Compilation of Terms to Programs for a Stack Machine

Stack Machine

Instructions:

push(n): the number n goes to stack s

load(n): the element $(s)_n$ is pushed on the stack

incr$_i$: $(s)_0 := (s)_0 + 1$

decr$_i$: $(s)_0 := (s)_0 \dot{-} 1$

pair: $(s)_1 := (s)_1, (s)_0$ and pop.

head$_i$: $(s)_0 := H(s)_0$

tail$_i$: $(s)_0 := T(s)_0$

call$(q); p$: stack is r, v, s_1; new stack is p, r, v, s_1 and r is executed; **Note** p is **return address**. q should end with **ret**(2)

if$_i(q_1, q_2); p$: stack is v, s_1; new stack p, s_1. q_1 or q_2 is executed according to whether $v > 0$ or not. Both programs should end with **ret**(0).

ret(n): stack is $(w, q, s_1) \oplus s_2$ where $L(s_1) = n$; new stack is w, s_2, saved return address q is executed.

Interpreter for the Stack Machine

$Run(p, s) = v$ is a (partial) function interpreting the program p - which is a list of instructions - for the stack machine on the stack s. At then end: $Run(0, v, s) = v$ it yield the value v at the **top** of the stack.

We wish to write a **compiler** $Comp(i, b) = p$ taking a **functional term** b and yielding a program p **evaluating** b.
Suppose that $f(v) = a$, $Comp(0, a) = q$ and b is a part of a then the situation during the execution will be $Run(p, s \oplus (c, q, v, t)$
where i is the **offset** on the stack: $L(s) = i$ containing already computed intermediate results of the body a.

Compilation versus Interpretation

For a function $f(v) = a$ we wish that the **compiled** program $q = Comp(0, a)$ satisfies the following:

$$Run(q, (c, q, v, 0)) = [r]^v_a$$

Note the **initial offset** 0 when the execution of q starts.

The general situation for a subterm b of a is:

$$Run(Comp(L(s), b) \oplus p, s \oplus (c, q, v, t)) =$$
$$Run(p, ([b]^v_r, s) \oplus (c, q, v, t))$$

Finite Sets

Coding of Finite Sets

There are many ways of coding of **finite sets** of natural numbers as natural numbers.

Probably the simplest one is as **powersets**: The **empty** set $\emptyset$ is coded by 0 and for $n > 0$ we code by **bits**:

$$\{s_1, \ldots, s_n\} \text{ as } \sum_{i=1}^{n} 2^{s_i}$$

Note that s_i can again code a set.

However, the numbers coding relatively small **sparse sets**, say $\{2, 10^6, 3 \cdot 10^9\}$, are very **large**.

Coding of finite sets by ordered lists

The problem of sparse sets is solved by coding the sets as **lists** x without **repetition**:

$$\forall y \forall z \forall v \forall a \forall b (x = y \oplus (a, z \oplus (b, v)) \rightarrow a \neq b)$$

For instance, **increasing lists** x are without repetition:

$$Set(x) \leftrightarrow \forall y \forall z \forall a \forall b (x = y \oplus (a, b, z) \rightarrow a < b)$$

For **list membership** predicate $a \mathrel{\varepsilon} x$ such that

$$a \mathrel{\varepsilon} x \leftrightarrow \exists y \exists z \, x = y \oplus (a, z)$$

we have for sets x:

$$x = y \oplus (a, z) \land b \mathrel{\varepsilon} y \land c \mathrel{\varepsilon} z \rightarrow b < a < c$$

If $Set(x)$ and $a \mathrel{\not\varepsilon} x$ then the **insertion** satisfies

$$x \cup \{a\} = y \leftrightarrow \exists v \exists w (x = v \oplus w \land y = v \oplus (a, w) \land \\ \forall b \mathrel{\varepsilon} v (b < a) \land \forall b \mathrel{\varepsilon} w (a < b))$$