Caltech

CS/IDS 142: Lecture 3.1
Progress Properties and Metrics

Richard M. Murray

14 October 2019

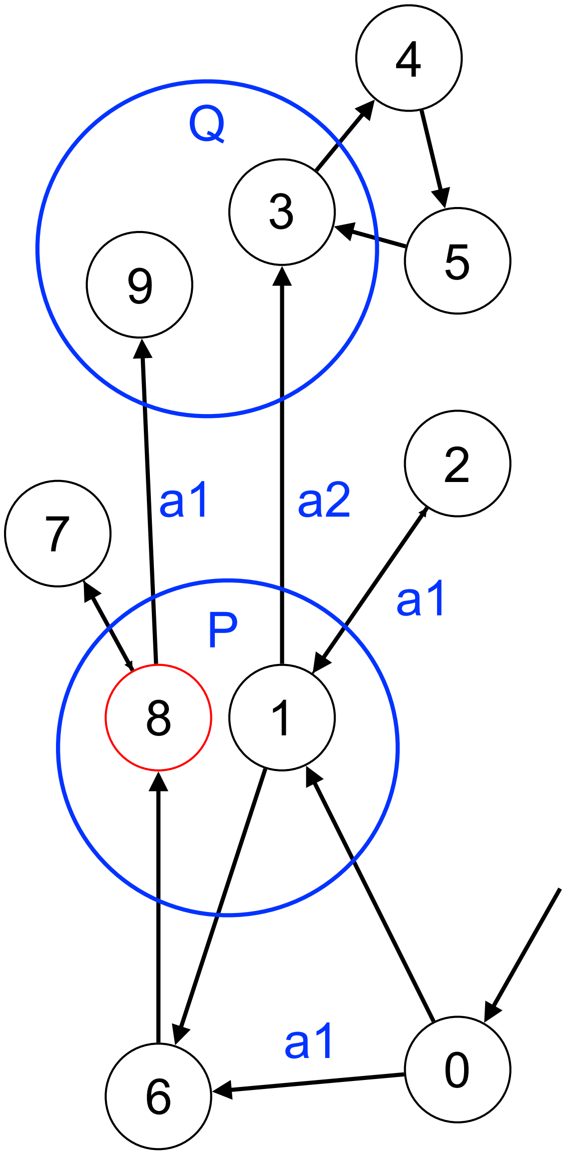
Goals:

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

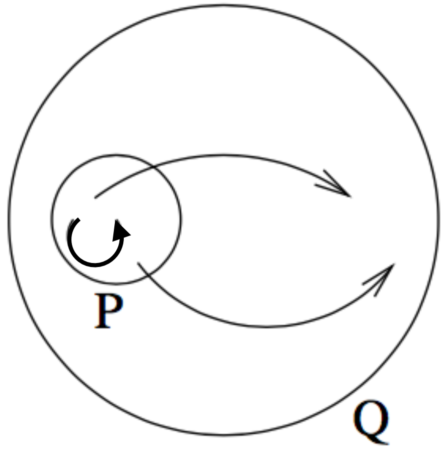
Reading:

- P. Sivilotti, *Introduction to Distributed Algorithms*, Section 3.5

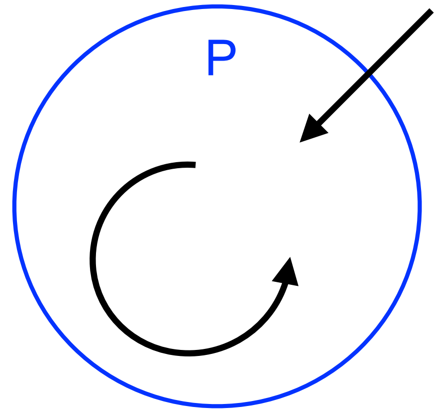
Summary: Reasoning About Programs



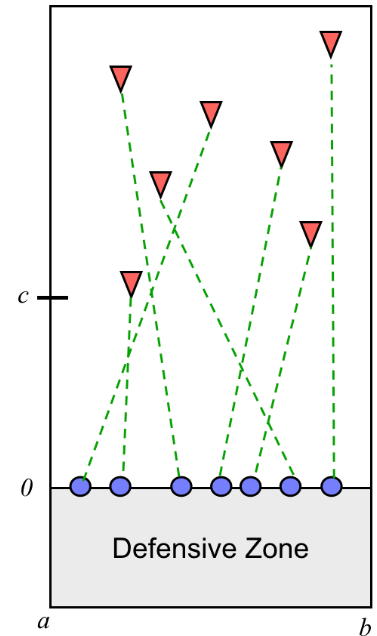
Hoare triple: {P} a {Q}



P **next** Q



stable(P)



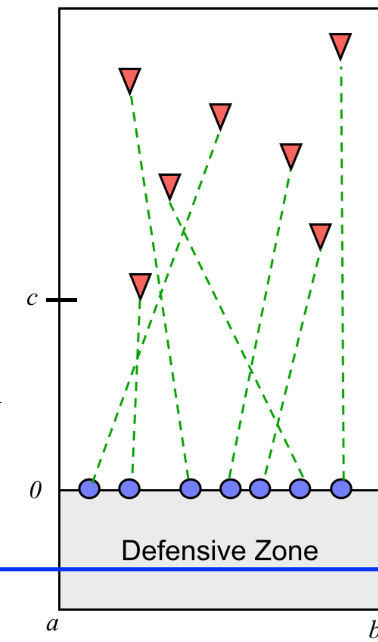
Initial tools for reasoning about program properties

- 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**:

$$(\forall a \,:\, a \in G \,:\, \{P\} \quad a \quad \{Q\}\,)$$

- Stability: **stable**(P) ≡ P **next** P
- Invariants: $\textbf{invariant}(P) \equiv \textbf{initially}(P) \wedge \textbf{stable}(P)$

Properties for RoboFlag program



Safety (Defenders do not collide)

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

Stability (switch predicate stays false)

$$\forall i \;.\; \underbrace{y_i > 2\delta \wedge z_i + 2\delta < z_{i+1}}_{\text{Robots are "far enough" apart.}} \wedge \neg switch_{i,i+1} \textbf{ next } \neg switch_{i,i+1}$$

Progress (we eventually reach a fixed point)

- 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)

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

- Can show that V always decreases whenever a switch occurs

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

Next week

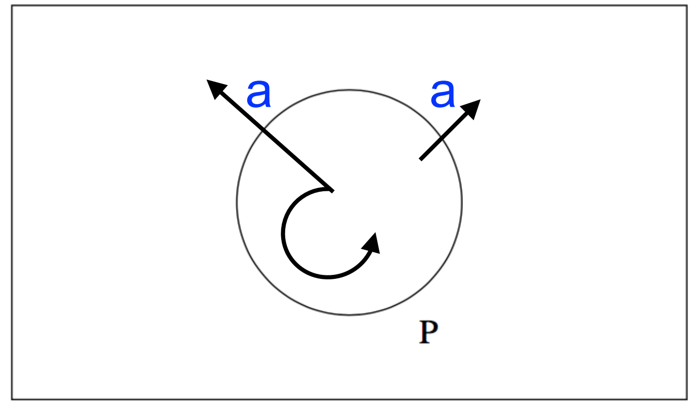
The 'Transient' Property

Definition

- Informally: "if P becomes true at some point in the computation, it is guaranteed to become false at some later point ⇒ P is false infinitely often" [not quite accurate]

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

- Compare to **next**: use ∃ instead of ∀
- 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

Program	*Count*
var	n : natural number
initially	$n = 0$
assign	
$n := n + 1$	

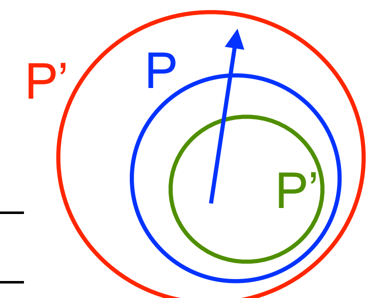
$$\textbf{transient}(n = 1) \equiv \textbf{true}$$

$$\textbf{transient}(n = 0) \equiv \textbf{true}$$

$$\textbf{transient}(n = 0 \vee n = 1) \equiv \textbf{false}$$

Which of the following hold (show formally in HW #3):

- Weakening: $\textbf{transient}(P) \wedge [P \implies P'] \implies \textbf{transient}(P')$ ______
- Strengthening: $\textbf{transient}(P) \wedge [P' \implies P] \implies \textbf{transient}(P')$ ______
- Intuition: remember that P' ⇒ P (formula) is same as P' ⊆ P (for the program graph)



The 'Ensures' Property

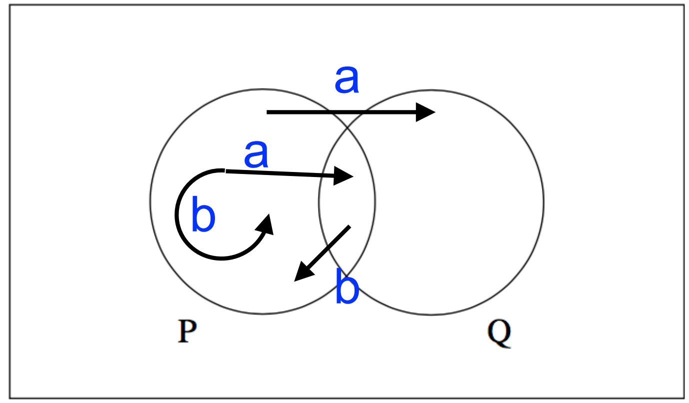
Definition

- If P holds, it will continue to hold as long as Q doesn't hold AND eventually Q holds

$$P \ \textbf{ensures} \ Q \ \equiv \ ((P \wedge \neg Q) \ \textbf{next} \ (P \vee Q)) \wedge \textbf{transient}.(P \wedge \neg Q)$$

Example

Program	*CountIfSmall*
var	n : natural number
initially	$n = 0$
assign	
$n \leq 2 \rightarrow n := n + 1$	



$(n = 1 \vee n = 2)$ **ensures** $(n \geq 2)$? ______

$n = 1$ **ensures** $n = 3$? ______

Some properties

- Weakening: (P **ensures** Q) ∧ [Q ⇒ R] ⇒ (P **ensures** R)
- Disjunction: (P **ensures** Q) ⇒ (P ∨ R) **ensures** (Q ∨ R)

Remarks

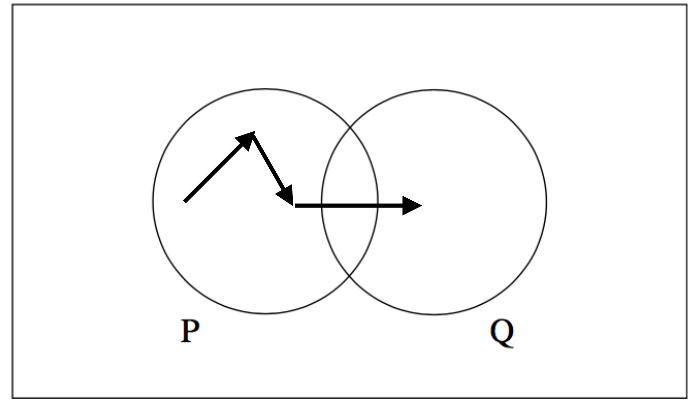
- Ensures is still "low level": defines properties at the level of *single actions*

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

$$
\begin{aligned}
P \textbf{ ensures } Q \quad &\Rightarrow \quad P \leadsto Q \\
(P \leadsto Q) \wedge (Q \leadsto R) \quad &\Rightarrow \quad P \leadsto R \\
(\forall i :: P_i \leadsto Q\ ) \quad &\Rightarrow \quad (\exists i :: P_i\ ) \leadsto Q
\end{aligned}
$$



Example

Program	$CountIfSmall$
var	n : natural number
initially	$n = 0$
assign	
	$n \leq 2 \rightarrow n := n + 1$

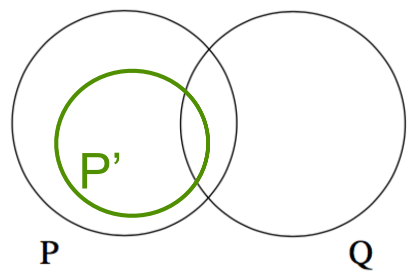
$(n = 1 \vee n = 2) \leadsto (n \geq 2)$? ______

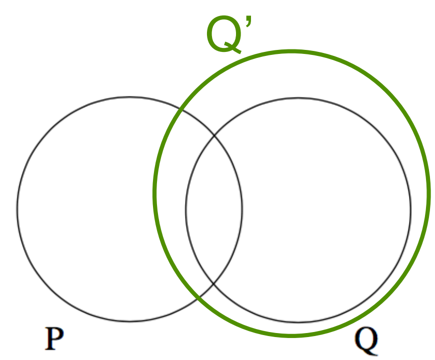
$(n = 1) \leadsto (n = 3)$? ______

Which of the following is true?

$(P \leadsto Q) \wedge [P' \Rightarrow P] \Rightarrow P' \leadsto Q$ ______

$(P \leadsto Q) \wedge [Q \Rightarrow Q'] \Rightarrow P \leadsto Q'$ ______





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

- $(P \leadsto Q) \wedge (R \leadsto Q) \Rightarrow (P \vee R) \leadsto Q$ ______
- $(P \leadsto Q) \wedge (P \leadsto R) \Rightarrow P \leadsto (Q \wedge R)$ ______
- $(P \leadsto Q) \wedge (P' \leadsto Q') \Rightarrow (P \wedge P') \leadsto (Q \wedge Q')$ ______

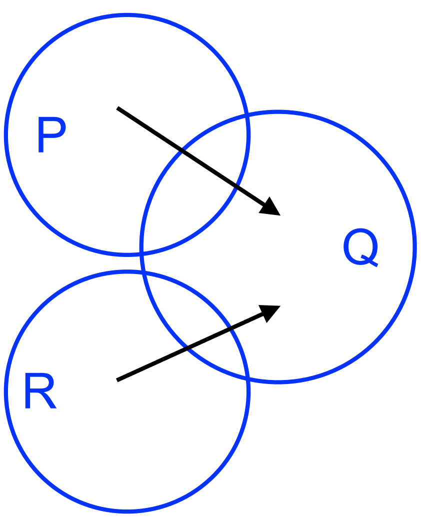
Stable Strengthening

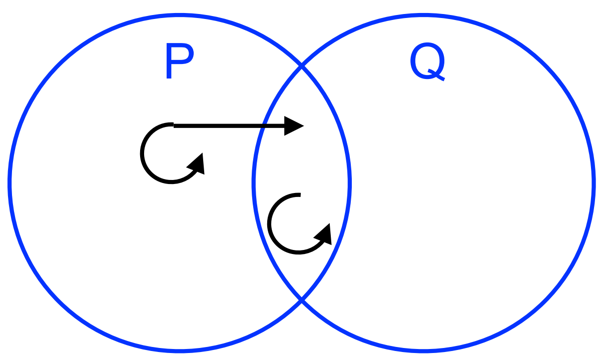
- $\textbf{stable}.P \wedge \textbf{transient}.(P \wedge \neg Q) \Rightarrow P \leadsto (P \wedge Q)$ ______

Progress-Safety-Progress (PSP)

- $(P \leadsto Q) \wedge (R \ \textbf{next} \ S) \Rightarrow (P \wedge R) \leadsto ((R \wedge Q) \vee (\neg R \wedge S))$

______



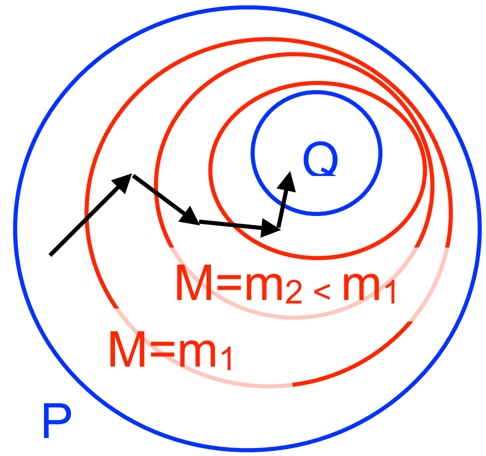


- PSP allow us to combine a safety proof with a progress proof
- Either stay in R and satisfy Q or move out of R and satisfy S
- Very useful in progress proofs

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)



Theorem 10 (Induction for $\rightsquigarrow$). *For a metric M,*

$$(\forall m :: \; P \wedge M = m \rightsquigarrow (P \wedge M < m) \vee Q \;) \quad \Rightarrow \quad P \rightsquigarrow Q$$

- 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

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

$$\begin{aligned} &(\forall m :: \; P \wedge M = m \;\textbf{next}\; (P \wedge M \leq m) \vee Q \;) \\ &\wedge \; (\forall m :: \; \textbf{transient}.(P \wedge M = m) \;) \\ \Rightarrow \quad & P \rightsquigarrow Q \end{aligned}$$

OK for M to remain = m, as long as there is *some* action that decreases m

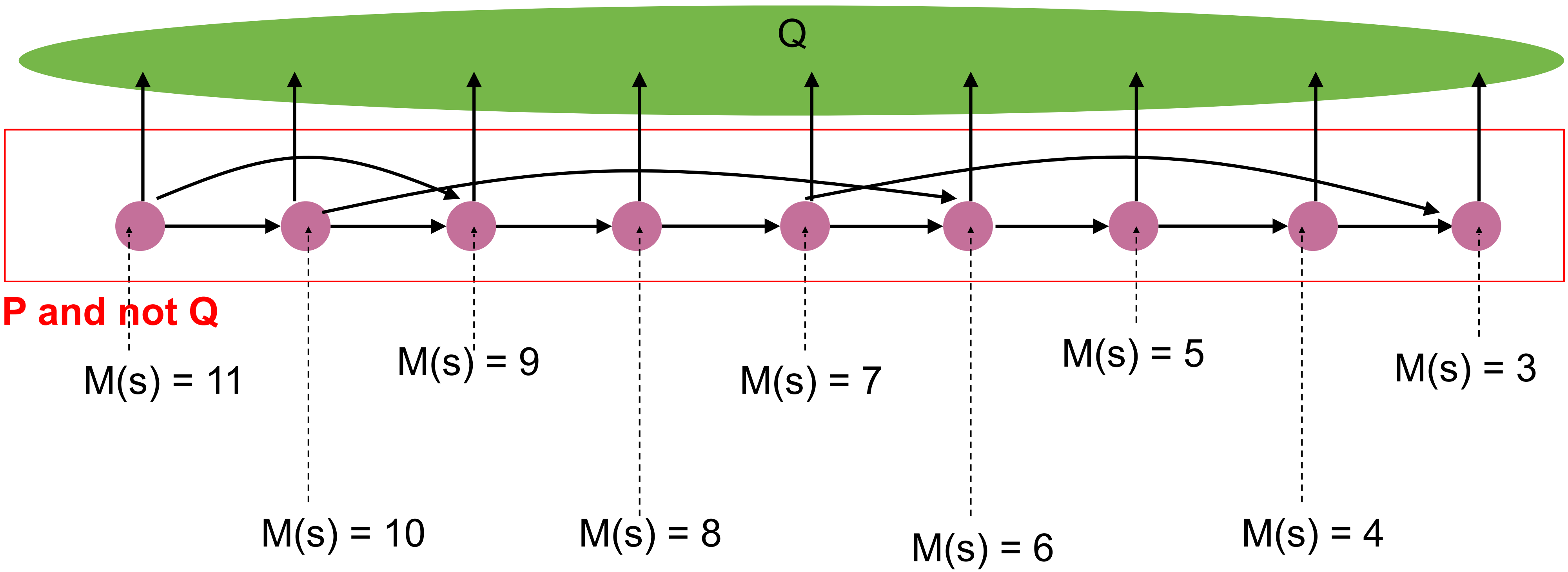
Reasoning about Fixed Points

Variant: show that all *enabled* actions decrease the metric

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

$$
\begin{aligned}
& (\forall i, m :: \{P \wedge M = m \wedge g_i\} \quad g_i \longrightarrow a_i \quad \{(P \wedge M < m) \vee Q\} \ ) \\
& \wedge \ (\forall i :: \neg g_i \ ) \Rightarrow Q \\
\Rightarrow \quad & P \leadsto Q
\end{aligned}
$$

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



Example: FindMax

$$\begin{array}{lll}
\textbf{Program} & FindMax & \\
\textbf{var} & A : \text{array } 0..N-1 \text{ of int}, & \\
 & r : \text{int} & \\
\textbf{initially} & r = A[0] & \\
\textbf{assign} & & \\
\multicolumn{3}{l}{\qquad (\ \| \ x \ : \ 0 \le x \le N-1 \ : \ r := max(r, A[x]) \ )}
\end{array}$$

$$\begin{array}{ll}
 & \textbf{transient}.(r = k \ \wedge \ r < M) \\
\Rightarrow & \qquad \{ \ \ \textbf{transient}.P \ \Rightarrow \ (P \ \rightsquigarrow \ \neg P) \ \ \} \\
 & r = k \ \wedge \ r < M \ \rightsquigarrow \ r \ne k \ \vee \ r \ge M \\
\Rightarrow & \qquad \{ \ \ \textbf{stable}.(r \ge k) \ \ \} \\
 & r = k \ \wedge \ r < M \ \rightsquigarrow \ r > k \ \vee \ r \ge M \\
\equiv & \qquad \{ \ \ [X \ \vee \ Y \ \equiv \ (\neg Y \ \wedge \ X) \ \vee \ Y] \ \ \} \\
 & r < M \ \wedge \ r = k \ \rightsquigarrow \ (r < M \ \wedge \ r > k) \ \vee \ r \ge M \\
\Rightarrow & \qquad \{ \ \ \text{induction} \ \ \} \\
 & r < M \ \rightsquigarrow \ r \ge M \\
\equiv & \qquad \{ \ \ \text{definition of } FP \ \ \} \\
 & r < M \ \rightsquigarrow \ FP \\
\equiv & \qquad \{ \ \ \textbf{initially}.(r < M) \ \ \} \\
 & \textbf{true} \ \rightsquigarrow \ FP
\end{array}$$

Will show on Wed

Specification

- Safety: **stable**(r = M) [Lecture 2.2]
- Progress: **true** $\rightsquigarrow$ (r = M)

Structure of the proof

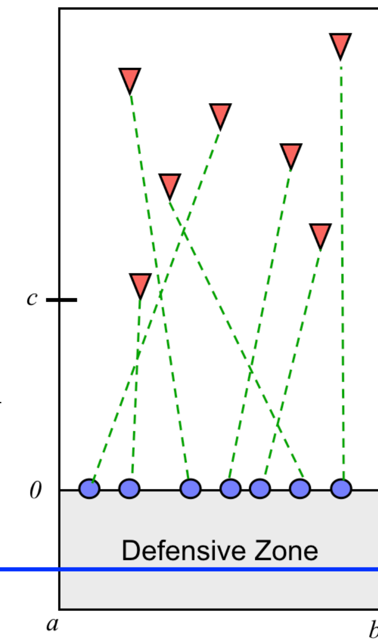
- Fixed point: $FP \ \equiv \ (\forall x \ : \ 0 \le x \le N-1 \ : \ r = \mathbf{max}(r, A[x]) \ )$

$$\equiv \ r \ge \ (\mathbf{Max}\, x \ : \ 0 \le x \le N-1 \ : \ A[x] \ )$$

$$\equiv \ r \ge M$$

- Invariant: $\textbf{invariant}.(r \le M)$
  - Combined with FP, this means that if we terminate at FP then r = M
- Metric: r
  - Never decreases and must increase at some point if r < M

Properties for RoboFlag program



Safety (Defenders do not collide)

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

Stability (switch predicate stays false)

$$\forall i \;.\; \underbrace{y_i > 2\delta \wedge z_i + 2\delta < z_{i+1}}_{\text{Robots are "far enough" apart.}} \wedge \neg switch_{i,i+1} \textbf{ next } \neg switch_{i,i+1}$$

Progress (we eventually reach a fixed point)

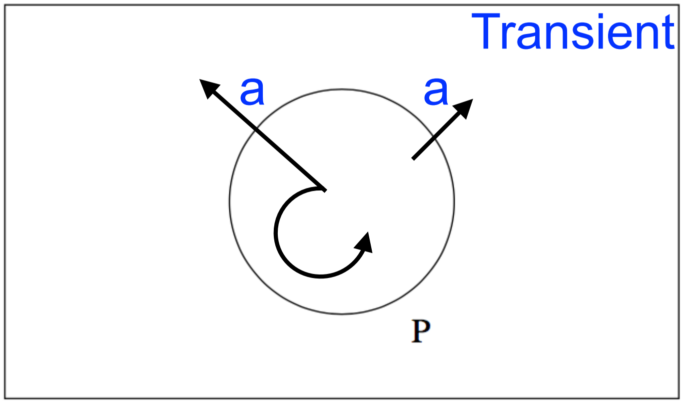
- Let ρ be the number of blue robots that are too far away to reach their red robots
- Let β be the total number of conflicts in the current assignment
- Define the *metric* that captures "energy" of current state (V = 0 is desired)

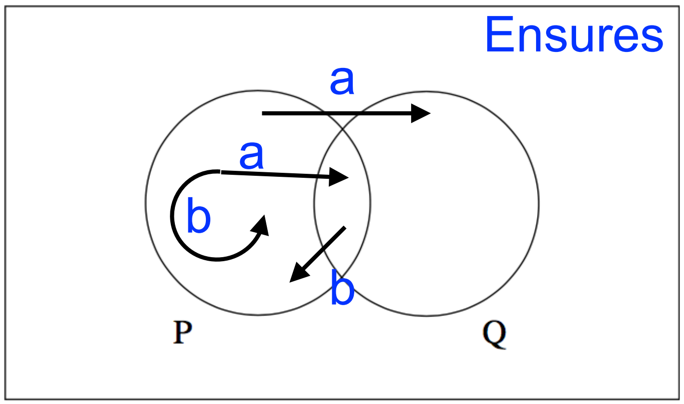
$$V = \left[\binom{n}{2} + 1\right]\rho + \beta \qquad \rho = \sum_{i=1}^{n} r(i,i) \qquad \beta = \sum_{i=1}^{n}\sum_{j=i+1}^{n} \gamma(i,j) \quad \text{where} \quad \gamma(i,j) = \begin{cases} 1 & \text{if } x_{\alpha(i)} > x_{\alpha(j)} \\ 0 & \text{otherwise} \end{cases}$$

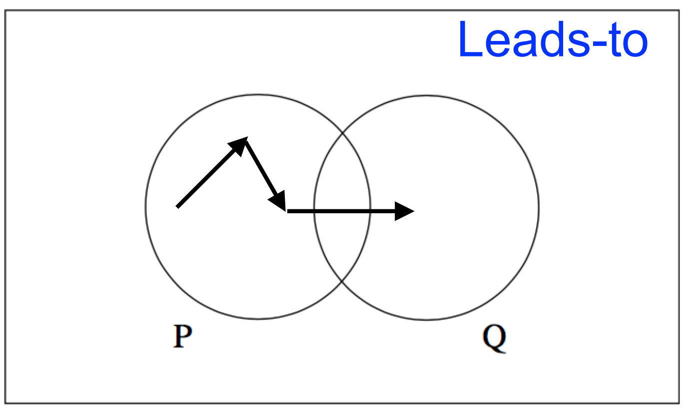
- Can show that V always decreases whenever a switch occurs

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

Summary: Progress Properties and Metrics







Establish *progress* properties

- Transient: $(\exists a : a \in G : \{P\}\ a\ \{\neg P\})$
- Ensures:

$$((P \wedge \neg Q)\ \textbf{next}\ (P \vee Q)) \wedge \textbf{transient}.(P \wedge \neg Q)$$

- Leads-to:

$$\begin{aligned} P\ \textbf{ensures}\ Q &\Rightarrow P \leadsto Q \\ (P \leadsto Q) \wedge (Q \leadsto R) &\Rightarrow P \leadsto R \\ (\forall i :: P_i \leadsto Q\ ) &\Rightarrow (\exists i :: P_i\ ) \leadsto Q \end{aligned}$$

  - This is the main property that we care about for proving that computations terminate correctly

- Metrics:

$$\begin{aligned} &(\forall m :: P \wedge M = m\ \textbf{next}\ (P \wedge M \leq m) \vee Q\ ) \\ &\wedge\ (\forall m :: \textbf{transient}.(P \wedge M = m)\ ) \\ &P \leadsto Q \end{aligned}$$

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