Consciousness-Based Education:
A Foundation for Teaching and Learning in the Academic Disciplines

* A Series of 12 Volumes *

Managing Editor, Dara Llewellyn
Executive Editor, Craig Pearson

2011

Consciousness-Based Books
Maharishi University of Management
Fairfield, Iowa 52557
VOLUME X

Consciousness-Based Education and Computer Science

Volume Editor, Keith Levi
Volume Editor, Paul Corazza
Consciousness-based education and Computer Science

© 2011 by Maharishi University of Management. Transcendental Meditation®, TM®, TM-Sidhi®, Maharishi Transcendental MeditationSM, Maharishi TM-SidhiSM, Yogic Flying®, Consciousness-BasedSM, Maharishi Vedic ScienceSM, Maharishi University of Management®, Maharishi International University, Maharishi School of the Age of Enlightenment, Science of Creative Intelligence®, Maharishi Science of Creative IntelligenceSM, Maharishi Vedic Science and TechnologySM, Maharishi Technology of ConsciousnessSM, Maharishi Ayurveda®, Maharishi Ayurveda Medical Center, Vedic ScienceSM, Maharishi Vedic University, and Maharishi European Research University are protected trademarks and are used in the U.S. under license or with permission.

Cover and book design by Shepley Hansen
Printed in the United States of America

Published by Consciousness-Based Books
an Imprint of Maharishi University of Management Press
Fairfield, IA 52557 [www.mumpress.com]

Library of Congress Cataloging-in-Publication Data

Consciousness-based education : a foundation for teaching and learning in the academic disciplines / managing editor, Dara Llewellyn ; executive editor, Craig Pearson.

v. cm.

“A series of 12 volumes.”

Includes bibliographical references.


LB2322.2.C67 2011
378.001--dc23

2010051616

ACKNOWLEDGEMENTS

Special acknowledgement goes to Cathy Zhang and Yali Jiang for their contribution to the production of this series; Consciousness-Based Education: A Foundation for Teaching and Learning in the Academic Disciplines. Recognition is also due the staff of the Maharishi University of Management Library for their reference, electronic, and archival support.
# Table of Contents

Introduction to the Series ................................................................. 1

Introduction to the Volume ............................................................. 13

**Part I:**

**Computer Science and *Maharishi Vedic Science***

Unified Field-Based Computer Science:  
Toward a Universal Science of Computation. ................................... 23  
Bruce P. Lester, Ph.D.

Vedic Computation: Redefining Computer Science  
in the Light of *Maharishi Vedic Science* ........................................... 107  
Gregory Guthrie, Ph.D.

**Part II:**

**Self-Referral Dynamics in Foundations of Computer Science**

The Reflexive Domain: Unified Field of Pure Knowledge at the Foundation of Computer Science ................................................................. 155  
Bruce P. Lester, Ph.D.

The Self-Referral Dynamics of Computation:  
An Introduction to a Course on Algorithms.................................173  
Paul Corazza, Ph.D.
Part III:
Principles of Maharishi Vedic Science in Software Engineering

Beyond Extreme Programming:
Absolute Programming, Broad Awareness, and Fine Focus of Attention......................................................... 199
Keith R. Levi, Ph.D.

Part IV:
Principles of Maharishi Vedic Science in the Theory of Algorithms

Computable Functions in the Theory of Algorithms: Collapsing Infinity to a Point .............................................. 241
Paul Corazza, Ph.D.

Part V:
Appendices

Modern Science and Vedic Science:
An Introduction .................................................................................................................................................. 289
Kenneth Chandler

Resources......................................................................................................................................................... 325
Higher education faces a complex set of challenges today. We are seeing resources diminish at the same time we are hearing calls for greater access and affordability. Demands for greater transparency and accountability are being sounded by both the general public and the government. Government is exerting increasing controls in this long-independent area.

These challenges, however, are merely financial and political, and they are hardly limited to colleges and universities. The fundamental challenges are educational and center around the students themselves. Challenges include high levels of stress, pervasive substance abuse (particularly binge drinking), lack of preparedness for college-level work, and mental and emotional disabilities. In most of these areas, the problem is serious and worsening. Though colleges and universities are striving to address these challenges, few would claim we are turning the tide.

An encouraging trend is the increasing focus in higher education nationwide on promoting student learning. Yet these laudable efforts do not take into account the powerful forces working in opposition. It is well known that learning is inhibited by stress, sleep deprivation, alcohol, and poor diet—and these are among the most conspicuous features of the college student experience.

Something new is required. Education needs a reliable means of developing students directly from within. We need a systematic method for cultivating their creative intelligence, their capacity to learn, and their natural humanity. All education aims at these goals, of course—but the approach thus far has been from the outside in, and the results have been haphazard at best.

Consciousness-Based education was established to address this need. It integrates the best practices of education and places beneath them a proper foundation—direct development of the student from inside out.

The outcomes of Consciousness-Based education have been unprecedented and scientifically verified. These outcomes include significant
growth of intelligence, creativity, learning ability, field independence, ego development, and moral maturity, among others. These results are remarkable because many of these values typically plateau in adolescence—but Consciousness-Based education promotes this growth in students of all ages, developing potentials that otherwise would have remained unexpressed.

Beyond this rich cognitive growth, Consciousness-Based education significantly reduces student stress, boosts self-esteem, improves health, reduces substance use, and enhances interpersonal relationships. All of this comes together to create exceptional learning environments. This approach even measurably improves the quality of life in the surrounding society.

Consciousness-Based education was founded by Maharishi Mahesh Yogi, the world authority on the science of consciousness. First pioneered at Maharishi University of Management (previously Maharishi International University, 1971–1995) in Fairfield, Iowa, Consciousness-Based education is being adopted by schools, colleges, and universities around the world. It is easily integrated into any school, without any change in mission or curriculum.

Consciousness-Based education recognizes that student learning depends fundamentally on students’ levels of consciousness or alertness. The more alert and awake the student, the more successful and satisfying the learning.

Consciousness-Based education consists of three components:

• a practical technology for directly developing students’ potential from within,
• a theoretical understanding of consciousness that gives rise to a unifying framework for knowledge, enabling students to easily grasp the fundamental principles of any discipline and to connect these principles to their own personal growth, and
• a set of classroom practices, arising from this understanding, that also helps promote effective teaching and learning.
The Transcendental Meditation program

At the heart of Consciousness-Based education is the practice of the Transcendental Meditation technique. The technique was brought to light by Maharishi Mahesh Yogi from the Vedic tradition of India, the world’s most ancient continuous tradition of knowledge. It is practiced for 20 minutes twice daily, once in the morning and once in the afternoon, while sitting comfortably with eyes closed. It is simple, natural, and effortless—so simple, in fact, that ten-year-old children can learn and practice it. It has been learned by more than six million people worldwide, of all ages, religions, and cultures.

The Transcendental Meditation technique differs from other procedures of meditation and relaxation in its effortlessness. It involves no concentration or control of the mind. Neither is it a religion, philosophy, or lifestyle. It involves no new codes of behavior, attitudes, or beliefs, not even the belief it will work.

The Transcendental Meditation program is the most extensively validated program of personal development in the world. It has been the subject of more than 600 scientific research studies, conducted at more than 250 universities and research institutions in more than 30 countries worldwide. These studies have been published in more than 150 scientific and scholarly journals in a broad range of fields, including Science, Scientific American, American Journal of Physiology, International Journal of Neuroscience, Memory and Cognition, Social Indicators Research, Intelligence, Journal of Mind and Behavior, Education, Journal of Moral Education, Journal of Personality and Social Psychology, Business and Health, British Journal of Educational Psychology, Journal of Human Stress, Lancet, Physiology and Behavior, and numerous others. No approach to education has as much empirical support as Consciousness-Based education.

This approach, moreover, has been successfully field-tested over the past 35 years in primary, secondary, and post-secondary schools all over the world, in developed and developing nations, in a wide variety of cultural settings—the United States, Latin America, Europe, Africa, India, and China.

The Transcendental Meditation technique enables one to “dive within.” During the practice, the mind settles inward, naturally and spontaneously, to a state of deep inner quiet, beyond thoughts and per-
ceptions. One experiences consciousness in its pure, silent state, uncolored by mental activity. In this state, consciousness is aware of itself alone, awake to its own unbounded nature.

The technique also gives profound rest, which dissolves accumulated stress and restores balanced functioning to mind and body.

This state of inner wakefulness coupled with deep rest represents a fourth major state of consciousness, distinct from the familiar states of waking, dreaming, and sleeping, known as Transcendental Consciousness.

In this restfully alert state, brain functioning becomes highly integrated and coherent. EEG studies show long-range spatial communication among all brain regions. This coherence is in sharp contrast to the more or less uncoordinated patterns typical of brain activity.

With regular practice, this integrated style of functioning carries over into daily activity. Research studies consistently show a high statistical correlation between brainwave coherence and intelligence, creativity, field independence, emotional stability, and other positive values. The greater one’s EEG coherence, in other words, the greater one’s development in these fundamental areas. At Maharishi University of Management, students even have the option of a Brain Integration Progress Report—an empirical measure of growth of EEG coherence between their first and last years at the University.

The brain is the governor of all human activity—and therefore personal growth and success in any field depend on the degree to which brain functioning is integrated. The increasingly integrated brain functioning that spontaneously results from Transcendental Meditation practice accounts for its multiplicity of benefits to mind, body, and behavior.

Every human being has the natural ability to transcend, to experience the boundless inner reality of life. Every human brain has the natural ability to function coherently. It requires only a simple technique.

**Theoretical component—**
**a unified framework for teaching and learning**

Scholars have long called for a way to unify the diverse branches of knowledge. Current global trends are making this need ever more
apparent. The pace of progress is accelerating, the knowledge explosion continues unabated, and knowledge is becoming ever more specialized.

Academic disciplines offer a useful way of compartmentalizing knowledge for purposes of teaching, learning, research, and publication. But each academic discipline explores only one facet of our increasingly complex and interrelated world. The real world, however, is not compartmentalized—an elephant is not a trunk, a tusk, and a tail. Academic disciplines, consequently, are criticized as inadequate, in themselves, for understanding and addressing today’s challenging social problems.

Today, more than ever, we need a means of looking at issues comprehensively, holistically. We need a way of discovering and understanding the natural relationships among all the complex elements that compose the world, even among the complex elements that compose our own disciplines.

Various attempts to address this need have been made under the rubric of interdisciplinary studies—programs or processes that aim to synthesize the perspectives and promote connections among multiple disciplines. Some of these efforts have been criticized as superficial joinings of disciplinary knowledge. But the chief criticism of interdisciplinary studies—leveled even by its proponents—is that looking at an issue from multiple perspectives does not, in itself, enable one to find the common ground among contrasting viewpoints, to resolve conflicts, and to arrive at a coherent understanding.

The diverse academic disciplines can be properly unified at only one level—at their source. All academic disciplines are expressions of human consciousness—and if the fundamental principles of consciousness can be identified and understood, then one would gain a grasp of all human knowledge in a single stroke.

This brings us to the theoretical component of Consciousness-Based education. Consciousness-Based education does precisely this—and not as an abstract, theoretical construct but as the result of students’ direct experience of their own silent, pure consciousness. In this sense, practice of the Transcendental Meditation technique forms the laboratory component of Consciousness-Based education, where the theoretical predictions of Consciousness-Based education can be verified through direct personal experience.
This theoretical component offers a rich and deep yet easy-to-grasp intellectual understanding of consciousness—its nature and range, how it may be cultivated, its potentials when fully developed. This theoretical component also identifies how the fundamental dynamics of consciousness are found at work in every physical system and in every academic discipline at every level.

With this knowledge as a foundation, teachers and students in all disciplines enjoy a shared and comprehensive understanding of human development and a set of deep principles common to all academic disciplines—a unified framework for knowledge. With this unified framework as a foundation, students can move from subject to subject, discipline to discipline, and readily understand the fundamental principles of the discipline and recognize the principles the discipline shares with the other disciplines they have studied. This approach makes knowledge easy to grasp and personally relevant to the student.

**Pure consciousness and the unified field**

Consciousness has traditionally been understood as the continuous flux of thoughts and perceptions that engages the mind. Thoughts and perceptions, in turn, are widely understood to be merely the by-product of the brain’s electrochemical functioning.

Maharishi has put forward a radically new understanding of human consciousness. In Consciousness-Based education, pure consciousness is understood as the foundation and source of all mental activity, the most silent, creative, and blissful level of the mind—the field of one’s total inner intelligence, one’s innermost Self. (This unbounded value of the Self is written with an uppercase “S” to distinguish it from the ordinary, localized self we typically experience.) Direct experience of this inner field of consciousness awakens it, enlivens its intrinsic properties of creativity and intelligence. Regular experience of pure consciousness through the Transcendental Meditation technique leads to rapid growth of one’s potential, to the development of higher states of human consciousness—to *enlightenment*.

But consciousness is more, even, than this.

Throughout the twentieth century, leading physicists conjectured upon the relation between mind and matter, between consciousness and the physical world; many expressed the conviction that mind is,
somehow, the essential ingredient of the universe. But Maharishi goes further. He has asserted that mind and matter have a common source, and that this source is pure consciousness. Consciousness in its pure, silent state is identical with the most fundamental level of nature’s functioning, the unified field of natural law that has been identified and described by quantum theoretical physicists over the past several decades. Everyone has the potential to experience this field in the simplest form of his or her own awareness. Considerable theoretical evidence, and even empirical evidence, has been put forward in support of this position.

Maharishi has developed these ideas in two bodies of knowledge, the first known as the Science of Creative Intelligence, the second as Maharishi Vedic Science and Technology. The Science of Creative Intelligence examines the nature and range of consciousness and presents a model of human development that includes seven states of consciousness altogether, including four higher states beyond the familiar states of waking, dreaming, and sleeping. These higher states, which develop naturally and spontaneously with Transcendental Meditation practice, bring expanded values of experience of one’s self and the surrounding world. Each represents a progressive stage of enlightenment. Maharishi Vedic Science and Technology examines the dynamics of pure consciousness in fine detail. It reveals the fundamental principles of consciousness that may then be identified in every field of knowledge and every natural system.

Most important for teaching and learning, these sciences reveal how every branch of knowledge emerges from the field of pure consciousness and how this field is actually the Self of every student.

**Strategies for promoting teaching and learning**

Consciousness-Based education also includes a battery of educational strategies that promotes effective teaching and learning. Foremost among these is the precept that parts are always connected to wholes and that learning is most effective when learners are able to connect parts to wholes. In Consciousness-Based education, the parts of knowledge are always connected to the wholeness of knowledge, and the wholeness of knowledge is connected to the Self of the student.
One means of doing this is through *Unified Field Charts*. These wall charts, developed by the faculty at Maharishi University of Management and used in every class, do three things: (1) They show all the branches of the discipline at a glance; (2) They show how the discipline emerges from the field of pure consciousness, the unified field of natural law at the basis of the universe; (3) They show that this field is the Self of the student, which the student experiences during practice of the Transcendental Meditation technique.

In this way students can always see the relation between what they are studying and the discipline as a whole, and they can see the discipline as an expression of their own pure consciousness. Again, this is more than an intellectual formulation—it is the growing reality of students’ experience as they develop higher states of consciousness.

Another strategy is *Main Point Charts*. Developed by the faculty for each lesson and posted on the classroom walls, these charts summarize in a few sentences the main points of the lesson and their relationship to the underlying principles of consciousness. In this way students always have the lesson as a whole in front of them, available at a glance.

### The next paradigm shift

If higher education is fundamentally about student learning and growth, then Consciousness-Based education represents a major paradigm shift in the history of education. To understand this change, it is useful to reflect on the encouraging paradigm shift that has already been taking place in education over the past several decades.

This shift involves a move from what many call an *instruction paradigm* to a *learning paradigm*. In the instruction paradigm, the mission of colleges and universities is to provide instruction; this is accomplished through a transfer of knowledge from teacher to student. In the learning paradigm, the mission is to produce student learning; this mission is achieved by guiding students in the discovery and construction of knowledge.

This shift is a vitally important advance in education, leading to more successful outcomes and more rewarding experiences for students and teachers alike. But a further paradigm shift remains, and we can understand it by examining a fundamental feature of human experience.
Maharishi observes that every human experience consists of three fundamental components: a knower, a known, and a process of knowing linking knower and known. We may also use the terms experiencer, object of experience, and process of experiencing, or observer, observed, and process of observation.

This threefold structure of experience is nowhere more evident than in schools: The knowers are the students, the known is the knowledge to be learned, and the process of knowing is what the full range of teaching and learning strategies seek to promote.

Understanding this threefold structure helps us understand the paradigm shifts that are taking place.

The instruction paradigm places emphasis on the known. It focuses on the information students are to absorb and the skills they are to learn. In this paradigm, the instructor’s role is to identify what students need to know and deliver it to them.

The learning paradigm emphasizes the process of knowing. It recognizes that students must be actively involved in the learning process, that knowledge is something individuals create and construct for themselves, that students have differing learning styles and differing interests that must be taken into account. In this paradigm, the instructor’s role is to create learning environments and experiences that promote the process of learning.

The Consciousness-Based paradigm embraces the known and the process of knowing but places primary emphasis on the knower—on
developing the knower’s potential for learning from within. The following diagram shows the respective emphases of each approach:

But the learning paradigm does not so much abandon the instruction paradigm as enlarge it, so that it includes the process of knowing as well as the known. And the Consciousness-Based approach completes the enlargement to include the knower:

Consciousness-Based education, in summary, is a theory and practice grounded in a systematic science and technology of consciousness, making available the complete experience, systematic development, and comprehensive understanding of the full range of human consciousness. More than 30 years’ experience and extensive scientific research
confirm the success of this approach and its applicability to any educational institution.

**About this book series**

This series of twelve volumes is the result of a unique faculty-wide project that began with the founding of Maharishi University of Management in 1971 and continues to this day. Each volume in the series examines a particular academic discipline in the light of our Consciousness-Based approach to education.

Volumes include

- an introductory paper introducing the Consciousness-Based understanding of the discipline,
- a Unified Field Chart, if available for publication for the discipline—a chart that conceptually maps all the branches of the discipline and illustrates how the discipline emerges from the field of pure consciousness and how that field is the Self of every individual. Thus, these charts connect the “parts” of knowledge to the “wholeness” of knowledge and the wholeness of knowledge to the Self of the student;
- subsequent papers that show how this understanding may be applied in various branches of the discipline,
- occasional examples of student work exploring how the Consciousness-Based approach enhances learning in the discipline, and
- an appendix describing Maharishi Vedic Science and Technologies of Consciousness in detail.
<table>
<thead>
<tr>
<th>VOLUME</th>
<th>DISCIPLINE</th>
<th>VOLUME EDITOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Maharishi Vedic Science</td>
<td>Frederick Travis</td>
</tr>
<tr>
<td>2</td>
<td>Education</td>
<td>Christopher Jones</td>
</tr>
<tr>
<td>3</td>
<td>Physiology &amp; Health</td>
<td>Kenneth Walton, Janet Kernis, Robert Schneider, Paul Morehead</td>
</tr>
<tr>
<td>4</td>
<td>Physics</td>
<td>Gerald Geer</td>
</tr>
<tr>
<td>5</td>
<td>Mathematics</td>
<td>Paul Corazza, Anne Dow</td>
</tr>
<tr>
<td>6</td>
<td>Literature</td>
<td>Terrance Fairchild</td>
</tr>
<tr>
<td>7</td>
<td>Art</td>
<td>Matthew Beaufort</td>
</tr>
<tr>
<td>8</td>
<td>Management</td>
<td>Dennis Heaton, Jane Schmidt-Wilk, Bruce McCollum</td>
</tr>
<tr>
<td>9</td>
<td>Government</td>
<td>Rachel Goodman, William Sands</td>
</tr>
<tr>
<td>10</td>
<td>Computer Science</td>
<td>Keith Levi, Paul Corazza</td>
</tr>
<tr>
<td>11</td>
<td>Sustainable Living</td>
<td>Mabel Scaroni-Fisher, David Fisher</td>
</tr>
<tr>
<td>12</td>
<td>World Peace</td>
<td>Rachel Goodman</td>
</tr>
</tbody>
</table>

We welcome inquiries and further contributions to this series.

**CONTACT INFORMATION**

Dara Llewellyn, Managing Editor  
Consciousness-Based Education Book Series  
Maharishi University of Management  
Fairfield, Iowa 52557  
Phone: 641-472-7000
Consciousness-Based education makes a rich contribution to the study of computer science. Similar to other fields of study, Consciousness-Based education in computer science has both theoretical and experiential aspects.

Theoretical Contributions of Consciousness-Based Education to Computer Science

The principles of Maharishi Vedic Science and Science of Creative Intelligence are principles of natural law and of consciousness. Computer science is a science of information, its organization, and its processing. As the articles in this volume point out, computer science involves applications of mathematics and logical reasoning to information processing. Computer science also involves understanding and modeling laws of nature in almost every other field of science, technology, business, and even art. Thus, computer science, like Maharishi Vedic Science, deals with the full range of natural law and human consciousness. It is not surprising then, that there are many rich connections that arise between Maharishi Vedic Science and the study of computer science.

In Consciousness-Based education, principles from Maharishi Vedic Science and Science of Creative Intelligence\(^1\) are used to illuminate the student’s understanding of computer science, and conversely, principles and practices in computer science can make the concepts of Maharishi Vedic Science better understood and more concrete. Students sometimes have a very deep and direct experience and understanding of principles of consciousness from Maharishi Vedic Science. They might have seen these same principles appearing not only in their experience of their own awareness, but also in different disciplines and different subfields of computer science. When they see them arising again in

---

\(^1\) All volumes in this series contain an appendix entitled “Modern Science and Vedic Science: An Introduction” that gives a general introduction to principles and concepts in Maharishi Vedic Science and the Science of Creative Intelligence. Some of these concepts and associated terminology may occur in this introduction and throughout this volume. The reader can refer to the appendix for a more thorough introduction to these terms and concepts.
a new area of computer science then they already have some feeling of familiarity and comfort with the principle. From the other direction, computer science principles and practices are often very concrete to students. When they see a strong connection of how a computer science principle illustrates or embodies a principle from Maharishi Vedic Science, this makes Maharishi Vedic Science more concrete to the students.

One of the most basic and fundamental principles of Maharishi Vedic Science is that the unified field of modern physics, which is the basis of all laws of nature, is a field of pure consciousness, and that this same field of pure consciousness can be directly experienced by individuals in a systematic repeatable method through the Transcendental Meditation and TM-Sidhi programs. Individuals can verify through direct experience the properties of this field that are also described in objective means by modern physics—that it is a field of perfect orderliness, harmony, unboundedness, great liveliness and energy at the same time as perfectly restful and nonactive. These are highly abstract and possibly contradictory properties at their face value. However, they become concrete and meaningful through Consciousness-Based education when students have both the direct personal experience of this state of consciousness, and further when they see examples of these abstract properties in their studies of computer science.

A fundamental characteristic of the unified field, according to Maharishi Vedic Science, is the property of self-awareness, awareness of awareness, or awareness of itself. Another fundamental and closely related principle is that creation arises from this fundamental characteristic or inherent relationship of awareness being aware of itself—the property of self-referral awareness. Maharishi Vedic Science describes how this property gives rise to three fundamental aspects of creation: knower, known, and process of knowing (Rishi, Chhandas, and Devatā in Vedic terminology).

At first impression, this principle or postulation that creation arises from the fundamental feature of pure consciousness as being self-aware might seem surprising or abstract and hard to comprehend. However, it is quite compatible with the modern physics description of how basic particles and forces in the universe can be described as different types of vibrations within the unified field of physics (e.g., vibration modes
of superstrings). According to Maharishi Vedic Science, the process of awareness being aware of itself produces a liveliness and some vibration at this most fundamental level of creation. This self-referral property means there is a fundamental relationship of knower, known, and process of knowing at this basic level. Pure awareness simultaneously plays the roles of knower (Rishi), the known (Chhandas), process of knowing (Devatā) as it becomes aware of itself.

This three-part structure of knower, known, and process of knowing is found throughout computer science, and every field of study. One simple example in computer science is the common design pattern and also a common system architecture known as model-view-controller. For example, in the case of a graphical user interface, the view is what is displayed on the screen; the model is the underlying information that is displayed, and the controller is the logic and processing that enables the underlying information to be displayed in various formats (for example, as text or graphics). The view is directly analogous to the known, the controller to a process of knowing, and the model to the knower or underlying intelligence.

These and many other principles of Maharishi Vedic Science arise naturally throughout the study of computer science. The above principles are discussed at length in terms of fundamental characteristics of computer science in several of the articles in this volume.

Experiential Contributions of Consciousness-Based Education to Computer Science

The most important contribution of Consciousness-Based education to computer science is experiential and extremely practical. The practical and experiential aspect of Consciousness-Based education, the Transcendental Meditation and TM-Sidhi programs, are fundamental to Consciousness-Based education in computer science just as they are for Consciousness-Based education in any discipline. Students can only correctly understand the principles of Maharishi Vedic Science if they have the direct experience of these principles, which can be done in a systematic repeatable manner through the Transcendental Meditation and TM-Sidhi programs. This requirement for experience to support knowledge is directly analogous to the standard component of all computer science studies, which is that students must experience the prin-

principles of computer science by implementing them through laboratory exercises in which they analyze, design, and implement all aspects of computing systems.

Having direct experience of pure awareness as part of Consciousness-Based education is also important for computer science because of the applied nature of the field and its demanding requirement for practitioners to have very highly developed abilities of orderly and precise thinking, and in particular, the ability to have broad awareness of an entire computing system and application domain at the same time as the ability to focus on very refined details of system implementation. The paper by Levi in this volume carefully analyzes this requirement for the field of software engineering. It describes how almost all of the current software engineering best practices can be seen as techniques to support this particular capability. It goes on to describe how the Transcendental Meditation and TM-Sidhi programs directly develop these capabilities in individuals and reviews an extensive body of empirical scientific evidence supporting this outcome.

The paper by Lester in Part I describes how the definition of computing can be extended by Consciousness-Based education in the sense that the eternal and ongoing operation of the laws of nature can be considered as an ultimate computing system. Further, the paper proposes that Maharishi Vedic Science offers a fulfillment of the goals of computer science in that any individual can gain the full benefit of nature’s computing simply by aligning his or her awareness with natural law through the Transcendental Meditation and TM-Sidhi programs.

S E C T I O N H E A D I N G S

Part I
In Part I, Lester describes how many important principles of computer science can be seen as limited reflections of the properties of the unified field of physics. He also provides an expanded definition of computing that includes the nervous system and self-referral computing of the unified field. Guthrie describes how Maharishi Vedic Science encompasses the full range of intelligence from the level of the unified field through all its expressions as investigated by different disciplines, and provides an expanded context for understanding some of the fundamental principles underlying computer science and other disciplines.
Part II
Part II contains two papers that describe how several of the mathematical underpinnings of computer science closely reflect self-referral properties of the unified field. These papers represent examples of insights into the foundations of computation provided by Maharishi Vedic Science.

Lester’s paper describes an analogy between the unified field of physics and the reflexive domain at the foundation of computation. He describes how procedures and information can be completely unified by the concept of lambda functions. In parallel to physics, this unification locates a self-interacting field called the reflexive domain at the source of computing. The vast variety of computing in nature can be understood as essentially consisting of the self-referral interaction of the reflexive domain at the source of computation. Based on these observations and a further survey of a set of properties of the reflexive domain he asserts that the reflexive domain represents the unified field of pure knowledge at the basis of computer science. He concludes: “Just as physics can be said to have glimpsed the unified field of pure knowledge, so now computer science has also glimpsed this field at its own foundation.”

Corazza’s paper on the Self-Referral Dynamics of Computation offers another analogy between Maharishi Vedic Science and the mathematical underpinnings of computation. Corazza reviews the perspectives offered by Maharishi Vedic Science and recent discoveries in quantum field theory, both of which conclude that nature’s computational dynamics are based on self-referral performance at a hidden, unmanifest level. Corazza shows how the mathematics of computation has its basis in self-referral dynamics by showing that every computable function may be defined as the fixed point of an operator on a vast (and therefore, in a sense, unmanifest) function space.

Parts III and IV
Parts III and IV of the volume provide articles on a specific field of study in computer science. Part III contains the paper on software engineering by Levi discussed above, which presents practical implications of Consciousness-Based education for software engineering. It describes how Consciousness-Based education develops a fundamental ability required by good software engineers, and analyzes how this ability is
already the basis for most modern software engineering practices. The article shows how many—if not all of—the fundamental practices in the field of software engineering can be seen as supporting the ability of software engineers to have broad awareness along with fine focus of attention, and that this ability is critical to successful software engineering.

The article by Corazza in Part IV presents an analogy between fundamental principles underlying Consciousness-Based education and computation in the field of algorithms and computability theory. He considers the Maharishi Vedic Science principle that creation arises in the collapse of the infinite unbounded value of wholeness to a point, from which creation arises in a sequential unfoldment. He shows how the class of functions that are actually used in computer science arises from a much vaster class of number-theoretic functions through a sequence of “collapses” to successively narrower function classes. He discusses how this sequence of collapses parallels the creative dynamics of pure consciousness as it brings forth creation through the collapse of the abstract, indescribable infinite value of wholeness to the concrete, specific point value within its nature.

**Part V**

This section includes Dr. Kenneth Chandler’s “Modern Science and Vedic Science: An Introduction,” which served as the introduction to the inaugural issue of the journal Modern Science and Vedic Science and which presents an overview of Maharishi Vedic Science and the new technology of consciousness developed by Maharishi Mahesh Yogi. The second appendix in this section provides a list of relevant links and resources for this volume.
Part I

Computer Science
and *Maharishi Vedic Science*
Unified Field-Based Computer Science:
Toward a Universal Science of Computation

Bruce P. Lester, Ph.D.
ABOUT THE AUTHOR

Bruce P. Lester, Ph.D., is Professor of Computer Science at Maharishi University of Management. He received his B.S. in electrical engineering in 1971, M.S. in computer science in 1971, and Ph.D. in computer science in 1974 from M.I.T. Dr. Lester was a Lecturer on the faculty of Princeton University in the Department of Electrical Engineering and Computer Science from 1973 to 1975. In 1979, he joined the faculty of Maharishi University of Management, where he founded the Department of Computer Science and served as Chair for eight years. For six years, he worked as a Senior Software Engineer at Primus Telecommunications. Dr. Lester is a widely published researcher in the area of parallel computing. He is author of the book *The Art of Parallel Programming*, which has been adopted as a course textbook by 62 universities.
ABSTRACT

This article presents an overview of a new approach to computer science based on the recent identification of the unified field of all the laws of nature by quantum physics and on the direct experience of the unified field provided by Maharishi Vedic Science through its applied aspect, the Maharishi Transcendental Meditation and TM-Sidhi programs. In this approach, every aspect of computer science is studied in its relation to the whole discipline, which is understood as emerging from the unified field of natural law.

The principles of computer science are seen as limited reflections of the dynamical properties of the unified field, which is the source of all activity in the universe. Any individual may directly experience this integrated source of computer science in the simplest state of his awareness through the Transcendental Meditation and TM-Sidhi programs, and thereby gain access to the highest value of computer science, the “cosmic computing” of natural law in the unified field. Part I of the paper presents an integrated view of computer science with the unified field as its ultimate foundation. Part II presents an expanded view of the future of computer science, and redefines the traditional focus of the field to include not only electronic computing, but all computational processes in nature including the computing of the human nervous system and the self-referral computing within the structure of the unified field.

Part I:
The Foundation of Computer Science

1.1 Introduction

The recent identification of the unified field of all the laws of nature by quantum physics, and the compelling evidence that it is a field of consciousness (Hagelin, 1987), have profound implications for all areas of science and technology. Science seeks to identify and formulate specific laws of nature, and technology utilizes these laws to produce desired effects. The unified field is understood by quantum physics as the ultimate origin of all the specific laws of nature studied in all fields of science and applied in all areas of technology. Thus, the knowledge of the unified field can lead to accelerated progress in science and technology, and more rapidly achieve the ultimate purpose of technology, to improve the quality of human life. Moreover,
the direct experience of the unified field provided by the Transcendental Meditation and TM-Sidhi programs presents the possibility for any individual to utilize the total potential of natural law inherent in the unified field and live the highest quality of life.

The knowledge of the unified field has particular significance to computer science and technology. Computer science has always been an interdisciplinary field of knowledge incorporating many diverse areas such as abstract mathematics, semiconductor technology, electronics, language theory, systems organization, and personnel management. As the boundaries of the field have continued to expand, it has become increasingly necessary to develop an integrated and coherent view of the field as a whole and its relationship to other fields of knowledge. Unified field-based computer science satisfies this need by presenting an integrated conceptual framework, in which each aspect of computer science is seen in relation to the whole field and in relation to all the laws of nature inherent in the unified field. By incorporating the knowledge of the unified field as the foundation of computer science, this new integrated approach to computer science also presents a new expanded vision of the discipline with the potential to accelerate progress in the field and bring it to fulfillment.

Through the systematic investigation of finer levels of matter during the past few hundred years, physical science has arrived at an increasingly unified understanding of natural law. Progress in this area has especially accelerated during the past twenty years and has culminated in the recent development of unified quantum field theories in which all the fundamental force fields and fine particles known to physics are unified into one single field (Hagelin, 1987). Although only recently discovered by modern physics, this unified field of natural law has been known throughout the ages by the ancient Vedic tradition, the oldest tradition of human knowledge. However, due to misinterpretation and partial loss of this knowledge over time, it has not been available in its completeness until revived and properly reinterpreted by Maharishi Mahesh Yogi (Chandler, 1987). This revival of the ancient Vedic tradition by Maharishi and its expression in the language of the modern scientific age is called Maharishi Vedic Science. (For brevity, the term Vedic Science will also be used with the same meaning.)
According to Vedic Science, the vast diversity of the universe has its basis and origin in a fundamental field of unity whose description parallels the unified field as known to physics. Unified quantum field theories are still evolving and have not yet reached the completeness of the knowledge of the unity of nature available from Vedic Science, but the descriptions are similar enough that Maharishi, in conjunction with leading theoretical physicists, has stated that modern physics has glimpsed the “unified field of natural law” found in the ancient Vedic tradition.

This growing knowledge of the unified field of natural law from quantum physics, and the complete knowledge and experience of the unified field from Maharishi Vedic Science, form the foundation of the new approach to computer science discussed in this paper: unified field-based computer science. The paper begins in Part I with a discussion of the relationship between the unified field and all the traditional areas of computer science, including hardware, software, and theory. The knowledge of the unified field is seen as the source and basis of all aspects of computer science, thus presenting a foundation for a new integrated approach to the whole discipline of computer science. This integrated vision, summarized by the chart of Figure 1 [which is an abbreviated version of the full Unified Field Chart, which appeared in the original publication (Lester, 1987)—Ed.] is especially useful in computer science education since it presents the student with a complete picture of the full range of computer science knowledge.

Part I of the paper (especially sections 1.2 and 1.4) will discuss the structure of this chart in more detail and explain how its organization reflects some of the basic organizational principles of the unified field. Part I also presents a more complete discussion of the understanding of the unified field from unified quantum field theories and its proposed identity with the field of “pure consciousness” described by Maharishi Vedic Science (see section 1.3). Some of the basic concepts in computer science are discussed in sections 1.5 and 1.6 as limited reflections of similar principles operating in the domain of the unified field. The evolution of computer technology is shown in section 1.6 to be leading toward deeper levels of natural law in the direction of a technology of the unified field. Sections 1.7 and 1.8 discuss techniques for direct experience of the unified field called the Transcendental Meditation.
and TM-Sidhi programs and their potential contribution to the field of computer science. This technology of the unified field plays a central role in unified field-based computer science because it provides the direct experience of the integrated source of computer science, as will be fully explained throughout Part I of the paper.

There is a growing trend in the field of computer science toward a more expanded definition of the discipline, incorporating the study of phenomena in nature that appear to have “computational” properties. Although the primary focus of computer science is to create more efficient and useful electronic computers, scientists are beginning to use some of the dynamical organizational principles found in the natural sciences as a basis for design of hardware and software in computer systems. The rapidly expanding research area termed neural networks uses the properties of neurons in the human brain as a model for creating computational systems in an electronic computer to do complex pattern recognition, learning, and language processing.

Researchers in this area consider the activity of the human brain as a form of “computation,” and they have attempted to simulate some aspects of this computation with an electronic computer, both in an attempt to produce more useful electronic computing, and also to understand more completely the functioning of the human nervous system. The dynamical properties of natural law in nonliving systems have also been successfully used as a model for electronic computing in some cases. The self-organization that takes place in many physical systems as they are cooled has been simulated mathematically with computer software as a basis for solving optimization problems that arise in business organizations. Thus, there is an increasing recognition that “computation” is not just some isolated phenomenon found only in electronic computers, but an activity of nature governed by natural law. This viewpoint provides the opportunity to develop a true science of computation that studies the dynamical laws of computing found in all areas of nature as a basis for engineering more powerful computing machines.

With this expanded vision of the whole field of computer science, the unified field is considered not only as the basis and source of the science, but an intimate part of the study of computer science itself. Part II of the paper discusses this theme thoroughly, beginning with a
presentation of some of the research trends in computer science leading toward this broader definition of the field, especially the neural network research based on the study of neurons in living systems. In Part II, the dynamical properties of the unified field as known through Maharishi Vedic Science will be examined for their “computational” aspects in an attempt to extend the knowledge of computer science into this most fundamental realm of natural law.

The proposition is put forward and developed in Part II that the historical evolution of computer science has been leading in the direction of more fundamental levels of natural law and will eventually arrive at a consideration of the computational properties of the unified field, the most fundamental level of natural law. It will be seen that many of the important foundational concepts found in computer hardware and software can be located in the internal dynamics of the unified field as expressed in Vedic Science. The ideas in Part II are not presented as final and fully developed, but as the first results that have emerged from this exciting new research area, exploring the computational laws of nature at the basis of the structure and functioning of the universe. From the results developed so far, it seems that this approach has the potential to greatly accelerate progress in all areas of computer science by providing a model of the most perfected value of computing—the “cosmic” computing of natural law in the unified field. Part II of the paper will also discuss how this expansion of the boundaries of computer science can dramatically improve the quality of computer science education by providing students with a picture of the goals of the evolution of computer technology.

1.2 A New Integrated Approach

The best way to begin a discussion of unified field-based computer science is with reference to the chart of Figure 1, showing the whole range of computer technology in its sequential emergence from the unified field of all the laws of nature, the total potential of natural law. From this unified basis come the three fundamental areas of computer science: theory, software, and hardware, which form a foundation of knowledge for computer applications in all areas. Within each major area of the chart, the specific subdivisions of knowledge are shown in relation to one another, progressing from fundamentals at the bottom of the chart to more applied values toward the top. This chart is especially useful as
Figure 1. This chart illustrates the sequential emergence of all aspects of computer science from their integrated source in the unified field of natural law. All the specific principles of computer science are seen as limited reflections of the dynamical properties of the unified field.
an educational tool because this vision of the whole at a glance helps the student to learn and apply the specific aspects of knowledge studied in each computer science course.

If computer science is to evolve as a true “science” rather than just a collection of loosely related engineering techniques, it is important to have an integrated vision of all aspects of the science and how they relate to and are derived from each other. The chart of Figure 1 is an attempt to provide such an integrated framework with each major area of computer science seen in the context of other areas. The chart also locates foundational principles and logical steps of sequential derivation of higher-level principles from these foundations. The foundation of computer hardware is shown as semiconductor technology and electronics, which have their basis in the knowledge of the fundamental force fields and matter fields of natural law, especially the electromagnetic field and its interaction with charged particles such as electrons. According to recent discoveries in quantum physics, these fundamental force and matter fields ultimately arise as excitations of an underlying universal level of natural law, the unified field of natural law. The properties of the unified field as known from quantum physics and from Vedic Science will be discussed in more detail in the next two sections. Justification will also be presented for placing the unified field as the source of software and theory.

Both Hardware and Software are organized in Figure 1 in hierarchical levels, where each successive level is created or logically derived from concepts or objects found at the previous level. Readers familiar with computers and computer programming will easily be able to see this property in both the large Hardware and Software boxes. Theory is much more diverse and therefore not as well structured as Hardware and Software. The basis of Theory is shown as discrete mathematics, which provides the language and formal conceptual tools for all the areas of theoretical computer science. The five other boxes inside Theory represent some of the major subdivisions of computer science theory. The arrows that connect boxes in the chart show some of the logical relationships and conceptual dependencies. [The original version of this chart (Lester, 1987) shows many additional relationships that do not appear in the present version.—Ed.] Naturally there are many more relationships than those shown since the three areas of
hardware, software, and theory are intimately connected at every level. The major division into theory, software, and hardware shows a natural progression from the most abstract to most concrete aspects of the discipline. Theory is conceived in pure logic and is therefore most abstract and fundamental, arising from pure intelligence. Software is also organized and conceived on the basis of logical concepts and relationships, but has a more concrete expression with the organizing power to guide the internal activity of the computer. The computer hardware is of course the most concrete physical expression and realizes the potential of theory and software. Although apparently quite different on the surface, each of these three areas has similar dynamical principles forming a conceptual foundation for the whole field of computer science. In subsequent sections of the paper it will be shown how many of these fundamental dynamical principles are found in the internal functioning of the unified field of natural law, thereby justifying its position in the chart at the source of theory, software, and hardware. The organization of the diverse areas of the field into a coherent framework as in Figure 1 is a good first step toward creating an integrated science of computation. The next step must be to locate universal natural laws of computation from which all aspects of computer technology are derived. To accomplish this it is necessary to investigate more deeply the properties and dynamics of the source of computer science, the unified field of natural law. By studying the unified field from a “computational” viewpoint, it is possible to discover fundamental laws of computation that can form the basis for a universal science of computation.

1.3 Unified Field—The Basis of Computer Technology

The real significance of Figure 1 lies not just in the relationships between the specific areas of computer science, but in the identification of a common source for all branches of computer science: the unified field of all the laws of nature. Computer technology has always relied on electronics and physical materials technology to create the hardware vehicle for the process of computing. As integrated circuit technology has advanced toward utilizing smaller distance scales, hardware technology has become increasingly connected with semiconductor physics and the knowledge of subtle properties of the electromagnetic field as derived from quantum physics. Thus, the recent progress in experimen-
tal and theoretical physics toward a unified understanding of natural law could have great significance for the future of computer technology. Modern physics has isolated a few fundamental force fields and matter fields whose interactions are responsible for all aspects of the functioning of natural law in the universe. As shown at the bottom of the chart of Figure 1, the four fundamental force fields are electromagnetism, the weak and strong nuclear forces, and gravitation. The fundamental matter fields are quarks, neutrinos, and charged leptons (including electrons). Unified quantum field theories describe a completely unified field, whose self-interacting dynamics give rise to all the fundamental force and matter fields of natural law. Through a process known as spontaneous dynamical symmetry breaking, the essential unity of the unified field is broken (Campbell, Ellis, Hagelin, Nanopoulos, & Ticciati), allowing the sequential emergence of the more limited values of natural law, including the electromagnetic field and all the charged particles with which it interacts (Antoniadis, Ellis, Hagelin, & Nanopoulos).

Since the unified field is the ultimate source and basis of all aspects of the universe, it must also be the source of human life and human consciousness. The exact relationship between consciousness and the unified field is central to an understanding of unified field-based computer science. An analysis of one of the mathematical equations of unified field theory by Dr. John Hagelin, a leading theoretical physicist, has located properties that are the same as those of consciousness: self-interaction, self-referral, dynamism, orderliness, and intelligence (Hagelin, 1984). The quality of self-interaction makes the unified field dynamic in its nature, and this internal dynamism serves as the basis for the emergence of specific force and matter fields in the structure of the unified field. Since all of natural law emerges from the unified field, the extreme order and precision exhibited by the expressed levels of natural law must have their origin in the perfect orderliness inherent in the unified field. As the ultimate source of nature’s functioning, the unified field can also be considered as the most concentrated state of intelligence in nature, underlying all natural phenomena and giving a direction to all activity through the various channels of natural law (Hagelin, 1987, p. 58).

There is a growing body of theoretical and experimental evidence that the unified field of natural law is actually a field of conscious-
ness, a unified state of “pure” consciousness (Hagelin, 1987). Individual human consciousness and all the subjective values of life—ego, intellect, mind, and senses—arise as excitations or “waves” of this unified state of consciousness, which resides in its pure unbounded form at the source of thought deep within the human mind.

This identification of the unified field with pure consciousness was first proposed by Maharishi Mahesh Yogi (1986), based on the ancient Vedic tradition, which describes a “unified” state of pure consciousness as the ultimate source of all aspects of natural law. According to Maharishi Vedic Science, the universe emerges from this all-pervading, unbounded field of pure consciousness (or pure intelligence), which becomes aware of itself due to its own nature as consciousness, thereby creating a self-interacting dynamical relationship of knower, process of knowing, and known within itself. This self-referral interaction of pure consciousness with itself is responsible for (as if) breaking the unity and creating three from within the structure of its own oneness. Further interactions of the three (knower, process of knowing, and known) with each other and with the unity of pure consciousness then give rise sequentially to increasingly diverse levels of nature, resulting in all the localized values of natural law, including physical matter and life. In Vedic Science, the unity of pure consciousness is termed Saṁhitā and the three values created from that are Rishi (knower), Devatā (process of knowing), and Chhandas (known) (Maharishi Mahesh Yogi, 1985).

Thus, both modern theoretical physics and Maharishi Vedic Science describe a “unified field” whose self-referral dynamics gives rise to all forms and phenomena in nature. Quantum physics uses the language of modern mathematics, gradually developed through experimental and theoretical investigation of finer levels of physical matter, to describe the unified field. Vedic Science is derived from the direct experience of the unified field of consciousness by the ancient Vedic seers, who were able to directly cognize the mechanics of the creation of diversity from unity, and give expression to this knowledge in the form of the primordial sounds of natural law as it begins to manifest within the unified field through a process of self-interaction (Maharishi Mahesh Yogi, 1985). These primordial sounds are known as Veda, which means “pure knowledge.” Now through Maharishi’s revival and reinterpretation of the ancient knowledge of the Veda, and through the efforts of theo-
retical physicists in working with Maharishi, it has become possible to see that the Veda and unified quantum field theories are merely using different languages to describe the same underlying reality of nature’s functioning: the unified field of natural law.

This identification of pure consciousness with the unified field is just one more step in the historical theme of unification in modern science, and especially physics, that has taken place during the past hundred years. In the nineteenth century, the two physical phenomena of electricity and magnetism were thought to be completely separate until James Maxwell developed his now famous equations for the electromagnetic field, which unified electricity and magnetism as different expressions of the same underlying field (Maxwell, 1873/1954). The development of unified quantum field theories during the past fifteen years has continued this theme of unification to include all the fundamental force fields and matter fields. Also, in the classical physics of the nineteenth century, observer (consciousness) and the observed (physical matter) were considered as completely separate so that no interaction between them could take place. However, with the advent of Einstein’s theory of relativity in the early twentieth century and the subsequent development of quantum physics, it has been discovered that there is an intimate connection between observer and observed. Wigner (1970) has proposed that it is consciousness itself that causes the collapse of the quantum wave function during the process of measurement of a quantum field. Thus, the proposed identity of the unified field with pure consciousness is just the final step in this historical process of unification of the diverse aspects of natural law, a step that unifies the observer, observed, and process of observation into one self-interacting wholeness of the unified field.

1.4 Fundamentals of Computer Science
The discovery of the unified field and its identification with pure consciousness forms the conceptual foundation for unified field-based computer science. Since the unified field is the ultimate source of all physical matter, including the electromagnetic field, it is clearly the basis of computer hardware, as shown in the chart of Figure 1. However, the chart also shows the unified field as the source of software and theory. The justification for this lies in the nature of the unified field
as pure consciousness, whose excitations give rise to human thought and ordinary waking consciousness. Both software and theory, being expressions of thought and arising from consciousness, therefore have their ultimate source in the unified field. The unified field is described by Maharishi (1978) as an unbounded field of pure creative intelligence, the ultimate source of all creativity and intelligence in every individual. All aspects of theory, software, and hardware are outer expressions of this field of creative intelligence. As described above, pure creative intelligence has its own universal dynamics of self-interaction that allow it to express itself into all the subjective and objective aspects of natural law. It is to be logically expected that the outer expressions of this field of creative intelligence in computer science will reflect some of these same universal dynamical principles.

Unified quantum field theories have glimpsed the nature of this field of pure intelligence and provide a mathematical formulation of some aspects of its universal dynamical principles. Since these same dynamical principles are the source of human creative intelligence and its manifest expression into computer technology, it is not surprising to find that the basic building blocks of theory, software, and hardware are parallel to the basic values of the unified field, as described by these unified quantum field theories. Through the process of spontaneous symmetry breaking (Campbell, et al.), the unified field gives rise to the first manifestation of diversity, in the form of matter fields and force fields (see bottom of Figure 1). Matter fields are the source of all elementary particles or basic objects in creation, and force fields govern the interactions and basic transformations between objects that give rise to all activity in the universe. Computer science has long recognized the important role of elementary objects and transformations in the foundations of theory, software, and hardware. The fundamental structures and dynamics found at the basis of theory, software, and hardware are not isolated concepts of computer science alone, but specific expressions of the general principles of natural law as described by quantum field theory.

The two fundamentals emerging from the unified field (objects and transformations) provide the guiding principles for the basic elements of programming languages in the form of data types and control structures, as shown in Figure 1 at the bottom of the large Software box.
Data types, the basic kinds of objects in programming languages, and control structures, the basic transformational structures, give rise to the next level of software organization, consisting of data structures, file structures, and subprograms. These in turn are the foundation of software engineering, the knowledge of how to create complete software packages. Finally, database structures and systems analysis are concerned with the application of software, giving rise to the whole range of application programs at the top of Figure 1: educational applications, management information systems, scientific applications, communication systems, control systems, and artificial intelligence.

The two basic values of objects and transformations found in the self-referral dynamics of the unified field can also be located at the basis of the mathematical language used in computer science theory, in the form of set theory and logic. Set theory deals with combining objects into wholes, and logic provides formal rules for the sequential transformation of axioms into theorems in any mathematical theory. The complementary approach to the foundations of mathematics provided by category theory and topos theory also locates the basic concepts of objects and transformations at the source of mathematics (Weinless, 1987). The Discrete Mathematics box, shown in Figure 1 at the bottom of the big Theory box on the right, is the basis for the more applied areas of mathematical modeling and formal language and automata theory. The next level of theory, the theory of parallelism and the theory of programming languages, both find application in systems programs (center of Figure 1). The theory of algorithms (represented in a box at the top of the Theory box) is applied directly, along with mathematical modeling in the application programs (shown at the top of Figure 1).

Again, as in the foundation of the Software and Theory boxes, hardware begins with knowledge of objects (semiconductor technology) and knowledge of transformations (electronics). These two form the basis for the various levels of computer circuit organization shown in the microelectronics box. The next level of hardware organization is computer architecture and input-output devices, which combine to form the three categories of computers (microcomputers, minicomputers, and mainframe computers) and the collective level of computer networks. This whole field of hardware supports machine language, which along with some fundamentals from the large software and theory
boxes, combine to create the first level of actual computer programs called systems programs. The internal activity of the systems programs supports the various high level programming languages (see the wide box toward the top of the chart), which are used to write all the applications programs at the top.

Since computer science is such a vast, interdisciplinary field of knowledge, it can be unified only by knowledge as basic and comprehensive as that of the unified field. To unify software and hardware, the fields of consciousness and matter must be unified. Software is exclusively an expression of consciousness, while hardware is structured from physical laws of electronics. The unified field is the source of all physical and mental phenomena and is, therefore, the unified source of all branches of computer science (theory, software, and hardware). From this perspective, it is natural to find that the fundamentals of theory, software, and hardware reflect the basic dynamics of the unified field—matter fields (objects) and force fields (transformations). This theme of comparing the basic principles of computer science with the dynamical principles of the unified field will be continued and expanded throughout the subsequent sections of the paper, especially in Part II. As mentioned earlier, unified quantum field theories are still evolving and are not yet fully developed enough to provide the complete description of the unified field that is available from Vedic Science. Thus, the knowledge of the mechanics of creation from the unified field found in Vedic Science will provide a much richer foundation for computer science, and allow the identification of natural laws governing computation as a basis for formulating a universal science of computation.

1.5 Self-referral in Computer Science

As briefly mentioned in section 1.3, one of the most important dynamical properties of the unified field (as described in Vedic Science) is the property of self-referral, the process by which the unity of pure consciousness becomes aware of itself and assumes the role of knower, process of knowing, and known. This initial symmetry breaking of the unified field is the creative process at the source of the universe and the origin of all activity in nature. This knowledge is clearly expressed in the following verse from the Bhagavad-Gitā:
This fundamental dynamics of creation through self-referral is reflected in the functioning of computer software and hardware. One of the most basic control structures in computer software, responsible for much of its ability to compactly define a complex process of computation is the loop, a “self-referral” structure in which the flow of control of the program returns to itself again and again. Structural and functional loops are also found at many levels of computer hardware in the electronic circuits. In fact the word “circuit” means circle or loop, and in computer science this is a fundamental structure of self-referral allowing the electronic activity to flow back onto itself in cycles, thus creating the steps of computational activity in the hardware. Since subsequent sections of the paper will focus mostly on the hardware and electronics aspect of the computer, this section will deal mainly with software and analyze more deeply the nature of loops in computer programs as a reflection of the self-referral activity of the unified field of natural law. A loop in a program is an example of sequential self-referral: the program refers back to itself through a sequence of computing steps. This is not as complete and perfect as the instantaneous self-referral in the unified field, which is eternally aware of itself at the basis of all the laws of nature.

The function and importance of loops in computer programs can be illustrated through a simple example of a program that sorts a list of names into alphabetical order, a common and important computational task performed often by computers. The names are stored electronically in the memory of the computer in the form of an array, a sequence of memory cells each of which holds a single name. If there are 5000 names in the list, the array will be a sequence of 5000 cells numbered from 1 to 5000. A computer program may compare and move these names by referring to the cells of the array by number. To use the first name in the array, the program refers to NAME(1). To use the second name in the array, the program refers to NAME(2), and so on down to NAME(5000). To compare two names alphabetically, the program may use an instruction of the form:

\[
\text{IF NAME}(l) > \text{NAME}(2) \text{ THEN exchange NAME}(l) \text{ with NAME}(2)
\]
The “>” symbol compares NAME(1) and NAME(2) alphabetically to see if NAME(1) is “greater than” NAME(2); that is, whether NAME(1) occurs later in the alphabet than NAME(2). If so, then these two names should be exchanged in the array so NAME(2) comes first, since it is earlier in the alphabet. This process of comparison and possible exchange takes the list one step closer to being in full alphabetical order. The first step in writing a computer program to sort the list into alphabetical order is to write a preliminary program that moves down through the array comparing each successive pair of names and exchanging them if they are out of order. That is, after comparing NAME(1) with NAME(2), then compare NAME(2) with NAME(3), then NAME(3) with NAME(4), etc., until the end of the list is reached. To save having to write out 5000 copies of the IF instruction shown above, a loop is created in which the same IF instruction is repeated 5000 times, once for each successive pair of names. The two compared names will now be referred to symbolically as NAME(FIRST) and NAME(SECOND). Initially FIRST is set to 1 and SECOND to 2. Then FIRST is increased to 2 and SECOND to 3, and so on through each pair of names in the whole list until eventually FIRST reaches 4999 and SECOND reaches 5000. Following is a short computer program to perform this activity:

1 LET FIRST = 1
2 LET SECOND = 2
3 IF NAME(FIRST) > NAME(SECOND) THEN
   exchange NAME(FIRST) with NAME(SECOND)
4 LET FIRST = FIRST + 1
5 LET SECOND = SECOND + 1
6 IF FIRST < 5000 THEN GOTO 3
7 END
In the above program, steps 1 and 2 initialize the values of FIRST and SECOND to start with the first two names in the list. Step 3 does the actual comparison and exchange of the names if they are out of alphabetical order. Steps 4 and 5 increase the values of FIRST and SECOND by 1, so that FIRST is increased to 2 and SECOND to 3. If the end of the list is not yet reached, then step 6 will cause the program to go back to step 3 again where the comparison of NAME(2) and NAME(3) will now take place. This loop, from steps 3 to 6 and back to 3 again, repeats approximately 5000 times until all successive pairs of names in the list have been compared and possibly exchanged. Through this self-referral process of the flow of the program coming back to itself, the computation progresses and the list of names is gradually sorted into alphabetical order. The loop allows this seven-line program to produce nearly 20,000 individual computational steps to greatly transform a very long list of names. The program may be viewed as a compact structure of intelligence guiding the activity of the computer with a dynamical behavior that resembles the internal dynamics of pure intelligence, the process of complete self-referral in the unified field. The loop in the program is actually a structure of “sequential” self-referral: the process of self-referral only happens after a sequence of computational steps. In the dynamics of pure intelligence, on the other hand, the self-referral is instantaneous and beyond the boundaries of space and time. In Part II of the paper, this property will be shown to be partially responsible for the infinite computational power of the unified field, with which the unified field organizes the whole universe.

The above seven-step sorting program actually does not complete the job of sorting the list of names alphabetically. Even after reaching the end of the list, there still may be names out of order. For example, if the name ADAMS appears at the very end of the list in cell 5000, it will only have moved up to cell 4999 after the program has finished. Thus, to be sure that the entire list is sorted completely, the whole program must be repeated again and again for 5000 repetitions. At first this may seem a monumental computing task, but it can easily be defined by simply adding another outer loop that surrounds the whole program. The symbol COUNT is used to count the number of times that the whole program is repeated, beginning from 1 and eventually ending at 5000:
0 LET COUNT = 1
1 LET FIRST = 1
2 LET SECOND = 2
3 IF NAME(FIRST) > NAME(SECOND) THEN
   exchange NAME(FIRST) with NAME(SECOND)
4 LET FIRST = FIRST + 1
5 LET SECOND = SECOND + 1
6 IF FIRST < 5000 THEN GOTO 3
7 LET COUNT = COUNT + 1
8 IF COUNT < 5000 THEN GOTO 1
9 END

Steps 1 through 6 are the same as the earlier program: they just perform a single pass through the list to compare and possibly exchange each successive pair of names. The loop from 3 to 6 and back to 3 again is repeated approximately 5000 times during a given pass. However, after the first pass is done, the value of COUNT increases to 2 in step 7 and then a new loop is created from step 8 back to step 1. This outer loop causes the whole inner loop to start again a second time, resulting in a second pass through the whole list of names to compare all successive pairs of names again. The outer loop is itself repeated about 5000 times and the inner loop is therefore repeated about 5000 X 5000 times, or nearly 25 million times! This program to sort 5000 names alphabetically would run on a computer for approximately 10 minutes to 1 hour depending on the speed of the particular computer. The use of what are called nested loops has created an extremely compact yet powerful nine-step program that has the concentrated intelligence to guide the computer in performing 25 million computational steps, all leading toward the goal of sorting a list. This double level of self-referral has increased
the computational power of the program, and again is an example of the principle of creation through self-referral located by Maharishi Vedic Science in the self-interacting dynamics of the unified field of natural law. This important control structure found in computer programs, the loop, is a limited reflection of the internal dynamics of the unified field, which is the source of all of computer science.

The amount of time taken by a particular computer to run the above sorting program depends on how long it takes to execute each loop. Even though the computer executes each one very quickly, it still takes some time for them all to finish. Time is required for computation because the self-referral process of a loop takes place through a sequence of steps in “time.” Since the self-referral in the unified field is instantaneous, one could predict that if the unified field can be considered as capable of “computation,” then this computation could take place instantaneously with no time required for computing. This theme will be thoroughly discussed in Part II of the paper.

1.6 The Evolution of Computer Technology

History has shown that knowledge of deeper and more comprehensive levels of natural law always leads to more powerful technologies with wider application. Knowledge of the mechanical level of natural law during the nineteenth century led to the industrial revolution, and knowledge of the deeper, more refined level of electromagnetism resulted in the electronics and computer technology of the twentieth century. The recent discoveries of more fundamental quantum fields, and especially the unified field, present the possibility of technologies much more powerful and comprehensive than current computer technology. Even the great speed of today’s electronic computers cannot compare to the infinite frequency discovered by modern physics in the dynamism of the unified field. Functioning in the domain of quantum geometry, the internal dynamics of the unified field does not involve movement in the classical field of space-time differences (Hawking, 1984). As technology advances, it is going to begin to utilize this aspect of the unified field more and more, where space-time differences are not involved and therefore no loss of time occurs.

The tremendous growth of computer technology during the past twenty years has been based on the great speed and small size of the
integrated electronic circuit chips used in computers. Throughout the history of computer technology, there has been a continual evolution toward utilization of smaller time and distance scales from which greater computing power becomes available at a lower cost. The first commercial computing devices began to be manufactured during the nineteenth century: mechanical calculators capable of performing addition, subtraction, multiplication, and division using complex systems of mechanical gears and wheels. Even though some of the fundamental concepts of general-purpose computation were known at that time, including the concept of a computer program, general-purpose computers could not be developed because mechanical technology based on knowledge of the mechanical level of natural law was not sufficient to realize the refined functioning needed by computers. One of the first general-purpose computers, the Harvard Mark I built in 1940, actually used a combination of mechanical and electrical technology. The central computational element of the Harvard Mark I used to do arithmetic was the electromechanical relay switch, which has two states as shown in Figure 2: switch open or switch closed.

![Relay Switch Diagram](image)

Figure 2. The central computational element of the Harvard Mark I computer, built in 1940, was the electromechanical relay switch. The relay has a metal armature with the ability to move up and down to control the flow of current through the relay.
Figure 3. The vacuum tube performed the same switching function as the relay but in a purely electrical manner, thereby making it much faster and less expensive. A vacuum tube has three major components—cathode, grid, and plate—surrounded by a vacuum enclosed in an outer container made of glass. Heating the cathode causes it to emit electrons, which flow through the grid and are absorbed by the plate.

The movement of the armature is electrically controlled by an electromagnet, but the movement is purely mechanical in nature. The Harvard Mark I also used mechanical counter wheels to store numbers during computations. It could multiply two ten-digit numbers in three seconds.

Both the relay switches and the mechanical counter wheels were quickly replaced in the 1940s by the vacuum tube, which could perform the same switching function as the relay and the same storage function as the mechanical counter wheels in a purely electrical manner. Figure 3 shows a schematic cross-section of a vacuum tube with its
three major components—cathode, grid, and plate—surrounded by a vacuum enclosed in an outer container made of glass. Placing a voltage on the cathode and heating it at the same time causes it to emit electrons, which are drawn through the vacuum and are absorbed by the plate at the top of the tube, just as an electrical current can pass through a relay switch in the response to a voltage. In the vacuum tube, the grid can be used to perform the “switching” function, to either open or close the switch to the flow of current. This is illustrated in Figure 4. When a positive voltage is placed on the grid, the negatively charged electrons are attracted and may pass through the grid to be eventually absorbed by the plate at the top, forming a flow of current. In this case the switch is closed (see left side of Figure 4). Placing a negative voltage on the grid as shown on the right side of Figure 4 causes the grid to repel the electrons. The electrons then collect in a cloud between the cathode and grid and cannot flow through the tube. Thus there is no current flow permitted—the switch is open.
The first completely electronic general-purpose computer was the ENIAC (Electronic Numerical Integrator And Calculator) built at the University of Pennsylvania in 1946 from 18,000 vacuum tubes. It could do arithmetic a thousand times faster than the Harvard Mark I because it used electronic rather than mechanical components—computer technology had moved from the mechanical level of natural law to the more refined electronic level of natural law. Many of the subtle and powerful properties of the electromagnetic field could now be used for computation, including its ability to propagate information at the speed of light and make rapid changes with great flexibility. However, the large size and high power requirements of vacuum tubes did not allow them to take full advantage of the computing potential of the electromagnetic field. The ENIAC weighed over 30 tons, and burnt-out vacuum tubes had to be replaced constantly to keep it running. During the 1950s, the vacuum tube was replaced by a much smaller and faster electronic device called the transistor, made from silicon, a semiconductor material. By adding small quantities of chemical impurities such as boron or indium to the silicon, free negative or positive charges could be created capable of carrying current. As shown on the right side of Figure 5, a transistor is formed by two regions with free negative charges (collector and emitter) and one region with free positive charges (the base). By changing the voltage at the base, the transistor can perform the same switching function as the vacuum tube but much more quickly and reliably. Furthermore, it is less than one tenth the size of a vacuum tube.

In the 1960s, a new generation of powerful electronic computers was developed, which used printed circuit boards with discrete electronic components: transistors, resistors, and capacitors. The evolution from vacuum tubes to transistors represented a movement toward smaller time and distance scales for computing and allowed more of the computing potential of the electromagnetic field to be realized. During the late 1960s, many individual transistors began to be fabricated on one single piece of silicon. This was the beginning of the age of the integrated circuit. Progress continued during the 1970s with larger and larger numbers of transistors (and other electronic components) being fabricated on each small chip. The result was that the transistors became increasingly smaller and faster, allowing computers with greater computational speed, increased memory storage, and a wider
Current Flow

Figure 5. In a transistor, free electrons in the emitter flow through the base to the collector, forming an electric current. By changing the voltage at the base, the transistor can perform the same switching function as a vacuum tube but much more quickly and reliably. The transistor is also much smaller, consumes less power, and is less expensive than the vacuum tube.

range of computational abilities at a lower cost. Typical integrated circuit chips during the mid-1980s have several hundred thousand or even one million transistors on a single chip of silicon one quarter of an inch square. Current feature size in Very Large Scale Integrated (VLSI) circuit chips is as small as 1 micron (10^{-4} centimeters), and transistor switching times are approaching 10^{-10} seconds. The electromagnetic field itself operates at fundamental distance scales as small as 10^{-14} centimeters, so the full potential of that field has not yet been realized and progress is continuing in integrated circuit technology.

The recent discoveries in quantum physics of more fundamental quantum fields may open the possibility in the future for more powerful computing devices that utilize the potential of these deeper levels of natural law. As illustrated at the bottom of Figure 1, the electroweak field unifies electromagnetism and the weak nuclear force into one force field functioning at the level of 10^{-16} centimeters. This electroweak force and the strong nuclear force are combined in grand unification at the scale of 10^{-29} centimeters. Unified quantum field theories have provided a framework for a completely unified understanding of natural law that
unifies all the fundamental force fields and matter fields. The unified field of natural law, according to quantum field theory, operates at the ultimate point-like scale of distance \(10^{-33}\) centimeters and time \(10^{-43}\) seconds (Hagelin, 1987). If a technology could be developed to draw upon this most fundamental level of natural law, then such a technology would make mankind the master of all the forces of nature and give us the ability to utilize the full potential of natural law.

The continuing progress in quantum physics and the natural evolution of computer technology toward smaller and faster circuits, utilizing finer laws of nature, will gradually lead computer science toward incorporating knowledge of the unified field. Recent breakthroughs in high temperature superconductors are one example of a new development that could potentially lead to a step in this direction, by allowing much higher circuit densities and reducing gate switching times by several orders of magnitude. It is interesting that this phenomenon of superconductivity results when the property of “frictionless flow,” usually found only at the quantum mechanical level of natural law, is brought to ordinary time and distance scales. Although the exact mechanics of the phenomenon are not yet fully understood, it is certain that the superconductivity results from the creation of a macroscopic quantum state, which brings these quantum properties of matter from finer levels up to the gross physical level. The result is a “frictionless flow” of electrons through the superconductor with no resistance and therefore no generated heat, thus removing two of the major barriers to smaller and faster integrated circuits. Although superconductivity is still operating within the range of the electromagnetic field, it is an example of how knowledge of deeper levels of natural law approaching the unified field will lead to progress in computer technology.

1.7 Technology of the Unified Field

Even though the organization of computer software and hardware has undergone tremendous evolution since the early electronic computers of the 1940s and 1950s, the electronic switch has remained one of the central functional computing elements. A careful analysis of the activity of the transistor switch as illustrated in Figure 5 shows that it reflects some of the fundamental dynamical properties of the unified field of natural law. The regulation of the electromagnetic field by the transistor
is parallel to the self-regulation of the unified field during its process of manifestation. A more thorough analysis of this parallel will lead to insights into future possibilities for computer technology.

According to Maharishi Vedic Science, the whole mechanics of creation of manifest activity from within the unmanifest structure of the unified field is expressed by the word *Akshara*. The first syllable “A” represents the fullness of all possibilities of natural law in its completely unified state. Since it is the field of all possibilities, this fullness must also contain within itself the value of emptiness. The second part of the word, *Kshara*, means “collapse.” The fullness of “A” collapses down to the point value of emptiness. By becoming aware of its own point value the fullness of “A” then begins to raise this value of emptiness to fullness again through the process of expansion and evolution from this point. This mechanics of creation of activity from within the unified field is illustrated schematically in Figure 6. The origin of knower, process of knowing, and known within the structure of the unified field is also explained by *Akshara* as illustrated in Figure 6. The fullness of the unified field as a knower (represented by “A”) becomes aware of its own point value as the known through a process of collapse (*Kshara*), the process of knowing. This three-in-one structure then gives rise to the manifest activity of natural law and the process of evolution.

This eternal dynamics of creation of activity from the unified field is reflected to a limited extent in the functioning of the transistor, the fundamental electronic circuit element of the electronic computer. As shown in Figure 7, a voltage is applied at the emitter, representing the “fullness” of the electromagnetic field. A voltage is simply the unmanifest potential of the electromagnetic field containing the potential for all modes of manifest electrical activity. In fact another name for voltage is electrical potential. This fullness of the electromagnetic field then collapses down to the point value at the base of the transistor where it no longer has any effect due to the electrical properties of the base. A small voltage applied directly to the base from below will determine whether the electrons are permitted to flow across to produce a current in the collector. Thus the base is the fundamental controlling point to regulate the pure unmanifest potentiality of voltage into pulsations of electrical activity. In Maharishi Vedic Science, the point value illustrated in Figure 6 also has this fundamental controlling power, and the
Mechanics of Creation from the Unified Field

Figure 6. The word *Akshara* expresses the mechanics of creation of manifest activity within the manifest structure of the unified field. The fullness of all possibilities represented by the first letter “A” collapses to its own point value of emptiness (*Kshara* means “collapse”). When fullness becomes aware of its own point value, the process of evolution begins and raises this point back to fullness.

Mechanics of Creation from the Electromagnetic Field

Figure 7. The mechanics of creation expressed by *Akshara* are reflected to a limited extent in the functioning of the transistor. The full potential of the voltage applied at the emitter collapses down to a point value in the base where the flow of current is regulated.
collapse of fullness to the point is the unmanifest origin of creative intelligence that gives direction to all activity in the universe.

In the functioning of the transistor, the voltage contains the total potential of all activity of the transistor. Thus, the range of computational power of the transistor is limited by the range of computing possibilities inherent in the electromagnetic field. As the transistors have become smaller and faster they have been able to utilize more of this potential of the electromagnetic field, and future generations of computers that utilize superconducting devices may unfold even more of the potential of the electromagnetic field. Eventually as technology advances it may be possible for computing devices to begin to directly utilize the electroweak field, the field of grand unification, or even possibly the unified field. One might imagine some kind of computing element like a transistor but having the electroweak field applied to one end instead of the electromagnetic field, and then somehow regulating the potential of this electroweak field to produce much more powerful computation than is possible in any electronic computer. Although this is very speculative, the continued investigation of the properties and dynamics of these more fundamental quantum fields could eventually lead to computing technologies that operate on these smaller time and distance scales approaching the ultimate level of the unified field.

Figures 6 and 7 illustrate two basic requirements for the expression of activity from pure potentiality: access to the full potential of the field and some mechanism to regulate and direct the flow of that potential. Current mathematical formulation of unified quantum field theories is not sufficiently developed to indicate any practical means of gaining access to the full potential of the unified field or of regulating its expression. To investigate properties of finer levels of matter, experimental physicists rely primarily on particle accelerators that create influences on small time and distance scales through the production of extremely high energy collisions. However, even the largest planned accelerators of several miles in diameter can only begin to go beyond the electroweak level and are at least fifteen orders of magnitude too weak for the unified field. Fortunately, a more complete understanding of the unified field available from Maharishi Vedic Science provides a practical technology for utilizing the unified field: a technology of
consciousness. Identification of the unified field of natural law with pure consciousness suggests that a technology of the unified field is possible through a technology of consciousness. Vedic Science provides a practical technology, the Transcendental Meditation and TM-Sidhi programs, whereby the individual mind may directly experience and utilize the full potential of the unified field. Since the mind is itself a localized set of excitations of pure consciousness, it is possible for the mind to simply and naturally settle down to experience its own least excited state, which is the unified field.

Thus the human mind itself becomes the means to enliven the full potential of the unified field. The mechanism to regulate and express this potential into specific modes of activity is the human nervous system. Once the individual mind begins to experience the unified field on a regular daily basis through the Maharishi Technology of the Unified Field, the functioning of the nervous system is gradually refined and purified to allow the full expression of the unbounded potentiality of the unified field into concrete thought and action. The neurons of the individual brain then begin to function as “transistors” to regulate the flow of the unified field into manifest activity within the individual and within the environment as well.

Current physiological understanding of the functioning of the neurons of the brain has focused mainly on the electrochemical aspects of their behavior. However, according to an aspect of Vedic Science called Maharishi Āyurveda (the science of life), every part of the body is intimately connected with consciousness, and through the use of specific techniques to remove impurities, this connection may be made so perfect that the level of pure consciousness is directly and fully expressed by the nervous system at all times. Due to misinterpretation and partial loss of this ancient knowledge during the past few thousand years, the practical technology of the unified field has not been available, except perhaps to a few privileged individuals living deep in the Himalayas. However, Maharishi, through the inspiration of his teacher Swami Brahmananda Saraswati, has been able to revive and properly reinterpret the ancient Vedic tradition of knowledge about pure consciousness, express it in modern language, and make it available throughout the world (Maharishi Mahesh Yogi, 1966). This has resulted in the current widespread availability of the Transcendental Meditation and
TM-Sidhi programs for allowing any individual to simply and naturally experience pure consciousness, the unified field of all the laws of nature (Maharishi Mahesh Yogi, 1986).

1.8 The Transcendental Meditation Program and Computer Science

One of the important aspects of Maharishi Vedic Science is the Transcendental Meditation (TM) program, an effortless mental technique to allow the conscious mind to naturally settle down to its own least excited state, where it is fully identified with the unified field of all the laws of nature. The Transcendental Meditation technique is practiced for twenty minutes twice daily, sitting comfortably with the eyes closed, and requires no change in lifestyle or beliefs. Through personal instruction by a teacher trained under the supervision of Maharishi, the individual learns during a four-day course how to practice the Transcendental Meditation technique and simply experience the unified field. According to Maharishi Vedic Science, regular practice of the Transcendental Meditation technique refines the mind and body so that they are increasingly able to maintain contact with the unified field during daily activity and to make use of its infinite creative potential for improving the quality of life. After some time, the individual is eligible for an advanced phase of the Transcendental Meditation program, the TM-Sidhi program, which helps to accelerate this process of integration of the unified field into activity (see section 2.9 for a further discussion of the TM-Sidhi program). Maharishi Ayurveda also prescribes additional techniques which are designed to help create perfect health in the body and thereby increase the infusion of pure consciousness into the daily life of the individual, thus improving all aspects of life.

Hundreds of research studies conducted during the past fifteen years on the Transcendental Meditation program have documented the unique psychological, physiological, and sociological changes that result from regular practice (Orme-Johnson & Farrow, 1977). An integrated set of physiological changes takes place during the Transcendental Meditation technique that indicate a fourth major state of consciousness, different from the usual states of waking, sleeping, or dreaming (Orme-Johnson, Wallace, Dillbeck, Lukenbach, & Rosenberg, 1979). Regular experience of this state produces certain long-term changes in
the physiology, such as release of stress (Goleman & Schwartz, 1976), improved neuromuscular integration (Warshal, 1980), and improved sensory perception (Martinetti, 1976). Other studies show increased learning ability (Dillbeck, Aron, & Dillbeck, 1979), increased intelligence growth rate (Tjoa, 1977), and improved memory (Miskiman, 1977). All of these studies taken together indicate a long-term culturing of the nervous system toward superior functioning and increased expression of intelligence.

These research studies help corroborate the subjective experiences of individuals practicing the Transcendental Meditation program, who report increasingly clear and profound experiences of the unified field and a growing ability to utilize the creative potential of the unified field in daily thought and activity. Through the regular practice of the Transcendental Meditation program over a period of years, the brain of the individual is cultured to maintain contact with the unified field—the source of all areas of computer science (see chart of Figure 1). Thus the practice of the Transcendental Meditation program is an important aspect of unified field-based computer science because it takes the science from mere intellectual analysis to direct experience of the unified source and foundation of computer science. The universal principles of functioning of pure intelligence are directly experienced by the individual and therefore more clearly understood as the source and basis of all the principles of computer science. The specific technical ideas found in all areas of computer science can then be studied as limited expressions of the eternal dynamics of the unified field functioning within itself at the basis of the universe. This can begin to take computer science from a limited technology of creating electronic computing devices to a universal science of computation that studies the general principles of computation found in all areas of natural law. Part II of this paper will explain and develop this theme in more detail and give many examples of specific concepts from traditional computer science in relation to universal dynamical principles of computation found in the self-interaction of the unified field.
Part II:
Computing Technology
of The Unified Field

2.1 An Expanded Definition of Computation

Ever since its inception, the field of computer science has been continuously expanding to incorporate new areas of knowledge from other disciplines. The theory of computation began as an aspect of mathematical logic and number theory in the 1930s with the work of Turing (1936) and Kleene (1936) in formulating mathematical models for the process of computation. Turing developed a definition of abstract automata called Turing machines, and Kleene defined recursive functions, which later proved completely equivalent to Turing machines in computational power. The practical field of computer technology began around the same time as an aspect of electrical engineering with the first electronic computers being built in the 1940s. Throughout the 1940s and 1950s, computing was considered to be a part of either mathematics or electrical engineering, and it was not until the 1960s that computer science was recognized as a separate discipline in its own right. Since then the boundaries of the field have been growing to encompass many aspects of science, engineering, and management.

In the late 1960s, it was found that software development of large programs was an extremely costly, time-consuming, and error-prone activity. This resulted in the growth of a new aspect of computer science called software engineering, drawing on concepts from organizational and personnel management (Brooks, 1974). In the late 1970s and 1980s, the growing importance of the human-computer interface has been recognized, resulting in the adaptation of principles and techniques from psychology into mainstream computer science. Throughout its history and especially in the past five years, there has been a growing movement in artificial intelligence (a branch of computer science) toward the study of the functioning of the human brain as a guide for future methods of computation. Also, as the miniaturization of integrated circuits has continued during the 1970s, the field of semiconductor physics has become an important aspect of computing technology. Thus, computer science has always been an interdisciplin-
ary field of knowledge and is recognizing the need to utilize still other areas of knowledge to promote continued progress.

There is a growing trend in computer science toward a more expanded definition of “computation” beyond current electronic computer technology. John Hopfield (1987), one of the leading researchers in the area of neural networks, recently defined computation as evolution of a system in “state space.” He presents a diagram similar to that shown in Figure 8 where the initial state of the system is represented as a point in an $n$-dimensional state space, and each dimension represents the range of values of some specific system variable. For the physical system of a gas in a container, the state space might consist of three variables: temperature, pressure, and volume. For a neural network with

Figure 8. John Hopfield (1987) defines computation as evolution of a system in state space from an initial to a final state. This figure shows only two dimensions in the system state space. In general the system will evolve in an $n$-dimensional state space with each dimension representing the range of values of some particular system variable.
one hundred neurons, the possible states of one neuron would be one dimension in the state space of the whole system; whereas for an electronic computing circuit the possible states of each transistor would be one dimension in the overall system state space. As illustrated schematically for a simple system with a two-dimensional state space, Hopfield defines computation as letting an initial point in state space evolve through a path to a final point in state space. The underlying system itself may be electronic, biological, or physical. In his research, Hopfield has been able to develop useful models for electronic computing from the behavior of neurons in living systems and from the physical behavior of inanimate systems such as the self-organization of magnetized atoms in a material as it is cooled (Hopfield, 1982). There are many examples in the computer science literature of such natural processes, both in living and nonliving systems, that have served as useful models for electronic computing.

This trend of expansion in computer science is taking it away from being simply a set of techniques for electronic computing toward becoming a true science of computation that studies the fundamental laws governing computational processes in all aspects of natural law. As more of these natural laws of computation are discovered, progress in electronic computer technology will be enhanced, as is illustrated by the contribution that neural network research has already made to computer technology. From this point of view, the knowledge and experience of the unified field of natural law has great relevance to computer science and could make major contributions to the understanding of computational processes in nature as a basis for a universal science of computation. Part II of this paper presents the proposal that the study of the “computational” aspects of the functioning of the unified field should be included as a central part of the discipline of computer science and will help the discipline to locate universal principles of computation in natural law. As the ultimate source and driving force behind all aspects of evolution in nature, the unified field must contain all dynamical principles of computation in seed form. Since the complete knowledge and experience of the self-interacting dynamics of the unified field is available from Vedic Science, this could prove to be a useful source for beginning to formulate universal laws of computation in their simplest and most general form.
The first step in this process of formulating a universal science of computation is to compare some of the fundamental aspects of current computer technology to the dynamical principles of the unified field as described in Maharishi Vedic Science. Part II of this paper presents some of the preliminary ideas in this area developed in research during the past few years. These ideas are not presented as final and definitive but as attempts to forge a link between existing electronic computer technology and the knowledge of the unified field from Vedic Science in an effort to secure a foundation for a computational science of natural law. Sections 2.2 and 2.3 give an overview of neural computing networks and compare the overall structure of the human body to the architecture of an electronic computer. Section 2.4 introduces and describes one of the most fundamental concepts of traditional computer science: the algorithm, a concrete expression of intelligence that guides the activity of computing. An attempt is made in sections 2.5 through 2.7 to locate the concept of “algorithm” in the dynamics of the unified field and analyze the language in which these algorithms are structured. The practical technology for the human nervous system to utilize these algorithms of natural law is presented in sections 2.8 and 2.9. A deeper consideration of the properties of the unified field in sections 2.10 and 2.11 reveals that it is self-sufficient in its computational power. Moreover, all other computational systems such as the human nervous system or the electronic computer are simply limited reflections of that cosmic level of computing in the unified field. Finally, the applications of unified field-based computer science to the field of education are considered in section 2.12.

2.2 Neural Computing Networks

The similarity of the functioning of the human brain to that of the electronic computer has long been recognized and studied. In fact, the original decision to use the binary system of on and off functioning in the early electronic computers was strongly influenced by the knowledge of the on and off firing of neurons in the human brain. During that time, a new field of scientific study was developed, which Norbert Wiener called “cybernetics” (1954), the comparative study of electronic computers and the human nervous system. Today this field is considered by many to be an aspect of computer science. In addition, new
areas of related research in computer science have developed such as neural networks, which examines the properties of nerve cells as a basis for designing powerful computing devices. This section presents some of the important concepts of neural networks as an introduction to the consideration of the human body as a computer.

In the early 1940s, while the first electronic computers were being built, an attempt to understand some of the computing aspects of the human nervous system appeared in a famous landmark paper by McCulloch and Pitts (1943). Minsky (1967, chapter 3) gives a complete discussion of the cells or neurons of McCulloch and Pitts as a universal model of the process of computation. The McCulloch and Pitts cells were designed as an oversimplified model for some aspects of the functioning of human nerve cells with the purpose of investigating the construction of complex computing machines from simple cells, whether that machine is the human brain or an electronic computer. Minsky (1967) represents each of these cells graphically as a circular figure with one output fiber, which may branch out after leaving the cell to act as an input connection to other cells. Two types of input connections are allowed: an excitatory input (represented graphically as an arrow) and an inhibitory input (represented graphically by a small open circle). A cell may have any number of input connections (see Figure 9 for three examples of such cells).

At each moment the cell is either firing or quiet. A firing cell produces a pulse on the output fiber, whereas a quiet cell produces no output pulse. An output pulse is transmitted to the input of other cells and influences whether they enter the firing or quiet state. Excitatory inputs to a cell influence it toward firing, while inhibitory inputs influence the cell toward being quiet. The exact firing rule for each cell is determined by the number written in the center of the cell (see Figure 9). If any inhibitory input is firing, then the cell remains quiet. However, if no inhibitory input is firing, then the cell will begin to fire if the number of firing excitatory inputs equals (or exceeds) the threshold number written in the center of the cell. In Figure 9, the top cell will fire if both inputs are firing. The center cell will fire if the inhibitory input is not firing and the excitatory input is firing. The bottom cell will fire if the inhibitory input is not firing and both excitatory inputs are firing. When the cells are connected into networks as shown in Figure 10,
they will begin to change state in response to the state of their inputs, causing new state changes in other cells. The propagation of these changes throughout the network results in a dynamical activity that closely resembles the electronic logic circuits of conventional computers.

**McCulloch-Pitts Cells**

![Diagram of McCulloch-Pitts cells]

Figure 9. McCulloch-Pitts cells are an abstraction of some aspects of nerve cells in the brain. In the above examples, the input arrows are *excitatory* inputs and influence the cell *firing* state. The inputs with a small circle on the end are *inhibitory* inputs and influence the cell toward a *quiet* state.
Minsky (1967, chapter 3) shows how these “neural networks” of McCulloch-Pitts cells can be used to create all the basic logic circuits traditionally used to create electronic computers, including memory units, decoders, counters, and binary adders. In fact, it is easily shown that simple McCulloch-Pitts cells can be used to create the fundamental \textit{and}, \textit{or}, and \textit{not} gates used as the basis of all computer logic circuits. The cell at the top of Figure 9 performs the same function as a two-input \textit{and} gate. By changing the threshold to 1, it becomes an \textit{or} gate. A \textit{not} gate is created by a one-input inhibitory input to a cell with threshold 0. It is interesting that memory units can be formed through the use of feedback fibers, which connect the output fiber of a cell back to its input as shown in Figure 11. After receiving a firing pulse on the excitatory input \textit{On}, the cell goes into the firing state and, because of the influence of the feedback fiber, it remains in that firing state until
Figure 11. A memory unit may be formed from a McCulloch-Pitts cell by using a feedback fiber that connects a cell’s output back to the input. In this figure, the cell may be placed into the *On* or *Off* state, and will remember that state until changed by a new input signal. This cell is functionally equivalent to a *flip-flop* for storing one bit of information in a conventional electronic computer.

the *Off* inhibitory input turns it off. Thus, the cell “remembers” the most recent *On* or *Off* pulse from its two inputs. This cell is essentially equivalent to what is called a *flip-flop* in conventional electronic logic circuits, the basic unit of memory for storing one binary digit of information (*On* or *Off* corresponding to binary 1 or 0, respectively).

During the past five years, there has been a great upsurge of interest in these types of neural networks as a model for general-purpose computing, especially in tasks involving learning and pattern recognition. The basic definition of the neuron has been generalized to allow large numbers of connections between neurons, more closely modeling the highly connected character of the human nervous system in which each individual nerve cell may have synapse connections from several hundred or many thousands of other nerve cells. Hopfield (1982) defines neurons that have a number (called the *weight factor*) associated with every connection between neurons. The weight factor for each input connection from a firing neuron is added to produce a single sum \( S \). If \( S \) is greater than the specific threshold value \( U \) associated with that neuron, the neuron enters the firing state, otherwise it remains in the quiet state. Positive input connection weights act as “excitatory” inputs by increasing \( S \), and negative weights tend to act as “inhibitory” inputs.
by decreasing the $S$. The neural networks can exhibit the characteristics of *learning* by gradually adjusting the value of the weight factors in response to external inputs from the environment. For a review of current research in neural networks and their application to computing and to understanding cognitive processes in the human nervous system, the reader is referred to Rumelhart and McClelland (1986).

To illustrate the use of neural networks it will be useful to consider a simple application program developed by the Hecht-Nielson Corporation for recognizing handwritten numerical digits from 0 to 9. The digit is written by the human operator on a special digitizer tablet connected to the computer, which records the position of the pen in a two-dimensional array of tiny points called *pixels*. Whatever the size of the digit drawn, the computer reduces it to a block of pixels with height 15 and width 10 for a total of 150 pixels, each of which is either *on* (for black) or *off* (for white). This is illustrated in Figure 12 for the digit 6. Each of these 150 pixels is used to control a single neuron. The value of the pixel determines whether the neuron is on or off. These 150 neurons are shown on the left side of Figure 13 as the inputs to two more layers of neurons in a network that can distinguish the digits 0 to 9. There are 40 neurons in the center for recognizing various important “features” that

![Pixel Block for Digit Six](image)

Figure 12. A handwritten digit is translated by the computer into a 10 by 15 block of pixels, each of which is either *on* (black) or *off* (white).
Neural Network for Recognizing Digits

Figure 13. Each of the 150 pixels (left side of the figure) is connected to 40 “feature” detecting neurons (center of figure). The feature-detecting neurons recognize characteristic features found in handwritten digits such as vertical or horizontal lines in certain positions. The ten “digit” neurons on the right then weigh these features to determine the most likely digit. The network can be made to learn which features are most important and refine its ability to recognize input digits.

can be found in handwritten digits. Typical features might include such things as the following: a vertical line in a specific area of the block, a horizontal line in a specific area, a leftward sloping curve near the top, a small circle at the bottom. Each of the 150 input neurons is connected
to each of the 40 feature-detecting neurons, and each of these 6,000 connections has its own specific weight factor. The process of adding weights is performed at each of the 40 feature neurons as described in the above paragraph to determine whether the feature neuron assumes a state of on or off, corresponding to the presence or absence of that particular feature in the input digit.

Each of the 40 feature neurons is connected to each of the ten-digit neurons shown on the right side of Figure 13. If a particular feature is usually found in a particular digit, then the connection weight is high, causing the digit to turn on. Features never found in particular digits have negative connection weights to those digits, causing them to turn off. For example, the feature of a small circle near the bottom of the pixel block will have a highly positive connection weight for the digits 8 and 6, but highly negative weights for the digits 1, 4, and 7. Other digits like 2, 3, and 9 do not normally have this feature, but might have something that looks similar to a circle near the bottom when they are handwritten quickly. Thus, the connection weights for these digits may be slightly positive, indicating the possibility of these digits. Each of the ten-digit neurons sums the weights of its incoming connections from all the features, and the one with the largest sum is chosen as a prediction for the original handwritten digit.

To program such a digit recognizer in the usual way on a computer, one would have to select the specific character of the forty features and choose the values of over 6,000 connection weights. However, the beauty of neural networks is that they can be programmed to learn all the features and connection weights for themselves! Initially, the network is set up with randomly chosen weights with no set pattern for the features. Then the network is presented with a handwritten digit, say 6. If the network makes a mistake and predicts the digit to be 4, then the connection weights along the connection paths that caused it to select the 4 are reduced. Thus, the likelihood of the network making the same mistake again are reduced. Then another handwritten digit is presented as input, say 2; if the network this time predicts the proper value of 2, then all of the connection weights in the network that most strongly caused this choice are increased to make it more likely in the future that the network makes this same correct choice again. In this way, a long sequence of handwritten digits is fed into the network. After each
digits, the weights are adjusted to make errors less likely to be repeated and correct choices more likely to be repeated. Gradually, the network learns how to recognize the handwritten digits. The author recently saw a version of this network being simulated on an IBM Personal Computer. The network had been trained with 150 sample handwritten digits and was able to recognize a wide range of handwritten digits correctly. All the digits that appeared to the human eye as unambiguous were recognized by the system.

2.3 The Human Computing Hardware

One of the goals of neural network research is to gain a more complete understanding of the computational processes in the brain when viewed as a computing hardware. Rumelhart and McClelland (1986, pp. 3–4), who are leading researchers in this area, state the following:

In our view, people are smarter than today’s computers because the brain employs a basic computational architecture that is more suited to deal with a central aspect of the natural information processing tasks that people are so good at. . . . After examining these points, we will introduce a computational framework for modeling processes that seems well suited to exploiting these constraints and that seems closer than other frameworks to the style of computation as it might be done by the brain.

Having considered some of the research on the relationship between electronic computing and the activity of the human nervous system, it will be interesting to consider the architecture of the computer in relation to the human brain as a whole. Figure 14 shows the nine major levels of hierarchical organization in the electronic computer as they parallel the nine major levels in the human brain. Both the electronic computer and the human brain have their ultimate origin in the unified field. The diagram suggests a correspondence between various levels. We have already seen that the central level of logic circuits in the electronic computer is similar to the level of nerve cells in the brain.

The human body has its basis in DNA, which is located within the nucleus of every cell of the body and contains the total information used to structure every aspect of the body. This corresponds to the basic circuit diagrams for the electronic computer, which are the “blueprints” for the entire structure. DNA gives rise to RNA, which carries the information from the DNA into the cell body where it is expressed in
Levels of Computer Architecture

**Architecture of Electronic Computer**
- Computer Hardware
- Processor-Memory
- Circuit Boards
- Integrated Circuit Chips
- Logic Circuits
- Circuit Elements
- Chip Regions
- Masks
- Circuit Diagrams

**Architecture of Human Brain**
- Human Brain
- Hemispheres
- Lobes
- Tissues
- Nerve Cells
- Cellular Components
- Proteins
- RNA
- DNA

**Unified Field**

Figure 14. The nine major levels of electronic computer architecture parallel the nine major levels of organization in the human brain. The DNA at the source of the human brain and the fundamental circuit diagrams of the electronic computer both have their ultimate origin in the unified field of all the laws of nature. The human brain has the enormous advantage that through the mind it can contact and utilize the full potential of the unified field. In this way the human brain hardware can use the cosmic software inherent in the unified field.
the form of proteins, the fundamental material from which the body is built (see Figure 14). Correspondingly, in the electronic computer, the circuit diagrams are used to create *masks*, which are necessary in the process of integrated circuit fabrication to create the proper patterns of *chip regions* that combine to form the fundamental circuit elements of the computer. These circuit elements can be compared to the cellular components in the human brain, built from the proteins, and used to form the cells, the basic building blocks of the body. The cells combine to form nerve tissue. In the electronic computer, the circuit elements combine to form logic gates, and the gates combine to form logic circuits, which are contained in large numbers on each integrated circuit chip of the computer.

These integrated circuit chips are the basic physical components of the electronic computer, and behave in a manner parallel to the nerve tissues in the brain. The tissues combine to constitute the various lobes of the brain, including the frontal lobes, occipital lobe, temporal lobe, and parietal lobe. In the same way, the integrated circuit chips in the electronic computer are put together on large printed circuit boards, where they are interconnected as major functional units. The lobes of the brain form the hemispheres (left and right), one of which is predominantly engaged in analytical activity and the other in synthetic activity. In a parallel structure, the circuit boards of the electronic computer are used to form the processor (for doing calculation) and the memory (for storing information). The processor and memory are then combined with a power supply, an outer case, and input-output devices to allow concrete physical expression of the computing activity. In a similar manner, the human brain is provided with a source of energy through the digestive process, an outer enclosing skeleton, and organs of sense and action to act as input-output devices.

Research has already shown that knowledge of the computing in the human nervous system can contribute to progress in computer technology. Thus, the study of the structure and functioning of the human brain and human body will become increasingly important to the future of computer science. From this point of view, Maharishi Vedic Science becomes significant in computer science because it has a profound influence on the computational abilities of the human brain. As already mentioned in section 1.6, the Transcendental Meditation
program, one of the important aspects of Maharishi Vedic Science of the unified field, has been shown to increase learning ability (Dillbeck, Aron, & Dillbeck, 1979), increase intelligence growth rate (Tjoa, 1977), and improve memory (Miskiman, 1977), as well as improve the general quality of functioning of the nervous system and body as a whole (Orme-Johnson & Farrow, 1977). A more complete understanding of the physiological changes that occur in the body as a result of the Transcendental Meditation program could lead to progress in developing more powerful and efficient models of computing, with potential application in electronic computer technology.

One of the most interesting physiological phenomena in this regard is the increase in brainwave coherence that occurs during the Transcendental Meditation program (Orme-Johnson & Haynes, 1981). By measuring voltage fluctuations at various positions on the scalp using electrodes, waves of electrical activity in the brain can be recorded as the EEG patterns with which most people are familiar. A characteristic phenomenon is found during the Transcendental Meditation program—called coherence—in which the wave patterns in different parts of the brain are synchronized with each other. Even widely separated parts of the brain display an increase in coherence during the Transcendental Meditation program indicating a more integrated style of functioning of the brain as a whole. Regular practice of the Transcendental Meditation program over a period of time is found to gradually increase EEG coherence even during that part of the day when the individual is not practicing the Transcendental Meditation program. Orme-Johnson and Haynes (1981) found that high EEG coherence is strongly correlated with creativity, and Nidich et al. (1983) concluded that EEG coherence is significantly correlated with a higher stage of moral reasoning.

Research has found that an overall style of functioning with properties similar to sleep and dreaming can be simulated in neural networks to help improve and refine their computational abilities. Hopfield, Feinstein, and Palmer (1983) state the following in their paper describing how to improve memory in neural networks: “Although our model was not motivated by higher nervous function, our system displays behaviors which are strikingly parallel to those needed for the hypothesized role of ‘unlearning’ in rapid eye movement (REM) sleep.” With the
knowledge that EEG coherence during the Transcendental Meditation program is highly correlated with improvements in computational abilities of the human brain, neural network researchers might well investigate phenomena of global coherence in the hope of discovering some universal laws of computation not yet known.

This example of the physiological phenomenon of EEG coherence is given in order to indicate that a better understanding of the physiological and psychological effects of the Transcendental Meditation program in refining the computing processes in the individual could suggest fruitful directions for research in computer science. Although the exact physiological effects of the Transcendental Meditation program are not yet fully known and are still the subject of scientific research, the ancient Vedic tradition (from which the Transcendental Meditation program is derived) contains a complete knowledge of the exact mechanics of the Transcendental Meditation program and its influence on the mind and body of the individual. In the following sections of the paper, we will consider some of this knowledge as expressed by Maharishi Vedic Science, and analyze its potential contribution to the field of computer science.

The study of the physiological organization and functioning of the human brain has provided insights into its computational processes and allowed it to be usefully viewed as a computing hardware. This was illustrated by the parallel levels of organization of the electronic computer hardware and the human brain in Figure 14. However, these hardware parallels do not address the issue of software, which in many ways is even more important than hardware in the process of electronic computation. It would be useful to also locate a concept of “software” for the human body, but from the experience of computer software, it is clear that the activity of the software will be hidden deep within and not obvious from an analysis of structure or function. Software in the computer is that which guides the internal activity of the hardware, the unseen “intelligence” that gives direction to the steps of computation.

From this point of view, we might well consider mental processes in the human being as the source of software, since thinking gives overall direction to physiological activities in the body. When the mind conceptualizes the achievement of some specific goal as a series of mental images or thoughts, this may be considered as a “program” that will
manifest itself through the nervous system into concrete physical activities to achieve the goal. One might argue that these mental images or thoughts are just a result of past knowledge and impressions that reside on a deeper level of the mind. Thus, to really locate the most fundamental level of human “software,” the most fundamental level of mental functioning must be considered, where the actual impulses of human intelligence originate. According to Vedic Science, this is the level of the unified field of natural law, the field of pure intelligence at the source of thought, deep within the mind of every individual.

Just as the hardware of the computer is organized in levels as shown in Figure 14, the software also has hierarchical levels of organization with each level being supported by a more fundamental level. Figure 1 distinguishes four major levels: application programs, high-level programming languages, systems programs, and machine language. In the same way, the levels of mental functioning in the individual can be considered as levels of software with the unified field as the most fundamental level giving rise to all the other levels and thus ultimately providing the impulses of intelligence to guide the activity of the brain hardware. According to Vedic Science, however, the individual human mind may function primarily from a more manifest and bounded level of intelligence and therefore may not be able to directly utilize the level of the unified field. This can be remedied by the Transcendental Meditation and TM-Sidhi programs, which bring the mind into contact with the unified field and allow it to utilize the full potential of the impulses of pure intelligence. These impulses of intelligence act as software for the activity of the brain hardware. This idea is summarized in a statement by Maharishi (1983):

Human brain physiology is that hardware of the cosmic computer, which through proper programming, can draw upon the cosmic software of Nature to accomplish anything.

The next several sections of the paper will consider more thoroughly this proposal that impulses of natural law inherent in the unified field can be viewed as “cosmic” software that guides the activity of natural law and can be directly utilized by the individual human brain hardware. If such a viewpoint is valid, then the knowledge of the dynamics of the unified field from Vedic Science could prove useful in helping to formulate universal laws of computation that operate in all com-
putational systems including electronic computers, living systems, and nonliving systems.

2.4 Abstract Algorithms

The most fundamental concept normally considered to be at the basis of computer software is the concept of an abstract *algorithm*, defined as an effective procedure that governs computing activity. One of the earliest known algorithms was formulated by Euclid in 500 B.C. for finding the greatest common divisor of two positive integers $m$ and $n$, which is defined as the largest number that evenly divides both $m$ and $n$.

*Euclid’s Algorithm:* (Knuth, 1973, p. 2)

1. **Step 1:** [Find remainder.] Divide $m$ by $n$ and let $r$ be the remainder.
2. **Step 2:** [Is it zero?] If $r = 0$, the algorithm terminates; $n$ is the answer.
3. **Step 3:** [Interchange.] Set $m$ to $n$, set $n$ to $r$, go back to Step 1.

As an example, consider the use of Euclid’s algorithm for finding the greatest common divisor of $m = 35$ and $n = 20$. In Step 1, $m$ is divided by $n$ to get the remainder $r = 15$. Since $r$ is not zero in Step 2, the algorithm continues to Step 3 where $m$ becomes 20 and $n$ becomes 15. Going back to Step 1, a new value of $r$ is computed as the remainder of 20 divided by 15, that is, $r = 5$. In Step 2, $r$ is still not zero, so the algorithm proceeds to Step 3 again where $m$ becomes 15 and $n$ becomes 5. Then in Step 1, the remainder of $m$ divided by $n$ is zero, so the algorithm terminates in Step 2 with $n = 5$ as the answer. An example of another algorithm was presented earlier in the paper in section 1.5: a sorting algorithm for a list of names.

To communicate an algorithm it is necessary to express it in a particular language, whether an informal one like that used above, or a more precise mathematical language, or one of the many computer programming languages. However, it is important to understand that the abstract algorithm itself transcends all of these diverse concrete expressions: the same abstract algorithm may have an unlimited variety of symbolic expressions in programming languages and also an unlimited variety of physical implementations in computer hardware. All computer software can ultimately be traced to a source in abstract algo-
rithms, and all computer hardware is designed to allow the concrete manifestation of abstract algorithms. In fact, the very architecture of the computer hardware embodies many specific algorithms, and contains the fundamental primitives needed to give concrete expression to any algorithm (Mano, 1982). The theory of computer science also deals with various aspects of the properties of algorithms, including their conceptual basis in pure mathematics, the formal mathematical language needed to define and express them, and the detailed analysis of their properties.

The three essential values that must be present in any abstract algorithm are control, operations, and data. The basic objects found in any abstract algorithm are called the data, which may be numeric quantities as in the Euclidean algorithm, symbolic entities such as words, or combinations of these individual entities into complex data structures such as arrays, lists, or files. The operations represent the basic transformations or manipulations of these data objects, either the combining of existing data to compute new data, the moving or restructuring of data objects, or the sending and receiving of data as input and output of the algorithm. The control aspect of an abstract algorithm is that which determines the actual sequence of operations to be performed, and may be based on the actual data values produced by various operations. The control may also allow many operations to be performed in parallel. Basically, all three values are needed in every aspect of any abstract algorithm: the data objects to be transformed by the operations under the guidance of the control. The three are intimately connected and are continually interacting with each other in each phase of the algorithm, but all three must always be present. Programming languages allow the symbolic expression of the data, operations, and control for specific algorithms as the basis for creating computer software. Computer hardware architecture is designed to physically represent the data objects and to implement the operations of any algorithm under the guidance of the control. The internal functioning of the hardware itself also has these three essential values at its basis: control, data, and operations.

In Euclid’s algorithm shown above, the data consists of the values of $m$, $n$, and $r$ during the computation. The operations on the data are the division (Step 1), the comparison of $r$ with 0 (Step 2), the output of $n$ as the answer (Step 2), setting $m$ to $n$ (Step 3), and setting $n$ to $r$ (Step 3).
The control aspect of the algorithm consists of the sequential execution rule of moving from Step 1 to Step 2 to Step 3, plus the conditional termination of the algorithm in Step 2, and the go back in Step 3. All three (control, operations, data) are necessary for the algorithm to function: any computational activity requires operations, these operations require data to operate on, and the control must be there to determine the sequence of operations. The three values are intimately connected as they interact with each other. This is especially obvious in Step 2 where the operation of comparing $r$ with 0 allows the control to determine whether the algorithm terminates or moves to Step 3.

2.5 The Cosmic Algorithms of Natural Law

As mentioned before, based on the direction of current trends in computer science, this paper puts forward the idea that computing is not just a limited phenomenon of some electronic machines created during the past forty years, but a phenomenon of nature found in many aspects of natural law, including the human nervous system. Thus, it should be fruitful to study the computational aspects of natural law as a basis for a true science of computation with its own fundamental laws distinct from any specific technological application of these laws. This hypothesis that computing is an activity of natural law is supported by a careful analysis of computer applications. For the most part, current electronic computer technology is used for the simulation of various phases of natural law, especially applications of computers in science and engineering. This simulation is done by forming exact mathematical models of the properties of natural law, then encoding these models into programs that guide the activity of the computer.

Due to limitations of electronic computer hardware and the limitations of our mathematical understanding of natural law, the simulation is only a rough approximation of certain simple aspects of natural law, but nevertheless a useful one. For example, the turbulent airflow around an airplane wing may be represented mathematically by a complex system of differential equations that may be approximately solved by Navier-Stokes simulation on supercomputers (Stevens, 1987). Many fields of natural science now have branches devoted to formulating algorithms to simulate or compute specific laws of nature on electronic computers. The example of the airflow around an airplane wing is an
aspect of a field called computational fluid dynamics. Another example is a growing field called computational physics.

There are some examples of computational properties in natural systems that have proved useful as computational models for solving a wide range of problems in seemingly unrelated areas. A primary example is the use of simulated annealing to solve combinatorial optimization problems that arise in operations research, an aspect of business administration (Kirkpatrick, Gelatt, & Vecchi, 1983). When a physical material is cooled gradually from a high temperature, it will naturally organize itself into a low energy state with useful properties of strength and orderliness. This process is called annealing. An example of a computational problem that can be solved through simulated annealing on a computer is the travelling salesman problem in which the goal is to minimize the total travelling distance needed to visit a group of cities one at a time in any order. An energy junction may be set up for the travelling salesman problem with the property that the energy tends to get lower as the total travelling distance between cities becomes less. Then some initial starting choice is made for the path of travel and it is incrementally changed in such a way as to gradually reduce the energy and drift toward a low energy state, just as a physical system will naturally move toward a low energy state.

One of the problems that must be overcome is that these systems have local minima, points where the energy is lower than surrounding points but not as low as other energy minima that are far away. In the formulation of these problems, the concept of an energy landscape is often used: a two-dimensional surface with hills and valleys like a landscape, where the valleys represent low energy states and the hills high energy states. The goal of finding the deepest valley can only be achieved if one has the ability to temporarily increase energy to jump over mountain ranges and locate deeper valleys that may be far away. The ability of a system to make jumps is provided by introducing the concept of the temperature of a system.

In a physical system during annealing, the high initial temperature puts a great deal of energy into the system, allowing it to jump around the energy landscape. As the temperature is reduced, the system naturally reduces the size of its jumps and gradually settles into one of the deepest valleys, a low energy state. In simulated annealing, the tem-
Temperature is simulated with a random number generator that forces the computational system to take random jumps to new possibilities. In the travelling salesman problem, this means that the salesman may have to consider longer trips (as the system climbs an energy hill), but thereby finds a much shorter trip (as the system slides down the other side of the hill into a deep energy valley). As the computation progresses, this temperature variable is reduced in magnitude to simulate “cooling” so the solution gradually settles into a low energy state (short total distance). Note that the reference to a two-dimensional energy landscape is an oversimplification. The energy landscape will be an $n$-dimensional surface with one dimension for each degree of freedom in the system’s solution space.

Many other optimization problems have been successfully solved with simulated annealing including graph partitioning, matching, and placement (Rammal, 1986) with applications in areas such as partitioning and wiring of electronic circuits. This process of minimization in an $n$-dimensional energy landscape has also been successfully used in neural networks as a computational model for memory and pattern classification (Rumelhart and McClelland, 1986). Thus, a phenomenon of natural law (annealing) that allows self-organization in complex physical systems has been understood as a general computational principle in natural law with applications to computational problems in electronic computers and neural networks. This direction of evolution of computer science indicates that it is moving toward a computational viewpoint of natural phenomena: a universal science of computation is beginning to emerge. Since Maharishi Vedic Science provides a complete understanding of the dynamical functioning of natural law at its most fundamental level in the unified field, a consideration of its principles can form a useful starting point for formulating universal principles of computation in natural law.

Through the knowledge of the unified field provided by Maharishi Vedic Science, it is now possible to begin to study the “algorithms” of natural law directly within the unified field. According to Vedic Science, there are guiding structures of intelligence in the unified field that control all the expressed activity of the laws of nature. The following sections of this paper present the proposition that these guiding structures can usefully be considered as “abstract algorithms” of natural
law with properties similar to conventional electronic computer algorithms as discussed in section 2.4.

The three essential values of control, data, and operations can be located in their perfect and totally integrated state within the unified field of natural law, which is the source of every abstract algorithm. It was mentioned in section 1.4 that according to Maharishi Vedic Science, the fundamental values of the unified field are Saṁhitā, which means togetherness or unity, and the three values of Ṛishi, Devatā, and Chhandas, which arise when Samhitā as knower (Ṛishi) becomes aware of itself as known (Chhandas) and thereby creates a process of knowing (Devatā) within the unmanifest structure of the unified field. The internal dynamics of this three-in-one structure is the basis for all functioning of natural law in every aspect of the universe. The English words “knower,” “process of knowing,” and “known” do not, however, capture the full meaning of Ṛishi, Devatā, and Chhandas. According to Maharishi Vedic Science (Maharishi Mahesh Yogi, 1985), Devatā embodies the fundamental impulses of intelligence that govern all activity emerging from the unified field of natural law. Thus, Devatā is the essence of all “operations” in the unified field. Chhandas, as the known, is the more concrete objective value within the unified field, the “data” objects of the unified field. Ṛishi, the knower, is the unmanifest intelligence guiding Devatā and Chhandas, the pure intelligence value at the basis of every aspect of natural law. Thus, Ṛishi is the “control” aspect of the unified field. In its role as “knower,” Ṛishi is the starting point for all steps of evolution and thus controls the direction of the sequential flow of natural law.

In his explanation of Vedic Science, Maharishi (1985) has described the process by which the self-interacting dynamical relationship of Saṁhitā with Ṛishi, Devatā, and Chhandas sequentially gives rise to all the most fundamental laws of nature in the unified field. Saṁhitā is Ṛishi, Devatā, and Chhandas, and these three together are nothing other than Saṁhitā. According to Maharishi, Saṁhitā may be characterized as a three-in-one structure that is continually pulsating within itself from one to three to one at infinite frequency, creating the primordial activity at the basis of natural law. Saṁhitā becomes Ṛishi, Devatā, and Chhandas, and each of the three becomes Saṁhitā. Ṛishi, Devatā, and Chhandas are also constantly being transformed into each other.
through their roles as knower, process of knowing, and known. All of these mutually interacting relationships create new dynamical relationships and continue the process of differentiation and manifestation of unity into diversity. Maharishi has explained that the first phase of this manifestation is the emergence of the Veda, the primordial sounds of natural law that exist within the structure of the unified field itself, and are therefore nonchanging and eternal. These primordial sounds of the Veda, which govern every phase of natural law, can be called the “abstract algorithms” of natural law, structured within the unified field, which govern all activity in the universe.

### 2.6 The Eternal Structure of the Veda

Maharishi Vedic Science provides a detailed description of the structure and internal self-referral dynamics of the Veda. This section will examine some of the major aspects of that structure from the point of view of computer software and abstract algorithms in order to understand more completely the nature of this “cosmic software” of natural law. The most essential core of the Veda, giving rise to all other aspects of the Veda, is called Rk Veda. Rk Veda is traditionally divided into ten parts called Manḍalas, which may be considered analogous to the modules that constitute ordinary computer software packages. In his description of the eternal structure of the Veda, Maharishi (1978, May) has shown that each of the ten Manḍalas is primarily responsible for the manifestation of some specific aspect of natural law.

The first Mandala contains the fullness of knowledge of natural law and is considered as the seed of the entire Rk Veda and the entire functioning of natural law in the universe. The second through sixth Manḍalas contain the laws responsible for governing the five subtle elements, or Tanmatras, forming the basis for all the physical aspects of the universe: Prithivi (earth) Tanmatra, Jala (water) Tanmatra, Agni (fire) Tanmatra, Vāyu (air) Tanmatra, and Ākāsha (space) Tanmatra. Hagelin (1987) has equated these Tanmatras with the five fundamental categories of quantum fields, or “spin types,” responsible for the entire material universe. These are the spin-2 graviton (responsible for the force of gravity), the spin-3/2 gravitino, spin-1 force fields, spin-1/2 matter fields, and spin-0 Higgs fields. The seventh through tenth Manḍalas
are concerned with the subjective aspects of life: mind, intellect, ego, and Self, respectively.

This description from Maharishi Vedic Science closely parallels the mathematical formulation of unified quantum field theories, especially superstring theory, in which all the fundamental force and matter fields of natural law arise from vibrations of a self-referral loop called the superstring (see Green, 1986, for a description of superstrings). In superstring theory, the five quantum-mechanical spin types correspond to massless modes of vibration of the superstring at the basis of natural law. Hagelin (1987) has suggested that the four subjective aspects of natural law governed by the seventh through tenth Maṇḍalas may correspond to massive superstring vibrational modes within the unified field.

The ten Maṇḍalas of Rk Veda exist eternally within the structure of the unified field and contain in seed form all the laws of nature responsible for governing all processes in the universe. To continue the analysis of the Veda as the cosmic software of natural law, these ten Maṇḍalas at the core of the Veda may be considered as the ten fundamental “modules” in the cosmic software package of the unified field of natural law. Figure 15 is a schematic diagram developed by Maharishi (1979) to illustrate the structure of the ten Maṇḍalas. Since this structure exists within the unified field beyond the boundaries of space-time geometry, it is eternal and nonchanging: this eternal structure contains the fundamental knowledge at the basis of the functioning of natural law at all times. In Figure 15 each Mandala is drawn as a circle (the word “Maṇḍala” means “circle”), and this circular structure is partially responsible for the eternal and invincible nature of natural law. Although this property is not shown in Figure 15, each of the ten Maṇḍalas arises from and is connected to one central “point,” which contains the total potential of natural law. Maharishi (1979) explains that this total structure of the ten Maṇḍalas of Rk Veda is contained at each point of the universe. Thus, the total potential of natural law is available at every point of the universe.

It is important to understand, however, that the Veda is not an intellectual formulation of knowledge “about” the unified field. According to Maharishi, the Veda is the structure of natural law within the unified field as directly cognized by the ancient Vedic seers during their direct experience of the unified field (Maharishi International Univer-
A verse of Ṛk Veda itself explains the nature and location of the Veda:

*Richo akshare parame vyoman yasmin devā adhi viśve nisheduḥ*

The verses of the Veda exist in the collapse of fullness, in the transce-ndental field, in which reside all the impulses of creative intelligence, the Laws of Nature responsible for the whole manifest universe.¹ (1.164.39)

According to Maharishi, the one key to Vedic study is the experience of our consciousness as described in another verse of Ṛk Veda:

*Yo jāgāra tam ričah kāmayante*

The hymns seek out him who is awake. (5.44.14)

To be fully “awake” means to be established in pure consciousness—a goal that may be practically realized through the Transcendental Meditation and TM-Sidhi programs.

**Mandalas of the Ṛk Veda**

![Diagram of Mandalas](image)

**Figure 15.** Ṛk Veda, the cosmic software of natural law, is traditionally divided into ten parts called Maṇḍalas, which may be considered as the modules that constitute ordinary computer software packages. The first Maṇḍala contains the fullness of knowledge of natural law. The second through sixth Maṇḍalas contain the Laws responsible for governing the five subtle elements or Tanmatras. Hagelin (1987) has equated these Tanmatras with the five fundamental categories of quantum fields. The seventh through ninth Maṇḍalas govern the subjective aspects of life.

Examining the detailed structure of each Maṇḍala will reveal more about the structure of the programming language of nature used to construct modules. Each Maṇḍala is divided into parts called Sūktas, ranging in number from 70 to 192 in each Maṇḍala (the average is about 100 Sūktas per Maṇḍala). These Sūktas are analogous to subrou-tines or procedures in conventional computer programming languages,
each having its own control (Ṛṣī), operations (Devatā), and data (Chhandas). Each Śūkta consists of five to fifty verses with the average being about ten. The entire Ṛk Veda consists of approximately 10,000 verses. Each verse has an average of ten words and may be considered as a basic “instruction” in the programming language of natural law. The length of a computer program is usually defined as a number of “lines of code,” the total number of instructions in the program. The kernel of the operating system, which is the central controlling software package for all the activity of the computer, may typically be about 10,000 lines of code. Thus, the Ṛk Veda with its 10,000 verses may be considered as the “kernel” of the operating system of natural law.

2.7 The Programming Language of Natural Law

Maharishi (1976) has done an elaborate analysis of the first verse of Ṛk Veda and concludes that it contains the concentrated knowledge of the total functioning of natural law and is the seed for the entire first Mandala and the whole of Ṛk Veda. Considering this first verse as an “instruction” in a programming language, it is possible to locate many of the properties of control, operations, and data ordinarily associated with computer programming languages. The first verse of Ṛk Veda is as follows:

Agnim ile purohitam yagyasya devam ritvijam hotaram ratna dhatamam

The first word Agnim is itself a complete expression of all knowledge (Maharishi Mahesh Yogi, 1976, p. 128) with its own internal dynamical structure having aspects of control, operations, and data. The word Agnim contains in seed form the complete knowledge of natural law, all evolutionary processes, and all structures and activities in the universe. The first letter “A” also has values of Gyaña (knowledge), Gamana (action), Prāpti (achievement), and Moksha (fulfillment) (MIU, 1974, p. 197). Thus the entire source, course, and goal of action is contained in the first letter of Ṛk Veda, “A,” representing the fullness of all possibilities. (Here the letter “A” has the same significance as it did in section 1.6.) There is an aspect of Maharishi Vedic Science that analyzes the meaning of roots contained in the words of the Veda. However, the Vedic sounds have the property that there is an exact correspondence between the sound and its meaning. Therefore, it is possible to gain insights into the meaning of a letter or word just from its sound. The
first letter “A” is pronounced “ah” with a full opening of the mouth and throat that corresponds to its meaning as fullness. The second letter of Agnim is “G” (as in “got”), pronounced by closing the throat and stopping the continuous flow of the first letter “A.” In contrast to the fullness of “A,” the second letter “G” represents emptiness, the unmanifest state of the unified field. Between the fullness of all possibilities in “A” and the unmanifest emptiness of “G”—the two most extreme values of natural law—lies the entire range of natural law.

In the first two letters of Rk Veda, the origin of the binary number system used in electronic computers for all computation can be located: the fullness of “A” is symbolized by the digit 1 and the emptiness of “G” is symbolized by the digit 0. In a computer, all possible information and all possible numbers can be represented as patterns of these 1s and 0s, and all computation consists of transformations on these patterns. Thus, 1 and 0 in the electronic computer represent all possibilities for computation just as “A” and “G” in the Veda contain the seed of all possibilities of natural law. For a further discussion of the meaning of the first word of Rk Veda, the reader is referred to MIU (1974, pp. 197–198).

The dynamics of “AG” are the same as those illustrated for the word Akshara in section 1.6 and Figure 6: this syllable represents the collapse of the fullness of all possibilities to a point of emptiness so that it can reemerge as the processes of creation and evolution on the other side. Since the letter “A” contains the fullness of all possibilities (infinity), it must have within itself the possibility of a point value (emptiness) represented by “G.” The first syllable, “AG,” symbolizes the process of fullness becoming aware of its own point value as emptiness. In this process, fullness will naturally raise this emptiness back to the original fullness, as symbolized by the third letter of Rk Veda, “N,” indicating negation of emptiness. This negation of emptiness leads to the progression of the emptiness of “G” back to the original fullness of “A.” This process of progression is indicated by the fourth letter, “I,” which means to progress or to lead. The continuation of this progression from emptiness to fullness and back again results in a vibration of infinite frequency within the unified field, represented by the final letter, “M.”

The collapse from the fullness of “A” to the emptiness of “G” is the origin of the three values of control, operations, and data, containing all three in a balanced and integrated state. The next three letters
“N,” “I,” “M”) represent control, operations, and data, respectively. The negation, indicated by the letter “N,” controls the manifestation of emptiness back to fullness, and therefore is the primary “control” aspect of all processes in nature. The progression indicated by “I” is the primary “operation” of the unified field, the basis of all operations in the universe. The vibration of the letter “M” is the source of all manifest objects in the universe, the source of all “data.”

In Maharishi’s analysis of the concentrated dynamics of natural law contained in Agnim, many of the fundamental principles of software, hardware, and theory may be located as a basis for beginning to formulate universal laws of computation. Section 2.4 has already shown that control, operations, and data are the foundation of all the algorithms that are expressed in computer software, hardware, and theory. This theme will be further elaborated below with respect to the first verse of Rk Veda: the principles of control, operations, and data will be located in the meaning of the words in this verse.

A careful consideration of the word Agnim also reveals some of the other fundamental dynamical structures at the foundation of computer hardware and theory. Computer hardware is built from electronic logic gates that operate on binary digits (1 and 0). As was shown above, the origin of 1 and 0 can be located in the first two letters of Agnim. In principle, all electronic logic gates can be created from one fundamental gate called the NOR gate as illustrated in Figure 16. The NOR gate has two inputs and one output, all of which may propagate binary digits 1 or 0. The NOR gate begins by taking the “or” of its two inputs and then negating the result (indicated by the small circle in Figure 16). This negation is parallel to the negation value of the letter “N” in Agnim which negates “G” to recreate “A.” The flow of 1s and 0s through the gate from input to output is parallel to the dynamics represented by the letter, “I,” meaning progression or the process of evolution. Thus, the fundamental objects (1 and 0) and the transformational dynamics (the NOR gate) of computer hardware may be located in seed form in the first word of Rk Veda: Agnim.

The chart of Figure 1 has already located the basis of computer theory in discrete mathematics, especially in certain aspects of logic and set theory. The origins of discrete mathematics may be found in the first two letters of the Veda, “A” and “G.” When the unbounded fullness of
“A” is stopped by the restriction of “G,” this represents the first quantification of pure intelligence and is therefore the origin of all numbers and discrete values used in discrete mathematics. In mathematical logic the two truth values, true and false, are the primary discrete values.

\[
\begin{array}{ccc}
 a & b & c \\
0 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 1 & 0 \\
\end{array}
\]

Figure 16. Computer hardware is built from electronic logic gates, which in principle can be created from one fundamental gate called the NOR gate. The operation of the NOR gate is parallel to the dynamics of creation from the unified field as expressed by the first word of the Veda, Agnim.

The value true can be located in the letter “A,” the truth of all possibilities in fullness. The value false arises from the opposite value of “G” in falsifying the fullness of “A.” Mathematical logic deals with rules governing the sequential steps of derivation of theorems from axioms in mathematical theories. This sequential progression has its seed in the flow of the fullness of “A” into the emptiness of “G” as expressed by “NI.” Thus, Agnim may be understood as the source of discrete mathematics and of mathematical logic. Finally, the primary concepts in set theory may also be derived from the internal dynamics of Agnim. The letter “A” in its fullness is also pure “wholeness” in its nature, and thus is the source of the mathematical concept of a set, which is an abstract notion of “collection” or “wholeness.” The letter “G” with its value of emptiness is the origin of the concept of the empty set, used in set the-
ory along with the concept of a set to create all possible mathematical objects. The logical steps of progression of knowledge in set theory as derived from its fundamental axioms are parallel to the dynamics of “NI,” the evolutionary value of the unified field, which brings about steps of progress in the universe. For an excellent discussion of relationship between the foundations of mathematics and Maharishi Vedic Science, the reader is referred to Weinless (1987), whose analysis proceeds along different lines from those presented here.

In Maharishi Vedic Science, the first word of Rk Veda (Agnim) is understood to contain the total knowledge of creation and all processes of evolution found in nature. As such, Agnim should also contain in seed form all the natural laws governing the process of computation, which is just a particular type of evolutionary process in a physical system. Just the brief analysis of the word Agnim presented here indicates that it should prove to be a rich source of ideas for formulating universal principles of computation as a basis for future progress in the field of computer science. It should be emphasized, however, that the primary use of the Veda is not through intellectual analysis of its sounds. As discussed in section 2.6, the verses of the Veda exist in the transcendental field, in the field of pure consciousness, and can only be fully comprehended through direct experience on that level. Only from direct experience of the dynamic of Agnim on the level of pure consciousness can one fully appreciate and understand it as the seed of all creation and evolution in the universe.

The concentrated value of knowledge contained in Agnim is further elaborated in the entire first verse of Rk Veda. It will be useful to consider this first verse as an “instruction” in a programming language, and briefly analyze its computational properties from the point of view of control, operations, and data. The second word of the first verse Île reveals the value of consciousness present in the fullness of Agnim. Consciousness emerges from its full potentiality in Agnim and curves back on to itself, as represented by the word Île (which may be literally translated as “I adore”). This process is well described by Maharishi (1976) in the following paragraph:

By virtue of being awareness, transparent to itself, consciousness emerges from within its pure potentiality (Agnim) and, curving back on to itself (Île), establishes an “observer-observed” relationship within its own
structure. This process of consciousness becoming aware of itself creates an unmanifest space-time geometry within the field of consciousness. The unmanifest space-time curve within the field of consciousness is at the source of spacetime curvature, which Einstein’s general theory of relativity shows to be the basis of all objective creation (p. 129).

As discussed earlier in section 2.5, this observer-observed relationship creates the three-in-one structure of Rishi (control), Devatā (operation), and Chhandas (data) within the unified field. This curving back indicated by the word Īle may be understood as a primordial “loop” of the program of natural law. (There is also a “loop” within the word Agnim formed by the progression of the letter, “I,” leading back to the fullness of “A.”) In the discussion of the sorting program in section 1.5, it was seen that a “loop” in conventional programming languages is a structure in which the flow of control of the program returns back to itself after a sequence of operations. The process of returning back indicated by Īle likewise forms a loop, but it occurs within the unified field prior to the emergence of space-time boundaries, and thus is instantaneous self-referral. The “sequential” self-referral of loops in computer programming languages happens in a very short time (10⁻⁶ seconds), but cannot compare with the perfect self-referral of the unified field beyond the Planck scale (10⁻⁴³ seconds). Thus, the second word of Rk Veda, Īle, can be associated mainly with the “control” aspect of programming language in its role as creator of a loop.

The next three words of Rk Veda, Purohitāṁ Yagyasya Devam, are further elaborations of control, data, and operations, respectively, as contained in concentrated form in Agnim Īle. When consciousness emerges from its state of fullness (Agnim) and becomes aware of itself (Īle), it then becomes capable of controlling and initiating action, as indicated by the third word Purohitāṁ, the inner controller of all action. This word Purohitāṁ is the basis of the “control” aspect of the software of natural law governing the universe. The fourth word Yagyasya expresses the code of action to achieve any goal, the structure of law itself, and thus can most closely be associated with the “data” aspect of programming languages. The fifth word Devam is the impulse of the unified field that spontaneously leads activity in an evolutionary direction, the “operation” aspect of evolution in the universe.
The next three words of Ṛk Veda, *Ritvijam Hotaram Ratna*, also respectively symbolize control, data, and operations, but at a more fulfilled level, resulting in the goal of all computational processes. *Ritvijam* is the non-active, absolute value of creative intelligence that witnesses all activity—the Rishi or “control” aspect of computational processes in natural law. *Hotaram* symbolizes action to yield the highest possible results, and as such is indicative of the “operation” value of algorithms. The absolute silence and action represented by *Ritvijam Hotaram* results in the culmination of all processes of evolution, *Hotaram*, defined as the grace of life in its highest form (Maharishi Mahesh Yogi, 1976, p. 131). This goal of all computational processes embodied in *Hotaram* is the highest value of “data,” symbolic of the final data that is the solution to a given computational problem. Finally, the ninth word of Ṛk Veda, *Dhātamam*, results from the overflow of the fullness and purity of *Hotaram* into the environment. *Hotaram* means the giver of supreme life, and is analogous to the final output of any computation, the fulfilled value of all control, operations, and data in the computation. In this case, it is the radiance of enlightenment making the supreme value of natural law universally available that is the ultimate goal of all computational processes in nature. For a more complete discussion of this first verse of Ṛk Veda see (Maharishi, 1976, pp. 128–131).

### 2.8 Action in Accordance with Natural Law

The purpose of the previous two sections was to discuss some of the computational aspects of the Veda in an attempt to provide a basis for a universal science of computation with application to all types of computational systems in nature, including electronic computers, the human brain, and self-organizing inanimate systems. With respect to computation in the human brain hardware, it was hypothesized in section 2.3 that the mental processes could usefully be considered as playing the role of software by providing overall direction to the steps of computation. Since the unified field (pure consciousness) is the most fundamental level of mental processes, it was decided to investigate the dynamical structure and functioning of the unified field from the point of view of computer software, looking for properties similar to conventional computer algorithms and programming languages. According to Vedic Science, the actual structure of natural law itself exists
eternally within the unified field in the form of the primordial sounds of the Veda, the concentrated state of pure knowledge responsible for guiding the process of creation and all evolution in the universe. Therefore, the structure of the Veda was analyzed in sections 2.6 and 2.7 using concepts from traditional computer software, algorithms, and languages. The overall conclusion is that the dynamical structure of the Veda has computational properties, according to the expanded definition of computation used in Part II of this paper (see section 2.1). Based on this conclusion, the remainder of the paper will use a model of the Veda as the *algorithms* or *programs* of natural law that guide all activity in the universe. The next two sections of the paper will consider the significance of this view of the Veda with respect to computation in the human brain and the practical possibility of raising that computation to its highest value.

If the human mind has all of the perfect algorithms of natural law at its basis, one might wonder why many individuals are constantly making mistakes and are unable to fulfill their desires in life. From the framework derived so far in this paper, this weakness must be due to lack of computational ability in the individual’s brain and inability to fully utilize the algorithms of natural law existing within the structure of the unified field. According to Vedic Science, the direct experience of the unified field provided by the Transcendental Meditation program allows the individual to gain access to those perfect algorithms, or programs of natural law, the Veda, and thereby utilize the infinite organizing power of natural law in daily life. Functioning through the mind of the individual, the algorithms of natural law can then directly guide the computational processes in the human brain hardware, so that an individual acts spontaneously in accordance with natural law.

According to Vedic Science, individual action in accordance with natural law will produce a problem-free life of health, happiness, and success. It is commonly known in the field of medicine (Chopra, 1987) that certain “unnatural” activities such as smoking, improper diet, or overwork can aggravate or cause illness. And it is common knowledge that problems and suffering in life often result from “mistakes,” activity that is not suitable to the particular time and circumstances. Maharishi has generalized this principle to state that all problems, suffering, and ill health result from violation of natural law (Maharishi
Mahesh Yogi, 1966). According to Vedic Science, the laws of nature at their basis are all evolutionary, that is, they naturally promote the evolution of everything in the universe toward higher values. However, when human awareness loses contact with the source of these laws of nature in the unified field, the individual is ignorant of natural law, and therefore will violate natural law. This violation may be unintentional, but nevertheless leads to problems and suffering in life. By reestablishing contact with the unified field through direct experience, the Transcendental Meditation program brings activity back into accordance with natural law, resulting in problem-free and healthy living.

From a computational point of view, the Transcendental Meditation program simply gives the individual access to the programs of natural law within the unified field, which are already computing the activity of everything in the universe in complete harmony without a problem. From those eternally perfect programs of nature, the thought and activity of the individual and society can be computed spontaneously in accordance with all the laws of nature, resulting in life in alliance with the total potential of natural law. The unified field is actually the unmanifest basis of thought of everyone anyway, but by having awareness restricted to localized values of natural law only, the individual cannot make use of the total potential of natural law, and will continue to violate some laws of nature. By regaining contact with the Samhitā of natural law and the total value of the Veda, the individual human brain is then capable of spontaneously utilizing these abstract algorithms of natural law to compute thoughts and actions in accordance with all the laws of nature. This computing happens spontaneously through the simple and natural process of thinking, without any effort or intellectual analysis by the individual.

2.9 The TM-Sidhi Program

It will be useful to consider more deeply the mechanics by which the individual mind becomes capable of utilizing the programs of natural law in the unified field. An advanced aspect of Maharishi Vedic Science, the TM-Sidhi program, allows the individual to become familiar with the basic impulses of natural law in the unified field. From a computational viewpoint, these basic impulses correspond to the primitive instruction types in the programming language of natural law. In the
field of electronic computing, every computer program is written in some specific programming language, as indicated in the chart of Figure 1 by the high-level programming language box across the center. The chart also shows that this diversity of languages has its source in only a few basic concepts and structures shown at the base of the software box. Actually, all computer programming languages are built up from just a few basic types of instructions that embody the fundamental impulses of computer activity. These basic types are used to specify the control, operations, and data of every computer program, which is a concrete realization of some abstract algorithm. Any computer program, no matter how vast and complex, consists of combinations and permutations of these few basic instruction types, which together are capable of bringing out all possible activity of the computer.

In order to create an actual computer program, however, a human programmer is needed to supply that lively intelligence that can conceive of an integrated set of individual instructions to achieve the desired computation and realize the potential of the hardware. Through repeated practice the programmer must gain complete familiarity with the basic instruction types on the quiet levels of inner consciousness, where his awareness is unbounded and is capable of comprehending all possible relationships and combinations of the basic instruction types. By having a thorough knowledge of the basic features of the programming language embedded at the fundamental level of the mind, a programmer can easily create a computer program to accomplish any task within the range of capability of the hardware.

This theme of programming in computer science is reflected in a similar theme of programming in Maharishi Vedic Science. The human brain hardware can be cultured to utilize the cosmic software of natural law, the Veda, and can therefore function as a “cosmic” computer. In Maharishi Vedic Science a few key instructions, called Yoga Sūtras, serve as the programming language to unfold the total potential of natural law for the individual. The Yoga Sūtras represent the fundamental operations of the unified field, which is the source of all the laws of nature that govern the universe. By gaining a thorough familiarity with these few Sūtras in the simplest form of awareness through the daily practice of the TM-Sidhi program, the total potential of these basic
impulses of natural law and all their possible interrelationships become spontaneously available to the individual.

The word “Yoga” means unity and “Sūtra” means aphorism or “thread.” Thus, the Yoga Sūtras are the “threads of unity,” the primordial impulses of the unified field that structure all the abstract algorithms of natural law. The Yoga Sūtras can be considered as the basic instructions of the programming language of natural law, which arise within the self-referral dynamics of the unified field. A new programmer can usually learn the meaning of the individual instructions of ordinary computer programming languages in a few weeks, but it takes several years of experience to become an accomplished programmer. The reason is that the mind must be gradually cultured to perceive various combinations and relationships of these instructions, so that the accomplishment of a given computing task can be formulated in an integrated way as a series of instructions that coordinate the control, operations, and data aspects of the underlying algorithm. In the same way, the individual must gain experience with the basic instructions of the programming language of the unified field, the Yoga Sūtras, through the practice of the TM-Sidhi program.

The Transcendental Meditation technique, as we have seen, is an effortless mental technique that allows the conscious mind to identify itself with the unified field of all the laws of nature in the simplest state of awareness. The TM-Sidhi program allows the mind to be active in that state and experience the fundamental impulses of natural law within the self-interacting dynamics of the unified field (Maharishi Mahesh Yogi, 1978). This experience purifies and cultures the functioning of the brain to be able to utilize these basic impulses, the basic instructions of the programming language of nature. Through the daily practice of the TM-Sidhi program, the brain is cultured to function as a computing hardware that can compute from the level of the unified field and spontaneously make use of the perfect, preexisting algorithms of natural law.

Thus, the human brain can become a kind of “cosmic” computer hardware: it can use the cosmic software of natural law to compute the thoughts and actions of the individual in full accordance with all the laws of nature, so that the individual does not violate any laws of nature and does not create the ground for suffering or problems in life (Wal-
lace, 1986). Just like an experienced programmer, the individual then becomes capable of spontaneously bringing out any desired mode of activity from within his own unbounded Self, the unified field, through the most simple and natural process of thinking. That is why the basic principle of action according to Maharishi Vedic Science is Yogasthāḥ kuru karmāṇi, which means “established in the unified field, perform action” (Bhagavad-Gītā 2.48; see Maharishi Mahesh Yogi, 1969). This principle is enlivened in individual life by the practice of the Transcendental Meditation and TM-Sidhi programs, because they give the direct experience of the unified field and the ability to perform action using the programming language of nature, the impulses of the Veda. This full development of the human computing hardware may be considered as the fulfillment of the field of computer science. Through the new integrated approach to computer science based on the unified field, this highest achievement in the field of computer science is available to every individual.

2.10 Toward Perfection in Computing

Having considered the Veda as the abstract algorithms of natural law and the human body as the computing hardware capable of utilizing this cosmic software, it will be interesting to consider now the activity of natural law itself in guiding the activity of the universe. The functioning of natural law, originating from the Veda, is constantly being displayed in every aspect of the universe, not just human activity. Thus, there must be some mechanism for the abstract algorithms of natural law to express themselves directly through some kind of “hardware” that is independent of the human body. A more careful analysis of the structure and internal dynamics of the unified field as known from Maharishi Vedic Science and from quantum field theory will show that it has the properties of hardware as well as software, and therefore has the ability to directly manifest the expressed activity of natural law from its own abstract algorithms. This section will discuss the recent trends in the design of computer hardware and show how these trends are leading in the direction of a computing structure with properties normally associated with fields, thus reflecting more closely the computational dynamics of the unified field.
One of the major trends in the architecture of the next generation of computers is the development of highly parallel computers, which have enormous computing power resulting from the ability to perform many operations simultaneously. The continuing reduction in size and cost of the basic processing elements of the computer, through higher circuit densities on integrated circuit chips, has made it economically attractive to put large numbers of processing elements in a single computer (Gajski & Peir, 1985). There are already many commercially available systems with five to thirty processors, and a few new computer systems with several hundred or several thousand processors all capable of working simultaneously, resulting in enormous computing potential at a reasonable cost. One of the largest of the commercially available parallel computers is the Connection Machine with up to 64,000 individual processing elements all capable of communicating with each other after the execution of each instruction. The Connection Machine is capable of performing ten billion operations per second, which is ten times the speed of today’s largest supercomputer. In a recent interview, C. Gordon Bell, the assistant director of the computer funding division at the National Science Foundation, said that the Connection Machine “carried out in about one hour all of the experiments in image processing that had been done in the last four decades” (Bell, 1987). John L. Gustafson, one of the leading researchers in highly parallel computers, has put forward the “idea that somehow the Connection Machine simulates the field properties of the universe” (Frenkel, 1986, p. 755).

In a recent paper presented at a major conference, B.J. MacLennan (1987) said that in computers with massive parallelism, the computing capability can be considered as a continuous field rather than a collection of discrete computing elements. He spoke of such computers as field computers and defined the computing process as a field transformation. In his presentation, MacLennan was seeking to identify a “set of universal field transformations” that could form a foundation for formulating a “universal field computer” as a basis for understanding the computational properties of all field computers. Based on the computational viewpoint of natural law developed so far in Part II, this paper puts forward the proposition that the unified field of natural law can be considered as the ultimate “universal field computer” and that the dynamics of *Agnim* as briefly discussed in section 2.7 provides a “set of universal
field transformations” for this computer. The remainder of this section of the paper will discuss the properties of the unified field as known from quantum field theory and Vedic Science to reveal a structure that resembles highly parallel computing hardware. This discovery of the “hardware” aspects of the unified field explains how the algorithms of natural law can manifest themselves directly into the expressed activity of nature everywhere in the universe at the same time.

According to the understanding of the unified field from both quantum field theory and Vedic Science, the total potential of natural law in the unified field is present at every point of the universe: every point of the universe is lively with all possible aspects of natural law (Hagelin, 1987). Even from our common everyday observation of nature, it seems that the same laws of nature function everywhere at the same time. For example, the law of gravity continues to function in the same way wherever we move and operates simultaneously all over the earth and in space. Unified quantum field theory states that at the point-like scale of distance (10^{-33} centimeters), the total potential of the unified field is lively. Thus, the unified field must be capable of infinite parallelism, with a complete “processor” for all the laws of nature at every point of the universe.

As explained in section 2.6 on the eternal structure of the Veda, the totality of the Veda exists at every point in the universe, and its self-interaction has three aspects: Rishi (knower), Devatā (process of knowing), and Chhandas (known). Although the discussion of the Veda as software associated Chhandas mainly with data, the Chhandas of the Veda is more than that: Chhandas is the structure or flow of the Veda (Maharishi Mahesh Yogi, 1985). The Veda is completely self-sufficient and has its own structure through which the impulses of intelligence (Devatā) manifest into the expressed aspects of natural law. Thus, the Chhandas of the Veda provides a computational hardware for the direct expression of the laws of nature throughout the universe, making the unified field the ultimate field computer with the total computing power at every point.

Furthermore, the perfect harmony and integrated activity of natural law indicates that these computing points function with perfect coherence and correlation. Hagelin (1984) has located, in the mathematical field equation for the unified field, qualities of infinite correlation and
frictionless flow of information, resulting from its property of perfect self-referral. The cosmic computing of the unified field computes orderly relationships even at enormous intergalactic distances, as evidenced by the orderliness and symmetry found in the cosmos. This knowledge from unified quantum field theories is also found in the Bhagavad-Gitā, which is the summary of the practical teaching of Vedic Science (Maharishi Mahesh Yogi, 1969):

\[ \text{Pratyayavo na vidyate} \]
\[ \text{No obstacle exists. (2.40)} \]

The major technological difficulty in the design and utilization of highly parallel computers is the excessive overhead of coordination and communication between the large number of concurrent activities in different processing elements. It has been especially difficult for programmers to develop software that integrates the many concurrent computing activities into one single computational task, and research is currently underway to develop new programming languages and algorithms for parallel computing (Gelernter, 1986). All of the technological problems now facing parallel computing do not exist at that level of the cosmic computing of the unified field of natural law, the ultimate parallel computer. As discussed above, the unified field has an infinite number of “processors” (one at each point of the universe) that function with perfect coherence and correlation giving rise to the simultaneous coordinated activity of every aspect of the universe in complete harmony. For the human mind to achieve this supreme level of computing, it is obviously necessary to function from a field beyond space-time differences. That technique is available in Vedic Science through the Transcendental Meditation program, which allows human awareness to transcend the limits of space-time, and identify itself with the unified field. According to quantum physics, this occurs at the Planck scale of $10^{-33}$ centimeters and $10^{-43}$ seconds, where classical space-time dissolves into a quantum mechanical superposition of all geometries, giving rise to infinite correlation between all classical space-time points (Hawking, 1984). At this level, computation is instantaneous and information flows without friction.
2.11 Summary: Realizing the Self—The Center of All Computing

The unified field is the cosmic computer of nature, computing all activity in the universe using its own self-referral language, the eternal programming language of natural law—the Veda. Being the self-sufficient source of all activity in nature, the unified field contains not only the abstract algorithms of natural law, but also its own cosmic level of software and hardware. As we saw in Figure 1, computer science has located three fundamentals—theory, software, and hardware—as the agencies by which abstract algorithms are manifested into concrete computing activity. We saw that any computer always contains all three values, intimately connected and interacting with each other, to the point where it is sometimes difficult to tell where one ends and the other begins. Although the discussion so far has focused only on the software and hardware aspects of the unified field, the theory or underlying “knowledge” value must also be present within the structure of the Veda. As the premier computing device, the unified field also has these three aspects, but in one eternal, self-referral, self-interacting state.

In its ability to utilize the cosmic software of the unified field to compute action in accordance with the laws of nature, the human nervous system can be considered as a miniature reflection of the cosmic computing of natural law within the unified field. In fact any nervous system of any species of life may be viewed as a computing hardware, a limited expression of the cosmic computer. This presents a vision of the full range of computer science, from the most perfected value of cosmic computing in the unified field to the computing of the human nervous system, the nervous systems of various species, and the computing of electronic computers. The cosmic computer, which computes the activity of the cosmos, is available at every point of creation and is reflected in varying capacities in the computing systems represented by different species, each having its own level of theory, software, and hardware. Man’s computing system, however, is unique in that it can be cultured to function from the level of the cosmic computer. The human brain with its 100 billion neurons is so perfect and complete that, if properly cultured by the Transcendental Meditation and TM-Sidhi programs, it can gain access to those perfect algorithms of the cosmic computer, which are governing all activity of the universe in the most orderly and
perfect manner. This allows individual computing to take place spontaneously from the level of cosmic computing.

The electronic computer is designed from knowledge of the electronic level of natural law, an expression of the electromagnetic field, which is only one of the four basic forces of natural law. Thus, electronic computer technology operates from a partial value of natural law and has its limitations, both practically and theoretically. In the theory of algorithms, it has been proven that there are certain computational tasks beyond the capability of any algorithm formulated as a finite series of mechanical steps (Minsky, 1967). One example of such a noncomputable problem is deciding whether a particular type of algebraic equation called a Diophantine equation has any integral roots. There can never be any electronic computer, no matter how large or fast, that could solve this problem for all Diophantine equations. However, Vedic Science states that the nature of the unified field is infinite intelligence, and that it is, therefore, a field of all possibilities. The cosmic computing of the unified field is from the total potential of natural law and has no limitations. The steps of computation and the self-referral loops of electronic computer programs are extremely fast but still take place within the boundaries of space-time and have limited computing power. The infinite self-referral and frictionless flow of information within the unified field (see section 2.10) gives it infinite computing power. Fortunately, the human computer has access to this level of cosmic computing of natural law through the Transcendental Meditation and TM-Sidhi program. Thus, the human body itself can be called a “cosmic” computer, whose full development represents the highest value of the field of computer science and the goal of unified field-based computer science.

2.12 Applications to Education

The main area of application of unified field-based computer science so far has been in computer science education. The integrated view of computer science, provided by the chart of Figure 1, has great value in helping students learn the specific knowledge presented in each course and to put it in the proper context of the whole of computer science. Even at the beginning of their first computer science classes, the students feel more at home with the vast complex range of knowledge in
computer science. As they study each separate area of computer science, they always see how it relates to other areas and how all the areas fit together into a whole. Both beginning and advanced students benefit from the chart; it covers the full range of computer science, from the fundamentals of data types and control structures studied in the first computer programming course to the most advanced research areas such as theory of algorithms and theory of programming languages. The chart helps students appreciate how all the areas of computer science emerge sequentially from one source, the unified field of natural law.

Through the daily practice of the Transcendental Meditation program the student gains direct experience of the dynamics of cosmic computing in the unified field. Then the concepts of computer technology studied in the classroom can be presented with reference to these principles of cosmic computing, thus increasing motivation and making learning easier. Maharishi University of Management is the first university in the world to adopt a unified field-based system of computer science education. At Maharishi University of Management, each lecture in computer science is supplemented with a main point chart giving three to five brief main points that summarize the major concepts presented in the lecture. For each of these computer science concepts, a corresponding principle is given from the cosmic computing of the unified field as known from Vedic Science. Thus, the students appreciate the concepts of electronic computer technology as specific examples of universal natural laws of computation found in the self-referral dynamics of the unified field. Unified field-based computer science education is helping the field of computer science move away from simply being a set of techniques for electronic computing and move in the direction of a universal science of computation that studies computational principles in natural law.

Many examples have been given in this paper of how the knowledge of the unified field from Vedic Science can help illumine specific principles of computer science. In section 2.5 it was shown how the three-in-one structure of Rishi, Devatā, and Chhandas in the Samhitā of the Veda could be related to the interaction of control, operations, and data in programming languages. Section 2.6 described parallels between the structure of the Rk Veda and the organization of computer software packages. The origin of loops in computer programs was located in sec-
tion 2.7 as the second word of the Veda, Īle, representing the process of the pure potentiality of consciousness in Agnim curving back onto itself. These specific parallels between electronic computing and cosmic computing have proved useful at Maharishi University of Management in courses in computer programming and software development by providing students with a deeper, more intimate understanding of the principles of computer science. Other sections of the paper, especially sections 2.8 and 2.9, described the mechanics by which the human brain and human mind can directly utilize the computing power of the unified field. This knowledge is also an important part of unified field-based computer science education because it allows students to realize that the electronic computer is just a limited reflection of the infinite computing potential of their own brain, thus making them a “master” of the computer rather than a “slave” of the computer.

There is a growing emphasis in computer science education on the appreciation by students of the wider impact of their activity on society as a whole. The culturing of each student’s own brain that results from the Transcendental Meditation and TM-Sidhi programs raises the individual computing of his nervous system more and more in the direction of cosmic computing. The individual human brain hardware of the student is then able to make use of the cosmic software of natural law to compute action in accordance with natural law. Graduates of such an educational system will tend not to violate the laws of nature, and thus not create problems for themselves or society. Spontaneously, these computer scientists will be able to guide the future growth and application of computer technology to be in accordance with all the laws of nature, and thus produce a beneficial influence for the whole human race. Graduates of a unified field-based educational system will live life free from problems and be ideal citizens fulfilling the highest ideal of education.

2.13 Future Directions
This paper has presented an expanded vision of computer science based on the unified field of natural law. In this new integrated approach, we have shown all the major areas of computer science to emerge from a common source, the unified field of natural law, which can be directly experienced by any individual in the simplest state of aware-
ness through the Transcendental Meditation program. As the origin of all laws of nature, the unified field can be viewed as the most powerful “computer,” responsible for computing the functioning of natural law everywhere in the universe. A deeper consideration of the self-referral structure and dynamics of the unified field, as known from Maharishi Vedic Science, brings out many of the computational aspects of the unified field in relation to the computing concepts used in current electronic computer technology. This analysis of the perfected value of the cosmic computing of the unified field provides a framework for understanding current computer technology more thoroughly and a model for future growth of technology.

In addition to the educational value of unified field-based computer science, it also has the potential to contribute to the future growth of computer science through research. The whole understanding of the computational properties of the unified field as found in the Veda is just beginning, and has not yet reached the point where direct applications in computer technology have been achieved. However, already there have been some attempts to use the model of cosmic computing as a guide for future directions in programming languages for highly parallel computers (Lester & Guthrie, 1987). As this new approach becomes more widely known among researchers, it can accelerate the progress in computer technology toward utilization of finer levels of natural law in the direction of the universal level of computation in the unified field. However, as was mentioned earlier in the paper, knowledge of the Veda is not gained through intellectual analysis of the Vedic sounds as recorded in books, but through direct experience of pure consciousness from which the Veda is structured. One verse of Ṛk Veda clearly expresses this fact:

\[ Yastanna veda kim pichā karishyati \\
\text{ya ittadvīdu tas ime samāsate} \]

*He whose awareness is not open to this field of pure consciousness, what can the verses of the Veda accomplish for him? Those who know this level of reality are established in evenness, wholeness of life.* (1.164.39)

Thus, computer scientists will be able to derive maximum from Maharishi Vedic Science through the Transcendental Meditation and TM-Sidhi programs, which provide the direct experience of the uni-
versal dynamics of computation in nature. The result will be not only greater creativity in developing more efficient electronic computers, but more importantly, as stated in the above verse from Rk Veda, the establishment of wholeness of life. This brings the state of fulfillment to the individual and allows that fulfillment to be lived in the boundaries of daily life. Computer technology, and in fact all technology, uses the organizing power resulting from knowledge of natural law to provide greater efficiency and productivity in human activity, making life more secure and comfortable. Now through the Transcendental Meditation and TM-Sidhi programs, every individual has the opportunity to refine the functioning of his own brain hardware to directly utilize the cosmic software of the unified field and thereby fulfill the purpose of computer technology: living life in happiness and abundance. The future of computer technology lies in living the infinite creative potential of the unified field.

Note

1. Translations of Sanskrit passages into English are based on Maharishi Mahesh Yogi’s translation.

References


Maharishi Mahesh Yogi (1978, May 22). *Principles of invincibility from Veda and from physics: Discovery of creation in the structure of pure knowledge—study of the Mandalas of Rig Veda* [Videotaped lecture, Seelisberg, Switzerland].


This article, “Unified field-based computer science: Toward a universal science of computation,” by Bruce Lester, Ph.D., here revised/updated, and reprinted with permission, was originally published in *Modern Science and Vedic Science, 1*(3), 1987, 267–321.
Vedic Computation: Redefining Computer Science
in the Light of Maharishi Vedic Science

Gregory Guthrie, Ph.D.
ABOUT THE AUTHOR

Gregory Guthrie, Ph.D., received his B.S. and M.S. in Engineering Science from Purdue University, and his Ph.D. degree also from Purdue in 1978. He was a computer systems architect on the technical staff at Bell Telephone Laboratories, after which he joined the faculty at Maharishi University of Management (previously Maharishi International University, 1971–1995) in 1983. Dr. Guthrie is the Chief Technical officer and President of AeTec Consulting, and in this role has consulted and taught courses at many major companies around the U.S. Dr. Guthrie is a former Dean of the Graduate School and Chairman of the Computer Science Department at Maharishi University of Management, where he is currently the Dean of the School of Computer Science and Mathematics. He has presented at technical conferences around the U.S., and has lectured on Maharishi Technology of ConsciousnessSM programs around the world since 1973. His current research and teaching interests are in the area of programming languages and systems.
ABSTRACT

Computer science has a unique and important role in today’s world. With its powerful synthesis of new tools and mechanisms through which to express our growing knowledge of natural law, it has extended and redefined many areas of life to create new disciplines and new possibilities for the application and development of existing fields.

This rapid development is based on progress in two fundamental areas: the discipline of computer science and the increasing knowledge of natural law in the application areas. Together these have combined to give us new methods and new depths and extents of inquiry and achievement. With this phenomenal growth in the capabilities and applications of computing, technology has expanded into areas critical to the safety and stability of our society. This raises important questions about the soundness of these foundations, and therefore the future development and applications of computer science and computing technologies. Issues of the correctness, limits, and applications of computation and computing systems are now vital to our future, yet cannot be resolved within the current scope of the discipline. This is because these fundamental issues require a clear understanding and precise description of the structures of intelligence expressed in computing systems, and this is not present in computer science, or any other modern scientific discipline.

Maharishi Vedic Science, a science of the full range of intelligence including its representations in objective forms and its realizations in human awareness, provides a unique context for understanding and resolving these problems. Because it provides both a theoretical model and practical techniques for investigating the abstract structure of intelligence and principles of natural law common to all application areas, it encompasses and integrates the areas of application knowledge and computational representations. Reconsidering the fundamental structure of the discipline of computer science and its applications in computing systems in this context results in new and more comprehensive understanding of the status and future of the entire field of computing.

Introduction

Computer science is perhaps the quintessential modern science; it merges the most advanced applied technologies of electronics and semiconductors with the abstract areas of theoretical mathematics, to form a new capability which has become the hallmark
of modern technology. These two divergent foundations are combined through the principal innovation of computing, the field of software, which has become the ubiquitous tool of modern society. In its applications, computer science provides the tools, which are at the forefront of every field of modern endeavor, supporting, extending, and often redefining the nature of our lives.

Computing has risen to be the universal technology of our modern age. It permeates all aspects of modern life, from applied areas of entertainment, economics, commerce, and business, to the more foundational areas of science, research, and education. Computing has extended the methods and capabilities of every area in such a manner as to redefine the possibilities, and often the approaches, of that area. New disciplines have been created and old disciplines redefined.

The result is an extension of our senses, actions, and intellect into realms which were hitherto inaccessible, both conceptually and physically. Computing systems and technologies have literally transformed life on our planet and become a key ingredient for future progress and technology.

This rapid development is based on developments in two fundamental areas: the discipline of computer science and the increasing knowledge of natural law in the application areas. Together these have combined to give us new tools, new methods, and new depths and extents of inquiry and achievement. Each succeeding new level of expansion yields new knowledge, which then recursively enables even greater capabilities. It is this self-enabling loop of development that has created the tremendous growth in all areas, including that of computing itself.

The pace of the development and application of computing technology has progressed at a pace unknown to other fields. Moore’s law (Moore, 1964), which successfully predicted the doubling of the capacity of integrated circuits every 18 months from 1964 (an increase of over 200,000), is one striking example of the ongoing and long-term nature of this progress. This exponential growth of computing technology has allowed us to build computing systems that are the most complex constructions ever created by man. Today’s programs may consist of many millions of lines of code and run on very sophisticated desktop processors. These small systems with millions of transistors in their microprocessor controllers execute instructions at several million
instructions per second, exceeding the capacity of multimillion-dollar supercomputers from only a few years ago. Individual computers are connected to the global Internet, with millions of communicating systems and even more millions of individual users. And yet, this emerging computational technology, connectivity, and capability is still in its early years, and still growing rapidly. While there is speculation that we will soon reach the physical limits of growth on the hardware level, architectural advances within and between systems promise to sustain the growth for the foreseeable future.

With this explosive growth of technology, it is vital that we locate a sound and complete theoretical foundation to guide its development and application. This is the liability of current computing technology. The entire field of computer science is still in its infancy, and is attempting to define its theoretical foundations, and create from them appropriate and effective methodologies for its application areas in the computing technologies.

This paper begins with an overview of the nature and components of the discipline of computer science, which presents the fundamental concepts and issues of the Science of Creative Intelligence and Maharishi Vedic Science. The impact of this expanded scope on the methods and limitations of traditional computer science is examined, and a vision of the future of computing presented.

The Science of Computation

Computer science is a relatively new field, and in its formation has combined aspects of existing disciplines and given rise to new areas of study. The history of computer science is an interesting interplay and, finally, merger of the disparate fields of theoretical mathematics and applied electronics. Theoretical mathematicians were interested in the descriptive power of mathematics, and investigating the limits of such mathematical models. Electronic engineers were simultaneously seeking to discover smaller, faster, and cheaper electronic devices. These explorations converged when it was realized that the discrete states of mathematical logic could be easily represented as the binary modes of an electronic switch. From this basic connection, and the resulting empowerment of both fields, the synergy and success of this combination has created new areas of specialization within the existing disci-
plines, and completely new areas of study. There is some latitude in the terminology of the related sub-disciplines of computer science, so an overview is given here.

Computer science studies the structures, dynamics, and methods of representing information, and the limits and extents of these structures to model aspects of natural law. It attempts to discover, describe, and create methods which apply this knowledge to create computational models of the real world. These models can be purely symbolic, similar to the symbolic descriptions of nature created in physics or mathematics, or they can be embodied in physical structures, to create computing machines. Computing systems combine through both their software and hardware components, and thus also include the interfaces between abstract and concrete computing structures.

Computer science studies the theory, tools, and methods at the basis of all computing systems. From this, several related fields arise, creating technologies that apply this underlying science toward specific practical goals. Software engineering investigates pragmatics of constructing large computing models and representative software systems. These large systems are the most complex objects constructed by mankind, and they are created in a new abstract media which provides great freedom of expression and also introduces new challenges. Computer engineering considers the creation of physical realizations of these systems, machines which physically reflect and embody the desired computational models. For an effective and practical result, the physical structure of these machines must correlate to the abstract structure of the computation to be run on them, as specified in the software model. The newly delineated discipline of computational science looks at combining these technologies and using them to model the most advanced aspects of various application areas.

The distinction between the science and technologies of computing is important, as the nature of the methods, goals, and limits of each is quite different. Figure 1 shows the relationship between computer science and its related and component disciplines. Strengths or weaknesses in the underlying science are naturally reflected in deficiencies in its related technologies, improper or incomplete understanding resulting in errors in application. To strengthen the applications of computing, one must look to its basis in computer science.
Computer Science

The goal of computer science is to understand, express, and apply knowledge of natural law in some situation to create computing systems which represent some essential nature of the qualities of the original system. While it could be said that an appreciation and expression of natural law is the goal of all sciences, and indeed arts, there are unique and important aspects of computer science, which distinguish it from all others. First, computer science attempts to provide a universal representational model for information. It does not limit its scope to representation of any specific or particular form of information, or description of structure, but strives to provide an ability to describe understandings of all areas. In this sense computer science is a meta-science; it hopes to discover the common structure(s) of knowledge at the foundation of all expressions of nature. Further, having such a description, it aspires to create machines which can automatically express the implications of this knowledge, to utilize the information in the symbolic structural descriptions to guide the operation of physical computing machines. The symbolic descriptions of system structure are the software, and the machines the hardware, which together form a computing system.
Science, Knowledge, and Veda

With computer science as the basis of so many great accomplishments and so much of modern technology, it is important to clearly define and understand the scope and nature of the discipline. If computer science is a science, a science of what? What is the fundamental element studied by the discipline?

The fundamental premises of all science are the following:
- There is an underlying order in nature
- This orderly structure can be known
- This knowledge can be expressed in concrete (symbolic) forms

Against these scientific premises computer science argues the following:
- Symbolic forms can be manipulated in systematic ways to represent other structures of knowledge.
- This symbolic manipulation can itself be represented as a symbolic structure of transformational states.
- This transformational process can be automated.

We can say that computer science is the science of information structures. It seeks methods to discover structures and patterns in systems, to create models which represent these structures, and to express these models in symbolic forms, which can then be automatically interpreted.

For example, scientists may study the phenomena of waves and flow in a fluid system and determine systems of mathematical equations, which describe their structure and behavior. These descriptions are a model of the actual phenomena, which can be encoded into a computer program. This program can then be used to investigate and explore many new aspects of potential system behavior: how the fluid will behave at other temperatures, pressures, or in other situations. The knowledge of the system represented in the model, and then in the program, contains adequate information to represent the essential properties of interest.

In this approach, computer science is a meta-science; it attempts to provide a universal modeling and representational system that spans all disciplines, including the sciences, arts, and humanities. With its unique approach of creating informational structures to represent our knowledge of a system, it attempts to be a general science of informa-
tion. Because this information represents an understanding within our intelligence, it can be seen as an attempt to universally quantify knowledge, to create a science of intelligence.¹

Computer science is in this sense an applied science. It does not study and analyze existing elements like physics, chemistry, or mathematics. One does not encounter naturally existing software programs or machines and study them; all software and computers are created systems. Rather, it forms new methods and models to describe existing natural systems. The success of computer science is a measure of how much we have been able to learn about natural law in all other disciplines, and to express in our computing systems.

As with any discipline, computer science is both synthetic and analytic. It is our intellect that connects these two areas; it defines what we know and what we are able to express. Yet, although the intellect is at this fundamental junction point of computer science, it is not currently a part of the discipline.

Scientific Models

Because of its roots as a modern science, computer science shares a common ideology with the other sciences. To achieve the desired repeatability, regularity, and reliability of a science, modern sciences have chosen to take a purely objective approach, to explicitly exclude subjective aspects which may vary from researcher to researcher and from time to time. Lacking any reliable standards, or absolute basis in the field of subjectivity, this was a logical and reasonable approach. Over the duration of the past few centuries this objective approach has yielded great results, thoroughly investigating and charting many structures, laws, and technologies of creation.

The approach, and thus the scope, of computer science is typical of any modern science, based on this objective approach. Unfortunately, this approach is intrinsically limited in that it excludes the most fundamental source of all science, the subjectivity of the scientist. It is the awareness of the scientist (analyst, programmer) that appreciates the structures of natural law at the source of all scientific discoveries.

¹ Note that this is not related to the field of cognitive science, which studies the mechanisms of human processes. Computer science focuses directly on relations between systems and their formal symbolic models.
and expressions. In computer science, this subjectivity is the basis of all computational expressions, in theory, software, or hardware. Further, the final result of any computation is in some interpretation by the user; all meaning is in the subjective valuation of the objective results of a computation. Thus, the subjective intellect of the (computer) scientist is the source and goal of all computation.

A different and more comprehensive approach to the issue of subjectivity and science is taken by Maharishi Science of Creative Intelligence (SCI), a new discipline which integrates the full nature and development of creative intelligence in nature and human life. SCI locates the source of all aspects of natural law as a unified field, an abstract field of natural law, which underlies all activities in creation and identifies this field as a field of pure intelligence. Various modes of this field of abstract pure consciousness are expressed as the various qualities and phenomena of creation. Modern physics also identifies a unified field as the most abstract basis of all of creation, and similarly describes all expressed qualities and structures of the universe as modes of this most fundamental field (unified field). Recently, physicists have proposed that this field is indeed a field of consciousness (Hagelin, 1987). In this model of the unified field as a field of pure consciousness, SCI unifies traditional objective science and a new subjective science.

SCI includes both the subjective and objective fields of knowledge, but starts with the field of intelligence as primary, forms a detailed description of the nature, dynamics, and properties of this field, and then describes its expression into various objective forms and phenomena. SCI studies the field of intelligence itself first, independent of any particular expression of it. This general approach, which identifies universal principles and qualities of intelligence, has major implications for the range of knowledge available compared to a purely objective approach. Considering any traditional discipline in this expanded scope drastically alters the context and definitions of the discipline, often providing a more natural context for resolving fundamental questions which may have existed with prior approaches.

A study of computer science in the context of SCI has similar impacts and important results. First, it provides a more reasonable and realistic
scope for studying computation. It does not exclude the source of all semantics which are the primary source and ultimate goal of all computational expressions. Second, because it provides universal models of structure, form, and process which describe the functioning of natural law in all systems, it directly realizes the goal of computer science to locate universal structures for modeling all areas. Third, it has an important and practical value in that it provides specific technologies for improving the ability of individuals to directly perceive and understand structures of natural law.

Figure 2. The objective approach of modern science considers only relationships between objective aspects of study and thus limits its scope by excluding the subjectivity of the scientist. Maharishi Science of Creative Intelligence expands this scope to include the full range of subjectivity, including its source in the unified field, and Maharishi Vedic Science provides greater detail of the structure and mechanics of the self-referral dynamics of consciousness.

Figure 2 shows the differences between these approaches. Computer science examines various disciplines looking for common unifying principles and descriptive qualities. It analyzes common aspects, unifying structural elements and patterns which are common to all areas. The fundamental agent of this analysis is itself purely subjective, the intellect of the analyst. Yet, computer science provides no theory, models, or techniques to address this subjective basis of all computing systems. While the education and training one receives do help structure specific skills and understandings, there is no mechanism provided to enhance one’s ability to recognize the patterns in systems that these techniques are the vehicles to describe. Computer science provides the
tools and methods to express understandings, but cannot augment the ability to realize the knowledge to be expressed. This objective outside-in approach is intrinsically limited; there are many fundamental areas of investigation completely beyond its reach.

Maharishi Science of Creative Intelligence provides a completely different approach to the search for common structures of natural systems. SCI describes the unified field as a field of intelligence and as the primary element responsible for not only the appreciation and analysis of structure in all systems, but also as the source of this structure. SCI also describes how this underlying field of pure intelligence expresses itself into the various structures and dynamics in creation. Therefore, SCI provides both a description of the common source of all structures and a unified theory of how this purely abstract dynamics of pure consciousness expresses into the universe of concrete observed phenomena. Further, SCI provides methods for the direct experience of this abstract field of pure intelligence, through the Maharishi Transcendental MeditationSM technique. This technique allows the mind to systematically experience more refined levels of thinking, until it is able to directly experience the most settled state of awareness, the field of pure consciousness, the unified field.3 Through this practice the unified field is realized as the simplest state of one’s own awareness, a self-referral field of pure consciousness (Maharishi Mahesh Yogi, 1995b, p. 280).

Thus, SCI could be termed an inside-out approach to knowledge. It starts with general unifying principles, experienced as aspects of the unified field of one’s own pure consciousness, and then connects them to their expressions in various disciplinary areas. Maharishi describes SCI as the fulfillment of the search for an interdisciplinary science based on this foundational and integrating role (Boothby, 1996). He explains how this approach fulfills the requirement for complete knowledge:

Complete knowledge should mean total knowledge of the object of inquiry and total knowledge of the subject: total knowledge of both the known and the knower. . . . The Science of Creative Intelligence, by opening one’s awareness to the infinite, unbounded value of intel-

---

3 See the series introduction, at the beginning of this volume, for a more complete description of the Maharishi Transcendental Meditation program.
Maharishi Vedic Science

Maharishi SCI locates the field of pure intelligence as the common source of all creation, and describes its qualities and expressions into the manifold properties of creation. Probing deeper into the nature of this field, Maharishi Vedic Science investigates the nature and mechanics of this unmanifest field of pure intelligence. It provides detailed descriptions of the internal structure and dynamics of the purely abstract field of pure consciousness, and locates the most fundamental dynamics within the unified field responsible for creation. Maharishi explains:

There was a time when modern physics declared that the entire creation emerged from four fundamental forces, which were thought to be fundamental. As research advanced, these so-called fundamental forces were understood in terms of one holistic Unified Field of all the Laws of Nature.

According to Maharishi Vedic Science, the entire classical world is the expression of the Veda—natural law—eternally lively within itself in terms of the self-interacting dynamics of consciousness. . . . (Maharishi Mahesh Yogi, 1995a, p. 274)

Maharishi Vedic Science finds that within this field there is a specific structure at the basis of all other structures; one most simple, primordial dynamic of natural law which is the source of all other expressions. This most fundamental dynamic is the eternal self-referral awareness of this field of pure consciousness. Being a field of consciousness, by its very nature it continually knows itself, and this process initiates the fundamental mechanics of creation. By experiencing this most fundamental quality of consciousness, one gains direct knowledge of the most universal level of natural law. Maharishi describes how this level of the unified field, termed the Veda, defines all the laws of nature, every aspect of structure and dynamics in creation. By knowing Veda one knows the source of every aspect of natural law. Maharishi describes this Veda as pure knowledge:

Veda is the structure and function of pure knowledge. It encompasses the whole range of science and technology; it is the theory and practice
Maharishi Vedic Science, by locating this self-referral structure of pure consciousness as the source of all expressed structure in creation, answers the quest for universal structure. In doing so, it introduces a new aspect to computer science, in that this source is located not within any aspect of any application area, but within the unified field. This means that not only is the source of all structure not located within the area of study, but it is found at a level of abstract pre-expression; all structure is an expression of a purely abstract dynamics of this unmanifest field of consciousness. Further, because this field is one of consciousness, traditional approaches of science lack any methods for investigation or application of properties of this abstract field. Maharishi Vedic Science provides this missing knowledge; it is a complete science of this level of pure abstraction. While other sciences may have great overlap in their domains, and one may deduce properties of one field from the structure of another, that is not the case for this level of inquiry; only through the direct experience of pure consciousness can the nature of the unified field be known. Intellectual investigation and inquiry can only infer general properties and descriptions of the field. Only through the knowledge and technologies provided by Maharishi Vedic Science can this level of universal structure be known, and its benefits applied (Maharishi Mahesh Yogi, 1996a, p. 34).

A New Paradigm
The different approaches of traditional computer science and Maharishi Vedic Science predictably locate different results in their quest for universality. The principles of SCI and Maharishi Vedic Science subsume those of traditional computer science, in a manner reminiscent of the relationship of traditional Newtonian physics and quantum mechanics. Newtonian physics describes the physical behaviors and interactions among macroscopic objects and provided an adequate model for the observations of early science. However, as science progressed, it was unable to describe some observed phenomena, which led to the postulation, the formulation, and finally the verification of an entirely new field. Quantum mechanics described aspects of natural law far beyond our direct experience, presenting a model in terms of abstract fields,
with many very unexpected and startling results. This new paradigm revolutionized the field of physics, and because it successfully subsumed and extended traditional physics, it is now the accepted theory of modern physics. Similarly, the partial views of traditional objective sciences have their value within their limited domain, but with the advent of SCI and Maharishi Vedic Science, are now seen to exclude fundamental areas required for complete knowledge of the discipline (Figure 3).

![Diagram](image)

**Figure 3.** The scope of modern science continually expands as newer paradigms and levels of knowledge are discovered. Maharishi Science of Creative Intelligence and Maharishi Vedic Science provide a complete framework for all knowledge, by providing knowledge of the unified field, the Veda, the home of all the laws of nature.

Computer science combines the knowledge of natural law in other disciplines with its own methods to model, describe, and, thereby, automate this knowledge in computing systems. Traditional computer science with its objective approach would always be limited in the degree to which it can investigate, understand, and thus represent the processes and dynamics of natural law in a system. With the availability of SCI and Maharishi Vedic Science, a new scope of knowledge of natural law is available, to comprehend, model, and express the full range of natural law, and thus form a truly universal basis for computing systems. This new paradigm for computer science has impacts in all aspects of the discipline—theory, methods, and applications—which are explored in subsequent sections.
Computational Structures

Computer science defines itself in terms of a number of general concepts, including data, information, knowledge, structure, relationships, and a number of more technical concepts such as algorithms, abstractions, languages, and machines. With the overlap of the intended technical definitions and common understandings of these terms, questions arise about the relationship between these properties of computing systems, and related concepts like intelligence and consciousness. Indeed, it is often presumed that because of the vast computational power inherent in modern computers, they must possess some attribute of intelligence, and thereby somehow a quality of consciousness. To properly address these broad questions, first we need to establish clear definitions of these terms in their technical context as a model of computation.

The relationship of structure, information, and knowledge is fundamental to the definition of computing. We will first define these in the context of the basic aspects of computer science, which are as follows:

1. Modeling, the creation of abstractions, which represent the structure of the desired system.
2. Computation, in which one representational form is transformed into another with equivalent semantics.
3. Interpretation, whereby the results of the computation are understood in terms of some new structural aspect of the original system.

Modeling is the process where the structure of the original system is studied, understood, and represented in some symbolic form, which describes the essential relevant features of the system. This description is then evaluated, or executed, on a computing machine to determine its meaning. In some cases, this evaluation will be the interpretation of the software as a series of steps of calculation, or of logical inference, or in others as a series of numerical or symbolic relationships. In any case the result is a transformed representation, which represents some alternate state or configuration of the original system. This result is itself symbolic, and is then interpreted by the programmer to determine its meaning (Figure 4).
Figure 4. Computation is the manipulation of abstract models of real world systems. Symbolic representations link subjective understandings to concrete computing systems. The symbolic results of a computation are then interpreted to provide some meaning to the user.

Note that computation is the manipulation of abstractions of a system, the symbolic software representations. This separation of the system and its description distinguishes computers from other mechanical systems and machines, and is the basis of general-purpose computing systems.

These representations are described as information structures. The correspondence of the forms of the programs and the structures of the original system is the basis of their representational value. The correspondence of these forms can be described in a variety of representational methods and languages, but there must be a well-defined mapping between the two structures.

In this context, we define structure as an abstract set of relationships between entities. It could be the logical structure of relationships in a set of numbers, or the numerical relationships expressed by standard arithmetic, or the more concrete physical structure of a physical system. In an arithmetic domain, there are many relationships between the numbers, and the operations on them, which give the rich structural properties seen in the methods of arithmetic manipulation of equations. For example, $2 + 2$ is the same as $1 + 3$, or $4$; they all represent the
number four. They are three different concrete symbolic representations of a single abstract concept. They are clearly distinct forms, but they all denote the same meaning. The form of the description is called the syntax, and its meaning the semantics of the expression. In arithmetic this same distinction is given between the numerals and the numbers which they represent. The fundamental requirement of such a representation is an isomorphism between the structure of the domains, between the representative symbolic notation and the actual system.

Figure 5. The structure of a system is represented in the informational structures of a software description of the system. This mapping between the two must preserve the knowledge of the system. This complete, reflexive, symmetric mapping is called an isomorphism.

This means that there is a one-to-one correspondence between the elements of the two domains; everything in the source domain can be represented, and every representation has a meaning. Further, an isomorphism guarantees the integrity of the mapping, that if one interprets a representation, it will have the same meaning as the original source object (Figure 5).

**Abstractions**

---

4 More precisely, they represent numeric structures which all evaluate to the same meaning.

5 The numeral 2 and the word “two” and the symbol “II” all represent the abstract mathematical concept of two. However, since we can generally express ideas in words and symbols, the distinction between the syntactic representation and its meaning is sometimes obscured. In some sense this is a benefit, as the notation becomes transparent, and the reader directly invokes the concept.
Software is an abstract description of some system, which describes the important properties in a precise and formal manner. An important principle embodied by software is that as one forms an abstraction of a system, the structure of the abstraction captures some general properties of the system. As one forms more and more general abstractions, one should be capturing more and more general properties of the system, and indeed of all systems. This implicit assumption is based on the premise that all systems are instances of a general set of natural laws—a basic tenet of all science.

For example, a standard differential equation represents the structure of an abstract set of qualities, which correspond to the physical behavior and relationships of actual systems. Consider a simple spring-mass system; its defining equation is the same as that of a capacitive circuit. Two vastly different systems, one of physical motion of an object, the other the changes in field potential in a circuit, with exactly the same mathematical description. It is this ability of an abstract structure (the equation) to represent a variety of actual systems that is the basis of the broad application range of computational systems.

If this equation is encoded into a simple program, that program can model the behavior of a wide range of systems. One description abstractly represents many systems.

Is there a limit to this model of software abstraction? Is there a complete mapping from any abstract structure, to a symbolic description in language, and/or to an actual physical or conceptual system? Are there any abstractions that are inexpressible or unrealizable? Or conversely, are there concrete systems for which no abstraction exists, and thus no symbolic description? To be able to answer these fundamental questions, we first define our basic model of how such abstractions are formed and expressed into software descriptions.

**Software and Hardware**

The existence of structure in a system means that there is a stable, orderly pattern to the underlying system dynamics. It is this pattern that is of interest to the designer of a computing system. To describe this structure, the system designer will write a software program. This term contrasts the soft, or abstract, symbolic, flexible nature of this
description to the concrete, physical, fixed structure of the hardware machine on which it will be run.

This separation of computing into distinct aspects of software and hardware is the principle innovation that marked the creation of general-purpose computing systems. Previously computation had been done on mechanical analog devices, in which the physical structure of the device modeled some relationship of interest. In the simplest case, the representation of numbers by distances was used to create mechanical adders. Linear concatenation of physical measures was used to find a new measured distance, which represented the numeric summation of the composite values. Multiplication was done in a similar manner, by using logarithmic rulings. Since adding logarithms represents multiplication of the numbers, by ruling the scales properly the multiplicand could be directly read from the device. This encoding of numbers as distances was the basis of slide rules, a principal computational device of engineering prior to electronic computing. Other devices used similar principles, with for example geared wheels measuring angles of rotation, with different ratios of movement representing different numerical relationships. In the early 1940s these mechanical calculating devices reached the height of their mechanical complexity and capability, able to do not only simple addition subtraction, and multiplication, but also complex integration and differentiation problems (Kidwell, 1994).

The concept of a separation of the program to specify and the machine to perform action was first inspired by mechanical looms. It was discovered that since many different patterns could be woven on the same mechanical loom, with a way to encode and automate the control of the loom mechanisms, one could create a very flexible general-purpose weaving system. The extrapolation to the ability to weave patterns of logic was quickly made. Further refinements in the implementation and technologies of the machines followed in the realms of modern electronics and computer engineering.

---

6 The Jacquard Loom, created in France in the 1830s, used a series of punched cards to direct the weaving shuttles.

7 Lady Lovelace Ada, generally recognized as the first programmer, said in an 1843 letter to Charles Babbage about his calculating machine, attributed as the first mechanical computer, “We may say most aptly that the Analytical Engine weaves algebraical patterns just as the Jacquard-loom weaves flowers and leaves.” (Toole, 1996)
This separation of the software description of a computation, and the hardware machine to perform the steps of the process has allowed the parallel coevolution of the two disciplines. Software and programming technology could focus on the problems of representations and abstractions and the methods for construction of large software systems. Hardware engineers could focus on the creation of faster, smaller, larger, cheaper systems to execute these programs. The field of Computer Engineering sits at the junction; finding good matches for the abstract structures of the programs and the actual physical structures of the machines.

While this separation has been a major foundation of the entire discipline, it also introduces many issues and challenges to assure proper integration across their interface. Some of the most visible have been the ongoing and prominent debates over two differing approaches to computer system architectures, termed RISC/CISC. One proposes a very simple reduced instruction set, whereby the complexity of the hardware is minimal, and thus it can be very simple, regular, and fast. The price is that software complexity increases correspondingly. The alternate proposal is to keep software simple, by increasing the complexity of the hardware. The outcome has been a combination of the approaches, but the main result has been to motivate a greater awareness of the flexibility of the hardware/software boundary, and the importance of proper integration between them.

In areas of high performance computing, there is always a concern for how well the physical structure and processes of the computing machine match the required structure of the computation as specified by the program. Further, much research is done on alternate mappings of the program into equivalent computations which would better match the hardware structure and thus perform better.

It is a very interesting and revealing point, that one can flexibly move computation across this hardware/software boundary. Specialized hardware devices are built at this boundary, in which a specific computation is directly mapped into a very regular structure as a system of logical equations, and then this system of equations is directly mapped into an array of logical hardware devices. In these devices, the structure of the computation has been directly implemented as a physical structure of the hardware. By using standard techniques that convert logical
expressions into very simple regular forms, and by implementing these as large arrays of logical hardware elements, in theory any computation can be converted into a direct hardware representation. However, in practice this is only useful for very simple and small evaluations.

The whole area of hardware or software interface has become a major focus of computer science, represented in the areas of computer system architecture and computer engineering. It is an important and interesting area, as it looks at the relationships of abstract and concrete structures, the models, programs, and machines used in today’s computing systems. Later we will see that Maharishi Vedic Science significantly expands this integration by considering more refined methods of implementation, which dissolve this hardware/software interface issue, and introduce new models of computation.

Models and Programming

Software is a concrete representation of a model. A model is an abstraction of an actual system; it is a symbolic encoding of the structure of the system. By necessity it describes only some subset of the overall system properties, and this simplification makes the representing program simpler than the actual system. Consider the modeling of a simple falling ball. The situation is very complex; but to simplify it the designer ignores aspects not important for the desired outcome (color, texture, composition, elasticity, temperature), and eliminates others which will not significantly affect the result by making simplifying assumptions (round shape, smooth surface, uniform mass, no turbulence, no rotation). From this simplified problem one can calculate a result, e.g. rate of fall, through a very simple process, which is not exact, but which adequately represents the essential aspects of the problem for the desired result.

This discrimination process depends on the analyst to intellectually determine the relevant properties, the appropriate representations, and then form an effective description. There are a range of languages designed to support this description process. These design and programming languages are universal or abstract enough to support a broad range of applications and to allow the description of structures of all types from all areas. These language systems are specifically designed to support the construction of complex hierarchical descriptions.
languages, methods, tools, and systems support the description of systems into programs, the source of this is the intellect of the programmer/analyst. It is the intellectual understanding of the original system, the goals of the program, and the methods and forms of programming languages which together form the knowledge which is the source of a program. It is this intellectual structure which is expressed in the structures and forms of the program.

Therefore, the most fundamental source of all computation is the scientist’s understanding, the appreciation and intellectual comprehension of some structure of natural law. This may be the relationships of various measurements of a physical system, or the relationships of various changes in a system with time, or descriptions of interactions of various systems or sub-systems. In every case, it is this subjective appreciation of structure, which is the basis of any further expressions. It is this mental model which is subsequently expressed into increasingly concrete and specific forms as specification, analysis, and design models, and finally a software program.

**Computational Universality**

A fundamental tenet of science is the existence of, and our ability to appreciate, order or structure in all systems. While this is a necessary axiom for modern science, an assumption that is unprovable within its scope, Maharishi Vedic Science describes and provides experiential verification of a specific, precise, well-defined structure at the basis of creation. This most fundamental level of mechanics of natural law can be directly perceived and appreciated by the most refined level of the intellect. The Maharishi Transcendental Meditation and TM-Sidhi programs are applied aspects of Maharishi Vedic Science which culture this ability through a process of systematic refinement of awareness. This gives a system analyst or programmer a method to develop the ability to gain direct familiarity with these abstract structures of natural law, and thereby the ability to more easily express them in their programs and designs.

It is the programmer’s intellect that links the structures of natural law in the application to the structures expressed in the program. It is the programmer’s appreciation and understanding of the structures of natural law, his knowledge of the system, that is expressed in the pro-
gram. When the programmer studies the system, his awareness appreciates various aspects of structure in it and forms internal intellectual models of that structure. This appreciation is the process whereby the awareness of the analyst automatically recognizes and focuses on the relevant aspects of a system, and forms an understanding, a mental model of them. Maharishi (1994) describes how awareness actually assumes a structure representative of the object of study:

Knowledge blossoms when the knower’s attention (awareness) falls on the object and allows the object to occupy the knower’s awareness. This occupation of the knower’s awareness by the object structures knowledge of the object in the knower’s attention, or awareness. (p. 72)

Here Maharishi describes the basic mechanics of gaining knowledge. By identifying this most fundamental structure of knowledge, and by providing a method to develop the ability to function from the simplest level of awareness, the unified field of pure consciousness, Maharishi Vedic Science provides a capability to gain complete and reliable knowledge of any system. This means that one can understand, and model all systems in a reliable manner. Maharishi (1972) locates these principles in the first expressions of the Veda:

This is beautifully expressed in Rk Veda: “Knowledge is structured in consciousness.” This means that if we want knowledge to be invariably true we should have a level of consciousness that is equally invariable. If we want complete knowledge we should have a state of consciousness which is most comprehensive. That expanded, unbounded state of consciousness should be permanently established. On that basis, whatever knowledge is gained will be true and complete. (SCI course, lesson 9)

The principles of SCI describe that the ability of awareness to reflect different levels of structure, and different aspects of a system, depends on the quality of the programmer’s consciousness. To have reliable knowledge, one must establish one’s awareness on a reliable level of consciousness. Only from the level of the unified field can a fully reliable and universal level of consciousness be gained.

The availability of such a universal and nonchanging level of awareness simultaneously fulfills the need to expand the breadth of contemporary objective sciences, and yet avoid the uncertainties and partial comprehension of less refined states of awareness.
Maharishi describes the role of refined subjectivity in forming mathematical models of systems, and the power of the ability to appreciate the refined structures of natural law in a system:

This universality of applications can be traced back to the fact that all aspects of Nature and areas of life are governed by the same principles of order and intelligence that have been discovered subjectively by mathematicians by referring back to the principles of intelligence in their own consciousness. Great scientists like Einstein have marveled in the past about this profound relation between the subjective and objective aspects in creation, a relation which has its foundation in the identity of the Unified Field of natural law and the field of pure self-referral consciousness displaying the universal principles of intelligence and order. (Maharishi Mahesh Yogi, 1996, p. 305)

By developing more refined awareness, the programmer has greater access to and appreciation of the structures of natural law in any system, and the ability to recognize their fundamental common attributes for expression into software. With the ability to develop unbounded awareness, programmers now have the opportunity to create software models of any aspect of natural law.

Note that the software model is not the same as the structure in the original system. While the programmer may appreciate the system in a very broad context, the model is intrinsically and intentionally limited to particular properties which relate to the goals of the program being created. A program is a partial expression of knowledge. Yet the scope of knowledge is not bounded; what is the relation of these two fields, what can be known, and what can be expressed into a program, i.e., what can be computed? To address this fundamental issue, we first define computation in terms of traditional computer science, and then consider the impacts of our new paradigm based on Maharishi Vedic Science.

**Computation**

The roots of computation are in mathematics. Mathematical logic deals with fundamental relations between structures. In logic the structures are the basic ideas of truth; a logical proposition is either true or false. Operations of implication connect various more complex structures of logical assertion. Using these logical formulae and operations (and,
or, not), an entire domain of logical propositions, laws, and results is formed. From logic, numerical domains can then be constructed which provide a richer expressive system, capturing all of the familiar arithmetic values and operations. From numbers, codes can be used to represent any structure or quality, including language, shape, motion, or color. These coded representations are expressed in the symbols of a language, often mathematical in its nature.

In mathematics, a sequential enumeration of the steps of a proof is called the process of deduction. It forms theorems, statements of truth, which can be proven through a direct derivation from the initial assertions, known truths. This deductive process traverses from one logical theorem to another by applying the axioms, and at each step presents a new expression of the original truth. The axioms are guaranteed to preserve this logical correctness.

Since the statements of logic are all represented symbolically as mathematical formulae, this entire process can be seen as one of symbolic manipulation. Indeed, it was the hope of early mathematicians that such logical systems could automate the entire reasoning process, and all of mathematics could be relegated to a mechanical system. This striking proposal was completely eliminated by Kurt Gödel in 1934 when he showed that there could be no formal systematic, complete, consistent finite description of mathematics (Gödel, 1934). For any consistent logical system expressive enough to include a description of mathematics, there would always be statements in the system that were true, but which could not be proven within the system. Mathematics could never be completely mechanized. Further, the truth of these statements would be available to logicians; we could know their veracity, but it would be beyond the formalized system. Mathematical truth has no finite representations.

Gödel attributed his inspiration for this theorem to the realization that “truth has no finite description” (Rucker, 1987, p. 227). Maharishi SCI presents the same conclusion; the complete nature of natural law can only be known from the unified field. One can never deduce the full nature of pure intelligence by studying the expressions of intelligence; only through direct experience of the field of pure intelli-

---

8 Gödel's Incompleteness Theorem (Gödel, 1934).
gence can one know its true nature. To realize a universal basis for all computation, we must adopt this new expanded paradigm in terms of Maharishi Vedic Science. Only with that scope can we realize the universal abstractions at the basis of all systems, and gain the ability to directly comprehend and express these structures.

**Symbolic Computation**

Computation can be defined as the systematic symbolic transformation of a system description into alternate syntactic forms, which maintain some semantic property. Therefore, computation is a purely syntactic process. In the actual computational domain, there are no intrinsic semantics; computation is merely automated manipulation of forms (see Figure 4).

However, the manipulation of syntactic forms has little value until and unless some meaning is attached to them. This meaning comes from the programmer. Any computing process starts with the subjective appreciation of some orderly structure in nature. From this the programmer defines the computational representations and processes, evaluates these to produce a new representational structure, and ends with an interpretation of this new symbolic structure to attach some meaning. The process and result of a computation has no intrinsic meaning; it is entirely a syntactic result. Any meaning or semantics of the result is attributed by the user of the system.

There are two parts to the software definition of a computation: the program and the data. The program defines the operations, the actual transformations or steps of the computation. The data are the actual values that participate in these operations. They parameterize this process to allow one specification to be used for many applications. This is similar to a general equation in mathematics, and the specific data used to find a particular solution of interest. For example, given the general definition of the value of an investment $P$ at a rate of interest $r$ after $y$ years; $V = P(1+r)^y$. To find a specific result, one provides input data for $(P, r, y)$ and solves the equation. This solution process involves the application of a series of rules, each of which simplifies the equation, until the resultant value is produced.

Similarly, a software program specifies the steps which will transform an input data specification to the desired result. The software is an
encoding of an algorithm, a general description of this solution process. Algorithms are procedurally oriented. They define a series of steps and rules which describe how to compute the desired result. Like a proof or deduction, these steps each transform one state into another. This sequence of states can be viewed as a tree, or more generally a graph in some abstract high dimensional state space. The nodes of the graph are valid states of the computation, and the arcs are valid state transitions. Paths through this space from the initial data specification state to the final result state are all valid methods of computation, and trajectories which traverse these paths are all correct algorithms.

For example, any game can be described as a state space, and the strategy for playing the game as an algorithm. For simple games, a tree can be drawn, with each node representing the choices of alternating players, and leading to the new state of the game based on the choice made. Any path through this decision tree represents a particular set of choices, one playing of the game. Any path leading to a winning configuration (game state) is a winning strategy. Other examples of computational state spaces include the evaluation of arithmetic expressions, in which there may be many different ways to simplify an expression like \((1 + 1) \times (3 - 1)\) to reach a final result of 4. It is the algorithm, in this case the rules of the game, or those of arithmetic, which guarantees that the path traverses only valid states.

It is the great richness of even a small state space and the resulting variety of possible traversals that makes computing so rich in its descriptive power and complex in its descriptions and evaluations. In a digital computer system, states are represented in a very simple manner, using numbers. Consider even a simple single number, represented in a computer as a string of binary digits, or bits.\(^9\) One 32-bit number or word in the computer’s memory, typical in today’s standard computers, can represent \(2^{32}\) or approximately four billion unique states. Using four such values, a modest 128 bits of information, represents over 300 trillion trillion trillion states \((2^{128} \approx 3 \times 10^{38})\). Ten words can represent more states than every particle in the universe. Today’s programs typically use thousands, or millions of words of data in their representations.

\(^9\) A bit is the smallest quantum of information. It means a binary digit, which has two states, 1 and 0, or On/Off. Because of its simplicity in both concept and implementation, the bit has become the standard unit of information both for mathematical theory and for actual computing hardware systems.
of a system, giving them an essentially infinite representational state space.

Note that this great representational power does not simplify the solution of complex tasks, it just allows their description. One must still evaluate the possible solutions, state space paths leading to proper (winning) states. With a complex space of possible states, this may take a very long time, in some cases so long as to make the problem intractable.10 For example, in the game of chess it is often estimated that a typical game is about 40 turns long, with each turn allowing about 30 different choices of play by each side. Thus the total number of possible moves in a game is approximately $30^{40}$ ($10^{118}$). If we assume a computer that could evaluate a billion moves per second, it would take $10^{101}$ years to make the first move; more than 1,090 times the age of the current universe.11

Clearly, given this vast complexity, and the resulting richness of structure of the state space, determining proper representations and correct algorithms for sequential traversal is a significant practical challenge. All such traversals use only local state at each node to determine the computational path. This is a necessity, as by definition this is the only information that they maintain. While some more sophisticated programs add some meta-state information to optimize the process of finding a solution, this can just be considered state augmentation, and thus still described in this model of locality of information. While the fields of algorithms, software engineering, and computational science attempt to deal with the pragmatics of this problem, there is a more fundamental alternative approach which avoids it altogether.

**Vedic Computation**

While a traditional computation can use only local information to guide its sequential traversal of state space, there are other possibilities. The state space is itself just a structure, and anyone with the knowledge of the relations and rules of this space could visualize the entire space and

---

10 A problem that is theoretically possible to solve, but would require resources that are not practical, is termed *intractable* in computer science.

11 Programs that play chess clearly do not take this brute-force approach. Instead, they use selective evaluation of possible moves and explore only a small subset of all possible states. Even this is good enough for them to play at the level of a world champion grandmaster chess player.
thereby draw immediate conclusions about all possible computations, or all possible results. Such direct insight into the nature of this abstract structure would allow one to understand properties of the entire range of all possible computations and all possible results that would not be apparent, or perhaps even feasible, by enumeration of individual sequential computations. The difference between the sequential and the simultaneous understanding of all possibilities is enormous. It is rather like the difference between a ground level traversal of a maze and a solution found by an aerial view; one is sequential exploration, the other direct perception of the result.

This ability to directly comprehend and understand such abstract systems of relations is the basis of mathematical intuition, which gives direct insight to the nature of mathematical problems and solutions. Maharishi describes the ultimate value of this familiarity with the abstract structure of natural law in terms of his Vedic Mathematics. Maharishi’s Vedic Mathematics describes the absolute, unmanifest, complete structure of natural law in terms of mathematical concepts of order, structure, and relation; it is the mathematics of the Veda, the unmanifest structure of pure knowledge. It is the full knowledge of this abstract unmanifest structure and dynamics of natural law, appreciated in terms of mathematics, in terms of structure and relationships: “As Veda is structured in consciousness, Vedic Mathematics is the mathematics of consciousness” (Maharishi Mahesh Yogi, 1996, p. 339).

Maharishi explains that because Vedic Mathematics is a direct appreciation of this structure of natural law, the direct experience of the holistic nature of natural law, it is a simultaneous cognition of all aspects and their composite wholeness; “coexistence of simultaneity and sequence characterize Vedic Mathematics” (Maharishi Mahesh Yogi, 1995a, p. 339). It is not a sequential logical intellectual exploration of individual aspects, descriptions, and implications of some particular theorem or system, as described by standard mathematics; rather Vedic Mathematics simultaneously comprehends the total value of all relationships in a system. This is similar to the traditional idea of Aha! in mathematics, where one is able to comprehend the entire structure and meaning of a problem (Gardner, 1978). However, in Maharishi

---

This term is often used to describe the process of gaining a clear insight into the structure and solution to a problem.
Vedic Mathematics the scope of such an insight is not limited to a specific problem, but includes the full unified value of natural law, as well as all of its parts, and the processes and relationships therein: “This system of simultaneously sustaining all values of relationship is Vedic Mathematics, which we call the mathematics of relationship; it handles all diversifying and unifying values of evolution simultaneously” (Maharishi Mahesh Yogi, 1996, p. 344).

![Diagram: Veda—The Unmanifest Structure of Natural Law]

Figure 6. By expanding the scope of computer science to include the absolute structures of the unified field, the traditional modes of relative symbolic computing are expanded to include the field of Vedic Computation, which deals with absolute computation.

While mathematics emphasizes the understanding of mathematical structures, computing emphasizes their application to guide some process. Computation is envisioned as a process, which produces a transformation of an input to an output. Maharishi has described how, through the regular practice of the technologies of his Vedic Science, one gains the ability to operate from the level of the unified field. Based on Maharishi’s definition of Vedic Mathematics, one can describe this ability to not only directly know the structure of natural law in a precise and systematic, mathematical manner, but also to compute from this same experiential level of the field of pure consciousness, the most fundamental level of natural law, the Veda. Such Vedic Computation would have the property that, by gaining the ability to directly
appreciate the finest level of structure of natural law, one could realize appropriate transformations of natural law within the field of pure consciousness, the Veda, to create any result.

From Maharishi’s description of this process of computation within the field of pure consciousness, it is characterized by what could be termed absolute computation (Figure 6), in that its components are all aspects of the unmanifest, abstract unified field. In this situation, there is no separation between concept, model, and machine as in relative computing as described in traditional computer science. In this case, the absolute structure of natural law is experienced as one’s own simplest state of awareness, and all expressions in creations are realized as self-referral transformations within this field of pure consciousness. These transformations of the qualities of absolute pure consciousness into the various expressed forms and structures of the universe are simply computations within this field of pure consciousness.

Because this entire process takes place within the unified field, from Maharishi’s descriptions it is a computation without steps. It realizes the result by direct enlivenment of the appropriate quality of natural law necessary to achieve any desired result. This direct cognition of the desired result spontaneously contains within it all qualities of natural law required, without the need to specifically traverse the individual intermediate transformations. Maharishi describes this quality of his Vedic Mathematics:

In Vedic Mathematics all steps are synthesized to promote the result without the need for going through the steps and stages to arrive at the goal. Vedic Mathematics is a spontaneous revelation, it is not a step-by-step derivation. (Maharishi Mahesh Yogi, 1995a, p. 389)

Maharishi Vedic Science describes the complete details of the unmanifest states of natural law and the transformations through which this unmanifest field of pure abstract intelligence expresses itself into the concrete values of creation. From the most fundamental self-referral dynamics of pure consciousness, which gives rise to the first structure of creation, further interactions between these various modes of consciousness continue to give rise to further differentiation, and so on through successive layers of further expression, until physical manifest structures arise (Maharishi Mahesh Yogi, 1995a, p. 40). Each successive stage of transformation gives rise to some new aspect
of natural law; described here as the states of the unified field, which constitute this process of Vedic computing. Maharishi describes this process in great detail, and identifies the sequential transformations from pure consciousness into its three values, and then into 40 fundamental modes which express the fundamental qualities of natural law. These 40 qualities are the canonical expressions of structure and dynamics in creations. As any further expression is some further interaction of them, they could be described as the fundamental transformations of nature, the most fundamental functions from which anything else can be computed.

Veda as the Programs of Natural Law

Maharishi describes this entire process in terms of the Vedic literature. He explains that, unlike traditional literature, where the meaning is contained in the printed symbolic forms, the Vedic literature is an expression of the unmanifest structure of natural law in the Veda:

The mathematical descriptions of nature available in the quantum field theories are descriptions by the intellect, which are grasped by the intellect. The intellectual description can at best view reality from the objective angle, in which the knower and known are separate from one another; the intellect and Being are separate. But the self-referral value is not on the level of the intellect, it is the reality itself. (Maharishi Mahesh Yogi, 1986b, p. 497)

The primary value of these descriptions of natural law in the Vedic literature is in the states of consciousness which they embody, not in the descriptions themselves. This principle is directly parallel to the concept that the meaning of software is not in its syntactic forms, but only in relation to its connections to the systems it represents. Maharishi notes that for the structures represented by, and described in the Veda, this relationship is unique in that the forms represented are not intellectual descriptions of natural law, but are the actual structures of intelligence which are those aspects of natural law. This relationship is described in Rk Veda as translated by Maharishi:

---

13 The number of qualities has expanded over time from 37 to 39 and finally to 40.
14 Maharishi uses the term Being to denote the existence aspect of the unified field.
15 Rk Veda is the first section of the Veda, and describes the overall structure of the entire Veda.
The verses of the Veda exist in the collapse of fullness in the transcendental field in which reside all the Devas, the impulses of creative intelligence, the Laws of Nature, responsible for the whole manifest universe. He whose awareness is not open to this field, what can the verses accomplish for him? Those who know this level of reality are established in evenness, wholeness of life. (Maharishi Mahesh Yogi, 1991, p. 243)

Here, Maharishi indicates that the verses of the Veda can be described as the direct expression of the structures and dynamics of natural law, the laws which govern nature’s functioning. Maharishi sometimes refers to this as the programs of natural law (MIU, 1983). However, while a program describes the laws of a particular system, the Veda is the unified field of all the laws of nature. The phrase “he whose awareness is not open to this field . . . ” emphasizes the syntactic nature of the actual verses; the symbolic descriptions of Veda are not Veda. Intellectual knowledge of the syntactic structures (the texts) of the Vedic literature does not provide access to the programs of natural law. The full values of natural law are available only to “Those who know this level of reality . . . ,” the direct self-referral experience of the Veda.

Just as in software, where the syntactic value is only from a semantic connection, this verse describes the syntactic nature of the verses when viewed as texts; the real value is in the semantics of natural law they represent. Here, however, Maharishi emphasizes that these Vedic programs are not an intellectual description of natural law. The syntactic texts describe qualities of natural law, but their real value and importance lies in the fact that they are direct expressions of the state of consciousness which is the totality of natural law.

Maharishi describes this connection in terms of the experience of the sound quality of the fundamental transformations of consciousness within the unified field. He describes that the unified field of all the laws of nature “is available to us as Shruti—vibrancy of intelligence in the form of sound generated by the self-referral dynamics of consciousness” (Maharishi Mahesh Yogi, 1995a, p. 272). These sounds, literally
the reverberations of natural law within the field of pure consciousness, are then represented as the texts of Vedic literature. He explains that “The sound of each of the 39 values of the Vedic literature is the name of a specific quality of [Natural] Law” (Maharishi Mahesh Yogi, 1995a, p. 58). This connection of the Vedic literature to the structure of natural law is therefore not based on intellectual understanding, but on a refined quality of consciousness from which one directly experiences those fundamental functions of nature. Without this level of consciousness, any meanings one might attribute to the verses would be partial, and of little value; syntactical forms with no semantic meaning. Maharishi indicates, however, that from this level of consciousness anything can be computed—this is the full value of Vedic computing.

Cosmic Computing

Maharishi presents a computational model of nature’s functioning by describing the functioning of natural law as a Cosmic Computer (Maharishi Mahesh Yogi, 1996a, p. 32). This description emphasizes the precise, systematic, and known systems of sequential transformations of all processes of natural law. The structure and descriptions of these patterns of natural law are what could be called Vedic algorithms, represented in the Vedic programs of the Vedic literature, by which nature computes the entire universe. These algorithms are the fundamental patterns of the operations of natural law, the fundamental interactions and transformations responsible for all phenomena.

In traditional computing, we emphasize the separateness of the model from the software, and the software from the hardware. This threefold division is the unique, and indeed fundamental distinguishing aspect of current computing systems. Because computer science studies the relations of these three areas, we could call it relative computing. In Vedic computing, although we can describe the transformations of natural law in terms of separate aspects of models, programs, and hardware, Maharishi explains that this separateness is in concept only (Maharishi Mahesh Yogi, 1985, p. 24). It is just an intellectual distinction used to describe the one self-referral abstract transformation of

---

16 Just as Maharishi has presented corresponding values of modern mathematics in his Vedic Mathematics (Maharishi Mahesh Yogi, 1995, p. 388), it would seem that one could locate aspects of Vedic knowledge corresponding to each major area of computer science.
consciousness within itself. We can describe the Veda as programs, but unlike traditional computing, here the programs are not distinct from the actual system which they represent; the verses of the Veda are the impulses of natural law which they describe. Similarly, the knowledge of natural law expressed as a model at this level of computation is not distinct from the system it describes; it is the system. Maharishi locates this identity of the knower of natural law and the field of natural law itself in the Vedic expression:

*Brahmavit brahmaiva bhavati*

*The knower of Brahman is Brahman itself.*

(Mundaka Upanishad, 3.2.9)

Maharishi explains that this verse indicates that the only level of consciousness comprehensive enough to know the totality of natural law (Brahman), is Cosmic Intelligence itself. At the level of Vedic computing, software and the system, which it describes, are not disjoint; they are both the unmanifest structure of self-referral pure consciousness. Similarly, the hardware on which this process is evaluated, or executes, is also the finest structure of natural law, the level of pure consciousness. Therefore, at this level of computing, the system, software, and hardware are all the same (Table 1).

<table>
<thead>
<tr>
<th>Source of Computing (semantics)</th>
<th>Process of Computing (description)</th>
<th>Hardware (evaluation)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Computer Science</strong></td>
<td>Models</td>
<td>Software</td>
</tr>
<tr>
<td><strong>Vedic Computing</strong></td>
<td>Self (Rishi)</td>
<td>Self (Devatā)</td>
</tr>
</tbody>
</table>

Table 1. Traditional computer science considers three separate aspects of description and representation of a system. Vedic Computing describes the self-referral transformations of the unified field within itself, three modes of one unified state of pure consciousness, experienced as one’s Self.
The entire process of computing is just self-referral transformations of pure consciousness within itself. This is what Maharishi describes as the Samhitā of Rishi, Devatā and Chhandas, the three most fundamental modes of pure consciousness as knower, known, and process of knowing itself (Maharishi Mahesh Yogi, 1994, p. 36). He explains that although there are three aspects or modes of pure consciousness described, these attributes are just intellectual conceptualizations of a self-referral process within the unified state (Samhitā) of pure consciousness, experienced as the Self. 

This model of Vedic Computation represents the natural conclusion of the ongoing evolution and refinement of computing systems. From mechanical to electrical, electronic, and now micro-electronic systems, computers have continued to evolve toward the limits of physical systems. Future architectures of optical computing, which compute via the interactions of fields of light, and quantum computers, which utilize quantum mechanical effects to achieve instantaneous coherence, indicate this continuing trend toward using the most refined structures known to implement our computing systems. With the discovery that these electronic and quantum fields are partial aspects of a unified field, which itself is a field of consciousness (Hagelin, 1987), computation within this unified field is the logical extension, and conclusion of all computing systems; it is the ultimate computing system.

Machine Intelligence

Because of the great sophistication and growing capabilities of computing systems, it is not surprising that the aspiration of computer scientists has been to express the maximal amount of their own intelligence, their own knowledge of natural law, in these systems. Simultaneously, users and clients of these systems, impressed by these capabilities, also begin to wonder about the quality, and limits of intelligence in these systems.

The syntactic definition and view of symbolic computation gives a clear and simple answer to questions about machine intelligence. By considering systems of symbolic computation, the actual source of intelligence, the scientist, is excluded. Programs and machines repre-

---

17 Because this experience is of the full value of one’s own simplest state of consciousness, it is often denoted with a capital letter.
sent intelligence, but the source of that intelligence is the programmer. Does this mean that computing systems are not intelligent, that they do not somehow embody intelligence, perhaps via their programs, perhaps emergent from their complex interacting components? The simple answer is yes; and this is the basis of arguments against machine or artificial intelligence. However, as programs get more and more complex, and represent or embody more and more of our knowledge of natural law, their behavior certainly becomes more closely modeled to ours, and thus it is more and more tempting to label the source of this behavior as intelligent.

Maharishi SCI describes various levels of intelligence, and how different systems embody these different levels. He explains, “A stone exists; it has its own level of consciousness,” and explains that “deep inside a rock all the electrons are moving around the nucleus and there is great activity among the finer particles. So much activity is going on, so there is intelligence, lively and active, in a stone,” and concludes that “Stone or non-stone, all have their levels of consciousness” (Maharishi Mahesh Yogi, 1972, pp. 8–9).

While Maharishi locates intelligence everywhere, and thus attributes some intelligence to all structures, he notes that different systems reflect different levels of pure consciousness, the fundamental quality at the basis of all intelligence. There is a limit to the quality of intelligence that can be expressed by different kinds of nervous systems. He indicates that the human nervous system is unique in this regard in that it can realize the full value of intelligence; it can directly experience the field of pure intelligence, the Veda. Maharishi describes that

Human brain physiology is the hardware of that Cosmic Computer which can create anything through proper programming. Human awareness has the ability to identify completely with the total potential of Natural Law, the Unified Field, which is Transcendental Consciousness, the self-referral state of consciousness. The Transcendental Meditation and TM-Sidhi programs train human brain physiology and human awareness to function completely in accord with the total poten-

---

18 This view is called “weak” AI, which proposed that no matter how sophisticated computing systems become, they only simulate intelligence, but are not themselves intelligent. The opposing view, “strong” AI, proposes that if something acts in an intelligent manner, then it is intelligent.
tial of Natural Law and spontaneously exhibit Natural Law in daily life. (Maharishi Mahesh Yogi, 1986a, p. 32)

Given this perspective on intelligence, we can see that the confusion over machine intelligence is based on a lack of clear definitions and understandings of intelligence itself. A computer is perhaps more intelligent than a rock, but less so than a human being. Further, Maharishi Vedic Science describes different levels of consciousness experienced by humans—from typical waking state of consciousness, to the full value of self-referral pure consciousness, traditionally described as enlightenment, or Unity Consciousness (Maharishi Mahesh Yogi, 1986a, p. 34). Comparing the highest values of intelligence displayed in a computing system with the limited value of intelligence of waking-state consciousness could easily confuse the actual potential of the two systems. Machine intelligence is intrinsically limited not only by the theoretical and practical limits of its programs, but also by its physical hardware structures.

Maharishi notes that the full value of human intelligence is realized only when one experiences the simplest state of awareness, the unified field of pure intelligence (Maharishi Mahesh Yogi, 1995a, p. 176; 1994, p. 260). It is this perfect self-referral state of pure consciousness, which is the field of Vedic Computation. Any more expressed physical system can display some degree of self-referral through its programs and hardware structures, but due to limitations of its relative physical structures this process will be bounded by space and time; all self-referral processes will be sequential.

This limited self-referral quality of all conventional computing systems limits the level of intelligence they can display. Maharishi indicates that there is something unique in the physical structure of the human physiology, especially the human brain, which allows it to support this highest level of consciousness. New research indicates that every level of human physiology has a direct correlation in both form and function to the absolute structure of natural law in the Veda (Nader, 1995). Maharishi describes the impact of this discovery: This knowledge has bridged the gap between mind and body, between consciousness and physiology, and between the individual, the environment, and the universe.
This discovery of the one-to-one relationship between the structure and function of the Veda and Vedic literature and the human physiology has established beyond a doubt that human physiology is the expression of Natural Law. (Maharishi Mahesh Yogi, 1995a, p. 129)

It is this perfection of structure, a physiology that reflects the structure of Veda, that is the basis of this unique and fortunate capability of the human nervous system to realize the full capability of nature’s cosmic computing.

**The Hazards of Computation**

Computing systems, based on the software programs which structure their behavior, are limited not only by their physical hardware structure and components, but by the degree to which programmers can express the abstract qualities and experiences of refined levels of consciousness and structures of intelligence into software programs. The formulation of computer science based on partial knowledge of an objective-only approach is more than limiting, it is hazardous; “A little knowledge is a dangerous thing.” Considering this aphorism in the context of the creation of machines that automate their actions based on programs, which represent this partial knowledge of natural law, raises a concern for the current and potential hazards of contemporary approaches to computing and computing technologies. The use of more and more sophisticated systems that automate partial knowledge is an intrinsically dangerous approach. Regular reports appear about incidents in which computing systems cause loss of service, value, and even life due to incomplete or incorrect programs (Neumann, 1996).

These hazards will only continue to grow with increasing applications of computing technology unless some new direction is taken. Within computer science, ongoing discussions have not yet resolved even the existence of a solution to this problem, let alone practical approaches (Fetzer, 1988). From the context of Maharishi Vedic Science, it is clear that these problems can only be overcome by providing individuals with more comprehensive knowledge and experience of the holistic values of natural law at the same time that they explore and implement more advanced technologies.

Modern scientific approaches are intrinsically deficient in this regard. Their fundamental approach excludes this—their one-sided
focus on objectivity limits their success and automatically makes them partial, and ultimately dangerous in their results. Maharishi Vedic Science and its associated technologies automatically fulfill this requirement; they provide systematic methods, which allow one to expand his awareness, and explore and investigate the full range of natural law. Only through expanding definitions of computer science to include this domain of Vedic Computation can it achieve its goals without hazardous side effects.

In addition, like all professions, the ongoing technical, professional, and personal demands of computer science and affiliated vocations can have a tiring and stressful effect. The requirement for great focus and technical precision, coupled with the required creativity for modeling and construction of large systems, requires ongoing clarity, creativity, and clear thinking. Without some reliable systematic method to culture these subjective qualities, one is not able to maintain the required refined levels of intellect and thinking required, and the result is a loss of effectiveness and resulting stress. Through the inclusion of Maharishi’s technologies for development of consciousness, the discipline and profession are transformed into activities that not only promote the welfare of the world, but also fulfill the personal and professional aspirations of the practitioners. The same growth of awareness that supports the appreciation of natural law in one’s profession, also produces the ability to act in accord with natural law in all aspects of one’s life.

The Future of Computing

Computer science provides a powerful set of computing methods, based on traditional approaches to knowledge and computing machines. However, its heritage as an objective modern science restricts it to domains that are intrinsically limited in both theory and practice. The basis of all computation is information structures, which are expressions of our understandings of natural law. Maharishi Science of Creative Intelligence and Maharishi Vedic Science locate the source of these structures in the abstract, unmanifest, unified field of natural law. By providing a precise and complete science of this abstract field, they expand the previous understanding of computing, and the entire scope of computer science.
It is only through this expanded definition of computation that the goals of computer science to realize a universal model of computation can be realized. Maharishi Vedic Science locates the universal structure of natural law within the field of pure consciousness, and indicates that to fully know the field of computer science, one must fully know the field of pure consciousness; one must fully know one’s Self.

It is fulfilling that having begun as a science investigating subjective knowledge expressed into objective informational structures, and the variety of physical machines to automate these structures, that through the integration with Maharishi Vedic Science computing now comes full circle to arrive at an ultimate definition: computation within the field of pure subjectivity, the field of pure consciousness. Einstein once said that “It is my inner conviction that the development of science seeks in the main to satisfy the longing for pure knowledge” (Einstein, 1996, p. 168). He also noted that “He who finds a thought that enables him to obtain a slightly deeper glimpse into the eternal secrets of nature has been given great grace.” It is indeed this deeper knowledge of natural law sought by modern science, that is now fortunately provided by Maharishi Vedic Science, in the ability to directly know the most fundamental levels of nature, the Veda.

Maharishi explains that by knowing this level, the source of nature’s own Cosmic Computing, one lives the full knowledge of all the laws of nature. Living this unbounded level of knowledge of nature completely changes the life of the scientist. It is not just an intellectual appreciation, but the realization that the entire process of computation at the source of creation is within one’s own simplest state of unbounded awareness. Living this level of consciousness spontaneously fulfills the highest aspirations of the discipline, theory, practice, results, and also the practitioners of computer science, or any discipline:

The objective approach to modern mathematics does not allow the subjective value of the individual to be always mathematically precise; it is only the subjective approach of Vedic Mathematics that enables the individual to be always precise and orderly—to spontaneously compute all the Laws of Nature necessary to fulfill every desire. (Maharishi Mahesh Yogi, 1996, p. 388)

Science progresses through regular changes in its fundamental world view. Each such advancement answers old questions, solves cur-
rent problems, and opens new areas of knowledge, technology, and accomplishment. The expansion of the discipline of computer science to include Vedic Computation, based on Maharishi Vedic Science, provides a complete and comprehensive framework which fulfills many fundamental limitations of the discipline, and opens the realm of computer science to include the full scope of natural law.

References


This article, “Vedic computation: Redefining computer science in the light of Maharishi Vedic Science,” by Gregory Guthrie, Ph.D., here revised/updated, and reprinted with permission, was originally published in *Modern Science and Vedic Science, 7*(1), 1997, 193–223.
Part II

Self-Referral Dynamics in Foundations of Computer Science
The Reflexive Domain:

Unified Field of Pure Knowledge

at the Foundation of Computer Science

Bruce P. Lester, Ph.D.
ABOUT THE AUTHOR

Bruce P. Lester, Ph.D., is Professor of Computer Science at Maharishi University of Management. He received his B.S. in electrical engineering in 1971, M.S. in computer science in 1971, and Ph.D. in computer science in 1974 from M.I.T. Dr. Lester was a Lecturer on the faculty of Princeton University in the Department of Electrical Engineering and Computer Science from 1973 to 1975. In 1979, he joined the faculty of Maharishi University of Management, where he founded the Department of Computer Science and served as Chair for eight years. For six years, he worked as a Senior Software Engineer at Primus Telecommunications. Dr. Lester is a widely published researcher in the area of parallel computing. He is author of the book *The Art of Parallel Programming*, which has been adopted as a course textbook by 62 universities.
ABSTRACT

When most people think of computer science, what usually comes to mind is the computer hardware—the physical box of electronics that sits on their desk. However, the intelligence and organizing power displayed by this hardware originates in the computer software, which is a symbolic representation of knowledge. The conceptual basis of computer software actually arose from the study of the laws of thought. The whole art and science of computing emerged from the study of logic, which is an attempt to codify the laws of orderly and precise thinking. Therefore, the fundamental laws of computer science are just the laws of thought, and efforts to understand the foundations of computer science are just efforts to understand the source of thought. From this point of view, it is not surprising that recent study of the foundations of computer science has located a unified field of pure knowledge, whose structure and properties closely parallel that of the Veda, as brought to light by Maharishi.

Unification in Computer Science

In its systematic investigation of the laws of nature, the science of physics has located two fundamentals at every level of organization of the universe: matter and forces. As we progress from gross to subtle levels, there is a gradual unification of the wide variety of different types of matter (fermions), and also a unification of the different types of forces (bosons). At the deepest level, “superunification” brings about the unity of fermions and bosons in the “unified field.”

A similar theme of unification is found in computer science, which has located two fundamentals at every level of computing in nature: information and procedures. Information is the “matter” or objects in computing, and procedures are the “forces” that create transformations in those objects. As in physics, investigation of subtler levels of computing reveals a gradual unification of a wide variety of different types of information and procedures. At the deepest level, information and procedures are unified into a self-interacting field of pure knowledge.

This unification is illustrated in Figure 1. The large variety of information found in various areas of computer applications can be represented with two basic types of data structures called records and lists, whose components are numbers and characters (see top line of Figure 1). Similarly, all the procedures required for applications can be encoded
using a few basic classes of statements found in programming languages: conditionals, iteration, assignment, and data structure operations.

At a deeper level of organization within the computer called the “machine language,” procedures are represented with arithmetic operations and conditionals, and information consists solely of numbers and arrays. The foundational theories of computer science have shown that a deeper level consists of recursive functions that operate on numbers. Finally, as illustrated in Figure 1, the procedures and information are completely unified into one kind of entity called lambda functions. In parallel to physics, this unification locates a self-interacting field called the reflexive domain at the source of computing. The vast variety of computing in nature can be understood as essentially consisting of the self-referral interaction of the reflexive domain at the source of computation.

**The Saṁhitā Domain**

To understand the character of the reflexive domain, it is useful to consider more deeply the unification of functions and numbers that is illustrated at the bottom of Figure 1. A function is an abstract mathematical representation of a computer program. A computer program receives some specific input values and computes some corresponding output values. A function is defined as a set of potential transformations between input and output values, as expressed in the following traditional notation:

\[ f : D \rightarrow C. \]

The function \( f \) is defined as a transformation from the set of all possible input values \( D \) to the set of all possible output values \( C \). The set \( D \) is called the domain of the function, and the set \( C \) is called the codomain. In the language of Maharishi Vedic Science, the domain is the Rishi, the codomain is the Chhandas, and the transformation is the Devatā.

For a given domain and codomain, there is a very large number of different possible functions. The notation “\( D \rightarrow C \)” can be used to represent the set of all functions having \( D \) as domain and \( C \) as codomain, as illustrated in the following equation:

\[ F = D \rightarrow C. \]
This equation defines $F$ as the set of all functions having domain $D$ and codomain $C$. Therefore, each object in $F$ is itself a function. The function $f$ defined earlier is one specific function that is a member of this set $F$.

![Figure 1. Unification of Computer Science](image)

With reference to Figure 1, the members of the set $F$ are functions and therefore belong in the “procedures” region on the left of the figure. The members of the sets $D$ and $C$ are values and therefore belong in the “information” region on the right of the figure. To unify procedures with information, it is necessary to unify $F$ with $D$ and $C$, giving rise to the following equation:

$$D = D \rightarrow D.$$
This equation states that the domain $D$ consists of the set of all functions from itself to itself. This type of domain is called a reflexive domain because the word “reflexive” refers to the unity of subject and object. The above equation states that the reflexive domain is equivalent to its own self-referral transformations. In the language of Maharishi Vedic Science, the reflexive domain is the togetherness of Rishi, Devatā, and Chhandas. The reflexive domain is the parallel in computer science to Samhitā in Maharishi Vedic Science.

**Source of Computing**

The reflexive domain brings about the desired unification of functions and values. Ordinarily, functions and the values on which they operate are distinct from each other. The reflexive domain, however, consists of objects that may play the role of functions or values. Consider any two members $f$ and $g$ of the reflexive domain. $f$ may be considered as a function that operates on $g$ yielding another element $h$, which is also a member of the reflexive domain. Similarly, $g$ or $h$ may also act as functions that operate on $f$ or on each other. $f$, $g$, $h$ may even operate on themselves! For example, $f$ can operate as a function on itself, yielding yet another element of the reflexive domain.

From this point of view, computing is just the self-interacting dynamics of the reflexive domain operating on itself. As illustrated in Figure 1, all of the diverse procedures and information found in every area of computer application can be represented by the self-interacting dynamics of the reflexive domain. Each computer program will correspond to some specific element in the reflexive domain, and each data structure corresponds to some specific element in the reflexive domain. All of the computational dynamics of any computer program operating on any data structure is contained in the self-interacting dynamics of the reflexive domain.

The existence of the reflexive domain was postulated about sixty years ago by the famous mathematician Alonzo Church as part of his development of lambda calculus. Lambda calculus is a mathematical language for expressing functions and their composition to create other functions. Church proved that lambda calculus was a universal model for computing—any computation that can conceivably be done by any computer is expressible in the lambda calculus.
However, Church’s theory of lambda calculus was incomplete because of the lack of a reflexive domain of functions to form a proper basis for the theory. It was not until 1970 that Dana Scott finally solved the self-referral domain equation \((D = D \rightarrow D)\) and proved the existence of the reflexive domain, usually denoted \(D_\infty\). This is considered to be one the most important achievements in the foundations of computer science, and laid the basis for the entire modern theory of programming languages. In recognition of this work, Scott was given the ACM Turing Award in 1976, the highest international award in computer science.

**Infinite Silence and Infinite Dynamism**

To further compare the reflexive domain to the Veda, it will be useful to consider the internal structure of \(D_\infty\) in more detail. \(D_\infty\) is defined using a sophisticated mathematical construction called the “co-limiting cone.” This co-limiting cone resembles Maharishi’s description of the “Ṛk whirlpool” that characterizes the infinite silence and infinite dynamism inherent in the Veda. This is illustrated in Figure 2. The base of the co-limiting cone is a domain \(D_0\) consisting of only two elements: an unmanifest value \(\bot\) called “bottom” containing zero information, and a value \(\top\) called “top” containing the totality of all information. As shown in Figure 2, each successive level in the cone is defined as the set of all continuous functions on the previous level. This is expressed in the following formula:

\[
D_0 = \{ \bot, \top \}
\]

for all \(i = 1, 2, 3, \ldots : D_i = D_{i-1} \rightarrow D_{i-1} \)

The size of these domains \(D_i\) grows very rapidly. Although \(D_2\) contains only ten functions, \(D_3\) contains 120,549 different functions. The functions at each successive level grow in complexity and dynamism, as illustrated in Figure 2. The base of the cone contains primitive elements that are not functions, and thus have zero activity—pure silence. Since the co-limiting cone grows to infinite height, it can be said that it ranges from infinite silence at the base to infinite dynamism at the top. The structure and properties of this co-limiting cone resemble Maharishi’s description of the structure of Veda as “Ṛk”: a whirlpool ranging from infinite silence to infinite dynamism.
The definition of the co-limiting cone also shows that each successive level is created from the self-referral interactions at the previous level. Each new $D_i$ adds yet another level of self-referral. Thus, it can be seen that the infinite dynamism of $D_\infty$ results from infinite self-referral, which is another important property of the Veda.

Figure 2. Co-limiting Cone of $D_\infty$
Pure Knowledge

The transformation between each successive level of the cone is created by a specific function denoted $\Phi_i$ which operates in the gaps between levels. This is illustrated more clearly in Figure 3. Each domain $D_i$ has an internal structure called a complete partial order. The transformation in the gap between levels creates an embedding of the structure at each level into the next level. This is illustrated in Figure 3 by the shaded groups of functions. As we progress up the cone, each new level contains a much more elaborate and complex internal structure, yet always has a complete embedding of all structures from the previous levels.

Figure 3. Transformation of the Gap
Figure 4. Each Element of $D_\infty$ is a Path in the Cone

The embedding function between levels is used to define "paths" in the cone from the base up toward the top, as illustrated in Figure 4. Each path is rooted in $D_0$ and moves upward, passing through exactly one element at each level. Since the cone has infinite height, each path is infinite in length. With all the paths drawn, the structure looks like a huge infinite tree. Each infinite path represents one single element of the domain $D_\infty$. Thus, each point in the domain $D_\infty$ contains the full range from infinite silence to infinite dynamism.

Each infinite path, which is a single point of $D_\infty$, can be represented as an infinite sequence. Consider any two points $f$ and $g$ of $D_\infty$. They will have the following general form as infinite sequences:

$$f = (f_0, f_1, f_2, \ldots, f_i, \ldots)$$
$$g = (g_0, g_1, g_2, \ldots, g_i, \ldots)$$

where each $f_i$ and $g_i$ is a member of domain $D_i$.

Recall from Figure 2 that each domain $D_i$ is defined as the set of all continuous functions on the previous level $D_{i-1}$. Therefore, each $f_i$ in the $f$ sequence is actually a function that operates on elements contained in
THE REFLEXIVE DOMAIN

Since $g_{i-1}$ is an element of $D_{i-1}$, $f_i$ can operate on $g_{i-1}$. This is represented by the following traditional notation for function application: $f_i(g_{i-1})$. It is now possible to consider the entire infinite sequence $f$ as a single function and apply it to the infinite sequence $g$ to yield a new infinite sequence $h$ as follows:

$$h = f(g) = (f_1(g_0), f_2(g_1), f_3(g_2), \ldots, f_i(g_{i-1}), \ldots)$$

$$= (h_0, h_1, h_2, \ldots)$$

By definition of the co-limiting cone, this infinite sequence $h$ is also an element of $D_\infty$. In a similar way, $g$ can be applied as a function to $f$, or $f$ can be applied to itself, or $h$ can be applied to $f$, etc. Thus, $D_\infty$ has the required “reflexive” property that brings about the unification of functions and values as illustrated at the bottom of Figure 1.

Dana Scott proved that any computer program $P$ has a corresponding element $\mu(P)$ in $D_\infty$, which contains the complete knowledge of that program in an abstract form. He also showed that any information structure $I$ has a corresponding element $\mu(I)$ in $D_\infty$. Furthermore, all of the computational dynamics of program $P$, operating on information structure $I$, is captured by the dynamics of $\mu(P)$ operating on $\mu(I)$ within the reflexive domain $D_\infty$. Thus, $D_\infty$ is the field of total knowledge in computer science. Since Rishi, Devatā, and Chhandas are unified in $D_\infty$, it can also be considered as the field of pure knowledge.

To make this concept more concrete, consider an example computer program $P$ that computes the stresses and forces on an airplane body during flight. The input to this program will be an information structure $I$ containing a mathematical description of the structure and shape of the airplane body. The program $P$ is based on the laws of aerodynamics and will solve a large system of mathematical equations that describe the properties of the airflow around the airplane body. Such a program will typically be 100,000 lines long. The input information structure may contain 100 million numbers. Even on a large supercomputer that executes 20 billion operations per second, the program may run for several hours to compute the output information structure $O$.

Within the reflexive domain $D_\infty$, there is a point $\mu(P)$ corresponding to the program $P$ and a point $\mu(I)$ corresponding to the input information structure $I$. If $\mu(P)$ is applied to $\mu(I)$ within $D_\infty$, the result will be another point $g$ of $D_\infty$. This $g$ computed within the reflexive domain will be
exactly the point \( \mu(O) \) that corresponds to the output information structure computed by the supercomputer. This example illustrates how the reflexive domain contains the total knowledge of computing in an abstract form, and therefore may be called the “unified field” of computing.

**The Reflexive Domain as AGNI**

The co-limiting cone illustrated in Figures 2 and 3 also brings out the relationship between “Ṛk” and “AK” as described in Maharishi Vedic Science. Maharishi has said that within the infinite silence of Ṛk is found the fullness of “A” and the emptiness of “K,” which form AK, the first syllable of the Veda. A similar property is found in the domain \( D_\infty \), which represents the infinite silence at the basis of the co-limiting cone of \( D_\infty \). \( D_0 \) contains two elements \( \top \) (top) and \( \bot \) (bottom). \( \top \) is defined as the value that contains all information, and therefore is parallel to the letter “A” in the Veda. Similarly, \( \bot \) is defined as the value that contains zero information, and therefore is parallel to the letter “K” in the Veda. The relationship between \( \top \) and \( \bot \) within \( D_0 \), combined with the dynamics of transformation in the first gap between \( D_0 \) and \( D_1 \), is sufficient to give rise to the entire reflexive domain.

To further investigate this parallel, it will be useful to consider more deeply the internal structural properties of the reflexive domain. Recall from Figure 3 that each level in the cone has an orderly relationship among its elements called a complete partial order, also sometimes called a lattice. This lattice becomes very elaborate at higher levels of the cone, and is a vital aspect of the reflexive domain. To help understand the nature and role of the lattice, let us consider a few simple examples of different lattices which are not part of the reflexive domain, but serve to illustrate the properties of lattices.

Consider two distinct pieces of information denoted \( t, f \). These two pieces of information may be combined into the following four sets: \{\}, \{t\}, \{f\}, \{t, f\}. These four sets have a natural relationship according to their information content. Let us use the symbol “\( \sqsubseteq \)” to denote the relationship “is contained in.” Then we can say \( \{t\} \sqsubseteq \{t, f\} \), because \( \{t, f\} \) contains \( \{t\} \) plus some additional information. Similarly, \( \{f\} \sqsubseteq \{t, f\} \). Also, since \{\} contains no information it “is contained in” all of the other sets. Regarding the \{t\} and \{f\}, it cannot be said that either is contained in the other because they contain different informa-
tion. Therefore, they are said to be *incomparable*. The relationships between these four sets are illustrated with the lattice diagram of Figure 5. A connecting line indicates the “$$\sqsubseteq$$” relationship.

![Figure 5. Lattice with Four Elements](image)

Now consider three distinct pieces of information $a$, $b$, $c$. These may be combined into eight sets whose relationship is shown in the lattice of Figure 6. In this lattice, \{a, b, c\} contains the total information of the entire lattice and so is the $\top$ (top) element. Similarly, the empty set {} is the $\bot$ (bottom) element.

![Figure 6. Lattice with Eight Elements](image)

To consider an even more complex lattice, let us generate the lattice consisting of all continuous functions on the four-element lattice of Figure 5. “Continuous” functions are those that do not disrupt the continuity of the original lattice. These 36 continuous functions are
shown in the lattice of Figure 7. For brevity, in Figure 7 the empty set is denoted $\bot$, and the set $\{t, f\}$ is denoted $\top$. Each small group of four symbols in Figure 7 indicates a single function mapping the original lattice into those four values.

Figure 7. Lattice of Functions

In all of the lattices of Figures 5–7, the bottom of the lattice is $\bot$, containing no knowledge, and the top of the lattice is $\top$, containing the totality of all knowledge in the lattice. (The term “knowledge” is especially appropriate for function lattices like Figure 7 because a function is a symbolic representation of knowledge.) The internal structure of the lattice is created by the gradual move from $\bot$ to $\top$. This is parallel to the dynamics of AGNI, the first word of the Veda, in which the collapse of fullness of “A” to emptiness of “K” negates the emptiness (“N”), giving rise to the move of emptiness to fullness (“I”). The entire diversity of natural law arises from this move. Similarly, the diverse structure of a lattice is created by the move of $\bot$ to $\top$.

This parallel to AGNI is especially appropriate with respect to the reflexive domain $D_\infty$ itself, which is also a lattice. This lattice has an infinite number of elements, each one of which is an infinite path of the co-limiting cone. This lattice of the reflexive domain has infinite
width and infinite height. It contains the universe of all possible computer programs and information structures in an abstract form. Notice in Figure 4 that there is one path going upward at the far left side of the cone rooted in $\top$. This path becomes the $\top$ element of the reflexive domain corresponding to the letter “A” in the Veda. Similarly, the path going upward at the far right side of the cone becomes the $\bot$ element of the reflexive domain corresponding to the letter “K.” The universe of all possible computing is created by the collapse of this “A” to “K” within the reflexive domain.

**The Eight Prakṛitis**

According to Maharishi Vedic Science, there are eight fundamentals at the basis of natural law, called the eight Prakṛitis: Āhaṁkār (ego), Buddhi (intellect), Manas (mind), Ākāś (space), Vāyu (air), Agni (fire), Jal (water), and Prithivī (earth). Within the self-interacting dynamics of the reflexive domain, one can locate eight abstract tendencies that correspond to these eight Prakṛitis, as summarized in the following table:

<table>
<thead>
<tr>
<th>Veda</th>
<th>Reflexive Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Āhaṁkār</td>
<td>$\Phi(D_\infty)$ (reflexive domain as functions)</td>
</tr>
<tr>
<td>Buddhi</td>
<td>$a = b \sqcap c$ (lattice meet operator)</td>
</tr>
<tr>
<td>Manas</td>
<td>$b \sqcup c = d$ (lattice join operator)</td>
</tr>
<tr>
<td>Ākāś</td>
<td>$b \not\sqsubseteq c$ and $c \not\sqsupset b$ (lattice incomparability)</td>
</tr>
<tr>
<td>Vāyu</td>
<td>$f(g)$ (functional application)</td>
</tr>
<tr>
<td>Agni</td>
<td>$\Phi$ (functional embedding operator)</td>
</tr>
<tr>
<td>Jal</td>
<td>$f \circ g$ (functional composition)</td>
</tr>
<tr>
<td>Pritivī</td>
<td>$\lambda f:E$ (lambda function)</td>
</tr>
</tbody>
</table>

Now let us discuss each of these eight in more detail. Āhaṁkār (ego) is the “I” sense or the knower aspect of pure knowledge. This corresponds to the lively value of $D_\infty$ itself when it is converted to a domain of functions by the operator $\Phi$. When $\Phi$ is applied to the entire field $D_\infty$, it converts each element into its corresponding form as a function. This gives the flat unmanifest field a lively potential for knowledge. Thus, we associate $\Phi(D_\infty)$ with Āhaṁkār.

The next three Prakṛitis (Buddhi, Manas, Ākāś) can be located in the lattice properties of the reflexive domain. Referring back to the example lattice of Figure 7, one can see the interplay of the two oppos-
ing tendencies of diversity and unity. As we move upward from the bottom, diversity grows by adding new pieces of knowledge. This growth of diversity causes the lattice to spread out horizontally. Among the Prakritis of Maharishi Vedic Science, Buddhi (intellect) is responsible for the manifestation of diversity from unity. Therefore, we associate this diversifying tendency in the reflexive domain lattice with Buddhi. This tendency in the lattice is called the meet operator as illustrated in Figure 8, showing three points \( a, b, c \) in the lattice. Adding a piece of knowledge to \( a \) causes it to manifest into \( b \). Adding a different piece of knowledge to \( a \) causes it to manifest into \( c \). The lattice relationship between \( a, b, c \) is expressed by the meet operator \( \cap \) in the following formula: \( a = b \cap c \).

\[ a = b \cap c \]

**Figure 8. Lattice Meet Operation**

In addition to this diversifying tendency, a lattice also has a unifying tendency. As illustrated in Figure 7, there is a tendency for elements at a lower level to come together at higher levels. This unifying tendency, called the join operation, gradually brings the lattice into the total unity of all knowledge at the top. Among the Prakritis, Manas (mind) unifies the five senses and five elements, and therefore corresponds to the unifying tendency of the reflexive domain lattice. Figure 9 illustrates

\[ b \sqcup c = d \]

**Figure 9. Lattice Join Operation**
the join operator for three points \( b, c, d \) in the lattice. Unifying the knowledge contained in \( b \) and \( c \) produces \( d \), as expressed in the following formula: \( b \sqcup c = d \).

Ākāśha (space) can be located in the reflexive domain as the horizontal extension of the lattice, as illustrated in Figure 7. This spatial extension of the lattice is a result of elements that are “incomparable” with respect to their knowledge content. For any two elements \( b, c \) in the lattice, this is expressed in the following formula: \( b \not\leq c \) and \( c \not\leq b \). Ākāśha is associated with sound and the sense of hearing. In Figure 7, the horizontal fluctuations of the lattice can be interpreted as vibrations of knowledge, or “sound,” within the unmanifest space of the reflexive domain.

The remaining Prakritis (Vāyu, Agni, Jal, Prithivi) are all associated with the functional aspects of the reflexive domain. Vāyu belongs to the sense of touch. In the self-referral dynamics of the reflexive domain, touch is produced by functional application which brings together a function and its input value. This is expressed in the traditional notation \( f(g) \), where \( f \) and \( g \) are any two elements of \( D_\infty \). Before any element of \( D_\infty \) can actually operate on its input, it must first be converted to its corresponding functional form by the functional embedding operator. Thus, \( \Phi \) puts the “fire” into an element, thereby giving it the potential to actually transform its input. Therefore, \( \Phi \) corresponds to Agni (fire).

Jal (water) has the property of flow. In the reflexive domain, “flow” is produced by functional composition, expressed by the notation \( f \circ g \), where \( f \) and \( g \) are any two functions in \( D_\infty \). When the composition \( f \circ g \) is applied to an input value, first function \( g \) is applied to the input, then the resultant output value becomes the input value for function \( f \). The repeated use of functional composition results in a sequential “flow” of values into different functions. Therefore, Jal corresponds to functional composition.

This flow of functional composition can be “frozen” into a structure by creating a lambda function. Since Prithivi (earth) is the structural aspect of pure knowledge, we associate Prithivi with lambda functions. A lambda function has the following general form: \( \lambda f.E \), where \( E \) is an expression involving many elements of \( D_\infty \), combined with functional application and functional composition. Thus, the lambda function “freezes” the Vāyu, Agni, and Jal qualities of the reflexive domain into a structure.
Conclusion

Having completed this brief survey of some of the properties of the reflexive domain, we feel confident in calling it the unified field of pure knowledge at the basis of computer science. The reflexive domain $D_{\infty}$ has properties of infinite self-referral, unboundedness, unmanifest, all possibilities, infinite silence, infinite dynamism, and pure knowledge. Its self-interacting structure has an internal dynamics that remind us of Ṛk and AGNI in Maharishi’s description of the Veda. The eight Prakṛitis also have corresponding abstract tendencies within $D_{\infty}$.

Studying the reflexive domain, one has the feeling of looking at a photograph of the Veda, but taken through a fog. Some of the important major features of the Veda can be seen in the reflexive domain, but the subtle details are missing. Just as physics can be said to have glimpsed the unified field of pure knowledge, so now computer science has also glimpsed this field at its own foundation. As computer science progresses in the future, we can expect more of the details of pure knowledge to emerge with increasing clarity.

References

The following references provide additional information about domain theory, lambda calculus, and $D_{\infty}$:


The Self-Referral Dynamics of Computation:
An Introduction to a Course on Algorithms

Paul Corazza, Ph.D.
ABOUT THE AUTHOR

Paul Corazza, Ph.D., received his Bachelor of Arts degree in Western Philosophy from Maharishi International University in 1978 and his M.S. and Ph.D. degrees in Mathematics from Auburn University in 1981 and 1988, respectively. He was awarded a Van Vleck Assistant Professorship at University of Wisconsin for the years 1987–1990. He worked in the Mathematics Department at Maharishi International University in the years 1990–95. Following a career as a software engineer, he rejoined faculty at Maharishi University of Management in 2004 and currently serves a joint appointment in the Departments of Mathematics and Computer Science. Dr. Corazza has published more than a dozen papers in Set Theory, focused primarily on the quest for providing an axiomatic foundation for large cardinals based on a paradigm derived from Maharishi Vedic Science.
ABSTRACT

The immense range of processes and behaviors that make up our universe is somehow managed by nature’s computational dynamics, but how exactly is all of it done? Physics has discovered mathematical models for virtually every observable phenomenon, but does nature use anything like “mathematical formulas” to accomplish its ends? Maharishi Vedic Science suggests that, at its deepest level, nature’s performance is a self-referral performance; the mistake-free creation and governance that characterize nature’s expression are due to a certain subtle level of activity that is ordinarily hidden from view—the self-interacting dynamics of silence itself.

Moreover, quantum field theory has provided a modern-day validation of this ancient wisdom in its discovery that all observables in the form of forces acting on objects are the expression of the self-interacting computational dynamics of a single unified field, a field whose all-encompassing, unified status is also ordinarily hidden from view.

In this paper we show that, in fact, the very mathematical notion of computation is based on self-referral dynamics of an expanded (in fact, unmanifest) domain of operators. Having described the way in which every computable function arises as the fixed point of such an operator, we discuss how the dynamics of creation from within this self-referral field, as described by Patanjali’s Yoga Sutras, are reflected in this mathematical space of operators as they give rise to all possible computations.

Introduction

An algorithm is a procedure or sequence of steps for computing values from given inputs. Whenever we write a program in Java or C, for example, we are providing a sequence of commands that tell the computer how to proceed in order to output the required values on a given input.

A course on algorithms considers problems that can be solved by finding a suitable algorithm and analytical methods which help to determine if a particular algorithm is more or less efficient than another one that happens to solve the same problem.

In this article, we look more deeply into what underlies algorithms—what is a computation, really? This is interesting since much of what we do depends on efficient computation, because we rely on technology (whose functioning is based on algorithms of various kinds), and
also because we do our own form of “computation” in the very way we conduct our lives—from working out a plan for the day to managing shopping lists to building a career.

In the history of scientific endeavor, man has attempted to understand nature by providing mathematical models. Modern physics has done an excellent job in providing models and computational techniques that allow us to understand and predict nature’s functioning in various ways. For example, we have a very accurate model of the motion of planets, so that we can compute with an extremely high degree of accuracy where any of the planets of our solar system will be at any time, within an enormous time span.

Procedures for modeling natural processes are, of course, algorithms, and it follows that, in modern times, understanding and predicting nature’s behavior is facilitated by the implementation of sophisticated algorithms on high-powered computers.

But how does nature itself compute the actions for which we have models? What are nature’s computations that underlie the motion of the planets and stars? Does nature somehow make use of equations of motion that have been obtained in physics, or does it use some other mechanism for computing?

Again, this is inherently interesting because nature is so powerful. So much is accomplished by the intelligence of nature—it is natural to aspire after the ability to harness the power of nature in some way.

In looking for how nature itself carries out its behavior, one must be willing to accept that many of the secrets underlying the world we see may be hidden beneath the surface. Consider this analogy: Suppose you go to see a magician perform. Perhaps he does the well-known trick of sawing a woman in half. The woman gets into an oblong box, the box is covered with its lid, and the magician pulls out a saw and starts cutting the box in half. It appears that the woman is being cut into two, and this impression is augmented by the fact that the magician moves the two pieces of the box apart, with both halves of the woman apparently also being separated. At the end of the trick, the pieces of the box are returned to their original position and the woman steps out of the box unharmed.
We must answer the question “What just happened? What was the procedure behind the performance just witnessed?” If you try the trick at home, you quickly discover that there was more to it than meets the eye.

The general principle is this: There is more to the intelligence underlying the performance than is revealed in the performance itself.

If you attempt to perform the trick by simply copying the steps that you were able to observe, you discover that your algorithm, though partially correct, falls short of the full effect.

This example is deeply related to all that we do in our lives and all that science does as it attempts to understand nature. Suppose you want to build a successful career. You may watch the behavior of other people who appear successful and so you may decide that your “algorithm” for success is to copy the steps that you have observed from these successful individuals. The problem with this algorithm is the same as the problem in reproducing the magic trick: You may not be seeing the true computational dynamics that underlie the success of these individuals. Attempting to copy their steps may therefore not be fruitful at all.

Modern physics has made a similar discovery: to understand nature’s true computational dynamics requires an investigation that probes deep beneath the surface. Physics began with an understanding that the world consists only of particles (or objects) and forces (like gravity and electromagnetism) that act on them. But a deeper understanding that emerged was that both material particles and forces are expressions of a deeper reality, called quantum fields—all particles, like electrons and protons, and all forces, like gravity and electromagnetism, are excitations of an abstract unbounded field. For instance, electrons are excitations of the electron field; gravity is an excitation of the gravitational field. The next step was the recognition that many of these fields that seem to be quite different from each other are, at a very fine level of nature’s functioning—that is, at small time and distance scales—in fact identical. Electrons and protons, at a small scale, are seen to be in fact excitations of the same field. And at the finest level of all, known as the Planck scale, all force and matter fields are seen to arise from a single unified field. (See Hagelin, 1987.)

What physics has discovered is that everything that we see—all objects, processes, and dynamics—is ultimately the consequence of certain computational dynamics of a single field, at a very fine level
of nature. It is by virtue of the unseen dynamics of this unified field that the “show” we observe in day-to-day life is made possible. These dynamics can be described in a simple way as the interaction of the unified field with itself—self-referral dynamics.

This is a remarkable discovery. It tells us that life is not truly as it appears. Things happen according to dynamics that are not obvious on the surface. We see a world of objects and forces, but both of these are effects of a much subtler cause—an abstract field interacting with itself somehow appears as this world of objects and forces.

Now we return to our initial question: What are the true computational dynamics of nature? Physics has discovered that the answer lies in this abstract unified field. And, although the mathematical description of these dynamics is quite complicated, one point about these dynamics is certain and easy to express: they are inherently self-referral. Indeed, it is by virtue of the unified field’s self-interaction that all that is observed and experienced in the physical world takes place. This discovery alerts us to an important fact about the universe in which we live: behind the scenes, self-referral dynamics are at work in the structuring of all that we experience.

The Truth about Computation Itself

We wish to examine the very idea of computation itself to discover what is fundamental. Ultimately, we would like to harness the essence of computation in some way to enhance our “computing power” both in a scientific sense and also in the subjective sense mentioned earlier. Since it seems that self-referral dynamics are fundamental for computation in nature, we are interested to see if self-referral dynamics are in fact fundamental to the very notion of computation as it is understood in mathematics.

In mathematics, a computation is performed as an implementation of an algorithm—a sequence of procedural steps is applied to a concrete set of input values in order to produce an output value. Though these steps can often be carried out “by hand,” we can describe the action of algorithms in a more general way by saying that algorithms are implemented by computer programs. For example, let us consider a Java program that implements the factorial function. Recall that the factorial of an integer \( n \) is the product of all natural numbers less than or equal to \( n \).
```java
int factorial(int n) {
    int accum = 1;
    for(int i = 2; i <= n; ++i){
        accum *= i;
    }
    return accum;
}
```

This particular implementation of the factorial function is called an **iterative program** because computations are performed iteratively in a loop. The same function can be computed in a different way using the technique of recursion:

```java
int factorial(int n) {
    if(n < 0) return 0;
    if(n == 0 || n == 1) return 1;
    return n * factorial(n-1);
}
```

Here, recursive calls are made to `factorial` in order to perform the computation. Recursion has a noticeable flavor of self-referral in it since, to compute `factorial` in this case we call `factorial` itself: we are in effect declaring `factorial(n)` to be equal to `n * factorial(n - 1)`. This fact tells us that the mathematical notion of computation involves, at least in some cases, certain self-referral dynamics. However, even this example of recursion is not a pure self-referral expression since `factorial(n)` is being computed in terms of `factorial` applied to a smaller argument `n - 1`; we do not see `factorial` being defined entirely in terms of itself. At this point in our investigation, we see very little evidence that mathematical computation is really grounded in self-referral dynamics.

As we look more deeply into the question, a first step is to notice the remarkable fact that *every* program can be rewritten as a recursive program, not just the `factorial` program. This fact can be demonstrated in the following way:

Let $f : \mathbb{N} \to \mathbb{N}$ be a computable function; that is, $f$ is a function for which we can define a Java method (or a method in any procedural language) `int java_f(int n)` with the property that, for any
natural number \( k \), if \( r = f(k) \), then the output of `java_f` on input \( k \) is \( r \). Let \( t_0 = f(0) \).

We can, in a computable way, obtain a function \( en \) that outputs, for each \( n \), the value \( f(n+1) \) from the pair of inputs \( n, f(n) \), namely,

\[
en(n, y) = \begin{cases} 
  f(n+1) & \text{if } y = f(n) \\
  0 & \text{otherwise}
\end{cases}
\]

Now we can define a function \( g : \mathbb{N} \to \mathbb{N} \) by recursion whose input/output behavior is identical to that of \( f \):

\[
g(0) = t_0 \\
g(n + 1) = en(n, g(n)).
\]

To prove \( f = g \), one reasons by induction: Certainly \( f(0) = t_0 = g(0) \). Assuming \( f(n) = g(n) \), then \( f(n + 1) = en(n, f(n)) = en(n, g(n)) = g(n + 1) \).

We have shown that all computation is reducible to recursive computation. And we can see that definition by recursion has some flavor of self-referral dynamics. A question that was asked by logicians many decades ago is this: We see that using recursion, a function like the factorial function \( \text{fact} \) can be defined by

\[
\text{fact}(n) = F(n, \text{fact}(n-1))
\]

where \( F(x, y) = x * y \). But is there a more elegant kind of self-referral dynamics whereby \( \text{fact} \) could be defined in the following, seemingly circular way?

\[
\text{fact} = F(\text{fact})
\]

Such a “definition” would be an unqualified expression of self-referral dynamics in the mathematics of computation, since, if it were possible to make sense of such a definition, the function \( \text{fact} \) would be seen to arise entirely in terms of itself, through some unseen self-interacting dynamics.

In fact, it is indeed possible to define the factorial function in this way. We examine the details of how this can be done. (See Aczel, 1977, and Weinless, 1987, for a fuller discussion.) Let \( S \) denote the collection of all functions \( g : A \to \mathbb{N} \), where \( A \subseteq \mathbb{N} \). For example, functions like
\( g(n) = 2n \) and \( g(n) = n^2 \) are in \( S \), but also functions having restricted domain are in \( S \), such as \( g(n) = \log_2 n \) whenever \( n \) is a power of 2 (here, the domain \( A \) of \( g \) is just the set of powers of 2: \( \{1, 2, 2^2, 2^3, \ldots \} \)).

We define a function \( F: S \rightarrow S \) as follows: For any \( g: A \rightarrow N \) in \( S \), let \( A' \) denote the set consisting of 0 together with all successors of elements of \( A \). For instance, if \( A = \{0, 2, 5\} \), then \( A' = \{0, 1, 3, 6\} \). Given \( g \), we let \( F(g) \) be the function \( h: A' \rightarrow N \) defined so that for all \( n \in A' \):

\[
    h(n) = \begin{cases} 
        1 & \text{if } n = 0, 1 \\
        n \cdot g(n-1) & \text{otherwise}
    \end{cases}
\]

Notice that, for \( h(n) \) to be defined, \( n - 1 \) must be in the domain of \( g \)—and this is guaranteed by our definition of \( A' \). Notice in particular that, if \( \bot \) denotes the “empty function”—the function that has no inputs or outputs, and so has domain the empty set \( \emptyset \)—then \( F(\bot) \) is a function \( h \) defined on the set \( \{0\} \) so that \( h(0) = 1 \).

Let’s show that \( \text{fact} = F(\text{fact}) \) by induction: Let \( h = F(\text{fact}) \). Then \( h(0) = h(1) = 1 \), and so likewise \( \text{fact}(0) = \text{fact}(1) = 1 \). Assuming \( h(n) = \text{fact}(n) \), we have

\[
    h(n + 1) = (n + 1) \cdot \text{fact}(n) = \text{fact}(n + 1).
\]

By induction, it follows that \( h = \text{fact} \). We therefore have a completely “circular” or self-referral definition of the factorial function:

\[
    \text{fact} = F(\text{fact}).
\]

Notice that here the input argument \( \text{fact} \) of \( F \) is the same as the output—\( \text{fact} \) is called a fixed point of \( F \). This observation is an instance of an important mathematical result:

**Recursive Operator Theorem.**\(^1\) For every recursively defined number-theoretic function \( f \), there is an operator \( F: S \rightarrow S \) such that \( f \) is the least fixed point of \( F \); that is, \( f \) can be defined by

\[
    f = F(f).
\]

---

\(^1\)The name we have given to this theorem is for the convenience of this paper; in the literature, this result has not been formally assigned a name. The idea behind the proof is that the rules (in the sense of Aczel, 1977) that define a given recursion can be canonically transformed into a monotone continuous operator \( F: S \rightarrow S \) whose least fixed point is the solution to the recursion.
We have shown that all computation is fundamentally self-referral computation. However, this truth about computation is not obvious on the surface—in fact, this insight is not available in the ordinary framework in which computation is done in applications. In order to discover this truth about computation, it was necessary to consider an expanded context: our discovery was made in the world of operators $S \rightarrow S$, which is a much larger infinite domain than that of computable functions.

This fact, that an expanded context is necessary to discover the self-referral nature of computation, parallels the discovery of self-referral dynamics at the basis of the physical universe: the self-referral dynamics at the basis of the universe are discovered in physics at the Planck scale—that is, at extremely small time and distance scales. Therefore, these dynamics are ordinarily not apparent in the ordinary world of forces acting on objects.

**Self-Referral Dynamics in Life**

The questions “How does nature compute?” and “What is the ultimate truth about the nature of computation?” have also been addressed by the ancient texts and brought into the contemporary stream of knowledge by Maharishi Mahesh Yogi. According to Maharishi Vedic Science, the life that we see, with one step of the story of our lives coming after another, is in fact a manifest, obvious unfoldment of unseen self-referral dynamics, the self-referral dynamics of pure intelligence (Maharishi, 1995, pp. 165–171). Nature’s computation is in fact a self-referral performance. Each expression in the manifest world is the result of pure consciousness curving back on itself, collapsing from its infinitely expanded value to a point (Maharishi, 1996, p. 539). The Veda declares: *Richo akshare parame vyoman.* . . . This means that the fundamental structuring impulses of the creation arise in the collapse of the fully expanded value of intelligence to its point, and subsequent expansion from point to unboundedness (Maharishi, 1996, p. 484).

The sequential flow that is observed in natural processes and the sequential nature of computations as we do them in mathematics and computer science are, from the Vedic perspective, expressions of an eternal self-referral process, a “hum” within the unboundedness of pure intelligence (see Lester, 1987, pp. 306–309).
These mechanics are embodied in the first word of Rik Veda: AGNIM. ‘A’ represents the fully expanded value of pure intelligence, and ‘K’ (note: A + K = AG) represents the fully contracted value, the point value. ‘N’ is negation, which in this case means “negation of the contraction,” in other words, expansion back to unboundedness. ‘I’ is continuation, so we find a continuation of collapse and expansion. And ‘M’ is the simultaneous occurrence of all frequencies, representing the infinitely frequent collapse and expansion embodied in the previous letters (for a more detailed discussion with references, see Lester, 1987, pp. 306–309).

Let us now ask: What is the practical value of this knowledge? For an answer, let us return for a moment to the magic show mentioned earlier. Suppose you decide you want to be able to do the trick the magician did. As we have observed already, without gaining access to the unseen dynamics of the magician’s performance, it will be impossible to do so. The key to success in this case is the ability to gain access to, and make use of, these hidden dynamics.

Analogously, to achieve the greatest success in pursuing the many goals we have for our own lives, the key is to gain access to, and make use of, the hidden dynamics of life itself; in this case, we seek to harness the unseen dynamics by which nature governs the universe. Such an immense intelligence and power would of course be of great value for our own individual lives. In building a career, a family, wealth, personal fulfillment—there really would be no limit to the benefits from tapping nature’s structuring dynamics. Just as in the case of the magician, these dynamics are not obvious on the surface; it is necessary to dig deeper to access this level of life.

Achieving this goal is the purpose of the Transcendental Meditation technique that we teach as part of the curriculum at Maharishi University of Management. It provides a key to tap this vast field of intelligence and provides the practical value—the “technology”—of the Vedic wisdom we have been discussing. One of the ancient Vedic texts, called the Bhagavad-Gita, summarizes in three verses the practical application of the wisdom of the ancient texts (Maharishi, 1969):
Regular dives into the unbounded aspect of our own intelligence automatically accomplish these aims. Contact with the transcendental field is contact with the field from which nature operates. The Veda describes the transcendent as *yasmin deva adhi visbve nishedub*—the home of all the laws of nature (Maharishi, 1996, pp. 113, 515). Waking up to this field automatically brings nature’s self-referral dynamics into the flow of our individual lives, leaving “action devoid of greatness” far behind.

**Directly Activating Nature’s Self-Referral Dynamics**

In the Transcendental Meditation program, we stir the transcendental field and automatically bring benefits into our daily life. But it is possible to more directly engage nature’s self-referral dynamics through an advanced program, the TM-Sidhi program.

In the TM-Sidhi program, acting from the settled state of awareness arising in meditation, one introduces a faint intention for a particular result or “Sidhi.” The intention is a kind of point value, and our settled state of awareness, which entertains this intention, is at the same time awake to the unbounded value of intelligence. So there is an interaction between point and unboundedness, and in the togetherness of these, a “result” comes—the result that was originally intended.

Maharishi Patanjali, part of the Vedic tradition of knowledge, described this technique of engaging nature’s intelligence in the following way in the Vedic text called the Yoga Sutra (see Aranya, 1977). He explains that it is by virtue of the togetherness of *Dharana* (intention), *Samadhi* (transcendence), and *Dhyana* (flow of awareness) (**Yoga Sutra**...
3.1, 3.2, and 3.3) that the result of the intention, a particular Sidhi, is produced. Patanjali calls (Yoga Sutra 3.4) the togetherness of these three Samyama. The TM-Sidhi program makes use of Patanjali’s principles in a simple and automatic way. The result is that a Sidha is able to more directly engage nature’s self-referral dynamics (see Wallace, 1986).

If we take a closer look at the deeper self-referral dynamics that underlie computation itself, we can see that the way nature computes in its self-referral way, particularly in the way in which Sidhis arise from Samyama, is strikingly analogous to the way computation itself works at the deeper mathematical levels. To see the analogy, let’s return to our computation of the factorial function fact using the operator $F: S \to S$. Recall we were able to show that

$$\text{fact} = F(\text{fact}).$$

Let us now see how the function fact actually emerges by way of self-referral dynamics within the expanded domain of operators $S \to S$ in much the same way as a Sidhi arises from Samyama.

We begin with $F$; $F$ is the operator that has been specifically encoded with information that is intended to produce the factorial function. It represents the intent behind our self-referral computation and corresponds to Dharana.

Next, we allow $F$ to interact with the empty function $\bot$ by application: $F(\bot)$. The empty function is the function that accepts no input and has no output, corresponding in programming terms to a function such as the following:

```cpp
void empty() {
    while (true) {  ;  }
}
```

The empty function can be viewed as a field of all possibilities—no commitment has been made to particular input/output behavior; it is an analogue to the unbounded nature of intelligence, and corresponds in our analogy to Samadhi. We now apply $F$ to $\bot$ not just once but repeatedly.

$$(\star) \quad \bot \subseteq F(\bot) \subseteq F(F(\bot)) \subseteq \ldots$$

This sequential unfoldment is like the flow of awareness described by Patanjali—Dhyana.
In this togetherness of intent (F), transcendence (⊥), and flow (*), what emerges is a result, a computation. The result in this case is obtained by forming the union

\[ h = ⊥ \cup F(⊥) \cup F(F(⊥)) \cup \ldots. \]

One can then show that \( h \) is in fact the (unique) fixed point of \( F \):

\[
\begin{align*}
F(h) &= F(⊥ \cup F(⊥) \cup F(F(⊥)) \cup \ldots) \\
&= F(⊥) \cup F(F(⊥)) \cup \ldots \\
&= h.
\end{align*}
\]

Moreover, as we showed before by induction, such a fixed point for \( F \) must actually be the factorial function itself. That is,

\[ h = \text{fact}. \]

**The Language of Self-Referral in the Field of Computation**

We have seen how, by expanding the context of study from computable functions \( \mathbb{N} \rightarrow \mathbb{N} \) to the domain of operators \( \mathcal{S} \rightarrow \mathcal{S} \), we can locate the self-referral dynamics underlying all computation. In the 1930s, a mathematical language, called the \( \lambda \)-calculus, was developed by Alonzo Church (1932/3) to provide a mathematical foundation for computation. This language expresses these self-referral dynamics in a more foundational way.

Church was seeking to build all of mathematics from the concept of transformation, the concept of a function. Though the result of his research fell short of being a foundation for all of mathematics, it turned out to be a foundation for the notion of computation. Church envisioned a universe built up entirely of functions. An important implication of such a universe is that, not only does a function \( f \) act on its inputs (as is usually the case with functions), but the domain (and range) of \( f \) includes all possible functions as well, including \( f \) itself. In particular, this means that some meaning must be attached to a term of the form \( f(f) \). Any such universe would have to be highly self-referral.

Church’s formal \( \lambda \)-calculus has the following elements: An infinite set of variables \( v_1, v_2, \ldots \) together with \( \lambda \)-terms defined recursively by:
(1) Any variable is a term.
(2) (Application) If $M$ and $N$ are terms, $(MN)$ is a term.
(3) (Abstraction) If $M$ is a term and $x$ is a variable, $(\lambda x.M)$ is a term.

The $\lambda$-notation is a way of specifying a function. Stepping outside the formal $\lambda$-calculus for a moment, one can use the $\lambda$-notation to specify ordinary functions in a convenient way. For instance, the function that takes $n$ to $2n$ can be expressed by writing,

$$\lambda n.2n.$$  

The expression is to be understood as saying, “$\lambda n.2n$ is the function that takes argument $n$ to the value $2n$.”

In our definition of $\lambda$-terms, clause (3) says, intuitively, that if $M$ is a term, the “function” that takes $x$ to $M$ is also a term. Multiple applications of clause (3) are typically expressed in abbreviated form. For instance $\lambda x.\lambda y.M$ is written $\lambda xy.M$. The complexity of nested parentheses that can occur in practice, in writing down terms according to rules (2) and (3), tends to become unmanageable, so there is an “association to the left” convention: It is understood that the expression $MNP$ is shorthand for $((MN)P)$.

$\lambda$-terms can often be “reduced” to simpler terms. Many of these reductions follow our intuitive expectation. For instance, the term $(\lambda x.y)M$ reduces to simply $y$. The intuitive reason is that $\lambda x.y$ is the function that takes any $x$ to constant value $y$. If this function is applied to the input $M$, the output is simply $y$. We write $(\lambda x.y)M \rightarrow y$. As an exercise, the reader may wish to verify (intuitively, as we have done here) why $(\lambda x.x(xy))M \rightarrow M(My)$.

More formally, the $\lambda$-calculus specifies rules for transforming $\lambda$-terms to other $\lambda$-terms (see Barendregt, 1984). The key axioms of the $\lambda$-calculus are the $\beta$-rule, for performing a reduction (as described in the previous paragraph), and the $\xi$-rule, which says that equal terms remain equal after abstraction. Here are the formal statements:

$\beta$-rule:  $(\lambda x.M)N = M[x/N]$.

$\xi$-rule:  if terms $M$, $N$ are equal, then terms $\lambda x.M$, $\lambda x.N$ are also equal.
Here the notation $M[x/N]$ means that all occurrences of $x$ are replaced by inserting the term $N$ (there is a small technical point about how $N$ is to be inserted, but we skip over this issue in this brief introduction).

Notice that the $\lambda$-calculus doesn’t have much in it—just variables, two rules for combining the variables into terms, and two rules to regulate interaction of terms. This simplicity is reminiscent of pure intelligence itself—there’s nothing much there! The formula for the dynamics of this simple field of pure intelligence is given by the first word of Rik Veda: AGNIM. Pure intelligence collapses to a point and expands to infinity—this is the character of its self-interacting dynamics. Analogously, in the $\lambda$-calculus, the self-interacting dynamics of $\lambda$-terms are governed by two rules: Application (the $\beta$-rule), which describes how the infinitely expanded value of an abstraction term is collapsed to a point, and Abstraction (the $\xi$-rule), which regulates the process by which points may be expanded to the infinitely expanded value of an abstraction term or “function.”

As we have seen, the $\lambda$-calculus gives rise to $\lambda$-terms. Among the terms of the $\lambda$-calculus are the objects that are familiar to us for use in computation: We can locate a copy of the natural numbers and representations of the concept of an ordered pair, for example. Even more significantly, we can represent all computable functions in the language of the $\lambda$-calculus, and when this is done, the $\lambda$-calculus allows us to prove that every computable function arises as a fixed point of a $\lambda$-term, and, moreover, the very process of recursion itself is seen to be the result of locating a fixed point for a suitable $\lambda$-term. (Details are given in the Appendix.) Therefore, the insight discussed earlier—that every computation arises through self-referral dynamics of an expanded operator domain, in the sense that each arises as the (least) fixed point of one such operator—is given precise expression in the language of the $\lambda$-calculus. In other words, the language of the $\lambda$-calculus reveals the hidden self-referral dynamics of computation.

As a final observation showing the natural link between the language of the $\lambda$-calculus and the self-referral dynamics of pure intelligence, we recall that the transformations of the field of intelligence naturally divide into three: knower (Rishi), object of knowing (Chhandas), and process of knowing (Devata). The $\lambda$-calculus itself has this
natural threefold division: Being a purely functional language, its essence is Devata, or transformation. What emerges within this field of pure dynamism are “precipitations”: numbers, ordered pairs, computable functions, etc. These are the objects of knowledge, the Chhandas value. And the driving intelligence behind all of it is the simple set of axioms for the λ-calculus which regulate the dynamism of this domain; these two axioms (the β-rule and the ξ-rule) happen to correspond to the two fundamental directions of flow of pure consciousness itself: collapse to a point (application) and expansion to infinite (abstraction).

Conclusion
In this introduction to Algorithms, we began by asking about the nature of the computing process, and in particular, how nature performs the computations that produce the flow of existence that we observe in the world. An analogy from everyday experience suggested to us that the truth concerning nature’s way of computing, and concerning computation in general, is very likely to be hidden from view, not obvious from casual observation. Moreover, a survey of recent discoveries from quantum physics suggests that nature’s functioning, at its basis, is inherently self-interacting and entirely beyond the range of observable phenomena.

We asked whether this self-interacting characteristic, inherent in nature’s fundamental computational dynamics, is actually a characteristic of the mathematics of computation itself. Our investigation showed that this is indeed the case: Every computation, represented in the form of an algorithm or computer program, can be formulated as a recursive computation, and every recursive computation can be seen to arise as a fixed point of an operator on the class of partial functions from N to N; that is, from purely self-referral dynamics arising in an expanded function space. Moreover, we showed how this phenomenon, that recursive computations arise as fixed points of operators, finds natural expression in the language of the λ-calculus, in which one can demonstrate that every computation arises as a fixed point of a λ-term. These mathematical self-referral dynamics, at the heart of mathematical computation on N, are, like the dynamics of the unified field, “hidden from view” in the sense that they can be discovered only in a domain (the domain of operators from S to S) that is far vaster than the ordinary domain of computable functions.
Finally, we looked into the practical application of this insight about computation and how nature computes. A natural desire is to harness for oneself the enormous intelligence and organizing power of nature’s self-referral functioning; the practical application is to dramatically uplift every area of life by bringing to bear nature’s cosmic level of functioning to the realm of individual life. The key to accomplish this goal, as we discussed, is to open individual awareness to the level at which nature carries out its self-referral performance—the level of the unified field of all the laws of nature. Simply by contacting this field, through the Transcendental Meditation program, one immediately begins to experience the influence of this field in daily life. Moreover, we saw that one could further engage nature’s intelligence through the Maharishi TM-Sidhi program, a program that provides a modern, easily practiced formulation of the Sutra practice described in the classic Vedic text, Patanjali’s Yoga Sutras.

The vision proclaimed by the ancient texts—that nature’s organizing power has its basis in the self-referral dynamics within the field of pure consciousness, and that simply allowing awareness to dive into this field of pure consciousness is enough to draw nature’s powerful organizing dynamics to support individual life—finds validation in the very mathematics of computation: modern mathematics itself proclaims that all computation is, at its basis, a self-referral performance of a deep, infinitely vast field.

References


**Appendix**

We give here some of the details of the syntax of the $\lambda$-calculus and illustrate the point made earlier that computable functions can be represented in the $\lambda$-calculus and arise in every case as fixed points of $\lambda$-terms (for a more detailed treatment, see Barendregt, 1984). To illustrate these points, we once again use the factorial function ‘fact’ as a point of reference.

We list some $\lambda$-terms that are used frequently and mention a few results that can be proven:

1. $I = \lambda x. x.$
2. $\omega = \lambda x. xx.$
3. $\Omega = \omega \omega$.
4. (Truth values) $T = \lambda xy.x$, $F = \lambda xy.y$.
5. (Conditionals) Suppose $B$, $M$, $N$ are $\lambda$-terms. Represent “if $B$ then $M$ else $N$” by the term $BMN$
6. Note that
$$TMN \to M$$
$$FMN \to N$$
7. (Ordered pairs) Given $\lambda$-terms $M$, $N$, we represent the ordered pair $(M, N)$ by
$$[M, N] = \lambda z. zMN$$
We also represent the operations that extract the first and second elements of a pair by:
$$\langle X \rangle_0 = X T$$
$$\langle X \rangle_1 = X F$$
It therefore follows that $(\langle [M, N] \rangle)_0 \to M$ and $(\langle [M, N] \rangle)_1 \to N$.
8. (Natural numbers) We represent the natural numbers $0, 1, 2, \ldots$ as $\lambda$-terms using the notation $\bar{0}$, $\bar{1}$, $\bar{2}$, $\ldots$, defined as follows:
$$\bar{0} = I$$
$$\bar{1} = [F, \bar{0}] \to [F, I]$$
$$\bar{2} = [F, \bar{1}] \to [F, [F, \bar{0}]]$$
$$\bar{3} = [F, \bar{2}] \to [F, [F, \bar{1}]] \to [F, [F, [F, \bar{0}]]]$$
$$n + 1 = [F, \bar{n}]$$
9. (Predecessor and Successor) Represent the predecessor and successor functions of arithmetic by $P$ and $S$, respectively:
$$P x = \lambda x.xF$$ (it follows that $P \bar{n + 1} = \bar{n}$)
\[ S \ x = \lambda x. [F, x] \] (it follows that \( S \ n = n + 1 \)).

10. \textbf{(Test for zero)} The term \textit{Zero} allows us to test whether a term is the term 0.

\[
\text{Zero } x = \lambda x. xT.
\]

This definition allows us to prove that

\[
\text{Zero } 0 = T
\]

\[
\text{Zero } n + 1 = F.
\]

11. \textbf{(\(\lambda\)-definable functions)} We can represent all computable functions \( N \to N \) in the \(\lambda\)-calculus. We focus on functions defined on all natural numbers (total functions), but there is an analogous treatment for partially defined functions as well. We say that \( f: N \to N \) is \(\lambda\)-definable if there is a \(\lambda\)-term \( F \) such that, for all \( k, n \), we have

\[
f(k) = n \quad \text{if and only if} \quad Fk = n.
\]

In other words, for all \( k \),

\[
Fk = f(k).
\]

12. \textbf{(Multiplication is \(\lambda\)-definable)} It can be shown that ordinary addition and multiplication are \(\lambda\)-definable. We assume this here and give the name \textit{Mult} to the \(\lambda\)-term for multiplication. Therefore, for any natural numbers \( m, n \), we have by \(\lambda\)-definability:

\[
\text{Mult } m \ n = m \cdot n.
\]

13. \textbf{(Fixed-Point Theorem)} We think of \(\lambda\)-terms as functions, defined on this highly dynamic domain. The Fixed-Point Theorem tells us that every \(\lambda\)-term has a fixed point.
**Fixed-Point Theorem.** For every $F$ there is $X$ such that $FX = X$.

**Proof.** Let $W = \lambda x. F(xx)$. Let $X = WW$. Then

$$X = (\lambda x. F(xx))W \rightarrow F(WW) = FX.$$ 

**Generalized Fixed-Point Theorem.** Suppose $C = C(f, x, y, z)$ is a $\lambda$ term involving only the free variables $f, x, y, z$ (and similarly for bigger sets of variables). Then there is a $\lambda$-term $F$ such that for all $\lambda$-terms $X, Y, Z$,

$$FXY Z \rightarrow C(F, X, Y, Z)$$

(where $C(F, X, Y, Z)$ is obtained from $C(f, x, y, z)$ by substitution). (Notice that in the theorem statement, $F$ is defined in terms of itself.)

The Generalized Fixed-Point Theorem is proved by using the Fixed-Point Theorem to obtain a fixed point for

$$\lambda fxyz. C(f, x, y, z).$$

**Example of Generalized Fixed-Point Theorem.** We show that there is a $\lambda$-term $F$ such that

$$Fxy = FyxF.$$ 

We use the Generalized Fixed Point Theorem to prove this. Let $C(f, x, y)$ be the term $fyxf$. Let $F$ be the $\lambda$-term given by the theorem. Then $F$ satisfies $Fxy \rightarrow FyxF$.

14. (Recursive factorial defined in $\lambda$-calculus) This example shows how factorial is represented in the $\lambda$-calculus as a fixed point of a $\lambda$-term. Equally important, it shows how recursion itself arises via the existence of fixed points for $\lambda$-terms, illustrating explicitly the self-referral dynamics underlying recursion and computation generally. We show that factorial function $\text{fact}$ is $\lambda$-defined by
the $\lambda$-term $F$ specified by the following self-referral expression (guaranteed to exist by the Generalized Fixed-Point Theorem):

$$Fx = (\text{if } \text{Zero } x \text{ then } 1 \text{ else } \text{Mult } (F(Px))x).$$

We prove that $F$ $\lambda$-defines fact by induction on $n$. If $n = 0$, then

$$F0 = 1 = \text{fact}(0).$$

Assuming $F n = \text{fact}(n)$,

$$Fn + 1 = \text{Mult } (F(P n + 1)) n + 1.
= \text{Mult } F n + 1.
= \text{Mult } \text{fact}(n) n + 1.
= \text{fact}(n) \cdot (n + 1)
= \text{fact}(n + 1).$$
Part III

Principles of *Maharishi Vedic Science* in Software Engineering
Beyond Extreme Programming:
Absolute Programming,
Broad Awareness, and Fine Focus of Attention

Keith R. Levi, Ph.D.
Keith Levi, Ph.D., is Professor and Chairman of the Department of Computer Science at Maharishi University of Management. Dr. Levi received his B.S. Summa Cum Laude from Maharishi University of Management in 1979, and then an M.A. in Statistics, M.S. in Computer Science, and Ph.D. in Mathematical Psychology from the University of Michigan in 1985. Dr. Levi was an Associate Fellow of the NASA Center of Excellence in Manned Systems, a Doctoral Fellow for the VA Health Services Research and Development Office, and a National Science Foundation Graduate Fellow. Prior to joining the faculty at Maharishi University of Management, Dr. Levi worked for Honeywell Systems and Research Center as a Senior Principal Research Scientist; his work there involved applications of artificial intelligence and expert systems to human-machine interface problems. Dr. Levi has been a principal or associate investigator on research awards from the Air Force’s Wright Research and Development Center, the Office of Naval Research, the United Nations’ Industrial Development Office, the Office of Naval Research, and the Army Research Institute. Dr. Levi has been a committee member or reviewer for conferences including the Eleventh Applications of Artificial Intelligence Conference, the International Conference on Machine Learning, and the International Joint Conference on Artificial Intelligence. Dr. Levi has been a reviewer for the journals: *Heuristics; IEEE Pattern Analysis and Machine Intelligence; IEEE Expert;* and *IEEE Systems, Man, and Cybernetics.* He has also been a grant reviewer for the National Science Foundation and the Air Force Office of Scientific Research. Dr. Levi has published or presented over 50 technical papers.
A B S T R A C T

Successful creation of a large modern software system requires broad situational awareness and fine focus of attention to detail. Developers must take into account the overall holistic structure of the system and the impacts on various other parts as they make local extensions and changes. Large software systems now involve millions of lines of code, often developed and deployed in multiple sites around the globe, are developed by large teams of people working under demanding budget and schedule constraints, and involve intricate and complex coding of many different subsystems that must change and evolve over time. This means that developers must take into account a broad range of factors at the same time as they are focused on the fine details of code implementation.

Introduction

Modern software engineering techniques share a common feature that they promote successful software development in a challenging environment by supporting the requirement for developers to maintain global awareness even while they are focusing on details of implementing specific pieces of code. This paper will explain this aspect of modern software engineering practice, including how it is realized by standard practices in requirements analysis, design, reviews, and testing.

Consciousness-Based education (Dillbeck and Dillbeck, 1987) is an educational approach that directly expands this critical capability of broad awareness along with fine focus of attention. It has long been established that there are huge individual differences in productivity between developers, up to a twenty-fold difference for top developers (McConnell, 2004). Consciousness-Based education offers proven techniques to unfold the full potential of developers in this critical regard. Consciousness-Based education does not radically change traditional educational methods other than to add principles and techniques for developing the consciousness of the student.

This paper introduces a consciousness-based software development process called “Absolute Programming.” Absolute Programming does not radically change traditional software engineering processes other than to add principles and techniques for developing the consciousness of the software engineer. Absolute Programming, however, goes
beyond methodological procedures to directly develop the consciousness of the developer so that he or she is always thinking and working from a level of awareness that appreciates the bigger picture of the solution while simultaneously supporting fine focus of attention to detail, even in the context of intense activity and demanding schedule and performance requirements.

The paper will introduce Absolute Programming by considering not only standard software engineering processes, but also recent developments in software engineering processes. In particular, the paper examines agile development methodologies and the agile process called Extreme Programming (XP). Agile development methodologies are quickly becoming accepted as industry “best practices.” Extreme Programming is based on a number of techniques that are quite surprising when first encountered. This paper will explain how these practices support the requirement for broad awareness along with fine focus.

The paper has two parts. The first part of the paper examines software engineering best practices in terms of how they all promote broad awareness along with fine focus. The second part presents an explicit model of the concept of broad awareness and fine focus and shows how this critical ability can be fully developed in individual software engineers through the Transcendental Meditation and TM-Sidhi programs.

Part I: Understanding Software Engineering Best Practices in Terms of Broad Awareness and Fine Focus

Software Engineering Is an Inherently Complex Process
In his classic paper, “Essence and Accident in Software Engineering: No Silver Bullet,” Brooks (1987) made the point that there is a set of defining characteristics for developing software that are intrinsic to this activity and which make it inherently a difficult and demanding activity. He grouped these inherent characteristics into four categories:

- Complexity: Modern software systems are complex in the sense that they are extremely large in size; further, they are often required to be interactive, distributed, and networked; and they are required to process combinations of states and
processes that far exceed the capacity of any hardware systems that have yet been imagined.

- Conformity: Software is generally expected to conform to hardware and existing business processes rather than hardware and business processes being redesigned to accommodate software requirements.

- Changeability: Software systems are expected to be modified and adapted to changing requirements in their area of application, even though software systems tend to be far more complex and costly these days than the underlying hardware on which they run.

- Invisibility: Software is abstract relative to most other artifacts created by humans. It is much easier to envision and visually model things such as buildings, vehicles, bridges, etc. than software.

In contrast to the essential characteristics of software development, Brooks listed other characteristics of software development that are not inherent to the activity. He called these characteristics “accidental” characteristics. He noted that we had already made significant advances in the field by eliminating accidental features, but that most of the obvious advances had already been made and future progress in these areas would be more limited. Aspects of software development he characterized as “accidental” included things such as the syntax of early low-level programming languages, lack of time-sharing on early computers, and lack of good development environments. In contrast, the characteristics given above which he described as “essential” or inherent in the activity were ones that he claimed could not be removed by automation but rather were inherent in the activity and ensured that software development would always, by its underlying nature, be a challenging and difficult undertaking.

“Studies have shown that for every six new large-scale software systems that are put into operation, two others are canceled. The average software development project overshoots its schedule by half; larger projects generally do worse. And some three quarters of all large systems are ‘operating failures’ that either do not function as intended or are not used at all” (Gibbs, 1994).
Only 16.2% of software projects are completed on time and on budget. In larger companies, a meager 9% of technology projects come in on-time and on-budget. In addition, about one third of all projects will be canceled before they ever get completed. Further results indicate 53% of projects will cost an average of 189% of their original estimates. In financial terms this analysis revealed that over $100 billion in cancellations and $60 billion in budget overruns occur in the Software Sector annually (The Standish Group, 1995).

It is not surprising that software development is so challenging when one considers the requirements for successfully developing a large software system. As Brooks pointed out, modern systems are huge in size (tens and even hundreds of millions of instructions); they involve highly complex interactions among many subsystems; developers are expected to understand the business or domain area in which they work in order to make their systems conform to the business practices; the system requirements often change over time and typically even as the system is being developed; and it can be difficult to visualize the large, abstract, and interacting system one is developing. In addition, developers must be programmers and highly skilled at writing extremely detailed instructions for executing complex processing of the many individual parts of the system.

Therefore, one of the most significant challenges of complexity is just getting the small parts to work by themselves. Far more challenging, however, is to have all of the small parts work together properly in the complete system; and even more challenging than getting the software system to work is getting the software system to work properly in the context of the business or application environment. In other words, to be a successful software developer, one must be able to have very fine focus of attention on the details of individual software components, and at the same time maintain a broad awareness of the ramifications of local changes on the entire software system and the entire application environment.

**Software Engineering Processes Provide Aids for Broad Awareness**

Modern software engineering processes provide a general context to support detailed programming. A variety of software engineering pro-
cesses have developed over the last 20 years to support the development of large software systems.

The best software organizations control their projects to meet defined quality targets. They accurately predict software delivery dates months or years in advance. They deliver their software projects within budget, and their productivity is constantly improving. Their staff morale is high, and their customers are highly satisfied. (McConnell, p. xv, 2004)

The presence of a well-defined and well-managed process is a key discriminator between hyperproductive projects and unsuccessful ones. (Jacobson, *The Unified Software Development Process*, 1999, pp. 3–4)

The original and most widely known software engineering process is known as the waterfall model. In this process model development proceeds sequentially through a series of five software phases:

- system analysis—specifying the application context and scope
- requirement analysis—specifying functionality and constraints on implementation and usage
- software design—stepwise refinement from more abstract to more specific procedures that will implement the specified requirements
- implementation—writing the code to implement the design
- test—testing the written code

Breaking the overall project into a set of separate phases reduces the breadth of what the developers need to keep in their awareness. They do not need to think about the entire project all at once. They can restrict their attention to the issues of the current phase. For example, the systems and then requirements analysis phases are done almost completely independently of any considerations of code implementation. The systems requirements phase analyzes the entire application environment and specifies which part of the application should be automated by the specified software system. Once that analysis is complete, then the software requirements analysis specifies the details of what the software system will do.

The design phase then works out the general software architecture and design for how the software will work. Traditionally, this is done
using a technique referred to as “stepwise refinement” in which the
design is developed in terms of interacting modules. Each module is
designed separately; this approach serves to restrict the breadth of issues
the developer needs to consider. He or she can attend to one module at
a time. Often a module will have internal component modules that are
not fully designed with that module, but rather are designed separately
at a later time. Again, this reduces the scope of what a developer needs
to consider at a given time.

Much of software engineering practice is concerned with the arti-
facts produced by each phase—e.g., specification documents from the
requirements phases, design documents from the design phase, etc.
These documents represent aids so that developers do not need to keep
all of this information in their awareness all the time. They can refer to
the documents as external sources of memory about issues that need to
be considered.

Most large modern software development projects have adopted the
object-oriented programming paradigm. Several software develop-
ment process models have been created for this programming para-
digm. Almost all of the object-oriented process models are “iterative”
development models. This means that instead of a single pass through
each phase as is done in the waterfall model, there are multiple itera-
tions through the waterfall phases. In effect, the project is broken into
a number of mini-waterfall projects (typically 3–10). This clearly has
the effect of reducing the scope or breadth of what needs to be consid-
ered during any particular iteration of the process, and hence is another
example of how software development methodologies assist the devel-
oper in dealing with the breadth of awareness requirement.

A recent development in software engineering methodologies are
the “agile development methodologies,” e.g., Extreme Programming,
Scrum, Dynamic Systems Development Method, Feature Driven
Development, Crystal Methods, Lean Development, Adaptive Soft-
ware Development, Dynamic Systems Development Method (Cohen,
et al., 2003) and (Agile software development).

The original agile methodology was the methodology of Extreme
Programming (Beck, 2000); see also (Wells, 2009). Extreme program-
mimg introduced a number of revolutionary concepts into the field of
software engineering methodologies. The driving motivation behind
Extreme Programming (XP) was to get away from what its originators perceived as an overemphasis on documentation, design, analysis, and other non-coding activities. They characterized other process methodologies as being “heavyweight” in the sense that they had come to focus too much time and effort on documentation and analysis activities, whereas the only result that mattered for a software project was to produce working code. The originators of XP considered themselves master coders, people among the top few percent of developers in terms of productivity in producing working code. Their objective was to take all the practices that had been found to be most effective in industry practice in quickly producing working code, and take those processes to their extreme value—hence the name “Extreme” Programming.

In the rest of this section we will consider the main features of XP and will see how they provide broad awareness of the overall system in support of the fine focus needed when implementing detailed code. XP is particularly interesting in this regard because it takes a radically different approach in its process methodology compared to more traditional development methodologies. Some of its methods are surprising in the sense that they appear to be directly in opposition to its own underlying mission of quickly creating code as fast as possible. However, we will see that all of the XP features, like almost all the more traditional processes, can be understood in terms of providing support for broad awareness along with fine focus of attention.

We will examine a dozen XP methods and then several process methods used by Microsoft developers. Each of these will have its own connection to the requirement for broad awareness and fine focus. After examining them we will categorize them into a smaller set of six types of techniques for supporting broad awareness and fine focus. These six types can be further reduced into two categories. The first category includes many techniques that reduce the requirement for broad awareness. The second category includes a smaller number of techniques that have the effect of increasing the breadth of awareness that can be brought to bear.

XP is defined by a set of twelve core processes (Jeffries, 2001). We will consider each of these processes and how they either reduce the need for broad awareness or increase the amount of awareness brought to bear.
Small Releases

XP teams put a simple system into production early, and update it frequently on a very short cycle.

Developing small pieces of functionality is a hallmark of all iterative development processes, which is a defining characteristic of almost all modern software development processes. Developing small units of functionality obviously reduces the breadth of awareness required by the developers as they are developing each individual component.

Of course, the different small units of functionality must eventually be combined into an overall complete system. This requires that somewhere in the project there must be a grand vision or plan or architecture that gives guidance for how all of the pieces will successfully fit together. Typically a very experienced member of the team is the system architect who supplies this grand vision or architecture for the system. In XP all members of the team are expected to function as system architects as well as developers of the fine details of coding. That is to say, XP expects all team members to have a broad awareness of the system design and requirements along with the ability to have a fine focus on detail in developing code.

Many small releases also enable stakeholders to constantly evaluate point values in relation to the entire system and its application. A small change in each release allows stakeholders (and programmers) to focus on just the new features in that release and how those features operate in the context of the entire system. If there are only minimal changes to the entire system this makes it easier to have a broad awareness of the entire system.

Simple Design

The philosophy of XP is to build the simplest system possible that meets the current application requirements. There is relatively little attention paid to creating general components to be directly reused in future projects. General, reusable components can require more complex designs and coding than application-specific components, and often the more general designs are not ever used. In XP, generalization of components is accomplished through “refactoring” as the need for reuse arises. This means that XP favors simpler designs that might be less general than
more complex ones, and then creates more general ones only when the demand is evident.

Simple systems make systems easier for developers to create in terms of the requirements of broad awareness and fine focus. Simple systems will be smaller and less abstract, and hence require less breadth of comprehension. Simple systems will also have less complicated, less intricate code, and hence the demands on fine focus of attention will be less.

Test-Driven Design
An important but counter-intuitive feature of XP is test-driven design. Test-driven design requires that developers write tests for components before they actually write the component. XP developers generally spend about half of their time writing these unit tests.

From a surface level analysis it seems quite surprising that a process developed to focus on writing and delivering working code as quickly as possible with minimal overhead would include as a critical step writing test code for up to half of the developer’s time. Test code is not used in the real application. Further, this activity takes place before the operational code is written. From a surface level analysis, this seems like a major waste of time and expense. However, test-driven design and unit testing is a practice that is being adopted widely in industry even outside XP because it has been found to be valuable in reducing overall development time and expense. XP proponents claim that it is completely in accord with the goal of quickly producing working code. They deliver working code more quickly by taking this time to write tests than if they did not write the tests.

There are many reasons why unit tests decrease programming time, including the fact that by far the most time-consuming activity in software development is debugging, and unit tests allow developers to almost eliminate this activity because a comprehensive set of unit tests quickly isolate most bugs.

Unit tests reduce requirements for broad awareness in several ways. First, tests are written with the specifications for the component in mind. The developer does not need to worry about the fine details of implementing the component when he or she writes the unit tests, only the inputs and expected results of the operations of the component. The unit tests then serve as executable specifications that guarantee the
component specifications are met by the actual implementation of the component. Without the unit tests it is easy for a developer to become lost in the details of coding the component and implement functionality that is different from the original intention. The unit tests guarantee that the original intention must be satisfied by the final implementation. This is a breadth of awareness feature.

Unit tests also serve to break the implementation process into smaller steps. Unit tests are generally very fine grained and require tests to be written to verify that every single operation in a class is working. Developers thus get immediate feedback as they implement each operation, and they must have each operation working before proceeding to the next operation. This eliminates complex bugs that arise where one operation does not work because there is a problem in some other operation on which it is dependent.

Thus test-driven design supports broad awareness in software development by providing an explicit enforcement of the system specifications on the detailed implementation. Test-driven design also breaks the implementation task into smaller steps, and thereby reduces the breadth of awareness requirement. The complete set of unit tests can also be written to insure that all of the individual subtasks work together properly to give the required overall functionality of the component, thus providing more support for that level of broad comprehension.

**Refactoring**

Refactoring is another important XP activity that is different from traditional development wisdom. Traditionally, most methodologies try to avoid changing working code if at all possible. In contrast, XP takes the view that even working code should be updated and improved throughout the entire development. This is an important feature in XP because of the underlying philosophy in XP that a software development process must allow requirements to be developed and modified throughout the development process. The XP philosophy is that requirements change is unavoidable and should be embraced rather than resisted.

Refactoring involves steps such as moving duplicated code into a common method, eliminating complex conditional logic in favor of polymorphism, breaking complex methods or classes into simpler ones if they become too large, etc.
Refactoring complex code to make it simpler and easier to understand supports breadth of awareness and fine focus by reducing the scope of awareness required to understand a piece of code, and by simplifying the level of complexity of the code. Simpler code is typically less coupled to other components, and hence a developer does not have to keep as many interactions between different components in his or her awareness. Simplifying complex logic or expressions means that even the fine focus of attention in developing a given component does not have to be as involved and intricate.

Those are ways that the refactoring process serves to reduce the breadth of awareness and detail of focus that is required in the development process. On the other hand, the refactoring process itself requires broad awareness and fine focus from the developer. Refactoring involves changing working code. It typically, by definition, involves changing complex pieces of code in order to simplify them. Such code is often brittle and probably not well written, which is why it needs to be refactored. Changing such code requires the developer to understand what is often complex and poorly written logic that might involve complex interactions with other components.

Pair Programming
Another highly innovative and surprising core practice in XP is pair programming. XP programmers write all production code in pairs, two programmers working together at one machine.

Pair programming is an exceptionally clear example of a process methodology that is designed to foster broad awareness and fine focus in software development. In pair programming one programmer is considered the “driver” and is typing code on the keyboard. The driver provides the fine focus of attention in developing the current component. The other programmer is considered the “navigator.” The navigator is responsible for providing broad awareness. The navigator keeps in mind how the current method or component being developed interacts with other components, and watches for any errors or details that the driver might overlook while focused on correctly typing the current expression, statement, branch, loop, etc.

Many people are surprised that pair programming can be cost effective since it has one (typically highly skilled) programmer always
watching another highly skilled programmer. However, “pair programming has been shown by many experiments to produce better software at similar or lower cost than programmers working alone” (Wells, 2009). This practice and the observed results are a strong example of the importance of breadth of awareness and fine focus of attention required for effective software development. Software development is a very challenging activity, and requires both broad awareness and fine focus of attention. The success of pair programming illustrates that these factors are so important that even skilled programmers can more than double their effectiveness by increasing the awareness given to writing code. Developing effective code is much more than simply typing instructions one after another.

**Planning**

XP has its own planning process, in which the customer specifies the value and importance of desired features, and developers provide time estimates for implementation.

Like any software development plan, the plan represents a global strategic context to direct which components should be developed when. The plan itself represents a broad awareness about the project. It enables developers to work on individual components without needing to be aware of the global issues that have been captured in the project plan. If this global strategic plan is not present in the project or in the awareness of the developers, then projects will quickly become disorganized and ineffective due to development of pieces of software that are not needed at a given time, or components that may not have other prerequisite components already developed, or components that duplicate or conflict with functionality being developed by other team members.

**Metaphor**

XP teams use a common “system of names” and a common system description that guides development and communication.
Having a common view and understanding of a system across a team of developers helps insure that everyone has a unified view of the entire system. With large systems containing hundreds of thousands or millions of lines of code, it is impossible for any one person to know the entire system in detail. Different team members will know the details of different parts of the system. If the entire team shares a common language and high level view of the system, this will allow them to make the entire system coherent by understanding and relating their individual components to the commonly shared view of the system.

**Continuous Integration**

XP teams integrate and build the software system multiple times per day.

Integrating more often means that the system development proceeds in smaller steps of progress. Smaller steps mean smaller pieces are changed, which means the requirements for refined focus and broad awareness are smaller for each of these individual steps. It is less likely that there are long chains of intricate logic to comprehend and less likely that there are many complex interactions between different parts of the system.

Requiring parts of the system to be continuously integrated with the whole of the system also promotes a broader comprehension of the system by individual developers. They are forced to continuously consider how their individual components work with the entire system. Having a completely integrated and hopefully a working system throughout development also promotes a global awareness of the entire system for each of the individual developers.

**40-Hour Week**

Tired programmers make more mistakes. XP teams do not work excessive overtime, keeping themselves fresh, healthy, and effective.
Experience throughout the software industry has shown that working harder and longer is not a good formula for developing working code quickly. This illustrates the importance of having broad awareness and an ability for fine focus in order to be a successful software developer. Developing a large software system is a very complex and demanding mental task. Having clear awareness is critical to avoiding mistakes, that is, avoiding bugs. Working shorter hours in a mistake-free manner is far more effective for producing working code quickly than working longer hours continuously without proper rest.

A developer can only have fine focus when she or he has clear awareness. A developer can only have clear and broad awareness if she or he is rested and fresh.

**Collective Ownership**

All the code belongs to all the programmers. Any programmer can change any code at any time.

Programmers can work on any part of a large system; this means that the programmer must have some breadth of awareness that can encompass and comprehend the entire system. Also, the fact that multiple programmers look at and work on the same pieces of code brings a breadth of awareness to the development process. Multiple programmers looking at a piece of code obviously means there is more awareness focused on that code. Since different programmers will be more familiar with different parts of the system than others, having these different programmers looking at a piece of code helps ensure that global awareness is brought to bear on this particular piece of code.

**On-site Customer**

An XP project is steered by a dedicated individual who is empowered to determine requirements, set priorities, and answer questions as the programmers have them.

The on-site customer represents the overall vision of the project. The customer provides a broad perspective of the entire system being developed as well as the application domain in which it will be used. The
on-site customer might be considered a sort of global-navigator counterpart of the pair-programming navigator.

**Coding Standards**

For a team to work effectively in pairs, and to share ownership of all the code, all the programmers need to write the code in the same way, with rules that make sure the code communicates clearly.

Coding standards promote better comprehension and shared awareness of the code across the team of developers. Coding standards are especially valuable in supporting the XP practices of collective ownership and pair programming, both of which support breadth of awareness and fine focus as discussed above.

Microsoft is the largest and most successful software company in the world. Although Microsoft has not widely adopted agile methodologies, many of their groups are experimenting with them. More significant is the fact that many of the XP practices are quite similar to longstanding practices developed at Microsoft. One widely used practice at Microsoft is that they have small teams of developers who work very closely, often one-on-one, with small teams of testers. Pair programming and test-driven development can be considered as extreme variants of this common practice of Microsoft. Having a tester working with a developer obviously increases the amount of human awareness focused on a component under development. Indeed, all of the pair programming and test-driven development effects on breadth of awareness and fine focus mentioned above should apply to this practice of testers working closely with developers throughout the development process.

Another best practice widely used at Microsoft is code reviewing. The entire development team (typically 3–8 developers and associated testers) conducts code reviews. Some Microsoft developers report that they spend up to half their time in code reviews. Reviewers share responsibility for any bugs in code with the developer of the code. Pair programming is sometimes described as a continuous real-time code review. Collective ownership of code can be thought of as taking the code-review-by-teams practice to its extreme value. Clearly, hav-
ing many developers reviewing code brings a lot of awareness and fine focus to all the reviewed code components.

We have now done a careful analysis of how many common best practices in software engineering methodologies serve to support the broad awareness and fine focus that is required for successful software development. This capability is a fundamental requirement for software development—part of the “essence” of software engineering described by Brooks. Below we organize the various techniques we have described into six different categories of techniques for supporting broad awareness and fine focus in software development practices.

<table>
<thead>
<tr>
<th>Techniques for supporting broad awareness and fine focus</th>
<th>Relevant XP practices</th>
</tr>
</thead>
</table>
| reducing scope, abstraction, intricacy                 | • do the simplest thing possible  
|                                                        | • continuous integration and small releases  
|                                                        | • refactoring           |
| breaking down scope, abstraction, intricacy into smaller pieces | • continuous integration  
|                                                        | • small releases        
|                                                        | • refactoring           |
| creating an external record that has the effect of reducing or breaking down scope of awareness | • requirements documents  
|                                                        | • on-site customer      
|                                                        | • test-driven design    
|                                                        | • planning              
|                                                        | • testers working closely with developers throughout the development process |
| increasing abstraction in well understood ways in order to remove details from immediate consideration | • test-driven design    
|                                                        | • refactoring           
|                                                        | • analysis and design techniques—design patterns |
| multiple shared views of same component               | • pair programming      
|                                                        | • reviews               
|                                                        | • collective ownership  
<p>|                                                        | • metaphor and coding standards |</p>
<table>
<thead>
<tr>
<th>Techniques for supporting broad awareness and fine focus</th>
<th>Relevant XP practices</th>
</tr>
</thead>
<tbody>
<tr>
<td>increasing ability of breadth of awareness and capability for focus of individual developer</td>
<td>• 40-hour work week</td>
</tr>
</tbody>
</table>

Note that the first four techniques could all be grouped together in the sense that they all have the effect of reducing the need for broad awareness. The last two techniques have the effect of increasing breadth of awareness. The first of these increases awareness being applied through group awareness. The second increases the awareness and ability to focus for an individual software engineer. This strategy of this last technique will be the focus of the second half of this paper.

Finally, we noted several aspects of XP that require broad awareness and fine focus in order for XP to work. These aspects included:

- There must be a grand vision of how all the small releases fit together into a complete system.
- Refactoring often requires a developer to understand complex and poorly written logic that might involve complex interactions with other components.
- The pair-programming navigator must keep in mind how the current method or component being developed interacts with other components.

The thesis of this paper is that programming is inherently difficult and requires both broad comprehension and fine focus on the part of programmers. Many studies and general industry experiences have shown that there is a great variability among the productivity of individual programmers (Sackman et al., 1968; McConnell, 2005). One analysis concluded,

Finally, programming demands the most of users and thus shows the highest variability. This [analysis] shows why the number one guideline for managing software development is to hire the best developers: Good developers are three times faster than slow ones and offer companies tremendous gain—even when they require higher salaries. The
difference between the very best and very worst developers is typically about a factor of twenty. (Nielsen, 2006)

This great level of variability is consistent with the claim that software development is especially difficult. There is a high variance because it is difficult and only the best can do it well, and weaker developers run into many difficulties. The claim of this paper is that good developers must be able to maintain broad awareness along with fine focus of attention. We have shown how almost every best practice of the software industry supports this requirement. In the following sections we will discuss a complementary approach to improving software development. In addition to using software development practices that provide awareness aids to developers, there are techniques that can directly increase developers’ ability to have broad awareness along with fine focus.

**Part II: The Transcendental Meditation and TM-Sidhi Programs Develop Broad Awareness and Fine Focus**

In Part II of this paper we describe how the foundations of Consciousness-Based education, the Transcendental Meditation and TM-Sidhi programs, directly develop this critical capacity in the software developer. We start with the idea that one of the greatest potential areas for increasing productivity of software engineering is to directly develop the awareness of the individual software engineers. We then define the concept of broad awareness and fine focus in terms of a model of levels of mind and levels of thought, including how the Transcendental Meditation and TM-Sidhi programs develop the ability of an individual to expand their awareness to encompass the full range of the mind. Next we describe how structuring deep principles into the awareness of an individual is critical to successful activity, and particularly how the Transcendental Meditation and TM-Sidhi programs structure an absolute level of knowledge into the awareness of an individual. Finally, we review relevant studies from a large body of scientific research on the effectiveness of the Transcendental Meditation and TM-Sidhi programs.

The Transcendental Meditation technique is a simple mental technique that can be practiced by anyone. The technique allows one’s mind
to settle down, and one’s physiology gains a state of deep rest. In this state of reduced mental activity and deep physical rest one maintains full awareness, even a sense of heightened mental wakefulness. One experiences finer states of thought along with expanded awareness. It is a completely natural process in that there is no intention or effort to achieve any particular experience or condition. People begin experiencing the benefits of the Transcendental Meditation technique immediately, and the effects increase and are cumulative over time.

Training in management requires the development of the ability to focus on one specific area of any project without losing the broad comprehension of the entire field of the project. The most successful procedure for the development of this quality in the manager is regular practice of Transcendental Meditation, which has demonstrated the growth of field independence [Pelletier, 1974]—the spontaneous ability to maintain broad comprehension while focusing sharply on any one area. (Maharishi Mahesh Yogi, 1995)

Developing the Developer

We noted above that there are great individual differences in the productivity of software developers—the best developers being as much as 20 times more productive than others. The obvious implication of this observation is that probably the largest single factor in software engineering success and productivity is the ability of the developer.

In his “Essence and Accident in Software Engineering” paper, Brooks (1987) gives proposals for the most promising avenues to advance the field in the future. One of his main proposals is to “create great designers.” He does not give a specific formula for how to do this, but his direction is to increase productivity through improving the human. This direction is the foundation of “Absolute Programming.”

The main practice of Absolute Programming is to create great developers by directly increasing their capacity to maintain broad awareness along with fine focus of attention. Absolute Programming provides a scientifically proven set of technologies for accomplishing this. The main technologies are the Transcendental Meditation technique and the TM-Sidhi program (The Transcendental Meditation Program [online]).

Guthrie (1997) examined the foundations of computer science and noted the critical role of the computer scientist:
While languages, methods, tools, and systems support the description of systems into programs, the source of this is the intellect of the programmer/analyst. It is the intellectual understanding of the original system, the goals of the program, and the methods and forms of programming languages that together form the knowledge that is the source of a program. It is this intellectual structure that is expressed in the structures and forms of the program. Therefore the most fundamental source of all computation is the [computer] scientist’s understanding, the appreciation and intellectual comprehension of some structure of natural law.

Dillbeck and Dilleck (1997) made a similar observation about the critical subjective role of the human in any scientific (or engineering) endeavor:

The limits of science are the limitations of the scientist. Being a human activity, its realms of explorations, its conclusions and its applications depend ultimately on the quality of humans doing the investigating— their alertness, intelligence, balance, comprehension, vision—their quality of consciousness. Maharishi describes that as scientists enlivening the holistic knowledge of natural law in their own awareness, their comprehension will become vast, their intuition flawless.

**Defining Broad Awareness and Fine Focus in Terms of Levels of the Mind**

An important concept underlying the Transcendental Meditation and TM-Sidhi programs is that just like there are levels of physical reality there are more and less refined levels of thought. In physics, everyone is familiar with the concept of levels of physical elements: chemicals, molecules, atoms, subatomic particles, and the physical fields that underlie particle physics. The finer levels are more abstract and more powerful. The amounts of energy available at atomic and subatomic levels of physics are far greater than those available at the molecular and physical levels, and at the ultimate level of physical reality, the unified field, the amount of energy in a cubic centimeter is greater than the entire mass-energy of the known universe (Hagelin, 1987).

There is also a sense in which intelligence is more concentrated at more fundamental levels of nature. Laws of nature dealing with molecular phenomena must be in accord with laws of nature dealing with atomic and subatomic phenomena. Laws of nature of more fundamen-
tal levels are the basis of the laws of nature of more expressed levels. The laws of nature of the unified field are the basis of all laws of nature. “If, as particle theorists are inclined to believe, all the laws of nature have their ultimate origin in the dynamics of the unified field, then the unified field must itself embody the total intelligence of nature’s functioning” (Hagelin, 1987, p. 58).

The Transcendental Meditation technique leads to the experience of finer levels of the mind, and ultimately to the experience of a state of awareness known as “pure awareness” or awareness of awareness itself—no thoughts, but just wakefulness and awareness of one’s Self. This level of awareness is called pure awareness or pure consciousness or Transcendental Consciousness. This is a unified state of consciousness parallel to the unified field of physics. It is an experience beyond space and time that underlies all other experiences and states of awareness, parallel to the unified field of physics being a field that is beyond space and time which underlies all states of matter in the universe. All matter and energy in the universe is the result of some excitation or vibration within this unmanifest unified field (Hagelin, 1987). All of the laws of nature are embodied in this unified field.

Figure 1 (below, from Dillbeck and Dillbeck, 1987) shows a further elaboration of this concept of levels of thought. It shows levels of mental activity in terms of knower, process of knowing, and known. The level of awareness of the knower determines the level of the process of knowing as well as the nature of the knowledge gained. Deeper levels of awareness allow one to have broader awareness along with fine focus of attention. The deepest level of awareness corresponds to Transcendental Consciousness, which corresponds to the direct experience of the unified field of physics (Hagelin, 1987; Maharishi Mahesh Yogi, 1985). The full range of mental activity ranges from the unbounded awareness of Transcendental Consciousness to the specific point values of sensory perception. Deeper levels of the mind—intellect, feeling, and Transcendental Consciousness—are successively quieter and less active states of awareness. One can have very fine focus of attention at these quiet levels and also a broad awareness of many considerations. Thus, access to deep levels of awareness provides one type of broad awareness and fine focus.
A more complete type of broad awareness and fine focus would be for an individual to have the full range of the mind open to his or her awareness. In such a state of awareness a person would maintain the unbounded and refined awareness of Transcendental Consciousness along with all other levels of mental activity. In such a state of awareness a person will have maximal ability to keep a broad range of considerations in mind even while engaged in dynamically focusing their senses, thoughts, and attention to specific details.

Dillbeck and Dillbeck (1987) discuss this model of levels of the mind and the value of broad awareness over the full range of the mind, and note the value of expanded awareness for any field of study:

The more expanded their awareness, the more easily they can comprehend fundamental principles, make profound connections, and bring a clear sense of values to whatever they study. (p. 417)
Structuring the Home of All the Laws of Nature in One’s Awareness

When people have an experience or learn some concept, they begin to structure their awareness according to that experience or understanding. For example, when programmers deeply understand the principles underlying process management and memory management in an operating system, then their understanding of how to program and debug a web application will be much deeper and more powerful than those who know nothing about processes and threads but have only learned to use some software package. We can say that the programmer who understands the underlying principles has structured his or her awareness or understanding in terms of those principles. The kinds of thoughts that such a programmer will have, for designing new programs or debugging existing ones, will be in accord with those underlying principles.

A basic principle of modern cognitive psychology is that our thoughts, memories, and perceptions are strongly influenced by our past experiences and our “mental model” of the world (Anderson, 1999; Gardner, 1987). That is, if a person has an underlying belief system or understanding of the underlying principles of a domain, then it is far more likely that person will have ideas and thoughts (and perceptions and memories) that are consistent with those basic principles than someone who does not have this understanding. This is why all good educational programs have a first priority to ensure students understand underlying principles, and that they understand more point-value details in the context of the underlying principles.

Consciousness-Based education and Absolute Programming take this principle to an absolute value. Absolute Programming seeks to structure the deepest level of nature into the awareness of the programmer, the level of absolute pure consciousness (Transcendental Consciousness), which is the experience of the unified field that underlies all of creation and embodies all the laws of nature. Consciousness-Based education provides both an intellectual understanding of this experience, and more importantly, a systematic and repeatable method for any individual to have this experience.

Through the practice of the Transcendental Meditation and TM-Sidhi programs an individual can develop this level of awareness to become the underlying frame of reference for all thoughts and actions.
As mentioned above this state of awareness corresponds to a direct experience of the unified field of all the laws of nature (Hagelin, 1987). At this level all thoughts and actions become consistent with the unified field of nature, similar to the sense in which a programmer who understands the underlying principles of computer science will have thoughts that are consistent with the underlying principles of computer science.

Consider the following analogy to help clarify this concept that thoughts and ideas can be structured according to a set of underlying principles when the awareness is functioning from a deep level. Imagine an architect who needs to think deeply about a large complex project and the implications of various systems interacting on some fine level. The architect would likely want to do this in a quiet setting where his or her mind is settled, but still awake and lively. This is a common thing for anyone who has some task requiring him or her to consider fine details along with broad comprehension over many considerations. In some quiet area the architect might look through a set of blueprints that represent the underlying “seed” information for the entire project. The blueprints are certainly not the finished building; however, there is a real sense in which the finished building is defined by the compact information of the blueprints. If the architect has a deep understanding of the blueprints, then his or her thoughts and actions and directions to the builders will be consistent with this underlying reality.

The Transcendental Meditation and TM-Sidhi programs give an individual the experience of pure awareness, the experience of the unified field, the source of all the laws of nature. Becoming familiar with this level of awareness is analogous to an architect becoming familiar with the blueprints of a building. The immense software systems being developed today might be enormously and inherently complex in some respects, but they are trivial in complexity and sophistication in comparison to the natural universe. The laws of nature automatically manage the entire universe, from its creation and dissolution to the ongoing operation of galaxies, planets, organisms, chemistry, molecules, atoms, particles, and matter and energy fields. There is an elegant orderliness in the laws of nature, which is captured by the intricate and elegant mathematics and logic of modern physics. This is the level of organizing power that is structured into one’s awareness through the Tran-
scendental Meditation and TM-Sidhi programs. The ability to think clearly and logically and organize a wide array of concerns is critical to being a successful software developer, and is increasingly important as one becomes more senior in responsibilities.

The Unified Field is the field of all knowledge in seed form. All the Laws of Nature are absolutely vital on that level. It is the source of all organizing power, the source of all streams of power. Once human awareness is open to the field of all knowledge in the Unified Field, then human awareness is lively in all the organizing power of Nature. (Maharishi Mahesh Yogi, quoted in Dillbeck and Dillbeck, 1987, p. 392).

**Effects of the Transcendental Meditation Program in Developing Broad Awareness and Fine Focus Are Immediate and Cumulative**

In the Scientific Validation section below we reference many unique physiological changes that occur during the experience of Transcendental Consciousness. These changes are consistent with what one should expect as indicators supporting broad awareness and fine focus. Indicators such as restfulness and global EEG coherence along with wakefulness should promote an ability to have many things accessible to one’s awareness at a very quiet level. In addition to the objective physiological studies, the subjective experience of Transcendental Consciousness also supports the claim that one gains in the ability to have broad comprehension along with fine focus on detail. The subjective experience is one of inner calm, expanded awareness, enhanced wakefulness and the ability to have thoughts at a very quiet and refined level. Physiological research has found that the physiological measures of Transcendental Consciousness in new meditators are very similar to those of long-term meditators during meditation. This is not surprising since the TM technique is a very simple mental technique that can be quickly learned by anyone. In addition, it is common for new meditators to quickly notice effects in activity. The following are a few anecdotal comments from students in the Masters of Computer Science program at Maharishi University of Management regarding the effects of their having recently learned the TM program and its effect in their jobs as professional software developers:
• “TM helps me keep focus.”
• “I had a job change this month. As usual TM helped me to stay alert and active amid turbulence.”
• “A lot of benefits, first I come early in the morning and feel fresh the whole day. I do not feel frustrated on any problem and because of this I am able to resolve the problem.”
• “Staying calm under pressure, where my non-meditating colleagues were panicking.”

An important physiological difference develops as individuals have been practicing the Transcendental Meditation and TM-Sidhi programs over longer periods of time. A style of functioning develops where the mental and physiological characteristics of Transcendental Consciousness become integrated with the waking and sleeping states of consciousness. Below is a typical description of the subjective experience:

The flurry of waking activity comes and goes; the inertia of sleep comes and goes. Yet, throughout these changing values of waking and sleeping, there is a silent, unbounded continuum of awareness that is me; I am never lost to myself. (Travis et al. 2002)

Physiological studies have found that individuals who report these experiences of pure consciousness along with waking and dreaming states maintain the unique brainwave indicators of pure consciousness along with brainwave activity normally seen in waking and dreaming states of consciousness. Thus, new meditators quickly show similar physiological indicators of pure awareness as long-term meditators during meditation. However, studies show that over time with continued practice these physiological measures of global brainwave coherence become significantly more pronounced outside of meditation, even in the midst of daily activities (Mason et al., 1997; Travis et al., 2002). This indicates that there should be a significant increase over time in the ability to maintain broad awareness along with fine focus of attention even during dynamic activity. This is broad awareness and fine focus encompassing the full range of the mind as we discussed in the previous section.
**TM-Sidhi Program Directly Develops Ability to Think and Act with Expanded Awareness**

The TM-Sidhi program is a set of advanced techniques for individuals practicing the TM technique. These techniques involve a practice known as *Sanyama*, in which an individual practices having a specific set of thoughts or intentions at a very fine level of awareness. One Sidhi that has been widely discussed is the intention to levitate. This TM-Sidhi is known as Yogic Flying® (The Transcendental Meditation Program [online]), (Pearson, 2008). The process of *Sanyama* requires that one is familiar with the quiet state of Transcendental Consciousness and is able to introduce a thought or intention at this very fine level of awareness. This fine level of awareness, Transcendental Consciousness, is the experience of the finest level of nature, the unified field of modern physics. This fine level of awareness, the experience of the unified field or the source of all the laws of nature, is a level of infinite potential and unbounded broad awareness. Thoughts or intentions grounded at this level will be very powerful and in accord with all the laws of nature.

Thus, this practice of *Sanyama* is an advanced program of mental development that trains the individual to have maximally broad (unbounded) awareness and to simultaneously be able to maintain a very fine level of focused attention at the same time. This is exactly the ability that is fundamental to being successful as a software engineer—broad awareness along with fine focus of attention.

**Scientific Validation of the Transcendental Meditation and TM-Sidhi Programs**

The effectiveness of the Transcendental Meditation and TM-Sidhi programs has been thoroughly validated by over 600 published studies conducted at over 250 universities and research institutions in 30 countries (Orme-Johnson and Farrow, 1977; Chalmers, Clements, Schenkluhn, and Weinless, 1989a, b, c; Wallace, Orme-Johnson, and Dillbeck, 1990) and the experience of millions of people (The Transcendental Meditation Program [online]).

The following are a few of the scientific studies demonstrating that practice of the Transcendental Meditation technique produces deep rest and relaxation:
• skin conductance response and heart rate deceleration at the onset of breath changes during a Transcendental Meditation session mark the transition of awareness from active thinking processes to the mental quiescence of Transcendental Consciousness (Travis & Wallace 1997).
• reduced basal skin conductance (increased skin resistance), and reduced plasma lactate (Dillbeck & Orme-Johnson, 1987).
• a natural decline in respiration rate during the practice, indicating a state of deep rest, in contrast to two groups of controls (Gallois, 1984).
• a natural decrease in the volume of respiration during the practice, in contrast to controls (Wolkove et al., 1984).
• increased skin resistance during the practice, in contrast to controls. Skin resistance is an electrophysiological measure of calmness or restfulness (O’Halloran, 1985).
• increased muscle relaxation during the practice, in comparison to controls (Kemmerling, 1978).
• decreased blood lactate during the practice, in comparison to controls. High levels of lactate in the blood have been associated with anxiety and high blood pressure, and decreased blood lactate is a physiological indicator of relaxation (Jevning, 1978).
• decreased plasma levels of cortisol during the practice, in contrast to controls. Increased plasma cortisol is a response to stress, and decreased plasma cortisol thus indicates reduced stress (Jevning, 1978).

The following studies demonstrated that practice of the Transcendental Meditation technique not only produces a state of deep rest and relaxation, but at the same time it is a state of wakefulness, restful alertness.

• increased cerebral blood flow during the practice, in contrast to controls, both in the occipital (back) area of the brain and also in the frontal area (Jevning, 1996).
• displayed electroencephalographic (EEG) indicators of increased wakefulness during the practice, in contrast to controls (Banquet, 1974).
• after two weeks of practice, individuals showed significantly increased EEG coherence during the practice. EEG coherence across many different areas of the brain indicates a global coordination of brain functioning, which is associated with the experience of broad awareness (Dillbeck, Bronson, 1981).

• during the Transcendental Meditation technique individuals often report the subjective experience of Transcendental Consciousness or pure consciousness, the state of least excitation of consciousness. This study found that many experiences of pure consciousness were associated with periods of natural respiratory suspension, and that during these respiratory suspension periods individuals displayed higher mean EEG coherence over all frequencies and brain areas, in contrast to control periods where subjects voluntarily held their breath (Badawi, Wallace, Orme-Johnson, & Rouzeré, 1984).

• increased global EEG coherence during the practice of the Transcendental Meditation technique in comparison to an eyes-closed control period before the practice. Controls who had been randomly assigned to a control technique showed no significant change (Gaylord, Orme-Johnson, & Travis, 1989).

• “During Transcendental Meditation practice, experiences of ‘unboundedness’ and the ‘loss of time, space, and body sense’ (Travis & Pearson, 2000) were associated with spontaneous breath quiescence (breath periods from 10 to 40s) (Badawi et al., 1984; Farrow & Hebert, 1982), with autonomic orienting at the onset of breath changes (Travis and Wallace, 1997). These breath changes occurred on the background of high EEG coherence, which rises to high levels in the first minute of TM practice (Travis and Wallace, 1999)” (Travis, Tecce, Arenander, Wallace, 2002).

The above studies provide evidence that this simple mental technique has profound physiological effects, producing a state of deep physiological rest and relaxation along with a state of heightened mental awareness and acuity. These effects result in increased mental capacity and an increased ability to have fine focus along with broad awareness. The following studies document positive effects of the TM technique on performance of tasks that are associated with the ability to be a
good software developer—e.g., academic performance, performance on intelligence tests, and field independence.

- broader comprehension with increasing ability to focus. (Pelle-tier, 1974)
- master degree engineering students who learned the Maharishi Transcendental Meditation program showed improved performance on their standard examinations after six months, compared with randomly assigned controls. (Kember, 1985)
- Cambodian students taking a year of remedial study in preparation for college who learned the Transcendental Meditation program showed a significant increase in intelligence test scores in contrast to other students after three months. (Fergusson, 1995)
- within one school year, elementary school students who practiced the Transcendental Meditation program showed significant gains on a nationally standardized test of basic skills. (Nidich, Nidich, Rainforth, 1986; Nidich & Nidich, 1989)

Large software projects are always done by teams of developers who must work closely and communicate effectively with each other as well as with the stakeholders in the project. Studies on the Transcendental Meditation and TM-Sidhi programs have shown improved job performance, increased job satisfaction, improved work and personal relationships, and decreased tension on the job.

- a study of executives and workers in the automotive industry found that after three months of regular practice of the Transcendental Meditation program, employees showed improved work and personal relationships, in comparison to controls from the same work sites. (Alexander et al., 1993)
- employees who learned the Maharishi Transcendental Meditation program showed improved relationships with their supervisors and with coworkers in comparison to control subjects. (Frew, 1974)
- a study conducted at Sumitomo Heavy Industries by the Japanese National Institute of Industrial Health found that workers who learned the Transcendental Meditation program fell
asleep more easily at night, in comparison to control workers. (Haratani & Hemmi, 1990)
• employees who learned the Transcendental Meditation program showed, over a four-month study period, significant increase in a composite scale of leadership behavior, in contrast to control employees who did not participate in the program. (McCollum, 1999)

Although the above studies did not deal directly with software developers, the physiological results should generalize to any population, and the mental task performance results are consistent with the hypothesized increase in the capacity for an individual to maintain broad awareness along with fine focus. The Transcendental Meditation and TM-Sidhi programs develop this capacity in two closely related modalities—mind and body. Physiologically, as the above studies have shown, and it is the obvious experience during the practice, one’s physiology becomes very rested and relaxed, which allows the natural mechanisms of the body to restore and rejuvenate both the mind and body. In addition, this natural and automatic practice allows mental activity to settle down in a manner that awareness becomes expanded at the same time that refined mental activity is maintained. This naturally establishes a habit of one’s awareness being able to have broad awareness along with fine focus.

Conclusion
Even new meditators experience the inner calm of Transcendental Consciousness in their meditations and begin to feel the benefits in daily activity. This promotes the ability to have broad awareness along with fine focus of attention. The benefits and effects are immediate, but over time they are cumulative and grow to a point where this unbounded awareness can be maintained at all times and in the midst of even dynamic and intense activity. That is, the ability to maintain broad awareness along with fine focus of attention is developed in the individual. This ability is fundamental to effective software development and this paper has shown that almost all software engineering methodologies can be seen as supporting this ability. In particular, the paper has looked at the practices of the software process methodol-
ogy of Extreme Programming and how those practices support broad comprehension along with fine focus. Absolute Programming is the addition of techniques such as the Transcendental Meditation and TM-Sidhi programs to standard software development methodologies in order to directly develop this ability on the level of the individual software developer. Absolute Programming operates on the level of the programmer’s own awareness, to create master programmers, to create those programmers who have orders of magnitude greater productivity than average programmers.

References


Meditation and TM-Sidhi program: Collected papers, 3. Vlodrop, the Netherlands: MVU Press.


Gallois, P. Modifications neurophysiologiques et respiratoires lors de la pratique des techniques de relaxation. L’Encéphale, 10, 139–144.


Part IV

Principles
of Maharishi Vedic Science
in the Theory of Algorithms
Computable Functions in the Theory of Algorithms:
Collapsing Infinity to a Point

Paul Corazza, Ph.D.
ABOUT THE AUTHOR

Paul Corazza, Ph.D., received his Bachelor of Arts degree in Western Philosophy from Maharishi International University in 1978 and his M.S. and Ph.D. degrees in Mathematics from Auburn University in 1981 and 1988, respectively. He was awarded a Van Vleck Assistant Professorship at University of Wisconsin for the years 1987–1990. He worked in the Mathematics Department at Maharishi International University in the years 1990–95. Following a career as a software engineer, he rejoined faculty at Maharishi University of Management in 2004 and currently serves a joint appointment in the Departments of Mathematics and Computer Science. Dr. Corazza has published more than a dozen papers in Set Theory, focused primarily on the quest for providing an axiomatic foundation for large cardinals based on a paradigm derived from Maharishi Vedic Science.
ABSTRACT

According to Maharishi Vedic Science, creation arises in the collapse of the infinite unbounded value of wholeness to a point, from which creation emerges in a sequential unfoldment. By analogy, the creative dynamics found in the field of computation arise from the collapse of a vast function field—whose elements are almost entirely indescribable—to a tiny packet of functions that are actually used: the polynomial time bounded functions. The class of functions that are actually used in Computer Science arises from a much vaster class of number-theoretic functions through a sequence of “collapses” to successively narrower function classes.

The vast majority of number-theoretic functions are essentially indescribable. By contrast, it is possible to know in full detail the behavior of the functions belonging to the much more restricted class of definable functions. Restricting further to the class of computable functions, it becomes possible to describe the behavior of each function in terms of a computer program; that is, one can describe how to compute each function in this class. And the narrowest class of functions—the class of polynomial time bounded functions—consists of functions that can be computed in a feasible way, efficiently enough to be of practical value in real applications. We discuss how this sequence of collapses parallels the creative dynamics of pure consciousness itself, as described by Maharishi Vedic Science, as it brings forth creation through the collapse of the abstract, indescribable infinite value of wholeness to the concrete, specific point value within its nature.

Introduction

According to Maharishi Vedic Science, all that we see in this manifest world of existence arises because of the transformational dynamics of a single field, the field of pure consciousness, the unified field of all the laws of nature (see Maharishi, 1996, pp. 252, 504). These creative dynamics arise from a fundamental movement within pure consciousness, which repeats over and over again: the infinite expanded value of this field collapses to its own point value, and then emerges into infinite diversity. What we see in the form of sequential unfoldment of life in the manifest world is an expression of unseen self-referral dynamics at the level of pure consciousness. In this introduction to Algorithms, we examine how core notions in the field of computer science arise as a kind of collapse of expanded, noncom-
putable mathematical domains to a highly restricted domain in which the usual forms of computation and computational analysis become possible.

The starting point for our investigation into these various classes of functions is the simple idea of a computer program. A computer program can be understood as a procedure for taking certain inputs—perhaps numbers, character strings, or other types—and transforming them in a sequential manner to produce well-defined output values. This simple idea is the core notion at the root of all of today’s complex and remarkable software.

This idea of a computer program is a modern-day way of expressing the mathematical notion of a function. A function, like a computer program, acts on certain inputs and outputs well-defined values. A simple example is the function that acts on the natural numbers 0, 1, 2, . . . according to the following rule: The function transforms any natural number input to the natural number that is twice as big as the input. Formally, we express this rule by the formula \( f(n) = 2 \times n \). Thus, for example, \( f(3) = 2 \times 3 = 6 \) and \( f(7) = 2 \times 7 = 14 \) — the function transforms each input to an output value twice as big.

As we will discuss in more detail later, it is not true that every function defined on the natural numbers can be computed using a computer program. This insight leads to a remarkable discovery: the field of all functions on the natural numbers is a kind of wholeness, a kind of vast field, whose contents are fundamentally indescribable. To emphasize the point, we can say that, if one were to place all natural number functions in a bucket, and choose one from the bucket at random, the probability that this function’s behavior could actually be represented in a computer program is 0! That is, there is zero probability that, for a function chosen at random from the bucket, a computer program could produce exactly the same outputs as the given function on given inputs.

As we will show more formally, the functions that can be computed using computer programs represent an infinitesimal speck in the vast wholeness of all number-theoretic functions.

This phenomenon provides a striking analogy to the parallel journey from the field of manifest existence that we experience with the senses back to its source in the infinitely expanded value of intelligence itself. This infinitely expanded value is truly beyond description,
indeed, beyond the intellect itself. In the Bhagavad-Gita, Lord Krishna describes this field with the words (Maharishi, 1969),

The senses, they say, are subtle; more subtle than the senses is mind; yet finer than mind is intellect; that which is beyond even the intellect is he.

–Bhagavad-Gita, 3.42

In this paper, we describe the stages through which this fundamentally indescribable field of all functions as if “collapses” to the infinitesimal package of functions that we actually use in the field of computer science and in the context of software development.

We will describe the mechanics of singling out successively smaller collections of functions until we arrive at the class of functions that will be our primary concern—the class of polynomial time computable functions.

We will begin by discussing the class of definable functions—those functions that can be specified at all in a reasonable way. The class of definable functions can be said to emerge from the class of all number-theoretic functions in a way analogous to the emergence of the notion of “two” from “one”—existence and intelligence from oneness of consciousness as Samhita—as described in Maharishi’s Vedic Mathematics (Maharishi, 1972, Lesson 8).¹ Each function can be viewed as a collection of ordered pairs, but when a function is definable, this means that there is, associated with the “existence” of the function as a set of ordered pairs, a formula that embodies the intelligence that underlies the operation of that function; the formula defines the function.

We will then discuss the computable functions—those functions whose input/output behavior can be captured by an algorithm or computer program. This class of functions is distinct from the class of definable functions in the sense that these functions are in fact derivable; by way of sequential steps of logic, each function in this class, with the unique features that characterize each, can be derived. By analogy, this quality of emergence by way of sequential unfoldment is characteristic of “three” emerging from “two” in Maharishi’s Vedic Mathematics. In the togetherness of intelligence and existence, intelligence becomes

¹The emergence of two from one is also expressed elsewhere by Maharishi (Nader, 1995, p.33) in terms of the sprouting of infinite silence and infinite dynamism from the singularity, and elsewhere still as the emergence of unity and diversity (Maharishi, 1996, p. 345).
intelligent, becomes aware of its existence, by the very nature of intelligence as pure awareness.

In other words, the togetherness of intelligence and existence results in a flow of intelligence. This flow is the flow of the knower toward the known, and introduces a third element: the process of knowing. In Sanskrit, knower, process of knowing, and object of knowledge have the names Rishi, Devata, and Chhandas, respectively (Maharishi, 1992, pp. 20–21). Being aspects of pure awareness, each of these is capable of being aware of any of the others; Rishi sees Devata and Chhandas; Devata sees Rishi and Chhandas; and so forth. In the act of knowing, the knower affects the known, and so, for example, the Chhandas that is seen by Rishi is now a slightly modified value of Chhandas, which in turn sees each of the other values, and in turn has an effect on each of them. In this way, all possible combinations and flavors of pure awareness sequentially unfold and give rise to all possible expressions. It is therefore at the level of the “three”—Rishi, Devata, Chhandas—that sequential unfoldment of the universe begins to take place. And, by analogy, we find that the distinguishing feature of the class of computable functions is their derivability in a sequence of logical steps or computations. Moreover, we will see that, using one of several formalizations of computable functions, all such functions can be shown to be derivable from just three.

Finally, we will discuss the class of polynomial time computable functions—those computable functions that can be implemented in a feasible way.
The concept of a function in this course will be intimately related to certain kinds of problems that we may wish to solve. Our analysis will show that certain kinds of problems are inherently unsolvable; others, though solvable in principle, are not known to be solvable in a feasible way; while still others are solvable, with solutions that can be implemented practically.

The course will focus primarily on the techniques for solving the problems that have feasible solutions, and demonstrating the degree of feasibility of the solutions. However, we will also discuss the applied value of problems with no known feasible solutions—particularly in the case of cryptography.

**Definability vs. Computability**

Recall that \( N = N^1 \) is the set \( \{ 0, 1, 2, \ldots \} \) of natural numbers, \( N \times N = N^2 \) is the set \( \{ (m, n) \mid m, n \text{ are natural numbers} \} \), and for each \( k \geq 2 \), \( N^k \) is the set \( \{ (a_1, a_2, \ldots, a_k) \mid a_1, a_2, \ldots, a_k \text{ are natural numbers} \} \) of all \( k \)-tuples of natural numbers. Let \( A \) be a subset of \( N^k \).

A function \( f: A \to N \) is an assignment of a unique natural number \( b \) to each \( k \)-tuple \( (a_1, \ldots, a_k) \) in \( A \); we write \( f(a_1, \ldots, a_k) = b \). Such an \( f \) is called a partial function \( N^k \to N \). If \( A = N^k \), \( f \) is called a total function \( N^k \to N \). Intuitively, if we can give a description of the input/output behavior of \( f \) using just natural number concepts, then \( f \) is said to be definable (we give a more precise definition below).

As a final warm-up definition, let us say that, given natural numbers \( a, b \), \( a \) divides \( b \), and write \( a \mid b \), if there is a number \( c \) such that \( b = ac \) (in other words, \( b \) is divisible by \( a \) with no remainder).

To a large extent, the significance of definability of a function is that it informs us, in a mathematical way, that it exists. This “mathematical way” begins with a mathematical formula; this formula represents the fundamental intelligence underlying the behavior of that function. A proof of the existence of the function depends critically on the presence of such a formula; the formula gives a precise description of how the function behaves, and without this information, there would be no way to provide a proof that such a function exists. Therefore, although there are in reality a vast number of number-theoretic functions, the collection of those that we can actually “get our hands on” is much narrower, and these are called the definable functions.
Interestingly, the definability of a function, though it informs us of its precise input/output behavior and is the basis for a proof of the existence of the function, does not necessarily provide us with a *procedure* for computing the values of the function; definability informs us about what the function is and does, but does not tell us *how* it does it (or how its behavior could be simulated computationally).

To illustrate the distinction between our ability to demonstrate the existence of a function using a defining formula on the one hand and being able to compute the values of the function on the other, we consider the following assertion:

**Assertion.** There is a partial function $\text{gcd} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, which, on input $(m, n)$, outputs the greatest number $d$ that divides both $m$ and $n$.

The $\text{gcd}$ function is the function that, on input $m, n$, outputs the greatest common divisor of $m$ and $n$. We give a proof of the existence of this function without giving any idea about how to compute its values. As we will see, the proof relies entirely on the definition of the function.

**Theorem.** Let $A = \{ (m, n) \mid \text{either } m \text{ or } n \text{ is not } 0 \}$. Then there is a function $\text{gcd} : A \to \mathbb{N}$ such that, on input $(m, n)$, $\text{gcd}$ outputs the greatest integer $d$ that divides both $m$ and $n$.

**Proof.** Given natural numbers $m, n$, not both 0, it is clear that 1 divides them both. Since at least one of them is not 0, it is also clear that there are fewer than $m + n$ divisors of both $m$ and $n$. So there must be a largest divisor. □

The proof of the theorem establishes that the partial $\text{gcd}$ function exists, but does not tell how to go about computing $\text{gcd}(m, n)$ for particular values of $m$ and $n$. As we will illustrate in a moment, the proof is founded on the precise definition of the function in question, and this definition can be represented as a formula. In fact, a formula $\psi(m, n, g)$ can be given informally here to illustrate the point:

$\psi(m, n, g) : g \text{ divides } m \text{ and } n \text{ and, if } h \text{ also divides } m \text{ and } n, \text{ then } h \leq g$. 

248
The formula $\psi$ defines the $\text{gcd}$ function in the sense that for any natural numbers $m$, $n$, $g$, $\text{gcd}(m, n) = g$ if and only if $\psi(m, n, g)$ is true. Notice that $\psi$ tells us the characteristic and defining features of $\text{gcd}$ without telling us how to compute values procedurally. For instance, it doesn’t tell us how to compute the $\text{gcd}$ of 2342 and 54.

For some theoretical purposes in mathematics, just knowing that a certain function exists is all that is necessary. Usually though, it is desirable to come up with a procedure for computing the values of a function, once it has been demonstrated to be definable. “Coming up with a procedure” is another way of saying “finding an algorithm” that computes the function.

Typically, once one has a (definable) function in hand, if the input/output values of the function can indeed be computed at all, there will be many different algorithms that can be used for this purpose. We show here two different ways to compute values of the $\text{gcd}$ function. The first approach is naturally called a brute force method that one might think of based on the existence proof just presented.

```java
public static int gcd1(int m, int n) {
    if(m==0 && n==0) {
        while(true) { ; } //infinite loop
    }
    int gcd = 1;
    for(int i = 1; i < m + n; ++i) {
        if(divides(i,m) && divides(i,n)){
            gcd = i;
        }
    }
    return gcd;
}

public static boolean divides(int i, int n){
    return n % i == 0;
}
```

It turns out there is a much more efficient way of computing $\text{gcd}$:
public static void gcd2(int m, int n) {
    if(m==0 && n==0) {
        while(true){ ; } //infinite loop
    }
    return computeGcd(m,n);
}
public static int computeGcd(int m, int n) {
    if(n==0) return m;
    return computeGcd(n, m % n);
}

As an example of the increased efficiency of the second approach, on a Windows XP laptop, computing gcd(123055509, 222535599) with the first algorithm requires more than 10 seconds, but with the second algorithm, less than a millisecond is necessary. This example motivates one of the main objectives of the course: to learn techniques for efficient algorithm design and for analyzing algorithm efficiency.

As a final observation about this example, we remark that the gcd function is indeed partial in the sense that it is not defined on all possible pairs \((m, n)\) belonging to \(\mathbb{N} \times \mathbb{N}\). In particular, gcd is not defined on the input \((0, 0)\). This is the reason why, in both programs above, we have put the function into an infinite loop when the input is \((0, 0)\); this is the conventional way of indicating, in a programmatic way, that the input value does not belong to the domain of the function.

**The Standard Model of Arithmetic**

In order to make the concept of definability more precise, we take a short detour and define the standard model of arithmetic. As we will see, when we say that a function is definable, what we actually mean in this paper\(^2\) is that it is definable in the *standard model of arithmetic*. Toward this goal, we give a brief introduction to the first-order logic of

---

\(^2\) There are several ways to make the notion of definability precise. The way we discuss here is standard in computability theory, but in set theory a broader definition is used; in that context, one would say that a number-theoretic function is definable if it belongs to the constructible universe (again, there are other variants). We mention this here because our computation of the size of the set of definable functions depends on our definition of “definable.” See the second footnote about this point later in the paper.
arithmetic (see Keisler and Robbins, 1996, or Enderton, 1972, for more on this topic).

We begin with a language—the language of arithmetic—that provides special symbols for the binary operations of addition and multiplication, the unary successor function, and the constants 0 and 1. In particular:

\[ \mathcal{L} = \{ s, +, *, 0, 1 \} \]

Our language \( \mathcal{L} \) consists of formal symbols. The symbols are chosen to suggest a particular intended interpretation. For instance the symbol ‘+’ suggests addition, even though, as a symbol belonging to \( \mathcal{L} \), it has not yet been interpreted as addition (and could actually be interpreted to be virtually any operation).

Another part of the language is an infinite set of variables. Formally, these variables can be listed as \( v_1, v_2, v_3, \ldots \), but for our purposes, it will be good enough to use boldface letters at the end of the alphabet, such as \( x, y, z \).

We will now give rules for combining the various symbols (language elements and variables) to form terms, and then introduce some additional logical symbols that will allow us to create formulas. Once we have these terms and formulas, we will describe how they will be given meaning by interpreting them in the standard model of arithmetic. All this effort will then allow us to state precisely, in the next section, what it means for a function to be definable.

We give the following recursive definition of terms: A term \( t \) is one of the following:

1. one of the constant symbols 0, 1;
2. a variable;
3. \( s(u) \), where \( u \) is another term;
4. either \( (u + v) \) or \( (u * v) \), where \( u \) and \( v \) are other terms.

This recursion is legitimate because, in the recursive clauses (3) and (4), the term \( t \) must always end up being longer than any of its component terms.

Here are some examples of terms:
We now build up formulas. We begin with atomic formulas. An atomic formula is an expression of the form \((t = u)\) where \(t, u\) are terms. Here is an example of an atomic formula:

\[(s(s(0)) = (1 + 1)).\]

We introduce the following symbols from mathematical logic: The symbol ‘∧’ stands for ‘and’; the symbol ‘∨’ stands for ‘or’; the symbol ‘→’ stands for ‘implies’; the symbol ‘¬’ stands for ‘not’; ‘∀’ stands for ‘for each’; and ‘∃’ stands for ‘there exists.’ ∀ and ∃ are called quantifiers. Starting from atomic formulas, we build up all formulas recursively as follows:

A formula \(ψ\) is one of the following:

1. an atomic formula;
2. \(¬ψ\), where \(ψ\) is a formula;
3. \((ψ ∧ θ), (ψ ∨ θ), \) or \((ψ → θ)\), where \(ψ, θ\) are formulas;
4. ∀\(x\)ψ or ∃\(x\)ψ, where \(ψ\) is a formula and \(x\) is a variable.

The recursive steps (2)–(4) are legitimate because new formulas are longer than the formulas from which they are built. Here are some examples of formulas:

\[((1 + 0) = 1)\]
\[¬(s(1) = 0)\]
\[((0 = 1) \lor ¬(0 = 1))\]

Now we give meaning to terms and formulas by interpreting the symbols we have defined so far in a model. The standard model of arithmetic, denoted \(\mathcal{N}\), consists of a base set \(N\) (the set of natural num-
bers 0, 1, 2, . . . ), together with the usual operations of addition (+), multiplication (∗), and a successor function \( s : \mathbb{N} \rightarrow \mathbb{N} \) defined by \( s(n) = n + 1 \). We think of the addition operation as an interpretation of the addition symbol + defined above. Likewise, ordinary multiplication is an interpretation of ∗; the successor function is an interpretation of \( s \).

And we will consider the two ordinary natural numbers 0 and 1 to be interpretations of the symbols 0 and 1, respectively.

More formally, we write:

\[
\begin{align*}
0^\mathbb{N} &= 0 \\
1^\mathbb{N} &= 1 \\
+^\mathbb{N} &= + \\
\ast^\mathbb{N} &= \ast \\
s^\mathbb{N} &= s.
\end{align*}
\]

Given a term \( t \), we give an interpretation \( t^\mathbb{N} \) of \( t \). We wish to define an interpretation in such a way that the term is evaluated to a number. However, if the term happens to contain a variable, we cannot hope to determine the value. For instance, the term \( s(x) \) cannot be evaluated uniquely to a single number. Recall that \( s \) is a symbol whose standard interpretation is the successor function. So, it follows that, once we know which number the variable \( x \) is supposed to stand for, we can evaluate the term as “1 plus the value of \( x \),” but not until then. So, as part of our interpretation scheme, we define how terms are to be evaluated when particular numbers are used to replace the variables.

A convention we observe is that if the variables in the term \( t \) are among \( x, y \), we will express \( t \) as \( t(x, y) \). (In the general case, we would use \( n \) variables instead of two.)

We can now finally define how terms are to be interpreted in \( \mathbb{N} \). Given a term \( t(x, y) \), we seek to interpret \( t \) as a number, but, as we just observed, can do so only relative to an interpretation of the variables \( x, y \) as particular numbers. Thus we formally define the interpretation of \( t(x, y) \) at numbers \( a, b \) in \( \mathbb{N} \), denoted \( t^\mathbb{N}[a, b] \), by the following recursive clauses:
1. if $t(x, y)$ is $x$, $t^N[a, b] = x^N[a] = a$
2. if $t(x, y)$ is $0$, $t^N[a, b] = 0$
3. if $t(x, y)$ is $1$, $t^N[a, b] = 1$
4. if $t(x, y)$ is $u(x, y) + v(x, y)$, then $t^N[a, b] = u^N[a, b] + v^N[a, b]$
5. if $t(x, y)$ is $u(x, y) \cdot v(x, y)$, then $t^N[a, b] = u^N[a, b] \cdot v^N[a, b]$

(Variables suppressed)

This interpretation of terms defines a function $\{\text{terms}\} \rightarrow \mathbb{N}$ by means of evaluation. For example, interpreting the term

$$(ss(1 + s(0)) \cdot s(x))$$

at $a = 1$ (so that the variable $x$ is replaced by the value $a = 1$) leads to the following steps, which result finally in a natural number:

- $(s(s(1 + s(0)))) \cdot s(x)^N[1]$
- $s(s(1 + s(0))) \cdot s(1)$
- $4 \cdot 2$
- $8$

Finally, we interpret formulas in the standard model $\mathcal{N}$. The result of this interpretation will be a boolean value—either true or false. We will explain this step somewhat informally, but accurately enough for our purposes. In order to perform the interpretation, we need to observe a distinction in the ways in which variables are used in formulas. To see the distinction, consider the following two formulas:

$$\phi : \forall x \ (x \cdot 1 = x)$$
$$\psi : \forall x \ (x \cdot y = x).$$

Notice that the variable $x$ in $\phi$ is “quantified” by $\forall$. Intuitively, $\phi$ says that, for every number, multiplying by 1 gives back the same number. Notice that the variable $x$ in $\psi$ is also “quantified” by $\forall$, but the variable $y$ is not associated with a quantifier. Intuitively, $\psi$ says that, for every number, multiplying by the (unspecified) number $y$ will give back the original number.

Based on our intuitive interpretations of $\phi$ and $\psi$, it seems clear that $\phi$ is true, but $\psi$ may or may not be true, depending on which number
the variable $y$ stands for. For instance, if $y$ is interpreted as the number 1, $\psi$ would be true, but if $y$ is interpreted as 2, $\psi$ would be false.

The formula $\phi$ is an example of a sentence—a sentence is a formula in which no variable occurs free. This means that each occurrence of each variable is tied to a quantifier. The formula $\psi$ is an example of a formula having a free variable $y$. In order to interpret formulas that have free variables, we will need to supply numbers to fill in for these variables, so that a final value of true or false can be obtained.

To facilitate this plan, we will adopt the following convention: To denote a formula $\theta$ whose free variables are among $x, y$, we write $\theta(x, y)$. If $z$ also occurs in $\theta$ but does not occur free, we do not list $z$ among the variables in the presentation of $\theta$.

Now we can now interpret a formula $\phi(x, y)$ in $\mathcal{N}$ as follows (to be formal, we should list $n$ free variables, but for readability, we just use two). The result of interpreting $\phi(x, y)$ in $\mathcal{N}$ at natural numbers $a, b$, denoted $\phi^\mathcal{N}[a, b]$, is defined recursively as follows:

1. if $\phi(x, y)$ is atomic, of the form $t(x, y) = u(x, y)$, then $\phi^\mathcal{N}[a, b]$ is true if and only if $t^\mathcal{N}[a, b] = u^\mathcal{N}[a, b]$;

2. if $\phi$ is $\lnot \psi$, then $\phi^\mathcal{N}$ is true if and only if $\psi^\mathcal{N}$ is false;

3. if $\phi$ is $\psi \land \theta$, then $\phi^\mathcal{N}$ is true if and only if both $\psi^\mathcal{N}$ and $\phi^\mathcal{N}$ are true;

4. if $\phi$ is $\psi \lor \theta$, then $\phi^\mathcal{N}$ is true if and only if at least one of $\psi^\mathcal{N}$ and $\phi^\mathcal{N}$ is true;

5. if $\phi$ is $\psi \rightarrow \theta$, then $\phi^\mathcal{N}$ is false if and only if $\psi^\mathcal{N}$ is true and $\phi^\mathcal{N}$ is false;

6. if $\phi(y, z)$ is $\forall x \psi(x, y, z)$, then $\phi^\mathcal{N}[b, c]$ is true if and only if $\psi^\mathcal{N}[a, b, c]$ is true for all $a$ in $\mathbb{N}$.

7. if $\phi(y, z)$ is $\exists x \psi(x, y, z)$, then $\phi^\mathcal{N}[b, c]$ is true if and only if $\psi^\mathcal{N}[a, b, c]$ is true for at least one $a$ in $\mathbb{N}$.
Example. Consider the following formula:

$$\phi(x, y) : \exists z \ (y = x \times z).$$

We evaluate $\phi^N[3, 12]$ and $\phi^N[2, 5]$:

$$\phi^N[3, 12]$$

is true iff $(\exists z \ (y = x \times z))^N[3, 12]$ is true

iff $(y = x \times z)^N[a, 3, 12]$ for some $a$ in $\mathbb{N}$

iff $(12 = 3 \times a)^N$ for some $a$ in $\mathbb{N}$

iff $12 = 3 \times a$ for some $a$ in $\mathbb{N}$.

Clearly, the last line is true, since $12 = 3 \times 4$. So the formula $\phi$ is true at $3, 12$. However:

$$\phi^N[2, 5]$$

is true iff $(\exists z \ (y = x \times z))^N[2, 5]$ is true

iff $(y = x \times z)^N[a, 2, 5]$ for some $a$ in $\mathbb{N}$

iff $(5 = 2 \times a)^N$ for some $a$ in $\mathbb{N}$

iff $5 = 2 \times a$ for some $a$ in $\mathbb{N}$.

In the second case, since there is no integer $a$ for which $5 = 2 \times a$, we conclude that $\phi$ is false at $2, 5$. These considerations lead to the observation that the formula $\phi$ is true in $\mathbb{N}$ at numbers $b, c$ if and only if $b$ divides $c$.

**Defining Functions and Relations in the Standard Model**

We can now explain what it means for a function to be definable in the standard model of arithmetic. We will give this explanation for functions of two variables, but generalizing to $n$ variables is a straightforward exercise. Also, we explain what it means for two-place relations (which are a generalization of unary functions) to be definable in $\mathbb{N}$; in fact the latter will be a natural starting point for understanding definability of functions in $\mathbb{N}$.

Suppose $R$ is a subset of $\mathbb{N} \times \mathbb{N}$. (A familiar example is the “less than relation,” expressed as ordered pairs: define $R$ to consist of those pairs $(a, b)$ for which $a < b$.) We say that $R$ is *definable in $\mathbb{N}$* if there is a formula $\phi(x, y)$ such that, for all pairs of natural numbers $(a, b)$, we have

$$(a, b)$$

is a member of $R$ if and only if $\phi^N[a, b]$ is true.
Example. Let $R$ be the set of all ordered pairs $(a, b)$ for which it is true that $a$ divides $b$. Let $\phi(x, y)$ be the formula defined in the previous example, namely

$$\phi(x, y) : \exists z (y = x \ast z).$$

Then, as the previous example explains, we have

$$(a, b) \in R \text{ if and only if } \phi^N[a, b] \text{ is true.}$$

Therefore, the “divides” relation is definable in $\mathbb{N}$.

Exercise. Let $R$ be the set of all ordered pairs $(a, b)$ for which $a \leq b$. Let $\phi(x, y)$ be the formula defined by

$$\phi(x, y) : \exists z (y = x + z).$$

Show that

$$(a, b) \in R \text{ if and only if } \phi^N[a, b].$$

Therefore the “$\leq$” relation is definable in $\mathbb{N}$.

Suppose $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is a partial function, with domain $A$. Then we say $f$ is definable in $\mathbb{N}$ if there is a formula $\psi(x, y, z)$ such that, for all natural numbers $a, b, c$, we have

$$c = f(a, b) \text{ if and only if } \psi^N[a, b, c].$$

Example. We show that the function $\text{gcd}$ defined earlier is definable in $\mathbb{N}$. Since we already know that “divides” and “less than or equal to” are definable relations, we will use convenient symbols for these relations (‘|’ and ‘$\leq$’ respectively) in the formula we will build, with the understanding that these symbols could be replaced by the defining formulas in a straightforward way. We define $\psi(x, y, z)$ as follows:

$$\psi(x, y, z) : (z \mid x \land z \mid y \land [\forall u (u \mid x \land u \land y \rightarrow u \leq z)])$$

Now, given $(a, b, c)$, we verify that $c = \text{gcd}(a, b)$ if and only if $\psi^N[a, b, c]$.

$$c = \text{gcd}(a, b) \text{ iff } c \mid a \text{ and } c \mid b \text{ and } c \text{ is the largest such number}$$

iff $c \mid a$ and $c \mid b$ and for any $d$ that divides both $a$ and $b$, $d \leq c$

iff $c \mid a$ and $c \mid b$ and for each $d$ in $\mathbb{N}$ ($d \mid a$ and $d \mid b$
implies \( d \leq c \)
\[ \text{iff } \psi^N[a, b, c]. \]

This example demonstrates the pattern that we have for all definable functions: the input/output behavior of such a function—its “existence” as a set of ordered pairs—is captured in a formula which defines the function relative to the standard model \( N \). It is this “intelligence” characteristic provided by such a formula that allows us to conceive of the function at all. Functions that have, beyond their bare existence as a set of ordered pairs, a formula that defines them from a specialized subcollection of the class of all number-theoretic functions form the class of definable functions.

Is Every Function \( N^k \rightarrow N \) Definable?

We show here that, although it’s reasonably accurate to say that “any function you can think of” is definable in \( N \), in reality, nearly all functions \( N^k \rightarrow N \) are not definable.\(^3\) This observation should help provide perspective on the scope of our course. The finely honed tools that we will develop in our analysis of algorithms will be relevant only to a tiny sliver of the world of functions from \( N \rightarrow N \).

For convenience, we restrict ourselves to the case \( k = 1 \): we show that there are functions \( N \rightarrow N \) that are not definable. Remember that every definable function \( N \rightarrow N \) is defined by a formula \( \phi(x, y) \). \( \phi \) itself is an expression involving finitely many symbols. We could arrange all formulas in a sequence, according to length. First we list all the length-1 formulas, then the length-2 formulas, and so forth.\(^4\) (To be more careful in handling the fact that we have infinitely many variables in our language, we can devise the list more carefully as follows: List all length-1 formulas that use only variable \( v_1 \) first. Then list all length-1 and length-2 formulas that use variables only among \( v_1, v_2 \). Then list

---

3 As was discussed in the first footnote, this point about the size of the set of definable functions depends on our definition of “definable.” If instead we take “definable” to mean “constructible” (as a set theorist might do), the thread of reasoning still makes the same basic point but in a slightly different way. One shows in this case that one of the following two assertions must be true: (a) The set of definable functions is a smaller infinity than the set of all functions; (b) the set of computable functions is a smaller infinity than the set of definable functions. Either way, the set of computable functions always ends up being a tiny speck in the very large set of all number-theoretic functions.

4 For any \( m \), a length-\( m \) formula is a formula that has exactly \( m \) symbols.
all length-1, length-2, and length-3 formulas that use variables only among \( v_1, v_2, v_3 \). And so forth.) In this way, we have a list

\[ \phi_0, \phi_1, \phi_2, \ldots \]

of all formulas that define functions \( \mathbb{N} \rightarrow \mathbb{N} \). Replacing each \( \phi_i \) by the function that it defines gives us the list:

\[ f_0, f_1, f_2, \ldots \]

Now we have a list of all the definable functions \( \mathbb{N} \rightarrow \mathbb{N} \). The question is, does this list exhaust all the functions from \( \mathbb{N} \rightarrow \mathbb{N} \)? If we come up with a function not on this list, the answer must be “no.”

Consider the following function \( g : \mathbb{N} \rightarrow \mathbb{N} : \)

for each \( k \), \( g(k) = f_k(k) + 1 \).

Is \( g \) in the list? Notice that \( g \neq f_0 \) because \( g \) disagrees with \( f_0 \) at 0: Namely, \( g(0) = f'_0(0) + 1 \neq f_0(0) \). But the same can be said for each \( k \): \( g \neq f_k \) because \( g \) disagrees with \( f_k \) at \( k \), since \( g(k) = f_k(k) + 1 \neq f_k(k) \).

We have found a function \( g \) not on the list. Therefore, not every function \( \mathbb{N} \rightarrow \mathbb{N} \) is definable. But we have shown much more. In fact we have shown that, no matter what list of functions \( \mathbb{N} \rightarrow \mathbb{N} \) one comes up with,

\[ b_0, b_1, b_2, \ldots , \]

there will always be a function \( g \), defined exactly as above, that does not lie on the list. This means that the collection of all functions \( \mathbb{N} \rightarrow \mathbb{N} \) simply cannot be arranged in a list! The collection of all such functions is too big to be put in a list. This fact demonstrates that the size of the set of all functions \( \mathbb{N} \rightarrow \mathbb{N} \) is a higher order of infinity than the set of natural numbers (see Jech, 1978, for more on higher orders of infinity).

The point of all these esoteric considerations is that the collection of all functions that are definable represents only a miniscule fraction of the vast collection of all functions \( \mathbb{N} \rightarrow \mathbb{N} \).

**Diagonalization and Fixed Points**

The result proved in the previous section uses a technique that appears over and over in the theory of computability. We wished to show that some function existed outside a particular class of functions. The technique for building such a function is called *diagonalization*—we built
g by examining each $f_k$ and forcing $g$ to disagree with it. One says that
one builds $g$ by “diagonalizing out” of the class.

Diagonalization provides a rich analogy to the process of transcending. While “diagonalizing out” describes the phenomenon of “going beyond,” another process that arises in computability theory describes another aspect of transcending equally well: The process of finding fixed points. If $\alpha : Y \to Y$ is a function, a fixed point for $\alpha$ is a $y \in Y$ such that $\alpha(y) = y$. Finding fixed points involves locating the self-referral dynamics inherent in a process.

Both the process of diagonalization and of locating fixed points have been largely captured in a striking abstract diagram (Yanofsky, 2003):

![Abstract Diagram]

The following two theorems (actually two variants of a single theorem), which give significance to the diagram, are the following:

**Diagonalization Theorem.** Suppose $f : T \times T \to Y$ and $\alpha : Y \to Y$ are functions. Define $\Delta : T \to T \times T$ by $\Delta(t) = (t, t)$. Then if $\alpha$ has no fixed point, there is a function $g : T \to Y$ that is not represented by $f$—that is, there is no $t_0$ for which $f(t, t_0) = g(t)$ for each $t$. Moreover, $g$ can be defined by $\alpha \circ f \circ \Delta$.

**Fixed Point Theorem.** Suppose $f : T \times T \to Y$ and $\alpha : Y \to Y$ are functions. Define $\Delta : T \to T \times T$ by $\Delta(t) = (t, t)$. If $\alpha \circ f \circ \Delta$ can be represented by $f$ (that is, for some $t_0$, $f(t, t_0) = \alpha(f(\Delta(t)))$ for every $t$), then $\alpha$ must have a fixed point.

By way of review, if $f : A \to B$ and $g : B \to C$ are functions, then the composition of $g$ with $f$, denoted $g \circ f$, is defined by

- $\text{dom } g \circ f = A$
- $g \circ f(a) = g(f(a))$. 

260
The diagram asserts that $\alpha \circ f \circ \Delta = g$: Following the bottom arrow $g$ from $T$ to $Y$ is the same as following the arrow $\Delta$ from $T$ to $T \times T$, followed by the arrow $f$ from $T \times T$ to $Y$, followed by the arrow $\alpha$ from $Y$ to $Y$. For more on commutative diagrams, (see Pierce, 1991).

As an example, we use the Diagonalization Theorem to reprove the result in the previous section: Let $f_0, f_1, f_2, \ldots$ be a list of all the definable functions (or for that matter, any list of functions $\mathbb{N} \to \mathbb{N}$), as described earlier. Define $F : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ by

$$F(m, n) = f_n(m).$$

Define $\alpha : \mathbb{N} \to \mathbb{N}$ by $\alpha(k) = k + 1$. (The connection to the diagram is that we have set $T = \mathbb{N}$ and $Y = \mathbb{N}$.) The definition of $\alpha$ makes it clear that $\alpha$ has no fixed point. So the theorem applies, and we can find a $g$ that is not represented by $F$; in particular, $g = \alpha \circ F \circ \Delta$ does the job. This means that, for each choice of $n$, the function $f_n$ does not agree with $g$. But this is precisely what we were seeking: a function $g$ that disagrees with every one of the functions $f_n$ on the list. (Notice how the composition $\alpha \circ F \circ \Delta$ reveals the mechanics of diagonalization in this case: Requiring $\Delta$ to precede $F$ forces us to consider the values $F(n, n)$; further applying $\alpha$ makes sure that $\alpha(F(n, n))$ always differs from $F(n, n)$, and so, in particular, that $g(n)$ is different from $f_n(n)$.)

### Computable Functions

Prior to the 1950s, there was considerable interest in giving a mathematical characterization of those functions $\mathbb{N} \to \mathbb{N}$ (or $\mathbb{N}^k \to \mathbb{N}^m$) whose input/output behavior could be captured by an algorithm. An inherent difficulty in making this determination was that there were many views about the definition of “algorithm” (see Rogers, 1988, and Bell & Machover, 1977).

A natural, modern-day answer to the question is: A partial function $f : \mathbb{N} \to \mathbb{N}$ is computable if and only if there is a Java program which, on input $n$, either fails to halt (in case $n$ is not in the domain of $f$) or halts with output $f(n)$.

For example, the function $f$ defined by $f(n) = 2n$ is computable because the following Java method demonstrates that it is:
int f(int n) {
    return n + n;
}

For this criterion for computability to be workable, we need to introduce one abstraction in our concept of a “Java program”—we must assume that the program can be run on a computer with “as much memory as necessary.” Otherwise, the program given above for $2n$ will fail to compute values (correctly, or at all) once the input integer exceeds memory limits. Later in this section, we will make the necessary assumptions more precise.

Historically, many classes of functions have been proposed as candidates for the “class of all computable functions.” The definitions of these classes have varied widely. The following is a partial list (see Bell & Machover, 1977; Rogers, 1988; and Barendregt 1984).

(1) The class of partial functions computable by a Turing machine. Turing machines are extremely simple structures that behave like low-level programs—like assembly code for an extremely simple processor. Input and output are handled by a tape containing infinitely many cells which may be blank or contain any character from a predetermined finite alphabet—however, at any time, all but finitely many of the cells are blank. The tape head is used both for reading and writing characters on the tape. When the machine begins to run, it reads the current value of the tape and checks its instruction set to determine what to do next. Its next step depends upon the value read and its current state (it may be in any of finitely many states). The tape head can move one cell to the left or right, erase a value, or write a value. In taking such an action, it may enter a new state, or remain in its present state. On a given input, the machine may halt, but on other inputs it may not. A partial function $f: \mathbb{N} \rightarrow \mathbb{N}$ is Turing computable if there is a Turing machine $M$ which, when started on the first of $n$ consecutive cells containing the symbol ‘1’ (and the rest of the tape blank), the machine eventually halts with head on the first of $f(n)$ consecutive 1s (and the rest of the tape is blank) if $n$ is in the domain of $f$, or else never halts. Similarly, a partial function $f: \mathbb{N}^2 \rightarrow \mathbb{N}$ is Turing computable if there is a Turing machine $M$ which, when started on the leftmost 1 on a tape having $m$ consecutive 1s fol-
followed by a blank followed by \( n \) consecutive 1s (and the rest of the tape blank), eventually halts with head on the first of \( f(m, n) \) consecutive 1s (and the rest of the tape is blank) if \((m, n)\) is in the domain of \( f \), or else never halts.

(2) The class of partial functions computable by a nondeterministic Turing machine. A nondeterministic Turing machine is just like a Turing machine, except that, when reading a value on the tape, and when in a particular state, its next step is not, in some cases, uniquely determined—it may do any of a number of things. In this case, \( f \) is considered computable if there is a nondeterministic machine \( M \) which, when started on the first 1 of \( n \) consecutive 1s (and the rest of the tape blank), has an execution path that will cause \( M \) to eventually halt with tape head on the first 1 of \( f(n) \) consecutive 1s (and the rest of the tape blank). A similar description provides a definition for a partial function \( \mathbb{N}^2 \to \mathbb{N} \) to be computable by a nondeterministic Turing machine.

(3) The class of \( \lambda \)-definable partial functions. Church devised a formal system called the \( \lambda \)-calculus in an effort to build a foundation for all of mathematics based on the concept of a function. The formal system consists of an infinite set of variables \( v_1, v_2, \ldots \), together with \( \lambda \)-terms defined recursively by:

(a) Any variable is a term;

(b) If \( M \) and \( N \) are terms, \((MN)\) is a term;

(c) If \( M \) is a term and \( x \) is a variable, \((\lambda x.M)\) is a term.

The \( \lambda \)-notation is a way of specifying a function. Stepping outside the formal system for a moment, one can use the \( \lambda \)-notation to specify the function that takes \( n \) to \( 2n \) by writing \( \lambda n.2n \). (\( \lambda n.2n \) is the function that takes argument \( n \) to \( 2n \).)

Clause (c) says, intuitively speaking, that if \( M \) is a term, the “function” that takes \( x \) to \( M \) is also a term. Multiple applications of clause (c) are typically written in abbreviated form. For instance \( \lambda x.\lambda y.M \) is written \( \lambda xy.M \). The number of parentheses involved in writing down terms according to rules (b) and (c) tends to become unmanageable, so
there is an “association to the left” convention: It is understood that the expression \(MNP\) is shorthand for \(((MN)P)\). If \(x\) is a variable, \(x^2\) denotes the term \(xx\), \(x^3\) denotes \(x(xx)\), \(x^4\) denotes \(x(x(xx))\), and so forth.

Lambda-terms can often be “reduced” to simpler terms. Many of these reductions follow our intuitive expectation. For instance, the term \((\lambda x.y)M\) reduces to simply \(y\). The intuitive reason is that \(\lambda x.y\) is the function that takes any \(x\) to constant value \(y\). If this function is applied to the input \(M\), the output is simply \(y\). We write \((\lambda x.y)M \rightarrow y\). As an exercise, the reader is invited to verify intuitively why \((\lambda x.x(xy))M \rightarrow M(My)\).

Intuitively, the \(\lambda\)-terms that have no “free” variables correspond to functions with actual input/output behavior. For instance, the \(\lambda\)-term \(\lambda x.xx\) binds the variable \(x\); this makes it possible to uniquely determine an output from a given input. In this case, still speaking intuitively, \(\lambda x.xx\) is the function that, for any input \(a\), outputs \(aa\). By contrast, the \(\lambda\)-term mentioned in the previous paragraph—\(\lambda x.x(xy)\)—has the free variable \(y\), so any computation using this term will still leave the value of \(y\) undetermined. \(\lambda\)-terms that have no free variable are called closed terms. As was mentioned in the introduction, it can be shown that all closed terms can be derived from an initial set of just three closed terms—usually denoted \(I, K, S\). For completeness of the presentation, we give their definitions here:

\[I: \quad \lambda x.x \]
\[K: \quad \lambda xy.x \]
\[S: \quad \lambda xy.z.x(z(yz)).\]

Natural numbers are represented in the \(\lambda\)-calculus in the following way: the natural number \(n\) is represented by the \(\lambda\)-numeral \(\lambda xy.x^n y\). In the special case of \(n = 0\), the definition asserts that 0 is represented by the term \(\lambda xy.y\). A shorthand notation for the \(\lambda\)-numeral for \(n\) is \(\tilde{n}\).

We can now define what it means for a number-theoretic function to be computable in the sense of the \(\lambda\)-calculus. We do this for total functions; a bit of additional development would be needed to handle the case of partial functions. A total function \(f: \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}\) is said to be \(\lambda\)-definable if there is a \(\lambda\)-term \(X\) such that for each \(m, n \in \mathbb{N}\),

\[X \tilde{m} \tilde{n} \rightarrow f(m,n).\]
(This definition generalizes to arbitrarily many variables in the obvious way.)

As we will indicate shortly, the $\lambda$-definable functions turn out to be precisely the computable functions, the functions whose behavior is computable with an algorithm. The fact that all $\lambda$-definable functions (and in fact, all closed terms) are derivable from just three is one of many striking analogies between the way in which the self-referral computational dynamics of pure consciousness find a direct parallel in the foundational dynamics of the $\lambda$-calculus. According to Maharishi Vedic Science, the sequential unfoldment of the Veda, and, ultimately, all of manifest existence, arises from self-referral dynamics in which pure awareness interacts with itself, simultaneously playing the roles of knower (Rishi), process of knowing (Devata), and known (Chhandas). In the $\lambda$-calculus context, the three fundamental closed terms play a similar role: The term $I$, known as the identity combinator, represents pure silence, the witness, the Rishi value, since its action under application is to leave its argument unchanged. The term $K$ acts by “forgetting” its second argument; this “hiding” influence corresponds naturally to Chhandas. And the term $S$ has a strongly dynamic flavor as it distributes its third argument $z$ across the other two arguments, and so corresponds naturally to Devata. Moreover, all three can be seen as the “play” of a single term $X$ interacting with itself (the definition of $X$ is rather involved, but can be expressed succinctly using $K$ and $S$ as follows: $X = \lambda z.zKSK$).

(4) The class of partial recursive functions. One begins the definition of this class by specifying a Base Set of functions, all of which are obviously computable. Then one specifies a collection $\mathcal{O}$ of fundamental operations for building new functions from old ones. The final class consists of all those functions that can be obtained by finitely many applications of these operations to the functions in the Base Set. For this discussion, we will consider functions $\mathbb{N}^k \to \mathbb{N}$, for all $k \geq 1$.

**Base Set.** The Base Set $\mathcal{B}$ consists of the following functions:

1. The successor function: The function $f : \mathbb{N} \to \mathbb{N}$ defined by $f(n) = n + 1$. 

265
2. All constant functions: For each \( n \), the function \( f : \mathbb{N}^k \to \mathbb{N} \) defined by \( f(x_1, x_2, \ldots, x_k) = n \).

3. All projection functions: For each \( i \), with \( 1 \leq i \leq k \), the function \( f : \mathbb{N}^k \to \mathbb{N} \) defined by \( f(x_1, x_2, \ldots, x_k) = x_i \).

**Recursive Operations.** The recursive operations \( \mathcal{O} \) will be applied to the Base Set to build up a large class of functions. Here we define these operations on arbitrary functions and then specify how partial recursive functions are to be built up later on.

1. **Composition.** Given functions \( g_1, g_2, \ldots, g_m : \mathbb{N}^n \to \mathbb{N} \) and a function \( h : \mathbb{N}^m \to \mathbb{N} \), the composition \( h \circ (g_1, g_2, \ldots, g_m) \) is the function \( f : \mathbb{N}^n \to \mathbb{N} \) defined by

\[
f(x_1, \ldots, x_n) = h(g_1(x_1, \ldots, x_n), g_2(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n)).
\]

For the case \( m = 1 \), we write \( h \circ g_1 \) rather than \( h \circ (g_1) \). For the case in which any or all of the \( g_i \) are only partially defined, we adopt the convention that \( f(x_1, \ldots, x_n) \) is undefined unless \( g_i(x_1, \ldots, x_n) \) is defined for all \( i \).

2. **Primitive Recursion.** Given functions \( g : \mathbb{N}^{n-1} \to \mathbb{N} \) and \( h : \mathbb{N}^{n-1} \to \mathbb{N} \), the function \( f = \text{pr}(g, h) : \mathbb{N}^n \to \mathbb{N} \), obtained by primitive recursion from \( g \) and \( h \), is defined inductively as follows:

\[
f(0, x_2, \ldots, x_n) = g(x_2, \ldots, x_n)
\]

\[
f(x_1 + 1, x_2, \ldots, x_n) = h(x_1, f(x_1, x_2, \ldots, x_n), x_2, \ldots, x_n).
\]

When \( n = 1 \), we take \( g \) to be a constant (rather than a function). If \( g \) is only partially defined, \( f(0, x_2, \ldots, x_n) \) will be defined only when \( g(x_2, \ldots, x_n) \) is defined, and similarly in the definition of \( f(x_1 + 1, x_2, \ldots, x_n) \).

3. **Unbounded Search.** Suppose \( g : \mathbb{N}^{n+1} \to \mathbb{N} \) is a partial function. Then \( f = \text{us}(g) : \mathbb{N}^n \to \mathbb{N} \) is defined by

\[
f(x_1, \ldots, x_n) = \text{least } y \text{ such that } g(x_1, \ldots, x_n, y) = 0 \text{ and } (x_1, \ldots, x_n, z) \text{ is in the domain of } g \text{ for each } z \leq y.
\]
Let us say that a function $f : \mathbb{N}^n \to \mathbb{N}$ is derivable from $B$ and $O$ if there are functions $f_1, \ldots, f_m$, with $f = f_m$, having the following properties: For each $k \leq m$, $f_k$ is obtained in one of the following ways:

a. $f_k$ belongs to $B$;

b. for some $i_1, \ldots, i_r, j$ all less than $k$, $f_k = f_j \circ \langle f_{i_1}, \ldots, f_{i_r} \rangle$;

c. for some $i, j$ both less than $k$, $f_k = pr(f_i, f_j)$;

d. for some $j < k$, $f_k = us(f_j)$.

As a simple example, we give a derivation of the function $f(x_1, x_2) = x_1 + x_2$ from $B$, $O$:

- $f_1$ is the successor function.
- $f_2 : \mathbb{N} \to \mathbb{N}$ is defined by $f_2(x) = x$.
- $f_3 : \mathbb{N}^3 \to \mathbb{N}$ is defined by $f_3(x_1, x_2, x_3) = x_2$.
- $f_4 : \mathbb{N}^3 \to \mathbb{N}$ is obtained by composition: $f_4 = f_1 \circ f_3$. Therefore, $f_4(x_1, x_2, x_3) = x_2 + 1$.
- $f_5 : \mathbb{N}^2 \to \mathbb{N}$ is obtained by primitive recursion: $f_5 = pr(f_2, f_4)$. Therefore,

\[
\begin{align*}
    f_5(0, x) &= f_2(x) = x \\
    f_5(1, x) &= f_5(0, f_5(0, x), x) = x + 1 \\
    f_5(2, x) &= f_5(1, f_5(1, x), x) = x + 1 + 1 = x + 2 \\
    f_5(n + 1, x) &= f_5(n, f_5(n, x), x) = x + n + 1.
\end{align*}
\]

It follows that $f = f_5$, and so $f$ has been derived from $B$ and $O$.

Finally, the class of all partial recursive functions is defined to be the set of all functions that are derivable from $B$ and $O$.

It may not be obvious that some of the functions belonging to the class of partial recursive functions are only partially defined, since the base set of functions includes only total functions. Partially defined functions do arise, and they do so by virtue of applications of unbounded search. As an example, we show how the partially defined function

\[
f(x) = \begin{cases}
0 & \text{if } x \geq 1 \\\n\infty & \text{if } x = 0
\end{cases}
\]

is derivable.
• $f_1$ is the constant function with one argument having value 1 (i.e. $f_1(x) = 1$).
• $f_2$ is the constant function with three arguments having value 0 (i.e. $f_2(x, y, z) = 0$).
• $f_3 : \mathbb{N}^2 \rightarrow \mathbb{N}$ is obtained by primitive recursion: $f_3 = \text{pr}(f_1, f_2)$. Therefore,

$$f_3(0, y) = f_1(y) = 1; \quad f_3(x + 1, y) = f_2(x, f_3(x, y), y) = 0.$$

• $f_4 : \mathbb{N} \rightarrow \mathbb{N}$ is obtained by unbounded search: $f_4 = \text{us}(f_3)$.

Notice that, by the definition of unbounded search, $f_4(0) = 1$, but for all $x > 0$, $f_4(x) = 0$. It follows that $f = f_4$, and so the partially defined function $f$ has been derived from $B$ and $O$.

(5) The class of partial functions that are weakly representable in Peano Arithmetic. Peano Arithmetic (PA) consists of a set of sentences in the language of first-order arithmetic (described earlier). PA is an example of an axiomatic theory for arithmetic. As such, it provides a way of systematically deriving sentences that are true in the standard model $\mathbb{N}$. PA begins with a set of obviously true (in $\mathbb{N}$) sentences as its set of axioms, and then allows one to derive one theorem (true sentence) after another using a single rule of inference. Kurt Gödel showed that, although nearly every true sentence of arithmetic that would ever arise in mathematical practice is derivable from PA, PA is not rich enough to allow one to derive all the true sentences.\footnote{Moreover, he showed that there is no recursive axiomatic theory of arithmetic from which every true sentence of arithmetic can be derived. An axiomatic theory is said to be \textit{recursive} if there is a computer program that can, for any formula in the language given as input, correctly answer the question: Is this formula one of the axioms of the theory? Intuitively, this means that we have a clear idea of what the axioms are to begin with. Without this requirement, one could start with all kinds of axioms that may well permit a derivation of every true sentence in arithmetic, but would be so complicated that it would be impossible to obtain proofs from the axioms, since one would never know which sentences are axioms and which are not. A simple example of this phenomenon occurs if one takes as one’s set of axioms all true sentences of arithmetic!}

The following is a list of axioms for Peano Arithmetic (see Keisler & Robbins, 1996).
Axioms of Peano Arithmetic

1. Every axiom of first-order logic
2. \( s(0) = 1 \)
3. \( \forall x \neg (s(x) = 0) \)
4. \( \forall x \forall y (s(x) = s(y) \rightarrow x = y) \)
5. \( \forall x (x + 0 = x) \)
6. \( \forall x \forall y (x + s(y) = s(x + y)) \)
7. \( \forall x (x \ast 0 = 0) \)
8. \( \forall x \forall y (x \ast s(y) = (x \ast y) + x) \)
9. For each formula \( \phi(x, y) \), the following is an axiom:
   \[ \forall y (\phi(0, y) \land \forall x [\phi(x, y) \rightarrow \phi(s(x), y)] \rightarrow \forall x \phi(x, y)). \]

The axioms indicated in Axiom 1 are the axioms from which all theorems of first-order logic—which must hold true in every model of the language of arithmetic—can be derived (we do not list these here).

Axiom 9 is the Principle of Mathematical Induction. It says that if a formula is true at 0, and if, whenever it is true at some \( n \) it can be shown to be true at \( n + 1 \), then the formula must be true for all natural numbers.

Suppose \( \sigma \) is a sentence in the language of arithmetic. We say that \( \sigma \) is provable from PA, and we write PA \( \vdash \sigma \), if there are sentences \( \psi_1, \ldots, \psi_n \), with \( \sigma = \psi_n \), such that for each \( m \leq n \), \( \psi_m \) is obtained in one of the following ways:

1. (axiom) \( \psi_m \) is one of the axioms;
2. (modus ponens) there are \( j, k \) both less than \( m \) such that \( \psi_k \) is of the form \( \psi_j \rightarrow \psi_m \).

Modus ponens says that, in our derivation list of sentences, if we find \( \psi_j \) somewhere in our derivation, and somewhere else we find \( \psi_j \rightarrow \psi_m \), then it is legitimate for the sentence \( \psi_m \) to be included as part of the derivation. This is the only way that new sentences are derived from other sentences, and is called inference by modus ponens. Modus ponens says, in essence, that if you know the sentences \( p \) and \( p \rightarrow q \) are true, you may conclude that \( q \) is true too.
We can now define weak representability. As a matter of notation, let us specify some terms in arithmetic that will be useful. We will write $s^2(0)$ for $s(s(0))$, $s^3(0)$ for $s(s(s(0)))$, and, in general, $s^{n+1}(0)$ for $s(s^n(0))$. $s^n(0)$ provides us with a natural way of representing the natural number $n$ in the formal language of arithmetic. Now, suppose $f : \mathbb{N}^2 \rightarrow \mathbb{N}$ is a partial function. (We can provide a similar definition for any $f : \mathbb{N}^k \rightarrow \mathbb{N}$.) We say that $f$ is weakly representable in PA if there is a formula $\phi(x, y, z)$ such that, for all natural numbers $a, b, c$,

$$c = f(a, b) \text{ iff } PA \vdash \phi(s^a(0), s^b(0), s^c(0)).$$

**Example.** We give an example of a proof from PA. One of the true sentences of arithmetic—a well-known and popular fact—is that $1 + 1 = 2$. Formulating this sentence in the formal language of arithmetic gives us:

$$1 + 1 = s^2(0).$$

For practice, let us verify that this sentence is indeed true in $\mathbb{N}$. First, the term $1 + 1$ evaluates to 2, because

$$(1 + 1)^{\mathbb{N}} = 1^{\mathbb{N}} + 1^{\mathbb{N}} = 1 + 1 = 2.$$

But $s^2(0) = s(s(0))$ also evaluates to 2:

$$(s(s(0)))^{\mathbb{N}} = s^{\mathbb{N}}(s^{\mathbb{N}}(0^{\mathbb{N}})) = s(s(0)) = 2.$$

Therefore,

$$(1 + 1 = s^2(0))^{\mathbb{N}} \text{ is true.}$$

We now write a formal proof of this sentence from PA. In actual mathematical practice, proofs from any theory (PA, or any other) are written in a much more abbreviated and readable form, but can, in principle, always be transformed to a format like the following. We use a table format in order to make evident the reason for each step in the proof. The proof will demonstrate that

$$PA \vdash 1 + 1 = s(0).$$

<table>
<thead>
<tr>
<th>Statement</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\forall x (x + 0 = x)$</td>
<td>Axiom 5</td>
</tr>
<tr>
<td>$\forall x (x + 0 = x) \rightarrow 1 + 0 = 1$</td>
<td>Axiom 1</td>
</tr>
<tr>
<td>$1 + 0 = 1$</td>
<td>Modus ponens, (1), (2)</td>
</tr>
</tbody>
</table>
Having examined briefly each of five classes of functions, let us observe that all are obtained in significantly different ways, and in some cases, their relationship to the notion of computability seems remote. One of the remarkable discoveries in mathematical logic is that these five classes are all identical! We state this fact as a theorem below. Since we would like to include among these classes the class of all partial functions that are computable by a Java program, we take a moment to be more precise about this latter notion.

Suppose \( f : \mathbb{N}^k \rightarrow \mathbb{N} \) is a partial function. We declare that \( f \) is computable by a Java program if, given a computer, supporting operating system, and JVM in which there is no limit on memory, we can in principle (ignoring time constraints) write a Java method \( m_f \) (which can be executed by instantiating its enclosing class and making a method call with appropriate arguments) that accepts a BigInteger array as its argument, has a BigInteger return value, and behaves in the following way: If \( (a_1, a_2, \ldots, a_k) \) is in the domain of \( f \), then, on input consisting of the BigInteger array of \( a_1, a_2, \ldots, a_k \), \( m_f \) returns (in principle, ignoring time constraints) the BigInteger version of \( f(a_1, a_2, \ldots, a_k) \). If

<table>
<thead>
<tr>
<th>Statement</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>(4) ( \forall x \forall y (x + s(y) = s(x + y)) )</td>
<td>Axiom 6</td>
</tr>
<tr>
<td>(5) ( \forall x \forall y (x + s(y) = s(x + y)) \rightarrow 1 + s(0) = s(1 + 0) )</td>
<td>Axiom 1</td>
</tr>
<tr>
<td>(6) ( 1 + s(0) = s(1 + 0) )</td>
<td>Modus ponens, (4), (5)</td>
</tr>
<tr>
<td>(7) ( 1 + 0 = 1 \rightarrow s(1 + 0) = s(1) )</td>
<td>Axiom 1</td>
</tr>
<tr>
<td>(8) ( s(1 + 0) = s(1) )</td>
<td>Modus ponens, (3), (7)</td>
</tr>
<tr>
<td>(9) ( 1 + s(0) = s(1 + 0) \rightarrow 1 + s(0) = s(1) )</td>
<td>Axiom 1, (8)</td>
</tr>
<tr>
<td>(10) ( 1 + s(0) = s(1) )</td>
<td>Modus ponens, (6), (9)</td>
</tr>
<tr>
<td>(11) ( s(0) = 1 )</td>
<td>Axiom 2</td>
</tr>
<tr>
<td>(12) ( 1 + s(0) = s(1) \rightarrow 1 + 1 = s(1) )</td>
<td>Axiom 1, (11)</td>
</tr>
<tr>
<td>(13) ( 1 + 1 = s(1) )</td>
<td>Modus ponens, (10), (12)</td>
</tr>
<tr>
<td>(14) ( s(0) = 1 \rightarrow s(s(0)) = s(1) )</td>
<td>Axiom 1</td>
</tr>
<tr>
<td>(15) ( s(s(0)) = s(1) )</td>
<td>Modus ponens, (11), (14)</td>
</tr>
<tr>
<td>(16) ( 1 + 1 = s(1) \rightarrow 1 + 1 = s(s(0)) )</td>
<td>Axiom 1, (15)</td>
</tr>
<tr>
<td>(17) ( 1 + 1 = s(s(0)) )</td>
<td>Modus ponens, (13), (16)</td>
</tr>
</tbody>
</table>
\((a_1, a_2, \ldots, a_k)\) is not in the domain of \(f\), then, with input consisting of the BigInteger array of \(a_1, a_2, \ldots, a_k\), \(m_f\) never returns a value (this may occur because \(m_f\) has entered an infinite loop, because an Exception has been thrown, or any other reason). Such a Java method \(m_f\) will be called a *regular Java program* (even though it is just a Java method), or the *regular Java program that computes* \(f\).

As a simple example of a regular Java program that demonstrates computability, we show that \(f(n) = 2^n\) is computable:

```java
BigInteger f(BigInteger[] n) {
    return n[0].add(n[0]);
}
```

We can now state our theorem (see Bell & Machover, 1977):

**Unification Theorem.** The following classes of functions \(\mathbb{N}^k \to \mathbb{N}\), \(k \geq 1\), are identical:
1. The class of Turing-computable partial functions.
2. The class of nondeterministic Turing-computable partial functions.
3. The class of \(\lambda\)-definable partial functions.
4. The class of partial recursive functions.
5. The class of partial functions that are weakly representable in Peano Arithmetic.
6. The class of partial functions that are computable by a Java program.

On the basis of evidence similar to our Unification Theorem, Church proposed (1936) that the class of all partial functions whose input/output behavior could be captured by an algorithm was also identical to each of the classes mentioned above. This proposal has come to be known as *Church’s Thesis*. Since no definition of algorithm has acquired universal assent, Church’s Thesis cannot actually be proved or disproved. Nonetheless, the Unification Theorem (and other even stronger results of this kind) provides compelling evidence for the truth of Church’s Thesis. By now, it has achieved nearly universal acceptance by the mathematical and computer science community, and we will take it to be true in this course.
One general feature of all the classes of functions mentioned here is that the notion of computability of a function seems always to require that it exhibit a *sequence of steps* from a well-defined starting point, in order to arrive at a “computation.” This is apparent from the versions of computable function that involve some notion of a computing machine, such as Turing machines or Java programs: To compute the value of a function on any input, either the machine goes into an infinite loop (when the input is not in the domain of the function), or it performs a finite sequence of steps to arrive at an output. In the case of the classes of partial recursive functions and \( \lambda \)-definable functions, the derivations involved have a different flavor, but still involve finite sequences of steps. A function is partial recursive if it can be derived from a simple set of functions via finitely many applications of fundamental computable operations. In the case of \( \lambda \)-definable functions, all are derivable through reductions from the three basic closed terms (see above), though the steps are restricted by the need to operate on natural number inputs rather than on arbitrary domains. The sequential flow is seen in the form of finite-length proofs from Peano Arithmetic for the other two classes. Therefore, in general, a distinguishing characteristic of the class of computable functions is that they arise in the context of finite sequential unfoldment, expressed in various ways.

As we discussed in the introduction, sequential unfoldment, in the dynamics of pure consciousness according to Maharishi Vedic Science, begins with the emergence of “three” from “two”; with the emergence of Rishi, Devata, and Chhandas. For this reason, while we associated the class of definable functions with the emergence of existence and intelligence, or “two,” from the unified, Samhita, value of consciousness, we associate the class of computable functions with the emergence of the flow of intelligence in the form of Rishi, Devata, and Chhandas, which gives rise to the sequential unfoldment of creation.

**Is Every Definable Function Computable?**

In this section, we show that the class of computable functions is a proper subset of the class of definable functions. This result shows that the requirement on a function that it should be possible to compute its input/output behavior in finite-length computations is more stringent and demanding than the bare assertion that such a function exists (in
the sense that it can be defined in the standard model). To say it another way, knowing the behavior of an existing function does not suffice to tell one how to compute its values. After showing that every computable function is indeed definable (as might be expected), we move on to an important example of a function that is definable but not computable (for a complete proof, see Keisler and Robbins, 1996).

**Computable Implies Definable Theorem.** Every partial function \( \mathbb{N}^k \to \mathbb{N} \) that is computable is also definable in \( \mathcal{N} \).

**Partial Proof.** We give a proof here for total computable functions; the proof for partial functions is considerably more difficult. Suppose \( f : \mathbb{N}^2 \to \mathbb{N} \) is a computable total function (the more general case in which \( k \) is arbitrary is similar). By part (5) of the Unification Theorem, there is a formula \( \phi(x, y, z) \) that weakly represents \( f \) in \( \text{PA} \); namely, for all natural numbers \( a, b, c \), we have

\[ c = f(a, b) \iff \text{PA} \vdash \phi(s^a(0), s^b(0), s^c(0)). \]

Recall that every sentence derivable from \( \text{PA} \) is true in \( \mathcal{N} \). Also notice that for any natural number \( e \), \( s^e(0) = e \). Therefore, if \( c = f(a, b) \), we have by (*) that \( \text{PA} \vdash \phi(s^a(0), s^b(0), s^c(0)) \). It follows that \( \phi(s^a(0), s^b(0), s^c(0)) \) is true, and so \( \phi^\mathcal{N}[a, b, c] \) is true.

On the other hand, suppose \( c \neq f(a, b) \). Then, since \( f \) is total, there is some \( d \neq c \) such that \( f(a, b) = d \), and again we have

\[ \text{PA} \vdash \phi(s^a(0), s^b(0), s^d(0)), \]

and again

\[ \phi^\mathcal{N}[a, b, d] \] is true.

Since \( c \neq d \), it follows that

\[ \phi^\mathcal{N}[a, b, c] \] is false.

We have therefore shown that

\[ c = f(a, b) \iff \phi^\mathcal{N}[a, b, c] \] is true,

and so \( f \) is definable. □
In order to give our example of a definable function that is not computable, we develop an encoding scheme that will allow us to associate to each regular Java program an integer code, from which we will be able to reconstruct the program. We make the assumption that there are fewer than 500 distinct characters that ever occur in a regular Java program. Therefore, each character can be represented as a bit string of length 9 (since $2^9 = 512$). For the moment, let us assume that some assignment of characters to length-9 bit strings has been made. The first step of encoding a regular Java program is to minimize whitespace by removing all line breaks and reducing multiple consecutive occurrences of a blank space to a single blank space, and then to translate all characters to length-9 bit strings, and concatenating. The result will be a long bit string. We prepend the character ‘1’ to this long bit string so that we can view this string as a binary number (the initial 1 takes care of the cases in which the long bit string happens to begin with one or more 0s).

**Example.** We show how to encode a simple regular Java program. There is nothing special about the way we have chosen to assign bit strings to characters. Codes have not been given for all possible characters—we have just given here enough codes to illustrate this example.

```java
BigInteger f(BigInteger[] n) {
    return n[0].add(n[0]);
}
```

<table>
<thead>
<tr>
<th>Char</th>
<th>Code</th>
<th>Char</th>
<th>Code</th>
<th>Char</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0000000001</td>
<td>d</td>
<td>0000000010</td>
<td>e</td>
<td>0000000011</td>
</tr>
<tr>
<td>f</td>
<td>0000000100</td>
<td>g</td>
<td>0000000101</td>
<td>i</td>
<td>0000000110</td>
</tr>
<tr>
<td>n</td>
<td>000000111</td>
<td>r</td>
<td>000001000</td>
<td>t</td>
<td>000001001</td>
</tr>
<tr>
<td>u</td>
<td>000001010</td>
<td>B</td>
<td>000001011</td>
<td>I</td>
<td>000001100</td>
</tr>
<tr>
<td>.</td>
<td>000001101</td>
<td>{</td>
<td>000001110</td>
<td>)</td>
<td>000001111</td>
</tr>
<tr>
<td>(</td>
<td>000010000</td>
<td>)</td>
<td>000010001</td>
<td>;</td>
<td>000010010</td>
</tr>
<tr>
<td>(space)</td>
<td>000010011</td>
<td>[</td>
<td>000010100</td>
<td>]</td>
<td>000010101</td>
</tr>
<tr>
<td>0</td>
<td>000010110</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The following bit string (including the prepended ‘1’) represents the regular Java program above:
We shall call the bit string code for a regular Java program $P$ the Gödel number of $P$, and denote it $\#P$. Notice that the code table can be used to reconstruct $P$ from $\#P$. Another thing to notice is that, even though the Java program we have given in the example is quite short, the bit string is rather long. This suggests that “most” (binary) integers will not be codes for any Java program. However, it is not hard to see that one could write a regular Java program IsGodel that accepts an integer (by convention, this means a BigInteger type) and outputs 1 if the binary equivalent of the integer is a Gödel number, or 0 if not. Such a program would have to embed the reconstructed Java method into a test class and attempt to compile; it would verify that the method had an appropriate return value and an input argument of type BigInteger[].

Since all the Gödel numbers of regular Java programs are integers, they can be listed in an ordered sequence:

$$g_0 < g_1 < g_2 < \ldots .$$

We will now arrange, once and for all, the regular Java programs in a sequence—programs will be arranged according to the order of their Gödel numbers. The sequence of programs

$$P_0, P_1, P_2, \ldots$$

is devised so that for each integer $e$, $P_e$ is the Java program whose Gödel number is $g_e$. Let us now make one further observation: There is a regular Java program Gen which, given any integer $e$, generates the code for program $P_e$. The program works like this: It examines the natural numbers 0, 1, 2, \ldots one by one, in order, seeking Gödel numbers (it will make use of IsGodel). Each time it finds a Gödel number, it adds a tally to an initially empty list. When the number of tallies finally becomes $e$, this means that the program has located the Gödel number
g\textsuperscript{e}. It will then reconstruct the program \( P_e \) from \( g \) (we mentioned above how this can be done). We will say that the number \( e \) is the index for the Java program \( P_e \).

Now observe that, by the conventions we have adopted, every regular Java program computes, in one way or another, a function \( \mathbb{N} \rightarrow \mathbb{N} \), another function \( \mathbb{N}^2 \rightarrow \mathbb{N} \), another function \( \mathbb{N}^3 \rightarrow \mathbb{N} \), etc. Although we may create such a program with a function \( \mathbb{N} \rightarrow \mathbb{N} \) in mind, nonetheless, the other types of functions are implicitly defined as well. As an example, consider the code given above for computing the function \( f: \mathbb{N} \rightarrow \mathbb{N} \) given by \( f(n) = 2^n \). But if we pass in a 2-element array of integers, the program will still return the result of adding the 0th array element to itself. In other words, the program computes the function \( g: \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N} \) defined by \( g(m, n) = m + m \).

With these points in mind, we will denote the function \( \mathbb{N}^n \rightarrow \mathbb{N} \) computed by \( P_e \) by \( \phi_e^{(n)} \). The superscript will be omitted when the number of arguments of the function is clear from the context. Our earlier discussion shows that the \( n \)-argument computable function \( \phi_e^{(n)} \) can be computed from \( e \): First compute \( P_e \) from \( e \); then for any \( n \)-ary argument \((a_1, \ldots, a_n)\), run \( P_e \) on these arguments and output the result. We will say that \( e \) is the index of \( \phi_e^{(n)} \).

Summarizing the points above, we have the following (see Rogers, 1988, or Keisler & Robbins, 1996):

**Enumeration Theorem.**

1. Every regular Java program \( P \) has a Gödel number \( \#P \), which is a binary integer (sometimes identified with its base 10 equivalent).
2. There is a regular Java program IsGodel that accepts one natural number for input, and returns 1 if that number is a Gödel number, 0 otherwise.
3. The regular Java programs can be effectively arranged in a sequence; that is, we can write the regular Java programs in a sequence \( P_0, P_1, \ldots, P_e, \ldots \) so that one can, with a regular Java program, generate the Java code for program \( P_e \) from the integer \( e \).
4. For each \( n \geq 1 \), each of the \( n \)-argument computable functions \( \phi_0^{(n)}, \phi_1^{(n)}, \ldots, \phi_e^{(n)}, \ldots \) can be computed from its index by a regular Java program.
We can now give an example of a definable function that is not computable.

**Example: The Halting Problem.** We define a function $H(x, y): \mathbb{N}^2 \to \mathbb{N}$ as follows:

$$H(e, n) = \begin{cases} 1 & \text{if program } P_e \text{ halts on input } n \\ 0 & \text{otherwise} \end{cases}$$

The function $H$ tells whether a given program will halt on a given input. $H$ is a well-defined function and we may conclude, intuitively at least, that $H$ is definable. We will show more formally later that it is indeed definable in $\mathcal{N}$. We show here, however, that there is no algorithm that computes the values of $H$. We will use the Diagonalization Theorem to establish this result.

First, we create a Java program $A$ that does the following: On any input other than 1, the program outputs the number 1. On input 1, $A$ goes into an infinite loop. For this example, we denote by $\alpha$ the 1-argument function computed by $A$. $\alpha$ is the partial function $\mathbb{N} \to \mathbb{N}$ that is undefined at 1, but has constant value 1 for all other input values. Notice that $\alpha$ does not have a fixed point. Let $f(x, y) = H(y, x)$. By the Diagonalization Theorem, the function $g = \alpha \circ f \circ \Delta$ is not represented by $f$—that is, for each $e$, there is an $n$ such that $g(n) \neq f(n, e) = H(e, n)$. Notice that $\Delta$ and $\alpha$ are computable.

Now let us suppose that $H$ is computable. It follows that both $f$ and $g$ are also computable. Let $P_e$ be a program that computes $g$. We arrive at an absurdity in trying to determine the value of $H(e, e)$. There are two possibilities: either $H(e, e) = 1$ or $H(e, e) = 0$. We consider each separately.

If $H(e, e) = 1$, then by definition of $H$, $P_e$ must halt on input $e$. But by definition of the program $P_e$, because $H(e, e) = 1$, $P_e$ must go into an infinite loop on input $e$, and therefore does not halt on input $e$.  

278
If \( H(e, e) = 0 \), then by definition of \( H \), \( P_e \) does not halt on input \( e \). However, by the definition of \( P_e \), since \( H(e, e) \neq 1 \), \( P_e \) must halt on input \( e \).

Either way, the conclusion is absurd. This demonstrates that the assumption that \( H \) is computable must be false.

Notice that the reasoning given above doesn’t go through if we do not assume \( H \) is computable. When we make this assumption, we can find a program index \( e \) (the index for \( g \)) that leads to a paradox. But without this assumption, no such paradoxical \( e \) arises. Therefore, as a function, \( H \) is entirely legitimate; the only issue is that its behavior is not computable.

The definition of \( H \) given above suggests another function that is both computable and extremely useful. Let us define the partial function \( U(x, y) \) by

\[
U(e, n) = \begin{cases} 
\phi_e(n) & \text{if } n \text{ is in the domain of } \phi_e \\
\text{undefined} & \text{otherwise}
\end{cases}
\]

We have already discussed an algorithm for computing \( U \): By part (4) of the Enumeration Theorem, there is a program which, on input \( e, n \), outputs the value that is returned from \( P_e \) when running on input \( n \), if \( P_e \) halts on input \( n \); otherwise, it does not halt. \( U \) is called the universal function for 1-argument functions. There are similar universal functions for \( n \)-argument functions, for every \( n \).

Using \( U \), we can show more formally that the halting function \( H \) is definable in \( \mathbb{N} \): Since \( U \) is computable, it is also definable by a formula \( \phi(x, y, z) \). One can now show that \( H \) is definable in \( \mathbb{N} \) by the following formula \( \psi(x, y, u) \):

\[
\psi(x, y, u) : (\exists z \phi(x, y, z) \rightarrow u = 1) \wedge (\neg \exists z \phi(x, y, z) \rightarrow u = 0).
\]

The formula says that if \( U(x, y) \) has a value \( z \), then the value computed by \( H(x, y) \) will be 1, whereas if \( U(x, y) \) is undefined, then the value computed by \( H(x, y) \) will be 0. This correctly specifies \( H \), and one proves fairly easily that

\[ e = H(a, b) \iff \psi^\mathbb{N}[a, b, e]. \]

The Complexity of Algorithms

Once we know that a function \( f : \mathbb{N}^k \rightarrow \mathbb{N} \) is computable, we can ask, “Can the input/output behavior of \( f \) be captured by an efficient algo-
algorithm?” Earlier, we considered two algorithms that implemented the function `gcd` and observed that one of them was considerably more efficient than the other. What is it that causes one algorithm to “run faster” than another? The basic measure of algorithm efficiency is the relationship between the size of the input and the number of steps required to arrive at the output. Since different inputs of size \( n \) may require a different number of steps, depending on the particulars of the input, we are most often concerned with the performance in the worst case.

**Definition.** The worst-case complexity of an algorithm \( A \) is a function \( w : \mathbb{N} \to \mathbb{N} \) defined as follows: For each \( n \), \( w(n) \) denotes the maximum number of steps performed by \( A \) in computing (and returning) output when the size of the input is \( n \). In other words, \( w(n) \) equals the number of steps required to process an input of size \( n \) in the worst case.

A familiar example can be found among sorting algorithms. We start with a list of comparable objects (say, integers) and we ask our algorithm to produce a sorted list. The algorithm’s sorting speed may depend on the initial arrangement of integers. For instance, for many sorting algorithms, if the initial list is already sorted, the output will be returned more quickly than if the initial list is ordered in some other way, say in reverse order or randomly.

In practice, it has been found that algorithms with a worst-case complexity that is not bounded by a polynomial, or indeed by a function of the form \( cn^k \) for some constants \( c, k \), run too slowly to be useful, unless inputs are guaranteed to be of small size. In particular, if the only known algorithms for a function have a running time that is exponential in the size of the input, implementation of the function is viewed as infeasible. (See Cormen, 2001, for a full treatment of these topics.)

**Definition.** An algorithm is *polynomial-bounded* if there is a polynomial \( p(n) \) such that for any input of size \( n \), the algorithm returns output in fewer than \( p(n) \) steps. The class of polynomial-bounded algorithms is denoted \( P \).

In this course, we will focus on polynomial-bounded algorithms, and develop analytical tools for improving their efficiency and investi-
gating their degree of complexity. As a secondary topic, we will take a look at some functions whose only known algorithms have exponential complexity. There is a vast collection of such algorithms. Perhaps surprisingly, some of these are known to be “more intractable” than others. Functions that admit no feasible algorithm sometimes do have a tractable feature that can be exploited.

Usually, the algorithms we study will be concerned with producing a “solution” to a problem—producing a sorted list, finding a shortest path through some data structure, optimizing a cost. Typically, then, one thinks of the class $P$ as representing the class of all problems that admit a polynomial-bounded solution. Many of the “hard” problems, which do not belong to $P$, are known to have a special feature that makes them more tractable: if a solution to the problem is given, the number of steps required to check that the solution is correct is polynomial bounded. Such problems are called nondeterministically polynomial bounded. The class of all such algorithms is denoted $NP$.

It should be apparent that every problem in $P$ also belongs to $NP$; however the converse inclusion is not known to be true. As we have said, the only known algorithms for many of the problems in $NP$ are exponentially slow, but this does not eliminate the possibility that one day someone will discover a polynomial-bounded algorithm that solves the problem. It is widely believed, however, that the classes $P$ and $NP$ are different. One striking feature of the class $NP$ is the existence of $NP$-complete problems—these are problems with the remarkable property that if a polynomial-bounded algorithm for the problem is ever found, there will automatically be polynomial-bounded algorithms for all problems in $NP$! We will discuss this remarkable phenomenon at the end of the course.

We close this section of the paper with a short list of some well-known $NP$-complete problems.

**Subset Sum.** The inputs are positive integers $C, m_1, \ldots, m_k$. The problem is: Among subsets of $\{m_1, \ldots, m_k\}$ having sum at most $C$, what is the largest subset sum?

**CNF-Satisfiability.** Consider the following language. We have a list of boolean symbols $p, q, r, \ldots$. We understand these to be variables
that can be assigned a value of either true or false. We have the usual
connectives and (‘∧’), or (‘∨’), and not (‘¬’). (For this discussion, we omit
the connective “implies.”) A boolean combination of boolean symbols is
obtained by applying the following rules:

1. any boolean symbol is a boolean combination;
2. if \( A \) is a boolean combination, so is \( \neg A \);
3. if \( A \) and \( B \) are boolean combinations, so are \( A \land B \) and \( A \lor B \).

We are interested in determining the truth value of a boolean com-
bination, given an assignment of truth values to the boolean symbols.
For instance, if we make the assignment
\[ p = \text{true}, \quad q = \text{false}, \]
what is the truth value of \( \neg p \lor q \)? Since \( p \) is true, \( \neg p \) is false, and since \( q \)
is also false, we conclude that the whole statement is false.

The problem we want to solve is this: Given a boolean combination,
is there a truth assignment for the boolean symbols for which the bool-
ean combination evaluates to true?

To make the problem more tractable, theorists observed early on that
any boolean combination is equivalent to (i.e. has the same truth val-
ues as) a boolean combination in \textit{conjunctive normal form}. A boolean
combination is in conjunctive normal form if it consists of one or more
Boolean combinations connected by “ands,” like this:
\[ A \land B \land C \land D \land \ldots. \]
Each of these component boolean combinations contains no “and” con-
nective, and must be in the following form:
\[ X \lor Y \lor Z \lor \ldots. \]
Each of the components in this latter boolean combination is either a
boolean symbol or the negation of a boolean symbol. Here is an exam-
ple of a boolean combination in conjunctive normal form:
\[ (p \lor q \lor \neg r) \land (\neg s \lor \neg p \lor t) \land (\neg p \lor s). \]

Finally, then, the \textit{CNF-Satisfiability Problem} is the following: The
input is a set of boolean symbols and a CNF boolean combination. The
problem is to find a truth assignment for the boolean symbols for which
the CNF boolean combination evaluates to true (or the algorithm outputs false if no such assignment exists).

**Knapsack.** Suppose we have a knapsack of capacity $C$ (a positive integer) and $n$ objects with sizes $s_1, \ldots, s_n$ and “profits” $p_1, \ldots, p_n$ (all of which are positive integers). The problem is to find the largest total profit of any subset of the object for which the sum of the sizes does not exceed $C$.

**Traveling Salesman Problem.** Suppose we have $n$ cities. Find the shortest path that one can follow which passes through each city once and only once.

**Conclusion**

We have shown in this article how the class of functions that are actually used for purposes of developing software and that rest at the heart of a highly creative and rapidly developing industry represent a mere speck in a much vaster wholeness consisting of all number-theoretic functions. This fact mirrors the dynamics of the unfoldment of creation itself, as described in Maharishi Vedic Science: The infinite dynamism and creativity at the basis of the sequential unfoldment of manifest existence arises in the collapse of the unbounded silent aspect of wholeness to a point. In computer science, the unbounded value of wholeness corresponds to the class of all possible number-theoretic functions, whereas the “point” corresponds to the extremely meager and narrowly defined class of polynomial-time computable functions.

In examining the stages of “collapse” from all functions to the polynomial-time-bounded computable functions, we have also observed another theme of unfoldment that parallels the dynamics of pure consciousness: Just as wholeness moves from the state of Samhita, unified consciousness or “one,” to the state of “two,” existence and intelligence, to the state of “three” as Rishi, Devata, Chhandas, from which all possible transformations of Samhita sequentially unfold as creation, so likewise does the “togetherness of all functions” move to the class of definable functions which display the duality of existence (each function exists as a set of ordered pairs) and intelligence (each definable function’s behavior is captured in a formula), to the class of computable functions, which are characterized by being specifiable via finite-
length derivations (and in the $\lambda$-calculus formulation, all derivations arise from three fundamental closed terms, which in turn arise from the self-interacting dynamics of the single $X$ combinator). Moreover, this class of computable functions lies at the basis of all modern-day creative applications of computability.

These parallels serve to bring into focus the hidden dynamics of pure intelligence in the workings of the modern-day theory of computation. It suggests that much of the great power and creativity that infuses the business of building software is due to a fundamental collapse within the very structure of mathematical transformation, from an unbounded field of all possibilities to a “point value”—the tiny packet of algorithms that contains only the most efficiently computable types of transformations, which lend themselves to application in the concrete world of software development.

References


Maharishi Mahesh Yogi (1992). *Constitution of the universe: The source of all order and harmony displayed throughout the universe*. Fairfield,
Iowa: Maharishi International University Press.
Part V

Appendices
Modern Science and *Vedic Science*:
An Introduction

Kenneth Chandler, Ph.D.
ABOUT THE AUTHOR

Kenneth Chandler holds a Ph.D. in Philosophy from the University of Texas at Austin. He served as Head of the Department of the Science of Creative Intelligence at Maharishi International University (today, Maharishi University of Management). Dr. Chandler continues his research into consciousness and is currently at work on a book on descriptions of the experience of transcending and pure consciousness in the mainstream classics of philosophy, science, religion, and the arts. It will be a three-volume set covering from the Vedic tradition to the present.
Modern Science and *Vedic Science*: An Introduction

This journal (*Modern Science and Vedic Science*) provides a forum for research on the forefront of mankind’s expanding knowledge of the universe. It is devoted to exploration of the unified field of all the laws of nature through the combined approaches of modern science and ancient Vedic science, as brought to light by Maharishi Mahesh Yogi. The identification of the unified field by modern physics is only the first glimpse of a new area of investigation that underlies all disciplines of knowledge, and which can be explored not only through objective science but through a new technology of consciousness developed by Maharishi.

The unified field is now beginning to be understood through modern physics as the unified source of the entire universe, as a unified state of all the laws of nature from which all force and matter fields sequentially emerge according to exact dynamical principles. As each science and each academic discipline progresses to uncover its own most basic laws and foundational principles, each is beginning to discover that the roots of these laws and principles can be traced to the unified field.

This journal recognizes a new method of gaining knowledge of the unified field that combines the approach of the modern sciences with that of the most ancient of sciences, the ancient tradition of Vedic science. Many thousands of years ago, the seers of the Himalayas discovered, through exploration of their silent levels of awareness, a unified field where all the laws of nature are found together in a state of wholeness. This unity of nature was directly experienced to be a self-referral state of consciousness which is unbounded, all-pervading, unchanging, and the self-sufficient source of all existing things. They experienced and gave expression to the self-interacting dynamics through which this unified field sequentially gives rise to the diversity of all laws of nature. That experience is expressed in the ancient Vedic literature.
In our own time, Maharishi has brought to light the knowledge of this ancient science and integrated it with the modern sciences in such a way that Vedic science and modern science are now seen as complementary methods of gaining knowledge of the same reality—the unified field of all the laws of nature. The knowledge of this ancient science that Maharishi has brought to light is known as Maharishi Vedic Science.

Maharishi Vedic Science is to be understood, first of all, as a reliable method of gaining knowledge, as a science in the most complete sense of the term. It relies upon experience as the sole basis of knowledge, not experience gained through the senses only, but experience gained when the mind, becoming completely quiet, is identified with the unified field. This method, examined in relation to the modern sciences, proves to be an effective means of exploring the unified field of all the laws of nature. On the basis of this method, complete knowledge of the unified field becomes possible. It is possible to know the unified field both subjectively on the level of direct experience through exploration of consciousness and objectively through the investigative methods of modern science. Maharishi Vedic Science gives complete knowledge of consciousness, or the knower, complete knowledge of the object known, and complete knowledge of the process of knowing. In knowing the unified field, all three—knower, known, and process of knowing—are united in a single unified state of knowledge in which the three are one and the same.

Maharishi has developed and made available a technology for the systematic exploration of the unified field. This technology is a means by which anyone can gain access to the unified field and explore it through experience of the simplest and most unified state of consciousness. As this domain of experience becomes universally accessible, the unified field becomes available as a direct experience that is a basis for universal knowledge. The technology for gaining access to the unified field is called the Transcendental Meditation technique and its advanced programs, and the science based on this experience, which links modern science and Maharishi Vedic Science in a single unified body of knowledge, is called the Science of Creative Intelligence.

Maharishi is deeply committed to applying the knowledge and technology of the unified field for the practical benefit of life. He has
developed programs to apply this knowledge to every major area of human concern, including the fields of health, education, rehabilitation, and world peace. These applications of Maharishi’s technologies of consciousness have laid it open to empirical verification and demonstrated its practical benefit to mankind. Hundreds of scientific studies have already established its usefulness. From these results, it is clear that Maharishi’s technologies of consciousness are far more beneficial than technologies based on present day empirical science; they promise to reduce and even eliminate war, terrorism, crime, ill health, and all forms of human suffering.

These technologies, which are the applied value of Maharishi Vedic Science, represent a great advance in methods for gaining knowledge. Past science was based on a limited range of knowledge gained through the senses. This new technology opens to mankind a domain of experience of a deeper and more far-reaching import. It places within our grasp a new source of discovery of laws of nature that far exceeds the methods of modern science, yet remains complementary to these methods.

Modern science and Maharishi Vedic Science, explored together, constitute a radically new frontier of knowledge in the contemporary world, opening out vistas of what it is possible for mankind to know and to achieve, which extend far beyond present conceptions, and which demand a re-evaluation of current paradigms of reality and a reassessment of old conceptions of the sources and limits of human knowledge.

This introductory essay will provide a preliminary understanding of what the unified field is, what Maharishi Vedic Science is, and how Maharishi Vedic Science and modern science are related. It also defines fundamental concepts and terminology that will be frequently used in this journal and surveys the practical applications of this new technology. We begin with a description of the unified field as understood in modern science.

**The Unified Field of Modern Science**

Within the last few years, modern theoretical physics has identified and mathematically described a unified field at the basis of all observable states of physical nature. Einstein’s hope of finding a unified field theory to unite the electromagnetic, gravitational, and other known
force fields has now been virtually realized in the form of unified quantum field theories. Instead of having several irreducible and distinct force fields, physics can now mathematically derive all four known force fields from a single supersymmetric field located at the Planck scale ($10^{-33}\text{cm}$ or $10^{-43}\text{sec}$), the most fundamental time-distance scale in nature. This field constitutes an unbounded continuum of non-changing unity pervading the entire universe. All matter and energy in the universe are now understood to be just excitations of this one, all-pervading field.

Physics now has the capacity to describe accurately the sequence by which the unified field of natural law systematically gives rise, through its own self-interacting dynamics, to the diverse force and matter fields that constitute the universe. With a precision almost undreamed of a few years ago, the modern science of cosmology can now account for the exact sequence of dynamical symmetry breaking by which the unified field, the singularity at the moment of cosmogenesis, sequentially gave rise to the diverse force fields and matter fields. It is now possible to determine the time and sequence in which each force and matter field decoupled from the unified field, often to within a precision of minute fractions of a second. This gives us a clear understanding of how all aspects of the physical universe emerge from the unified field of natural law.

Mathematics, physiology, and other sciences have also located a unified source and basis of all the laws of nature in their respective disciplines. In mathematics, the foundational area of set theory provides an account of the sequential emergence of all of mathematics out of the single concept of a set and the relationship of set membership. The iterative mechanics of set formation at the foundation of set theory directly present the mechanics of an underlying unified field of intelligence that is self-sufficient, self-referral, and infinitely dynamic in its nature. Investigations into the foundations of set theory are ultimately investigations of this unified field of intelligence from which all diversity of the discipline emerge in a rigorous and sequential fashion. In physiology, it is the DNA molecule that contains, either explicitly or implicitly, the information specifying all structures and functions of the individual physiology. In this sense, therefore, it is DNA that unifies the discipline by serving as a unified source to which the diversity of physiological functioning can be traced.
Each of the modern sciences may indeed be said to have glimpsed a unified state of complete knowledge in which all laws of nature are contained in seed form. Each has gained some knowledge of how the unified field of natural law sequentially unfolds into the diverse expressions of natural law constituting its field of study. Modern science is now discovering and exploring the fundamental unity of all laws of nature.

**Maharishi Vedic Science**

Maharishi Vedic Science is based upon the ancient Vedic tradition of gaining knowledge through exploration of consciousness, developed by the great masters in the Himalayas who first expressed this knowledge and passed it on over many thousands of years in what is now the oldest continuous tradition of knowledge in existence. Maharishi’s work in founding Maharishi Vedic Science is very much steeped in that ancient tradition, but his work is also very much imbued with the spirit of modern science and shares its commitment to direct experience and empirical testing as the foundation and criterion of all knowledge. For this reason, and other reasons to be considered below, it is also appropriately called a science. The name “Maharishi Vedic Science” thus indicates both the ancient traditional origins of this body of knowledge and the modern commitment to experience, system, testability, and the demand that knowledge be useful in improving the quality of human life.

The founders of the ancient Vedic tradition discovered the capability of the human mind to settle into a state of deep silence while remaining awake, and therein to experience a completely unified, simple, and unbounded state of awareness, called pure consciousness, which is quite distinct from our ordinary waking, sleeping, or dreaming states of consciousness. In that deep silence, they discovered the capability of the mind to become identified with a boundless, all-pervading, unified field that is experienced as an eternal continuum underlying all existence. They gave expression to the self-sufficient, infinitely dynamic, self-interacting qualities of this unified state of awareness; and they articulated the dynamics by which it sequentially gives rise, through its own self-interacting dynamics, to the field of space-time geometry, and subsequently to all the distinct forms and phenomena that constitute the universe. They perceived the fine fabric of activity, as Maharishi explains it, through which this unity of pure consciousness, in the pro-
cess of knowing itself, gives rise sequentially to the diversity of natural law and ultimately to the whole of nature.

This experience was not, Maharishi asserts, on the level of thinking, or theoretical conjecture, or imagination, but on the level of direct experience, which is more vivid, distinct, clear, and orderly than sensory experience, perhaps much in the same way that Newton or Einstein, when they discovered the laws of universal gravitation or special relativity, enjoyed a vivid experience of sudden understanding or a kind of direct “insight” into these laws. The experience of the unified field of all the laws of nature appears to be a direct experience of this sort, except that it includes all laws of nature at one time as a unified totality at the basis of all existence—an experience obviously far outside the range of average waking state experience.

The ancient Vedic literature, as Maharishi interprets it, expresses, in the sequence of its flow and the structure of its organization, the sequence of the unfoldment of the diversity of all laws of nature out of the unified field of natural law. The Veda is thus to be understood as the sequential flow of this process of the oneness of pure consciousness giving rise to diversity; and Maharishi Vedic Science is to be understood as a body of knowledge based on the direct experience of the sequential unfoldment of the unified field into the diversity of nature. It is an account, according to Maharishi, of the origin of the universe from the unified field of natural law, an account that is open to verification through direct experience, and is thus to be understood as a systematic science.

These ancient seers of the Vedic tradition developed techniques to refine the human physiology so that it can produce this level of experience, techniques that were passed on over many generations, but were eventually lost. Maharishi’s revival and reinterpretation of ancient Vedic science is based on his revival of these techniques which have now been made widely accessible through the training of thousands of teachers of the Transcendental Meditation program. He has thus provided a reliable method of access to this field of direct experience where the oneness of pure consciousness gives rise to the diversity of the laws of nature; and he has also developed applications of this technology that render it open to experimental testing. These applications will be considered below.
Maharishi describes the experience of this unified field of consciousness as an experience of a completely unchanging, unbounded unity of consciousness, silently awake within itself. Gaining intimate familiarity with the silence of pure consciousness, Maharishi holds, one gains the ability to experience within that silence an eternal “fabric” or “blueprint” of all laws of nature that govern the universe, existing at the unmanifest basis of all existence. This unmanifest basis of life, where all laws of nature eternally reside in a collected unity, is experienced as the fabric of the silent field of consciousness itself, which is not in space and time, but lies at the unmanifest basis of all manifest activity in space and time. Through Maharishi’s work, this experience comes to be understood (as we see below) as a normal state of consciousness that arises in the natural course of human development.

Glimpses of this universal domain of experience, where all possibilities reside together in an eternally unified state, have been reported in almost every culture and historical epoch, from Plato to Plotinus and Augustine, and from Leibniz to Hegel and Whitehead. Scientists like Kepler, Descartes, Cantor, and Einstein also appear to have written of it and seemingly drew their insights into the laws of nature from this experience. Descartes (1908) writes, for example, of an experience that he had as a young man of “penetrating to the very heart of the kingdom of knowledge” and there comprehending all the sciences, not in sequence, but “all at once.” Scientists and writers from many traditions have described this experience of unity, which confirms that it is completely universal, and not a product of a particular cultural tradition. Just as the Vedic tradition has been misunderstood, however, so have those descriptions of consciousness found in these different cultural traditions; for without a technique that makes the experience systematically accessible to everyone, the understanding that this is a universal experience of the most fundamental level of nature’s activity has been obscured, and has not before now emerged into the light of universal science.

According to Maharishi Vedic Science, it is not only possible to gain direct experience of the unity of natural law at the basis of the manifest universe, but one can also directly experience the unity of nature sequentially giving rise to the diversity of natural law through its own self-interacting dynamics. Maharishi’s most recent research has
centered on delving deeply into the analysis of these self-interacting dynamics of consciousness.

**The Self-Interacting Dynamics of Consciousness**

When one gains the capability, through practice of the Transcendental Meditation technique, of remaining awake while becoming perfectly settled and still, one gains the ability to experience a completely simple, unified, undifferentiated, self-referral state of pure consciousness, which is called Saṁhitā in the Vedic literature, in which knower, known, and process of knowing are one and the same. Consciousness is simply awake to itself, knowing its own nature as simple, unified pure consciousness. Yet in knowing itself, the state of pure consciousness creates an intellectually conceived distinction between itself as knower, itself as known, and itself as process of knowing. In Vedic literature, this is reflected in the distinction between Ṛishi (knower), Devatā (process of knowing), and Chhandas (object of knowledge). According to Maharishi, from the various interactions and transformations of these three intellectually conceived values in the unified state of pure consciousness, all diverse forms of knowledge, all diverse laws of nature, and ultimately all diversity in material nature itself sequentially emerge.

The conscious mind, awake at this totally settled and still level of awareness, can witness the mechanics by which this diversification of the many out of the unity of pure consciousness takes place. The mechanics of Ṛishi, Devatā, and Chhandas transforming themselves into Saṁhitā, Saṁhitā transforming itself into Ṛishi, Devatā, and Chhandas, and Ṛishi, Devatā, and Chhandas transforming themselves into each other are the mechanics by which the unity of pure consciousness gives rise to the diversity of natural law. These mechanics are expressed in the sequential unfoldment of Vedic literature. These are the self-interacting dynamics of consciousness knowing itself, which, Maharishi asserts, sequentially give rise to all diversity in nature.

Maharishi (1986) describes this self-referral state of consciousness as the basis of all creative processes in nature:

This self-referral state of consciousness is that one element in nature on the ground of which the infinite variety of creation is continuously emerging, growing, and dissolving. The whole field of change emerges from this field of non-change, from this self-referral, immortal state of
consciousness. The interaction of the different intellectually conceived components of this unified self-referral state of consciousness is that all-powerful activity at the most elementary level of nature. That activity is responsible for the innumerable varieties of life in the world, the innumerable streams of intelligence in creation. (pp. 25–26)

The Structure of Maharishi Vedic Science
One of Maharishi’s most important contributions to Vedic scholarship has been his discovery of the Apaurusheya Bhashyā, the “uncreated commentary” of the Rk Veda, which brings to light the dynamics by which the Veda emerges sequentially from the self-interacting dynamics of consciousness. According to Maharishi’s analysis, the Veda unfolds through its own commentary on itself, through the sequential unfoldment, in different-sized packets of knowledge, of its own knowledge of itself. All knowledge of the Veda is contained implicitly even in the first syllable “Ak” of the Rk Veda, and each subsequent expression of knowledge elaborates the meaning inherent in that packet of knowledge through an expanded commentary. The phonology of that syllable, as analyzed by Maharishi, expresses the self-interacting dynamics of consciousness knowing itself. As pure consciousness interacts with itself, at every stage of creation a new level of wholeness emerges to express the same self-interacting dynamics of Rishi, Devatā, and Chhandas.

Thus the body of Vedic literature reflects, in its very organization and structure, the sequential emergence of all structures of natural law from the unity of pure consciousness. Each unit of Vedic literature—Rk Veda, Sāma Veda, Yajur-Veda, Atharva Veda, Upanishad, Āranyakas, Brāhmaṇa, Vedāṅga, Upāṅga, Itihās, Purāṇ, Smriti, and Upaveda—expresses one aspect or level of the process. As Maharishi (1986) describes it:

The whole of Vedic literature is beautifully organized in its sequential development to present complete knowledge of the reality at the unmanifest basis of creation and complete knowledge of all of its manifest values. (p. 28)

Veda, Maharishi asserts, is the self-interaction of consciousness that ultimately gives rise to the diversity of nature. The diversity of creation sequentially unfolding from the unity of consciousness is the result of
distinctions being created within the wholeness of consciousness, as consciousness knows itself. Thus from the perspective of Maharishi Vedic Science, the entire universe is just an expression of consciousness moving within itself. All activity in nature is just activity within the unchanging continuum of the wholeness of consciousness.

Through the texts of ancient Vedic science, as interpreted by Maharishi, we possess a rich account of the emergence of diversity out of the unity of natural law. On the basis of this account, it becomes feasible to compare the Vedic description of the origin of the universe with that of the modern sciences.

**Modern Science and Maharishi Vedic Science**

When Maharishi heard from major scientists of the recent advances of unified field theory in physics, he asserted that modern science had glimpsed the unified field described in ancient Vedic science. “The knowledge of the unified field,” he said (1986, p. 29), “has been discovered by modern science during just the last few years, but the complete knowledge of the unified field has always been available in the Vedic literature.” Modern science, he proposed, had now arrived at the edge of comprehending, through unified quantum field theories, what Vedic science had described on the basis of exploration of the least excited state of consciousness since ancient times: that all diversity in nature sequentially emerges from a unified source through a precise self-interacting dynamics. Modern experimental science and Maharishi Vedic Science could now be seen as two diverse yet mutually complementary approaches to knowing the same underlying reality—one through the empirical method, the other through the exploration of the least excited state of consciousness. Through Maharishi’s inspiration, this has become a major research program that has engaged the attention of many scientists and that has yielded very rich results.

Over the past decade, Maharishi has participated in numerous symposia with major scientists on the theme of exploring modern science and Vedic science to discover detailed structural similarities in their descriptions of the unified field. These symposia have attracted eminent unified field theorists, mathematicians, and physiologists, including a number of Nobel laureates, as well as many of the most highly recognized Pandits of the Vedic tradition. Out of these interactions has come
a meeting of two traditions, East and West, on the ground of their common theme: the investigation of the unified field. Those who have followed these symposia have recognized a deep and impressive structure of knowledge common to both traditions. Both identify a boundless, all-pervading field underlying all states of matter and energy in the universe; both locate it on the most fundamental time-distance scale of nature; both assign to it the same properties of self-sufficiency, self-interaction, infinite dynamism, unboundedness, and unity, among many other common attributes; both identify a threefold structure at the basis of all nature; and both describe a dynamics by which the diversity of nature sequentially emerges from this unified field according to precise laws. The result of these symposia has been that many scientists, following Maharishi’s lead, now feel confident to assert that the unified field described by physics and the unified field of consciousness described by Vedic science are one and the same.

In the first issue of Modern Science and Vedic Science, the lead article by John Hagelin explored many of the deep connections between contemporary unified field theory in physics and Maharishi Vedic Science from the standpoint of an active field theorist. His work brought these two diverse methods of inquiry into close relation, drawing upon both the latest developments of unified field theories and the direct experience of the unified field.

Dr. Hagelin presented evidence for Maharishi’s assertion that the unified field of consciousness and the unified field of physics are the same. His main empirical evidence for this new paradigm was drawn from experimental research in the social sciences on the “Maharishi Effect”—the measurable effects on society resulting from the practice of the Transcendental Meditation and TM-Sidhi programs, including Yogic Flying. As further evidence for the identity of consciousness and the unified field, he cited deep parallels between the descriptions of the unified field found in physics and Maharishi Vedic Science. These strikingly similar descriptions support the conclusion that modern science and Maharishi Vedic Science are two complementary methods of approach to the same underlying unity of nature.
The New Paradigm of the Unity of Nature

It is a common belief that the unified field of physics is an objective reality of nature and that consciousness is a subjective experience, and that the two belong, consequently, to different categories of existence. According to this understanding, one is purely material, the other is purely mental, and the two cannot, therefore, be equated.

Through the experience of pure consciousness described in Maharishi Vedic Science, that unified level of intelligence is experienced, not as a mere subjective and localized phenomenon of thought or sensation, but as a non-changing, unbounded field of Being, pervading all forms and phenomena in the universe on a non-active, silent, unmanifest level. Objective and subjective aspects of nature are seen as but two manifest modes of this unified field at the unmanifest basis of existence. A thorough examination of the nature of the unified field in physics and the descriptions of unbounded consciousness brought to light by Maharishi support the thesis that they are but two complementary modes of apprehending a single underlying reality.

The view of nature as consisting of billiard-ball-type objects, each separate, discrete, and isolated from the other, belongs to the old classical Newtonian view of the world. Quantum field theory in modern physics no longer views nature in this way, but provides a new understanding in which the primary reality is that of quantum fields. All forms of matter and energy are understood to be excitations of these underlying fields. In the last year and a half, the apparently different fields of gravity, electromagnetism, and the weak and strong interactions have been theoretically unified as different levels of expression of one single underlying field. All forms and phenomena in the universe are just modes of vibratory excitation of this one all-pervading unified field.

Today, the success of modern physics in unifying our understanding of physical nature is mirrored in the success of Maharishi Vedic Science in unifying our understanding of consciousness. When the unbounded level of pure consciousness is gained as a direct experience, all activity in nature is experienced as an excited state of that one all-pervading field. Since quantum field theory also describes all activity in the universe as excitations of one underlying field, the simplest interpretation is that there is a single unified field which can be known both
through direct experience and through the objective sciences. In this new understanding of the unity of nature, mind and matter cease to be viewed as ultimately different and come to be seen as expressions of a deeper unity of unbounded consciousness.

The unity of nature is not merely a hypothetical unity, nor a unity of intellectual understanding or interpretation. It is a unity of direct experience that has been described in almost every tradition and every historical epoch. Maharishi Vedic Science only brings to light what has been the experience of many of the greatest minds throughout history. What is radically new is that Maharishi has provided a systematic and reliable method by which anyone can gain access to this level of experience. This method of access is the Transcendental Meditation and TM-Sidhi programs, including Yogic Flying.

**The Transcendental Meditation and TM-Sidhi Programs, including Yogic Flying**

The Transcendental Meditation and TM-Sidhi programs, including Yogic Flying, have been introduced by Maharishi as an effective means for opening the unified field to all as a direct experience. In this way, the unified field becomes universally accessible to systematic exploration.

The key component of these programs is the Transcendental Meditation technique, which provides a systematic procedure by which the mind is allowed to settle naturally into a state of restful alertness, the self-referral state of pure consciousness, in which the mind is completely silent and yet awake. In this way, the state of pure consciousness, which has been the subject of philosophical speculation throughout the centuries, can now be investigated on the basis of direct experience. Maharishi’s immensely important contribution to the clarification and elucidation of this experience of pure consciousness will be a theme for analysis in future issues of this journal.

This quiet, still level of consciousness has rarely been experienced in the past because no systematic and effective technique has been available for providing that experience. The Transcendental Meditation technique is a simple, natural, and effortless procedure for allowing the awareness to settle into a state of deep silence while remaining awake. It has proved to be uniquely effective in making this level of experience widely accessible. Through the deep rest gained during the
practice of the technique, balance is systematically created on all levels of physiological functioning, and the nervous system is habituated to a more settled, coherent, and alert style of functioning. In time, a state of completely integrated functioning is gained, in which pure consciousness is spontaneously and permanently maintained. Once this state is established, the silent, self-referral field of awareness is always present as a stable, non-changing ground underlying all changing states of awareness. This integrated state of consciousness, Maharishi holds, is the basis of all excellence in life and provides the foundation for the further development of higher states of consciousness through the practice of the Transcendental Meditation and TM-Sidhi programs, including Yogic Flying.

Maharishi’s Programs for the Development of Higher States of Consciousness

The ultimate purpose of all aspects of the Transcendental Meditation and TM-Sidhi programs, including Yogic Flying, and Vedic Science is the development of consciousness, the unfoldment of the full human potential to live life in enlightenment. Enlightenment is that fully developed state of life in which one enjoys complete knowledge and lives in total fulfillment. In this state, one lives in harmony with all the laws of nature, enjoying the full support of natural law to achieve any desire without making mistakes.

Maharishi has identified a specific sequence of higher states of consciousness, each distinct from waking, dreaming, and sleeping, which, he asserts, arise in the normal full course of human development. Each state of consciousness unfolds on the basis of a concrete shift in the mode of the individual’s neurophysiological functioning. These states can be distinguished from waking, dreaming, and sleeping on the basis of their distinct physiological correlates. The higher states of consciousness that arise in this developmental sequence are, Maharishi asserts, a source of greater joy, knowledge, and fulfillment than ordinary waking state life.

The attainment of these higher states of consciousness is the basis for fully understanding and applying the theoretical assertions of Maharishi Vedic Science. Maharishi Vedic Science is just the exposition of the full range of direct experience that unfolds during the course of the natural
development of human consciousness. These states of consciousness are universal stages of human development accessible to everyone through the practice of Maharishi’s technologies of consciousness. What before was shrouded in the veil of mysticism is now scientifically understood as a normal, natural stage of human life available to anyone.

An article in the first issue of *Modern Science and Vedic Science*, by Dr. Charles Alexander and others (1987) examined the empirical evidence, drawn from behavioral and neurophysiological research, for the existence of these higher stages of human development. This article unfolded the scientific basis for understanding and verifying higher states of consciousness from the standpoint of a developmental psychologist, and laid the basis for a new paradigm of human development.

**Research on the Relation between Modern Science and Maharishi Vedic Science**

Each individual nervous system, when refined through Maharishi’s technologies of consciousness, is an instrument through which the silent field of pure unbounded consciousness becomes accessible as a field of inquiry. Since the unified field is all-pervading and everywhere the same, a nervous system finely enough attuned in its functioning can gain the ability, according to Maharishi, to experience and identify itself with that unbounded, undifferentiated, and unified field underlying all activity in nature. By taking one’s awareness from the gross level of sensory objects to perception of finer levels of activity, one gains the ability to experience that level of nature’s functioning at which the unity of pure consciousness gives rise to diversity. Gaining this unified state of consciousness is the means by which anyone can experience and confirm the structure of knowledge and reality described in Maharishi Vedic Science. This is partly what makes Maharishi Vedic Science a precise, verifiable science: All theoretical structures of the science can be verified through a reliable, systematic, effective technology. Other foundational aspects of this science will be considered below.

Maharishi’s technologies of consciousness become, in the modern world, a method for the investigation of the unified field and the most refined level of nature’s activity through direct experience. Modern physics, through its objective method of inquiry, has glimpsed a unified field underlying all of nature, but physics has reached a fundamental
impasse in its ability to experimentally investigate the unified field, because the energies required to probe these finer scales exceed those attainable by any conceivable particle accelerator technology. When physics can go no further, Maharishi’s technologies of consciousness, facilitate inquiry beyond the limitations of the objective approach by providing an effective means of exploring the unified field on the level of direct experience.

This exploration of the unified field through the subjective experience of consciousness is a well-structured program of research. It is guided by the knowledge of Maharishi Vedic Science set forth by Maharishi in conjunction with the modern sciences. When descriptions of the unified field from the standpoint of modern science, of Maharishi Vedic Science, and of direct experience coalesce, the three together provide a basis for complete knowledge. This program of research is based on Maharishi’s exposition of the Vedic literature as a complete and detailed expression of the unified field.

According to Maharishi’s exposition of the Veda, the sequential emergence of the diverse laws of nature from the unified field can be directly experienced in the field of consciousness as a sequence of sounds; these are presented in the sequential emergence of phonological structures of the Vedic texts. Veda is just the structure of the self-interacting dynamics through which the unified field gives rise to the diverse expressions of natural law. Fundamental theoretical concepts in physics and other disciplines, insofar as they are valid descriptions of nature, should therefore correspond to different aspects of Vedic literature that describe these realities from the standpoint of direct experience.

The basic program of research of modern science and Maharishi Vedic Science, as conceived by Maharishi, thus has three major goals: (1) to develop an integrated structure of knowledge by fathoming the depth of correspondence between the principles of modern science and Vedic Science; (2) to provide, from Maharishi Vedic Science, a foundation in direct experience for the most profound theoretical concepts of modern science; and (3) to resolve the impasse faced by the objective approach of modern science through the addition of the subjective approach of Maharishi Vedic Science, which provides complete knowledge of nature on the basis of the complete development of the knower.
In another issue of *Modern Science and Vedic Science* [see Vol. 5, Pt. 1 of this series], Dr. M.H. Weinless (1987) explored set theory and other foundational areas of modern mathematics in relation to Maharishi Vedic Science. In a proposed issue, Drs. R.K. Wallace, D.S. Pasco, and J.B. Fagan (1988) explore the fundamental relationship between Maharishi Vedic Science and the foundational areas of modern physiology, such as molecular biology. Their paper also discusses the extent to which fundamental principles of Maharishi Vedic Science can be used to further investigation of DNA structure and function.

The discovery of deep structures of knowledge and principles common to Maharishi Vedic Science and modern science represents such a profound contribution to our understanding of nature that this journal was founded to foster continued scholarly investigation of the interrelations between these complementary methods of gaining knowledge. Knowledge gained by direct experience of the fine fabrics of nature’s activity, and knowledge gained by the experimental methods of modern science coalesce in a new integrated method of inquiry that offers both the fundamental principles of modern science and the expressions of direct experience in Maharishi Vedic Science as two facets of one reality of nature’s functioning.

Maharishi (1986) sums up the relation between Maharishi Vedic Science, modern science, and his technologies of consciousness:

Maharishi Vedic Science is applied through the Technology of the Unified Field. We speak of the unified field in connection with Maharishi Vedic Science because of the similarity of what has been discovered by physics and what exists in the self-referral state of human consciousness. The Technology of the Unified Field [That is, Transcendental Meditation and TM-Sidhi programs, including Yogic Flying—Eds.] is a purely scientific procedure for the total development of the human psyche, the total development of the race. This is a time when objective, science-based progress in the world is being enriched by the possibility of total development of human life on earth, and this is the reason why we anticipate the creation of a unified field-based civilization. (p. 35)

On the basis of the universal availability of this domain of experience, an empirical science of consciousness becomes possible for the first time.
The Science of Creative Intelligence:
Foundations of a New Science of Consciousness

The unified science that links the objective method of modern science and the subjective method of Maharishi Vedic Science, while preserving the integrity of each, is called the Science of Creative Intelligence (SCI). Maharishi himself has laid the foundations of this new science by showing, first, how a precise subjective science of consciousness is established on the basis of the direct experience of consciousness in its pure form; and second, how the experimental method can be used to test empirically the assertions of the subjective science. Through Maharishi’s work, for the first time in history, the full potential of human consciousness can be investigated both through direct experience and through the objective methods of modern science. The foundations of this new science linking the subjective and objective method will now be considered.

Experiential Foundations

Prior to Maharishi’s work, the term consciousness was considered too vague and indefinite to be allowed into scientific discussion. It was excluded from science as a metaphysical term because consciousness was not objectively observable, and therefore apparently not amenable to scientific investigation. Through Maharishi’s work, the concept of consciousness has been given a precise, well-defined meaning on the basis of direct experience, and its relation to the objective framework of science has been precisely specified.

The experience of pure consciousness, available to anyone through regular practice of the Transcendental Meditation technique, is a basis for precise experiential knowledge of consciousness in its simplest, most fundamental, and most unified state. Even though consciousness can never be an object of experience, when the conscious mind becomes completely settled in a wakeful state, it experiences its own nature as pure wakefulness, pure consciousness, without any activity or objective content. Through the repeatable, systematic experience of this silent but wakeful state of mind, the concept of pure consciousness, which has been subject to conjecture and debate throughout the centuries, is now available to direct experience.
Having laid the basis for introducing consciousness into science as a precise concept, it remained for Maharishi to develop a program of applied research to test theoretical predictions of Maharishi Vedic Science. Identifying consciousness with the unified field provides a precise understanding of where consciousness is located in the framework of the sciences. To create an empirical science of consciousness, however, it was also necessary to account for how consciousness could be investigated through experimental research.

**Empirical Foundations**

Maharishi’s work has laid the foundation for an experimental investigation of consciousness. He has led the way in drawing out predictions of Vedic science that are open to testing, translating discussions of consciousness, derived from experience of higher states of consciousness, into predictions of experimentally observable phenomena. Three examples will illustrate this principle.

Pure consciousness, as was noted above, is experienced during the practice of the Transcendental Meditation technique as a state of pure restful alertness. This purely subjective experience does not, however, establish objectively whether it is in fact a state of deep rest and alertness, or only seems to be. If a person is in a deep state of rest and alertness, Maharishi has asserted, then physiological evidence of deep rest and alertness should be observable. Reduced levels of oxygen consumption, reduced breath rate, and other measures of more refined physiological activity would be predicted. Patterns of EEG coherence in the alpha range, indicative of restful alertness, should also be observed. Early pioneering research by Dr. R.K. Wallace (1986) found that these changes do indeed occur. In this way, statements about the subjective experience of consciousness were translated into empirically verifiable assertions. The basis of this correlation between consciousness and physiology is a principle, fundamental to Maharishi’s thinking, that for every state of consciousness there is a corresponding state of physiological functioning. The range of physiological correlates of the experience of pure consciousness is a subject of continuing research.

Consider a second example. Pure consciousness is understood in Maharishi Vedic Science as a clear and settled state of awareness. Anyone who gains this state is said to have a mind like a placid lake, unrippled
by waves, and thus able to reflect the world in a precise, non-agitated manner. Maharishi drew from this several predictions. One is that a person growing in the ability to experience pure consciousness would experience more stable and orderly physiological functioning. This can be translated into the testable prediction that subjects regularly practicing the Transcendental Meditation program display increased stability of the autonomic nervous system. Another prediction is that the practice of the Transcendental Meditation program will produce greater perceptual clarity and greater orderliness of thinking. Translated into specific terms, this leads to the prediction that practicing the Transcendental Meditation program will produce measurable increases on such scales as auditory discrimination, brain wave coherence, and problem solving ability. Research has been designed, carried out, and reported in the literature which measures the growth of these parameters in groups practicing the Transcendental Meditation program by comparison to control groups, thus providing objective verification of the predicted correlates of the subjective experience of pure consciousness.

A third example of how assertions of Maharishi Vedic Science can be translated into testable form is found in the sociological experiments on the Transcendental Meditation and TM-Sidhi programs, including Yogic Flying. The hypothesis is that a group of people practicing this technology in one place, by bringing their awareness to the level of perfect orderliness in the unified field, will enliven qualities of harmony and orderliness in collective consciousness, thus producing measurable positive changes in the quality of societal life. Many experiments have been designed by Maharishi and carried out, demonstrating the power of this technology to produce significant changes in the level of coherence, positivity, balance, and stability in society, even on a global scale. (See Experimental Research, below.) The results of these experiments strongly support Maharishi’s assertion that consciousness is identical with the unified field.

Experimental Research
Over 600 hundred experimental studies in the areas of physiology, psychology, and sociology provide substantial confirmation of many basic assertions of Maharishi Vedic Science in the arena of empirical science. Many of these studies, now published in major scientific jour-
nals throughout the world, have been collected in the volumes called Scientific Research on the Transcendental Meditation Programme: Collected Papers, Vols. 1–6 (1977–1991). This research provides experimental validation of the efficacy of the Transcendental Meditation and TM-Sidhi programs, including Yogic Flying. Because this research—from over 600 scientific studies at over 300 universities and research institutions in 33 countries, published in more than 100 scientific journals—is too extensive to summarize here, the reader is referred to the Collected Papers for articles cited in this and other professional journals. Overall, this research probably represents the most concerted, well-designed research program on a potential means to benefit mankind ever conceived. Its present standing is that, taken together as a body of research, it is one of the most impressive confirmations of a theory of human potential ever executed.

Although it is beyond the scope of this introduction to go into the details of this research, it is worthwhile to mention some of the broad categories of scientific investigation that have evolved to guide the research program of the Science of Creative Intelligence. The main areas of research include studies on the individual and society. Research on benefits to the individual may be further subdivided into studies of physiological changes (both during and after the practice); cognitive, psychological, and behavioral changes; benefits to health and social behavior; and benefits to athletic performance, performance in business, and academic performance. Research on social benefits through collective practice may be further grouped into research on families, city populations, national populations, and global population. These research studies fall into the categories of crime prevention, accident prevention, benefits to economy, health, violence reduction, and world peace.

On the basis of this research, basic assertions of Maharishi Vedic Science become verifiable through empirical science. There is, moreover, a unity of theory underlying these diverse predictions and tests. These studies, taken as a whole, constitute a coherent research program that tests the prediction that repeated experience of the unified field results in greater orderliness, coherence, and positivity, in both individual and social life. Research on these changes not only tests fundamental theory, but demonstrates the practical benefits of this new
technology. Maharishi’s technologies of consciousness become open to experimental testing precisely because they have significant practical applications in improving every area of human life.

**Practical Applications of the Transcendental Meditation and TM-Sidhi Programs, including Yogic Flying**

Maharishi has frequently asserted that the purpose of Maharishi Vedic Science is to benefit life, not merely to give knowledge for its own sake. Knowledge, he holds, is for action, action for achievement, and achievement for fulfillment. The ultimate purpose of Maharishi Vedic Science and its applied technology is, therefore, to bring human life to fulfillment.

Maharishi’s technologies of consciousness bring fulfillment to individual life by unfolding the full potential of consciousness. When higher states of consciousness are realized, Maharishi emphasized, life is lived in “twenty-four-hour bliss.” Gaining contact with the unified field, one enjoys spontaneous right action, lives life in total accord with all the laws of nature, and accomplishes any life-supporting desire. Violations of natural law cease, and all suffering, which is caused by violation of natural law, comes to an end. Life is lived free from mistakes, in inner and outer fulfillment. Such is the fundamental purpose of the technologies Maharishi has created.

**Perfect Health**

Maharishi’s technologies of consciousness have important practical applications in the area of health. According to Maharishi, sickness arises from imbalance. Perfect health means wholeness, balance on all levels of life. When individual life is established in the unified field of all the laws of nature, all actions are spontaneously in accord with natural law. In terms of physiological functioning, this means perfect integration and balance, from the biochemical and molecular levels to the macroscopic, organismic levels.

Maharishi Ayurveda is an integral part of Maharishi Vedic Science. It is a revitalized form of the ancient ayurvedic science of life and health, restored to its original purity and effectiveness by Maharishi.
According to Maharishi, the cornerstone of Ayurveda is the development of consciousness. Perfect health in mind, body, and behavior is the result of perfect balance in consciousness and physiology. This develops through the Transcendental Meditation and TM-Sidhi programs, including Yogic Flying, when the mind identifies itself with the unified field, the field of perfect balance and wholeness.

Maharishi Ayurveda combines Maharishi’s technologies of consciousness with specific procedures to treat and prevent illness and promote longevity. Maharishi Ayurveda Medical Centers have been established in many countries to eliminate the basis of sickness, create perfect health, and reverse the aging process. Over the last fifteen years, research into the effects of Maharishi’s technologies of consciousness, on health have been carried out at research institutions all over the world, and Maharishi’s recent emphasis on Ayurveda provides many new research opportunities for investigating the applications of Vedic Science in the area of health.

Maharishi’s technologies of consciousness also include technologies to accomplish specific goals of individual and social life. The TM–Sidhi program has been founded by Maharishi to utilize the knowledge and the organizing power of the unified field for improving achievements in every area of human endeavor.

**Unfolding Full Human Potential through the Transcendental Meditation and TM–Sidhi programs**

When one gains the level of experience of the self-interacting dynamics of consciousness, Maharishi holds, one gains command over all the laws of nature. Stationed at the source of all the laws of nature, at the “central switchboard” of nature’s activity, human consciousness can command all the laws of nature to create any desirable effect in the material world. Maharishi has brought forth a program for gaining mastery over all the laws of nature, based on the formulations found in the ancient Yoga Sūtras of Patanjali, one of the principal books of Vedic literature. This is the TM–Sidhi program, in which the mind gains the ability to function from the level of the self-interacting dynamics of the unified field. Once established in pure self-referral awareness through the practice of the Transcendental Meditation program, an individual
gains the ability to draw upon the organizing power of the unified field to accomplish anything. Since the unified field is the source of all existence, its organizing power is infinite, and one who functions from this level has unlimited organizing ability. Established in that unified field of all possibilities on the unmanifest level of existence before consciousness assumes the form of matter, all possibilities open to one’s awareness and one can govern the expressions of the unified field as it transforms itself into matter. As Maharishi (1986) expresses it:

In this program, human awareness identifies itself with that most powerful level of nature’s functioning and starts to function from there. The purpose of the TM-Sidhi program is to consciously create activity from that level from where nature performs. (p. 74)

Through the practice of the TM-Sidhi program, Maharishi predicts, it will become possible to achieve levels of body-mind coordination hitherto deemed impossible. It will be possible, he asserts, to realize the ancient dream of flying through the air, and to develop highly enhanced powers of hearing, seeing, and intuition that extend the senses far beyond the limits currently conceived to be possible. In the Yogic Flying technique, which Maharishi developed from the Yoga Sūtras, the silent state of self-referral consciousness is integrated most fully with outer activity as the body lifts in spontaneous hops, generating inner bliss and maximum coherence in brain functioning. Other Vedic texts describe the ability to move through the air at will as a result of perfection of this Yogic Flying technique. By activating laws of nature that are now hidden to ordinary methods of scientific investigation, the TM-Sidhi program provides a research methodology to explore what is possible for mankind to achieve on the basis of functioning from that level where the conscious mind has become identified with the unified field. This is the basis of a technological revolution more powerful and beneficial to life than any conceived through empirical science.

The Maharishi Effect
The TM-Sidhi program, when practiced in groups, is even more powerful than the TM-Sidhi program practiced alone. The collective practice of the TM-Sidhi program can produce an influence that affects the entire world in measurable ways. This global influence of coherence
generated through the group practice of the Transcendental Meditation and TM-Sidhi programs, including Yogic Flying, has been called the “Maharishi Effect.”

As early as 1960, Maharishi predicted that when individuals practice the Transcendental Meditation and TM-Sidhi programs in sufficiently large groups, a measurable increase in orderliness, coherence, and positive trends would be observed in society. By enlivening the life-supporting and evolutionary qualities of the unified field, such as perfect orderliness, infinite dynamism, and self-sufficiency, Maharishi held, these qualities would be enlivened in collective consciousness and this would have positive, measurable effects on a wide social scale.

Over the years, social scientists developed formulas for predicting the size of the group necessary to create a “phase transition” in society to a measurably higher quality of life. These formulas, calculated on the basis of analogous phase transitions, from disorder to orderliness, studied in physics, came out to be approximately one percent of a population practicing the Transcendental Meditation program, and a much smaller percentage, on the order of the square root of one percent, practicing the TM-Sidhi program.

Since 1978, many experimental studies have been performed to measure the effect of large groups practicing the TM-Sidhi program. Experimental confirmation of the principle has been the consistent result. The Maharishi Effect is now as well documented as any principle of modern social science. In creating this technology, Maharishi has provided an effective method of social change that operates from the silent, harmonizing level of the unified field to produce a transformation in the quality of collective consciousness, thereby effortlessly creating coherence on a global scale. Maharishi (1986) describes how this effect is produced:

The transcendental level of nature’s functioning is the level of infinite correlation. When the group awareness is brought in attunement with that level, then a very intensified influence of coherence radiates, and a great richness is created. Infinite correlation is a quality of the transcendental level of nature’s functioning from where orderliness governs the universe. (p. 75)

D. W. Orme-Johnson and M. C. Dillbeck (1987) have summarized the empirical research on the Maharishi Effect. They surveyed
experimental studies documenting the sociological improvements resulting from the group practice of the TM-Sidhi program. Based on these results Maharishi asserts that the collective practice of the TM-Sidhi program in groups of 8000 (the square root of one percent of the world’s population) would produce coherence in the collective consciousness of the entire world. Statistically significant reductions in crime, accidents, fatalities, and disease, and other positive benefits on a global scale observed during experimental periods have established this as an effective means of changing collective consciousness and thereby changing the quality of life in the world—simply by enlivening the source of order and coherence at the basis of nature, from the level of the unified field.

**Maharishi’s Program to Create World Peace**

The most dramatic application of the Transcendental Meditation and TM-Sidhi programs, including Yogic Flying, is Maharishi’s program to create world peace through the creation of a permanent group of 8000 collectively practicing Maharishi’s technologies of consciousness. These technologies are a basis for eliminating negativity and destructive tendencies throughout the world. Large groups of experts in the TM-Sidhi program, creating coherence, during experimental periods, have provided ample opportunity for scientific research. During these experimental periods, conflict and violence have been reduced in war-torn areas and negative trends have been reversed. Over thirty studies have established the efficacy of this technology to eliminate conflict and promote life-supporting, positive trends throughout the world.

Maharishi clearly lays out the basis of his program to create world peace. Stress, he holds, is the basic cause of all negativity, violence, terrorism, and national and international conflicts. Stress generated by the violation of natural law causes strained trends and tendencies in the environment. Through the Transcendental Meditation and TM-Sidhi programs, including Yogic Flying, human intelligence can be identified with the unified field, and violations of natural law will cease. “Reinforcement of evolutionary power in world consciousness is the only effective way,” Maharishi holds, “to neutralize all kinds of negative
trends in the world and maintain world consciousness on a high level of purity” (*Maharishi’s Program to Create World Peace*, 1986, p. 7).

The global applications of this new science and technology are almost beyond present levels of imagination. Yet scientific research has found measurable reductions in levels of violence, crime, and other indications of negativity during the practice of the TM-Sidhi program in sufficiently large groups during experimental trial periods. Here for the first time in history is a scientific basis for creating world peace, ending terrorism, and reducing the negative trends of society.

On the basis of these studies, Maharishi holds that world peace can be guaranteed now, within a few years, through the establishment of groups of 8000; he holds that perfect health and unlimited longevity can be achieved for individual life, and that balance, coherence and health in society can be established in our generation. War, crime, poverty, and all problems that bring unhappiness to the family of man can be entirely eliminated. Life, he holds, can be lived in absolute abundance and fulfillment. Maharishi has called upon every significant individual in the world to act now to adopt this program for world peace by creating groups of 8000 collectively practicing the Transcendental Meditation and TM-Sidhi programs, including Yogic Flying, to establish world peace and guarantee its perpetuation.

The practical benefits that Maharishi foresees through these new technologies are far greater than those achieved by the technology based on present science. As science has investigated deeper levels of nature, from microbes to molecules to atoms, new technologies have emerged which apply the knowledge in areas such as medicine and nuclear power. In drawing upon the deepest and most powerful level of natural law, the level of the unified field, Maharishi Vedic Science lays the basis for much more powerful technologies still. Where modern medicine has been able to eliminate some diseases by drawing upon microscopic levels, Maharishi Vedic Science lays the basis for the elimination of all disease, and more importantly, for the creation of perfect health and reversal of aging. While modern science has produced nuclear technology but no technology for peaceful resolution of conflict, Maharishi Vedic Science draws upon the infinite organizing power of the unified field at the basis of nature to create social harmony.
and world peace while preserving cultural integrity and stimulating prosperity and progress.

**Maharishi’s Technologies of Consciousness as a New Method of Gaining Knowledge**

The bold assertions about what is practically possible through the application of Maharishi’s technologies of consciousness must be understood in the context of the new method of gaining knowledge that Maharishi has founded. The history of science testifies that as new methods of gaining knowledge of deeper and more unified levels of natural law become available, more powerful and useful technologies become available. Maharishi’s technologies of consciousness are based on the deepest and most unified level of knowledge of nature. It should not be surprising, therefore, that this technology provides a radically new source of organizing power to fulfill the highest goals of mankind.

These technologies of consciousness offer a fundamentally new approach to knowledge that has not been available before. In asserting that it is possible for one individual to know all the laws of nature and the entirety of the universe within his or her own consciousness, Maharishi is well aware that he is introducing an account of human potential that goes well beyond the concept of the limits of knowledge that has dominated in the scientific era. This new paradigm of knowledge must be examined in a new light.

It is a widespread belief in the modern age that the only valid method of gaining knowledge is by moving outward through the senses, that is, through the methods of the empirical sciences. It is, however, only the historical failure of subjective approaches that has led to this belief. It cannot be thought that the senses are the only way of gaining knowledge, and those who cling to the belief that it is, only allow old habits to stand in the way of exploring new possible sources of knowledge.

Subjective approaches to knowledge in the past failed to bear fruit because they failed to provide an effective and reliable method of access to an invariant and universal domain of direct experience. They thus failed to establish independent standards of knowledge, they failed to produce methods of distinguishing truth from error, they failed to produce consensus even among those practicing the same method, and
they failed to produce practical technological benefits through the practice of the method.

Maharishi’s technologies of consciousness are different from subjective approaches in the past, and must therefore be considered on separate grounds. They provide an effective, reliable method of opening the mind to an invariant and universal level of nature which is everywhere, and yet not ordinarily open to experience because the mind usually functions on more active levels. By providing a technology to make this non-active level of nature available as a direct experience, Maharishi has made this domain available to all as a new field of inquiry; and, where there is a new source of experience of something universal, unchanging, and objectively verifiable, a new source of knowledge is available.

The Science of Creative Intelligence gives a new account of how complete knowledge is possible. When the mind becomes completely settled and still, according to this account, it gains the ability to perceive on the most refined levels of nature’s functioning—the all-pervading unified field where all laws reside in a collective totality. It not only experiences this unified field, it becomes identified with it; it is the unified field and thus knows the unified field as its own universal Self. On this level of knowledge, there is no separation of knower from the known. Nothing lies outside the range of the knower. All laws of nature and everything in the universe can be known as intimately as one’s own Self. Mind and body cease to be seen as separate realities. Maharishi (1986) says:

In reality our self-referral state of consciousness is the unified field—not an object of knowledge as a rose is when we say, “I see that rose.” The unified field is not an object in this way; it is the subject itself. The unified field is a self-referral state of awareness that knows itself, and in knowing itself is the knower and the known, both together. (p. 96)

On this account, there is no distinction between the knower and the reality that it knows. Since it is the Self that knows itself, there is nothing ultimately outside the consciousness of the knower, and there are therefore no limits on what can be known. [This unbounded value of the Self is written with an uppercase “S” to distinguish it from the ordinary, localized self we typically experience.] If true, this account of knowledge provides a fundamentally new source of discovery of the
laws of nature, like the empirical sciences, in that it relies on experience as a source of knowledge, but distinct from these sciences in that it draws upon a wider range of experience. As a new source of discovery, it extends the power of scientific investigation; yet it remains within the scope of empirical science by being subject to procedures of objective verification.

**Maharishi University of Management**

Maharishi University of Management, formerly Maharishi International University, was founded by Maharishi in 1971, based on the principles of the Science of Creative Intelligence. One of the major functions of this University is to show how each discipline and each level of natural law arises from the unified field of pure consciousness. The specialty of Maharishi University of Management is the knowledge of the unified field of pure consciousness from the standpoint of each academic discipline. At Maharishi University of Management, each modern discipline traces the diversity of laws back to a unified source in the unified field of pure consciousness and shows how the diversity of laws emerge from this unified field through the self-interacting dynamics of consciousness. Just as physics and mathematics have discovered increasingly unified levels of natural law at the basis of their discipline, thus tracing the diversity of its laws to their source in the unified field, so every academic discipline can ultimately show how its laws derive sequentially from the unified field. This project of unification of knowledge, a long sought goal throughout Western intellectual history, is now being systematically pursued and completed at Maharishi University of Management.

This enterprise includes developing charts to show how each modern discipline arises from the unified field of pure consciousness. For each discipline, a Unified Field Chart has been constructed to show how the discipline sequentially emerges from the unified field through the self-interacting dynamics of knower, known, and process of knowing. These Unified Field Charts constitute a major unification of knowledge, showing at a glance how all the diversity of knowledge emerges from a unified source.

Since the unified field is understood as a field of consciousness, and consciousness is the most fundamental level of each student’s own Self,
the study of the unified field at Maharishi University of Management constitutes a method of systematically relating all knowledge to the student’s Self. The success of Maharishi University of Management’s Consciousness-Based education is due in part to this program of relating all knowledge to the unified field and the unified field to the Self. Because all students and faculty at Maharishi University of Management collectively practice the Transcendental Meditation technique, regularly gaining the direct experience of the unified field of pure consciousness, this unified field increasingly becomes a living reality. This unified field ceases to be an abstract concept and becomes as intimate as the Self. The experience of faculty and students has been that learning and inquiry is joyful and most fulfilling in this environment of Consciousness-Based education.

[The reader is referred to other issues of the journal Modern Science and Vedic Science as well as to other volumes in this book series Consciousness-Based Education: A Foundation for Teaching and Learning in the Academic Disciplines for articles illustrating how Maharishi Vedic Science is transforming our understanding of modern academic disciplines. —Eds.]

Maharishi’s Work in Historical Perspective:
An Appreciation

Maharishi has created a major watershed in world intellectual history. He has laid the foundation for a fundamental change both in intellectual history and in the history of technology and civilization itself. His work has created a new paradigm of the unity of human knowledge, and, we may expect, will unify the sciences and humanities in a more integrated way than ever before. He has, moreover, brought to an end the old notion that man is born to suffer and that life is a struggle. The practical programs he has founded provide a scientifically validated basis for reducing and even eliminating crime, war, terrorism, poverty, and other problems that beset mankind; more importantly, his discoveries make it possible to live life in the fulfillment of pure knowledge and permanent bliss consciousness and to achieve the highest goals of human endeavor. He has laid the basis for a new civilization, founded on new principles of complete, reliable, useful, fulfilling knowledge—
the knowledge of the unified field of pure consciousness as the perfectly orderly, unified source of nature.

Maharishi is unique in the world today. He has not offered conjectures and hypotheses about reality and human potential, nor does he set himself up as a final authority on matters of knowledge when he speaks rather of experience as the ultimate basis of knowledge. The experience of which he has spoken is derived from a new source, from the level of fully developed human life gained when one’s awareness is open to the unified field of pure consciousness. Maharishi’s life is an example of that which he taught. Unlike those whose teaching is based solely on the personal authority of the individual, Maharishi has founded universities, sciences, technologies, and other institutions based on universal principles through which any individual can gain the direct experience of the fully unfolded nature of life and validate the truth of what is described in the science. Because of this, Maharishi is held in highest esteem by millions of people around the world.

Maharishi has provided the means of unfolding the dormant creative genius within everyone, and he has established institutions through which the knowledge of how to unfold this potential will be perpetuated generation after generation. He has, moreover, used this knowledge to found programs to create perfect health, progress, prosperity, and permanent peace for the world—programs to end suffering and allow life to be lived in spontaneous accord with natural law. These institutions are not just ideals, but functioning institutions whose practical achievements are now well documented and available for all to examine.

Everyone now has the ability, with the availability of the Transcendental Meditation and TM-Sidhi programs, including Yogic Flying, to engage in this great experiment of identifying one’s awareness with the total potential of natural law and to spontaneously live in accord with all the laws of nature while established in the awareness of the unified field of pure consciousness. The experience of approximately three million people who have learned the Transcendental Meditation technique testifies to its practicality and its effortlessness and ease of practice. Experimental studies have shown that its benefits are real and concrete. On this basis, Maharishi has foreseen the creation of a new era of civilization—Heaven on Earth—in which life will be lived
in fullness and abundance without suffering. Maharishi’s work eliminates the very basis of stress and suffering and lays the ground for a new civilization, a unified field-based, ideal civilization that draws on the infinite organizing power of the unified field of pure consciousness to bring human life to fulfillment.

References


Maharishi’s program to create world peace: Removing the basis of terrorism and war. (1986). Washington, DC: Age of Enlightenment Press.


Wallace, R. K., Orme-Johnson, D. W., & Dillbeck, M. C. (Eds.).


Kenneth Chandler’s “Modern Science Vedic Science: An Introduction,” here revised/updated, was originally published in *Modern Science and Vedic Science, 1(2)*, p. v–xxvi. It is reprinted with permission of the publisher.
Electronic Resources and Publications

LINKS

Education

Maharishi University of Management: www.mum.edu
Maharishi School of the Age of Enlightenment:
   www.maharishischooliowa.org
Maharishi’s Consciousness-Based Education: www.CBEprograms.org
International Foundation of Consciousness-Based Education:
   www.CBEfoundation@ifcbe.org
David Lynch Foundation for Consciousness-Based Education and
World Peace: www.davidlynchfoundation.org

Transcendental Meditation Program

Maharishi’s Technologies of Consciousness: www.tm.org
Maharishi Channel: www.maharishichannel.in
Maharishi Lectures and Interviews (film clips): www.tm.org/maharishi
Invincible America Assembly: www.invincibleamerica.org
Global Country of World Peace: www.globalcountry.org
Global Good News Site: www.globalgoodnews.com
Fortune Creating Homes: www.FortuneCreatingHomes.com
Sthapati Veda: www.sthapathyaveda.com

Research

Center for Brain, Consciousness, and Cognition: www.drfredtravis.com
Truth about TM: www.truthabouttm.org

PHONE NUMBERS

1-888-LEARN TM (1-888-532-7686)
Maharishi University of Management (1-641-472-7000)
PUBLICATIONS

These publications are available from Maharishi University of Management Press: http://mumpress.com and at the MUM Bookstore.

Books by Maharishi Mahesh Yogi

Science of Being and Art of Living
Bhagavad-Gita: A New Translation and Commentary, Chapters 1–6
Celebrating Perfection of Education
Celebrating Perfection in Administration
Vedic Knowledge for Everyone
Inaugurating Maharishi Vedic University

Consciousness-Based Books Imprint from MUM Press

The series Consciousness-Based Education: A Foundation for Teaching and Learning in the Academic Disciplines contains 12 volumes, available in 2011.

Maharishi Vedic Science
Physiology and Health
Mathematics
Art
Government
Sustainable Living
Education
Physics
Literature
Management
Computer Science
World Peace

Each volume includes a paper introducing the Consciousness-Based understanding of the discipline and a Unified Field Chart that conceptually maps all branches of the discipline, illustrating how the discipline emerges from the field of pure consciousness, the Self of every individual. These charts connect the “parts” of knowledge to the “wholeness” of knowledge and the wholeness of knowledge to the Self of the student.

Subsequent papers show how a Consciousness-Based approach may be applied in various branches of the discipline; these papers include occasional examples of student work. Each volume ends with an appendix describing Maharishi Vedic Science and Technologies of Consciousness in detail.