

1 A One Instruction Language

A loop program uses only a handful of primitives: setting a variable to 0, incrementing a variable, composition and looping. One might wonder if one can get away with just a single type of instruction. In other words, is there are a programming language all of whose programs look like

$$I_1, I_2, \dots, I_n$$

where each instruction I_i is of the same type.

Surprisingly, the answer is yes, even when the instruction is required to be rather simple. Here is a classical example for such a language. The language OIL (one instruction language) has instructions of the form

$$k, x, y, l;$$

where k and l are positive integers, used as labels, and x and y are variables. Thus a program looks like

$$\begin{aligned} &k_1, x_1, y_1, l_1; \\ &k_2, x_2, y_2, l_2; \\ &\vdots \\ &k_n, x_n, y_n, l_n; \end{aligned}$$

We insist that $k_i = i$ and $1 \leq l_i \leq n + 1$, so we can run through the instructions using a program counter as usual and use the l -labels as targets for goto statements.

The semantics of OIL are defined as follows. First, the registers hold integers (not just non-negative integers as in LOOP). Second, instruction $k, x, y, l;$ corresponds to

```
x -= y;
if( x == 0 )
    goto l;
else
    goto k+1;
```

Any attempt to jump to $n + 1$ will cause the program to halt. Thus the execution of an instruction k, x, y, l comes down to this:

- Subtract the content of register y from register x , and store the result there.
- If x is now 0, goto instruction l , otherwise increment the program counter.
- Halt if the program counter is $n + 1$, otherwise continue execution.

To compute a function $f : \mathbb{N}^m \rightarrow \mathbb{N}$ fix some variables x_1, x_2, \dots, x_m as input variables and declare another variable y as output variable. Now for a little twist: let z be another variable. An OIL program P computes f (via x_1, x_2, \dots, x_m and y) if, for all input arguments $a_1, \dots, a_k \in \mathbb{N}$ the following holds: if the variables x_i are initialized to a_i and z is initialized to 1, then, upon execution of the program, the value of y is $f(a_1, \dots, a_k)$. Note that we may safely assume that all variables other than x_i and z in the program are initialized to 0: otherwise we can simply prepend instructions of the form $(1, u, u, 2)$.

Example 1.1 Here is an OIL program that adds the contents of variables x_1 and x_2 and writes the sum to register y . Of course, the comments below are not part of the actual program.

```

1, u, x1, 2;    //u = -x1
2, u, x2, 3;    //u = -x1 - x2
3, y, u, 4;     //y = x1 + x2

```

Example 1.2 Here is an OIL program that adds 1 to the contents of variable x , using u as an auxiliary variable. Recall that z is initialized to 1.

```

1, u, z, 2;     //u = -1
2, u, x, 3;     //u = -x - 1
3, x, x, 4;     //x = 0
4, x, u, 5;     //x = x + 1

```

Note that in these examples the goto-mechanism is not really needed: the program counter is always incremented by 1. Needless to say, in general goto's are indispensable.

2 Some Questions

Exercise 2.1 Show how to compute standard arithmetic functions such as times, exponentiation, mod and div by an OIL program.

Exercise 2.2 Write an interpreter for OIL programs and check that your programs from the previous problem really work.

Exercise 2.3 Invent a better notation for OIL programs. E.g., the programs in the examples should not require any labels whatsoever. You might add instructions such as $x = y$;

Exercise 2.4 Show that the functions computable by an OIL program are closed under composition.

Exercise 2.5 Show that any LOOP program can be simulated by an OIL program.

Exercise 2.6 Show that any computable function can be computed by an OIL program.

Exercise 2.7 What would happen if the special variable z were initialized to 0 rather than 1? What functions are still OIL-computable in this case?

Exercise 2.8 OIL programs use integers to compute functions over the natural numbers. Could one somehow use variables that range over \mathbb{N} rather than \mathbb{Z} ? How much more complicated would the single instruction have to be?

The last problem is a bit open-ended, there are probably several reasonable answers.