A Simple and Efficient Universal Reversible Turing Machine

Holger Bock Axelsen
Robert Glück

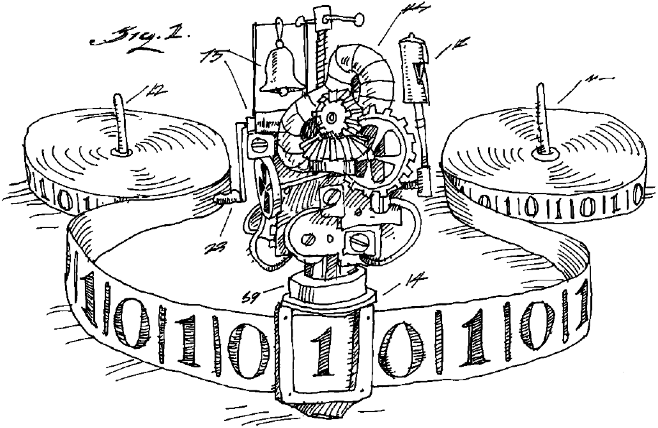
DIKU, Dept. of Computer Science, University of Copenhagen
`www.diku.dk/~funkstar`

LATA 2011, May 31, Tarragona

Overview

- Reversible Turing machines
- Universality for RTMs
- A first principles URTM

The setting: Turing machines



[Picture credit: Tom Dunne, American Scientist, March-April 2002]

Turing machines

Definition (Turing machine)

A TM T is a tuple $(Q, \Sigma, \delta, b, q_s, q_f)$ where Q is a finite set of states, Σ is a finite set of tape symbols, $b \in \Sigma$ is the blank symbol,

$$\delta \subseteq (Q \times [(\Sigma \times \Sigma) \cup \{\leftarrow, \downarrow, \rightarrow\}] \times Q)$$

is a partial relation defining the transition relation, $q_s \in Q$ is the starting state, and $q_f \in Q$ is the final state. There must be *no* transitions leading out of q_f nor into q_s.

Triple format for transition rules

$$\delta \subseteq (Q \times [(\Sigma \times \Sigma) \cup \{\leftarrow, \downarrow, \rightarrow\}] \times Q)$$

The form of a triple format rule in δ is either:

- a symbol rule $(q, (s, s'), q')$ where $s, s' \in \Sigma$, or
- a move rule (q, d, q') where $d \in \{\leftarrow, \downarrow, \rightarrow\}$.

(Triples can be converted to the usual quintuples, and vice versa. We use it for convenience.)

Reversible Turing machines (RTMs)

Intuition: RTMs are those where each configuration has a *unique* successor and predecessor configuration.

Definition (Reversible Turing machine)

A TM T is *reversible* iff it is (locally) forward and backward deterministic.

Local backward determinism: Examples

- $(\mathtt{q}, (\mathtt{a}, \mathtt{b}), \mathtt{p})$ and $(\mathtt{q}, (\mathtt{a}, \mathtt{c}), \mathtt{p})$ respects bwd determinism.
- $(\mathtt{q}, (\mathtt{a}, \mathtt{b}), \mathtt{p})$ and $(\mathtt{r}, (\mathtt{c}, \mathtt{b}), \mathtt{p})$ breaks bwd determinism.
- $(\mathtt{q}, (\mathtt{a}, \mathtt{b}), \mathtt{p})$ and $(\mathtt{r}, \rightarrow, \mathtt{p})$ breaks bwd determinism.

Local forward/backward determinism

Definition (Local forward determinism)

A TM T is *local forward deterministic* iff for any distinct pair of triples $(q_1, a_1, q_1') \in \delta$ and $(q_2, a_2, q_2') \in \delta$, if $q_1 = q_2$ then $a_1 = (s_1, s_1')$ and $a_2 = (s_2, s_2')$, and $s_1 \neq s_2$.

Definition (Local backward determinism)

A TM T is *local backward deterministic* iff for any distinct pair of triples $(q_1, a_1, q_1') \in \delta$ and $(q_2, a_2, q_2') \in \delta$, if $q_1' = q_2'$ then $a_1 = (s_1, s_1')$ and $a_2 = (s_2, s_2')$, and $s_1' \neq s_2'$.

RTM computability

Some important results:

- RTMs compute *injective functions*, only.
- *All* injective computable function are computable with RTMs.
- 1-tape, 3-symbol RTMs are sufficient.
- RTMs can be easily inverted.

Classical universality

A universal TM U is defined as a *self-interpreter* for Turing machines:

$$[\![U]\!](\ulcorner T \urcorner, x) = [\![T]\!](x) \ .$$

Here, $\ulcorner T \urcorner \in \Sigma^*$ is a Gödel number representing some TM T.

Problem: Does *not* work for RTMs - $[\![U]\!]$ is non-injective.

RTM-universality

An RTM-universal TM U_R is defined by

$$[\![U_R]\!](\ulcorner T \urcorner, x) = (\ulcorner T \urcorner, [\![T]\!](x)) \ .$$

where $\ulcorner T \urcorner \in \Sigma^*$ is a Gödel number representing some RTM T.

$[\![U_R]\!]$ is injective and computable $\Rightarrow$ computable by some RTM.

Why a first-principles approach?

Pioneering work by Bennett, Morita rely on *reversible simulations* of irreversible machines. Asymptotically *very* costly: As much space as time!

The URTM we give has better complexity: (Program dependent) constant factor slowdown, same space as interpreted program.

URTM overview

Scope:

- Interprets 1-tape, 3-symbol RTMs ($T = \{Q, \{b, 0, 1\}, \delta, b, q_s, q_f\}$).

Structure:

- Work tape: Identical to T's tape.
- Program tape: Contains the program $\ulcorner T \urcorner$.
- State tape: Encoding of T's internal state, q_c.

Program encoding $\ulcorner T \urcorner$

A program is a string $\ulcorner T \urcorner$ over $\Sigma = \{b, \mathtt{0}, \mathtt{1}, \mathtt{B}, \mathtt{S}, \mathtt{M}, \#\}$. $\ulcorner T \urcorner$ lists the rules δ of T, with q_s rule first, q_f rule last.

$$\begin{aligned} trans(q,(s,s'),q') &= \mathtt{S}\#e(q)\#e(s)e(s')\#(e(q'))^R\#\mathtt{S} \\ trans(q,d,q') &= \mathtt{M}\#e(q)\#e(d)\#(e(q'))^R\#\mathtt{M} \end{aligned}$$

$e : Q \to \{\mathtt{0},\mathtt{1}\}^{\lceil \log |Q| \rceil}$ is an injective binary encoding of states.

$$e(s) = \begin{cases} \mathtt{B} & \text{if } s = b \\ s & \text{otherwise} \end{cases} \qquad e(d) = \begin{cases} \mathtt{10} & \text{if } d = \leftarrow \\ \mathtt{BB} & \text{if } d = \downarrow \\ \mathtt{01} & \text{if } d = \rightarrow \end{cases}$$

Program encoding, example

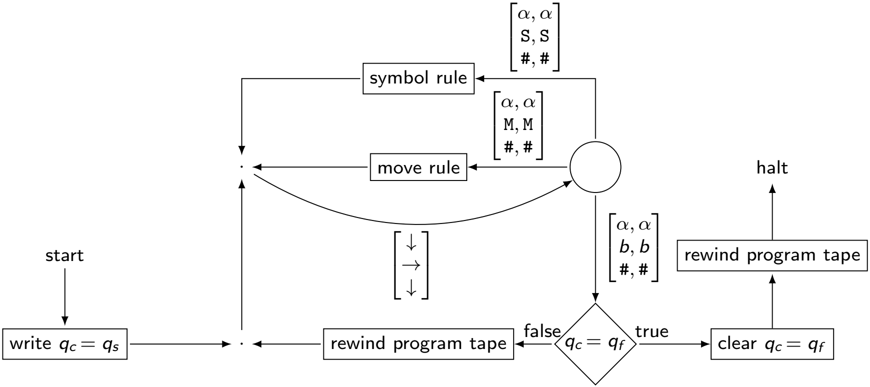
RTM $T = (\{q_0, q_1, q_2, q_3\}, \{b, 0, 1\}, \delta, b, q_0, q_3)$

$$\delta = \{(q_0, \rightarrow, q_1), (q_1, (0, 1), q_2), (q_1, (1, 0), q_2), (q_2, \leftarrow, q_3)\}$$

$\ulcorner T \urcorner =$ `M#00#01#10#MS#01#01#01#SS#01#10#01#SM#10#10#11#M`

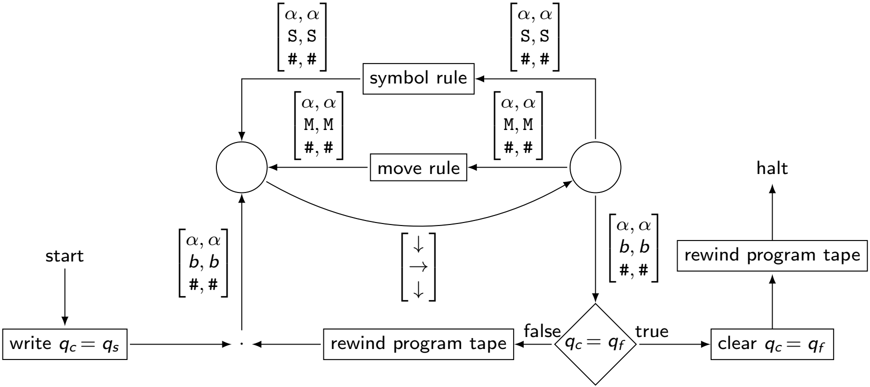
e is given by $q_0 \mapsto 00$, $q_1 \mapsto 01$, $q_2 \mapsto 10$ and $q_3 \mapsto 11$.

URTM program



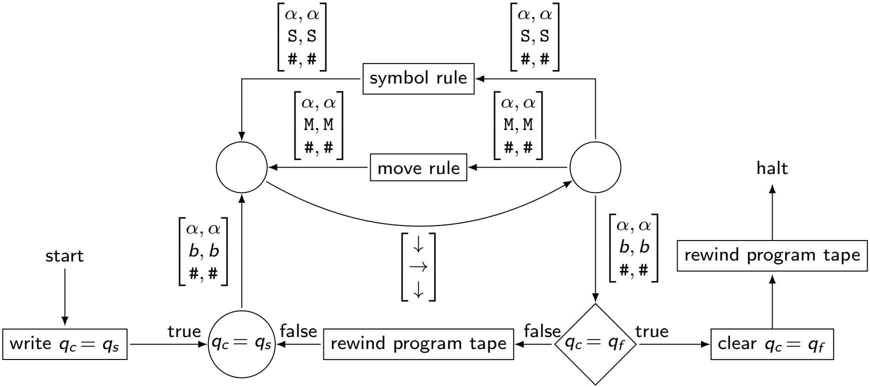
Problem: Lots of irreversibilities in control flow.

URTM program



Use enclosing S,M to join paths after rule tests.

URTM program



Works because q_s is only visited once.

String comparison

A key functionality we need to implement is string comparison.

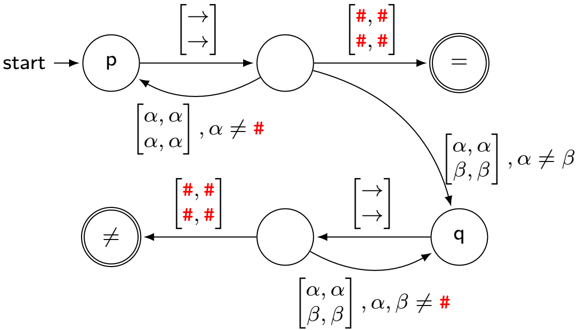
Assume a 2-tape structure

- $\#s_1 \cdots s_n\#$
- $\#t_1 \cdots t_n\#$

with tape heads on the *leftmost* #.

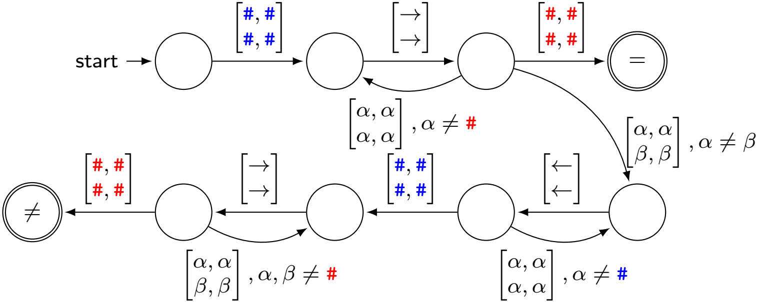
From internal state q, we want to pass over the strings, ending in internal state $q_=$ if the strings match, and in internal state $q_{\neq}$ if they don't, with the tape heads in either case on the *rightmost* #.

Irreversible string comparison of $\begin{matrix}\#s_1 \cdots s_n\# \\ \#t_1 \cdots t_n\#\end{matrix}$

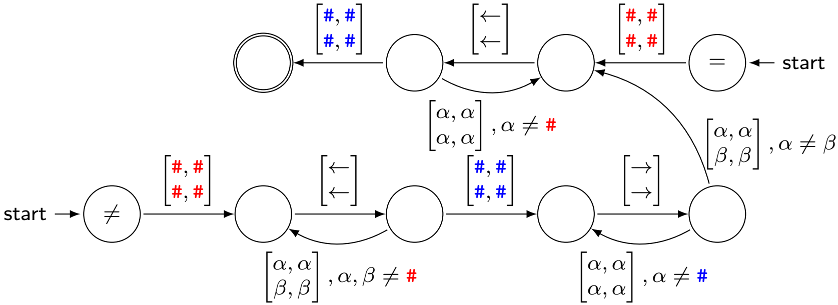


Irreversibility at state q (and probably p).

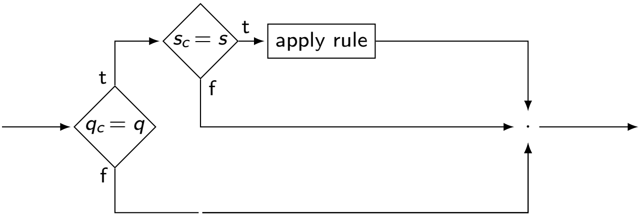
Reversible string comparison of $\begin{matrix}\#s_1 \cdots s_n\# \\ \#t_1 \cdots t_n\#\end{matrix}$



Inverse string comparison of $\begin{matrix} \#s_1 \cdots s_n\# \\ \#t_1 \cdots t_n\# \end{matrix}$ = join

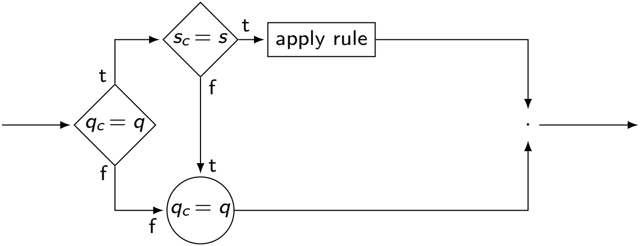


Testing a symbol rule $(q, (s, s'), q')$



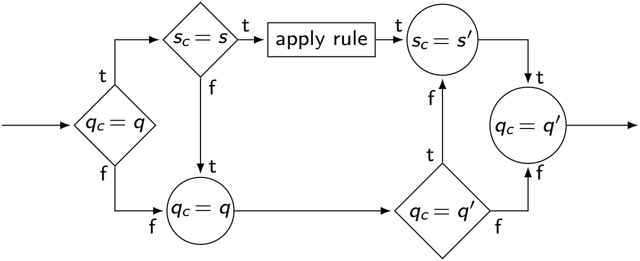
Must resolve the join in control flow.

Testing a symbol rule $(q, (s, s'), q')$



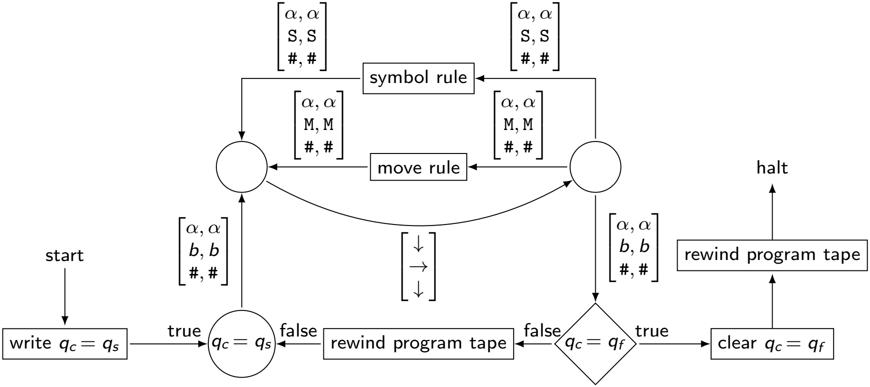
This assertion works for all T. Still irreversible...

Testing a symbol rule $(q, (s, s'), q')$



Only works because T is reversible!

URTM program



Move rule is analogous to symbol rule.
Write/clear are subparts of apply rule.
Rewind is a subpart of string comparison.

Bonus: Inverse interpretation

URTM can work as a (reversible) *inverse interpreter* with no extra overhead. A reversible inverse interpreter is a program *rinvint* s.t.

$$[\![rinvint]\!](p, y) = (p, x) \quad \text{iff} \quad [\![p]\!](x) = y$$

We intentionally designed the program encoding s.t.

$$\ulcorner T \urcorner^{R} = \ulcorner T^{-1} \urcorner$$

Let R perform string reversal. (Linear time using 2 tapes.)

$$[\![R \circ U \circ R]\!] (\ulcorner T \urcorner, y) \quad = \quad (\ulcorner T \urcorner, [\![T^{-1}]\!](y))$$

Conclusion

First URTM with

- Constant factor slowdown (proportional to $length(\ulcorner T \urcorner)$.)
- No space overhead (unlike all previous approaches.)
- Inverse interpretation for free.

The RTMs can simulate themselves efficiently.