Programming Language Semantics
Axiomatic Semantics

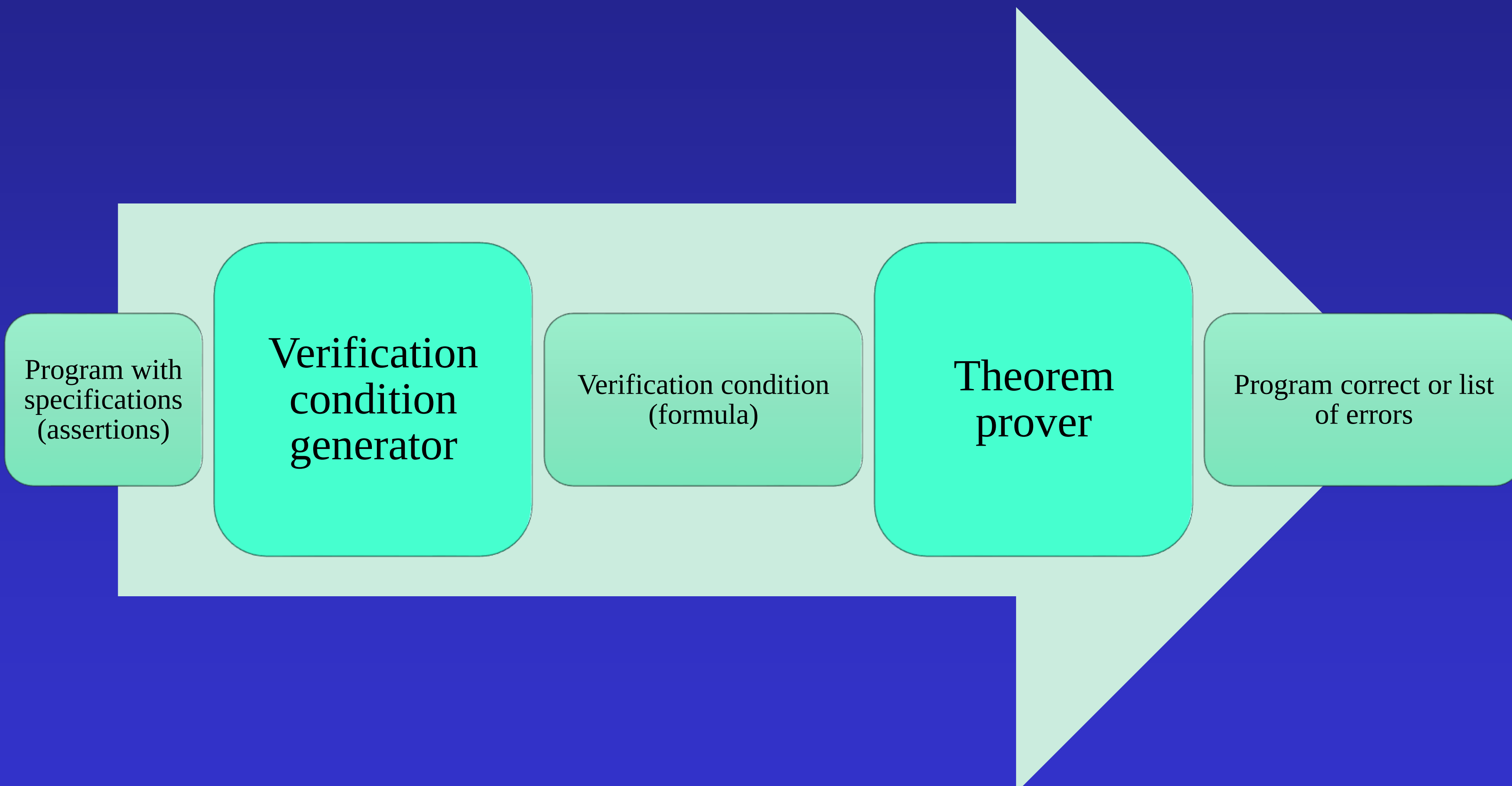
Chapter 6

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

Basic Verifier Architecture



Plan

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

Example Program

```
S:=0
N := 1
while ¬(N=101) do
        S := S + N ;
        N :=N+1
```

$N=101$

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

Example Program

S:=0

{S=0}

N := 1

{S=0 ∧N=1}

while ¬(N=101) do

    S := S + N ;

    N :=N+1

{N=101 ∧S=$\sum_{1\leq m\leq 100}$ m}

Example Program

$S:=0$

$\{S=0\}$

$N := 1$

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

while $\neg(N=101)$ do

$\quad S := S + N ;$

$\quad N := N+1$

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

Example Program (take 1)

S:=0

$\{S=0\}$

N := 1

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

while $\{1 \leq N \leq 101\}$ $\neg(N=101)$ do

$\{1 \leq N < 101\}$

S := S + N ;

$\{1 \leq N < 101\}$

N :=N+1

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

Example Program (take 2)

S:=0

$\{S=0\}$

N := 1

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

while $\{1 \leq N \leq 101 \wedge S=\sum_{1\leq m\leq 100} m\}$ $\neg(N=101)$ do

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

S := S + N ;

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

N :=N+1

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

Example Program (take 3)

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

$\{1 \leq N < 101 \wedge S=\sum 1\leq m\leq N\text{-}1\ m\}$

S := S + N ;

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

N :=N+1

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

Another Example

```
X := 1
Y: = 2;
while true {odd(X) } do
    X := X +  Y  ;
    Y := Y + 2
```

Ghost Variables

$\{P\}$
$X := X + 5$
$\{Q\}$

Partial Correctness

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

Total Correctness

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

Formalizing Partial Correctness

- $\sigma \models A$
  - A is true in σ
- $\{P\}\ c\ \{Q\}$
  - $\forall \sigma, \sigma' \in \Sigma.\ (\sigma \models P\ \&\ <c, \sigma> \rightarrow \sigma'\ ) \Rightarrow \sigma' \models Q$
  - $\forall \sigma \in \Sigma.\ (\sigma \models P\ \&\ \mathbf{C}\ [\![c]\!]\sigma \neq \bot) \Rightarrow \mathbf{C}\ [\![c]\!]\sigma \models Q$
- Convention for all A

  $\bot \models A$
- $\forall \sigma, \sigma' \in \Sigma.\ \sigma \models P\ \Rightarrow \mathbf{C}\ [\![c]\!]\sigma \models Q$

The Assertion Language

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

The 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 $M := M - N$

Free and Bound Variables

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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 $\sigma \models^{I} B$)
  - $\sigma \models^{I} \forall i A$ $\sigma \models^{I[n/i]} A$ 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

- $\{P\}c\{Q\}$
  - $P, Q \in \mathrm{Assn}$ and $c \in \mathrm{Com}$
- 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

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

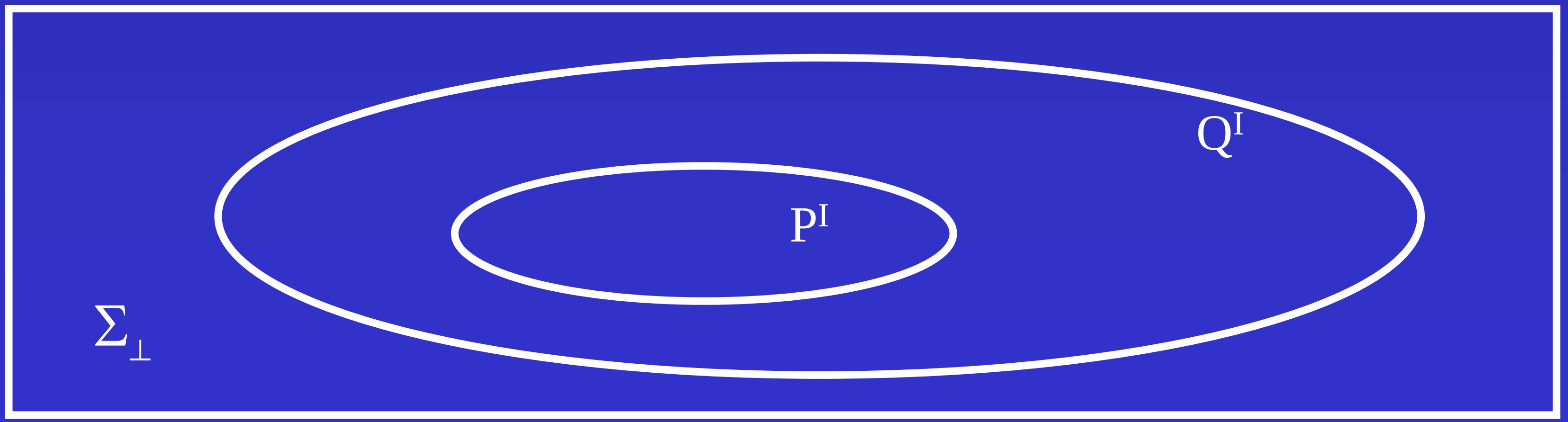
The extension of assertions

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

Then for any interpretation I

$\forall \sigma \in \Sigma_\perp .\ \sigma \models^I P \Rightarrow \sigma \models^I Q$

$P^I \subseteq Q^I$



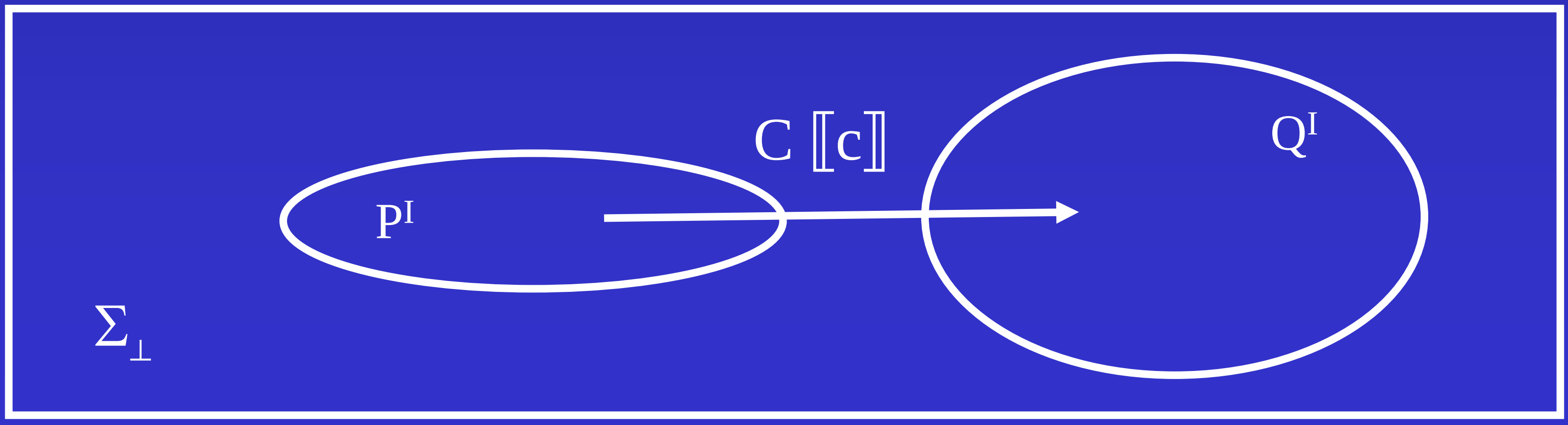
The extension of assertions

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 (take 1)

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

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

$$\frac{\{P\}\ c_0\ \{C\}\quad \{C\}\ c_1\ \{Q\}}{\{P\}\ c_0;c_1\{Q\}}$$

$$\frac{\{P\wedge b\}\ c_0\ \{Q\}\quad \{P\wedge\neg b\}\ c_1\ \{Q\}}{\{P\}\ \text{if } b \text{ then } c_0 \text{ else } c_1\{Q\}}$$

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

Hoare Proof Rules for Partial Correctness

$$\{A\}\ \mathrm{skip}\ \{A\}$$

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

$$\frac{\{P\}\ c_0\ \{C\}\quad \{C\}\ c_1\ \{Q\}}{\{P\}\ c_0;c_1\{Q\}}$$

$$\frac{\{P\wedge b\}\ c_0\ \{Q\}\quad \{P\wedge\neg b\}\ c_1\ \{Q\}}{\{P\}\ \mathrm{if}\ b\ \mathrm{then}\ c_0\ \mathrm{else}\ c_1\{Q\}}$$

$$\frac{\{I\wedge b\}\ c\ \{I\}}{\{I\}\ \mathrm{while}\ b\ \mathrm{do}\ c\{I\wedge\neg b\}}$$

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

Example

```
while X > 0 do
        Y := X ×Y;
        X := X – 1
```

Incomplete Proof Rules for Partial Correctness

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

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

$$\frac{\{P\}\ c_0\ \{C\}\ \ \{C\}\ c_1\ \{Q\}}{\{P\}\ c_0;c_1\{Q\}}$$

$$\frac{\{P\}\ c_0\ \{Q\}\ \ \{P\}\ c_1\ \{Q\}}{\{P\}\ \text{if } b \text{ then } c_0 \text{ else } c_1\{Q\}}$$

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

$$\frac{\models P\ \ \Leftrightarrow\Rightarrow P'\ \ \{P'\}\ c\ \{Q'\}\ \models Q' \Rightarrow Q}{\{P\}\ c\ \{Q\}}$$

Unsound Proof Rules for Partial Correctness

$$\{A\}\ \mathrm{skip}\ \{B\}$$

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

$$\frac{\{P\}\ c_1\ \{C\}\ \ \{C\}\ c_0\ \{Q\}}{\{P\}\ c_0;c_1\{Q\}}$$

$$\frac{\{P\wedge b\}\ c_0\ \{Q\}\ \ \{P\wedge\neg b\}\ c_1\ \{Q\}}{\{P\}\ \mathrm{if}\ b\ \mathrm{then}\ c_0\ \mathrm{else}\ c_1\{Q\}}$$

$$\{I\}\ \mathrm{while}\ b\ \mathrm{do}\ c\{I\wedge\neg b\}$$

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

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)
  - Requires user assistance
- Proof of soundness
  - Every rule preserves validity (Theorem 6.1)

Completeness

- Every valid theorem can be derived by the rule system is valid
  - $\models \{P\}\ c\ \{Q\} \Rightarrow\ \vdash \{P\}\ c\ \{Q\}$
- But what about Gödel's incompleteness?
- Relative completeness
  - Assume that every math theorem is valid
- Chapter 7
  - Uses Weakest Preconditions

Parallelism

Summary

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