URM computable

Canonical name	URMComputable
Date of creation	2013-03-22 19:03:42
Last modified on	2013-03-22 19:03:42
Owner	CWoo (3771)
Last modified by	CWoo (3771)
Numerical id	14
Author	CWoo (3771)
Entry type	Definition
Classification	msc 68Q05
Classification	msc 03D10
Synonym	URM-computable

Let M be an unlimited register machine (URM), and r a finite sequence of non-negative integers. Recall the following notations:

- $M(r)$ denotes the computation of r by the program of M,
- $M(r)\downarrow$ denotes that the computation halts (M converges on r),
- $M(r)\downarrow a$ denotes $M(r)\downarrow$, and a is the content of register 1 in the output,
- $M(r)\uparrow$ denotes that the computation does not halt (M diverges r).

In the case where all but finitely many values of r are 0, say $r = r_1, r_2, \ldots, r_n, 0, 0, \ldots$, we also write $M(r_1, \ldots, r_n)$ to emphasize the fact that $r_i = 0$ for all $i > n$.

Definition. Let $f : \mathbb{N}^n \to \mathbb{N}$ be an n-ary partial function on natural numbers (including 0 in this discussion). f is said to be *URM-computable* if there is a URM M such that $M(r_1, \ldots, r_n)\downarrow f(r_1, \ldots, r_n)$ precisely when $(r_1, \ldots, r_n) \in \operatorname{dom}(f)$. When f is URM-computable by M, we also say that M *computes* f.

In other words, if $(r_1, \ldots, r_n)$ is in the domain of f, then we have a halting computation

$$\text{start}\ \begin{array}{|c|c|c|c|c|c}\hline r_1 & \cdots & r_n & 0 & 0 & \cdots \\ \hline\end{array}$$

$$\vdots$$

$$\text{halt}\ \begin{array}{|c|c|c|c}\hline f(r_1, \ldots, r_n) & \cdot & \cdot & \cdots \\ \hline\end{array}$$

If on the other hand $(r_1, \ldots, r_n)$ is not in the domain of f, then the computation of the above input never terminates.

For example, $f(r_1, r_2) = r_1 + r_2$, addition of two non-negative integers, is URM-computable, as is shown in `http://planetmath.org/ExamplesOfUnlimitedRegisterMachi` entry.

Here are two more basic examples:

- (subtraction by 1): $f(r_1) = r_1 - 1$. Note that f is a partial function that is not total, because $f(0)$ is not defined. A URM that computes f is the following:

$$M = J(1, 4, 1), S(2), J(1, 2, 6), S(3), J(1, 1, 2), T(3, 1)$$

First, M compares the r_1 with $r_4 := 0$. If they are the same, it loops indefinitely. Otherwise, M increments r_2 by 1, and then compares r_1 with r_2. If they are the same, then M transfers $r_3 := 0$ in register 3 to r_1 in register 1. Otherwise, it increments r_3 by 1 and loops back to the second instruction. The computation continues until $r_1 = r_2$, and when this happens, r_1 is set to be r_3.

- (monus operation): $f(r_1) = r_1 \dot{-} 1$. This is like the last example, except $f(0) := 0$. All we have to do is to modify the URM above:

  $$M = J(1,4,6), S(2), J(1,2,6), S(3), J(1,1,2), T(3,1)$$

  so the first instruction jumps to the last instruction when $r_1 = r_4$, instead of looping.

- (parity checking): $f(r_1) = 1$ if r_1 is odd, and $f(r_1) = 0$ otherwise. In other words, $f(r_1)$ is the remainder of the division of r_1 by 2. A URM that computes f is the following:

  $$\begin{aligned} M \;=\; & J(1,2,14), T(1,2), S(2), S(3), S(3), J(1,3,9), J(2,3,11), J(1,1,4), \\ & Z(1), J(1,1,14), Z(1), S(1), J(1,1,14) \end{aligned}$$

  Basically, with input $r_1 := m$, M first sets $r_2 := m + 1$. Then by incrementing r_3 by 2, M tests whether $r_1 = r_3$ or $r_2 = r_3$. If the former, then M sets $r_1 := 0$, otherwise r_1 is set to 1. The computation stops when the program jumps to the non-existent instruction 14.

Remarks.

- For any URM M and any positive integer n, M computes a unique n-ary (partial) function f. This can be simply done as follows: take the contents r of the first n registers of the tape as input, and run M. Define a partial function $f : \mathbb{N}^n \to \mathbb{N}$ so that $r \in \mathrm{dom}(f)$ iff $M(r)\downarrow$, and when this is the case, set $f(r)$ to be the integer such that $M(r)\downarrow f(r)$.

  Examples.

  - $T(5,2)$ computes, for any $n > 0$, the n-ary function $f(x_1, \ldots, x_n) = x_1$.
  - $T(5,1)$ computes $f(x_1, \ldots, x_n) = 0$ for any $0 < n < 5$, and $g(x_1, \ldots, x_n) = x_5$ for any $n \geq 5$.

- $J(1,1,1)$ computes the empty function $\varnothing$ for all $n \geq 0$.

- More generally, a partial function $f : \mathbb{N}^n \to \mathbb{N}^m$ is said to be URM-computable iff there is a URM M such that $M(r_1, \ldots, r_n)\downarrow$, and the i-th coordinate of $f(r_1, \ldots, r_n)$ is the content of the i-th register, $i \in \{1, \ldots, m\}$, precisely when $(r_1, \ldots, r_n) \in \mathrm{dom}(f)$.

  The function f above can be expressed as $(g_1, \ldots, g_m)$, where each $g_i : \mathbb{N}^n \to \mathbb{N}$. Then it is not hard to show that f is URM-computable iff each g_i is URM-computable.

- One of the fundamental facts about URM computability is the following: a function is URM computable iff it is Turing computable. By Church's thesis, this means that URM computability is equivalent to effective computability.

References

[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).