Computability Theory II

Unlimited Register Machine

Guoqiang Li

Shanghai Jiao Tong University

Sep. 26, 2014

Review Tips

Computable Functions

- In a formal theory of computability, every problem instance can be represented by a number and every number represents a problem instance.
- A problem is a function $f : \mathbb{N} \rightarrow \mathbb{N}$ from numbers to numbers.
- A problem is computable if it can be calculated by a program.

Decision Problem

A problem $f : \mathbb{N} \rightarrow \mathbb{N}$ is a decision problem if the range $ran(f)$ of f is $\{0, 1\}$, where 1 denotes a 'yes' answer and 0 a 'no' answer.

A decision problem g can be identified with the set $\{n \mid g(n) = 1\}$.

Conversely a subset A of $\mathbb{N}$ can be seen as a decision problem via the characteristic function of A:

$$c_A(n) = \begin{cases} 1, & \text{if } x \in A, \\ 0, & \text{otherwise.} \end{cases}$$

Decision Problem as Predicate

A decision problem can be stated as a predicate $P(x)$ on number.

It relates to the problem-as-function viewpoint by the following characteristic function of $P(x)$:

$$c_P(n) = \begin{cases} 1, & \text{if } P(n) \text{ is valid,} \\ 0, & \text{otherwise.} \end{cases}$$

$$\begin{aligned}\text{Decision Problem} \quad &\Leftrightarrow \quad \text{Subset of } \mathbb{N} \\ &\Leftrightarrow \quad \text{Predicate on } \mathbb{N}\end{aligned}$$

Register Machine

Remark

Register Machines are more advanced than Turing Machines.

Remark

Register Machine Models can be classified into three groups:

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

Synopsis

1. Unlimited Register Machine
2. Definability in URM

Unlimited Register Machine

Unlimited Register Machine Model

The Unlimited Register Machine Model belongs to the CM class.

Computability and Recursive Functions, by J. Shepherdson and H. Sturgis, in 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	$\ldots$

R_1 R_2 R_3 R_4 R_5 R_6 R_7 $\ldots$

Every register can hold a natural number at any moment.

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	$\ldots$
R_1	R_2	R_3	R_4	R_5	R_6	R_7	$\ldots$

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 the 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.

Program Rules

- $P = \{I_1, I_2, \cdots, I_s\} \to \text{URM}$.
- URM starts by obeying instruction I_1.
- When URM finishes obeying I_k, it proceeds to the next instruction in the computation,
  - if I_k is not a jump instruction, then the next instruction is I_{k+1};
  - if $I_k = J(m, n, q)$ then next instruction is
    - I_q, if $r_m = r_n$; or
    - I_{k+1}, otherwise.
- Computation stops when the next instruction is I_v, where $v > s$.
  - if $k = s$, and I_s is an arithmetic instruction;
  - if $I_k = J(m, n, q)$, $r_m = r_n$ and $q > s$;
  - if $I_k = J(m, n, q)$, $r_m \neq r_n$ and $k = s$.

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.

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.

- $P(a_1, a_2, \ldots, a_m)$ is $P(a_1, a_2, \ldots, a_m, 0, 0, \ldots)$.
- $P(a_1, a_2, a_3, \ldots)$ 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.

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 $a_1, \ldots, a_n, b \in \mathbb{N}$.

URM-Computable Function

Suppose P is the program of a URM and $a_1, \ldots, a_n, b \in \mathbb{N}$.

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.

URM-Computable Function

Suppose P is the program of a URM and $a_1, \ldots, a_n, b \in \mathbb{N}$.

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 $P(a_1, \ldots, a_n) \downarrow b$.

URM-Computable Function

Suppose P is the program of a URM and $a_1, \ldots, a_n, b \in \mathbb{N}$.

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 $a_1, \ldots, a_n, b \in \mathbb{N}$, $P(a_1, \ldots, a_n) \downarrow b$ iff $f(a_1, \ldots, a_n) = b$.

URM-Computable Function

Suppose P is the program of a URM and $a_1, \ldots, a_n, b \in \mathbb{N}$.

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 $a_1, \ldots, a_n, b \in \mathbb{N}$, $P(a_1, \ldots, a_n) \downarrow b$ iff $f(a_1, \ldots, a_n) = b$.

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

We shall abbreviate “URM-computable” to “computable”.

Let

$$\mathscr{C}$$

be the set of computable functions and

$$\mathscr{C}_n$$

be the set of n-ary computable functions.

Example of URM I

Construct a URM that computes $x + y$.

Example of URM I

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 II

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 II

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 III

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 III

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

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

Example of URM IV

Construct a URM that computes $f(x) = \lfloor 3x/4 \rfloor$

Example of URM IV

Construct a URM that computes $f(x) = \lfloor 3x/4 \rfloor$

I_1 Z(2)
I_2 Z(3)
I_3 Z(4)
I_4 J(1,2,10)
I_5 S(2)
I_6 S(3)
I_7 S(3)
I_8 S(3)
I_9 J(1,1,4)
I_{10} Z(2)
I_{11} J(2,3,21)
I_{12} S(2)
I_{13} J(2,3,21)
I_{14} S(2)
I_{15} J(2,3,21)
I_{16} S(2)
I_{17} J(2,3,21)
I_{18} S(2)
I_{19} S(4)
I_{20} J(1,1,11)
I_{21} T(4,1)

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 $q \leq s + 1$.

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.

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 $P; Q$?

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

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