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

1. Program with specifications (assertions)
2. Verification condition generator
3. Verification condition (formula)
4. Theorem prover
5. Program correct or list of errors
Plan

• The basic idea
• An assertion language
• Semantics of assertions
• Proof rules
• An example
• Soundness
• Completeness
Example Program

\[ S := 0 \]
\[ N := 1 \]

while \( \neg (N=101) \) 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 \land N=1 \}

while \( \neg (N=101) \) do

\[ S := S + N ; \]

\[ N := N+1 \]

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

\[
S := 0
\]
\[
\{ S = 0 \}
\]

\[
N := 1
\]
\[
\{ S = 0 \land N = 1 \}
\]

while \( \neg (N = 101) \) do

\[
S := S + N ;
\]

\[
N := N + 1
\]

\[
\{ N = 101 \land S = \sum_{1 \leq m \leq 100} m \}
\]
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)

\( S := 0 \)

\( \{ S=0 \} \)

\( N := 1 \)

\( \{ S=0 \land N=1 \} \)

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

\( \{ 1 \leq N < 101 \land S=\sum_{1 \leq m \leq 100} m \} \)

\( S := S + N \)

\( \{ 1 \leq N < 101 \land S=\sum_{1 \leq m \leq 100} m \} \)

\( N := N + 1 \)

\( \{ N=101 \land S=\sum_{1 \leq m \leq 100} m \} \)
Example Program (take 3)

\[ S := 0 \]
\[ \{S = 0\} \]
\[ N := 1 \]
\[ \{S = 0 \land N = 1\} \]

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

\[ \{1 \leq N < 101 \land S = \sum_{1 \leq m \leq N-1} m\} \]

\[ S := S + N ; \]

\[ \{1 \leq N < 101 \land S = \sum_{1 \leq m \leq N} m\} \]

\[ N := N + 1 \]

\[ \{N = 101 \land S = \sum_{1 \leq m \leq 100} m\} \]
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 \(\sigma\) which satisfies P, if the execution of c from state \(\sigma\) terminates in state \(\sigma'\), then \(\sigma'\) 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 \(\sigma\) which satisfies P,
    • the execution of c from state \(\sigma\) must terminates in a state \(\sigma'\)
    • \(\sigma'\) satisfies Q
Formalizing Partial Correctness

• $\sigma \models A$
  – $A$ is true in $\sigma$

• $\{P\} \cdash \{Q\}$
  – $\forall \sigma, \sigma' \in \Sigma. (\sigma \models P \& \langle c, \sigma \rangle \to \sigma') \Rightarrow \sigma' \models Q$
  – $\forall \sigma \in \Sigma. (\sigma \models P \& C \llbracket c \rrbracket \sigma \neq \bot) \Rightarrow C \llbracket c \rrbracket \sigma \models Q$

• Convention for all $A$
  $\bot \models A$

• $\forall \sigma, \sigma' \in \Sigma. \sigma \models P \Rightarrow C \llbracket c \rrbracket \sigma \models Q$
The Assertion Language

- Extend Bexp
- Allow quantifications
  - ∀i: ...
  - ∃i: ...
    - ∃i. k=i×1
- Import well known mathematical concepts
  - n! = n ×(n-1) × · · · 2 ×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

\[ A := \text{true} \mid \text{false} \mid a_0 = a_1 \mid a_0 \leq a_1 \mid A_0 \land A_1 \mid A_0 \lor A_1 \mid \neg A \mid \]

\[ A_0 \Rightarrow A_1 \mid \forall i. A \mid \exists i. A \]
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 $\exists i. k = i \times L \quad (i + 100 \leq 77) \land \forall i. j + 1 = i + 3$

FV(n) = FV(X) = ∅

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) = ∅

FV(a_0 = a_1) = FV(a_0 \leq a_1) = FV(a_0) \cup FV(a_1)

FV(A_0 \land A_1) = FV(A_0 \lor 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 $A$
  
  - Consider a “pure” arithmetic expression
    
    $A[a/i] = \ldots a \ldots a$

\begin{align*}
  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]
\end{align*}
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 \land A_1)[a/i] = (A_0[a/i] \land A_1[a/i])  (A_0 \lor A_1)[a/i] = (A_0[a/i] \lor 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
  
  $A[a/X] \quad ---a---a---$
Example Assertions

• i is a prime number
• i is the least common multiple of j and k
Semantics of Assertions

- An **interpretation** $I: \text{intvar} \rightarrow \mathbb{N}$
- The meaning of $A\text{expv}$
  - $Av[n]I\sigma = n$
  - $Av[X]I\sigma = \sigma(X)$
  - $Av[i]I\sigma = I(i)$
  - $Av[a0+a1]I\sigma = Av[a0]I\sigma + Av[a1]I\sigma$
  - $\ldots$
- For all $a \in A\text{exp}$ states $\sigma$ 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 \( \sigma \in \Sigma_{\bot} \), define \( \sigma \models^I A \) by structural induction
  
  - \( \sigma \models^I \text{true} \)
  - \( \sigma \models^I (a_0 = a_1) \) if \( \text{Av}[[a_0]] I\sigma = \text{Av}[[a_1]] I\sigma \)
  - \( \sigma \models^I (A \land B) \) 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 \( n \in \mathbb{N} \)
  - \( \bot \models^I A \)
Proposition 6.4

For all \( b \in \text{Bexp} \) states \( \sigma \) and Interpretations \( I \)

\[
\begin{align*}
\text{B}[b]\sigma &= \text{true} \quad \text{iff} \quad \sigma \models^I b \\
\text{B}[b]\sigma &= \text{false} \quad \text{iff} \quad \text{not } \sigma \models^I b
\end{align*}
\]
Partial Correctness Assertions

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

- For a state \(\sigma \in \Sigma_\perp\) and interpretation I
  - \(\sigma \models^I \{P\}c\{Q\}\) if \((\sigma \models^I P \implies C \llbracket c \rrbracket \sigma \models^I Q)\)

- 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

\[ A^I = \{ \sigma \in \Sigma^I \mid \sigma \models^I A \} \]
The extension of assertions

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

Then for any interpretation $I$
$\forall \sigma \in \Sigma_{\bot}, \sigma \models^I P \Rightarrow \sigma \models^I Q$

$P^I \subseteq Q^I$
The extension of assertions

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

Then for any interpretation $I$

$\forall \sigma \in \Sigma_\perp. \sigma \models^I P \Rightarrow C [c] \sigma \models^I Q$

$C [c] P^I \subseteq Q^I$
Hoare Proof Rules for Partial Correctness (take 1)

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

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

\text{\{P\} } c_0 \{C\} \{C\} c_1 \{Q\}
\{P\} c_0;c_1\{Q\}

\text{\{P}\land b\} } c_0 \{Q\} \{P\land \neg b\} c_1 \{Q\}
\{P\} \text{if } b \text{ then } c_0 \text{ else } c_1\{Q\}

\text{\{I\land b\} } c \{I\}
\{I\} \text{while } b \text{ do } c\{I\land \neg b\}
Hoare Proof Rules for Partial Correctness

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

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

\{P\} c_0 \{C\} \{C\} c_1 \{Q\}

\{P\} c_0;c_1\{Q\}

\{P \land b\} c_0 \{Q\} \{P \land \neg b\} c_1 \{Q\}

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

\{I \land b\} \ c \{I\}

\{I\} \text{while } b \text{ do } c\{I \land \neg b\}

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

\{P\} \ c \{Q\}
Example

while $X > 0$ do

    $Y := X \times Y$;

    $X := X - 1$
Incomplete Proof Rules for Partial Correctness

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

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

\{P\} c_0 \{C\} \{C\} c_1 \{Q\}

\{P\} c_0;c_1\{Q\}

\{P\} c_0\{Q\} \{P\} c_1\{Q\}

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

\{I\} c \{I\}

\{I\} \text{while } b \text{ do } c\{I \land \neg b\}

\models P \iff P' \{P'\} c \{Q'\} \models Q' \Rightarrow Q

\models P \models Q
Unsound Proof Rules for Partial Correctness

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

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

\{P\} c_1 \{C\} \{C\} c_0 \{Q\}

\{P\} c_0;c_1\{Q\}

\{P\wedge b\} c_0 \{Q\} \{P \wedge \neg b\} c_1 \{Q\}

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

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

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

\{P\} c \{Q\}
Soundness

• Every theorem obtained by the rule system is valid
  – $\vdash \{P\} \subset \{Q\} \Rightarrow \models \{P\} \subset \{Q\}$
• 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
  \[ \models \{P\} \subset \{Q\} \implies \vdash \{P\} \subset \{Q\} \]

• 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