Safraless Procedures for Timed Specifications

Barbara Di Giampaolo (U Salerno)
Gilles Geeraerts and Jean-François Raskin (ULB)
Nathalie Sznajder (Paris 6)

Journées FAC 2011 - Toulouse

Safraless Procedures
Motivations

Language inclusion

Prog	Spec
A NBW	B NBW

Prog $\models$ Spec iff $L(A) \subseteq L(B)$

iff $L(A) \cap L(B^c) = \emptyset$

B^c obtained by **determinization** of B

Realizability

$\Sigma = \Sigma_1 \cup \Sigma_2$

Spec.
Φ
LTL

$?(\Sigma_1) \parallel Env(\Sigma_2) \models \Phi$

$\exists \lambda_1 \bullet \forall \lambda_2 \bullet \underline{\exists \text{run } r \text{ of } A_\Phi} \bullet r$ accepts $\text{Outcome}(\lambda_1, \lambda_2)$

Remove second alternation by **determinization** of A_Φ.

$\exists \lambda_1 \bullet \forall \lambda_2 \bullet$ unique r of A^d on $\text{Outcome}(\lambda_1, \lambda_2)$ is accepting

Realizability

$\Sigma = \Sigma_1 \cup \Sigma_2$

Spec.
Φ
LTL

$?(\Sigma_1) \mid\mid Env(\Sigma_2) \models \Phi$

$\exists \lambda_1 \bullet \forall \lambda_2 \bullet \underline{\exists \text{run } r \text{ of } A_\Phi} \bullet r \text{ accepts Outcome}(\lambda_1, \lambda_2)$

Remove second alternation by **determinization** of A_Φ.

$\exists \lambda_1 \bullet \forall \lambda_2 \bullet \underline{\text{unique}} \text{ } r \text{ of } A^d \text{ on Outcome}(\lambda_1, \lambda_2) \text{ is accepting}$

Lead to a reduction to games

Determinization is difficult for NBW

① DBWs are **strictly less expressive** than NBWs. Need Rabin or Parity acceptance conditions.

② Simple subset constructions are not sufficient: Safra's construction uses **trees of subsets** (encoding history of run).

... and resistent to efficient implementation

①

No good **symbolic** data structures
for the underlying state space.

②

LTL synthesis: Rabin (NP-complete) or
Parity games ($NP \cap coNP$)
on a doubly exponential state space.

... and resistent to efficient implementation

①

No good **symbolic**

for the un

Safra's determinization has been implemented by Tasiran et al. (1995) and Thomas et al.(2005): need of **intricate data structures** and **very low scalability** (8-12 states).

complete) or

(NP∩coNP)

y exponential state space.

... and resistent to efficient implementation

① **No** good **symboli**

②

Safra's de... (1995) and Th... **structures** and **very**

...y Tasiran et al. ...ed **data**

... (NP∩coNP)

...ly exponential state space.

With alternative approaches, we are able to treat automata with **hundreds of states**

How to avoid determinization ?

“Safraless” decision procedures

“Safraless” decision procedures

- Safraless complementation (with no determinization):
  - ★ Progress measure construction [Klarlund91].
  - ★ Rank construction [KupfermanVardi97,01]:
    NBW → UcoBW → ABW → NBW

“Safraless” decision procedures

- Safraless complementation (with no determinization):
  - ★ Progress measure construction [Klarlund91].
  - ★ Rank construction [KupfermanVardi97,01]:
    NBW → UcoBW → ABW → NBW
- Safraless realizability/synthesis:
  - ★ Rank construction [KupfermanVardi05]:
    LTL → UcoBW → ABT → NBT → Büchi game
  - ★ K-co-Büchi condition:
    [ScheweFinkbeiner07] application to distributed synthesis,
    [FiliotJinRaskin09] application to LTL synthesis.
    LTL → UcoBW → UKcoBW → Safety game

Plan of the talk

- How to avoid Safra construction ?

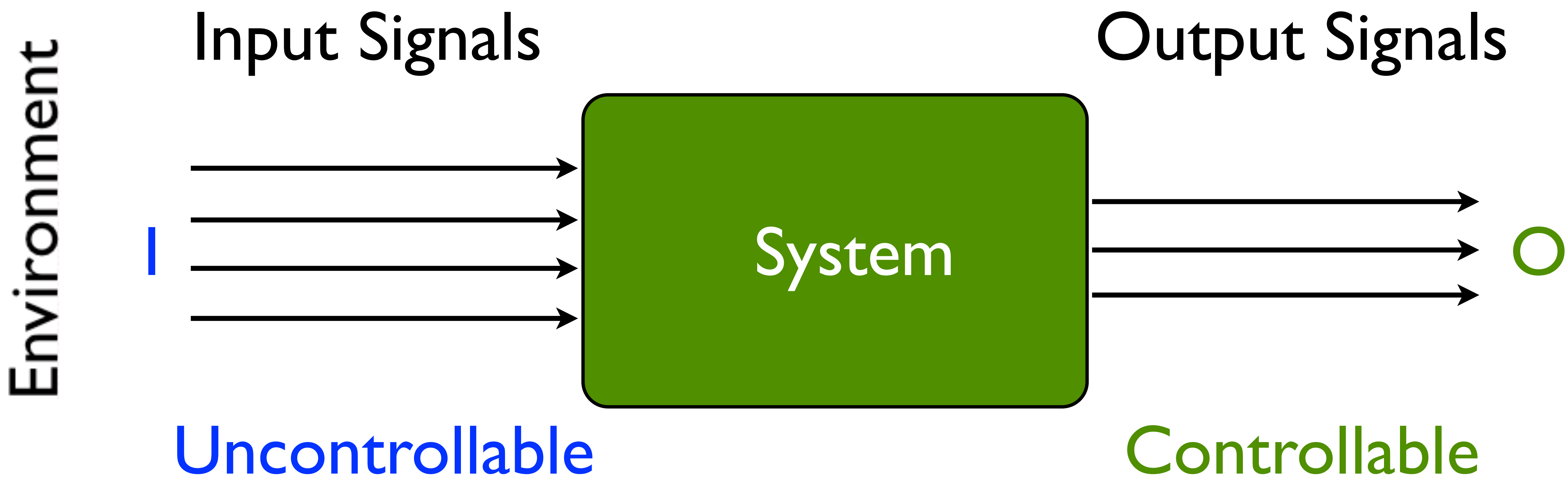
  focus on synthesis

- Extensions to timed specifications ?

  focus on synthesis

- Summary of the results of a paper published in FORMATS'2010.

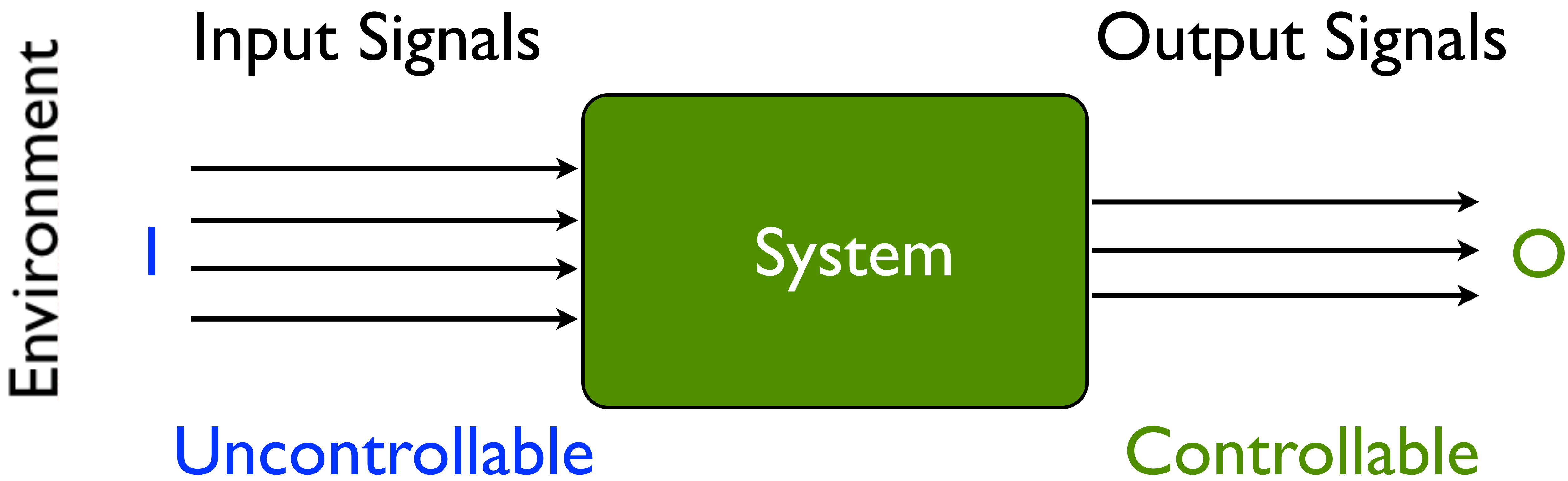
The Synthesis Problem



Interaction produces an infinite word w over $\Sigma = 2^{I \cup O}$

$(o_0 \cup i_0)(o_1 \cup i_1)(o_2 \cup i_2)\ldots \qquad o_j \subseteq O \quad i_j \subseteq I$

The Synthesis Problem



Realizability Problem

Given a LTL spec Φ, does there exist a way for the System to choose its signals along time, so that, **no matter how** the environment chooses its signals, the resulting execution satisfies the formula Φ ?

Synthesis as an ∞-game

Player 1

System M

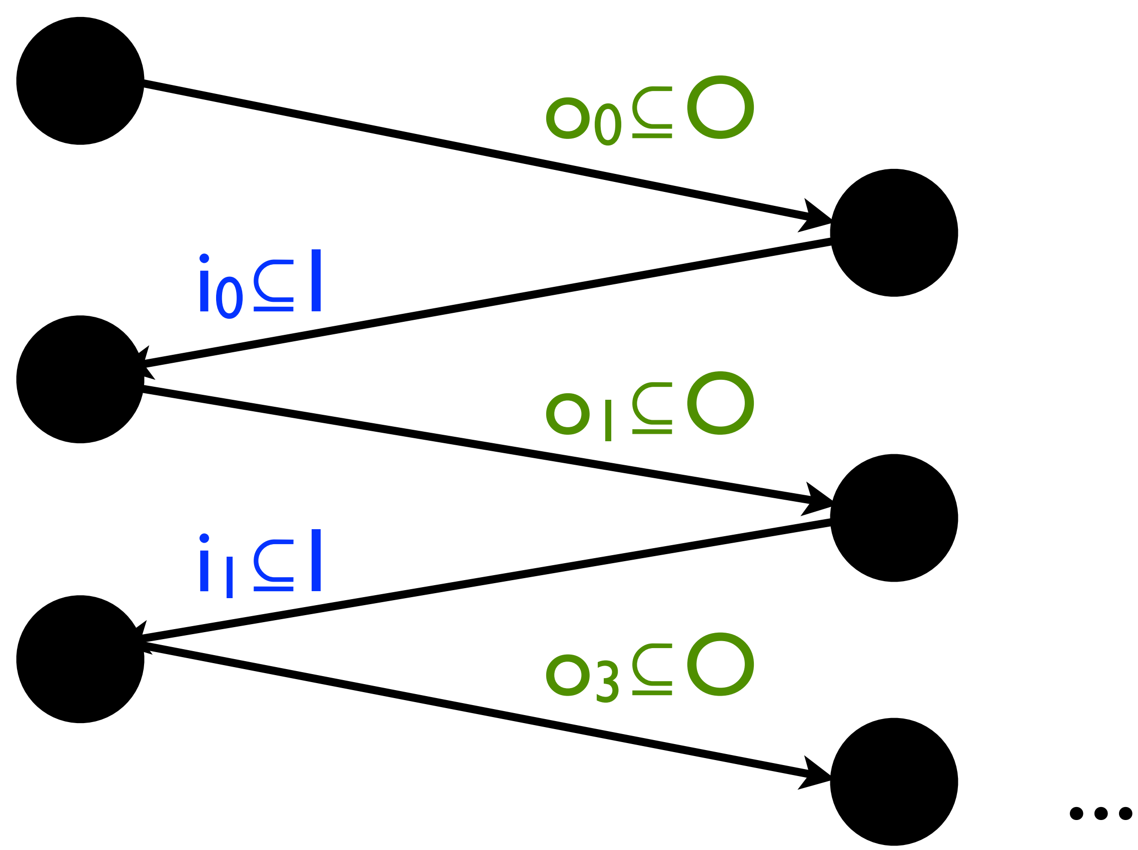
Player 2

Environment

Synthesis as an ∞-game

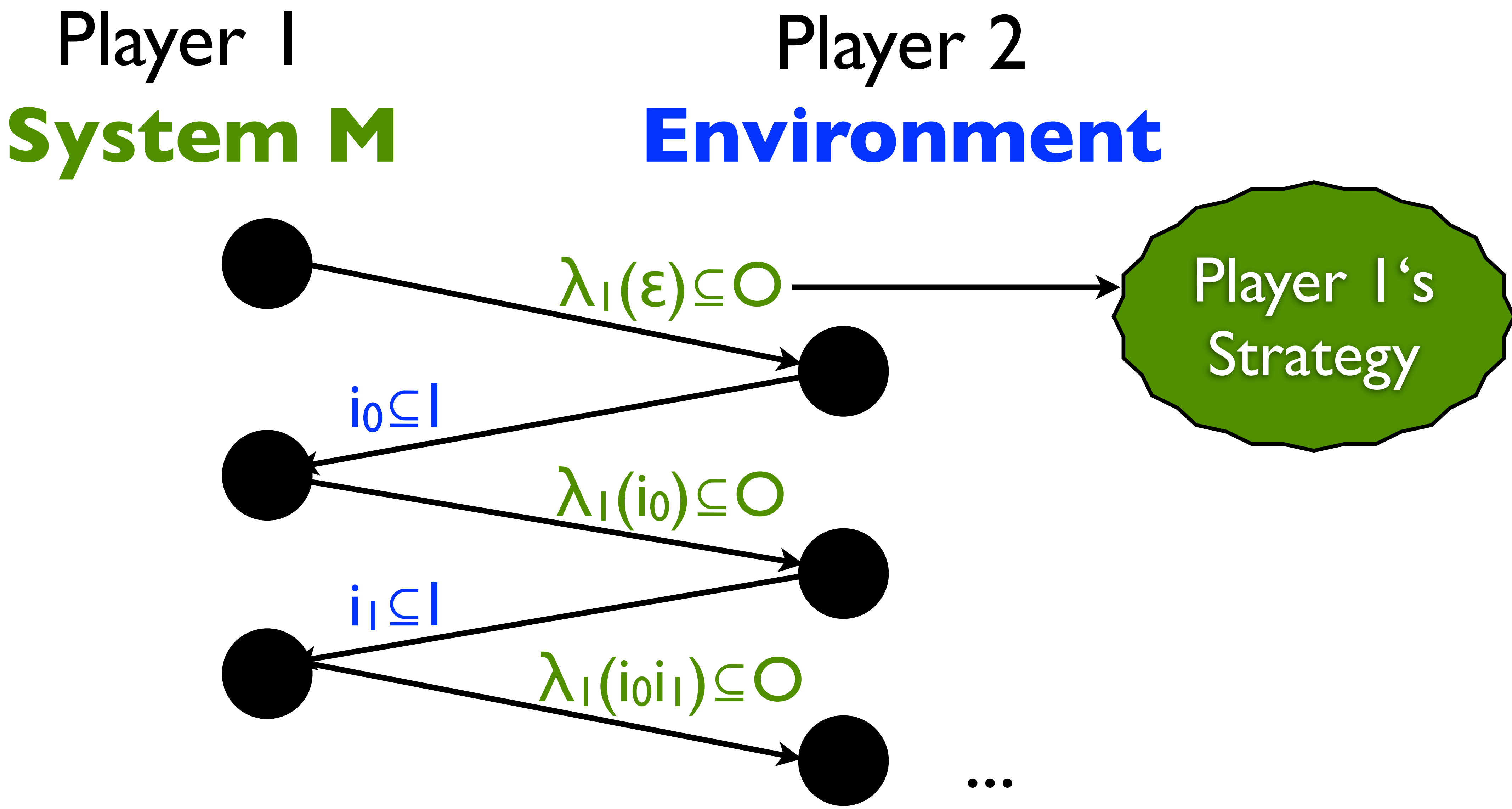
Player 1 — **System M**

Player 2 — **Environment**



$(o_0 \cup i_0)(o_1 \cup i_1)(o_2 \cup i_2)\ldots$

Synthesis as an ∞-game



The system wins the game if the play
$(\lambda_1(\varepsilon)\cup i_0)(\lambda_1(i_0)\cup i_1)(\lambda_1(i_0 i_1)\cup i_2)\ldots$ satisfies ϕ

The Synthesis Problem

Realizability Problem

Given a LTL spec Φ, does there exist a way for the System to choose its signals along time, so that, **no matter how** the environment chooses its signals, the resulting execution satisfies the formula Φ ?

Φ is **realizable**

iff

$$\exists \lambda_{\mathrm{I}} \cdot \mathbf{Outcome}(\lambda_{\mathrm{I}}) \subseteq [\![\Phi]\!]$$

"Classical" solution

Classical solution proposed by Pnueli and Rosner, 1989:

$$\text{LTL} \xrightarrow{2^{O(n)}} \textbf{Nd}.\ \text{Büchi Word Automata} \xrightarrow[\text{[Safra, 88]}]{2^{O(m \log m)}} \textbf{Det}.\ \text{Rabin Word Automata}$$

Realizability
= Rabin Game

The problem has been shown to be **2ExpTime-C** by the same authors.

An Alternative Solution

LTL

$\downarrow 2^{O(n)}$

Universal coBüchi Word automata

$\downarrow O(1)$

Universal **KcoBüchi** Word automata

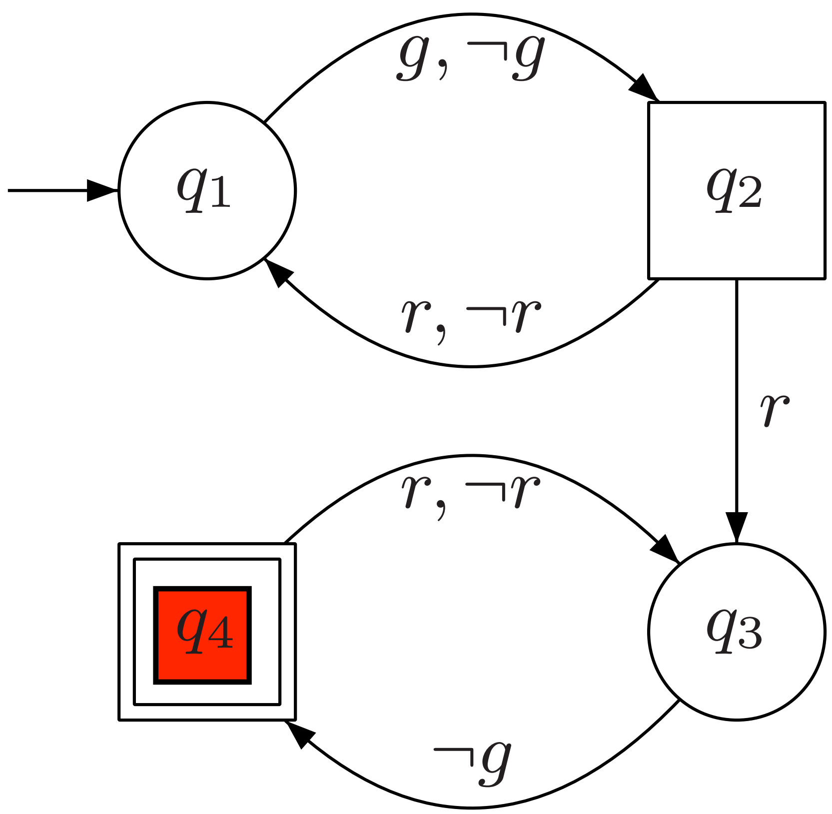
$\downarrow 2^{O(n2)}$

Det. KcoBüchi Word automata

Realizability

= Safety game

Universal coBüchi Word Automata



w ∈ L_{UcoB}(A)
iff
all runs of A on w visit
finitely many times α.

Σ^ω

¬g

r

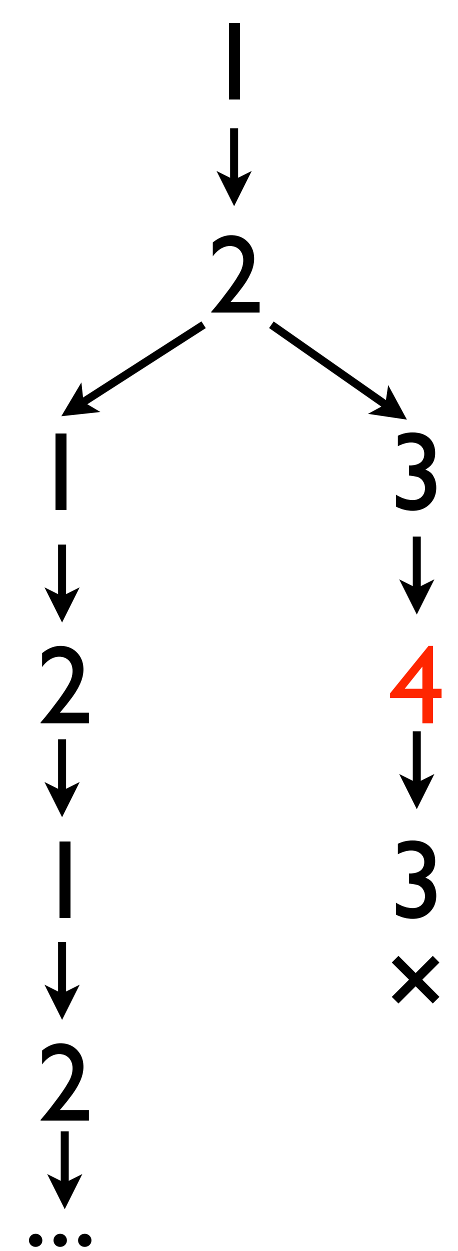
¬g

r

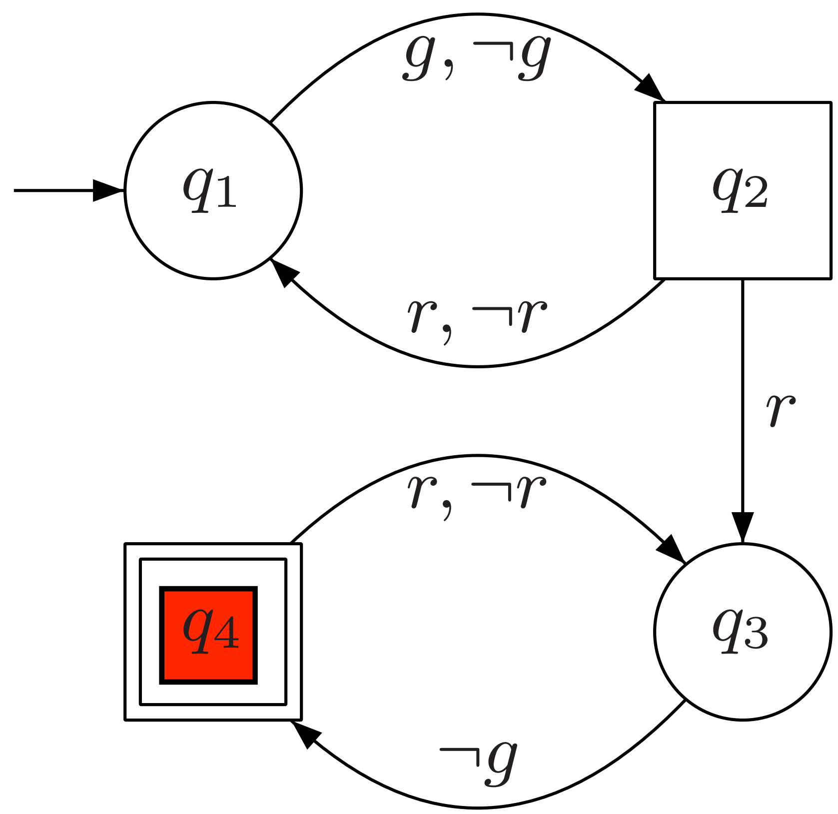
g

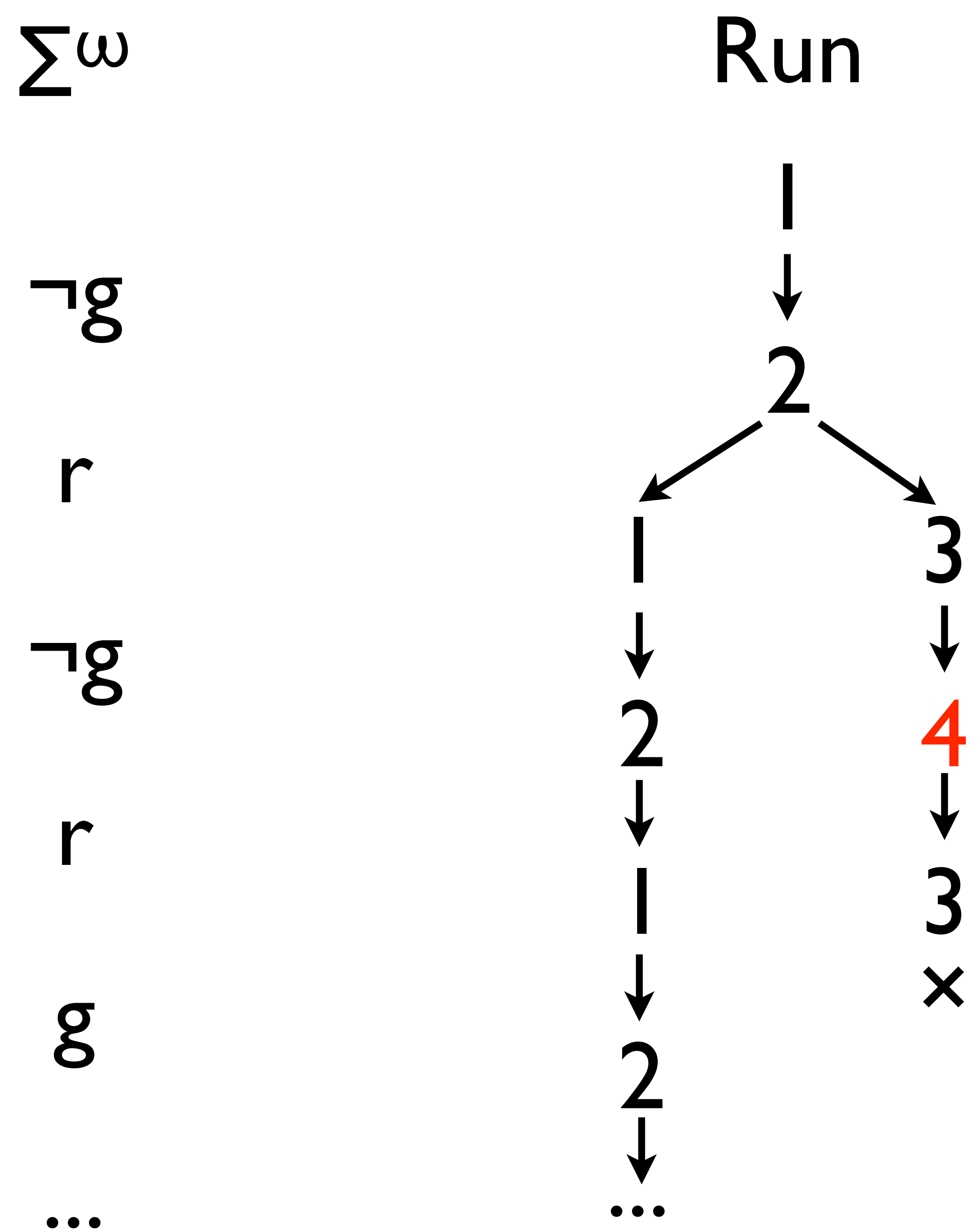
...

Run



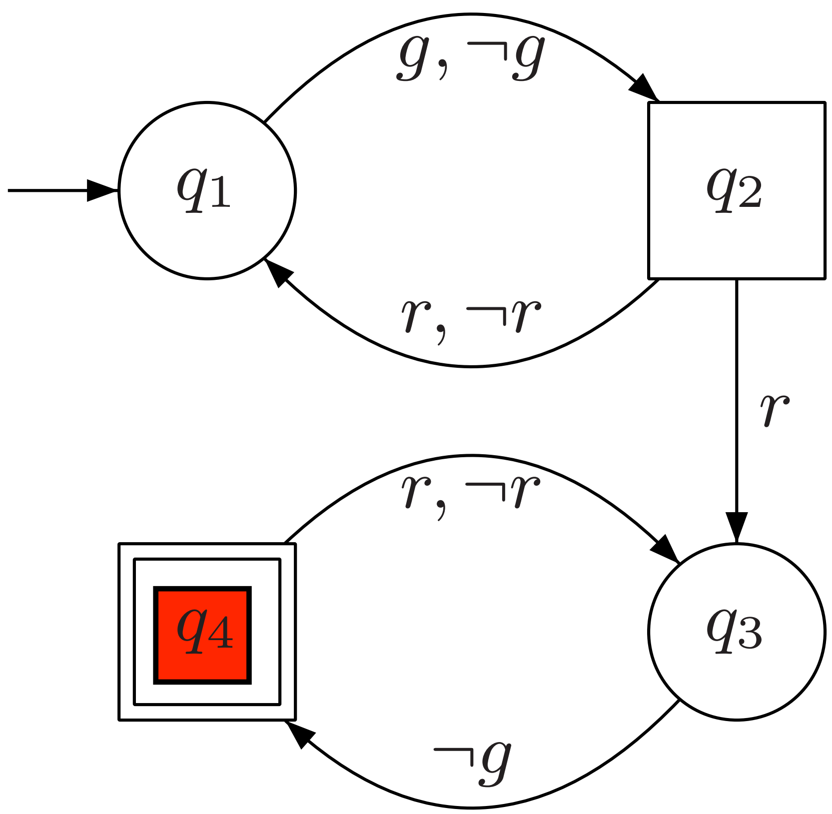
Universal **K**coBüchi Word Automata





w ∈ $L_{U,K}$(A)
iff
all runs of A on w visit
α **at most K times**.

Universal **K**coBüchi Word Automata



Σ^ω	Run
	1
$\neg g$	↓
	2
r	
…	…

Note that the ω-language accepted by a UKcoBW is a safety language.

w ∈ $L_{U,K}$

iff

all runs of A on

α **at most K t**

LTL, UcoBW and UKcoBW

Input

$$\Box(r \rightarrow \mathcal{X}(\Diamond g))$$

Output

LTL, UcoBW and UKcoBW

Input

$$\Box(r \rightarrow \mathcal{X}(\Diamond g))$$

Output

How to get an UcoBW ?

LTL, UcoBW and UKcoBW

Input

$$\Box(r \rightarrow \mathcal{X}(\Diamond g))$$

Output

LTL Φ

$\downarrow$

NBW $A_{\neg\Phi}$

$\downarrow$ Dualize

UcoBW $A_{\neg\Phi}$

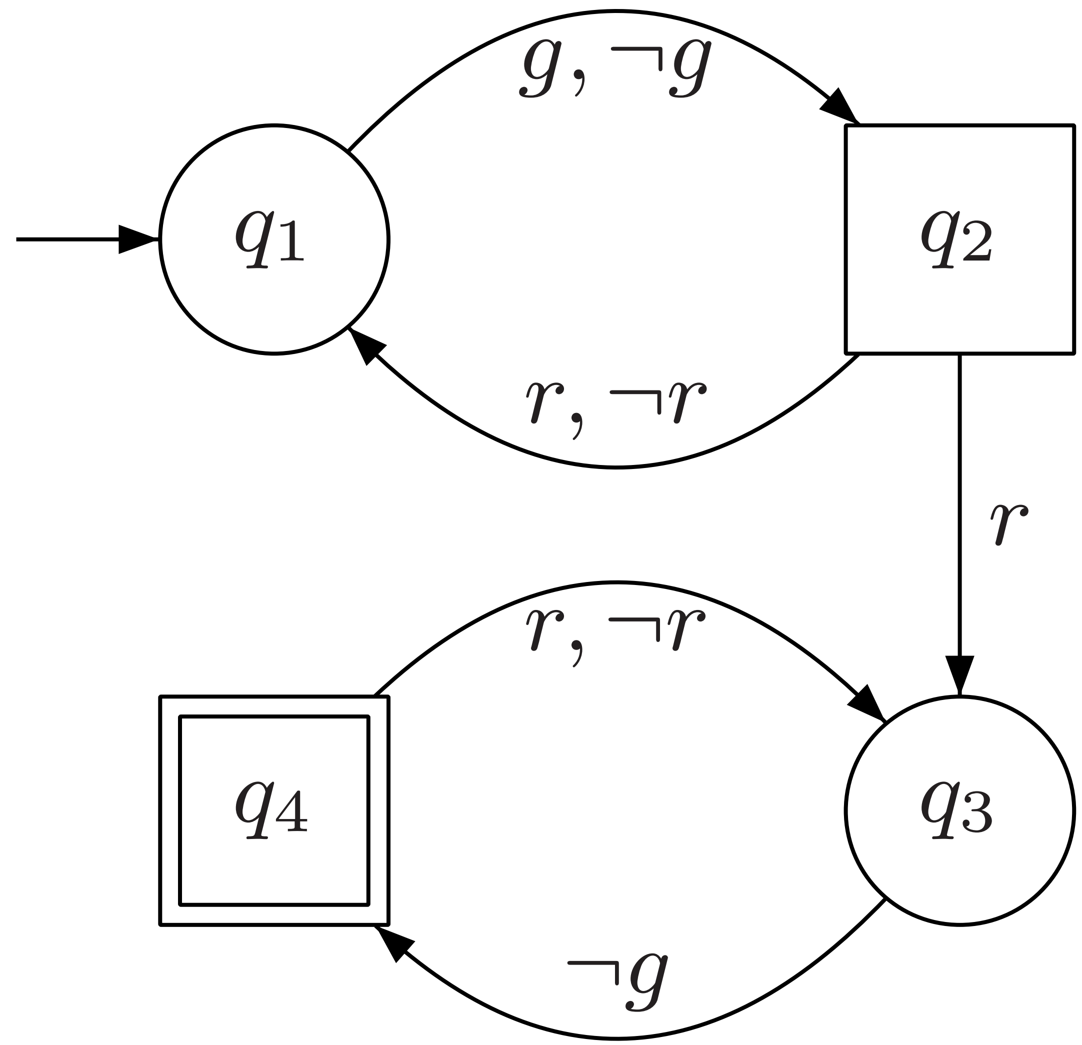
$$L_{UcoB}(A_{\neg\Phi}) = \{ w \mid w \models \Phi \}$$

LTL, UcoBW and UKcoBW

Input

$$\Box(r \rightarrow \mathcal{X}(\Diamond g))$$

Output



w ∈ $L_{UcoB}(A_\phi)$ iff **all** runs of A_ϕ on w visit **finitely many times** α.

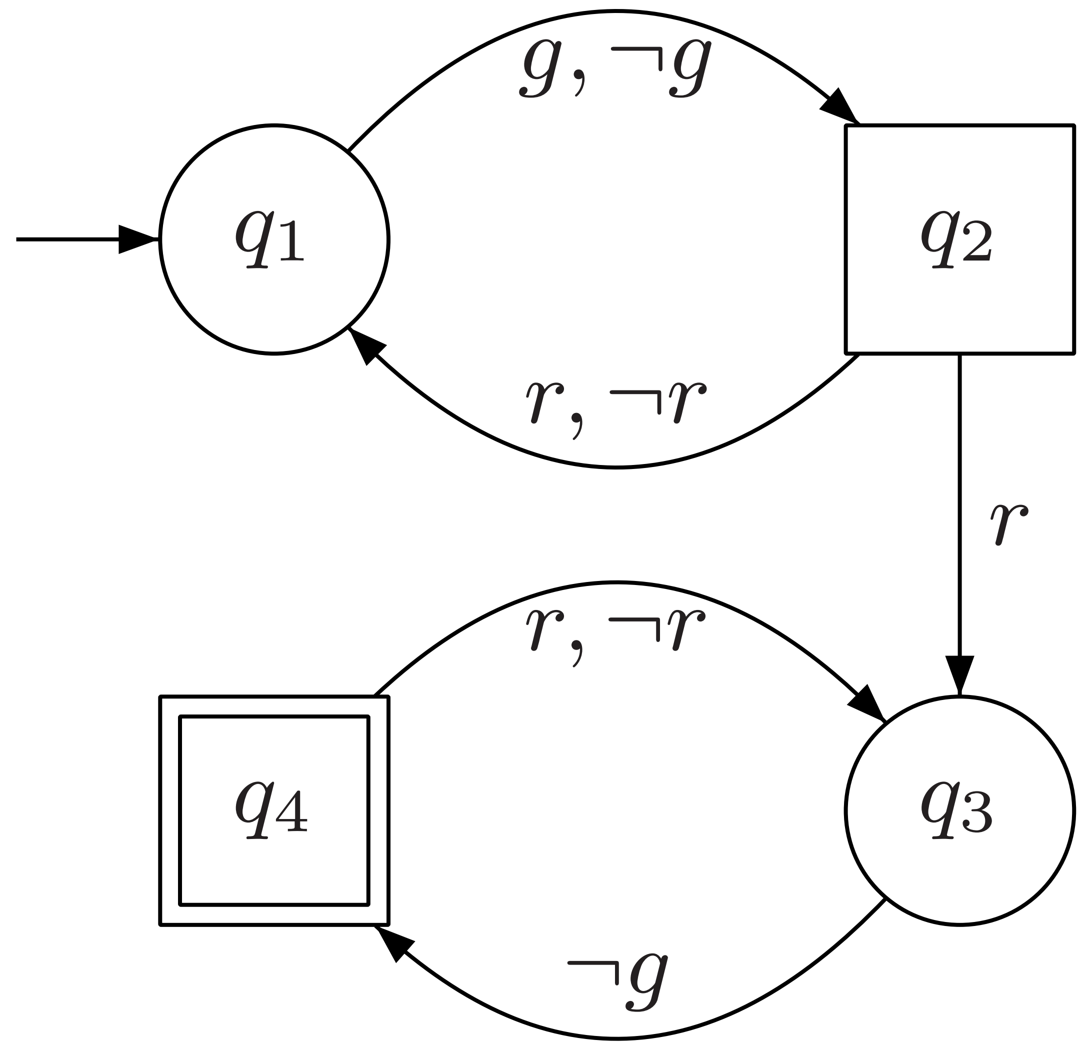
w ∈ $L_{U,K}(A_\phi)$ iff **all** runs of A on w visit **at most K times** α.

LTL, UcoBW and UKcoBW

Input

$$\Box(r \rightarrow \mathcal{X}(\Diamond g))$$

Output



$$L_{U,1}(A_\Phi) \subseteq L_{U,2}(A_\Phi) \subseteq \dots \subseteq L_{U,n}(A_\Phi) \subseteq \dots \subsetneq L_{UcoB}(A_\Phi) = [\![\Phi]\!].$$

(Finite Memory) Strategies

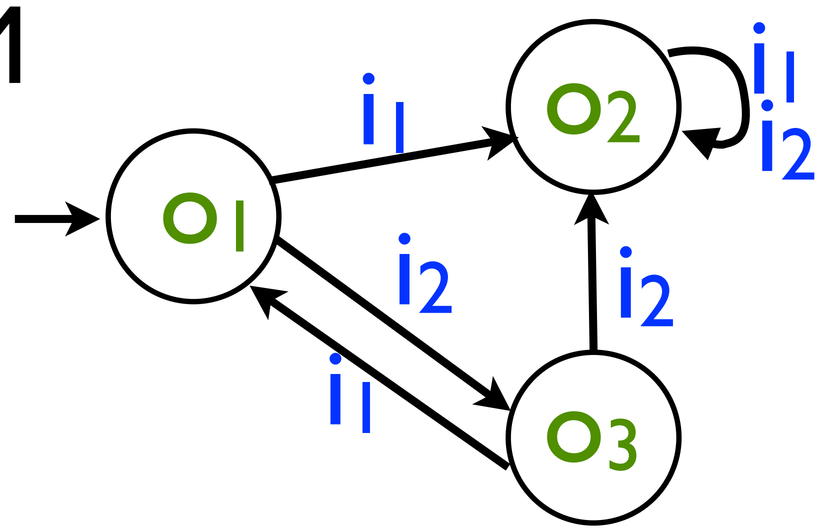
Strategies for Player 1:

$\lambda_1 : (\Sigma_1 \bullet \Sigma_2)^* \rightarrow \Sigma_1$

Finite Memory for Player 1:

(Complete) Moore Machines

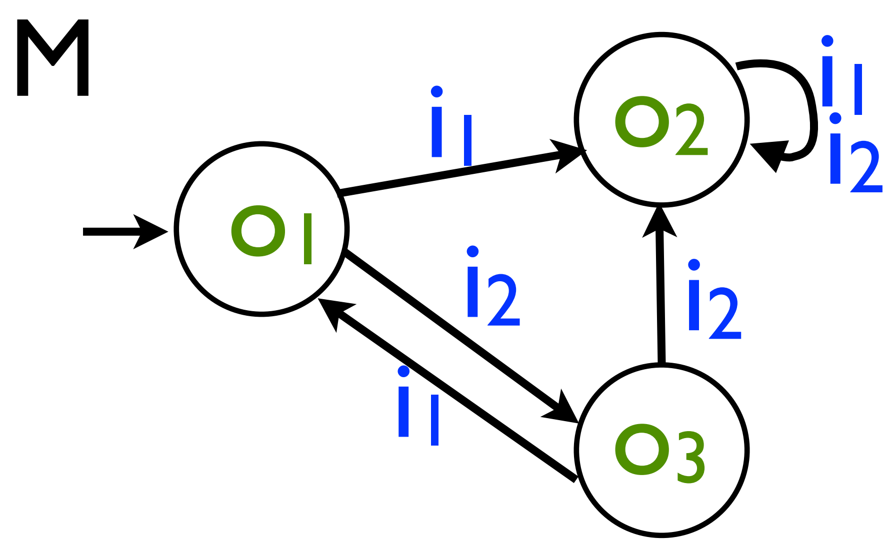
M



$L(M) = \{\text{infinite words over } \Sigma_1 \cup \Sigma_2\}$

Ex: $(o_1 \cup i_1)(o_2 \cup i_2)^\omega$

Finite Memory Strategies are Sufficient



$L(M) =$ infinite words over $\Sigma_1 \cup \Sigma_2$

Ex: $(o_1 \cup i_1)(o_2 \cup i_2)^\omega$

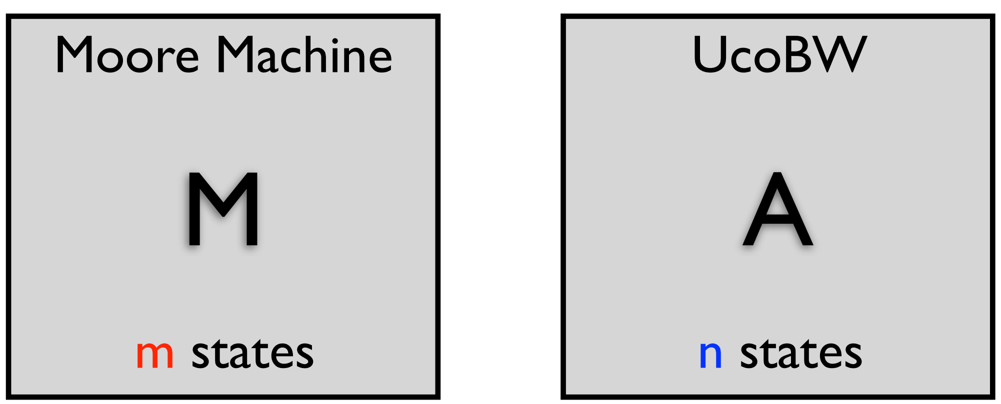
- ★ If a regular objective is realizable, then it is realizable by a **finite memory** strategy [Büchi69].
- ★ **Theorem [Safra88,Piterman08]** For an objective specified by a UCW, there is a Moore machine that realizes the objective iff there is a Moore machine with less than $2^{O(n2)}$.

Bounding Visits to Accepting States

Lemma. Let M be a Moore machine with m states, and A a UcoBW with n states. If $\mathbf{L}(M) \subseteq \mathbf{L}_{\mathbf{UcoB}}(A)$, then all runs on words of $\mathbf{L}(M)$ visit accepting states at most $m \times n$ times.

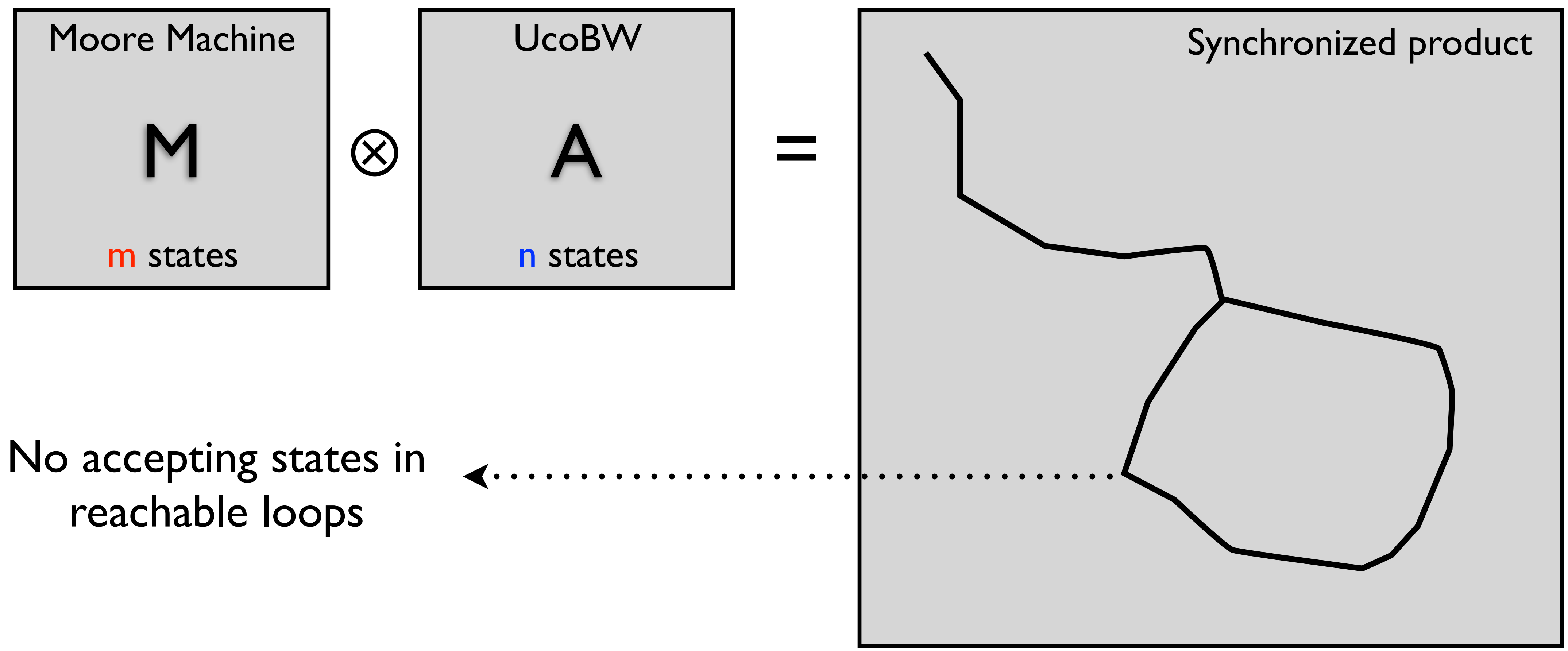
Bounding Visits to Accepting States

Lemma. Let M be a Moore machine with m states, and A a UcoBW with n states. If **L**(M)⊆$\mathbf{L}_{\mathbf{UcoB}}$(A), then all runs on words of **L**(M) visit accepting states at most m×n times.



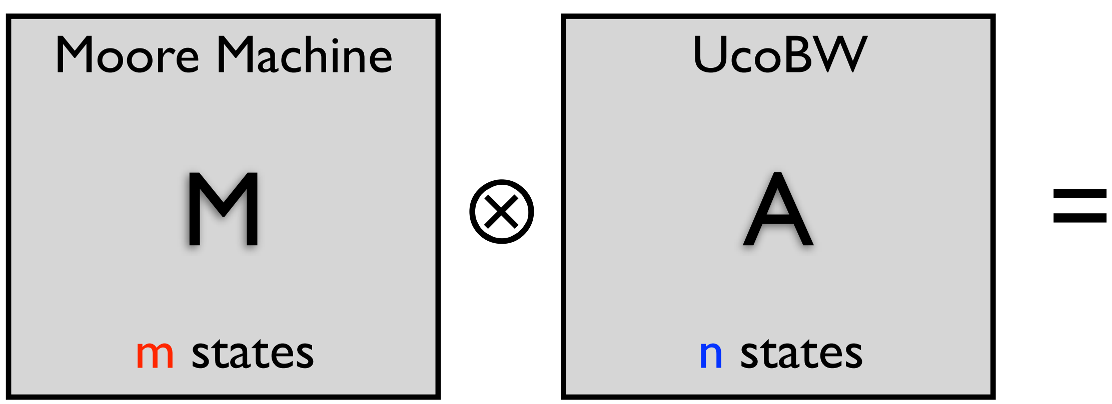
Bounding Visits to Accepting States

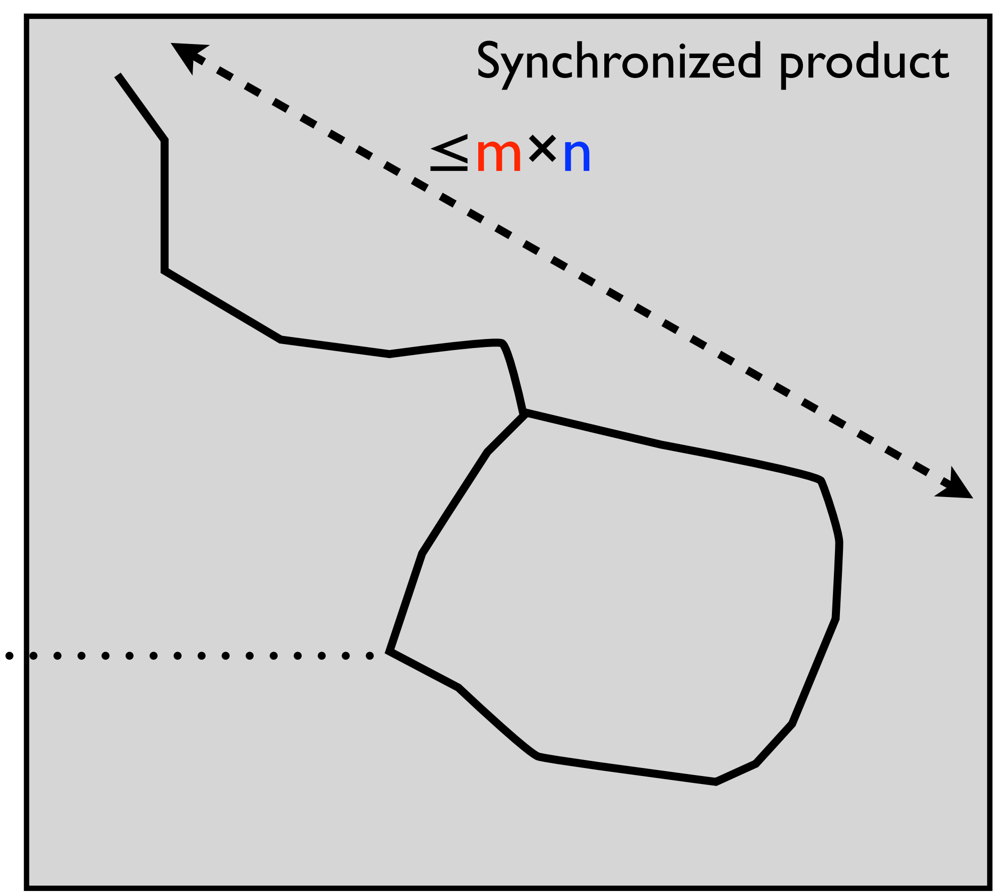
Lemma. Let M be a Moore machine with m states, and A a UcoBW with n states. If $\mathbf{L}(M) \subseteq \mathbf{L}_{\mathbf{UcoB}}(A)$, then all runs on words of $\mathbf{L}(M)$ visit accepting states at most $m \times n$ times.



Bounding Visits to Accepting States

Lemma. Let M be a Moore machine with m states, and A a UcoBW with n states. If $\mathbf{L}(M) \subseteq \mathbf{L}_{\mathbf{UcoB}}(A)$, then all runs on words of $\mathbf{L}(M)$ visit accepting states at most $m \times n$ times.





No accepting states in reachable loops

□

At most m×n accepting states on a path

Bounding Visits to Accepting States

Corollary 1. For all UcoBW A with n states, for all Moore machine M with m states, let $\mathbf{K}=n\times m$, then

$$\mathbf{L}(M)\subseteq\mathbf{L}_{\mathbf{UcoB}}(A) \text{ iff } \mathbf{L}(M)\subseteq\mathbf{L}_{\mathbf{u,K}}(A)$$

Corollary 2. If an objective $\mathbf{L}_{\mathbf{UcoB}}(A)$ defined by a UcoBW A with n states is realized by a Moore machine M with m states, then the strengthened objective $\mathbf{L}_{\mathbf{u,K}}(A)$, with $\mathbf{K}=n\times m$, is also realized by M.

K-Co-Büchi Objectives

Theorem:
Let A a UcoBW with n states and $\mathbf{K = n(n^{2n+1}+1)}$.
Then $\mathbf{L_{UcoB}}(A)$ is realizable iff $\mathbf{L_{U,K}}(A)$ is realizable.

Proof. Back direction is trivial. For the converse:

1/ UcoBW A → det. Parity automaton → Parity game G with

$$|G| = n^{2n+1}+1$$

2/ Parity games admit **memoryless strategies**

3/ Therefore A realizable ⇒ ∃M with |G| states that realizes it

4/ Apply previous Lemma → bound on the number of accepting states

Determinization of UKcoBWs

Σ^ω	Run	"Extended subsets"
	1	{1:0}
¬g	↓	
	2	{2:0}
r	↙ ↘	
	1 3	{1:0,3:0}
¬g	↓ ↓	
	2 4	{2:0,4:1}
r	↓ ↓	
	1 3	{1:0,3:1}
g	↓ ×	
	2	{2:0}
	↓	
…	…	…

Determinization of UKcoBWs

Lemma: UKCWs are determinizable (modulo exponential blow-up)

- **Sketch of Proof**: Let $A = (\Sigma, Q, q_0, \alpha, \Delta, K)$ be a UKCW.
- For each state q, count the maximal number of accepting states visited by runs ending up in q
- States are counting functions F from Q to [-1,0,...,K+1]
- Initial counting function $F_0: q \rightarrow (q_0 \in \alpha)$ if $q = q_0$, -1 otherwise
- Final states are functions F such that $\exists q: F(q) > K$

$$\Delta_d(F, \sigma) : q \rightarrow \max_{(q',\sigma,q)\in\Delta} \{ F(q') + (q \in \alpha) \mid F(q') \neq -1 \}$$

Determinization of UKcoBWs

Lemma: UKCWs are determinizable (modulo exponential blow-up)

- **Sketch of Proof**: Let A = ($\sum$,Q,q_0,α,Δ,K) be a UKCW.
- For each state q, count the maximal number of ac
ending up in q
- States are counting functi
- Initial cou
-

From Det(A,K), it is easy to construct
a safety game G(A,K).

Δ_d(F, $\max_{(q',\sigma,q)\in\Delta}$ { F(q') + (q$\in\alpha$) | F(q')$\neq$-1}

Incremental algorithm

Remember that for all UcoBW A, for all $K_1 \leq K_2$,

$L(A,K_1) \subseteq L(A,K_2) \subseteq L(A)$.

⇒ Incremental Realizability Checking Algorithm:

1. **Input**: an LTL formula Φ, a partition I,O
2. A ← UcoBW with n states equivalent to Φ
3. K ← $n(n^{2n+1}+1)$
4. **for** k=0...**K** **do**
5.   **if** Player I wins then G(A,k) **return** realizable
6. **endfor**
7. **return** unrealizable

Incremental algorithm

Remember that for all UcoBW ... 2,

L(A,K ...

⇒ Incre... n:

... on I,O

... es equivalent to Φ

... K **do**

if Player 1 wins then G(A,k) **return** realizable

6.**endfor**

7.**return** unrealizable

This is not reasonable for unrealizable specification !

Incremental algorithm

Remember that for all UcoBW … 2,
L(A,K …

⇒ Incr … n:

… not reasonable for
… ification !

Solution: run two instances of the algorithm:

1) one that checks realizability of Φ for Player 1
2) one that checks realizability of ¬Φ for Player 2

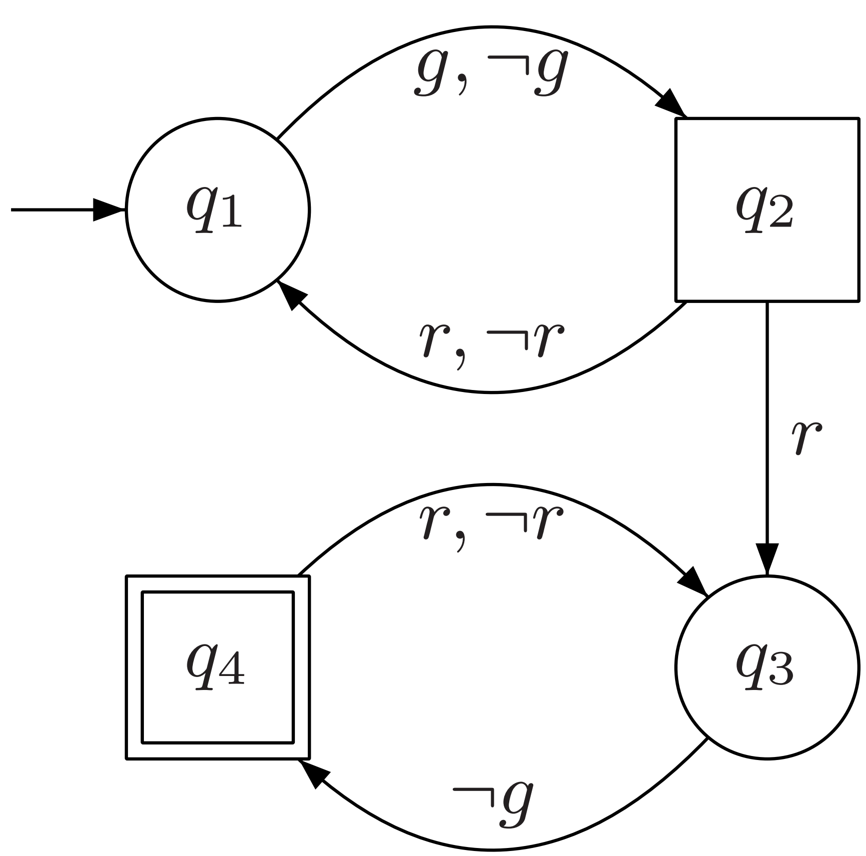
Justified by **determinacy** of ω-regular games !

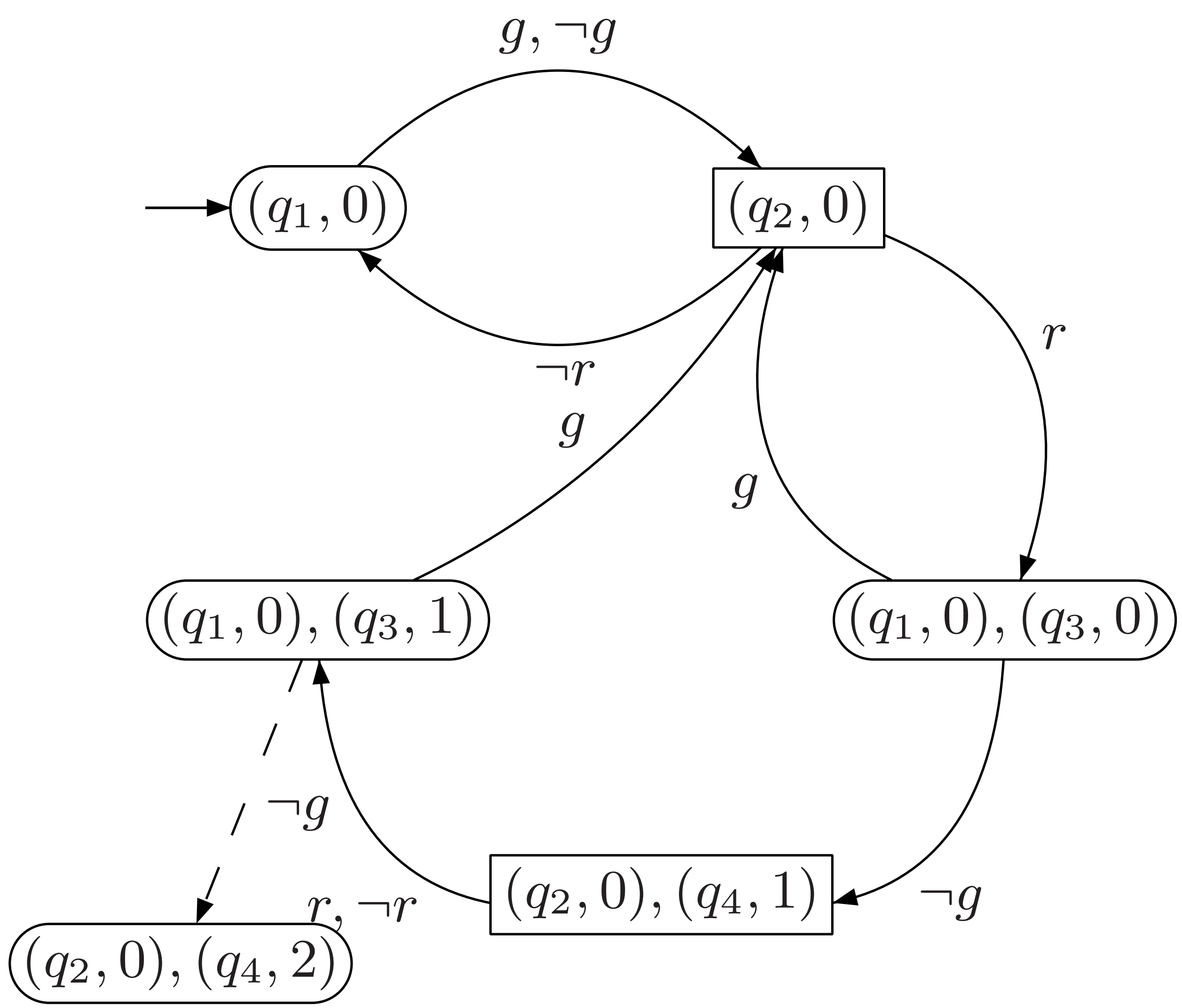
6.**end**
7.**ret**…

Illustration

Example, K=1

$$\Box(r \rightarrow \mathcal{X}(\Diamond g))$$



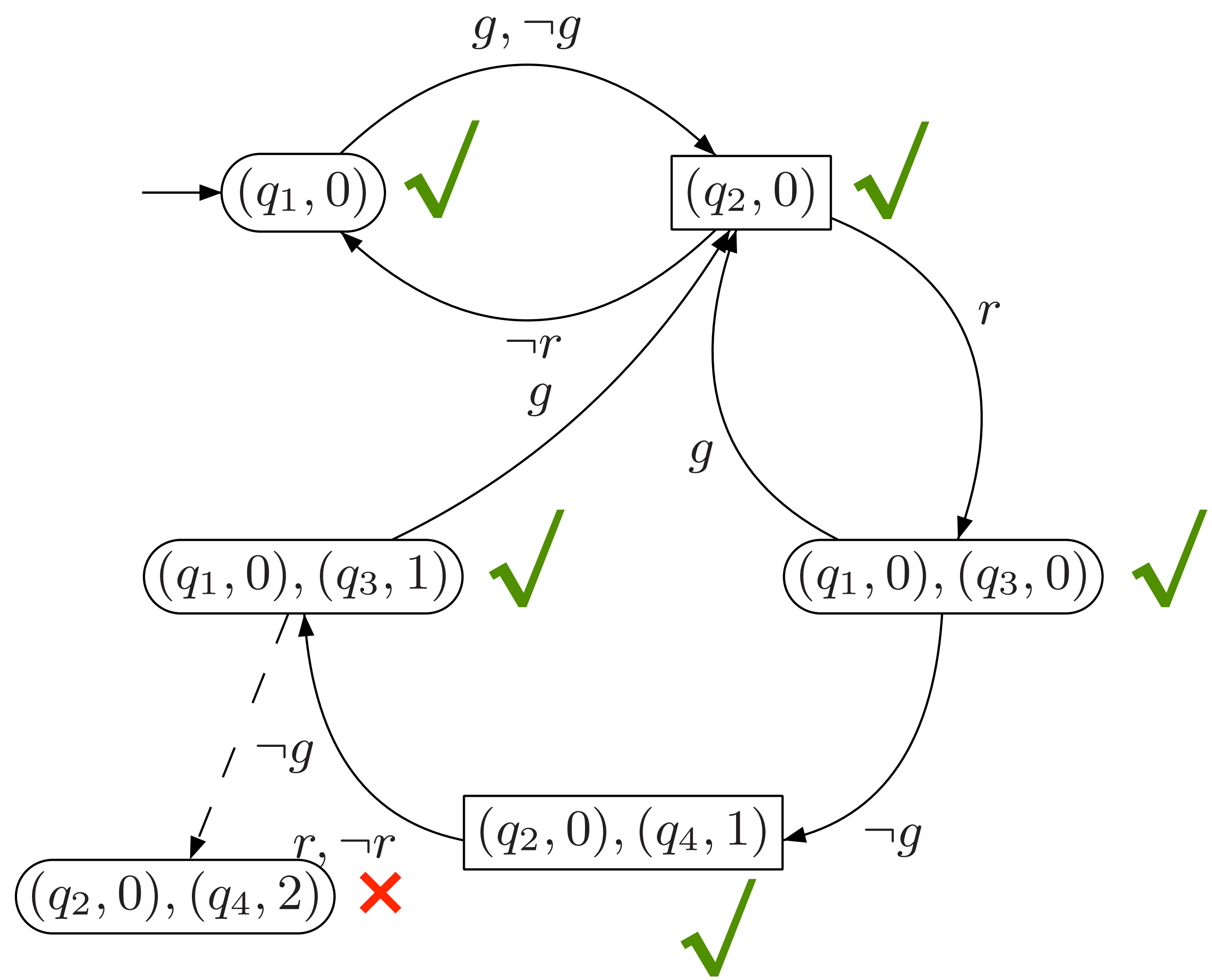


UCW of the formula

Safety game for K=1

Solving the safety game

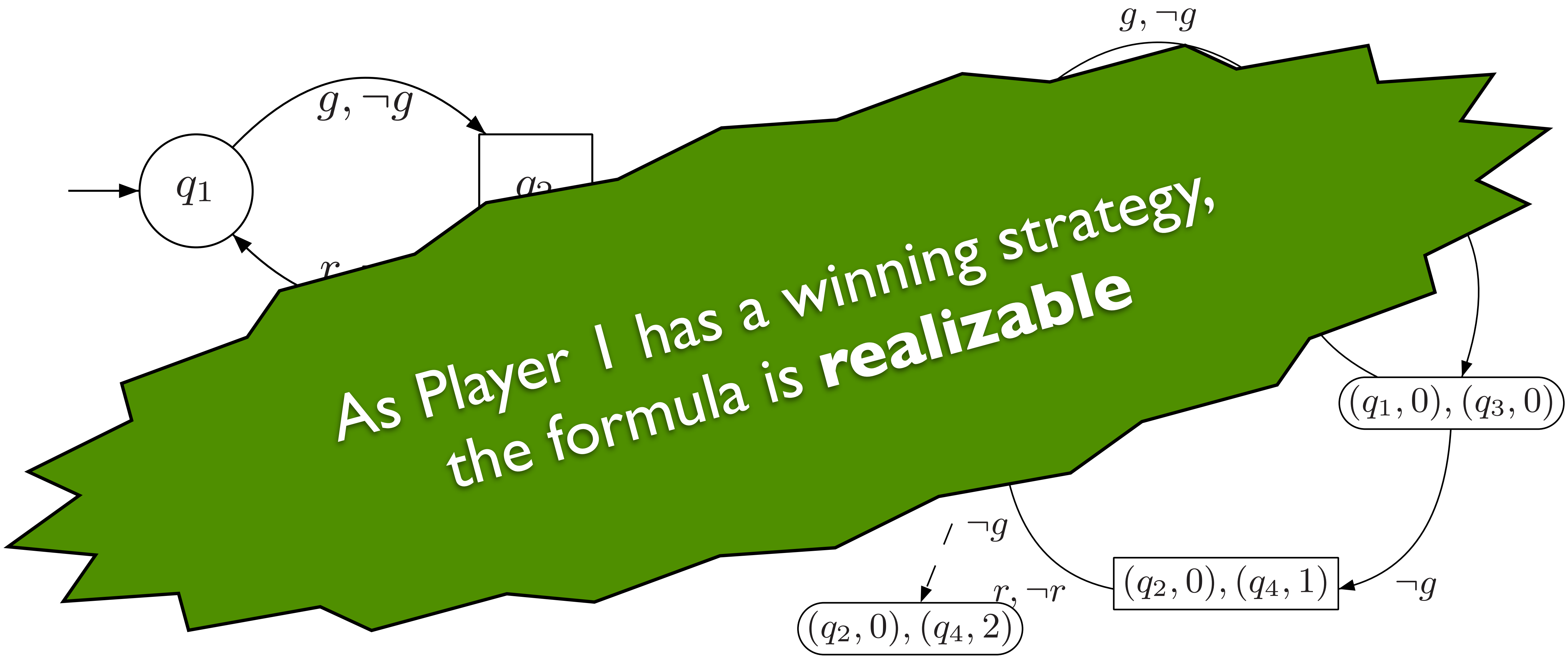
Safety game for K=1



√= winning for player 1

Example, K=1

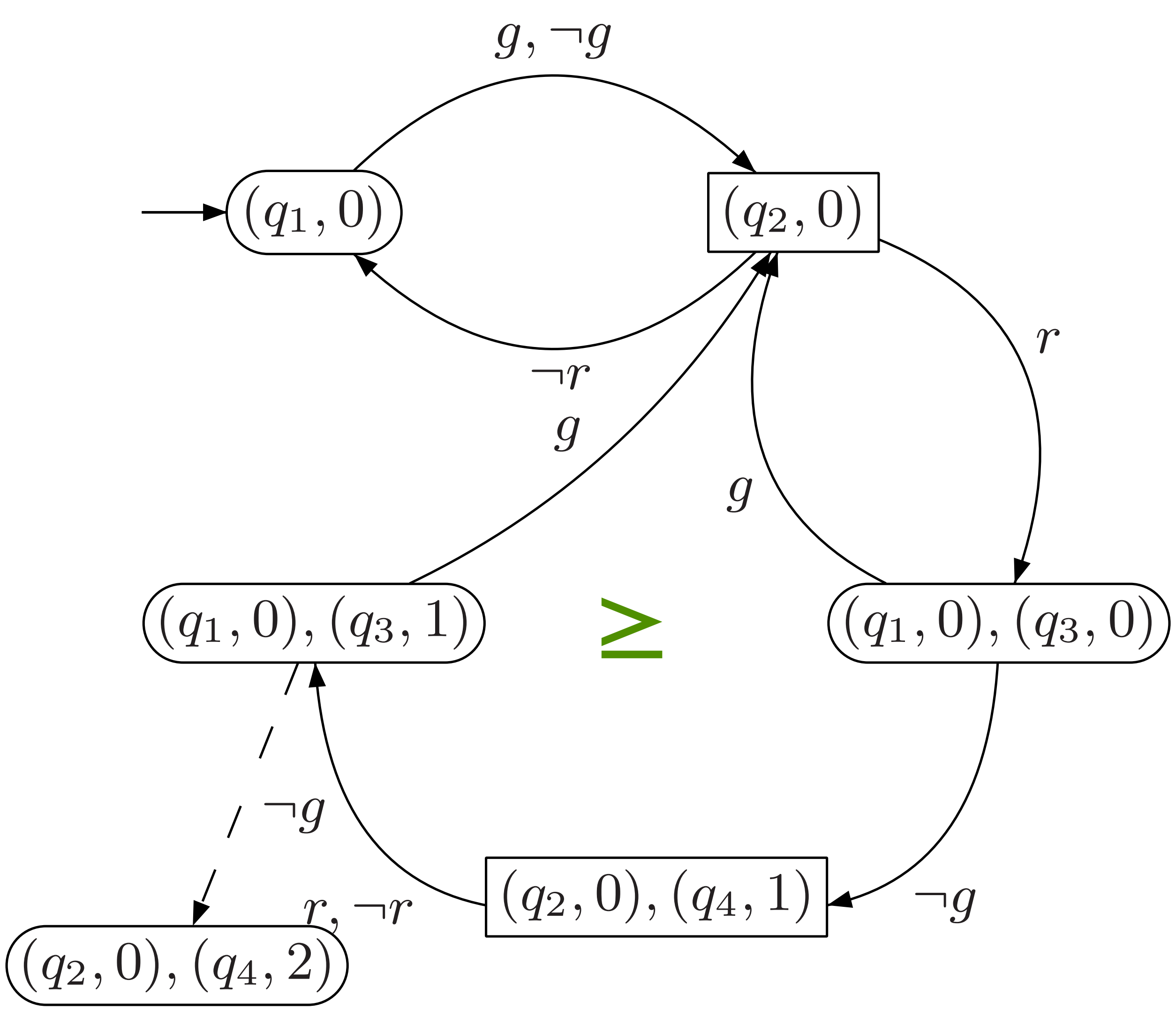
$$\Box(r \rightarrow \mathcal{X}(\Diamond g))$$



UCW of the formula

Safety game for K=1

Structure



Safety game for K=1

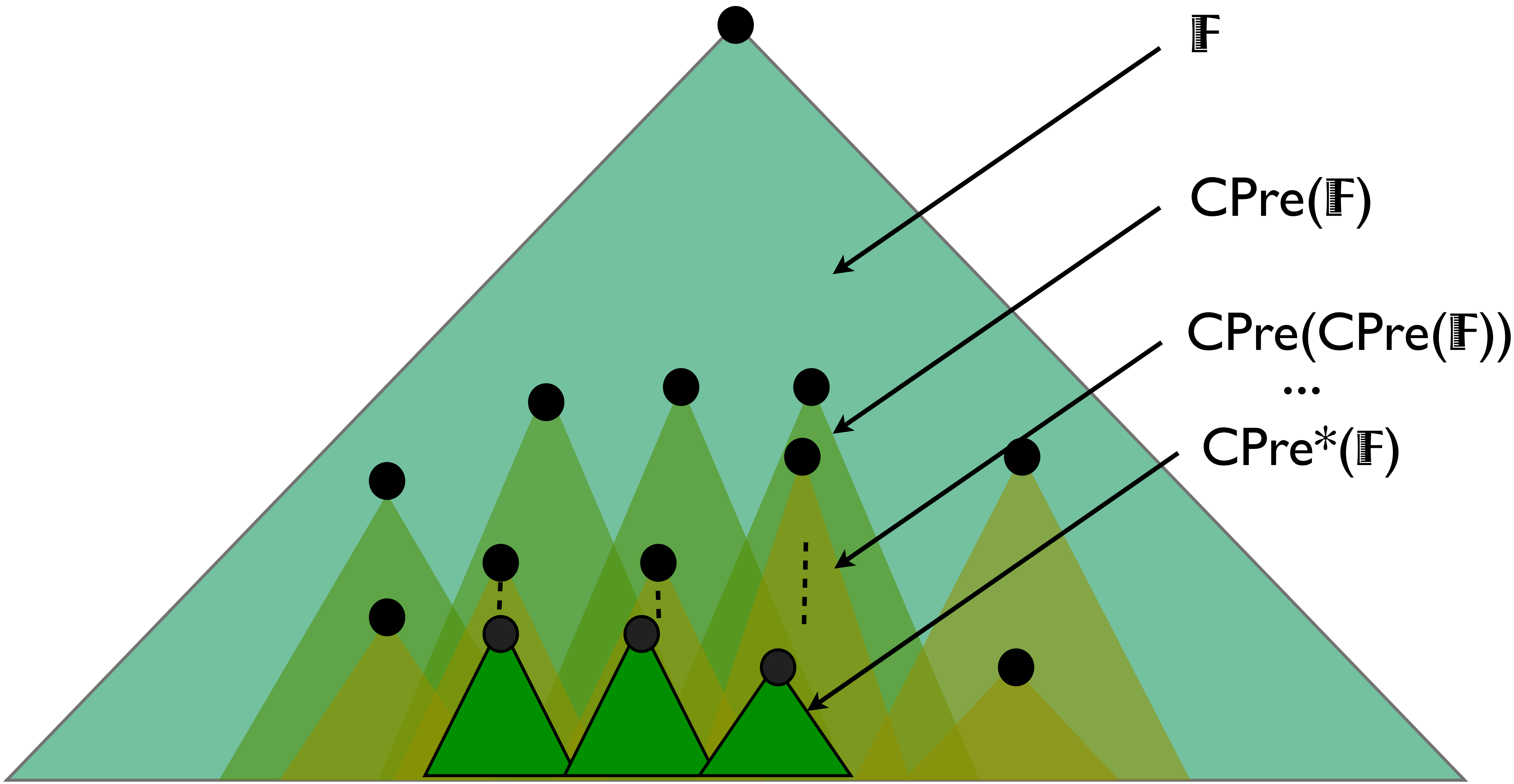
$((q_1,0),(q_3,1)) \geq ((q_1,0),(q_3,0))$

$((q_1,0),(q_3,1))$ winning
implies
$((q_1,0),(q_3,0))$ is winning

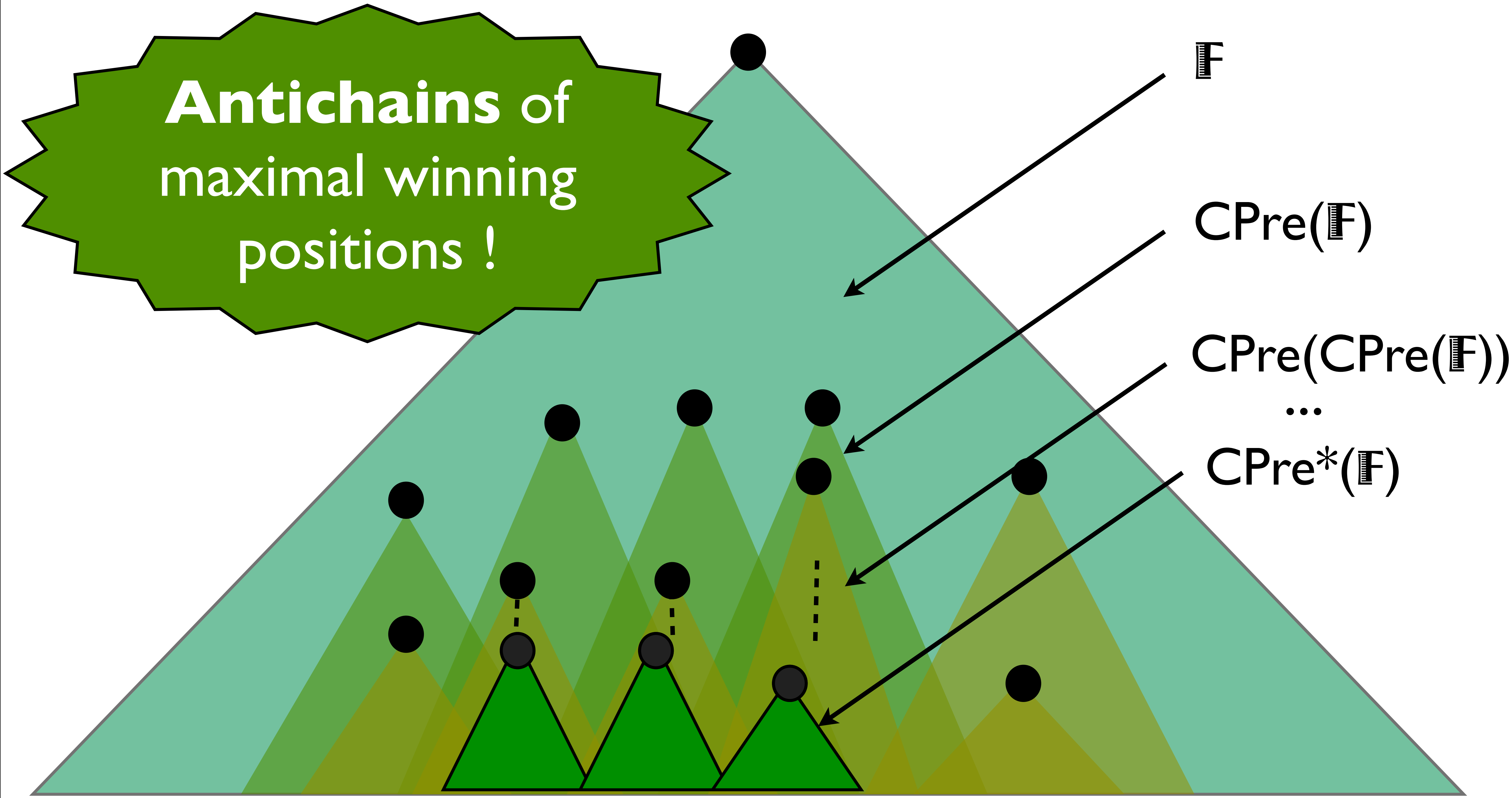
Set of winning positions are $\geq$-downward closed

$\geq$-downward closed sets are canonically represented by their maximal elements

Structure



Structure


Antichains of maximal winning positions !
$\mathbb{F}$
CPre($\mathbb{F}$)
CPre(CPre($\mathbb{F}$))
...
CPre*($\mathbb{F}$)

It works in practice !

- Implemented in Acacia [FJR09] (at ULB)
- .... and with BDDs [Ehlers10] (at U Saarbrucken)
- Acacia handles large LTL formulas (can be several pages long)
- Parameter K is usually very small (K=0,1,2,3).
- Synthesized strategies are very compact

☐ may lead to hardware implementations.

CAV'2009

ATVA'2010

An Antichain Algorithm for LTL Realizability*

Emmanuel Filiot Naiyong Jin Jean-François Raskin

CS, Faculty of Sciences
Université Libre de Bruxelles (U.L.B.), Belgium

Abstract. In this paper, we study the structure of underlying automata based constructions for solving the LTL realizability and synthesis problem. We show how to reduce the LTL realizability problem to a game with an observer that checks that the game visits a bounded number of times accepting states of a universal co-Büchi word automaton. We show that such an observer can be made deterministic and that this deterministic observer has a nice structure which can be exploited by an incremental algorithm that manipulates antichains of game positions. We have implemented this new algorithm and our first results are very encouraging.

1 Introduction

Automata theory has revealed very elegant for solving verification and synthesis problems. A large body of results in computer aided verification can be phrased and solved in this framework. Tools that use those results have been successfully used in industrial context, see [16] for an example. Nevertheless, there is still plenty of research to do and new theory to develop in order to obtain more efficient algorithms able to
larger or broader classes of practical examples. Recently, we and
shown in [4–6, 14, 21] that several automata-based co
erties that can be exploited to improve algorithms on
show how to solve more efficiently the language inclu
tic Büchi automata by exploiting a partial-order that ex
constructions used to solve this problem. Other struct
tionally exploited in [7]. In this paper, we pursue this
automata-based approach to LTL realizability and synthe
is 2EXPTIME-COMPLETE, we show that there are also
with adequate partial-orders that can be exploited to obt
procedure for it.

The realizability problem for an LTL formula ϕ is best seen as a game between two players [13]. Each of the players is controlling a subset of the set P of propositions on which the LTL formula ϕ is constructed. The set of propositions P is partitioned into I the set of *input signals* that are controlled by "Player input" (the environment

Compositional Algorithms for LTL Synthesis

Emmanuel Filiot, Nayiong Jin, and Jean-François Raskin

CS, Université Libre de Bruxelles, Belgium

Abstract. In this paper, we provide two compositional algorithms to solve safety games and apply them to provide compositional algorithms for the LTL synthesis problem. We have implemented those new compositional algorithms, and we demonstrate that they are able to handle full LTL specifications that are orders of magnitude larger than the specifications that can be treated by the current state of the art algorithms.

1 Introduction

Context and motivations The *realizability problem* a game between two players [12]. Given an LTL formula propositions P into I and O, Player 1 starts ions [1], Player 2 responds by o_1 and Player 2 he game is the f the resulting ce a winning ility problem and has been studied works by Pnueli and Rosner [12], been shown 2EXPTIME-C in [13].[2] Despite computation complexity, we believe that it is possible to solve LTL and synthesis problems in practice. We proceed here along recent research efforts that have brought new algorithmic ideas to attack this important problem.

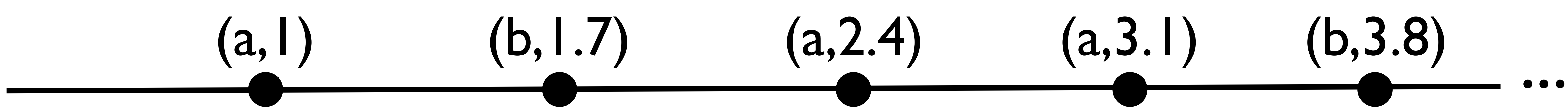
Contributions In this paper, we propose two compositional algorithms to solve the LTL realizability and synthesis problems. Those algorithms rely on previous works where the LTL realizability problem for an LTL formula Φ is reduced to the resolution of a safety game $G(\Phi)$ [5] (a similar reduction was proposed independently in [15] and ap-

See, among others...

Extensions to Timed Specifications

Timed words

Timed word on $\Sigma=\{a,b\}$:



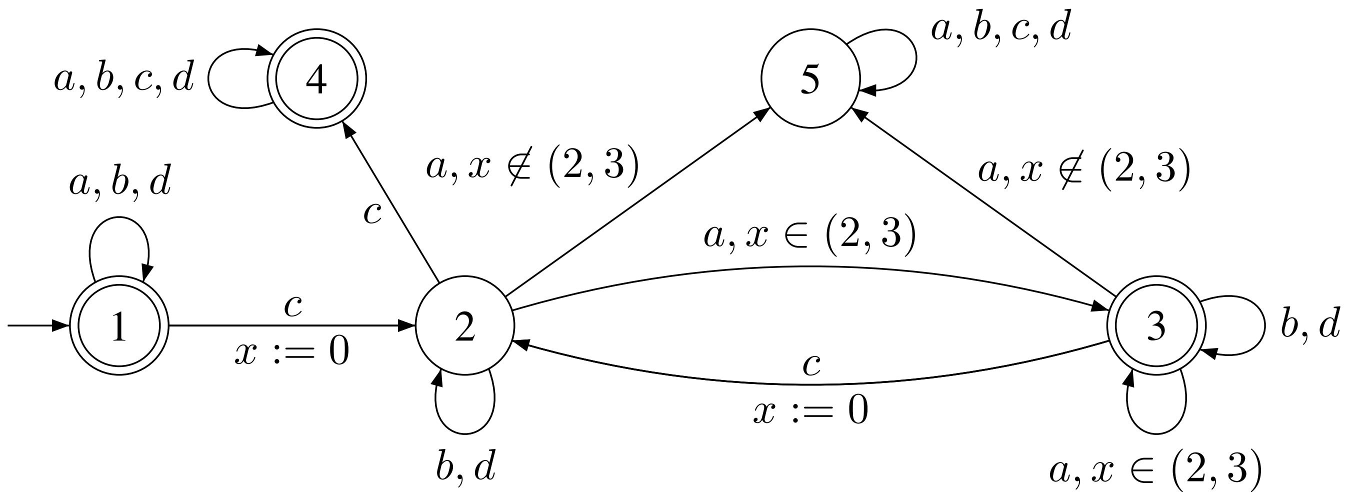
= infinite sequence of elements in $\Sigma \times \mathbb{R}^{\geq 0}$

$(\sigma_0,t_0)\ (\sigma_1,t_1)\ (\sigma_2,t_2)\ \ldots\ (\sigma_n,t_n)\ \ldots$

such that $\sigma_i \in \Sigma$ and $t_i \leq t_{i+1}$, for all $i \in \mathbb{N}$.

Timed Formalisms

Timed automata



Timed extensions of LTL

$\Box (a \rightarrow \Diamond_{=1} b)$

MTL [Koy89,AH89]

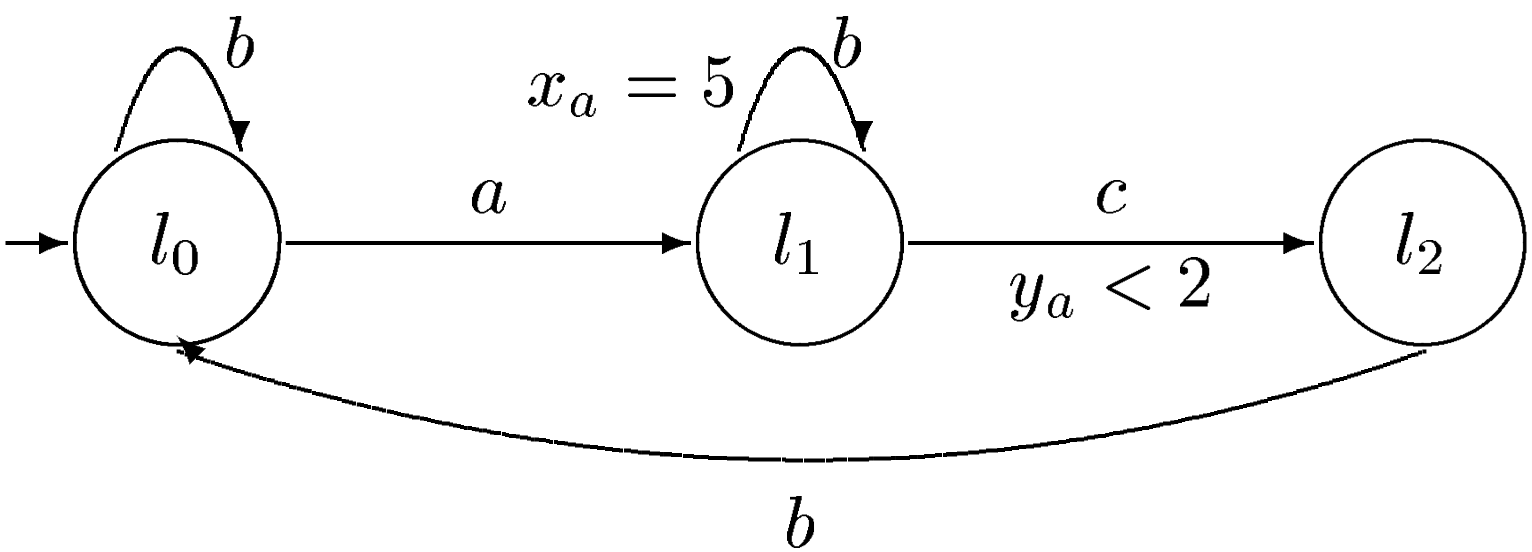
"Every a is followed by a b exactly one time unit later"

Undecidability

- ➡ Language inclusion for TA is undecidable [AD94].
- ➡ Emptiness of universal/alternating automata is undecidable.
- ➡ MTL satisfiability (over infinite timed words) is undecidable [AH93], and so is realizability/synthesis.

➢➢ no hope to apply the previous constructions to those timed formalisms !

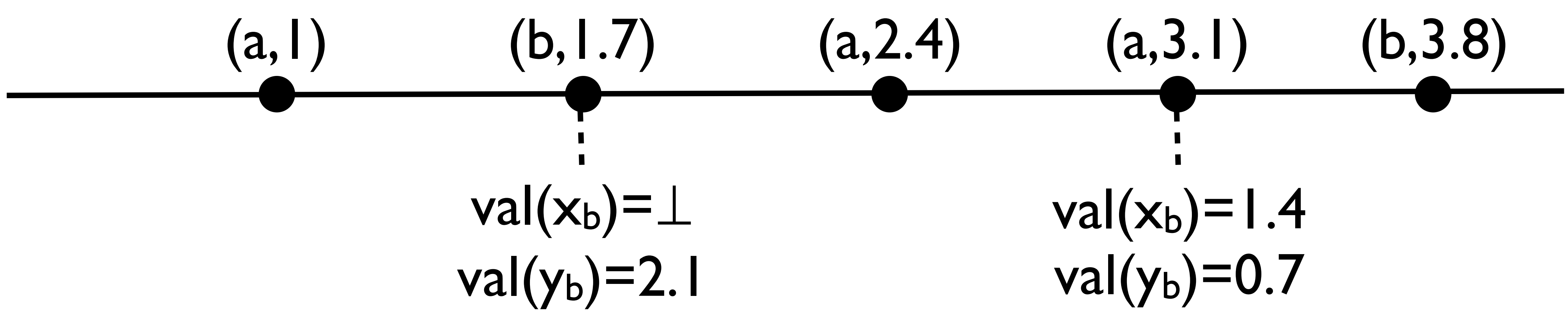
Recovering decidability



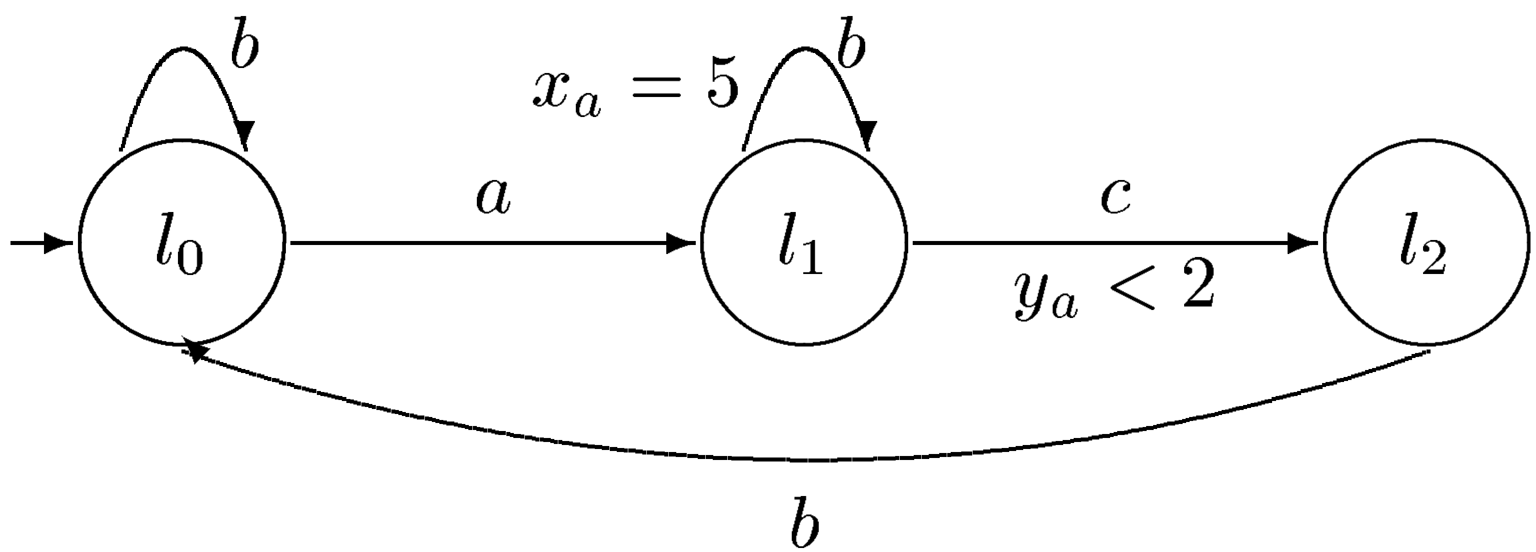
Event clock automata

[AFH99]

Clock are not reset and are associated to events: $\{ x_\sigma, y_\sigma \mid \sigma \in \Sigma \}$
Values of event-clocks are input determined:



Recovering decidability



Event clock automata
[AFH99]

Theorem [AFH99]. Unlike timed automata, event-clock automata are determinizable and their language inclusion problem is PSpace-C.

Recovering decidability

$\Box (a \rightarrow \Diamond_{(0,1]} b)$

MITL [AFH91]

prohibits punctuality in MTL
satisfiability ExpSpaceC [AFH96]
but synthesis undecidable [DGRR09]

$\Box (a \rightarrow \rhd_{=1} b)$

ECL [RS97,HRS98]

refers only to next/previous occ.
satisfiability PSpaceC [RS97,HRS98]
but synthesis undecidable [DGRR09]

$\Box (a \rightarrow \lhd_{=1} b)$

LTL+ $\lhd$ [DGRR09]

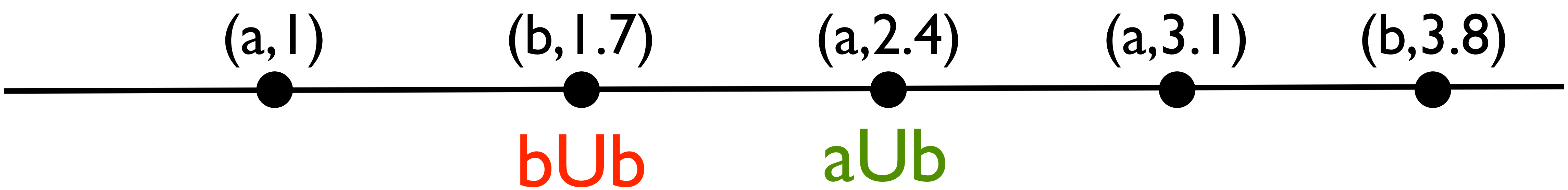
refers only to previous occ.
satisfiability PSpaceC [RS97,HRS98]
and synthesis 2ExpTimeC [DGRR09]

LTL + $\lhd$

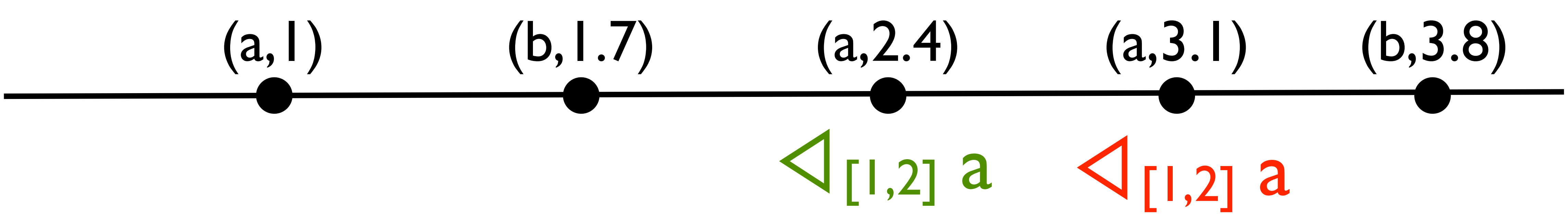
LTL+$\lhd$: $\Phi ::= \sigma \mid \Phi_1 \vee \Phi_2 \mid \Phi_1 \ U\ \Phi_2 \mid \Phi_1 \ U\ \Phi_2 \mid \lhd_I\ \sigma$

with I is an interval of $\mathbb{R}^{\geq 0}$ with integer bounds.

$(w,i) \models \Phi_1\ U\ \Phi_2$ **iff** $\exists j > i \bullet (\ (w,j) \models \Phi_2$ and $\forall k \bullet i < k < j \bullet (w,j) \models \Phi_1\ )$



$(w,i) \models\ \lhd_I\ \sigma$ **iff** $\exists j < i \bullet (w,j) \models \sigma$ and $\forall k \bullet j < k < i \bullet (w,k) \not\models \sigma$ and $t(i)-t(j) \in I$



Timed Games

- A timed game is a 3-tuple $\langle\Sigma_1,\Sigma_2,\mathbf{Win}\rangle$ where:
  - ★ Σ_1 is a finite alphabet of letters that belong to Player 1,
  - ★ Σ_2 belongs to Player 2,
  - ★ and **Win** is a language of timed words over $\Sigma_1 \cup \Sigma_2$.
- A timed game is played during an infinite number of rounds. In each round:
  - ★ Player 1 chooses a pair $(\sigma,t_1)\in\Sigma_1\times\mathbb{R}^{\geq 0}$
  - ★ Player 2 either lets Player 1 play or chooses $(\sigma,t_2)\in\Sigma_2\times\mathbb{R}^{\geq 0}$ with $t_2 \leq t_1$.
- This interaction generates an infinite timed word w.
- Player 1 wins the timed game iff $w \in \mathbf{Win}$.

Timed Strategies

Player 1's strategies: $\lambda_1 : (\Sigma \times \mathbb{R}^{\geq 0})^* \rightarrow (\Sigma_1 \times \mathbb{R}^{\geq 0})$

ex: $\lambda_1((a,0.6),(b,0.9))=(a,0.5)$

then **either** Player 2 let Player 1 play, and we obtain:

$(a,0.6),(b,0.9)(a,1.4)$

or he overtakes Player 1, for example by playing $(b,0.3)$, and we get

$(a,0.6),(b,0.9)(b,1.2)$

➢➢ λ_1 is winning in $\langle \Sigma_1, \Sigma_2, \mathbf{Win} \rangle$ if $\mathrm{Outcome}(\lambda_1) \subseteq \mathbf{Win}$

Realizability problem for LTL+$\lhd$

LTL+$\lhd$ realizability problem

Given a LTL+$\lhd$ spec Φ over the alphabet $\Sigma_1 \cup \Sigma_2$.
Does there exist a strategy λ_1 for Player 1 such that:

λ_1 is winning the timed game $\langle \Sigma_1, \Sigma_2, [\![\Phi]\!] \rangle$?

Example

$$\Sigma_1 = \{grant\}, \qquad \Sigma_2 = \{up, down\}$$

$$\begin{aligned}
\mathsf{Hyp} &\equiv \Box\Big(up \rightarrow (\neg down\, \mathcal{U} (down \wedge \lhd_{\geq 1} up))\Big) \wedge \\
&\quad \Box\Big(down \rightarrow (\neg up\, \mathcal{U} (up \wedge \lhd_{\geq 1} down))\Big) \\
\mathsf{Req}_1 &\equiv \Box\big((down \wedge \lhd_{>2} up) \rightarrow (\neg up\, \mathcal{U}\, grant)\big) \\
\mathsf{Req}_2 &\equiv \Box(grant \rightarrow \neg \lhd_{<3} grant)
\end{aligned}$$

Example

$$\Sigma_1 = \{grant\}, \qquad \Sigma_2 = \{up, down\}$$

$$\begin{aligned} \mathsf{Hyp} \equiv \Box\Big(up \rightarrow (\neg down\, \mathcal{U} (down \wedge \lhd_{\geq 1} up))\Big) \wedge \\ \Box\Big(down \rightarrow (\neg up\, \mathcal{U} (up \wedge \lhd_{\geq 1} down))\Big) \end{aligned}$$

"***Up*** and ***down*** events alternate. Distance between ***up*** and ***down*** is at least 1 t.u."

$$\mathsf{Req}_1 \equiv \Box\big((down \wedge \lhd_{>2} up) \rightarrow (\neg up\, \mathcal{U}\, grant)\big)$$

"If ***down*** follows ***up*** with at least 2 t.u. then it should be ***grant***ed before next ***up***"

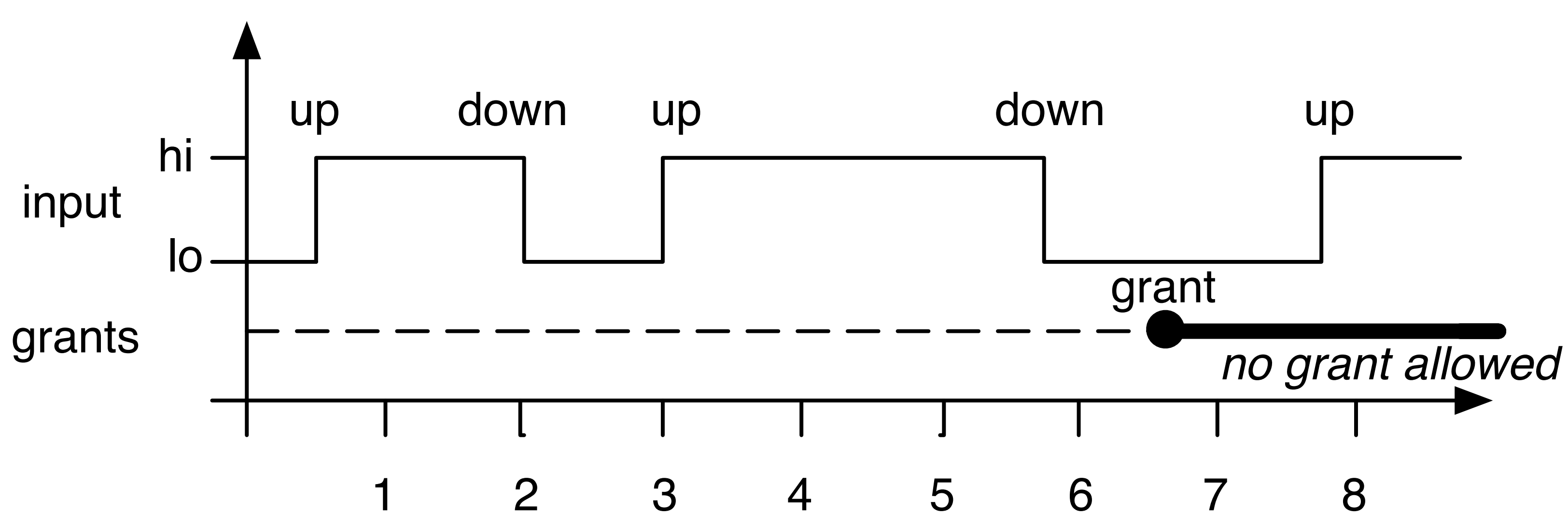
$$\mathsf{Req}_2 \equiv \Box(grant \rightarrow \neg \lhd_{<3} grant)$$

"Two ***grant*** events should be at least 3 t.u. apart"

Example

$$\Sigma_1 = \{grant\}, \qquad \Sigma_2 = \{up, down\}$$

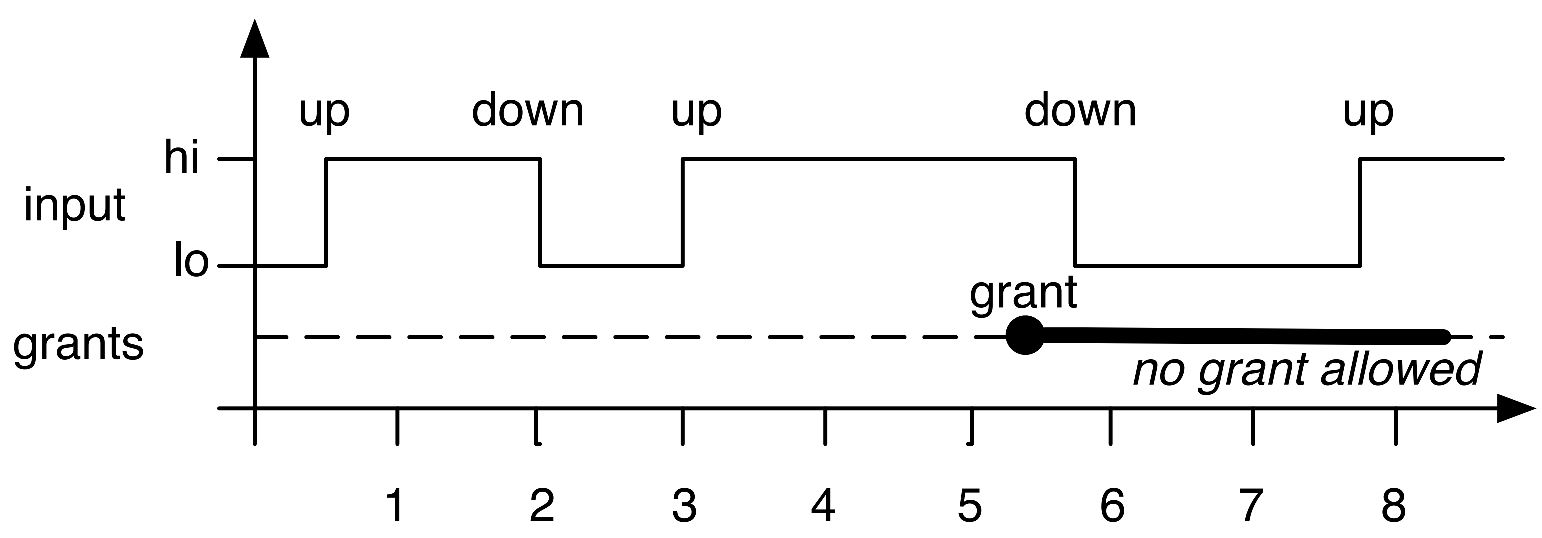
$$\begin{aligned}
\mathsf{Hyp} &\equiv \Box\Big(up \to \big(\neg down\,\mathcal{U}(down \wedge \lhd_{\geq 1}\, up)\big)\Big) \wedge \\
&\quad\ \Box\Big(down \to \big(\neg up\,\mathcal{U}(up \wedge \lhd_{\geq 1}\, down)\big)\Big) \\
\mathsf{Req}_1 &\equiv \Box\big((down \wedge \lhd_{>2}\, up) \to (\neg up\,\mathcal{U}\, grant)\big) \\
\mathsf{Req}_2 &\equiv \Box(grant \to \neg \lhd_{<3}\, grant)
\end{aligned}$$



Example

$\Sigma_1 = \{grant\}$, $\Sigma_2 = \{up, down\}$

$$\mathsf{Hyp} \equiv \Box\Big(up \rightarrow (\neg down\, \mathcal{U} (down \wedge \lhd_{\geq 1} up))\Big) \wedge$$
$$\Box\Big(down \rightarrow (\neg up\, \mathcal{U} (up \wedge \lhd_{\geq 1} down))\Big)$$
$$\mathsf{Req}_1 \equiv \Box\big((down \wedge \lhd_{>2} up) \rightarrow (\neg up\, \mathcal{U}\, grant)\big)$$
$$\mathsf{Req}_2 \equiv \Box(grant \rightarrow \neg \lhd_{<3} grant)$$



Ingredients for Safraless procedure

(1) A translation from LTL+$\triangleleft$ to a class of **universal** timed automata

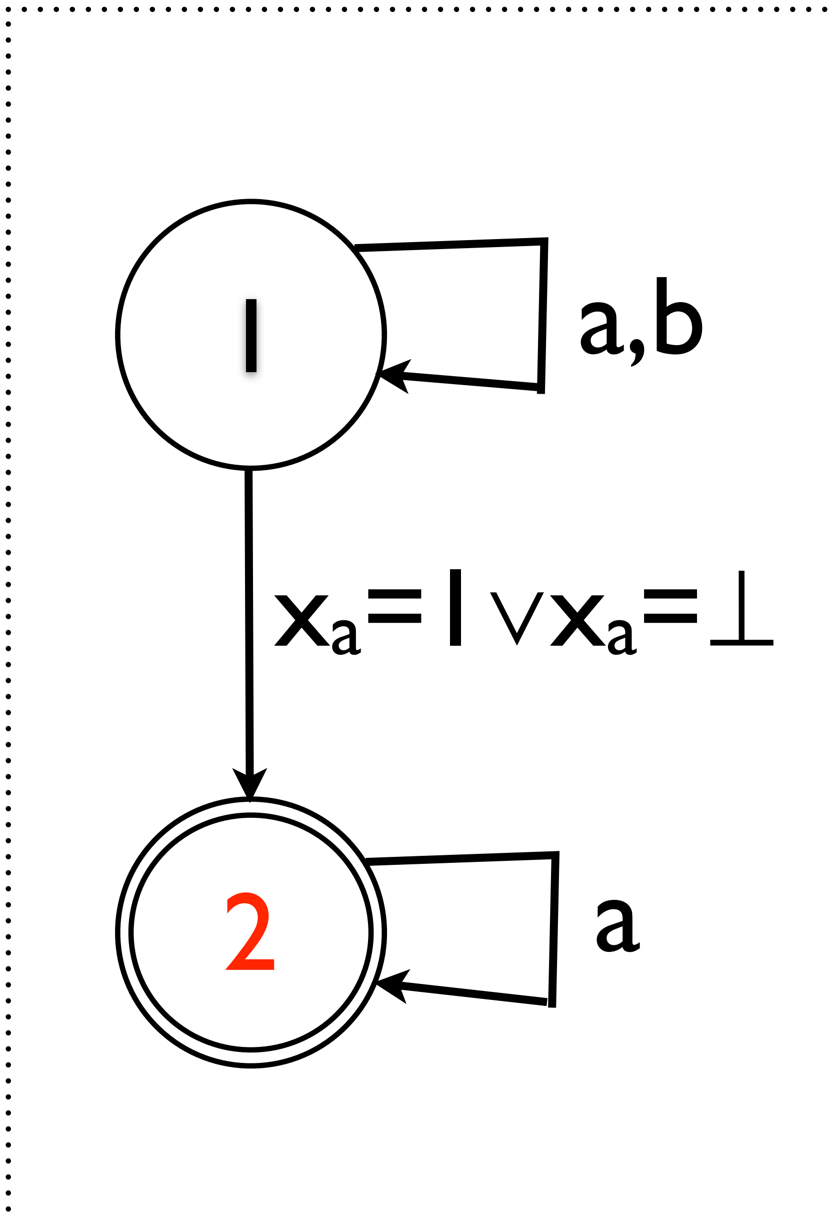
(2) A **bound** on the memory needed for winning realizable LTL+$\triangleleft$ specifications

(3) A translation to **timed safety games**

Ingredient 1

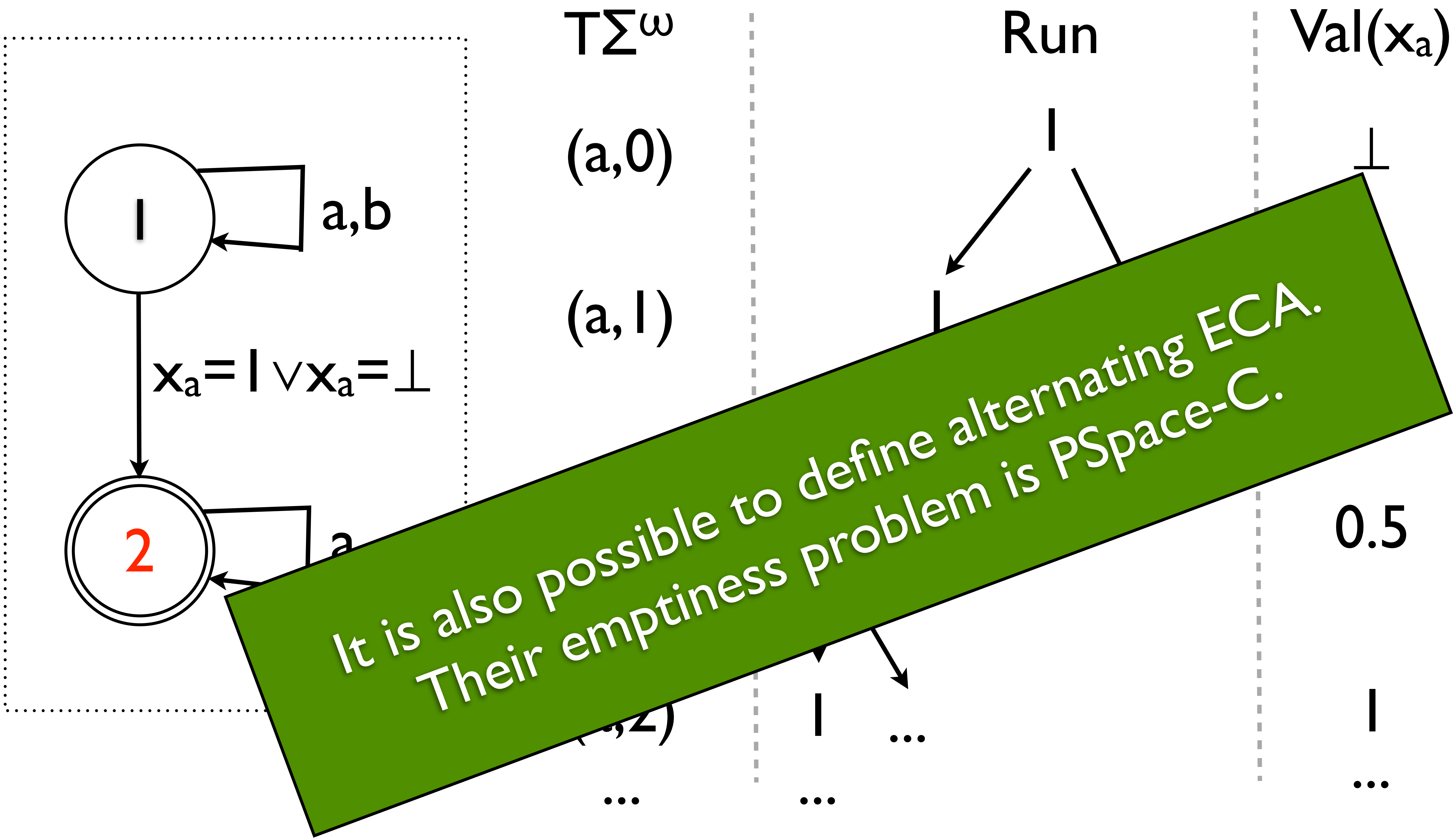
A Class of Universal Timed Automata

UniversalPastECA with coB a.c.

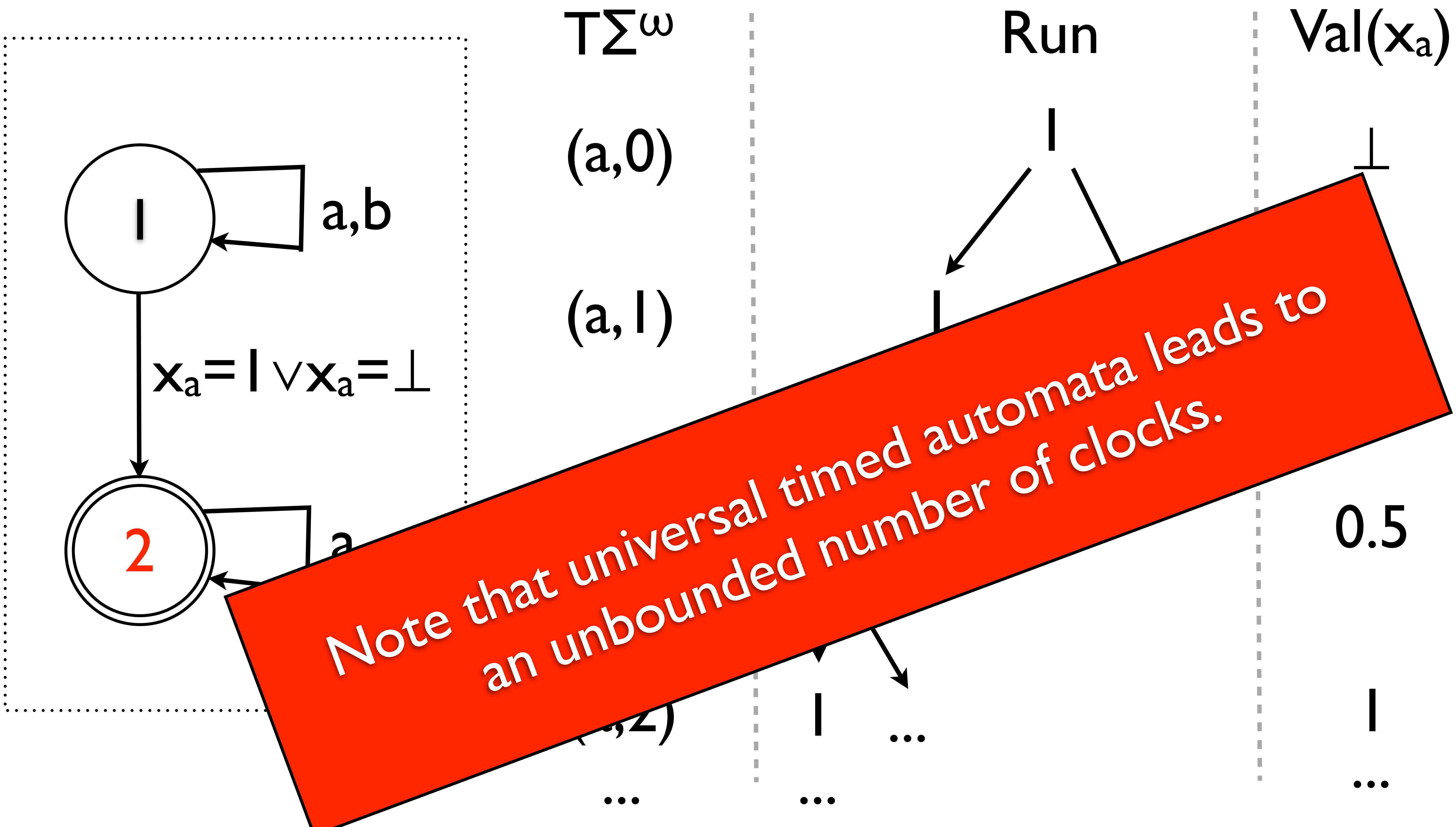


$T\Sigma^\omega$	Run	$Val(x_a)$
(a,0)	1	$\perp$
(a,1)	1 2	1
(b,1.5)	1 2 2	0.5
(a,2)	1 ...	1
...	...	...

UniversalPastECA with coB a.c.



UniversalPastECA with coB a.c.



Note that universal timed automata leads to an unbounded number of clocks.

Ingredient 2

Bounding memory

Region Games

- A **region game** is a 4-uple $\langle \Sigma_1, \Sigma_2, c_{max}, W \rangle$ where $c_{max} \in \mathbb{N}$ and $W \subseteq (\Sigma_1 \cup \Sigma_2) \times \mathrm{Reg}(\mathbb{H}_\Sigma, c_{max})$
  - ★ $\mathbb{H}_\Sigma$ is the set of history clocks over Σ
  - ★ $\mathrm{Reg}(\mathbb{H}_\Sigma, c_{max})$ is the set of regions for clocks in $\mathbb{H}_\Sigma$ and maximal constant c_{max}.
- A region game is played in rounds.
  - ★ In each round Pl. 1 proposes a pair (σ, r) where $\sigma \in \Sigma_1$ and $r_{current} \leq_{t.s.} r$.
  - ★ Then, either Pl. 2 lets Pl. 1 play, or plays (σ', r') s.t. $\sigma' \in \Sigma 2$ and $r_{current} \leq_{t.s.} r' \leq_{t.s.} r$.
- Such an interaction generate an infinite word over the alphabet $(\Sigma_1 \cup \Sigma_2) \times \mathrm{Reg}(\mathbb{H}_\Sigma, c_{max})$.

Region Games

Theorem

Let A be a universal PastECA with maximal constant cmax.

Player 1 has a winning strategy in the timed game $G=\langle\Sigma_1,\Sigma_2,L_{coB}(A)\rangle$

iff

Player 1 has a winning strategy in the region game $GR=\langle\Sigma_1,\Sigma_2,cmax,L_{coB}(\mathbf{Rg}(A))\rangle$.

Region Games

Theorem

Let A be a universal PastECA
with maximal constant cmax.

Player 1 has a winning strategy in
the timed game $G=\langle\Sigma_1,\Sigma_2,L_{coB}(A)\rangle$

iff

Player 1 has a winning strategy in
the region game $GR=\langle\Sigma_1,\Sigma_2,cmax,L_{coB}(\mathbf{Rg}(A))\rangle$.

Syntactic
Transformation

Region Games

Theorem

Let A be a universal PastECA with maximal constant cmax.

Player 1 has a winning strategy in the timed game $G=\langle\Sigma_1,\Sigma_2,L_{\kappa_{coB}}(A)\rangle$

↑

iff

Player 1 has a winning strategy in the region game $GR=\langle\Sigma_1,\Sigma_2,cmax,L_{\kappa_{coB}}(\mathbf{Rg}(A))\rangle$.

↑

Bounding the visits to accepting states

- Regions games = regular games.
- To win $\langle \Sigma_1, \Sigma_2, cmax, L_{UcoB}(Rg(A)) \rangle$, Player 1 needs a memory which is bounded by $(2n^{n+1}n! + n) \times |Reg(\mathbb{H}_{\Sigma}, c_{max})|$.

Bounding the visits to accepting states

Theorem

Let A be a universal PastECA with maximal constant cmax.
Let $\mathbf{K} = \mathbf{(\ 2n^{n+1}n! + n\ ) \times |Reg(\mathbb{H}_{\Sigma}, c_{max})|}$

Player I has a winning strategy in the timed game $G=\langle\Sigma_1,\Sigma_2,L_{coB}(A)\rangle$

iff

Player I has a winning strategy in the region game $GR=\langle\Sigma_1,\Sigma_2,cmax,L_{coB}(Rg(A))\rangle$

iff

Player I has a winning strategy in the region game $GR=\langle\Sigma_1,\Sigma_2,cmax,L_{\mathbf{K}coB}(Rg(A))\rangle$

iff

Player I has a winning strategy in the timed game $GR=\langle\Sigma_1,\Sigma_2,L_{\mathbf{K}coB}(A)\rangle$

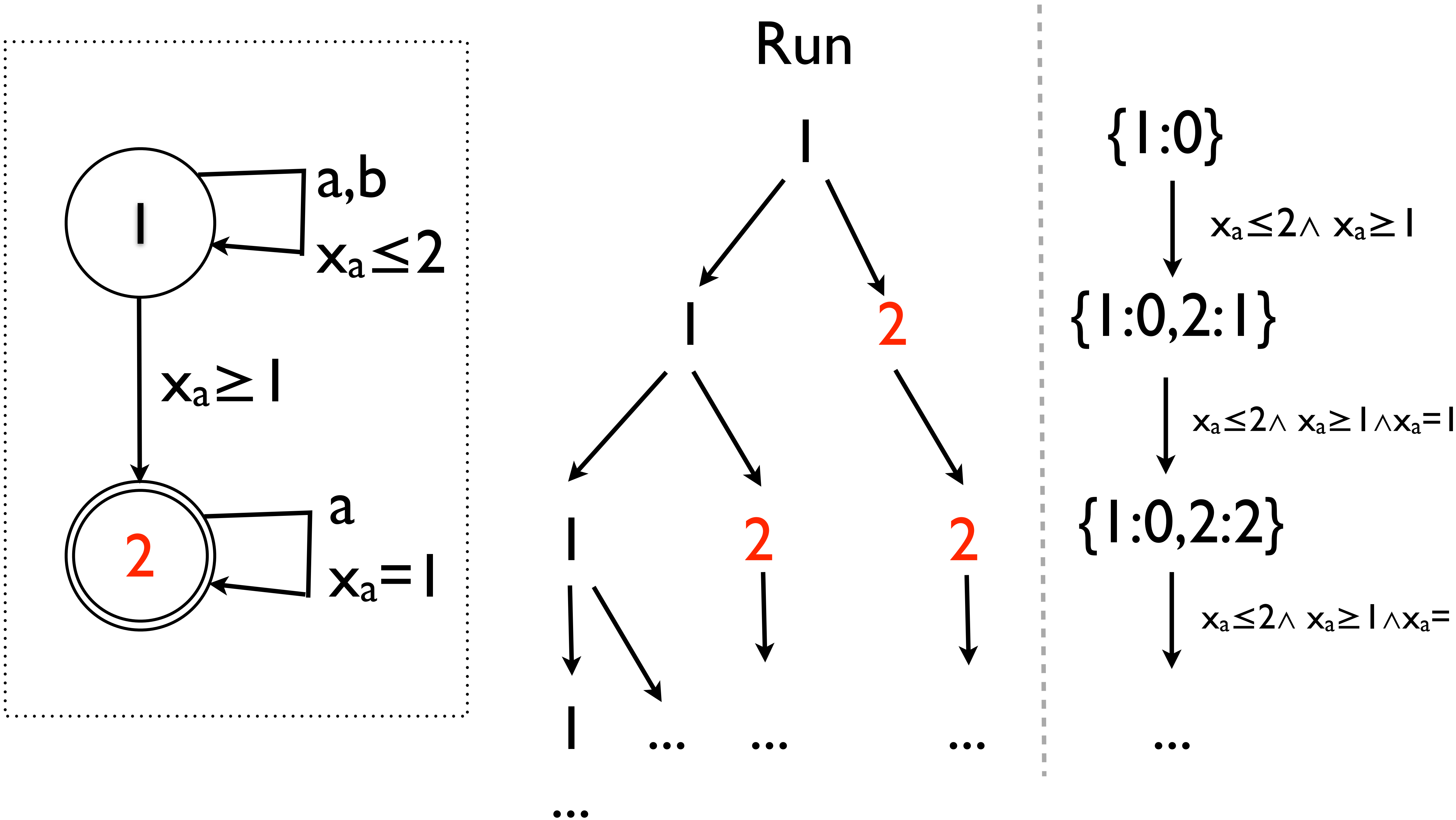
Ingredient 3

Timed Safety Games

Determinization of PastK$_{\text{UcoB}}$ECA

- "Counting subset construction" can be applied directly on Past$_{\text{UcoB}}$ECA.
- **No** need to construct the region automaton.

Determinization of PastK$_{UcoB}$ECA



Determinization of PastK$_{\mathrm{UcoB}}$ECA

- "Counting subset construction" can be applied directly on PastK$_{\mathrm{UcoB}}$ECA.
- No need to construct the region automaton.

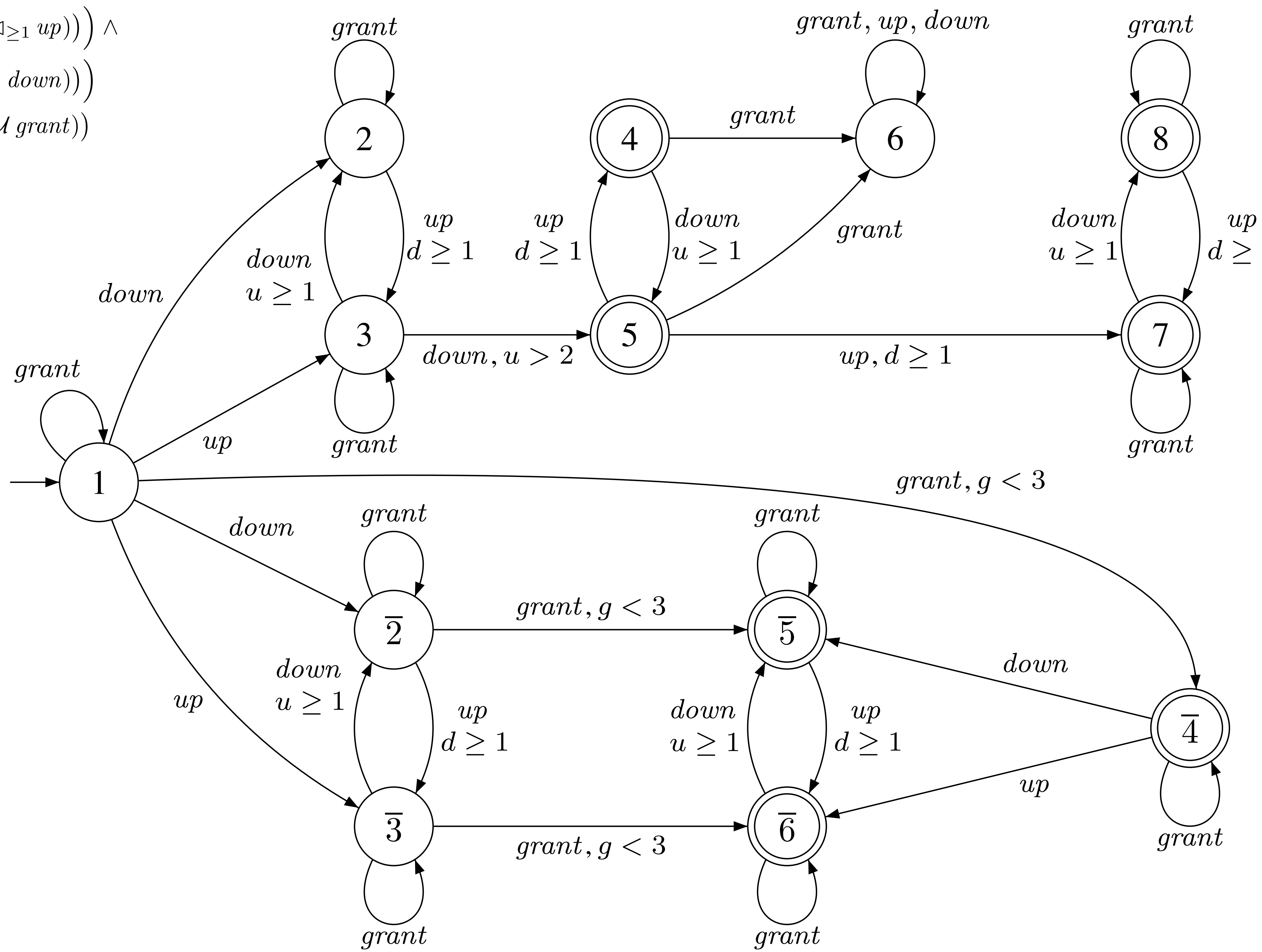
As deterministic PastECA are TA, we can use UppAal TiGa to analyze the underlying timed safety game.

Illustration

Illustration

$$\mathsf{Hyp} \equiv \Box\Big(up \rightarrow \big(\neg down\,\mathcal{U}(down \wedge \lhd_{\geq 1} up)\big)\Big) \wedge$$
$$\Box\Big(down \rightarrow \big(\neg up\,\mathcal{U}(up \wedge \lhd_{\geq 1} down)\big)\Big)$$
$$\mathsf{Req}_1 \equiv \Box\big((down \wedge \lhd_{>2} up) \rightarrow (\neg up\,\mathcal{U}\,grant)\big)$$
$$\mathsf{Req}_2 \equiv \Box(grant \rightarrow \neg \lhd_{<3} grant)$$

using extension of classical constructions



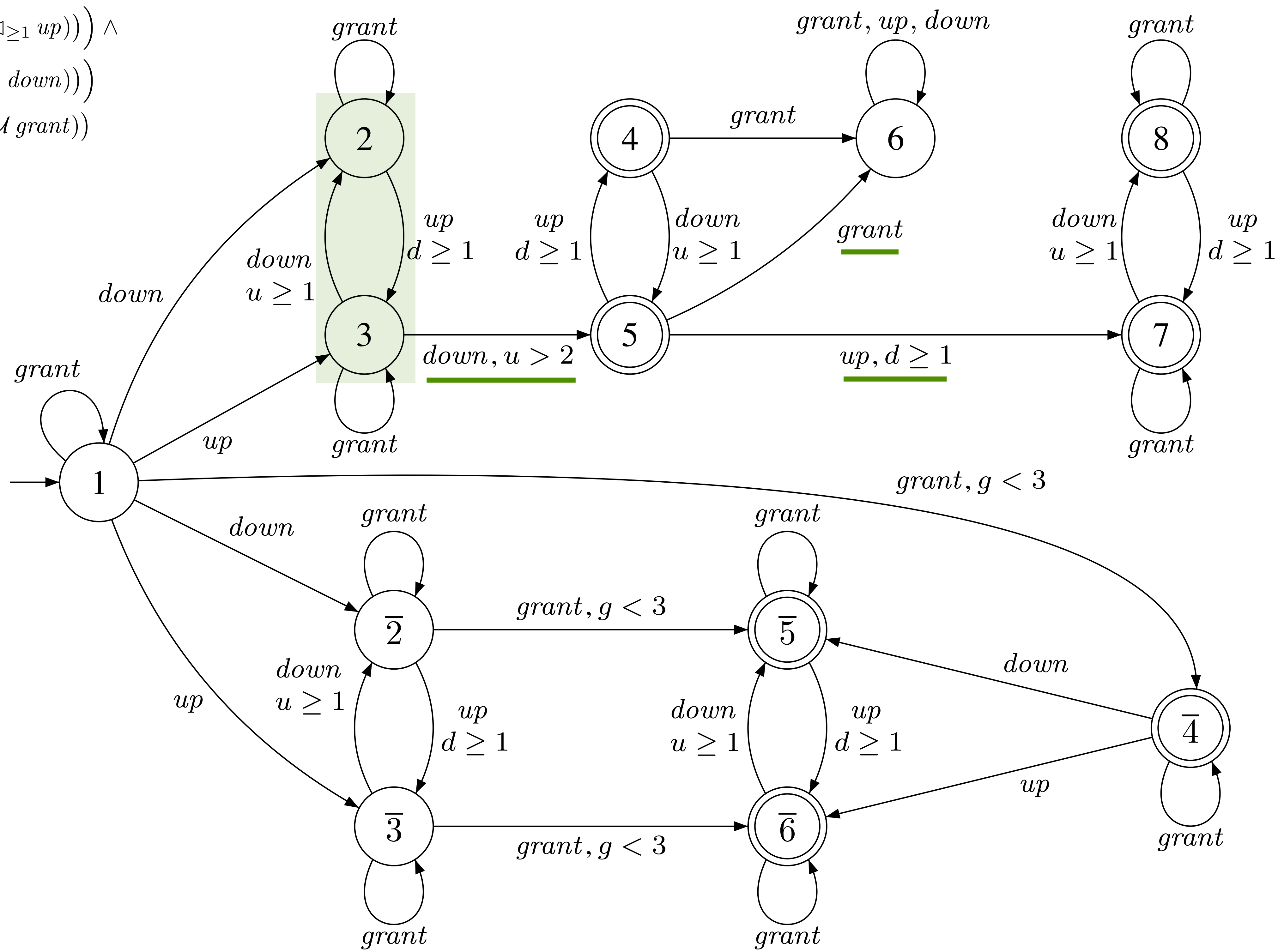
Illustration

$\mathsf{Hyp} \equiv \Box\Big(up \to \big(\neg down\,\mathcal{U}(down \wedge \lhd_{\geq 1}\, up)\big)\Big) \wedge$
$\Box\Big(down \to \big(\neg up\,\mathcal{U}(up \wedge \lhd_{\geq 1}\, down)\big)\Big)$

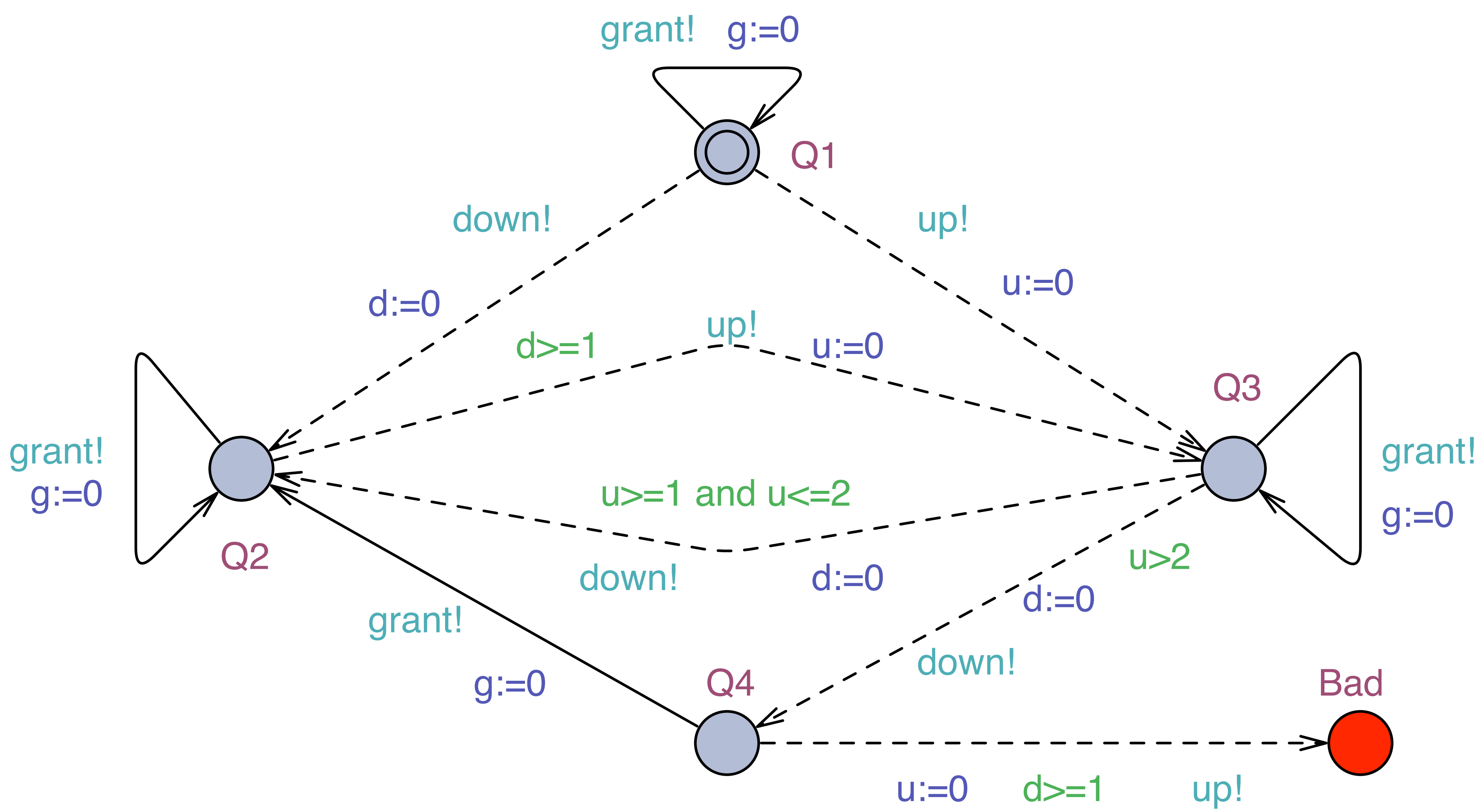
$\mathsf{Req}_1 \equiv \Box\big((down \wedge \lhd_{>2}\, up) \to (\neg up\,\mathcal{U}\, grant)\big)$

$\mathsf{Req}_2 \equiv \Box(grant \to \neg \lhd_{<3}\, grant)$

using extension of classical constructions

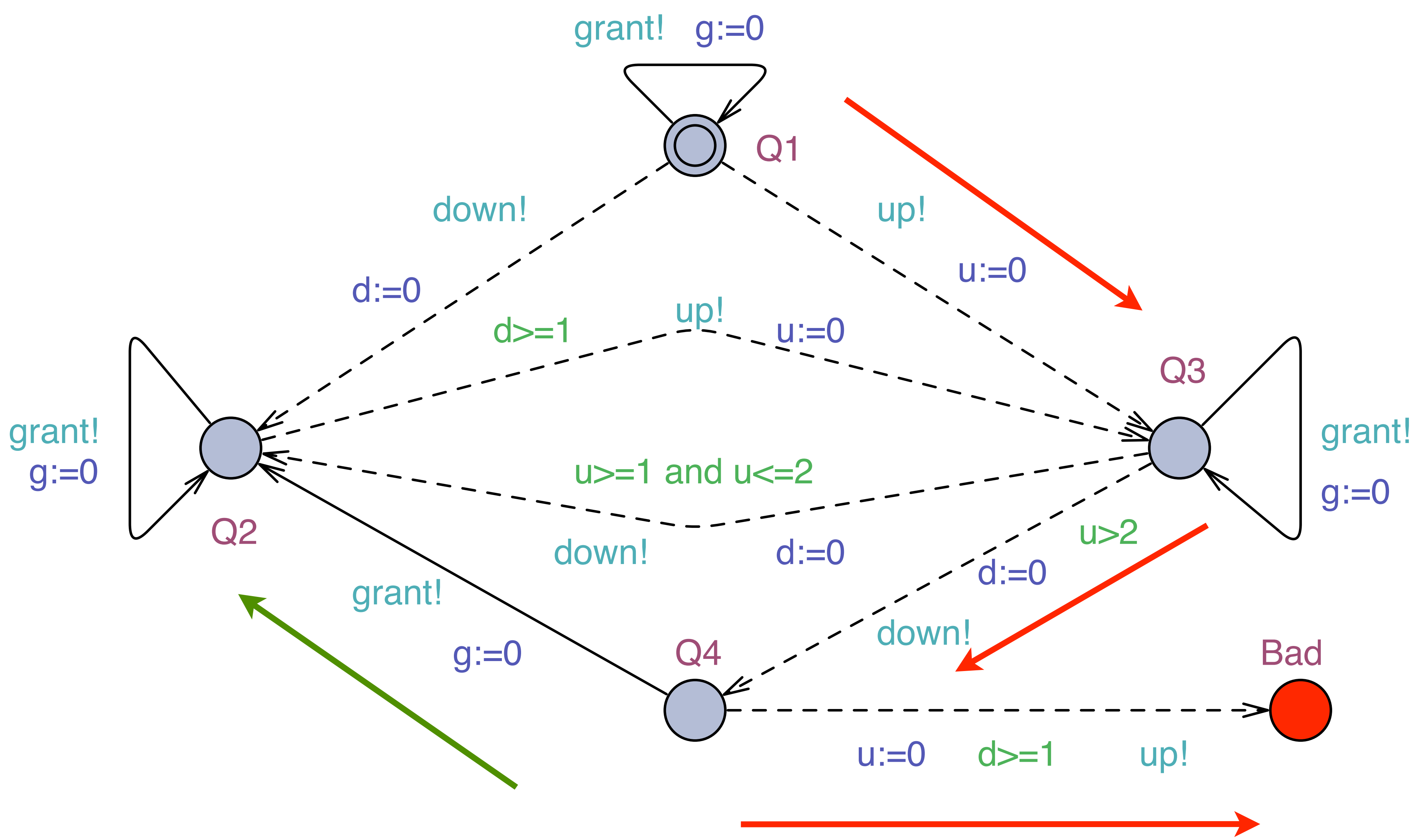


Illustration



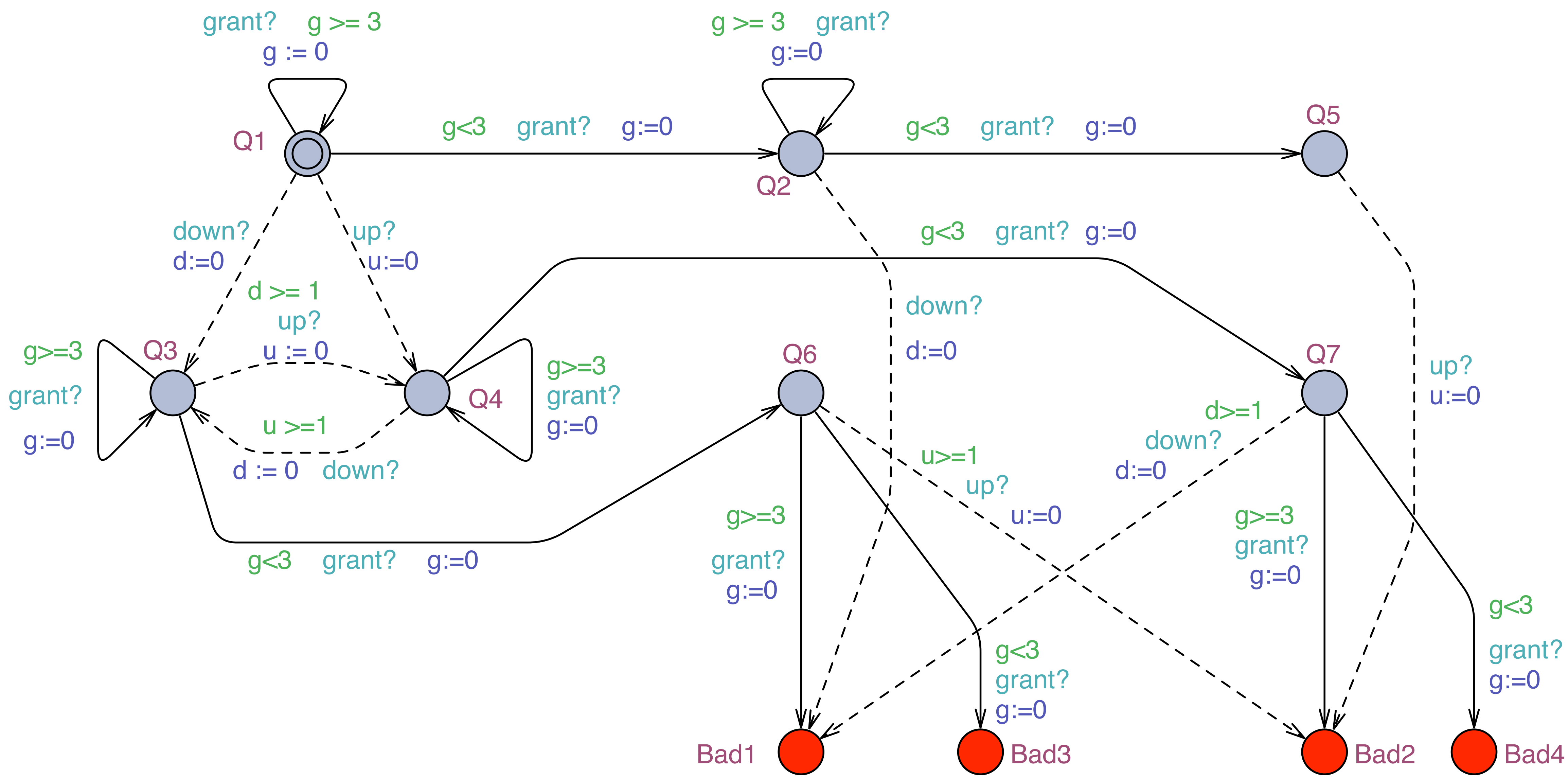
For K=1

Illustration



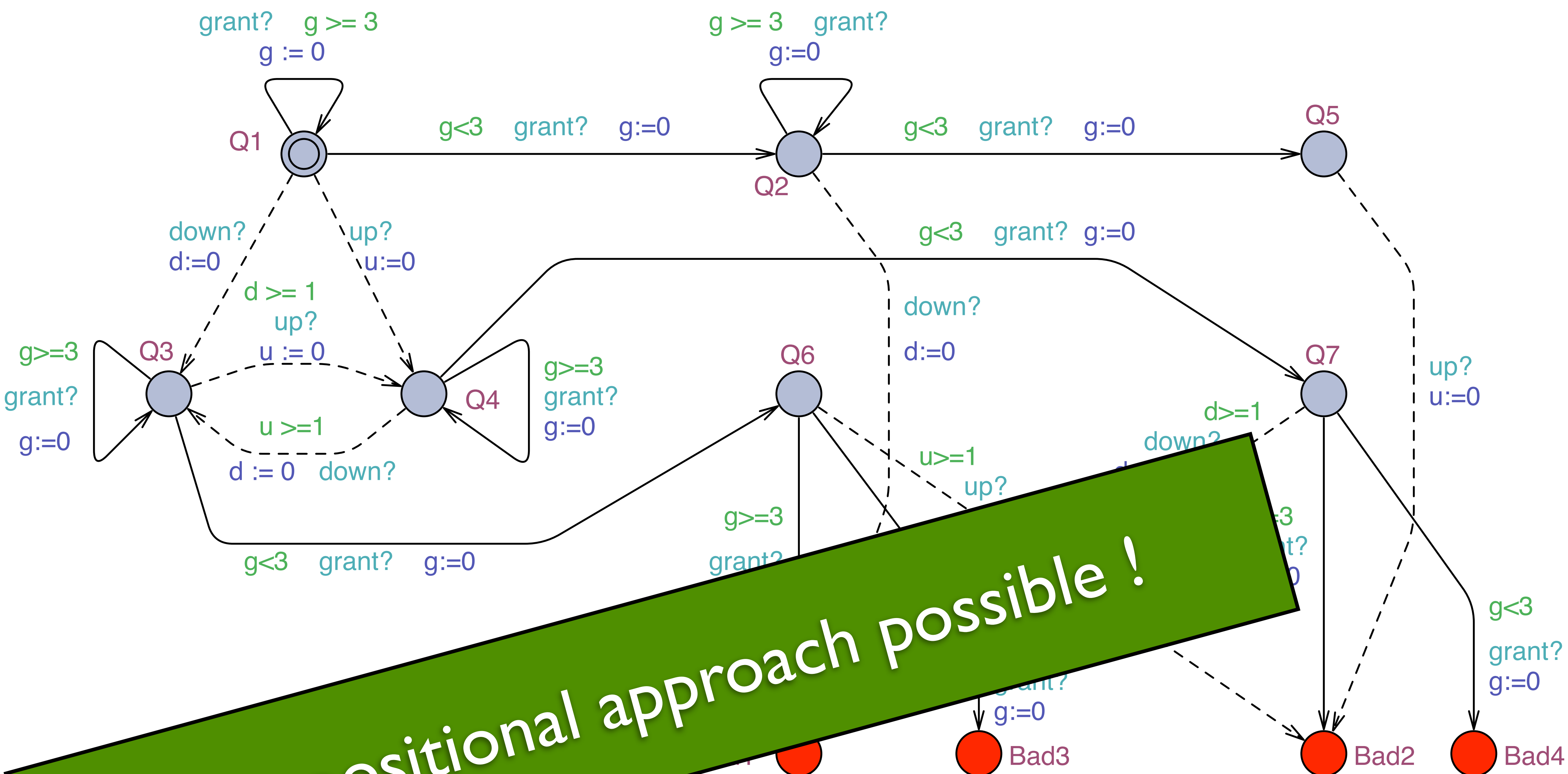
For K=1

Illustration



For K=1

Illustration



Compositional approach possible !

For K=1

Illustration

In this game, Player 1 has a winning strategy

$\Box$ the formula

$$
\begin{aligned}
\mathsf{Hyp} &\equiv \Box\Big(up \rightarrow (\neg down\,\mathcal{U}(down \wedge \triangleleft_{\geq 1}\, up))\Big) \wedge \\
&\ \Box\Big(down \rightarrow (\neg up\,\mathcal{U}(up \wedge \triangleleft_{\geq 1}\, down))\Big) \\
\mathsf{Req}_1 &\equiv \Box\big((down \wedge \triangleleft_{>2}\, up) \rightarrow (\neg up\,\mathcal{U}\, grant)\big) \\
\mathsf{Req}_2 &\equiv \Box(grant \rightarrow \neg \triangleleft_{<3}\, grant)
\end{aligned}
$$

is realizable

UppAal-TiGa can provide a winning strategy.

Conclusion

- ★ Safraless approaches makes LTL synthesis practical
- ★ ... this can be smoothly extended to LTL+$\lhd$
- ★ Existing tools like UppAal-TiGa can be used

- ★ More in the paper:
  - ★ Rank construction for AECA
  - ★ ... with application to the language inclusion problem for nondeterminstic Büchi ECA