Towards a Unified Theory of Operational and Axiomatic Semantics

Grigore Rosu and Andrei Stefanescu
University of Illinois, USA

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 -

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

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

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

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

Operational Semantics of IMP
- Sample Rules -

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

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

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

$$\texttt{proc}() \ \Rightarrow\ \texttt{body} \qquad \text{where ``}\texttt{proc}()\ \texttt{body}\text{''}$$

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 rewrite 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 (HOARE LOGIC)

Axiomatic Semantics

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

$$\{\psi\}\ \text{code}\ \{\psi'\}$$

Precondition

Postcondition

Axiomatic Semantics

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

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

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

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 reachability property (including Hoare triples).

Matching Logic

(AMAST'10, ICSE'11, ICALP'12, FM'12, OOPSLA'12)

- Logic for reasoning about structure
- Matching logic: extend FOL with *patterns*
  - Special predicates which are open configuration terms, whose meaning is "can you match me?"
- Examples of patterns:

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

SUM

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

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

Reachability Rule

- Pair of patterns, with meaning "reachability"

$$\varphi \Rightarrow \varphi'$$

- Reachability rules generalize both operational semantics rules and Hoare triples

Operational Semantics Rules are Reachability Rules

Operational semantics rule

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

is syntactic sugar for reachability rule

$$l \wedge b \Rightarrow r$$

We can associate a transition system to any set of reachability rules, and define validity; see paper

$$S \models \varphi \Rightarrow \varphi'$$

Hoare Triples are Reachability Rules

Hoare triple

$$\{\psi\}\,\mathrm{code}\,\{\psi'\}$$

is syntactic sugar for reachability rule

$$\exists X_{\mathrm{code}}(\langle \mathrm{code}, \sigma_{X_{\mathrm{code}}}\rangle \wedge \psi_X) \Rightarrow \exists X_{\mathrm{code}}(\langle \mathtt{skip}, \sigma_{X_{\mathrm{code}}}\rangle \wedge \psi'_X)$$

… but there are better ways to specify program properties; see the paper

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 language-independent rules -

Rules of operational nature

Reflexivity :

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

Axiom :

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

Substitution :

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

Transitivity :

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

Rules of deductive nature

Case Analysis :

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

Logic Framing :

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

Consequence :

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

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

Circularity :

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

Rule 1
Reflexivity

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

Rule 2
Axiom

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

Rule 3
Substitution

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

Rule 4

Transitivity

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

Rule 5
Case Analysis

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

Rule 6
Logic Framing

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

Rule 7

Consequence

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

Rule 8

Abstraction

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

Rule 9
Circularity

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

Main Result
Soundness

<u>**Theorem:**</u> If $\mathcal{S} \vdash \varphi \Rightarrow \varphi'$ derivable with the nine-rule proof system, then $\mathcal{S} \models \varphi \Rightarrow \varphi'$

Conclusion

- Proof system for reachability
- Works with any operational semantics, as is
- Requires no other semantics of the language
- Unlike Hoare logics, which are language-specific, our proof system is
  - Language-independent (takes language as axioms)
  - Proved sound only once, for all languages
- Has been implemented in MatchC and works
- Can change the way we do program verification