Space-Efficient Reversible Simulations

Ryan Williams

September 22, 2000

Abstract

We study constructions that convert arbitrary deterministic Turing machines to reversible machines; i.e. reversible simulations. Specifically, we study space-efficient simulations; that is, the resulting reversible machine uses $O(f(S))$ space, where S is the space usage of the original machine and f is very close to linear (say, $n \log n$ or smaller). We generalize the previous results on this reversibility problem by proving a general theorem incorporating two simulations: one is space-efficient ($O(S)$) and is due to Lange, McKenzie, and Tapp[5]; the other is time-efficient ($O(T^{1+\epsilon})$ for any $\epsilon > 0$, where T is the time usage of the original machine) and is due to Bennett[2]. Corollaries of our general theorem give interesting new time-space tradeoffs. One is that for any unbounded space constructible $f(n) = o(T(n))$, there is a reversible simulation using $O(S \log f(S))$ space and $O(f(S)^{1+\epsilon} c^{\min(\epsilon T/f(S))})$ time, for any $\epsilon > 0$. This gives the first reversible simulation that uses time subexponential in T, and space subquadratic in S.

1 Introduction

Reversible computation is, in essence, that which can be executed *backwards deterministically* as well as forwards deterministically. More formally, a Turing machine M is **reversible** if there is a deterministic TM M' with the property that there is a path from nodes C to D in the configuration graph of M iff there is a path from D to C in the configuration graph of M every configuration. It can be shown that this definition of reversibility is equivalent to the definition that requires every node in the configuration graph of M to have indegree ≤ 1. We will use this second characterization of reversibility, though the first one is closer to our intuitive idea.

Reversible models of computation are interesting for two important reasons. One is that reversible computers, unlike the more general models, are not known to require any heat dissipation during a computation.[3] This implies that reversible computational models may be very useful to us as we continue to miniaturize computing devices; in general, reductions in the temperature of the processor will allow us worry less about the materials and designs we must use when building smaller devices, since as the surface area of a chip lessens, more ingenuity is required to keep the chip cool. Another reason is that models of computation based on quantum mechanics are inherently reversible, since they obey the laws of quantum mechanics, which are reversible.

However, we would like to perform our usual irreversible operations, erasing and writing data with abandon. To this end, simulations of deterministic (irreversible) computation suitable for reversible computers have been studied. The most celebrated work in this area includes Bennett [2] and Lange, McKenzie, and Tapp [5] (hitherto referred to as LMT). The former (along with an analysis of Levine and Sherman[6]) showed that a reversible simulation of an arbitrary deterministic TM M using space S and time T could be performed in space $\Theta(c(\epsilon)S[1 + \log(T/S)])$ and time $\Theta(T^{1+\epsilon}/S^{\epsilon})$ for any $\epsilon > 0$, where $c(\epsilon) \approx \epsilon 2^{1/\epsilon}$. The latter gave a simulation that used only space $O(S)$, but the running time was $O(c^S)$, where c is a constant depending on M (roughly, about the size of the alphabet).

In the context of using quantum computing to simulate our usual computers, it has been suggested that space-efficient reversible simulation is more vital than time-efficiency; storing a qubit appears to be more difficult than performing a transformation on a qubit [5]. Our ultimate goal would be to construct a reversible simulation of determinism that requires only $O(T)$ time and $O(S)$ space. Oracle results by Frank and Amner seem to indicate such a simulation may not exist [3]; that is, there is a (reversible) oracle such that relative to this oracle deterministic time and space is not equal to reversible time and space.

An intermediate and perhaps more tractable goal is to find reversible simulations that require $o(S^2)$ space, yet $c^{\epsilon T}$ time for all $\epsilon > 0$ (subquadratic space, and subexponential time, i.e. $2^{o(T)}$). We introduce a simulation that is a straightforward conglomeration of the LMT simulation and Bennett's simulation. We will leave several parameters open in the construction of the simulation, such that by modifying parameters, different time-space tradeoffs result. We show that for any fixed $\epsilon > 0$ and "reasonable" $f(T,S)$ there is a simulation utilizing $O(S \log f(T,S))$ space and $O(f(T,S)^{1+\epsilon} \cdot c^{\min(\epsilon T/f(T,S),S)})$ time. Corollaries of this result include simulations that require space superlinear (but subquadratic) in S, and run in time subexponential in T.

2 Preliminaries

We begin by formally defining what we call a reversible simulation. In this paper we will only consider the offline TM model,with a two-way input tape, two-way worktapes, and a one-way output tape.

Let M be a deterministic TM, and x be an input to it.

Definition 2.1 *M is* **reversible** *if every node in the graph of all of its possible configurations has indegree ≤ 1.*

It is an exercise to see that this definition is equivalent to the more natural one: that there exists a Turing machine M' such that for all configurations C,D in the configuration graph of M, C has a path to D iff D has a path to C in the configuration graph of M'.

Without loss of generality, we will assume that M halts with blank worktapes. Let $I(M,x)$ and $F(M,x)$ be the initial and final configurations of TM M on x. To us, the final configuration will consist of the final state and the output.

Definition 2.2 *Let M be a TM. We say a TM M' is a* **reversible simulation** *of M if $L(M) = L(M')$, M' is reversible, and for all x, $F(M,x) = F(M',x)$.*

To express our results concisely, we will use the notations TISP$[T,S]$ and rTISP$[T,S]$. TISP$[T(n),S(n)]$ is the set of all languages accepted by deterministic TMs using time $O(T(n))$ and space $O(S(n))$ on inputs of size n. rTISP$[T(n),S(n)]$ is defined analogously for reversible offline TMs (so we explicitly require that the input to the machine is never erased during the computation). Observe that rTISP$[T,S] \subseteq$ TISP$[T,S]$. As another example, note that in this paper we are looking for results of the form TISP$[T,S] \subseteq$ rTISP$[f(T,S),g(T,S)]$, where $g(T,S) \approx S$, and $f(T,S)$ is as small as possible.

Before we begin stating reviewing the simulations, we will cite the following lemma which is easy to prove:

Let $O(M,x)$ be the final contents of the output tape of M on x.

Lemma 2.1 *(Bennett [1])*

Suppose M' is reversible, $L(M) = L(M')$, and for all x, $O(M,x) = O(M',x)$. Then there is a reversible simulation M'' of M that uses the twice the time and the same space as M'.

This lemma is very useful, since now we do not have to worry about making sure the worktapes are blank when we finish the simulation. For any reversible simulation we devise, anything we leave on worktape can be ignored without any loss of generality.

3 Review of Previous Simulations

We shall now describe the work of LMT[5] and Bennett[2], which our improvement uses crucially. For completeness, we will restate their theorems and corresponding simulations.

Theorem 3.1 *(Lange, McKenzie, Tapp [5])*

$TISP[T,S] \subseteq rTISP[2^{O(S)},S]$.

We prove a modular form of this theorem, so that we may use its result later in our results.

Theorem 3.2 *Let M be a deterministic Turing machine, C be a configuration of M, and q be a state of M. Then there is a $O(S)$ space, $O(2^{O(S)})$ time reversible algorithm DFS such that:*

> *$DFS(M, C, S, q)$ = the next configuration of M that is in state q, when M is executed starting from C and is required to only use $O(S)$ space.*

Proof. Suppose we are given an arbitrary M requiring time T and space S, C is some configuration of M, and q is a state of M. The basic idea is that $DFS(M, C, S, q)$ will, starting from C, perform a depth-first search of the directed graph of all $S(n)$ space-bounded configurations of $M(x)$ in such a way that only the current node (configuration of $M(x)$) ever needs to be stored. When the search finds a configuration D of M in state q, D is returned; if no such configuration is found, $DFS(M, C, S, q)$ is undefined.

More precisely, on input (M, C, q), N assigns a linear ordering on the transitions of M (the order in which the transitions appear in the string representing M). This extends to a linear ordering on the incoming edges for each node in the configuration graph of M.

The simulation is just a depth-first search as found in LMT[5] and Sipser[7].

$DFS(M, C, S, q)$:

(0) If C is in state q, then return C. If C uses more than space S, return *undefined*.

(1) If M entered C via the last incoming edge to C (with respect to the ordering),

(a) If M is in state q, then return C.

(b) Let (C, D) be the outgoing edge of C. If D uses space $S + 1$, then return *undefined*. Otherwise, return $DFS(M, D, S, q)$.

(2) If M entered C on an incoming edge e that is not last,

(a) If for all configurations D with a transition to C that is succeeded by e in the ordering, D uses space $S + 1$, then let (C, E) be the outgoing edge of C. Return $DFS(M, E, S, q)$.

(b) Otherwise, simulate M backwards one step, choosing the next applicable transition in the ordering. Simulating backwards means that the heads of M are moved in the direction opposite what is specified in the transition, the symbol that was to be read from worktape in the transition is written to worktape, and if a symbol was written to output, it is erased. Let the resulting configuration of this step be E. Return $DFS(M, E, S, q)$.

(3) If $M(x)$ entered C from its single outgoing edge,

(a) If there are edges incoming to C, let the edges coming into C be $e_1, \ldots, e_k$. Choose the first edge (wrt the ordering, which is of the form $e_i = (B, C)$. Return $DFS(M, B, S, q)$.

(b) If there are no edges incoming to C, let the outgoing edge be (C, D). Return $DFS(M, D, S, q)$.

(4) Otherwise, choose the first outgoing edge from C (with respect to the ordering); call it (C, D). Return $DFS(M, D, S, q)$.

The simulation is reversible, because if we reverse the ordering and start from the accept configuration searching for the initial configuration, precisely the same steps are performed in the above algorithm, but in reverse. It takes no more than $O(S)$ space to store the current configuration, and storing the forwards/backwards/edge information for determining cases (1), (2), or (3) of the algorithm requires only $O(1)$ space. It is easy to show that the search never visits any single node more than a constant number of times. Thus, the running time is bounded by the number of possible configurations in the graph, which is c^S for some $c > 0$ that depends on M.

□

It will be much easier for us if we formulate the "black box" in terms of running time:

Corollary 3.1 *Let M, C, S be the same as above. Then there is a $O(S + \log m)$ space, $O(2^{O(S)})$ time algorithm GeneralDFS such that*

> *$GeneralDFS(M, C, S, m)$ = the resulting configuration of M using space $\leq S$, when executed starting from C for $\leq m$ steps.*

Proof. The algorithm *GeneralDFS* is a simple modification of the above theorem.

First, we will conceptually divide the computation of $M(x)$ into k blocks of length m. We modify M so that it will have an m bit counter, which is initially set to 1. It will be incremented when a step of $M(x)$ is taken. The counter will reset to 1 when it is equal to m. (Note that its function is completely reversible.) Also, we will impose the condition that between the times where the counter is m and the counter resets to 1, M will be in one of two special states q_s and q'_s, alternating between them after each block, and this is the only time M is in one of these states. This restriction adds no real power; it will simply make it easier for us to state the algorithm in pseudocode.

The crucial function of the counter is to provide a tighter bound on the size of the configuration graph; we will use the information of the counter to prune the graph (as well as the space bound). When M' reaches a configuration C of $M(x)$ with m written on the counter, M' saves C to blank worktape. In other words, the configuration of $M(x)$ is saved at the end of each block.

We now describe the simulation more precisely. Let $M'[m]$ be the machine M augmented with an m-bit counter as described above.

```
$GDFS(M'[m], C, S, b)$:
    If $C$ is a final configuration, then accept (reject) if $C$ is accepting (rejecting).
    Let $q_s$ and $q'_s$ be the special states of $M'$. Save $C$ to blank worktape.
    If $s$ then call $GDFS(DFS(M'[m], C, S, q_s), \neg b)$, else call $GDFS(DFS(M'[m], C, S, q'_s), \neg b)$.

$GeneralDFS(M, C, S, m) := GDFS(M', C, S, \mathbf{true})$.
```

This concludes the description of the simulation.

First, we argue that *GeneralDFS* is reversible. First, note that $DFS(M, C, q)$ is reversible via Theorem 4.1, and the only writing done in the *GeneralDFS* procedure is to blank tape. Furthermore, the comparisons for the conditionals are reversible; knowing $\neg s$ we can obtain s, and knowing that the simulation accepts / rejects we know that C was accepting / rejecting.

Note that *GeneralDFS* is strictly speaking not a reversible simulation of M, as the worktapes in the final configuration of it are not blank, where the worktapes of M is blank in this case. This is easily corrected by applying Lemma 2.1. We make M' a reversible simulation of M by, when a final configuration has been reached, setting an "accept" or "reject" switch in M', and performing the simulation in reverse as described above. Then, when M' reaches the initial configuration, depending on the value of the switch, M' halts in the accept or reject state of M.

□

We move to the next major simulation used in this work. Bennett's simulation of determinism [2] with reversibility has a slightly more sophisticated construction. It divides the computation of $M(x)$ from initial state to final state into blocks of m steps, with an extra parameter n such that $T = mk^n$. Roughly, Bennett's simulation only saves intermediate configurations of the computation, and uses them to reversibly reconstruct the other configurations as needed.

In the base case of the simulation (where $n = 0$), a straightforward time-efficient simulation of M is performed. The simulation saves the index of each transition taken (there is an ordering on them, so there is an index for each one) to blank worktape. We will think of this history-saving simulation as an implementation of a black box $BASE(M, C, m)$ to Bennett's simulation. $BASE(M, C, m)$ will simply perform the base case function of Bennett's simulation, and return the configuration M results in starting from configuration C and executing for m steps. $RevBASE(M, C, m)$ will be the "reverse" of $BASE$. It will not return a configuration; its actions will simply be side-effects, changing the configuration of M to what it would be m steps before it reaches configuration C.

More precisely, to execute k^n m-length blocks of M reversibly starting from config C_0, there are two mutually recursive programs, *Bennett* and *RevBennett*. Their descriptions are as follows.

```
$Bennett(BASE, M, C_0, k, m, n)$:
    If $n = 0$ then return $BASE(M, C, m)$.
    For $i = 1, \ldots, k$,
        Let $C_i := Bennett(BASE, M, C_{i-1}, k, m, n-1)$, writing $C_i$ to blank worktape.
    End for
    For $i = (k-1), \ldots, 1$,
        $RevBennett(BASE, M, C_{i-1}, k, m, n-1)$
    End for
    Return $C_k$.
```

Similarly, to execute k^n blocks of $M(x)$ in reverse starting from config C_k:

```
$RevBennett(BASE, M, D, k, m, n)$ :
(We assume that if $n$ is sufficiently large, then $C_0, \ldots, C_k$ from Bennett are stored on worktape.)
    If $n = 0$, then return $RevBASE(M, C, m)$.
    For $i = 1, \ldots, (k-1)$,
        $RevBennett(BASE, M, C_{i-1}, k, m, n-1)$
    End for
    For $i = k, \ldots, 1$,
        Erase $C_i = Bennett(BASE, M, C_{i-1}, k, m, n-1)$ from worktape.
    End for
```

It is easy to see that *RevBennett* is the same algorithm as *Bennett* but is executed in reverse. We will leave the proof that the algorithms are indeed reversible to the reader.

We briefly describe the complexity analysis; it will be given in more detail later. In their paper, Levine and Sherman [6] calculate that the time requirement for this simulation is $T' = m(2k-1)^n$, and the space requirement is $S' = mn(k-1)$, when $BASE$ is taken to be the history-saving simulation. Bennett chooses $m = S$ (since otherwise his time-efficient simulation would use more space than saving a configuration) and holds k constant, so $n = \frac{\log(T/S)}{\log(k)}$, and the values obtained are $T' = S(2k-1)^{\frac{\log(T/S)}{\log(k)}}$, $S' = S\frac{\log(T/S)}{\log k}(k-1) = O(S\log(T/S))$. Letting $\epsilon = \log(2-(1/k))/\log(k)$, $T' = O(T^{1+\epsilon}/S^{\epsilon})$.

The resulting theorem is the following.

Theorem 3.3 *(Bennett, Levine and Sherman)*

$$TISP[T, S] \subseteq rTISP[2^{1/\epsilon}T^{1+\epsilon}/S^{\epsilon}, S\log(T/S)].$$

However, we wish for something more general. Suppose we let the algorithm $BASE$ be arbitrary. Then the following theorem holds:

Theorem 3.4 *There exists a simulation Bennett that, given a reversible simulation $BASE$ that uses time $T_B(T, S)$ and space $S_B(T, S)$, Bennett$(BASE, M, I(M, x), k, m, n)$ is a reversible simulation of $M(x)$ that uses $T_B(m, S)(2k-1)^n$ time and $S_B(m, S)n(k-1)$ space, where $T = mk^n$, i.e.*

$$TISP[mk^n, S] \subseteq rTISP[T_B(m, \min(m, S))(2k-1)^n, S_B(m, S)n(k-1)].$$

Proof. Let $\overline{T}$ be the running time of *Bennett*, and $\overline{T^{-1}}$ be the running time of *RevBennett* as described above. Given a $BASE$ algorithm that uses time $T_B(T, S)$ and space $S_B(T, S)$, $\overline{T}$ and $\overline{T^{-1}}$ satisfy the following recurrences:

$\overline{T}(n) = k\overline{T}(n-1) + (k-1)\overline{T^{-1}}(n-1)$, if $n > 0$

$\overline{T}(0) = T_B(m, \min(m, S))$.

$\overline{T^{-1}}(n) = k\overline{T}(n-1) + (k-1)\overline{T^{-1}}(n-1)$, if $n > 0$

$\overline{T^{-1}}(0) = T_B(m, \min(m, S))$.

The justification for the base cases are that when $n = 0$, $BASE$ runs for m steps; so the space usage over that time is the minimum of m and S.

Combining the two together and solving, we find that

$$\overline{T}(n) = T_B(m)(2k-1)^n.$$

Since we need S space to store a configuration; when $n = 0$ we use S_B (the space of $BASE$), and if $n > 0$ we store $(k-1)$ configurations, plus use the space of the $n-1$ stage.

The space usage $\overline{S}(n)$ obeys the following recurrence:

$\overline{S}(n) = (k-1)S + \overline{S}(n-1)$, if $n > 0$

$\overline{S}(0) = S_B(m, \min(m, S))$.

Solving the recurrence,

$$\overline{S}(n) = Sn(k-1) + S_B(m, \min(m, S)).$$

□

Corollary 3.2 *Let $f(\cdot,\cdot)$ be constructible in $O(n)$ space and $O(n)$ time, T (and S) be time (and space) constructible, and $0 < f(T,S) \le T$. Then for all $\epsilon > 0$,*

TISP[mk^n, S] $\subseteq$

rTISP[$T_B(\frac{T}{f(T,S)}, \min(\frac{T}{f(T,S}, S))f(T,S)^{1+\epsilon}, \min(\frac{T}{f(T,S)}, S)\log f(T,S) + S_B(\frac{T}{f(T,S)}, \min(\frac{T}{f(T,S)}, S))$].

Proof. Given f, we will let k be constant and set $k^n = f(T,S)$. Note that on inputs of size i, $f(T(i), S(i)) = k^n$ can be computed and thus m can be computed.

Note that $m = T/f(T,S)$, so using the previous theorem

TISP[mk^n, S] $\subseteq$

rTISP[$T_B(T/f(T,S), \min(\frac{T}{f(T,S)}, S))(2k-1)^n, \min(\frac{T}{f(T,S)}, S)n(k-1) + S_B(\frac{T}{f(T,S)}, \min(\frac{T}{f(T,S)}, S))$].

It remains to show that (1) for all $\epsilon > 0$, there is a k such that $(2k-1)^n \le f(T,S)^{1+\epsilon}$, and (2) $(k-1)n = c\log f(T,S)$ for some constant $c > 1$.

For (1), $\frac{(2k-1)^n}{f(T,S)} = \frac{(2k-1)^n}{k^n} = (\frac{2k-1}{k})^n = (2-(1/k))^{\log(k^n)/\log(k)} = (k^n)^{\log(2-(1/k))/\log(k)} \le (k^n)^{1/\log(k)} = f(T,S)^{1/\log(k)}$.

Therefore, $(2k-1)^n = f(T,S)^{1+1/\log(k)}$. For any $\epsilon > 0$, we can choose k to be a constant such that $f(T,S)^{1+1/\log(k)} \le f(T,S)^{1+\epsilon}$.

For (2), $n = \log(f(T,S))/\log(k)$, so $(k-1)n = \frac{k-1}{\log(k)}\log(f(T,S))$. Let $c = \frac{k-1}{\log(k)}$ and we are done.

□

Theorem 3.5 *TISP[T, S] $\subseteq \bigcup_{c>1}$ rTISP[$c^{\min(\frac{T}{f(T,S)}, S))}f(T,S)^{1+\epsilon}, \min(\frac{T}{f(T,S)}, S)\log f(T,S) + S$].*

Proof. Apply the previous corollary. Let $BASE$ be the $GeneralDFS$ algorithm from Theorem 3.1. Recall that the simulation of an m-step block using the space-efficient LMT method requires $c^{\min(m,S)}$ time and $O(S)$ space, for some c depending on M. So we apply the previous corollary, with $T_B(T,S) = c^{\min(T,S)}$ and $S_B(T,S) = O(S)$; the theorem follows.

□

The above result is our main result. It says that deterministic space and time is contained in reversible subexponential time, and very nearly linear space, as we can let f be any slow growing function like $\log n$, $\log\log n$, or even iterated logarithm ($\log^* n$), and it is still constructible in $O(n)$ space. A small factor of space usage ($\log S$) can give us a significant decrease in time when S is not exponentially separated from T.

Note that when $f(T,S) = 1 = k^n$, no recursion of the algorithm is ever done ($n = 0$), so the simulation has space/time usage comparable to LMT. If $f(T,S) = T = k^n$, then $m = 1$, and the DFS simulation in the base case

Simulation:	$S' + \log T'$	$S' \log T'$	$(S')^{S'} T'$
Bennett	$O[S \log(\frac{T}{S}) + \log(\frac{T^{1+\epsilon}}{S^{\epsilon}})]$	$O[S \log(\frac{T}{S}) \log(\frac{T}{S})]$	$(S \log(\frac{T}{S}))^{S \log(\frac{T}{S})} T^{1+\epsilon}/S^{\epsilon}$
LMT	$O[S]$	$O[S^2]$	$O[S^S c^S]$
Thm 3.5	$O[S \log f + \min(\frac{T}{f}, S)]$	$O[S \log f \cdot (\min(\frac{T}{f}, S) + \log f)]$	$(S \log f)^{S \log f} f^{1+\epsilon} c^{\min(T/f,S)}$

Table 1: A comparison of the different simulations presented, via different measures of complexity. S', T' are the space/time requirements for the reversible simulation in question.

of the simulation is only run for one step; hence we have a running time/space comparable to Bennett's original simulation.

It is here that we begin to realize the subtleties of how time and space actually are traded for one another in our simulation. One raises some fraction of the original to $1+\epsilon$, dividing that fraction from the exponential term in time, while incurring a charge in space of only a logarithmic factor. Intuitively, in this situation we can think of Bennett's simulation wishing to gain the fastest time possible, so the maximum amount of time is removed from the exponent, and the space usage becomes high (quadratic). The LMT simulation removes only a constant factor from the exponent, yielding a better space bound.

Now suppose a deterministic M we wish to simulate is such that $T = O(S^d \log^d(S))$ for some $d \geq 2$. Using the simulation from the above theorem, there exists a subquadratic space simulation of M, and choosing ϵ appropriately, $O(S^{d+\epsilon} \cdot c^{\epsilon \log^d(S)}) = O(S^{d+\epsilon} \cdot S^{\epsilon \log^{d-1}(S)}) = O(S^{d+\epsilon(1+\log^{d-1}(S))})$ time. In this case, we have subquadratic space and $S^{O(\log^{d-1}(S)}$ time, which is significantly subexponential.

Now, let us observe the time-space tradeoffs for particular values of k, m, and n.

Corollary 3.3

$$\mathrm{TISP}[T, S] \subseteq \bigcap_{\epsilon > 0} \mathrm{rTISP}[(1/\epsilon)^{1+\epsilon} 2^{O[\min(\epsilon T, S)]}, \min(\epsilon T, S) \log(1/\epsilon) + S].$$

Proof. Set $f(T, S) = 1/\epsilon$ in the above corollary.

□

This improves on the LMT result that $\mathrm{TISP}[T, S] \subseteq \mathrm{rTISP}[2^{O(S)}, S]$, but only in a strictly technical sense. The constant factors in the time and space both diverge as $\epsilon \to 0$.

Corollary 3.4

$$\mathrm{TISP}[T, S] \subseteq \bigcap_{k > 1} \mathrm{rTISP}[S^k \cdot 2^{O[\min(T/S^k, S)]}, \min(T/S^k, S) \log(S) + S].$$

Proof. Given any such k, set $f(T, S) = S^k$ in the above corollary.

□

4 Analysis of Improvement

The general improvement that our new tradeoffs introduce is difficult to quantify. It can be summed up informally by saying there is a reversible simulation that is very space efficient, and runs in time subexponential in T, for any time T and space S.

Table 1 gives a comparison of the three main simulations presented in this paper, with respect to various measures that weigh space usage much more heavily than time usage. The first measure does not punish much for exponential gains in time at all. The second measure is a product, so exponential gains in time are more prominent here. The third measure punishes greatly for slight increases in space.

It is hard to tell where our simulation gives improvement. One result shows us the limitations of our general theorem: our conglomeration of the two simulations in Theorem 3.5 cannot outperform both of them simultaneously in terms of time and space, regardless of what f is chosen.

Let S_B, T_B be the space and time used by Bennett's simulation, and S_D, S_D be that used by the DFS simulation.

Theorem 4.1 *If for all $c > 0$, $f(T,S) > c$, then applying $f(T,S)$ to Theorem 3.5 results in a simulation with running time T_f such that $O(T_f) > O(T_B)$; i.e. for every setting of f, the time usage is asymptotically more than Bennett.*

Proof. Suppose $T/S > f(T,S) > c$. Then $T_f = c^{\min(T/f(T,S),S))} f(T,S)^{1+\epsilon} = c^S f(T,S)^{1+\epsilon} > O(T_D) = O(c^S)$, since $f(T,S) > c$. So this case is uninteresting, as the resulting time performance is worse than even LMT.

Now suppose $T/S < f(T,S)$. Then $T_f = c^{T/f(T,S)} f(T,S)^{1+\epsilon} > c^{T/f(T,S)}(T/S)^{1+\epsilon}$.

Assuming $c^{T/f(T,S)} > S$, this time is worse than T_B, which is $T^{1+\epsilon}/S^{\epsilon}$. On the other hand, if $c^{T/f(T,S)} \leq S$, then $T \leq f(T,S)\log(S)$. Then $T_f = c^{T/f(T,S)} f(T,S)^{1+\epsilon} \geq S f(T,S)^{1+\epsilon} > T^{1+\epsilon}$, which is also worse than *Bennett*.

□

Throughout this paper, we have just been considering algorithms that, given as input an arbitrary machine M, output a machine M' that is reversible and simulates M. Suppose we also add to our input descriptions of functions T and S that are promised to be upper bounds on the resources used by M, and the compiler is capable of determining what T/S is. Then, our "reversible compiler" will be able to construct nice simulations of M.

Given that T/S is exponential (i.e. $T = O(d^S)$ for some $d > 1$), the compiler should perform the LMT simulation and get a running time polynomial in T, and use efficient space. Given that T/S is constant, it should perform a simple simulation where the index of each transition taken is saved to blank worktape, and also use (asymptotically) the same amount of time and space as the original. Given that T/S is bounded by a polynomial, it performs Bennett's simulation, with $O(S \log S)$ space and $O(T^{1+\epsilon})$ time.

The only cases that are difficult for such a compiler to find simulations where space and time are both subquadratic are those with T more than any polynomial in S, but subexponential in S, in which case the best course of action is to use our simulations, using $O(S)$ space with arbitrarily small exponents in time or $O(S \log S)$ space with a relatively small function in the exponent for time, or Bennett's simulation, with $O(S^2)$ space and polynomial time.

5 Conclusion

Our simulations are simple generalizations of the previously known methods, which give new time-space tradeoffs for reversible computation, where a little extra space (but still subquadratic) gives a relatively small function in the exponent for time. We are convinced that the procedure above probably can be improved to $O(S)$ space and $O(\max(T^{1+\epsilon}, 2^{\epsilon S}))$ time for all $\epsilon > 0$. These bounds would be close to the best possible, unless there exist reversible simulations that are non-relativizing. In such a situation, it may be possible to achieve linear space and linear time simulation.

6 References

[1] C.H. Bennett, *Logical reversibility of computation*, IBM J. Res. Devel., 17 (1973), 525-532.

[2] C.H. Bennett, *Time/space trade-offs for reversible computation*, SIAM J. Comput., 18 (1989), 766-776.

[3] M. Frank, M. J. Amner, *Separations of reversible and irreversible space-time complexity classes.* MIT Technical Report.

[4] R. Landauer, *Irreversibility and heat generation in the computing process*, IBM J. Res. Develop., 5 (1961), 183-191.

[5] K.J. Lange, P. McKenzie, and A. Tapp, *Reversible space equals deterministic space*, Proc. 28th ACM Symp. Theory of Computing (1996) 212-219.

[6] R.Y. Levine and A.T. Sherman, *A note on Bennett's time-space tradeoff for reversible computation*, SIAM J. Comput., 19:4 (1990), 673-677.

[7] M. Sipser, *Halting space-bounded computations*, Theor. Comput. Sci., 10 (1980) 335-338.