Adding Classes to SIMPL2

I, N, C left unchanged; B merged into E

\[
E ::= N \mid \text{true} \mid \text{false} \mid \text{unit} \mid \text{null} \mid I \mid E + E \mid E - E \mid E * E
\]

\[
E < E \mid E == E \mid E & E \mid E | E \mid \text{not } E
\]

\[
E ::= E \mid E : E \mid E : I \mid E : \text{Args} \mid \text{new } I \text{ Args} \mid (I) E
\]

\[
\text{Exps} ::= \_ \mid E \mid \text{Exps}
\]

\[
\text{ty} ::= \text{int} \mid \text{bool} \mid \text{unit} \mid \text{null} \mid I
\]

\[
\text{Ids} ::= \_ \mid \text{ty} \mid I \mid \text{Ids}
\]

\[
\text{Params} ::= \{ \text{Ids} \}
\]

\[
\text{VDec} ::= \text{ty} \mid I = E;
\]

\[
\text{Block} ::= C \mid VDec \mid \text{Block}
\]

\[
\text{MDec} ::= \text{ty} \mid \text{I Params} \{ \text{Block} \} \mid \text{Decs} ::= \_ \mid \text{VDec Decs} \mid \text{MDec Decs}
\]

\[
\text{CBody} ::= \{ \text{Decs} \} \mid \text{CDec} ::= \text{class } I \text{ extends } I \text{ CBody}
\]

\[
\text{Prog} ::= \text{CDec} \mid \text{CDec} : \text{Prog}
\]

Restriction on Classes

- Every class must define a method of the same name
- Prog must contain a class named main with params ( )
- Assumes a pre-existing “class” called Object
  - Object extends Object
  - Body of method Object just returns unit
- No instance variables; no other instance methods
- Assuming here only instance variables and instance methods in class definition
  - Will change the semantics of main some
  - Will simplify semantics of new some
- Include class variables and class methods later (if time allows)
- No overloading of methods or variables

Evaluation

- All evaluation (after Prog digested to prog) done in context of prog : Identifier → (Identifier × Decs). Also require the “current” class.

\[
(\text{prog}, C) \vdash (\text{E}, (\text{env}, \text{mem})) \downarrow (v, \text{mem}')
\]

\[
(\text{prog}, C) \vdash (C, (\text{env}, \text{mem})) \downarrow \text{mem}'
\]

\[
(\text{prog}, C) \vdash (\text{VDec}, (\text{env}, \text{mem})) \downarrow (\text{env}', \text{mem}')
\]

\[
(\text{prog}, C) \vdash (\text{MDec}, (\text{env}, \text{mem})) \downarrow (\text{env}', \text{mem}')
\]

\[
(\text{prog}, C) \vdash (\text{Decs}, (\text{env}, \text{mem})) \downarrow (\text{env}', \text{mem}')
\]

- Commands, old constructs for expressions (including boolean expressions) as before.

Notes on Syntax

- Identifiers I play four roles*
  - Expression Variable:
    \[
    E ::= I \quad \text{VDec} ::= \text{ty} I = E;
    \]
  - Field name:
    \[
    E ::= E . I
    \]
  - Method name:
    \[
    E ::= E : I \mid \text{new } I \text{ Args} \quad MDec ::= \text{ty} I \text{ Params} \{ \text{Block} \}
    \]
  - Class name:
    \[
    E ::= (\text{I} E \mid \text{new } I \text{ Args} * \quad CDec ::= \text{class } I \text{ extends } I \text{ CBody}
    \]
  *
  - C is both a class name and a method name in new C Args

Semantic Objects for Classes and Objects

- Still want Location and mem : Location → Value
- Want env : (Field/VarName × ClassName) → Location to interpret expression variables (and fields)
- Also want prog : ClassName → (ParentClassName × Decs) to interpret class names
- If prog(C1) = (C2, decs)
  - C2 is the parent class of C1
  - decs describes the additions to the instance variables / fields
- Where records evaluated to FieldName → Location, objects will eval to (CurClassName × ((FieldName × SrcClassName) → Location))
LVar Eval

As for Records, for Field assignment, need evaluation of expressions to locations (when it makes sense)

Identifiers:
- \( CI \) sees_field \( l \) at \( CI' \) in prog

\[
\text{prog}, \text{Cl} \vdash (l, (\text{env}, \text{mem})) \Downarrow (\text{env}(l, CI'), \text{mem})
\]

Field Expression:
- \( \text{objl sees_field } l \text{ at } CI' \text{ in prog} \)

\[
\text{prog}, \text{Cl} \vdash (\text{Field Expression}) \Downarrow ((\text{objl}, \text{onenv}), \text{mem''})
\]

Dec. Seq.:
- \( \text{objl sees_field } l \text{ at } CI' \text{ in prog} \)

\[
\text{prog}, \text{Cl} \vdash (E.l, (\text{env}, \text{mem})) \Downarrow (\text{env}(ci', l), \text{mem'})
\]

More Labels:

\[
\begin{align*}
(l & = E \text{. ExpRow} (\text{env}, \text{mem})) \Downarrow \{(l \rightarrow i), \text{ (env'}, \text{mem'') } [l \rightarrow v])
\end{align*}
\]

for \( i \notin \text{rng(env')} \cup \text{rng(env'')} \cup \text{dom(mem)} \)

Evaluation of Records

Record Expression:

\[
(\text{ExpRow}, (\text{env}, \text{mem})) \Downarrow ((\text{env'}, \text{mem'})]
\]

Label:

\[
(E, (\text{env}, \text{mem})) \Downarrow (v, (\text{env'}, \text{mem'})]
\]

\[
\begin{align*}
(l & = E \text{. ExpRow} (\text{env}, \text{mem})) \Downarrow \{(l \rightarrow i), \text{ (env'}, \text{mem'') } [l \rightarrow v])
\end{align*}
\]

for \( i \notin \text{rng(env')} \cup \text{rng(env'')} \cup \text{dom(mem)} \)