From Hoare Logic to Matching Logic Reachability

Grigore Rosu and Andrei Stefanescu
University of Illinois, USA

Matching Logic Reachability - Goal -

- Language independent program verification framework
  - Derives program properties based on the operational semantics of a language
- Language independent proof system (ICALP'12)
- *Question*: is this approach as expressive and powerful as Hoare logic?
- *Answer*: yes!
  - Hoare logic derivation translated into matching logic reachability derivation
  - Translation size is linear
- Consequences
  - Relative completeness
  - Alternative way of proving Hoare logic sound

Summary

- Operational semantics
- Axiomatic semantics (Hoare logic)
- Matching logic
- Matching logic reachability
- Proof translation from Hoare logic to matching logic reachability

Operational Semantics

Operational Semantics

- Easy to define and understand
  - Can be regarded as formal “implementations”
- Require little mathematical knowledge
  - Great introductory topics in PL courses
- Scale up well
  - C (>1000 rules), Java, Scheme, Verilog, …, defined
- Executable, so testable
  - C semantics tested against real benchmarks

Operational Semantics of IMP
- Sample Rules -

$$\texttt{if}(i)\ s_1\ \texttt{else}\ s_2 \Rightarrow s_1 \qquad \text{if } i \neq 0$$

$$\texttt{if}(0)\ s_1\ \texttt{else}\ s_2 \Rightarrow s_2$$

$$\texttt{while}(e)\ s \Rightarrow \texttt{if}(e)\ s; \texttt{while}(e)\ s\ \texttt{else skip}$$

$$\texttt{proc}() \Rightarrow \text{body} \qquad \text{where “}\texttt{proc}()\ \text{body”}$$

May need to be completed “all the way to top”, into rules between configurations:

$$\langle C, \sigma\rangle[\texttt{if}(i)\ s_1\ \texttt{else}\ s_2] \Rightarrow \langle C, \sigma\rangle[s_1] \qquad \text{if } i \neq 0$$

Operational Semantics
- Bottom Line (well-known) -

We can operationally define any programming languages only with rules of the form

$$l \Rightarrow r \text{ if } b$$

where l, r are "top-level" configuration terms, and b is a Boolean side condition

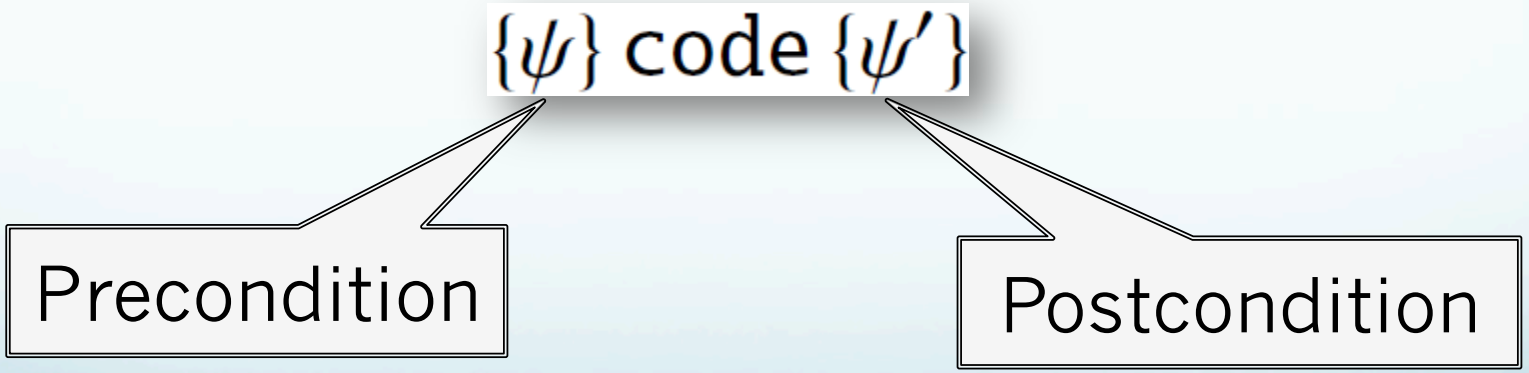
Unfortunately ...

- Operational semantics considered inappropriate for program reasoning
- Proofs based on operational semantics are low-level and tedious
  - Have to formalize and work with transition system
  - Induction on structure, number of steps, etc.

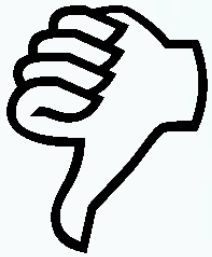
Axiomatic Semantics

Axiomatic Semantics (Hoare Logic)

- Focused on reasoning
- Programming language captured as a formal proof system that allows to derive triples



Axiomatic Semantics



- Not easy to define and understand, error-prone
  - Not executable, hard to test; require program transformations which may lose behaviors, etc.

$$\frac{\mathcal{H} \vdash \{\psi \wedge \mathtt{e} \neq 0\}\, \mathtt{s}\, \{\psi\}}{\mathcal{H} \vdash \{\psi\}\, \mathtt{while(e)}\, \mathtt{s}\, \{\psi \wedge \mathtt{e} = 0\}}$$

$$\frac{\mathcal{H} \cup \{\psi\}\, \mathtt{proc()}\, \{\psi'\} \vdash \{\psi\}\, \mathtt{body}\, \{\psi'\}}{\mathcal{H} \vdash \{\psi\}\, \mathtt{proc()}\, \{\psi'\}}$$

State-of-the-art in Certifiable Verification

- Define an operational semantics, which acts as trusted reference model of the language
- Define an axiomatic semantics, for reasoning
- Prove the axiomatic semantics sound for the operational semantics

- Now we have trusted verification ...
- ... but the above needs to be done for each language individually; at best uneconomical

Unified Theory of Programming
- (Hoare and Jifeng) -

- Framework where various semantics of the same language coexist, with systematic relationships (e.g., soundness) proved
- Then use one semantics or another ...

- This still requires two or more semantics for the same language (C semantics took >2years)
- Uneconomical, people will not do it

Unified Theory of Programming - Our Approach -

- Underlying belief
  - A language should have only one semantics, which should be easy, executable, and good for program reasoning. One semantics to rule them all.
- Approach
  - Devise language-independent proof system that takes operational semantics “as is” and derives any program property, stated as reachability rules (including Hoare triples).

Matching logic

Matching Logic
(AMAST'10)

- Logic for stating and reasoning about *static* properties of configurations
- Matching logic: extend FOL with *patterns*
  - Special predicates which are configuration terms
  - Configurations satisfy patterns iff they match them
- **IMP** configurations

$$\langle\ Code,\ Store\ \rangle$$

Matching Logic
- Sample Patterns -

$$\langle\ \texttt{if}(i)\ s_1\ \texttt{else}\ s_2,\ \sigma\ \rangle \wedge\ i \neq 0$$

SUM

$$\exists s\ (\langle\ \texttt{s:=0; while(n>0)(s:=s+n; n:=n-1)},\ (\texttt{s} \mapsto s,\ \texttt{n} \mapsto n)\ \rangle \wedge n \geq_{Int} 0)$$

$$\langle \texttt{skip},\ (\texttt{s} \mapsto n *_{Int} (n +_{Int} 1)/_{Int} 2,\ \texttt{n} \mapsto 0)\rangle$$

Matching logic reachability

Reachability Rule

- State and reason about *dynamic* properties of configurations
- Pair of patterns, with meaning “reachability”

$$\varphi \Rightarrow \varphi'$$

- We define the validity of reachability rules in terms of the transition system induced by the operational semantics

$$\mathcal{S} \models \varphi \Rightarrow \varphi'$$

- Reachability rules generalize both operational semantics rules and Hoare triples

Operational Semantics Rules as Reachability Rules

Operational semantics rule

$$l \Rightarrow r \text{ if } b$$

is syntactic sugar for reachability rule

$$l \wedge b \Rightarrow r$$

From Hoare Triples To Reachability Rules

$$\{\psi\}\ \texttt{code}\ \{\psi'\}$$

$$\exists X\,(\langle \texttt{code},\ \sigma_X\rangle \wedge \psi_{X,Y}) \Rightarrow \exists X\,(\langle \texttt{skip},\ \sigma_X\rangle \wedge \psi'_{X,Y})$$

- X, Y : sets of logical variables
- σ_X : state mapping program variables into variables in X
- $\psi_{X,Y}, \psi'_{X,Y}$: formulae over the variables in X, Y

H2ML Sample Application

- For the SUM program

```
s:=0; while(n>0)(s:=s+n; n:=n-1)
```

$$\{\mathtt{n} = \mathtt{oldn} \wedge \mathtt{n} \geq 0\}\ \mathtt{SUM}\ \{\mathtt{s} = \mathtt{oldn}*(\mathtt{oldn}+1)/2 \wedge \mathtt{n} = 0\}$$

$$\begin{aligned} & \exists s, n\ (\langle \mathtt{SUM},\ (\mathtt{s} \mapsto s,\ \mathtt{n} \mapsto n)\rangle \wedge n = oldn \wedge n \geq_{Int} 0) \\ \Rightarrow\ & \exists s, n\ (\langle \mathtt{skip},\ (\mathtt{s} \mapsto s,\ \mathtt{n} \mapsto n)\rangle \wedge s = oldn *_{Int} (oldn +_{Int} 1)/_{Int} 2 \wedge n = 0) \end{aligned}$$

Reasoning about Reachability

- Having generalized the elements of both operational and axiomatic semantics, we now want a proof system for deriving reachability rules from reachability rules:

$$\mathcal{A} \vdash \varphi \Rightarrow \varphi'$$

Reachability Proof System - 9 Rules (ICALP'12) -

Symbolic execution(one step)

Rules of operational nature

Reflexivity :

$$\frac{\cdot}{\mathcal{A} \vdash \varphi \Rightarrow \varphi}$$

Case Analysis :

$$\frac{\mathcal{A} \vdash \varphi_1 \Rightarrow \varphi \qquad \mathcal{A} \vdash \varphi_2 \Rightarrow \varphi}{\mathcal{A} \vdash \varphi_1 \vee \varphi_2 \Rightarrow \varphi}$$

Axiom :

$$\frac{\varphi \Rightarrow \varphi' \in \mathcal{A}}{\mathcal{A} \vdash \varphi \Rightarrow \varphi'}$$

Logic Framing :

$$\frac{\mathcal{A} \vdash \varphi \Rightarrow \varphi' \qquad \psi \text{ is a (patternless) FOL formula}}{\mathcal{A} \vdash \varphi \wedge \psi \Rightarrow \varphi' \wedge \psi}$$

Substitution :

$$\frac{\mathcal{A} \vdash \varphi \Rightarrow \varphi' \qquad \theta : Var \to \mathcal{T}_\Sigma(Var)}{\mathcal{A} \vdash \theta(\varphi) \Rightarrow \theta(\varphi')}$$

Consequence :

$$\frac{\models \varphi_1 \to \varphi_1' \qquad \mathcal{A} \vdash \varphi_1' \Rightarrow \varphi_2' \qquad \models \varphi_2' \to \varphi_2}{\mathcal{A} \vdash \varphi_1 \Rightarrow \varphi_2}$$

Transitivity :

$$\frac{\mathcal{A} \vdash \varphi_1 \Rightarrow \varphi_2 \qquad \mathcal{A} \vdash \varphi_2 \Rightarrow \varphi_3}{\mathcal{A} \vdash \varphi_1 \Rightarrow \varphi_3}$$

Abstraction :

$$\frac{\mathcal{A} \vdash \varphi \Rightarrow \varphi' \qquad X \cap FreeVars(\varphi') = \emptyset}{\mathcal{A} \vdash \exists X\, \varphi \Rightarrow \varphi'}$$

Rule for circular behavior

$$\frac{\ldots \Rightarrow \varphi'\} \vdash \varphi'' \Rightarrow \varphi'}{\varphi'}$$

Symbolic execution(multiple steps)

Circular behaviors

- Circularity proof rule

$$\frac{\mathcal{A} \vdash \varphi \Rightarrow^{+} \varphi'' \qquad \mathcal{A} \cup \{\varphi \Rightarrow \varphi'\} \vdash \varphi'' \Rightarrow \varphi'}{\mathcal{A} \vdash \varphi \Rightarrow \varphi'}$$

- Hoare logic rule for while loops

$$\frac{\mathcal{H} \vdash \{\psi \wedge \mathtt{e} \neq 0\}\, \mathtt{s}\, \{\psi\}}{\mathcal{H} \vdash \{\psi\}\, \mathtt{while(e)}\, \mathtt{s}\, \{\psi \wedge \mathtt{e} = 0\}}$$

Proof Translation

Main Result

Theorem: If $\{\psi\}\,\mathbf{code}\,\{\psi'\}$ is derivable in the Hoare logic of **IMP**, then $\mathcal{S}_{\text{IMP}} \vdash H2M(\{\psi\}\,\mathbf{code}\,\{\psi'\})$ is derivable by the matching logic reachability proof system, where $\mathcal{S}_{\text{IMP}}$ is the operational semantics of **IMP.**

Proof Idea

Proof by induction. For each Hoare logic proof rule

$$\frac{T_1 \ldots T_n}{T}$$

we derive the conclusion $\mathcal{S}_{\text{IMP}} \vdash H2M(T)$ from the premises $\mathcal{S}_{\text{IMP}} \vdash H2M(T) \ \ldots \ \mathcal{S}_{\text{IMP}} \vdash H2M(T)$ with the proof system.

For most proof rules, the first eight rules suffice.

The proof idea is generic, and should extend to any programming language.

While Loop

- While loop rule

$$\mathcal{S}_{\text{IMP}} \vdash \exists X\,(\langle \texttt{while(e) s},\ \sigma_X\rangle \wedge \psi_{X,Y}) \Rightarrow \exists X\,(\langle \texttt{skip},\ \sigma_X\rangle \wedge (\psi \wedge \texttt{e} = 0)_{X,Y})$$

- Steps
  - Circularity
  - Loop unrolling
  - Symbolic evaluation of the condition
  - Case Analysis
  - Use the premise + the rule itself

Size of the Translated Proof

- For each Hoare logic proof rule the reachability derivation has constant size
- The size of the mechanically generated reachability derivation is linear in the size of the HL derivation
- In practice, reasoning directly in matching logic reachability is better than the mechanical translation. See the paper for details

Conclusions

- Matching logic reachability is at least as expressive and as powerful as Hoare logic
- The size of reachability proofs is at most within a linear factor of the size of Hoare logic proofs
- Proved for a simple imperative language, but should work with any language
- Matching logic reachability is relatively complete
- Alternative way of proving Hoare logic sound