An Object-Oriented Dynamic Logic with Updates

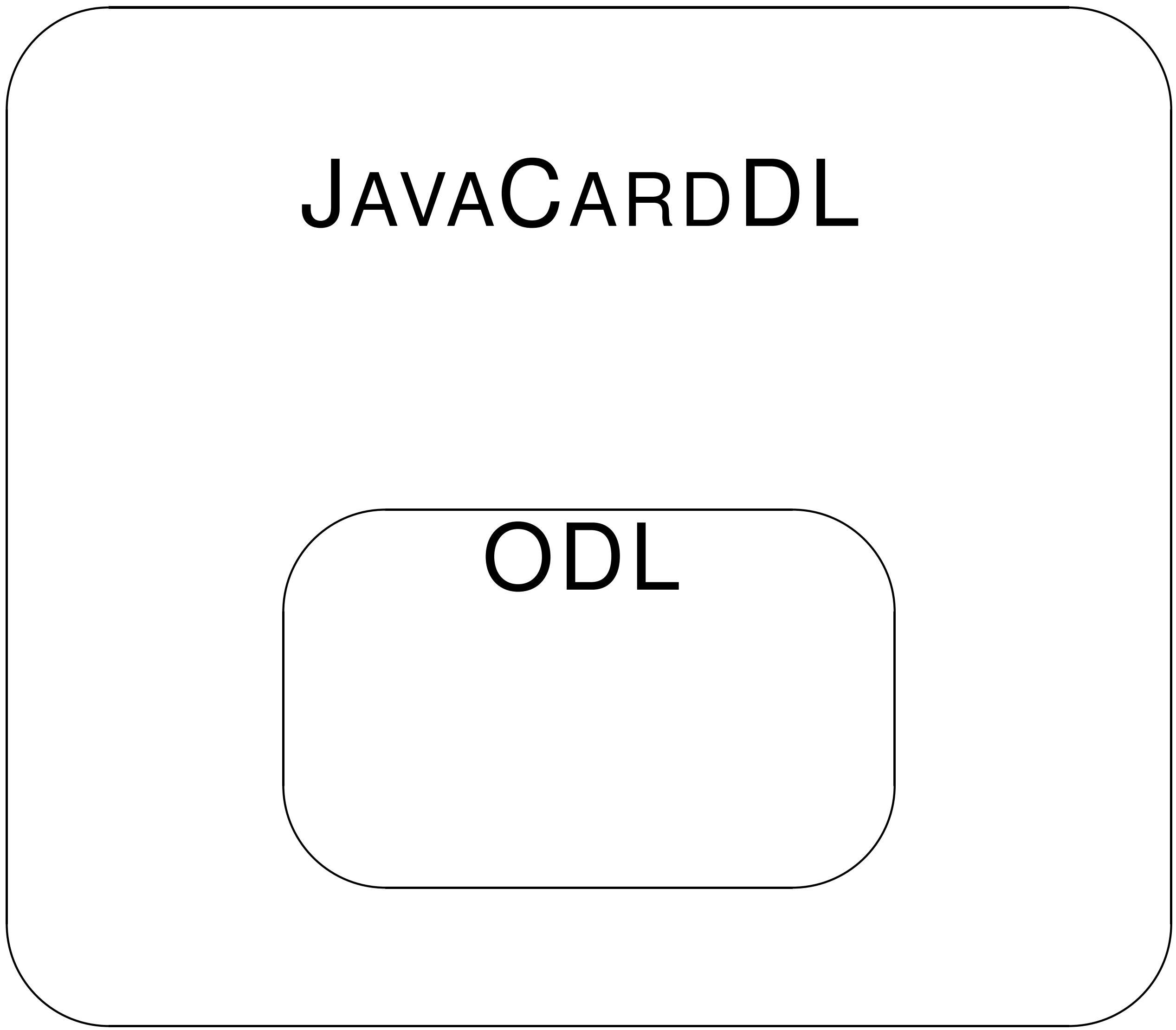
André Platzer

University of Karlsruhe

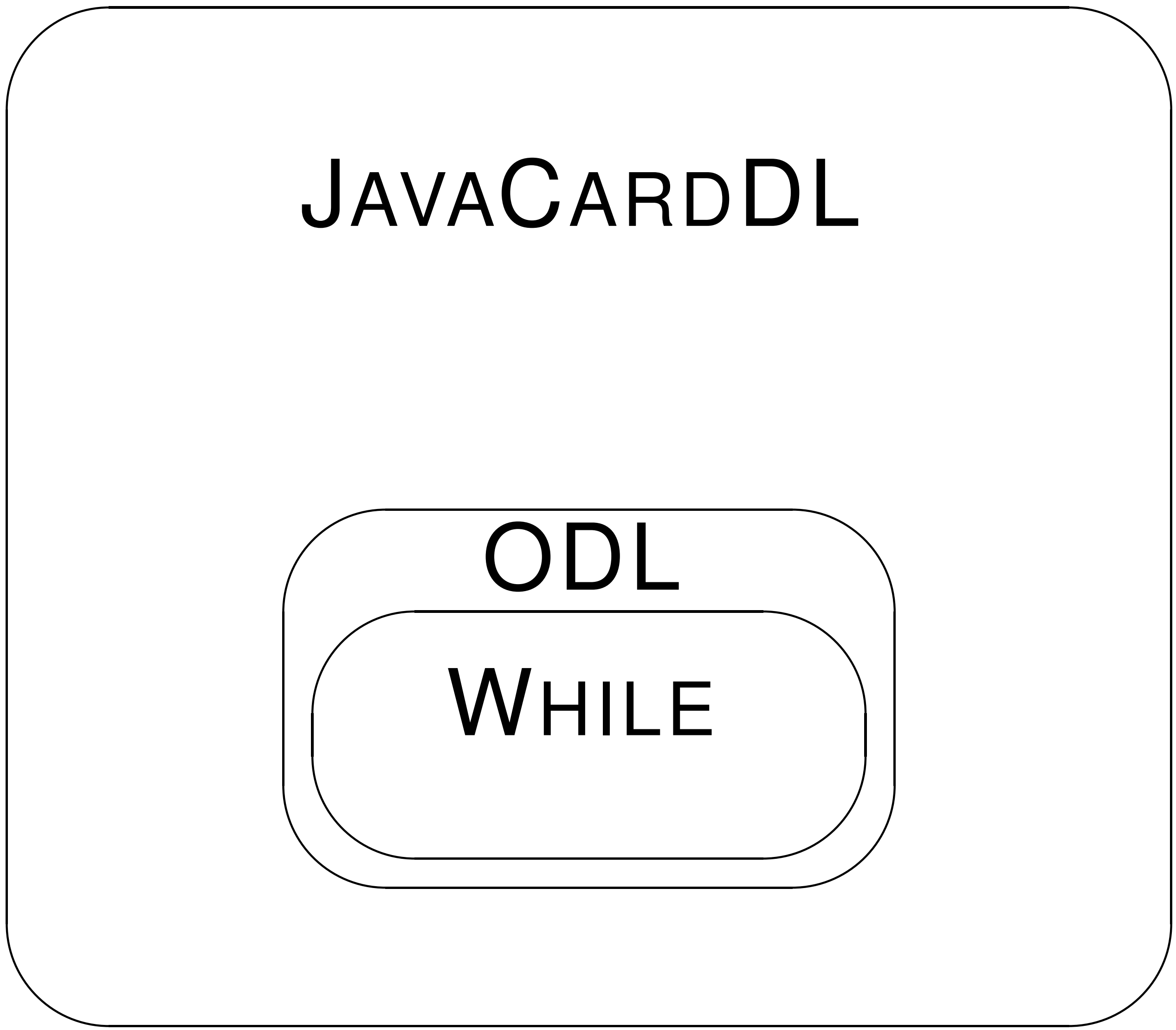
Motivation

JavaCardDL

Motivation



Motivation



Overview

- What's an Object-Oriented Dynamic Logic (ODL)
- Objective
- The language ODL
- JAVA $\rightsquigarrow$ ODL
- Calculus
- Summary

Object-Oriented DL

- ODL is a dynamic logic.
- “Natural” representation of OOP.
- ODL only contains essentials of OO.

Objective

- Characterise logical essentials of OO.
- Simple proofs *within* calculus and *about* calculus.
- Prove sound & rel. complete.
- Theoretical foundation of KeY and updates. KeY completeness?

The Language ODL

- Type-lattice with integers $\mathbb{Z}$
- Formulas ϕ
  - $\neg, \wedge, \vee, \rightarrow, \leftrightarrow, \exists\,, \forall\,, \doteq$
  - $[\alpha]\phi, \quad \langle\alpha\rangle\phi$
  - $\mathit{if}\ \phi\ \mathit{then}\ s\ \mathit{else}\ t\ \mathit{fi}, \quad t\ \mathrm{instanceof}\ \mathrm{C}$
- Programs α
  - $f(t) \coloneqq s$ (also simultaneous)
  - $\mathrm{if}(\phi)\,\{\alpha\}\,\mathrm{else}\{\gamma\}, \quad \mathrm{while}(\phi)\,\{\alpha\}, \quad \alpha; \gamma$

JAVA $\rightsquigarrow$ ODL

Software-Engineering features to ignore

- Coupling of state and behaviour
- Encapsulation
- Information hiding & visibility

JAVA $\rightsquigarrow$ ODL

Non-essentials to discard

- Inner classes
- Field overriding
- Associations
- Events
- Side-effects & evaluation order
- Exceptions

Simple translation $\Rightarrow$ syntactic sugar

JAVA $\rightsquigarrow$ ODL

Essentials to dispose

- Implementation inheritance
- Object creation
- Dynamic dispatch & polymorphism

Simple translation $\Rightarrow$ syntactic sugar

JAVA $\approx$ ODL

Features to keep

- Field access (functions)
- Subtyping ($\neq$ inheritance)

$\approx$ "object = state + behaviour"
especially

- ▶ Modifiable state
- ▶ Dynamic types

JAVA $\leadsto$ ODL (create)

Object creation has to support

- Dynamic type checks
- Object identity "$\mathrm{new} \neq \mathrm{new}$"
- Extension for varying domain

JAVA $\rightsquigarrow$ ODL (create)

Object creation has to support

- Dynamic type checks
- Object identity "$\mathrm{new} \neq \mathrm{new}$"
- Extension for varying domain
- Ex:

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

Calculus: object creation

- Dynamic type checks

$$\mathrm{obj}_{\mathtt{A}}(n)\,\mathrm{instanceof}\,\mathtt{C} = \begin{cases} \mathit{true} & \Leftarrow \mathtt{A} \text{ subty. of } \mathtt{C} \\ \mathit{false} & \text{otherwise} \end{cases}$$

Calculus: object creation

- Dynamic type checks

$$\mathrm{obj}_{\mathtt{A}}(n)\ \mathrm{instanceof}\ \mathtt{C} = \begin{cases} \mathit{true} & \Leftarrow \mathtt{A}\ \text{subty. of}\ \mathtt{C} \\ \mathit{false} & \text{otherwise} \end{cases}$$

- Ex:

$$\begin{aligned} \mathrm{obj}_{\mathtt{Car}}(n)\ \mathrm{instanceof}\ \mathtt{Vehicle} &= \mathit{true} \\ \mathrm{obj}_{\mathtt{Vehicle}}(k)\ \mathrm{instanceof}\ \mathtt{Car} &= \mathit{false} \\ f(a)\ \mathrm{instanceof}\ \mathtt{Car} &= \ ? \end{aligned}$$

Calculus: object creation

- Object identity "$\mathrm{new} \neq \mathrm{new}$"

$$\frac{}{\Gamma, i \neq j \ \vdash \Delta, \mathrm{obj}_{\mathrm{C}}(i) \neq \mathrm{obj}_{\mathrm{C}}(j)}$$

Calculus: object creation

- Object identity "$\text{new} \neq \text{new}$"

$$\overline{\Gamma, i \neq j \ \vdash \Delta, \mathrm{obj}_{\mathtt{C}}(i) \neq \mathrm{obj}_{\mathtt{C}}(j)}$$

- Ex:

$\mathbf{x} \mathrel{:=} \mathrm{new}\ \mathtt{C}();$	$\rightsquigarrow$	$\mathbf{x} \mathrel{:=} \mathrm{obj}_{\mathtt{C}}(1);$
$\mathbf{y} \mathrel{:=} \mathrm{new}\ \mathtt{C}();$	$\rightsquigarrow$	$\mathbf{y} \mathrel{:=} \mathrm{obj}_{\mathtt{C}}(2);$
$\mathrm{if}(\mathbf{x} \doteq \mathbf{y})\ \{\alpha\}\ \mathrm{else}\{\gamma\}$	$\overset{1\neq 2}{\rightsquigarrow}$	γ

Calculus: object creation

- Extension for varying domain

$$\forall n \left(n < next_{\mathrm{C}} \rightarrow \phi(\mathrm{obj}_{\mathrm{C}}(n))\right)$$

"All objects created so far satisfy ϕ"

Example: updates

$$\langle \mathtt{f}(\mathtt{s}) \coloneqq \mathtt{t} \rangle g(f(s))$$

$$\rightsquigarrow \quad g\big(\langle \mathtt{f}(\mathtt{s}) \coloneqq \mathtt{t} \rangle f(s)\big)$$

$$\text{“} \rightsquigarrow \text{”} \quad g(t)$$

Example: updates (alias)

$$\langle \mathtt{f}(\mathtt{s}) \doteq \mathtt{t} \rangle g(f(r))$$

$$\leadsto \quad g\big(\langle \mathtt{f}(\mathtt{s}) \doteq \mathtt{t} \rangle f(r)\big)$$

$$\leadsto \quad g\big(\mathit{if}\ s \doteq r\ \mathit{then}\ t\ \mathit{else}\ f(r)\ \mathit{fi}\big)$$

$$\text{“} \leadsto \text{”} \quad \big(s \doteq r \rightarrow\ g(t)\big) \land$$

$$\big(s \neq r \rightarrow\ g(f(r))\big)$$

Calculus: updates

- update (match)

$$\langle \mathtt{f}(\mathtt{s}) \coloneqq \mathtt{t} \rangle f(u) \rightsquigarrow$$

$$\mathit{if}\ s \doteq \langle \mathtt{f}(\mathtt{s}) \coloneqq \mathtt{t} \rangle u\ \mathit{then}\ t\ \mathit{else}\ f\big(\langle \mathtt{f}(\mathtt{s}) \coloneqq \mathtt{t} \rangle u\big)\ \mathit{fi}$$

- conditional term split

$$\frac{\Gamma \vdash \Delta, (e \to \phi(s)) \wedge (\neg e \to \phi(t))}{\Gamma \vdash \Delta, \phi(\mathit{if}\ e\ \mathit{then}\ s\ \mathit{else}\ t\ \mathit{fi})}$$

admissible

Relative Completeness

- Arithmetic is incomplete.
- How much worse is ODL calculus?
- Relatively complete:
  $\approx$ “in addition to domain of computation, program verification calculus ODL is complete”

Relative Completeness

- Arithmetic is incomplete.
- How much worse is ODL calculus?
- Relatively complete:
  $\approx$ “in addition to domain of computation, program verification calculus ODL is complete”

$\Rightarrow$ KeY is relatively complete, “suitable” JAVA transformations provided.

Summary

- ODL is an object-oriented dynamic logic.
- ODL only contains essentials of OO.
- “Natural” translation JAVA $\rightsquigarrow$ ODL .
- Updates for object aliasing.
- Calculus is sound.

▶ Plan: prove calculus rel. complete.

Repository

- The end of the presentation

Terminology: Admissible

- $[s \mapsto t]$ is *admissible* for $\phi \;:\Longleftrightarrow\; s, t$ do not trespass modalities for which they are not rigid during the formation of $\phi[s \mapsto t]$.
- *wary substitution* $[\widehat{s \mapsto t}]$ works like $[s \mapsto t]$ but quits in front of modalities for which s or t are not rigid.

JAVA $\rightsquigarrow$ ODL (throw) (I)

```
try {
  while (x >= y) {
    x = x - y;
    if (no progress) {
      throw new DivByZero(x, y);
    }
    z = z + 1;
  }
} catch (DivByZero r) {h}
```

JAVA $\rightsquigarrow$ ODL (throw) (II)

```
while (e == null && x >= y) {
      x = x − y;
       if (no progress) {
            e = new DivByZero(x, y);
       }
      if (e == null) {z = z + 1;}
}
if (e instanceof DivByZero) {h
} else {...}
```

JAVA $\rightsquigarrow$ ODL (dispatch)

- C extends B
- C and B provide m(String arg)
- Transformation of x.m(arg):

```
if (x instanceof C) {
      ((C)x).m(arg);
} else if (x instanceof B) {
      ((B)x).m(arg);
}
```

Example: updates (alias)

$$\langle \mathtt{f}(\mathtt{s}) \mathrel{:=} \mathtt{t} \rangle g(f(r))$$

$$\leadsto \quad g\Big(\langle \mathtt{f}(\mathtt{s}) \mathrel{:=} \mathtt{t} \rangle f(r)\Big)$$

$$\leadsto \quad g\Big(\mathit{if}\ s \doteq r \qquad\qquad \mathit{then}\ t\ \mathit{else}\ f(\langle \mathtt{f}(\mathtt{s}) \mathrel{:=} \mathtt{t} \rangle r)\ \mathit{fi}\Big)$$

Example: updates (alias)

$$\langle \mathtt{f}(\mathtt{s}) \coloneqq \mathtt{t} \rangle g(f(r))$$

$$\leadsto \quad g\big(\langle \mathtt{f}(\mathtt{s}) \coloneqq \mathtt{t} \rangle f(r)\big)$$

$$\leadsto \quad g\big(\mathit{if}\ s \doteq \langle \mathtt{f}(\mathtt{s}) \coloneqq \mathtt{t} \rangle r\ \mathit{then}\ t\ \mathit{else}\ f(\langle \mathtt{f}(\mathtt{s}) \coloneqq \mathtt{t} \rangle r)\ \mathit{fi}\big)$$

$$\leadsto \quad g\big(\mathit{if}\ s \doteq r\ \mathit{then}\ t\ \mathit{else}\ f(r)\ \mathit{fi}\big)$$

$$\text{“}\leadsto\text{”} \quad \big(s \doteq r \rightarrow\ g(t)\big) \land \big(s \neq r \rightarrow\ g(f(r))\big)$$

Example: updates

$$\langle \mathtt{f}(\mathtt{s}) \mathrel{:=} \mathtt{s} \rangle g(f(f(r)))$$

$$\rightsquigarrow \quad g\big(\langle \mathtt{f}(\mathtt{s}) \mathrel{:=} \mathtt{s} \rangle f(f(r))\big)$$

$$\rightsquigarrow \quad g\Big(\mathit{if}\ s \doteq \langle \mathtt{f}(\mathtt{s}) \mathrel{:=} \mathtt{s} \rangle f(r)\ \mathit{then}\ s\ \mathit{else}\ f\big(\langle \mathtt{f}(\mathtt{s}) \mathrel{:=} \mathtt{s} \rangle f(r)\big)\ \mathit{fi}\Big)$$

$$\rightsquigarrow \quad g\Big(\mathit{if}\ s \doteq \big(\mathit{if}\ s \doteq \langle \mathtt{f}(\mathtt{s}) \mathrel{:=} \mathtt{s} \rangle r\ \mathit{then}\ s\ \mathit{else}\ f(\langle \mathtt{f}(\mathtt{s}) \mathrel{:=} \mathtt{s} \rangle r)\ \mathit{fi}\big)\ \mathit{then}$$

$$s$$

$$\mathit{else}$$

$$f\big(\mathit{if}\ s \doteq \langle \mathtt{f}(\mathtt{s}) \mathrel{:=} \mathtt{s} \rangle r\ \mathit{then}\ s\ \mathit{else}\ f(\langle \mathtt{f}(\mathtt{s}) \mathrel{:=} \mathtt{s} \rangle r)\ \mathit{fi}\big)$$

$$\mathit{fi}\Big)$$

Example: updates

$$\leadsto \quad g\Big(\mathit{if}\ s \doteq \big(\mathit{if}\ s \doteq r\ \mathit{then}\ s\ \mathit{else}\ f(r)\ \mathit{fi}\big)\ \mathit{then}$$

$$s$$

$$\mathit{else}$$

$$f\big(\mathit{if}\ s \doteq r\ \mathit{then}\ s\ \mathit{else}\ f(r)\ \mathit{fi}\big)$$

$$\mathit{fi}\Big)$$

$$\text{“}\leadsto\text{”} \quad \Big(s \doteq r \rightarrow\ g\big(\mathit{if}\ s \doteq s\ \mathit{then}\ s\ \mathit{else}\ f(s)\ \mathit{fi}\big)\Big) \land$$

$$\Big(s \neq r \rightarrow\ g\big(\mathit{if}\ s \doteq f(r)\ \mathit{then}\ s\ \mathit{else}\ f(f(r))\ \mathit{fi}\big)\Big)$$

$$\text{“}\leadsto\text{”} \quad \Big(s \doteq r \rightarrow\ g(s)\Big) \land$$

$$\Big(s \neq r \rightarrow\ g\big(\mathit{if}\ s \doteq f(r)\ \mathit{then}\ s\ \mathit{else}\ f(f(r))\ \mathit{fi}\big)\Big)$$

Example: updates (quick)

$$\langle \mathtt{f}(\mathtt{s}) \coloneqq \mathtt{s} \rangle g(f(f(s)))$$

$$\rightsquigarrow \quad g\big(\langle \mathtt{f}(\mathtt{s}) \coloneqq \mathtt{s} \rangle f(f(s))\big)$$

$$\rightsquigarrow \quad g\Big(\mathit{if}\ s \doteq \langle \mathtt{f}(\mathtt{s}) \coloneqq \mathtt{s} \rangle f(s)\ \mathit{then}\ s\ \mathit{else}\ f\big(\langle \mathtt{f}(\mathtt{s}) \coloneqq \mathtt{s} \rangle f(s)\big)\ \mathit{fi}\Big)$$

$$\rightsquigarrow \quad g\Big(\mathit{if}\ s \doteq \big(\mathit{if}\ s \doteq \langle \mathtt{f}(\mathtt{s}) \coloneqq \mathtt{s} \rangle s\ \mathit{then}\ s\ \mathit{else}\ f(\langle \mathtt{f}(\mathtt{s}) \coloneqq \mathtt{s} \rangle s)\ \mathit{fi}\big)\ \mathit{then}$$

$$s$$

$$\mathit{else}$$

$$f\big(\mathit{if}\ s \doteq \langle \mathtt{f}(\mathtt{s}) \coloneqq \mathtt{s} \rangle s\ \mathit{then}\ s\ \mathit{else}\ f(\langle \mathtt{f}(\mathtt{s}) \coloneqq \mathtt{s} \rangle s)\ \mathit{fi}\big)$$

$$\mathit{fi}\Big)$$

Example: updates (quick)

$$g\Big(\mathit{if}\ s \doteq (\mathit{if}\ s \doteq \langle \mathtt{f}(\mathtt{s}) {:=} \mathtt{s} \rangle s\ \mathit{then}\ s\ \mathit{else}\ f(\langle \mathtt{f}(\mathtt{s}) {:=} \mathtt{s} \rangle s)\ \mathit{fi})\ \mathit{then}$$

$$s$$

$$\mathit{else}$$

$$f\big(\mathit{if}\ s \doteq \langle \mathtt{f}(\mathtt{s}) {:=} \mathtt{s} \rangle s\ \mathit{then}\ s\ \mathit{else}\ f(\langle \mathtt{f}(\mathtt{s}) {:=} \mathtt{s} \rangle s)\ \mathit{fi}\big)$$

$$\mathit{fi}\Big)$$

$$\leadsto \quad g\Big(\mathit{if}\ s \doteq s\ \mathit{then}\ s\ \mathit{else}\ f(s)\ \mathit{fi}\Big)$$

$$\text{“}\leadsto\text{”} \quad g(s)$$

Calculus: update promotion

- update (match)

$$\langle \mathtt{f}(\mathtt{s}) \mathrel{\dot{=}} \mathtt{t} \rangle f(u) \;\rightsquigarrow$$

$$\mathit{if}\ s \doteq \langle \mathtt{f}(\mathtt{s}) \mathrel{\dot{=}} \mathtt{t} \rangle u\ \mathit{then}\ t\ \mathit{else}\ f\big(\langle \mathtt{f}(\mathtt{s}) \mathrel{\dot{=}} \mathtt{t} \rangle u\big)\ \mathit{fi}$$

- update (promote)

$$\langle \mathtt{f}(\mathtt{s}) \mathrel{\dot{=}} \mathtt{t} \rangle \Upsilon(u) \;\rightsquigarrow\; \Upsilon\big(\langle \mathtt{f}(\mathtt{s}) \mathrel{\dot{=}} \mathtt{t} \rangle u\big)$$

$$\Leftarrow f \neq \Upsilon$$

Calculus: merge (last-win)

- update merge

$$\langle\mathcal{U}\rangle\langle \mathtt{f}(\mathtt{s}) \doteq \mathtt{t}\rangle\phi \;\rightsquigarrow$$

$$\langle\mathcal{U}, \mathtt{f}(\langle\mathcal{U}\rangle\mathtt{s}) \doteq \langle\mathcal{U}\rangle\mathtt{t}\rangle$$

Calculus parallel updates

- Simultaneous parallel update

$$\underbrace{f_1(s_1) \mathrel{:=} t_1, \ldots, f_n(s_n) \mathrel{:=} t_n}_{\mathcal{U}}$$

- update (match)

$$\langle \mathcal{U} \rangle f(u) \rightsquigarrow$$

$$\mathit{if}\ s_{i_r} \doteq \langle \mathcal{U} \rangle u\ \mathit{then}\ t_{i_r}\ \mathit{else}\ \ldots$$

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

$$\Leftarrow \{i_1, \ldots, i_r\} = \{i : f_i = f\}$$

Relative Completeness

- Rel. Complete:

  for each $\phi \in \mathrm{Fml}(\Sigma \cup V)$

  $($ for each arithmetic struct. ℓ $\ell \vDash \phi)$ implies $\vdash$

  assuming oracle for first-order arithmetic.