Programming Language Semantics
Axiomatic Semantics

The Formal Semantics of Programming Languages

Glynn Winskel

Chapter 6

Motivation

- What do we need in order to prove that the program does what it supposed to do?

- Specify the required behavior
- Compare the behavior with the one obtained by the denotational/operational semantics
- Develop a proof system for showing that the program satisfies a requirement
- Mechanically use the proof system to show correctness
- The meaning of a program is a set of verification rules

Plan

- The basic idea
- An assertion language
- Semantics of assertions
- Proof rules
- An example
- Soundness
- Completeness
- Verification conditions

Example Program

S:=0

N := 1

while ¬(N=101) do

S := S + N ;

N :=N+1

N=101

$S=\sum_{1\le m\le 100} m$

Example Program

S:=0

$\{S=0\}$

N := 1

$\{S=0 \wedge N=1\}$

while $\neg$(N=101) do

    S := S + N ;

    N :=N+1

$\{N=101 \wedge S=\sum_{1\leq m\leq 100} m\}$

Example Program

S:=0

$\{S=0\}$

N := 1

$\{S=0 \wedge N=1\}$

while $\{1 \leq N \leq 101 \wedge S=\sum_{1\leq m\leq N-1} m\}$ $\neg(N=101)$ do

S := S + N ;

$\{1 \leq N < 101 \wedge S=\sum_{1\leq m\leq N} m\}$

N :=N+1

$\{N=101 \wedge S=\sum_{1\leq m\leq 100} m\}$

Partial Correctness

- {P}S{Q}
  - P and Q are assertions (extensions of Boolean expressions)
  - S is a statement
  - For all states σ which satisfies P, if the execution of S from state σ terminates in state σ', then σ' satisfies Q
- {true}while true do skip{false}

Total Correctness

- [P]S[Q]
  - P and Q are assertions
    (extensions of Boolean expressions)
  - S is a statement
  - For all states σ which satisfies P,
    - the execution of S from state σ must terminates in a state σ'
    - σ' satisfies Q

Formalizing Partial Correctness

- $\sigma \models A$
  - A is true in σ
- $\{P\}\ S\ \{Q\}$
  - $\forall \sigma, \sigma' \in \Sigma.\ (\sigma \models P\ \&\ \langle S, \sigma \rangle \rightarrow \sigma'\ ) \Rightarrow \sigma' \models Q$
  - $\forall \sigma \in \Sigma.\ (\sigma \models P\ \&\ \mathbf{S}\ [\![S]\!]\sigma \neq \perp) \Rightarrow \mathbf{S}\ [\![S]\!]\sigma \models Q$
- Convention for all A
  $\perp \models A$
- $\forall \sigma, \sigma' \in \Sigma.\ \sigma \models P\ \Rightarrow \mathbf{S}\ [\![S]\!]\sigma \models Q$

An Assertion Language

- Extend Bexp
- Allow quantifications
  - $\forall$i: …
  - $\exists$i: …
    - $\exists i.\ k=i \times l$
- Import well known mathematical concepts
  - $n! \triangleq n \times (n-1) \times \cdots 2 \times 1$

Assertion Language

Aexpv

$$a := n \mid X \mid i \mid a_0 + a_1 \mid a_0 - a_1 \mid a_0 \times a_1$$

Assn

$$A := \text{true} \mid \text{false} \mid a_0 = a_1 \mid a_0 \leq a_1 \mid A_0 \wedge A_1 \mid A_0 \vee A_1 \mid \neg A \mid$$

$$A_0 \Rightarrow A_1 \mid \forall i.\ A \mid \exists i.\ A$$

Example

while $\neg(M=N)$ do

if $M \leq N$

then $N := N - M$

else $M := M - N$

Example

```
int power(int x, unsigned int y)
{
    int temp;
    if( y == 0)
        return 1;
    temp = power(x, y/2);
    if (y%2 == 0)
        return temp*temp;
    else
        return x*temp*temp;
```

Free and Bound Variables

- An integer variable is bound when it occurs in the scope of a quantifier
- Otherwise it is free
- Examples $\exists i.\ k = i \times L \quad (i + 100 \le 77) \wedge \forall i.\ j + 1 = i + 3)$

$FV(n) = FV(X) = \emptyset \qquad FV(i) = \{i\}$

$FV(a_0 + a_1) = FV(a_0 - a_1) = FV(a_0 \times a_1) = FV(a_0) \cup FV(a_1)$

$FV(true) = FV(false) = \emptyset \quad FV(a_0 = a_1) = FV(a_0 \le a_1) = FV(a_0) \cup FV(a_1)$

$FV(A_0 \wedge A_1) = FV(A_0 \vee A_1) = FV(A_0 \Rightarrow A_1) = FV(A_0) \cup FV(A_1)$

$FV(\neg A) = FV(A)$

$FV(\forall i.\ A) = FV(\exists i.\ A) = FV(A) \setminus \{i\}$

Substitution

- Visualization of an assertion A

  ---i---i----

- Consider a "pure" arithmetic expression

  A[a/i] ---a---a---

$n[a/i] = n$

$X[a/i]=X$

$i[a/i] = a$

$j[a/i] = j$

$(a_0 + a_1)[a/i] = a_0[a/i] + a_1/[a/i]$

$(a_0 - a_1)[a/i] = a_0[a/i] - a_1[a/i]$

$(a_0 \times a_1)[a/i]= a_0[a/i] \times a_1[a/i]$

Substitution

- Visualization of an assertion A

  ---i---i----

- Consider a "pure" arithmetic expression

  A[a/i] ---a---a---

true[a/i] = true

false[a/i]=false

$(a_0 = a_1)[a/i] = (a_0/[a/i] = a_1[a/i])$

$(a_0 \leq a_1)[a/i] = (a_0/[a/i] \leq a_1[a/i])$

$(A_0 \wedge A_1)[a/i] = (A_0[a/i] \wedge A_1[a/i])$

$(A_0 \vee A_1)[a/i] = (A_0[a/i] \vee A_1[a/i])$

$(A_0 \Rightarrow A_1)[a/i] = (A_0[a/i] \Rightarrow A_1[a/i])[a/i]$

$(\neg A)[a/i] = \neg(A[a/i])$

$(\forall i.A)[a/i] = \forall i.\ A$

$(\forall j.A)[a/i] = (\forall j.\ A[a/i])$

$(\exists i.A)[a/i] = \exists i.\ A$

$(\exists j.A)[a/i] = (\exists j.\ A[a/i])$

Location Substitution

- Visualization of an assertion A

  ---X---X----
- Consider a “pure” arithmetic expression

  A[a/X] ---a---a---

Example Assertions

- i is a prime number
- i is the least common multiple of j and k

Semantics of Assertions

- An interpretation $I:intvar \rightarrow N$
- The meaning of Aexpv
  - $Av[\![n]\!]I\sigma=n$
  - $Av[\![X]\!]I\sigma= \sigma(X)$
  - $Av[\![i]\!]I\sigma= I(i)$
  - $Av[\![a0+a1]\!]\ I\sigma = Av[\![a0]\!]I\sigma +Av\ [\![a1]\!]\ I\sigma$
  - …
- For all $a \in$ Aexp states σ and Interpretations I
  - $A[\![a]\!]\sigma=Av[\![a]\!]I\sigma$

Semantics of Assertions (II)

- $I[n/i]$ change i in I to n
- For I and $\sigma \in \Sigma_{\perp}$, define $\sigma \models^{I} A$ by structural induction
  - $\sigma \models^{I}$ true
  - $\sigma \models^{I} (a_0 = a_1)$ if $Av[\![a_0]\!] I\sigma = Av[\![a_1]\!] I\sigma$
  - $\sigma \models^{I} (A \wedge B)$ if $\sigma \models^{I} A$ and $\sigma \models^{I} B$
  - $\sigma \models^{I} \neg A$ if not $\sigma \models^{I} A$
  - $\sigma \models^{I} A \Rightarrow B$ if (not $\sigma \models^{I} A$) or $\sigma \models^{I} B$)
  - $\sigma \models^{I} \forall i.A$ if $\sigma \models^{I[n/i]} A$ for all $n \in N$
  - $\perp \models A$

Proposition 6.4

For all $b \in$ Bexp states σ and Interpretations I

$$\mathcal{B}[\![b]\!]\sigma = \text{true} \quad \text{iff} \quad \sigma \models^I b$$

$$\mathcal{B}[\![b]\!]\sigma = \text{false} \quad \text{iff not} \quad \sigma \models^I b$$

Partial Correctness Assertions

- $\{P\}c\{Q\}$
  - $P, Q \in Assn$ and $c \in Com$
- For a state $\sigma \in \Sigma_\perp$ and interpretation I
  - $\sigma \models^I \{P\}c\{Q\}$ if $(\sigma \models^I P \Rightarrow C\,[\![c]\!]\sigma \models^I Q)$
- Validity
  - When $\forall \sigma \in \Sigma_\perp$, $\sigma \models^I \{P\}c\{Q\}$ we write
    - $\models^I \{P\}c\{Q\}$
  - When $\forall \sigma \in \Sigma_\perp$, and I $\sigma \models^I \{P\}c\{Q\}$ we write
    - $\models \{P\}c\{Q\}$
    - $\{P\}c\{Q\}$ is valid

The extension of an assertion

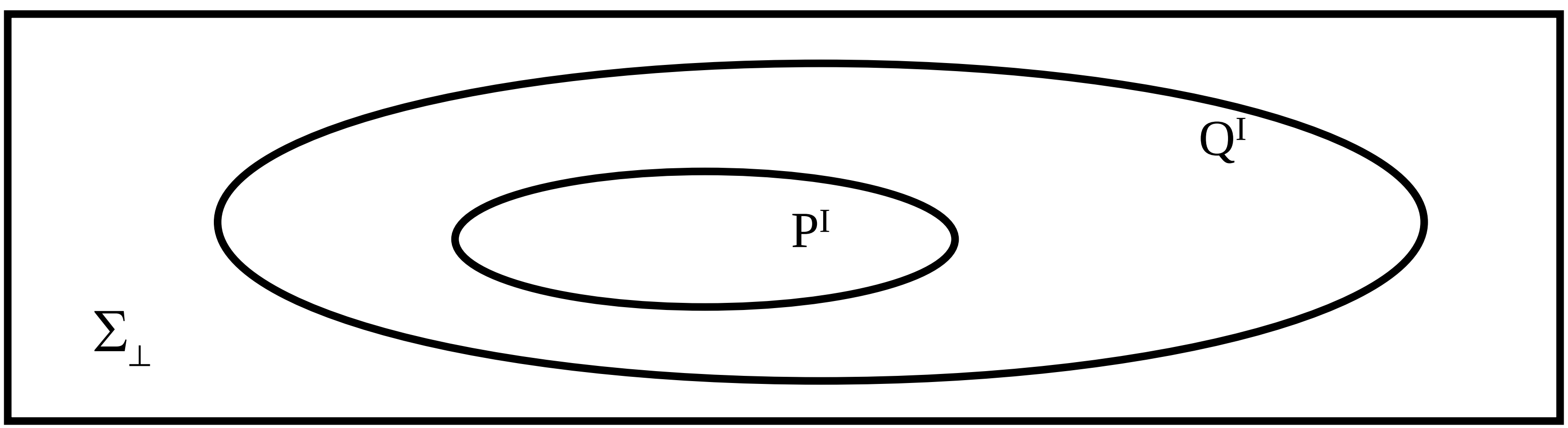
$$A^{I} \doteq \{\sigma \in \Sigma_{\perp} \mid \sigma \models^{I} A\}$$

The extension of assertions

Suppose that $\models (P \Rightarrow Q)$

Then for any interpretation I
$\forall \sigma \in \Sigma_\perp.\ \sigma \models^I P \Rightarrow \sigma \models^I Q$

$P^I \subseteq Q^I$

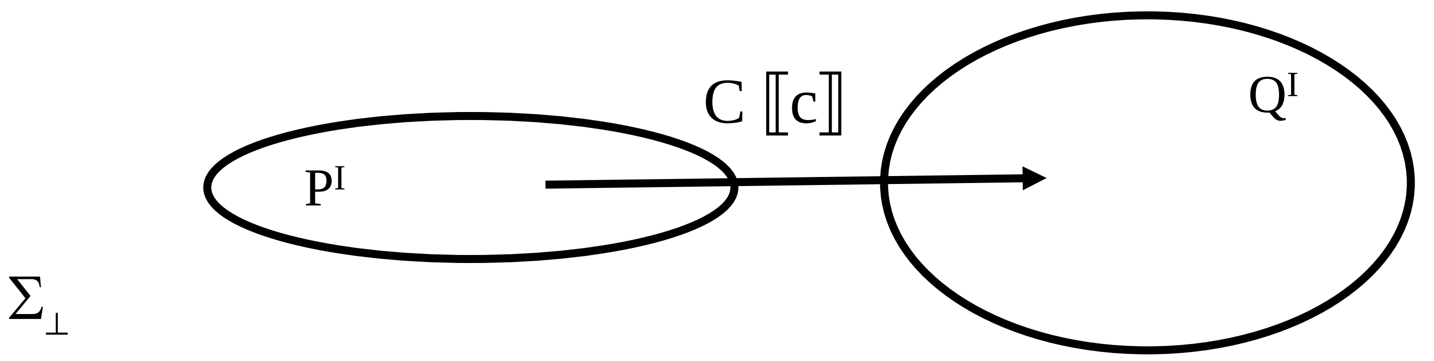


The extension of assertions

Suppose that $\models \{P\}c\{Q\}$

Then for any interpretation I
$\forall \sigma \in \Sigma_\perp .\ \sigma \models^I P \Rightarrow \mathcal{C}[\![c]\!]\sigma \models^I Q$

$\mathcal{C}[\![c]\!]P^I \subseteq Q^I$



Hoare Proof Rules for Partial Correctness

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

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

$$\frac{\{P\}\ S_0\ \{C\}\quad \{C\}\ S_1\ \{Q\}}{\{P\}\ S_0;S_1\{Q\}}$$

$$\frac{\{P \wedge b\}\ S_0\ \{Q\}\quad \{P \wedge \neg b\}\ S_1\ \{Q\}}{\{P\}\ \text{if } b \text{ then } S_0 \text{ else } S_1\{Q\}}$$

$$\frac{\{I \wedge b\}\ S\ \{I\}}{\{I\}\ \text{while } b \text{ do } S\{I \wedge \neg b\}}$$

$$\frac{\models P \Rightarrow P' \quad \{P'\}\ S\ \{Q'\}\quad \models Q' \Rightarrow Q}{\{P\}\ S\ \{Q\}}$$

Example

$\{X = n \wedge n \geq 0\}$

$Y := 1;$

$\{X = n \wedge Y{=}1 \wedge n \geq 0\}$

while $X > 0$ do

$Y := X \times Y;$

$X := X - 1$

$\{Y = n!\}$

Example

$\{X = n \wedge n \geq 0\}$

Y := 1;

$\{X = n \wedge Y=1 \wedge n \geq 0\}$

while X > 0 do $\{X \geq 0 \wedge n \geq 0 \wedge Y=n!/X!\}$

$\{X > 0 \wedge n \geq 0 \wedge Y=n!/X!\}$

Y := X $\times$Y;

$\{X > 0 \wedge n \geq 0 \wedge Y=n!/(X-1)!\}$

X := X – 1

$\{X > 0 \wedge n \geq 0 \wedge Y=n!/X!\}$

$\{Y = n!\}$

Example Formal

$\{X = n \wedge n \geq 0\}$ Y :=1 $\{X = n \wedge Y=1 \wedge n \geq 0\}$

$\{X = n \wedge n \geq 0\}$ Y :=1 $\{X \geq 0 \wedge n \geq 0 \wedge Y=n!/X!\}$

$\{X > 0 \wedge n \geq 0 \wedge Y=n!/X!\}$ Y := X ×Y; $\{X > 0 \wedge n \geq 0 \wedge Y=n!/(X-1)!\}$

$\{X > 0 \wedge n \geq 0 \wedge Y=n!/(X-1)!\}$ X := X-1; $\{X \geq 0 \wedge n \geq 0 \wedge Y=n!/X!\}$

$\{X > 0 \wedge n \geq 0 \wedge Y=n!/X!\}$ Y := X ×Y; X := X-1 $\{X \geq 0 \wedge n \geq 0 \wedge Y=n!/X!\}$

$\{X \geq 0 \wedge n \geq 0 \wedge Y=n!/X! \wedge X>0\}$ Y := X ×Y; X := X-1 $\{X \geq 0 \wedge n \geq 0 \wedge Y=n!/X!\}$

$\{X \geq 0 \wedge n \geq 0 \wedge Y=n!/X!\}$ while X > 0 do Y := X ×Y; X := X-1
$\{X \geq 0 \wedge n \geq 0 \wedge Y=n!/X! \wedge \neg X > 0\}$

$\{X \geq 0 \wedge n \geq 0 \wedge Y=n!/X!\}$ while X > 0 do Y := X ×Y; X := X-1 $\{Y=n!\}$

$\{X=n \wedge n \geq 0\}$ Y :=1; while X > 0 do Y := X ×Y; X := X-1 $\{Y=n!\}$

Soundness

- 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)

Unsound Proof Rules for Partial Correctness

$$\{A\}\ \text{skip}\ \{B\}$$

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

$$\frac{\{P\}\ S_0\ \{C1\}\ \{C2\}\ S_1\ \{Q\}}{\{P\}\ S_0;S_1\{Q\}}$$

$$\frac{\{P\wedge\neg b\}\ S_0\ \{Q\}\ \{P\wedge b\}\ S_1\ \{Q\}}{\{P\}\ \text{if } b \text{ then } S_0 \text{ else } S_1\{Q\}}$$

$$\frac{\{I\wedge b\}\ S\ \{I\}}{\{I\}\ \text{while } b \text{ do } S\{I\wedge\neg b\}}$$

$$\frac{\models P' \Rightarrow P\ \ \{P'\}\ S\ \{Q'\}\ \models Q \Rightarrow Q'}{\{P\}\ S\ \{Q\}}$$

Incomplete Proof Rules for Partial Correctness

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

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

$$\frac{\{P\}\ S_0\ \{C\}\ \{C\}\ S_1\ \{Q\}}{\{P\}\ S_0;S_1\{Q\}}$$

$$\frac{\{P\}\ S_0\ \{Q\}\ \{P\ \}\ S_1\ \{Q\}}{\{P\}\ \text{if b then}\ S_0\ \text{else}\ S_1\{Q\}}$$

$$\frac{\{I\}\ S\ \{I\}}{\{I\}\ \text{while b do}\ S\{I\}}$$

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
  $\models \{I\}\ \text{while } b \text{ do } S\ \{I \wedge \neg b\}$

Soundness of the consequence rule

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

Extensions to While

- Other control flow constructs
- Abort statement (like C exit w/o return value)
- Non determinism
- Parallelism
- Local Variables
- Procedures

(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? $\models \{\text{true}\}\ \text{skip}\ \{Q\}$
- What does $\models \{\text{true}\}\ c\ \{\text{false}\}$ mean?

Relative Completeness (Chapter 7)

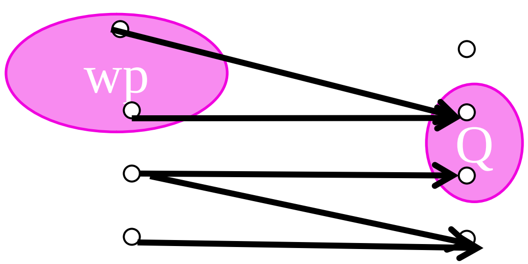
- Assume that every math theorem can be proved
  $\models \{P\}\ S\ \{Q\}$ implies $\vdash \{P\}\ S\ \{Q\}$

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 \{C\}
  and
  $\models$\{I\} S_1 \{Q\}

Weakest Precondition

- wp: Stm $\rightarrow$(Ass$\rightarrow$Ass)
- wp $[\![S]\!](Q)$ – the weakest condition such that every terminating computation of S results in a state satisfying Q
- $\sigma \models \text{wp } [\![S]\!](Q) \leftrightarrow \quad \forall \sigma' : \sigma\ [\![S]\!]\ \sigma' \rightarrow \sigma' \models Q$



- Can be used to compute verification conditions

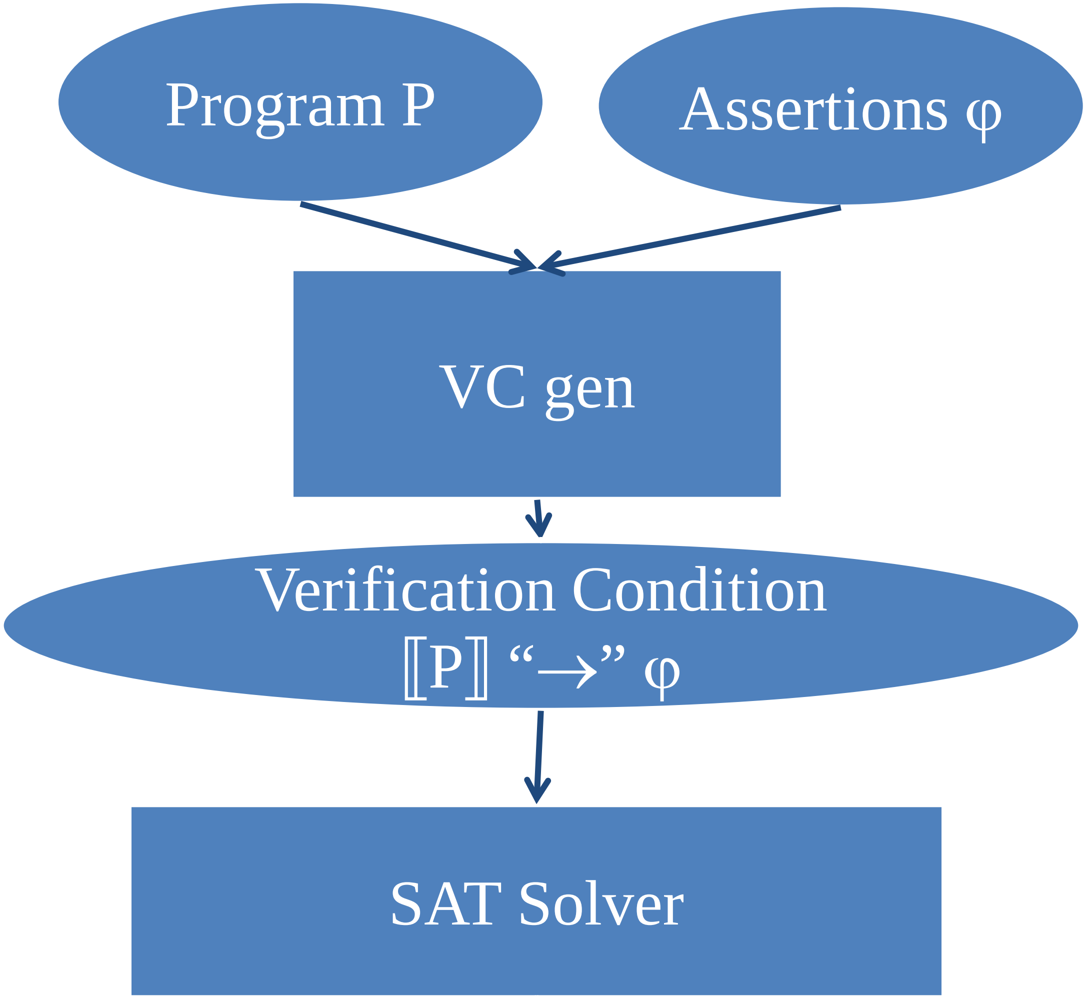
Weakest (Liberal) Precondition

- wp(S, Q) – the weakest condition such that every terminating computation of S results in a state satisfying Q
- $[\![wp^l(S, Q)]\!] = \{\sigma \in \Sigma^\perp \mid S[\![S]\!]\sigma \models^l Q\}$

Some WP rules

- wp(S, false) =
- 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(while B {I} do S, Q) =

Verification Process



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#

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)
- $VC_{aux}(S_1; S_2, Q) =$
  $VC_{aux}(S_1, wp(S_2, Q)) \cup VC_{aux}(S_2, Q)$
- $VC_{aux}(\text{if } C \text{ then } S_1 \text{ else } S_2, Q) =$
  $VC_{aux}(S_1, Q) \cup VC_{aux}(S_2, Q)$
- $VC_{aux}(\text{while } B \text{ do } \{I\}\ S, Q) = VC_{aux}(S, I) \cup$
  $\{I \wedge [\![B]\!] \rightarrow wp[\![S]\!](I)\} \cup$
  $\{I \wedge \neg[\![B]\!] \rightarrow Q\}$

Summary

- Axiomatic semantics provides an abstract semantics
- Can be used to explain programming
- Can be automated
- More effort is required to make it practical