Infinite-State Verification: From Transition Systems to Markov Chains

Parosh Aziz Abdulla

Uppsala University

September 13, 2009

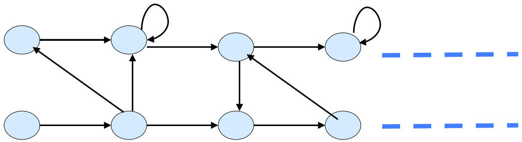
(Joint work with Noomene Ben Henda, Richard Mayr, and Sven Sandberg)

Infinite-State Transition Systems

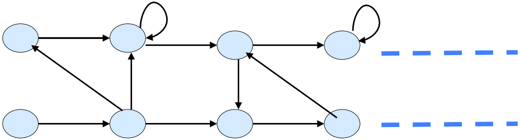
Infinite-State Transition Systems

- $\{C, \longrightarrow\}$
- C: (potentially infinite) set of configurations
- $\longrightarrow$: transition relation

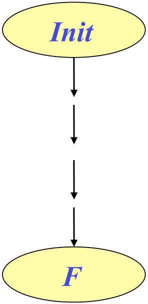
Infinite-State Transition Systems



Infinite-State Transition Systems

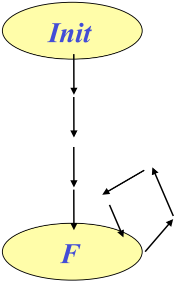


Reachability



$Init \models \Diamond F$?

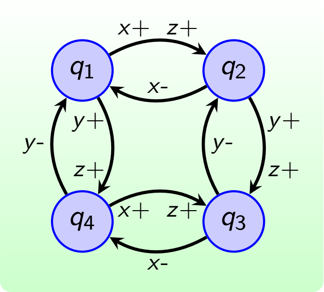
Repeated Reachability



$Init \models \Box\Diamond F$?

Vector Addition Systems with States (VASS)

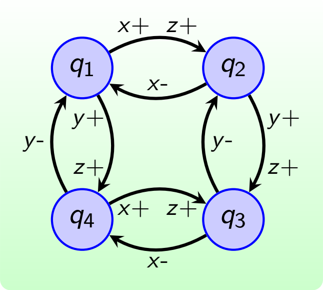
Model



- weak counters: can be incremented or decremented
- equivalent to Petri nets

Vector Addition Systems with States (VASS)

Model

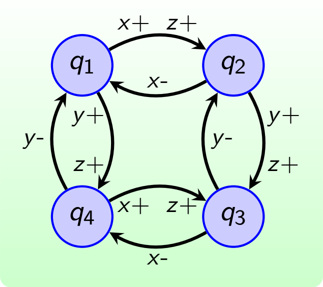


Configuration

$c = q_1(2, 0, 4)$

Vector Addition Systems with States (VASS)

Model



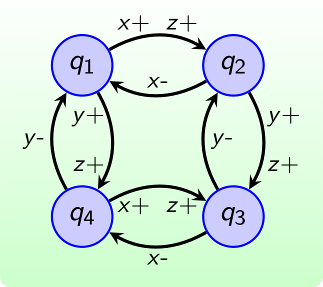
Configuration

$c = q_1(2, 0, 4)$

infinitely many configurations

Vector Addition Systems with States (VASS)

Model



Computation

$q_1(2,0,4) \longrightarrow q_2(3,0,5) \longrightarrow q_3(3,1,6) \longrightarrow q_2(3,0,6) \longrightarrow \cdots$

Vector Addition Systems with States (VASS)

Ordering

Ordering

- $q(x, y, z) \leq q'(x', y', z')$ iff
  - $q = q'$.
  - $x \leq x'$, $y \leq y'$, $z \leq z'$.

Vector Addition Systems with States (VASS)

Ordering

Ordering

- $q(x, y, z) \leq q'(x', y', z')$ iff
  - $q = q'$.
  - $x \leq x'$, $y \leq y'$, $z \leq z'$.

Examples

- $q_1(2, 0, 3) \leq q_1(4, 1, 3)$
- $q_1(2, 0, 3) \not\leq q_1(1, 6, 3)$
- $q_1(2, 0, 3) \not\leq q_2(5, 6, 3)$

Vector Addition Systems with States (VASS)

Ordering

Upward Closed Sets

$$(c \in U) \wedge (c \leq c') \implies (c' \in U)$$

Vector Addition Systems with States (VASS)

Ordering

Upward Closed Sets

$$(c \in U) \wedge (c \leq c') \implies (c' \in U)$$

Upward Closure

- $c\!\uparrow := \{c' | c \leq c'\}$
- $q_1(2,0,3)\!\uparrow = \{q_1(2,0,3), q_1(3,0,3), q_1(2,0,4), q_1(3,2,6), \ldots\}$
- $q_1(0,0,0)\!\uparrow = \{q_1(0,0,0), q_1(1,0,0), q_1(0,1,0), q_1(3,2,6), \ldots\}$

Vector Addition Systems with States (VASS)

Ordering

Upward Closed Sets

$$(c \in U) \wedge (c \leq c') \implies (c' \in U)$$

Upward Closure

- $c\uparrow := \{c' | c \leq c'\}$
- $q_1(2,0,3)\uparrow = \{q_1(2,0,3), q_1(3,0,3), q_1(2,0,4), q_1(3,2,6), \ldots\}$
- $q_1(0,0,0)\uparrow = \{q_1(0,0,0), q_1(1,0,0), q_1(0,1,0), q_1(3,2,6), \ldots\}$

Minimal Elements

- $\min(U)$:= minimal elements of U wrt. $\leq$.

Vector Addition Systems with States (VASS)

Ordering

Upward Closed Sets

$(c \in U) \wedge (c \le c') \implies (c' \in U)$

Upward Closure

- $c\uparrow := \{c' | c \le c'\}$
- $q_1(2,0,3)\uparrow = \{q_1(2,0,3), q_1(3,0,3), q_1(2,0,4), q_1(3,2,6), \ldots\}$
- $q_1(0,0,0)\uparrow = \{q_1(0,0,0), q_1(1,0,0), q_1(0,1,0), q_1(3,2,6), \ldots\}$

Minimal Elements

- $\min(U)$:= minimal elements of U wrt. $\le$.
- Properties:
  - $\min(U)$ is finite
  - $\min(U)\uparrow = U$.

Vector Addition Systems with States (VASS)

Coverability

Computation

$q_1(2,0,4) \longrightarrow q_2(3,0,5) \longrightarrow q_3(3,1,6) \longrightarrow q_2(3,0,6) \longrightarrow \cdots$

Vector Addition Systems with States (VASS)

Coverability

Computation

$q_1(2,0,4) \longrightarrow q_2(3,0,5) \longrightarrow q_3(3,1,6) \longrightarrow q_2(3,0,6) \longrightarrow \cdots$

K-Reachability

- $c_1 \xrightarrow{K} c_2$: c_1 can reach c_2 within K steps
- $q_1(2,0,4) \xrightarrow{5} q_2(3,0,6)$

Vector Addition Systems with States (VASS)

Coverability

Computation

$q_1(2,0,4) \longrightarrow q_2(3,0,5) \longrightarrow q_3(3,1,6) \longrightarrow q_2(3,0,6) \longrightarrow \cdots$

K-Reachability

- $c_1 \xrightarrow{K} c_2$: c_1 can reach c_2 within K steps
- $q_1(2,0,4) \xrightarrow{5} q_2(3,0,6)$

Reachability

- $c_1 \xrightarrow{*} c_2$: c_1 can reach c_2
- $q_1(2,0,4) \xrightarrow{*} q_2(3,0,6)$

Vector Addition Systems with States (VASS)

Coverability

Control State Reachability

- Instance:
  - c: configuration
  - q: control state
- Question: $c \xrightarrow{*} q(*,*,*)$?

Vector Addition Systems with States (VASS)

Coverability

Control State Reachability

- Instance:
  - c: configuration
  - q: control state
- Question: $c \xrightarrow{*} q(*,*,*)$?

Coverability

- Instance: c_1, c_2: configurations
- Question: $c_1 \xrightarrow{*} c_2\uparrow$?

Vector Addition Systems with States (VASS)

Coverability

From Control State Reachability to Coverability

- $c \xrightarrow{*} q(*,*,*)$?

Vector Addition Systems with States (VASS)

Coverability

From Control State Reachability to Coverability

- $c \xrightarrow{*} q(*,*,*)$?
- $c \xrightarrow{*} q(0,0,0)\uparrow$?

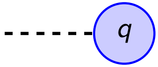
Vector Addition Systems with States (VASS)

Coverability

From Control State Reachability to Coverability

- $c \xrightarrow{*} q(*, *, *)$?
- $c \xrightarrow{*} q(0, 0, 0)\uparrow$?

From Coverability to Control State Reachability



Vector Addition Systems with States (VASS)

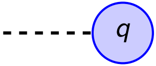
Coverability

From Control State Reachability to Coverability

- $c \xrightarrow{*} q(*, *, *)$?
- $c \xrightarrow{*} q(0, 0, 0)\uparrow$?

From Coverability to Control State Reachability

- $c \xrightarrow{*} q(2, 0, 1)\uparrow$?



Vector Addition Systems with States (VASS)

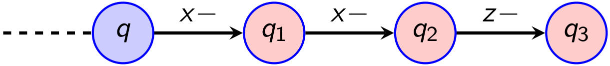
Coverability

From Control State Reachability to Coverability

- $c \xrightarrow{*} q(*,*,*)$?
- $c \xrightarrow{*} q(0,0,0)\uparrow$?

From Coverability to Control State Reachability

- $c \xrightarrow{*} q(2,0,1)\uparrow$?
- $c \xrightarrow{*} q_3(*,*,*)$?



Vector Addition Systems with States (VASS)

Backward Reachability Analysis

Monotonicity

$$\begin{array}{ccc} c_1 & \longrightarrow & c_2 \\ \text{\rotatebox{90}{$\leq$}} & & \\ c_3 & & \end{array}$$

Vector Addition Systems with States (VASS)

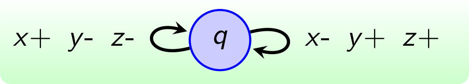
Backward Reachability Analysis

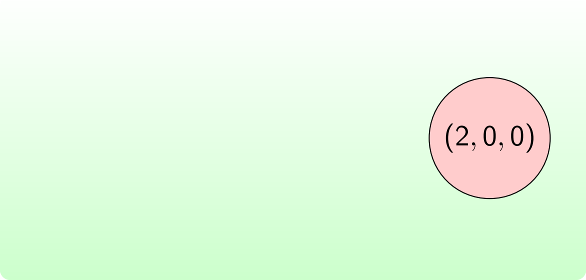
Monotonicity

$$
\begin{tikzcd}
c_1 \arrow[r] & c_2 \\
c_3 \arrow[u, phantom, "\leq" description] \arrow[r] & c_4 \arrow[u, phantom, "\leq" description]
\end{tikzcd}
$$

Vector Addition Systems with States (VASS)

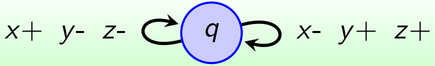
Backward Reachability Analysis

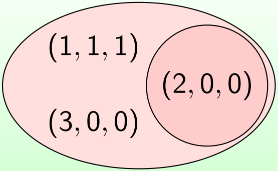




Vector Addition Systems with States (VASS)

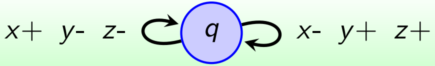
Backward Reachability Analysis

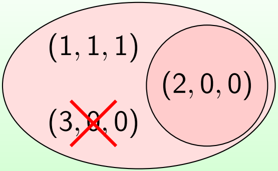




Vector Addition Systems with States (VASS)

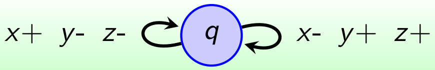
Backward Reachability Analysis

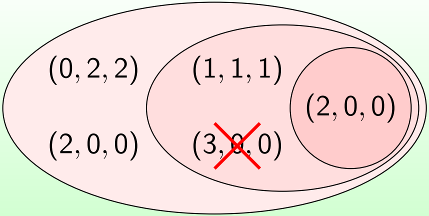




Vector Addition Systems with States (VASS)

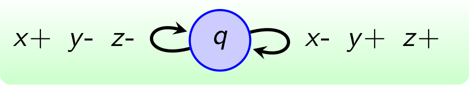
Backward Reachability Analysis

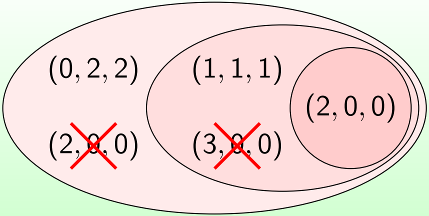




Vector Addition Systems with States (VASS)

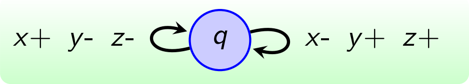
Backward Reachability Analysis

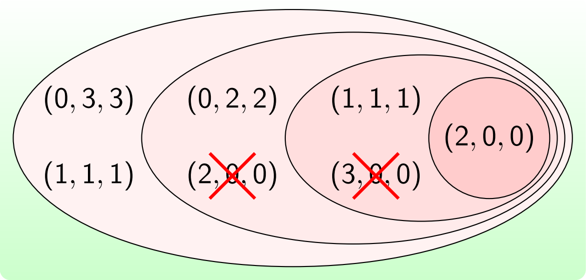




Vector Addition Systems with States (VASS)

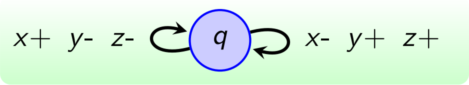
Backward Reachability Analysis

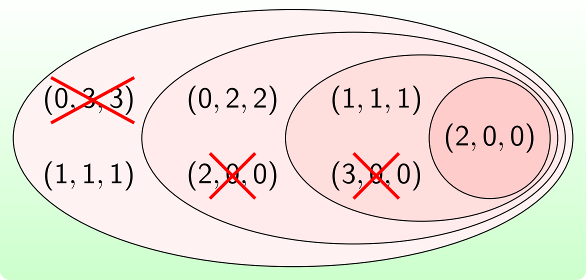




Vector Addition Systems with States (VASS)

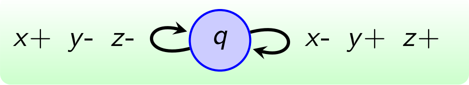
Backward Reachability Analysis

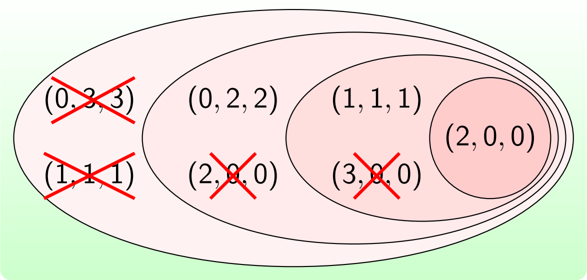




Vector Addition Systems with States (VASS)

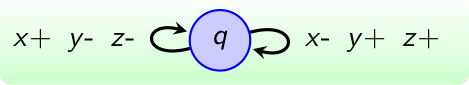
Backward Reachability Analysis

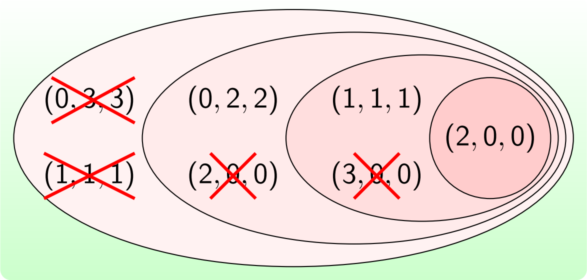




Vector Addition Systems with States (VASS)

Backward Reachability Analysis





By monotonicity

if $c \xrightarrow{*} (2,0,0)\uparrow$ then $c \xrightarrow{2} (2,0,0)\uparrow$

Vector Addition Systems with States (VASS)

Finite Spanning

Finitely Spanning

- F: set of target states
- $\exists K \; \forall c$:
  - if $c \xrightarrow{*} F$ then $c \xrightarrow{K} F$

Vector Addition Systems with States (VASS)

Finite Spanning

Finitely Spanning

- F: set of target states
- $\exists K\ \forall c$:
    - if $c \xrightarrow{*} F$ then $c \xrightarrow{K} F$

VASS

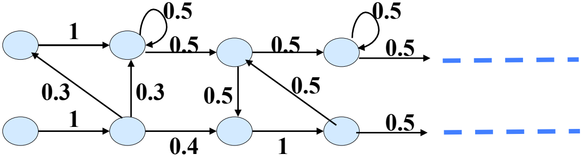
finitely spanning for upward closed F

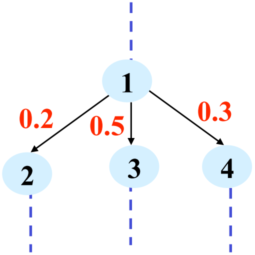
Infinite-State Markov Chains

Infinite-State Markov Chains

- infinite state space
- qualitative and quantitative properties

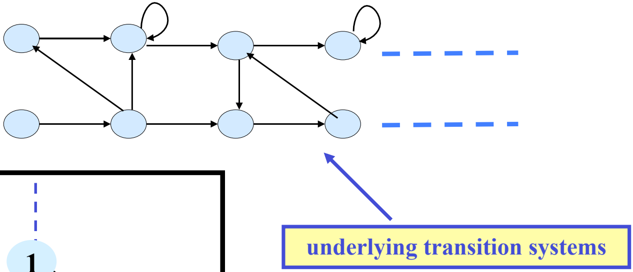
Infinite-State Markov Chains

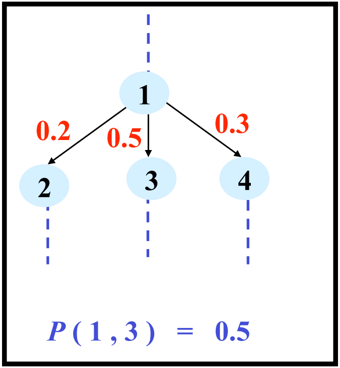




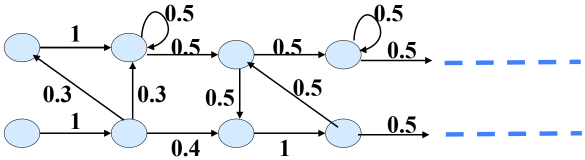
$P(1,3) = 0.5$

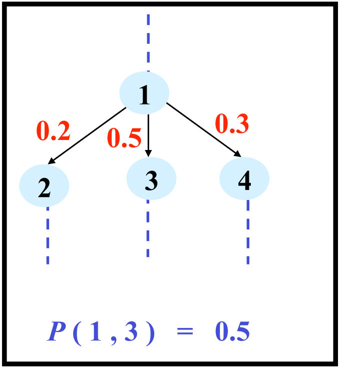
Infinite-State Markov Chains





Infinite-State Markov Chains



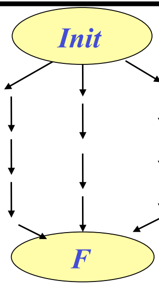


$Prob_s(\phi)$:

Probability that a computation from s satisfies ϕ

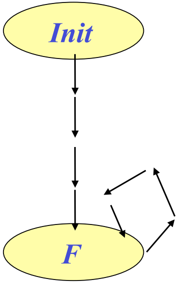
Qualitative Reachability Analysis

$$\boxed{Prob_{Init}\,(\Diamond F) = 1 \ ?}$$



Qualitative Repeated Reachability Analysis

$$\boxed{Prob_{Init}\,(\Box\Diamond F) = 1 \ ?}$$



Decisive Markov Chains

Definition

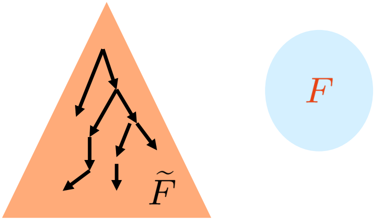
Decisive Markov Chains

- characterized by a simple property
- cover a wide class of systems
  - Probabilistic VASS (Probabilistic Petri Nets)
  - Probabilistic Lossy Channels Systems
  - Probabilistic Turing Machines
  - Probabilistic Pushdown Systems
- allow qualitative and quantitative properties

Decisive Markov Chains

Decisive Markov Chains

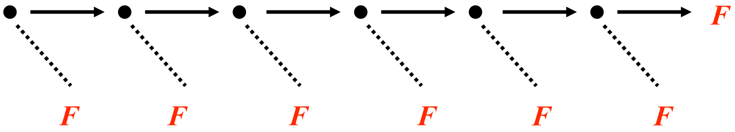
$\widetilde{F}$: states from which F unreachable



$$\widetilde{F} := \neg \exists \Diamond F$$

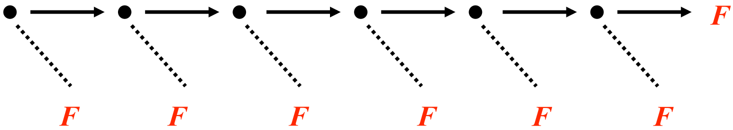
Decisive Markov Chains

F always reachable **implies** F almost certainly reached



Decisive Markov Chains

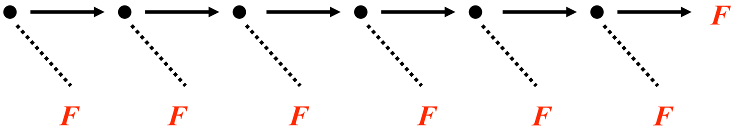
F always reachable implies **F almost certainly reached**



$$\boxed{Prob_s(\Diamond F \mid \Box \exists \Diamond F) = 1}$$

Decisive Markov Chains

F always reachable implies **F almost certainly reached**



$$Prob_s(\Diamond F \mid \Box \exists \Diamond F) = 1$$

$$Prob_s(\Diamond F \vee \Diamond \widetilde{F}) = 1$$

All finite-state Markov chains decisive

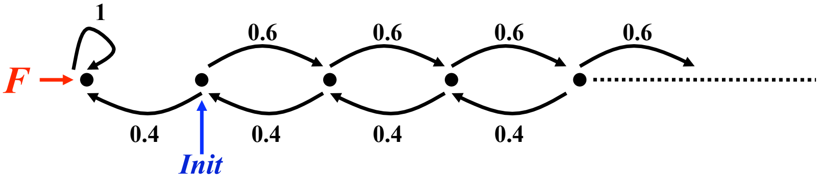
All finite-state Markov chains decisive

Are infinite-state Markov chains decisive?

All finite-state Markov chains decisive

Are infinite-state Markov chains decisive?

Not in general

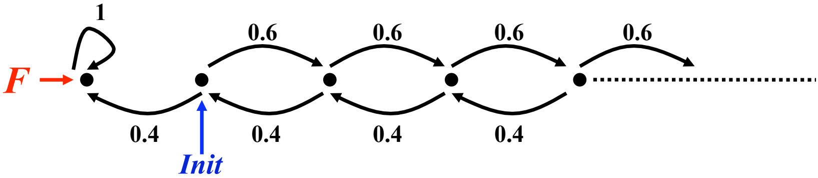


$$Prob_{Init}(\Diamond F) = \frac{2}{3} < 1$$

All finite-state Markov chains decisive

Are infinite-state Markov chains decisive?

Not in general



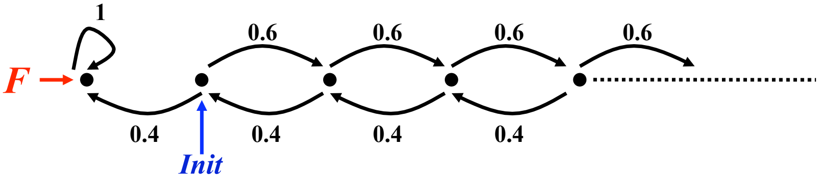
$$Prob_{Init}(\Diamond F) = \frac{2}{3} < 1$$

$$\widetilde{F} = \emptyset$$

All finite-state Markov chains decisive

Are infinite-state Markov chains decisive?

Not in general



$$Prob_{Init}(\Diamond F) = \frac{2}{3} < 1$$

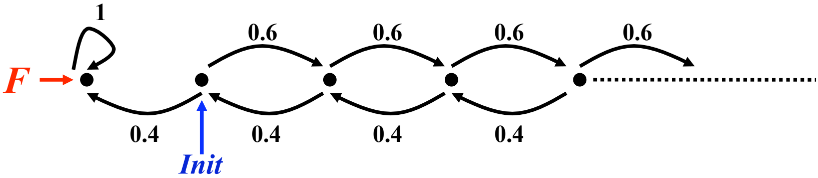
$$\widetilde{F} = \emptyset$$

$$Prob_{s}(\Diamond F \vee \Diamond \widetilde{F}) = \frac{2}{3} < 1$$

All finite-state Markov chains decisive

Are infinite-state Markov chains decisive?

Not in general



$$\boxed{Prob_{Init}(\Diamond F) = \frac{2}{3} < 1}$$

$$\boxed{\widetilde{F} = \emptyset}$$

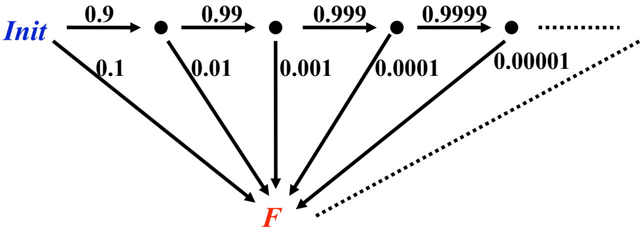
Not decisive

$$\boxed{Prob_s(\Diamond F \vee \Diamond \widetilde{F}) = \frac{2}{3} < 1}$$

All finite-state Markov chains decisive

Are infinite-state Markov chains decisive?

Not in general



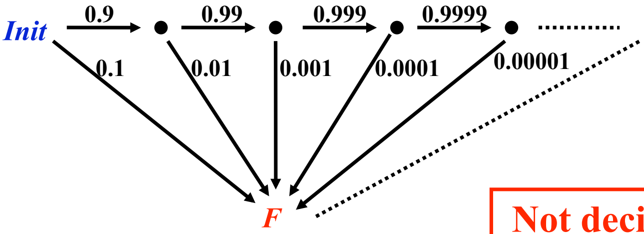
$$Prob_{Init}(\Diamond F) < 0.2 \cdots$$

$$Prob_{Init}(\Diamond F \vee \Diamond \widetilde{F}) < 0.2 \cdots$$

All finite-state Markov chains decisive

Are infinite-state Markov chains decisive?

Not in general



Not decisive

$$Prob_{Init}(\Diamond F) < 0.2 \cdots$$

$$Prob_{Init}(\Diamond F \vee \Diamond \widetilde{F}) < 0.2 \cdots$$

Decisive Markov Chains

Sufficient Conditions

Decisive Markov Chains – Sufficient Conditions

- coarseness and finite spanning
  - Probabilistic VASS (Probabilistic Petri Nets)
  - Probabilistic Turing Machines
- existence of finite attractors
  - Probabilistic Lossy Channels Systems
  - Probabilistic Pushdown Systems

Decisive Markov Chains

Sufficient Conditions

Decisive Markov Chains – Sufficient Condition 1

- coarseness and finite spanning
  - Probabilistic VASS (Probabilistic Petri Nets)
  - Probabilistic Turing Machines

α - coarseness

$P(s_1, s_2) > 0$ implies $P(s_1, s_2) > \alpha$

K-spanning

$s \xrightarrow{*} F$ implies $s \xrightarrow{K} F$

α - coarseness

$P(s_1, s_2) > 0$ implies $P(s_1, s_2) > \alpha$

K-spanning

$s \xrightarrow{*} F$ implies $s \xrightarrow{K} F$

coarseness
+
finite spanning

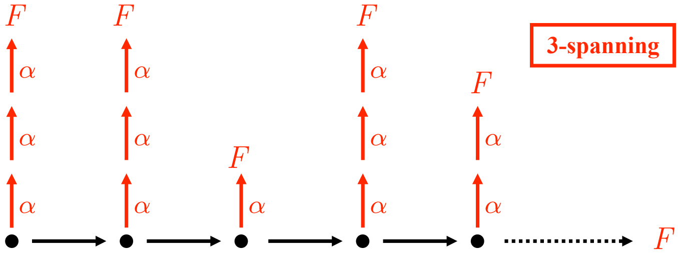
implies

decisiveness

coarseness
+
finite spanning

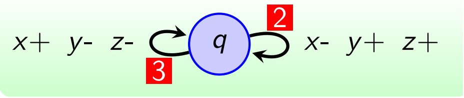
implies

decisiveness



Probabilistic Vector Addition Systems with States (PVASS)

PVASS

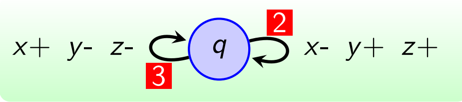


Weights

- Each transition has a weight
- $P(c_1, c_2)$ decided by:
  - relative weights of transitions

Probabilistic Vector Addition Systems with States (PVASS)

PVASS



Weights

- Each transition has a weight
- $P(c_1, c_2)$ decided by:
  - relative weights of transitions

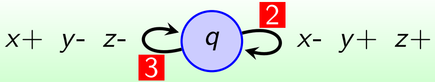
Example

$$P((1,0,2),(0,1,3)) = 1$$

$$P((1,1,2),(0,2,3)) = \frac{2}{5}$$

Probabilistic Vector Addition Systems with States (PVASS)

PVASS



Weights

- Each transition has a weight
- $P(c_1, c_2)$ decided by:
  - relative weights of transitions

Example

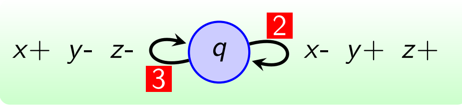
$P((1,0,2),(0,1,3)) = 1$

$P((1,1,2),(0,2,3)) = \frac{2}{5}$

finite set of weights $\Longrightarrow$ coarse

Probabilistic Vector Addition Systems with States (PVASS)

PVASS



Weights

- Each transition has a weight
- $P(c_1, c_2)$ decided by:
  - relative weights of transitions

Example

$$P((1,0,2),(0,1,3)) = 1$$

$$P((1,1,2),(0,2,3)) = \frac{2}{5}$$

finite set of weights $\Longrightarrow$ coarse

PVASS $\Longrightarrow$

- coarse
- finitely spanning

$\Longrightarrow$ Decisive

Decisive Markov Chains

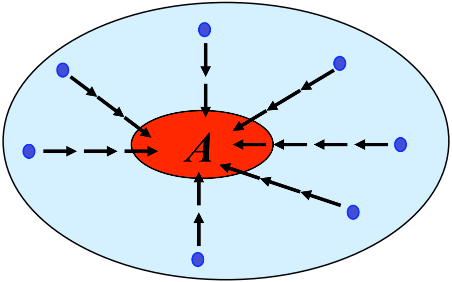
Sufficient Conditions

Decisive Markov Chains – Sufficient Condition 2

- existence of finite attractors
  - Probabilistic Lossy Channels Systems
  - Probabilistic Pushdown Systems

Attractors

for each s :
$Prob_s(\Diamond A) = 1$

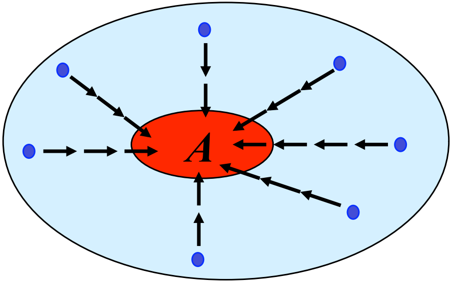


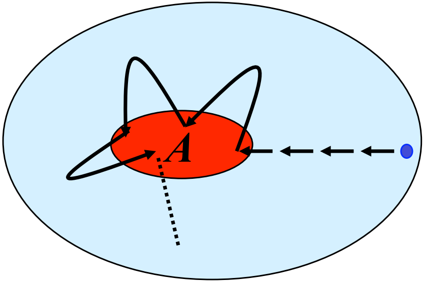
Attractors

for each s :
$Prob_s(\Diamond A) = 1$

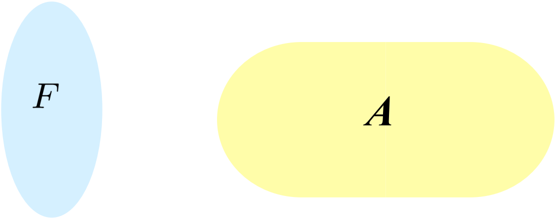
Implies

for each s :
$Prob_s(\Box\Diamond A) = 1$

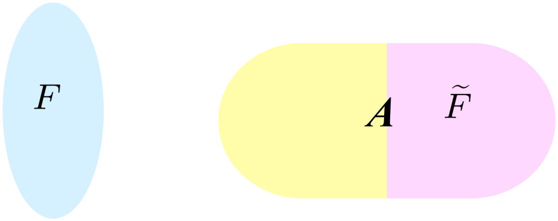




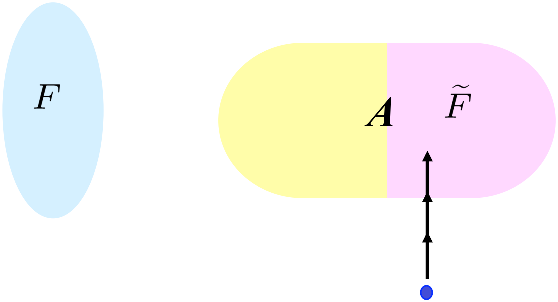
Finite attractor $\longrightarrow$ **Decisiveness**



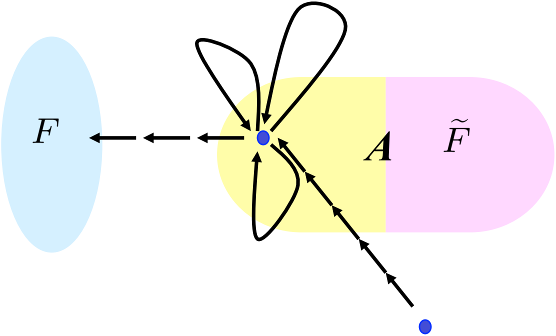
Finite attractor $\longrightarrow$ **Decisiveness**



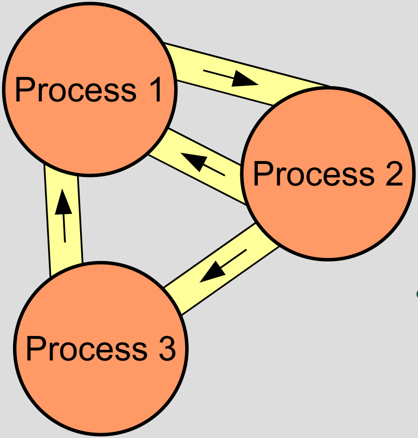
Finite attractor $\longrightarrow$ **Decisiveness**



Finite attractor $\longrightarrow$ **Decisiveness**

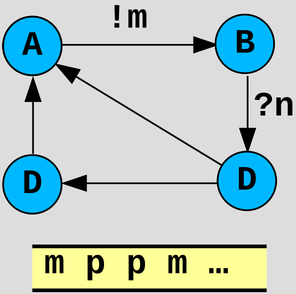


Probabilistic Lossy Channel Systems (PLCS)



- **Model:**
  - Finite state processes
  - Unbounded lossy channels
  - Send & receive operations
- **Motivation:**
  - Models of communication protocols

Probabilistic Lossy Channel Systems (PLCS)



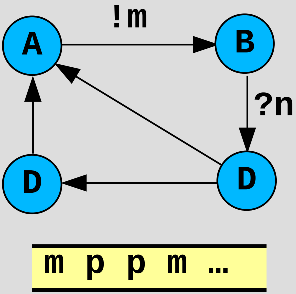
Sufficient:

- One channel
- One process

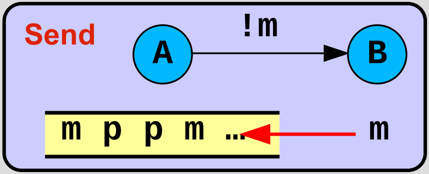
Properties:

- Infinite state space
- Perfect channel = Turing machine

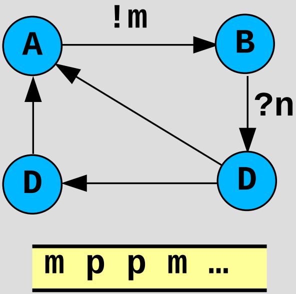
Probabilistic Lossy Channel Systems (PLCS)



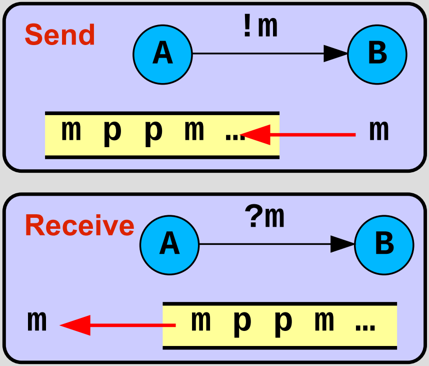
Transitions:



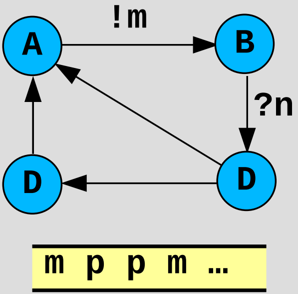
Probabilistic Lossy Channel Systems (PLCS)



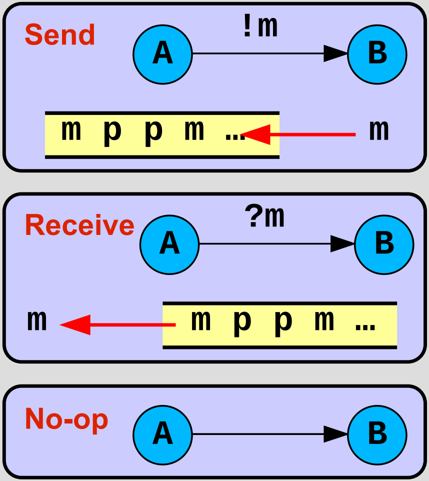
Transitions:



Probabilistic Lossy Channel Systems (PLCS)



Transitions:



Probabilistic Lossy Channel Systems (PLCS)

message loss:

p m n p n …

$\Downarrow$

p n n …

- **Each transition:**

  each message lost with prob $\lambda > 0$, independently

PLCS

Finite Attractor

set of configurations with empty channels

PLCS

Finite Attractor

set of configurations with empty channels

PLCS $\Longrightarrow$ Finite Attractor $\Longrightarrow$ Decisive

Qualitative Reachability Analysis

Qualitative Reachability Analysis

- analyze underlying transition system
- structural properties: reachability of F and $\tilde{F}$

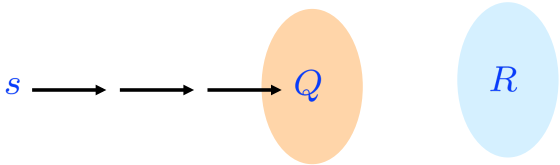
Qualitative Reachability Analysis

$$\boxed{Prob_{Init}(\Diamond F) = 1 \ ?}$$

Qualitative Reachability Analysis

$$\boxed{Prob_{Init}(\Diamond F) = 1 \ ?}$$

$s \models Q$ **Before** R



Qualitative Reachability Analysis

$$\boxed{Init \models \widetilde{F} \;\; \textbf{Before} \;\; F}$$

implies

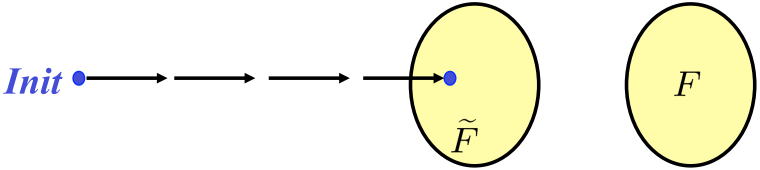
$$\boxed{Prob_{Init}(\Diamond F) < 1}$$

Qualitative Reachability Analysis

$$\boxed{Init \models \widetilde{F} \;\; \textbf{Before} \;\; F}$$

implies

$$\boxed{Prob_{Init}(\Diamond F) < 1}$$

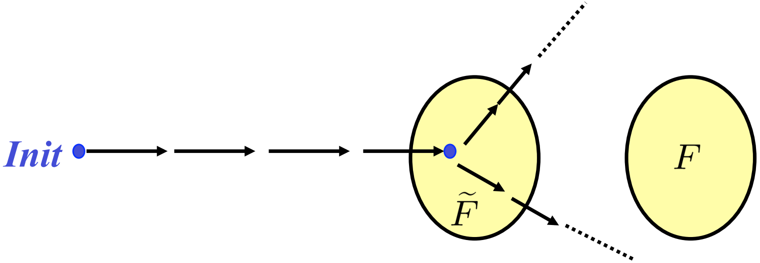


Qualitative Reachability Analysis

$$\boxed{Init \models \widetilde{F} \ \ \textbf{Before} \ \ F}$$

implies

$$\boxed{Prob_{Init}(\Diamond F) < 1}$$



Qualitative Reachability Analysis

$$\boxed{Init \not\models \widetilde{F} \quad \textbf{Before} \quad F}$$

Implies?

$$\boxed{Prob_{Init}(\Diamond F) = 1}$$

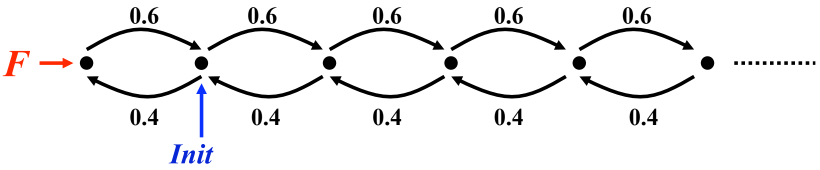
Qualitative Reachability Analysis

$$\boxed{Init \not\models \widetilde{F} \;\; \textbf{Before} \;\; F}$$

Implies?

$$\boxed{Prob_{Init}(\Diamond F) = 1}$$

Not in general !!



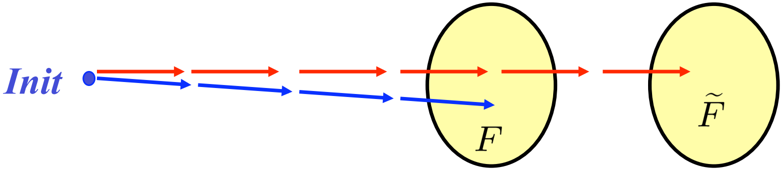
Qualitative Reachability Analysis

$$\boxed{Init \not\models \widetilde{F} \textbf{ Before } F}$$

Implies?

$$\boxed{Prob_{Init}(\Diamond F) = 1}$$

Yes if decisive !!



Qualitative Reachability Analysis

$$\boxed{Init \not\models \widetilde{F} \ \textbf{Before} \ F}$$

Implies?

$$\boxed{Prob_{Init}(\Diamond F) = 1}$$

Yes if decisive !!

Yes if coarse and finitely spanning !!

Yes if finite attractor exists !!

Qualitative Reachability Analysis

$$Init \not\models \widetilde{F} \ \textbf{Before} \ F$$

Implies?

$$Prob_{Init}(\Diamond F) = 1$$

Yes if decisive !!

Yes if coarse and finitely spanning !!

Yes if finite attractor exists !!

Yes: PVASS

Yes: NTM

Yes: PLCS

Qualitative Reachability Analysis

$$\boxed{Init \models \widetilde{F} \;\; \textbf{Before} \;\; F}$$

iff

$$\boxed{Prob_{Init}(\Diamond F) < 1}$$

Yes: PVASS

Yes: NTM

Yes: PLCS

Qualitative Reachability Analysis

Can we check

$$Init \models \widetilde{F} \ \textbf{Before} \ F \ ?$$

Qualitative Reachability Analysis

Can we check

$Init \models \widetilde{F}$ **Before** F ?

Yes:
PVASS -- ***F*** **set of control states**

Qualitative Reachability Analysis

Can we check

$Init \models \widetilde{F}$ **Before** F ?

Yes:
PVASS -- *F* set of control states

No:
PVASS -- *F* upward closed

undecidable

Qualitative Reachability Analysis

Can we check

$Init \models \widetilde{F}$ Before F ?

Yes:
PVASS -- *F* set of control states

No:
PVASS -- *F* upward closed

undecidable

Yes:
PLCS

Qualitative Repeated Reachability Analysis

Qualitative Repeated Reachability Analysis

- analyze underlying transition system
- structural properties: reachability of F

Qualitative Repeated Reachability Analysis

$$\boxed{Prob_{Init}(\Box\Diamond F) = 1 \ ?}$$

Qualitative Repeated Reachability Analysis

$$\boxed{Prob_{Init}(\Box\Diamond F) = 1 \; ?}$$

$$\boxed{Init \models \forall\Box\;\exists\Diamond\; F}$$

Qualitative Repeated Reachability Analysis

$$\boxed{Prob_{Init}(\Box\Diamond F) = 1 \ \ ?}$$

$$\boxed{Init \models \forall\Box\ \exists\Diamond\ F} \quad \equiv \quad \boxed{Init \not\models \exists\Diamond\ \widetilde{F}}$$

Qualitative Repeated Reachability Analysis

$$\boxed{Init \not\models \forall\Box\ \exists\Diamond\ F}$$

implies

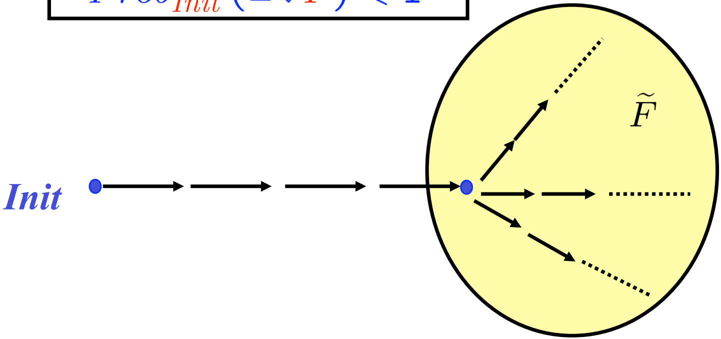
$$\boxed{Prob_{Init}(\Box\Diamond F) < 1}$$

Qualitative Repeated Reachability Analysis

$\boxed{Init \not\models \forall\Box\ \exists\Diamond\ F}$ $\equiv$ $\boxed{Init \models \exists\Diamond\ \widetilde{F}}$

implies

$\boxed{Prob_{Init}(\Box\Diamond F) < 1}$



Qualitative Repeated Reachability Analysis

$$\boxed{Init \models \forall\Box\ \exists\Diamond\ F}$$

Implies ?

$$\boxed{Prob_{Init}(\Box\Diamond F) = 1}$$

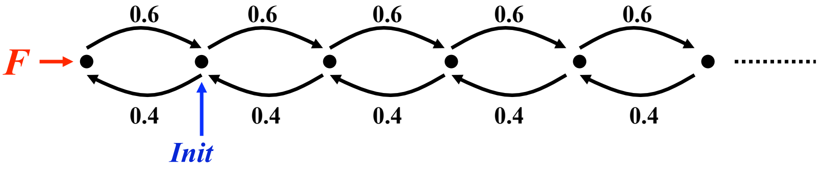
Qualitative Repeated Reachability Analysis

$$\boxed{Init \models \forall\Box\ \exists\Diamond\ F}$$

Implies ?

$$\boxed{Prob_{Init}(\Box\Diamond F) = 1}$$

Not in general !!



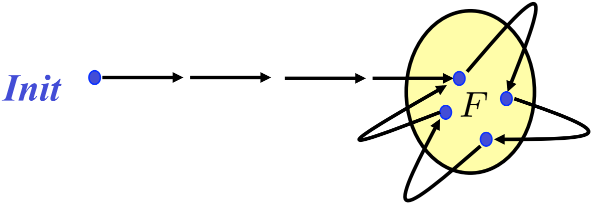
Qualitative Repeated Reachability Analysis

$$\boxed{Init \models \forall\Box\ \exists\Diamond\ F}$$

Implies ?

$$\boxed{Prob_{Init}(\Box\Diamond F) = 1}$$

Yes if decisive !!



Qualitative Repeated Reachability Analysis

$$\boxed{Init \models \forall\Box\ \exists\Diamond\ F}$$

Implies ?

$$\boxed{Prob_{Init}(\Box\Diamond F) = 1}$$

Yes if decisive !!

Yes if coarse and finitely spanning !!

Yes if finite attractor exists !!

Yes: PVASS

Yes: NTM

Yes: PLCS

Qualitative Repeated Reachability Analysis

$$\boxed{Init \models \forall\Box\ \exists\Diamond\ F}$$

iff

$$\boxed{Prob_{Init}(\Box\Diamond F) = 1}$$

Yes: PVASS

Yes: NTM

Yes: PLCS

Qualitative Repeated Reachability Analysis

Can we check

$Init \models \forall\Box\ \exists\Diamond\ F$?

Qualitative Repeated Reachability Analysis

Can we check

$Init \models \forall\Box\ \exists\Diamond\ F$?

$=$

$Init \in \widetilde{\left(\widetilde{F}\right)}$?

Qualitative Repeated Reachability Analysis

Can we check

$Init \models \forall\Box\ \exists\Diamond\ F$?

$=$

$Init \in \widetilde{\left(\widetilde{F}\right)}$?

Yes:
PVASS -- F set of local states

Yes:
PVASS -- F upward closed

decidable

Yes:
PLCS -- F set of local states

Qualitative Reachability Analysis

(Approximate) Quantitative Reachability Analysis

- expand the reachability tree

(Approximate) Quantitative Reachability Analysis

Given ϵ ; compute ρ s.t.

$$\rho \leq Prob_{Init}(\Diamond F) \leq \rho + \epsilon$$

Init

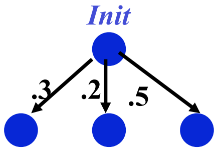
Yes: =

No: =

Quantitative Reachability Analysis

Given ϵ ; compute ρ s.t.

$$\rho \leq Prob_{Init}(\Diamond F) \leq \rho + \epsilon$$



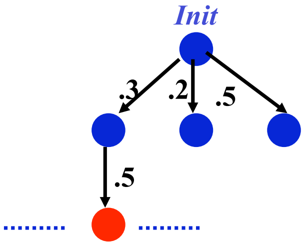
Yes: =

No: =

Quantitative Reachability Analysis

Given ϵ ; compute ρ s.t.

$$\rho \leq Prob_{Init}(\Diamond F) \leq \rho + \epsilon$$



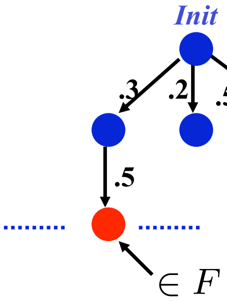
Yes: =

No: =

Quantitative Reachability Analysis

Given ϵ ; compute ρ s.t.

$$\rho \leq Prob_{Init}(\Diamond F) \leq \rho + \epsilon$$



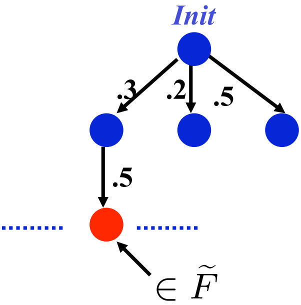
Yes: = Yes + .15

No: =

Quantitative Reachability Analysis

Given ϵ ; compute ρ s.t.

$$\rho \leq Prob_{Init}(\Diamond F) \leq \rho + \epsilon$$



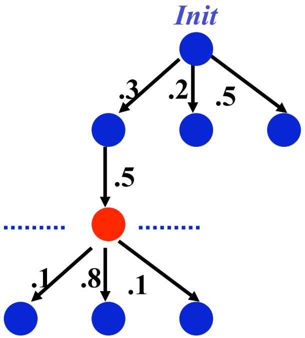
Yes: =

No: = No + .15

Quantitative Reachability Analysis

Given ϵ ; compute ρ s.t.

$$\rho \leq Prob_{Init}(\Diamond F) \leq \rho + \epsilon$$



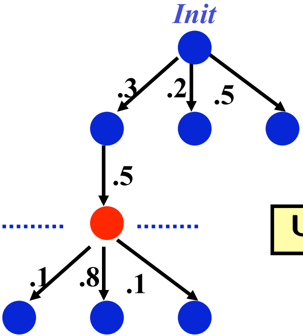
Yes: =

No: =

Quantitative Reachability Analysis

Given ϵ ; compute ρ s.t.

$$\rho \leq Prob_{Init}(\Diamond F) \leq \rho + \epsilon$$

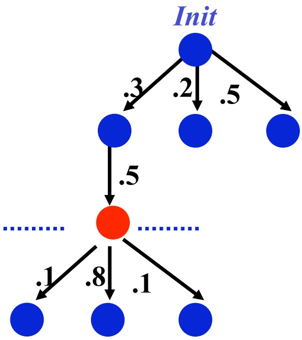


Until **Yes** $+$ **No** $> 1 - \epsilon$

Quantitative Reachability Analysis

Given ϵ ; compute ρ s.t.

$$\rho \leq Prob_{Init}(\Diamond F) \leq \rho + \epsilon$$



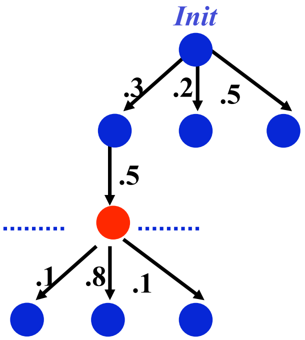
After Termination:

$$\textbf{Yes} \leq Prob_{Init}(\Diamond F) \leq \textbf{Yes} + \epsilon$$

Quantitative Reachability Analysis

Given ϵ ; compute ρ s.t.

$$\rho \leq Prob_{Init}(\Diamond F) \leq \rho + \epsilon$$

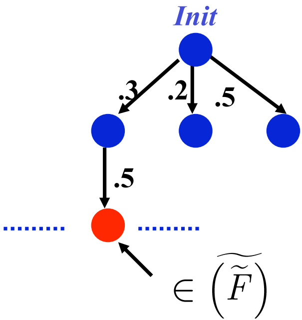


Decisiveness:
Termination Guaranteed

Quantitative Repeated Reachability Analysis

Given ϵ ; compute ρ s.t.

$$\rho \leq Prob_{Init}\,(\Box\Diamond F) \leq \rho + \epsilon$$



Yes: = Yes + .15

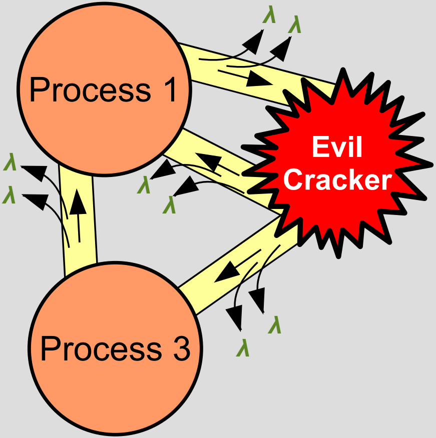
Finite Attractor:
Termination Guaranteed

Stochastic Games with Lossy Channels

Stochastic Games with Lossy Channels

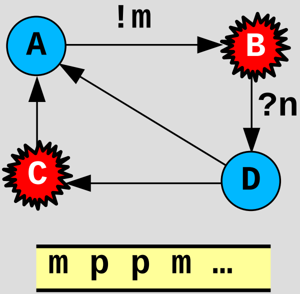
- turn-based stochastic games
- induced by PLCS (Probabilistic Lossy Channel Systems)
- repeated reachability objectives
- almost sure winning conditions
- we show:
  - pure memoryless determined
  - effective construction of winning set of configurations

Game Probabilistic Lossy Channel Systems (GPLCS)



- **Game:**
  Interaction with evil cracker
- **Probabilistic:**
  Messages lost randomly (probability λ)

Game Probabilistic Lossy Channel Systems (GPLCS)



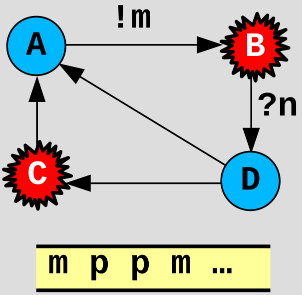
Sufficient:

- One channel
- One process
- Each state controlled by a player

Properties:

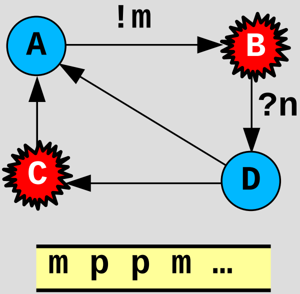
- Infinite state space
- Perfect channel = Turing machine

Game Probabilistic Lossy Channel Systems (GPLCS)

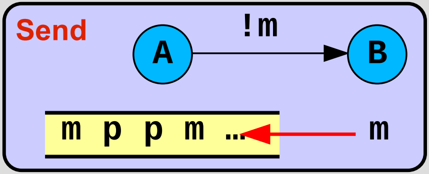


State: s = (A, mpnn)

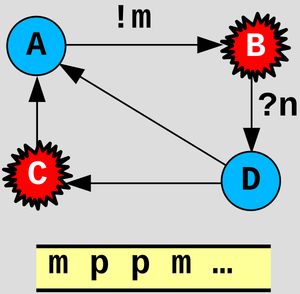
Game Probabilistic Lossy Channel Systems (GPLCS)



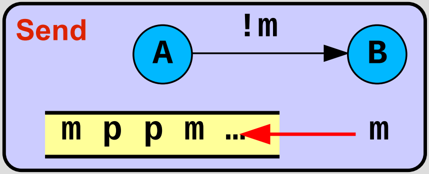
Transitions:

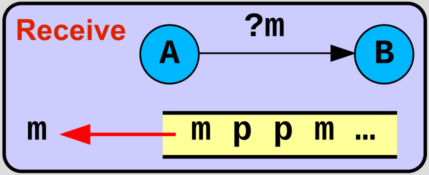


Game Probabilistic Lossy Channel Systems (GPLCS)

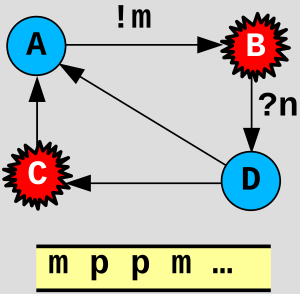


Transitions:

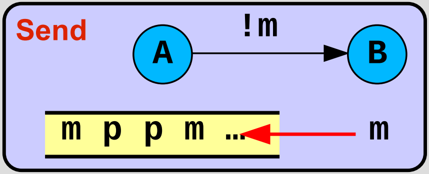


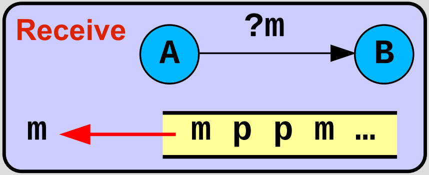


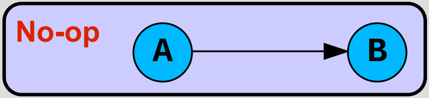
Game Probabilistic Lossy Channel Systems (GPLCS)



Transitions:







Game Probabilistic Lossy Channel Systems (GPLCS)

Probabilistic message loss:

p m n p n …

$\Downarrow$

p n n …

- **Each transition:**

  each message lost with prob $\lambda > 0$, independently

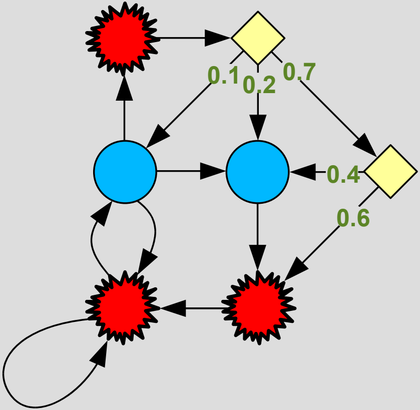
Stochastic Games

- **Every GPLCS induces a stochastic game**
- **Infinite state**
- **3 types of states:**

Player **Good**

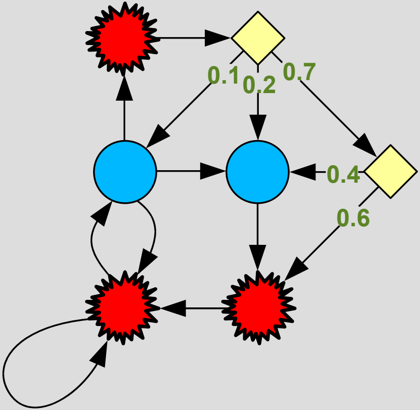
Player **Bad**

Player **Random**



Stochastic Games

- Every GPLCS induces a stochastic game
- Infinite state
- 3 types of states:



Stochastic Games

- **Strategy**
  = selection of outgoing transitions

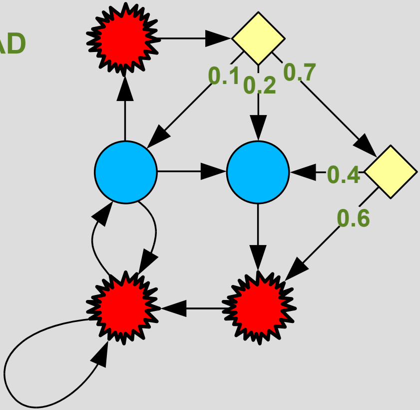
- **Strategies for GOOD & BAD**

$\Downarrow$

Only probabilistic choices remain

$\Downarrow$

"**Prob(event)**" well-defined



Stochastic Games

- **Strategy**
  = selection of outgoing transitions

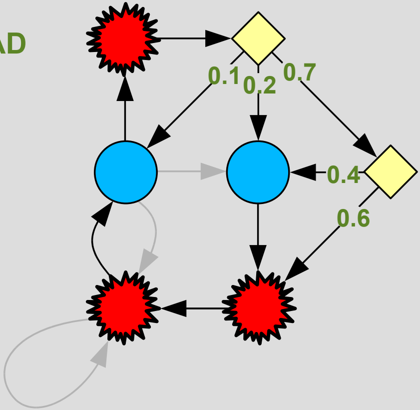
- **Strategies for GOOD & BAD**

$\Downarrow$

Only probabilistic choices remain

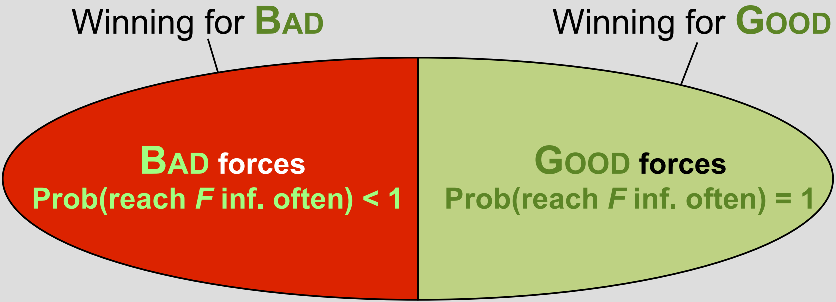
$\Downarrow$

"**Prob(event)**" well-defined



Repeated Reachability for GPLCS

- **Input:**
  - GPLCS
  - Set ***F*** of final states
- **Output: Partition of states:**



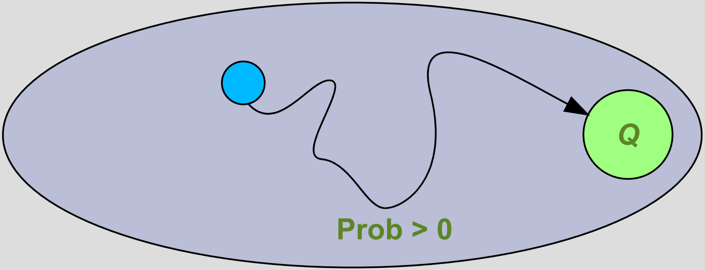
Stochastic Games with Lossy Channels

Algorithm

- Subroutine: Force-set
- algorithm

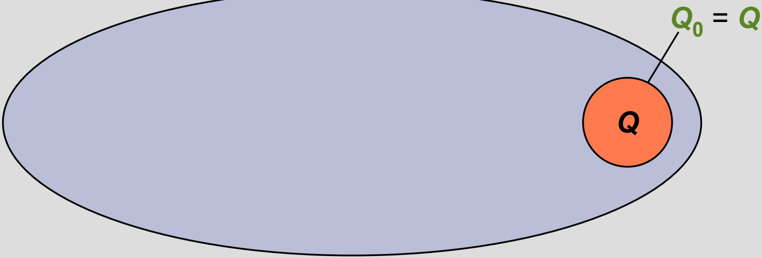
Subroutine: Force-set

- **Force-set ≈ "Reachability for games"**
  - Given target set ***Q***
  - Compute the set of states where **GOOD** can force **Prob(reach *Q*) > 0**
- **Backward search**



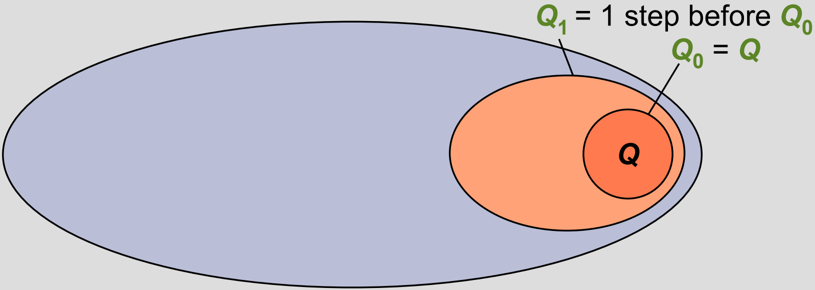
Subroutine: Force-set Construction

- **Force-set ≈ "Reachability for games"**
  - Given target set Q
  - Compute the set of states where GOOD can force **Prob(reach Q) > 0**
- **Backward search**



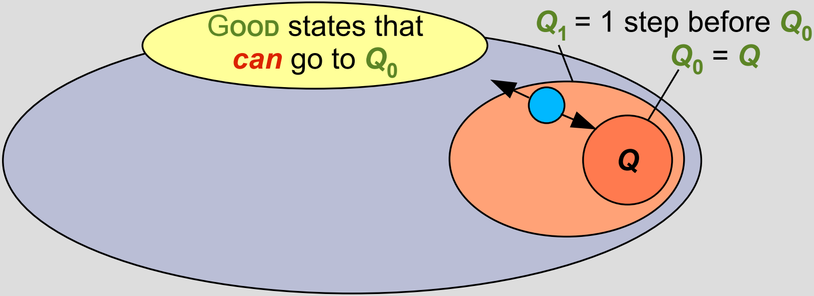
Subroutine: Force-set Construction

- **Force-set ≈ "Reachability for games"**
  - Given target set Q
  - Compute the set of states where GOOD can force **Prob(reach Q) > 0**
- **Backward search**



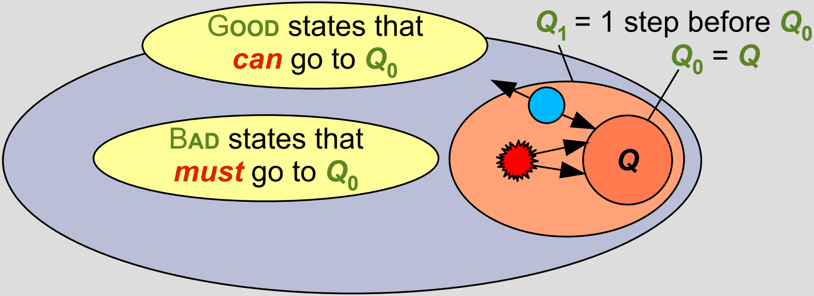
Subroutine: Force-set Construction

- **Force-set ≈ "Reachability for games"**
  - Given target set Q
  - Compute the set of states where GOOD can force **Prob(reach Q) > 0**
- **Backward search**



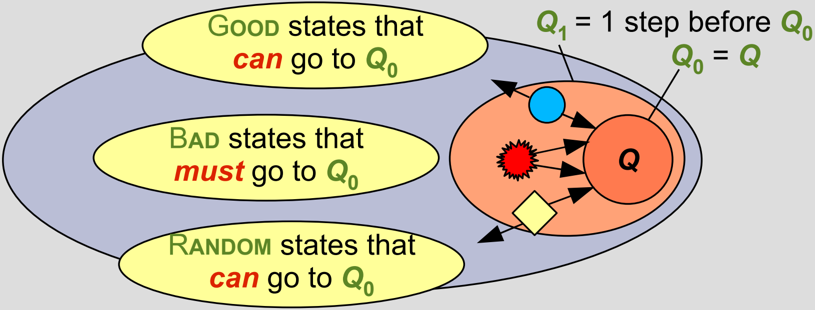
Subroutine: Force-set Construction

- **Force-set ≈ "Reachability for games"**
  - Given target set Q
  - Compute the set of states where GOOD can force **Prob(reach Q) > 0**
- **Backward search**



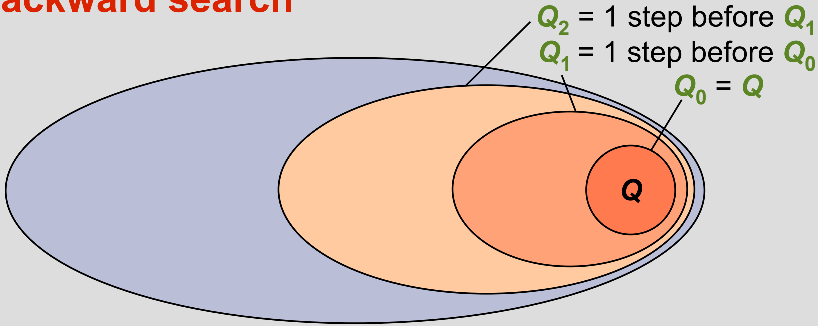
Subroutine: Force-set Construction

- **Force-set ≈ "Reachability for games"**
  - Given target set Q
  - Compute the set of states where GOOD can force **Prob(reach Q) > 0**
- **Backward search**



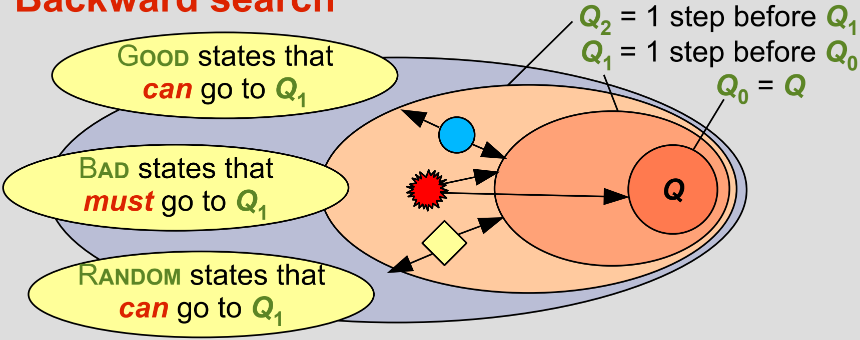
Subroutine: Force-set Construction

- **Force-set ≈ "Reachability for games"**
  - Given target set Q
  - Compute the set of states where GOOD can force **Prob(reach Q) > 0**
- **Backward search**



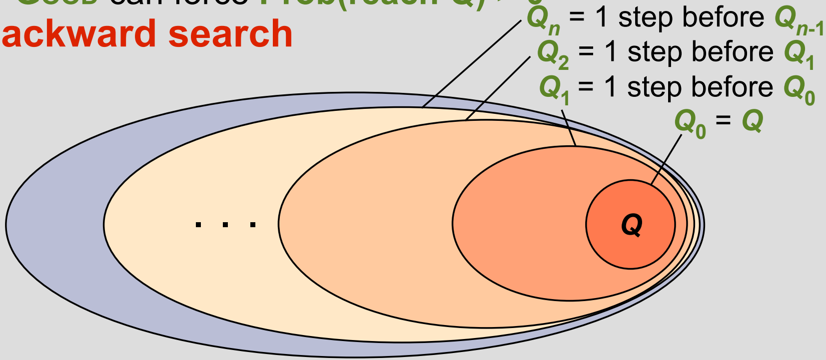
Subroutine: Force-set Construction

- **Force-set ≈ "Reachability for games"**
  - Given target set Q
  - Compute the set of states where GOOD can force Prob(reach Q) > 0
- **Backward search**



Subroutine: Force-set Construction

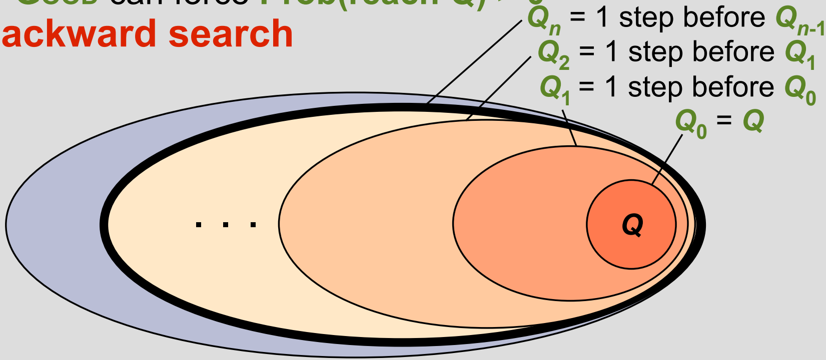
- **Force-set ≈ "Reachability for games"**
  - Given target set Q
  - Compute the set of states where GOOD can force **Prob(reach Q) > 0**
- **Backward search**



Subroutine: Force-set Convergence

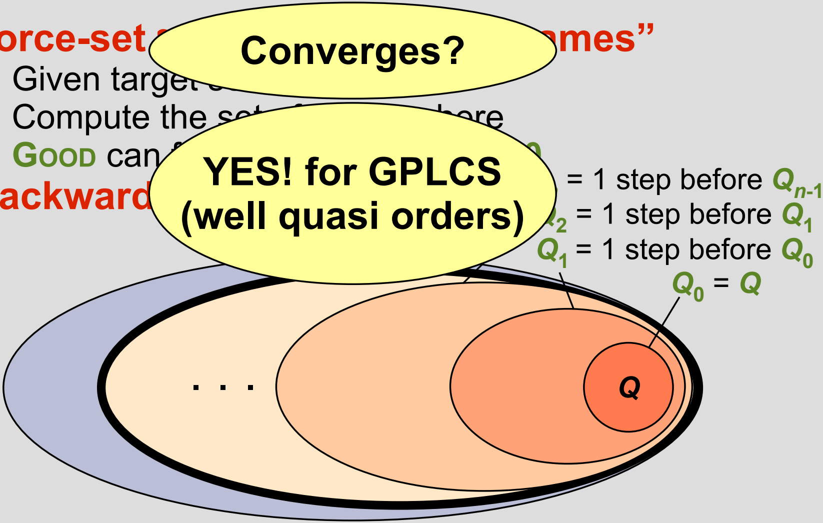
- **Force-set** … **mes"**
  - Given target …
  - Compute the set of states where GOOD can force **Prob(reach Q) > 0**
- **Backward search**

Converges?



Subroutine: Force-set Convergence

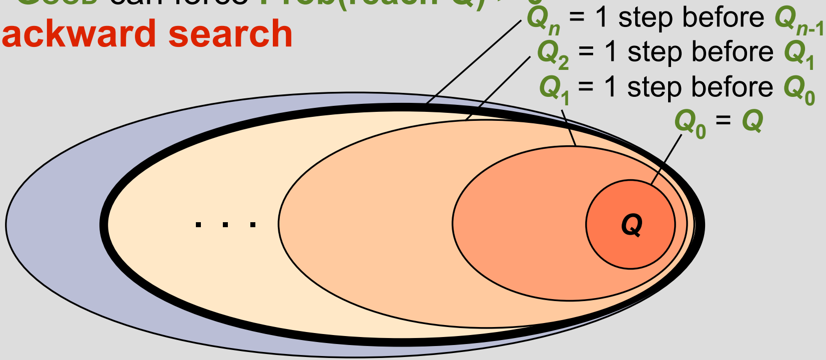
- **Force-set ... mes"**
  - Given target ...
  - Compute the set ... here
    GOOD can ...
- **Backward** ...



Subroutine: Force-set Correctness

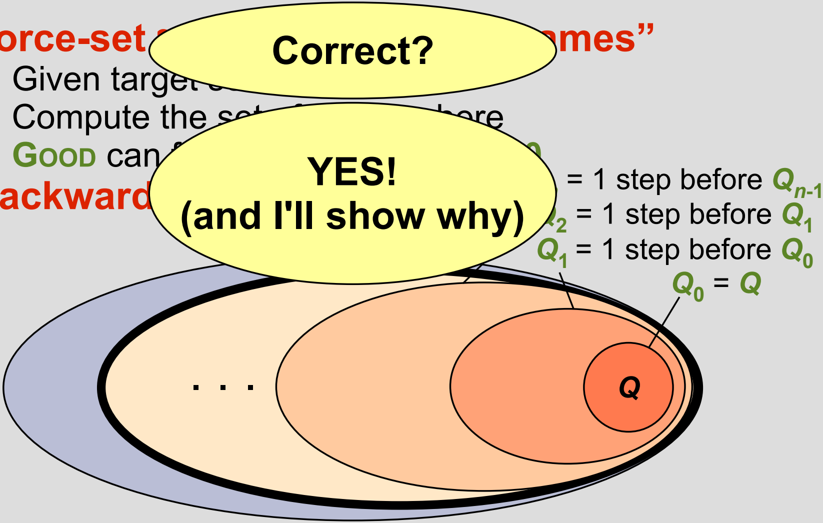
Correct?

- **Force-set … ames”**
  - Given target …
  - Compute the set of states where GOOD can force **Prob(reach Q) > 0**
- **Backward search**



Subroutine: Force-set Correctness

- **Force-set** [partially hidden] **mes"**
  - Given target [partially hidden]
  - Compute the se[partially hidden]here
    GOOD can [partially hidden]
- **Backward** [partially hidden]

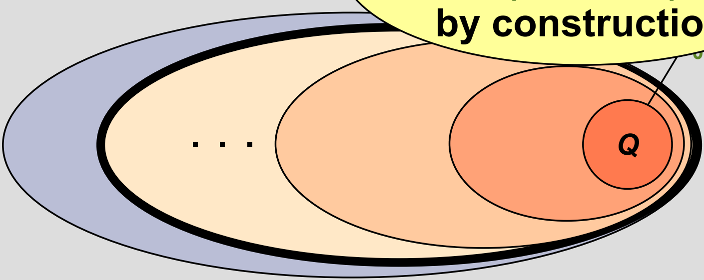


Subroutine: Force-set Correctness

- **Force-set** … **ames”**
  - Given target …
  - Compute the set of states whe…
    Good can force Prob(re…
- **Backward search**

Correct?

In Q_n, Good can force Prob(reach Q) > 0 by construction



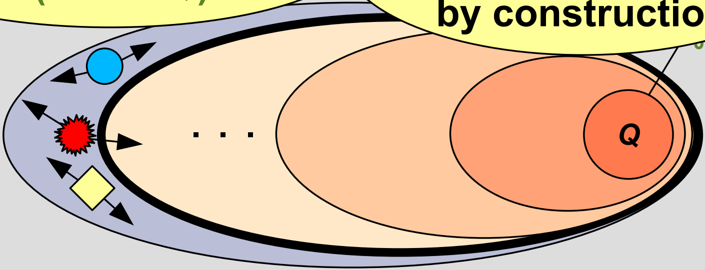
Subroutine: Force-set Correctness

- **Force-set** ... **mes"**
  - Given target ...
  - Compute the set of states where ...

Correct?

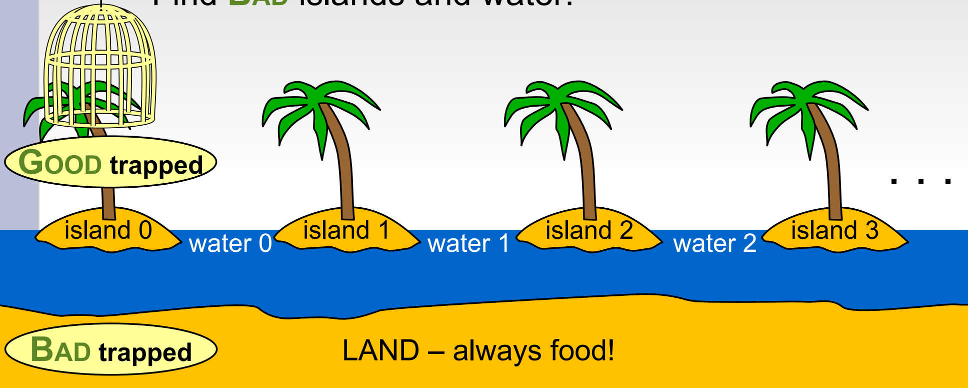
Outside Q_n, BAD can force Prob(reach Q) = 0

In Q_n, GOOD can force Prob(reach Q) > 0 by construction



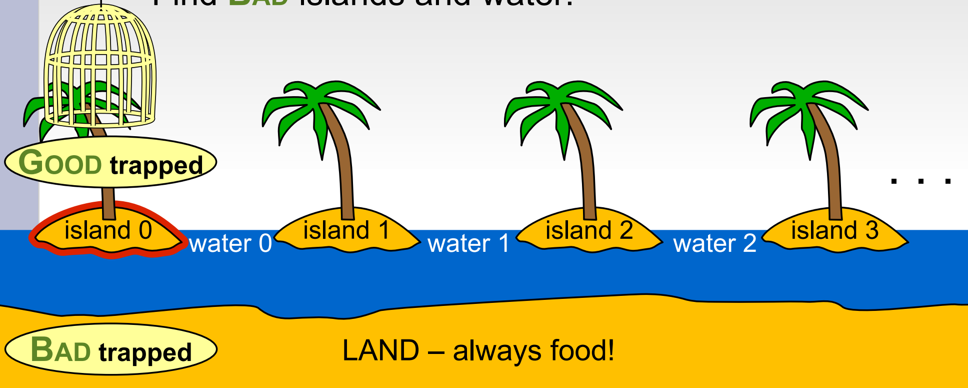
Algorithm Idea

- A world of islands and water
- No food (= ***F***) on islands
- In water, may float to island (possibly elsewhere, too)
- Find **BAD** islands and water!



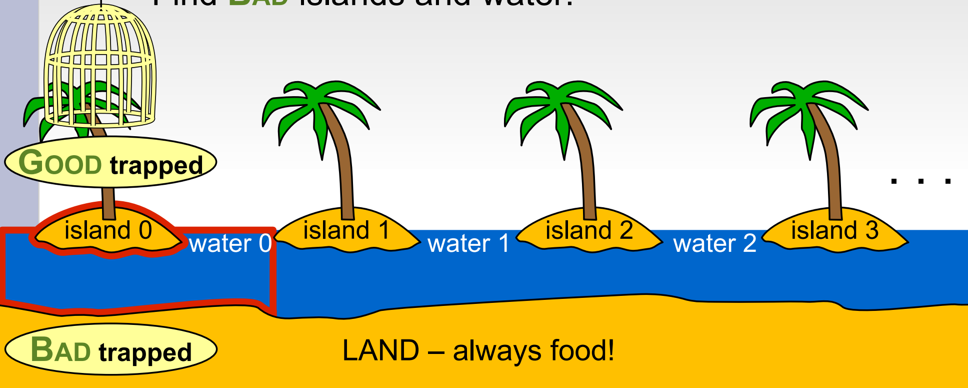
Algorithm Idea

- A world of islands and water
- No food (= ***F***) on islands
- In water, may float to island (possibly elsewhere, too)
- Find **BAD** islands and water!



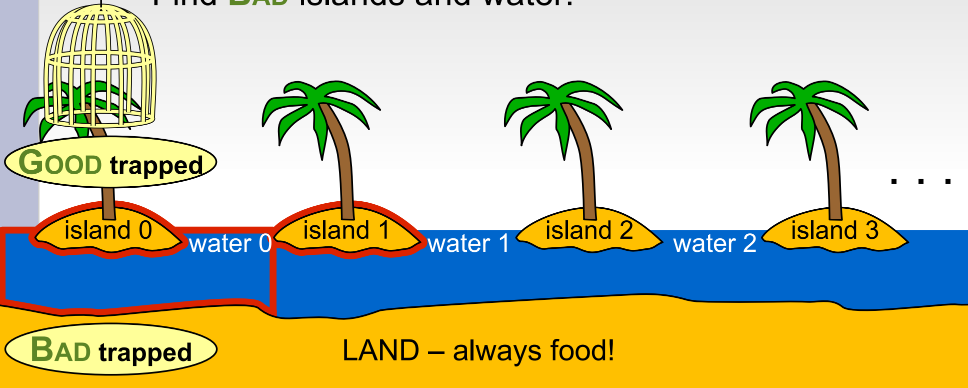
Algorithm Idea

- A world of islands and water
- No food (= ***F***) on islands
- In water, may float to island (possibly elsewhere, too)
- Find **BAD** islands and water!



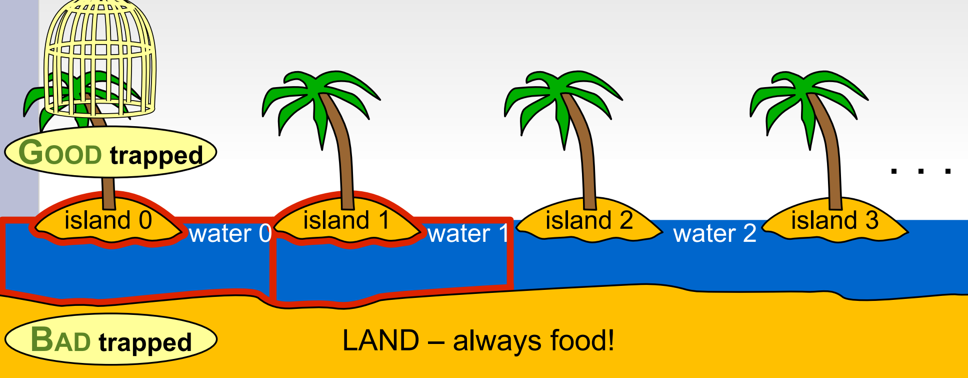
Algorithm Idea

- A world of islands and water
- No food (= ***F***) on islands
- In water, may float to island (possibly elsewhere, too)
- Find **BAD** islands and water!



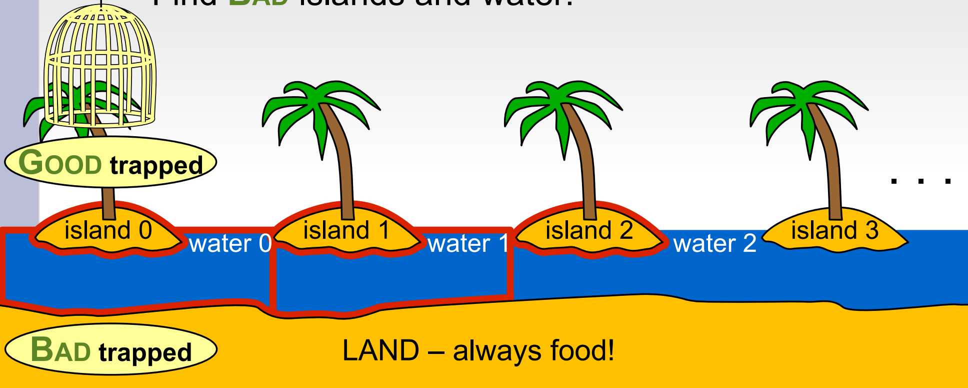
Algorithm Idea

- A world of islands and water
- No food (= *F*) on islands
- In water, may float to island (possibly elsewhere, too)
- Find **BAD** islands and water!



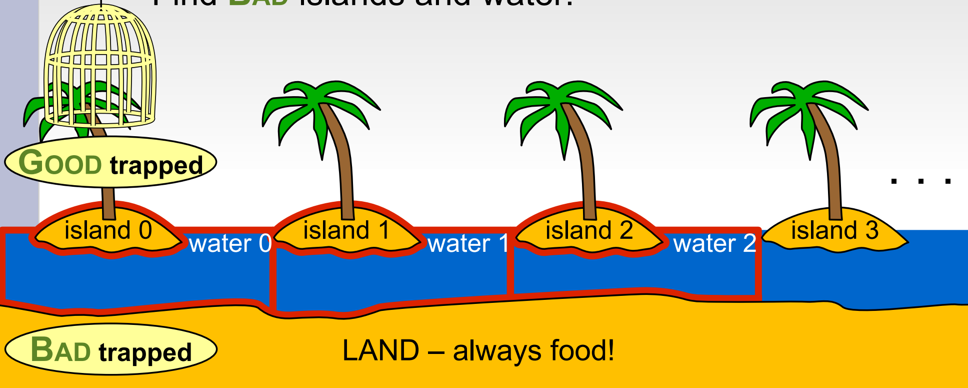
Algorithm Idea

- A world of islands and water
- No food (= ***F***) on islands
- In water, may float to island (possibly elsewhere, too)
- Find **BAD** islands and water!



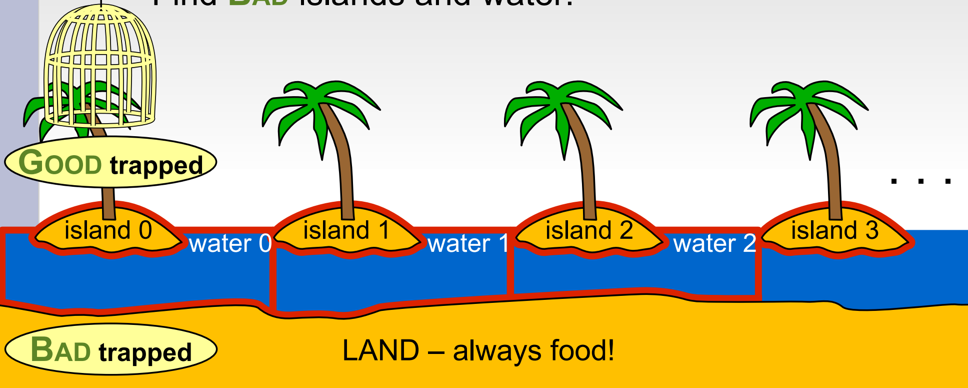
Algorithm Idea

- A world of islands and water
- No food (= ***F***) on islands
- In water, may float to island (possibly elsewhere, too)
- Find **BAD** islands and water!



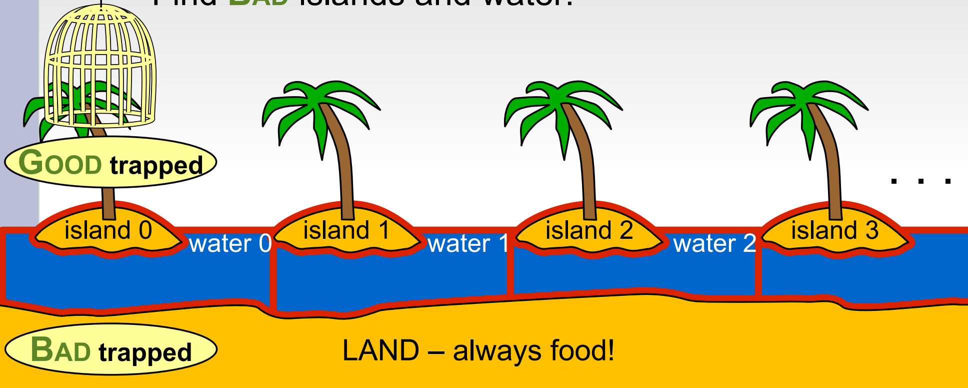
Algorithm Idea

- A world of islands and water
- No food (= *F*) on islands
- In water, may float to island (possibly elsewhere, too)
- Find **Bad** islands and water!

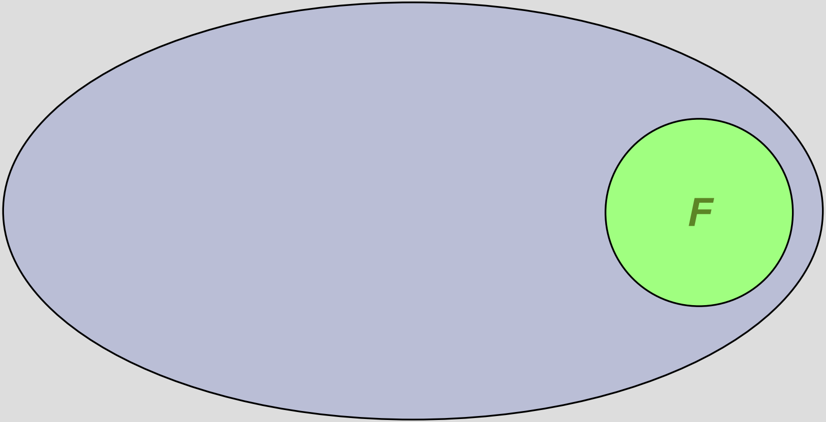


Algorithm Idea

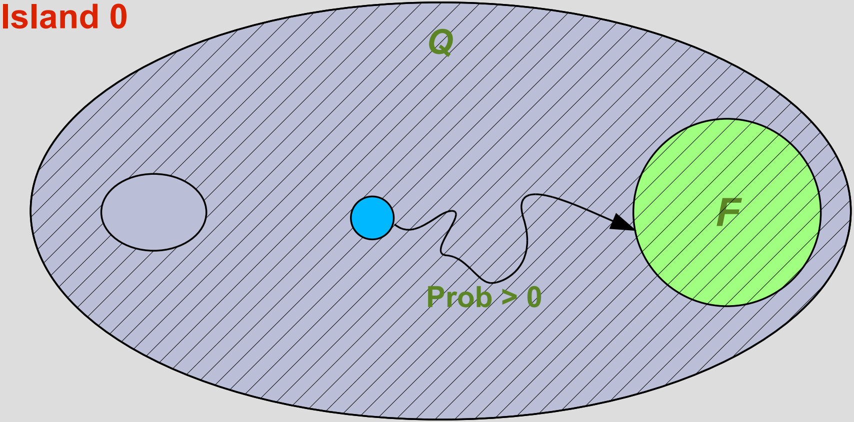
- A world of islands and water
- No food (= ***F***) on islands
- In water, may float to island (possibly elsewhere, too)
- Find **BAD** islands and water!



Algorithm Overview

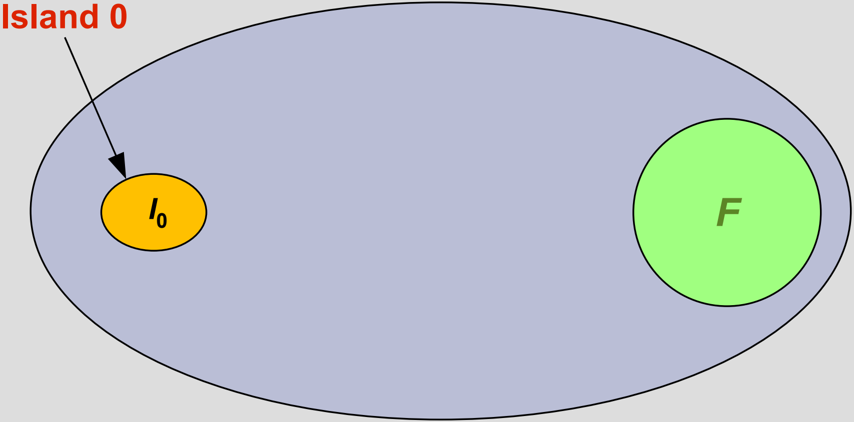


Algorithm Overview



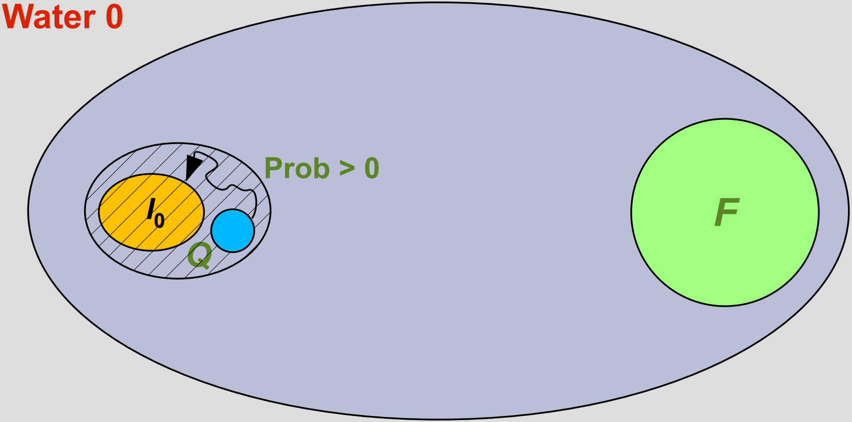
- Compute *Q*: **Good** can force **Prob(reach *F*) > 0**

Algorithm Overview



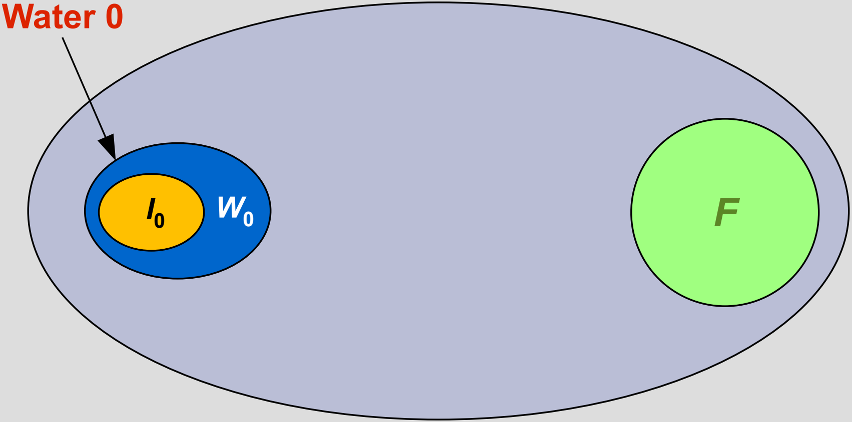
- Compute Q: **Good** can force **Prob(reach F) > 0**
- I_0 = complement(Q)
- So **Bad** can force **Prob(reach F) = 0** on I_0

Algorithm Overview



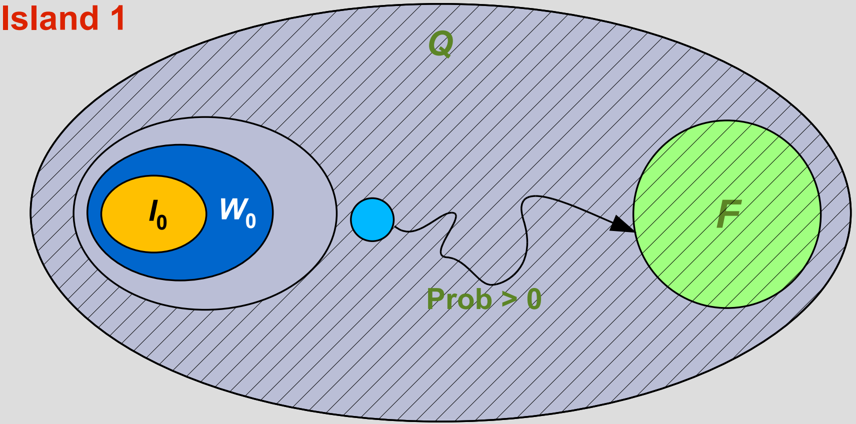
- Compute Q: BAD can force Prob(reach I_0) > 0

Algorithm Overview



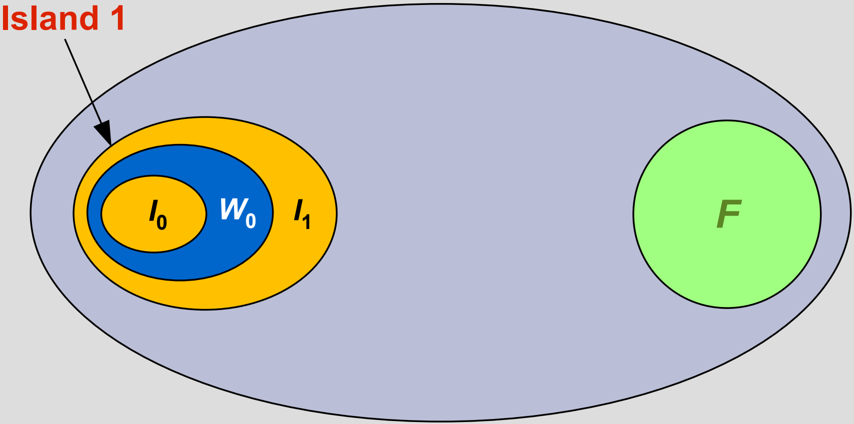
- Compute Q: BAD can force Prob(reach I_0) > 0
- $W_0 = Q$

Algorithm Overview



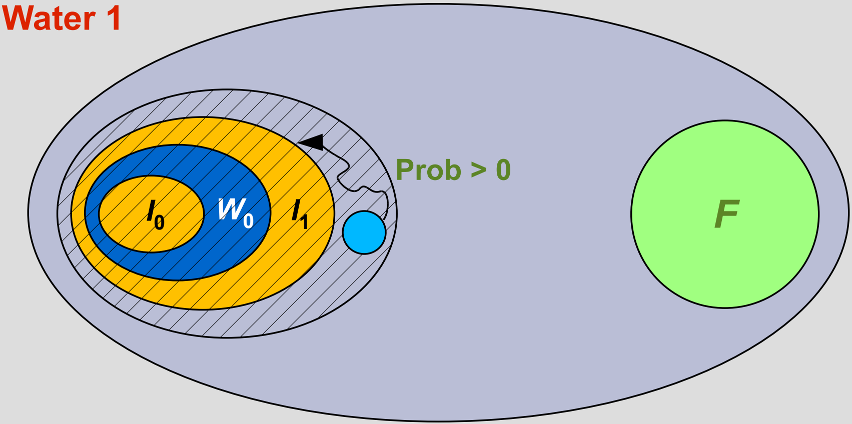
- Compute ***Q***:
  Good can force **Prob(reach *F*) > 0**, **avoiding I_0 and W_0**

Algorithm Overview



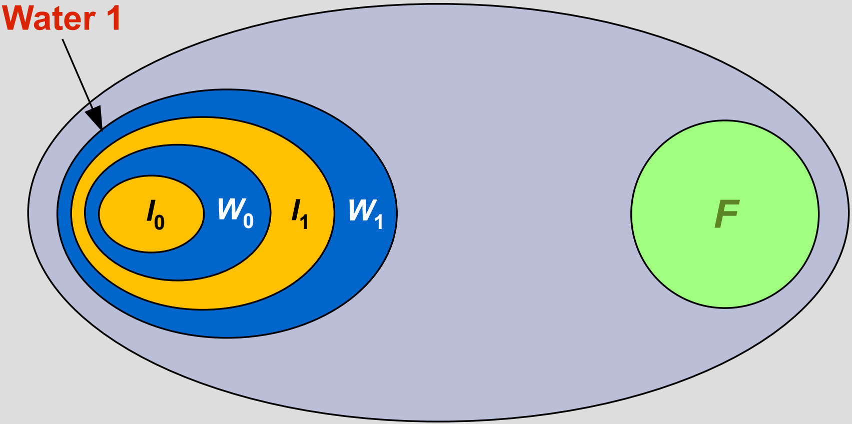
- Compute ***Q***:
  GOOD can force **Prob(reach *F*) > 0**, **avoiding I_0 and W_0**
- I_1 = complement(Q)

Algorithm Overview



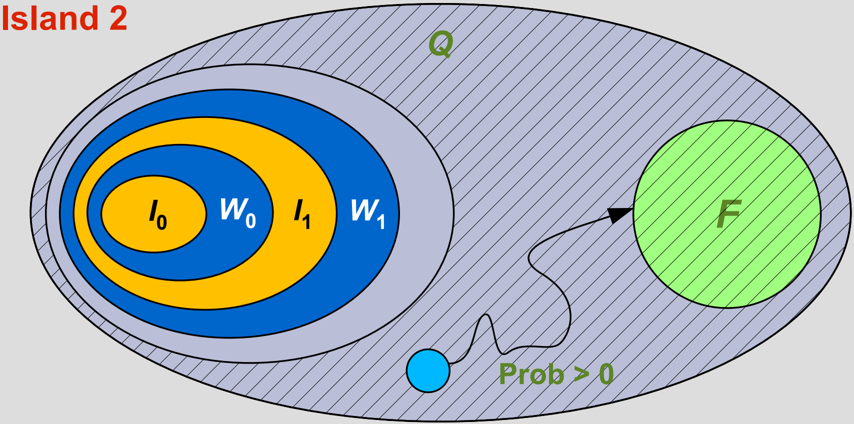
- Compute Q: Bad can force Prob(reach $I_0 \cup W_0 \cup I_1$) > 0

Algorithm Overview



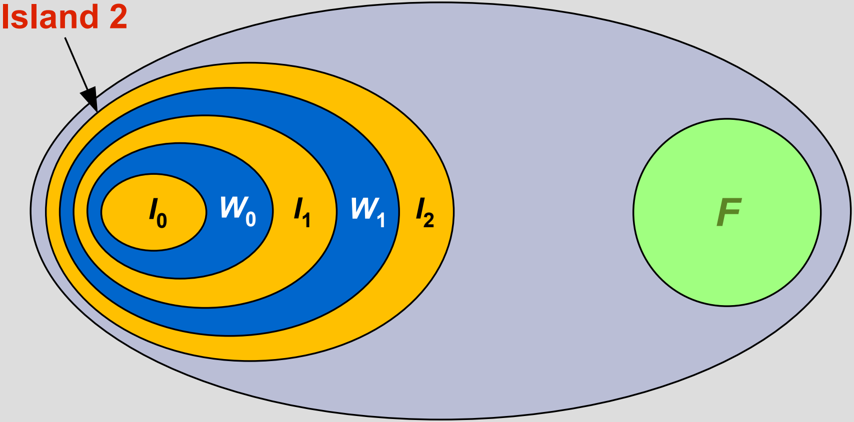
- Compute Q: BAD can force Prob(reach $I_0 \cup W_0 \cup I_1$) > 0
- $W_1 = Q$

Algorithm Overview



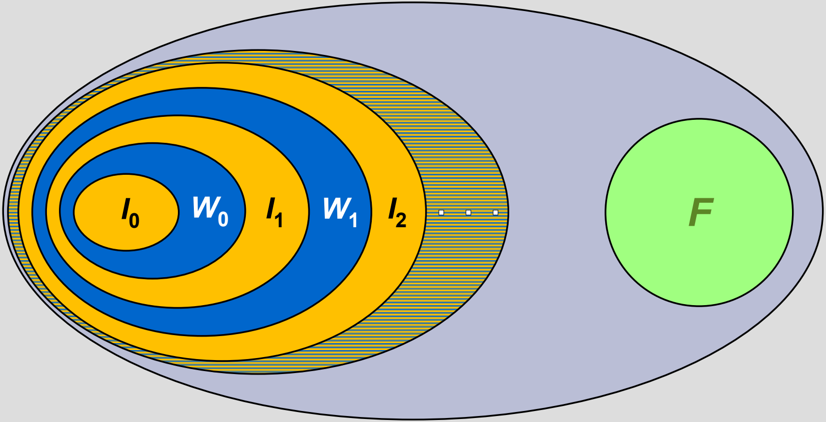
- Compute Q:
  GOOD can force Prob(reach F) > 0, avoiding I_0, W_0, I_1, W_1

Algorithm Overview

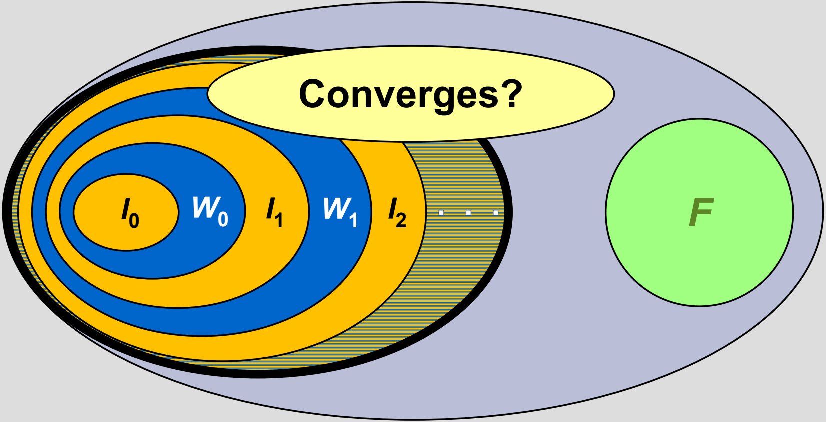


- Compute Q:
  Good can force Prob(reach F) > 0, avoiding I_0, W_0, I_1, W_1
- I_2 = complement(Q)

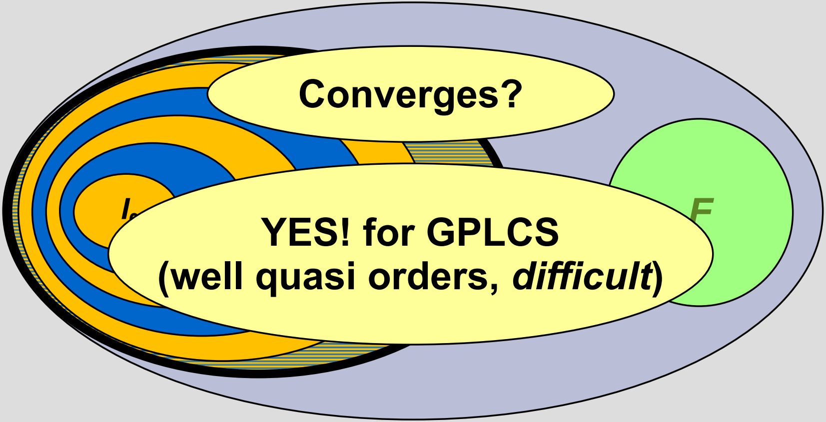
Algorithm Overview



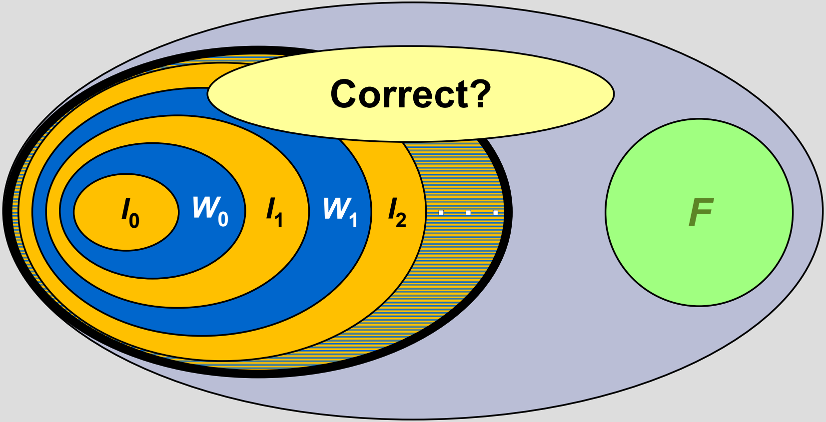
Algorithm Convergence



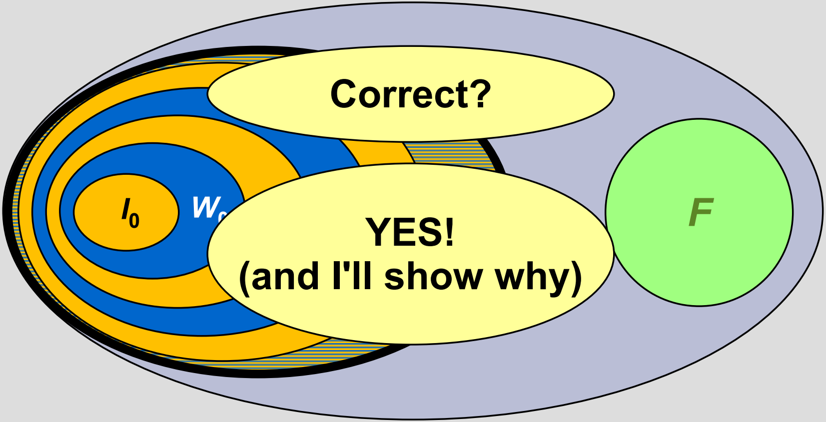
Algorithm Convergence



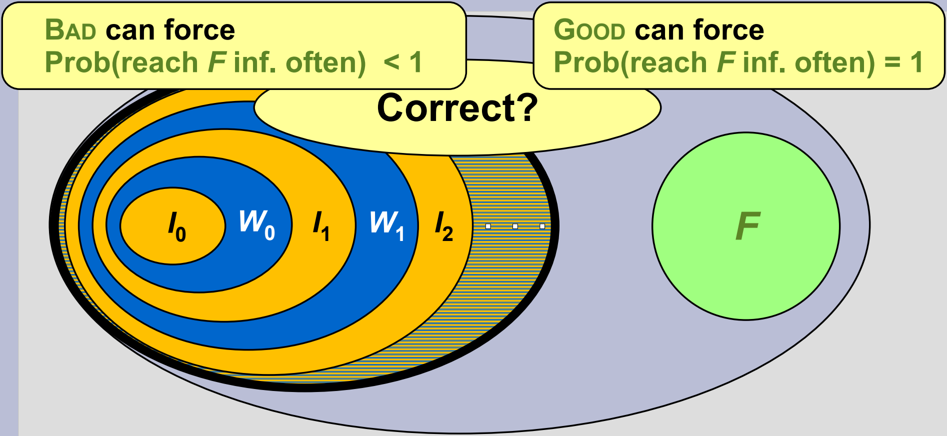
Algorithm Correctness



Algorithm Correctness

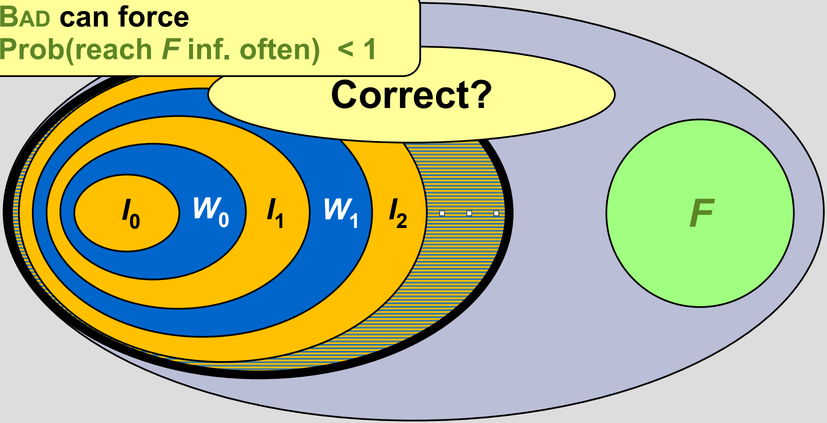


Algorithm Correctness



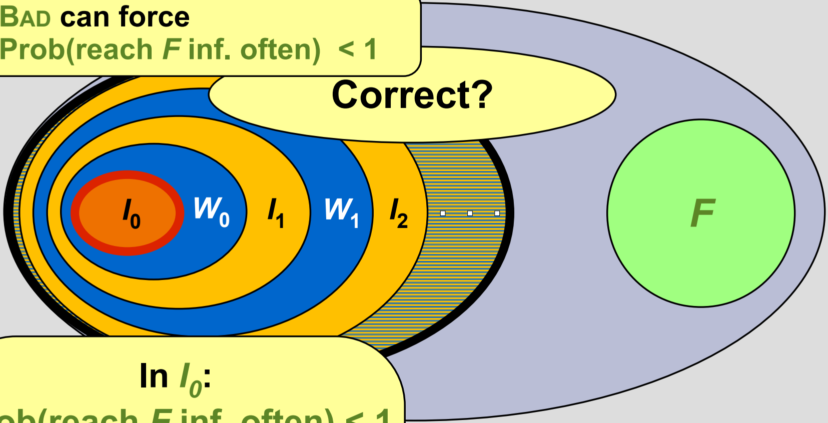
Algorithm Correctness

BAD can force
Prob(reach *F* inf. often) < 1



Algorithm Correctness

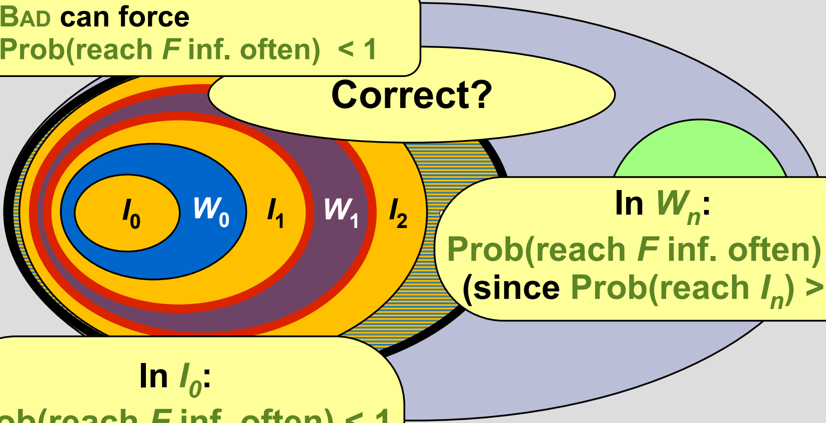
BAD can force
Prob(reach *F* inf. often) < 1



In I_0:
Prob(reach *F* inf. often) < 1
(since even
Prob(reach *F*) = 0)

Algorithm Correctness

BAD can force
Prob(reach *F* inf. often) < 1

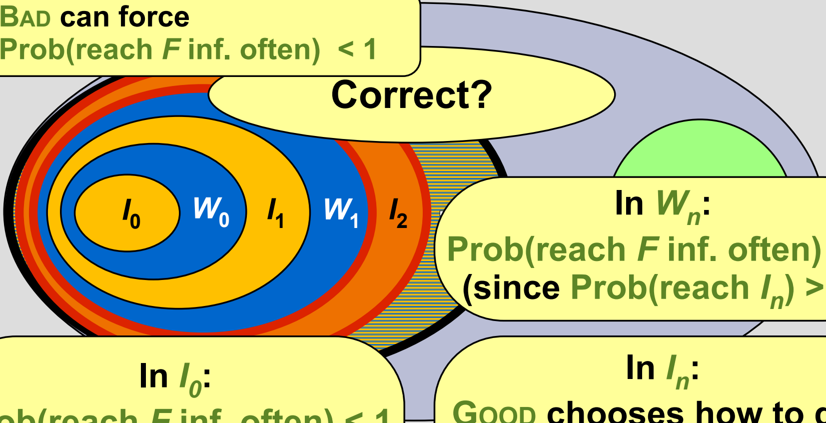


In W_n:
Prob(reach *F* inf. often) < 1
(since Prob(reach I_n) > 0)

In I_0:
Prob(reach *F* inf. often) < 1
(since even
Prob(reach *F*) = 0)

Algorithm Correctness

BAD can force
Prob(reach *F* inf. often) < 1



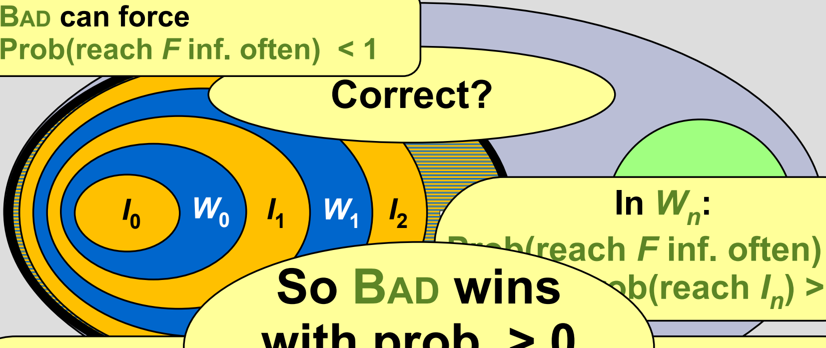
In W_n:
Prob(reach *F* inf. often) < 1
(since Prob(reach I_n) > 0)

In I_0:
Prob(reach *F* inf. often) < 1
(since even
Prob(reach *F*) = 0)

In I_n:
GOOD chooses how to die:
- stay in I_n and lose (no *F*)
- or go to W_{n-1} and lose

Algorithm Correctness

BAD can force
Prob(reach *F* inf. often) < 1



Correct?

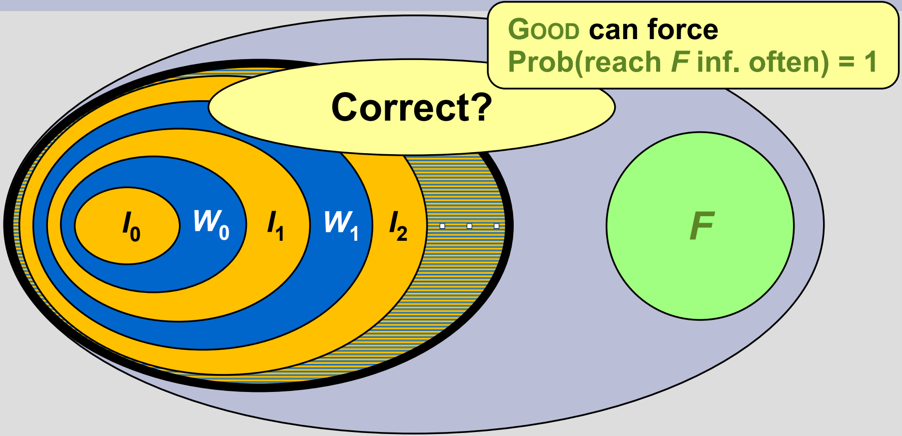
In W_n:
Prob(reach *F* inf. often) < 1
...ob(reach I_n) > 0)

So BAD wins with prob. > 0 in all W_n, I_n

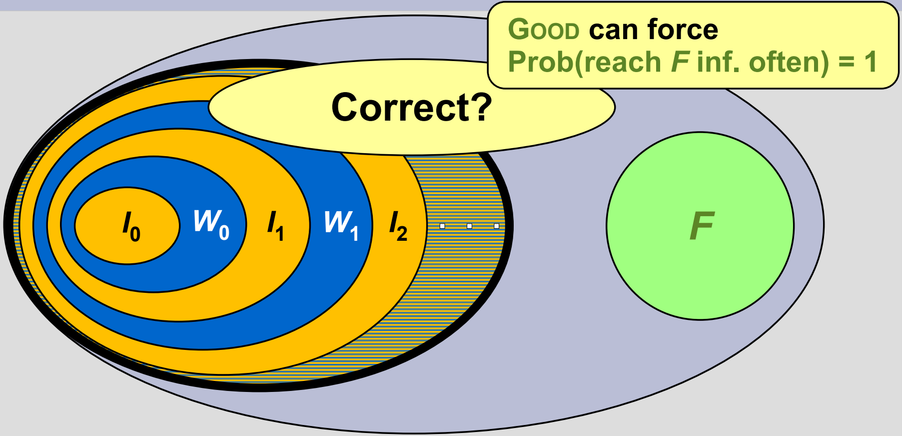
In I_0...
Prob(reach *F* inf. o...
(since even
Prob(reach *F*) = 0)

...n I_n:
...ooses how to die:
- stay in I_n and lose (no *F*)
- or go to W_{n-1} and lose

Algorithm Correctness

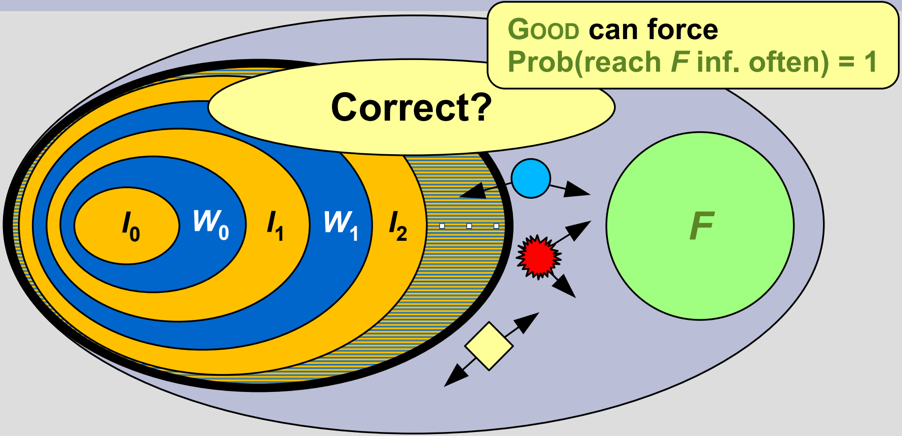


Algorithm Correctness



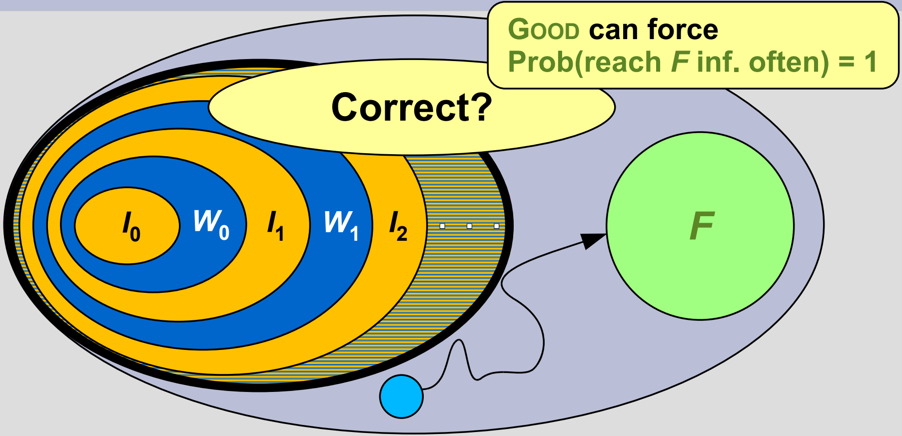
- **Convergence means:**
  - No more water
  - No more island

Algorithm Correctness



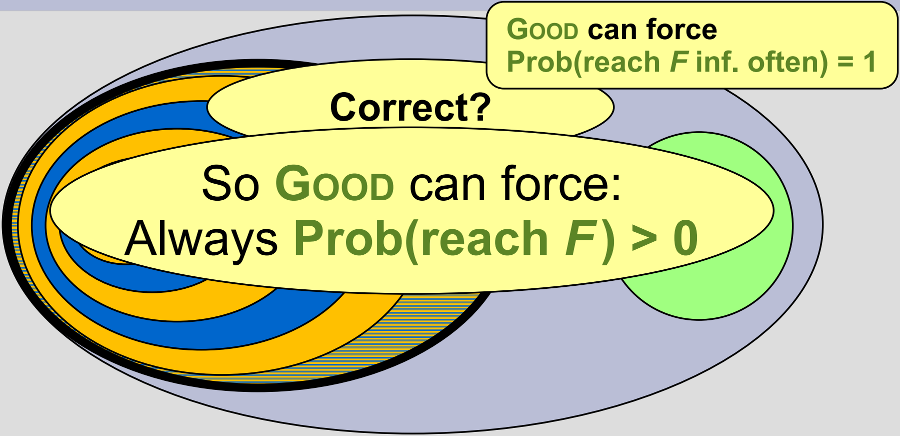
- **Convergence means:**
  - No more water $\Rightarrow$ Good can stay outside I_n, W_n
  - No more island

Algorithm Correctness



- **Convergence means:**
  - No more water ⟹ GOOD can stay outside I_n, W_n
  - No more island ⟹ GOOD can force Prob(reach F) > 0

Algorithm Correctness



- **Convergence means:**
  - No more water $\Rightarrow$ GOOD can stay outside I_n, W_n
  - No more island $\Rightarrow$ GOOD can force **Prob(reach *F*) > *0***

Algorithm Correctness

GOOD can force
Prob(reach *F* inf. often) = 1

Correct?

So GOOD can force:
Always **Prob(reach *F*) > 0**

For GPLCS, this implies
Prob(reach *F* inf. often) = 1
(using attractors, *difficult*)

- **Convergence**
  - No more water ⇛ GOOD can stay outside I_n, W_n
  - No more island ⇛ GOOD can force **Prob(reach *F*) > *0***

Conclusions

- Decisive Markov Chains
- Stochastic Games on LCS
- Other work: Eager Markov Chains:
  - Computing expected reward (cost) of runs
  - Expected residence time

Future Work

- probabilistic timed Petri nets
- distributed systems with probabilistic components