Properties of Axiomatic Semantics

Winskel: The Formal Semantics of Programming Languages, Chapter 6& 7

Krzysztof R. Apt, Frank S. de Boer, Ernst-Rüdiger Olderog:
Verification of Sequential and Concurrent Programs

Topics

- Reasoning about concurrent systems
- Soundness and completeness of Hoare rules

Parallel Composition Rule

$$\frac{\{p_1\}\ c_1\ \{q_1\}\ \{p_2\}\ c_2\ \{q_2\}..\ \{p_k\}\ c_k\ \{q_k\}}{\{p_1 \wedge p_2 \wedge \ldots\ \underline{p_k}\}\ \text{cobegin}\ c_1\ ||\ c_2\ ||\ \ldots ||\ c_k\ \text{coend}\ \{q_1 \wedge q_2 \wedge \ldots \wedge q_k\}}$$

Interference

- A command T with a precondition pre(T) does not interfere with the proof of {p} c {q} if:
  - $\{q \wedge pre(T)\}\ T\ \{q\}$
  - For any command c' inside c with a precondition pre(c')
    - $\{pre(c') \wedge pre(T)\}\ T\ \{pre(c')\}$
- $\{p_1\}\ c_1\ \{q_1\}\ \{p_2\}\ c_2\ \{q_2\}..\ \{p_k\}\ c_k\ \{q_k\}$ are interference free if for every $i \neq j$ and for every assignment in T in c_i does not interfere with $\{p_j\}\ c_j\ \{q_j\}$

Parallel Composition Rule

$\{p_1\}\, c_1\, \{q_1\}\ \{p_2\}\, c_2\, \{q_2\} .. \{p_k\}\, c_k\, \{q_k\}$

$\{p_1\} q_1\, \{q_1\}\ \{p_2\}\, c_2\, \{q_2\} .. \{p_k\}\, c_k\, \{q_k\}$ are interference free

---

$\{p_1 \wedge p_2 \wedge \ldots \underline{p_k}\}$ cobegin $c_1 \,||\, c_2 \,||\, \ldots || \, c_k$ coend $\{q_1 \wedge q_2 \wedge \ldots \wedge q_k\}$

Limitations of the Owicky & Gries proof rules

- Checking interference can be hard
  - Non-compositionality
    - Until you finished the local proofs cannot check interference
- A non-standard meaning of Hoare triples
  - Depends on the interference of other threads with the proof
  - Proofs need to be saved
  - Hard to handle libraries and missing code
  - Soundness is non-trivial
- Completeness depends on auxiliary variables
- Used in practice
- Generalization Rely/Guarantee

Example

{x ≥0}		{x ≥0}		{x ≥0}
x := x + 1;		x := x + 2;		x := x + 200;
y := 1	||	y := 2	|| …	y := 200
{x >0}		{x >0}		{x >0}

Soundness of Hoare Rules

- Every theorem obtained by the rule system is valid
  - $\vdash \{P\}\ c\ \{Q\} \Rightarrow \models \{P\}\ c\ \{Q\}$
- The system can be implemented (HOL, LCF, Coq)
  - Requires user assistance
- Proof of soundness
  - Every rule preserves validity (Theorem 6.1)

Soundness of skip axiom

$$\models \{A\}\ \text{skip}\ \{A\}$$

Soundness of the assignment axiom

$$\models \{B[a/X]\}\ X:=a\ \{B\}$$

Soundness of the sequential composition rule

- Assume that
  $\models\{P\}\ S_0\ \{C\}$
  and
  $\models\{C\}\ S_1\ \{Q\}$
- Show that
  $\models\{P\}\ S_0;S_1\{Q\}$

Soundness of the conditional rule

- Assume that
  $\models\{P \wedge b\}\ S_0\ \{Q\}$
  and
  $\models\{P \wedge \neg b\}\ S_1\ \{Q\}$
- Show that
  $\models\{P\}$ if b then S_0 else $S_1\{Q\}$

Soundness of the while rule

- Assume that
  $\models\{I \wedge b\}\ S\ \{I\}$
- Show that
  $\models\{I\}$ while b do S $\{I \wedge \neg b\}$

Soundness of the consequence rule

- Assume that
  $\models\{P'\}\ S\ \{Q'\}$
  and
  $\models P \Rightarrow P'$
  and
  $\models Q' \Rightarrow Q$
- Show that
  $\models\{P\}\ S\ \{Q\}$

(Ideal) Completeness

- Every valid theorem can be proved by the rule system
- For every P and Q such that $\models\{P\}\ S\ \{Q\}$
  there exists a proof such $\vdash \{P\}\ S\ \{Q\}$
- But what about Gödel's incompleteness?
  $\models\{true\}$ skip $\{Q\}$
- What does $\models\{true\}$ c $\{false\}$ mean?

Relative Completeness (Chapter 7)

- Assume that every math theorem can be proved
  $\models \{P\}\ S\ \{Q\}$ implies $\vdash \{P\}\ S\ \{Q\}$

Relative completeness of composition rule

- Prove that $\{P\}\ S_0;S_1\{Q\}$
- Does there exist an assertion I such that
  $\models\{P\}\ S_0\ \{I\}$
  and
  $\models\{I\}\ S_1\ \{Q\}$

Weakest (Liberal) Precondition [Dijkstra75]

- wp(S, Q) – the weakest condition such that every terminating computation of S results in a state satisfying Q
- $[\![wp^{l}(S, Q)]\!] = \{\sigma \in \Sigma^{\perp} \mid S[\![S]\!]\, \sigma \models^{l} Q\}$
- [Can employ predicate transformer semantics to formally define the meaning (Chapter 7.5)]
- Prove that $\{P\}\ S_0;S_1\{Q\}$ by proving
  $\models\{P\}\ S_0\ \{I\}$
  and
  $\models\{I\}\ S_1\ \{Q\}$ where $I=wp(S_1, Q)$
- $\models\{P\}\ S\ \{Q\}$ iff for all $I\ [\![P]\!] \subseteq [\![wp^{l}(S, Q)]\!]$
- $\models\{P\}\ S\ \{Q\}$ iff for $P \Rightarrow wp(S, Q)$

Some WP rules

- wp(skip, Q) = Q
- wp(X := a, Q) = Q[a/X]
- wp(S_0; S_1, Q) = wp(S_0, wp(S_1, Q))
- wp(if b then S_0 else S_1, Q) =
  b $\wedge$wp(S_0, Q) $\vee$ $\neg$ b $\wedge$wp(S_1, Q)
- wp(S, true) =
- wp(S, false)
- wp(S, P $\wedge$ Q) =

Detour Strongest Postcondition

- For every command c and assertion B
  - there exists an assertion A, such that A=wp(c, B) (Theorem 7.5)
  - $\vdash$\{wp(c, B)\} c \{B\}(Lemma 7.6)
- Theorem 7.7: The proof system is relatively complete
  - $\models$\{P\} c \{Q\} implies $\vdash$ \{P\} c \{Q\}

Theorem 7.5: Assn is expressive

- For every command c and assertion B, there exists an assertion w[c, B] such that for every assignment I, $w^I[c,B] = wp^I(c, B)$

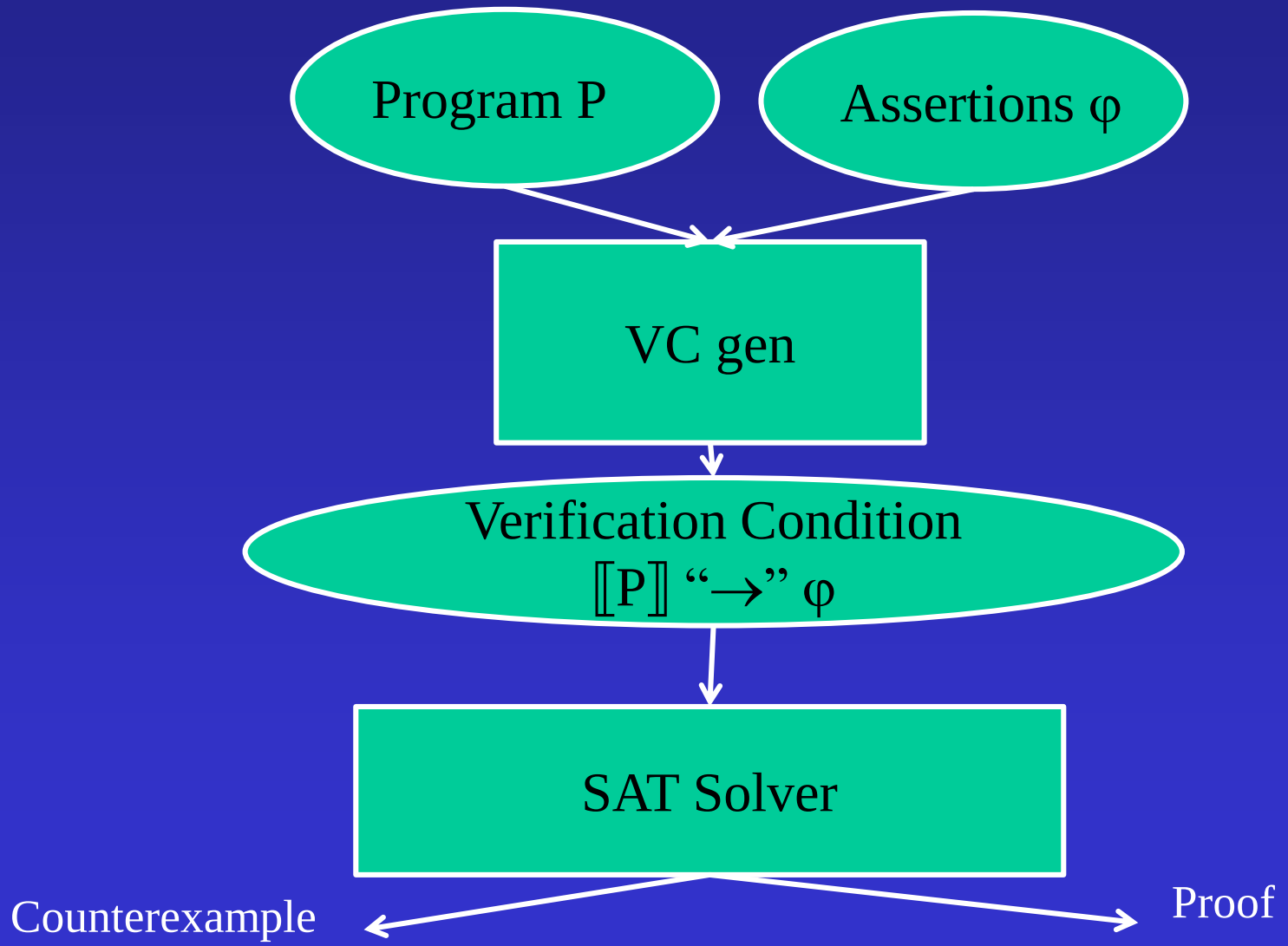
Lemma 7.6: Hoare Proof Rules are relatively complete

- For every command c and assertion B
  $\vdash$ {wp(c, B)} c {B}

Verification Conditions

- Generate assertions that describe the partial correctness of the program
- Use automatic theorem provers to show partial correctness
- Existing tools ESC/Java, Spec#, Dafny

Verification Process



WP Compound statements

- wp (skip, Q) = Q
- wp(x := e,Q) = Q[e / x]
- wp(S_1; S_2,Q) = wp(S_1,wp(S_2,Q)
- wp(if B then S_1 else S_2) =
  B $\wedge$wp(S_1 ,Q) $\vee$($\neg$B$\wedge$ wp(S_2 ,Q))
- wp(while B do {I} S) = I

VC rules

- $VC_{gen}(\{P\}\ S\ \{Q\}) = P \rightarrow wp(S, Q) \wedge \bigwedge VC_{aux}(S, Q)$
- $VC_{aux}(S, Q) = \{\}$ (for any atomic statement)
- $VC_{aux}(S_1; S_2, Q) =$
  $VC_{aux}(S_1, wp(S_2, Q)) \cup VC_{aux}(S_2, Q)$
- $VC_{aux}(\text{if C then } S_1 \text{ else } S_2, Q) =$
  $VC_{aux}(S_1, Q) \cup VC_{aux}(S_2, Q)$
- $VC_{aux}(\text{while B do S}, Q) = VC_{aux}(S, I) \cup$
  $\{I \wedge B \rightarrow wp(S, I)\} \cup$
  $\{I \wedge \neg B \rightarrow Q\}$

Summary

- Axiomatic semantics provides an abstract semantics
- Can be used to explain programming
- Extensions
  - Procedures
  - Concurrency
  - Events
  - Rely/Guarantee
  - Heaps
- Can be automated
- More effort is required to make it practical

Suggested Projects

- Verification of filesystems FSCQ (MIT) SOSP’15, SOSP’17
- Verification of distributed protocol IronFleet SOSP’15