Reversible Computation

S. Buss, 22 January 2013.

Part I:
[Bennett '73; Bennett '89; Levine-Sherman '90].
[Also: Lecher '63.]
Goal: Reversible simulation of deterministic computation.

"Poor man's" or "Global" reversibility.

→ Remember the input and count the number of steps.
→ Can reverse a computation step by returning to the initial configuration, and running the entire computation up to the previous step.

Bennett '73 defined "reversible" more stringently:
No two configurations can lead to the same successor configuration.

Slightly modified definition of TM so that an instruction can either:
(a) Read a symbol + overwrite with a new symbol, or
(b) Ignore the read symbol (not even the stack arm) + move tape head -1, 0, or 1. Square right.

(Options (a) + (b) can be chosen differently in different ways)

Some things that can be done reversibly:
(1) Make a copy of a string (on top of blanks!)
(2) Erase (overwrite with blanks) a string if it is a copy of another string.
Bennett\'s construction

Let $M$ be a deterministic (multipurpose) TM, with $k$ types. Reversibly simulate $M$ with a $(k+1)$ type machine.

- The new type holds a history of the transition rules executed during the computation of $M$: $\leq 2k$ symbols per step of $M$.

#### Tape $k+1$

- **blank**

#### Tape $k+1-k$

- **Input**

simulates $\downarrow \downarrow \downarrow \downarrow \leftarrow$ Run $M$ forward

#### Tape $k+1$

- **history**

#### Tape $k+1-k$

- **Work Tape**
  - **Output**

$\downarrow \downarrow \downarrow \downarrow \leftarrow$ Copy output

#### Tape $k+1$

- **history**

#### Tape $k+1-k$

- **Work Tape**
  - **Output**

$\downarrow \downarrow \downarrow \downarrow \leftarrow$ Run $M$ backwards

#### Tape $k+1$

- **blank**

#### Tape $k+1-k$

- **blank write**

- **Input**
  - **Output**

---

For machines that use space $O(n)$, the input is read only, not part of the modified types. We\'ll deal only w/ such machine that have a boolean output or short output.
The above simulation of $M$ uses

- Time $O(T)$
- Space $O(S+T)$

where $M \in TISP(T, S)$.

For poly-time, logspace machine this gives a poly-space procedure.

[Bennett'79]: Remarks you do a two level version of this and achieve Time $O(T)$ and Space $O(\sqrt{ST})$.

and that by using multiple nested levels can perhaps achieve Time $O(T^2)$ and Space $O(S\log T)$.

This is carried out, and improved, in [Bennett'89, Levin Shoup'90]

Parameters:
- $m = \#$ of steps simulated at base level à la [Bennett'73.]
- $k = \#$ of blocks between recursive calls

```
<table>
<thead>
<tr>
<th>Config</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>Config</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Config</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Config</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Config</td>
</tr>
</tbody>
</table>
```

$k$ recursive calls (Forward computation)

$k-1$ recursive calls (Reversing computation)
At level $n$:

- Number of states of $M$ simulated: $m^k = k^n$.
- Time used for one configuration: $T_{n-1}$, computation at level $n$:

$$P_n = (2k-1)P_{n-1}$$
$$P_0 = m$$

So:

$$P_n = m(2k-1)^n$$

Space used is given by:

$$S_n = \begin{cases} (k-1)m & \text{if } n = 1 \\ (k-1)m + S_{n-1} & \text{if } n > 1. \end{cases}$$

So:

$$S_n = m - m(k-1).$$

Take $m = S$ (for level 0, history uses same space as T.M. $M$).

Fix $k$.

Suppose $M$ is in $TISP(T, S)$. $T = mk^n$, $S = m$

[Lenine-Sherman]:

The reversible computation uses:

- Space: $S' = S_n = S - \log\frac{T}{S} \frac{k-1}{\log k}$
- Time: $T' = P_n = T \left(\frac{T}{S}\right)^k$

where $E = \frac{\log(2-1/k)}{\log k}$.

And $E \to 0$ as $k \to \infty$.

**Proof:**

$$T' = P_n = S \cdot (2k-1)^n \cdot S \cdot \left(\frac{2k-1}{k}\right)^n = T \cdot \left(\frac{2k-1}{k}\right)^n$$

$$= T \cdot \left(\frac{T}{S}\right)^k.$$
Let \( M \) be a deterministic, space-S Turing machine. Assume initial configurations have no predecessor configurations and \( M \) halts on all inputs.

On input \( x \): Initial configuration is \( C(x) \).

\( C(x) \) determines a component in the undirected graph of one-step reachability among configurations.

Lemma: The connected component of \( C(x) \) is tree-like with a single root (sink node).

![Graph Example](image)

From the directed graph of \( M \)'s configurations of depth \( S(x) \), form an "Eulerian tour" on the "edge ends" of \( 6 \).

Ex: The edge ends are a1, a3, b2, b4, c3, c6, d4, d6, e5, e6, f6, f7.

Each vertex of \( G \) has finite form, corresponding to \( M \) having only finitely many transition rules. Form an arbitrary cyclic permutation of the incident edge ends. Also let \( \pi \) swap the ends of two an edge.

Ex: \( \pi(a1) = a3 \)
\( \pi(a3) = a1, \) etc.
\( \tau(a3) = c3 \)
\( \tau(a1) = a1, \) etc.
\( \tau(c6) = d6 \)
\( \tau(d6) = e6 \)
\( \tau(e6) = f6 \)
\( \tau(f6) = c6. \)

Define \( \lambda = \pi \tau \).

Ex: \( a1 \xrightarrow{\lambda} a3 \xrightarrow{\lambda} c6 \xrightarrow{\lambda} d4 \xrightarrow{\lambda} b2 \xrightarrow{\lambda} b4 \xrightarrow{\lambda} d6 \xrightarrow{\lambda} \cdots \)
Thus $x$ is a cycle permutation of the edge ends of the connected component of $C(x)$.

The reversible simulation just iterates $x$ until reaching one (the) finalizing configuration.

It needs to remember only the current edge end; hence uses only $O(5)$ space.

Hence uses item 2 of $O(5)$.

Then $SPACE(5) \leq rev\_SPACE(5)$.

[Buttman-Trapp-Vitanyi '01; Williams, unpub.] gave since small representatives by combining Bennett's method with the LMT construction. Namely, the lower levels of Bennett's construction are replaced by a use of LMT.

Williams conjecture: $TISP(7, 5) \leq rev\_TISP(1, 5) + \begin{array}{c}
\text{rev-} TISP \left( \max \left(7 + \varepsilon \cdot 2^5, 5 \right) \right) \\varepsilon
\end{array}$

should be possible. However current methods are not close to this yet.