Model object oriented languages

Artur Jarocki

University of Wrocaw

May 28, 2014

O-1 language

O-1 is a simple language, that follows from a first-order calculi. It includes object and class-based constructs, first-order object types with subtyping and variance annotations, recursion and typecase construct.

Syntax of O-1 types

$A, B ::=$	types
X	type variable
Top	the biggest type
Object$(X)[l_i v_i : B_i^{i \in 1..n}]$	object type (l_i distinct)
Class(A)	class type

Syntax of O-1 types

$A, B ::=$	types
X	type variable
Top	the biggest type
Object$(X)[l_i v_i : B_i^{i \in 1..n}]$	object type (l_i distinct)
Class(A)	class type

O-1 includes object types and class types. For simplicity, it does not include standard basic types (like booleans), or function types.

Syntax of O-1 types

$A, B ::=$	types
X	type variable
Top	the biggest type
Object$(X)[l_i v_i : B_i^{i \in 1..n}]$	object type (l_i distinct)
Class(A)	class type

O-1 includes object types and class types. For simplicity, it does not include standard basic types (like booleans), or function types.

Object type is written in the form **Object**$(X)[l_i v_i : B_i\{X\}^{i \in 1..n}]$, where X is a type variable, each l_i, v_i, $B_i\{X\}$ is a label, variance annotation ($^-$,^{o} , or^+) and a type, respectively.

Variance annotations

The variance annotation $^{+}$ identifies read-only attributes, while ^{o} stands for read-write attributes. Typically proper methods are read-only attributes and fields are read-write attributes, but we don't require this.

Additionally, we include the variance annotation $^{-}$ for write-only attributes, but we won't use it in examples.

Syntax of O-1 terms

$a, b, c ::=$

x	variable
$\mathbf{object}(x : A) l_i = b_i^{i \in 1..n} \mathbf{end}$	direct object construction
$a.l$	field selection/method invocation
$a.l := b$	update with term
$a.l := \mathbf{method}(x : A) b\ \mathbf{end}$	update with a method
$\mathbf{new}\ c$	object construction from a class
$\mathbf{root}$	root class
$\mathbf{subclass\ of}\ c : C\ \mathbf{with}(x : A)$	subclass
$\quad l_i = b_i^{i \in n+1..n+m}$	additional attributes
$\quad \mathbf{override}\ l_i = b_i^{i \in Ovr \subseteq 1..n} \mathbf{end}$	overriden attributes
$c \hat{}\ l(a)$	class selection
$\mathbf{typecase}\ a\ \mathbf{when}(x : A) b_1$ $\mathbf{else}\ b_2\ \mathbf{end}$	typecase

O-1 includes object-based and class-based constructs. Dropping object-based constructs (object construction and method update) would result in a language similar to a traditional class-based programming, while dropping class-based (root class, subclass, new, class selection) constructs gives an object-based language.

O-1 includes object-based and class-based constructs. Dropping object-based constructs (object construction and method update) would result in a language similar to a traditional class-based programming, while dropping class-based (root class, subclass, new, class selection) constructs gives an object-based language.

Also, we don't formalize the operational semantics of these terms. It's important to distinguish fields from proper methods (for example by using naming convention), because different evaluation orders are appropriate for imperative semantics. Therefore syntax could be read either functionally or imperatively.

Additional notations

Root $\triangleq$
  Class(**Object**(X)[])

class with$(x : A)\ l_i = b_i^{i \in 1..n}$ **end** $\triangleq$
  subclass of root:Root with$(x : A) l_i = b_i^{i \in 1..n}$ **override end**

subclass of $c : C$ **with**$(x : A)$ ... **super**.l ... **end** $\triangleq$
  subclass of $c : C$ **with**$(x : A)$ ... $c\hat{}l(x)$ ... **end**

object$(x : A)$ ... l **copied from** c ... **end** $\triangleq$
  object$(x : A)$ ... $l = c\hat{}l(x)$ ... **end**

Example 1

$$\begin{aligned} \mathit{Point} \triangleq\ & \mathbf{Object}(X)[x : \mathit{Int}, \mathit{eq}^{+} : X \to \mathit{Bool}, \mathit{mv}^{+} : \mathit{Int} \to X] \\ \mathit{CPoint} \triangleq\ & \mathbf{Object}(X)[x : \mathit{Int}, c : \mathit{Color}, \mathit{eq}^{+} : \mathit{Point} \to \mathit{Bool}, \\ & \mathit{mv}^{+} : \mathit{Int} \to \mathit{Point}] \end{aligned}$$

Example 1

$$Point \triangleq \mathbf{Object}(X)[x : Int, eq^{+} : X \to Bool, mv^{+} : Int \to X]$$
$$CPoint \triangleq \mathbf{Object}(X)[x : Int, c : Color, eq^{+} : Point \to Bool, mv^{+} : Int \to Point]$$

$$pointClass : \mathbf{Class}(Point) \triangleq$$
$$\mathbf{class\ with}\ (self{:}Point)$$
$$x = 0,$$
$$eq = \mathbf{fun}(other{:}Point)\ self.x = other.x\ \mathbf{end},$$
$$mv = \mathbf{fun}(other{:}Point)\ self.x := self.x{+}dx\ \mathbf{end}$$
$$\mathbf{end}$$

Example 1

```
cPointClass : Class(CPoint) ≜
   subclass of pointClass: Class(Point)
   with (self:CPoint)
      c = black
   override
      eq = fun(other:Point)
            typecase other
            when (other':CPoint) super.eq(other') and self.c=other'.c
            else false
            end
         end
   end

cPoint:CPoint ≜ new cPointClass
```

Example 2

Calls to mv lose the color information. To access color of a moved point, a typecase is needed.

movedColor:Color $\triangleq$
typecase *cPoint.mv(1)*
when *(cp:CPoint) cp.c*
else *black*
end

$CPoint2 \triangleq \textbf{Object}(X)[x : Int, c : Color, eq^{+} : Point \to Bool, mv^{+} : Int \to X]$

```
cPointClass2:Class(CPoint2) ≜
  subclass of pointClass : Class(Point)
  with (self:CPoint2)
    c = black
  override
    eq = fun(other:Point)
      typecase other
      when (other':CPoint2)super.eq(other') and self.c = other'.c
      else false
      end
    end,
    mv = fun(dx:Int)
      typecase super.mv(dx)
      when (res:CPoint2)res
      else ... (error)
      end
    end
```

Judgements

$E \vdash \diamond$	environment E is well-formed
$E \vdash A$	A is a well-formed type in E
$E \vdash A <: B$	A is a subtype of B in E
$E \vdash vA <: v'B$	A is a subtype of B in E, with variance annotations v
$E \vdash a : A$	a has type A in E

Environments

$$\frac{}{\varnothing \vdash \diamond}\ (\text{Env } \varnothing)$$

$$\frac{E \vdash A \qquad X \notin dom(E)}{E, X <: A \vdash \diamond}\ (\text{Env } X <:)$$

$$\frac{E \vdash A \qquad X \notin dom(E)}{E, x : A \vdash \diamond}\ (\text{Env } x)$$

Types

$$\frac{E', X <: A, E'' \vdash \diamond}{E', X <: A, E'' \vdash X} \text{(Type } X\text{)}$$

$$\frac{E \vdash \diamond}{E \vdash \mathbf{Top}} \text{(Type } Top\text{)}$$

$$\frac{E, X <: \mathbf{Top} \vdash B_i \qquad \forall i \in 1..n}{E \vdash \mathbf{Object}(X)[l_i\, v_i\, B_i^{i \in 1..n}} \text{(Type Object) } (l_i \text{ distinct, } v_i \in (^o, ^-, ^+)$$

$$\frac{E \vdash A}{E \vdash \mathbf{Class}(A)} \text{(Type Class) (where } A \equiv \mathbf{Object}(X)[l_i\, v_i\, B_i\{X\}^{i \in 1..n}])$$

Subtyping

$$\frac{E \vdash A}{E \vdash A <: A} \text{(Sub Refl)}$$

$$\frac{E \vdash A <: B \qquad E \vdash B <: C}{E \vdash A <: C} \text{(Sub Trans)}$$

$$\frac{E, X <: A, E'' \vdash \diamond}{E', X <: A, E'' \vdash X <: A} \text{(Sub X)}$$

$$\frac{E \vdash A}{E \vdash A <: \textbf{Top}} \text{(Sub } \textit{Top}\text{)}$$

$$\frac{(A \equiv \mathbf{Object}(X)[l_i v_i : B_i\{X\}^{i\in 1..n+m}], A' \equiv \mathbf{Object}(X')[l_i v'_i : B'_i\{X'\}^{i\in 1..n}]) \quad E \vdash A \quad E \vdash A' \quad E, X <: A' \vdash v_i B_i\{X\} <: v'_i B'_i\{A'\} \quad \forall i \in 1..n}{E \vdash \mathbf{Top}}$$

(Sub Invariant)
$$\frac{E \vdash B}{E \vdash {}^{o}B <: {}^{o}B}$$

(Sub Covariant)
$$\frac{E \vdash B <: B' \quad v \in \{^{o}, ^{+}\}}{E \vdash vB <: {}^{+}B'}$$

(Sub Contravariant)
$$\frac{E \vdash B' <: B \quad v \in \{^{o}, ^{-}\}}{E \vdash vB <: {}^{-}B'}$$

Terms

$$\text{(Val subsumption)}\quad \frac{E \vdash a : A \qquad E \vdash A <: B}{E \vdash a : B}$$

$$\text{(Val x)}\quad \frac{E', x : A, E'' \vdash \diamond}{E', x : A, E'' \vdash x : A}$$

$$\text{(Val Object) (where } A \equiv \mathbf{Object}(X)[l_i v_i : B_i\{X\}^{i \in 1..n}]) \quad \frac{E, x : A \vdash b_i : B_i\{A\} \qquad \forall i \in 1..n}{E \vdash \mathbf{object}(x : A) l_i = b_i^{i \in 1..n}\ \mathbf{end} : A}$$

$$\text{(Val Select) (where } A \equiv \mathbf{Object}(X)[l_i v_i : B_i\{X\}^{i\in 1..n}]) \quad \frac{E \vdash a : A \qquad v_j \in \{^o, ^+\} \qquad j \in 1..n}{E \vdash a.l_j : B_j\{A\}}$$

$$\text{(Val Update) (where } A \equiv \mathbf{Object}(X)[l_i v_i : B_i\{X\}^{i\in 1..n}]) \quad \frac{E \vdash a : A \qquad E \vdash b : B_j\{A\} \qquad v_j \in \{^o, ^-\} \qquad j \in 1..n}{E \vdash a.l_j := b : A}$$

$$\text{(Val Method Update) (where } A \equiv \mathbf{Object}(X)[l_i v_i : B_i\{X\}^{i\in 1..n}]) \quad \frac{E \vdash a : A \qquad E, x : A \vdash b : B_j\{A\} \qquad v_j \in \{^o, ^-\} \qquad j \in 1..n}{E \vdash a.l_j := \mathbf{method}(x : A)b\ \mathbf{end} : A}$$

$$\text{(Val New)} \quad \frac{E \vdash c : \mathbf{Class}(A)}{E \vdash \mathbf{new}\ c : A}$$

$$\text{(Val Root)} \quad \frac{E \vdash \diamond}{E \vdash \mathbf{root{:}Class(Object}(X)[])}$$

(Val Subclass) (where $A \equiv \mathbf{Object}(X)[l_i v_i : B_i\{X\}^{i \in 1..n+m}]$, $A' \equiv \mathbf{Object}(X')[l_i v'_i : B'_i\{X\}^{i \in 1..n}]$)

$$E \vdash c' : \mathbf{Class}(A')$$
$$E \vdash A <: A'$$
$$E \vdash B'_i\{A'\} <: B_i\{A\} \qquad \forall i \in 1..n - Ovr$$
$$E, x : A \vdash b_i : B_i\{A\} \qquad \forall i \in Ovr \cup n+1..n+m$$

$$E \vdash \mathbf{subclass\ of} c' : \mathbf{Class}(A') \mathbf{with} (x : A) l_i = b_i^{i \in n+1...n+m}$$
$$\mathbf{override} l_i = b_i^{i \in Ovr} \mathbf{end} : \mathbf{Class}(A)$$

(Val Class Select) (where $A \equiv \mathbf{Object}(X)[l_i v_i : B_i^{i \in 1..n}]$)

$$\frac{E \vdash a : A \qquad E \vdash c : \mathbf{Class}(A) \qquad j \in 1..n}{E \vdash \mathrm{c}\hat{}\, l_j(a) : B_j\{A\}}$$

(Val Typecase)

$$\frac{E \vdash a : A' \qquad E, x : A \vdash b_1 : D \qquad E \vdash b_2 : D}{E \vdash \mathbf{typecase}\ a\ \mathbf{when}(x : A) b1\ \mathbf{else}\ b_2\ \mathbf{end} : D}$$

Translation

We relate O-1 to our calculi by translation. It is done by adding some type information to terms, and then proceeding by induction on the syntax of the source language.

We only need to add one rule and type information to field update: $(a : A).l_j := b$ (instead of $a.l_j := b$)

$$\text{(Val Update) (where } A \equiv \mathbf{Object}(X)[l_i v_i : B_i\{X\}^{i\in 1..n} \qquad \frac{E \vdash a : A \qquad E \vdash b : B_j\{A\} \qquad v_j \in \{^o, ^-\} \qquad j \in 1..n}{E \vdash (a : A).l_j := b : A}$$

Translation of O-1 types

$$\ll X \gg \triangleq X$$
$$\ll \textbf{Top} \gg \triangleq Top$$
$$\ll \textbf{Object}(X)[l_i v_i : B_i^{i \in 1..n}] \gg \triangleq \mu(X)[l_i v_i :\ll B_i \gg^{i \in 1..n}]$$
$$\ll \textbf{Class}(A) \gg \triangleq [new^{+} :\ll A \gg \rightarrow \ll B_i \gg \{\ll A \gg\}^{i \in 1..n}]$$

Translation of O-1 environments

$$\ll \varnothing \gg \triangleq \varnothing$$
$$\ll E, X <: A \gg \triangleq \ll E \gg, X <: \ll A \gg$$
$$\ll E, x : A \gg \triangleq \ll E \gg, x : \ll A \gg$$

Preliminary translation of O-1 terms

$$\ll x \gg \triangleq x$$

$$\ll \textbf{object}(x : A) l_i = b_i^{i \in 1..n} \textbf{ end} \gg \triangleq [l_i = \varsigma(x :\ll A \gg) \ll b_i \gg^{i \in 1..n}$$

$$\ll a.l \gg \triangleq \ll a \gg .l$$

$$\ll (a : A) l := b \gg \triangleq \ll a \gg .l :=\ll b \gg$$

$$\ll (a.l := \textbf{method}(x : A) b \textbf{ end} \triangleq \ll a \gg .l \Leftarrow \varsigma(x :\ll A \gg) \ll b \gg$$

$$\ll \textbf{new} c \gg \triangleq \ll c \gg .new$$

$$\ll \textbf{root} \gg \triangleq [new=[]]$$

$$\ll \textbf{subclass of} c' : \textbf{Class}(A') \textbf{with}(x : A) l_i = b_i^{i \in n+1..n+m}$$
$$\textbf{override } l_i = b_i^{i \in Ovr} \textbf{ end} \gg \triangleq$$
$$[new = \varsigma(z :\ll \textbf{Class}(A) \gg)[l_i = \varsigma(s :\ll A \gg) z.l_i(s)^{i \in 1..n+m}],$$
$$l_i =\ll c' \gg .l_i^{1..n-Ovr},\ l_i = \lambda(x :\ll A \gg) \ll b_i \gg^{i \in Ovr \cup n+1..n+m}$$

$$\ll c \hat{} l(a) \gg \triangleq \ll c \gg .l(\ll a \gg)$$

$$\ll \textbf{typecase } a \textbf{ when}(x : A) b_1 \textbf{ else } b_2 \textbf{ end} \gg \triangleq$$
$$typecase \ll a \gg |(x :\ll A \gg) \ll b_1 \gg | \ll b_2 \gg$$