Implementing the add() protocol

John Doe

July 17, 2017

1 Introduction

We start by writing syntax, type checking rules and the operational semantics of the *add()* protocol.

2 Syntax

$\langle program\rangle$ ::= '`protocol (`'$\langle args\rangle$'`) : party`' $\langle Idlist\rangle$ '{' $\langle body\rangle$ '}'

$\langle args\rangle$ ::= $\langle arg\rangle$
| $\langle arg\rangle$,$\langle args\rangle$

$\langle arg\rangle$ ::= $\langle \tau\rangle$ $\langle Id\rangle$

$\langle \tau\rangle$ ::= '`Int`'
| '`Field`'

$\langle Idlist\rangle$ ::= $\langle Id\rangle$
| $\langle Id\rangle$,$\langle Idlist\rangle$

$\langle Id\rangle$ ::= '`[a-z, A-Z]`$^{+}$'
| $\langle Id\rangle$.$\langle Id\rangle$
| '`environment`'

$\langle body\rangle$ ::= $\langle statement\rangle^{+}$

$\langle statement\rangle$ ::= $\langle pvtstmt\rangle$
| $\langle fwdstmt\rangle$
| $\langle assgnmntstmt\rangle$
| $\langle declstmt\rangle$

$\langle pvtstmt\rangle$ ::= '`in`' $\langle Id\rangle$ '{' $\langle statment\rangle$ '}'

$\langle fwdstmt\rangle$ ::= $\langle Id\rangle$ '$\Rightarrow$' $\langle Id\rangle$

$\langle assgnmntstmt \rangle$ ::= $\langle Id \rangle$ '=' $\langle exp \rangle$

$\langle exp \rangle$ ::= n
| $\langle Id \rangle$
| $\langle exp \rangle$ '+' $\langle exp \rangle$
| '(' $\langle exp \rangle$ ')'
| $\langle exp \rangle$ '*' $\langle exp \rangle$

$\langle declstmt \rangle$ ::= $\langle \tau \rangle$ $\langle Id \rangle$

$\langle \Gamma \rangle$ ::= $\langle Id \rangle \rightarrow (\langle \tau \rangle, \langle partyScope \rangle)$

$\langle partyScope \rangle$ ::= $\langle \zeta \rangle$
| $\langle Id \rangle$

3 Type checking

The following rules states how the types checks are going to be performed

3.1 Type Environment

The *type* environment of Cooler consists of three parts.

- A type environment that contains all possible scopes denoted by A
- Γ that maps Identifier to a tuple *(type, partyScope)*. Here *type* denotes the type of the Identifier while *partyScope* denotes a scope the Identifier belongs to. A *partyScope* is the scope local to each party.
  $\Gamma(Id) \rightarrow (type,\ partyScope)$
- s denotes the current scope

3.2 Type Checking Rules

The notation used for each of the type checking rule is as follows

$$\frac{\vdots}{A, \Gamma, s \vdash e : T}$$

here A, Γ and s represents the three type environments respectively and the expression e evaluates to type T.

$$\frac{A(Id)}{A, \Gamma, s \vdash Id : T}\ [\text{Var}]$$

$$\frac{}{A, \Gamma, s \vdash n : Int}\ [\text{Int}]$$

Here x is a special Int that is not of the form *Id.Id*

$$\frac{A(x) = Int}{A, \Gamma, s \vdash x : Int} \text{ [x]}$$

$$\frac{\begin{array}{l} A, \Gamma, s \vdash e_1 : Int \\ A, \Gamma, s \vdash e_2 : Int \\ op \in \{+, *\} \end{array}}{A, \Gamma, s \vdash e_1 \ op \ e_2 : Int} \text{ [exp op exp]}$$

$$\frac{A, \Gamma, s \vdash e_1 : Int}{A, \Gamma, s \vdash (e_1) : Int} \text{ [(exp)]}$$

$$\frac{(Id, s) \notin \Gamma}{A, \Gamma, s \vdash \tau Id : \tau} \text{ [Id]}$$

$$\frac{\Gamma, s \vdash e : \zeta, \Gamma(x) = (\zeta, s)}{A, \Gamma, s \vdash x = e : \ (\ )} \text{[]}$$

$$\frac{A(x)}{A, \_, \_ \vdash x : ()} \text{ [Id]}$$

$$\frac{}{\_, \_, \_ \vdash environment : ()} \text{ [environment]}$$

$$\frac{A, \Gamma, s \vdash Id : T}{A, \Gamma, s \vdash Id \Rightarrow environment : ()} \text{[Id to env]}$$

$$\frac{A, \Gamma, s \vdash Id : T}{A, \Gamma, s \vdash environment \Rightarrow Id : ()} \text{[env to Id]}$$

$$\frac{A, \Gamma, \mathscr{G} \vdash B : ()}{A, \Gamma, s \vdash inx\{B\} : ()} \text{ [in]}$$

$$\frac{\begin{array}{c} A, \Gamma, s \vdash Id_1 : T \\ A, \Gamma, s \vdash Id_2 : T \end{array}}{A, \Gamma, s \vdash Id_1 \Rightarrow Id_2 : ()} \text{ [in]}$$

$$\frac{\begin{array}{l} A, \_, \_ \vdash x : () \\ A, \Gamma, x \vdash y : \zeta \end{array}}{A, \Gamma, \mathscr{G} \vdash x.y : \zeta} \text{ [x.y]}$$

$$\frac{scopelist, f(args), \mathscr{G} \vdash B : ()}{protocol(args) : partyscopelist\{B\} : ()} \text{ [in]}$$

4 Operational Semantics

The following section describes the operational semantics of our language: The notation used is as follows.

$$\frac{\vdots}{S, E, s \vdash e : v, S'}$$

$$\frac{\begin{array}{l} E(Id) = l \\ S(l) = v \end{array}}{S, E, s \vdash Id : v, S}\text{[Var]}$$