A Logic for Secure Memory Access of ASMs

Stanislas Nanchen
Robert F. Stärk

Institute of Theoretical Computer Science, ETH Zurich

Overview

- Motivations
- Access and access sets
- A logic for secure memory access of ASMs
- Implementation in ASMKeY
- Conclusion

Abstract State Machines (ASMs)

A formal method for specification and verification

- specification of hardware and software
- specification of algorithms
- semantics of programming languages

ASM Archive: http://www.eecs.umich.edu/gasm/

Gurevich's thesis:

> *For each algorithm, one can build an ASM so that one step in the algorithm corresponds to one step on the ASM*

ASMs in a Nutshell

- State = abstract algebraic structure.
- Dynamic functions vs static functions.
- Semantic via update sets (firing a rule).
- Transition rules :

if *Condition* **then**

$$\left.\begin{array}{lcl} f_1(s_1) & := & t_1 \\ f_2(s_2) & := & t_2 \\ \vdots & \vdots & \vdots \\ f_n(s_n) & := & t_n \end{array}\right\} \text{parallel}$$

ASM rules

Name	Syntax	Semantics
Skip Rule	**skip**	[do nothing]
Update Rule	$f(t) := s$	[update f at argument t to s]
Block Rule	R **par** S	[parallel execution]
Conditional	**if** φ **then** R **else** S	[conditional execution]
Let Rule	**let** $x = t$ **in** R	[call by value]
Forall Rule	**forall** x **with** φ **do** R	[parallel for each x satisfying φ]
Sequence	R **seq** S	[sequential execution]
Try Rule	**try** R **else** S	[if R is inconsistent, then S]
Call Rule	$\rho(t)$	[call ρ with parameter t]

Motivations: Security

- In the world of ASMs
  - The states are global states
  - Focus on the updates of ASMs

- Access are equally important
  - Applet isolation on smart cards
  - Secure information flow

- Some properties
  - 'The program P does not read the location $f(x)$'
  - 'Whenever program P reads location $f(x)$, then $0 \leq x < 10$'

Motivations: Sequentialization

- Sequentialization

$$P \textbf{ par } Q \quad \rightsquigarrow \quad P \textbf{ seq } Q$$

- Q does not override locations updated by P

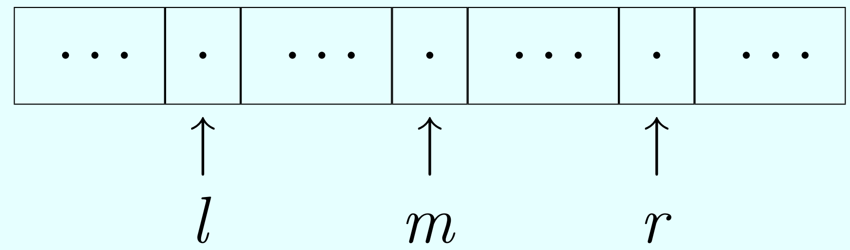
$$\mathsf{not\_over}(P, Q) \triangleq$$
$$\bigwedge_{f \text{ dyn.}} \forall x, y \, (\mathsf{upd}(P, f, x, y) \rightarrow \mathsf{inv}(Q, f, x) \vee \mathsf{upd}(Q, f, x, y))$$

- Q does not read locations updated by P

$$\mathsf{not\_read}(P, Q) \triangleq ???$$

Motivations: Avoiding Exceptions

Use a unary dynamic function f:



$\textsc{MergeSort}(l, r) =$
  if $l < r$ **then**
    let $m = \lfloor (l + r)/2 \rfloor$ **in**
      $(\textsc{MergeSort}(l, m)$ **par** $\textsc{MergeSort}(m + 1, r))$ **seq**
      $\textsc{Merge}(l, m, r)$

$\textsc{Merge}(l, m, r) =$
  (**forall** i **with** $l \leq i \leq r$ **do** $g(i) := f(i)$) **seq**
  $\textsc{MergeCopy}(l, m, m + 1, r, l)$

Motivations: Avoiding Exceptions (cont.)

$\textsc{MergeCopy}(i, m, j, r, k) =$
if $k \le r$ **then**
if $(i \le m \wedge j \le r \wedge g(i) \le g(j)) \vee (r < j)$ **then**
$f(k) := g(i)$ **par** $\textsc{MergeCopy}(i+1, m, j, r, k+1)$
else
$f(k) := g(j)$ **par** $\textsc{MergeCopy}(i, m, j+1, r, k+1)$

'In the absence of short-circuit boolean operators, if the greatest element is in the first half of the array, then $\textsc{MergeCopy}$ makes an access at $g(r+1)$.'

ASMs

- ASMs

  Deterministic Turbo ASMs (with recursive rule definitions) with **forall** transition rules

- Rule guards

  New boolean connectives in rule guards : short-circuit && and ||

- Logical equivalent of rule guards

$$\varphi \dashrightarrow \left\{ \begin{array}{ccc} \&\& & \rightsquigarrow & \wedge \\ || & \rightsquigarrow & \vee \end{array} \right\} \dashrightarrow \hat{\varphi}$$

Access Sets and Update Sets

- Update and access (for a dynamic function f)
  - Udpate: $\langle f, a, b\rangle$ with $a, b \in |\mathfrak{A}|$
  - Access: $\langle f, a\rangle$ with $a \in |\mathfrak{A}|$
- Update sets and access sets
  - Update sets yielded by transition rules;
  - Access sets yielded by terms, formulas and transition rules.
- Consistency
  - Update sets:
    $$\bigwedge_{f \text{ dyn.}} \forall x, y, z(\langle f, x, y\rangle \in U \wedge \langle f, x, z\rangle \in U \rightarrow y = z)$$
  - Access sets: no consistency notion needed.

Access Sets for Terms and Formulas

$$\mathsf{AccSet}(x, \mathfrak{A}, \zeta) = \emptyset$$

$$\mathsf{AccSet}(f(t), \mathfrak{A}, \zeta) = \begin{cases} \{\langle f, [\![t]\!]^{\mathfrak{A}}_{\zeta}\rangle\} \cup \mathsf{AccSet}(t, \mathfrak{A}, \zeta), \\ \text{if } f \text{ is dynamic;} \\ \mathsf{AccSet}(t, \mathfrak{A}, \zeta), \\ \text{otherwise.} \end{cases}$$

$$\mathsf{AccSet}(s = t, \mathfrak{A}, \zeta) = \mathsf{AccSet}(s, \mathfrak{A}, \zeta) \cup \mathsf{AccSet}(t, \mathfrak{A}, \zeta)$$

$$\mathsf{AccSet}(\neg\varphi, \mathfrak{A}, \zeta) = \mathsf{AccSet}(\varphi, \mathfrak{A}, \zeta)$$

$$\mathsf{AccSet}(\varphi \wedge \psi, \mathfrak{A}, \zeta) = \mathsf{AccSet}(\varphi, \mathfrak{A}, \zeta) \cup \mathsf{AccSet}(\psi, \mathfrak{A}, \zeta)$$

$$\mathsf{AccSet}(\varphi \ \&\& \ \psi, \mathfrak{A}, \zeta) = \begin{cases} \mathsf{AccSet}(\varphi, \mathfrak{A}, \zeta) \cup \mathsf{AccSet}(\psi, \mathfrak{A}, \zeta), \\ \text{if } [\![\hat{\varphi}]\!]^{\mathfrak{A}}_{\zeta} = \mathit{true}; \\ \mathsf{AccSet}(\varphi, \mathfrak{A}, \zeta), \\ \text{otherwise.} \end{cases}$$

Access Sets and Update Sets (cont.)

$$\frac{}{\mathsf{yields}(\mathbf{skip}, \mathfrak{A}, \zeta, \emptyset, \emptyset)}$$

$$\frac{}{\mathsf{yields}(f(s) := t, \mathfrak{A}, \zeta, \{\langle f, [\![s]\!]^{\mathfrak{A}}_{\zeta}, [\![t]\!]^{\mathfrak{A}}_{\zeta}\rangle\}, A)}$$

where $A = \mathsf{AccSet}(s, \mathfrak{A}, \zeta) \cup \mathsf{AccSet}(t, \mathfrak{A}, \zeta)$

$$\frac{\mathsf{yields}(P, \mathfrak{A}, \zeta, U, A) \quad \mathsf{yields}(Q, \mathfrak{A}, \zeta, V, B)}{\mathsf{yields}(P \ \mathbf{par} \ Q, \mathfrak{A}, \zeta, U \cup V, A \cup B)}$$

$$\frac{\mathsf{yields}(P, \mathfrak{A}, \zeta, U, A) \quad \mathsf{yields}(Q, \mathfrak{A} + U, \zeta, V, B)}{\mathsf{yields}(P \ \mathbf{seq} \ Q, \mathfrak{A}, \zeta, U \oplus V, A \cup B)} \text{ if } U \text{ is consistent}$$

$$\frac{\mathsf{yields}(P, \mathfrak{A}, \zeta, U, A)}{\mathsf{yields}(P \ \mathbf{seq} \ Q, \mathfrak{A}, \zeta, U, A)} \quad \text{if } U \text{ is inconsistent}$$

Basic Logic (Reminder)

Modal formulas and basic predicates

$$[\![\,[P]\varphi\,]\!]^{\mathfrak{A}}_{\zeta} := \begin{cases} \mathit{true}, & \text{if } [\![\varphi]\!]^{\mathfrak{A}+U}_{\zeta} = \mathit{true} \text{ for each consist. } U \text{ with} \\ & \text{yields}(P, \mathfrak{A}, \zeta, U, A); \\ \mathit{false}, & \text{otherwise.} \end{cases}$$

$$[\![\text{def}(P)]\!]^{\mathfrak{A}}_{\zeta} := \begin{cases} \mathit{true}, & \text{if there exists an (finite) update set } U \text{ with} \\ & \text{yields}(P, \mathfrak{A}, \zeta, U, A); \\ \mathit{false}, & \text{otherwise.} \end{cases}$$

$$[\![\text{upd}(P, f, s, t)]\!]^{\mathfrak{A}}_{\zeta} := \begin{cases} \mathit{true}, & \text{if yields}(P, \mathfrak{A}, \zeta, U, A) \text{ and} \\ & \langle f, [\![s]\!]^{\mathfrak{A}}_{\zeta}, [\![t]\!]^{\mathfrak{A}}_{\zeta} \rangle \in U; \\ \mathit{false}, & \text{otherwise.} \end{cases}$$

Derived Predicates

- Consistency

$$\mathsf{con}(P) \triangleq \bigwedge_{f \text{ dyn.}} \forall x, y, z \, (\mathsf{upd}(P, f, x, y) \wedge \mathsf{upd}(P, f, x, z) \rightarrow y = z)$$

$$\mathsf{Con}(P) \triangleq \mathsf{def}(P) \wedge \mathsf{con}(P)$$

- Invariance

$$\mathsf{inv}(P, f, s) \triangleq \mathsf{def}(P) \wedge \forall y \, \neg \mathsf{upd}(P, f, s, y)$$

Access Predicates

New Predicates

$$[\![\mathsf{acc}(s, f, t)]\!]^{\mathfrak{A}}_{\zeta} := \begin{cases} \mathit{true}, & \text{if } \langle f, [\![t]\!]^{\mathfrak{A}}_{\zeta}\rangle \in \mathsf{AccSet}(t, \mathfrak{A}, \zeta); \\ \mathit{false}, & \text{otherwise.} \end{cases}$$

$$[\![\mathsf{acc}(\varphi, f, t)]\!]^{\mathfrak{A}}_{\zeta} := \begin{cases} \mathit{true}, & \text{if } \langle f, [\![t]\!]^{\mathfrak{A}}_{\zeta}\rangle \in \mathsf{AccSet}(\varphi, \mathfrak{A}, \zeta); \\ \mathit{false}, & \text{otherwise.} \end{cases}$$

$$[\![\mathsf{acc}(P, f, t)]\!]^{\mathfrak{A}}_{\zeta} := \begin{cases} \mathit{true}, & \text{if } \mathsf{yields}(P, \mathfrak{A}, \zeta, U, A) \text{ and} \\ & \langle f, [\![t]\!]^{\mathfrak{A}}_{\zeta}\rangle \in A; \\ \mathit{false}, & \text{otherwise.} \end{cases}$$

Why only dynamic functions ?

- Static functions are "built-in" (or "hard-wired") operations.

- Substitution principle

$$\forall x \forall y\, (\mathsf{acc}(h(x) < 0, h, y) \rightarrow y = x)$$

$$\forall y\, (\mathsf{acc}(h(h(0)), h, y) \rightarrow y = h(0)) \quad \textit{WRONG}!$$

$$\forall x\, (x = 0 \rightarrow [f(0) := 1]x = 0)$$

$$f(0) = 0 \rightarrow [f(0) := 1]f(0) = 0 \qquad \textit{WRONG}!$$

Derived Predicates

- Q does not read locations updated by P

$$\mathsf{not\_read}(P, Q) \triangleq \bigwedge_{f \text{ dyn.}} \forall x \, (\mathsf{acc}(Q, f, x) \rightarrow \mathsf{inv}(P, f, x))$$

- Q does not change after firing P

$$\mathsf{same\_after}(P, Q) \triangleq \bigwedge_{f \text{ dyn.}} \begin{cases} \forall x, y \, (\mathsf{upd}(Q, f, x, y) \leftrightarrow [P]\mathsf{upd}(Q, f, x, y)) \\ \forall x \, (\mathsf{acc}(Q, f, x) \leftrightarrow [P]\mathsf{acc}(Q, f, x)) \end{cases}$$

- P and Q are equivalent

$$(P \equiv Q) \triangleq \mathsf{def}(P) \wedge \mathsf{def}(Q) \wedge \bigwedge_{f \text{ dyn.}} \begin{cases} \forall x, y \, (\mathsf{upd}(P, f, x, y) \leftrightarrow \mathsf{upd}(Q, f, x, y)) \wedge \\ \forall x \, (\mathsf{acc}(P, f, x) \leftrightarrow \mathsf{acc}(Q, f, x)) \end{cases}$$

Axioms

I.- IX. Axioms of the basic logic

X. Axioms for the acc predicates

1. Axioms for the new predicates
2. $\mathsf{acc}(P, f, x) \rightarrow \mathsf{def}(P)$
3. $\mathsf{Con}(P) \wedge \mathsf{def}(Q) \wedge \mathsf{not\_read}(P, Q) \rightarrow \mathsf{same\_after}(P, Q) \wedge [P]\mathsf{def}(Q)$

Axioms (cont.)

- An axiom for the acc predicate for terms.

  AT2. $\mathsf{acc}(f(t), f, x) \leftrightarrow x = t \vee \mathsf{acc}(t, f, x)$

- Some axioms for the acc predicate for formulas.

  AF3. $\mathsf{acc}(\varphi \wedge \psi, f, x) \leftrightarrow \mathsf{acc}(\varphi, f, x) \vee \mathsf{acc}(\psi, f, x)$

  AF4. $\mathsf{acc}(\varphi \,\&\&\, \psi, f, x) \leftrightarrow \mathsf{acc}(\varphi, f, x) \vee (\hat{\varphi} \wedge \mathsf{acc}(\psi, f, x))$

- Some axioms for the acc predicate for rules.

  AR2. $\mathsf{acc}(g(s) := t, f, x) \leftrightarrow \mathsf{acc}(s, f, x) \vee \mathsf{acc}(t, f, x)$

  AR3. $\mathsf{acc}(P \textbf{ par } Q, f, x) \leftrightarrow$
  $\mathsf{def}(P \textbf{ par } Q) \wedge (\mathsf{acc}(P, f, x) \vee \mathsf{acc}(Q, f, x))$

  AR7. $\mathsf{acc}(P \textbf{ seq } Q, f, x) \leftrightarrow$
  $(\mathsf{acc}(P, f, x) \wedge [P]\mathsf{def}(Q)) \vee (\mathsf{Con}(P) \wedge [P]\mathsf{acc}(Q, f, x))$

Applications: Sequentialization

- Q does not read locations updated by P

$$\mathsf{not\_read}(P, Q) \triangleq \bigwedge_{f \text{ dyn.}} \forall x\, (\mathsf{acc}(Q, f, x) \rightarrow \mathsf{inv}(P, f, x))$$

- Q does not overwrite updates of P

$$\mathsf{not\_over}(P, Q) \triangleq$$
$$\bigwedge_{f \text{ dyn.}} \forall x, y\, (\mathsf{upd}(P, f, x, y) \rightarrow \mathsf{inv}(Q, f, x) \vee \mathsf{upd}(Q, f, x, y))$$

- P **par** $Q \equiv P$ **seq** Q

$$\mathsf{Con}(P) \wedge \mathsf{def}(Q) \wedge \mathsf{not\_read}(P, Q) \wedge \mathsf{not\_over}(P, Q) \rightarrow$$
$$P \textbf{ par } Q \equiv P \textbf{ seq } Q$$

Applications: MergeSort

- Index overflow

$$\forall l, r, m \; (l < r \wedge m = (l+r)/2 \\ \wedge \exists i \, (l \leq i \leq m \wedge \forall j \, (m+1 \leq j \leq r \rightarrow f(j) < f(i))) \\ \rightarrow \mathsf{acc}(\textsc{MergeSortV}(l, r), g, r+1))$$

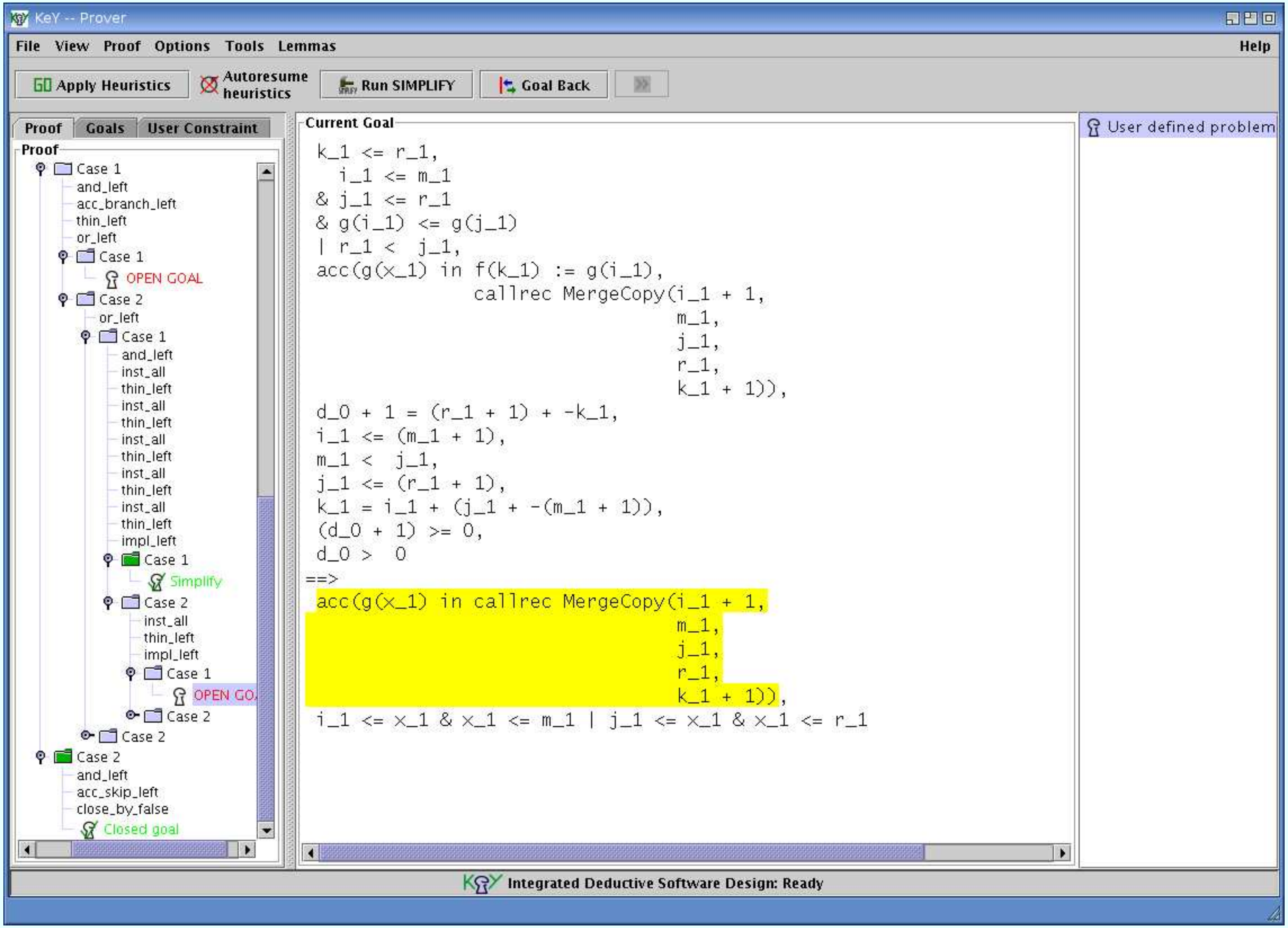
- Correct accesses

$$\forall l, r \, (\forall x \, (\mathsf{acc}(\textsc{MergeSort}(l, r), g, x) \rightarrow (l \leq x \wedge x \leq r)))$$

- Sequentialization

$$\forall l, r \, (\textsc{MergeSort}(l, r) \equiv \textsc{SeqMergeSort}(l, r))$$

ASMKeY


KeY -- Prover
File View Proof Options Tools Lemmas Help
Apply Heuristics
Autoresume heuristics
Run SIMPLIFY
Goal Back
Proof Goals User Constraint
Proof
Case 1
and_left
acc_branch_left
thin_left
or_left
Case 1
OPEN GOAL
Case 2
or_left
Case 1
and_left
inst_all
thin_left
inst_all
thin_left
inst_all
thin_left
inst_all
thin_left
inst_all
thin_left
impl_left
Case 1
Simplify
Case 2
inst_all
thin_left
impl_left
Case 1
OPEN GO
Case 2
Case 2
Case 2
and_left
acc_skip_left
close_by_false
Closed goal
Current Goal
k_1 <= r_1,
i_1 <= m_1
& j_1 <= r_1
& g(i_1) <= g(j_1)
| r_1 < j_1,
acc(g(x_1) in f(k_1) := g(i_1),
callrec MergeCopy(i_1 + 1,
m_1,
j_1,
r_1,
k_1 + 1)),
d_0 + 1 = (r_1 + 1) + -k_1,
i_1 <= (m_1 + 1),
m_1 < j_1,
j_1 <= (r_1 + 1),
k_1 = i_1 + (j_1 + -(m_1 + 1)),
(d_0 + 1) >= 0,
d_0 > 0
==>
acc(g(x_1) in callrec MergeCopy(i_1 + 1,
m_1,
j_1,
r_1,
k_1 + 1)),
i_1 <= x_1 & x_1 <= m_1 | j_1 <= x_1 & x_1 <= r_1
User defined problem
Integrated Deductive Software Design: Ready

ASMKeY

- Work in Progress
  - Step logic
  - Static analysis
- Future Work
  - Strategies
  - Cases studies: LiftControler, Shortest Path Algorithm, (C# Thread Model)
  - Source Code: create a 'real' ASMKeY distribution

Conclusion

- Summary
  - New predicates: acc(t, f, s), acc(φ, f, s) and acc(R, f, s)
  - Allows to express secure access properties
  - Gives a criterion for sequentialization of parallel ASMs
  - Implemented in ASMKeY
- Future Work
  - Extend the logic for ASMs with the **choose** rule
  - ASMKeY

Access Sets and Update Sets (cont.)

The problem of **forall**

$$\frac{\mathsf{yields}(P, \mathfrak{A}, \zeta[x \mapsto v], U_v, A_v) \quad \text{for each } v \in I = \mathrm{range}(x, \varphi, \mathfrak{A}, \zeta)}{\mathsf{yields}(\textbf{forall } x \textbf{ with } \varphi \textbf{ do } P, \mathfrak{A}, \zeta, \bigcup_{v \in I} U_v, \bigcup_{v \in I} A_v \cup \Phi)}$$

where

- $\mathrm{range}(x, \varphi, \mathfrak{A}, \zeta) \triangleq \{v \in |\mathfrak{A}| : [\![\varphi]\!]^{\mathfrak{A}}_{\zeta[x \mapsto v]} = \mathit{true}\}$
- $\Phi = \bigcup_{v \in |\mathfrak{A}|} \mathsf{AccSet}(\varphi, \mathfrak{A}, \zeta[x \mapsto v])$

Example:

forall x **with** $0 \leq x$ && $x \leq 10$ && $f(x) < 0$ **do** $f(x) := 0$
Access set: $\{\langle f, 0\rangle, \langle f, 1\rangle, \ldots, \langle f, 10\rangle\}$

Access Sets and Observable Terms (Critical Terms)

(For a non-recursive ASM P without **forall** rules)

$A \triangleq \{\langle f, a\rangle \mid P$ reads the location $\langle f, a\rangle\}$

$D \triangleq \{\langle f, a\rangle \mid P$ depends on the location $\langle f, a\rangle\}$

$O \triangleq \{\langle f, a\rangle \mid \langle f, a\rangle$ is the location of an observable term of $P\}$

- $A \not\subseteq O$, Example:

  if $f(0) := 0$ **then** $f(1) := 1$ **else** $f(1) := 1$
  Observable terms: $\{1\}$

- $D \not\subseteq O$, Example:

  if $f(f(0)) = 0$ **then** $f(0) := 1$
  Observable terms: $\{0, 1, f(f(0))\}$

- $D \subseteq \{\langle f, a\rangle \mid \langle f, a\rangle$ is the location of a subterm of an observable term of $M\}$

Theorems

- Soundness of the logic

  If $\Psi \vdash_M \varphi$, then $\Psi \models_M \varphi$.

- Independence lemma

  For an ASM Q, if yields$(Q, \mathfrak{A}, \zeta, U, A)$ and W is a consistent update set that does not contain updates for locations in A, then yields$(Q, \mathfrak{A} + W, \zeta, U, A)$.

- Completeness of the logic for hierachical ASMs.

  Let M be a hierchical ASM. If $\Psi \models_M \varphi$, then $\Psi \vdash_M \varphi$.