Automated Verification of Randomised Distributed Algorithms

Directed by Nathalie Bertrand[1]

Bastien Thomas

[1] SUMO team: Inria and Irisa

Table of Contents

Table of Contents

Randomised Distributed Algorithms

Framework

Asynchronicity Several processes advance at their own pace, messages can be delayed and arrive out of order

Fault-Tolerance Processes may fail, either by *crashing* or by executing any code (*bysantine* fault)

Randomisation Code may include random choices (coin toss)

Goals Providing tools for automatically verifying algorithm for *any* number of processes

Motivation: Consensus Problem

Situation

- 3 processes p_1, p_2, p_3.
- Each process has an initial value $v_i \in \{0, 1\}$.
- In the end all processes must agree on a common value $d \in \{v_1, v_2, v_3\}$.

No crash Simple Protocol

- Send v_i
- Wait until received two other values
- if $0 \in \{v_1, v_2, v_3\}$ then decide 0 else decide 1

One crash Naive protocol attempt

- Send v_i
- Wait until received *one* other value v_j
- if $0 \in \{v_i, v_j\}$ then decide 0 else decide 1

Problem: if $v_1, v_2, v_3 = 1, 1, 0$, if p_1 received v_2 and p_2 received v_3, then p_1 decides 1 and p_2 decides 0

Impossibility Result and Probabilistic Relaxation

With n processes and t allowed crashes.

Theorem [M. Fisher et al. 1982]

When $n \geq 3$ and with one possible crash, every consensus-solving algorithm have infinite executions.

Relaxation Termination with probability 1

Ben-Ors' Algorithm [M.Ben-Or. 1982]

Ben-Or's algorithm solves the consensus and terminate with probability 1 for $t < n/2$.

Proofs of such algorithms are complex and error-prone [Saias 1992]
Currently, automated methods only work for a fixed n

Table of Contents

Threshold Automata

Syntax

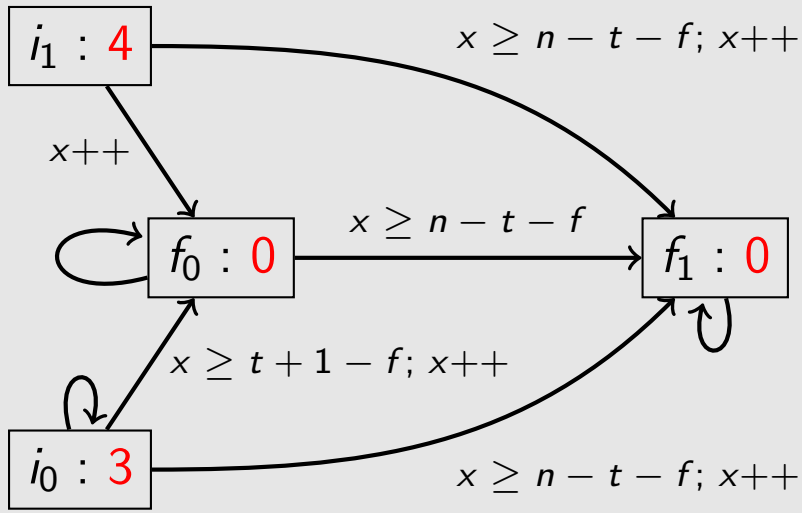
- L: *locations*
- X: *shared variables*
- Π: *parameters*
- $E \subset L^2 \times \text{Formulas} \times \mathbb{N}^X$: *edges*

Semantic

Configuration: $c \in \mathbb{N}^{L \cup X \cup \Pi}$

$e = (s, s', \varphi, u) \in E$, $e(c) = c'$ if:

- $c \models \varphi$
- $c(s) > 0$
- $c'_L = c_L - \mathbb{1}_{\{s\}} + \mathbb{1}_{\{s'\}}$
- $c'_X = c_X + u$
- $c'_\Pi = c_\Pi$



Parameter $n = 7,\ t = 2,\ f = 1$

Variables $x = 0$

Threshold Automata

Syntax

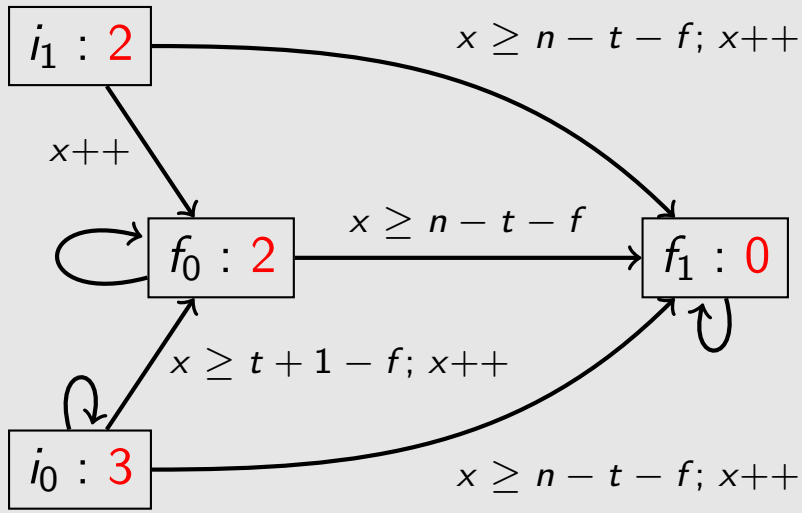
- L: *locations*
- X: *shared variables*
- Π: *parameters*
- $E \subset L^2 \times \text{Formulas} \times \mathbb{N}^X$: *edges*

Semantic

Configuration: $c \in \mathbb{N}^{L \cup X \cup \Pi}$

$e = (s, s', \varphi, u) \in E$, $e(c) = c'$ if:

- $c \models \varphi$
- $c(s) > 0$
- $c'_L = c_L - \mathbb{1}_{\{s\}} + \mathbb{1}_{\{s'\}}$
- $c'_X = c_X + u$
- $c'_\Pi = c_\Pi$



Parameter $n = 7,\ t = 2,\ f = 1$

Variables $x = 2$

Threshold Automata

Syntax

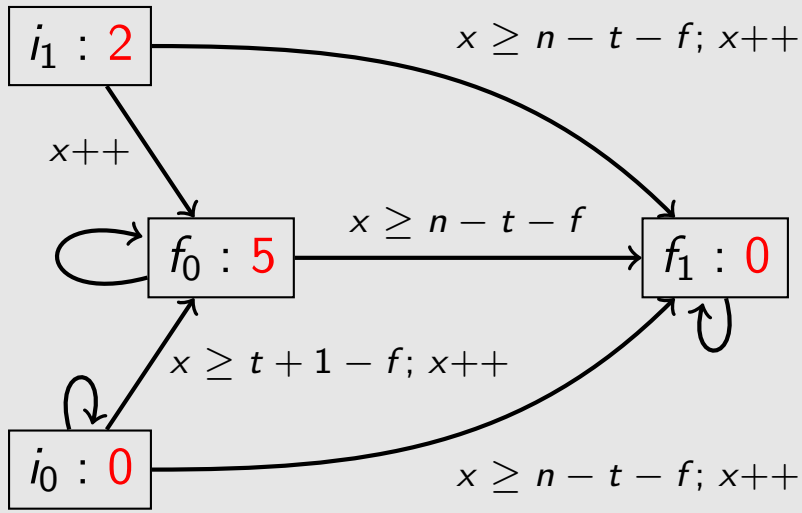
- L: *locations*
- X: *shared variables*
- Π: *parameters*
- $E \subset L^2 \times \text{Formulas} \times \mathbb{N}^X$: *edges*

Semantic

Configuration: $c \in \mathbb{N}^{L \cup X \cup \Pi}$

$e = (s, s', \varphi, u) \in E$, $e(c) = c'$ if:

- $c \models \varphi$
- $c(s) > 0$
- $c'_L = c_L - \mathbb{1}_{\{s\}} + \mathbb{1}_{\{s'\}}$
- $c'_X = c_X + u$
- $c'_\Pi = c_\Pi$



Parameter $n = 7,\ t = 2,\ f = 1$

Variables $x = 5$

Threshold Automata

Syntax

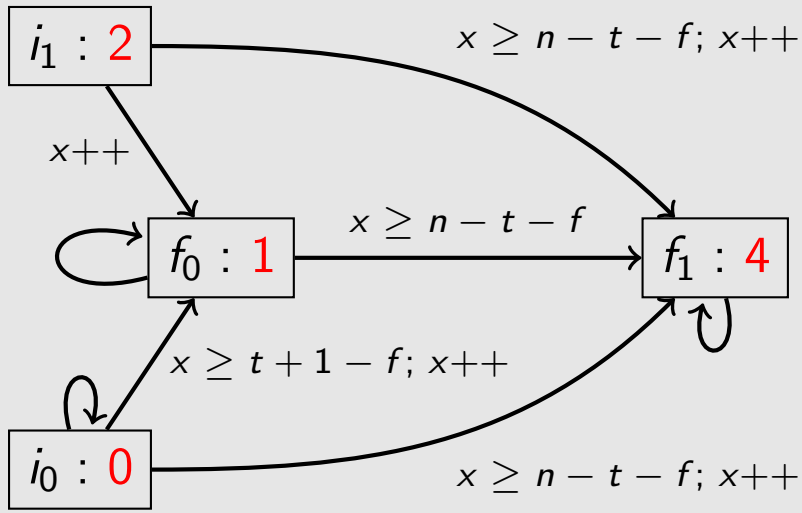
- L: *locations*
- X: *shared variables*
- Π: *parameters*
- $E \subset L^2 \times \text{Formulas} \times \mathbb{N}^X$: *edges*

Semantic

Configuration: $c \in \mathbb{N}^{L \cup X \cup \Pi}$

$e = (s, s', \varphi, u) \in E$, $e(c) = c'$ if:

- $c \models \varphi$
- $c(s) > 0$
- $c'_L = c_L - \mathbb{1}_{\{s\}} + \mathbb{1}_{\{s'\}}$
- $c'_X = c_X + u$
- $c'_\Pi = c_\Pi$



Parameter $n = 7,\ t = 2,\ f = 1$

Variables $x = 5$

LTL logic and Threshold Automata

Atomic Propositions

- Guards of the automaton e.g. $x \geq t + 1 - f$
- $c(l) = 0$

For a sequence $c_0 c_1 \ldots$ of configurations

- $c_0 c_1 \ldots \models \textit{proposition}$ if $c_0 \models \textit{proposition}$
- $c_0 c_1 \ldots \models \Box\varphi$ if for all $i \in \mathbb{N}$, $c_i c_{i+1} \ldots \models \varphi$
- $c_0 c_1 \ldots \models \Diamond\varphi$ if there exists $i \in \mathbb{N}$, $c_i c_{i+1} \ldots \models \varphi$

We will try to solve satisfiability (counter-example based approach)
If we add operators 'next' or 'until', *undecidable*

Safety of Threshold Automata

Theorem: Bounded Diameter [Konnov et al. 2017]

Hypothesis:

- Guard Formulas change status at most once
- No loop increase shared variables

Then there exists $d \in \mathbb{N}$ computable such that:
If $c \rightarrow^* c'$ then there exists $e_1 \dots e_m$, $k_1 \dots k_m$ with:
$c' = e_m^{k_m} \circ \cdots \circ e_1^{k_1}(c)$ and $m \leq d$.

Value d is independent of the parameters!
We can use SMT solvers to perform *complete* bounded model-checking.

Consequence

Satisfiability of reachability properties ($p_0 \wedge \Diamond p_1$) are decidable on Threshold Automata (NP-complete)

Liveness of Threshold Automata

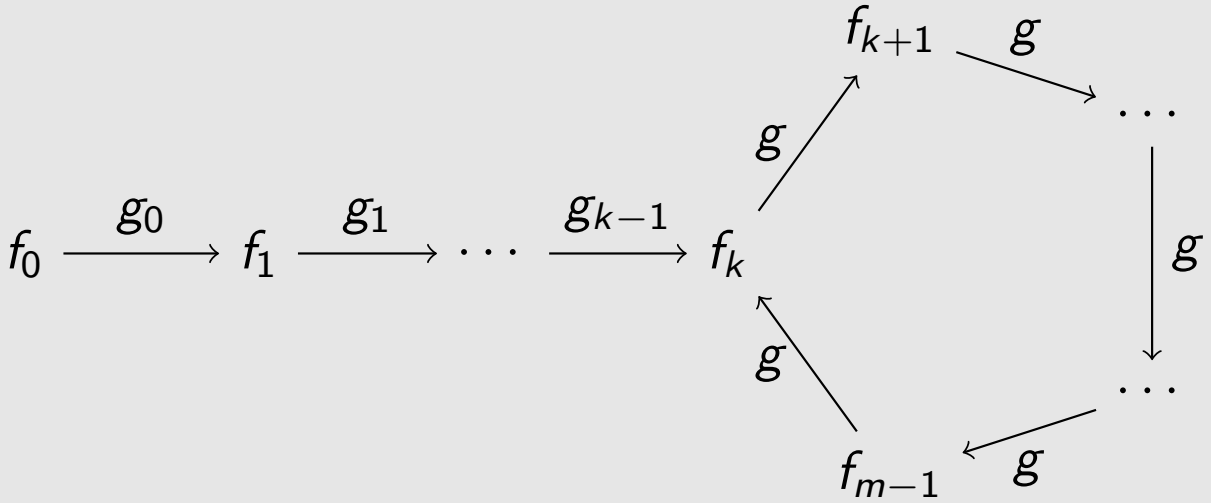
Additional constraint: if $c \models guard$ then $\mu c \models guard$ for $\mu \in \mathbb{N} \setminus \{0\}$

Theorem [Konnov et al. 2017]

The diameter stay bounded if we restrict the visited configurations to:

$$\left\{ c \mid \bigwedge_{i=1}^{m} \bigvee_{l \in A_i \subset L} (c(l) \neq 0) \wedge \bigwedge_{l \in A_0} (c(l) = 0) \right\}$$

By using this property multiple times, we can find *lasso frames* such as:

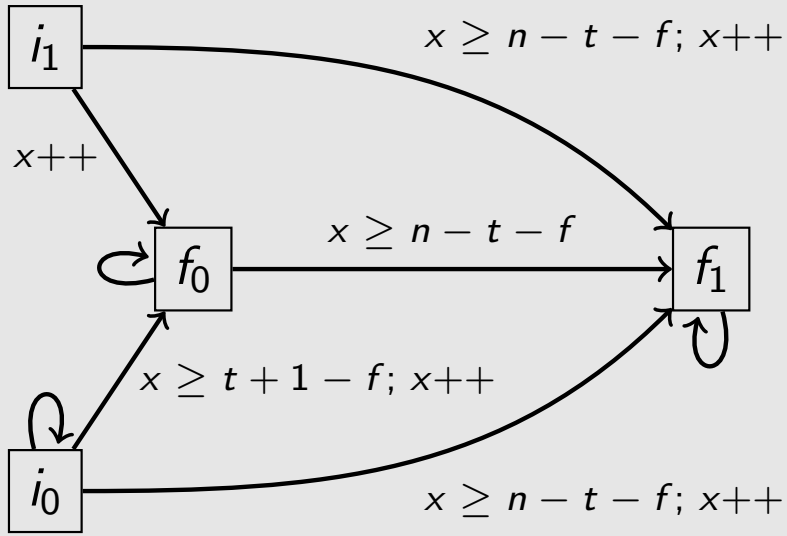


A lot of liveness properties have this kind of counter-examples

Example of Properties

Example 1

- Property:
  $(i_0 = n - f) \rightarrow \Box(f_1 = 0)$
- Negation:
  $(i_0 = n - f) \wedge \Diamond(f_1 > 0)$
- Safety property: OK



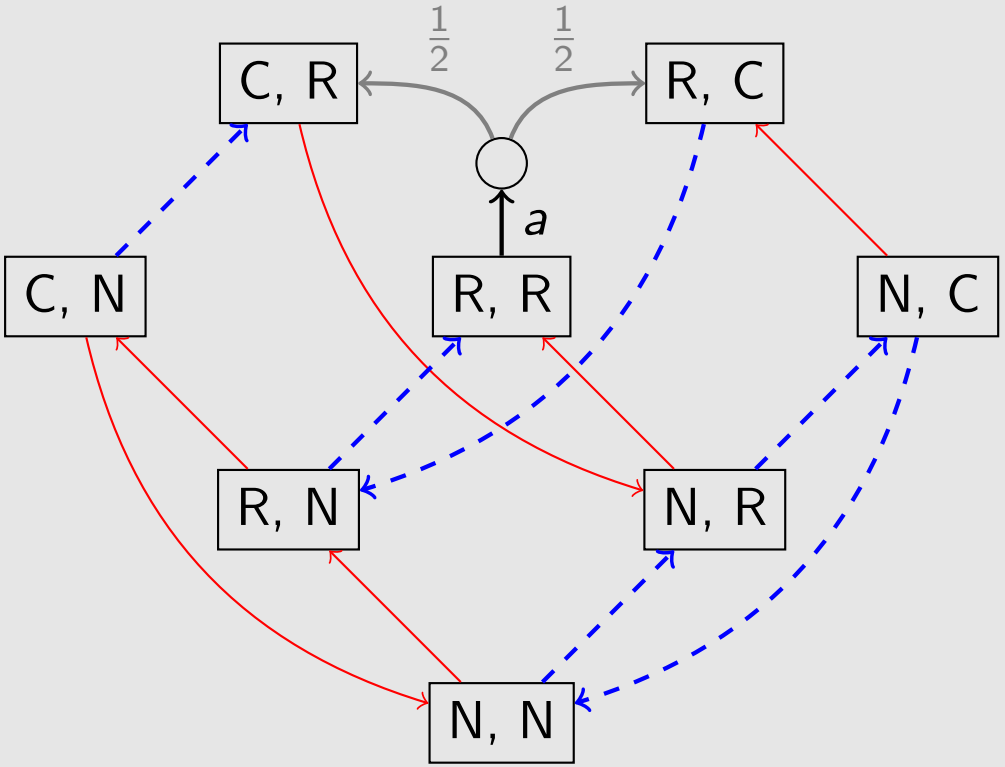
Example 2

- Property:
  $(i_1 = n - f) \wedge [\Box(x \geq n - t - f) \rightarrow \Diamond(f_0 = i1 = 0)] \rightarrow \Diamond(f_1 > 0)$
- Negation:

$$(i_1 = n - f) \wedge \Diamond(x \geq n + t - f) \wedge \Box(f_1 = 0)$$
$$\vee\, (i_1 = n - f) \wedge \Diamond(f_0 = i_1 = 0) \wedge \Box(f_1 = 0)$$

- Disjunction of lassos frames: OK

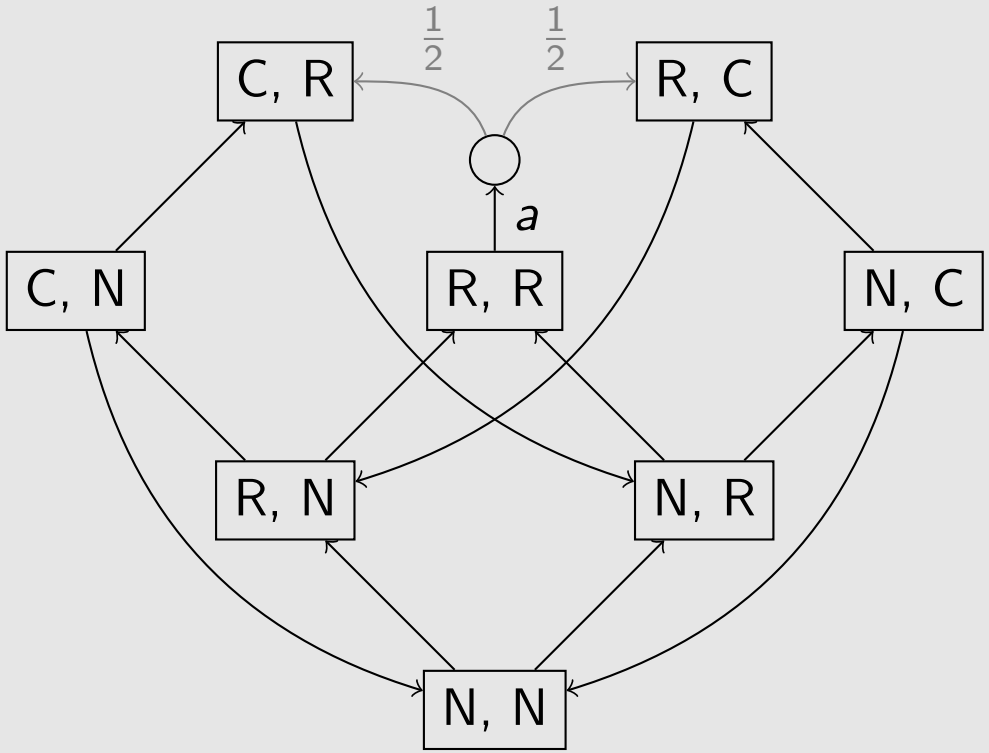
Markov Decision Processes



An MDP consists in:

- A set S of *states* $\{N, R, C\}^2$
- A set A of *actions* $\{\text{blue}, \text{red}, a\}$
- A partial function $\delta : S \times A \mapsto \text{distr}(S)$.

Finite MDP and End-Component



Definition

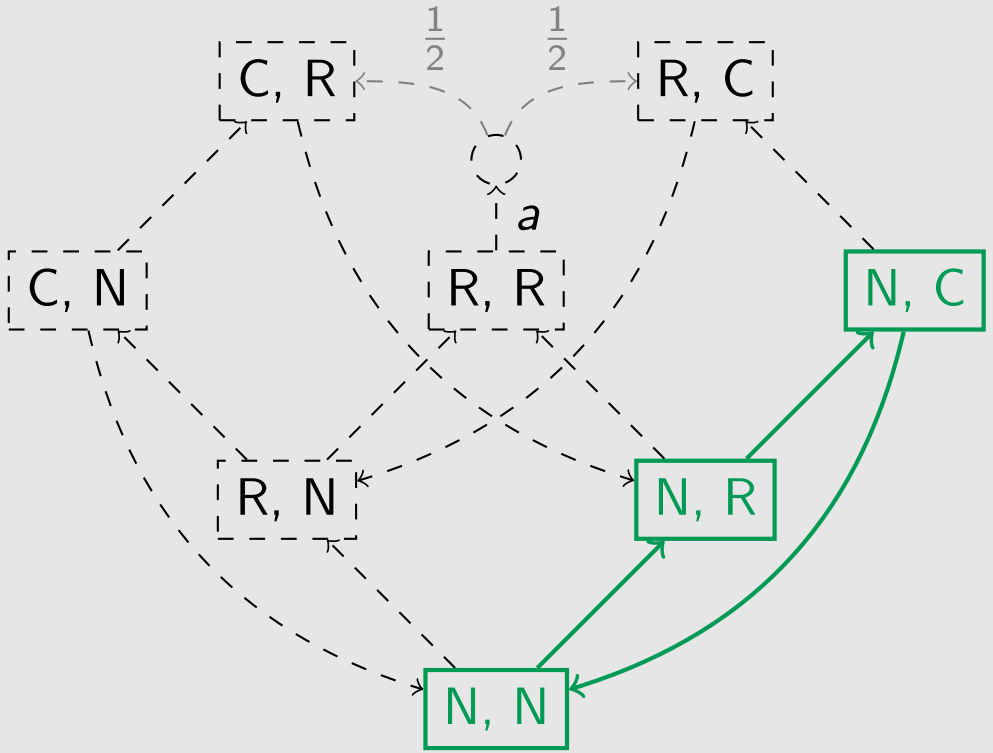
An *end-component* is a strongly connected sub-MDP.

Theorem

Almost surely, the nodes and actions taken infinitely often in a path form an end-component.

The verification of properties on *finite* MDP can be done by end-component analysis.

Finite MDP and End-Component



Definition

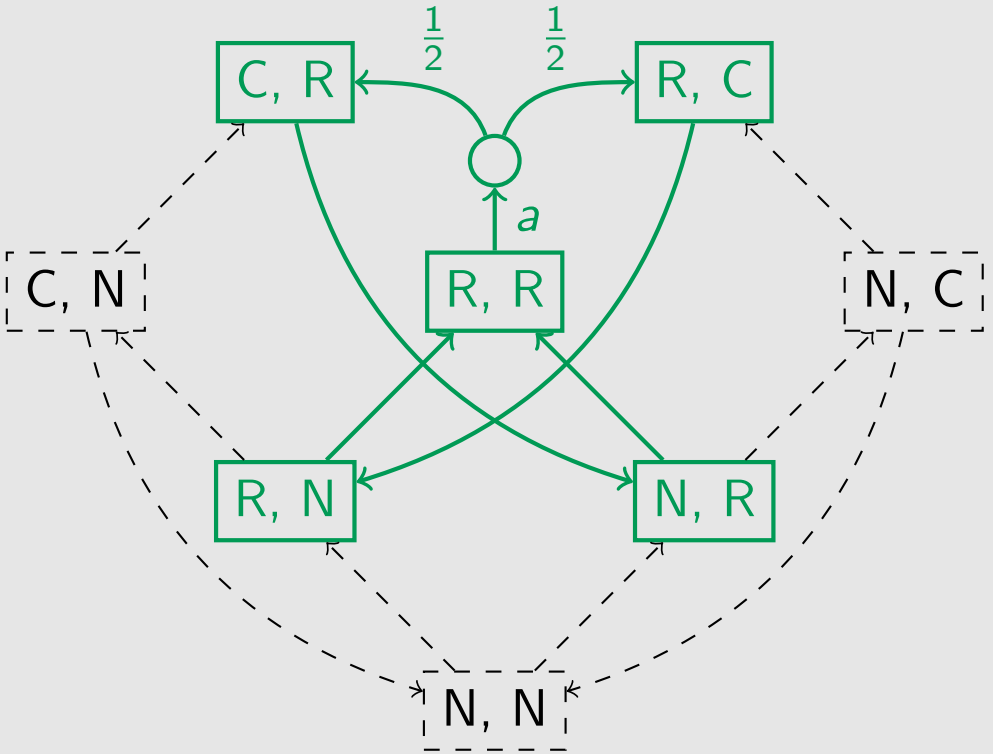
An *end-component* is a strongly connected sub-MDP.

Theorem

Almost surely, the nodes and actions taken infinitely often in a path form an end-component.

The verification of properties on *finite* MDP can be done by end-component analysis.

Finite MDP and End-Component



Definition

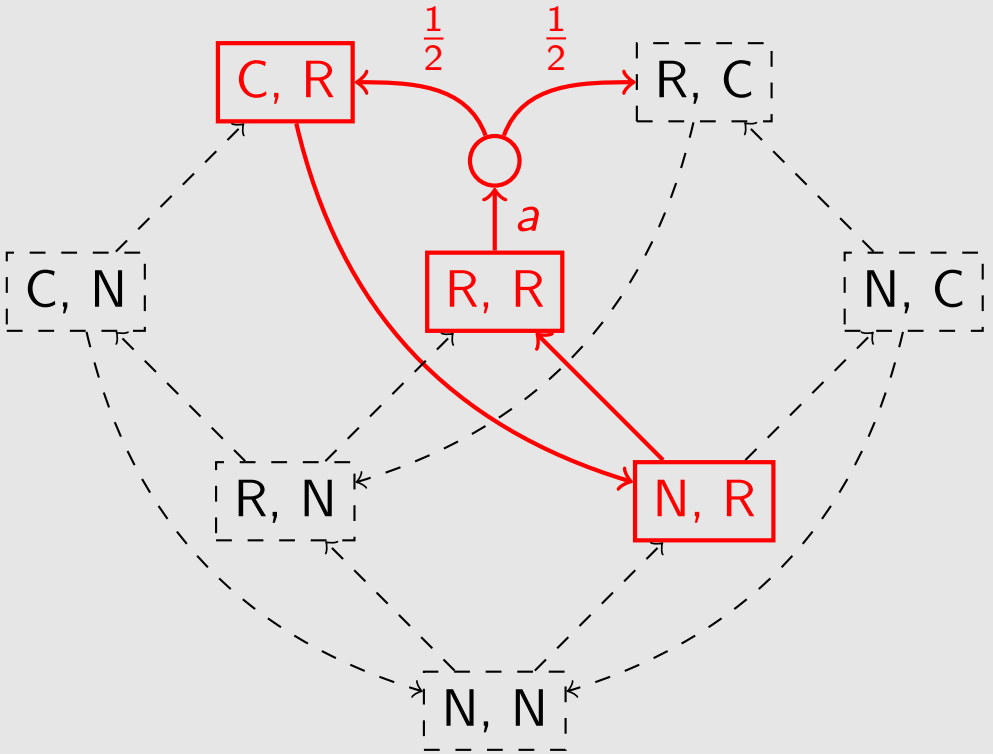
An *end-component* is a strongly connected sub-MDP.

Theorem

Almost surely, the nodes and actions taken infinitely often in a path form an end-component.

The verification of properties on *finite* MDP can be done by end-component analysis.

Finite MDP and End-Component



Definition

An *end-component* is a strongly connected sub-MDP.

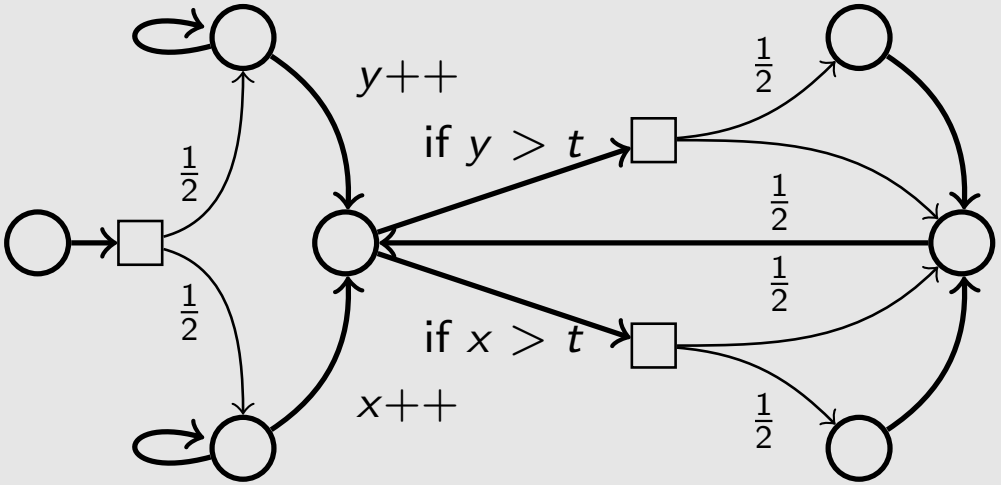
Theorem

Almost surely, the nodes and actions taken infinitely often in a path form an end-component.

The verification of properties on *finite* MDP can be done by end-component analysis.

Table of Contents

Probabilistic Threshold Automata

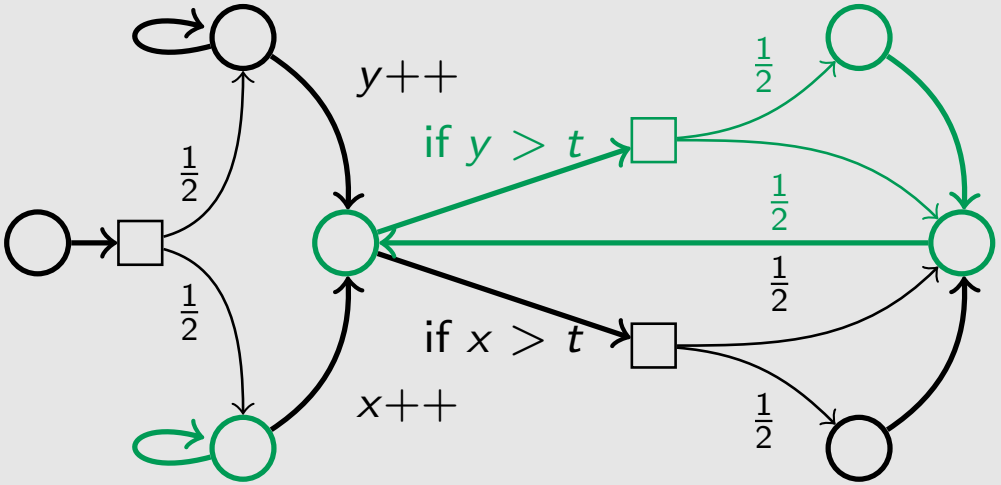


Edges destinations are now a distribution of locations.
The semantic is given as an infinite union of *finite* Markov Decision Processes.

Theorem

The edges taken by an infinite path forms almost surely an union of strongly connected sub-automata (called *connected cells*).

Probabilistic Threshold Automata



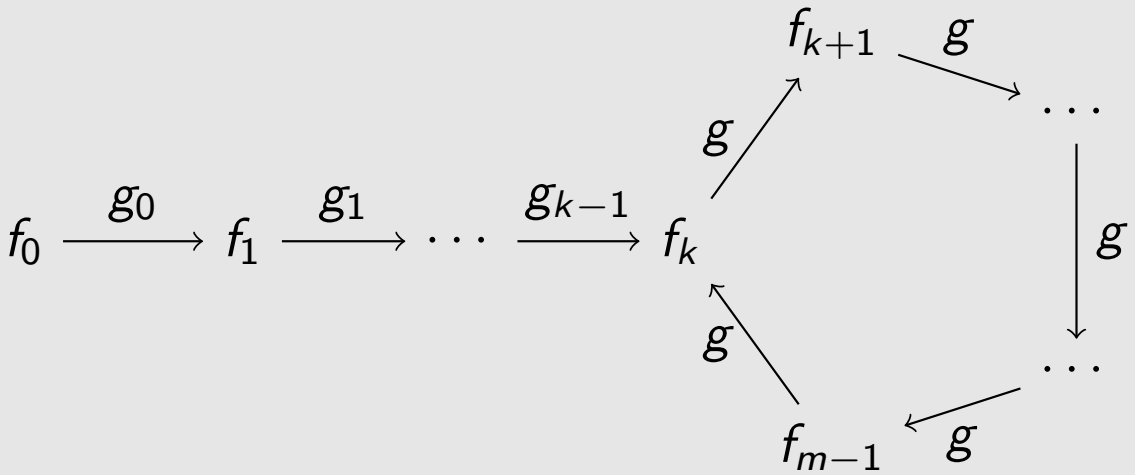
Edges destinations are now a distribution of locations.
The semantic is given as an infinite union of *finite* Markov Decision Processes.

Theorem

The edges taken by an infinite path forms almost surely an union of strongly connected sub-automata (called *connected cells*).

Verification of Probabilistic Threshold Automata

Consider the lasso frame τ:



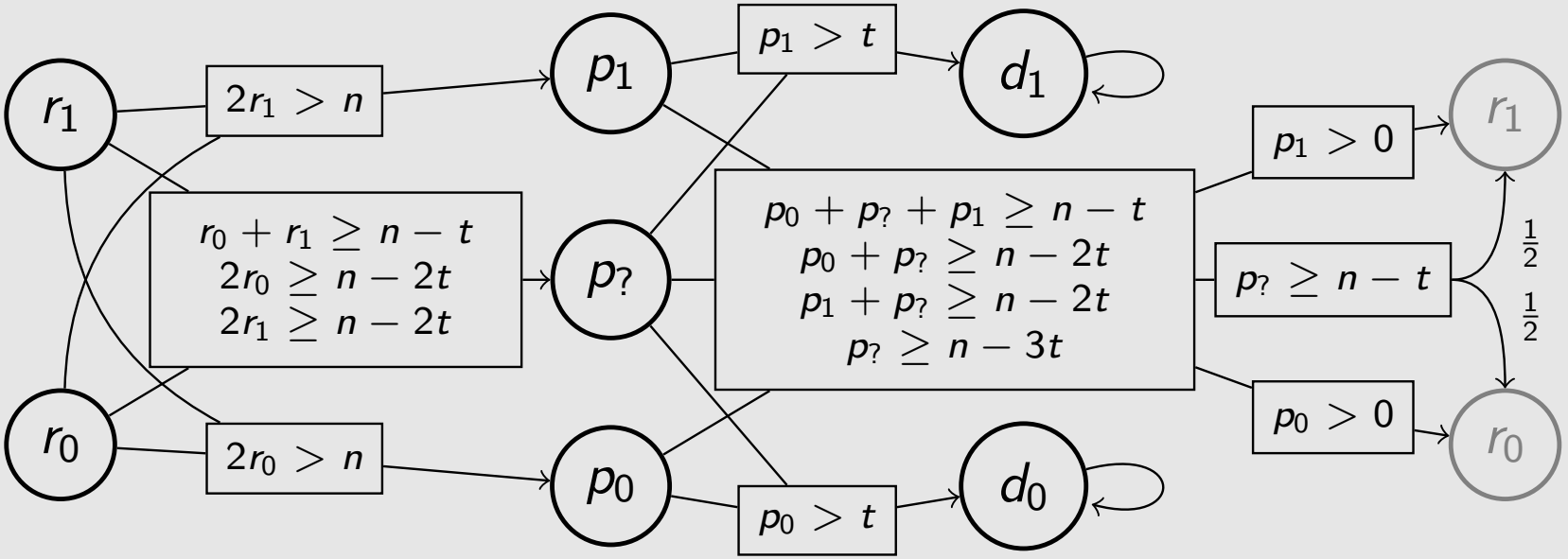
Theorem

If $g = \bigwedge_{i=1}^{m} \bigvee_{l \in A_i \subset L} (c(l) \neq 0) \wedge \bigwedge_{l \in A_0} (c(l) = 0)$ then:

- there exists a strategy for achieving $\mathbb{P}(\tau) > 0$
- iff there exists a lasso path $c_0 \ldots c_{l-1} c_l \ldots c_{n-1} c_l$ and a connected cell C such that $C \cap A_0 = \emptyset$ and only edges in C appear infinitely often in the path.

We can verify almost sure termination on probabilistic threshold automata

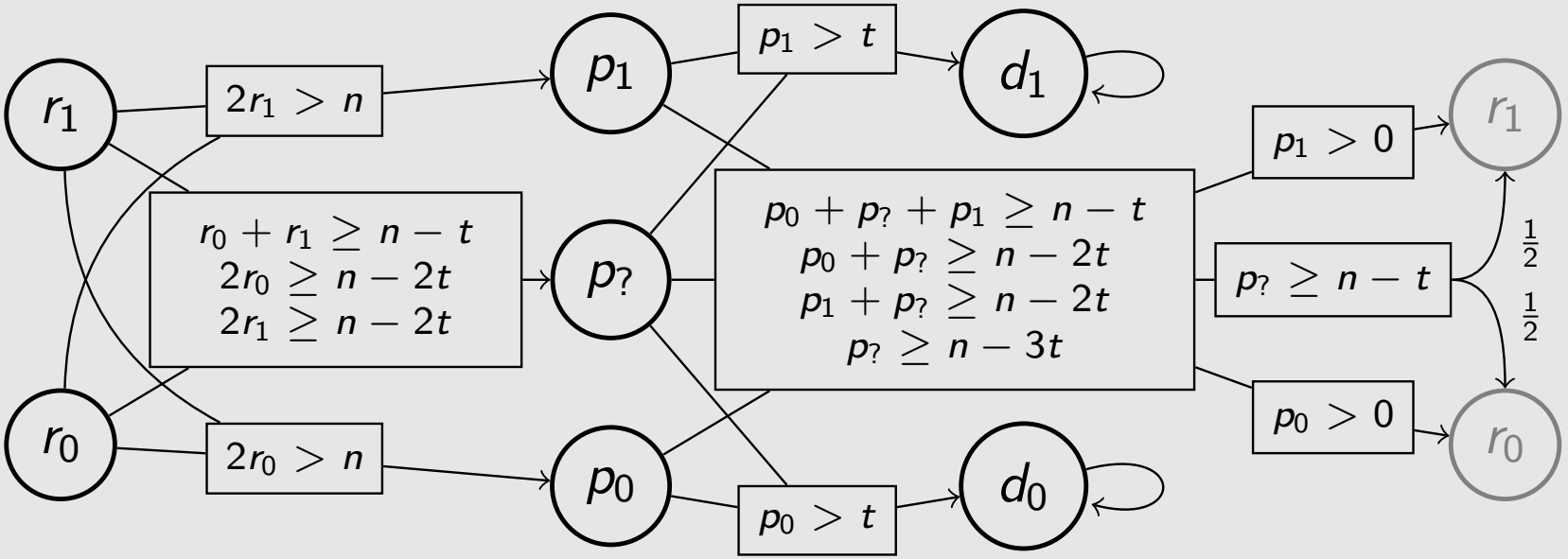
Ephemeral Threshold Automata



Example: 3 processes $a_1 a_2 a_3$, $n = 3$, $t = 1$

a_1 :	r_0				
a_2 :	r_0				
a_3 :	r_1				
r_0 :	2	0	0	0	$\cdots$
r_1 :	1	0	0	0	$\cdots$
p_0 :	0	0	0	0	$\cdots$
$p_?$:	0	0	0	0	$\cdots$
p_1 :	0	0	0	0	$\cdots$

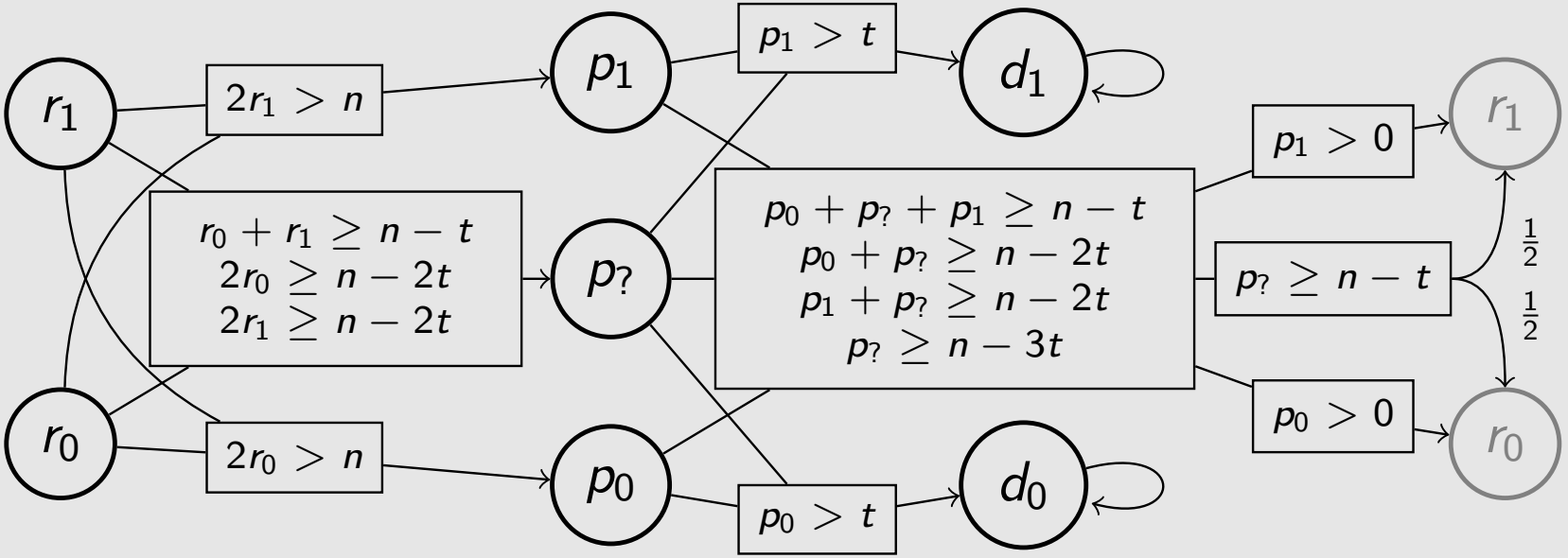
Ephemeral Threshold Automata



Example: 3 processes $a_1 a_2 a_3$, $n = 3$, $t = 1$

a_1 :	r_0	$p_?$			
a_2 :	r_0				
a_3 :	r_1				
r_0 :	2	0	0	0	$\cdots$
r_1 :	1	0	0	0	$\cdots$
p_0 :	0	0	0	0	$\cdots$
$p_?$:	0	1	0	0	$\cdots$
p_1 :	0	0	0	0	$\cdots$

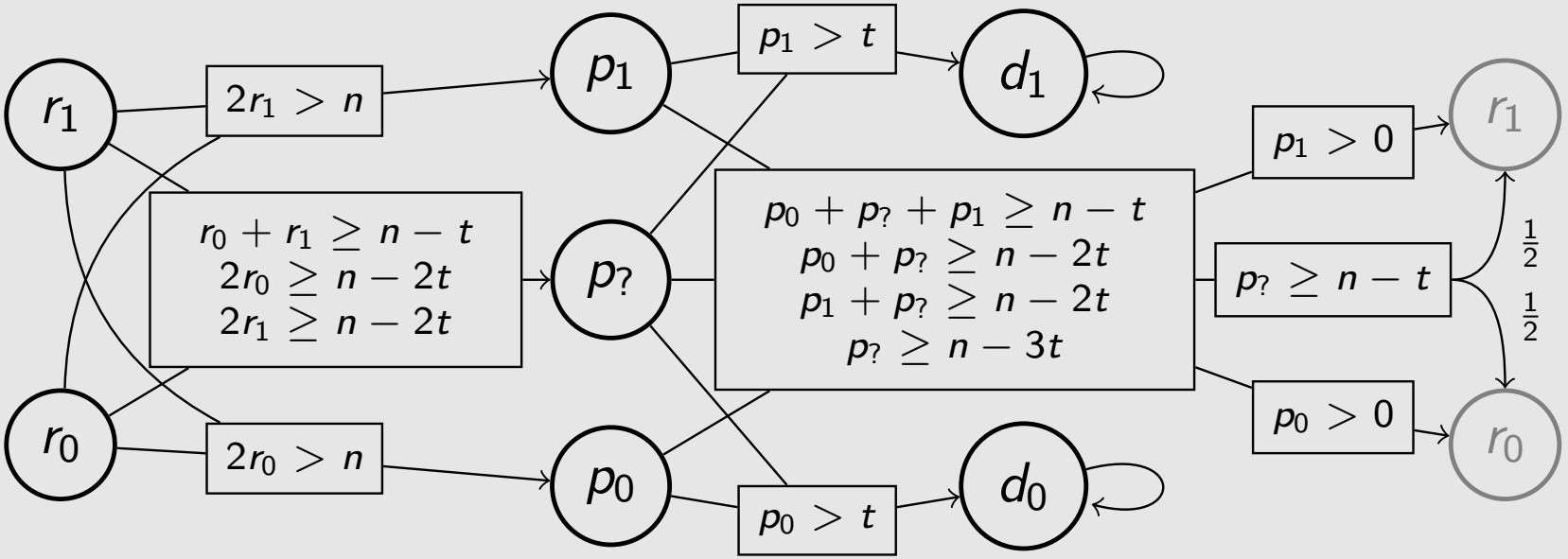
Ephemeral Threshold Automata



Example: 3 processes $a_1 a_2 a_3$, $n = 3$, $t = 1$

a_1 :	r_0	$p_?$			
a_2 :	r_0	p_0			
a_3 :	r_1				
r_0 :	2	0	0	0	$\cdots$
r_1 :	1	0	0	0	$\cdots$
p_0 :	0	1	0	0	$\cdots$
$p_?$:	0	1	0	0	$\cdots$
p_1 :	0	0	0	0	$\cdots$

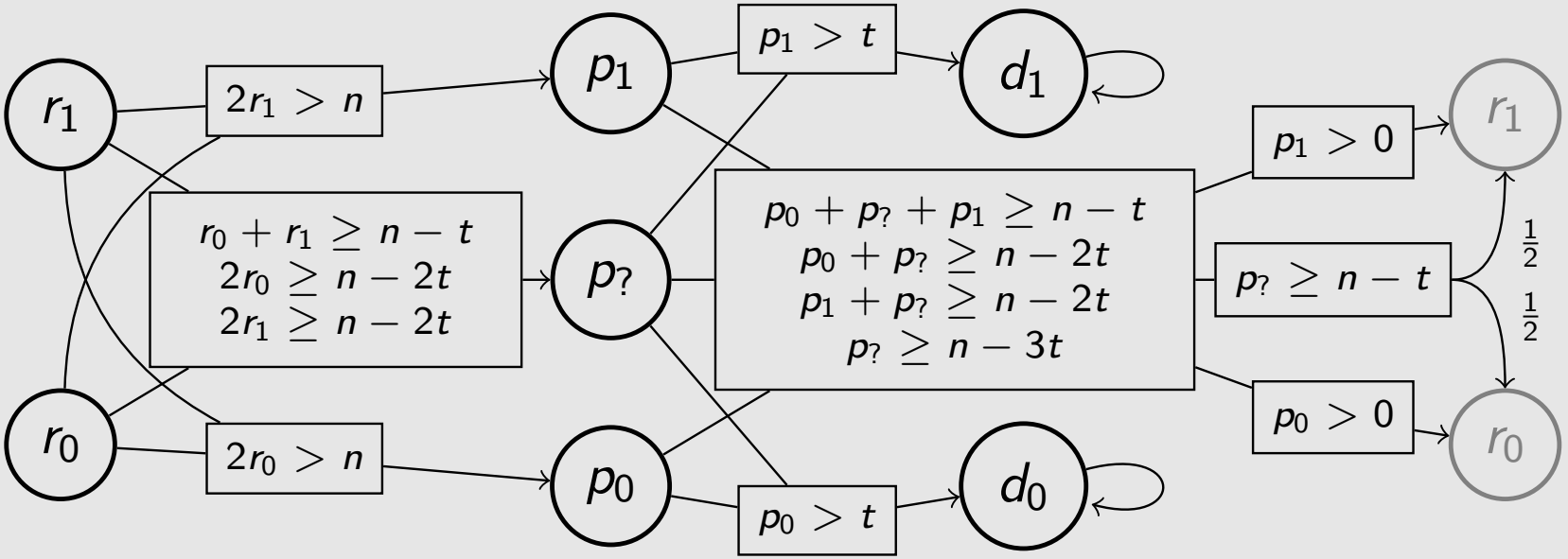
Ephemeral Threshold Automata



Example: 3 processes $a_1 a_2 a_3$, $n = 3$, $t = 1$

a_1 :	r_0	$p_?$			
a_2 :	r_0	p_0	r_0		
a_3 :	r_1				
r_0 :	2	0	1	0	$\cdots$
r_1 :	1	0	0	0	$\cdots$
p_0 :	0	1	0	0	$\cdots$
$p_?$:	0	1	0	0	$\cdots$
p_1 :	0	0	0	0	$\cdots$

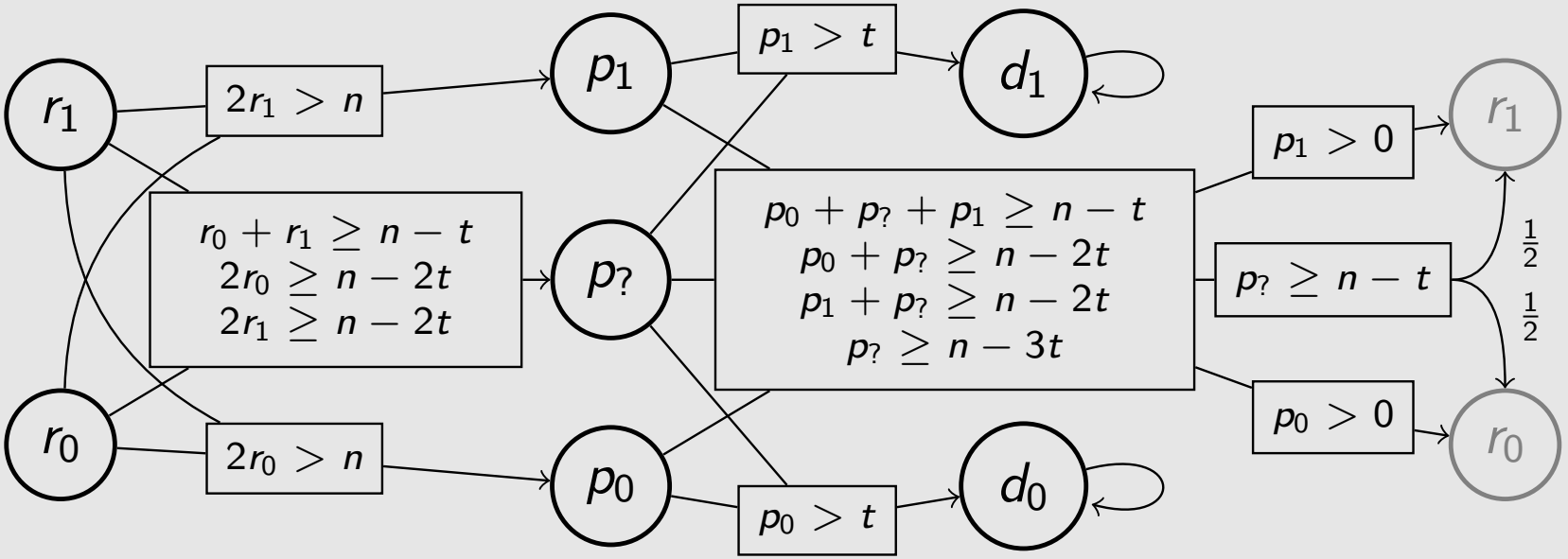
Ephemeral Threshold Automata



Example: 3 processes $a_1 a_2 a_3$, $n = 3$, $t = 1$

a_1 :	r_0	$p_?$			
a_2 :	r_0	p_0	r_0		
a_3 :	r_1	p_0			
r_0 :	2	0	2	0	$\cdots$
r_1 :	1	0	0	0	$\cdots$
p_0 :	0	2	0	0	$\cdots$
$p_?$:	0	1	0	0	$\cdots$
p_1 :	0	0	0	0	$\cdots$

Ephemeral Threshold Automata



Example: 3 processes $a_1 a_2 a_3$, $n = 3$, $t = 1$

a_1 :	r_0	$p_?$	r_1	$p_?$	$\cdots$
a_2 :	r_0	p_0	r_0	$p_?$	$\cdots$
a_3 :	r_1	p_0	r_0	p_0	$\cdots$
r_0 :	2	0	2	0	$\cdots$
r_1 :	1	0	1	0	$\cdots$
p_0 :	0	2	0	1	$\cdots$
$p_?$:	0	1	0	2	$\cdots$
p_1 :	0	0	0	0	$\cdots$

Verification of Ephemeral Threshold Automaton

r_0	2	0	2	0	$\cdots$
r_1	1	0	1	0	$\cdots$
p_0	0	2	0	1	$\cdots$
$p_?$	0	1	0	2	$\cdots$
p_1	0	0	0	0	$\cdots$
	$2r_0 > n$	$p_0 > t$	$2r_0 > n$	$p_0 + p_? \geq n - 2t$	$\cdots$
	$2r_0 \geq n - 2t$	$p_0 + p_? \geq n - 2t$	$2r_0 \geq n - 2t$	$p_1 + p_? \geq n - 2t$	$\cdots$
	$2r_1 \geq n - 2t$	$p_1 + p_? \geq n - 2t$	$2r_1 \geq n - 2t$	$p_? \geq n - 3t$	$\cdots$
	$-$	$p_? \geq n - 3t$	$-$	$-$	$\cdots$

Observation

The set of guards true in a column gives constrain the set of guards of the next column.

Guard Automaton

We build an automaton with:

- The set of guards valuations as *states*.
- An *edge* between two states if possible to reach one from the other.

Correction

To any 'real' execution of the ephemeral threshold automaton corresponds one in the guard automaton.

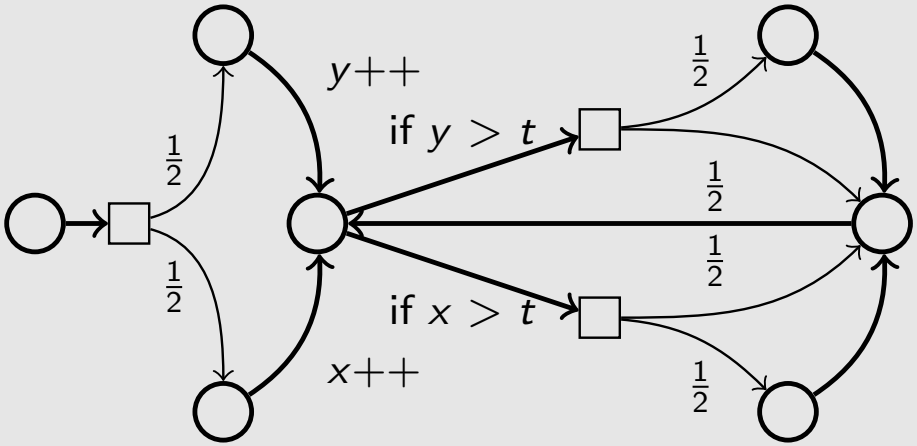
Completeness in practice

On concrete example, the converse holds (can be checked automatically).

Limitations For now, restricted to 'weak' schedulers.

Table of Contents

Conclusion

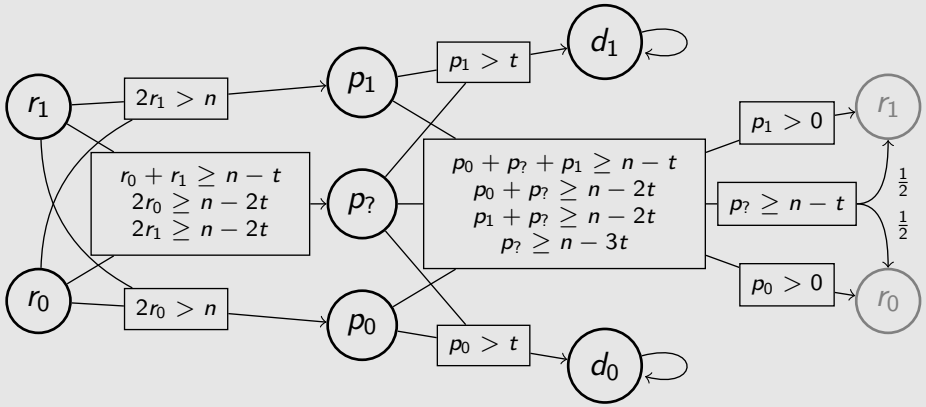


Probabilistic TA

- Extends Threshold Automata
- Extends verification
- No realistic algorithm
- Qualitative properties only

Ephemeral TA

- Represents real algorithm
- Restriction on schedulers
- Qualitative properties only
- Undecidable in general but promising practical results



PhD Objectives

Improve Current Models

- Weaken requirements
- Optimize verification of ETA
- Unify ETA and PTA

Understand Theory

- Why can we decide so often on case studies?
- Establish complexity
- Find realistic restrictions on schedulers

Correct-by-Design Synthesis

- Generate optimal guards
- Generate resilience condition ($2t < n$)

Verify Quantitative Properties

- *e.g.* if $n < 40$, algorithm terminates in 3 rounds with probability at least 0.7

Implementation

Tools to ease the verification and design of correct/optimal algorithms