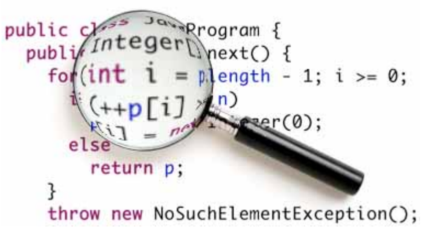
15-819M: Data, Code, Decisions

08: Essentials of Object-oriented Dynamic Logic

André Platzer

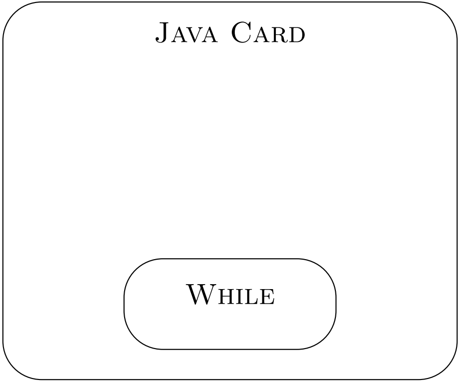
aplatzer@cs.cmu.edu
Carnegie Mellon University, Pittsburgh, PA



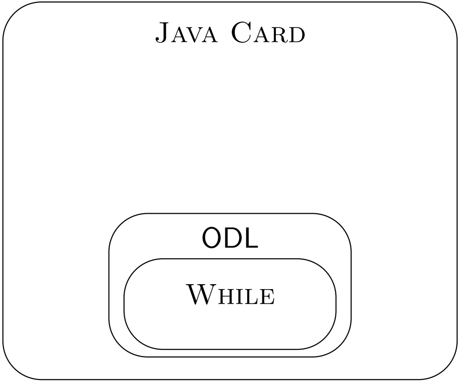
Outline

Outline

Object-oriented Dynamic Logic

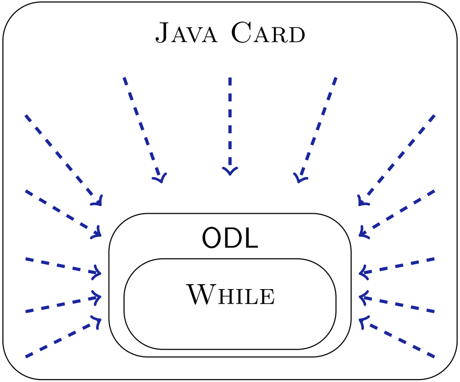


Object-oriented Dynamic Logic



- ODL only contains essentials of object-orientation.

Object-oriented Dynamic Logic

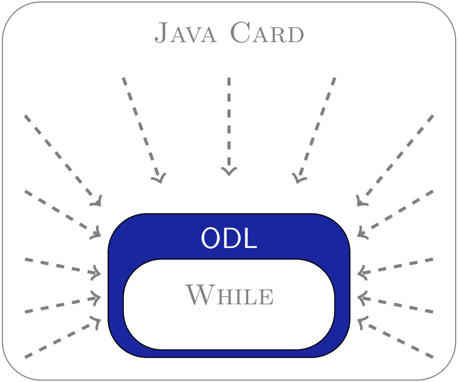


✓ Theoretical investigations

✓ Program verification

- ODL only contains essentials of object-orientation.
- "Natural" representation of object-orientation.

Object-oriented Dynamic Logic



✓ Theoretical investigations

✓ Program verification

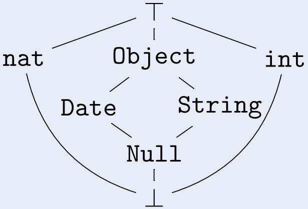
- ODL only contains essentials of object-orientation.
- “Natural” representation of object-orientation.

Outline

Outline

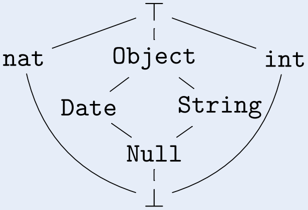
The Logic ODL

Definition (Type system)



The Logic ODL

Definition (Type system)



- nat
- finite subtypes (object-types)

The Logic ODL

Definition (Object enumerator symbols)

$\texttt{obj}_{\texttt{C}} : \texttt{nat} \to \texttt{C}$ (disjoint bijections for object-types $\texttt{C}$)

Example

$$\texttt{obj}_{\texttt{C}}(3)$$

Definition (Flexible function symbols)

...change value during program execution & represent object attributes.

Example (Object attribute representation)

$$\boxed{\mathsf{x}.a} \rightsquigarrow \boxed{a(\mathsf{x})}$$

The Logic ODL

Definition (Formulas ϕ)

$\neg, \wedge, \vee, \rightarrow, \leftrightarrow, \ \forall, \exists, \ \doteq$ (first-order part)

$[\alpha]\phi, \quad \langle\alpha\rangle\phi$ (dynamic part)

Definition (Programs α)

$\alpha;\gamma, \quad \texttt{if}(\phi)\,\alpha\,\texttt{else}\,\gamma, \quad \texttt{while}(\phi)\,\alpha$ (control-structure)

$f(t) := t', g(r) := r'$ (state update)

The Logic ODL: Semantics

Definition (State)

State = first-order structure

Definition (Semantics of programs)

1. $(s, s') \in \rho_{I,\beta}(f_1(t_1^1, \ldots, t_1^{k_1}) := t_1, \ldots, f_n(t_n^1, \ldots, t_n^{k_n}) := t_n) :\Longleftrightarrow$
   - $s = s_0, s' = s_n$, and
   - $s_i = s_{i-1}[f_i(val_{I,\beta}(s, t_i^1), \ldots, val_{I,\beta}(s, t_i^{k_i})) \mapsto val_{I,\beta}(s, t_i)]$ $(1 \leq i \leq n)$.
2. $(s, s') \in \rho_{I,\beta}(\alpha; \gamma) :\Longleftrightarrow (s, u) \in \rho_{I,\beta}(\alpha)$ and $(u, s') \in \rho_{I,\beta}(\gamma)$ for some state u.
3. $(s, s') \in \rho_{I,\beta}(\texttt{if}(\phi)\, \alpha\, \texttt{else}\, \gamma) :\Longleftrightarrow$
   - $val_{I,\beta}(s, \phi) = true$ and $(s, s') \in \rho_{I,\beta}(\alpha)$, *or*
   - $val_{I,\beta}(s, \phi) = false$ and $(s, s') \in \rho_{I,\beta}(\gamma)$.
4. $(s, s') \in \rho_{I,\beta}(\texttt{while}(\phi)\, \alpha)$ iff there are $n \in \mathbb{N}$ and $s = s_0, \ldots, s_n = s'$
   - for $0 \leq i < n$, $val_{I,\beta}(s_i, \phi) = true$ and $(s_i, s_{i+1}) \in \rho_{I,\beta}(\alpha)$, and
   - $val_{I,\beta}(s_n, \phi) = false$.

Outline

Java $\leadsto$ ODL

Simple & natural translation of

- Object creation
- Dynamic dispatch
- Side-effects
- Exception handling
- Inner classes

JAVA $\rightsquigarrow$ ODL: Object Creation

Example (Transformation)

$$\boxed{\mathtt{x} := \mathtt{new}\ \mathtt{C}()} \rightsquigarrow \boxed{\begin{array}{l} \mathtt{x} := \mathtt{obj}_{\mathtt{C}}(\mathit{next}_{\mathtt{C}}), \\ \mathit{next}_{\mathtt{C}} := \mathit{next}_{\mathtt{C}} + 1 \end{array}}$$

JAVA $\rightsquigarrow$ ODL: Object Creation

Example (Transformation)

$$\boxed{x := \texttt{new}\, \texttt{C}()} \rightsquigarrow \boxed{\begin{array}{l} x := \texttt{obj}_{\texttt{C}}(\mathit{next}_{\texttt{C}}), \\ \mathit{next}_{\texttt{C}} := \mathit{next}_{\texttt{C}} + 1 \end{array}}$$

- Object identity "new $\neq$ new"

Example

$x := \texttt{new}\, \texttt{C}();$	$\rightsquigarrow$	$x := \texttt{obj}_{\texttt{C}}(\mathit{next}_{\texttt{C}});$ $//\mathit{next}_{\texttt{C}} := \mathit{next}_{\texttt{C}} + 1$
$y := \texttt{new}\, \texttt{C}();$	$\rightsquigarrow$	$y := \texttt{obj}_{\texttt{C}}(\mathit{next}_{\texttt{C}} + 1)$ $//x \neq y$

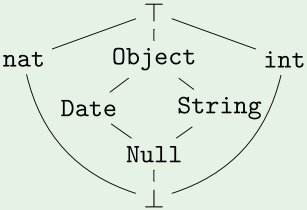
Java $\rightsquigarrow$ ODL: Object Creation

Example (Transformation)

$$\boxed{x := \mathtt{new}\,\mathtt{C}()} \rightsquigarrow \boxed{\begin{array}{l} x := \mathtt{obj}_{\mathtt{C}}(next_{\mathtt{C}}), \\ next_{\mathtt{C}} := next_{\mathtt{C}} + 1 \end{array}}$$

- Object identity “new $\neq$ new”
- Dynamic type checks

Example



$$\boxed{t\ \mathtt{instanceof}\ \mathtt{String}}$$

$$\rightsquigarrow$$

$$\boxed{\exists n : \mathtt{nat}\ t \doteq \mathtt{obj}_{\mathtt{String}}(n)}$$

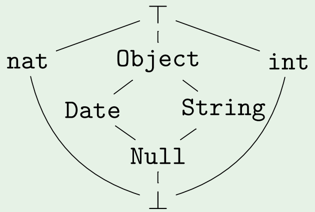
JAVA $\rightsquigarrow$ ODL: Object Creation

Example (Transformation)

$$\boxed{\mathtt{x} := \mathtt{new}\ \mathtt{C}()} \rightsquigarrow \boxed{\begin{array}{l}\mathtt{x} := \mathtt{obj}_{\mathtt{C}}(next_{\mathtt{C}}),\\ next_{\mathtt{C}} := next_{\mathtt{C}} + 1\end{array}}$$

- Object identity "new $\neq$ new"
- Dynamic type checks

Example



$$\boxed{t\ \mathtt{instanceof}\ \mathtt{Object}}$$

$$\Downarrow$$

$$\boxed{\exists n : \mathtt{nat}\ (\ t \doteq \mathtt{obj}_{\mathtt{Date}}(n) \lor t \doteq \mathtt{obj}_{\mathtt{String}}(n) \lor t \doteq \mathtt{obj}_{\mathtt{Object}}(n))}$$

JAVA $\leadsto$ ODL: Object Creation

Example (Transformation)

$$\boxed{\mathtt{x} := \mathtt{new}\,\mathtt{C}()} \leadsto \boxed{\begin{array}{l}\mathtt{x} := \mathtt{obj}_{\mathtt{C}}(next_{\mathtt{C}}),\\ next_{\mathtt{C}} := next_{\mathtt{C}} + 1\end{array}}$$

- Object identity "new $\neq$ new"
- Dynamic type checks

JAVA $\rightsquigarrow$ ODL: Side-effects

```
a[i++] = b-- + b
```

$$vi := i;\ \ i := i + 1;\ \ vb := b;\ \ b := b - \overset{\lightning}{1};\ \ a(vi) := vb + b$$

$$i := i + 1, b := b - 1, a(i) := b + (b \overset{\lightning}{-} 1)$$

Return

Java ⇝ ODL: Exception Handling

```
try { while (d != 0)
       {if (d < 0) {throw new RangeEx(d);} else {d=d-1;}}
      /* do something */
    } catch (RangeEx r) {/* handle range */}
```

⇝

```
Exception r = null;
while (r == null && d != 0)
  {if (d < 0) {r = new RangeEx(d);} else {d=d-1;}}
if (r == null) {/* do something */}
else if (r instanceof RangeEx) {/* handle range */}
else {return r; /* pass up the call trace */}
```

Return

Java ⇝ ODL: Dynamic Dispatch

```
class Car { int follow(Car d) {...} }
class Van extends Car { int follow(Car d) {...} }
... return b.follow(d);
```

$\rightsquigarrow$

```
class Car { int Car_follow(Car d) {...} }
class Van extends Car { int Van_follow(Car d) {...} }
...  if(b instanceof Van) {return ((Van)b).Van_follow(d);}
else if(b instanceof Car) {return ((Car)b).Car_follow(d);}
else {/* cannot happen when all types are known */}
```

Type-casts expressible as

$$\exists v{:}\mathrm{Van}\, v \doteq b$$

◂ Return

Outline

Calculus: State Updates (merge & promote)

- Merge updates

$$\langle \mathcal{U} \rangle \langle f(t) := t' \rangle \phi \quad \rightsquigarrow \quad \Big\langle \mathcal{U},\ f(\langle \mathcal{U} \rangle t) := \langle \mathcal{U} \rangle t' \Big\rangle \phi$$

"Last change is most recent"

Example

$$\big\langle g(a) := t \big\rangle \big\langle f(g(a)) := h(g(a)) \big\rangle \phi \ \rightsquigarrow\ \big\langle g(a) := t, f(t) := h(t) \big\rangle \phi$$

Calculus: State Updates (merge & promote)

- Merge updates

$$\langle\mathcal{U}\rangle\langle f(t):=t'\rangle\phi \quad\rightsquigarrow\quad \Big\langle\mathcal{U},\ f(\langle\mathcal{U}\rangle t):=\langle\mathcal{U}\rangle t'\Big\rangle\phi$$

"Last change is most recent"

- Promote updates

$$\langle f(t):=t',\ \ g(t):=c,\ \ f(s):=s'\rangle\ \ f(a)$$

Calculus: State Updates (merge & promote)

- Merge updates

$$\langle \mathcal{U} \rangle \langle f(t) := t' \rangle \phi \quad \rightsquigarrow \quad \Big\langle \mathcal{U},\ f(\langle \mathcal{U} \rangle t) := \langle \mathcal{U} \rangle t' \Big\rangle \phi$$

"Last change is most recent"

- Promote updates

$$\langle \underbrace{f(t) := t',\ \ g(t) := c,\ \ f(s) := s'}_{f} \rangle \ \ f(a)$$

Calculus: State Updates (merge & promote)

- Merge updates

$$\langle \mathcal{U} \rangle \langle f(t) := t' \rangle \phi \quad \rightsquigarrow \quad \Big\langle \mathcal{U},\ f(\langle \mathcal{U} \rangle t) := \langle \mathcal{U} \rangle t' \Big\rangle \phi$$

"Last change is most recent"

- Promote updates

$$\langle f(t) := t',\ \ g(t) := c,\ \ f(s) := s' \rangle \quad f(a)$$

$$f$$

$$\langle f(t) := t',\ \ f(s) := s' \rangle \qquad f(a)$$

Calculus: State Updates (merge & promote)

- Merge updates

$$\langle \mathcal{U} \rangle \langle f(t) := t' \rangle \phi \quad \rightsquigarrow \quad \Big\langle \mathcal{U},\ f(\langle \mathcal{U} \rangle t) := \langle \mathcal{U} \rangle t' \Big\rangle \phi$$

"Last change is most recent"

- Promote updates

$$\langle f(t) := t',\ \ g(t) := c,\ \ f(s) := s' \rangle \quad f(a)$$

$$f$$

$$\langle f(t) := t',\ \ f(s) := s' \rangle \qquad f(a)$$

$$\mathit{if}\, s \doteq a \,\mathit{then}\, s' \,\mathit{else}\, \ldots \ \mathit{fi}$$

Calculus: State Updates (merge & promote)

- Merge updates

$$\langle\mathcal{U}\rangle\langle f(t) := t'\rangle\phi \quad\rightsquigarrow\quad \Big\langle\mathcal{U},\ f(\langle\mathcal{U}\rangle t) := \langle\mathcal{U}\rangle t'\Big\rangle\phi$$

"Last change is most recent"

- Promote updates

$$\langle f(t) := t',\ \ g(t) := c,\ \ f(s) := s'\rangle\ \ f(a)$$

$$f$$

$$\langle f(t) := t',\ \ f(s) := s'\rangle \qquad f(a)$$

$$\mathit{if}\, s \doteq a\ \mathit{then}\ s'\ \mathit{else}\ \mathit{if}\ t \doteq a\ \mathit{then}\ t'\ \mathit{else}\ f(a)\ \mathit{fi}\ \mathit{fi}$$

Calculus: State Updates (merge & promote)

- Merge updates

$$\langle\mathcal{U}\rangle\langle f(t) := t'\rangle\phi \quad\leadsto\quad \Big\langle\mathcal{U},\ f(\langle\mathcal{U}\rangle t) := \langle\mathcal{U}\rangle t'\Big\rangle\phi$$

"Last change is most recent"

- Promote updates

$$\overbrace{\langle f(t) := t',\ \ g(t) := c,\ \ f(s) := s'\rangle}^{\mathcal{U}} \quad f(a)$$

$$f$$

$$\langle f(t) := t',\ \ f(s) := s'\rangle \qquad f(a)$$

$$\mathit{if}\ s \doteq \langle\mathcal{U}\rangle a\ \mathit{then}\ s'\ \mathit{else}\ \mathit{if}\ t \doteq \langle\mathcal{U}\rangle a\ \mathit{then}\ t'\ \mathit{else}\ f(\langle\mathcal{U}\rangle a)\ \mathit{fi}\ \mathit{fi}$$

Calculus: State Updates (merge & promote)

- Merge updates

$$\langle\mathcal{U}\rangle\langle f(t) := t'\rangle\phi \quad\rightsquigarrow\quad \Big\langle\mathcal{U},\ f(\langle\mathcal{U}\rangle t) := \langle\mathcal{U}\rangle t'\Big\rangle\phi$$

"Last change is most recent"

- Promote updates

$$\phi\Big(\ \overbrace{\langle f(t) := t',\ \ g(t) := c,\ \ f(s) := s'\rangle}^{\mathcal{U}}\ \ f(a)\ \Big)$$

$$f$$

$$\phi\Big(\ \langle f(t) := t',\ \ f(s) := s'\rangle\ \ f(a)\ \Big)$$

$$\phi\Big(\ \mathit{if}\, s \doteq \langle\mathcal{U}\rangle a\ \mathit{then}\ s'\ \mathit{else}\ \mathit{if}\ t \doteq \langle\mathcal{U}\rangle a\ \mathit{then}\ t'\ \mathit{else}\ f(\langle\mathcal{U}\rangle a)\ \mathit{fi}\ \mathit{fi}\ \Big)$$

Calculus: State Updates (merge & promote)

- Merge updates

$$\langle\mathcal{U}\rangle\langle f(t) := t'\rangle\phi \quad\rightsquigarrow\quad \Big\langle \mathcal{U},\ f(\langle\mathcal{U}\rangle t) := \langle\mathcal{U}\rangle t' \Big\rangle\phi$$

"Last change is most recent"

- Promote updates

$$\phi\Big(\overbrace{\langle f(t) := t',\ \ g(t) := c,\ \ f(s) := s'\rangle}^{\mathcal{U}}\ f(a) \Big)$$

$$f$$

$$\phi\Big(\langle f(t) := t',\ \ f(s) := s'\rangle \quad f(a) \Big)$$

$$\phi\Big(\mathit{if}\, s \doteq \langle\mathcal{U}\rangle a\ \mathit{then}\ s'\ \mathit{else}\ \mathit{if}\ t \doteq \langle\mathcal{U}\rangle a\ \mathit{then}\ t'\ \mathit{else}\ f(\langle\mathcal{U}\rangle a)\ \mathit{fi}\ \mathit{fi} \Big)$$

$$\Big(s \doteq \langle\mathcal{U}\rangle a \ \rightarrow\ \phi(s')\Big) \ \wedge\ \Big(s \neq \langle\mathcal{U}\rangle a \ \rightarrow\ \ldots\phi(f(\langle\mathcal{U}\rangle a))\Big)$$

Calculus: State Updates

$\langle \mathcal{U} \rangle f(u) \rightsquigarrow$
$\mathit{if}\, s_{i_r} \doteq \langle \mathcal{U} \rangle u \;\mathit{then}\; t_{i_r} \;\mathit{else}\; \ldots \mathit{if}\, s_{i_1} \doteq \langle \mathcal{U} \rangle u \;\mathit{then}\; t_{i_1} \;\mathit{else}\; f(\langle \mathcal{U} \rangle u) \;\mathit{fi} \ldots \mathit{fi}$
where $i_1 < \cdots < i_r$ are all those indices with $f_{i_j} = f$, for some $r \geq 0$

Example (State updates with aliasing)

$$\begin{array}{ll} & \langle f(s) := t \rangle g(f(r)) \\ \rightsquigarrow & g\big(\langle f(s) := t \rangle f(r)\big) \\ \rightsquigarrow & g\big(\mathit{if}\, s \doteq r \;\mathit{then}\; t \;\mathit{else}\; f(r) \;\mathit{fi}\big) \\ \text{“}\rightsquigarrow\text{”} & \big(s \doteq r \rightarrow g(t)\big) \wedge \\ & \big(s \neq r \rightarrow g(f(r))\big) \end{array}$$

(R1) $\quad \langle \tilde{\mathcal{U}} \rangle \langle \mathcal{U} \rangle \phi \rightsquigarrow \Big\langle \tilde{\mathcal{U}}, f_1(\langle \tilde{\mathcal{U}} \rangle s_1) := \langle \tilde{\mathcal{U}} \rangle t_1, \ldots, f_n(\langle \tilde{\mathcal{U}} \rangle s_n) := \langle \tilde{\mathcal{U}} \rangle t_n \Big\rangle \phi$

Calculus: First-order Part (18 rules)

$$\frac{\vdash A}{\neg A \vdash} \qquad \frac{A, B \vdash}{A \land B \vdash} \qquad \frac{A \vdash \quad B \vdash}{A \lor B \vdash} \qquad \frac{\vdash A \quad B \vdash}{A \to B \vdash}$$

$$\frac{A \vdash}{\vdash \neg A} \qquad \frac{\vdash A \quad \vdash B}{\vdash A \land B} \qquad \frac{\vdash A, B}{\vdash A \lor B} \qquad \frac{A \vdash B}{\vdash A \to B}$$

$$\frac{\vdash A_x^X}{\vdash \forall x\, A} \qquad \frac{A_x^t, \forall x\, A \vdash}{\forall x\, A \vdash} \qquad \frac{A_x^X \vdash}{\exists x\, A \vdash} \qquad \frac{\vdash A_x^t, \exists x\, A}{\vdash \exists x\, A}$$

$$\frac{}{A \vdash A} \qquad \frac{\Gamma_t^{t'}, t \doteq t' \vdash \Delta_t^{t'}}{\Gamma, t \doteq t' \vdash \Delta} \qquad \frac{}{\vdash t \doteq t}$$

$$\frac{A \vdash \quad \vdash A}{\vdash} \qquad \frac{\Gamma_t^{t'}, t' \doteq t \vdash \Delta_t^{t'}}{\Gamma, t' \doteq t \vdash \Delta} \qquad \frac{\vdash \phi(0) \quad \phi(X) \vdash \phi(X{+}1)}{\vdash \forall n\, \phi(n)}$$

Calculus: Program Logic Part (12 rules)

$$\frac{\langle\alpha\rangle\langle\gamma\rangle\phi}{\langle\alpha;\gamma\rangle\phi}$$

$$\frac{}{\vdash \mathtt{obj_C}(i) \doteq \mathtt{obj_C}(j) \rightarrow i \doteq j}$$

$$\frac{(e \rightarrow \langle\alpha\rangle\phi) \land (\neg e \rightarrow \langle\gamma\rangle\phi)}{\langle \mathtt{if}(e)\,\alpha\,\mathtt{else}\,\gamma\rangle\phi}$$

$$\frac{}{\vdash \neg(\mathtt{obj_C}(i) \doteq \mathtt{obj_D}(j))}$$

$$\frac{(e \rightarrow \phi(t)) \land (\neg e \rightarrow \phi(t'))}{\phi(\mathit{if}\; e\; \mathit{then}\; t\; \mathit{else}\; t'\; \mathit{fi})}$$

$$\frac{}{\vdash \forall o : \mathtt{C}\; (o\; \mathtt{instanceof}\; \mathtt{C} \lor o \doteq \mathtt{null})}$$

$$\frac{\langle \mathtt{if}(e)\,\{\alpha; \mathtt{while}(e)\,\alpha\}\rangle\phi}{\langle \mathtt{while}(e)\,\alpha\rangle\phi}$$

$$\frac{\Gamma \vdash \langle\mathcal{U}\rangle p \quad p, e \vdash [\alpha]p \quad p, \neg e \vdash \phi}{\Gamma \vdash \langle\mathcal{U}\rangle[\mathtt{while}(e)\,\alpha]\phi}$$

$$\frac{A \vdash B}{\exists x\, A \vdash \exists x\, B}$$

$$\frac{A \vdash B}{\langle\alpha\rangle A \vdash \langle\alpha\rangle B}$$

$$\langle\mathcal{U}\rangle f(u) \;\rightsquigarrow\; \mathit{if}\; s_{i_r} \doteq \langle\mathcal{U}\rangle u\; \mathit{then}\; t_{i_r}\; \mathit{else}\; \ldots\; \mathit{if}\; s_{i_1} \doteq \langle\mathcal{U}\rangle u\; \mathit{then}\; t_{i_1}\; \mathit{else}\; f(\langle\mathcal{U}\rangle u)\; \mathit{fi} \ldots \mathit{fi}$$

$$\langle\tilde{\mathcal{U}}\rangle\langle\mathcal{U}\rangle\phi \;\rightsquigarrow\; \langle\!\![\tilde{\mathcal{U}}, f_1(\langle\tilde{\mathcal{U}}\rangle s_1) := \langle\tilde{\mathcal{U}}\rangle t_1, \ldots, f_n(\langle\tilde{\mathcal{U}}\rangle s_n) := \langle\tilde{\mathcal{U}}\rangle t_n]\!\!\rangle\phi$$

Verification Example

```
class E { static int g; int id;
    E generate() {E r=new E(); r.id=g;g=g+5; return r;}}
```

$$\forall x{:}\mathrm{E}\,(x.\mathrm{id} < g \to [\mathrm{generate}]\,(x.\mathrm{id} < r.\mathrm{id}))$$

Verification Example

```
class E { static int g; int id;
    E generate() {E r=new E(); r.id=g;g=g+5; return r;}}
```

$$\forall x{:}\mathrm{E}\,(x.\mathrm{id} < g \to [\mathsf{generate}]\,(x.\mathrm{id} < r.\mathrm{id}))$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{*}{X.\mathrm{id} < g, \neg o(n) \doteq X \vdash X.\mathrm{id} < g} \qquad \dfrac{\dots}{X.\mathrm{id} < g, o(n) \doteq X \vdash g < g}}{X.\mathrm{id} < g \vdash (\neg o(n) \doteq X \to X.\mathrm{id} < g) \land (o(n) \doteq X \to g < g)}}{X.\mathrm{id} < g \vdash (\mathit{if}\, o(n) \doteq X\ \mathit{then}\, g\ \mathit{else}\, X.\mathrm{id}\ \mathit{fi}) < g}}{X.\mathrm{id} < g \vdash \langle r := o(n), n := n{+}1, o(n).\mathrm{id} := g, g := g + 5\rangle\ (X.\mathrm{id} < r.\mathrm{id})}}{X.\mathrm{id} < g \vdash \langle r := o(n), n := n{+}1, o(n).\mathrm{id} := g\rangle[g := g + 5]\ (X.\mathrm{id} < r.\mathrm{id})}}{X.\mathrm{id} < g \vdash \langle r := o(n), n := n{+}1\rangle[r.\mathrm{id} := g][g := g + 5]\ (X.\mathrm{id} < r.\mathrm{id})}}{X.\mathrm{id} < g \vdash \langle r := o(n), n := n{+}1\rangle[r.\mathrm{id} := g; g := g + 5]\ (X.\mathrm{id} < r.\mathrm{id})}}{X.\mathrm{id} < g \vdash [\alpha]\ (X.\mathrm{id} < r.\mathrm{id})}}{\vdash X.\mathrm{id} < g \to [\alpha]\ (X.\mathrm{id} < r.\mathrm{id})}}{\vdash \forall x{:}\mathrm{E}\,(x.\mathrm{id} < g \to [\alpha]\ (x.\mathrm{id} < r.\mathrm{id}))}$$

Soundness & Completeness

Theorem (Soundness)

ODL calculus is sound:

$$\vdash \phi \quad \textit{implies} \quad \vDash \phi$$

Soundness & Completeness

Theorem (Soundness)

ODL calculus is sound:

$$\vdash \phi \quad \textit{implies} \quad \vDash \phi$$

Theorem (Incompleteness)

ODL calculus is non-axiomatizable, i.e., there is no sound and complete effective calculus.

Soundness & Completeness

Theorem (Soundness)

ODL calculus is sound:

$$\vdash \phi \quad \textit{implies} \quad \vDash \phi$$

Theorem (Incompleteness)

ODL calculus is non-axiomatizable, i.e., there is no sound and complete effective calculus.

Theorem (Relative completeness)

ODL calculus is complete w.r.t. first-order arithmetic: ▸ Proof Outline

$$\vDash \phi \quad \textit{implies} \quad Taut_{Arith} \vdash \phi$$

Outline

Conclusions

- ODL is essentials-only object-oriented dynamic logic:
  1. object type system
  2. object enumerators
  3. flexible functions
- Natural translation JAVA $\rightsquigarrow$ ODL.
- Calculus proven sound & relatively complete w.r.t. arithmetic.

B. Beckert and A. Platzer.
Dynamic logic with non-rigid functions: A basis for object-oriented program verification.
In U. Furbach and N. Shankar, editors, *IJCAR*, volume 4130 of *LNCS*, pages 266–280. Springer, 2006.

Outline

6. Completeness Proof
   - Misc

Relative Completeness Proof

$$\vDash \phi \quad \text{implies} \quad \text{Taut}_{\mathbf{N}} \vdash \phi$$

Proof.

1. propositionally complete
2. first-order complete
3. first-order expressible: $\forall \phi \; \exists F \in \mathrm{FOL} \quad \vDash \phi \leftrightarrow F$
4. term rewrites are Noetherian
5. (rel.) complete for first-order $F \rightarrow \langle \alpha \rangle G$
6. (rel.) complete for first-order dynamic typing

□

◂ Return

Use Cases in JAVA CARD DL

- Use ODL object enumerators for object creation.
- Use quicker and rel. complete ODL within KeY in automatic verification scenarios. (Trafo combined with compiler construction technology)
- Import simpler ODL calculus into JAVA CARD DL, for formulas that are "sufficiently" ODL.