V. Register Machine

Yuxi Fu

BASICS, Shanghai Jiao Tong University

Register Machines are more abstract than Turing Machines.

Register Machine Models can be classified into three groups:

- CM (Counter Machine Model)
  - Unlimited Register Machine Model
- RAM (Random Access Machine Model)
- RASP (Random Access Stored Program Machine Model)

Synopsis

1. Unlimited Register Machine
2. Definability in URM
3. Simulation of TM by URM

1. Unlimited Register Machine

Unlimited Register Machine Model

Unlimited Register Machines are introduced by Shepherdson and Sturgis in

Computability and Recursive Functions.

Journal of Symbolic Logic, 32:1-63, 1965.

Register

An Unlimited Register Machine (URM) has an infinite number of register labeled $R_1, R_2, R_3, \ldots$.

r_1	r_2	r_3	r_4	r_5	r_6	r_7	...
R_1	R_2	R_3	R_4	R_5	R_6	R_7	...

Every register can hold a natural number at any moment.

The registers can be equivalently written as for example

$$[r_1, r_2, r_3]_1^3[r_4]_4^4[r_5, r_6, r_7]_5^7[0, 0, 0, \ldots]_8^\infty$$

or simply

$$[r_1, r_2, r_3]_1^3[r_4]_4^4[r_5, r_6, r_7]_5^7.$$

Program

A URM also has a program, which is a finite list of instructions.

Instruction

type	instruction	response of URM
Zero	$Z(n)$	Replace r_n by 0.
Successor	$S(n)$	Add 1 to r_n.
Transfer	$T(m, n)$	Copy r_m to R_n.
Jump	$J(m, n, q)$	If $r_m = r_n$, go to the q-th instruction; otherwise go to the next instruction.

Computation

Registers:

9	7	0	0	0	0	0	...
R_1	R_2	R_3	R_4	R_5	R_6	R_7	

Program:

$I_1: \; J(1,2,6)$
$I_2: \; S(2)$
$I_3: \; S(3)$
$I_4: \; J(1,2,6)$
$I_5: \; J(1,1,2)$
$I_6: \; T(3,1)$

Configuration and Computation

Configuration: register contents + current instruction number.

Initial configuration, computation, final configuration.

Some Notation

Suppose P is the program of a URM and $a_1, a_2, a_3, \ldots$ are the numbers stored in the registers.

- $P(a_1, a_2, a_3, \ldots)$ is the initial configuration.
- $P(a_1, a_2, a_3, \ldots) \downarrow$ means that the computation converges.
- $P(a_1, a_2, a_3, \ldots) \uparrow$ means that the computation diverges.
- $P(a_1, a_2, \ldots, a_m)$ is $P(a_1, a_2, \ldots, a_m, 0, 0, \ldots)$.

Every URM uses only a fixed finite number of registers, no matter how large an input number is.

2. Definability in URM

URM-Computable Function

Let $f(\widetilde{x})$ be an n-ary partial function.

What does it mean that a URM computes $f(\widetilde{x})$?

URM-Computable Function

Suppose P is the program of a URM and $a_1, \ldots, a_n, b \in \omega$.

The computation $P(a_1, \ldots, a_n)$ converges to b if $P(a_1, \ldots, a_n) \downarrow$ and $r_1 = b$ in the final configuration.

In this case we write $P(a_1, \ldots, a_n) \downarrow b$.

P URM-computes f if, for all $a_1, \ldots, a_n, b \in \omega$, $P(a_1, \ldots, a_n) \downarrow b$ iff $f(a_1, \ldots, a_n) = b$.

The function f is URM-definable if there is a program that URM-computes f.

Example of URM

Construct a URM that computes $x + y$.

Example of URM

Construct a URM that computes $x + y$.

$I_1 :\ J(3, 2, 5)$
$I_2 :\ S(1)$
$I_3 :\ S(3)$
$I_4 :\ J(1, 1, 1)$

Example of URM

Construct a URM that computes $x \dot{-} 1 = \begin{cases} x - 1, & \text{if } x > 0, \\ 0, & \text{if } x = 0. \end{cases}$

Example of URM

Construct a URM that computes $x \dot{-} 1 = \begin{cases} x-1, & \text{if } x > 0, \\ 0, & \text{if } x = 0. \end{cases}$

$I_1 :\ J(1,4,8)$
$I_2 :\ S(3)$
$I_3 :\ J(1,3,7)$
$I_4 :\ S(2)$
$I_5 :\ S(3)$
$I_6 :\ J(1,1,3)$
$I_7 :\ T(2,1)$

Example of URM

Construct a URM that computes

$$x \div 2 = \begin{cases} x/2, & \text{if } x \text{ is even,} \\ \text{undefined}, & \text{if } x \text{ is odd.} \end{cases}$$

Example of URM

Construct a URM that computes

$$x \div 2 = \begin{cases} x/2, & \text{if } x \text{ is even,} \\ \text{undefined}, & \text{if } x \text{ is odd.} \end{cases}$$

$I_1 : \ J(1,2,6)$
$I_2 : \ S(3)$
$I_3 : \ S(2)$
$I_4 : \ S(2)$
$I_5 : \ J(1,1,1)$
$I_6 : \ T(3,1)$

Function Defined by Program

$$f_P^n(a_1,\ldots,a_n) = \begin{cases} b, & \text{if } P(a_1,\ldots,a_n)\downarrow b, \\ \text{undefined}, & \text{if } P(a_1,\ldots,a_n)\uparrow. \end{cases}$$

Program in Standard Form

A program $P = I_1, \ldots, I_s$ is in standard form if, for every jump instruction $J(m, n, q)$ we have $q \leq s + 1$.

For every program there is a program in standard form that computes the same function.

We will focus exclusively on programs in standard form.

Program Composition

Given Programs P and Q, how do we construct the sequential composition $P; Q$?

The jump instructions of P and Q must be modified.

Some Notations

Suppose the program P computes f.

Let $\rho(P)$ be the least number i such that the register R_i is not used by the program P.

Some Notations

The notation $P[l_1, \ldots, l_n \to l]$ stands for the following program

$$\begin{array}{rcl} I_1 & : & T(l_1, 1) \\ & \vdots & \\ I_n & : & T(l_n, n) \\ I_{n+1} & : & Z(n+1) \\ & \vdots & \\ I_{\rho(P)} & : & Z(\rho(P)) \\ \_ & : & P \\ \_ & : & T(1, l) \end{array}$$

Definability of Initial Function

Fact. The initial functions are URM-definable.

Definability of Composition

Fact. If $f(y_1, \ldots, y_k)$ and $g_1(\widetilde{x}), \ldots, g_k(\widetilde{x})$ are URM-definable, then the composition function $h(\widetilde{x})$ given by

$$h(\widetilde{x}) \simeq f(g_1(\widetilde{x}), \ldots, g_k(\widetilde{x}))$$

is URM-definable.

Definability of Composition

Let $F, G_1, \ldots, G_k$ be programs that compute $f, g_1, \ldots, g_k$.
Let m be $\max\{n, k, \rho(F), \rho(G_1), \ldots, \rho(G_k)\}$.

Registers:

$$[\ldots]_1^m [\widetilde{x}]_{m+1}^{m+n} [g_1(\widetilde{x})]_{m+n+1}^{m+n+1} \ldots [g_k(\widetilde{x})]_{m+n+k}^{m+n+k}$$

Definability of Composition

The program for h:

$$
\begin{array}{rcl}
I_1 & : & T(1, m+1) \\
 & \vdots & \\
I_n & : & T(n, m+n) \\
I_{n+1} & : & G_1[m+1, m+2, \ldots, m+n \ \rightarrow \ m+n+1] \\
 & \vdots & \\
I_{n+k} & : & G_k[m+1, m+2, \ldots, m+n \ \rightarrow \ m+n+k] \\
I_{n+k+1} & : & F[m+n+1 \ldots, m+n+k \ \rightarrow \ 1]
\end{array}
$$

Definability of Recursion

Fact. Suppose $f(\widetilde{x})$ and $g(\widetilde{x}, y, z)$ are URM-definable.
The recursion function $h(\widetilde{x}, y)$ defined by the following recursion

$$\begin{aligned} h(\widetilde{x}, 0) &\simeq f(\widetilde{x}), \\ h(\widetilde{x}, y+1) &\simeq g(\widetilde{x}, y, h(\widetilde{x}, y)) \end{aligned}$$

is URM-definable.

Definability of Recursion

Let F compute f and G compute g. Let m be $\max\{n, \rho(F), \rho(G)\}$.

Registers: $[\ldots]_1^m[\widetilde{x}]_{m+1}^{m+n}[y]_{m+n+1}^{m+n+1}[k]_{m+n+2}^{m+n+2}[h(\widetilde{x},k)]_{m+n+3}^{m+n+3}$.

Program:

$$
\begin{array}{lcl}
I_1 & : & T(1, m+1) \\
 & & \vdots \\
I_{n+1} & : & T(n+1, m+n+1) \\
I_{n+2} & : & F[1, 2, \ldots, n \ \rightarrow \ m+n+3] \\
I_{n+3} & : & J(m+n+2, m+n+1, n+7) \\
I_{n+4} & : & G[m+1, \ldots, m+n, m+n+2, m+n+3 \ \rightarrow \ m+n+3] \\
I_{n+5} & : & S(m+n+2) \\
I_{n+6} & : & J(1, 1, n+3) \\
I_{n+7} & : & T(m+n+3, 1)
\end{array}
$$

Definability of Minimization

Fact. If $f(\widetilde{x}, y)$ is URM-definable, then the minimization function $\mu y(f(\widetilde{x}, y) = 0)$ is URM-definable.

Definability of Minimization

Suppose F computes $f(\widetilde{x}, y)$. Let m be $\max\{n+1, \rho(F)\}$.

Registers: $[\ldots]_1^m[\widetilde{x}]_{m+1}^{m+n}[k]_{m+n+1}^{m+n+1}[0]_{m+n+2}^{m+n+2}$.

Program:

$$
\begin{array}{rcl}
I_1 & : & T(1, m+1) \\
 & \vdots & \\
I_n & : & T(n, m+n) \\
I_{n+1} & : & F[m+1, m+2, \ldots, m+n+1 \ \rightarrow \ 1] \\
I_{n+2} & : & J(1, m+n+2, n+5) \\
I_{n+3} & : & S(m+n+1) \\
I_{n+4} & : & J(1, 1, n+1) \\
I_{n+5} & : & T(m+n+1, 1)
\end{array}
$$

Main Result

Theorem. All recursive functions are URM-definable.

3. Simulation of TM by URM

Simulating TM by URM

Suppose $\mathbb{M}$ is a 3-tape TM with the alphabet $\{0, 1, \Box, \triangleright\}$.

The URM that simulates $\mathbb{M}$ can be designed as follows:

- Suppose that R_m is the right most register that is used by a program calculating $x \dot{-} 1$.
- The head positions are stored in $R_{m+1}, R_{m+2}, R_{m+3}$.
- The three binary strings in the tapes are stored respectively in
  $R_{m+4}, R_{m+7}, R_{m+10}, \ldots,$
  $R_{m+5}, R_{m+8}, R_{m+11}, \ldots,$
  $R_{m+6}, R_{m+9}, R_{m+12}, \ldots.$
- The states of $\mathbb{M}$ are encoded by the states of the URM.
- The transition function of $\mathbb{M}$ can be easily simulated by the program of the URM.

Exercise. Describe an algorithm that transforms a TM to a URM.