Caltech

CS 142: Lecture 3.2
Reasoning about Programs: Examples

Richard M. Murray

16 October 2019

Goals:

- Walk through some examples small programs and prove correctness
- Example 1: FindMax (from Sivilotti)
- Example 2: AverageConsensus
- Example 3 (if time): RoboFlag Drill

Reading:

- P. Sivilotti, *Introduction to Distributed Algorithms*, Chapter 4
- (Optional) E. Klavins, "A computation and control language for multi-vehicle systems," Proceedings of the 48th IEEE Conference on Decision and Control, pp. 4133–4139, 2003. DOI 10.1109/CDC.2003.1271797

Example: FindMax

Program	$FindMax$
var	A : array $0..N-1$ of int,
	r : int
initially	$r = A[0]$
assign	
	$(\; [\!] \; x \; : \; 0 \leq x \leq N-1 \; : \; r := max(r, A[x]) \;)$

Specification

- Safety: **stable**(r = M)
- Progress: **true** $\rightsquigarrow$ (r = M)

Structure of the proof

- Fixed point: identify the conditions under which the algorithm terminates
  - $$\begin{aligned} FP \quad &\equiv \quad (\forall x \; : \; 0 \leq x \leq N-1 \; : \; r = \mathbf{max}(r, A[x]) \;) \\ &\equiv \quad r \geq \; (\mathbf{Max}\, x \; : \; 0 \leq x \leq N-1 \; : \; A[x] \;) \\ &\equiv \quad r \geq M \end{aligned}$$
- Invariant: set of constraints on the behavior of the program
  - $\mathbf{invariant}.(r \leq M)$
  - Combined with FP, this means that if we terminate at FP then r = M
- Metric: upper or lower bounded function used to track progress
  - metric: *r* (current maximum)
  - Never decreases and must increase at some point if r < M

FindMax Proof Outline

Safety: stable(r = M) (once we reach the fixed point we will stay there => terminate)

- Same idea as earliest meeting time property shown in Homework #2

Progress: true $\leadsto$ (r = M)

- Use restricted form of induction theorem (Sivilotti, Section 3.5)

Theorem 11 (Restricted Form of Induction for $\leadsto$). *For a metric* L

$$
\begin{aligned}
& (\forall m :: P \wedge L = m \textbf{ next } (P \wedge L \geq m) \vee Q) \\
& \wedge (\forall m :: \textbf{transient}.(P \wedge L = m)) \\
\Rightarrow \quad & P \leadsto Q
\end{aligned}
$$

Use metric w/ *upper* bound

- For FindMax, we take P = true, L = r, Q = {r = M} [note: changed *M* to *L*]
- Need to show
  (1) for any action, the value or r does not get smaller
  (2) we cannot stay at r = m forever (eg, **transient**(r=m))
- r = m **next** r ≥ m ∨ r = M is true for all actions (by def'n of max) => just need to show **transient** property

Program	$FindMax$
var	A : array $0..N-1$ of int,
	r : int
initially	$r = A[0]$
assign	
	$(\parallel x : 0 \leq x \leq N-1 : r := max(r, A[x]))$

FindMax Proof: r < M is transient

Task: show that transient(r = k)

- Problem: this is only true for as long as r < M

Instead: show that r = k is transient as long as r < M

$$
\begin{array}{ll}
 & \textbf{transient}.(r = k \ \wedge \ r < M) \\
\equiv & \quad \{ \text{ definition of } \textbf{transient} \ \} \\
 & (\exists a :: \ \{r = k \ \wedge \ r < M\} \quad a \quad \{r \neq k \ \vee \ r \geq M\} \ ) \\
\Leftarrow & \quad \{ \text{ definition of program } \} \quad \text{with m selected so that A[m] = M} \\
 & \{r = k \ \wedge \ r < M\} \quad r := max(r, \ \ \text{M} \ \ ) \quad \{r \neq k \ \vee \ r \geq M\} \\
\equiv & \quad \{ \text{ assignment axiom } \} \\
 & r = k \ \wedge \ r < M \ \Rightarrow \ \textbf{max}\,(r, \ \ \text{M} \ \ ) \neq k \ \vee \ \textbf{max}\,(r, \ \text{M} \ ]) \geq M \\
\Leftarrow & \quad \{ \text{ from definition of max } \} \\
 & r = k \ \wedge \ r < M \ \Rightarrow \ \textbf{max}\,(r, \ \ \text{M} \ \ ) \neq k \ \ \text{OR true} \\
\equiv & \quad \{ \text{ definition of } m \ \} \qquad\qquad \text{Let m be index such that A[m] = M} \\
 & r = k \ \wedge \ r < M \ \Rightarrow \ \textbf{max}\,(r, M) \neq k \\
\Leftarrow & \quad \{ \text{ predicate calculus } \} \\
 & M > k \ \Rightarrow \ \textbf{max}\,(r, M) \neq k \\
\Leftarrow & \quad \text{since max(r,M)>k => max(r,M) !=k} \\
 & M > k \ \Rightarrow \ \textbf{max}\,(r, M) > k \\
\equiv & \quad \{ \text{ property of } \textbf{max} \ \} \\
 & \textbf{true}
\end{array}
$$

Program	$FindMax$
var	A : array $0..N-1$ of int,
	r : int
initially	$r = A[0]$
assign	
	$(\ [\!] \ x \ : \ 0 \leq x \leq N-1 \ : \ r := max(r, A[x]) \ )$

FindMax Proof: Showing Termination

Because we changed the transient property, can't *directly* use Theorem 11

- Need to prove a variant that fits our situation
- (Good example of how the proof of Theorems 10-12 in Sivilotti can be carried out)

Show that true $\leadsto$ (r = M)

$$
\begin{array}{ll}
 & \textbf{true} \\
\equiv & \quad \{ \ \textbf{transient}\text{property established above} \ \} \\
 & \textbf{transient}.(r = k \ \wedge \ r < M) \\
\Rightarrow & \quad \{ \ \textbf{transient}.P \ \Rightarrow \ (P \ \leadsto \ \neg P) \ \} \\
 & r = k \ \wedge \ r < M \ \leadsto \ r \neq k \ \vee \ r \geq M \\
\Rightarrow & \quad \{ \ \textbf{stable}.(r \geq k) \ \} \\
 & r = k \ \wedge \ r < M \ \leadsto \ r > k \ \vee \ r \geq M \\
\equiv & \quad \{ \ [X \ \vee \ Y \ \equiv \ (\neg Y \ \wedge \ X) \ \vee \ Y] \ \} \\
 & r < M \ \wedge \ r = k \ \leadsto \ (r < M \ \wedge \ r > k) \ \vee \ r \geq M \\
\Rightarrow & \quad \{ \ \text{induction} \ \} \\
 & r < M \ \leadsto \ r \geq M \\
\equiv & \quad \{ \ \text{definition of } FP \ \} \\
 & r < M \ \leadsto \ FP \\
\equiv & \quad \{ \ \textbf{initially}.(r < M) \ \} \\
 & \textbf{true} \ \leadsto \ FP
\end{array}
$$

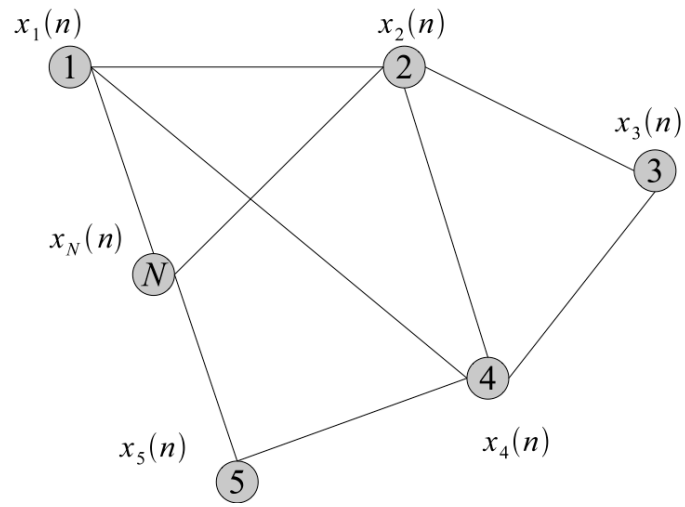
- Proved earlier than **stable**(r ≥ k)
- r = k $\Longrightarrow$ r can't get smaller
- r ≠ k eventually $\Longrightarrow$ r must eventually become > k

Program	$FindMax$
var	A : array $0..N-1$ of int,
	r : int
initially	$r = A[0]$
assign	
	$(\ [\!] \ x \ : \ 0 \leq x \leq N-1 \ : \ r := max(r, A[x]) \ )$

Example #2: Average Consensus

Program $AverageConsensus$
constant $N \quad \{number\ of\ agents\}$
$\mathcal{G} \quad \{interconnection\ graph\}$
$0 < \alpha < 1 \quad \{averaging\ factor\}$
var x : array of N numbers
assign

$$\big([]i,j : (i,j) \in \mathcal{G} : x_i := \alpha x_i + (1-\alpha)x_j$$
$$\| \; x_j := \alpha x_j + (1-\alpha)x_i)\big)$$



Structure of the proof

- Fixed point: identify the conditions under which the algorithm terminates
  - FP = {x_i = x_j for all pairs *i, j*}
  - Note that the fixed point *doesn't* say we reach the average
- Invariants: set of constraints on the behavior of the program
  - Claim: **invariant**(avg x) AND **invariant**(var x)
  - Avg *A* invariant $\Longrightarrow$ if we reach the fixed point, then we must have x_i = average(x)
- Metric: upper or lower bounded function used to track progress
  - Metric: variance = $\sum_i (x_i - A)^2$ [*A* = average of values]
  - Lower bounded by zero $\Longrightarrow$ if we can show it always decreases, we will be done
- Final result: show the for any ε, each x_i will eventually be within ε of the mean

Proof Obligations for AverageConsensus

1. If variance is non-zero then it decreases: $\forall K > 0 : V = K \rightsquigarrow V < K$

$$
\begin{aligned}
& V = K \rightsquigarrow V < K \\
\Longleftarrow \quad & \{\text{Definition of } \textbf{leadsto}\} \\
& V = K > 0 \;\textbf{ensures}\; V < K \\
\equiv \quad & \{\text{Definition of } \textbf{ensures}\} \\
& (V = K \wedge V \geq K)\;\textbf{next}\;(V = K \vee V \leq K) \wedge \textbf{transient}(V = K \wedge V \geq K) \\
\equiv \quad & \{\text{simpification}\} \\
& V = K \;\textbf{next}\; V \leq K \wedge \textbf{transient}(V = K) \\
\Longleftarrow \quad & \{\text{choose an action for some } x_i \neq x_j\} \\
& V = K \;\textbf{next}\; V \leq K \wedge \{V = K\} \quad x_i, x_j := y(x_i + x_j)/2, (x_i + x_j)/2 \quad \{V < K\} \\
\equiv \quad & \{\text{assignment axiom} + V \text{ decreases}\} \\
& V = K \;\textbf{next}\; V \leq K \wedge \textbf{true} \\
\equiv \quad & \underline{\qquad} \\
& V = K \;\textbf{next}\; V \leq K \\
\equiv \quad & \underline{\qquad} \\
& (\forall a :: \{V = K\} \quad a \quad \{V \leq K\} \\
\equiv \quad & \underline{\qquad} \\
& \textbf{true}
\end{aligned}
$$

Program	$AverageConsensus$
constant	$N \quad \{number\ of\ agents\}$
	$\mathcal{G} \quad \{interconnection\ graph\}$
	$0 < \alpha < 1 \quad \{averaging\ factor\}$
var	x : array of N numbers
assign	

$$
\big([\!] i, j : (i, j) \in \mathcal{G} : x_i := \alpha x_i + (1 - \alpha) x_j \;\|\; x_j := \alpha x_j + (1 - \alpha) x_i \big)
$$

Proof Obligations for AverageConsensus

2. Variance decreases by geometric factor α: $\forall K > 0 : V = K \rightsquigarrow V < \beta K$

- Claim: ∃ j such that $(x_j - A)^2 \geq \sqrt{K/N}$ {if not, then can't have V = K)
- Assume $x_j > A$ and find some k such that $x_k < A$ (must exist since A = average)
- Now sort all of the variables in decreasing order

$$x_{i_1} \geq x_{i_2} \geq \cdots \geq x_j \geq \cdots \geq x_k \geq \cdots \geq x_{i_n}$$

- Claim: there exists adjacent indices u, v such that $x_u - x_v \geq (x_j - x_k)/N$
  - Worst case is that all numbers between x_j and x_k are evenly spaced ⇒ $(x_j - x_k)/N$
- For these indices we have that

$$(x_u - x_v) \geq \frac{x_j - x_k}{N} \implies (x_u - x_v)^2 \geq \frac{(x_j - x_k)^2}{N^2} \geq \frac{K}{N^3} = \frac{V}{N^3}$$

- Next: show that replacing any pair by average reduces variance by factor of $\beta = 1/N^3$
- Represent out list in decreasing order, calling out x_u and x_v

$$x_{i_1} \geq x_{i_2} \geq \cdots \geq x_u \geq x_v \geq x_{i_{n-1}} \cdots \geq x_{i_n}$$

- Claim: $V = K \rightsquigarrow V < \beta K$
  - If we switch any pairs with indices $i, j \leq u$ or $i, j \geq v$ then bounds remain unchanged
  - If we switch u, v or any pairs "outside" u, v then we get reduction by at least β

Proof Obligations for AverageConsensus

1. If variance is non-zero then it decreases: $\forall K > 0 : V = K \rightsquigarrow V < K$

- This is stronger than invariance of K since it says that the variance will get *smaller*
- Doesn't bound how fast the decrease will occur => we may never terminate

2. Variance decreases by geometric factor α: $\forall K > 0 : V = K \rightsquigarrow V < \beta K$

- Since V decreases by geometric factor, can show (eventually) have V arbitrarily small

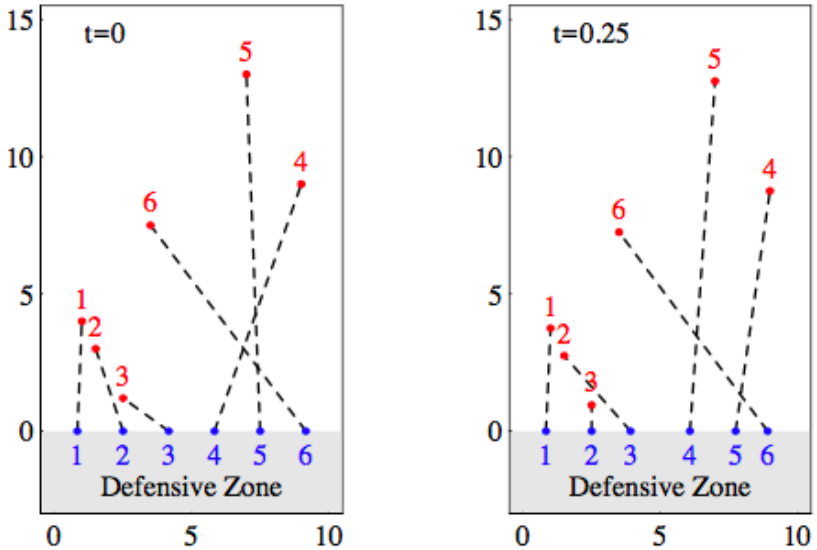
3. Final result - variance can be made arbitrarily small: $true \rightsquigarrow V < \varepsilon$

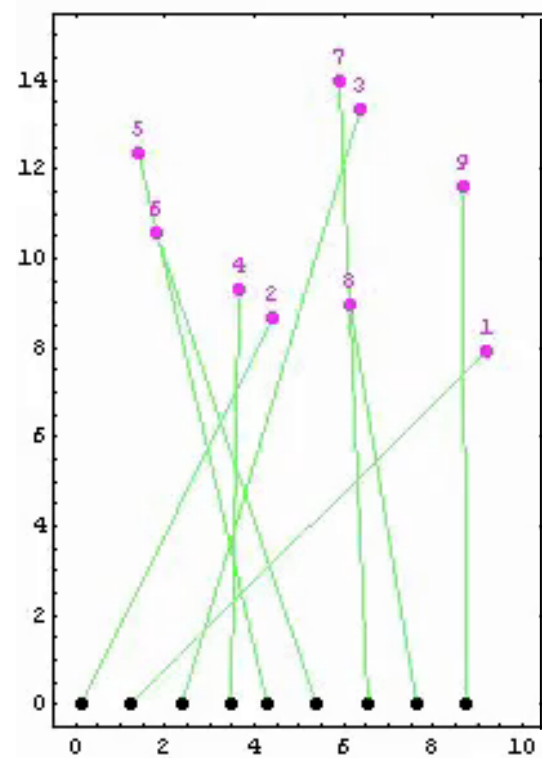
- From (2), we have that $V = K \rightsquigarrow V < \beta K \rightsquigarrow V < \beta^2 K \rightsquigarrow V < \beta^3 K \ldots$
- Choose m such that $\beta^m < \varepsilon \Rightarrow$ after m "iterations" (of leads-to) we will achieve bound

Going back to the overall structure of the proof

- Fixed point: FP = {x[i] = x[j] for all pairs i, j}
- Invariants: average and variance
  - Avg invariant => if we reach the fixed point, then we must have x[i] = average(x)
- Metric: variance = \sum_i {(x[i] - M)^2}
  - Lower bounded by zero => if we can show it always decreases, we will be done
- Final result: show the for any ε, each x[i] will eventually be within ε of the mean

RoboFlag Drill





Program $P_{red}(i)$

Initial	$x_i \in [min, max] \wedge y_i > \delta$
Commands	$y_i - \delta > 0 \quad : \quad y_i' = y_i - \delta$

Program $P_{blue}(i)$

Initial	$z_i \in [min, max] \;\wedge\; z_i < z_{i+1}$
Commands	$z_i < x_{\alpha(i)} \wedge z_i < z_{i+1} - \delta \quad : \quad z_i' = z_i + \delta$
	$z_i > x_{\alpha(i)} \wedge z_i > z_{i-1} + \delta \quad : \quad z_i' = z_i - \delta$

r(i, j) = 1 if defender i cannot reach incoming robot i in time to intercept

Program $P_{proto}(i)$

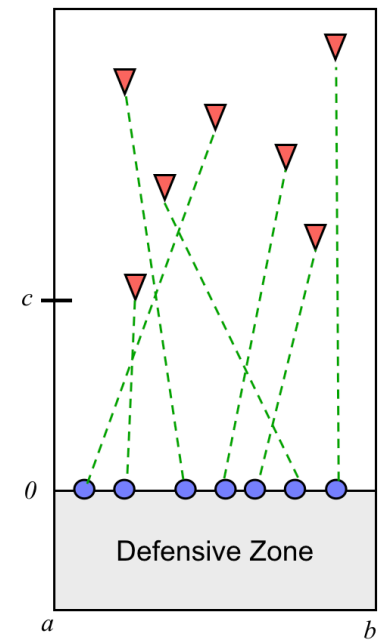
Initial	$\alpha(i) \neq \alpha(j)$ if $i \neq j$
Commands	$switch(i, i+1) \quad : \quad \alpha(i)' = \alpha(i+1)$
	$\alpha(i+1)' = \alpha(i)$

Will switching increase the number of incoming robots we can intercept?

$$r(i,j) \triangleq \begin{cases} 1 \text{ if } y_{\alpha(j)} < |z_i - x_{\alpha(j)}| - \delta \\ 0 \text{ otherwise.} \end{cases}$$

$$\begin{aligned} switch(i,j) \;\triangleq\; & r(i,j) + r(j,i) < r(i,i) + r(j,j) \\ \vee\; & (r(i,j) + r(j,i) = r(i,i) + r(j,j) \\ & \wedge\, x_{\alpha(i)} > x_{\alpha(j)}. \end{aligned}$$

Properties for RoboFlag program



Safety (Defenders do not collide) [invariant]

$$z_i < z_{i+1} \ \mathbf{next}\ z_i < z_{i+1}$$

Stability (switch predicate stays false) [fixed point]

$$\forall i \ .\ \underbrace{y_i > 2\delta \wedge z_i + 2\delta < z_{i+1}} \wedge \neg switch_{i,i+1} \ \mathbf{next}\ \neg switch_{i,i+1}$$

Robots are "far enough" apart.

Progress (we eventually reach a fixed point) [metric]

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

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

- *V* implements *lexicographic* ordering: $(\rho_1, \beta_1) > (\rho_2, \beta_2)$ if $\rho_1 > \rho_2$ or $\rho_1 = \rho_2 \wedge \beta_1 > \beta_2$
- Can show that V always decreases whenever a switch occurs

$$\forall i \ .\ z_i + 2\delta m < z_{i+1} \wedge \exists j \ .\ switch_{j,j+1} \wedge V = m \ \ \mathbf{next}\ \ V < m$$

Summary: Reasoning about Programs

Key elements of a specification

- Safety: properties that should always be true
- Progress: properties that should eventually be true

Key elements of a proof

- Fixed points: points at which the computation terminates
- Invariants: properties preserved during execution
- Metric: bounded function used to measure progress

What's next:

- Move from non-deterministic computation (UNITY) to distributed computation (still UNITY, but w/ messages)