1. Review: Partial Correctness
2. Proof Calculus for Total Correctness
3. Programming by Contract
Convert informal description $R$ of requirements for an application domain into formula $\phi_R$. 
Convert informal description $R$ of requirements for an application domain into formula $\phi_R$.

Write program $P$ that meets $\phi_R$. 
Convert informal description $R$ of requirements for an application domain into formula $\phi_R$.

Write program $P$ that meets $\phi_R$.

Prove that $P$ satisfies $\phi_R$. 
Framework for Software Verification

Convert informal description $R$ of requirements for an application domain into formula $\phi_R$.

Write program $P$ that meets $\phi_R$.

Prove that $P$ satisfies $\phi_R$.

Each step provides risks and opportunities.
Expressions come as arithmetic expressions $E$:

\[
E ::= n \mid x \mid (-E) \mid (E + E) \mid (E - E) \mid (E \times E)
\]
Expressions in Core Language

Expressions come as arithmetic expressions $E$:

$$E ::= n \mid x \mid (-E) \mid (E + E) \mid (E - E) \mid (E \times E)$$

and boolean expressions $B$:

$$B ::= \text{true} \mid \text{false} \mid (!B) \mid (B \& B) \mid (B \| B) \mid (E < E)$$
Commands cover some common programming idioms. Expressions are components of commands.

\[ C ::= x = E \mid C; C \mid \text{if } B \{ C \} \text{ else } \{ C \} \mid \text{while } B \{ C \} \]
Example

Consider the factorial function:

\[
0! \quad \overset{\text{def}}{=} \quad 1 \\
(n + 1)! \quad \overset{\text{def}}{=} \quad (n + 1) \cdot n!
\]

We shall show that after the execution of the following Core program, we have \( y = x! \).

\[
y = 1; \\
z = 0; \\
\textbf{while} \; (z \neq x) \; \{ \; z = z + 1; \; y = y \ast z; \; \}
\]
Shape of assertions

\{ \phi \} \ P \ \{ \psi \}
Assertions on Programs

Shape of assertions

\[ \{ \phi \} \ P \ {\psi} \]

Informal meaning

If the program \( P \) is run in a state that satisfies \( \phi \), then the state resulting from \( P \)'s execution will satisfy \( \psi \).
What program $P$ meets this triple

$\{ x > 0 \} P \{ y \cdot y < x \}$

One correct answer: $P =$

```plaintext
y = 0;
while (y * y < x) {
    y = y + 1;
}
y = y - 1;
```
Hoare Triples

**Definition**

An assertion of the form \( \{ \phi \} P \{ \psi \} \) is called a Hoare triple.

- \( \phi \) is called the precondition, \( \psi \) is called the postcondition.
- A state of a Core program \( P \) is a function \( l \) that assigns each variable \( x \) in \( P \) to an integer \( l(x) \).
- A state \( l \) satisfies \( \phi \) if \( M \models l \phi \), where \( M \) contains integers and gives the usual meaning to the arithmetic operations.
- Quantifiers in \( \phi \) and \( \psi \) bind only variables that do not occur in the program \( P \).
Definition

We say that the triple $\{\phi\} \ P \{\psi\}$ is satisfied under partial correctness if, for all states which satisfy $\phi$, the state resulting from $P$’s execution satisfies $\psi$, provided that $P$ terminates.
Partial Correctness

Definition
We say that the triple $\{ \phi \} P \{ \psi \}$ is satisfied under partial correctness if, for all states which satisfy $\phi$, the state resulting from $P$’s execution satisfies $\psi$, provided that $P$ terminates.

Notation
We write $\models_{\text{par}} \{ \phi \} P \{ \psi \}$.
Total Correctness

Definition

We say that the triple $\{\phi\} P \{\psi\}$ is satisfied under total correctness if, for all states which satisfy $\phi$, $P$ is guaranteed to terminate and the resulting state satisfies $\psi$. 
Definition
We say that the triple \( \{ \phi \} P \{ \psi \} \) is satisfied under total correctness if, for all states which satisfy \( \phi \), \( P \) is guaranteed to terminate and the resulting state satisfies \( \psi \).

Notation
We write \( \models_{\text{tot}} \{ \phi \} P \{ \psi \} \).
We are looking for a proof calculus that allows us to establish

\[ \vdash_{\text{par}} \{ \phi \} \quad P \quad \{ \psi \} \]
Strategy

We are looking for a proof calculus that allows us to establish

\[ \vdash_{\text{par}} \{ \phi \} \ P \ \{ \psi \} \]

where

\[ \models_{\text{par}} \{ \phi \} \ P \ \{ \psi \} \] holds whenever \[ \vdash_{\text{par}} \{ \phi \} \ P \ \{ \psi \} \]

(correctness)
We are looking for a proof calculus that allows us to establish

\[ \vdash_{\text{par}} \begin{cases} \phi \end{cases} P \begin{cases} \psi \end{cases} \]

where

\[ \models_{\text{par}} \begin{cases} \phi \end{cases} P \begin{cases} \psi \end{cases} \]

holds whenever \[ \vdash_{\text{par}} \begin{cases} \phi \end{cases} P \begin{cases} \psi \end{cases} \] (correctness), and

\[ \vdash_{\text{par}} \begin{cases} \phi \end{cases} P \begin{cases} \psi \end{cases} \]

holds whenever \[ \models_{\text{par}} \begin{cases} \phi \end{cases} P \begin{cases} \psi \end{cases} \] (completeness).
Rules for Partial Correctness

\[ \{ \phi \} \ C_1 \ {\eta} \quad \{ \eta \} \ C_2 \ {\psi} \]

\[ \frac{}{\{ \phi \} \ C_1 ; C_2 \ {\psi}} \quad \text{[Composition]} \]
Rules for Partial Correctness (continued)

\[
\{ [x \rightarrow E] \psi \} \ x = E \ {\psi}
\]
Rules for Partial Correctness (continued)

\[
\begin{align*}
\{ \phi \land B \} & \quad C_1 \quad \{ \psi \} \\
\{ \phi \land \neg B \} & \quad C_2 \quad \{ \psi \}
\end{align*}
\]

\[\text{[If-statement]}\]

\[
\{ \phi \} \text{ if } B \{ \ C_1 \ \} \text{ else } \{ \ C_2 \ \} \{ \psi \}
\]
Rules for Partial Correctness (continued)

\[
\begin{align*}
\{ \phi \land B \} & \quad C_1 \quad \{ \psi \} \quad \quad \{ \phi \land \neg B \} \quad C_2 \quad \{ \psi \} \\
\hline
\{ \phi \} & \quad \text{if} \quad B \quad \{ \quad C_1 \quad \} \quad \text{else} \quad \{ \quad C_2 \quad \} \quad \{ \psi \} \\
\end{align*}
\]

[If-statement]

\[
\{ \psi \land B \} \quad C \quad \{ \psi \} \\
\hline
\{ \psi \} \quad \text{while} \quad B \quad \{ \quad C \quad \} \quad \{ \psi \land \neg B \} \\
\]

[Partial-while]
Rules for Partial Correctness (continued)

\[
\begin{align*}
\vdash_{AR} \phi' & \rightarrow \phi & \{\phi\} \; C & \{\psi\} & \vdash_{AR} \psi & \rightarrow \psi' \\
\hline
\{\phi'\} \; C & \{\psi'\} & \text{[Implied]}
\end{align*}
\]
Factorial Example

We shall show that the following Core program $\text{Fac1}$ meets this specification:

\[
\begin{align*}
y &= 1; \\
z &= 0; \\
\text{while} \ (z \neq x) \ {\{ z = z + 1; \ y = y \times z; \} }
\end{align*}
\]

Thus, to show:

\[
\{ \top \} \text{Fac1} \ {\{ y = x! \}}
\]
Partial Correctness of \texttt{Fac1}

\begin{verbatim}
:
\{y = z!\} while ( z != x ) {
  \{y = z! \land z \neq x\} \quad \text{Invariant}
  \{y \cdot (z + 1) = (z + 1)!\} \quad \text{Implied}
  z = z + 1;
  \{y \cdot z = z!\} \quad \text{Assignment}
  y = y \ast z;
  \{y = z!\} \quad \text{Assignment}
}
\{y = z! \land \neg(z \neq x)\} \quad \text{Partial-while}
\{y = x!\} \quad \text{Implied}
\end{verbatim}
How To Discover an Invariant?

$$\{\eta \land B\} \ C \ \{\eta\}$$

$$\frac{}{\{\eta\} \ \text{while} \ B \ \{\ C \ \} \ \{\eta \land \neg B\}}$$

[Partial-while]
How To Discover an Invariant?

\[
\{\eta \land B\} \ C \ \{\eta\}
\]

\[
\{\eta\} \ \text{while} \ B \ \{\ C \ \} \ \{\eta \land \neg B\}
\]

To be proven:

\[
\{\phi\} \ \text{while} \ B \ \{\ C \ \} \ \{\psi\}
\]
How To Discover an Invariant?

\[
\begin{align*}
\{\eta \land B\} & \quad C \quad \{\eta\} \\
\hline
\{\eta\} & \quad \text{while} \quad B \quad \{C\} \quad \{\eta \land \neg B\}
\end{align*}
\]

To be proven:

\[
\{\phi\} \quad \text{while} \quad B \quad \{C\} \quad \{\psi\}
\]

1. \(\vdash_{AR} \phi \rightarrow \eta\)
How To Discover an Invariant?

\[
\{ \eta \land B \} \ C \ \{ \eta \} \\
\hline \\
\{ \eta \} \ \text{while} \ B \ \{ \ C \ \} \ \{ \eta \land \neg B \} \\
\]

To be proven: \[ \{ \phi \} \ \text{while} \ B \ \{ \ C \ \} \ \{ \psi \} \]

1. \[ \vdash_{\text{AR}} \phi \rightarrow \eta \]
2. \[ \vdash_{\text{AR}} \eta \land \neg B \rightarrow \psi \]
How To Discover an Invariant?

To be proven: \( \{ \phi \} \text{ while } B \{ C \} \{ \psi \} \)

1. \( \vdash_{AR} \phi \rightarrow \eta \)
2. \( \vdash_{AR} \eta \land \neg B \rightarrow \psi \)
3. \( \{ \eta \land B \} \ C \{ \eta \} \)
Partial Correctness of $\text{Fac1}$

$\{\top\}$

$\{(1 = 0!)\}$  Implied

$y = 1;$  Assignment

$\{y = 0!\}$  Assignment

$z = 0;$

$\{y = z!\}$  Assignment

while ( $z \neq x$ ) {
  
  ...

}

$\{y = z! \land \neg(z \neq x)\}$  Partial-while

$\{y = x!\}$  Implied
Review: Partial Correctness

Proof Calculus for Total Correctness

Programming by Contract
Ideas for Total Correctness

- The only source of non-termination is the `while` command.
- If we can show that the value of an integer expression decreases in each iteration, but never becomes negative, we have proven termination.
The only source of non-termination is the `while` command.

If we can show that the value of an integer expression decreases in each iteration, but never becomes negative, we have proven termination. Why?
The only source of non-termination is the `while` command.

If we can show that the value of an integer expression decreases in each iteration, but never becomes negative, we have proven termination. Why? Well-foundedness of natural numbers.
Ideas for Total Correctness

- The only source of non-termination is the `while` command.
- If we can show that the value of an integer expression decreases in each iteration, but never becomes negative, we have proven termination.
  Why? Well-foundedness of natural numbers
- We shall include this argument in a new version of the `while` rule.
Rules for Partial Correctness (continued)

\[
\begin{align*}
\{\psi \land B\} & \quad C \quad \{\psi\} \\
\{\psi\} & \quad \textbf{while} \quad B \quad \{\ C \} \quad \{\psi \land \neg B\}
\end{align*}
\]

\[
\begin{align*}
\{\psi \land B \land 0 \leq E = E_0\} & \quad C \quad \{\psi \land 0 \leq E < E_0\} \\
\{\psi \land 0 \leq E\} & \quad \textbf{while} \quad B \quad \{\ C \} \quad \{\psi \land \neg B\}
\end{align*}
\]
y = 1;
z = 0;
while (z != x) {
    z = z + 1;
y = y * z;
}

What could be a good variant $E$?
y = 1;
z = 0;
while (z != x) {
    z = z + 1; y = y * z;
}

What could be a good variant $E$?

$E$ must strictly decrease in the loop, but not become negative.
Factorial Example (Again!)

\[
y = 1; \\
z = 0; \\
\text{while } (z \neq x) \{ z = z + 1; \ y = y * z; \} \\
\]

What could be a good variant \(E\)?

\(E\) must strictly decrease in the loop, but not become negative.

Answer:

\[x - z\]
Total Correctness of \texttt{Fac1}

\begin{verbatim}
: {y = z! \land 0 \leq x - z}
while ( z != x ) {
    {y = z! \land z \neq x \land 0 \leq x - z = E_0}
    {y \cdot (z + 1) = (z + 1)! \land 0 \leq x - (z + 1) < E_0}
    z = z + 1;
    {y \cdot z = z! \land 0 \leq x - z < E_0}
    y = y \ast z;
    {y = z! \land 0 \leq x - z < E_0}
}
{y = z! \land \neg(z \neq x)}
{y = x!}
\end{verbatim}
Total Correctness of \texttt{Fac1}

\[
\begin{align*}
\{ x \leq 0 \} \\
\{ (1 = 0! \land 0 \leq x - 0) \} & \quad \text{Implied} \\
y = 1; \\
\{ y = 0! \land 0 \leq x - 0 \} & \quad \text{Assignment} \\
z = 0; \\
\{ y = z! \land 0 \leq x - z \} & \quad \text{Assignment} \\
\text{while } ( z \neq x ) \{ \\
\quad \vdots \\
\} \\
\{ y = z! \land \neg (z \neq x) \} & \quad \text{Total-while} \\
\{ y = x! \} & \quad \text{Implied}
\end{align*}
\]
1. Review: Partial Correctness
2. Proof Calculus for Total Correctness
3. Programming by Contract
Consider

\[ \{ \phi \} \; P \; \{ \psi \} \]

**Obligation for consumer of** \( P \)

Only run \( P \) when \( \phi \) is met.

**Obligation for producer of** \( P \)

Make sure \( \psi \) is met after every run of \( P \), assuming that \( \phi \) is met before the run.
Contracts as Documentation

```java
int factorial (x: int) { ... return y; }
```

Method name: factorial
Input: x of type int
Assumes: \(0 \leq x\)
Guarantees: \(y = x!\)
Output: y
Modifies only: y