Unlimited Register Machine

Xiaofeng Gao

Department of Computer Science and Engineering
Shanghai Jiao Tong University, P.R.China

CS363-Computability Theory

* Special thanks is given to Prof. Yuxi Fu for sharing his teaching materials.
Outline

1. Effective Procedures
   - Basic Concepts
   - Computable Function

2. Unlimited Register Machine
   - Definition
   - Instruction
   - An Example

3. Computable and Decidable
   - URM-Computable Function
   - Decidable and Computable

4. Notations
   - Register Machine
   - Joining Programs Together
Outline

1. Effective Procedures
   - Basic Concepts
   - Computable Function

2. Unlimited Register Machine
   - Definition
   - Instruction
   - An Example

3. Computable and Decidable
   - URM-Computable Function
   - Decidable and Computable

4. Notations
   - Register Machine
   - Joining Programs Together
What is Effective Procedure

- Methods for addition, multiplication · · ·
  - Given \( n \), finding the \( n \)th prime number.
  - Differentiating a polynomial.
  - Finding the highest common factor of two numbers \( HCF(x, y) \rightarrow \) Euclidean algorithm
  - Given two numbers \( x, y \), deciding whether \( x \) is a multiple of \( y \).

- Their implementation requires no ingenuity, intelligence, inventiveness.
Intuitive Definition

An *algorithm* or *effective procedure* is a *mechanical rule*, or *automatic method*, or *programme* for performing some mathematical operations.

Blackbox: input $\rightarrow$ output
What is “effective procedure”?

**An Example**: Consider the function $g(n)$ defined as follows:

$$g(n) = \begin{cases} 
1, & \text{if there is a run of exactly } n \text{ consecutive 7's in the decimal expansion of } \pi, \\
0, & \text{otherwise.}
\end{cases}$$

**Question**: Is $g(n)$ effective?
What is “effective procedure”?

An Example: Consider the function $g(n)$ defined as follows:

$$g(n) = \begin{cases} 
1, & \text{if there is a run of exactly } n \text{ consecutive 7's in the decimal expansion of } \pi, \\
0, & \text{otherwise.}
\end{cases}$$

Question: Is $g(n)$ effective?

▷ The answer is unknown $\neq$ the answer is negative.
What is “effective procedure”?

**An Example:** Consider the function $g(n)$ defined as follows:

$$g(n) = \begin{cases} 
1, & \text{if there is a run of exactly } n \text{ consecutive 7's} \\
0, & \text{otherwise.}
\end{cases}$$

Question: Is $g(n)$ effective?

▷ The answer is unknown ≠ the answer is negative.

**Other Examples:**

- *Theorem Proving* is in general not effective/algorithmic.
- *Proof Verification* is effective/algorithmic.
Algorithm

An algorithm is a procedure that consists of a finite set of *instructions* which, given an *input* from some set of possible inputs, enables us to obtain an *output* through a systematic execution of the instructions that *terminates* in a finite number of steps.
Outline

1. Effective Procedures
   - Basic Concepts
   - Computable Function

2. Unlimited Register Machine
   - Definition
   - Instruction
   - An Example

3. Computable and Decidable
   - URM-Computable Function
   - Decidable and Computable

4. Notations
   - Register Machine
   - Joining Programs Together
Computable Function

When an algorithm or effective procedure is used to calculate the value of a numerical function then the function in question is **effectively calculable** (or **algorithmically computable**, **effectively computable**, **computable**).
Computable Function

When an algorithm or effective procedure is used to calculate the value of a numerical function then the function in question is effectively calculable (or algorithmically computable, effectively computable, computable).

Examples:

- \( HCF(x, y) \) is computable;
- \( g(n) \) is non-computable.
Outline

1 Effective Procedures
   - Basic Concepts
   - Computable Function

2 Unlimited Register Machine
   - Definition
   - Instruction
   - An Example

3 Computable and Decidable
   - URM-Computable Function
   - Decidable and Computable

4 Notations
   - Register Machine
   - Joining Programs Together
An Unlimited Register Machine (URM) is an idealized computer.

- No limitation in the size of the numbers it can receive as input.
- No limitation in the amount of working space available.
- Inputs and outputs are restricted to natural numbers. (coding for others)

From Shepherdson & Sturgis [1963]’s description.

A URM has an infinite number of register labeled $R_1, R_2, R_3, \ldots$.

\[
\begin{array}{cccccccc}
R_1 & R_2 & R_3 & R_4 & R_5 & R_6 & R_7 & \cdots \\
\hline
r_1 & r_2 & r_3 & r_4 & r_5 & r_6 & r_7 & \cdots
\end{array}
\]

Every register can hold a natural number at any moment.

The registers can be equivalently written as for example

\[
[r_1, r_2, r_3]_1^3[r_4]_4^4[r_5, r_6, r_7]_5^\infty
\]

or simply

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

1. Effective Procedures
   - Basic Concepts
   - Computable Function

2. Unlimited Register Machine
   - Definition
   - Instruction
   - An Example

3. Computable and Decidable
   - URM-Computable Function
   - Decidable and Computable

4. Notations
   - Register Machine
   - Joining Programs Together
A URM also has a program, which is a finite list of instructions.

An instruction is a recognized simple operations (calculation with numbers) to alter the contents of the registers. \((I_1, \cdots, I_s)\)
## Instruction

<table>
<thead>
<tr>
<th>Type</th>
<th>Instruction</th>
<th>Response of the URM</th>
</tr>
</thead>
</table>

---

CSC363-Computability Theory @ SJTU

Xiaofeng Gao
### Instruction

<table>
<thead>
<tr>
<th>Type</th>
<th>Instruction</th>
<th>Response of the URM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero</td>
<td>$Z(n)$</td>
<td>Replace $r_n$ by 0. ($0 \rightarrow R_n$, or $r_n := 0$)</td>
</tr>
</tbody>
</table>
### Instruction

<table>
<thead>
<tr>
<th>Type</th>
<th>Instruction</th>
<th>Response of the URM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero</td>
<td>$Z(n)$</td>
<td>Replace $r_n$ by 0. (0 $\rightarrow R_n$, or $r_n := 0$)</td>
</tr>
<tr>
<td>Successor</td>
<td>$S(n)$</td>
<td>Add 1 to $r_n$. ($r_n + 1 \rightarrow R_n$, or $r_n := r_n + 1$)</td>
</tr>
</tbody>
</table>
### Instruction

<table>
<thead>
<tr>
<th>Type</th>
<th>Instruction</th>
<th>Response of the URM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero</td>
<td>$Z(n)$</td>
<td>Replace $r_n$ by 0. ($0 \rightarrow R_n$, or $r_n := 0$)</td>
</tr>
<tr>
<td>Successor</td>
<td>$S(n)$</td>
<td>Add 1 to $r_n$. ($r_n + 1 \rightarrow R_n$, or $r_n := r_n + 1$)</td>
</tr>
<tr>
<td>Transfer</td>
<td>$T(m, n)$</td>
<td>Copy $r_m$ to $R_n$. ($r_m \rightarrow R_n$, or $r_n := r_m$)</td>
</tr>
</tbody>
</table>
## Instruction

<table>
<thead>
<tr>
<th>Type</th>
<th>Instruction</th>
<th>Response of the URM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero</td>
<td>$Z(n)$</td>
<td>Replace $r_n$ by 0. ($0 \rightarrow R_n$, or $r_n := 0$)</td>
</tr>
<tr>
<td>Successor</td>
<td>$S(n)$</td>
<td>Add 1 to $r_n$. ($r_n + 1 \rightarrow R_n$, or $r_n := r_n + 1$)</td>
</tr>
<tr>
<td>Transfer</td>
<td>$T(m, n)$</td>
<td>Copy $r_m$ to $R_n$. ($r_m \rightarrow R_n$, or $r_n := r_m$)</td>
</tr>
<tr>
<td>Jump</td>
<td>$J(m, n, q)$</td>
<td>If $r_m = r_n$, go to the $q$-th instruction; otherwise go to the next instruction.</td>
</tr>
</tbody>
</table>
### Instruction

<table>
<thead>
<tr>
<th>Type</th>
<th>Instruction</th>
<th>Response of the URM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero</td>
<td>$Z(n)$</td>
<td>Replace $r_n$ by 0. $(0 \rightarrow R_n$, or $r_n := 0)$</td>
</tr>
<tr>
<td>Successor</td>
<td>$S(n)$</td>
<td>Add 1 to $r_n$. $(r_n + 1 \rightarrow R_n$, or $r_n := r_n + 1)$</td>
</tr>
<tr>
<td>Transfer</td>
<td>$T(m, n)$</td>
<td>Copy $r_m$ to $R_n$. $(r_m \rightarrow R_n$, or $r_n := r_m)$</td>
</tr>
<tr>
<td>Jump</td>
<td>$J(m, n, q)$</td>
<td>If $r_m = r_n$, go to the $q$-th instruction; otherwise go to the next instruction.</td>
</tr>
</tbody>
</table>

$Z(n)$, $S(n)$, $T(m, n)$ are arithmetic instructions.
Outline

1. Effective Procedures
   - Basic Concepts
   - Computable Function

2. Unlimited Register Machine
   - Definition
   - Instruction
   - An Example

3. Computable and Decidable
   - URM-Computable Function
   - Decidable and Computable

4. Notations
   - Register Machine
   - Joining Programs Together
Example: The initial registers are:

\[
\begin{array}{cccccccc}
R_1 & R_2 & R_3 & R_4 & R_5 & R_6 & R_7 & \cdots \\
9 & 7 & 0 & 0 & 0 & 0 & 0 & \cdots \\
\end{array}
\]

The program is:

\[
\begin{align*}
I_1 &: J(1, 2, 6) \\
I_2 &: S(2) \\
I_3 &: S(3) \\
I_4 &: J(1, 2, 6) \\
I_5 &: J(1, 1, 2) \\
I_6 &: T(3, 1)
\end{align*}
\]
Configuration and Computation

Configuration:
the contents of the registers + the current instruction number.

Initial configuration, computation, final configuration.
Operation of URM under a program $P$

- $P = \{I_1, I_2, \cdots, I_s\} \rightarrow \text{URM}$
- 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 (1) $I_q$, if $r_m = r_n$; or (2) $I_{k+1}$, 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 $k = s$. 
Flow Diagram

- \( J(m, m, q) \) is an unconditional jump
- Computations that never stop

Typical configuration:

\[
\begin{array}{c|c|c|c|c|}
R_1 & R_2 & R_3 \\
\hline
x & y & z & \ldots \\
\end{array}
\]

After k cycles round the loop in this program:

\[
\begin{array}{c|c|c|c|c|c|c|c|c|c|c}
R_1 & R_2 & R_3 \\
\hline
x & y+k & z+k & \ldots \\
\end{array}
\]

If \( x = y + k \):

\[
\begin{array}{c|c|c|c|c|c|c|c|c|c|c}
R_1 & R_2 & R_3 \\
\hline
z+k & y+k & z+k & \ldots \\
\end{array}
\]
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 $P(a_1, a_2, \ldots, a_m, 0, 0, \ldots)$. 
Outline

1. Effective Procedures
   - Basic Concepts
   - Computable Function

2. Unlimited Register Machine
   - Definition
   - Instruction
   - An Example

3. Computable and Decidable
   - URM-Computable Function
   - Decidable and Computable

4. Notations
   - Register Machine
   - Joining Programs Together
<table>
<thead>
<tr>
<th>Effective Procedures</th>
<th>Unlimited Register Machine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computable and Decidable</td>
<td>Notations</td>
</tr>
<tr>
<td>URM-Computable Function</td>
<td>Decidable and Computable</td>
</tr>
</tbody>
</table>

**URM-Computable Function**
URM-Computable Function

What does it mean that a URM computes a (partial) \( n \)-ary function \( f \)?
What does it mean that a URM computes a (partial) $n$-ary function $f$?

Let $P$ be the program of a URM and $a_1, \ldots, a_n, b \in \mathbb{N}$. When 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. We write $P(a_1, \ldots, a_n) \downarrow b$.

- $P$ URM-computes $f$ if, for all $a_1, \ldots, a_n, b \in \mathbb{N}$,

$$P(a_1, \ldots, a_n) \downarrow b \text{ iff } f(a_1, \ldots, a_n) = b$$

- Function $f$ is URM-computable if there is a program that URM-computes $f$.

(We abbreviate “URM-computable” to “computable”)
Let

$\mathcal{C}$ be the set of computable functions and

$\mathcal{C}_n$ be the set of $n$-ary computable functions.
Examples

Construct a URM that computes $x + y$. 
Examples

Construct a URM that computes $x + y$.

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

Construct a URM that computes $x + y$.

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

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

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

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

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

\[ \begin{align*}
I_1 & : J(1, 4, 8) \\
I_2 & : S(3) \\
I_3 & : J(1, 3, 7) \\
I_4 & : S(2) \\
I_5 & : S(3) \\
I_6 & : J(1, 1, 3) \\
I_7 & : T(2, 1)
\end{align*} \]
Examples

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

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

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

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

\[
\begin{align*}
I_1 & : J(1, 2, 6) \\
I_2 & : S(3) \\
I_3 & : S(2) \\
I_4 & : S(2) \\
I_5 & : J(1, 1, 1) \\
I_6 & : T(3, 1)
\end{align*}
\]
Function Defined by Program

Given any program $P$ and $n \geq 1$, by thinking of the effect of $P$ on initial configurations of the form $a_1, \ldots, a_n, 0, 0, \cdots$, there is a unique $n$-ary function that $P$ computes, denoted by $f_P^{(n)}$.

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

Unlimited Register Machine

Computable and Decidable

Notations

Outline

1. Effective Procedures
   - Basic Concepts
   - Computable Function

2. Unlimited Register Machine
   - Definition
   - Instruction
   - An Example

3. Computable and Decidable
   - URM-Computable Function
   - Decidable and Computable

4. Notations
   - Register Machine
   - Joining Programs Together
Predicate and Decision Problem

The value of a predicate is either ‘true’ or ‘false’.

The answer of a decision problem is either ‘yes’ or ‘no’.

**Example:** Given two numbers \( x, y \), check whether \( x \) is a multiple of \( y \).

Input: \( x, y \);

Output: ‘Yes’ or ‘No’.

The operation amounts to calculation of the function

\[
f(x, y) = \begin{cases} 
1, & \text{if } x \text{ is a multiple of } y, \\
0, & \text{if otherwise.}
\end{cases}
\]

Thus the property or predicate ‘\( x \) is a multiple of \( y \)’ is algorithmically or effectively decidable, or just decidable if function \( f \) is computable.
Suppose that $M(x_1, \ldots, x_n)$ is an $n$-ary predicate of natural numbers. The characteristic function $c_M(x)$, where $x = x_1, \ldots, x_n$, is given by

$$f_{P}^{(n)}(a_1, \ldots, a_n) = \begin{cases} 
1, & \text{if } M(x) \text{ holds,} \\
0, & \text{if otherwise.}
\end{cases}$$

The predicate $M(x)$ is \textit{decidable} if $c_M$ is computable; it is \textit{undecidable} otherwise.
Suppose $D$ is an object domain. A \textbf{coding} of $D$ is an explicit and \textbf{effective injection} $\alpha : D \rightarrow \mathbb{N}$. We say that an object $d \in D$ is \textbf{coded} by the natural number $\alpha(d)$.

A function $f : D \rightarrow D$ extends to a numeric function $f^* : \mathbb{N} \rightarrow \mathbb{N}$. We say that $f$ is computable if $f^*$ is computable.

$$f^* = \alpha \circ f \circ \alpha^{-1}$$
Example

Consider the domain $\mathbb{Z}$. An explicit coding is given by the function $\alpha$ where

$$\alpha(n) = \begin{cases} 
2n, & \text{if } n \geq 0, \\
-2n - 1, & \text{if } n < 0.
\end{cases}$$

Then $\alpha^{-1}$ is given by

$$\alpha^{-1}(m) = \begin{cases} 
\frac{1}{2}m, & \text{if } m \text{ is even}, \\
-\frac{1}{2}(m + 1), & \text{if } m \text{ is odd}.
\end{cases}$$
Consider the function $f(x) = x - 1$ on $\mathbb{Z}$, then $f^* : \mathbb{N} \rightarrow \mathbb{N}$ is given by

$$f^*(x) = \begin{cases} 
1 & \text{if } x = 0 \text{ (i.e. } x = \alpha(0)), \\
x - 2 & \text{if } x > 0 \text{ and } x \text{ is even (i.e. } x = \alpha(n), n > 0), \\
x + 2 & \text{if } x \text{ is odd (i.e. } x = \alpha(n), n < 0). 
\end{cases}$$

It is a routine exercise to write a program that computes $f^*$, hence $x - 1$ is a computable function on $\mathbb{Z}$. 
Outline

1. Effective Procedures
   - Basic Concepts
   - Computable Function

2. Unlimited Register Machine
   - Definition
   - Instruction
   - An Example

3. Computable and Decidable
   - URM-Computable Function
   - Decidable and Computable

4. Notations
   - Register Machine
   - Joining Programs Together
Remark

Register Machines are more advanced than Turing Machines.
Remark

Register Machines are more advanced than Turing Machines.

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).
Remark

Register Machines are more advanced than Turing Machines.

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).

The Unlimited Register Machine Model belongs to the CM class.
Finiteness

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

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

This is a fine property of Counter Machine Model.
Outline

1. Effective Procedures
   - Basic Concepts
   - Computable Function

2. Unlimited Register Machine
   - Definition
   - Instruction
   - An Example

3. Computable and Decidable
   - URM-Computable Function
   - Decidable and Computable

4. Notations
   - Register Machine
   - Joining Programs Together
Given Programs $P$ and $Q$, how do we construct the sequential composition $P; Q$?

The jump instructions of $P$ and $Q$ must be modified.
Sequential 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.

**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 $q \leq s + 1$. 
Lemma

For any program $P$ there is a program $P^*$ in standard form such that any computation under $P^*$ is identical to the corresponding computation under $P$. In particular, for any $a_1, \cdots, a_n, b$,

$$P(a_1, \cdots, a_n) \downarrow b \text{ if and only if } P^*(a_1, \cdots, a_n) \downarrow b,$$

and hence $f_P^{(n)} = f_{P^*}^{(n)}$ for every $n > 0$. 

Proof

Suppose that $P = I_1, I_2, \cdots, I_s$. Put $P^* = I_1^*, I_2^*, \cdots, I_s^*$ where

if $I_k$ is not a jump instruction, then $I_k^* = I_k$;

if $I_k$ is not a jump instruction, then $I_k^* = \begin{cases} I_k & \text{if } q \leq s + 1, \\ J(m, n, s + 1) & \text{if } q > s + 1. \end{cases}$
Join/Concatenation

Let $P$ and $Q$ be programs of lengths $s$, $t$ respectively, in standard form. The *join* or *concatenation* of $P$ and $Q$, written $PQ$ or $^PQ$, is a program $I_1, I_2, \ldots, I_s, I_{s+1}, \ldots, I_{s+t}$ where $P = I_1, \ldots, I_s$ and the instructions $I_{s+1}, \ldots, I_{s+t}$ are the instructions of $Q$ with each jump $J(m, n, q)$ replaced by $J(m, n, s + q)$. 
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 $P$. 
The notation $P[l_1, \ldots, l_n \to l]$ stands for the following program:

\[
\begin{align*}
I_1 & : T(l_1, 1) \\
& \vdots \\
I_n & : T(l_n, n) \\
I_{n+1} & : Z(n + 1) \\
& \vdots \\
I_{\rho(P)} & : Z(\rho(P)) \\
_1 & : P \\
_2 & : T(1, l)
\end{align*}
\]