Fundamental Computer Science
Sequence 1. Turing Machines
Random Access TM

Denis Trystram

February, 2021

Content

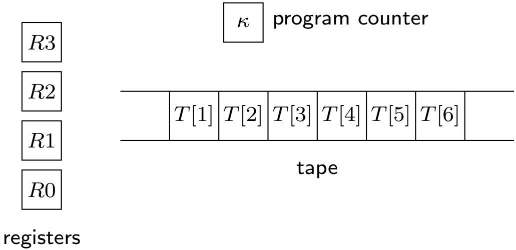
The goal here is to show **how to extend the abstract Turing Machine to a higher level concept**, closer to our *computers*.

Random Access Turing Machines

- Random Access Memory
  - access any position of the tape in a single step

Random Access Turing Machines

- Random Access Memory
  - access any position of the tape in a single step
- we also need:
  - finite number of *registers* $\rightarrow$ manipulate addresses of the tape
  - *program counter* $\rightarrow$ current instruction to execute



- program: a set of instructions

Random Access Turing Machines: Instructions set

instruction	**operand**	**semantics**
read	j	$R_0 \leftarrow T[R_j]$
write	j	$T[R_j] \leftarrow R_0$
store	j	$R_j \leftarrow R_0$
load	j	$R_0 \leftarrow R_j$
load	$= c$	$R_0 = c$
add	j	$R_0 \leftarrow R_0 + R_j$
add	$= c$	$R_0 \leftarrow R_0 + c$
sub	j	$R_0 \leftarrow \max\{R_0 - R_j, 0\}$
sub	$= c$	$R_0 \leftarrow \max\{R_0 - c, 0\}$
half		$R_0 \leftarrow \lfloor \frac{R_0}{2} \rfloor$
jump	s	$\kappa \leftarrow s$
jpos	s	if $R_0 > 0$ then $\kappa \leftarrow s$
jzero	s	if $R_0 = 0$ then $\kappa \leftarrow s$
halt		$\kappa = 0$

- register R_0: *accumulator*

Random Access Turing Machines: Formal definition

A Random Access Turing Machine is a pair $M = (k, \Pi)$, where

- $k > 0$ is the finite number of registers, and
- $\Pi = (\pi_1, \pi_2, \ldots, \pi_p)$ is a finite sequence of instructions (program).

Random Access Turing Machines: Formal definition

A Random Access Turing Machine is a pair $M = (k, \Pi)$, where

- $k > 0$ is the finite number of registers, and
- $\Pi = (\pi_1, \pi_2, \ldots, \pi_p)$ is a finite sequence of instructions (program).

Notations

- the last instruction π_p is always a *halt* instruction
- $(\kappa; R_0, R_1, \ldots, R_{k-1}; T)$: a **configuration**, where
  - κ: program counter
  - R_j, $0 \leq j < k$: the current value of register j
  - T: the contents of the tape
    (each $T[i]$ contains a non-negative integer, i.e. $T[i] \in \mathbb{N}$)
- **halted configuration:** $\kappa = 0$

Example 1 – write the configurations

```
1: load 1
2: add 2
3: sub =1
4: store 1
5: halt
```

$$(1; 0, 5, 3; \emptyset)$$

Example 1 – write the configurations

1: load 1
2: add 2
3: sub =1
4: store 1
5: halt

$(1; 0, 5, 3; \emptyset)$

$$\begin{aligned}(1; 0, 5, 3; \emptyset) \quad &\vdash \quad (2; 5, 5, 3; \emptyset) \vdash (3; 8, 5, 3; \emptyset) \vdash (4; 7, 5, 3; \emptyset) \\ &\vdash \quad (5; 7, 7, 3; \emptyset) \vdash (0; 7, 7, 3; \emptyset)\end{aligned}$$

Example 1 – write the configurations

1: load 1
2: add 2
3: sub =1
4: store 1
5: halt

$(1; 0, 5, 3; \emptyset)$

$$\begin{aligned} (1; 0, 5, 3; \emptyset) \quad &\vdash \quad (2; 5, 5, 3; \emptyset) \vdash (3; 8, 5, 3; \emptyset) \vdash (4; 7, 5, 3; \emptyset) \\ &\vdash \quad (5; 7, 7, 3; \emptyset) \vdash (0; 7, 7, 3; \emptyset) \end{aligned}$$

$R_1 \leftarrow R_2 + R_1 - 1$

Example 2

1: load 1
2: jzero 6
3: sub =3
4: store 1
5: jump 2
6: halt

$(1; 0, 7; \emptyset)$

$$
\begin{aligned}
(1; 0, 7; \emptyset) \quad &\vdash \quad (2; 7, 7; \emptyset) \vdash (3; 7, 7; \emptyset) \vdash (4; 4, 7; \emptyset) \vdash (5; 4, 4; \emptyset) \\
&\vdash \quad (2; 4, 4; \emptyset) \vdash (3; 4, 4; \emptyset) \vdash (4; 1, 4; \emptyset) \vdash (5; 1, 1; \emptyset) \\
&\vdash \quad (2; 1, 1; \emptyset) \vdash (3; 1, 1; \emptyset) \vdash (4; 0, 1; \emptyset) \vdash (5; 0, 0; \emptyset) \\
&\vdash \quad (2; 0, 0; \emptyset) \vdash (6; 0, 0; \emptyset) \vdash (0; 0, 0; \emptyset)
\end{aligned}
$$

while $R_1 > 0$ do $R_1 \leftarrow R_1 - 3$

Exercise

- Write a program for a Random Access Turing Machine that multiplies two integers.

  HINT: assume that the initial configuration is $(1; 0, a_1, a_2, 0; \emptyset)$

Power of the Random Access Turing Machines

Theorem

Every Random Access Turing Machine $M = (\kappa, \Pi)$ has an equivalent single tape Turing Machine $M' = (K, \Sigma, \Gamma, \delta, s, H)$.

If M halts on input of size n after t steps, then M' halts on after $O(poly(t, n))$ steps.

Power of the Random Access Turing Machines

Theorem

Every Random Access Turing Machine $M = (\kappa, \Pi)$ has an equivalent single tape Turing Machine $M' = (K, \Sigma, \Gamma, \delta, s, H)$.

If M halts on input of size n after t steps, then M' halts on after $O(poly(t, n))$ steps.

Proof (sketch):

- we pass through the multiple tape model
  - use $k + 3$ tapes
  - tape 1: the contents of the tape of M
  - tape 2: the program counter
  - tape 3: auxiliary
  - tape $3 + j$, $1 \leq j \leq k$: corresponds to R_j
- add appropriate delimiters
- simulate instructions

Proof (sketch):

- add 4
  1. copy the contents of tape 8 (R_4) on tape 3 (auxiliary)
  2. use the Turing Machine with two tapes seen in previous lecture to add the numbers in tapes 8 and 4 (R_0)
  3. store the result in tape 4
  4. increase the contents of tape 2 (program counter) by 1

Proof (sketch):

- add 4
  1. copy the contents of tape 8 (R_4) on tape 3 (auxiliary)
  2. use the Turing Machine with two tapes seen in previous lecture to add the numbers in tapes 8 and 4 (R_0)
  3. store the result in tape 4
  4. increase the contents of tape 2 (program counter) by 1

- write 2
  1. move the head of tape 1 (tape of M) to the position (address) indicted by tape 6 (R_2)
  2. copy the contents of tape 4 (R_0) in the indicated position of tape 1
  3. increase the contents of tape 2 (program counter) by 1

Proof (sketch):

- add 4
  1. copy the contents of tape 8 (R_4) on tape 3 (auxiliary)
  2. use the Turing Machine with two tapes seen in previous lecture to add the numbers in tapes 8 and 4 (R_0)
  3. store the result in tape 4
  4. increase the contents of tape 2 (program counter) by 1

- write 2
  1. move the head of tape 1 (tape of M) to the position (address) indicted by tape 6 (R_2)
  2. copy the contents of tape 4 (R_0) in the indicated position of tape 1
  3. increase the contents of tape 2 (program counter) by 1

- jpos 19
  1. scan tape 4 (R_0)
  2. if all cells are zero then increase the contents of tape 2 (program counter) by 1
  3. else replace the contents of tape 2 by 19

Proof (sketch):

- the size of the contents of all tapes cannot be bigger that a polynomial to t and n
  - initially: n
  - at each step: the size of the contents is increased by at most a constant c (instruction add $= c$)

Proof (sketch):

- the size of the contents of all tapes cannot be bigger that a polynomial to t and n
  - initially: n
  - at each step: the size of the contents is increased by at most a constant c (instruction add $= c$)
- each instruction can be implemented in time polynomial in the size of the contents of all tapes

Proof (sketch):

- the size of the contents of all tapes cannot be bigger that a polynomial to t and n
  - initially: n
  - at each step: the size of the contents is increased by at most a constant c (instruction add $= c$)
- each instruction can be implemented in time polynomial in the size of the contents of all tapes

- Thus, complexity polynomial in t and n

Proof (sketch):

- the size of the contents of all tapes cannot be bigger that a polynomial to t and n
  - initially: n
  - at each step: the size of the contents is increased by at most a constant c (instruction add $= c$)
- each instruction can be implemented in time polynomial in the size of the contents of all tapes
- Thus, complexity polynomial in t and n

Random Access is not more powerful !!!