

Senior Integration Paper Final Draft
Noel Weichbrodt
200311.13
Computer Science and Philosophy Reconsidered

Abstract

Computer Science, Philosophy, and Christianity are three disciplines that are commonly presumed to be mutually antithetical to each other. However, by investigating how we use each of these disciplines, there turns out to be a broad and necessary connection between them, due to the person of Jesus Christ. Computer science, a servant discipline, needs to be given meaning and direction, while Philosophy, a meta-discipline, needs grounding. Both find their redemption and fulfillment in the God and life of the Bible. The Christian faith makes one a better computer scientist and philosopher.

1.0.0 Definitions of Computer Science and Philosophy. Both disciplines are broad, and encompass a host of activities.

In the fall semester of my junior undergraduate year, I was taking two classes back to back: “ICS400: Data Structures and Algorithms” and “PHI204: Continental Philosophy”. They met in different buildings on campus, and that walk from the liberal arts building to the science building was the only time my mind could take a break from thinking about them. If the men (yes, mostly men) I studied knew how they were muddled together, Derrida with QuickSort, or Bill Joy and James Gosling with *hexaity*, I am sure they would combust in a conflagration of indignation. However, these things were not physically embodied in people, but thoughts and abstractions playing around inside my head.

Surprisingly, the two classes not only made peace with each other, but started affecting how I thought in each of them. Connections that no one else could make wired through my head during Continental, and sometimes poststructuralism made more sense when applied to vectors than to texts. Philosophy and Computer Science suddenly looked like happy bedfellows, and not quarreling extended family. That comfortable union surprised people, who often stated “Wow, those two majors have nothing in common!” What are the natures of these two disciplines, and what in them causes those unexpected connections that clanged around in my brain?

1.1.0 Computer Science is thinking about computation, what is possible, impossible, and may yet be possible. Computer Science also creates and maintains the computational infrastructure needed to carry out computation.

Computation, determining something by mathematical or logical methods, is what computers do, and what one part of Computer Science studies. As a base example of what computation is, consider “ $19 + 23$ ”. This is a computational command: take the union of the sets $\{19\}$, $\{23\}$; or, take the numbers 19 and 23 and perform the additive operation on them. Another example comes from the metamathematical Axiom of Choice—given the set of the subsets of all positive integers, choose a member from each set. We could write an algorithm that

recursively selects the smallest number in each set, and adds that number to a new set that it returns. We could run these computations through in our mind, a difficult but not undoable proposition. However, why spend the effort when at our command are computational devices whose very purpose is to assist and shorten the time it takes to solve both of these problems?

1.1.1 Computer Science explores the limits of what can and cannot be done with computation, and serves humanity, embodied as ‘users’, by understanding and mastering their algorithmic computation needs.

Both of the examples above, “19 + 23” and “find the smallest number in each subset of the set of positive integers”, can be computed. However, there are some things that cannot be computed. Some ideas might spring to your mind, but here is an example that is provably uncomputable. Suppose we want to do the same computation as described above, except on all the subsets of the set of positive real numbers. This computation results in an infinite loop during the first iteration: the smallest number between, say, the subset (0,1) will always be one smaller than the one picked. If the result is, say, .0000001, it’s trivial to come up with a smaller number, like .00000001. Perhaps there are other classes of things that cannot be computed—the android Data in *Star Trek: The Next Generation* is incapable of love because “it does not compute”.

So there are certain things that are computable, and other things that are not. Computer Science explores both sides, actively figuring out new things that are (or in some cases are not) computable. Recently, computer scientists found ways to computationally model certain behaviors of ants and bees. [Resnick 1994] And Deep Fritz attempts with much success to compute the best possible move on a given chess board. [Slashdot.org 2003] As yet, there is no way to computationally write a novel, or to fall in love. This exploration of well-defined procedures for solving a problem in a finite number of steps, the exploration of algorithms, is the vital core of what computer science is and does.

1.1.2 Computer Science creates and maintains hardware and software in a circular relationship between that infrastructure and the user’s computational needs.

Not only is computer science concerned with the ideas and abilities of computation, but it also covers the implementation of systems with the ability to compute. The computational infrastructure, which consists of software and hardware, is created by computer scientists. Hardware is what most folk normally think about when they think of ‘computers’—processors, memory, input and output devices. Each of those things must be designed, manufactured, and maintained. There’s also a “soft” side to the infrastructure—the software. Software is the structure which bridges the divide between the way humans think and the way computers compute.

The process of hardware/software creation and maintenance has an asymmetrical relationship to the user’s computational demands. Many times, hardware is made and software is written that does not meet a specific need, but rather creates the need for new computational ability. For example, the doubling of processor speed every eighteen months, known as Moore’s Law, is not driven by a specific programming need. Rather the hardware engineers at Intel and other places create processing power, and the software companies create programs that utilize the newly available processing power. The demand is created by the supply, as programmers dream up new things to make computers do that were not previously possible. Note that though the overarching drive in computer science is service, a significant portion of how computer science serves is through pursuing its own interests and research programs that often have no immediately known application to the end-user. There is a symbiosis between the theoretical and the practical sides of computer science, and a vital linkage between computational ability and computational use.

1.2.0 Philosophy is thinking. Notoriously slippery when asked to give a definition of itself, philosophy ranges over many realms. Perhaps the love of wisdom, or the analysis of thinking about sundry things.

Philosophy majors at most undergraduate institutions are characterized by a certain esoteric flair, a dysfunctional drive to gain the upper hand in arguments and confuse their peers with well-timed, ill-formed, but conversation-stopping questions and comments. This observation leads to a negative formulation of what philosophy

is: useless wranglings over ineffables and trifles. But since my degree is also in this area, I will give a defense against that definition, and maintain that philosophy is a vital, if necessarily ill-defined, discipline.

Specifically here, philosophy analyzes and abstracts the stuff of our thoughts about reality. Broken down etymologically, philosophy is rooted in “love” (from the Greek *filos*) and “wisdom” (from the Greek *sophia*). Often and rightly, that is taken together and made into “the love of wisdom”. But perhaps we should look at philosophy in a different way: as the discipline of combining love and wisdom into the world.

1.2.1 Philosophy involves the analysis of our thinking about disciplines.

One of the reasons undergraduate philosophy majors have such a bad reputation is that they push their philosophic noses in other undergrad’s business, claiming to know what is going on in each area of study better than the other students themselves. Critiquing by declaring to know the ultimate agendas and the full relativity of the discipline at stake, the philosophy majors maddeningly claim a special insight into the way things are, solely from their own area of study. Again, a negative formulation of what philosophy does, but the truth behind this is that philosophy analyzes thinking, and as such analyzes the different kinds of thinking that go on in every other discipline, from art to sociology to physics to computer science.

What does this analyzing and abstracting do? The idea of analysis is to break apart a thought, concept, what-have-you, into constituent parts, and then break it apart again, and so on until a satisfactory understanding of the fundamental ideas behind every phenomena is reached. Understanding the interrelatedness of everything involved, finding connections, determining the ground and motive for each thought, these are the entrees that philosophy serves up. Abstraction reduces the thought to a general case, stripping away what is judged accidental or nonessential and driving to the essence or core of just what a thought or thing is.

1.2.2 Philosophy involves talking about things on a meta-, or meta-meta-, level.

Philosophy is also confusing due to its propensity to discuss things on meta-levels. Swiftly, a meta-level discussion becomes quite confusing to follow and to

figure out. But oftentimes philosophy involves discussing how, say, science *does* science, or how language *means*. These things move discussion up from the level of talking about something (often more concrete) to talking about the talk about something (which gets more abstract). Philosophy can take these discussions up many levels. For example, the philosophy of language studies and thinks about how language creates and destroys possible meanings, while knowing full well that the whole discussion of this topic is done under the auspices and constraints of the language under discussion. It is for this reason that Heidegger [Heidegger 1928] and Derrida [Derrida 1998] become at times incomprehensible. They are attempting to talk past what the actual language means. If these are opaque waters, take heart—we are now discussing how philosophy discusses things, a meta-level, philosophic discussion itself!

2.0.0 A redefinition of the relationship between Computer Science, Philosophy, and my Christian faith.

Students of philosophy or computer science may have noticed something in the above descriptions of the other discipline that related strongly to something in their discipline. This is intentional, because there is a naturally strong relationship between computer science and philosophy. A list of specific areas would never be exhaustive—logic, languages, intelligence, isomorphisms, ethics, incompleteness, and meaning might start the process of generating such an infinite list. But there is further discussion to have about these disciplines, beyond their general overlap. The Christian faith, as lived and believed, is wrapped up in these disciplines, and in their areas of overlap.

Of course, this is not a testable proposition—though there are many books that mix philosophy and computer science, there are none that claim the link between a Christian faith and the two disciplines is anything stronger than ancillary. This is fine—any good computer scientist or philosopher will tell you that there are things that are not empirically testable but still necessary and true. That the Christian faith—bound up in a N:1 relationship between man and Jesus—is tied to the disciplines that its believers participate in is an article of, again, faith and not proof. It is not the intent of this paper to give conclusive proof that computer science and philosophy are

inexorably tied together, nor is it to show that the Christian faith is the only way (though faithful folks would not prefer any other way) to guide and ground philosophy and computer science.

With that disclaimer, there is indeed much mixing in the life of a practitioner. The best integration and proof of Christ's preeminence in these disciplines is the faithful life of a computer scientist and/or philosopher. In answer to the question "Please tell us how in the world a scientific or at least technical mind can believe in God, and what role religion has played in your work on Perl..." the computer scientist Larry Wall responded, "Well, hmm, that's a topic for an entire essay, or a book, or a life..." [Slashdot.org 2002] The proclamation, starting from the cultural mandate of the Old Testament in Genesis 1:26 [see Appendix 0] and moving on to Jesus' demand for priority in work and the rest of life in the New Testament Gospels and finally resting in the reality of God's upholding and sustaining the work of man is chiefly seen not in paper abstracts but in the lives of faithful followers of Jesus. Integration of this sort may be explained in academic essays, but it is only experienced and shown true by living it out as faithful service to God.

2.1.0 Computer Science is a servant discipline

All this talk has been revolving around the idea of "service", the act characteristic of servants. Computer science is perhaps best characterized by this notion, making it a servant discipline. [Hogg 2003] Just like regular servants faithfully come up with ways to provide and care for their master's needs, computer science serves other disciplines by providing for their computational needs and caring for their computational infrastructure. Harry Plantiga recognizes this, and proposes love and joy as the expressions of this service.

We are in the image of a creative God, and we also love to create. There is a joy in creating something beautiful and useful out of the raw material that God provides. Love for God and neighbor is expressed in creating systems that solve problems, meet needs, and build up the church. And a unique power of this tool is that programs can be copied at almost no cost. Serving millions of people is as easy as serving one. [Plantiga undated]

There is definitely an element of art, of originality and creativity, in this service, and that will be discussed. But mainly, computer science serves others.

Computer science also reflects what other disciplines consider important. Computers themselves are of course a chief object of study in computer science, but that study will always serve to further other, outside ends. For example, big computer clusters are inherently interesting entities to computer scientists. But it is big science research, like nanoscale electronics, that pushes behind the creation and use of the hardware and software used in clusters. While there are introverted motives in computer science, like math and other disciplines, the final use of computer science is inexorably tied to the real world.

2.1.1 Computer Science involves art (Knuth's Art of Computer Programming), a crafty and joyful creation and refinement. Computer Science involves math (Knuth again), a logical, disciplined, rigorous way of thought.

Like mathematical intuition, computer science combines a creative art and a logically rigorous discipline. The dual nature of computer science can be bifurcated many ways. There is art, and there is math. There is application and theory. Harry Plantiga dichotomizes it this way:

Computer science is a discipline with two aspects. On the one side it is an *engineering* discipline: it involves the planning, design, construction, and maintenance of computer systems...it is also a *science* in the sense that mathematics is a science. It is the study of computation and computability, the study of *algorithm*. It does not frequently use the hypothesis-test method of studying nature, but it does involve the study of nature. [Plantiga undated]

From one side of the influences we get books like Donald Knuth's *Art of Computer Programming*, [Knuth 1973] which finds that the best computer science is regulated by the liberal principles of art, not hard rules. Notice that art is used in two senses here: art as the joyful creation and refinement of ideas, and art in an older sense as a crafty practice that requires study and apprenticeship. Art may be seen as governing human-computer interaction, from the coding of programs to the design of circuits.

The mathematical side of computer science is what studies things like Turing's

Halting Problem, and Big-O notation. Mathematics underlies the logic of computation, and so from the ground up computer science is a mathematical discipline. The rigor of mathematical thinking influences the way computer scientists solve problems, while the procedural, iterative nature of mathematics give rise to the step-by-step nature of computer programs. Of course, any mention of iteration will remind a computer scientist of its more abstract and creative brother recursion, the other approach to algorithmic programming. There is a tension between the approaches to thought embodied in iteration versus recursion that illustrates the tension between art and math in the discipline as a whole.

2.1.2 Ethics in Computer Science

As a servant discipline, external interests and needs dictate computer science's research. Ethically this is important because there must be judgments made within computer science regarding the pursuit and use of this research. That computer science always involves making ethical judgments is not a dogmatic or loaded view, but simply a pragmatic acknowledgment of the way that computer scientists actually work. This means that the judgments made daily by computer scientists necessarily need grounding beyond the weak "code of ethics" propounded by the major professional associations, because they include both the actions of the individual computer scientist, as well as the overall actions of a research group or organization.

Ethical action is vital to the way computer science is done. Like Superman, computer science has to use its powers for good and not for evil. This, actually, is quite a minority view in computer science at large. Hacking, file-sharing, encryption, and closed-source reverse-engineering are all areas of research that are currently judged by our common law code to be wrong, but are still carried out with complete disregard for their rightness or wrongness. Not that these areas are morally wrong—in fact there are good arguments to be made that they are all morally permissible—but there is a complete refusal to acknowledge research as having an ethical dimension. This silence must not be kept.

2.2.0 Philosophy is a meta-discipline

Philosophy subsumes, as discussed above, other disciplines' discussions into its

own discussion. This act of creating higher-level dialogue helps distinguish philosophy from other disciplines. Philosophy as a meta-discipline extends its tentacles outside the instance of a specific discipline domain and subverts the normative discussion in ways not usually done within the discipline.

Do not take from all this the idea that philosophy is itself a meta-narrative, or an arrogant discipline that seeks to force everyone to explain their lives and realities on its terms. Though that can be the case with bad practice, philosophy can also be practiced humbly, with concern for the correct application and use of its deconstructive and revealing powers.

2.2.1 Philosophy concerns itself with thinking in abstractions, and also with the implementation details.

By creating abstractions of reality, philosophy works in the same fashion as a computer scientist seeking to model reality on a computer. Abstractions create maps of reality, or reduce a process to its significant components. Both of these ideas are familiar to any software engineer, who must take a business request or goal and abstract the computational tasks from it. The computer cannot model the whole world just to figure out a specific detail of it, and philosophy cannot consider the world as a whole when it seeks to analyze a specific issue. What the philosopher and the computer scientist deems significant in their abstracting process is dependent on their chosen narrative for their work, and life.

By working out the details of a certain claim or narrative, philosophy digs into the implementation details much like a programmer works out exactly how a project will be implemented on specific hardware. The process of finding counterexamples, for instance, is analogous to the testing phase of the software creation life cycle. Choosing the best way of implementing the theory or program is, again, dependent on the practitioner's chosen narrative, since the process of choice is value-laden and presuppositional.

2.2.2 Philosophy plays with structure, language, and logic.

Derrida is famous for saying that language is play. "Play is always play of presence and absence." [Derrida 2001 208] Philosophy as a whole plays with

structure, language, and logic much like a child plays with colored blocks and Legos. Philosophy analyzes the structure of things (e.g. scientific revolutions, or the way the world is ontologically ordered) and then plays with what it finds by removing and adding features, bringing out different aspects of the subject, and exposing the subject's foundations. With language, philosophy examines how power interacts with our words, how words both describe and prescribe, and what limits language brings to thought and action. Logic permeates the discipline, with logical objects like Pareto-rationality and higher-order modal systems, so much so that logic is an accepted sub-discipline within philosophy.

2.3.0 Christ is the architectonic for both of these.

By examining what computer science and philosophy are, and what they do, there seems to appear an architectonic need for something else beyond the disciplines themselves. That something is Christ, who is actively sustaining and upholding our practice of these disciplines. There are things in each of these disciplines that point to something more, a further reality that both grounds and directs them.

2.3.1 Computer Science needs a purpose and direction for its program, which cannot exist in an ethical vacuum. Christ provides the ethics and the imperative for a good Computer Science.

How computer science chooses and carries out its projects does not, as previously stated, exist in an ethical vacuum. There must be an ethical system that guides the discipline, and the ethics known as Christian virtue ethics [see MacIntyre 1984 for an introduction and defense of virtue ethics] is the only one that can give the full guidance and grounding needed.

An imperative is also needed for computer science—why does it serve? When the computer scientist wakes up in the morning, what notion save a transcendent imperative can get her out of bed? In a Christian system, the service that computer science provides is highly valued. The computer scientist is simply following Christ's command to serve one another [c.f. Zaleski 1997, 120], and a servant's heart and mind is most assuredly a worthy calling. Not only does computer science serve, but it

also fulfills the creation mandate given in Genesis 1:27 [see Appendix 0] to rule and subdue the earth. By modeling and taming computationally complex processes, computer science contributes to our ability to wisely control and steward the world.

Joel Adams notes,

“If computing is indeed a part of God’s creation, then men and women who work to increase mankind’s understanding and mastery of computation (i.e., computer scientists) are obeying God’s command with respect to this part of His creation.” [Adams 2002]

Finally, by thinking abstractly about problems using logic and math, computer science explores God’s immanent methods of creation and sustenance.

2.3.2 Philosophy needs grounding and upholding that only Christ can give.

The philosopher makes choices too, in abstracting and implementing and in how philosophy is carried out. There is no way to trust philosophy unless it is fully grounded in a specific ethical, metaphysical, and epistemological narrative, and even then the narrative itself is not fully trustworthy unless it derives its legitimacy from transcendent revelation. What to leave out and what to keep, what to say and left unsaid, are chaotically complex choices that no algorithm or formula can make, and their ground must be in Christ, for no other foundation can take the messiness of reality and faithfully uphold it.

3.0.0 General application of my worldview to Computer Science & Philosophy

That is the general outline of how the Christian worldview applies to computer science and philosophy. But as said earlier, the application and integration is best seen in the lived life. So, I want to present this final section with a bit more boldness, but in a humble way. I think that I can become a good computer scientist and philosopher by following the path of Jesus, and I can see a few examples of older practitioners who have successfully done so.

3.1.0 My Christianity does not conflict with my life as a Computer Scientist or as a Philosopher. Christianity makes me a better Philosopher and Computer Scientist, and the practice of Philosophy and Computer Science make me a better Christian.

My Christianity, far from conflicting with the life of science or philosophy,

makes me better in them. The Christian way of life provides the ethical and stochastic guidance for dilemmas in computer science while imbuing the work with meaning. Philosophy is grounded in both its day-to-day life and in its conceptual work.

3.1.1 Because Christ is my hope and glory, I can pursue my disciplines wholeheartedly while avoiding the emptiness and skewed perspective that intense involvement can cause.

In any intensive discipline, burnout and a loss of meaning are ever-present enemies to the involved practitioner. Imagine a computer scientist who has lost interest in working on morally strong projects due to her loss of commitments to the whole enterprise of researching in computer science. By grounding motivation and pleasure in the transcendent reality of the God of the Bible, Christianity keeps its follower from burnout and loss of meaning. The life of the philosopher is easily swept up in the grand ideas, the esoteric examination of abstractions and counterfactuals. On top of this, any student of philosophy must wrestle with questions regarding their life's ultimate meaning, questions that cannot be easily answered in any fashion, and are especially problematic without divine intervention. The Christian faith faithfully lived grounds the life of the believer in both of these senses.

3.1.2 The insights and modes of thinking I learn in Philosophy and Computer Science deepen my faith, and my wonder at God and his creations.

The study of philosophy continually pushes the student back to the basic tenets of Christianity. A well-practiced discipline is a mark of a faithful Christian. Applying the analytical and critical thinking skills to the study of the Bible and to living the life commanded therein increases obedience and wisdom. The joy of thinking, displayed in Christians from Augustine to Kepler to Kierkegaard to Knuth, invokes wonder at God and His creations.

3.2.0 Modern Computer Scientists and Philosophers who are Christians, and what they have said regarding the integration of their faith and their disciplines.

There are faithful lives that follow the above path and display the fruits of a well-lived life in their disciplines of computer science and philosophy. Donald Knuth, mentioned above, is perhaps the foremost Christian intellectual in computer science.

His *Art of Programming* series is the most well-respected work on algorithms and programs by broad consensus. Larry Wall created the popular computer language Perl, and delights in using biblical principles to lead the wild-and-woolly open-source community that has sprouted around his language. And Alvin Plantiga, called by W.V.O. Quine the smartest student ever to pass under his tutelage, combines a mesmerizing rhetorical style with whip-smart analytic skill to defend a full-feathered Reformed Christian faith in philosophy.

3.2.1 Don Knuth explores algorithms as a way of seeing God's creativity and glory. Larry Wall creates and guides a computer language that strives to serve its users.

Don Knuth delights in exploring algorithms and programming. He finds the study of the mathematical side of computer science to be a reflection of God's delight in creation. The delight is two-way, too: Knuth himself finds that his study of things like finite numbers to influence the way he approaches an infinite God. [Knuth 2003] Knuth's humility is reflected in his outstanding offer of \$2.53 for every error found in his published material. [Knuth undated]

Larry Wall created and now guides the popular scripting language Perl, which he describes as "the first postmodern computer language." [Leonard 1998] Perl has been infused with certain principles by Larry, principles that reflect Larry's commitments to living out Christ's preeminence in every part of life. Wall says that "Perl always has been, and always will be (I hope) a *humble* language" [Slashdot.org 2002] Perl's motto is "easy things should be easy, hard things should be possible." [Technomaniesto.org 2003] Larry refuses to endorse any one style of Perl programming as better than another, but rather focuses on what the intent of the program is. And Larry, unlike most other open-source community leaders, refuses to kick out a trouble-raising programmer in all but the most extreme cases, instead preferring to keep the programmer close to him and working through whatever issues are raised. His tolerance stems from his belief that "The theological notion that we are each individually valuable in the sight of God puts a different viewpoint on what humans really want and how they should treat each other." [Wall 1999]

3.2.2 Alvin Plantiga is a rigorous, renown philosopher and unapologetic Christian in

whom the two are tied extraordinarily close.

Alvin Plantinga's most recent work, *Warranted Christian Belief*, is perhaps the most important breakthrough work in epistemology from the last twenty years. It is also the best full-throated defense of a complete version of Christianity to be published since C.S. Lewis' *Mere Christianity*. With work this academically respectable and intellectually important, there is proof that faith and philosophy can be successfully integrated.

3.3.0 Conclusion

The disciplines of Philosophy and Computer Science are defined, and explained. The relationship between the two of them, and Christianity, is shown. Christianity is maintained to provide not only the vital grounding and imperative for both of them, but also to provide the believer with the ability to be a better practitioner due to her faith commitments. Finally, the way the Christian faith has influenced three famous men in these disciplines is examined.

Appendix: Full Quotations of Biblical Citations Used

Genesis 1:26-30

Then God said, “Let us make man in our image, after our likeness. And let them have dominion over the fish of the sea and over the birds of the heavens and over the livestock and over all the earth and over every creeping thing that creeps on the earth.” So God created man in his own image,

in the image of God he created him;

male and female he created them.

And God blessed them. And God said to them, “Be fruitful and multiply and fill the earth and subdue it and have dominion over the fish of the sea and over the birds of the heavens and over every living thing that moves on the earth.” And God said, “Behold, I have given you every plant yielding seed that is on the face of all the earth, and every tree with seed in its fruit. You shall have them for food. And to every beast of the earth and to every bird of the heavens and to everything that creeps on the earth, everything that has the breath of life, I have given every green plant for food.” And it was so. And God saw everything that he had made, and behold, it was very good. And there was evening and there was morning, the sixth day.

Bibliography

Adams, Joel C.:

–“Computing Technology: Created, Fallen, In Need Of Redemption?”, unpublished manuscript, available at [http://Computer Science.calvin.edu/Computer Science/Xian/](http://ComputerScience.calvin.edu/ComputerScience/Xian/).

–”Why Christians Should Study Computer Science (and other technical disciplines)”, unpublished manuscript, available at [http://Computer Science.calvin.edu/Computer Science/Xian/](http://ComputerScience.calvin.edu/ComputerScience/Xian/), 2002.

Borgmann, Albert: *Power Failure: Christianity in the Culture of Technology*, Brazos Press, 2003.

Derrida, Jacques: *Of Grammatology*, English translation by Spivak, Johns Hopkins Univeristy Press, 1998.

–“Structure, Sign and Play in the Discourse of the Human Sciences.” *Modern Literary Theory*, 4th Edition, edited by Rice and Waugh, Oxford University Press, 2001.

Heidegger, Martin: *Being and Time*, first published 1928. English translation by Macquarrie and Robinson, Harper SanFrancisco, 1962.

Hogg, Matt: “Senior Integration Paper”, unpublished manuscript, Covenant College, May 2003.

Knuth, Donald:

–*The Art of Computer Programming, Vol. 1*, Addison-Wesley, 1973.

–*Things a Computer Scientist Rarely Talks About*, CSLI Publications, 2003.

–Personal website, available at

<http://www-cs-faculty.stanford.edu/~knuth/books.html>, visited 18 November 2003.

Leonard, Andrew: “The Joy of Perl”, *Salon.com*, available at

http://archive.salon.com/21st/feature/1998/10/cov_13feature.html, 1998.

McGrath, Alister: *Science & Religion: An Introduction*, Blackwell, 1999.

MacIntyre, Alasdair: *After Virtue: A Study In Moral Theory*, University of Notre Dame

Press, 1984.

Plantinga, Harry: "Christianity and Computer Science at Calvin College", unpublished manuscript, available at <http://cs.calvin.edu/CS/Xian/plantinga.htm>.

Slashdot.org: "Kasparov Wins Game 3 Against X3D Fritz", available at <http://games.slashdot.org/article.pl?sid=03/11/16/2320211&mode=thread&tid>

=

[127&tid=186](http://games.slashdot.org/article.pl?sid=03/11/16/2320211&mode=thread&tid), visited 2003.11.18.

—"Larry Wall On Perl, Religion, and..." , available at

<http://interviews.slashdot.org/interviews/02/09/06/1343222.shtml?tid=145>, visited 18 November 2003.

Technomanifesto.org: "Perl", available at

<http://www.technomanifestos.net/index.pl?Perl>, visited 18 November 2003.

Resnick, Mitchel: *Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds*, MIT, 1994.

Vander Liden, Keith: "Computing in the Image of God", unpublished manuscript, available at <http://Computer Science.calvin.edu/CS/Xian/>, October 2002.

Wall, Larry: "Divine Intervention: An interview with Larry Wall", available at <http://www.techgnosis.com/wall1.html>, originally published in *Feed*, 10 February 1999.

Zaleski, Jeff: *The Soul of Cyberspace: How New Technology is Changing Our Spiritual Lives*, HarperEdge, 1997.