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
Example: FindMax

Program: `FindMax`

```
var
  A: array 0..N-1 of int,
  r: int
initially
  r = A[0]
assign
  ( \[ x : 0 \leq x \leq N - 1 : r := \text{max}(r, A[x]) \] )
```

Specification
- Safety: \(\text{stable}(r = M)\)
- Progress: \(\text{true} \leadsto (r = M)\)

Structure of the proof
- **Fixed point**: identify the conditions under which the algorithm terminates
  - \(FP \equiv (\forall x: 0 \leq x \leq N - 1 : r = \text{max}(r, A[x]))\)
    \[ \equiv r \geq (\text{Max } x: 0 \leq x \leq N - 1 : A[x]) \]
    \[ \equiv r \geq M \]
- **Invariant**: set of constraints on the behavior of the program
  - \(\text{invariant.}(r \leq M)\)
    - 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 $\Rightarrow (r = M)$
- Use restricted form of induction theorem (Sivilotti, Section 3.5)

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

$$(\forall m :: P \land L \downarrow = m \text{ next } (P \land L \downarrow \geq m) \lor Q)$$

$$\land (\forall m :: \text{transient.}(P \land L \downarrow = m))$$

$$\Rightarrow P \leadsto Q$$

- 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 $r \geq m \lor r = M$ is true for all actions (by def’n of max) => just need to show transient property

**Program**

<table>
<thead>
<tr>
<th>FindMax</th>
</tr>
</thead>
<tbody>
<tr>
<td>var</td>
</tr>
<tr>
<td>$A : \text{array } 0..N - 1 \text{ of int,}$</td>
</tr>
<tr>
<td>$r : \text{int}$</td>
</tr>
<tr>
<td>initially</td>
</tr>
<tr>
<td>$r = A[0]$</td>
</tr>
<tr>
<td>assign</td>
</tr>
<tr>
<td>( $\parallel x : 0 \leq x \leq N - 1 : r := \text{max}(r, A[x])$ )</td>
</tr>
</tbody>
</table>
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 $r < M$

\[
\text{transient}(r = k \land r < M) \\
\equiv \{ \text{definition of transient} \} \\
( \exists a : (r = k \land r < M) \land (r \neq k \lor r \geq M) ) \\
\iff \{ \text{definition of program} \} \text{ with } m \text{ selected so that } A[m] = M \\
\{ r = k \land r < M \} \land r := \max(r, M) \land (r \neq k \lor r \geq M) \\
\equiv \{ \text{assignment axiom} \} \\
\begin{align*}
M > k & \Rightarrow \max(r, M) \neq k \\
\text{since } \max(r, M) > k & \Rightarrow \max(r, M) \neq k \\
M > k & \Rightarrow \max(r, M) > k
\end{align*}
\]

\[
\equiv \{ \text{from definition of max} \} \\
\begin{align*}
r = k \land r < M & \Rightarrow \max(r, M) \neq k \lor \text{true} \\
\implies \{ \text{definition of } m \} \\
r = k \land r < M & \Rightarrow \max(r, M) \neq k
\end{align*}
\]

Let $m$ be index such that $A[m] = M$

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

\[
\begin{align*}
\text{true} & \equiv \{ \text{transient property established above} \} \\
\text{transient} & \implies (r = k \land r < M) \\
\implies & \{ \text{transient} \quad P \implies (P \leadsto \neg P) \} \\
& (r = k \land r < M) \leadsto r \neq k \lor r \geq M \\
& \{ \text{stable} \quad (r \geq k) \} \\
& r = k \land r < M \leadsto r > k \lor r \geq M \\
& \equiv \{ [X \lor Y \equiv (\neg Y \land X) \lor Y] \} \\
& r < M \land r = k \leadsto (r < M \land r > k) \lor r \geq M \\
& \implies \{ \text{induction} \} \\
& r < M \leadsto r \geq M \\
& \equiv \{ \text{definition of} \quad FP \} \\
& r < M \leadsto FP \\
& \equiv \{ \text{initially} \quad (r < M) \} \\
& \text{true} \leadsto FP
\end{align*}
\]

- Proved earlier than stable$(r \geq k)$
- $r = k \implies r$ can’t get smaller
- $r \neq k$ eventually $\implies r$ must eventually become $> k$

Program

\[
\begin{align*}
\text{FindMax} & \\
\text{var} & \quad A : \text{array} \quad 0..N - 1 \quad \text{of int}, \quad r : \text{int} \\
\text{initially} & \quad r = A[0] \\
\text{assign} & \quad (\quad x : \quad 0 \leq x \leq N - 1 : \quad r := \max(r, A[x]) \quad )
\end{align*}
\]
Example #2: Average Consensus

Program $\text{AverageConsensus}$

constant $N \{\text{number of agents}\}$

$G \{\text{interconnection graph}\}$

$0 < \alpha < 1 \{\text{averaging factor}\}$

var $x : \text{array of } N \text{ numbers}$

assign

$([i, j] : (i, j) \in G : x_i := \alpha x_i + (1 - \alpha) x_j$

$\| x_j := \alpha x_j + (1 - \alpha) x_i)$

Structure of the proof

• Fixed point: identify the conditions under which the algorithm terminates
  
  - FP = $\{x_i = x_j \text{ for all pairs } i, j\}$
  
  - Note that the fixed point doesn’t say we reach the average

• Invariants: set of constraints on the behavior of the program
  
  - Claim: $\text{invariant}(\text{avg } x) \text{ AND } \text{invariant}(\text{var } x)$
    
  - Avg $A$ invariant $\implies$ if we reach the fixed point, then we must have $x_i = \text{average}(x)$

• Metric: upper or lower bounded function used to track progress
  
  - Metric: variance $= \sum_i (x_i - A)^2 \ [A = \text{average of values}]$

  - Lower bounded by zero $\implies$ if we can show it always decreases, we will be done

• Final result: show the for any $\varepsilon$, each $x_i$ will eventually be within $\varepsilon$ of the mean
Proof Obligations for AverageConsensus

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

\[
V = K \leadsto V < K
\]
\[
\iff \{\text{Definition of leadsto}\}
V = K > 0 \text{ ensures } V < K
\]
\[
\equiv \{\text{Definition of ensures}\}
(V = K \land V \geq K) \text{ next } (V = K \lor V \leq K) \land \text{transient}(V = K \land V \geq K)
\]
\[
\equiv \{\text{simpification}\}
V = K \text{ next } V \leq K \land \text{transient}(V = K)
\]
\[
\iff \{\text{choose an action for some } x_i \neq x_j\}
V = K \text{ next } V \leq K \land \{V = K\} \quad x_i, x_j := y(x_i + x_j)/2, (x_i + x_j)/2 \quad \{V < K\}
\]
\[
\equiv \{\text{assignment axiom } + V \text{ decreases}\}
V = K \text{ next } V \leq K \land \text{true}
\]
\[
\equiv \{\text{ }\}
V = K \text{ next } V \leq K
\]
\[
\equiv \{\text{ }\}
(\forall a : \{V = K\} \land \{V \leq K\}
\]
\[
\equiv \{\text{ }\}
\text{true}
\]
Proof Obligations for AverageConsensus

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

- Claim: $\exists j$ 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 $\Rightarrow$ $(x_j-x_k)/N$

- 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: $V = K \leadsto V < \beta K$
  - 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 $\beta$
Proof Obligations for AverageConsensus

1. If variance is non-zero then it decreases: $\forall K > 0 : V = K \leadsto V < K$
   - 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 $\Rightarrow$ we may never terminate

2. Variance decreases by geometric factor $\alpha$: $\forall K > 0 : V = K \leadsto V < \beta K$
   - Since $V$ decreases by geometric factor, can show (eventually) have $V$ arbitrarily small

3. Final result - variance can be made arbitrarily small: true $\leadsto V < \varepsilon$
   - From (2), we have that $V = K \leadsto V < \beta K \leadsto V < \beta^2 K \leadsto V < \beta^3 K \ldots$
   - 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 $\Rightarrow$ if we reach the fixed point, then we must have $x[i] = \text{average}(x)$
- **Metric**: variance $= \sum_i (x[i] - M)^2$
  - Lower bounded by zero $\Rightarrow$ if we can show it always decreases, we will be done
- **Final result**: show the for any $\varepsilon$, each $x[i]$ will eventually be within $\varepsilon$ of the mean
RoboFlag Drill

**Program** $P_{\text{red}}(i)$

<table>
<thead>
<tr>
<th>Initial</th>
<th>$x_i \in [\min, \max] \land y_i &gt; \delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commands</td>
<td>$y_i - \delta &gt; 0$ : $y'_i = y_i - \delta$</td>
</tr>
</tbody>
</table>

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

$r(i, j) = 1$ if $y_{\alpha(j)} < |z_i - x_{\alpha(j)}| - \delta$

0 otherwise.

**Program** $P_{\text{blue}}(i)$

<table>
<thead>
<tr>
<th>Initial</th>
<th>$z_i \in [\min, \max] \land z_i &lt; z_{i+1}$</th>
</tr>
</thead>
</table>
| Commands | $z_i < x_{\alpha(i)} \land z_i < z_{i+1} - \delta$ : $z'_i = z_i + \delta$
|          | $z_i > x_{\alpha(i)} \land z_i > z_{i-1} + \delta$ : $z'_i = z_i - \delta$ |

Will switching increase the number of incoming robots we can intercept?
Properties for RoboFlag program

Safety (Defenders do not collide) [invariant]

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

Stability (switch predicate stays false) [fixed point]

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

Robots are "far enough" apart.

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

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

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

\[
\forall i . \; z_i + 2\delta m < z_{i+1} \land \exists j . \; \text{switch}_{j,j+1} \land V = m \quad \text{next} \quad V < m
\]
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)