Programming Reversible Computers

Ph.D. Thesis Proposal

(Concise Version for Submission)
$Revision: 2.8 .

Michael P. Frank
MIT AI Lab, Rm. 747
545 Technology Sq.
Cambridge, MA 02139
`http://www.ai.mit.edu/~mpf`

Started August 27, 1996.
Concise version started December 5, 1996.
Formatted on December 20, 1996.
Revision checkpoint on
$Date: 1996/12/20 17:37:03 .

A current version and the earlier,
more verbose version are available online at
`http://www.ai.mit.edu/~mpf/proposal/proposal.html`

Abstract

Reversibility offers tantalizing prospects for enabling computations to take place in a denser, faster, and more energy efficient manner by making the total heat production arbitrarily small. Unfortunately, known general-purpose reversible computing techniques can only reduce energy dissipation by a small factor before incurring prohibitive asymptotic increases in the space and/or time complexity of algorithms. Therefore it is still an open question whether the impact of reversibility on the limits of general-purpose computing can ever be very significant.

I propose, for my Ph.D. thesis research, to delve into this question, and study in detail the relationship between the efficiency of a computation, and the amount of irreversible information destruction it performs. This investigation will involve both theoretical and experimental work.

On the theoretical side, I propose to formalize a new measure of computational complexity in terms of the amount of information that a computer erases, and determine where existing algorithms studied by computer scientists fall on this spectrum. I will attempt to develop efficient reversible versions of these algorithms. Some particular algorithms can be made reversible without affecting their asymptotic efficiency; I will describe these algorithms and attempt to characterize how this sort of transformation might be applied more generally. Finally, I will study how the efficiency of algorithms is affected if only partial reversibility is required.

On the experimental side, I propose to develop new abstractions, programming techniques, languages, and compilers that will allow programmers to easily write partially or fully reversible programs that are as space and time efficient as can be provided for, given the best known theoretical results to date. To the degree that time permits, I will write and compile example programs using this system, simulate the programs' execution on a reversible computer architecture, and experimentally measure the programs' efficiency and degree of reversibility, to validate the theoretical predictions.

By this work, I hope to show that a requirement for reversibility is not actually prohibitive for general-purpose computation, or at least, to show that the amount of information erasure in a computation can be greatly reduced (if not eliminated) without severely affecting its efficiency. If successful, this work will provide a crucial foundation for the programming of possible future generations of extremely high density computers, by allowing them to be energy-efficient enough to operate at high speeds without suffering severe cooling problems.

The proposed nominal period of research is one year, from December 1996 to December 1997. Out of this, I propose to spend approximately three months on theoretical work, five months on experimental work, and four months writing and revising the thesis. The period of research may be extended by approximately one semester, if necessary.

1 Motivation for Reversible Computing

For several decades, researchers studying the physics of computation have known that physics, as it is currently understood, sets no fundamental lower bound on the energy expenditure required for computation, in the case where the computation is performed reversibly, *i.e.*, without discarding information.

If, on the other hand, a bit of information is thrown away (meaning that a computational state is reached that could have been reached from either of two preceding states), then the second law of thermodynamics requires an unavoidable energy dissipation of $k_B T \ln 2$, where k_B is Boltzmann's constant and T is the temperature of the system. Current digital circuit technologies actually dissipate around six orders of magnitude more energy than this minimum whenever a bit is erased, because bit erasure is done by dumping large numbers of electrons from a high-voltage circuit node down to ground. But even this much larger energy dissipation can be avoided if the computation is done reversibly, using adiabatic circuit techniques such as SCRL [7].

The energy dissipated when bits are erased becomes a serious problem as we extrapolate current trends in computer technology to ever-higher component densities and speeds. If every logic element erases a bit (namely, its previous output) on every cycle, as typically occurs in current digital circuits, the rate of heat production will eventually exceed the heat flow capacity of the computing material, leading to high temperatures and thermal noise that can prevent circuits from functioning properly. Active cooling, *e.g.*, with coolant flowing through channels in the computing material, will eventually be required, leading to severe logistical problems of its own. Moreover, in mobile computing devices, the total power and/or energy available for computation is usually limited, and will thus restrict the total rate and/or amount of computation that the device can perform.

Reversibility offers a way to deal with these problems. If we can avoid discarding most or all of the information that current computers throw away, we can reduce the energy that is dissipated to heat, allowing faster computation with a given rate of cooling, or more computation for a given supply of energy.

Full reversibility also offers potential benefits unrelated to energy usage, namely, for detecting errors, preventing tampering, and diagnosis of software problems. I will discuss these other benefits in more detail in the thesis, but not in this short proposal.

2 The Efficiency of Reversible Computing

Theorists have shown that reversible computers can do anything that normal computers can do, via the simple expedient of saving all the information (such as intermediate results) that a normal computer would throw away ([1], [5]). Unfortunately, this approach simply transforms the problem of how to get rid of unwanted heat into the problem of how to get rid of unwanted information. The memory capacity of any computer is limited, just like its heat capacity, and actively sending all unwanted information outside the computer presents logistical problems analogous to those involved in active cooling. Therefore, we would prefer to compute in a way that produces less of this "garbage data" in the first place.

Bennett [2] has shown that any algorithm can be transformed into a form that produces garbage data at a rate that decreases steadily as the algorithm runs, so that the total accumulation of garbage data is only a logarithmic function of run time, and is thus reasonable to store. However, Bennett's technique suffers from the drawback that the run time t of the algorithm is multiplied by a polynomial factor of $O(t^{\epsilon})$; ϵ can be made arbitrarily small, but only at the cost of exponentially increasing space [4].

2.1 Important Open Questions

Other techniques besides the above are known for transforming arbitrary irreversible algorithms into reversible ones, but all known techniques involve prohibitive asymptotic increases in either time complexity, space complexity, or both. However, no one has proven that such inefficiencies are actually required to compute reversibly. Moreover, no one has even thoroughly investigated the question of whether we can compute efficiently if we relax the reversibility constraint a little bit, and merely require that very little information (rather than none at all) be discarded. This suggests several very interesting open questions:

1. Can we prove whether full reversibility actually requires greater asymptotic (ignoring constant factors) space and/or run time on some classes of problems?

2. Can we find and characterize other classes of problems or algorithms in which full reversibility does *not* decrease efficiency?

3. Can we describe how to reduce the amount of information erasure needed for a computation by arbitrarily large finite factors, without incurring asymptotically decreased efficiency? How quickly does efficiency fall off as erasure decreases?

4. Can we organize and facilitate the programming process so that the theoretical techniques for providing reversibility (to various degrees) can be applied to make it practical to actually write fairly efficient programs for reversible computers?

The first question is extremely difficult, since proving it would involve showing that no reversible algorithm, out of the enormous space of all possible algorithms, could solve a problem with a certain efficiency; this sort of proof has historically been very difficult for complexity theorists to come up with. (Witness the unresolved status of the P = NP question.) I am sure I cannot do it, but it is an interesting question to raise.

The second question might be more approachable, since I am aware of certain classes of algorithms that can be transformed into a reversible form without increasing either their asymptotic time or space complexity. However, this does not rule out the existence of a better irreversible algorithm for solving the same problem. Still, it would be of interest to know what features of a given algorithm allow it to be made reversible without incurring inefficiency.

I believe the third question can be answered straightforwardly, using a restricted version of Bennett's full-reversibility transformation. Other techniques such as traditional data compression might be used to improve the answer even further.

The fourth question is practical rather than theoretical, and is very straightforward since it is mostly involves just some software engineering design work, rather than proving new and difficult theorems.

These questions can be summed up as a single question: "Can we make it practical to program computers to compute highly efficiently while requirements for reversibility become more stringent, up to and including the limit of full reversibility?" This question will be crucially important if we ever wish our computing technology to achieve extremely high computational speeds and densities, without being limited by the need to export to the outside world all the unwanted intermediate results that are generated, in some form such as heat.

The thesis will explore this question, and the various components of it that are listed above.

3 Research Plan

The thesis work can be subdivided into two major parts: the theoretical work, and the practical work. I will start with the theoretical work, since in the course of it I might discover some new techniques and algorithms that could be applied in the practical work.

3.1 Theory of Reversible Computational Complexity

The aim of the theoretical work will be to develop the beginnings of a theoretical study of computational complexity that accounts for reversibility, and measures it by counting the amount of information that is erased during a given procedure (this is zero for a fully-reversible computation). We might call this the "thermodynamic complexity" of an algorithm, as opposed to its time or space complexity. Because of the $k_B T \ln 2$ energy dissipation per bit erased, the thermodynamic complexity of an algorithm is the fundamental measure of the minimum amount of energy required to execute the algorithm.

As is usual with computational complexity, we will characterize thermodynamic complexity in a way that ignores constant factors, since the exact number of bits erased internally during a particular operation are assumed to vary depending on the particular computer architecture being used. However, we will assume that the architecture provides a universal set of reversible computational primitives which all incur a thermodynamic cost of zero, that is, they are implemented in a way that does not erase any bits internally. The Pendulum architecture designed by Carlin Vieri [6] is an example of such an architecture. In fact, Pendulum does not provide any irreversible primitives, and is not meant to be used to program partially-reversible algorithms. In contrast, we will assume an architecture that also provides an "ERASE" instruction that erases a fixed-size unit of information at a thermodynamic cost that is bounded by a machine-specific constant. The zero-cost reversible primitives, together with the ERASE instruction, constitute a sufficient language for describing arbitrary algorithms and analyzing their thermodynamic complexity.

My plan for my study of thermodynamic complexity is as follows:

1. For a number of practical and interesting algorithms from traditional computer science (e.g., from [3]), such as algorithms for sorting, searching, and optimization problems, I will determine the algorithms' thermodynamic complexity when they are translated to the form of reversible primitives plus ERASE in the most straightforward way. I expect most of the traditional algorithms will constantly perform irreversible operations, and thus have thermodynamic complexity equal to their time complexity.

2. I will attempt to find ways to modify the existing algorithms in ways that reduce their thermodynamic complexity without increasing their asymptotic time or space complexity, or else increasing it as little as possible.

3. I already know some examples of algorithms that can be transformed into an equally-efficient reversible form. I will describe these algorithms and attempt to characterize more generally what it is about these algorithms that allows this transformation to be done, and apply that characterization to try to translate other algorithms to an efficient reversible form.

4. I will study and describe the existing very general techniques such as Bennett's for translating arbitrary algorithms into a less-efficient but reversible form. I will attempt to improve on these techniques, but I do not anticipate finding a better transformation for achieving total reversibility. However, I will describe variations on these techniques that achieve greater space and run-time efficiency by incurring a non-zero, but small, cost in thermodynamic complexity.

I plan to spend only three months, January through March of 1997, exploring these issues. Some of the theoretical questions I am asking are very difficult, and so I will not expect to be able to answer all of them to my satisfaction. It will be necessary to cut short the time spent on them, in order to avoid the risk of using up all the time available for the thesis on what might be an impossibly difficult task that may produce no new hard results.

However, I expect that even if all I can do is to formalize the questions, and describe particular examples of efficient reversible algorithms, it will still be worth the effort, and may provide important insights necessary for more far-reaching future work.

3.2 Programming Reversible Computers

The theoretical study of the efficiency of reversible computing is certainly very interesting, but it's difficult to make progress on.

In contrast, the accompanying area, of how to apply the known theory to actually program reversible computers effectively, is ripe for advancement. My plan for my work in this area will involve the following steps, in approximately the following order:

1. Define a high-level programming language that leverages off of appropriate new reversible programming abstractions in order to make it easy to describe algorithms having arbitrary degrees of reversibility, up to and including full reversibility.

2. Demonstrate the expressiveness of the language by writing programs in it that implement the partially-to-fully reversible algorithms that will have already been studied in the theoretical work.

3. Write a compiler that can translate programs in the high-level language into instructions for a processor architecture that supports reversible computing, such as Pendulum or something like it.

4. Enable the compiler to optionally apply its own transformations for automatically reducing the thermodynamic complexity of the program as part of the compilation process. These transformations might include Bennett's technique, traditional data compression, or any new techniques that I might have discovered in my theoretical work. The user of the compiler should be able to specify how he wishes to trade off thermodynamic complexity versus space and time complexity.

5. Compile my various example programs using a variety of different automatic transformations for reducing thermodynamic complexity.

6. Execute the compiled programs on a simulated reversible processor. Check the correctness of the results to validate the compiler. Measure the actual time, space, and thermodynamic complexities of the executions to verify the theoretical predictions.

7. Compare the thermodynamic, time, and space efficiency of programs that are hand-designed to be highly reversible against that of nominally irreversible programs whose thermodynamic efficiency is optimized by the compiler using the general, automatic techniques.

I intend to spend approximately five months, from April through August of 1997, on this experimental work. If it seems to be moving too slowly, I can always limit the ambitiousness of the compiler, reduce the number of example programs being explored, or extend the period of research a few months. Since this experimental side of the research mainly consists of just software design and programming, as opposed to an all-or-nothing struggle to prove a possibly false theorem, I have high confidence in the success of the work, so there is less risk involved in extending the amount of time spent on it, if necessary. However, I would very much like to wrap it up by, at latest, the end of the year, if not in August as currently planned.

3.3 Thesis Writeup

Finally, after the above-mentioned theoretical studies and experimental work have been completed, I will spend the Fall semester writing up the results in the thesis document. In addition to relating the work done as described above, the thesis document will, as time permits, discuss a variety of related issues, many of which were mentioned in the earlier, longer version of this proposal (available at `http://www.ai.mit.edu/~mpf/proposal/proposal.html`).

For example, there are possible applications of reversibility for error detection, debugging, and auditability which should be discussed. There are also questions about how to manage multiple, inter-communicating reversible processes within a multiprocessing framework, and about how to relate reversibility to rollback in transactional database systems.

Also, there is the very interesting issue of how to extend computational complexity theory to incorporate facts of physical reality other than just the second law of thermodynamics, such as, notably, the finiteness of the speed of light (which traditional complexity theory conveniently ignores by assuming that any of an unlimited number of RAM cells can be accessed in constant time). The longer version of the proposal mentions numerous other interesting issues as well.

However, all these tangential discussions will be optional, and will be undertaken only after the main core of the thesis research, proposed here, has been successfully completed and documented. If no time remains to pursue these other directions, the coverage of these topics in the thesis will be limited to the short discussions already written up in the longer version of this proposal.

4 Schedule Summary

The following chart summarizes the planned schedule for the thesis work.

```
1997

Jan \ Theoretical Studies---   - Characterize known algorithms
Feb  > Reversible Complexity   - Invent new algorithms
Mar /  Theory                  - Generalize techniques
Apr \ Experimental Studies---  \_ Design reversible language,
May  | Programming             /    write compiler.
Jun  |                         \ Implement example programs,
Jul  |                          >  run them, measure results.
Aug /                          /
Sep \ Thesis Writeup           - First draft.
Oct  | and Revision            - Second draft.
Nov  |                         - Optional extra discussion.
Dec /                          - Final draft, defense.
```

References

[1] C. H. Bennett. Logical reversibility of computation. *IBM Journal of Research and Development*, 6:525–532, 1973.

[2] C. H. Bennett. Time/space trade-offs for reversible computation. *Society for Industrial and Applied Mathematics Journal on Computing*, 18(4):766–776, 1989.

[3] Thomas H. Cormen, Charles E. Leiserson, and Ronald R. Rivest. *Introduction to Algorithms.* MIT Press/McGraw-Hill, 1990.

[4] Robert Y. Levine and Alan T. Sherman. A note on Bennett's time-space tradeoff for reversible computation. *Society for Industrial and Applied Mathematics Journal on Computing*, 19(4):673–677, 1990.

[5] Tommaso Toffoli. Computation and construction universality of reversible cellular automata. *Journal of Computer and System Sciences*, 15:213–231, 1977.

[6] Carlin J. Vieri. Pendulum: A reversible computer architecture. Master's thesis, MIT Artificial Intelligence Laboratory, 1995.

[7] S. G. Younis. *Asymptotically Zero Energy Computing Using Split-Level Charge Recovery Logic.* PhD thesis, MIT Artificial Intelligence Laboratory, 1994.