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

\begin{align*}
S &:= 0 \\
N &:= 1 \\
\text{while } \neg (N=101) \text{ do} \\
& \quad S := S + N \ ; \\
& \quad N := N + 1 \\
& \quad N = 101 \\
& \quad S = \sum_{1 \leq m \leq 100} m
\end{align*}
Example Program

\[
S := 0 \\
\{S = 0\} \\
N := 1 \\
\{S = 0 \land N = 1\} \\
\text{while } \neg (N = 101) \text{ do} \\
\quad S := S + N ; \\
\quad N := N + 1 \\
\{N = 101 \land S = \sum_{1 \leq m \leq 100} m\}
\]
Example Program

\[ S := 0 \]

\[ \{ S = 0 \} \]

\[ N := 1 \]

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

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

\[ S := S + N ; \]

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

\[ N := N + 1 \]

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

- \{P\}S\{Q\}
  - P and Q are assertions (extensions of Boolean expressions)
  - S is a statement
  - For all states \(\sigma\) which satisfies P, if the execution of S from state \(\sigma\) terminates in state \(\sigma'\), then \(\sigma'\) 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 \(\sigma\) which satisfies P,
    • the execution of S from state \(\sigma\) must terminates in a state \(\sigma'\)
    • \(\sigma'\) satisfies Q
Formalizing Partial Correctness

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

• $\{P\} S \{Q\}$
  - $\forall \sigma, \sigma' \in \Sigma. (\sigma \models P & <S, \sigma> \rightarrow \sigma' ) \Rightarrow \sigma' \models Q$
  - $\forall \sigma \in \Sigma. (\sigma \models P & S \llbracket S \rrbracket \sigma \neq \bot) \Rightarrow S \llbracket S \rrbracket \sigma \models Q$

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

• $\forall \sigma, \sigma' \in \Sigma. \sigma \models P \Rightarrow S \llbracket S \rrbracket \sigma \models Q$
An Assertion Language

• Extend Bexp

• Allow quantifications
  – ∀i: ...
  – ∃i: ...
    • ∃i. k=i×1

• Import well known mathematical concepts
  – n! = n ×(n-1) × ··· 2 ×1
Assertion Language

\[
A_{expv}
\]

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

\[
Assn
\]

\[
A := \text{true} \mid \text{false} \mid a_0 = a_1 \mid a_0 \leq a_1 \mid A_0 \land A_1 \mid A_0 \lor A_1 \mid \neg A \mid A_0 \Rightarrow A_1 \mid \forall i. A \mid \exists i. A
\]
Example

while \neg (M=N) do
    if M \leq N
        then N := N - M
    else M := M - N
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: $\exists i. \ k = i \times L \ (i + 100 \leq 77) \land \forall i. j + 1 = i + 3$

FV(n) = FV(X) = $\emptyset$

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) = $\emptyset$

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 $A[a/i]$ 

\[
\begin{align*}
n[a/i] &= n & X[a/i] &= X \\
i[a/i] &= a & j[a/i] &= j \\
(a_0 + a_1)[a/i] &= a_0[a/i] + a_1[a/i] & (a_0 - a_1)[a/i] &= a_0[a/i] - a_1[a/i] \\
(a_0 \times a_1)[a/i] &= a_0[a/i] \times a_1[a/i]
\end{align*}
\]
Substitution

- Visualization of an assertion A

- Consider a “pure” arithmetic expression
  $A[a/i] \rightleftharpoons a \rightleftharpoons a$

true[$a/i$] = true
false[$a/i$] = false

$(a_0 = a_1)[a/i] = (a_0/[a/i] = a_1[a/i])$
$(a_0 \leq a_1)[a/i] = (a_0/[a/i] \leq a_1[a/i])$
$(A_0 \land A_1)[a/i] = (A_0[a/i] \land A_1[a/i])$
$(A_0 \lor A_1)[a/i] = (A_0[a/i] \lor A_1[a/i])$
$(A_0 \Rightarrow A_1)[a/i] = (A_0[a/i] \Rightarrow A_1[a/i])[a/i]$
$(\neg A)[a/i] = \neg(A[a/i])$

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

$(\forall j. A)[a/i] = (\forall j. A[a/i])$
$(\exists j. A)[a/i] = (\exists j. A[a/i])$
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 $I:\text{intvar} \to \mathbb{N}$

• The meaning of $A\exp v$
  
  – $A\exp[n]I_\sigma = n$
  
  – $A\exp[X]I_\sigma = \sigma(X)$
  
  – $A\exp[i]I_\sigma = I(i)$
  
  – $A\exp[a0+a1]I_\sigma = A\exp[a0]I_\sigma + A\exp[a1]I_\sigma$
  
  – ...

• For all $a \in A\exp$ states $\sigma$ and Interpretations $I$
  
  – $A[a]_\sigma = A\exp[a]I_\sigma$
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 $\text{Av}[a_0] \models_I \sigma = \text{Av}[a_1] \models_I \sigma$
  - $\sigma \models_I (A \land B)$ 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 $\sigma \models_I B$
  - $\sigma \models_I \forall i. A$ if $\sigma \models_I[n/i] A$ for all $n \in \mathbb{N}$
  - $\bot \models A$
Proposition 6.4

For all $b \in B\text{exp}$ states $\sigma$ and Interpretations $I$

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

$B[b]\sigma = \text{false}$  iff not $\sigma \models^I b$
Partial Correctness Assertions

• \{P\}c\{Q\}
  – \(P, Q \in \text{Assn} \) and \(c \in \text{Com}\)

• For a state \(\sigma \in \Sigma_{\bot}\) and interpretation \(I\)
  – \(\sigma \models^I \{P\}c\{Q\}\) if (\(\sigma \models^I P \Rightarrow C \llbracket c \rrbracket \sigma \models^I Q\))

• Validity
  – When \(\forall \sigma \in \Sigma_{\bot}, \sigma \models^I \{P\}c\{Q\}\) we write
    • \(\models^I \{P\}c\{Q\}\)
  – When \(\forall \sigma \in \Sigma_{\bot}, \text{and} I \sigma \models^I \{P\}c\{Q\}\) we write
    • \(\models \{P\}c\{Q\}\)
    • \(\{P\}c\{Q\}\) is valid
The extension of an assertion

\[ A^1 = \{ \sigma \in \Sigma_{\perp} \mid \sigma \models^1 A \} \]
The extension of assertions

Suppose that \( \vdash (P \Rightarrow Q) \)

Then for any interpretation \( I \)
\[
\forall \sigma \in \Sigma_I. \sigma \vdash^I P \Rightarrow \sigma \vdash^I Q
\]

\( P^I \subseteq Q^I \)
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

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

\{B[a/X]\} \ X:=a \ {B}\n
\{P\} \ S_0 \ {C} \ {C} \ S_1 \ {Q}\n
\{P\} \ S_0;S_1 \ {Q}\n
\{P \land b\} \ S_0 \ {Q} \ \{P \land \neg b\} \ S_1 \ {Q}\n
\{P\} \ \text{if} \ b \ \text{then} \ S_0 \ \text{else} \ S_1 \ {Q}\n
\{I \land b\} \ S \ {I}\n
\{I\} \ \text{while} \ b \ \text{do} \ S \{I \land \neg b\}

\models P \Rightarrow P' \ {P'} \ S \ {Q'} \ \models Q' \Rightarrow Q \n
\{P\} \ S \ {Q}\
Example

{\(X = n \land n \geq 0\)}

\(Y := 1;\)

{\(X = n \land Y=1 \land n \geq 0\)}

while \(X > 0\) do

\(Y := X \times Y;\)

\(X := X - 1\)

{\(Y = n! \) }
Example

\{X = n \land n \geq 0\}

\[Y := 1;\]

\{X = n \land Y=1 \land n \geq 0\}

while \(X > 0\) do

\{X \geq 0 \land n \geq 0 \land Y=n!/X!\}

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

\[Y := X \times Y;\]

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

\[X := X - 1\]

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

\{Y = n! \}
Example Formal

\{X = n \land n \geq 0\} \ Y := 1 \ \{X = n \land Y = 1 \land n \geq 0\}

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

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

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

\{X > 0 \land n \geq 0 \land Y = n!/X!\} \ Y := X \times Y; \ X := X - 1 \ \{X \geq 0 \land n \geq 0 \land Y = n!/X!\}

\{X \geq 0 \land n \geq 0 \land Y = n!/X! \land X > 0\} \ Y := X \times Y; \ X := X - 1 \ \{X \geq 0 \land n \geq 0 \land Y = n!/X!\}

\{X \geq 0 \land n \geq 0 \land Y = n!/X!\} \ while \ X > 0 \ do \ Y := X \times Y; \ X := X - 1
\{X \geq 0 \land n \geq 0 \land Y = n!/X! \land \neg X > 0\}

\{X \geq 0 \land n \geq 0 \land Y = n!/X!\} \ while \ X > 0 \ do \ Y := X \times Y; \ X := X - 1 \ \{Y = n!\}

\{X = n \land n \geq 0\} \ Y := 1; \ while \ X > 0 \ do \ Y := X \times Y; \ X := X - 1 \ \{Y = n!\}
Soundness

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

• 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]\}

\{P\} \ S_0 \ \{C1\} \ \{C2\} \ S_1 \ \{Q\}

\{P\} \ S_0;S_1 \{Q\}

\{P \land \neg b\} \ S_0 \{Q\} \ \{P \land b\} \ S_1 \{Q\}

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

\{I \land b\} \ S \ \{I\}

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

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

\{P\} \ S \ \{Q\}
Incomplete Proof Rules for Partial Correctness

{A} skip {A}

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

{P} S_0 {C} C S_1 {Q}

{P} S_0;S_1 {Q}

{P} S_0 {Q} {P} S_1 {Q}

{P} if b then S_0 else S_1 {Q}

{I} S {I}

{I} 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
  \[ \vdash \{ P \} S_0 \{ C \} \]
  and
  \[ \vdash \{ C \} S_1 \{ Q \} \]

• Show that
  \[ \vdash \{ P \} S_0; S_1 \{ Q \} \]
Soundness of the conditional rule

• Assume that
  \[ \vdash \{ P \land b \} S_0 \{ Q \} \]
  and
  \[ \vdash \{ P \land \lnot b \} S_1 \{ Q \} \]

• Show that
  \[ \vdash \{ P \} \text{ if } b \text{ then } S_0 \text{ else } S_1 \{ Q \} \]
Soundness of the while rule

• Assume that
  \[ \implies \{ I \land b \} S \{ I \} \]

• Show that
  \[ \implies \{ I \} \text{ while } b \text{ do } S \{ I \land \neg b \} \]
Soundness of the consequence rule

• Assume that
  $\models \neg\{P'\} \ S \ \{Q'\}$
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? $\models \{true\}$ skip $\{Q\}$
• What does $\models \{true\} \ c \ \{false\}$ mean?
Relative Completeness (Chapter 7)

• Assume that every math theorem can be proved

\[ \vdash \{P\} S \{Q\} \implies \not\vdash \{P\} S \{Q\} \]
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

• \( \text{wp: Stm} \rightarrow (\text{Ass} \rightarrow \text{Ass}) \)
• \( \text{wp} \left[ S \right](Q) \) – the weakest condition such that every terminating computation of \( S \) results in a state satisfying \( Q \)
• \( \sigma \models \text{wp} \left[ S \right](Q) \iff \forall \sigma': \sigma \left[ S \right] \sigma' \rightarrow \sigma' \models Q \)

• 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 \( Q \)
- \( \lceil wp^I(S, Q) \rceil = \{ \sigma \in \Sigma^\perp \mid S[\lceil S \rceil] \sigma =^I Q \} \)
Some WP rules

- \( \text{wp}(S, \text{false}) = \) 
- \( \text{wp}(	ext{skip}, Q) = Q \) 
- \( \text{wp}(X := a, Q) = Q[a/X] \) 
- \( \text{wp}(S_0; S_1, Q) = \text{wp}(S_0, \text{wp}(S_1, Q)) \) 
- \( \text{wp}(\text{if } b \text{ then } S_0 \text{ else } S_1, Q) = b \land \text{wp}(S_0, Q) \lor \neg b \land \text{wp}(S_1, Q) \) 
- \( \text{wp}(\text{while } B \{I\} \text{ do } S, Q) = \)
Verification Process

Program P

Assertions \( \varphi \)

VC gen

Verification Condition

\([P] \quad \rightarrow \quad \varphi\)

SAT Solver
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

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

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