CS/IDS 142: Lecture 3.1
Progress Properties and Metrics

Richard M. Murray
14 October 2019

Goals:
• Define liveness (progress) properties and metrics (variant functions)
• New properties: transient, ensures, leads-to, induction

Reading:
• P. Sivilotti, Introduction to Distributed Algorithms, Section 3.5
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: $\text{stable}(P) \equiv P \text{ next } P$
- Invariants: $\text{invariant}(P) \equiv \text{initially}(P) \land \text{stable}(P)$
Properties for RoboFlag program

Safety (Defenders do not collide)
\[ z_i < z_{i+1} \quad \text{next} \quad z_i < z_{i+1} \]

Stability (switch predicate stays false)
\[ \forall i . \ y_i > 2\delta \wedge z_i + 2\delta < z_{i+1} \wedge \neg\text{switch}_{i,i+1} \quad \text{next} \quad \neg\text{switch}_{i,i+1} \]

Robots are "far enough" apart.

Progress (we eventually reach a fixed point)
- Let \( \rho \) be the number of blue robots that are too far away to reach their red robots
- Let \( \beta \) be the total number of conflicts in the current assignment
- Define the metric that captures “energy” of current state (\( V = 0 \) is desired)

\[
V = \left( \binom{n}{2} + 1 \right) \rho + \beta \\
\rho = \sum_{i=1}^{n} r(i,i) \\
\beta = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \gamma(i,j) \\
\gamma(i,j) = \begin{cases} 
1 & \text{if } x_{\alpha(i)} > x_{\alpha(j)} \\
0 & \text{otherwise}
\end{cases}
\]

- Can show that \( V \) always decreases whenever a switch occurs

\[ \forall i . \ z_i + 2\delta m < z_{i+1} \wedge \exists j . \text{switch}_{j,j+1} \wedge V = m \quad \text{next} \quad V < m \]
The ‘Transient’ Property

Definition
• Informally: “if $P$ becomes true at some point in the computation, it is guaranteed to become false at some later point $\Rightarrow P$ is false infinitely often” [not quite accurate]

$$\exists a : a \in G : \{ P \} a \{ \neg P \}$$

• Compare to next: use $\exists$ instead of $\forall$
• Allowable for $P$ to remain true for one or more actions, as long as there is always one action that falsifies $P$ for every state for which $P$ is true (strong property!)

Simple example

<table>
<thead>
<tr>
<th>Program</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>var</td>
<td>$n : \text{natural number}$</td>
</tr>
<tr>
<td>initially</td>
<td>$n = 0$</td>
</tr>
<tr>
<td>assign</td>
<td>$n := n + 1$</td>
</tr>
</tbody>
</table>

Which of the following hold (show formally in HW #3):
• Weakening: $\text{transient}(P) \land [P \implies P'] \implies \text{transient}(P')$ _____
• Strengthening: $\text{transient}(P) \land [P' \implies P] \implies \text{transient}(P')$ _____
• Intuition: remember that $P' \Rightarrow P$ (formula) is same as $P' \subseteq P$ (for the program graph)
The ‘Ensures’ Property

Definition
• If $P$ holds, it will continue to hold as long as $Q$ doesn’t hold AND eventually $Q$ holds

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

Example

Program $CountIfSmall$

\begin{align*}
\text{var} & \quad n : \text{natural number} \\
\text{initially} & \quad n = 0 \\
\text{assign} & \\
& \quad n \leq 2 \rightarrow n := n + 1
\end{align*}

$(n = 1 \lor n = 2) \text{ ensures } (n \geq 2) \text{?} \quad \underline{\text{_____}}$

$n = 1 \text{ ensures } n = 3 \text{?} \quad \underline{\text{_____}}$

Some properties
• Weakening: $(P \text{ ensures } Q) \land [Q \Rightarrow R] \Rightarrow (P \text{ ensures } R)$
• Disjunction: $(P \text{ ensures } Q) \Rightarrow (P \lor R) \text{ ensures } (Q \lor R)$

Remarks
• Ensures is still “low level”: defines properties at the level of single actions
The ‘Leads-To’ Property

Definition
- If $P$ is true at some point, $Q$ will be true (at that same or a later point) in the computation
  
  $P$ ensures $Q$ $\implies$ $P \leadsto Q$
  
  $(P \leadsto Q) \land (Q \leadsto R) \implies P \leadsto R$
  
  $(\forall i :: P_i \leadsto Q) \implies (\exists i :: P_i) \leadsto Q$

Example

Program $\text{CountIfSmall}$

<table>
<thead>
<tr>
<th>var</th>
<th>n : natural number</th>
</tr>
</thead>
<tbody>
<tr>
<td>initially</td>
<td>$n = 0$</td>
</tr>
<tr>
<td>assign</td>
<td>$n \leq 2 \rightarrow n := n + 1$</td>
</tr>
</tbody>
</table>

Which of the following is true?

- $(P \leadsto Q) \land [P' \implies P] \implies P' \leadsto Q$
- $(P \leadsto Q) \land [Q \implies Q'] \implies P \leadsto Q'$

Remarks
- Leads-to is key property we will use in proofs (show that program leads to fixed point)
Which of the Following Properties are True?

Disjunction
- \((P \rightarrow Q) \land (R \rightarrow Q) \Rightarrow (P \lor R) \rightarrow Q\)
- \((P \rightarrow Q) \land (P \rightarrow R) \Rightarrow P \rightarrow (Q \land R)\)
- \((P \rightarrow Q) \land (P' \rightarrow Q') \Rightarrow (P \land P') \rightarrow (Q \land Q')\)

Stable Strengthening
- \(\text{stable}.P \land \text{transient}.(P \land \neg Q) \Rightarrow P \rightarrow (P \land Q)\)

Progress-Safety-Progress (PSP)
- \((P \rightarrow Q) \land (R \text{ next } S) \Rightarrow (P \land R) \rightarrow ((R \land Q) \lor (\neg R \land S))\)

- PSP allow us to combine a safety proof with a progress proof
- Either stay in R and satisfy Q or move out of R and satisfy S
- Very useful in progress proofs
Induction (and Metrics)

Approach: use metric to show that a property is eventually satisfied

- Definition: a metric (or variant function) is a function from the state space to a “well-founded set” (e.g., set with lower bound)

\[ \text{Theorem 10 (Induction for } \sim \text{). For a metric } M, \]
\[ (\forall m :: P \land M = m \sim (P \land M < m) \lor Q) \Rightarrow P \sim Q \]

- This theorem gives us a way to prove properties of programs: find a metric that shows that we eventually get to a desired fixed point (= termination)

Problem: can be hard to find a function that strictly decreases

- Alternative: make sure that P doesn’t increase and eventually decreases

\[ \text{Theorem 11 (Restricted Form of Induction for } \sim \text{). For a metric } M \]
\[ (\forall m :: P \land M = m \text{ next } (P \land M \leq m) \lor Q) \]
\[ \land (\forall m :: \text{ transient.}(P \land M = m)) \]
\[ \Rightarrow P \sim Q \]

OK for M to remain \( = m \), as long as there is some action that decreases m
Reasoning about Fixed Points

Variant: show that all enabled actions decrease the metric

Theorem 12 (Induction for $\leadsto$). For a metric $M$,

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

$$\land (\forall i :: \neg g_i ) \Rightarrow Q$$

$$\Rightarrow \quad P \leadsto Q$$

- Allows you to reason about fixed point (metric at min or all guards disabled)
Example: FindMax

Program: \textit{FindMax}

\begin{align*}
\text{var} & \quad A : \text{array } 0..N - 1 \text{ of int,} \\
\text{r : int} & \\
\text{initially} & \quad r = A[0] \\
\text{assign} & \quad ( \quad x : 0 \leq x \leq N - 1 : r := \text{max}(r, A[x]) \quad )
\end{align*}

Specification

- Safety: \textbf{stable}(r = M) \ [\text{Lecture 2.2}]
- Progress: \textbf{true} \leadsto (r = M)

Structure of the proof

- Fixed point: \textbf{FP} \equiv (\forall x : 0 \leq x \leq N - 1 : r = \text{max}(r, A[x]) )
  \equiv r \geq (\text{Max } x : 0 \leq x \leq N - 1 : A[x] )
  \equiv r \geq M

- Invariant: \textbf{invariant.}(r \leq M)
  - Combined with FP, this means that if we terminate at FP then \( r = M \)

- Metric: \textbf{r}
  - Never decreases and must increase at some point if \( r < M \)

\begin{align*}
\text{transient.}(r = k \land r < M) & \\
\Rightarrow & \quad \{ \text{transient.}\text{P} \Rightarrow (P \rightsquigarrow \neg P) \} \\
r = k \land r < M \rightsquigarrow r \neq k \lor r \geq M & \\
\Rightarrow & \quad \{ \text{stable.}(r \geq k) \} \\
r = k \land r < M \rightsquigarrow r > k \lor r \geq M & \\
\equiv & \quad \{ X \lor Y \equiv (\neg Y \land X) \lor Y \} \\
r < M \land r = k \rightsquigarrow (r < M \land r > k) \lor r \geq M & \\
\Rightarrow & \quad \{ \text{induction} \} \\
r < M \rightsquigarrow r \geq M & \\
\equiv & \quad \{ \text{definition of FP} \} \\
r < M \rightsquigarrow FP & \\
\equiv & \quad \{ \text{initially.}(r < M) \} \\
\text{true} \leadsto FP & \text{ Will show on Wed}
\end{align*}
Properties for RoboFlag program

Safety (Defenders do not collide)

\[ z_i < z_{i+1} \text{ next } z_i < z_{i+1} \]

Stability (switch predicate stays false)

\[ \forall i . \ y_i > 2\delta \land z_i + 2\delta < z_{i+1} \land \neg \text{switch}_{i,i+1} \text{ next } \neg \text{switch}_{i,i+1} \]

Robots are "far enough" apart.

Progress (we eventually reach a fixed point)

- Let \( \rho \) be the number of blue robots that are too far away to reach their red robots
- Let \( \beta \) be the total number of conflicts in the current assignment
- Define the metric that captures "energy" of current state (\( V = 0 \) is desired)

\[ V = \left( \binom{n}{2} + 1 \right) \rho + \beta \quad \rho = \sum_{i=1}^{n} r(i,i) \quad \beta = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \gamma(i,j) \quad \text{where} \quad \gamma(i,j) = \begin{cases} 1 & \text{if } x_{\alpha(i)} > x_{\alpha(j)} \\ 0 & \text{otherwise} \end{cases} \]

- Can show that \( V \) always decreases whenever a switch occurs

\[ \forall i . \ z_i + 2\delta m < z_{i+1} \land \exists j . \text{switch}_{j,j+1} \land V = m \text{ next } V < m \]
Summary: Progress Properties and Metrics

Establish *progress* properties

- **Transient**: $(\exists a : a \in G : \{P\} \ a \ \{\neg P\})$
- **Ensures**:
  $$(P \land \neg Q) \ \land \ \text{next} \ (P \lor Q) \ \land \ \text{transient.}(P \land \neg Q)$$
- **Leads-to**:
  $$P \ \text{ensures} \ Q \ \Rightarrow \ P \leadsto Q$$
  $$(P \leadsto Q) \land (Q \leadsto R) \ \Rightarrow \ P \leadsto R$$
  $$(\forall i :: P_i \leadsto Q) \ \Rightarrow \ (\exists i :: P_i) \leadsto Q$$
  - This is the main property that we care about for proving that computations terminate correctly

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

Next (Wed): show that we can use all of this to do something useful