09—The Language imPL

CS 4215: Programming Language Implementation

Martin Henz

March 16, 2012

Generated on Thursday 15 March, 2012, 22:56

Introduction

- simPL, rePL: an identifier refers to a value
- once computed, the value does not change
- pass-by-need exploits this fact
- referential transparency
- good for formal reasoning

Motivation

- many algorithms are more natural when presented using random-access memory
- assigment allows to change the value stored in the memory location associated with an identifier
- imPL0 allows assignment
- imPL1 adds assignment to record properties
- many interesting variants...

Syntax of imPL0

Assignment $$\frac{E}{x := E}$$

Sequence $$\frac{E_1 \quad E_2}{E_1 \ ; \ E_2}$$

Loop $$\frac{E_1 \quad E_2}{\texttt{while } E_1 \texttt{ do } E_2 \texttt{ end}}$$

Example

```
let x = 0 in
    x := 1;
    x := x + 2;
    x := x + 3;
    x
end
```

Another Example

```
fun x ->
   let i = 1
       f = 1 in
       while \ i > x do
          f := f * i;
          i := i + 1
       end;
       f
   end
end
```

Yet Another Example

```
let gcd = fun a b ->
             while \(a = b) do
                if a > b
                then a := a - b
                else b := b - a
                end
             end;
             a
          end
in
   (gcd 6 10)
end
```

Denotational Semantics: Then How?

```
let x = 0 in
    x := 1;
    x := x + 2;
    x := x + 3;
    x
end
```

Denotational Semantics: Idea

- identifiers refer to locations
- a store maps locations to values
- the store is passed to and returned from the semantic function

Semantic Domains

Domain name	Definition
EV	$\mathbf{Int} + \mathbf{Bool} + \mathbf{Fun} + \{\mathbf{error}\}$
SV	$\mathbf{Int} + \mathbf{Bool} + \mathbf{Fun}$
DV	**Loc**
Fun	$\mathbf{DV} * \cdots * \mathbf{DV} * \mathbf{Store} \rightsquigarrow \mathbf{(EV, Store)}$
Store	$\mathbf{Loc} \rightsquigarrow \mathbf{SV}$
Env	$\mathbf{Id} \rightsquigarrow \mathbf{DV}$

Example

Let us say we have a store with the value 1 at location l

$$\Sigma = \emptyset_{\mathbf{Store}}[l \leftarrow 1]$$

and an environment that carries location l at identifier x

$$\Delta = \emptyset_{\mathbf{Env}}[x \leftarrow l]$$

Then we can access the value of x in the store as follows:

$$\Sigma(\Delta(x)) = 1$$

The Main Semantic Function

$$\cdot \rightarrowtail \cdot : \mathbf{imPL0} \rightarrow \mathbf{EV}$$

$$\frac{\emptyset_{\mathbf{Store}} \mid \emptyset_{\mathbf{Env}} \Vdash E \rightarrowtail (v, \Sigma)}{E \rightarrowtail v}$$

$$\cdot \mid \cdot \Vdash \cdot \rightarrowtail \cdot : \mathbf{Store} * \mathbf{Env} * \mathbf{imPL0} \rightarrow \mathbf{EV} * \mathbf{Store}$$

Let Expressions

$$\frac{\Sigma'[l_1 \leftarrow v_1] \mid \Delta[x_1 \leftarrow l_1] \Vdash E \rightarrowtail (v, \Sigma'')}{\Sigma \mid \Delta \Vdash \texttt{let}\ x_1 = E_1\ \texttt{in}\ E\ \texttt{end} \rightarrowtail (v, \Sigma'')}$$

if $\Sigma \mid \Delta \Vdash E_1 \rightarrowtail (v_1, \Sigma')$, and
l_1 is a new location,
which means $\Sigma'(l_1)$ is undefined

Identifiers

$$\frac{}{\Sigma \mid \Delta \Vdash x \rightarrowtail (\Sigma(\Delta(x)), \Sigma)}$$

Assignment

$$\frac{\Sigma \mid \Delta \Vdash E \rightarrowtail (v, \Sigma')}{\Sigma \mid \Delta \Vdash x := E \rightarrowtail (v, \Sigma'[\Delta(x) \leftarrow v])}$$

Example

$\emptyset_{\mathbf{Store}}[l \leftarrow 1] \mid \emptyset_{\mathbf{Env}}[\mathtt{a} \leftarrow l] \Vdash$
$\mathtt{a := 2} \rightarrowtail (2, \emptyset_{\mathbf{Store}}[l \leftarrow 1][l \leftarrow 2])$
The resulting store $\emptyset_{\mathbf{Store}}[l \leftarrow 1][l \leftarrow 2])$ is of course the same as $\emptyset_{\mathbf{Store}}[l \leftarrow 2]$.
The original binding of l to 1 is overwritten by the new value 2.

Function Definition

$$\frac{}{\Sigma \mid \Delta \Vdash \texttt{fun}\ x\ \texttt{->}\ E\ \texttt{end} \rightarrowtail (f, \Sigma)}$$

where $f(l, \Sigma') = (v', \Sigma'')$,
where $\Sigma' \mid \Delta[x \leftarrow l] \Vdash E \rightarrowtail (v', \Sigma'')$

Function Application

$$\frac{\Sigma \mid \Delta \Vdash E_1 \rightarrowtail (f, \Sigma') \qquad \Sigma' \mid \Delta \Vdash E_2 \rightarrowtail (v_2, \Sigma'')}{\Sigma \mid \Delta \Vdash (E_1\ E_2) \rightarrowtail f(l, \Sigma''[l \leftarrow v_2])}$$

where l is a new location in Σ''.

Sequence

$$\frac{\Sigma \mid \Delta \Vdash E_1 \rightarrowtail (v_1, \Sigma') \qquad \Sigma' \mid \Delta \Vdash E_2 \rightarrowtail (v_2, \Sigma'')}{\Sigma \mid \Delta \Vdash E_1 ; E_2 \rightarrowtail (v_2, \Sigma'')}$$

While Loop

$$\frac{\Sigma \mid \Delta \Vdash E_1 \rightarrowtail (v_1, \Sigma')}{\Sigma \mid \Delta \Vdash \texttt{while}\ E_1\ \texttt{do}\ E_2\ \texttt{end} \rightarrowtail (\mathit{true}, \Sigma')}\ \text{if } v_1 = \mathit{false}$$

While Loop

$$\frac{\Sigma \mid \Delta \Vdash E_1 \rightarrowtail (v_1, \Sigma')}{\Sigma \mid \Delta \Vdash \texttt{while}\ E_1\ \texttt{do}\ E_2\ \texttt{end} \rightarrowtail (v, \Sigma''')}$$

if $v_1 = \mathit{true}$,
where $\Sigma' \mid \Delta \Vdash E_2 \rightarrowtail (v_2, \Sigma'')$ and
$\Sigma'' \mid \Delta \Vdash \texttt{while}\ E_1\ \texttt{do}\ E_2\ \texttt{end} \rightarrowtail (v, \Sigma''')$

Record Property Assignment

$$\frac{E_1 \qquad E_2}{E_1.q := E_2}$$

Semantic Domains

Domain name	Definition
EV	$\mathbf{Int} + \mathbf{Bool} + \mathbf{Fun} + \mathbf{Rec} + \{\bot\}$
SV	$\mathbf{Int} + \mathbf{Bool} + \mathbf{Fun} + \mathbf{Rec}$
DV	**Loc**
Rec	$\mathbf{Id} \rightsquigarrow \mathbf{Loc}$

Record Construction

$$\frac{\Sigma^{(0)} \mid \Delta \Vdash E_1 \rightarrowtail (v_1, \Sigma^{(1)}) \quad \Sigma^{(n-1)} \mid \Delta \Vdash E_n \rightarrowtail (v_n, \Sigma^{(n)})}{\Sigma^{(0)} \mid \Delta \Vdash [q_1 : E_1, \ldots, q_n : E_n] \rightarrowtail (f, \Sigma')}$$

where $l_1, \ldots, l_n$ are new locations in $\Sigma^{(n)}$,
$\Sigma' = \Sigma^{(n)}[l_1 \leftarrow v_1] \cdots [l_n \leftarrow v_n]$,
and $f(q_i) = l_i$ for $1 \leq i \leq n$.

Property Access

$$\frac{\Sigma \mid \Delta \Vdash E \rightarrowtail (f, \Sigma')}{\Sigma \mid \Delta \Vdash E.q \rightarrowtail (\Sigma'(f(q)), \Sigma')}$$

Property Assignment

$$\frac{\Sigma \mid \Delta \Vdash E_1 \rightarrowtail (f, \Sigma') \qquad \Sigma' \mid \Delta \Vdash E_2 \rightarrowtail (v, \Sigma'')}{\Sigma \mid \Delta \Vdash E_1.q := E_2 \rightarrowtail (v, \Sigma''[f(q) \leftarrow v])}$$

Java: Passing Objects to Methods

```
void f(MyClass myobject) {
   myobject.myfield = 1;
   myobject = new MyClass();
   myobject.myfield = 2;
}
```

imPL: Passing Records to Functions

```
let a = [Myfield: 0]
in
   (fun b ->
       b.Myfield := 1;
       b := [Myfield: 2];
       b.Myfield := 3
    end
    a);
   a.Myfield
end
```

Pass-by-Reference

$$\frac{\Sigma \mid \Delta \Vdash E_1 \rightarrowtail (f, \Sigma')}{\Sigma \mid \Delta \Vdash (E_1\ x) \rightarrowtail f(\Delta(x), \Sigma')}$$

Pass-by-Copy

$$\frac{\Sigma \mid \Delta \Vdash E_1 \rightarrowtail (f, \Sigma') \qquad \Sigma' \mid \Delta \Vdash E_2 \rightarrowtail (v_2, \Sigma'')}{\Sigma \mid \Delta \Vdash (E_1\ E_2) \rightarrowtail f(l, \Sigma''')}$$

if $v_2 \in \mathbf{Rec}$,
where $l, l'_1, \ldots, l'_n$ are new locations in Σ'',
$\{q_1, \ldots, q_n\} = dom(v_2)$,
$\Sigma''' = \Sigma''[l \leftarrow \{(q_1, l'_1), \ldots, (q_n, l'_n)\}]$
$[l'_1 \leftarrow \Sigma''(v_2(q_1))] \ldots [l'_n \leftarrow \Sigma''(v_2(q_n))]$

Division by Zero

$$\frac{\Sigma \mid \Delta \Vdash E_1 \rightarrowtail (v_1, \Sigma') \qquad \Sigma' \mid \Delta \Vdash E_2 \rightarrowtail (0, \Sigma'')}{\Sigma \mid \Delta \Vdash E_1 / E_2 \rightarrowtail (e, \Sigma'')}$$

if $v_1 \notin \mathbf{Exc}$
and where $e =$ `[DivisionByZero:true]`,
and $e \in \mathbf{Exc}$

Execution of try-expressions

$$\frac{\Sigma \mid \Delta \Vdash E_1 \rightarrowtail (v, \Sigma')}{\Sigma \mid \Delta \Vdash \texttt{try}\ E_1\ \texttt{catch}\ x\ \texttt{with}\ E_2\ \texttt{end} \rightarrowtail (v, \Sigma')} \text{ if } v \notin \textbf{Exc}$$

$$\frac{\Sigma \mid \Delta \Vdash E_1 \rightarrowtail (v_1, \Sigma') \qquad \Sigma'[l \leftarrow v_1] \mid \Delta[x \leftarrow l] \Vdash E_2 \rightarrowtail (v_2, \Sigma'')}{\Sigma \mid \Delta \Vdash \texttt{try}\ E_1\ \texttt{catch}\ x\ \texttt{with}\ E_2\ \texttt{end} \rightarrowtail (v_2, \Sigma'')}$$

if $v_1 \in \textbf{Exc}$ and l new location

Rolling back the State

$$\frac{\Sigma \mid \Delta \Vdash E_1 \rightarrowtail (v_1, \Sigma') \qquad \Sigma[l \leftarrow v_1] \mid \Delta[x \leftarrow l] \Vdash E_2 \rightarrowtail (v_2, \Sigma'')}{\Sigma \mid \Delta \Vdash \texttt{try}\ E_1\ \texttt{catch}\ x\ \texttt{with}\ E_2\ \texttt{end} \rightarrowtail (v_2, \Sigma'')}$$

if $v_1 \in \mathbf{Exc}$ and l new location

Idea

Reuse heap for implementing imperative constructs

Translation of Assignment

$$\frac{E \hookrightarrow s}{x := E \hookrightarrow s.\texttt{ASSIGNS}\ x}$$

Execution of Assignment

$$\frac{s(pc) = \texttt{ASSIGNS}\ x}{(os, pc, e, rs, h) \rightrightarrows_s (os, pc + 1, e, rs, update(e, x, v, h'))}$$

where $(v, h') = pop(os, h)$

Translation of Sequences

$$\frac{E_1 \hookrightarrow s_1 \qquad E_2 \hookrightarrow s_2}{E_1 \,;\, E_2 \hookrightarrow s_1.\texttt{POP}.s_2}$$

Implementation of Sequences

$$\frac{s(pc) = \texttt{POP}}{(os, pc, e, rs, h) \rightrightarrows_s (os, pc + 1, e, rs, h')} \text{where } (v, h') = pop(os, h)$$

Translation of Loops

$$\frac{E_1 \hookrightarrow s_1 \qquad E_2 \hookrightarrow s_2}{\texttt{while } E_1 \texttt{ do } E_2 \hookrightarrow s_1.(\texttt{JOFR } |s_2 + 3|).s_2.\texttt{POP}.(\texttt{GOTOR } -(|s_1| + 2 + |s_2|)).\mathit{LDCB}\ \texttt{true}}$$

Overview of Next Two Lectures

- Object-oriented programming languages
  - defining an object system
  - runtime support
  - typing
- Concurrent languages