A Calculus for Data Abstraction

18 May 2009

Roman Krenický

Function updates

$\{u\}\, f(t)$ – in state s ⇨ $f(t)$ – in $u(s)$

Function updates

$\{u\}\, f(t)$ – in state s ⇨ $f(t)$ – in $u(s)$

- $(\{u\}\ f\ )(\ t\ )$

Function updates

$\{u\}\, f(t)$ – in state s ⇨ $f(t)$ – in $u(s)$

- $(\{u\}\, \underbrace{f}_{u(s)})(\underbrace{t}_{s})$

Function updates

$\{u\}\, f(t)$ – in state s ⇨ $f(t)$ – in $u(s)$

- $(\{u\}\, \underbrace{f}_{u(s)})(\underbrace{t}_{s})$

- in s: $f(5) = 5,\ f(4) = 3$

$$\{f(5) := 4\}\, f(\underbrace{f(5)}_{u(s)\ [=4]}) \quad = \quad 3$$

$$(\{f(5) := 4\}\, f)(\underbrace{f(5)}_{s\ [=5]}) \quad = \quad 4$$

Update Simplifier

- term rewriting system by Philipp Rümmer

$$\{u\}\, f(t) \;\rightarrow\; (\{u\}\, f)(\{u\}t)$$

$$(\{f(s) := r\}\, f)(t) \;\rightarrow\; \text{if } s{=}t \text{ then } r \text{ else } f(t)$$

Update Simplifier

- term rewriting system by Philipp Rümmer

$$\{u\}\, f(t) \;\rightarrow\; (\{u\}\, f)(\{u\}t)$$

$$(\{f(s) := r\}\, f)(t) \;\rightarrow\; \text{if } s{=}t \text{ then } r \text{ else } f(t)$$

- all updates in terms and formulae are eliminated

Update Simplifier

- term rewriting system by Philipp Rümmer

$$\{u\}\, f(t) \;\rightarrow\; (\{u\}\, f)(\{u\}t)$$

$$(\{f(s) := r\}\, f)(t) \;\rightarrow\; \text{if } s{=}t \text{ then } r \text{ else } f(t)$$

- all updates in terms and formulae are eliminated
- soundness and completeness proven using Isabelle/HOL

New Expressions

- observer symbols
- location descriptors

  for $x.f(t)$

- *locEqual, locSubset, locDisjoint*
- anonymising updates

  $f(t) := *3$

New Constructors

- $dom(u)$ — domain of u
- $RESTRICT(u, ld) := u|_{ld}$
- $ld_1 \setminus ld_2$
- $ld_1 \cap ld_2$

A Calculus for Data Abstraction

- update simplifier for new expressions

$$(\{obs(s) := *n\}\, f)(t) \;\rightarrow\; \text{if } \mathit{locSubset}(f(t), obs(s))$$
$$\text{then } (\{\mathit{everything} := *n\}\, f)(t)$$
$$\text{else } f(t)$$

A Calculus for Data Abstraction

- update simplifier for new expressions

$$(\{obs(s) := *n\}\, f)(t) \;\rightarrow\; \text{if } \mathit{locSubset}(f(t), obs(s))$$
$$\text{then } (\{\mathit{everything} := *n\}\, f)(t)$$
$$\text{else } f(t)$$

$$\mathit{locSubset}(f(t), \text{for } x.\mathit{ld}) \;\rightarrow\; \exists\, x.\mathit{locSubset}(f(t), \mathit{ld})$$

A Calculus for Data Abstraction

- update simplifier for new expressions

$$(\{obs(s) := *n\}\, f)(t) \;\rightarrow\; \text{if } \mathit{locSubset}(f(t), obs(s))$$
$$\text{then } (\{\mathit{everything} := *n\}\, f)(t)$$
$$\text{else } f(t)$$

$$\mathit{locSubset}(f(t), \text{for } x.\mathit{ld}) \;\rightarrow\; \exists\, x.\mathit{locSubset}(f(t), \mathit{ld})$$

$$(\{f(t) := r\}\, obs)(s) \;\rightarrow\; ?$$

Remaining Expressions

- $(\{everything := *n\}\ f)(t)$ *[Term]*

Remaining Expressions

- $(\{everything := *n\}\ f)(t)$ *[Term]*
- $(\{u\}\ obs)(t)$ *[Term]*

Remaining Expressions

- $(\{everything := *n\}\ f)(t)$ *[Term]*
- $(\{u\}\ obs)(t)$ *[Term]*
- *locSubset*($f(t)$, $obs(s)$) *[For]*
- *locSubset*($obs(s)$, *ld*) *[For]*
- *locSubset*(*everything*, *ld*) *[For]*

Remaining Expressions

- $(\{everything := *n\}\ f)(t)$ *[Term]*
- $(\{u\}\ obs)(t)$ *[Term]*
- *locSubset*($f(t)$, $obs(s)$) *[For]*
- *locSubset*($obs(s)$, *ld*) *[For]*
- *locSubset(everything, ld)* *[For]*
- $(\{u\}\ obs)(t)$ *[LocDesc]*

Remaining Expressions

- $(\{everything := *n\}\ f)(t)$ *[Term]*
- $(\{u\}\ obs)(t)$ *[Term]*
- *locSubset*$(f(t), obs(s))$ *[For]*
- *locSubset*$(obs(s)$, *ld*$)$ *[For]*
- *locSubset(everything, ld)* *[For]*
- $(\{u\}\ obs)(t)$ *[LocDesc]*
- $(\{u\}\ obs)(t) := *n$ *[Upd]*

Remaining Expressions

- $(\{everything := *n\}\ f)(t)$ *[Term]*
- $(\{u\}\ obs)(t)$ *[Term]*
- *locSubset*$(f(t), (\{u\}\ obs)(s))$ *[For]*
- *locSubset*$((\{u\}\ obs)(s)$, *ld)* *[For]*
- *locSubset(everything, ld)* *[For]*
- $(\{u\}\ obs)(t)$ *[LocDesc]*
- $(\{u\}\ obs)(t) := *n$ *[Upd]*

Observer Updates

$$(\{u\}\ obs)\ (t)$$

Observer Updates

$$(\{u\}\ obs\ )\ (t)$$

$$\rightarrow$$

$$(\{\text{if } \neg \mathit{locDisjoint}(\mathit{dom}(u), \mathit{obs}(t))\ .\ u\}\ obs\ )\ (t)$$

Observer Updates

$(\{u\}\ obs\ )\ (t)$

$\rightarrow$

$(\{\text{if } \neg locDisjoint(dom(u), obs(t)) \ .\ u\}\ obs\ )\ (t)$

- reduces to
  - $obs(t)$
  - $(\{u\}\ obs)(t)$

Observer Updates

$(\{u\}\ obs\ )\ (t)$

$\rightarrow$

$(\{\text{if } \neg locDisjoint(dom(u), obs(t))\ .\ u\}\ obs\ )\ (t)$

- reduces to
  - $obs(t)$
  - $(\{u\}\ obs)(t)$
- no such rule for location descriptors
  - no framing for ld-s:
    dependencies of "$(\{u\}\ obs)(t)$" unknown

Example

axiom: $\forall$ *int[] a . locEqual(nonNullArray(a),* for *int i . a[i])*

$$\{arr2[0] := null\}\ nonNullArray(arr1)$$

Example

axiom: $\forall$ *int[] a . locEqual(nonNullArray(a),* for *int i . a[i])*

$$\{arr2[0] := null\}\ nonNullArray(arr1)$$

$$\text{—(rewrite)}\rightarrow$$

if *locSubset*$(arr2[0], nonNullArray(arr1))$
then $(\{arr2[0] := null\}nonNullArray)(arr1)$
else $nonNullArray(arr1)$

Example

axiom: $\forall$ *int[] a . locEqual(nonNullArray(a),* for *int i . a[i])*

{arr2[0] := null} nonNullArray(arr1)

—(rewrite)→

if *locSubset(arr2[0], nonNullArray(arr1))*
then *({arr2[0] := null}nonNullArray)(arr1)*
else *nonNullArray(arr1)*

—(axiom)→

if *locSubset(arr2[0],* for *int i . arr1[i])*
then *({arr2[0] := null}nonNullArray)(arr1)*
else *nonNullArray(arr1)*

Example

axiom: $\forall$ *int[] a . locEqual(nonNullArray(a),* for *int i . a[i])*

{arr2[0] := null} nonNullArray(arr1)

—(rewrite)→

if *locSubset(arr2[0], nonNullArray(arr1))*
then *({arr2[0] := null}nonNullArray)(arr1)*
else *nonNullArray(arr1)*

—(axiom)→

if *locSubset(arr2[0],* for *int i . arr1[i])*
then *({arr2[0] := null}nonNullArray)(arr1)*
else *nonNullArray(arr1)*

—(rewrite + axiom: $arr1 \neq arr2$)→

nonNullArray(arr1)

Properties

- correctness: proven manually
- not reducible to FOL any more
- termination: unproven
- confluence: …

Confluence

- $(\text{for } x.\mathit{ld}) \setminus (\text{for } x.\mathit{ld}) \;\rightarrow\; \mathit{empty}$

  $(\text{for } x.\mathit{ld}) \setminus (\text{for } x.\mathit{ld}) \;\rightarrow\; \text{for } x.(\mathit{ld} \setminus \text{for } x.\mathit{ld})$

- $\{\text{if } \varphi \,.\, \mathit{skip}\}\, f \;\rightarrow\; f$

  $\{\text{if } \varphi \,.\, \mathit{skip}\}\, f \;\rightarrow\; \text{if } \varphi \text{ then } f \text{ else } f$

Equivalence of Updates

- old system (no observers):

$$u_1 \equiv_{\mathsf{md}} u_2 \;\Leftrightarrow\; \bigwedge_{f \in u_1 \cup u_2} \forall\, y \,.\, (\{u_1\}\, f(y) = \{u_2\}\, f(y))$$

- new system (with observers):

$$(f(t) := *3) \equiv (obs(s) := *3) \;\Leftrightarrow\; locEqual(f(t), obs(s))$$

…

Summary

- ~180 rules (P.R.: ~80)
- reduction to logic + set theory (location descriptors)
- sometimes remaining updates in expressions with observers