Compiler Construction

Lecture 19: Code Generation IV (Translation Example & Correctness)

Thomas Noll

Lehrstuhl für Informatik 2
(Software Modeling and Verification)

RWTH Aachen University

noll@cs.rwth-aachen.de

http://www-i2.informatik.rwth-aachen.de/i2/cc08/

Summer semester 2008

Outline

The Abstract Machine AM

Definition (Abstract machine for EPL)

The abstract machine for EPL (AM) is defined by the state space

$$S := PC \times DS \times PS$$

with

- the program counter $PC := \mathbb{N}$,
- the data stack $DS := \mathbb{Z}^*$ (top of stack to the right), and
- the procedure stack (or: runtime stack) $PS := \mathbb{Z}^*$ (top of stack to the left).

Thus a state $s = (l, d, p) \in S$ is given by

- a program label $l \in PC$,
- a data stack $d = d.r : \ldots : d.1 \in DS$, and
- a procedure stack $p = p.1 : \ldots : p.t \in PS$.

Structure of Procedure Stack

The semantics of procedure and transfer instructions requires a particular structure of the procedure stack $p \in PS$: is must be composed of frames (or: activation records) of the form

$$sl : dl : ra : v_1 : \ldots : v_k$$

where

static link sl: points to frame of surrounding declaration environment
$\Longrightarrow$ used to access non-local variables

dynamic link dl: points to previous frame (i.e., of calling procedure)
$\Longrightarrow$ used to remove topmost frame after termination of procedure call

return address ra: program label after termination of procedure call
$\Longrightarrow$ used to continue program execution after termination of procedure call

local variables v_i: values of locally declared variables

The Initial Symbol Table

By Definition 17.9, an EPL program $P = \texttt{in/out}\ I_1, \ldots, I_n; K\ . \in Pgm$ has the semantics $\mathfrak{M}[\![P]\!] : \mathbb{Z}^n \dashrightarrow \mathbb{Z}^n$.

Given $(z_1, \ldots, z_n) \in \mathbb{Z}^n$, we choose the initial state

$$s := (1, \varepsilon, \underbrace{0 : 0 : 0 : z_1 : \ldots : z_n}_{\texttt{I/O}\text{ frame}}) \in S = PC \times DS \times PS$$

Thus the corresponding initial symbol table has n entries:

$$\mathrm{st}_{\texttt{I/O}}(I_j) := (\texttt{var}, 0, j) \qquad \text{for every } j \in [n]$$

Translation of EPL into AM Programs

Translation mapping trans : $Pgm \dashrightarrow AM$ defined using

- symbol table:

$$Tab := \{\mathrm{st} \mid \mathrm{st} : Ide \dashrightarrow (\{\texttt{const}\} \times \mathbb{Z}) \cup (\{\texttt{var}\} \times Lev \times Off) \cup (\{\texttt{proc}\} \times PC \times Lev \times Size)\}$$

- update : $Dcl \times Tab \times Lev \dashrightarrow Tab$
  for extending the symbol table by declarations
- kt : $Block \times Tab \times PC \times Lev \dashrightarrow AM$ for translating blocks
- dt : $Dcl \times Tab \times Lev \dashrightarrow AM$
  for translating the bodies of procedure declarations
- ct : $Cmd \times Tab \times PC \times Lev \dashrightarrow AM$
  for translating commands
- bt : $BExp \times Tab \times PC \times Lev \dashrightarrow AM$
  for translating Boolean expressions
- at : $AExp \times Tab \times PC \times Lev \dashrightarrow AM$
  for translating arithmetic expressions

Outline

Example: Factorial Function I

Example 19.1 (Factorial function)

Source code:

```
in/out x;
  var y;
  proc F;
    if x > 1 then
      y := y * x;
      x := x - 1;
      F()
  y := 1;
  F();
  x := y.
```

$\text{trans}(\texttt{in/out}\ I_1, \ldots, I_n; K.) :=$

$1 : \texttt{CALL}(a, \texttt{0}, \text{size}(K))$;
$\text{kt}(K, \text{st}_{\texttt{I/O}}, a, 1)$
$\text{dt}(D, \text{update}(D, \text{st}, l), l)$
$\text{ct}(C, \text{update}(D, \text{st}, l), a, l)$
$a' : \texttt{RET};$

$\text{kt}(K_1, \text{st}, a_1, l+1)$
$\vdots$

$\text{update}(\texttt{var}\ I_1, \ldots, I_n;, \text{st}, l) :=$
$\text{st}[I_1 \mapsto (\texttt{var}, l, 1), \ldots, I_n \mapsto (\texttt{var}, l, n)]$

$\text{update}(\texttt{proc}\ I_1; K_1; \ldots; I_n; K_n;, \text{st}, l) :=$
$\text{st}[I_1 \mapsto (\texttt{proc}, a_1, l, \text{size}(K_1)), \ldots, I_n \mapsto (\texttt{proc}, a_n, l, \text{size}(K$

$\text{ct}(\texttt{if}\ B\ \texttt{then}\ C_1\ \texttt{else}\ C_2, \text{st}, a, l) :=$
$\text{ct}(C_B, \text{st}, a, 1)$
$a_2 : \texttt{RET};$
$\text{ct}(C_1, \text{st}, a_1, 2)$
$a_3 : \texttt{RET};$

Intermediate code:

$\text{trans}(\texttt{in/out}\ x; K.)$
$1 : \texttt{CALL}(a_0, \texttt{0}, \texttt{1});$
$2 : \texttt{JMP}(\texttt{0});$
$\text{kt}(K, \text{st}_{\texttt{I/O}}, a_0, 1)$
$1 : \texttt{CALL}(a_0, \texttt{0}, \texttt{1});$
$2 : \texttt{JMP}(\texttt{0});$
$\text{dt}(D, \text{update}(D, \text{st}_{\texttt{I}}$
$\text{ct}(C, \text{update}(D, \text{st}_{\texttt{I}}$
$a_2 : \texttt{RET};$
$1 : \texttt{CALL}(a_0, \texttt{0}, \texttt{1});$
$2 : \texttt{JMP}(\texttt{0});$
$\text{dt}(D, \text{st}', 1)$
$\text{ct}(C, \text{st}', a_0, 1)$
$a_2 : \texttt{RET};$
$1 : \texttt{CALL}(a_0, \texttt{0}, \texttt{1});$
$2 : \texttt{JMP}(\texttt{0});$
$\text{kt}(K_{\texttt{F}}, \text{st}', a_1, 2)$
$\text{ct}(C, \text{st}', a_0, 1)$
$a_2 : \texttt{RET};$
$1 : \texttt{CALL}(a_0, \texttt{0}, \texttt{1});$
$2 : \texttt{JMP}(\texttt{0});$
$\text{ct}(C_{\texttt{F}}, \text{st}', a_1, 2)$
$a_3 : \texttt{RET};$

Example: Factorial Function II

Example 19.2 (Factorial function; continued)

Code with symbolic addresses:

```
 1 : CALL(a0,0,1);
 2 : JMP(0);
a1 : LOAD(2,1);
     LIT(1);
     GT;
a4 : JFALSE(a3);
     LOAD(1,1);
     LOAD(2,1);
     MULT;
     STORE(1,1);
     LOAD(2,1);
     LIT(1);
     SUB;
     STORE(2,1);
     CALL(a1,1,0);
a3 : RET;
a0 : LIT(1);
     STORE(0,1);
     CALL(a1,0,0);
     LOAD(0,1);
     STORE(1,1);
a2 : RET;
```

Linearized ($a_0 = 17, a_1 = 3, a_2 = 22, a_3 = 16, a_4 = 6$):

```
 1 : CALL(17,0,1);
 2 : JMP(0);
 3 : LOAD(2,1);
 4 : LIT(1);
 5 : GT;
 6 : JFALSE(16);
 7 : LOAD(1,1);
 8 : LOAD(2,1);
 9 : MULT;
10 : STORE(1,1);
11 : LOAD(2,1);
12 : LIT(1);
13 : SUB;
14 : STORE(2,1);
15 : CALL(3,1,0);
16 : RET;
17 : LIT(1);
18 : STORE(0,1);
19 : CALL(3,0,0);
20 : LOAD(0,1);
21 : STORE(1,1);
22 : RET;
```

Example: Factorial Function II

Example 19.3 (Factorial function; continued)

Computation for $\mathtt{x} = 2$:

```
 1 : CALL(17,0,1);
 2 : JMP(0);
 3 : LOAD(2,1);
 4 : LIT(1);
 5 : GT;
 6 : JFALSE(16);
 7 : LOAD(1,1);
 8 : LOAD(2,1);
 9 : MULT;
10 : STORE(1,1);
11 : LOAD(2,1);
12 : LIT(1);
13 : SUB;
14 : STORE(2,1);
15 : CALL(3,1,0);
16 : RET;
17 : LIT(1);
18 : STORE(0,1);
19 : CALL(3,0,0);
20 : LOAD(0,1);
21 : STORE(1,1);
22 : RET;
```

PC	*DS*	*PS*
1	ε	0 : 0 : 0 : 2
17	ε	4 : 3 : 2 : 0 : 0 : 0 : 0 : 2
18	1	4 : 3 : 2 : 0 : 0 : 0 : 0 : 2
19	ε	4 : 3 : 2 : 1 : 0 : 0 : 0 : 2
3	ε	3 : 2 : 20 : 4 : 3 : 2 : 1 : 0 : 0 : 0 : 2
4	2	3 : 2 : 20 : 4 : 3 : 2 : 1 : 0 : 0 : 0 : 2
5	2 : 1	3 : 2 : 20 : 4 : 3 : 2 : 1 : 0 : 0 : 0 : 2
6	1	3 : 2 : 20 : 4 : 3 : 2 : 1 : 0 : 0 : 0 : 2
7	ε	3 : 2 : 20 : 4 : 3 : 2 : 1 : 0 : 0 : 0 : 2
8	1	3 : 2 : 20 : 4 : 3 : 2 : 1 : 0 : 0 : 0 : 2
9	1 : 2	3 : 2 : 20 : 4 : 3 : 2 : 1 : 0 : 0 : 0 : 2
10	2	3 : 2 : 20 : 4 : 3 : 2 : 1 : 0 : 0 : 0 : 2
11	ε	3 : 2 : 20 : 4 : 3 : 2 : 2 : 0 : 0 : 0 : 2
12	2	3 : 2 : 20 : 4 : 3 : 2 : 2 : 0 : 0 : 0 : 2
13	2 : 1	3 : 2 : 20 : 4 : 3 : 2 : 2 : 0 : 0 : 0 : 2
14	1	3 : 2 : 20 : 4 : 3 : 2 : 2 : 0 : 0 : 0 : 2
15	ε	3 : 2 : 20 : 4 : 3 : 2 : 2 : 0 : 0 : 0 : 1
3	ε	6 : 2 : 16 : 3 : 2 : 20 : 4 : 3 : 2 : 2 : 0 : 0 : 0 : 1
4	1	6 : 2 : 16 : 3 : 2 : 20 : 4 : 3 : 2 : 2 : 0 : 0 : 0 : 1
5	1 : 1	6 : 2 : 16 : 3 : 2 : 20 : 4 : 3 : 2 : 2 : 0 : 0 : 0 : 1
6	0	6 : 2 : 16 : 3 : 2 : 20 : 4 : 3 : 2 : 2 : 0 : 0 : 0 : 1
16	ε	6 : 2 : 16 : 3 : 2 : 20 : 4 : 3 : 2 : 2 : 0 : 0 : 0 : 1
16	ε	3 : 2 : 20 : 4 : 3 : 2 : 2 : 0 : 0 : 0 : 1
20	ε	4 : 3 : 2 : 2 : 0 : 0 : 0 : 1
21	2	4 : 3 : 2 : 2 : 0 : 0 : 0 : 1
22	ε	4 : 3 : 2 : 2 : 0 : 0 : 0 : 2
2	ε	0 : 0 : 0 : 2
0	ε	0 : 0 : 0 : 2

Outline

Correctness of the Translation

Theorem 19.4 (Correctness of translation)

For every $P \in Pgm$, $n \in \mathbb{N}$, *and* $(z_1, \ldots, z_n), (z'_1, \ldots, z'_n) \in \mathbb{Z}^n$:

$$\begin{aligned} & \mathfrak{M}[\![P]\!](z_1, \ldots, z_n) = (z'_1, \ldots, z'_n) \\ \iff & [\![\text{trans}(P)]\!](1, \varepsilon, 0 : 0 : 0 : z_1 : \ldots : z_n) = (0, \varepsilon, 0 : 0 : 0 : z'_1 : \ldots : z'_n) \end{aligned}$$

Proof.

see M. Mohnen: *A Compiler Corectness Proof for the Static Link Technique by means of Evolving Algebras*, Fundamenta Informaticae 29(3), 1997, pp. 257–303 □

Correct Translation of Arithmetic Expressions

The simplest subcase of the correctness proof is the following.

Lemma 19.5 (Correctness of at)

Let $A \in AExp$, $\rho \in Env$, $\sigma \in Stt$, $z \in \mathbb{Z}$, $\mathrm{st} \in Tab$, $a \in PC$, $l \in Lev$, $d \in DS$, *and* $p \in PS$ *where, for every* $I \in Ide$ *in* A,

- $\rho(I) = z \in \mathbb{Z} \implies \mathrm{st}(I) = (\texttt{const}, z)$ *and*
- $\rho(I) = \alpha \in Loc, \sigma(\alpha) = z \implies \mathrm{st}(I) = (\texttt{var}, lev, off)$ *for some* $lev \in Lev$ *and* $off \in Off$ *such that* $p.(\mathrm{base}(p, l - lev) + off + 2) = z$.

Then, with $P := \mathrm{at}(A, \mathrm{st}, a, l)$,

$$\mathfrak{A}[\![A]\!]\ \rho\ \sigma = z \implies [\![P]\!](a, d, p) = (a + |P|, d : z, p).$$

Proof.

on the board □