Course Notes: Operational Semantics and the Parameterized Aspect Calculus

Curtis Clifton and Gary T. Leavens
Dept. of Computer Science
Iowa State University
226 Atanasoff Hall
Ames, IA 50011-1040 USA
{cclifton,leavens}@cs.iastate.edu

December 8, 2003

1 Motivation

1.1 Review [4, 7]

- Quantification

  Defn. 1.1 (Quantified Statements) *have an effect on many places* in the program

- Obliviousness

  Defn. 1.2 (Obliviousness) *the execution of cross-cutting code A without any reference to A from the client code that A cross-cuts*

  - interaction
  - without coupling

- Modular Reasoning

  Understanding a module M based on:

  -
  -
  -

- Behavioral Subtyping Analogy

- Behavioral subtyping in OOP:
  an overriding method must
- Behavioral subtyping is a *discipline*
  - It places constraints on
  - It provides the benefit of modular reasoning
- What about AOP?
  Q: Can a language have quantification and obliviousness *and* allow modular reasoning?

1.2 Spectators and Assistants [3]

- Assistants
  - can change the behavior of
  - must be explicitly accepted by either
    - the module containing the advised join points,
    - or a client of that module
- Spectators

  Defn. 1.3 *A* spectator *is an aspect that*

  Q: What might that mean? What is "spectator-ness"?

  - Safety and Liveness [10]

    Defn. 1.4 *A* safety property *says that*

    Defn. 1.5 *A* liveness property *says that*

    - Before-advice that immediately went into an infinite loop would
    - Before-advice that deleted all the files on your hard drive and then proceeded to the original method would
  - Spectators and Safety
    Some possible interpretations:
    - A spectator cannot

    * A spectator cannot

  Q: Is it that simple? Are there any problems with these notions?

  - Spectators and Liveness
    Goal: Spectators must always allow the advised method

    Q: Is this decidable?

    What if we:

    * Restrict control flow constructs in spectator advice

    * Run spectators

    * Approximate by

- Do you buy it?
  - Which of these notions of "spectator-ness" could be statically enforced?
  - Do spectators and assistants provide modular reasoning? How do we know?
  - Can we implement reasonable aspect-oriented programs under these restrictions?

1.3 Why formal semantics?

Defn. 1.6 *A* formal semantics *is a*

- Makes proofs about language properties tractable
- *Lingua franca* of programming language researchers

1.4 Why core calculi?

Defn. 1.7 *A* core calculus *is a programming language*

Q: What is "essential"?
A core calculus:

- Eliminates
- Makes construction of
- Can be used to define
- Examples
  - λ calculus and
  - Object calculus and
  - Parameterized aspect calculus and

2 Introduction to Formal Semantics

2.1 Kinds of Formal Semantics

Example: the semantics of a while loop

- Denotational [9]
  - Strength:
  - Map values in language to
  - Model operations in language as
  - Example:

$$[\![\textsf{while } E \textsf{ do } C;]\!]_s = w(s), \text{ where } w(s) = \mathit{if}([\![E]\!]_s, w([\![C]\!]_s), s)$$

  $[\![]\!]_s$ is overloaded:

    * $[\![E]\!]_s$: $boolean$
    * $[\![C]\!]_s$: $state$
    * **Q:** what is the type of the if function?

- Axiomatic [2]

– Strength:
– Map values in language to
– Describe operations using

– Uses *Hoare triples*: $\{P\}C\{Q\}$
  * P is a
  * Q is a
  * For two states s and s' we write:

$$(s, s') \models \{P\}C\{Q\} \text{ iff}$$

– Example:

$$\frac{C\{I\}}{\{I\}\text{while } E \text{ do } C;}$$

I is the

- Operational
  – Strength:

  – Values in language
  – Operations are described by

  General form:

$$\frac{premise_1 \quad \ldots \quad premise_n}{Env \vdash a \rightsquigarrow b}$$

– Two sorts of operational semantics

  * Small Step: a sub-term of a is replaced with a new sub-term to form b

    Example:
    The semantics of the if statement is:

$$\frac{}{\vdash \textsf{if true then } C_0 \textsf{ else } C_1 \cdot s \to C_0 \cdot s} \qquad \frac{}{\vdash \textsf{if false then } C_0 \textsf{ else } C_1 \cdot s \to}$$

$$\frac{\vdash E \cdot s \to E' \cdot s'}{\vdash \textsf{if } E \textsf{ then } C_0 \textsf{ else } C_1 \cdot s \to}$$

    and the semantics of statement sequencing is:

$$\frac{}{\vdash \textsf{skip; } C_1 \cdot s \to C_1 \cdot s} \qquad \frac{}{\vdash C_0; C_1 \cdot s \to}$$

    Using these, the semantics of the while statement is [8]:

$$\frac{}{\vdash \textsf{while } E \textsf{ do } C; \cdot s \to \textsf{if } E \textsf{ then } \qquad\qquad \textsf{else skip} \cdot s}$$

  * Big Step (a.k.a. "natural"): a is reduced to a value in one (big) step

    Example:

$$\frac{\vdash E \cdot s \rightsquigarrow \textsf{false} \cdot s'}{\vdash \textsf{while } E \textsf{ do } C; \cdot s \rightsquigarrow s'}$$

$$\frac{\vdash E \cdot s \rightsquigarrow \textsf{true} \cdot s_e \qquad \vdash C \cdot s_e \rightsquigarrow s'}{\vdash \textsf{while } E \textsf{ do } C; \cdot s \rightsquigarrow s''}$$

- Other kinds of formal semantics
  – Labelled transition systems
  – Chemical semantics

2.2 Operational semantics for the λ calculus

- Small step semantics
  - Rules
    - Top-level, one-step reduction

$$\frac{\beta}{\vdash ((\lambda x.e)\ e') \rightarrowtail e\{\!\{x \leftarrow e'\}\!\}}$$

    - One-step reduction

      Defn. 2.1 *A* context $\mathcal{C}[\![-]\!]$ *is a term with*
      $\mathcal{C}[\![e]\!]$ *represents the result of*

$$\frac{\vdash e \rightarrowtail e' \qquad \mathcal{C}[\![-]\!] \text{ is any context}}{\vdash \mathcal{C}[\![e]\!] \rightarrow \mathcal{C}[\![e']\!]}$$

    - Many-step reduction
      $\twoheadrightarrow$ is the
    - Example
  - Non-deterministic:

    Can be made deterministic by restricting the shape of contexts.
    - Normal order:
    - Applicative order?

- Big step semantics
  - Judgment: $\vdash e \rightsquigarrow v$
    The term e
  - Values
    - *
    - *
  - Rules

$$\beta \quad \frac{}{\vdash ((\lambda x.e)\ e') \rightsquigarrow v} \qquad \textsc{Rator} \quad \frac{}{\vdash (e\ e') \rightsquigarrow v} \qquad \textsc{Val} \quad \frac{}{\vdash v \rightsquigarrow v}$$

    Q: Do these rules describe applicative order? normal order? some other order?

  - Examples

$$\dfrac{\dfrac{}{\vdash 3 \rightsquigarrow 3}\ \textsc{Value}}{\vdash ((\lambda \mathsf{y}.3)\ ((\lambda \mathsf{z}.\mathsf{z})\ 2)) \rightsquigarrow 3}\ \beta$$

  - **Q:** Is this semantics deterministic?

- Abadi and Cardelli Proof Style [1, pp. 79–80]

$Judg_2$	(RULE 2)
$Judg_3$	(RULE 3)
$Judg_4$	REASON
$Judg_5$	(RULE 5)
$Judg_6$	

  Example:

$\vdash (\lambda y.3) \leadsto (\lambda y.3)$	VALUE
$\vdash ((\lambda x.x)\ (\lambda y.3)) \leadsto (\lambda y.3)$	β
$\vdash 3 \leadsto 3$	VALUE
$\vdash ((\lambda y.3)\ ((\lambda z.z)\ 2)) \leadsto 3$	β
$\vdash (((\lambda x.x)\ (\lambda y.3))\ ((\lambda z.z)\ 2)) \leadsto 3$	RATOR

2.3 Untyped Object Calculus, ς

- Syntax

$$
\begin{array}{rrcl}
\text{variables} & x & \in & Vars \\
\text{labels} & l & \in & Labels \\
\text{terms} & a,b,c & ::= & x \\
 & & \mid & [\overline{l_i = \varsigma(x_i)b_i}^{\,i\in I}] \\
 & & \mid & a.l \\
 & & \mid & a.l \Leftarrow \varsigma(x)b
\end{array}
$$

- Big step semantics

  - Object

RED OBJECT

$$\frac{}{\vdash [\overline{l_i = \varsigma(x_i)b_i}^{\,i\in I}] \leadsto [\overline{l_i = \varsigma(x_i)b_i}^{\,i\in I}]}$$

Example: [pos=ς(x)x.n, n=ς(x)2]

  - Method Selection

RED SELECT

$$\frac{\vdash a \leadsto [\overline{l_i = \varsigma(x_i)b_i}^{\,i\in I}] \qquad \vdash b_j\{\!\{x_j \leftarrow [\overline{l_i = \varsigma(x_i)b_i}^{\,i\in I}]\}\!\} \leadsto v \qquad j \in I}{\vdash a.l_j \leadsto v}$$

Example: [pos=ς(x)x.n, n=ς(x)2].pos

$\vdash [\text{pos} = \varsigma(\text{x})\text{x.n}, \text{n} = \varsigma(\text{x})2] \leadsto [\text{pos} = \varsigma(\text{x})\text{x.n}, \text{n} = \varsigma(\text{x})2]$	RED OBJECT
$\text{pos} \in \{\text{pos}, \text{n}\}$	
$\vdash [\text{pos} = \varsigma(\text{x})\text{x.n}, \text{n} = \varsigma(\text{x})2] \leadsto [\text{pos} = \varsigma(\text{x})\text{x.n}, \text{n} = \varsigma(\text{x})2]$	RED OBJECT
$\text{n} \in \{\text{pos}, \text{n}\}$	
$\vdash 2 \leadsto 2$	RED OBJECT
$\vdash [\text{pos} = \varsigma(\text{x})\text{x.n}, \text{n} = \varsigma(\text{x})2].\text{n} \leadsto 2$	RED SELECT
$\vdash [\text{pos} = \varsigma(\text{x})\text{x.n}, \text{n} = \varsigma(\text{x})2].\text{pos} \leadsto 2$	RED SELECT

– Method update

$$\frac{\vdash a \rightsquigarrow [\overline{l_i = \varsigma(x_i)b_i}^{\,i\in I}] \qquad j \in I}{\vdash a.l_j \Leftarrow \varsigma(x)b \rightsquigarrow [l_j = \varsigma(x)b, \overline{l_i = \varsigma(x_i)b_i}^{\,i\in I\setminus j}]} \text{ (Red Update)}$$

Q: What's the result of reducing this term: [pos=ς(x)x.n, n=ς(x)2].n ⇐ς(x)3

Q: What about this one: [pos=ς(x)x.n, n=ς(x)2].pos ⇐ς(x)x.n.succ

Q: What happens if we select pos on the result?

- Syntactic sugar
  - Fields: methods in which
    [pos=ς(x).n, n=2] desugars to
    [pos=ς(x).n, n=2].n := 3 desugars to
  - Lambda expressions
    Can translate untyped λ calculus into the ς calculus.
    Let $\langle\!\langle\rangle\!\rangle$ map λ calculus to ς calculus as follows:

$$\begin{aligned} \langle\!\langle x \rangle\!\rangle &= x \\ \langle\!\langle (e_1\ e_2) \rangle\!\rangle &= (\langle\!\langle e_1 \rangle\!\rangle.arg{:=}\langle\!\langle e_2 \rangle\!\rangle).val \\ \langle\!\langle (\lambda x.e) \rangle\!\rangle &= \end{aligned}$$

3 Parameterized Aspect Calculus, ς_{asp} [5, 6]

3.1 Changes vs. the object calculus

Object calculus plus aspects

- Join point abstraction
  - Each reduction step triggers
  - Search uses a four-part abstraction of the reduction step
    - *Reduction kind*, ρ
    - *Evaluation context*, $\mathcal{K}$
    - *Target signature*
      - either the set of labels in the target object, or
      - the name of a constant
    - Invocation or update *message*

  · either a label, or
  · a functional constant

– The search semantics is specified by a

  ∗ PCDL is a parameter to the calculus, various PCDL may be used
    Q: How might this be useful?

    Q: What problems might this cause?

  ∗ PCDL consists of two parts:
    ·
    ·

- Syntax of ς_{asp}
  - All object calculus terms
  - Constants

$$d \in \mathit{Consts} \qquad f \in \mathit{FConsts} \qquad \text{terms} \quad a, b, c \ ::= \ \ldots \ \mid \ d \ \mid \ a.f$$

  - Advice

$$pcd \in \boldsymbol{\mathcal{C}} \qquad \begin{array}{lrcl} \text{programs} & \mathcal{P} & ::= & a \otimes \overrightarrow{\mathcal{A}} \\ \text{advice} & \mathcal{A} & ::= & pcd \triangleright \varsigma(\overrightarrow{y})b \end{array}$$

– Proceeding

$$
\begin{array}{rrcl}
\text{terms} & a, b, c & ::= & \ldots \\
 & & | & \mathsf{proceed}_{\mathrm{VAL}}() \\
 & & | & \mathsf{proceed}_{\mathrm{IVK}}(a) \\
 & & | & \mathsf{proceed}_{\mathrm{UPD}}(a, \varsigma(x)b) \\
 & & | & \pi \\
\text{proceed closures} & \pi & ::= & \Pi_{\mathrm{VAL}}\{\!|B, v|\!\}() \\
 & & | & \Pi_{\mathrm{IVK}}\{\!|B, S, k|\!\}(a) \\
 & & | & \Pi_{\mathrm{UPD}}\{\!|B, k|\!\}(a, \varsigma(x)b)
\end{array}
$$

- Semantics
  - Changes
    - Object calculus reduction rules are changed to
    - Rules are added for:
      - Constants
      - Object calculus terms to which advice applies
      - Proceeding
  - Helper functions
    - Advice lookup

$$advFor_{\boldsymbol{M}}(jp, \bullet) = \bullet$$

$$advFor_{\boldsymbol{M}}(jp, (pcd \triangleright \varsigma(\overrightarrow{y})b) + \overrightarrow{\mathcal{A}}) = match(pcd \triangleright \varsigma(\overrightarrow{y})b, jp) + advFor_{\boldsymbol{M}}(jp, \overrightarrow{\mathcal{A}})$$

* Proceed closure

$$close_{\text{VAL}}(\textsf{proceed}_{\text{VAL}}(), \{\!|B, v|\!\}) = \Pi_{\text{VAL}}\{\!|B, v|\!\}()$$

$$close_{\text{IVK}}(\textsf{proceed}_{\text{IVK}}(a), \{\!|B, S, k|\!\}) = \Pi_{\text{IVK}}\{\!|B, S, k|\!\}(close_{\text{IVK}}(a, \{\!|B, S, k|\!\}))$$

$$close_{\text{UPD}}(\textsf{proceed}_{\text{UPD}}(a, \varsigma(x)b), \{\!|B, k|\!\}) = \Pi_{\text{UPD}}\{\!|B, k|\!\}(close_{\text{UPD}}(a, \{\!|B, k|\!\}), \varsigma(x)close_{\text{UPD}}(b, \{\!|B, k|\!\}))$$

– Objects and Basic Constants

$$\text{values} \quad v \quad ::= \quad d \mid [\overline{l_i = \varsigma(x_i)b_i}^{\,i \in I}]$$

RED VAL 0

$$\frac{\mathcal{K} \vdash_{\boldsymbol{M},\overrightarrow{\mathcal{A}}} \diamond \qquad advFor_{\boldsymbol{M}}(\langle \text{VAL}, \mathcal{K}, sig(v), \epsilon\rangle, \overrightarrow{\mathcal{A}}) = \bullet}{\mathcal{K} \vdash_{\boldsymbol{M},\overrightarrow{\mathcal{A}}} v \rightsquigarrow v}$$

RED VAL 1

$$\frac{\mathcal{K} \vdash_{\boldsymbol{M},\overrightarrow{\mathcal{A}}} \diamond \qquad advFor_{\boldsymbol{M}}(\langle \text{VAL}, \mathcal{K}, sig(v), \epsilon\rangle, \overrightarrow{\mathcal{A}}) = \varsigma()b + B \qquad close_{\text{VAL}}(b, \{\!|B, v|\!\}) = b' \qquad \textsf{va} \cdot \mathcal{K} \vdash_{\boldsymbol{M},\overrightarrow{\mathcal{A}}} b' \rightsquigarrow v'}{\mathcal{K} \vdash_{\boldsymbol{M},\overrightarrow{\mathcal{A}}} v \rightsquigarrow v'}$$

Q: What, in plain English, is the meaning of these two rules?

Things to note:

* subscripts on the turnstile
* wellformedness premise
* RED VAL 0 correspondence to RED OBJECT
* advice lookup
  · join point abstraction

· Required shape of result in RED VAL 1

* proceed closure, and information stored
* evaluation context in last premise of RED VAL 1

– Method Selection

RED SEL 0 (where $o \triangleq [\overline{l_i = \varsigma(x_i)b_i}^{\,i\in I}]$)

$$\frac{\mathcal{K} \vdash_{\boldsymbol{M},\vec{\mathcal{A}}} a \rightsquigarrow o \qquad l_j \in \overline{l_i}^{\,i\in I} \qquad advFor_{\boldsymbol{M}}(\langle \mathrm{IVK}, \mathcal{K}, \overline{l_i}^{\,i\in I}, l_j\rangle, \vec{\mathcal{A}}) = \bullet \qquad \mathsf{ib}(\overline{l_i}^{\,i\in I}, l_j) \cdot \mathcal{K} \vdash_{\boldsymbol{M},\vec{\mathcal{A}}} b_j\{\!\{x_j \leftarrow o\}\!\} \rightsquigarrow v}{\mathcal{K} \vdash_{\boldsymbol{M},\vec{\mathcal{A}}} a.l_j \rightsquigarrow v}$$

RED SEL 1 (where $o \triangleq [\overline{l_i = \varsigma(x_i)b_i}^{\,i\in I}]$)

$$\frac{\mathcal{K} \vdash_{\boldsymbol{M},\vec{\mathcal{A}}} a \rightsquigarrow o \qquad l_j \in \overline{l_i}^{\,i\in I} \qquad advFor_{\boldsymbol{M}}(\langle \mathrm{IVK}, \mathcal{K}, \overline{l_i}^{\,i\in I}, l_j\rangle, \vec{\mathcal{A}}) = \varsigma(y)b + B \qquad close_{\mathrm{IVK}}(b, \{\!|(B + \varsigma(x_j)b_j), \overline{l_i}^{\,i\in I}, l_j|\!\}) = b' \qquad \mathsf{ia} \cdot \mathcal{K} \vdash_{\boldsymbol{M},\vec{\mathcal{A}}} b'\{\!\{y \leftarrow o\}\!\} \rightsquigarrow v}{\mathcal{K} \vdash_{\boldsymbol{M},\vec{\mathcal{A}}} a.l_j \rightsquigarrow v}$$

Q: What, in plain English, is the meaning of these two rules?
Q: Where does the final value come from?

Things to note:

* correspondence of RED SEL 0 and RED SELECT
* join point abstraction
* shape of returned advice
* information stored in proceed closure
* evaluation context

– Functional Constant Application

RED FCONST 0

$$\frac{\mathcal{K} \vdash_{\boldsymbol{M},\overrightarrow{\mathcal{A}}} a \leadsto v' \qquad advFor_{\boldsymbol{M}}(\langle \text{IVK}, \mathcal{K}, sig(v'), f\rangle, \overrightarrow{\mathcal{A}}) = \bullet \qquad \mathsf{ib}(sig(v'), f) \cdot \mathcal{K} \vdash_{\boldsymbol{M},\overrightarrow{\mathcal{A}}} \delta(f, v') \leadsto v}{\mathcal{K} \vdash_{\boldsymbol{M},\overrightarrow{\mathcal{A}}} a.f \leadsto v}$$

RED FCONST 1

$$\frac{\mathcal{K} \vdash_{\boldsymbol{M},\overrightarrow{\mathcal{A}}} a \leadsto v' \qquad advFor_{\boldsymbol{M}}(\langle \text{IVK}, \mathcal{K}, sig(v'), f\rangle, \overrightarrow{\mathcal{A}}) = \varsigma(y)b + B \qquad close_{\text{IVK}}(b, \{\!|B, sig(v'), f|\!\}) = b' \qquad \mathsf{ia} \cdot \mathcal{K} \vdash_{\boldsymbol{M},\overrightarrow{\mathcal{A}}} b'\{\!\{y \leftarrow v'\}\!\} \leadsto v}{\mathcal{K} \vdash_{\boldsymbol{M},\overrightarrow{\mathcal{A}}} a.f \leadsto v}$$

Q: What is the meaning of these two rules?

Things to note:

- **Q:** Aren't these rules non-deterministic given the selection rules?
- **Q:** How do these rules differ from the selection rules?

– Method Update

RED UPD 0 (where $o \triangleq [\overline{l_i = \varsigma(x_i)b_i}^{\,i \in I}]$)

$$\frac{\mathcal{K} \vdash_{\boldsymbol{M},\overrightarrow{\mathcal{A}}} a \leadsto o \qquad l_j \in \overline{l_i}^{\,i\in I} \qquad advFor_{\boldsymbol{M}}(\langle \text{UPD}, \mathcal{K}, \overline{l_i}^{\,i\in I}, l_j\rangle, \overrightarrow{\mathcal{A}}) = \bullet}{\mathcal{K} \vdash_{\boldsymbol{M},\overrightarrow{\mathcal{A}}} a.l_j \Leftarrow \varsigma(x)b \leadsto [\overline{l_i = \varsigma(x_i)b_i}^{\,i \in I\setminus\{j\}}, l_j = \varsigma(x)b]}$$

RED UPD 1 (where $o \triangleq [\overline{l_i = \varsigma(x_i)b_i}^{\,i \in I}]$)

$$\frac{\mathcal{K} \vdash_{\boldsymbol{M},\overrightarrow{\mathcal{A}}} a \leadsto o \qquad advFor_{\boldsymbol{M}}(\langle \text{UPD}, \mathcal{K}, \overline{l_i}^{\,i\in I}, l_j\rangle, \overrightarrow{\mathcal{A}}) = \varsigma(targ, rval)b' + B \qquad close_{\text{UPD}}(b', \{\!|B, l_j|\!\}) = b'' \qquad \mathsf{ua} \cdot \mathcal{K} \vdash_{\boldsymbol{M},\overrightarrow{\mathcal{A}}} b''\{\!\{rval \hookleftarrow b\{\!\{x \leftarrow targ\}\!\}\}\!\}_{targ}\{\!\{targ \leftarrow o\}\!\} \leadsto v}{\mathcal{K} \vdash_{\boldsymbol{M},\overrightarrow{\mathcal{A}}} a.l_j \Leftarrow \varsigma(x)b \leadsto v}$$

Things to note:

- Correspondence of RED UPD 0 and RED UPDATE
- Evaluation context in RED UPD 1
- Data used for proceed closure
- Shape of returned advice: *two* parameters
  - $targ$, corresponds to
  - $rval$, corresponds to
- *two* kinds of substitution
  - $b\{\!\{x \leftarrow c\}\!\}$ is normal capture-avoiding substitution
    Key rules:

$$\begin{array}{rcl}(\varsigma(y)b)\{\!\{x \leftarrow c\}\!\} & \triangleq & \varsigma(y')(b\{\!\{y \leftarrow y'\}\!\}\{\!\{x \leftarrow c\}\!\}) \\ & & \text{where } y' \notin FV(\varsigma(y)b) \cup FV(c) \cup \{x\} \\ x\{\!\{x \leftarrow c\}\!\} & \triangleq & c \\ y\{\!\{x \leftarrow c\}\!\} & \triangleq & y \qquad\qquad \text{if } x \neq y\end{array}$$

  - $b''\{\!\{x \hookleftarrow c\}\!\}_z$ says: in b'' replace all occurances of x with c, capturing any occurances of z in c
    Key rules:

$$\begin{array}{rcl}(\varsigma(z)b)\{\!\{x \hookleftarrow c\}\!\}_z & \triangleq & \varsigma(z)(\{\!\{x \hookleftarrow c\}\!\}_z) \\ (\varsigma(y)b)\{\!\{x \hookleftarrow c\}\!\}_z & \triangleq & \varsigma(y')(b\{\!\{y \leftarrow y'\}\!\}\{\!\{x \hookleftarrow c\}\!\}_z) \\ & & \text{if } y \neq z \text{, where } y' \notin FV(\varsigma(y)b) \cup FV(c) \cup \{x\}\end{array}$$

    Q: Which of these rules does the capturing?

- Why two kinds of substitution?
  - $b\{\!\{x \leftarrow targ\}\!\}$:

  - $targ$-capturing substitution for $rval$ in the advice body, b'', lets advice author:
    capture occurrences of the self-parameter

    or
    not capture occurrences of the self-parameter

- Examples:

[n=ς(y)0, pos=ς(p)p.n].pos ⇐ ς(x)x.n.succ

  - In the absence of advice, this would reduce to:

    Q: What happens if we update n to 2 in this object and then select pos?

· Advice designed to avoid capture:

$$\varsigma(\text{targ},\text{rval})\text{proceed}_{\text{UPD}}(\text{targ},\ \varsigma(\text{z})\text{rval})$$

Assuming no other advice:

$$b'' = \Pi_{\text{UPD}}\{\!|\bullet, \text{pos}|\!\}(\text{targ},\ \varsigma(\text{z})\text{rval})$$

$$\begin{aligned}
&\Pi_{\text{UPD}}\{\!|\bullet, \text{pos}|\!\}(\text{targ},\ \varsigma(\text{z})\text{rval})\{\!\{\text{rval} \hookleftarrow \underline{\text{x.n.succ}}\{\!\{\text{x} \leftarrow \text{targ}\}\!\}\}\!\}_{\text{targ}} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad \{\!\{\text{targ} \leftarrow [\text{n}=\varsigma(\text{y})0,\ \text{pos}=\varsigma(\text{p})\text{p.n}]\}\!\} \\
&= \underline{\Pi_{\text{UPD}}\{\!|\bullet, \text{pos}|\!\}(\text{targ},\ \varsigma(\text{z})\text{rval})}\{\!\{\text{rval} \hookleftarrow \qquad\qquad \}\!\}_{\text{targ}} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad \{\!\{\text{targ} \leftarrow [\text{n}=\varsigma(\text{y})0,\ \text{pos}=\varsigma(\text{p})\text{p.n}]\}\!\} \\
&= \underline{\Pi_{\text{UPD}}\{\!|\bullet, \text{pos}|\!\}(\qquad\qquad\qquad)}\{\!\{\text{targ} \leftarrow [\text{n}=\varsigma(\text{y})0,\ \text{pos}=\varsigma(\text{p})\text{p.n}]\}\!\} \\
&= \Pi_{\text{UPD}}\{\!|\bullet, \text{pos}|\!\}([\text{n}=\varsigma(\text{y})0,\ \text{pos}=\varsigma(\text{p})\text{p.n}],\ \varsigma(\text{z})[\text{n}=\varsigma(\text{y})0,\ \text{pos}=\varsigma(\text{p})\text{p.n}].\text{n.succ})
\end{aligned}$$

The last term will reduce to:

$$[\text{n}=\varsigma(\text{y})0,\ \text{pos}=\varsigma(\text{z})[\text{n}=\varsigma(\text{y})0,\ \text{pos}=\varsigma(\text{p})\text{p.n}].\text{n.succ}]$$

Q: What happens if we update n to 2 in this object and then select pos?

· Advice designed to capture:

$$\varsigma(\text{targ},\text{rval})\text{proceed}_{\text{UPD}}(\text{targ},\varsigma(\text{targ})\text{rval.succ})$$

Assuming no other advice was found in the advice lookup, then after closing the $\text{proceed}_{\text{UPD}}$ sub-term, the substitutions for this advice are:

$$\begin{aligned}
&\Pi_{\text{UPD}}\{\!|\bullet, \text{pos}|\!\}(\text{targ},\varsigma(\text{targ})\text{rval.succ})\ \{\!\{\text{rval} \hookleftarrow \underline{\text{x.n.succ}}\{\!\{\text{x} \leftarrow \text{targ}\}\!\}\}\!\}_{\text{targ}} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad \{\!\{\text{targ} \leftarrow [\text{n}=\varsigma(\text{y})0,\ \text{pos}=\varsigma(\text{p})\text{p.n}]\}\!\} \\
&= \underline{\Pi_{\text{UPD}}\{\!|\bullet, \text{pos}|\!\}(\text{targ},\varsigma(\text{targ})\text{rval.succ})}\{\!\{\text{rval} \hookleftarrow \text{targ.n.succ}\}\!\}_{\text{targ}} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad \{\!\{\text{targ} \leftarrow [\text{n}=\varsigma(\text{y})0,\ \text{pos}=\varsigma(\text{p})\text{p.n}]\}\!\} \\
&= \underline{\Pi_{\text{UPD}}\{\!|\bullet, \text{pos}|\!\}(\text{targ},\varsigma(\text{targ})\qquad\qquad .\text{succ})} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad \{\!\{\text{targ} \leftarrow [\text{n}=\varsigma(\text{y})0,\ \text{pos}=\varsigma(\text{p})\text{p.n}]\}\!\} \\
&= \Pi_{\text{UPD}}\{\!|\bullet, \text{pos}|\!\}([\text{n}=\varsigma(\text{y})0,\ \text{pos}=\varsigma(\text{p})\text{p.n}],\ \varsigma(\text{targ})\qquad\qquad\qquad .\text{n.succ.succ})
\end{aligned}$$

This term will reduce to:

[n=ς(y)0, pos=ς(targ)targ.n.succ.succ]

Q: What happens if we update n to 2 in this object and then select pos?

- Proceeding
  - General ideas:
    - Two rules for each kind of advice
    - Rules are very similar to the regular operations, *except* ...
    - No additional advice lookup
    - Proceed closure formed
  - Proceeding from Value Advice

$$\textsc{Red VPrcd 0}\quad \frac{\mathcal{K} \vdash_{M,\vec{\mathcal{A}}} \diamond}{\mathcal{K} \vdash_{M,\vec{\mathcal{A}}} \Pi_{\mathrm{VAL}}\{\!|\bullet, v|\!\}() \rightsquigarrow v}$$

$$\textsc{Red VPrcd 1}\quad \frac{\mathcal{K} \vdash_{M,\vec{\mathcal{A}}} \diamond \qquad close_{\mathrm{VAL}}(b, \{\!|B, v|\!\}) = b' \qquad \mathsf{va} \cdot \mathcal{K} \vdash_{M,\vec{\mathcal{A}}} b' \rightsquigarrow v'}{\mathcal{K} \vdash_{M,\vec{\mathcal{A}}} \Pi_{\mathrm{VAL}}\{\!|(\varsigma()b + B), v|\!\}() \rightsquigarrow v'}$$

  - Proceeding from Selection Advice

$$\textsc{Red SPrcd 0}\quad \frac{\mathcal{K} \vdash_{M,\vec{\mathcal{A}}} a \rightsquigarrow o \qquad \mathsf{ib}(\bar{l}, l) \cdot \mathcal{K} \vdash_{M,\vec{\mathcal{A}}} b\{\!\{y \leftarrow o\}\!\} \rightsquigarrow v}{\mathcal{K} \vdash_{M,\vec{\mathcal{A}}} \Pi_{\mathrm{IVK}}\{\!|\varsigma(y)b, \bar{l}, l|\!\}(a) \rightsquigarrow v}$$

$$\textsc{Red SPrcd 1}\quad \frac{\mathcal{K} \vdash_{M,\vec{\mathcal{A}}} a \rightsquigarrow o \qquad B \neq \bullet \qquad close_{\mathrm{IVK}}(b, \{\!|B, \bar{l}, l|\!\}) = b' \qquad \mathsf{ia} \cdot \mathcal{K} \vdash_{M,\vec{\mathcal{A}}} b'\{\!\{y \leftarrow o\}\!\} \rightsquigarrow v}{\mathcal{K} \vdash_{M,\vec{\mathcal{A}}} \Pi_{\mathrm{IVK}}\{\!|(\varsigma(y)b + B), \bar{l}, l|\!\}(a) \rightsquigarrow v}$$

Q: Where does the target object in the 0 rule come from?

Q: Where does the method body evaluated in the 0 rule come from?

∗ Proceeding from Application Advice

$$\textsc{Red FPrcd 0}\quad \frac{\mathcal{K} \vdash_{M,\overrightarrow{\mathcal{A}}} a \rightsquigarrow v' \qquad \mathsf{ib}(S,f) \cdot \mathcal{K} \vdash_{M,\overrightarrow{\mathcal{A}}} \delta(f,v') \rightsquigarrow v}{\mathcal{K} \vdash_{M,\overrightarrow{\mathcal{A}}} \Pi_{\mathrm{IVK}}\{\!|\bullet, S, f|\!\}(a) \rightsquigarrow v}$$

$$\textsc{Red FPrcd 1}\quad \frac{\mathcal{K} \vdash_{M,\overrightarrow{\mathcal{A}}} a \rightsquigarrow v' \qquad close_{\mathrm{IVK}}(b, \{\!|B,S,f|\!\}) = b' \qquad \mathsf{ia} \cdot \mathcal{K} \vdash_{M,\overrightarrow{\mathcal{A}}} b'\{\!\{y \leftarrow v'\}\!\} \rightsquigarrow v}{\mathcal{K} \vdash_{M,\overrightarrow{\mathcal{A}}} \Pi_{\mathrm{IVK}}\{\!|(\varsigma(y)b + B), S, f|\!\}(a) \rightsquigarrow v}$$

∗ Proceeding from Update Advice

$$\textsc{Red UPrcd 0}\quad \frac{\mathcal{K} \vdash_{M,\overrightarrow{\mathcal{A}}} a \rightsquigarrow [\overline{l_i = \varsigma(x_i)b_i}^{\,i \in I}] \qquad l_j \in \overline{l_i}^{\,i \in I}}{\mathcal{K} \vdash_{M,\overrightarrow{\mathcal{A}}} \Pi_{\mathrm{UPD}}\{\!|\bullet, l_j|\!\}(a, \varsigma(x)b) \rightsquigarrow [\overline{l_i = \varsigma(x_i)b_i}^{\,i \in I \setminus j}, l_j = \varsigma(x)b]}$$

$$\textsc{Red UPrcd 1}\quad \frac{\begin{array}{c}\mathcal{K} \vdash_{M,\overrightarrow{\mathcal{A}}} a \rightsquigarrow o \qquad close_{\mathrm{UPD}}(b', \{\!|B, l_j|\!\}) = b'' \\ \mathsf{ua} \cdot \mathcal{K} \vdash_{M,\overrightarrow{\mathcal{A}}} b''\{\!\{rval \hookleftarrow b\{\!\{x \leftarrow targ\}\!\}\}\!\}_{targ}\{\!\{targ \leftarrow o\}\!\} \rightsquigarrow v\end{array}}{\mathcal{K} \vdash_{M,\overrightarrow{\mathcal{A}}} \Pi_{\mathrm{UPD}}\{\!|(\varsigma(targ, rval)b' + B), l_j|\!\}(a, \varsigma(x)b) \rightsquigarrow v}$$

4 Sample Point Cut Description Languages

4.1 Natural Selection, M_s

Let $\boldsymbol{M}_s = \langle \boldsymbol{\mathcal{C}}_s, match_s \rangle$, where $\boldsymbol{\mathcal{C}}_s$ ::= $[\bar{l}].l$ and:

$$match_s([\bar{l}].l \triangleright \varsigma(\overrightarrow{y})b, \langle \rho, \mathcal{K}, S, k \rangle) = \begin{cases} \langle \varsigma(\overrightarrow{y})b \rangle & \text{if } (\rho = \text{IVK}) \wedge (S = \bar{l}) \wedge (k = l) \\ \bullet & \text{otherwise} \end{cases}$$

Example:

- Without advice:
$$[\mathsf{pos} = \varsigma(\mathsf{p})\mathsf{p.n}, \mathsf{n} = \varsigma(\mathsf{y})2].\mathsf{pos} \rightsquigarrow 2$$
- With before advice $[\mathsf{pos}, \mathsf{n}].\mathsf{pos} \triangleright \varsigma(\mathsf{x})\mathsf{proceed}_{\text{IVK}}((\mathsf{x.n} \Leftarrow \varsigma(\mathsf{y})0))$:
$$[\mathsf{pos} = \varsigma(\mathsf{p})\mathsf{p.n}, \mathsf{n} = \varsigma(\mathsf{y})2].\mathsf{pos} \rightsquigarrow$$
- With after advice $[\mathsf{pos}, \mathsf{n}].\mathsf{pos} \triangleright \varsigma(\mathsf{x})\mathsf{proceed}_{\text{IVK}}(\mathsf{x}).\mathsf{succ}$:
$$[\mathsf{pos} = \varsigma(\mathsf{p})\mathsf{p.n}, \mathsf{n} = \varsigma(\mathsf{y})2].\mathsf{pos} \rightsquigarrow$$

4.2 General Matching, M_G

- Allows queries over all portions of the join point abstraction.
  - Reduction Kind
$$\boldsymbol{\mathcal{C}}_G ::= \text{VAL} \mid \text{IVK} \mid \text{UPD} \mid \ldots$$
  - Message
$$\boldsymbol{\mathcal{C}}_G ::= \ldots \mid \mathsf{k} = k \mid \ldots$$
  - Target signature
$$\boldsymbol{\mathcal{C}}_G ::= \ldots \mid \mathsf{S} = k \mid \ldots$$
  - Evaluation Context
$$\boldsymbol{\mathcal{C}}_G ::= \ldots \mid \mathsf{K} \in r \mid \ldots$$

$$\begin{array}{rrcl} \text{context expr.} & r & ::= & \epsilon \mid \mathsf{ib}(M, m) \mid \mathsf{va} \mid \mathsf{ia} \mid \mathsf{ua} \mid \\ & & & \mathbf{.} \mid r + r \mid rr \mid r^* \\ \text{signatures} & M & ::= & d \mid \bar{l} \mid \mathbf{.} \\ \text{messages} & m & ::= & f \mid l \mid \mathbf{.} \end{array}$$

- Boolean Combinations

$$\mathcal{C}_G ::= \ldots \mid \neg pcd \mid pcd \wedge pcd \mid pcd \vee pcd \mid$$

- M_G is sufficient to model AspectJ
  - Join points

AspectJ Point Cut	**Modeled In $\varsigma_{asp}(M_G)$**
call(void Point.pos())	
call(Point.new())	
execution(void Point.pos())	
get(int Point.n)	
set(int Point.n)	
adviceexecution()	
within(Point)	
withincode(Point.pos)	
cflow(Point.pos)	
cflowbelow(Point.pos)	
this(Point)	
target(Point)	

Q: Does cflowbelow consider advice execution to be "below" a cflow?
Q: Does our model?

Q: What about the variable binding form of this?

Q: What else is missing?

– Open Classes (a.k.a. intertype declarations)

```
int Point.color = 0;
```

A model of this in M_G uses two pieces of advice:

```
(VAL ∧ S = {n,pos}) ▷ ς()
  [ orig=ς(s)proceed_VAL(),
    n=ς(s)s.orig.n,
    pos=ς(s)s.orig.pos, color=ς(s)0 ]

(UPD ∧ S = {orig,n,pos,color} ∧ (k = n ∨ k = pos))▷
   ς(t,r)   [ orig=ς(s)proceed_UPD(t.orig, ς(t)r),
              n=ς(s)s.orig.n,
              pos=ς(s)s.orig.pos, color=ς(s)t.color ]
```

Q: Why is the second piece of advice needed?

4.3 Other Models

- Modeling HyperJ
  - Can use M_G
  - Like Open Classes, but two key differences:
    - ∗ Special basic constants represent module names
    - ∗ A model for abstact methods allows composed modules to call each other while remaining oblivious to the other modules implementation
- Modeling Adaptive Methods
  - Basic Idea

    Adaptive methods allow a specification of a over an .

    Specify:

    ∗

  *

  Example:

- Is M_G sufficient?

- Keys to model in ς_{asp}
  * Use distinguished names to indicate fields of objects
  * Extend M_G with

  * Use the two parameters of update advice in a unique way
    · Target object is used for dispatching to the appropriate code for the node
    · R-value is used to pass a visitor (accumulator) object

4.4 Insights

- Spectators and Assistants

  Q: Can we study them using ς_{asp}?

  Q: How might we add imperative features?

  Q: Can we eliminate any features from ς_{asp}? Should we?

- Interaction of PCDL and base language

  Q: How does the design of the PCDL effect reasoning in the base language?

- Comparisons

  Q: What do we learn about similarities between the modeled langauges?

  Q: Differences?

4.5 Decisions in the design of ς_{asp}

- Big step or little step?

- Functional or imperative?

- Include constants?

- Advice declarations or terms?

References

[1] M. Abadi and L. Cardelli. *A Theory of Objects*. Monographs in Computer Science. Springer-Verlag, New York, NY, 1996.

[2] G. Baumgartner. Axiomatic semantics, Jul 2000. http://www.cis.ohio-state.edu/~gb/cis755/slides/week4-wednesday.pdf.

[3] C. Clifton and G. T. Leavens. Spectators and assistants: Enabling modular aspect-oriented reasoning. Technical Report 02-10, Iowa State University, Department of Computer Science, Oct. 2002.

[4] C. Clifton and G. T. Leavens. Obliviousness, modular reasoning, and the behavioral subtyping analogy. Technical Report 03-01a, Iowa State University, Department of Computer Science, Mar. 2003.

[5] C. Clifton, G. T. Leavens, and M. Wand. Formal definition of the parameterized aspect calculus. Technical Report 03-12b, Iowa State University, Department of Computer Science, Nov. 2003.

[6] C. Clifton, G. T. Leavens, and M. Wand. Parameterized aspect calculus: A core calculus for the direct study of aspect-oriented languages. Technical Report 03-13, Iowa State University, Department of Computer Science, Oct. 2003. Submitted for publication.

[7] R. E. Filman and D. P. Friedman. Aspect-oriented programming is quantification and obliviousness. In M. Akşit, S. Clarke, T. Elrad, and R. E. Filman, editors, *Aspect-Oriented Software Development*. Addison-Wesley, Reading, MA, to appear.

[8] R. Rugina. Small-step operational semantics, Sep 2002. http://www.cs.cornell.edu/courses/cs611/2002fa/lectures/lec05.ps.

[9] D. A. Schmidt. *The Structure of Typed Programming Languages*. Foundations of Computing Series. MIT Press, Cambridge, Mass., 1994.

[10] F. W. Vaandrager. Safety and liveness, Nov 2003. http://www.cs.kun.nl/~fvaan/PV/SLIDES/liveness.pdf.