examples of unlimited register machines

Canonical name	ExamplesOfUnlimitedRegisterMachines
Date of creation	2013-03-22 19:03:13
Last modified on	2013-03-22 19:03:13
Owner	CWoo (3771)
Last modified by	CWoo (3771)
Numerical id	8
Author	CWoo (3771)
Entry type	Example
Classification	msc 68Q05
Classification	msc 03D10

In this entry, we illustrate the basic computing power of unlimited register machines by giving some examples.

Example (Addition). Here, we show how the addition of two non-negative integers can be achieved by a URM. Let M be the URM with the instructions:

$$I_1, I_2, I_3, I_4, I_5 = J(2,3,5), S(1), S(3), J(1,1,1), Z(3)$$

Let the input content be a, b in the first two registers, and 0 everywhere else. Then the ouput content has $a+b, b$ in the first two registers, and 0 everywhere else. This is how the computation works:

1. compare the contents of registers 2 and 3,
2. if they are different, increase the content of register 1 by 1,
3. increase the content of register 3 by 1,
4. jumps back to instruction 1 (loops here) and continue the computation until the contents of registers 2 and 3 are the same, then jump to instruction 5,
5. erase the content of register 2,
6. erase the content of register 3.
7. the computation halts, because instruction 6 does not exist.

Below is an actual computation carried out where $a = 3$ and $b = 2$: This is how a computation works with input

	3	2	0	0	0	0	0	$\cdots$	input
c_1	3	2	0	0	0	0	0	$\cdots$	I_1
c_2	4	2	0	0	0	0	0	$\cdots$	I_2
c_3	4	2	1	0	0	0	0	$\cdots$	I_3
c_4	4	2	1	0	0	0	0	$\cdots$	I_4
c_5	4	2	1	0	0	0	0	$\cdots$	I_1
c_6	5	2	1	0	0	0	0	$\cdots$	I_2
c_7	5	2	2	0	0	0	0	$\cdots$	I_3

c_8	5	2	2	0	0	0	0	$\cdots$	I_4
c_9	5	2	2	0	0	0	0	$\cdots$	I_1
c_{10}	5	2	0	0	0	0	0	$\cdots$	I_5

Note that the last instruction I_5 above may be removed without affecting the outcome (in register 1).

Example. Let M be the URM with a single instruction $I_1 = J(n, n, 2)$. This machine, when run, halts immediately after the first computation step. If I_1 were $J(n, n, 1)$ instead, then the machine loops forever when run, because it keeps jumping back to I_1. In both cases, the tape contents do not change. Nevertheless, we shall see that such instruction $J(n, n, p)$ is very useful in the next example.

Example (Transfer Instruction). Here, we show how the transfer instruction $T(m, n)$ may be simulated by other instructions. Let M be the URM with the following instructions:

$$I_1, I_2, I_3, I_4, I_5 = J(m, n, 6), Z(n), S(n), J(m, n, 6), J(m, m, 3)$$

When a computation is started with any input,

1. M first compares the contents of registers m and n, if they are the same, it jumps to the 6th instruction, which does not exist, so the computation halts.

2. Otherwise, it goes to the next step, which reduces the content of register n to 0,

3. Then, step by step, M increases the content of register n by 1.

4. During each increment, it compares the contents of registers m and n. If they are not the same, the loops back to instruction 3, and increases the content of n by 1.

5. However, if they are the same, then it jumps to instruction 6, so that the computation halts.

Below is a computation of input where the contents of registers m and n are 9 and 7 respectively (assume $m < n$)

r_1	$\cdots$	9	$\cdots$	7	$\cdots$	input

c_1	r_1	$\cdots$	9	$\cdots$	7	$\cdots$		I_1
c_2	r_1	$\cdots$	0	$\cdots$	7	$\cdots$		I_2
c_3	r_1	$\cdots$	1	$\cdots$	7	$\cdots$		I_3
c_4	r_1	$\cdots$	1	$\cdots$	7	$\cdots$		I_4
c_5	r_1	$\cdots$	1	$\cdots$	7	$\cdots$		I_5
c_6	r_1	$\cdots$	2	$\cdots$	7	$\cdots$		I_3

looping instructions 3,4,5 until content of register m reads 9

c_{21}	r_1	$\cdots$	7	$\cdots$	7	$\cdots$		I_3
c_{22}	r_1	$\cdots$	7	$\cdots$	7	$\cdots$		I_4

The result is precisely the same as running a URM with a single instruction $T(m, n)$. However, without $T(m, n)$, it takes 28 steps to achieve the same goal.

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).