Script **generated by TTT**

Title:	Petter: Virtual Machines (28.05.2019)
Date:	Tue May 28 10:11:38 CEST 2019
Duration:	93:56 min
Pages:	7

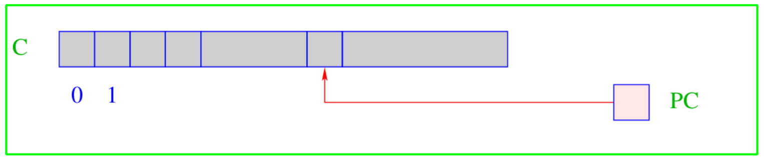
A program p is constructed as follows:

$$\begin{array}{rcl} \boxed{t} & ::= & \boxed{a} \mid \boxed{X} \mid \boxed{\_} \mid \boxed{f(t_1, \ldots, t_n)} \\ \boxed{g} & ::= & \boxed{p(t_1, \ldots, t_k)} \mid \boxed{X = t} \\ c & ::= & \boxed{p(\boxed{X_1}, \ldots, \boxed{X_k})} \leftarrow \boxed{g_1, \ldots, g_r} \\ p & ::= & \boxed{c_1 . \ldots . c_m} ? \boxed{g} \end{array}$$

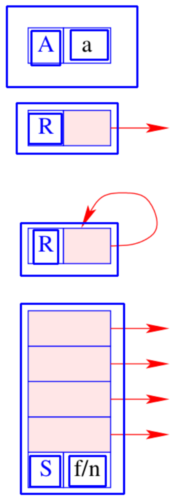
- A term t either is an atom, a variable, an anonymous variable or a constructor application.
- A goal g either is a literal, i.e., a predicate call, or a unification.
- A clause c consists of a head $p(X_1, \ldots, X_k)$ with predicate name and list of formal parameters together with a body, i.e., a sequence of goals.
- A program consists of a sequence of clauses together with a single goal as query.

28 Architecture of the WiM

The Code Store



C	=	Code store – contains WiM program; every cell contains one instruction;
PC	=	Program Counter – points to the next instruction to executed;



atom	1 cell
variable	1 cell
unbound variable	1 cell
structure	(n+1) cells

29 Construction of Terms in the Heap

Parameter terms of goals (calls) are constructed in the heap before passing.

Assume that the address environment ρ returns, for each clause variable X its address (relative to FP) on the stack. Then code$_A$ t ρ should ...

- construct (a presentation of) t in the heap; and
- return a reference to it on top of the stack.

Idea

- Construct the tree during a post-order traversal of t
- with one instruction for each new node!

Example $t \equiv f(g(X,Y),a,Z).$

Assume that X is initialized, i.e., S[FP + ρ X] contains already a reference, Y and Z are not yet initialized.

29 Construction of Terms in the Heap

Parameter terms of goals (calls) are constructed in the heap before passing.

Assume that the address environment ρ returns, for each clause variable X its address (relative to FP) on the stack. Then code$_A$ t ρ should ...

- construct (a presentation of) t in the heap; and
- return a reference to it on top of the stack.

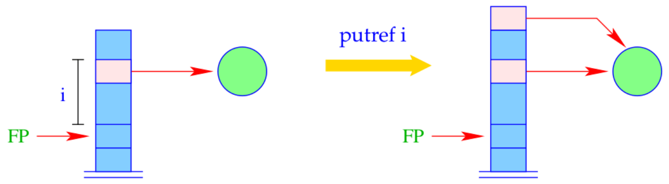
Idea

- Construct the tree during a post-order traversal of t
- with one instruction for each new node!

Example $t \equiv f(g(X,Y),a,Z).$

Assume that X is initialized, i.e., S[FP + ρ X] contains already a reference, Y and Z are not yet initialized.

The instruction putref i pushes a reference to the value of the variable onto the stack:

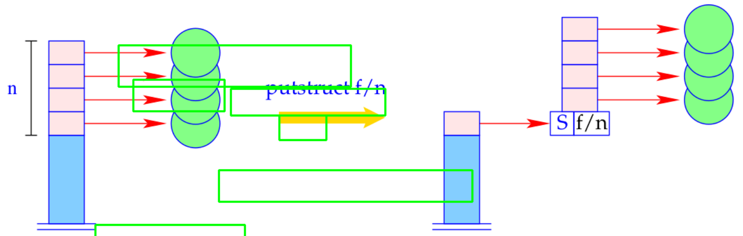


```
SP = SP + 1;
S[SP] = deref S[FP + i];
```

The auxiliary function deref contracts chains of references:

```
ref deref (ref v) {
     if (H[v]==(R,w) && v!=w) return deref (w);
     else return v;
}
```

The instruction putstruct f/n builds a constructor application in the heap:



```
v = new (S, f, n);
SP = SP - n + 1;
for (i=1; i¡=n; i++)
    H[v + i] = S[SP + i -1];
S[SP] = v;
```