Reasoning about Programs

Murat Demirbas

SUNY Buffalo

03 September 2013
1 Specification

2 Reasoning about a single action

3 Safety

4 Progress
Cloud computing explained

Video is unrelated to today’s lecture but related to the course content
http://www.youtube.com/watch?v=QJncFirhjPg
Reasoning about a program involves showing that the program meets its specification. A specification is a high-level description of program behavior.

We will use program properties to specify our programs. A program property is a predicate on an execution.

We say that a program property $R$ holds for a program $P$ exactly when $R$ holds for every possible execution of $P$. 
There are two fundamental categories of program properties that we will use to describe program behavior: safety properties, and progress properties. Before considering these properties on computations (consisting of an infinite sequence of actions), we first examine how to reason about the behavior of a single action, executed once.
A Hoare triple is an expression of the form
\{P\}S\{Q\}

where \(S\) is a program action and \(P\) and \(Q\) are predicates. The predicate \(P\) is often called the “precondition” and \(Q\) the “postcondition”.

Informally, this triple says that if \(S\) begins execution in a state satisfying predicate \(P\), it is guaranteed to terminate in a state satisfying \(Q\).
Saying that a triple \( \{P\} \ x := y + 1 \ {\text{even}.x} \) holds is the same as saying:
\[ [P \Rightarrow \text{even.}(y + 1)] \]

To prove \( \{P\} \ x := E \ \{Q\} \), we must show \( [P \Rightarrow Q^x_E] \).

\( Q^x_E \) is used to indicate writing expression \( Q \) with all occurrences of \( x \) replaced by \( E \).
Example

Does this triple hold?
\[ \{ x \geq -2 \} \ x:=x-y+3 \ \{ x + y \geq 0 \} \]
To prove the triple $\{P\} g \xrightarrow{\quad} x := E \{Q\}$, we need to show:

$$[(P \land g \Rightarrow Q^x_E) \land (P \land \neg g \Rightarrow Q)]$$
What is $P$?
Safety properties

A safety property is a property that can be violated by a finite computation.

- Invariant property we had discussed earlier is a safety property.
- Two processes are not in critical section concurrently is a safety property.
- The program will terminate eventually is not a safety property.
A **next** property (i.e., a predicate on programs) is written:

\[ P \text{ next } Q \]

where \( P \) and \( Q \) are predicates on states in the program.

\( P \text{ next } Q \) means that if a program is in a state satisfying \( P \), its very next state (i.e., after choosing and executing exactly one action) must satisfy \( Q \).

Since any action could be chosen as the next one to be executed, we must show that for every action, if it begins in \( P \), must terminate in \( Q \).
To prove 
\((P \text{ next } Q).G\)
we must show 
\((\forall x : a \in G : \{P\}a\{Q\})\)

Since *skip* is always part of any program, we have \(P \Rightarrow Q\)
false \texttt{next} Q

\texttt{P next true}

\((P1 \texttt{next} Q1) \land (P2 \texttt{next} Q2) \implies (P1 \land P2) \texttt{next} (Q1 \land Q2)\)

\((P1 \texttt{next} Q1) \land (P2 \texttt{next} Q2) \implies (P1 \lor P2) \texttt{next} (Q1 \lor Q2)\)

\((P \texttt{next} Q) \land [Q \implies Q'] \implies (P \texttt{next} Q')\)

\((P \texttt{next} Q) \land [P' \implies P] \implies (P' \texttt{next} Q)\)
stable.\( P \) means that once \( P \) becomes true, it remains true.

\[
\text{stable.} P \equiv \text{next } P
\]

- \text{stable.}\text{true}
- \text{stable.}\text{false}

- \text{stable.} P \land \text{stable.} Q \Rightarrow \text{stable.} (P \land Q)
- \text{stable.} P \land \text{stable.} Q \Rightarrow \text{stable.} (P \lor Q)

???

- ??? \text{stable.} P \land [P \Rightarrow P'] \Rightarrow \text{stable.} P'
- ??? \text{stable.} P \land [P' \Rightarrow P] \Rightarrow \text{stable.} P'
Invariant property is very important for reasoning about safety of your program.
Unlike safety, a progress (liveness) property can not be violated by a finite execution.

Progress is a predicate on possible computation suffixes.

All program properties of interest can be expressed as a conjunction of safety and progress.
Transient.

\[
\text{ transient.} P . G \equiv \\
(\exists a : a \in G : \{ P \} a \{ \neg P \})
\]

\[
\text{ transient.} P \land [ P' \Rightarrow P] \Rightarrow \text{ transient.} P'
\]
\[
\text{ transient.} P \land [ P \Rightarrow P'] \Rightarrow \text{ transient.} P' \ ???
\]
Transient (example)

\[\text{even}.x \quad \rightarrow \quad x := x + 1\]
\[\text{transient}.(x = 2) \quad ?\]

\[n \leq 2 \quad \rightarrow \quad n := n + 1\]
\[\text{transient}.(n = 0 \vee n = 1) \quad ???\]
Ensures

$P$ ensures $Q$ means that if $P$ holds, it will continue to hold so long as $Q$ does not hold, and eventually $Q$ does hold.

$P$ ensures $Q \equiv ((P \land {\neg}Q) \text{ next } (P \lor Q)) \land \text{ transient.}(P \land {\neg}Q)$
Ensures (example)

\[
even.x \quad \longrightarrow \quad x := x + 1
\]
\[
(x = 2 \lor x = 6) \text{ ensures } (x = 3 \lor x = 7)
\]

\[
n \leq 2 \quad \longrightarrow \quad n := n + 1
\]
\[
n = 1 \text{ ensures } n = 3
\]
$P \leadsto Q$ means that if $P$ is true at some point, $Q$ will be true (at that same or a later point) in the computation.

$P \text{ ensures } Q \implies P \leadsto Q$

$(P \leadsto Q) \land (Q \leadsto R) \implies P \leadsto R$
What is the relation between:

\textit{transient}. \( P \)

\( P \models \neg P \)
$P \rightsquigarrow true$

false $\rightsquigarrow P$

$P \rightsquigarrow P$

$(P \rightsquigarrow Q) \land [Q \Rightarrow Q'] \Rightarrow P \rightsquigarrow Q'$

$(P \rightsquigarrow Q) \land [P' \Rightarrow P] \Rightarrow P' \rightsquigarrow Q$

stable. $P \land trans. (P \land \neg Q) \Rightarrow P \rightsquigarrow (P \land Q)$

$\text{???} \; (P \rightsquigarrow Q) \land (P' \rightsquigarrow Q') \Rightarrow (P \land P') \rightsquigarrow (Q \land Q')$
A metric (or “variant function”) is a function from the state space to a well-founded set (e.g., the natural numbers). The well-foundedness of the range means that the value of the function is bounded below (i.e., can only decrease a finite number of times).

Theorem 10 (Induction). For a metric $M$,\\
$(\forall m :: P \land M = m \Rightarrow (P \land M < m) \lor Q) \Rightarrow P \Rightarrow Q$
Theorem 11. For a metric $M$,

$(\forall m :: P \land M = m \text{ next } (P \land M \leq m) \lor Q) \land (\forall m :: \text{ transient.}(P \land M = m)) 
\Rightarrow P \Rightarrow Q$ 

$(\forall i, m :: \{P \land M = m \land g_i\}g_i \rightarrow a_i\{(P \land M < m) \lor FP\}) 
\Rightarrow P \Rightarrow FP$