Unlimited Register Machine

Informal Description

An *unlimited register machine*, or *URM* for short, is a mathematical abstraction of a very primitive computer, which consists of a tape, which is left-ended, and stretches indefinitely to the right (hence the word ``unlimited''). The tape is divided into squares called registers, as illustrated by the diagram below

r_1	r_2	r_3	r_4	r_5	r_6	r_7	$\cdots$

Each register can store a single non-negative integer. The integer in the register is called the content of the register. In the diagram above, the content of the i-th register is r_i . The function of the tape is storage, such as inputs and outputs.

The core of a URM is its program, which is a finite sequence of instructions (to be described below). The program of a URM is responsible for carrying out computations. When a program is run, the contents of the registers may change on the tape. The execution of the program may or may not end.

The pre-defined instructions that make up the program of a URM come in four basic types:

1. zero operation $Z(n)$, which changes the content of register n to 0 ;
2. successor operation $S(n)$, which increases the content of register n by 1 ;
3. transfer operation $T(m,n)$, which writes (or transfers) the content of register m to that of register n ;
4. jump operation $J(m,n,p)$, which, when encountered in a program, ``jumps'' to the p -th instruction whenever the contents of registers m and n are the same.

Whereas the first three types of instructions alter the contents of the tape, the last type alters the flow of the program. Therefore, the three types are generally called *arithmetical instructions*, while the last is known as a *control instruction*.

Formal Description

Based on the information description above, we can write down precisely what a URM is.

First, the contents of the tape is just an infinite sequence of non-negative integers $r_1, r_2, \ldots$. Let $\mathfrak{R}$ be the set of tape contents (sequences just described).

Definition. A *configuration* is a pair (r, i) where $r \in \mathfrak{R}$, and $i \in \mathbb{N}$. The set of all configurations is denoted by $\mathfrak{C}$.

Next, we define the four types of instructions, the first three of which are arithmetical, and the last one is a control.

Definition. Let m, n, p be arbitrary positive integers. An *instruction* I is a function on $\mathfrak{C}$, and is any of the four following types:

1. $Z(n)(r, i) := (r', i+1)$, where

$$r'_k := \begin{cases} 0 & \text{if } k = n, \\ r_k & \text{otherwise.} \end{cases}$$

2. $S(n)(r, i) := (r', i+1)$, where

$$r'_k := \begin{cases} r_n + 1 & \text{if } k = n, \\ r_k & \text{otherwise.} \end{cases}$$

3. $T(m, n)(r, i) := (r', i+1)$, where

$$r'_k := \begin{cases} r_m & \text{if } k = n, \\ r_k & \text{otherwise.} \end{cases}$$

4. $J(m, n, p)(r, i) := (r, j)$, where

$$j := \begin{cases} p & \text{if } r_m = r_n, \\ i+1 & \text{otherwise.} \end{cases}$$

For each instruction I , a number $R(I)$ may be associated: $R(Z_n)=R(S_n):=n$, and $R(T(m,n))=R(J(m,n,p)):=\max(m,n)$.

Definition. Given an instruction I , a *computation step* of I is a pair of configurations (c_1,c_2) such that $c_2=I(c_1)$. c_1,c_2 are called the *input* and *output configurations* of I

A *computation sequence* is just a sequence of computation steps $c_1,c_2,\ldots$ such that for each $i=1,2,\ldots$, there is some instruction I (dependent on i) such that $c_{i+1}=I(c_i)$.

One often writes $c_1 \Longrightarrow c_2$ to denote a computation step, and $c_1 \Longrightarrow c_2 \Longrightarrow \ldots \Longrightarrow c_k \Longrightarrow \ldots$ to denote a computation sequence.

Definition. An *unlimited register machine* M is a finite sequence of instructions $I_1,I_2,\ldots,I_q$ (where each instruction is a function from one of the four types above). The *program* of M is also identified with the sequence of instructions.

In other words, a URM is just a list of instructions. What sets a URM apart from other computing machines is in the types of instructions used, as well as how computations are done based on the instructions.

Definition. Suppose M is a URM, and $r\in\mathcal{R}$. A *computation* of r by M is a computation sequence $c_1 \Longrightarrow c_2 \Longrightarrow \ldots \Longrightarrow c_k \Longrightarrow \ldots$ such that

1. $c_1=(r,1)$, and $c_2=I_1(c_1)$, and
2. if $c_k=(s,j)$, then $c_{k+1}=I_j(c_k)$.

The computation is deterministic in that the computation starts with a fixed initial configuration, and each computation step determines the next computation step. We denote $M(r)$ the computation of r by M .

Since $M(r)$ is a sequence of computation steps, it is either a finite sequence or an infinite sequence:

- If it is finite, we say that the computation *halts*, *terminates*, or that M *converges* on r, and we write $M(r)\downarrow$. This means that in the last computation step$(s,j) \Rightarrow (t,k)$, we have $k>q$, so that the next computation step is impossible, as the index of the instructions only goes up to q.

  If a is the content of register 1 in the ouptput, we also write $M(r)\downarrow a$, and we say that M *converges* (on r) to a

- Otherwise, we say that the computation *runs forever*, *non-terminating*, or that M *diverges* on r, and is denoted by $M(r)\uparrow$.

Given a URM M, one can define two numbers that are input independent:

Definition. $|M|$ is the number of instructions in the program of M, and

$$\rho(M) := \max\{R(I_k) \mid I_k \text{ is an instruction of } M\}.$$

In other words, $\rho(M)$ is the largest register that may be affected by any computation of M.

Remark. Unlimited register machines were introduced by Shepherdson and Sturgis in 1963. In their paper, the transfer instruction was not used (in fact, it is not hard to see that the transfer instruction is unnecessary in the current definition, as it can be ``simulated'' by a program consisting of the other instructions), and the jump instruction had a different form. But the basic setup is the same, having a tape of consisting of infinite squares or registers. It can be shown that URMs are equivalent to Turing machines in terms of their computing power. However, URMs are easier to work with, as there is an unlimited supply of integers rather than a finite set of symbols, and going from cell to the next can be done by a single instruction, rather than a series of moves.

Bibliography

1

J. C. Shepherdson, H. E. Sturgis, *Computability of Recursive Functions*. Journal Assoc. Comput. Mach. 10, 217-255, (1963).

2

N. Cutland, *Computability: An Introduction to Recursive Function Theory*. Cambridge University Press, (1980).