Function Definition

Gabriel Dos Reis

Parasol Lab
Department of Computer Science and Engineering
Texas A&M University
http://parasol.tamu.edu/~gdr/

February 18, 2011

Parse Tree Revised

$$
\begin{array}{llcl}
\textit{Toplevel} & \mathtt{d} & ::= & \mathtt{f} \mid \mathtt{u} \\
\\
\textit{FunDef} & \mathtt{f} & ::= & \mathit{FunDef}\left(\mathit{Identifier}(i), \mathtt{t}, \overrightarrow{\mathtt{p}}, u\right) \\
\textit{Parm} & \mathtt{p} & ::= & \mathit{Parameter}(\mathit{Identifier}(i), \mathtt{t}) \\
\textit{Stmt} & \mathtt{u} & ::= & \mathit{ExprStmt}(\mathtt{x}) \\
 & & \mid & \mathit{VarDef}(\mathit{Identifier}(i), \mathtt{t}, \mathtt{x}) \\
 & & \mid & \mathit{IfStmt}(\mathtt{x}, \mathtt{u}_1, \mathtt{u}_2) \\
 & & \mid & \mathit{RetStmt}(\mathtt{x}) \\
 & & \mid & \mathit{CmpdStmt}(\vec{\mathtt{u}}) \\
\\
\textit{RawExpr} & \mathtt{x}, \mathtt{y}, \mathtt{t} & ::= & \mathit{BooleanLiteral}(\mathtt{b}) \\
 & & \mid & \mathit{IntegerLiteral}(\mathtt{n}) \\
 & & \mid & \mathit{StringLiteral}(\mathtt{s}) \\
 & & \mid & \mathit{Identifier}(i) \\
 & & \mid & \mathit{Modify}(\mathtt{x}, \mathtt{y}) \\
 & & \mid & \mathit{UnaryExpr}(\mathtt{o}, \mathtt{x}) \\
 & & \mid & \mathit{BinaryExpr}(\mathtt{o}, \mathtt{x}, \mathtt{y}) \\
 & & \mid & \mathit{CallExpr}(\mathtt{x}, \vec{\mathtt{y}})
\end{array}
$$

Expression Trees

$$
\begin{array}{lrcl}
\textit{Expression} & e & ::= & \textit{Boolean}\,(b) \\
 & & \mid & \textit{Integer}\,(n) \\
 & & \mid & \textit{String}\,(s) \\
 & & \mid & \textit{Symbol}\,(i) \\
 & & \mid & \textit{Unary}\,(o, e) \\
 & & \mid & \textit{Binary}\,(o, e_1, e_2) \\
 & & \mid & \textit{Call}\,(e, \vec{e}) \\
 & & \mid & \textit{Return}\,(e) \\
 & & \mid & \textit{If}\,(e_1, e_2, e_3) \\
 & & \mid & \textit{Block}\,(\vec{e}) \\
 & & \mid & \textit{Bind}\,(i, \tau, e) \\
 & & \mid & \textit{Store}\,(e_1, e_2)
\end{array}
$$

$$
\textit{Elaboration} \quad \epsilon \quad ::= \quad \langle e, \tau \rangle \mid \langle \tau, \natural \rangle
$$

$$
\begin{array}{lrcl}
\textit{Type} & \tau & ::= & \textit{bool} \\
 & & \mid & \textit{int} \\
 & & \mid & \textit{string} \\
 & & \mid & (\vec{\tau}) \quad \rightarrow \quad \tau
\end{array}
$$

Design Decisions

General source-level principles:

- Uniform rules for variable and function definitions
- Function definitions may be recursive
  - so can variable definitions
- No syntactic distinction beetwen recursive and non-recursion definition
  - e.g. no `rec let` vs. `let`

Semantics and implementation challenges:

- How to give sound semantics?
- How to compile general recursion?
  - See: *Compilation of extended recursion in call-by-value functional languages*, by T. Hirschowitz, X. Leroy, J.B. Wells

A little issue with parameters

- An elaboration is a pair:

  $$\mathit{Elaboration}\ \epsilon\ ::=\ \langle e, \tau \rangle \mid \langle \tau, \natural \rangle$$

  - The first component is an expression to be evaluated at runtime
- A Typing context maps identifiers to elaborations
  - this works well for (variable) definitions
- What about parameters?
  - they are not "defined" with initializers in the source code

- Various solutions were proposed:
  - see fixpoint denotational semantics techniques with $\perp$, thunks, reify runtime environment, etc.

Parameter elaboration

Problem:

- Generate expression for a parameter declaration without too much cleverness, nor large extension to the existing expression intermediate language

Solution:

- Generate a symbolic reference to the parameter name:
  - symbol name looked up in the runtime environement ehen the parameter is referenced.
  - avoids reification of runtime environment
  - uses an existing general mechanism
  - if done right, may also support overloading (when added)

Expression Trees Revised

$$
\begin{array}{rcl}
\textit{Expression}\ e & ::= & \textit{Boolean}\,(b) \\
& | & \textit{Integer}\,(n) \\
& | & \textit{String}\,(s) \\
& | & \textit{Symbol}\,(i,\tau) \\
& | & \textit{Unary}\,(o,e) \\
& | & \textit{Binary}\,(o,e_1,e_2) \\
& | & \textit{Call}\,(e,\vec{e}) \\
& | & \textit{Return}\,(e) \\
& | & \textit{If}\,(e_1,e_2,e_3) \\
& | & \textit{Block}\,(\vec{e}) \\
& | & \textit{Bind}\,(i,\tau,e) \\
& | & \textit{Store}\,(e_1,e_2) \\
& | & \textit{Lambda}\left(\vec{i_k};e\right)
\end{array}
$$

$$
\textit{Elaboration}\ \epsilon\ ::=\ \langle e,\tau\rangle\ |\ \langle \tau,\natural\rangle
$$

$$
\begin{array}{rcl}
\textit{Type}\ \tau & ::= & \textit{void} \\
& | & \textit{bool} \\
& | & \textit{int} \\
& | & \textit{string} \\
& | & (\vec{\tau}) \quad \rightarrow \quad \tau
\end{array}
$$

Parameter elaboration

$$\frac{\texttt{t} \vdash_{\Gamma} \langle \tau, \natural \rangle}{\mathit{Parameter}\,(\mathit{Identifier}\,(i)\,,\texttt{t})\vdash_{\langle i \mapsto \epsilon \rangle \Gamma} \epsilon \quad \text{where} \quad \epsilon = \langle \mathit{Symbol}\,(i, \tau)\,, \tau \rangle}$$

Function elaboration

$$\frac{\mathtt{t} \vdash_\Gamma \langle \tau, \natural \rangle \quad \mathtt{p}_k \vdash_{\Gamma_k} \langle \tau_k, \natural \rangle \; k = 1..n \quad u \vdash^{\tau}_{\Gamma_{n+1}} e}{\mathit{FunDef}\left(\mathit{Identifier}\left(i\right), \mathtt{t}, \vec{\mathtt{p}}, u\right) \vdash_{\langle i \mapsto \epsilon \rangle \Gamma} \epsilon}$$

where

$$\begin{aligned}
\epsilon &= \left\langle \mathit{Bind}\left(i, \tau', \mathit{Lambda}\left(\vec{i_k}; e\right)\right), \tau' \right\rangle \\
\tau' &= \left(\vec{\tau_k}\right) \longrightarrow \tau \\
\Gamma_1 &= \langle i_1 \mapsto \left\langle \mathit{Symbol}\left(i_1, \tau_1\right), \tau_1 \right\rangle \rangle \Gamma \\
\Gamma_2 &= \langle i_2 \mapsto \left\langle \mathit{Symbol}\left(i_2, \tau_2\right), \tau_2 \right\rangle \rangle \Gamma_1 \\
\vdots &= \vdots \\
\Gamma_{n+1} &= \langle i \mapsto \left\langle \mathit{Symbol}\left(i, \tau'\right), \tau' \right\rangle \rangle \Gamma_n
\end{aligned}$$

Statement elaboration

- Expression statement

$$\frac{x \vdash_{\Gamma} \langle e, \tau \rangle}{\mathit{ExprStmt}\,(x) \vdash_{\Gamma}^{\bullet} \langle e, \tau \rangle}$$

- Conditional statement

$$\frac{\mathtt{x} \vdash_{\Gamma} \langle e, \mathit{bool} \rangle \quad \mathtt{u}_1 \vdash_{\Gamma}^{\tau} \langle e_1, \tau_1 \rangle \quad \mathtt{u}_2 \vdash_{\Gamma}^{\tau} \langle e_2, \tau_2 \rangle}{\mathit{IfStmt}\,(e, \mathtt{u}_1, \mathtt{u}_2) \vdash_{\Gamma}^{\tau} \langle \mathit{If}\,(e, e_1, e_2)\,, \mathit{void} \rangle}$$

- Return statement

$$\frac{\mathtt{x} \vdash_{\Gamma} \langle e, \tau \rangle}{\mathit{RetStmt}\,(\mathtt{x}) \vdash_{\Gamma}^{\tau} \langle \mathit{Return}\,(e)\,, \tau \rangle}$$

Statement elaboration

- Variable definition

$$\frac{\mathtt{t} \vdash_{\Gamma} \langle \tau, \natural \rangle \quad \mathtt{x} \vdash_{\langle i \mapsto \langle \mathit{Symbol}(i,\tau),\tau \rangle\rangle \Gamma} \langle e, \tau \rangle \quad i \notin \operatorname{dom} \Gamma_1}{\mathit{VarDef}\,(\mathit{Identifier}\,(i)\,,\mathtt{t},\mathtt{x}) \vdash^{\bullet}_{\langle i \mapsto \langle \mathit{Bind}(i,\tau,e),\tau \rangle\rangle \Gamma} \mathit{Bind}\,(i,\tau,e)}$$

where Γ_1 is the most recent lexical scope

$$\Gamma = \Gamma_1 \overleftarrow{\oplus} \Gamma_2$$

- Compound statements (local blocks)

$$\frac{\mathtt{u}_1 \vdash^{\tau}_{\Gamma_1} \langle e_1, \tau_1 \rangle \quad \cdots \quad \mathtt{u}_n \vdash^{\tau}_{\Gamma_n} \langle e_n, \tau_n \rangle}{\mathit{CmpdStmt}\,(\mathtt{u}_1, \ldots, \mathtt{u}_n) \vdash^{\tau}_{\Gamma} \langle \mathit{Block}\,(e_1, \ldots, e_n)\,, \mathit{void} \rangle}$$

where Γ_k is obtained from Γ_{k-1} by adding at most one binding and $\Gamma_0 = \cdot \overleftarrow{\oplus} \Gamma$