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
\begin{align*}
\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 ( \ | \ x : 0 \leq x \leq N - 1 : r := \text{max}(r, A[x]) )
\end{align*}

Specification
\begin{itemize}
\item Safety: stable(r = M)
\item Progress: true \leadsto (r = M)
\end{itemize}

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

**Safety:** stable\((r = M)\) (once we reach the fixed point we will stay there => terminate)
- Shown in homework #2

**Progress:** true $\leadsto (r = M)$
- Use restricted form of induction theorem (Sivilotti, Section 3.5)

\[
\text{Theorem 11 (Restricted Form of Induction for } \leadsto \text{). For a metric } L
\]

\[
(\forall m :: P \land L \\geq m \text{ next } (P \land L \\geq m) \lor Q) \\
\land (\forall m :: \text{ transient.}(P \land L = m))
\]

$\Rightarrow P \leadsto Q$

- For FindMax, we take $P = \text{true}$, $L = r$, $Q = \{r = M\}$ [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) $\Rightarrow$
  just need to show transient property

\[
\begin{array}{|l|l|}
\hline
\text{Program} & \text{FindMax} \\
\hline
\text{var} & \text{A : array 0..N-1 of int,} \\
& \text{r : int} \\
\text{initially} & \text{r = A[0]} \\
\text{assign} & \text{( } \left[ x : 0 \leq x \leq N-1 : r := max(r, A[x]) \right) } \\
\hline
\end{array}
\]
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 $r < M$

$$\text{transient.}(r = k \land r < M)$$

$$\equiv \{ \text{definition of transient} \}$$

$$\left( \exists a :: \{ r = k \land r < M \} \quad a \quad \{ r \neq k \lor r \geq M \} \right)$$

$$\iff \{ \text{definition of program} \}$$

$$\{ r = k \land r < M \} \quad r := \max(r, M) \quad \{ r \neq k \lor r \geq M \}$$

$$\equiv \{ \text{assignment axiom} \}$$

$$r = k \land r < M \implies \max(r, A[m]) \neq k \lor \max(r, A[m]) \geq M$$

$$\iff \{ \text{weakening antecedent} \}$$

$$r = k \land r < M \implies \max(r, A[m]) \neq k$$

$$\equiv \{ \text{definition of } m \}$$

$$r = k \land r < M \implies \max(r, M) \neq k$$

$$\iff \{ \text{properties of inequalities} \}$$

$$M > k \implies \max(r, M) \neq k$$

$$\iff \{ \text{weakening antecedent} \}$$

$$M > k \implies \max(r, M) > k$$

$$\equiv \{ \text{property of } \max \}$$

true

---

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

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

Mani has some issues with this version: antecedent versus consequent?

See alternative version at the end.
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 $\leadsto (r = M)$

true

$\equiv \{ \text{transient property established above} \}$

transient.$(r = k \land r < M)$

$\Rightarrow \{ \text{transient.} P \Rightarrow (P \leadsto \neg P) \}$

$r = k \land r < M \leadsto r \neq k \lor r \geq M$

$\Rightarrow \{ \text{stable.} (r \geq k) \}$

$r = k \land r < M \leadsto r > k \lor r \geq M$

$\equiv \{ \ [X \lor Y \equiv (\neg Y \land X) \lor Y] \}$

$r < M \land r = k \leadsto (r < M \land r > k) \lor r \geq M$

$\Rightarrow \{ \text{induction} \}$

$r < M \leadsto r \geq M$

$\equiv \{ \text{definition of } FP \}$

$r < M \leadsto FP$

$\equiv \{ \text{initially.} (r < M) \}$

true $\leadsto FP$

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

Program \textit{AverageConsensus}

\textbf{constant} \quad N \quad \{number of agents\}
\textbf{G} \quad \{interconnection graph\}
0 < \alpha < 1 \quad \{averaging factor\}

\textbf{var} \quad x : \text{array of } N \text{ numbers}

\textbf{assign}

\begin{align*}
(\forall i, j : (i, j) \in G : x_i &:= \alpha x_i + (1 - \alpha)x_j \\
\| x_j &:= \alpha x_j + (1 - \alpha)x_i) \end{align*}

\textbf{Structure of the proof}

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

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

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

Program **AverageConsensus**

- **constant**: $N$ \{number of agents\}
- **G**: \{interconnection graph\}
- **0 < \alpha < 1**: \{averaging factor\}
- **var**: $x$ : array of $N$ numbers
- **assign**
  \[
  \begin{array}{l}
  (\forall i, j : (i, j) \in G : x_i := \alpha x_i + (1 - \alpha)x_j \\
  \parallel x_j := \alpha x_j + (1 - \alpha)x_i)
  \end{array}
  \]
Proof Obligations for AverageConsensus

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

- 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 $A = \text{average}$)
- 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: $V = K \sim V < \beta K$
  - 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 => 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 $V = K \leadsto V < \beta K \leadsto V < \beta^2 K \leadsto V < \beta^3 K ...$
   • 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: $FP = \{x[i] = x[j] \text{ for all pairs } i, j\}$
• Invariants: average and variance
  - Avg invariant $\Rightarrow$ if we reach the fixed point, then we must have $x[i] = \text{average}(x)$
• Metric: $\text{variance} = \sum_i (x[i] - M)^2$
  - 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

\[ r(i, j) = 1 \text{ if defender } i \text{ cannot reach incoming robot } j \text{ in time to intercept} \]

Program \( P_{\text{red}}(i) \)
\[
\] Inital \( x_i \in [\text{min}, \text{max}] \land y_i > \delta \)
Commands \( y_i - \delta > 0 : y_i' = y_i - \delta \)

Program \( P_{\text{blue}}(i) \)
\[
\] Inital \( z_i \in [\text{min}, \text{max}] \land z_i < z_{i+1} \)
Commands \( z_i < x_{\alpha(i)} \land z_i < z_{i+1} - \delta : z_i' = z_i + \delta \)
\[ z_i > x_{\alpha(i)} \land z_i > z_{i-1} + \delta : z_i' = z_i - \delta \]

Program \( P_{\text{proto}}(i) \)
\[
\] Inital \( \alpha(i) \neq \alpha(j) \text{ if } i \neq j \)
Commands \( \text{switch}(i, i+1) : \alpha(i)' = \alpha(i+1) \)
\[ \alpha(i+1)' = \alpha(i) \]

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

\[ r(i, j) \triangleq \begin{cases} 
1 & \text{if } y_{\alpha(j)} < |z_i - x_{\alpha(j)}| - \delta \\
0 & \text{otherwise.} 
\end{cases} \]
Properties for RoboFlag program

Safety (Defenders do not collide) [invariant]

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

Stability (switch predicate stays false) [fixed point]

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

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 \textit{metric} that captures “energy” of current state (\( V = 0 \) is desired)

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

- \( V \) implements \textit{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

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

transient.($r = k$ ∧ $r < M$)

≡

{ definition of transient }

(∃a :: {r = k ∧ r < M} a \{r ≠ k ∨ r ≥ M\})

≡

{ definition of program } with m selected so that $A[m] = M$

{r = k ∧ r < M} r := max(r, M) \{r ≠ k ∨ r ≥ M\}

≡

{ assignment axiom }

$r = k ∧ r < M$ ⇒ max (r, M) ≠ k ∨ max (r, A[m]) ≥ M

≡

{ from definition of max }

$r = k ∧ r < M$ ⇒ max (r, M) ≠ k OR true

≡

{ definition of max }

$r = k ∧ r < M$ ⇒ max (r, M) ≠ k

≡

{ predicate calculus }

$M > k$ ⇒ max (r, M) ≠ k

≡

since max(r,M) > k => max(r,M) ! = k

$M > k$ ⇒ max (r, M) > k

≡

{ property of max }

true

Program FindMax

var A : array 0..N - 1 of int,

r : int

initially $r = A[0]$

assign ($\parallel x : 0 ≤ x ≤ N - 1 : r := max(r, A[x])$)