Verification of Randomized Consensus Algorithms under Round-Rigid Adversaries

Marijana Lazić
TU München

Nathalie Bertrand Igor Konnov Josef Widder



CONCUR, August 30, 2019

European Research Council
Established by the European Commission

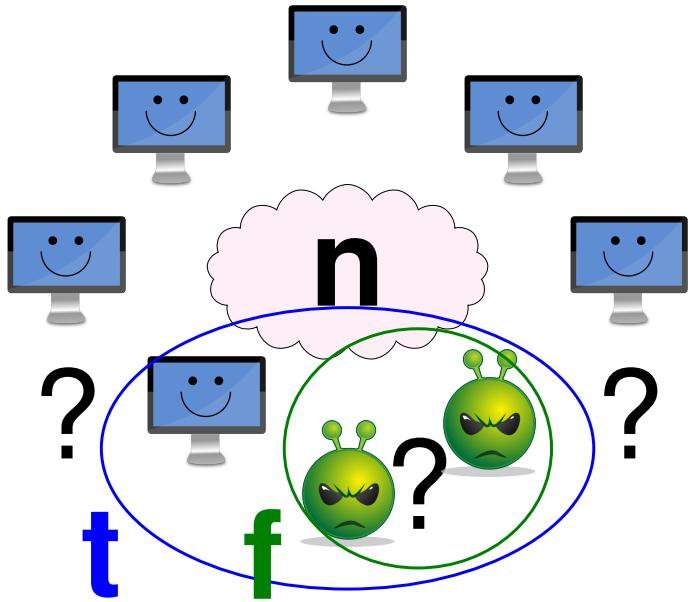
Technische Universität München

logics LOGICAL METHODS IN COMPUTER SCIENCE

FWF Der Wissenschaftsfonds.

WWTF VIENNA SCIENCE AND TECHNOLOGY FUND

Fault-tolerant distributed algorithms



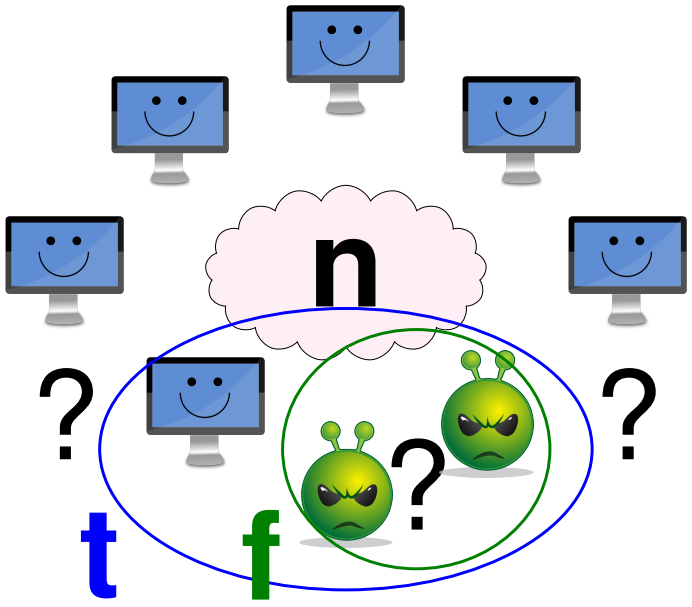
n processes communicate by sending messages **asynchronously**

f processes are faulty (unknown)

t is an upper bound on ***f*** (known)

resilience condition on n, t, and f, e.g., $n > 3t \wedge t \geq f \geq 0$

Parameterized Verification



correct or faulty

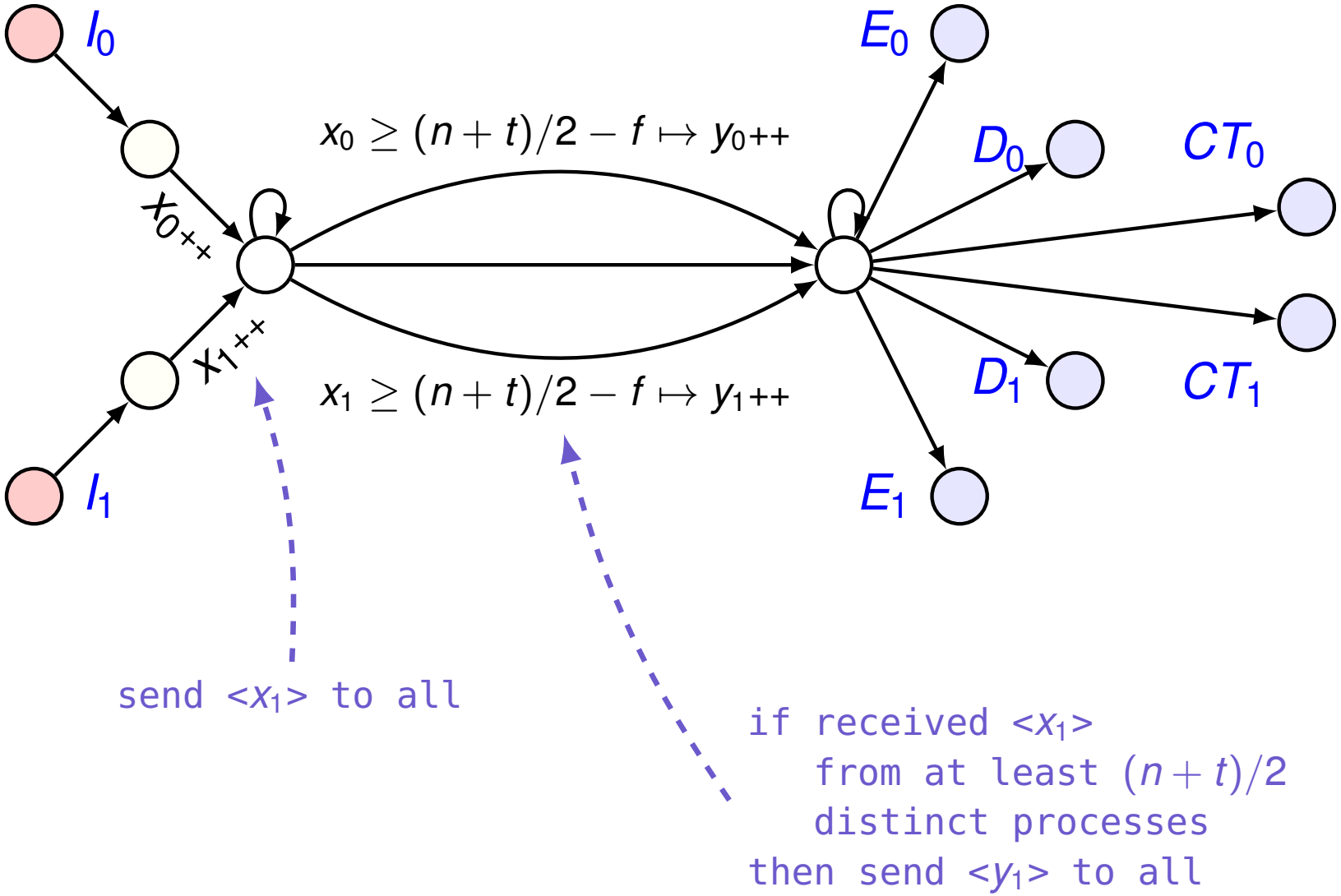
$\forall n, t, f$ with $n > 3t$ and $t \geq f \geq 0$.

$$\underbrace{P(n,t) \parallel P(n,t) \parallel \ldots \parallel P(n,t)}_{\text{n-f correct}} \parallel \underbrace{\text{Faulty} \parallel \ldots \parallel \text{Faulty}}_{\text{f faulty}} \models \mathit{Specs}$$

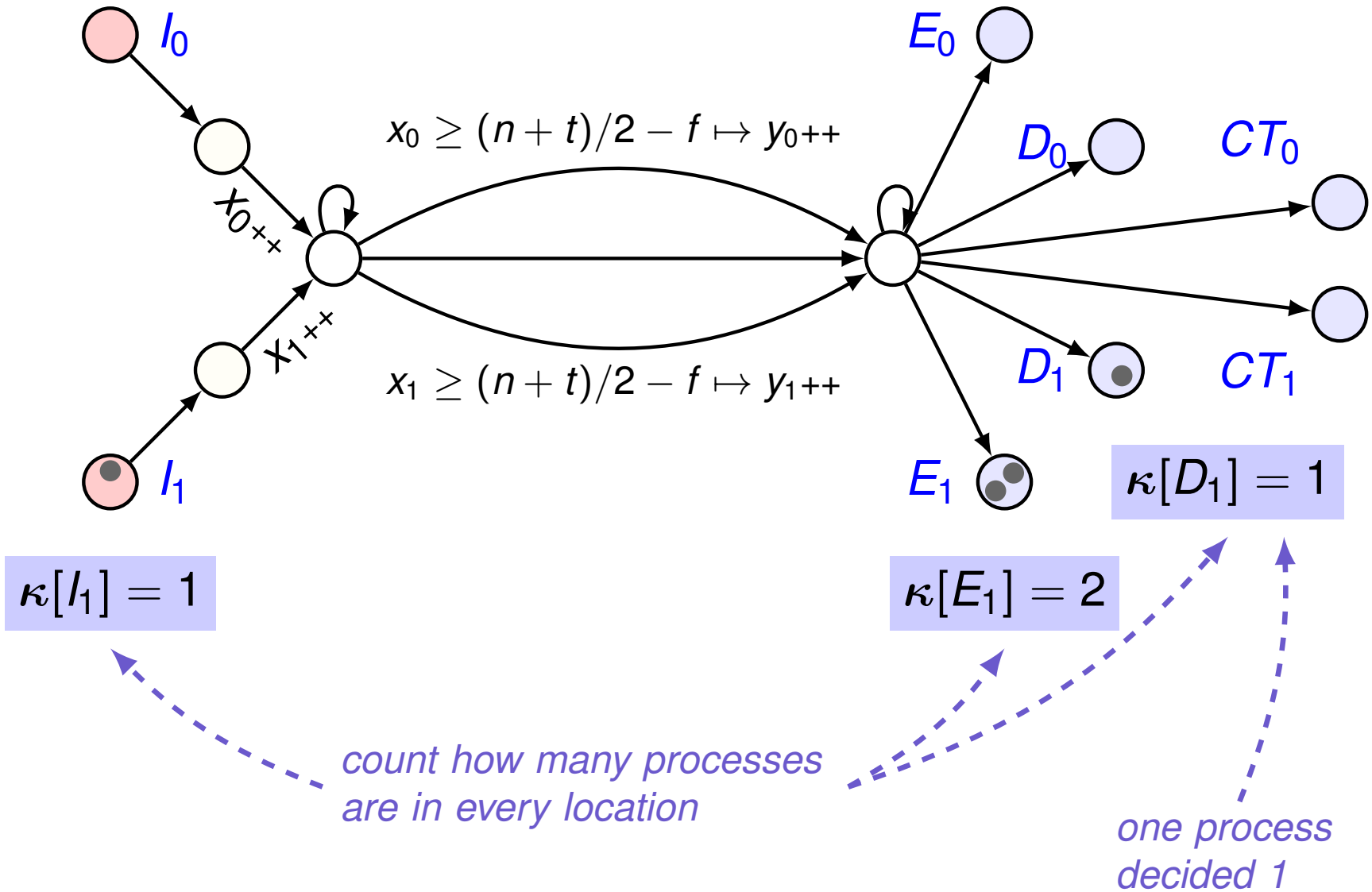
Verification of **non-randomized** distributed algorithms

(Konnov, L., Veith, Widder, POPL 2017)

Threshold automata



Counter System as a Semantic of a TA



Specifications in LTL$_{-X}$ with counters

Agreement : No two correct processes decide differently (safety)

$$\mathbf{F}\ \kappa[D_v] > 0 \quad \rightarrow \quad \mathbf{G}\ \kappa[D_{1-v}] = 0$$

Termination : Eventually all correct processes decide (liveness)

$$\mathbf{F} \bigwedge_{\ell \in \mathcal{L} \setminus \{D_0, D_1\}} \kappa[\ell] = 0$$

We denote this fragment by ELTL$_{FT}$

Verification of Threshold-based Algorithms

Does Sys(TA) $\models \varphi$? (Konnov, L., Veith, Widder, POPL'17)

Given:

- a threshold automaton TA,
- a specification φ in ELTL_{FT}, and
- a resilience condition *RC*,

we can check whether for all parameters satisfying *RC* holds that

$$\text{Sys(TA)} \models \varphi$$

[forsyte.at/software/bymc]

What about consensus?

Theorem (FLP'85)

There is no asynchronous consensus algorithm!

- faulty processes
- asynchrony
- safety requirements
- liveness requirement

What about consensus?

Theorem (FLP'85)

There is no asynchronous consensus algorithm!

- faulty processes
- asynchrony
- safety requirements
- ~~liveness requirement~~
  almost sure termination

Solution: Randomized consensus algorithms

Contribution:

Extend previous result
to verify
randomized consensus algorithms

Randomized consensus algorithm Ben-Or

```
bool v := input_value({0, 1});
int r := 1;
while (true) do
 send (R,r,v) to all;
 wait for n - t messages (R,r,*);

 if received (n + t) / 2 messages (R,r,w)
 then send (P,r,w,D) to all;
 else send (P,r,?) to all;
 wait for n - t messages (P,r,*);

 if received at least t + 1
    messages (P,r,w,D) then {
  v := w;                    /* enough support -> update estimate */
  if received at least (n + t) / 2
   messages (P,r,w,D)
  then decide w;                    /* strong majority -> decide */
 } else v := random({0,1});       /* unclear -> coin toss       */
 r := r + 1;
od
```

[Ben-Or, PODC 1983]

Randomized consensus algorithm Ben-Or

```
bool v := input_value({0, 1});
int r := 1;
while (true) do
 send (R,r,v) to all;
 wait for n - t messages (R,r,*);

 if received (n + t) / 2 messages (R,r,w)
 then send (P,r,w,D) to all;
 else send (P,r,?) to all;
 wait for n - t messages (P,r,*);

 if received at least t + 1
    messages (P,r,w,D) then {
  v := w;                    /* enough support -> update estimate */
  if received at least (n + t) / 2
   messages (P,r,w,D)
  then decide w;                    /* strong majority -> decide */
 } else v := random({0,1});      /* unclear -> coin toss      */
 r := r + 1;
od
```

[Ben-Or, PODC 1983]

Randomized consensus algorithm Ben-Or

```
bool v := input_value({0, 1});
int r := 1;
while (true) do
 send (R,r,v) to all;
 wait for n - t messages (R,r,*);

 if received (n + t) / 2 messages (R,r,w)
 then send (P,r,w,D) to all;
 else send (P,r,?) to all;
 wait for n - t messages (P,r,*);

 if received at least t + 1
    messages (P,r,w,D) then {
  v := w;                    /* enough support -> update estimate */
  if received at least (n + t) / 2
   messages (P,r,w,D)
  then decide w;                  /* strong majority -> decide */
 } else v := random({0,1});      /* unclear -> coin toss      */
 r := r + 1;
od
```

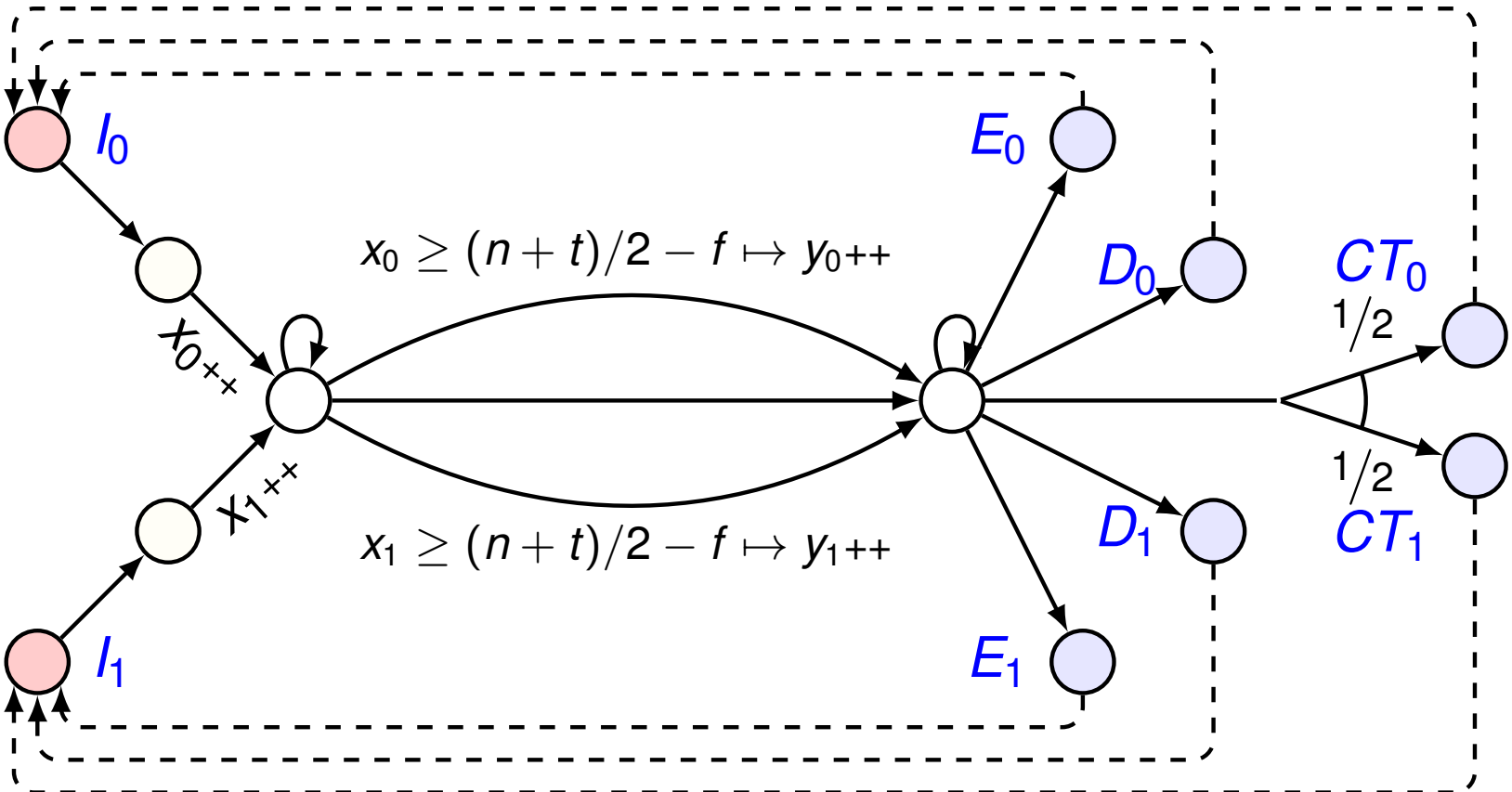
[Ben-Or, PODC 1983]

Randomized consensus algorithm Ben-Or

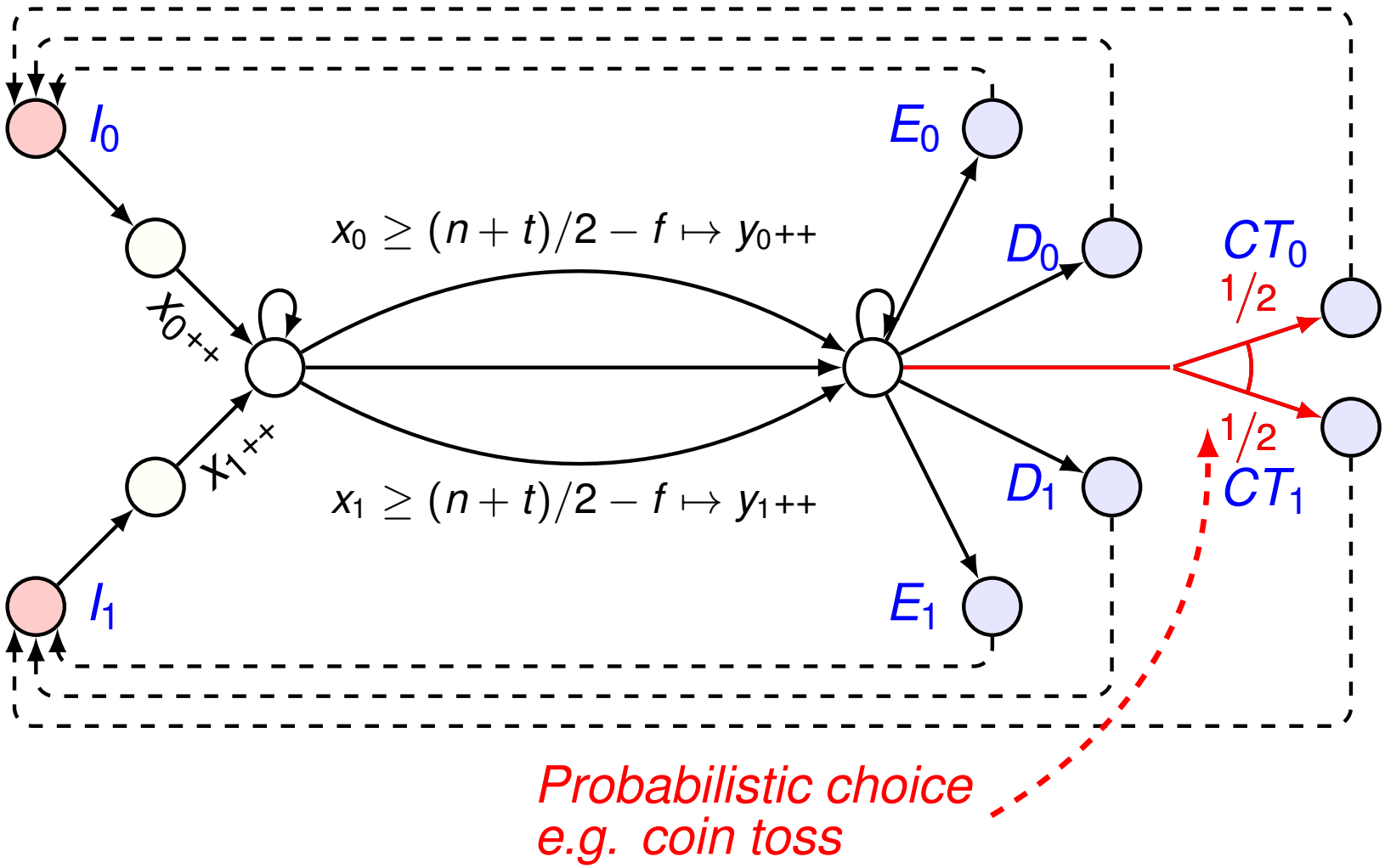
```
bool v := input_value({0, 1});
int r := 1;
while (true) do
 send (R,r,v) to all;
 wait for n - t messages (R,r,*);

 if received (n + t) / 2 messages (R,r,w)
 then send (P,r,w,D) to all;
 else send (P,r,?) to all;
 wait for n - t messages (P,r,*);

 if received at least t + 1
    messages (P,r,w,D) then {
  v := w;                    /* enough support -> update estimate */
  if received at least (n + t) / 2
   messages (P,r,w,D)
  then decide w;                /* strong majority -> decide */
 } else v := random({0,1});     /* unclear -> coin toss      */
 r := r + 1;
od
```

[Ben-Or, PODC 1983]

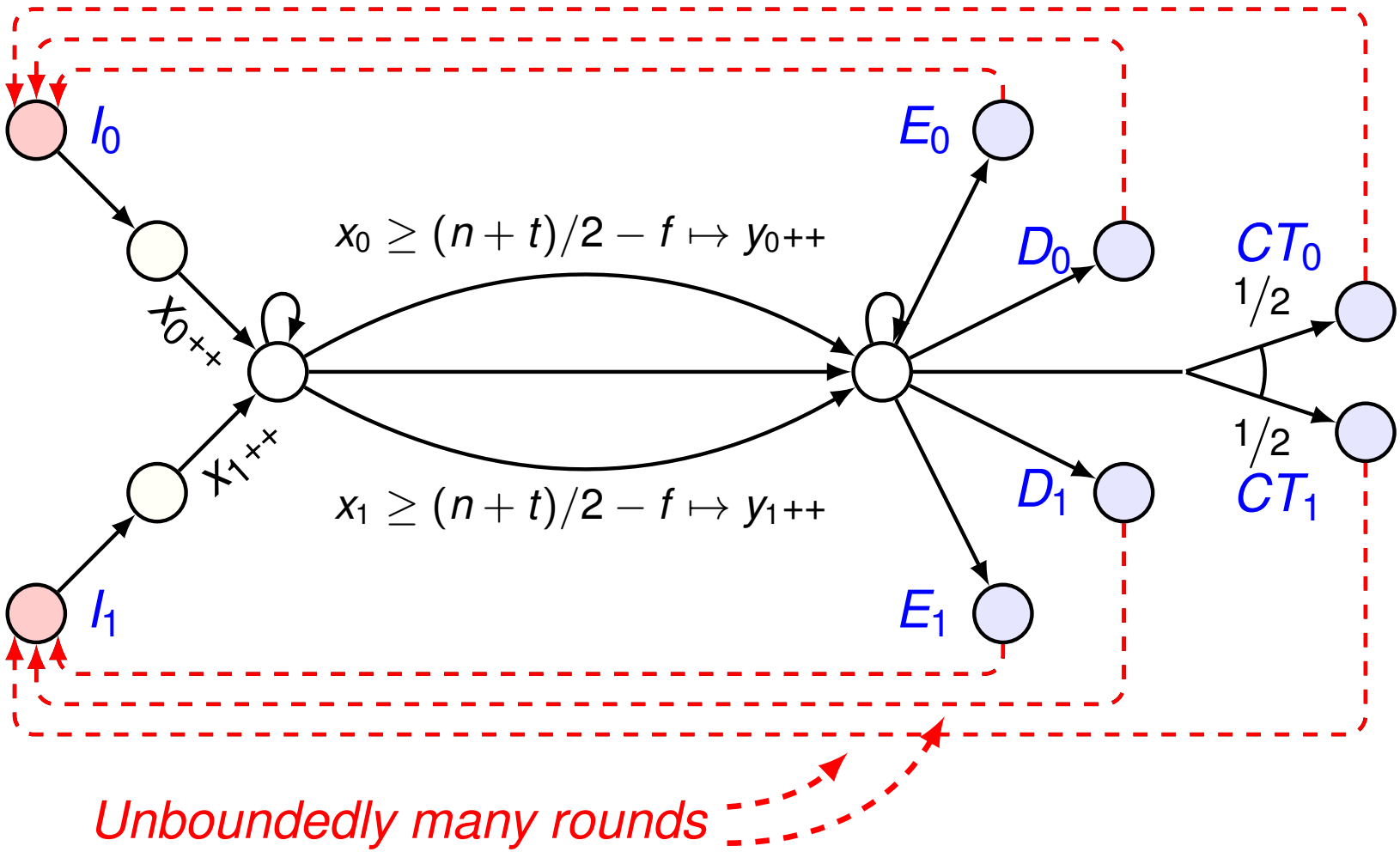
Probabilistic Threshold Automata (PTA)



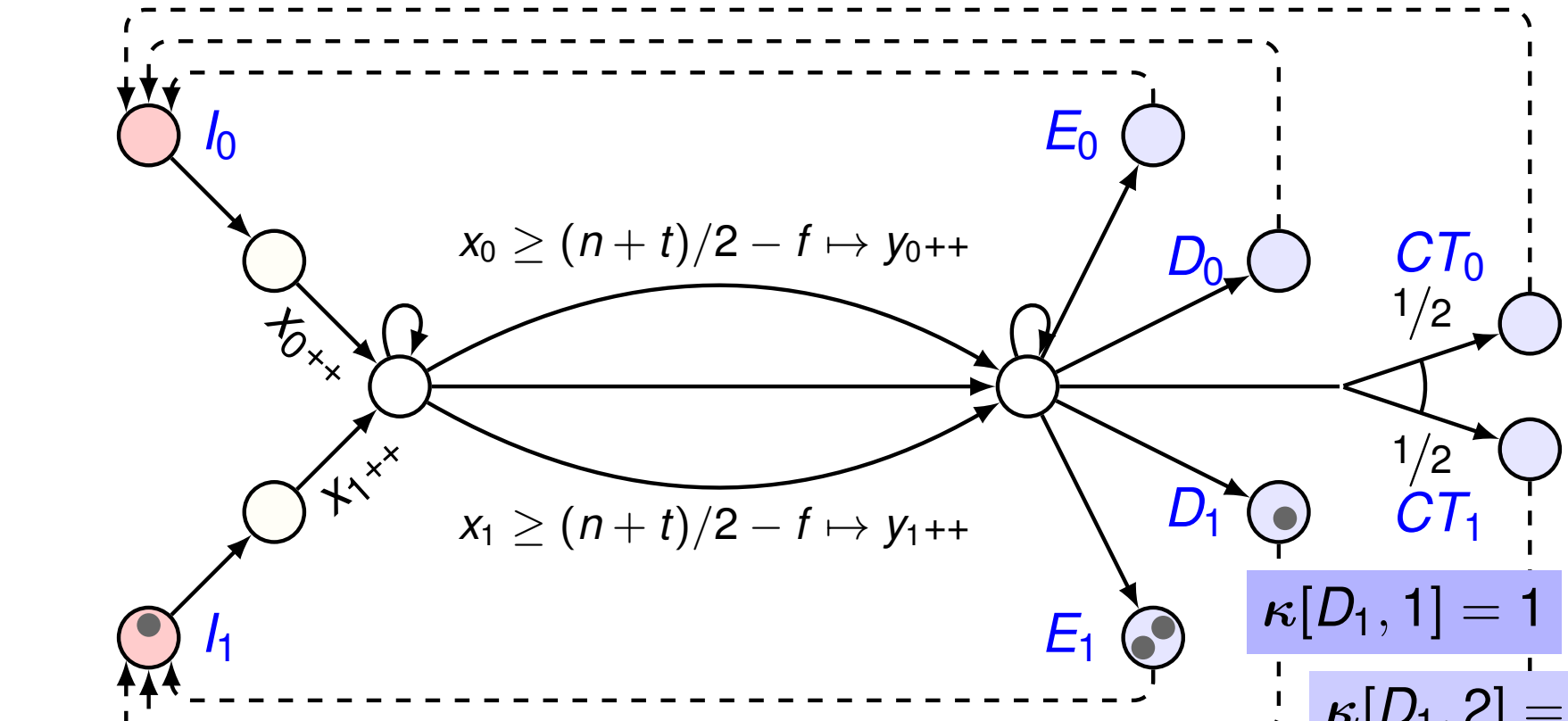
Probabilistic Threshold Automata (PTA)



Probabilistic Threshold Automata (PTA)



Probabilistic Counter System for a PTA



$\kappa[D_1, 1] = 1$

$\kappa[D_1, 2] = 0$

$\kappa[D_1, 3] = 0$

$\kappa[D_1, 4] = 0$

...

$\kappa[l_1, 1] = 1$

$\kappa[l_1, 2] = 0$

$\kappa[l_1, 3] = 0$

...

how many processes are in every location ***for every round***

$\kappa[E_1, 1] = 2$

$\kappa[E_1, 2] = 0$

$\kappa[E_1, 3] = 0$

...

Randomized consensus properties

Agreement : No two correct processes decide differently
(no matter in which two rounds they are)

Validity : If all correct processes have initial value v (in the 1st round)
then no process should decide $1 - v$ (in any other round)

Almost Sure Termination : Under every round-rigid adversary,
with probability 1 every correct process eventually decides

Randomized consensus properties

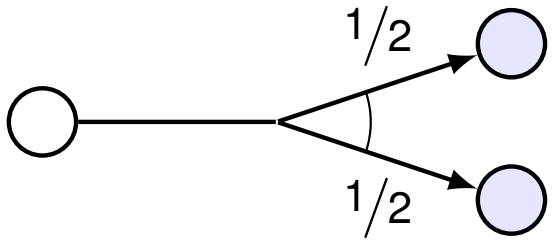
Agreement : No two correct processes decide differently
(no matter in which two rounds they are)

Validity : If all correct processes have initial value v (in the 1st round)
then no process should decide $1 - v$ (in any other round)

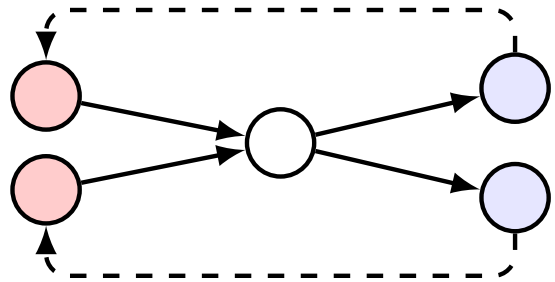
Almost Sure Termination : Under every round-rigid adversary,
with probability 1 every correct process eventually decides

Challenges

Probabilistic choice



Unboundedly many rounds



Specs with multiple rounds

$$\forall k,\ \forall k'.\ \mathbf{A}\ \varphi[k, k']$$

Specs with probability 1

$$\mathbb{P}_{\mathsf{s}}\big(\psi[k]\big) = 1$$

The Key Idea

Elimination of these brings us to the previous **non-randomized** setting

Two types of specifications

Specs with multiple rounds

$$\forall k,\ \forall k'.\ \mathbf{A}\ \varphi[k, k']$$

Specs with probability 1

$$\mathbb{P}_{\mathsf{s}}\big(\psi[k]\big) = 1$$

Two types of specifications

Specs with multiple rounds

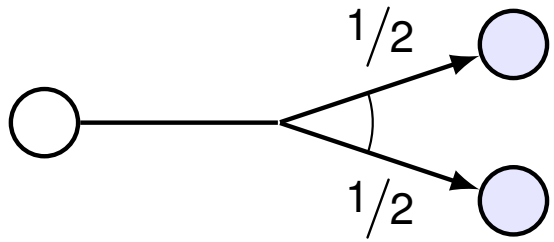
$$\forall k,\ \forall k'.\ \mathbf{A}\ \varphi[k, k']$$

Specs with probability 1

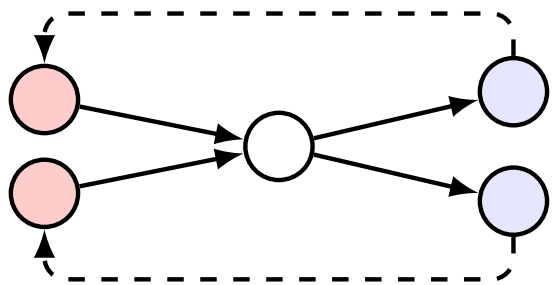
$$\mathbb{P}_{\mathsf{s}}\big(\psi[k]\big) = 1$$

Non-probabilistic properties

Probabilistic choice



Unboundedly many rounds

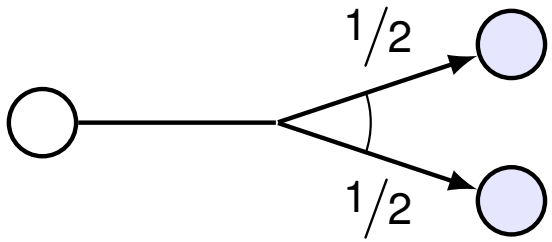


Specs with multiple rounds

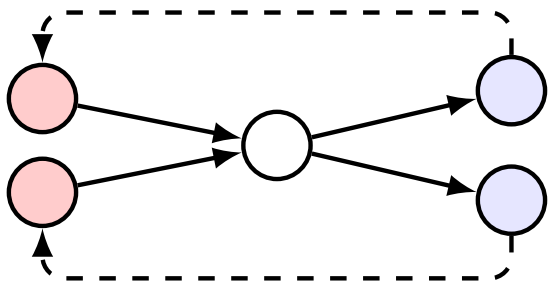
$$\forall k,\ \forall k'.\ \mathbf{A}\ \varphi[k, k']$$

Non-probabilistic properties

Probabilistic choice



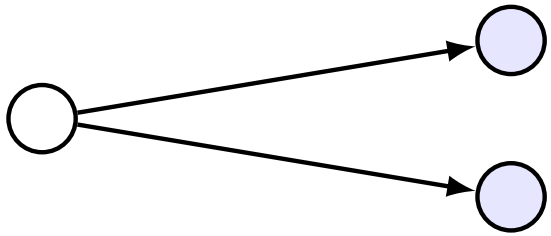
Unboundedly many rounds



Specs with multiple rounds

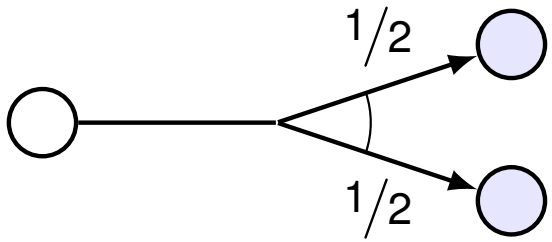
$$\forall k,\ \forall k'.\ \mathbf{A}\ \varphi[k, k']$$

Non-determinism

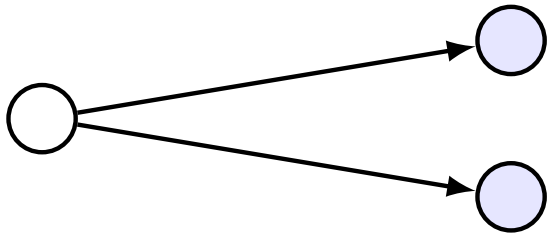


Non-probabilistic properties

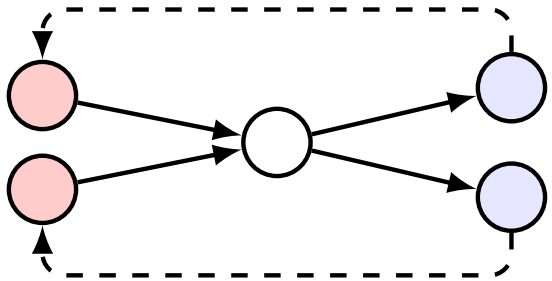
Probabilistic choice



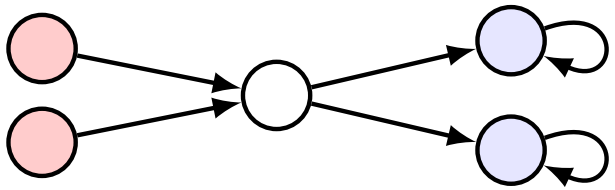
Non-determinism



Unboundedly many rounds



One-round system

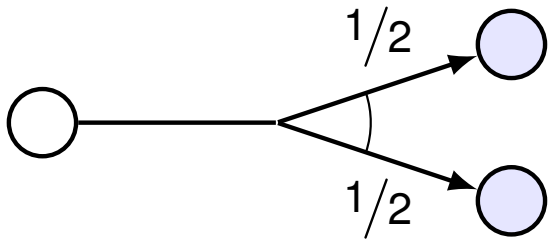


Specs with multiple rounds

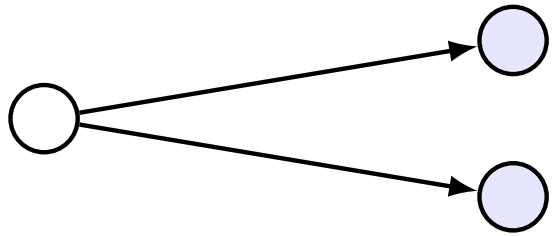
$$\forall k,\ \forall k'.\ \mathbf{A}\ \varphi[k, k']$$

Non-probabilistic properties

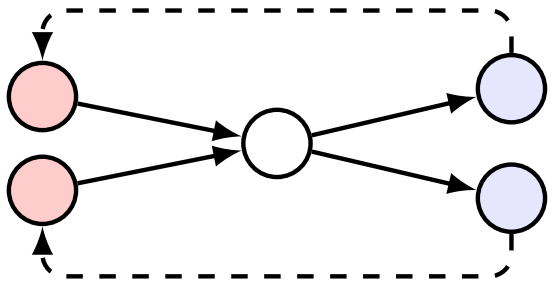
Probabilistic choice



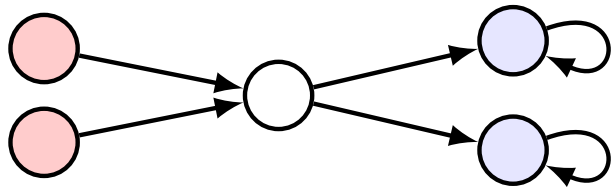
Non-determinism



Unboundedly many rounds



One-round system



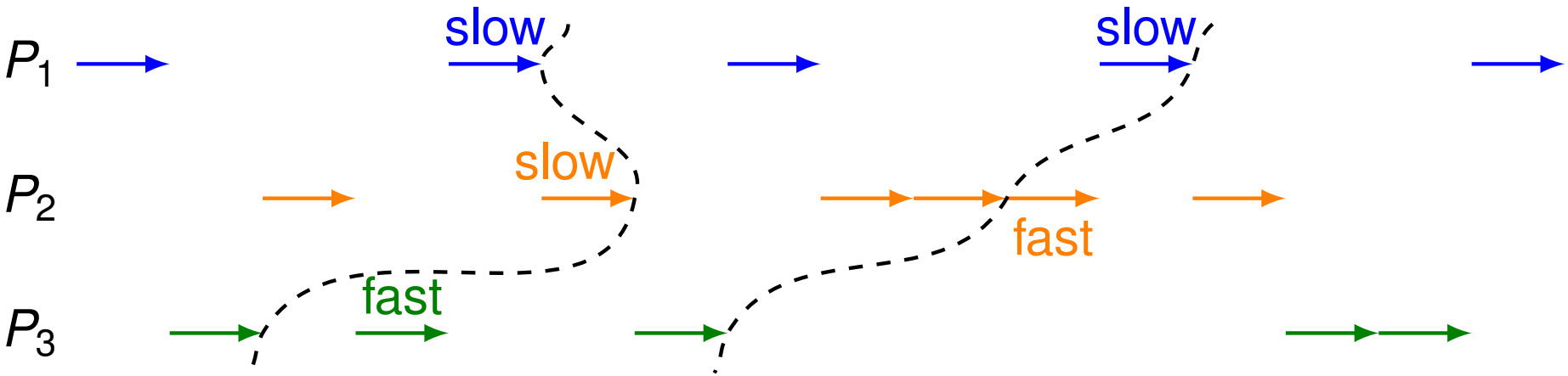
Specs with multiple rounds

$$\forall k,\ \forall k'.\ \mathbf{A}\ \varphi[k, k']$$

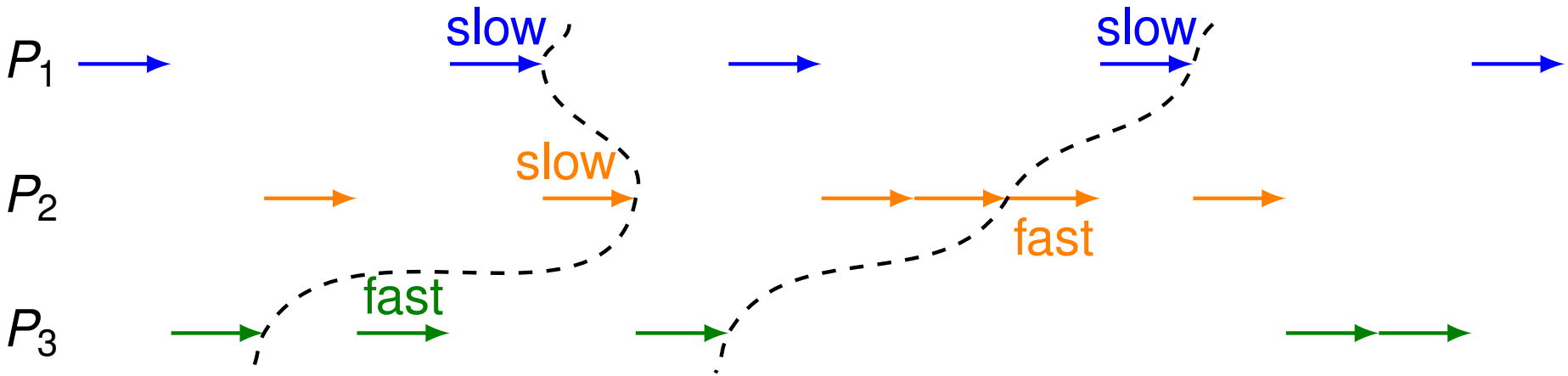
One-round specs

$$\forall k.\ \mathbf{A}\ \varphi'[k]$$

Reduction to one-round system

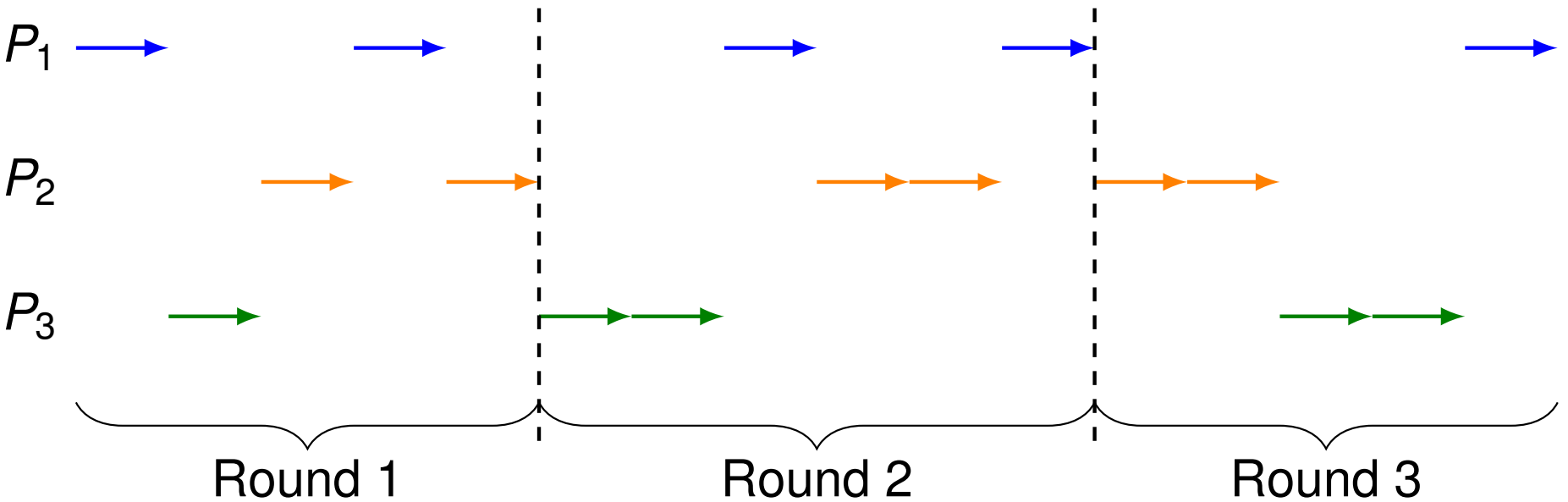


Reduction to one-round system



Solution:

reordering transitions allows us to analyze rounds in isolation



Reasoning about round boundaries

Original System:

$$P_1 \parallel P_2 \parallel \cdots \parallel P_n, \quad \text{with} \quad P_i = R_i^1 \;;\; R_i^2 \;;\; R_i^3 \;;\; \ldots$$

reduced to

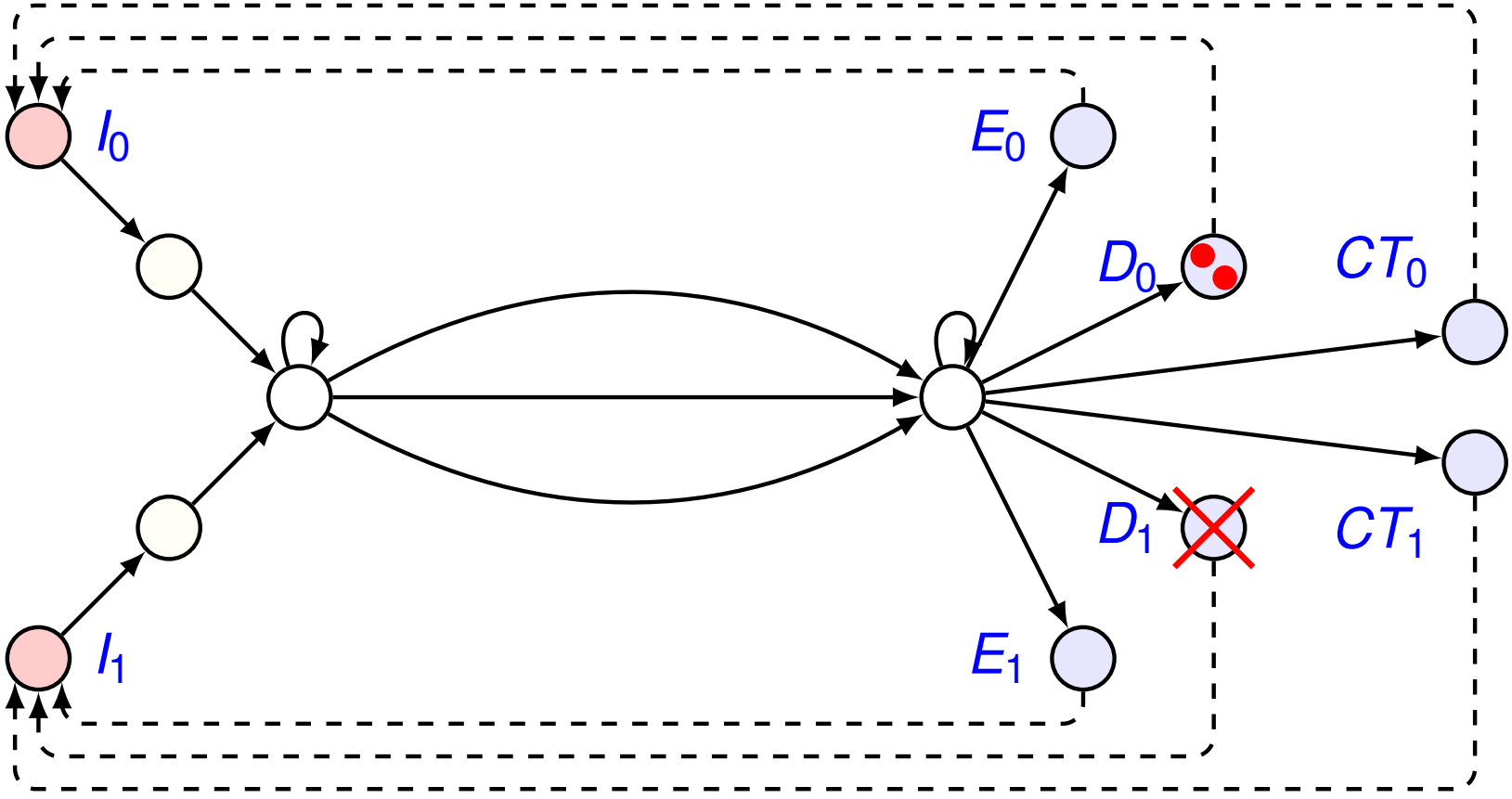
$$R_1^1 \parallel R_2^1 \parallel \cdots \parallel R_n^1 \;;\; R_1^2 \parallel R_2^2 \parallel \cdots \parallel R_n^2 \;;\; \ldots$$

$\Rightarrow$ Reason about round boundaries only!

$$\{init\} \; R_1^1 \parallel R_2^1 \parallel \cdots \parallel R_n^1 \; \{\phi_1\}; \; R_1^2 \parallel R_2^2 \parallel \cdots \parallel R_n^2 \; \{\phi_2\}; \; \ldots$$

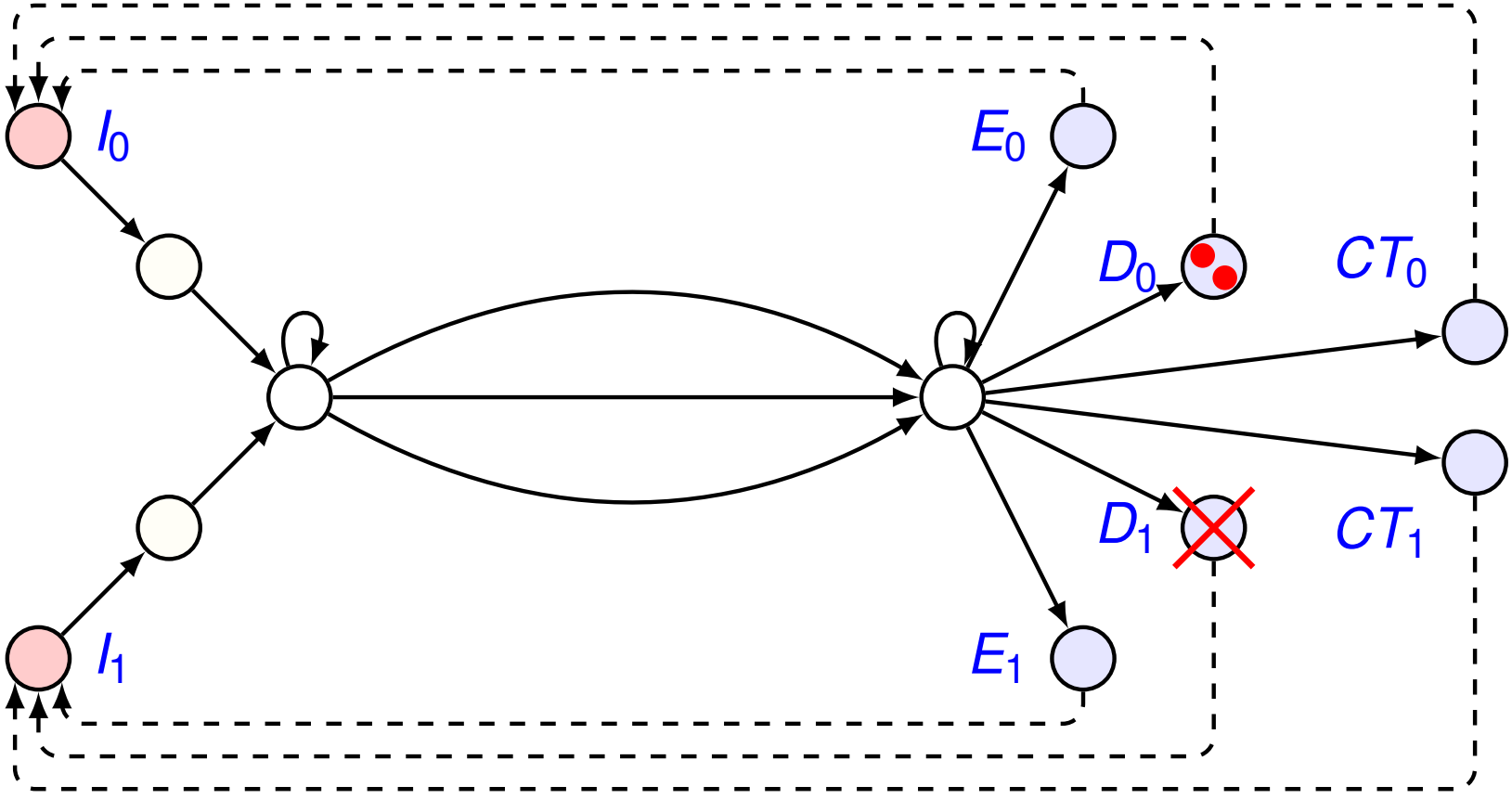
Reduction to one-round specs

Agreement: if **F** decision v in k then **G** no decision $1-v$ in k'



Reduction to one-round specs

Agreement: if **F** decision v in k then **G** no decision $1-v$ in k'

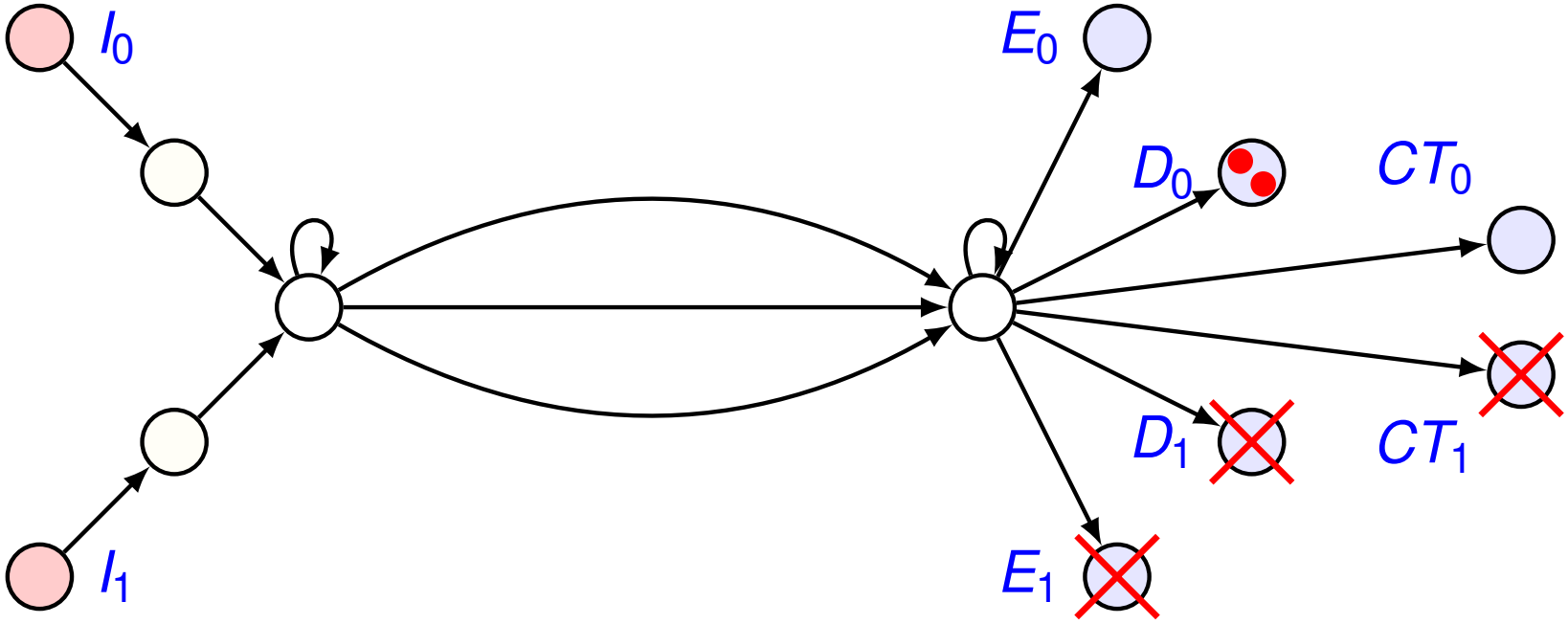


(A)∧(B) → Agreement

Both are one-round specs

Reduction to one-round specs

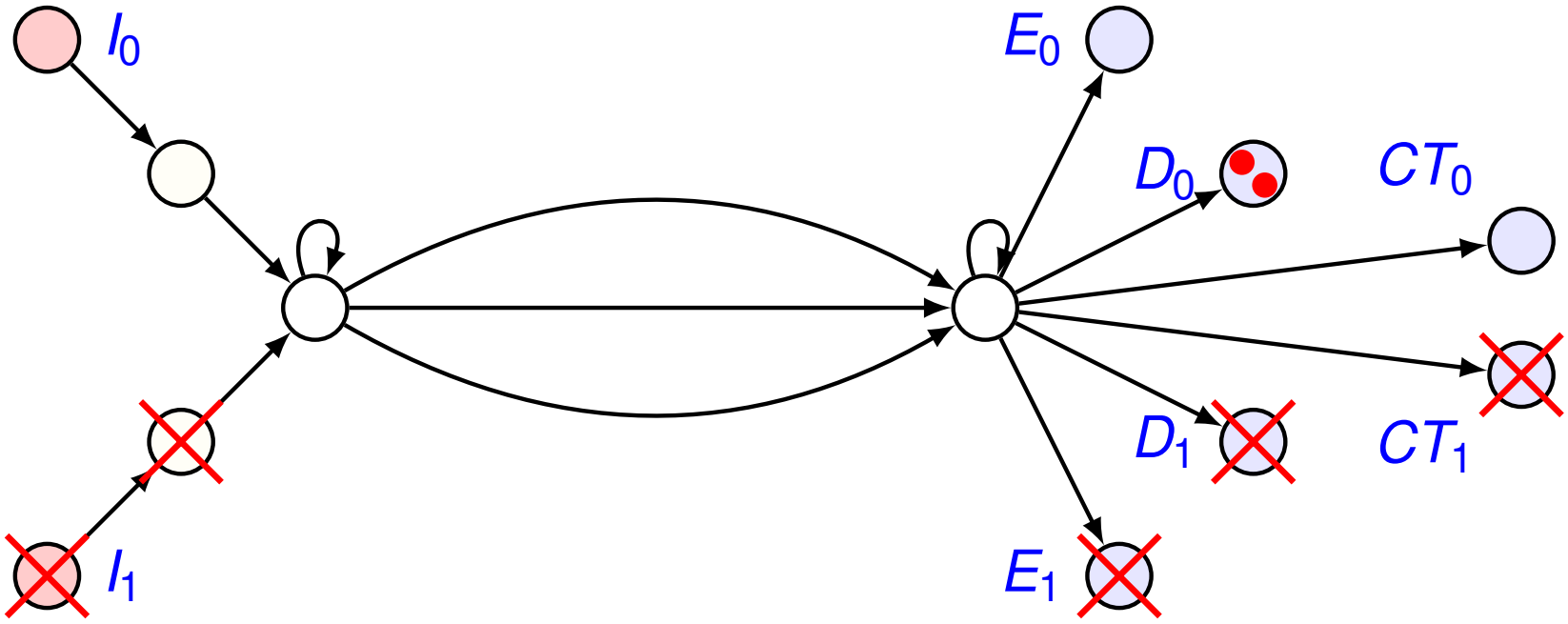
Agreement: if **F** decision v in k then **G** no decision $1-v$ in k'



(A) if **F** decision v in k then **G** empty final states with $1-v$ in k

Reduction to one-round specs

Agreement: if **F** decision v in k then **G** no decision $1-v$ in k'



(B) if **G** empty initial with $1-v$ in k then **G** empty final with $1-v$ in k

Two types of specifications

Specs with multiple rounds

$$\forall k,\ \forall k'.\ \mathbf{A}\ \varphi[k, k']$$

Specs with probability 1

$$\mathbb{P}_{\mathsf{s}}\big(\psi[k]\big) = 1$$

Two types of specifications

Specs with multiple rounds

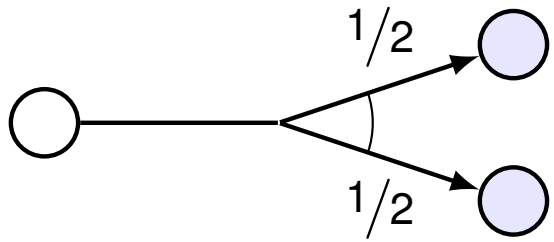
$$\forall k,\ \forall k'.\ \mathbf{A}\ \varphi[k, k']$$

Specs with probability 1

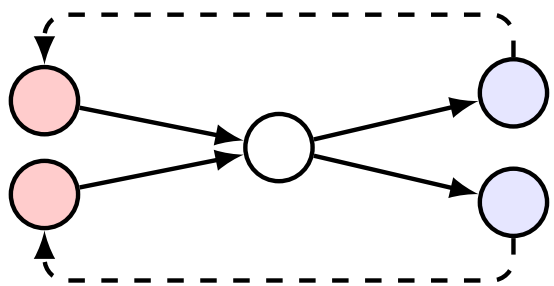
$$\mathbb{P}_{\mathsf{s}}\big(\psi[k]\big) = 1$$

Probabilistic properties

Probabilistic choice



Unboundedly many rounds

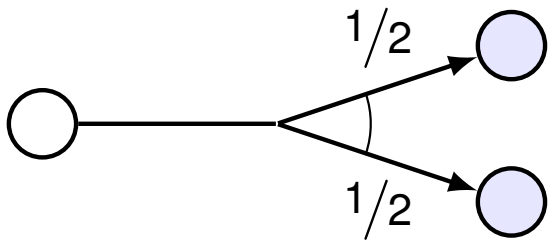


Almost sure termination

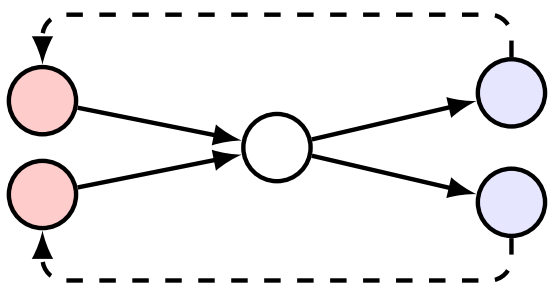
$$\mathbb{P}_s\big(\psi[k]\big) = 1$$

Probabilistic properties

Probabilistic choice



Unboundedly many rounds



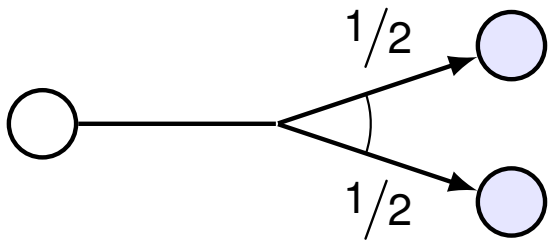
Almost sure termination

$$\mathbb{P}_s\big(\psi[k]\big) = 1$$

Separately reason about probabilities

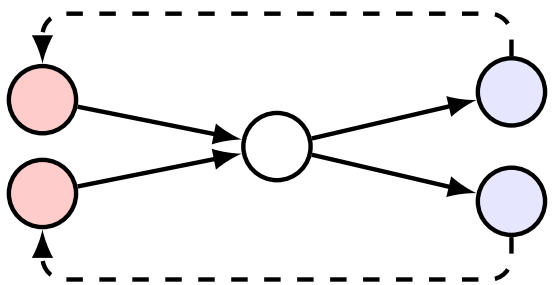
Probabilistic properties

Probabilistic choice



Separately reason about probabilities

Unboundedly many rounds



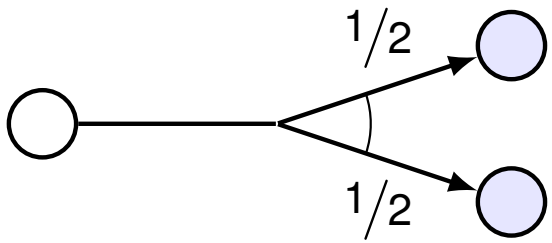
Restriction to round-rigid adversaries

Almost sure termination

$$\mathbb{P}_s\big(\psi[k]\big) = 1$$

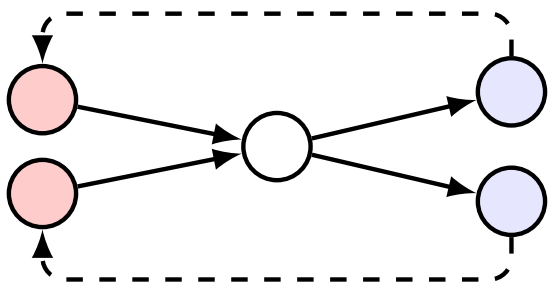
Probabilistic properties

Probabilistic choice



Separately reason about probabilities

Unboundedly many rounds



Restriction to round-rigid adversaries

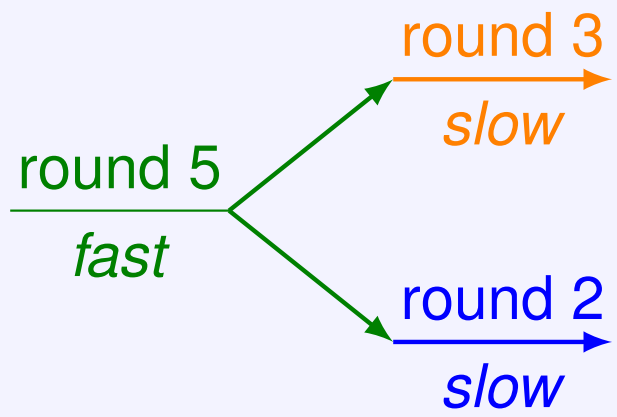
Almost sure termination

$$\mathbb{P}_s\big(\psi[k]\big) = 1$$

Transformation to familiar formulas

Reduction to one-round system

Adversary : prefix $\mapsto$ next transition or branching for coin toss



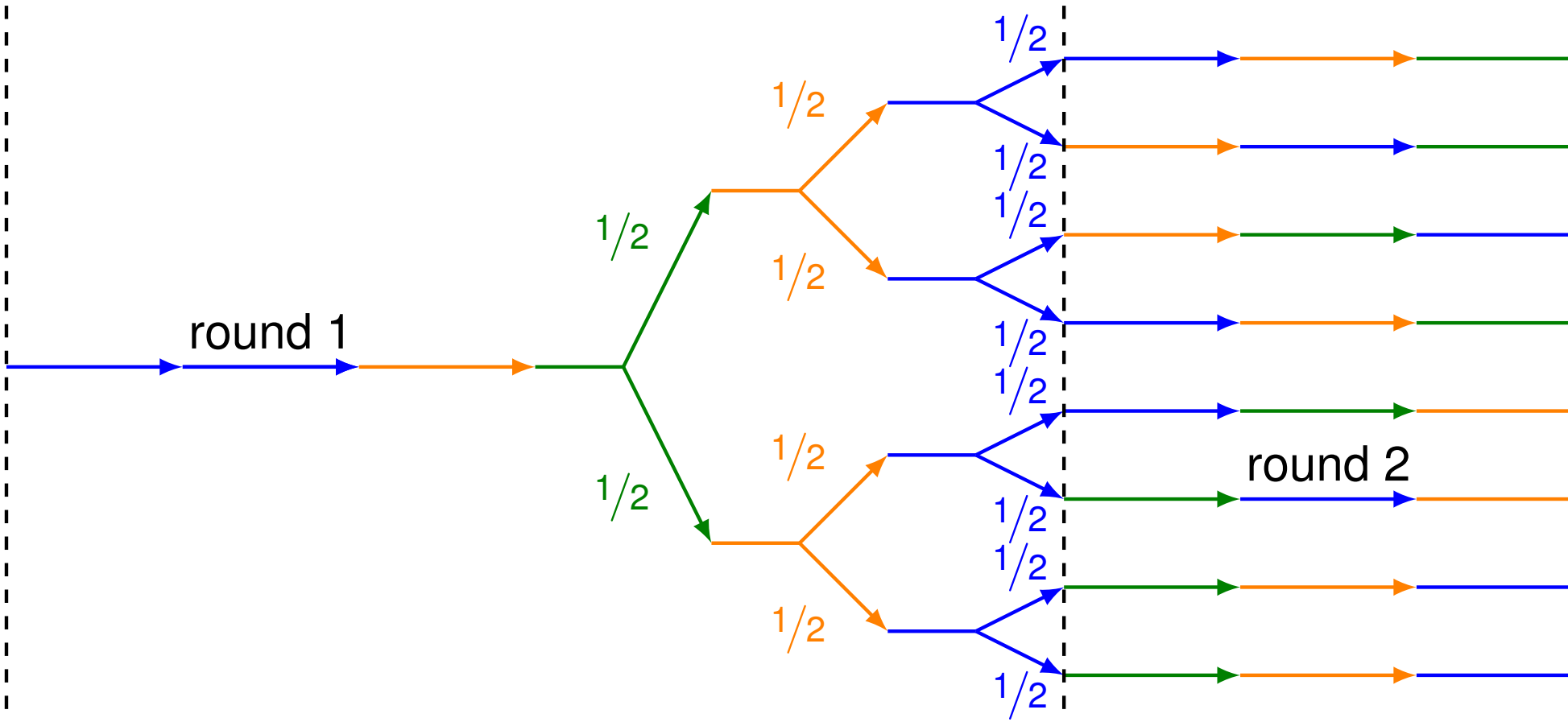
How can we now swap “fast” and “slow”?

Our swapping trick does not work for arbitrary adversaries!

Restriction to round-rigid adversaries

Round-rigid adversary

- respects round order
- branching at the end of each round



Reduction to one-round specs

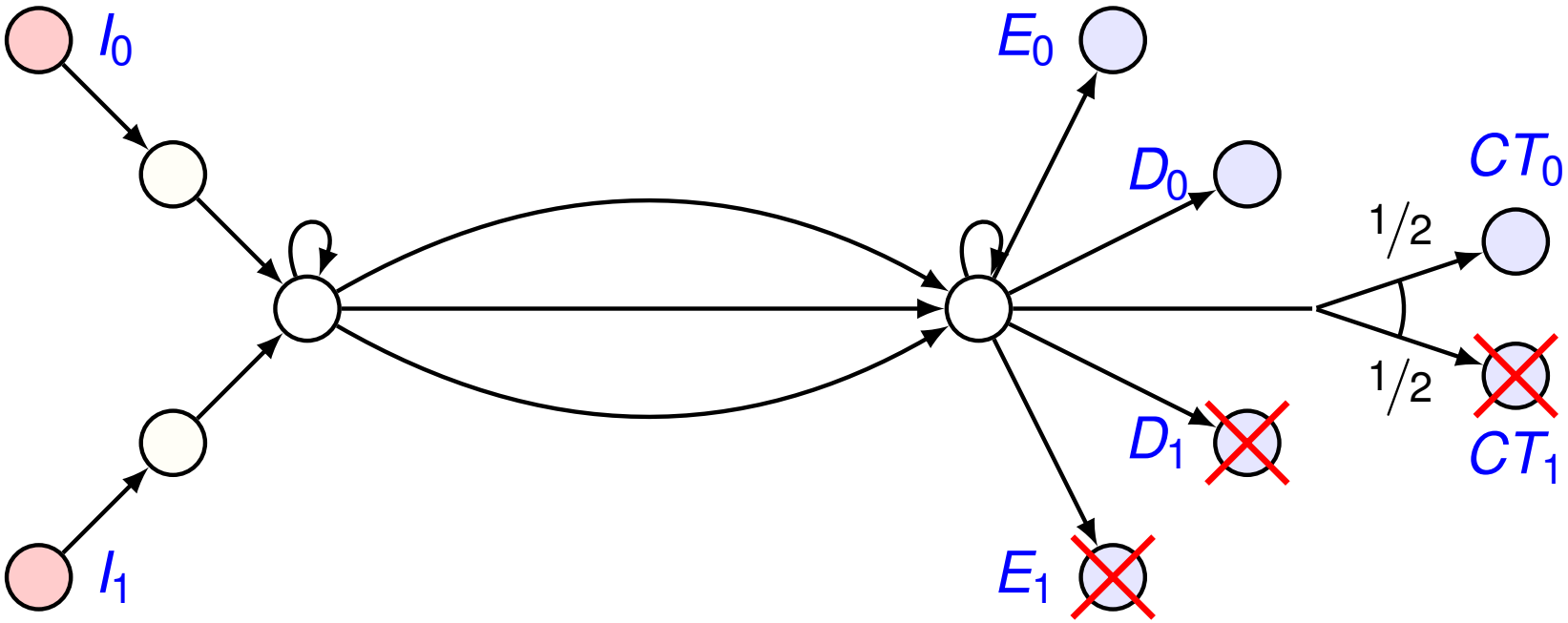
Almost sure termination: Under every round-rigid adversary, with probability 1 every correct process eventually decides.

Simplifying the problem:

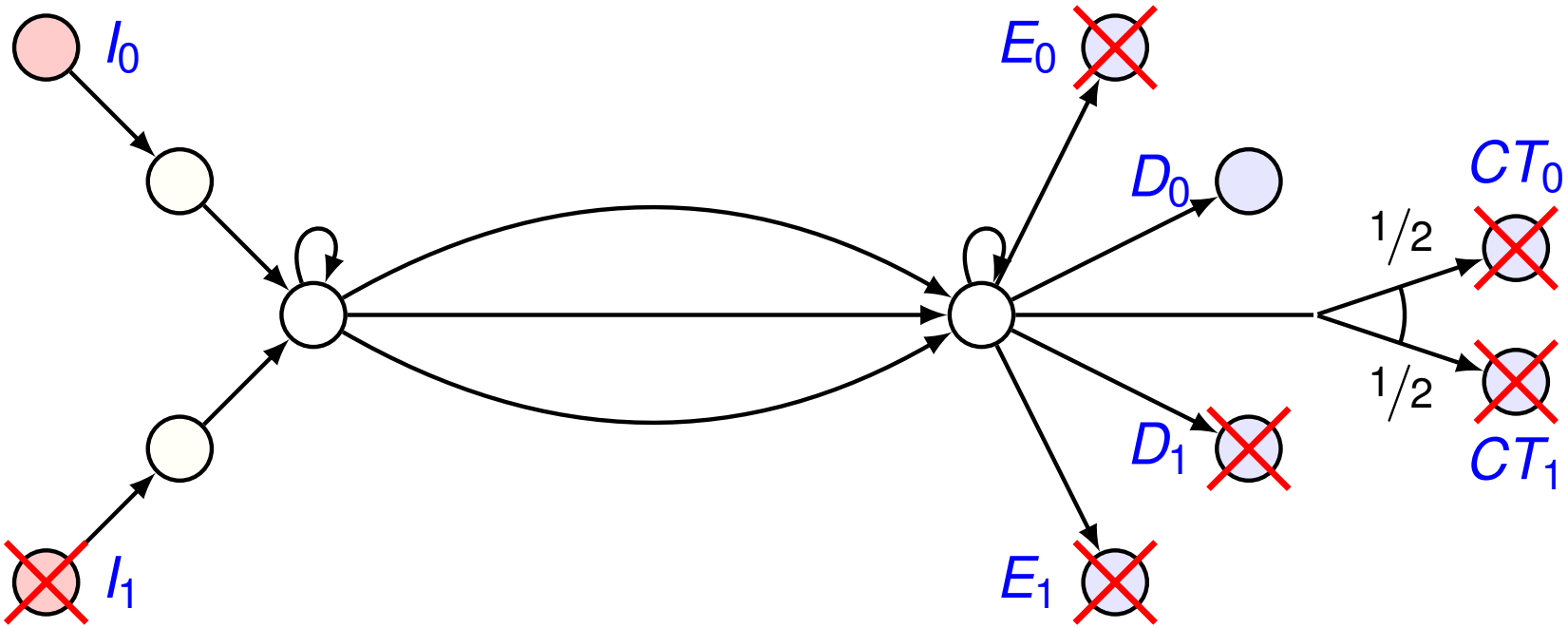
1. Define a “lucky” situation for a round k
2. **Check**: Lucky in k means everyone decides in $k + 1$
3. **Check**: Lucky happens with non-zero probability

This implies Almost sure termination!

1. Defining a “lucky” situation for a round

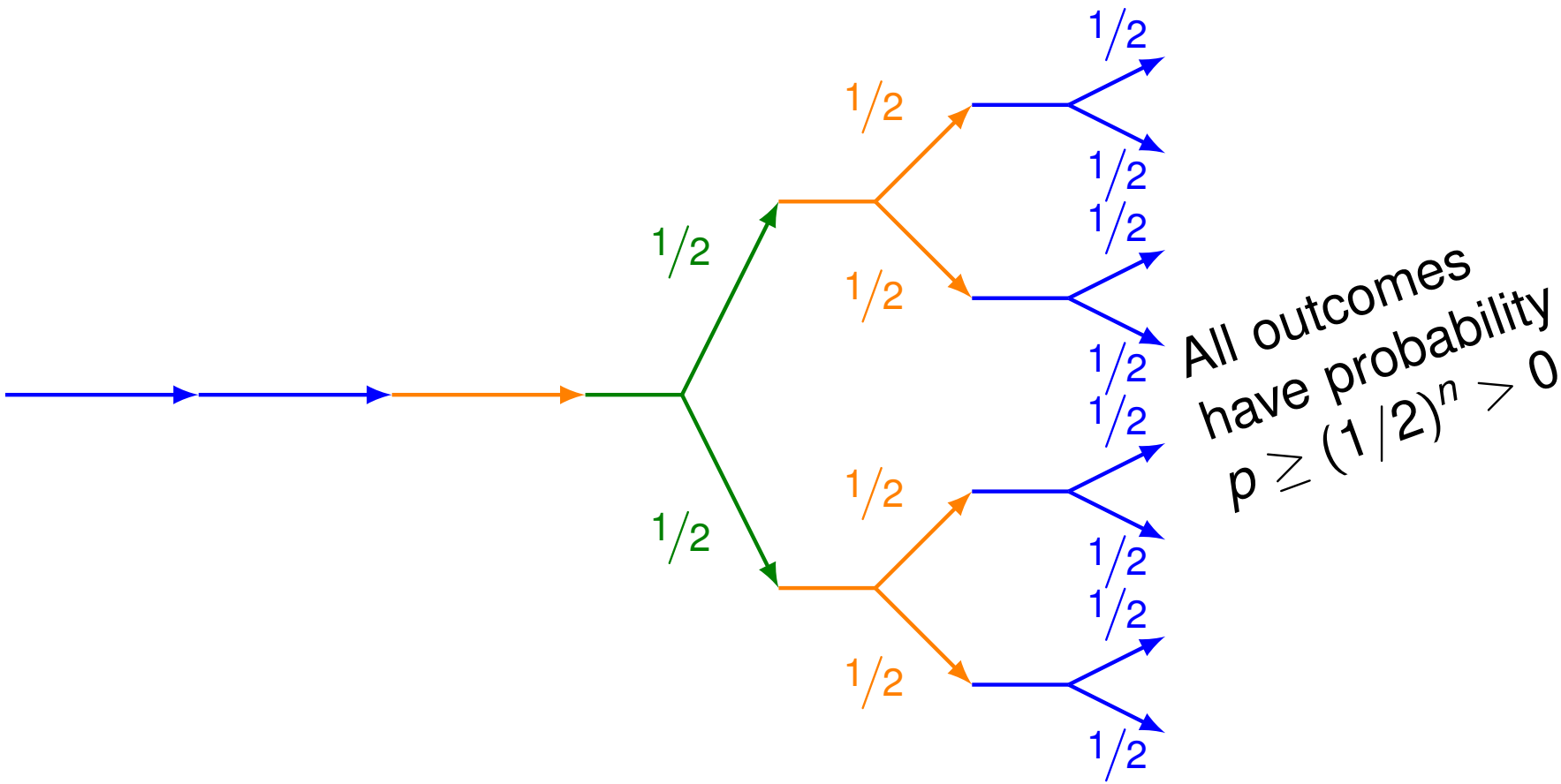


2. Lucky in k means termination in $k+1$

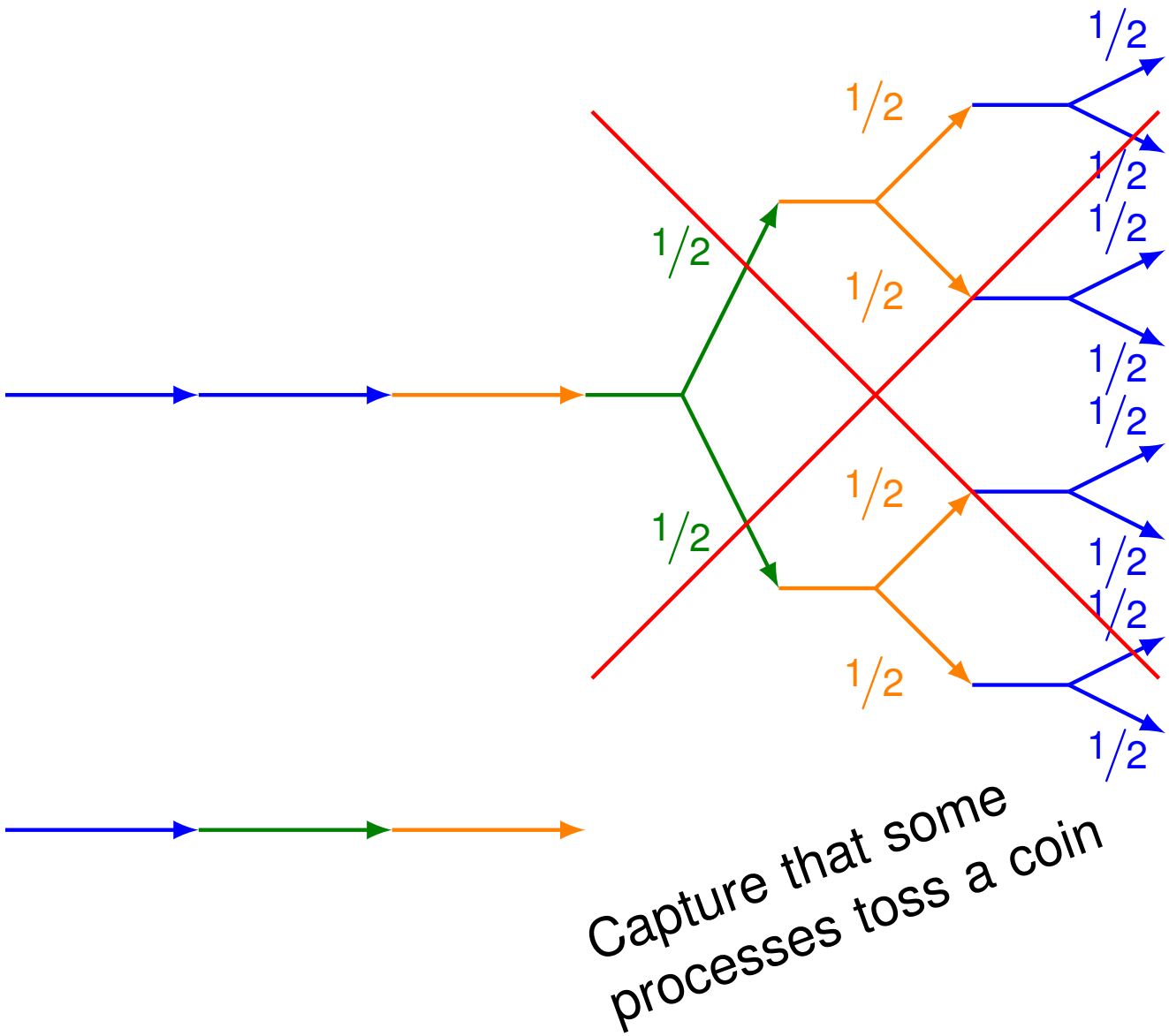


if **G** empty initial with $1-v$ in k then **F** all decide v in k

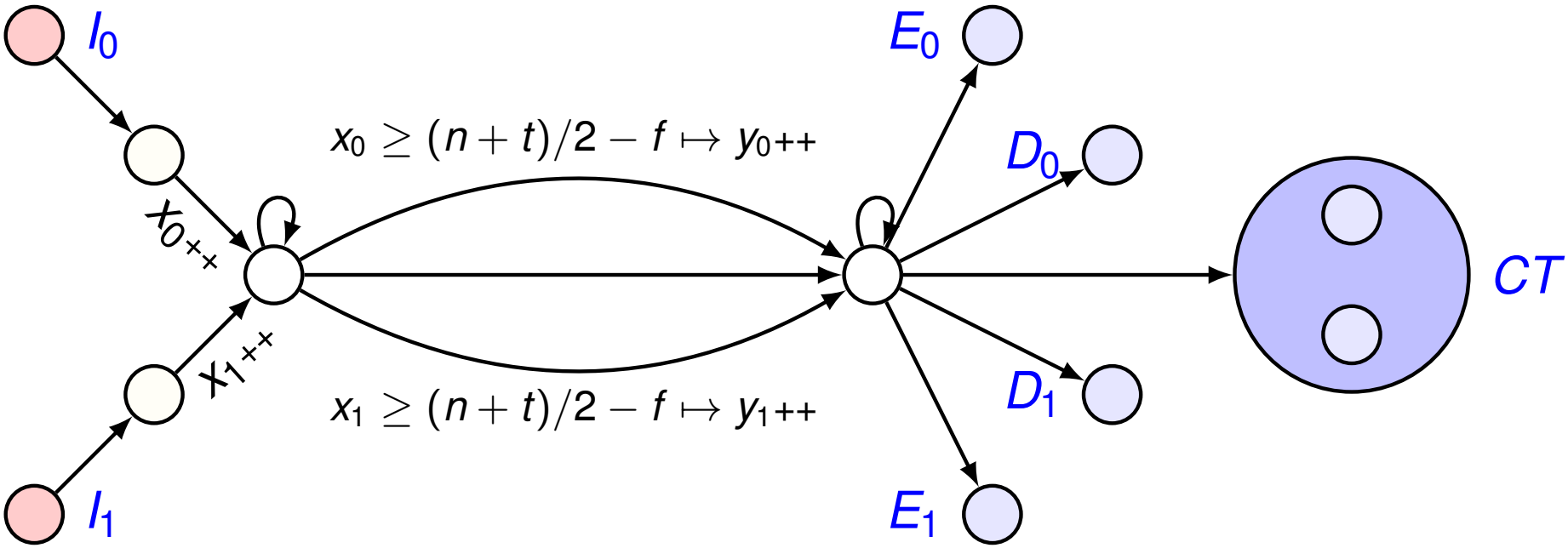
3. Lucky happens with non-zero probability



3. Lucky happens with non-zero probability



Abstracting Coin-Toss Outcomes



A **non-probabilistic** threshold automaton

Proving Almost sure termination

	Meaning	Probability
lucky in k	all decide in $k+1$	$p > 0$
"unlucky" in k	NOT all decide in $k+1$	$1-p < 1$
"constantly unlucky"	non-termination	$\lim_{k\to\infty}(1-p)^k = 0$
	termination	1

Proving Almost sure termination

	Meaning	Probability
lucky in k	all decide in $k+1$	$p > 0$
"unlucky" in k	NOT all decide in $k+1$	$1-p < 1$
"constantly unlucky"	non-termination	$\lim_{k\to\infty}(1-p)^k = 0$
	termination	1

Proving Almost sure termination

	Meaning	Probability
lucky in k	all decide in $k+1$	$p > 0$
"unlucky" in k	NOT all decide in $k+1$	$1-p < 1$
"constantly unlucky"	non-termination	$\lim_{k\to\infty}(1-p)^k = 0$
	termination	1

Proving Almost sure termination

	Meaning	Probability
lucky in k	all decide in $k+1$	$p > 0$
"unlucky" in k	NOT all decide in $k+1$	$1-p < 1$
"constantly unlucky"	non-termination	$\lim_{k\to\infty}(1-p)^k = 0$
	termination	1

Two types of specifications

Specs with multiple rounds

$$\forall k,\ \forall k'.\ \mathbf{A}\ \varphi[k, k']$$

Specs with probability 1

$$\mathbb{P}_{\mathsf{s}}\big(\psi[k]\big) = 1$$

Experimental evaluation

We have verified 6 parameterized randomized consensus algorithms with several one-round safety and liveness properties:

Algorithm	Verification per property
- Ben-Or's Byzantine random. consensus	$\leq$ 1 sec
- Ben-Or's crash random. consensus	$\leq$ 1 sec
- Ben-Or's clean crash random. consensus	$\leq$ 1 sec
- Bracha's randomized consensus	$\leq$ 1 sec
- Raynal's k-set agreement	3–40 sec
- Song's and van Renesse's BOSCO	3 hours on a cluster

[forsyte.at/software/bymc]

Conclusions

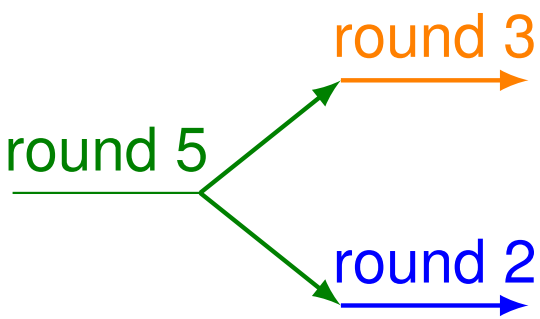
Parameterized verification of **randomized consensus** algorithms

Main contributions:

- Round reduction for **all** non-deterministic systems of threshold automata
- Compositional reasoning for non-probabilistic **consensus** specifications
- Probabilistic reasoning for almost-sure termination in **consensus**

Future work

- more general adversaries



Conclusions

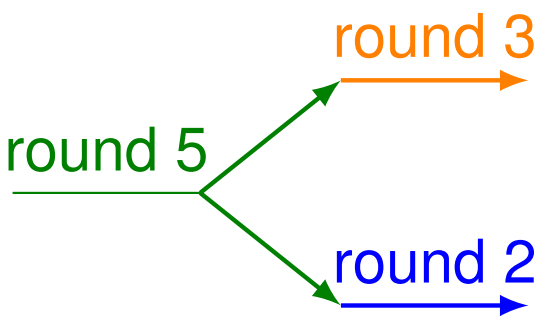
Parameterized verification of **randomized consensus** algorithms

Main contributions:

- Round reduction for **all** non-deterministic systems of threshold automata
- Compositional reasoning for non-probabilistic **consensus** specifications
- Probabilistic reasoning for almost-sure termination in **consensus**

Future work

- more general adversaries



Thank you!