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:

$$\text{while } E \text{ do } C; s = w(s), \text{ where } w(s) = \text{if } (\text{eval } E, w(\text{eval } C), s)$$

$\text{eval } E$ is overloaded:

- $\text{eval } E$: boolean
- $\text{eval } C$: 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:
    \[
    (s, s') \vdash \{ P \} C \{ Q \} \text{ iff } \]

– Example:

\[
\begin{align*}
\{ I \} C I \\
\{ I \} \text{while } E \text{ do } C; \\
I \text{ is the }
\end{align*}
\]

• Operational
  – Strength:
  – Values in language
  – Operations are described by

General form:

\[
\begin{array}{c}
\text{premise}_1 \quad \ldots \quad \text{premise}_n \\
\hline
\text{Env} \vdash a \rightsquigarrow b
\end{array}
\]
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:

\[
\vdash \text{if true then } C_0 \text{ else } C_1 \cdot s \rightarrow C_0 \cdot s \\
\vdash \text{if false then } C_0 \text{ else } C_1 \cdot s \rightarrow \\
\vdash E \cdot s \rightarrow E' \cdot s' \\
\vdash \text{if } E \text{ then } C_0 \text{ else } C_1 \cdot s \rightarrow \\
\]

and the semantics of statement sequencing is:

\[
\vdash \text{skip; } C_1 \cdot s \rightarrow C_1 \cdot s \\
\vdash C_0; C_1 \cdot s \rightarrow \\
\]

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

\[
\vdash \text{while } E \text{ do } C; \cdot s \rightarrow \text{if } E \text{ then } \\
\vdash \text{else skip; } s \\
\]

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

Example:

\[
\vdash E \cdot s \leadsto \text{false} \cdot s' \\
\vdash \text{while } E \text{ do } C; \cdot s \leadsto s' \\
\vdash E \cdot s \leadsto \text{true} \cdot s_e \\
\vdash C \cdot s_e \leadsto s' \\
\vdash \text{while } E \text{ do } C; \cdot s \leadsto s'' \\
\]

Other kinds of formal semantics

- Labelled transition systems
- Chemical semantics
2.2 Operational semantics for the \( \lambda \) calculus

- Small step semantics
  - Rules
    - Top-level, one-step reduction
      \[ \beta \]
      \[ \vdash (\lambda x.e) e' \rightsquigarrow e \{ x \leftarrow e' \} \]
    - One-step reduction
      \textbf{Defn. 2.1} A context \( C[-] \) is a term with \( C[e] \) represents the result of
      \[ \vdash e \rightarrow e' \quad C[-] \text{ is any context} \]
      \[ \vdash C[e] \rightarrow C[e'] \]
    - Many-step reduction
      \( \rightarrow^* \) 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 \( e \) reduces to the value \( v \)
- Values
  * \( \ast \)
  * \( \ast \)
- Rules

\[
\begin{align*}
\beta & \quad \text{RATOR} & \quad \text{VAL} \\
\vdash ((\lambda x.e) e') \rightsquigarrow v & \quad \vdash (e e') \rightsquigarrow v & \quad \vdash v \rightsquigarrow v
\end{align*}
\]

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

- Examples

\[
\begin{align*}
\vdash 3 \rightsquigarrow 3 & \quad \text{VALUE} \\
\vdash ((\lambda y.3) ((\lambda z.z) 2)) \rightsquigarrow 3 & \quad \beta
\end{align*}
\]

- Q: Is this semantics deterministic?

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

\[
\begin{align*}
\text{(RULE 2)} \\
\text{(RULE 3)} \\
\text{REASON} \\
\text{(RULE 5)}
\end{align*}
\]

Example:
2.3 Untyped Object Calculus, $\varsigma$

- Syntax

variables $x \in Vars$

labels $l \in Labels$

terms $a, b, c ::= x$

| $\mid \left[ l_i = \varsigma(x_i)b_i \right]_{i \in I}$ |
| $\mid a.l$ |
| $\mid a.l \leftarrow \varsigma(x)b$ |

- Big step semantics

- Object

RED OBJECT

$\vdash \left[ l_i = \varsigma(x_i)b_i \right]_{i \in I} \rightsquigarrow \left[ l_i = \varsigma(x_i)b_i \right]_{i \in I}$

Example: $[pos=\varsigma(x)x.n, n=\varsigma(x)2]$

- Method Selection

RED SELECT

$\vdash a \rightsquigarrow \left[ l_i = \varsigma(x_i)b_i \right]_{i \in I}$

$\vdash b_j \left\{ x_j \leftarrow \left[ l_i = \varsigma(x_i)b_i \right]_{i \in I} \right\} \rightsquigarrow v$

$\vdash a.l_j \rightsquigarrow v$

Example: $[pos=\varsigma(x)x.n, n=\varsigma(x)2], pos$

$\vdash \left[ pos = \varsigma(x)x.n, n = \varsigma(x)2 \right] \rightsquigarrow \left[ pos = \varsigma(x)x.n, n = \varsigma(x)2 \right]$  RED OBJECT

$\vdash pos \in \{pos, n\}$

$\vdash \left[ pos = \varsigma(x)x.n, n = \varsigma(x)2 \right] \rightsquigarrow \left[ pos = \varsigma(x)x.n, n = \varsigma(x)2 \right]$  RED OBJECT

$\vdash n \in \{pos, n\}$

$\vdash 2 \rightsquigarrow 2$  RED OBJECT

$\vdash \left[ pos = \varsigma(x)x.n, n = \varsigma(x)2 \right], n \rightsquigarrow 2$  RED SELECT

$\vdash \left[ pos = \varsigma(x)x.n, n = \varsigma(x)2 \right], pos \rightsquigarrow 2$  RED SELECT
- Method update

\[
\text{RED UPDATE} \quad \frac{\vdash a \rightsquigarrow [l_i = \varsigma(x)b_i \, i \in I]}{\vdash a.l_j \leftarrow \varsigma(x)b \rightsquigarrow [l_j = \varsigma(x)b, l_i = \varsigma(x)b_i \, i \in I \setminus j]}
\]

Q: What’s the result of reducing this term: \([\text{pos}=\varsigma(x).n, n=\varsigma(x)2].n \leftarrow \varsigma(x)3\]

Q: What about this one: \([\text{pos}=\varsigma(x).n, n=\varsigma(x)2].\text{pos} \leftarrow \varsigma(x).n.\text{succ}\]

Q: What happens if we select \text{pos} on the result?

- Syntactic sugar
  - Fields: methods in which
    \([\text{pos}=\varsigma(x).n, n=2]\) desugars to
    \([\text{pos}=\varsigma(x).n, n=2].n := 3\) desugars to
  - Lambda expressions
    Can translate untyped \(\lambda\) calculus into the \(\varsigma\) calculus.
    Let \(\langle \rangle\) map \(\lambda\) calculus to \(\varsigma\) calculus as follows:
    \[
    \begin{align*}
    \langle x \rangle & = x \\
    \langle (e_1 \ e_2) \rangle & = (\langle e_1 \rangle).\text{arg}:=\langle e_2 \rangle).\text{val} \\
    \langle (\lambda x.e) \rangle & =
    \end{align*}
    \]

3 Parameterized Aspect Calculus, \(\varsigma_{asp}\) [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, \(\rho\)
    * Evaluation context, \(K\)
    * Target signature
      - either the set of labels in the target object, or
      - the name of a constant
    * Invocation or update message
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 $\varsigma_{asp}$
    - All object calculus terms
    - Constants

\[
\begin{align*}
  d & \in \text{Consts} & f & \in \text{FConsts} & \text{terms} & a, b, c & ::= & \ldots \\
& & & & & & & | d \\
& & & & & & & | a.f
\end{align*}
\]

- Advice

\[
\begin{align*}
  \text{ programs } \mathcal{P} & ::= a \otimes \overrightarrow{A} \\
  \text{ advice } \mathcal{A} & ::= pcd \downarrow \varsigma(\overrightarrow{y})b
\end{align*}
\]
- Proceeding

$$\text{terms } a, b, c \ ::= \ldots$$  
$$| \ \text{proceed}_{VAL}()$$  
$$| \ \text{proceed}_{IVK}(a)$$  
$$| \ \text{proceed}_{UPD}(a, \varsigma(x)b)$$  
$$| \ \pi$$

$$\text{proceed closures } \pi \ ::= \Pi_{VAL}\{B, v\}(\cdot)$$  
$$| \ \Pi_{IVK}\{B, S, k\}(a)$$  
$$| \ \Pi_{UPD}\{B, k\}(a, \varsigma(x)b)$$

- 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
      $$advFor_M(jp, \bullet) = \bullet$$
      $$advFor_M(jp, (\text{pcd}\triangleright \varsigma(\overline{y})b) + \overline{A}) =$$
      $$\text{match}(\text{pcd}\triangleright \varsigma(\overline{y})b, jp) + advFor_M(jp, \overline{A})$$

12
* Proceed closure

\[ \text{close}_{\text{VAL}}(\text{proceed}_{\text{VAL}}(), \{B, v\}) = \Pi_{\text{VAL}} \{B, v\}() \]

\[ \text{close}_{\text{IVK}}(\text{proceed}_{\text{IVK}}(a), \{B, S, k\}) = \Pi_{\text{IVK}} \{B, S, k\}(\text{close}_{\text{IVK}}(a, \{B, S, k\})) \]

\[ \text{close}_{\text{UPD}}(\text{proceed}_{\text{UPD}}(a, \varsigma(x)b), \{B, k\}) = \Pi_{\text{UPD}} \{B, k\}(\text{close}_{\text{UPD}}(a, \{B, k\}), \varsigma(x)\text{close}_{\text{UPD}}(b, \{B, k\})) \]

- Objects and Basic Constants

\[
\text{values } v ::= d \mid [v_i = \varsigma(x_i)b_i]_{i \in I}
\]

RED VAL 0

\[
\frac{\Pi_{\text{M}, \overline{\text{A}}} \circ \text{advFor}_M(\langle \text{VAL}, \mathcal{K}, \text{sig}(v), \epsilon \rangle, \overline{\text{A}}) = \bullet}{\Pi_{\text{M}, \overline{\text{A}}} v \rightsquigarrow v}
\]

RED VAL 1

\[
\frac{\Pi_{\text{M}, \overline{\text{A}}} \circ \text{advFor}_M(\langle \text{VAL}, \mathcal{K}, \text{sig}(v), \epsilon \rangle, \overline{\text{A}}) = \varsigma()b + B}{\text{close}_{\text{VAL}}(b, \{B, v\}) = b' \quad \mathcal{V}a : \Pi_{\text{M}, \overline{\text{A}}} b' \rightsquigarrow v'}
\]

\[
\frac{\Pi_{\text{M}, \overline{\text{A}}} v \rightsquigarrow v'}{\Pi_{\text{M}, \overline{\text{A}}} v \rightsquigarrow v'}
\]

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

\[
\text{RED SEL 0 (where } o \triangleq \{ l_i = \varsigma(x_i) b_i \}_{i \in I} \}\]

\[
K_{M, \mathcal{A}}^\vdash a \rightsquigarrow o \quad l_j \in \overline{I}_i^\vdash
\]

\[
\text{advFor}_M((IVK, K, \overline{I}_i^\vdash, l_j, \mathcal{A})) = \bullet \quad ib(\overline{I}_i^\vdash, l_j) \cdot K_{M, \mathcal{A}}^\vdash b_j \{ x_j \rightarrow o \} \rightsquigarrow v
\]

\[
K_{M, \mathcal{A}}^\vdash a.l_j \rightsquigarrow v
\]

\[
\text{RED SEL 1 (where } o \triangleq \{ l_i = \varsigma(x_i) b_i \}_{i \in I} \}\]

\[
K_{M, \mathcal{A}}^\vdash a \rightsquigarrow o \quad l_j \in \overline{I}_i^\vdash
\]

\[
\text{advFor}_M((IVK, K, \overline{I}_i^\vdash, l_j, \mathcal{A})) = \varsigma(y) b + B
\]

\[
\text{close}_{IVK}(b, \{(B + \varsigma(x_j) b_j), \overline{I}_i^\vdash, l_j\}) = B' \quad ia \cdot K_{M, \mathcal{A}}^\vdash b' \{ y \rightarrow o \} \rightsquigarrow v
\]

\[
K_{M, \mathcal{A}}^\vdash a.l_j \rightsquigarrow v
\]

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**

\[
\frac{K \vdash M, \overrightarrow{A} \triangleright a \leadsto v'}{\text{advFor}_M(\langle \text{IVK}, K, \text{sig}(v'), f \rangle, \overrightarrow{A}) = \bullet \quad \text{i}b(\text{sig}(v'), f) \cdot K \vdash M, \overrightarrow{A} \delta(f, v') \leadsto v}
\]

\[
K \vdash M, \overrightarrow{A} a.f \leadsto v
\]

**RED FCONST 1**

\[
\frac{K \vdash M, \overrightarrow{A} a \leadsto v'}{\text{advFor}_M(\langle \text{IVK}, K, \text{sig}(v'), f \rangle, \overrightarrow{A}) = \varsigma(y)b + B \quad \text{close}_{\text{IVK}}(b, \{B, \text{sig}(v'), f \}) = b' \quad \text{i}a \cdot K \vdash M, \overrightarrow{A} b'\{y \leftarrow v'\} \leadsto v}
\]

\[
K \vdash M, \overrightarrow{A} a.f \leadsto v
\]

**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 \( o \triangleq \{l_i = \varsigma(x_i)b_{i\in I}\}\))

\[
\frac{K \vdash_M, \overrightarrow{A} a \leadsto o \quad l_j \in \Gamma_i \forall i \quad \text{advFor}_M(\langle \text{UPD}, K, l_j \rangle, \overrightarrow{A}) = \bullet}{K \vdash_M, \overrightarrow{A} a.l_j \leftarrow \varsigma(x)b \leadsto \{l_i = \varsigma(x_i)b_{i\in I\setminus\{j\}}, l_j = \varsigma(x)b\}}
\]

**RED UPD 1** (where \( o \triangleq \{l_i = \varsigma(x_i)b_{i\in I}\}\))

\[
\frac{K \vdash_M, \overrightarrow{A} a \leadsto o \quad \text{advFor}_M(\langle \text{UPD}, K, l_j \rangle, \overrightarrow{A}) = \varsigma(\text{targ}, \text{rval})b' + B \quad \text{close}_{\text{UPD}}(b', \{B, l_j\}) = b'' \quad \text{ua} \cdot K \vdash_M, \overrightarrow{A} b''\{\text{rval} \leftarrow b\{x \leftarrow \text{targ}\}\}_{\text{targ}}\{\text{targ} \leftarrow o\} \leadsto v}{K \vdash_M, \overrightarrow{A} a.l_j \leftarrow \varsigma(x)b \leadsto v}
\]

**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:

\[
(\varsigma(y)b)\{x \leftarrow c\} \triangleq \varsigma(y')(b\{y \leftarrow y'\}x \leftarrow c) \\
\text{where } y' \notin FV(\varsigma(y)b) \cup FV(c) \cup \{x\} \\
x\{x \leftarrow c\} \triangleq c \\
y\{x \leftarrow c\} \triangleq y \quad \text{if } x \neq y
\]

\( b''\{x \leftarrow c\}_x \) says: in \( b'' \) replace all occurrences of \( x \) with \( c \), capturing any occurrences of \( z \) in \( c \)

Key rules:

\[
(\varsigma(z)b)\{x \leftarrow c\}_z \triangleq \varsigma(z)\{x \leftarrow c\}_z \\
(\varsigma(y)b)\{x \leftarrow c\}_z \triangleq \varsigma(y')(b\{y \leftarrow y'\}x \leftarrow c)_z \\
\text{if } y \neq z, \text{where } y' \notin FV(\varsigma(y)b) \cup FV(c) \cup \{x\}
\]

Q: Which of these rules does the capturing?

Why two kinds of substitution?
  · \( b\{x \leftarrow \text{targ}\} \):

  · 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:

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

In the absence of advice, this would reduce to:

Q: What happens if we update \( n \) to 2 in this object and then select \( \text{pos} \)?
Advice designed to avoid capture:

\[ \langle \text{targ, rval} \rangle \text{proceed}_{\text{UPD}} (\text{targ, } \langle \text{z} \rangle \text{rval}) \]

Assuming no other advice:

\[ b'' = \Pi_{\text{UPD}} \{ \bullet, \text{pos} \} (\text{targ, } \langle \text{z} \rangle \text{rval}) \]

\[
\Pi_{\text{UPD}} \{ \bullet, \text{pos} \} (\text{targ, } \langle \text{z} \rangle \text{rval}) \{ \text{rval} \leftarrow x.\text{n.succ} \{ x \leftarrow \text{targ} \} \} \]
\[
\text{targ} \{ \text{targ} \leftarrow [n=\langle \text{y} \rangle 0, \text{pos}=\langle \text{z} \rangle \text{p}.\text{n}] \}
\]
\[
= \Pi_{\text{UPD}} \{ \bullet, \text{pos} \} (\text{targ, } \langle \text{z} \rangle \text{rval}) \{ \text{rval} \leftarrow \} \]
\[
\text{targ} \{ \text{targ} \leftarrow [n=\langle \text{y} \rangle 0, \text{pos}=\langle \text{z} \rangle \text{p}.\text{n}] \}
\]
\[
= \Pi_{\text{UPD}} \{ \bullet, \text{pos} \} (\text{targ, } \langle \text{z} \rangle \text{rval}) \{ \text{rval} \leftarrow \}
\]
\[
\text{targ} \{ \text{targ} \leftarrow [n=\langle \text{y} \rangle 0, \text{pos}=\langle \text{z} \rangle \text{p}.\text{n}] \}
\]
\[
= \Pi_{\text{UPD}} \{ \bullet, \text{pos} \} ([n=\langle \text{y} \rangle 0, \text{pos}=\langle \text{z} \rangle \text{p}.\text{n}], \langle \text{z} \rangle [n=\langle \text{y} \rangle 0, \text{pos}=\langle \text{z} \rangle \text{p}.\text{n}].\text{n.succ})
\]

The last term will reduce to:

\[ [n=\langle \text{y} \rangle 0, \text{pos}=\langle \text{z} \rangle [n=\langle \text{y} \rangle 0, \text{pos}=\langle \text{z} \rangle \text{p}.\text{n}].\text{n.succ}] \]

Q: What happens if we update \( n \) to 2 in this object and then select \( \text{pos} \)?

Advice designed to capture:

\[ \langle \text{targ, rval} \rangle \text{proceed}_{\text{UPD}} (\text{targ, } \langle \text{targ} \rangle \text{rval}.\text{succ}) \]

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:

\[
\Pi_{\text{UPD}} \{ \bullet, \text{pos} \} (\text{targ, } \langle \text{targ} \rangle \text{rval}.\text{succ}) \{ \text{rval} \leftarrow x.\text{n.succ} \{ x \leftarrow \text{targ} \} \}
\]
\[
\text{targ} \{ \text{targ} \leftarrow [n=\langle \text{y} \rangle 0, \text{pos}=\langle \text{z} \rangle \text{p}.\text{n}] \}
\]
\[
= \Pi_{\text{UPD}} \{ \bullet, \text{pos} \} (\text{targ, } \langle \text{targ} \rangle \text{rval}.\text{succ}) \{ \text{rval} \leftarrow \text{targ.\text{n.succ}} \}
\]
\[
\text{targ} \{ \text{targ} \leftarrow [n=\langle \text{y} \rangle 0, \text{pos}=\langle \text{z} \rangle \text{p}.\text{n}] \}
\]
\[
= \Pi_{\text{UPD}} \{ \bullet, \text{pos} \} ([n=\langle \text{y} \rangle 0, \text{pos}=\langle \text{z} \rangle \text{p}.\text{n}], \langle \text{targ} \rangle \text{n.succ.succ})
\]
This term will reduce to:

\[ n = 0, \text{pos} = \text{targ.n.succ.succ} \]

**Q:** What happens if we update \( n \) to 2 in this object and then select \( \text{pos} \)?

- 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

\[
\text{RED VPRCD 0} \\
\frac{\mathcal{K} \vdash_M \mathcal{A} \circ}{\mathcal{K} \vdash_M \mathcal{A} \Pi_{\text{VAL}}(\bullet, v) \rightarrow v}
\]

\[
\text{RED VPRCD 1} \\
\frac{\mathcal{K} \vdash_M \mathcal{A} \circ \quad \text{close}_{\text{VAL}}(b, \langle B, v \rangle) = b' \quad \text{va} \cdot \mathcal{K} \vdash_M \mathcal{A} b' \rightarrow v'} \quad \frac{\mathcal{K} \vdash_M \mathcal{A} \Pi_{\text{VAL}}(\langle \text{succ}(b + B), v \rangle) \rightarrow v'}
\]

- Proceeding from Selection Advice

\[
\text{RED SPRCD 0} \\
\frac{\mathcal{K} \vdash_M \mathcal{A} a \rightarrow o \quad ib(\bar{t}, l) \cdot \mathcal{K} \vdash_M \mathcal{A} b y \leftarrow o \rightarrow v}{\mathcal{K} \vdash_M \mathcal{A} \Pi_{\text{IVK}}(\langle \text{succ}(y) b + B, \bar{t}, l \rangle(a) \rightarrow v)}
\]

\[
\text{RED SPRCD 1} \\
\frac{\mathcal{K} \vdash_M \mathcal{A} a \rightarrow o \quad B \neq \bullet \quad \text{close}_{\text{IVK}}(b, \langle B, \bar{t}, l \rangle) = b' \quad \text{ia} \cdot \mathcal{K} \vdash_M \mathcal{A} b' y \leftarrow o \rightarrow v}{\mathcal{K} \vdash_M \mathcal{A} \Pi_{\text{IVK}}(\langle \text{succ}(y) b + B, \bar{t}, l \rangle(a) \rightarrow v)}
\]

**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

\[ \text{RED FPRCD 0} \]
\[
\frac{\mathcal{K} \vdash_{M, \mathcal{A}} a \leadsto v' \quad \mathcal{I} = (S, f) \cdot \mathcal{K} \vdash_{M, \mathcal{A}} \delta(f, v') \leadsto v}{\mathcal{K} \vdash_{M, \mathcal{A}} \Pi_{\text{IVK}} \{ *, S, f \} (a) \leadsto v}
\]

\[ \text{RED FPRCD 1} \]
\[
\frac{\mathcal{K} \vdash_{M, \mathcal{A}} a \leadsto v' \quad \text{close}_{\text{IVK}}(B, \{ B, S, f \}) = b' \quad \mathcal{I} = \mathcal{I} \cdot \mathcal{K} \vdash_{M, \mathcal{A}} b' \{ y \leftarrow v' \} \leadsto v}{\mathcal{K} \vdash_{M, \mathcal{A}} \Pi_{\text{IVK}} \{ (\varsigma(y)b + B), S, f \} (a) \leadsto v}
\]

∗ Proceeding from Update Advice

\[ \text{RED UPRCD 0} \]
\[
\frac{\mathcal{K} \vdash_{M, \mathcal{A}} a \leadsto [l_i = \varsigma(x_i)b_i] \quad l_j \in l_i}{\mathcal{K} \vdash_{M, \mathcal{A}} \Pi_{\text{UPD}} \{ *, l_j \} (a, \varsigma(x)b) \leadsto [l_i = \varsigma(x_i)b_i, l_j = \varsigma(x)b]}
\]

\[ \text{RED UPRCD 1} \]
\[
\frac{\mathcal{K} \vdash_{M, \mathcal{A}} a \leadsto o \quad \text{close}_{\text{UPD}}(b', \{ B, l_j \}) = b'' \quad \mathcal{I} = \mathcal{I} \cdot \mathcal{K} \vdash_{M, \mathcal{A}} b'' \{ \text{rval} \leftarrow b \{ x \leftarrow \text{targ} \} \} (\text{targ} \leftarrow o) \leadsto v}{\mathcal{K} \vdash_{M, \mathcal{A}} \Pi_{\text{UPD}} \{ (\varsigma(\text{targ}, \text{rval})b' + B), l_j \} (a, \varsigma(x)b) \leadsto v}
\]
4 Sample Point Cut Description Languages

4.1 Natural Selection, $M_s$

Let $M_s = \langle C_s, match_s \rangle$, where $C_s ::= [\ ] . l$ and:

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

Example:

- Without advice:
  $$[\text{pos} = \varsigma(p).p.n, \text{n} = \varsigma(y)2].\text{pos} \rightsquigarrow 2$$

- With before advice $[\text{pos}, \text{n}].\text{pos} \triangleright \varsigma(x)\text{proceed}_{\text{IVK}}((x.\text{n} = \varsigma(y)0))$:
  $$[\text{pos} = \varsigma(p).p.n, \text{n} = \varsigma(y)2].\text{pos} \rightsquigarrow$$

- With after advice $[\text{pos}, \text{n}].\text{pos} \triangleright \varsigma(x)\text{proceed}_{\text{IVK}}(x).\text{succ}$:
  $$[\text{pos} = \varsigma(p).p.n, \text{n} = \varsigma(y)2].\text{pos} \rightsquigarrow$$

4.2 General Matching, $M_G$

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

context expr. $r ::= \epsilon \mid \text{ib}(M, m) \mid \text{va} \mid \text{ia} \mid \text{ua} \mid \star \mid r + r \mid rr \mid r^*$

signatures $M ::= d \mid I \mid \star$

messages $m ::= f \mid l \mid \star$
- Boolean Combinations

\[ C \models ::= \ldots \mid \neg pcd \mid pcd \land pcd \mid pcd \lor pcd \mid \]

- \( M_G \) is sufficient to model AspectJ
  - Join points

<table>
<thead>
<tr>
<th>AspectJ Point Cut</th>
<th>Modeled In ( \varsigma_{asp}(M_G) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>call(void Point.pos())</td>
<td></td>
</tr>
<tr>
<td>call(Point.new())</td>
<td></td>
</tr>
<tr>
<td>execution(void Point.pos())</td>
<td></td>
</tr>
<tr>
<td>get(int Point.n)</td>
<td></td>
</tr>
<tr>
<td>set(int Point.n)</td>
<td></td>
</tr>
<tr>
<td>adviceexecution()</td>
<td></td>
</tr>
<tr>
<td>within(Point)</td>
<td></td>
</tr>
<tr>
<td>withincode(Point.pos)</td>
<td></td>
</tr>
<tr>
<td>cflow(Point.pos)</td>
<td></td>
</tr>
<tr>
<td>cflowbelow(Point.pos)</td>
<td></td>
</tr>
<tr>
<td>this(Point)</td>
<td></td>
</tr>
<tr>
<td>target(Point)</td>
<td></td>
</tr>
</tbody>
</table>

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?

A: Would need to change core calculus to track this in evaluation context.

Q: What else is missing?

A: Non-sensicle: Constructor advice, initialization advice, handler advice, args point cut

A: Omitted: if (but could be handled in PCDL without changing core calculus)

Homework: Are there interesting things that can be said in $M_G$ that would give insight into join points “missing” from AspectJ?

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

int Point.color = 0;

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

\[
(V_{AL} \land S = \{n, pos\} \triangleright \varsigma() \begin{cases}
    \text{orig} = \varsigma(s).\text{proceed}_{V_{AL}}(), \\
    n = \varsigma(s).\text{orig}.n, \\
    \text{pos} = \varsigma(s).\text{orig}.\text{pos}, \text{color} = \varsigma(s).0
\end{cases}
\]

\[
(U_{PD} \land S = \{\text{orig}, n, \text{pos}, \text{color}\} \land (k = n \lor k = \text{pos}) \triangleright \varsigma(t, r) \begin{cases}
    \text{orig} = \varsigma(s).\text{proceed}_{U_{PD}}(t.\text{orig}, \varsigma(t)r), \\
    n = \varsigma(s).\text{orig}.n, \\
    \text{pos} = \varsigma(s).\text{orig}.\text{pos}, \text{color} = \varsigma(s).t.\text{color}
\end{cases}
\]

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 abstract 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 $\varsigma_{asp}$
  * 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 $\varsigma_{asp}$?
  Q: How might we add imperative features?
  Q: Can we eliminate any features from $\varsigma_{asp}$? 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 languages?
  Q: Differences?

4.5 Decisions in the design of $\varsigma_{asp}$

● Big step or little step?
● Functional or imperative?
● Include constants?
● Advice declarations or terms?
References


