CS 142: Lecture 2.1
Reasoning About Programs

Richard M. Murray
2 October 2017

Goals:
- Introduce the concept of proving correctness of programs
- New concepts: Hoare triples, assignment axiom, stable operator

Reading:
- P. Sivilotti, *Introduction to Distributed Algorithms*, Section 3.1-3.3
Last Week: Models of Computation

UNITY model provides (seemingly) simple description of programs

- Program = variables + actions [assignments] (that’s it!)
- Guarded assignment \((g \rightarrow a)\) allows modeling of finite state automata
- Distributed programs captured by nondeterministic execution model
- Termination = reaching a fixed point (variables remain constant)

Next: how to we prove that specifications are satisfied?

- A1: exhaustive testing [remember ZA002!]
- A2: model checking [for specific instantiation]
- A3: formal proof [often generalizable]

Fri: how to prove things using predicate calculus and quantification (review + some new stuff)

This week: reasoning about program behavior and safety properties (invariants)
Examples to Consider

**FindAverage (average consensus)**
- Given a set of N sensors on a graph, would like to agree on the value of the average measurements
- Example: agree that it is too cold and warm up the room
- Q1: what protocol should we implement to solve this problem?
- Q2: is it *always* possible to agree?

**ChooseDefenders**
- Given a set of initial assignments in the RoboFlag drill, communicate with left and right neighbors and switch assignments such that we end up with no “crossed” assignments
- Q1: What are the properties we want to guarantee?
  - Termination: program terminates (variables remain constant)
  - Correctness: only fixed points are the desired ones
- Q2: What could go wrong?
  - Deadlock: get stuck in a state (= undesired fixed point)
  - Livelock: never terminate (eg, assignments “oscillate”)

Consensus is reached when:

\[ x_i(n) = \frac{1}{N} \sum_{j=1}^{N} x_j(0), \forall i \]

*n*: Time index

N: number of nodes
Example: Average Consensus

Problem setup
• Variables: local estimate of average, initialized to local measurement
• Assignments: two agents communicate and share information

Program \( \text{AverageConsensus} \)

\[
\begin{align*}
\text{constant} & \quad N \quad \{ \text{number of agents} \} \\
& \quad G \quad \{ \text{interconnection graph} \} \\
& \quad \alpha : 0 < \alpha < 1 \\
\text{var} & \quad x : \text{array of } N \text{ numbers} \\
\text{assign} & \quad (\forall i, j : j \in \mathcal{N}_i : x[i] := \alpha x[i] + (1 - \alpha) x[j] \\
& \quad \quad \quad || x[j] := \alpha x[j] + (1 - \alpha) x[i]))
\end{align*}
\]

Specification
• Show that we converge to a consensus (everyone agrees on average value)
• In practice, usually good enough to show that we get close within finite time

Why do we need a “proof”?
• Want to understand conditions under which this is not true (eg, directed graphs)
• Can extend to understand more interesting cases (eg, what happens if someone lies)
Properties of Programs

Notation: property(P) or property(P, Q) or P property Q

- A property operates on a set of states that satisfy a formula (predicate) P (and/or Q)
- The property is true if it holds for all possible executions

Reasoning about properties using graphs

- Formulas define subsets of the state space
- Can reason about whether a property holds by looking at how the transitions map to the (sets of states representing) properties

Reasoning about properties using formulas

\[
\text{stable.}(r \leq M) \\
\equiv \frac{(r < M) \quad \text{next } (r < M)}{(r < M) \quad \text{next } (r < M)}
\]

\[
\equiv \frac{(\forall a :: \{r \leq M\} \quad a \quad \{r \leq M\})}{\{ \text{definition of program} \}}
\]

\[
\equiv \{ \text{assignment axiom} \}
\]

\[
(\forall x : 0 \leq x < N : \{r \leq M\} \quad r := \max(r, A[x]) \quad \{r \leq M\})
\]

Can also combine representations

\[
(P \subseteq Q) \land \text{stable}(Q) \Rightarrow \text{reachable}(P) \subseteq Q
\]
Reasoning About Actions

How are we going to prove things?
  • A: show that sets of properties hold for all executions

Two main parts of a proof: safety and liveness (or progress)
  • Safety: show that bad things don’t (ever) happen.
  • Liveness: show that good things eventually do happen
  • Roughly: can show that all specifications break down into safety and liveness

Notation: Hoare triple - {P} a {Q}
  • P = precondition (predicate), a = program action, Q = postcondition (predicate)
  • Interpretation: the triple evaluates to true if for any program state in which P holds, if we take the action a then Q will hold after the action is executed

Assignment axiom: {P} x := E {Q}
  • In what state must we being execution in order for Q to hold after executing x := E?
  • Written another way: find those states for which $[P \implies Q^x_{E}]$
Reasoning About Guarded Actions

Hoare triple with a guarded action: \( \{P\} g \rightarrow x := E \{Q\} \)

- What we need to show depends on whether the guard is true or false
- \( g = \text{true} \): same as assignment
- \( g = \text{false} \): need \( Q \) to be satisfied

\[
\left[ (P \land g \Rightarrow Q_x^E) \land (P \land \neg g \Rightarrow Q) \right]
\]

Example \( \{x > y = 7\} \ x > y \rightarrow x, y := y, x \ \{x > 3\} \)

\[
(x > y = 7 \land x > y \Rightarrow y > 3) \land (x > y = 7 \land \neg(x > y) \Rightarrow x > 3)
\]

\[
\iff \ \{ \text{antecedent strengthening of } \Rightarrow : [(X \Rightarrow Z) \Rightarrow (X \land Y \Rightarrow Z)] \} \]

\[
(y = 7 \Rightarrow y > 3) \land (x > y = 7 \land \neg(x > y) \Rightarrow x > 3)
\]

\[
\equiv \ \{ 7 > 3 \} \]

\[
x > y = 7 \land \neg(x > y) \Rightarrow x > 3 \]

\[
\equiv \ \{ \text{definition of } \neg \} \]

\[
x > y = 7 \land x \leq y \Rightarrow x > 3
\]

\[
\iff \]

\[
x > y \land x \leq y \Rightarrow x > 3
\]

\[
\equiv \]

\[
\text{false } \Rightarrow x > 3
\]

\[
\equiv \ \{ \text{property of } \Rightarrow : [\text{false } \Rightarrow X \equiv \text{true}] \} \]

true
The Next Relation: $P \text{ next } Q$

Use to reason about properties of a program $G$ as it executes

$$P \text{ next } Q \equiv (\forall a : a \in G : \{P\} \ a \ \{Q\})$$

- $P$ and $Q$ are predicates on states
- next is a binary relation between predicates

$P \text{ next } Q$ in terms of graphs means that
- (1) for all edges $(u, v)$ in a graph, if $u$ is in $P$ then $v$ is in $Q$,
- (2) furthermore for all $u$ in $P$, $u$ is also in $Q$ (why: _________ )

Some useful properties of next (prove in HW #2)

$$\begin{align*}
(P \text{ next } Q) \land (Q \subseteq Q') & \Rightarrow (P \text{ next } Q') \\
(P \text{ next } Q) \land (P' \subseteq P) & \Rightarrow (P' \text{ next } Q)
\end{align*}$$

RoboFlag Drill examples ($z =$ defender pos’n, $y =$ attacker height)
- Defenders never collide
  $$z_i < z_{i+1} \text{ next } z_i < z_{i+1}$$
- If attackers are far enough away, we won’t switch back and forth
  $$\forall i . \ y_i > 2\delta \land z_i + 2\delta < z_{i+1} \land \neg switch_{i,i+1} \text{ next } \neg switch_{i,i+1}$$
**Stable**

**Definition:** \( \text{stable}(P) \)
- Informal: once \( P \) becomes true, it remains true
- Formally: \( \text{stable}(P) \equiv P \text{ next } P \)
- Note: \( \text{stable}(P) \) does not mean that \( P \) is true for all (or even any) program executions

**When do we use stable in a proof?**
- Termination: \( \text{stable}([p]) \)
- Often combined with progress (Wed + W3)
  - Show that if we satisfy some conditions then we eventually get to a good set of states (and stay there)

**Some useful results (will prove on the homework)**
- \( \text{stable}(P) \wedge \text{stable}(Q) \implies \text{stable}(P \wedge Q) \)
  - Interpretation: if \( P \) is stable and \( Q \) is stable, then at the point that both of them are true, they will both remain true
- \( \text{stable}(P) \wedge \text{stable}(Q) \implies \text{stable}(P \vee Q) \)
  - Note: not true that \( \text{stable}(P) \vee \text{stable}(Q) \implies \text{stable}(P \vee Q) \)
**Reachable(P) is the smallest stable set that includes P**

- Reachable(P) = set of points that we can reach from states that satisfy predicate P
- Proof sketch (exercise: turn into a formal proof = sequence of implications/equivalents)
  - Let $Q = \text{reachable}(P)$. Clear that $P \subseteq Q$ and $\text{stable}(Q)$
  - Suppose $Q'$ is a smaller set ($Q' \subset Q$) with $P \subseteq Q'$ and $\text{stable}(Q')$
  - $Q' \subset Q$ and $\text{stable}(Q) \implies Q = \text{reachable}(P) \subset Q' \implies Q = Q'$
  - Algorithm for finding reachable(P): start with P add neighbors until you stop growing

---

**Which of the following formulas are true?**

\[
\text{stable}(P) \land (P \subseteq Q) \implies \text{stable}(Q) \quad \text{_____}
\]

\[
\text{stable}(P) \land (Q \subseteq P) \implies \text{stable}(Q) \quad \text{_____}
\]

\[
\forall P : \text{stable}(\text{reachable}(P)) \quad \text{_____}
\]

\[
(P \subseteq Q) \land \text{stable}(Q) \implies \text{reachable}(P) \subseteq Q \quad \text{_____}
\]
Examples: Properties for Average Consensus

Program \( \text{AverageConsensus} \)

constant \( N \) \{number of agents\}

\( G \) \{interconnection graph\}

var \( x \) : array of \( N \) numbers

assign

\[
\begin{align*}
( \forall i, j & : j \in \mathcal{N}_i : x[i] := \alpha x[i] + (1 - \alpha) x[j] \\
\| & x[j] := \alpha x[j] + (1 - \alpha) x[i] )
\end{align*}
\]

What are some stable properties for this program? [assume \( \alpha = 1/2 \)]

- stable\((x_i \leq x_i^0)\)?

- ___

- stable\((x_i + x_j \leq x_i^0 + x_j^0)\)?

- ___

- stable\((x_i \leq \max_i x_i^0)\)?

- ___

- stable\(((+i : 0 \leq j \leq N - 1 : x_i) \leq (+i : 0 \leq i \leq N - 1 : x_i^0))\)?

- ___
Summary: Reasoning About Programs

Initial tools for reasoning about program properties

- **UNITY approach**: assume that any (enabled) command can be run at any time
- **Hoare triple**: show that all (enabled) actions satisfying a predicate $P$ will imply a predicate $Q$
- “Lift” Hoare triple to define $next$:
  \[
  \forall a : a \in G : \{ P \} \ a \ \{ Q \}
  \]
- **Stability**: $stable(P) \equiv P \ next \ P$
- Wed: finish stability and introduce liveness properties