Caltech

CS/IDS 142: Lecture 2.2
Safety Properties

Richard M. Murray

9 October 2019

Goals:

- Define safety properties, program invariants
- New properties: next, stable, invariant

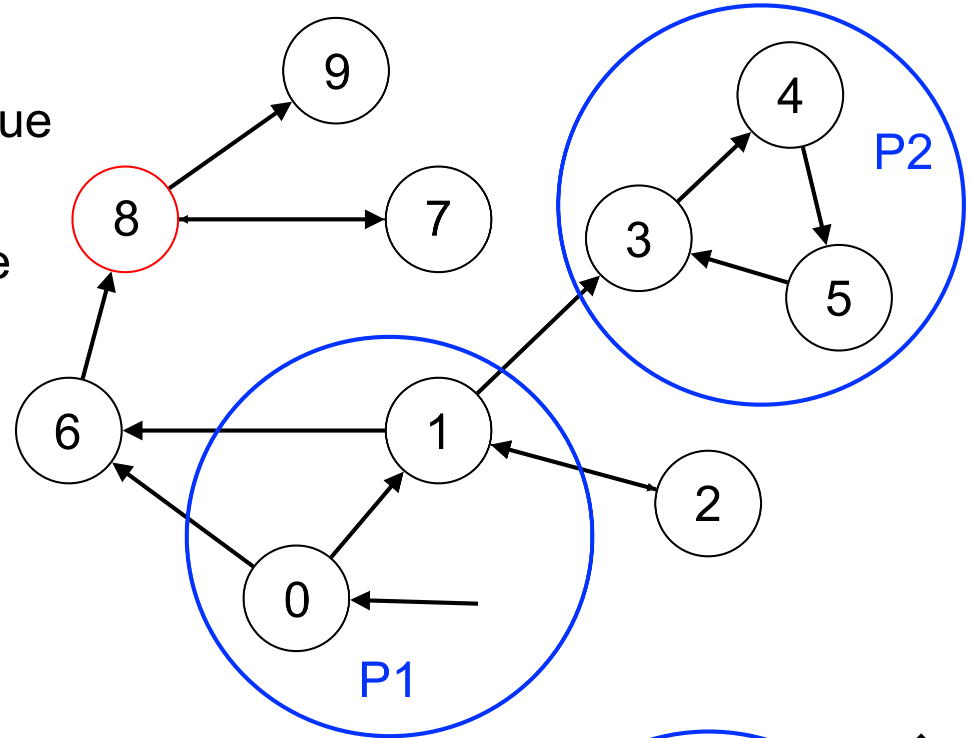
Reading:

- P. Sivilotti, *Introduction to Distributed Algorithms*, Section 3.3

The ‘Stable’ Property

Definition: **stable(P)**

- Informal: once P becomes true, it remains true
- Formally: **stable**(P) ≡ P **next** P
- Note: **stable**(P) does not mean that P is true for all (or even any) program executions



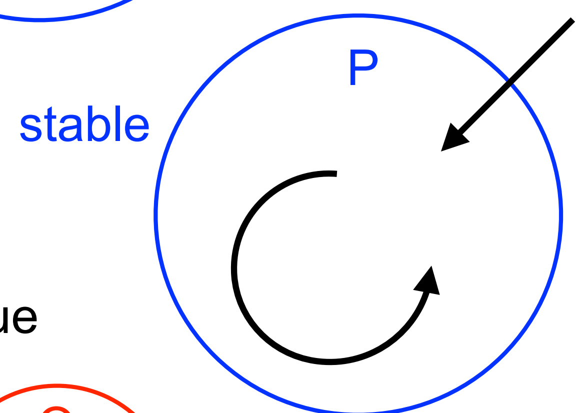
When do we use stable in a proof?

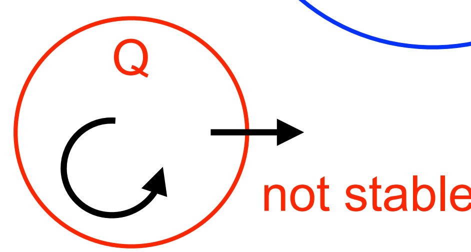
- Termination: **stable**({p})
- Often combined with progress (Wed + W3)
  - Show that if we satisfy some conditions then we eventually get to a good set of states (and stay there)

Some useful results (will prove on the homework)

- $\textbf{stable}(P) \wedge \textbf{stable}(Q) \implies \textbf{stable}(P \wedge Q)$
  - Interpretation: if P is stable and Q is stable, then at the point that both of them are true, they will both remain true
- $\textbf{stable}(P) \wedge \textbf{stable}(Q) \implies \textbf{stable}(P \vee Q)$
  - Note: *not* true that

$$\textbf{stable}(P) \vee \textbf{stable}(Q) \implies \textbf{stable}(P \vee Q)$$





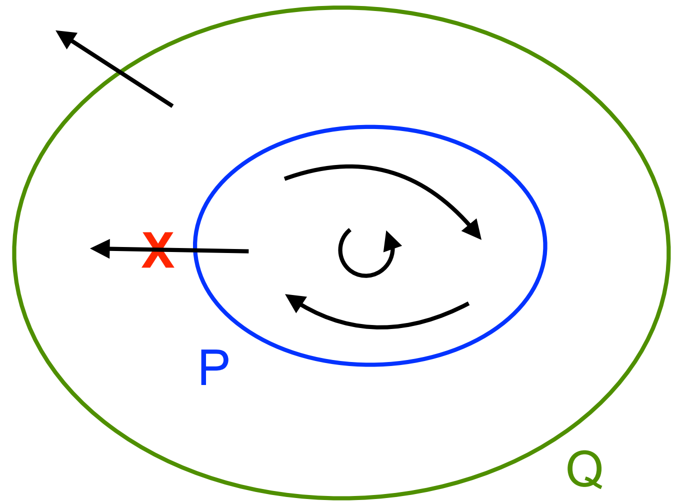
Which of the following formulas are true?

$\textbf{stable}(P) \wedge (P \subseteq Q) \implies \textbf{stable}(Q)$ _____ ← There can be an edge from a vertex which is in Q and not in P to a vertex outside Q

$\textbf{stable}(P) \wedge (Q \subseteq P) \implies \textbf{stable}(Q)$ _____

$\forall P : \textbf{stable}(\text{reachable}(P))$ _____

$(P \subseteq Q) \wedge \textbf{stable}(Q) \implies \text{reachable}(P) \subset Q$ _____



Reachable(P) is the smallest stable set that includes P

- Reachable(P) = set of points that we can reach from states that satisfy predicate P
- Proof sketch (exercise: turn into a formal proof = sequence of implications/equivals)
  - Let Q = reachable(P). Clear that $P \subseteq Q$ and **stable**(Q)
  - Suppose Q' is a smaller set ($Q' \subset Q$) with $P \subseteq Q'$ and stable(Q')
  - $Q' \subset Q \wedge \textbf{stable}(Q) \implies Q = \text{reachable}(P) \subset Q' \qquad \therefore Q = Q'$
- Algorithm for finding reachable(P): start with P add neighbors until you stop growing

Examples: Properties for Average Consensus

Program $\quad AverageConsensus$

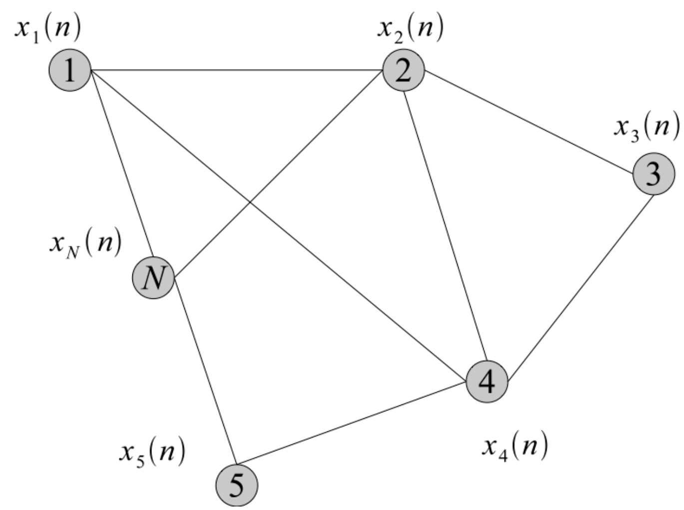
constant $\quad N \quad \{number\ of\ agents\}$

$\quad \mathcal{G} \quad \{interconnection\ graph\}$

var $\quad x$: array of N numbers

assign

$$\big(\llbracket i, j : j \in \mathcal{N}_i : x[i] := \alpha x[i] + (1-\alpha)x[j] \;||\; x[j] := \alpha x[j] + (1-\alpha)x[i]\rrbracket\big)$$



What are some stable properties for this program? [assume α = 1/2]

- $\textbf{stable}(x_i \leq x_i^0)$?
  - ___
- $\textbf{stable}(x_i + x_j \leq x_i^0 + x_j^0)$?
  - ___
- $\textbf{stable}(x_i \leq \max_i x_i^0)$?
  - ___
- $\textbf{stable}\big((+i : 0 \leq i \leq N-1 : x_i) \leq (+i : 0 \leq i \leq N-1 : x_i^0)\big)$?
  - ___

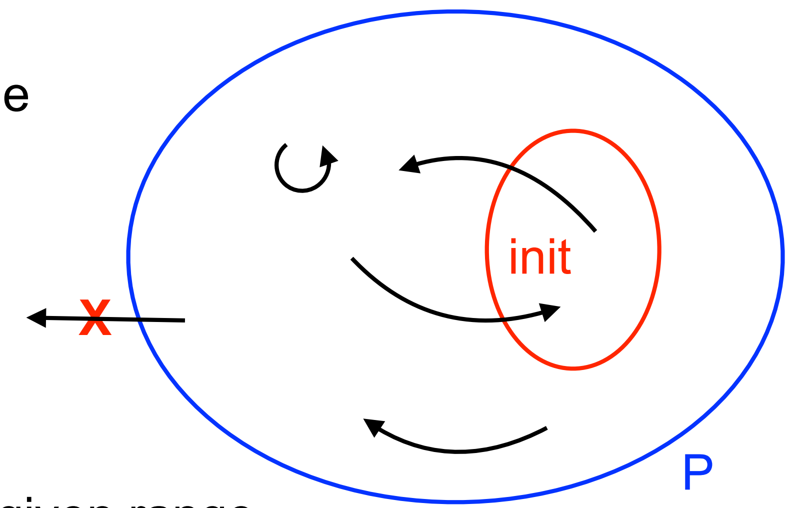
If time, add proof of the last property here?

The 'Invariant' Property

A predicate P is *invariant* if it is always true

$$\textbf{invariant}(P) \quad \equiv \quad \textbf{initially}(P) \land \textbf{stable}(P)$$

- Invariants are a critical part of proofs; establish the key properties that a problem *always* satisfies
- Invariants are not unique; a program can have many invariants



Some examples of useful invariants

- Amount of memory required is less than M
- Values of a variable (eg, address register) is in a given range

Proving properties about invariants comes down to evaluating Hoare triples

$$\textbf{initially}(P) \land (\forall a : a \in G : \{P\}\, a\, \{P\})$$

Example:

- For average consensus,

$$\textbf{invariant}\big((+i : 0 \leq i \leq N-1 : x_i) = (+i : 0 \leq i \leq N-1 : x_i^0)\big)$$

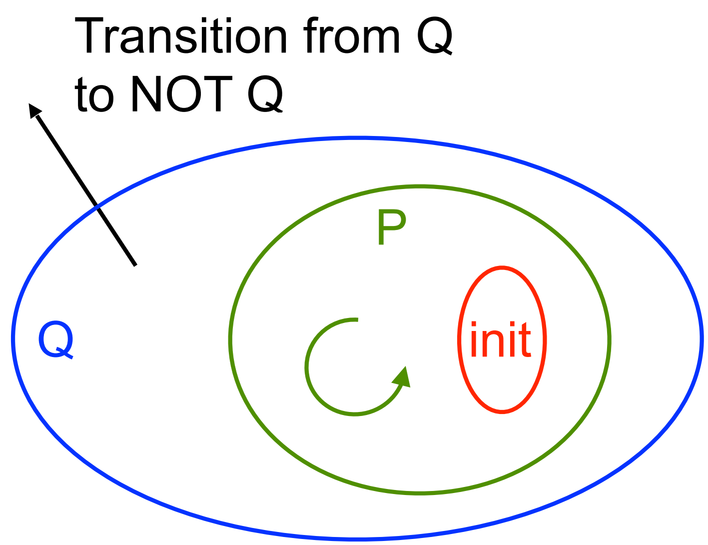
Reachability and invariants

- Recall that reachable(P) is the smallest stable set of vertices that includes P. Hence:

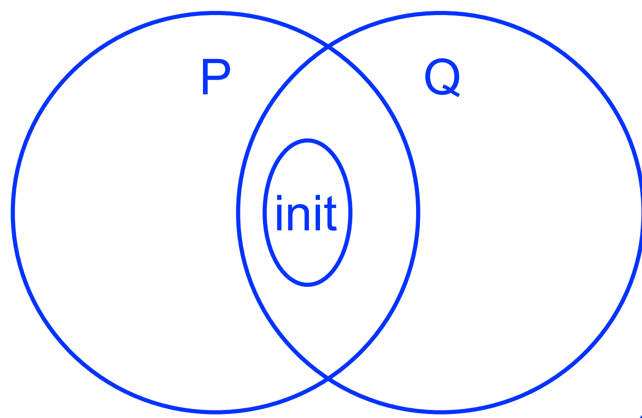
$$\textbf{invariant}\big(\text{reachable}(\text{init})\big) \qquad \textbf{invariant}(I) \implies \text{reachable}(\text{init}) \subseteq I$$

Which of the following formulas are true?

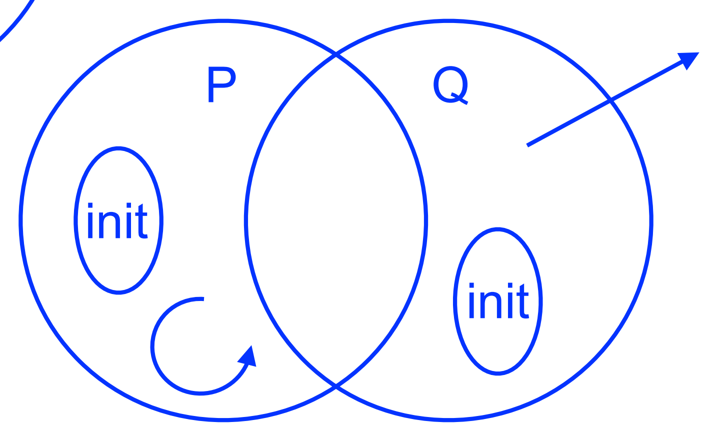
$$\text{invariant}(P) \wedge (P \subseteq Q) \;\Rightarrow\; \text{invariant}(Q)$$



$$\text{invariant}(P) \wedge \text{invariant}(Q) \Rightarrow \text{invariant}(P \bigcap Q)$$



$$\text{invariant}(P) \vee \text{invariant}(Q) \Rightarrow \text{invariant}(P \bigcup Q)$$



Example: FindMax

Let $M = (\mathbf{Max}\, x : 0 \leq x < N : A[x])$. **Prove that r ≤ M is an invariant**

1. $\mathbf{initially}.(r \leq M)$

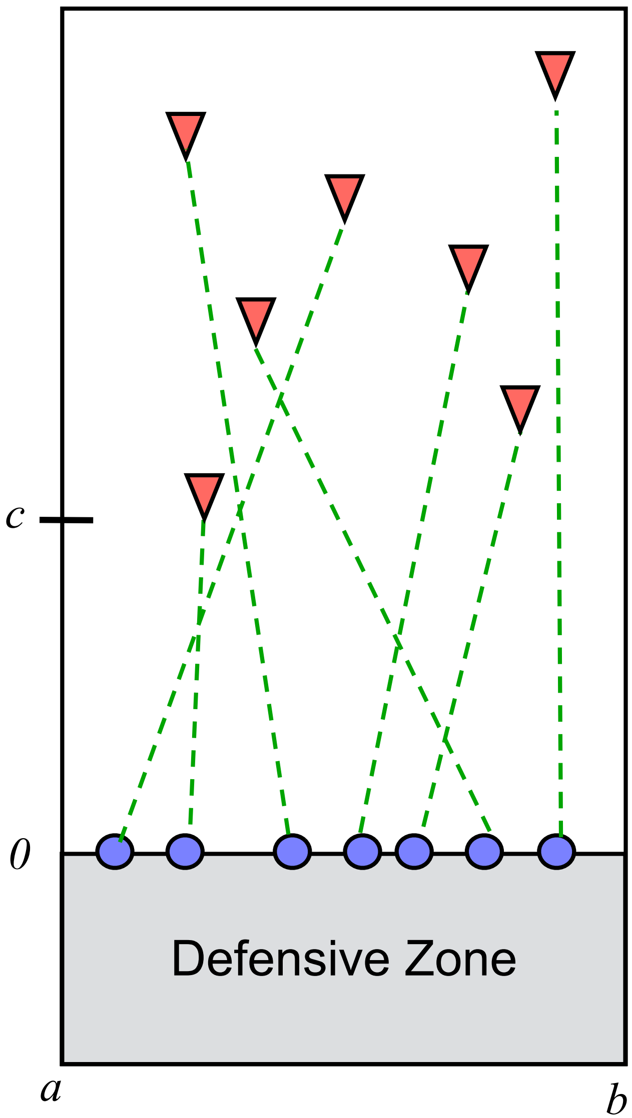
$$\begin{array}{ll} & r = A[0] \\ \Rightarrow & \{\ A[0] \leq M\ \} \\ & r \leq M \end{array}$$

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

2. $\mathbf{stable}.(r \leq M)$

$$\begin{array}{ll} & \mathbf{stable}.(r \leq M) \\ \equiv & \underline{\qquad\qquad\qquad} \\ & (r \leq M)\ \mathbf{next}\ (r \leq M) \\ \equiv & \underline{\qquad\qquad\qquad} \\ & (\forall a :: \{r \leq M\}\ a\ \{r \leq M\}) \\ \equiv & \{\ \text{definition of program}\ \} \\ & (\forall x : 0 \leq x < N : \{r \leq M\}\ r := max(r, A[x])\ \{r \leq M\}) \\ \equiv & \{\ \text{assignment axiom}\ \} \\ & (\forall x : 0 \leq x < N : \underline{r \leq M \Rightarrow \max(r, A[x]) \leq M)} \\ \equiv & \{\ x \leq \max(x, y)\ \} \\ & (\forall x : 0 \leq x < N : r \leq M \Rightarrow r \leq M\ ) \\ \equiv & \{\ \text{predicate calculus}\ \} \\ & \mathbf{true} \end{array}$$

Example: RoboFlag Drill



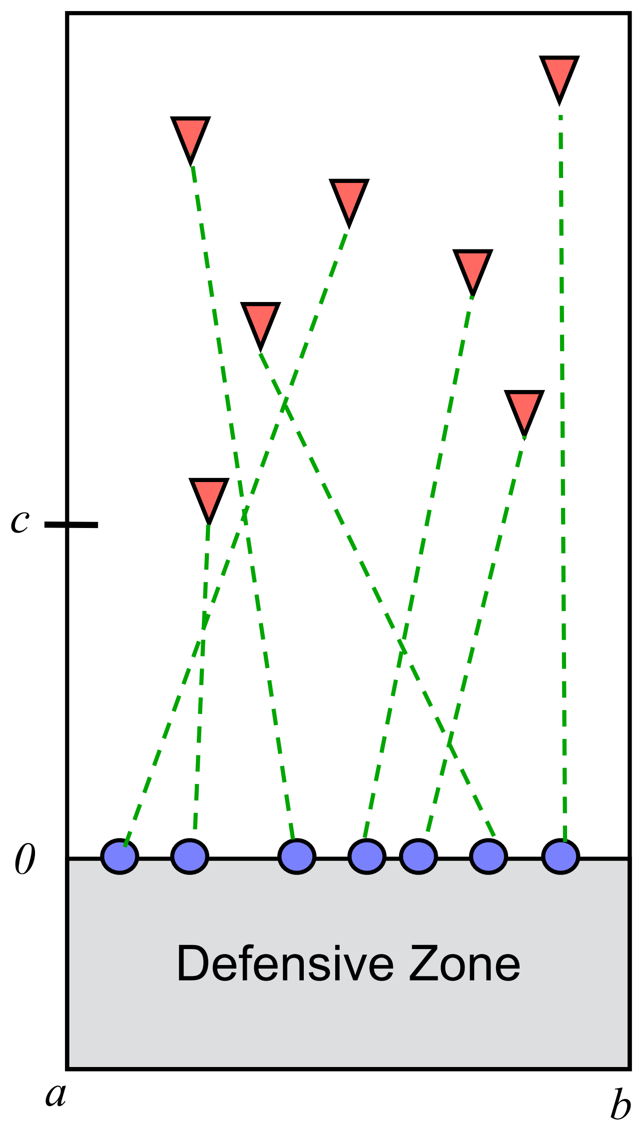
$Red(i)$	
Initial	$x_i \in [a, b] \wedge y_i > c$
Commands	$y_i > \delta \ : \ y_i' = y_i - \delta$
	$y_i \leq \delta \ : \ x_i' \in [a, b] \wedge y_i > c$

$$P_{Red}(n) = +_{i=1}^{n} Red(i)$$

$Blue(i)$	
Initial	$z_i \in [a, b] \wedge z_i < z_{i+1}$
Commands	$z_i < x_{\alpha(i)} \wedge z_i < z_{i+1} - \delta \ : \ z_i' = z_i + \delta$
	$z_i > x_{\alpha(i)} \wedge z_i > z_{i-1} + \delta \ : \ z_i' = z_i - \delta$

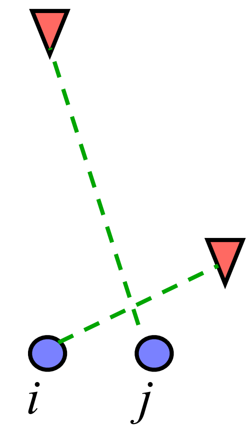
$$P_{Blue}(n) = +_{i=1}^{n} Blue(i)$$

RoboFlag Control Protocol



$\alpha(j)$ is too far down for i to get

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

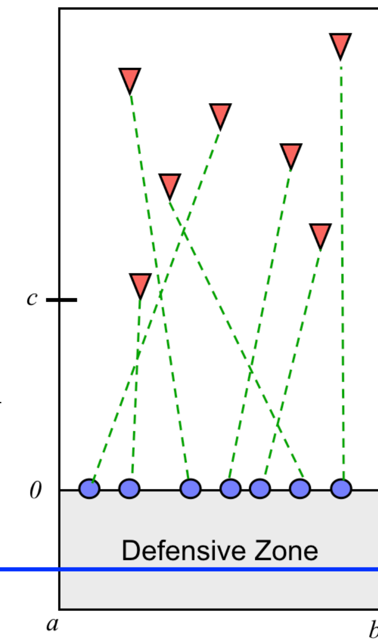


$$\begin{aligned} switch(i,j) \quad = \quad & r(i,j) + r(j,i) < r(i,i) + r(j,j) \\ \vee \quad & (r(i,j) + r(j,i) = r(i,i) + r(j,j) \\ & \wedge x_{\alpha(i)} > x_{\alpha(j)}) \end{aligned}$$

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

$$P_{Proto}(n) = +_{i=1}^{n-1} Proto(i)$$

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)

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

Robots are "far enough" apart.

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

What Goes Wrong: ZA002, Nov 2010

Official Word from Boeing: ZA002 787 Dreamliner fire and smoke details

By David Parker Brown, on November 10th, 2010 at 3:46 pm

Boeing 787 Dreamliner ZA002 at Paine Field on January 27, 2010 before its first flight.

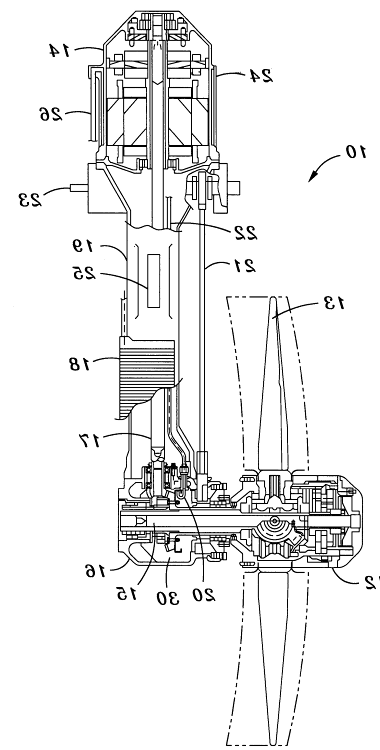
For the last day there are been bits and pieces of information coming from Boeing, inside sources and different media outlets on ZA002's sudden landing due to reported smoke in the cabin. Boeing has just released an official statement putting some of the rumors to rest and explaining what they know of ZA002's recent emergency landing in Laredo, TX.

Boeing confirms that ZA002 did lose primary electrical power that was related to an on board electrical fire. Due to the loss, the Ram Air Turbine (RAT), which provides back up power (photo of RAT from ZA003) was deployed and allowed the flight crew to land safely. The pilots had complete control of ZA002 during the entire incident.

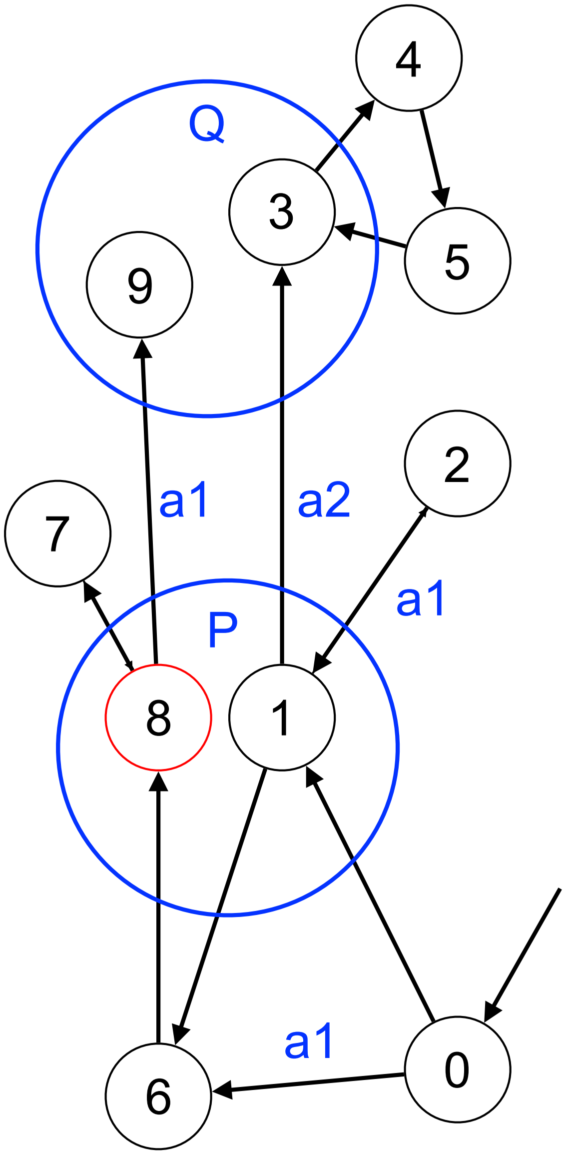
After their initial inspection, it appears that a power control panel in the rear of the electronics bay will need to be replaced. They are checking the surrounding areas for any additional damages. At this time, the cause of the fire is still being investigated and might take a few days until we have more answers.

Loss of primary electrical power => cockpit goes “dark”

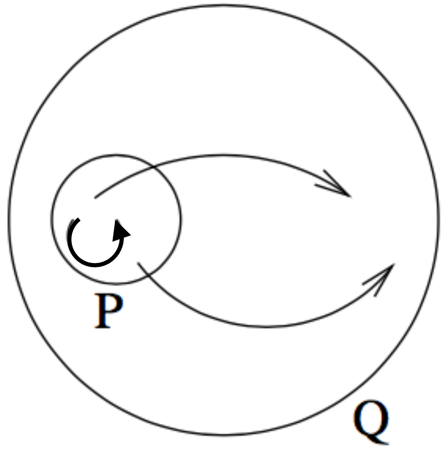
Ram Air Turbine (RAT) deployed and allows safe landing



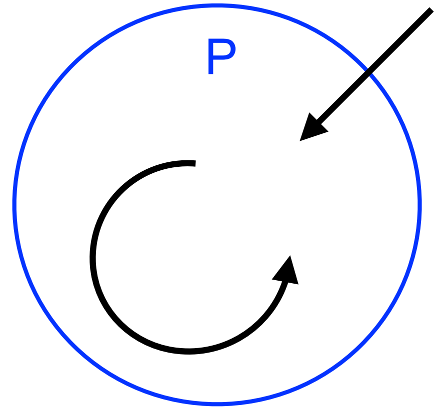
Summary: Reasoning About Programs



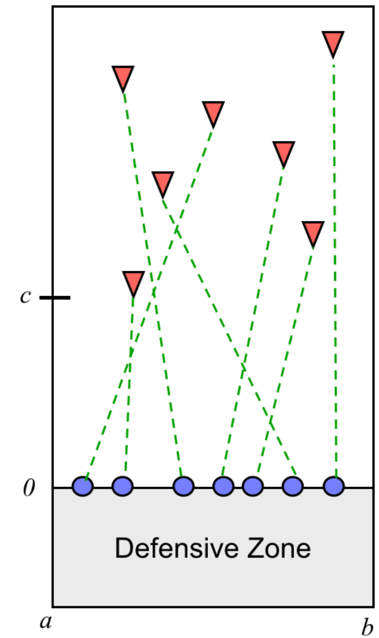
Hoare triple: {P} a {Q}



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: $\mathbf{invariant}(P) \equiv \mathbf{initially}(P) \wedge \mathbf{stable}(P)$