Dynamic Logic for Java Card

Bernhard Beckert

UNIVERSITÄT KOBLENZ-LANDAU

November 2002

Dynamic Logic

Syntax

- Basis: typed first-order logic
- Modal operators $[p]$ and $\langle p \rangle$ for each (Java-)program p, i.e., each executable piece of Java code
- Class/interface definitions are in seperate *context* (not in p)

Semantics

- $[p]\,F$: If p terminates normally, then in its final state F holds
  (partial correctness)
- $\langle p \rangle\,F$: p terminates normally and in its final state F holds
  (total correctness)

Simple Example

$$\texttt{card.balance} \doteq 1 \vdash \langle \texttt{card.charge(17);} \rangle \texttt{card.balance} \doteq 18$$

Rule for if-else

premisses

new proof obligation

$$\frac{\Gamma,\ b = \mathsf{TRUE} \vdash \langle p \rangle F \qquad\qquad \Gamma,\ b = \mathsf{FALSE} \vdash \langle q \rangle F}{\Gamma \vdash \langle \texttt{if}\,(b)\ \{p\}\ \texttt{else}\ \{q\} \rangle F}$$

conclusion

old proof obligation

Abrupt Termination

Problem

- `throw`, `return`, `break`, `continue` abruptly change control flow
- Transforming $\langle p; q\rangle\, F$ into $\langle p\rangle\langle q\rangle\, F$ not possible

Solution

- Rules apply to "first active statement" p in $\langle \pi\; p\; \omega\rangle F$
- Prefix π: sequence of `{` `try{` `method-call(x){`

  Postfix ω: the "rest´´ of the program

Example:

$$\vdash \langle \mathtt{try\{}\ \mathtt{i{+}{+};}\ \mathtt{j{+}{+};\}finally\{...\}}\rangle \mathtt{i} = 2$$

Assignment / Aliasing

Assignment Rule

$$\frac{???}{\Gamma \vdash \langle \pi \;\; loc \; \texttt{=} \; val\,\texttt{;} \;\; \omega \rangle F}$$

"$:=$" cannot be handled by syntactic substitution!
(even if loc, val free of side-effects)

Problem: Aliasing

$$\texttt{o1.a} \doteq 1 \vdash \langle \texttt{o2.a = 2;} \rangle F$$

is `o1.a` affected?

Depends on whether `o1`, `o2` refer to the same object

Naive calculus: numerous case disctinctions!

State Updates

Solution: Updates approximate "semantical" substitution

Syntax

$$\{loc := val\}$$

- loc a simple expression denoting a location:
  - a program variable `var`
  - a field access `obj.attr`
  - an array access `arr[i]`
- val a logical term (no side effects)

Rule for assignment

$$\frac{\Gamma \vdash \{loc := val\}\langle \pi\, \omega \rangle F}{\Gamma \vdash \langle \pi\;\; loc \;\texttt{=}\; val\,;\;\; \omega \rangle F}$$

State Updates

“Application” of updates

Effect of updates computed using simplification rules

- **for terms**
- **for formulas not containing any programs**
- **NOT: for programs**
  Effect: delayed until after (symbolic) execution of progam

State Updates

“Application” to program variable

$$\{loc := val\}\,\mathtt{var} \quad = \quad \begin{cases} val & \textbf{if } loc = \mathtt{var} \\ \mathtt{var} & \textbf{otherwise} \end{cases}$$

“Application” to field access

$$\{loc := val\}u\,.\,\mathtt{a} \quad = \quad (\{loc := val\}u)\,.\,\mathtt{a} \qquad \textbf{if } loc \neq o\,.\,\mathtt{a}$$

$$\{o\,.\,\mathtt{a} := val\}u\,.\,\mathtt{a} \quad = \quad ((\{o\,.\,\mathtt{a} := val\}u) \stackrel{?}{=} o)\,.\,\mathtt{a} \mapsto val$$

Example

$$\{o.a := o\}\, o.a.a.b \quad \leadsto \quad o.b$$

Rule for if-else Revisited

Simple version

$$\frac{\Gamma,\ b = \mathsf{TRUE} \ \vdash\ \langle p \rangle F \qquad\qquad \Gamma,\ b = \mathsf{FALSE} \ \vdash\ \langle q \rangle F}{\Gamma \ \vdash\ \langle \texttt{if}\,(b)\ \{p\}\ \texttt{else}\ \{q\} \rangle F}$$

Java version

$$\frac{\Gamma,\ \mathcal{U}\,(b = \mathsf{TRUE}) \ \vdash\ \mathcal{U}\,\langle \pi\ p\ \omega \rangle F \qquad \Gamma,\ \mathcal{U}\,(b = \mathsf{FALSE}) \ \vdash\ \mathcal{U}\,\langle \pi\ q\ \omega \rangle F}{\Gamma \ \vdash\ \mathcal{U}\,\langle \pi\ \texttt{if}\,(b)\ \{p\}\ \texttt{else}\ \{q\}\ \omega \rangle F}$$

$\mathcal{U}$: updates
π: prefix
ω: “rest” of program

Scope of Program Variable Declarations

Logical variables and program variables are different

Declarations within a block: **the block**

$\mathtt{i} = 0 \vdash \langle\texttt{\{int i = 1;\}}\rangle \mathtt{i} = 0$

Declarations outside any block: **the scope of the diamond**

$\mathtt{i} = 0 \vdash \langle\texttt{int i = 1;}\rangle \mathtt{i} = 1$

becomes $\mathtt{i} \doteq 0 \vdash \{\mathtt{i1} := 1\}(\mathtt{i1} = 1)$

Scope of “free” program variables: **the whole sequent**

Rule for Exception being Caught

$$\dfrac{\begin{array}{l}\Gamma \vdash \mathcal{U}\langle \pi\, \texttt{if (}exc\ \texttt{instanceof}\ T\texttt{) \{} \\ \qquad\qquad \texttt{try\{}e\texttt{=}exc\texttt{;}\ q\texttt{\}finally\{}r\texttt{\}} \\ \qquad \texttt{\} else \{} \\ \qquad\qquad r\ \texttt{throw}\ exc\texttt{;} \\ \qquad \texttt{\}}\,\omega\,\rangle F\end{array}}{\Gamma \vdash \mathcal{U}\langle \pi\, \texttt{try\{throw}\ exc\texttt{;}\ p\texttt{\}catch(}T\ e\texttt{)\{}q\texttt{\}finally\{}r\texttt{\}}\,\omega\rangle F}$$

Rule for Exception being Caught: Example

Part of a program

```
try {throw exc; return 3;}
catch (Exception e) {return 4;}
finally {return 5;}
```

$$\dfrac{\dfrac{\Gamma \vdash \{\mathtt{e} := \mathtt{exc}\}\langle\mathtt{return\ 5;\ return\ 4;}\rangle F}{\Gamma \vdash \{\mathtt{e} := \mathtt{exc}\}\langle\mathtt{try\{return\ 4;\}finally\{return\ 5;\}}\rangle F}}{\begin{array}{c}\Gamma \vdash \ \mathit{instanceof}(\mathtt{exc},\mathtt{Exception}) \\ \Gamma \vdash \ \langle\mathtt{try\{e{=}exc;\ return\ 4;\}finally\{return\ 5;\}}\rangle F\end{array}}$$

$$\dfrac{\begin{array}{c}\Gamma \vdash \ \mathit{instanceof}(\mathtt{exc},\mathtt{Exception}) \\ \Gamma \vdash \ \langle\mathtt{try\{e{=}exc;\ return\ 4;\}finally\{return\ 5;\}}\rangle F\end{array}}{\begin{array}{c}\Gamma \vdash \ \langle\mathtt{try\{throw\ exc;\ return\ 3;\}} \\ \mathtt{catch(Exception}\ e\mathtt{)\{return\ 4;\}} \\ \mathtt{finally\{return\ 5;\}}\rangle F\end{array}}$$

Rule for while Loops

Symbolic execution of one loop iteration

target `continue`

$$\frac{\Gamma \vdash \langle \texttt{if}\,(b)\; l_1 : \{ l_2 : \{ p\texttt{'} \}\ \ \texttt{while}\,(b)\,\{p\} \} \rangle F}{\Gamma \vdash \langle \texttt{while}\,(b)\,\{p\} \rangle F}$$

target `break`

Construction of p':

`break`	$\rightarrow$	`break` l_1
`continue`	$\rightarrow$	`break` l_2

Rule for while Loops: Example

Part of a program

$$\left.\begin{array}{l}\texttt{while (true) \{}\\ \quad\texttt{if (i==10)}\\ \qquad\texttt{break;}\\ \quad\texttt{i++;}\\ \texttt{\}}\end{array}\right\} p$$

Rule application

$$\frac{\Gamma \vdash \langle \texttt{if(true)l1:\{l2:\{if(i==10) break l1; i++;\}} \; \texttt{while(true)\{}p\texttt{\}} \; \texttt{\}}\rangle F}{\Gamma \vdash \langle \texttt{while(true)\{}p\texttt{\}}\rangle F}$$

More Java/OO Features

Attributes and Methods

- **Attributes**
  Treated as non-rigid functions
- **Dereferencing attributes and methods**
  Calculus must implement Java's binding rules
  (depending on context, visibility, etc.)
- **Dynamic binding**
- **Polymorphism**

Objects

- **Object creation**
- **Object initialisation**

More Java/OO Features II

Run-time exceptions (eg. `null` pointer)

Built-in data types (cyclic integers)

Complex expressions with side effects

Java vs. Java Card

Java features (not Java Card features)
that cannot be handled easily (long-term goals)

- **Threads** (unrestricted concurrency)
- **Floating-point arithmetic**
- **Dynamic class loading**
- **Graphical I/O**