Formal Verification - Inference Rules

1. \( \begin{align*} &\text{Q[E/x]D} \quad x = E \quad \text{DQD} \\
\text{We typically apply this rule from the end of a program forward.}
\end{align*} \)

2. \( \begin{align*} &\text{\(P \rightarrow P'\)} \quad \text{DQD} \\
\text{We apply this rule only implicitly, and never cite it.}
\end{align*} \)

3. \( \begin{align*} &\text{DQD C DQD} \\
\text{We apply this rule only implicitly, and never cite it.}
\end{align*} \)

4. \( \begin{align*} &\text{DQD} \\
\text{We apply this rule only implicitly, and never cite it.}
\end{align*} \)

5. \( \begin{align*} &\text{DQD} \\
\text{We apply this rule only implicitly, and never cite it.}
\end{align*} \)

6. \( \begin{align*} &\text{DQD} \\
\text{We apply this rule only implicitly, and never cite it.}
\end{align*} \)

7. \( \begin{align*} &\text{DQD} \\
\text{We apply this rule only implicitly, and never cite it.}
\end{align*} \)
Assignments

Complete the following annotations

\[ Q[E/x] = (2 = 2) \]

1. \((2 = 2)D\) \[ x = 2; \text{ E is } 2 \]
2. \((2 = y)D\) \[ x = 2; \]
3. \((2 = 0)D\) \[ x = 2; \]
4. \((x = 2)D\) assignment \[ x = y; \]
5. \((x = 0)D\) assignment \[ x = y. \]

\[ Q[E/x] = Q[x + 1/x] = (x + 1 = n + 1) \]

\[ Q[E/x] = [y/x] = (2 \cdot y = y + y) \]

The "assignment" inference rule.

\[ (Q[E/x])D \text{ then } Q \text{ must be true when replacing every } x \text{ by } E \]

\[ x = E; \]

\[ QD \text{ if } Q \text{ is true after assigning } x \text{ to } E, \]

Notes

1. \(E\) may contain \(x\) in it. Treat \(E\) as a whole expression and do not worry about what's inside.
2. When writing down \(Q[E/x]\), do NOT change the order of things in the formula and do NOT simplify it.

solutions
Assignments

Prove that the following program satisfies the given triple under partial correctness.

1. \( q ((x = x_0) \land (y = y_0)) \ D \)
   \( q ((y = y_0) \land (x = x_0)) \ D \) implied
   \( t = x_j \)
   \( q ((y = y_0) \land (t = x_0)) \ D \) assignment
   \( x = y_j \)
   \( q ((x = y_0) \land (t = x_0)) \ D \) assignment
   \( y = t_j \)
   \( q ((x = y_0) \land (y = x_0)) \ D \) assignment

Steps:
1. Push up assignments
2. Prove any implied's.

Proof of implied: \( - ( ((x = x_0) \land (y = y_0)) \rightarrow (y = y_0) \land (x = x_0)) \)

1. \( (x = x_0) \land (y = y_0) \) assumption
2. \( x = x_0 \) \( \Lambda e: 1 \)
3. \( y = y_0 \) \( \Lambda e: 1 \)
4. \( (y = y_0) \land (x = x_0) \) \( \Lambda i: 2, 3 \)
5. \( ((x = x_0) \land (y = y_0)) \rightarrow (y = y_0) \land (x = x_0)) \) \( \rightarrow i: 1-4 \)

2. \( q true \ D \)
   \( q ((x + y = x + y)) \ D \) implied
   \( z = x_j \)
   \( q (z + y = x + y) \ D \) assignment
   \( z = z + y_j \)
   \( q (z = x + y) \ D \) assignment
   \( u = z_j \)
   \( q (u = x + y) \ D \) assignment

Proof of implied: \( - (true \rightarrow (x + y = x + y)) \)

1. \( true \) assumption
2. \( (x + y = x + y) \) \( EQ1 + Ve. \)
3. \( (true \rightarrow (x + y = x + y)) \rightarrow i: 1-2 \)

Solutions
Conditional Statement \( (\text{if-then}) \) & \( (\text{if-then-else}) \)

1. "if-then"
   \[ \begin{align*}
   & \neg A \lor D \\
   & \text{if } (B) \land \neg C \Rightarrow \neg A \\
   & \neg A \lor D \\
   & \text{implied } (a) \quad 2
   \end{align*} \]

2. "if-then-else"
   \[ \begin{align*}
   & \neg A \lor D \\
   & \text{if } (B) \land \neg C \\
   & \neg A \lor D \\
   & \text{implied } (b) \quad 2
   \end{align*} \]

Steps:
1. Annotate the if-then's
2. Push up assignments
3. Prove any implied's

Proofs of implied (a) and (b)
Common Questions about if-then:

1. Why did you annotate the last line with "implied" and the implication \((P \land (\neg B)) \rightarrow Q\)?

For an if-then statement, there are 2 cases to consider. If \(B\) is true, then we go inside the if-block and execute \(C\). If \(B\) is false, we skip the if-block. The last "implied" annotation takes care of the second case. Even when we skip the if-block, we still need to show that \(Q\) is satisfied. This corresponds to proving the implication \((P \land (\neg B)) \rightarrow Q\).

2. How did you get \(P'\) and why do you know that \((P \land B)\) implies \(P'\)?

I derived \(P'\) by looking at \(C\) and \(Q\) and figuring out what precondition needs to be satisfied if \(Q\) is true after executing \(C\). For example, if \(C\) consists of assignments only, then \(P'\) is the result of pushing up \(Q\) through \(C\).

I don't know \((P \land B)\) implies \(P'\). By writing down \(P'\) and "implied" as the justification, I am saying that: "If this program satisfies my specification, then I need to prove that \((P \land B)\) implies \(P'\)."
Conditional Statements (If-Then)
Prove that the following program satisfies the given triple under partial correctness.

About line 3: you can immediately simplify it and write down \((\text{max} < \text{x})\) instead.

1. \(\text{if true} \quad \text{D} \)

2. \(\text{if (max < x)} \quad \text{D}\)

3. \(\text{if-then}\)

4. \(\text{D} \quad \text{if (x \geq x)} \quad \text{D}\)

5. \(\text{implied (a)}\)

6. \(\text{max = x}\)

7. \(\text{D (max \geq x)} \quad \text{assignment}\)

8. \(\text{if-then}\)

\(\text{implied (b) (true} \land (\neg(\text{max < x}))) \rightarrow (\text{max} \geq \text{x})\)

1. Proof of implied (a): \(\vdash \text{((true} \land (\text{max < x})) \rightarrow (\text{x} \geq \text{x}))\)

   (This is an informal proof. See a formal proof on the next page.)

   \((\text{x} \geq \text{x})\) is a tautology. Thus the implication holds.

2. Proof of implied (b): \(\vdash \text{((true} \land (\neg(\text{max < x}))) \rightarrow (\text{max} \geq \text{x}))\)

<table>
<thead>
<tr>
<th></th>
<th>((-\text{true} \land (\neg(\text{max &lt; x}))))</th>
<th>assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>((-\text{true} \land (\neg(\text{max &lt; x}))))</td>
<td>assumption</td>
</tr>
<tr>
<td>2</td>
<td>((-\text{true} \land (\neg(\text{max &lt; x}))))</td>
<td>assumption</td>
</tr>
<tr>
<td>3</td>
<td>((-\text{max &lt; x}))</td>
<td>(\Lambda e: 1)</td>
</tr>
<tr>
<td>4</td>
<td>((\text{max} \geq \text{x}))</td>
<td>def. of (\geq)</td>
</tr>
<tr>
<td>5</td>
<td>(((\text{true} \land (\neg(\text{max &lt; x}))) \rightarrow (\text{max} \geq \text{x})))</td>
<td>(\rightarrow i: 1-3)</td>
</tr>
</tbody>
</table>
Proof of implied (a): \( \vdash (\text{true} \land (\max < x)) \rightarrow (x \leq x) \)

<table>
<thead>
<tr>
<th>Step</th>
<th>Formula</th>
<th>Justification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( (\text{true} \land (\max &lt; x)) )</td>
<td>assumption</td>
</tr>
<tr>
<td>2</td>
<td>( x + 0 = x )</td>
<td>PA3 + A6e</td>
</tr>
<tr>
<td>3</td>
<td>( \exists z \ (x + z = x) )</td>
<td>( \exists i : 2 )</td>
</tr>
<tr>
<td>4</td>
<td>( x \leq x )</td>
<td>def of ( \leq )</td>
</tr>
<tr>
<td>5</td>
<td>( (\text{true} \land (\max &lt; x)) \rightarrow (x \leq x) )</td>
<td>( \Rightarrow \ 1-4 )</td>
</tr>
</tbody>
</table>
Conditional Statements (If-Then-Else)

Prove that the following program satisfies the given triple under partial correctness.

true D

if \( x > y \) \{ 
\[ \begin{align*}
  & (\text{true} \land (x > y)) \ \text{D} & \text{if-then-else} \\
  & ( (x > y) \land (x = x)) \lor ( (x < y) \land (x = y)) \ \text{D} & \text{implied (a)} \\
  & \max = x; \\
\end{align*} \]

\[ \begin{align*}
  & ((x > y) \land (\max = x)) \lor ( (x < y) \land (\max = y)) \ \text{D} & \text{assignment} \\
\end{align*} \]

\[ \begin{align*}
  & \max = y; \\
\end{align*} \]

\[ \begin{align*}
  & ((x > y) \land (\max = x)) \lor ( (x < y) \land (\max = y)) \ \text{D} & \text{assignment} \\
\end{align*} \]

\[ \begin{align*}
  & (x > y) \land (\max = x)) \lor ( (x < y) \land (\max = y)) \ \text{D} & \text{if-then-else} \\
\end{align*} \]

Proof of implied (a):

1. \( x > y \) \hspace{1cm} \text{premise}
2. \( x = x \) \hspace{1cm} \text{EQ1+VE}
3. \( ((x > y) \land (x = x)) \) \hspace{1cm} \text{\( \Lambda \tilde{t} : 1, 2 \)}
4. \( (((x > y) \land (x = x)) \lor ((x < y) \land (x = y))) \) \hspace{1cm} \text{\( V\tilde{t} : 3 \)}