Preliminaries

On-the-fly Strategy Synthesis for Event-Clock Linear Temporal Logic on Timed Games

Peter Bulychev, Barbara Di Giampaolo, Laurent Doyen,
Gilles Geeraerts, Jean-François Raskin, Julien Reichert,
Pierre-Yves Schobbens, Tali Sznajder

1 Universitá di Salerno
2 ENS Cachan
3 Université Libre de Bruxelles
4 Laboratoire d'Informatique de Paris 6
5 FUNDP Univ. of Namur
6 Aalborg Univ.

On-the-fly Strategy Synthesis for Event-Clock Linear Temporal Logic on Timed Games

Peter Bulychev, Barbara Di Giampaolo, Laurent Doyen, Gilles Geeraerts, Jean-François Raskin, Julien Reichert, Pierre-Yves Schobbens, Tali Sznajder

1 Universitá di Salerno
2 ENS Cachan
3 Université Libre de Bruxelles
4 Laboratoire d'Informatique de Paris 6
5 FUNDP Univ. of Namur
6 Aalborg Univ.

Realizability problem in a real-time setting

Realizability problem in a real-time setting

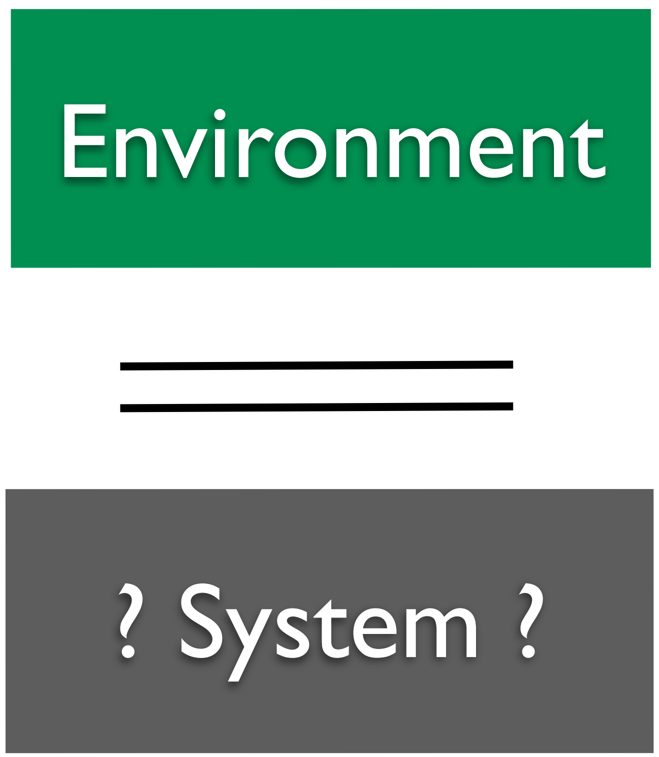
Environment

Realizability problem in a real-time setting

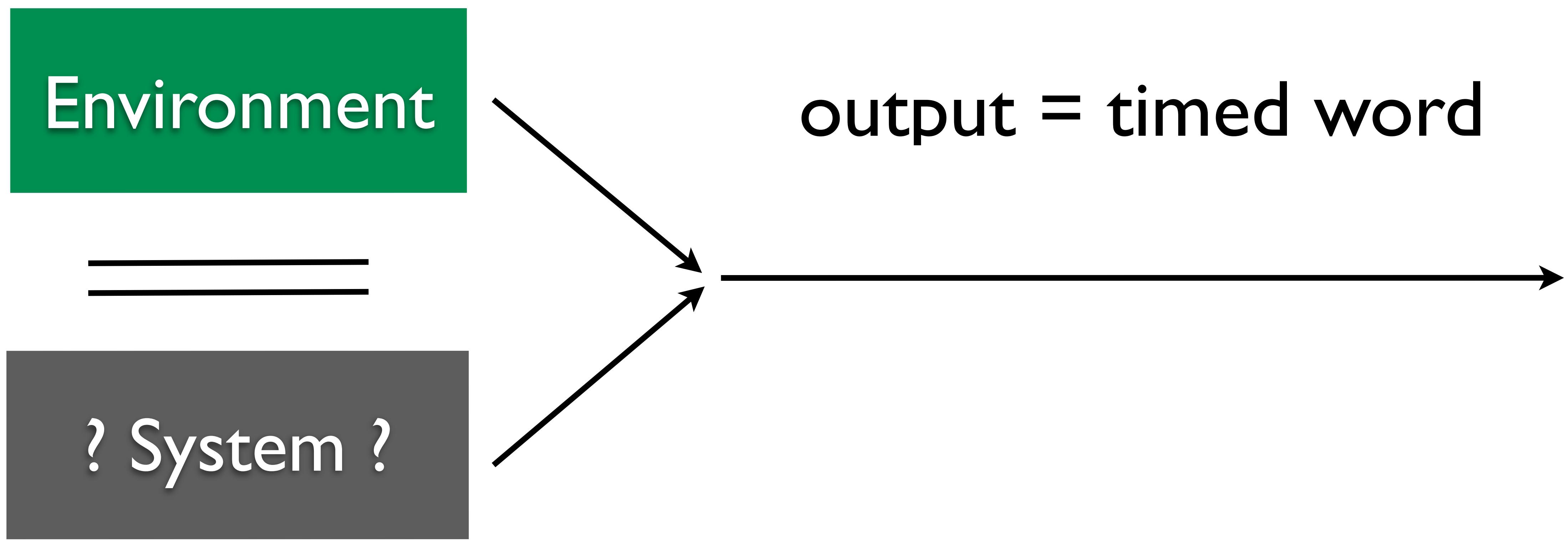
Environment

? System ?

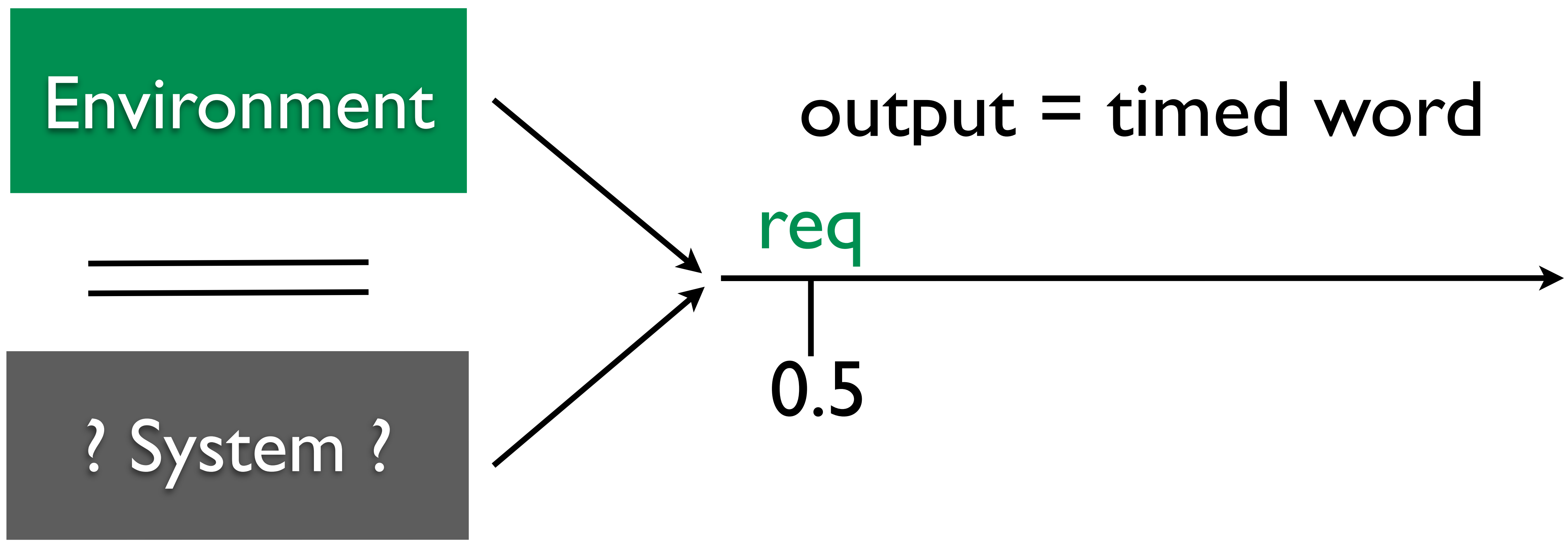
Realizability problem in a real-time setting



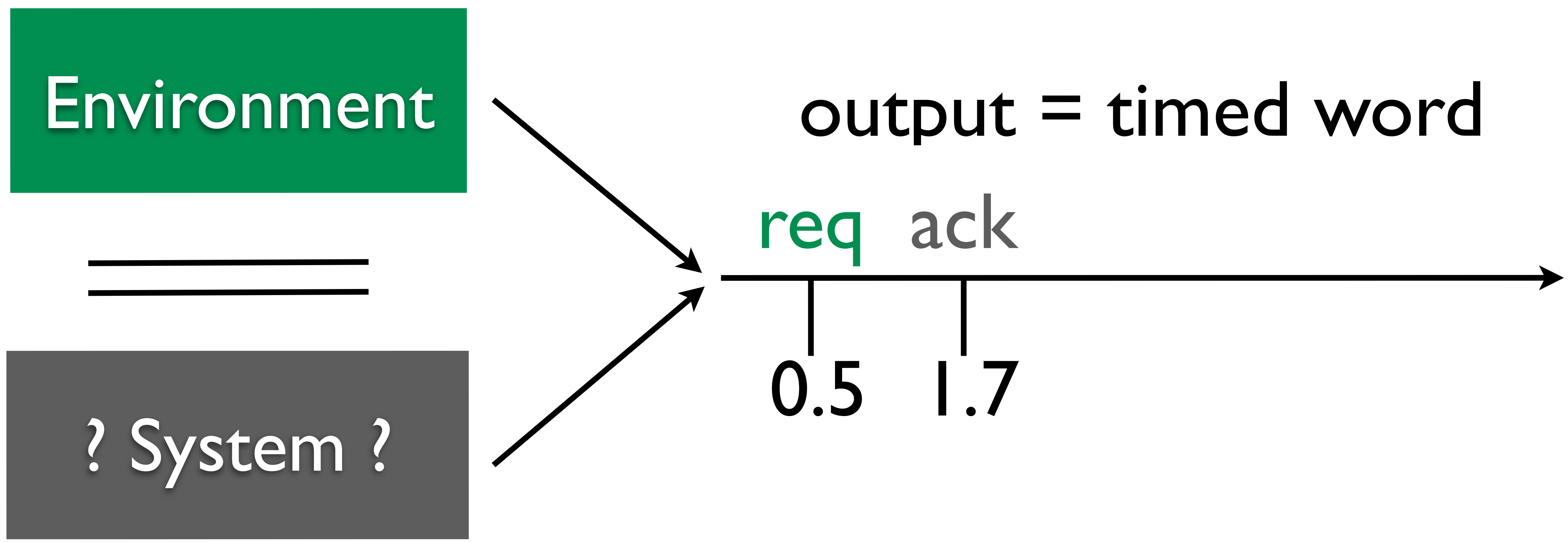
Realizability problem in a real-time setting



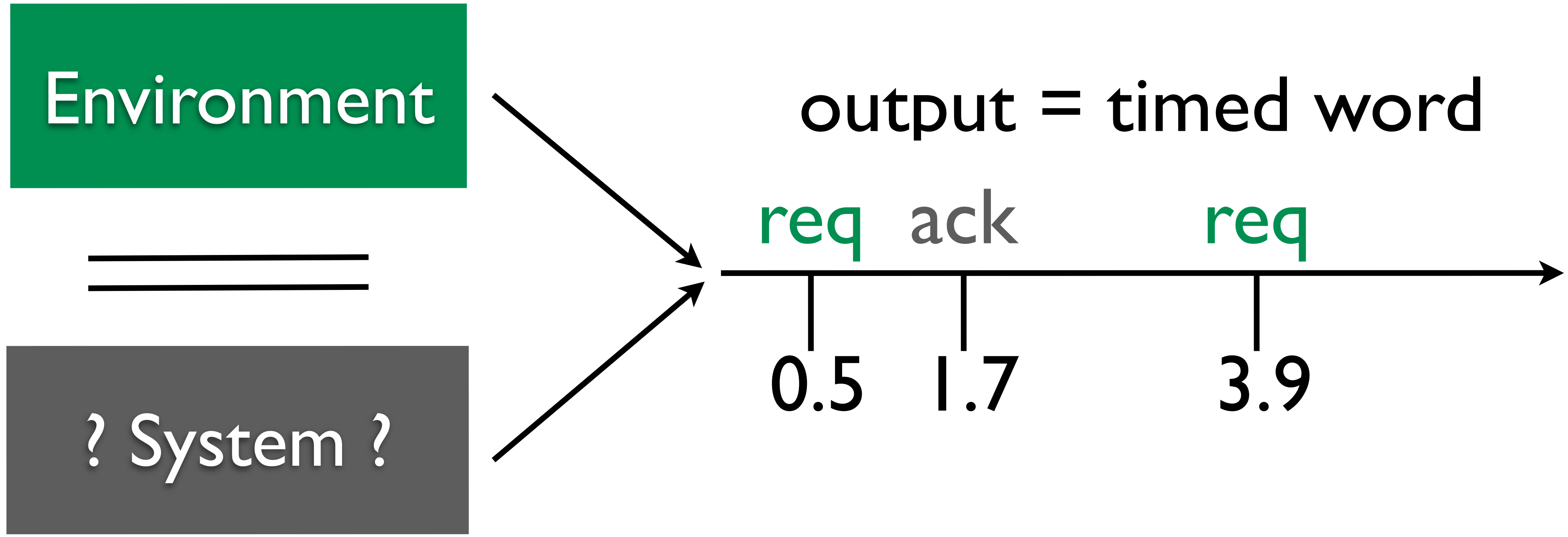
Realizability problem in a real-time setting



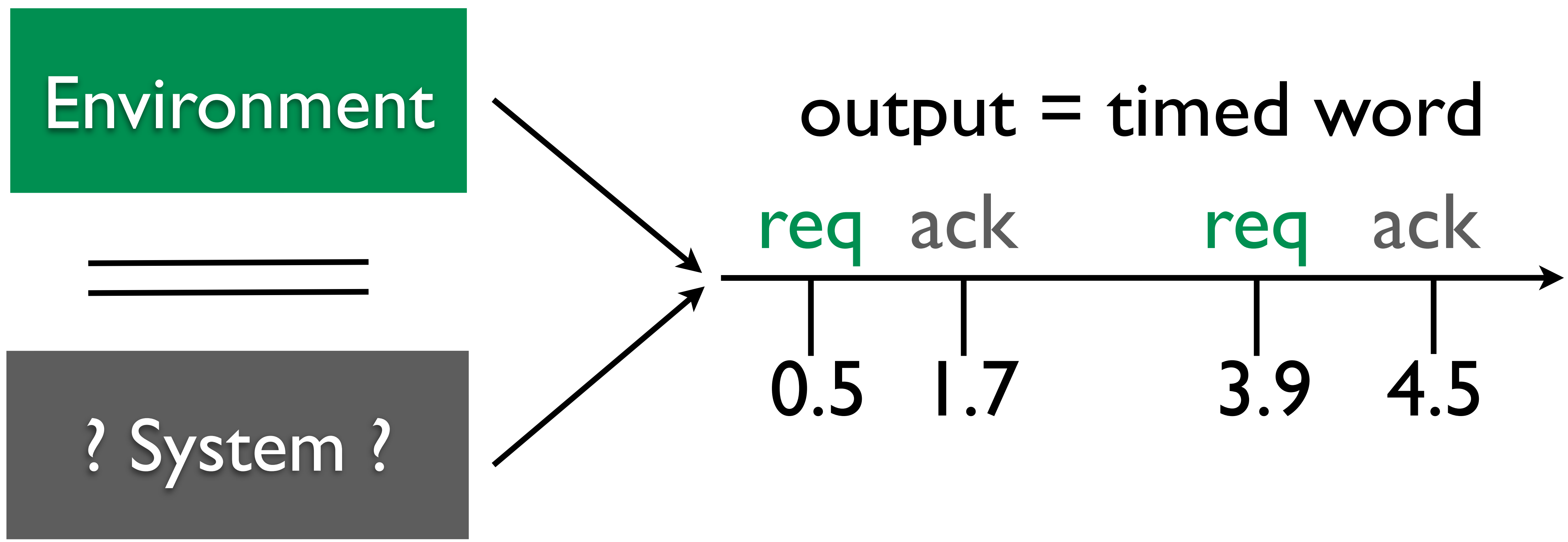
Realizability problem in a real-time setting



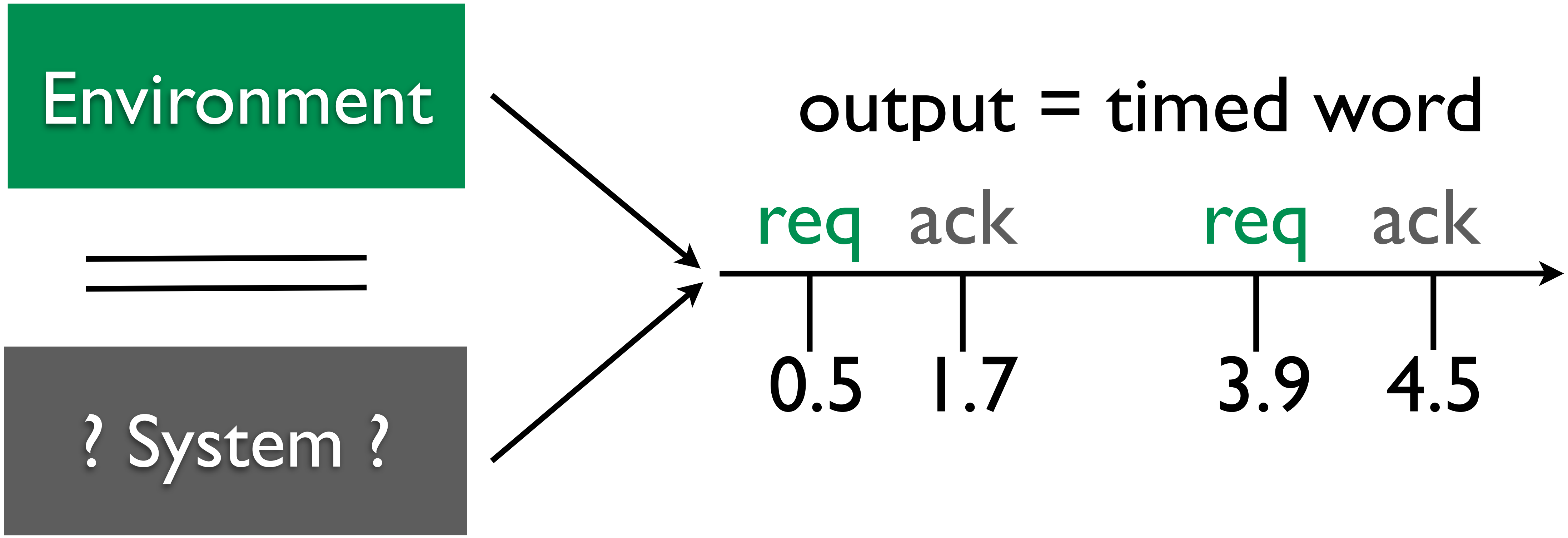
Realizability problem in a real-time setting



Realizability problem in a real-time setting



Realizability problem in a real-time setting

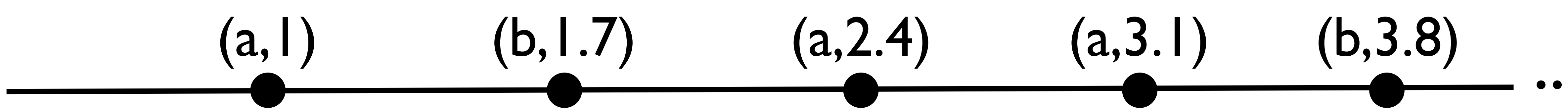


Problem

Given a 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 ?

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 $i\in\mathbb{N}$.

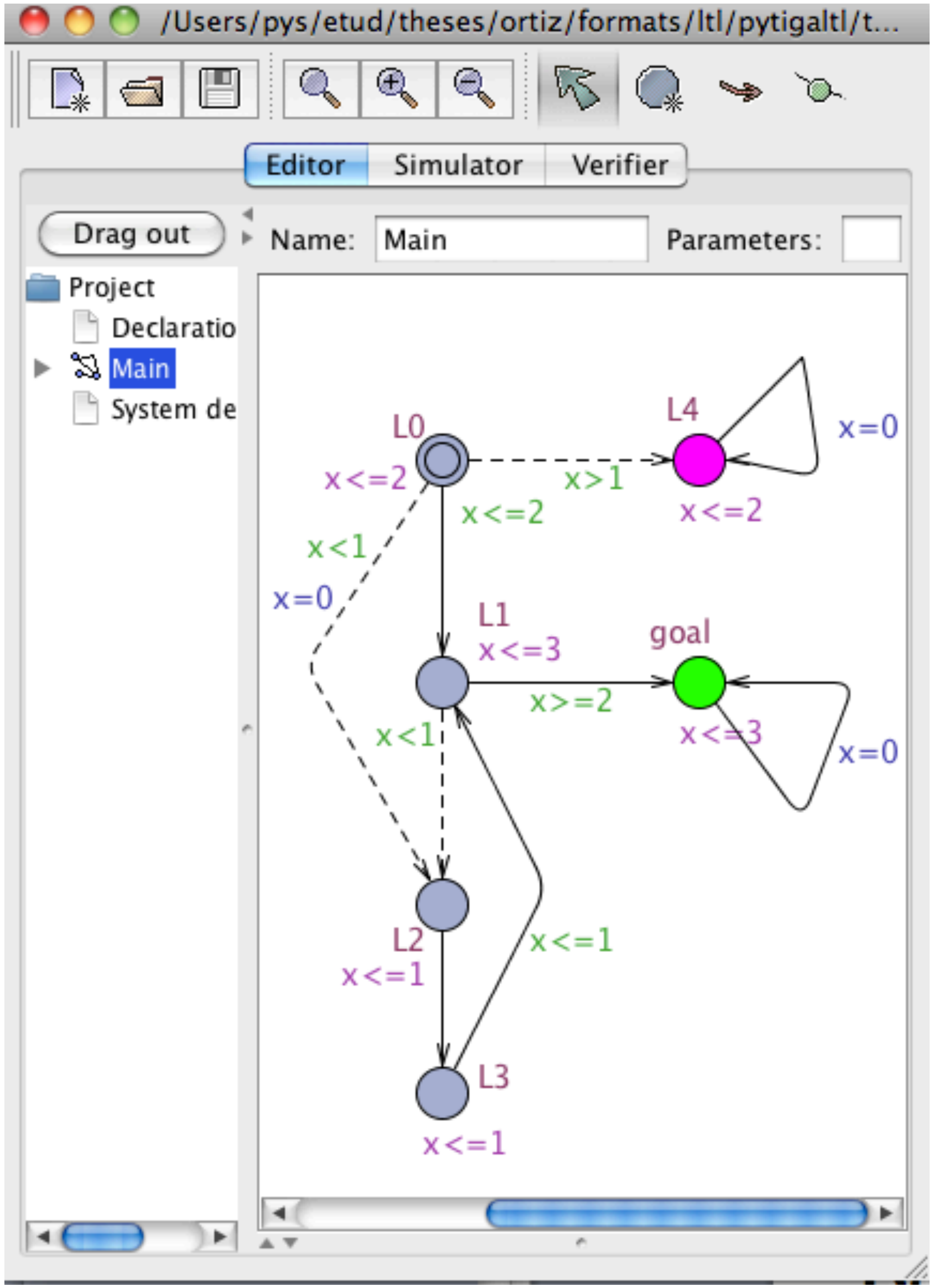
Timed Games

± Timed Automaton

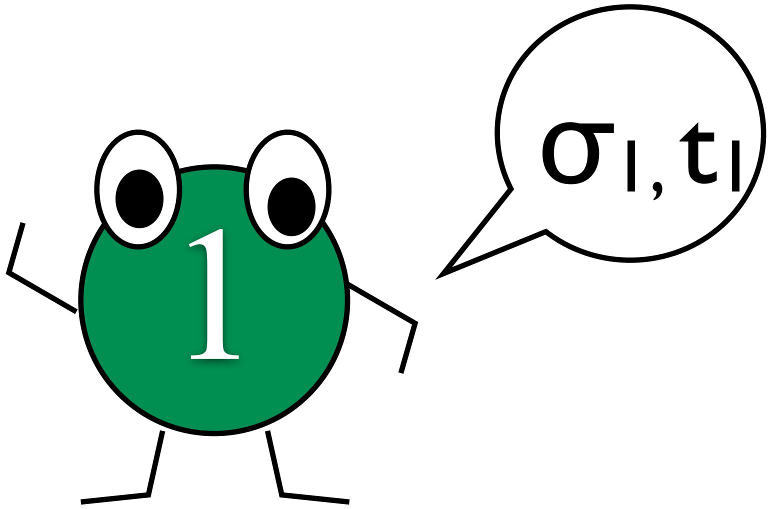
2 players: Sys and Env

Own transitions

Both players can agree to wait (as long as the location invariant stays true)



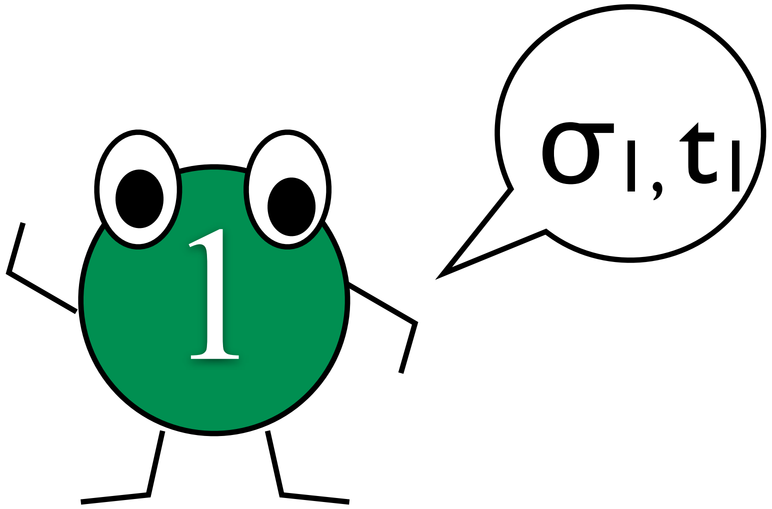
One round of the game



Player 1 chooses an action and a delay t_1

$(\sigma^1,\tau^1),\ldots,(\sigma^n,\tau^n),$

One round of the game

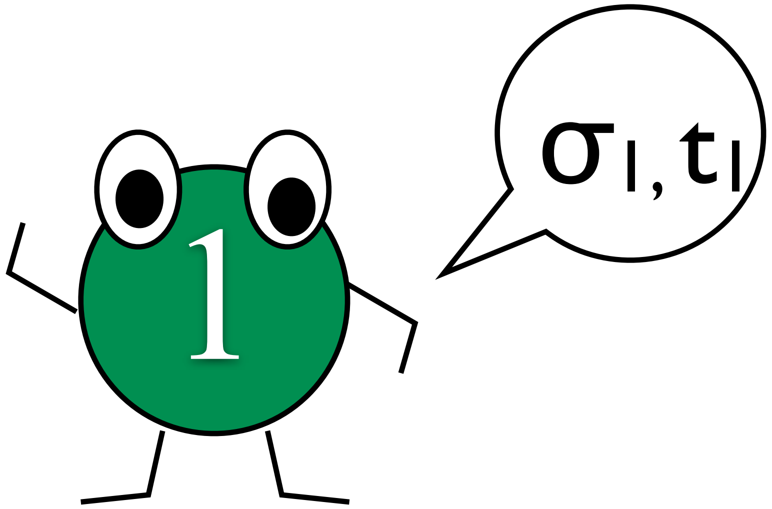


Player 1 chooses an action and a delay t_1

Player 2 may let Player 1 play

$(\sigma^1,\tau^1),\ldots,(\sigma^n,\tau^n),$

One round of the game

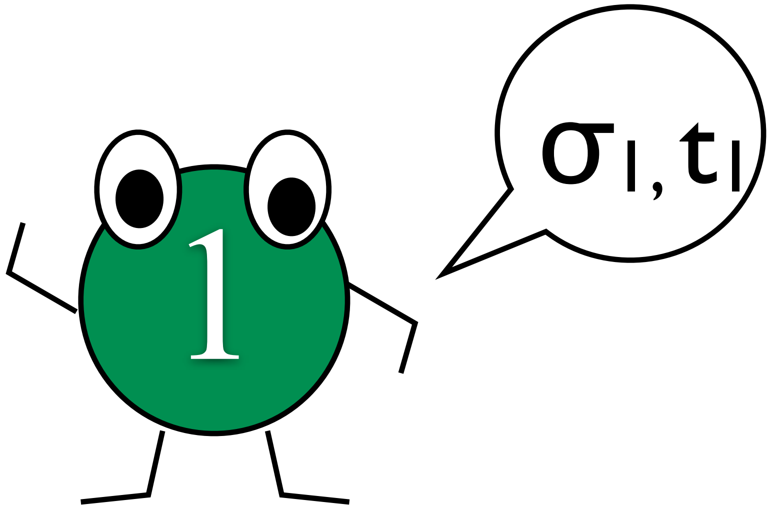


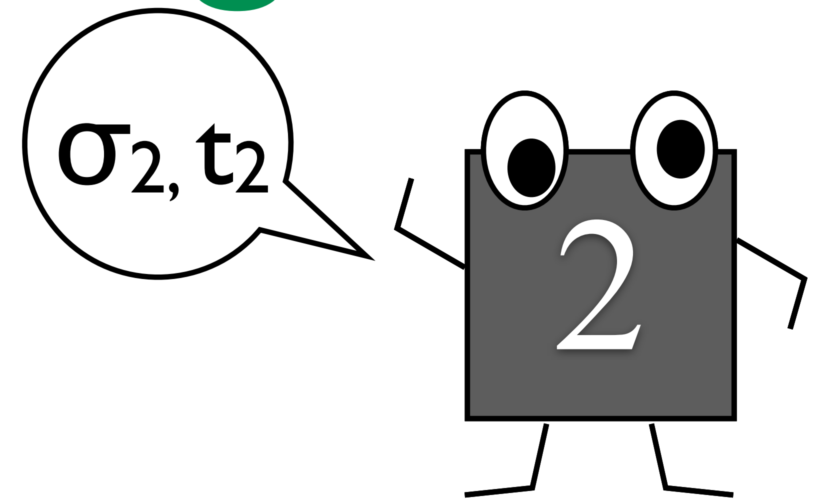
Player 1 chooses an action and a delay t_1

Player 2 may let Player 1 play

$(\sigma^1,\tau^1),\ldots,(\sigma^n,\tau^n),(\sigma_1, \tau^n+t_1)$

One round of the game



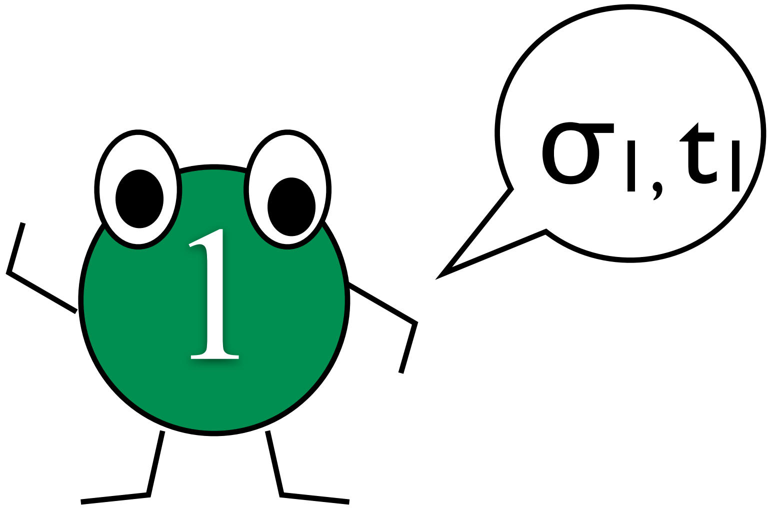


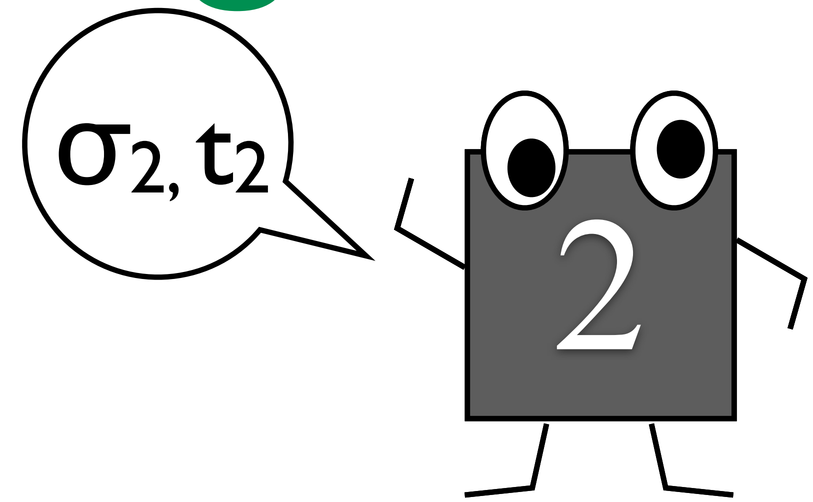
Player 1 chooses an action and a delay t_1

or chooses an action and a delay t_2, $t_2 \leq t_1$

$(\sigma^1,\tau^1),\ldots,(\sigma^n,\tau^n),$

One round of the game





Player 1 chooses an action and a delay t_1

or chooses an action and a delay t_2, $t_2 \leq t_1$

$(\sigma^1,\tau^1),\ldots,(\sigma^n,\tau^n), (\sigma_2, \tau^n+t_2)$

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 it 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 $\langle\Sigma_1,\Sigma_2,$**Win**$\rangle$ if Outcome(λ_1)$\subseteq$**Win**

Event-Clock Logic (ECL)

$$\varphi \in \text{ECL} ::= a \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi\,\mathcal{S}\,\varphi \mid \varphi\,\mathcal{U}\,\varphi \mid \triangleleft_{\mathrm{I}}\,\varphi \mid \triangleright_{\mathrm{I}}\,\varphi$$

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

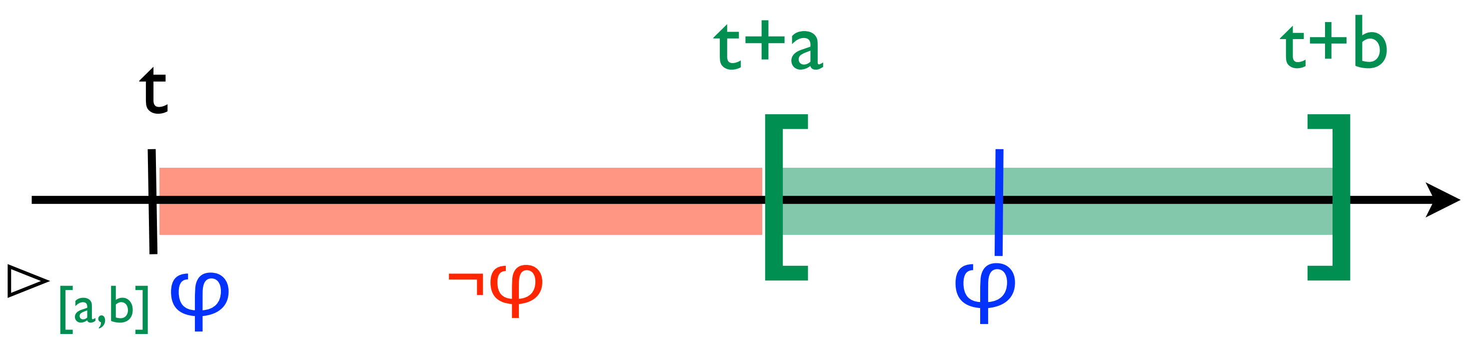
$\triangleright_{[a,b]}\ \varphi$

Event-Clock Logic (ECL)

$\varphi \in \text{ECL} ::= a \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi\, \mathcal{S}\, \varphi \mid \varphi\, \mathcal{U}\, \varphi \mid \lhd_{\mathrm{I}}\, \varphi \mid \rhd_{\mathrm{I}}\, \varphi$

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

$\rhd_{[a,b]}\, \varphi$

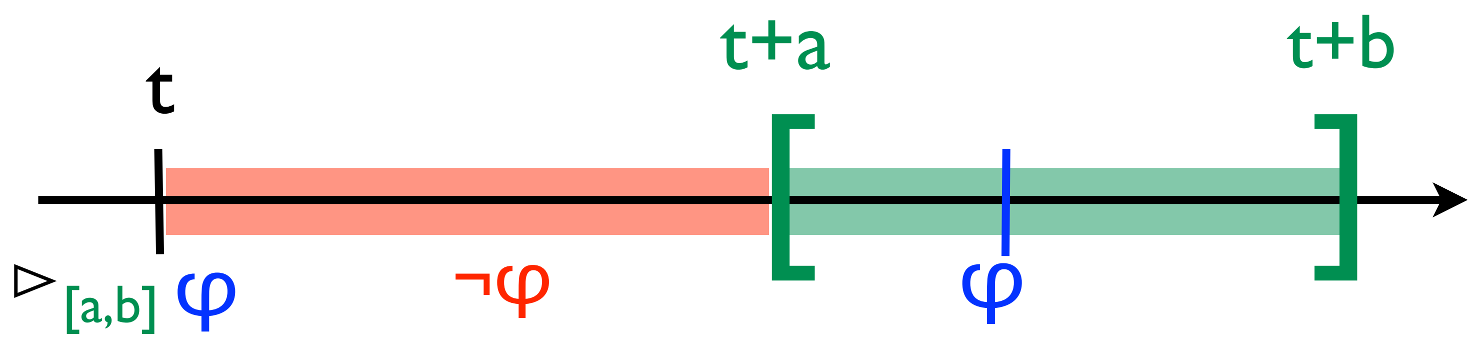


Event-Clock Logic (ECL)

$$\varphi \in \text{ECL} ::= a \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi\, \mathcal{S}\, \varphi \mid \varphi\, \mathcal{U}\, \varphi \mid \lhd_I\, \varphi \mid \rhd_I\, \varphi$$

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

$\rhd_{[a,b]}\ \varphi$



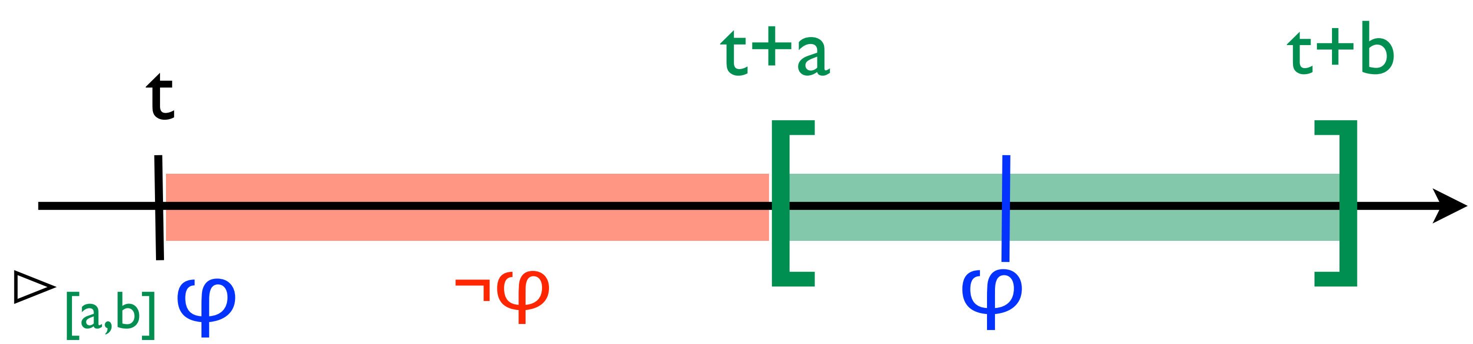
Remark: it is different from:

Event-Clock Logic (ECL)

$$\varphi \in \text{ECL} ::= a \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \mathcal{S} \varphi \mid \varphi \mathcal{U} \varphi \mid \lhd_I \varphi \mid \rhd_I \varphi$$

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

$\rhd_{[a,b]} \varphi$



Remark: it is different from:

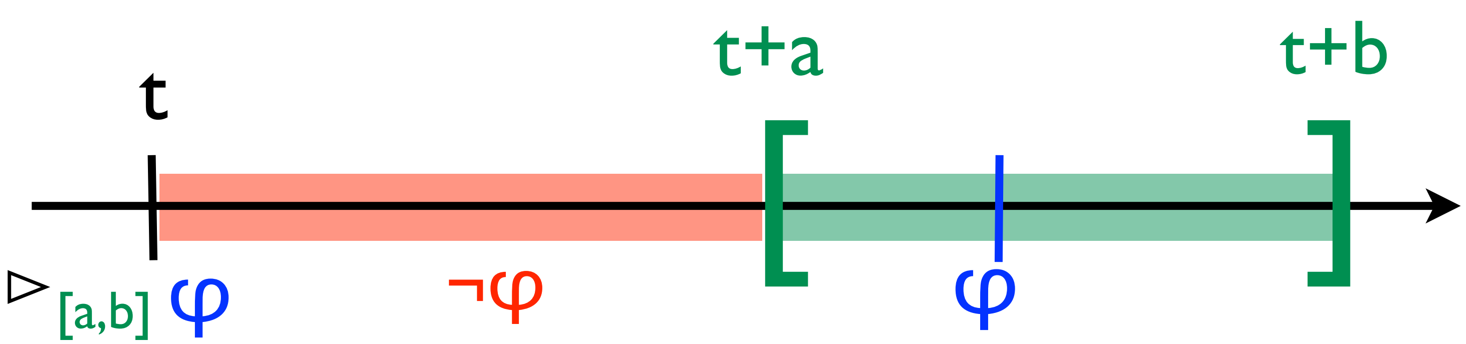
$\Diamond_{[a,b]} \varphi$

Event-Clock Logic (ECL)

$$\varphi \in \text{ECL} ::= a \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi\, \mathcal{S}\, \varphi \mid \varphi\, \mathcal{U}\, \varphi \mid \lhd_I\, \varphi \mid \rhd_I\, \varphi$$

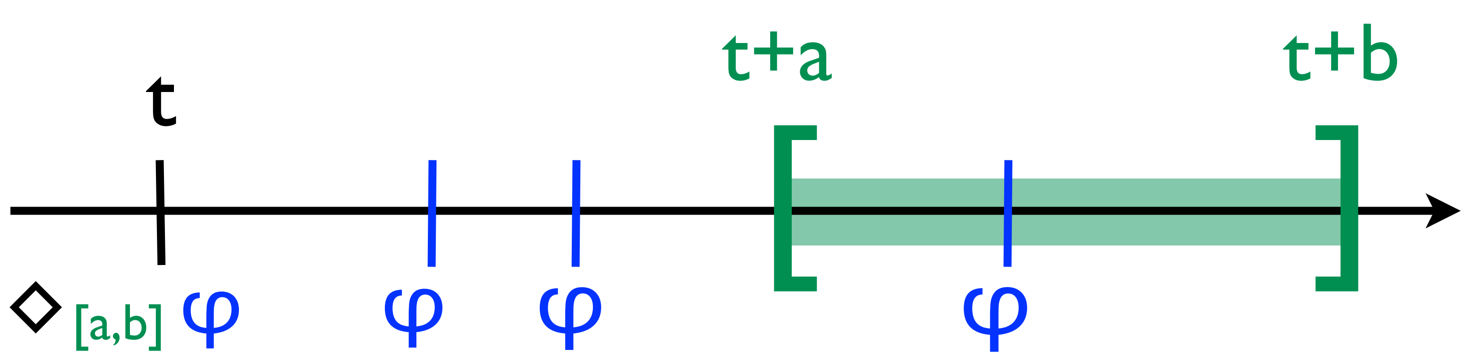
with I an interval of $\mathbb{R}^{\geq 0}$ with integer bounds

$\rhd_{[a,b]}\ \varphi$



Remark: it is different from:

$\Diamond_{[a,b]}\ \varphi$

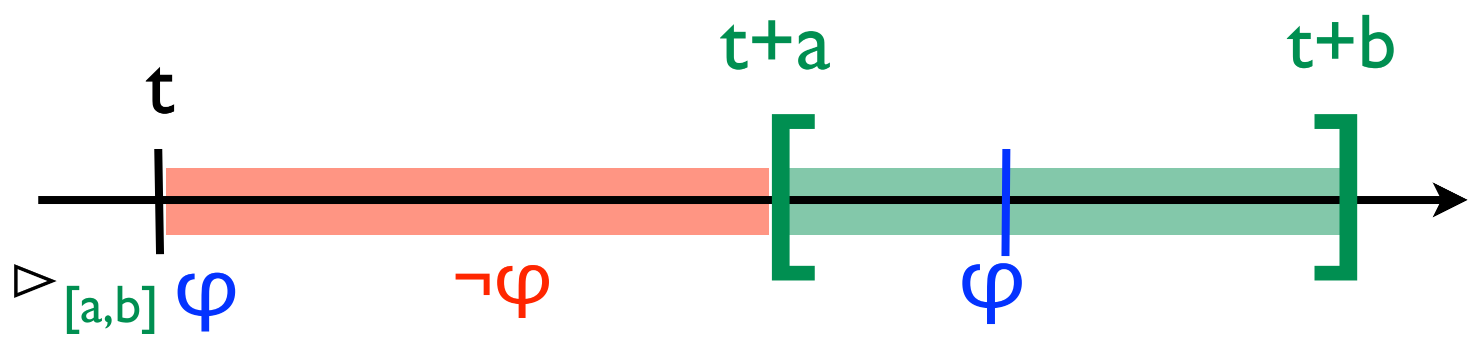


Event-Clock Logic (ECL)

$\varphi \in \mathsf{ECL} ::= a \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi\,\mathcal{S}\,\varphi \mid \varphi\,\mathcal{U}\,\varphi \mid \lhd_I\,\varphi \mid \rhd_I\,\varphi$

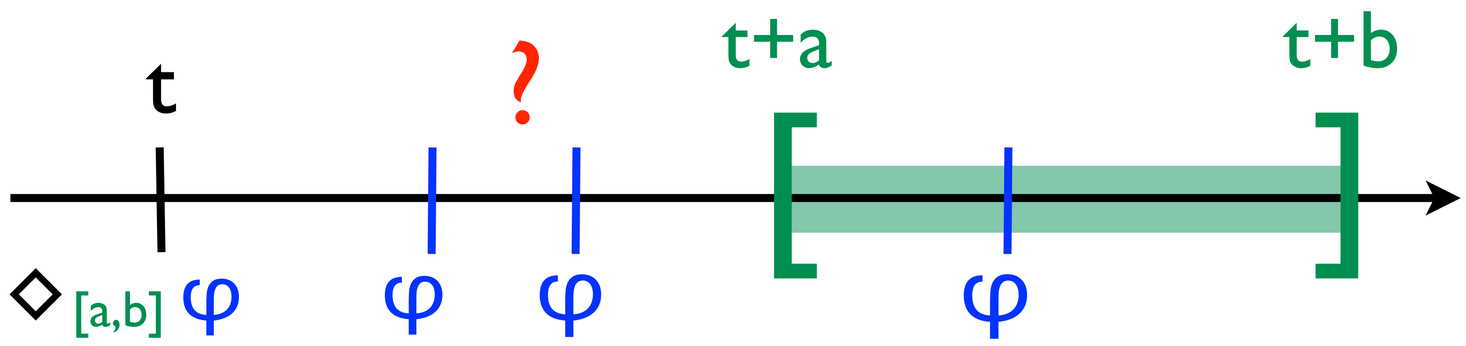
with I an interval of $\mathbb{R}^{\geq 0}$ with integer bounds

$\rhd_{[a,b]}\,\varphi$



Remark: it is different from:

$\Diamond_{[a,b]}\,\varphi$



Event-Clock Logic (ECL)

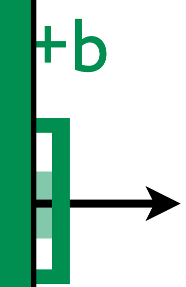
$\varphi \in \text{ECL} ::= a \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi\,\mathcal{S}\,\varphi \mid \varphi\,\mathcal{U}\,\varphi \mid \triangleleft_I\,\varphi \mid \triangleright_I\,\varphi$

We consider timed games of the form

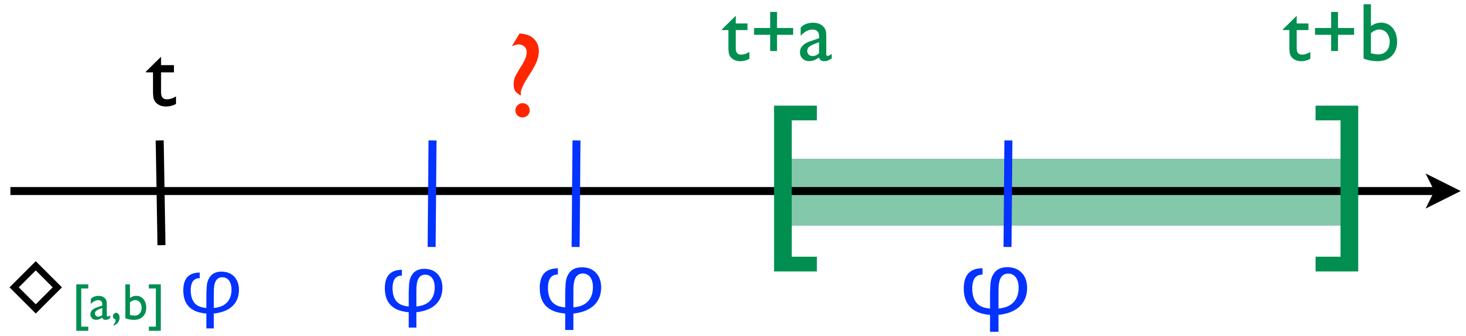
$\langle \Sigma_1, \Sigma_2, [\![\varphi]\!] \rangle$

where

φ is an ECL formula



$\Diamond_{[a,b]}\,\varphi$



Event-Clock Logic (ECL)

$\varphi \in \text{ECL} ::= a \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi\,\mathcal{S}\,\varphi \mid \varphi\,\mathcal{U}\,\varphi \mid \lhd_I\,\varphi \mid \rhd_I\,\varphi$

We consider timed games of the form

$\langle \Sigma_1, \Sigma_2, [\![\varphi]\!] \rangle$

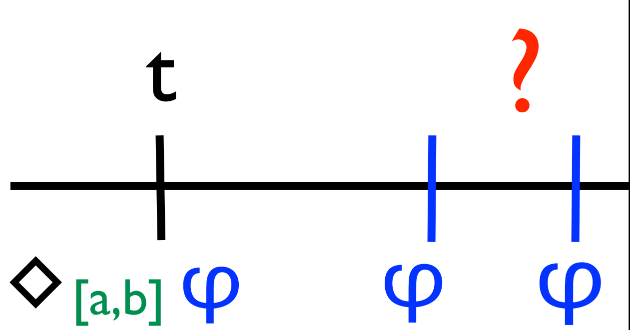
where

φ is an ECL formula

This problem is called

ECL «realizability»

$\Diamond_{[a,b]}\,\varphi$



Why ECL?

- Satisfiability of MTL **undecidable** on infinite words.
  $\Longrightarrow$ Realizability is thus **undecidable** too !
- ECL is an interesting subcase of MITL (equivalent to $MITL_{0,\infty}$).

Results

Undecidability of ECL realizability

Theorem: ECL realizability is undecidable

- Idea of the proof: encode computations of lossy three counters machines into timed words
- Build a game s.t. Player 1 has a winning strategy iff the machine admits an infinite bounded run
  - One has to use the interaction of the Players to check that the encoding is correct.

LTL$_{\triangleleft}$ realizability is decidable

$$\varphi \in \text{ECL} ::= a \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi\, \mathcal{S}\, \varphi \mid \varphi\, \mathcal{U}\, \varphi \mid \triangleleft_{\mathrm{I}}\, \varphi \mid \triangleright_{\mathrm{I}}\, \varphi$$

LTL$_\lhd$ realizability is decidable

$$\varphi \in \mathsf{ECL} ::= a \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi\,\mathcal{S}\,\varphi \mid \varphi\,\mathcal{U}\,\varphi \mid \lhd_{\mathrm{I}}\,\varphi \mid \rhd_{\mathrm{I}}\,\varphi$$

$$\psi \in \mathsf{LTL}_\lhd ::= a \mid \neg\psi \mid \psi \vee \psi \mid \psi\,\mathcal{S}\,\psi \mid \psi\,\mathcal{U}\,\psi \mid \lhd_{\mathrm{I}}\,a$$

$\mathsf{LTL}_{\triangleleft}$ realizability is decidable

$$\varphi \in \mathsf{ECL} ::= a \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi\,\mathcal{S}\,\varphi \mid \varphi\,\mathcal{U}\,\varphi \mid \triangleleft_{\mathrm{I}}\,\varphi \mid \triangleright_{\mathrm{I}}\,\varphi$$

$$\psi \in \mathsf{LTL}_{\triangleleft} ::= a \mid \neg\psi \mid \psi \vee \psi \mid \psi\,\mathcal{S}\,\psi \mid \psi\,\mathcal{U}\,\psi \mid \triangleleft_{\mathrm{I}}\,a$$

The real-time modality can «speak» about past events only

LTL$_\lhd$ realizability is decidable

$$\varphi \in \mathsf{ECL} ::= a \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi\,\mathcal{S}\,\varphi \mid \varphi\,\mathcal{U}\,\varphi \mid \lhd_{\mathrm{I}}\,\varphi \mid \rhd_{\mathrm{I}}\,\varphi$$

$$\psi \in \mathsf{LTL}_\lhd ::= a \mid \neg\psi \mid \psi \vee \psi \mid \psi\,\mathcal{S}\,\psi \mid \psi\,\mathcal{U}\,\psi \mid \lhd_{\mathrm{I}}\,a$$

The real-time modality can «speak» about past events only

- **Theorem**: The realizability problem for LTL$_\lhd$ is 2EXPTIME-complete

LTL$_\triangleleft$ realizability is decidable

$$\varphi \in \mathsf{ECL} ::= a \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi\,\mathcal{S}\,\varphi \mid \varphi\,\mathcal{U}\,\varphi \mid \triangleleft_{\mathrm{I}}\,\varphi \mid \triangleright_{\mathrm{I}}\,\varphi$$

$$\psi \in \mathsf{LTL}_{\triangleleft} ::= a \mid \neg\psi \mid \psi \vee \psi \mid \psi\,\mathcal{S}\,\psi \mid \psi\,\mathcal{U}\,\psi \mid \triangleleft_{\mathrm{I}}\, a$$

The real-time modality can «speak» about past events only

- **Theorem**: The realizability problem for LTL$_\triangleleft$ is 2EXPTIME-complete

- **Idea**: from ψ, build a deterministic timed automaton with parity condition

LTL$_{\lhd}$ realizability is decidable

Determinization of Büchi automata is already hard in practice !

$\ldots \varphi \mid \lhd_{\mathrm{I}}\, \varphi \mid \rhd_{\mathrm{I}}\, \varphi$

$\ldots \psi \mid \lhd_{\mathrm{I}}\, a$

…peak»

- **Theorem**: The realizability problem for LTL$_{\lhd}$ is 2EXPTIME-complete

- **Idea**: from ψ, build a deterministic timed automaton with parity condition

LTL$_\triangleleft$ realizability is decidable

$\varphi \mid \triangleleft_I \varphi \mid \triangleright_I \varphi$

$\psi \mid \triangleleft_I a$

Determinization of Büchi automata is already hard in practice !

Can we find «Safraless» procedures that avoid Safra's determinization ?

- **Theorem**
  LTL$_\triangleleft$ is 2EX

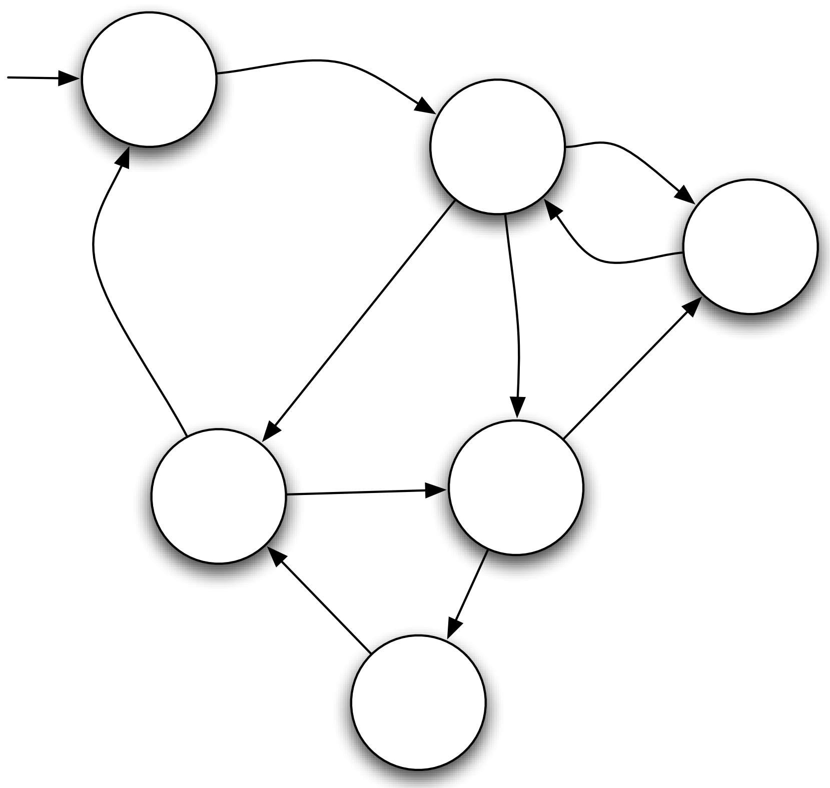
- **Idea**: from ψ, build a deterministic timed automaton with parity condition

Safraless procedures

- Safraless realizability/synthesis (untimed setting):
  - ★ 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

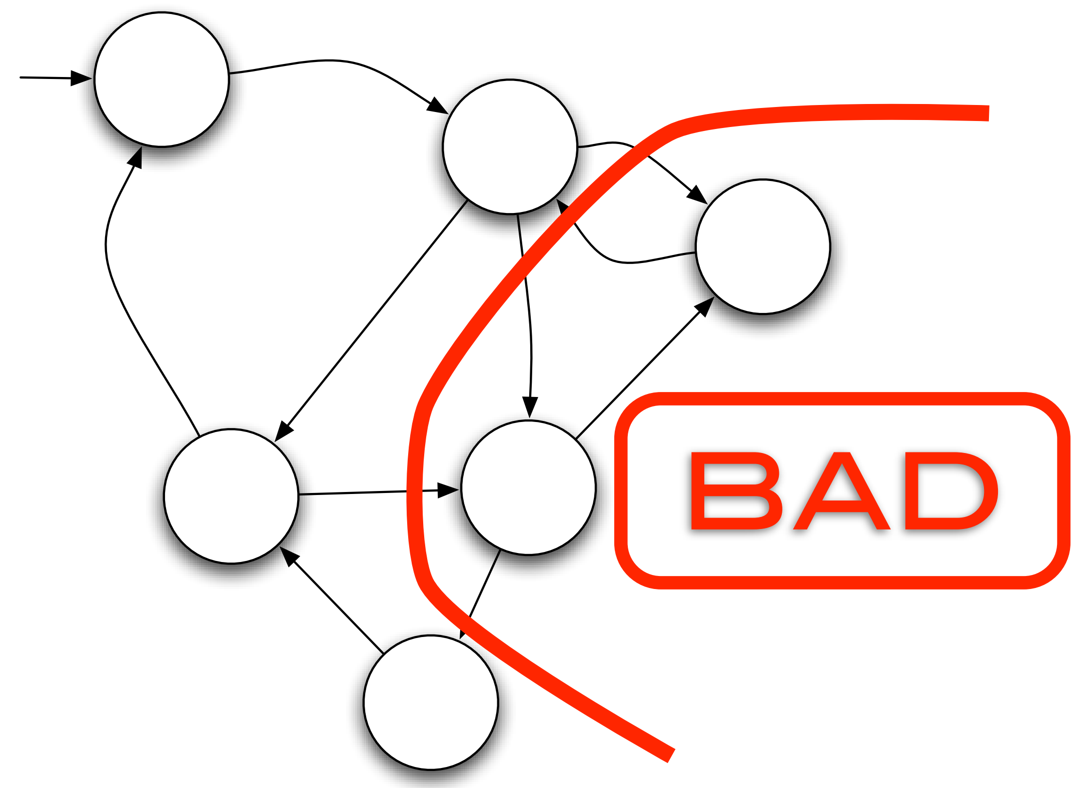
Idea of procedure

- **Reduce** the realizability problem to a **safety (timed) game**
  - Game played on a graph
  - Goal: avoid bad states
- Not a Büchi condition: avoid Safra !
- Allows incremental procedure
- Tools and algorithms exist to solve safety (timed) games
  - e.g.: UppAal TiGa



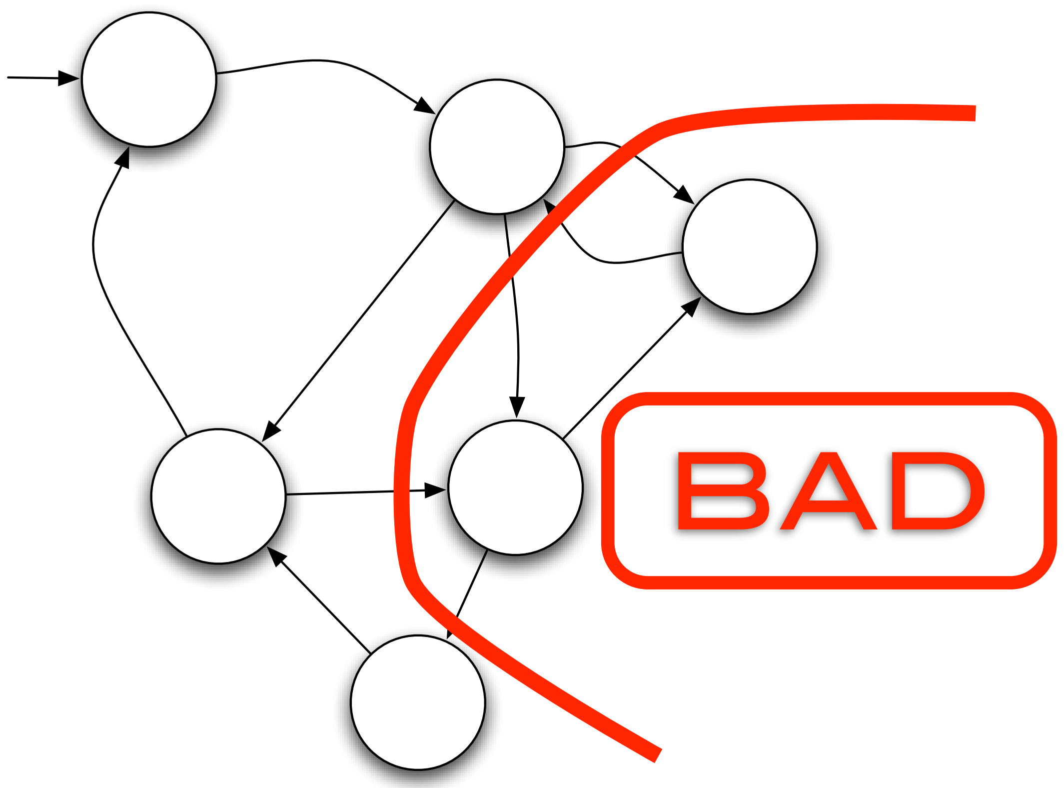
Idea of procedure

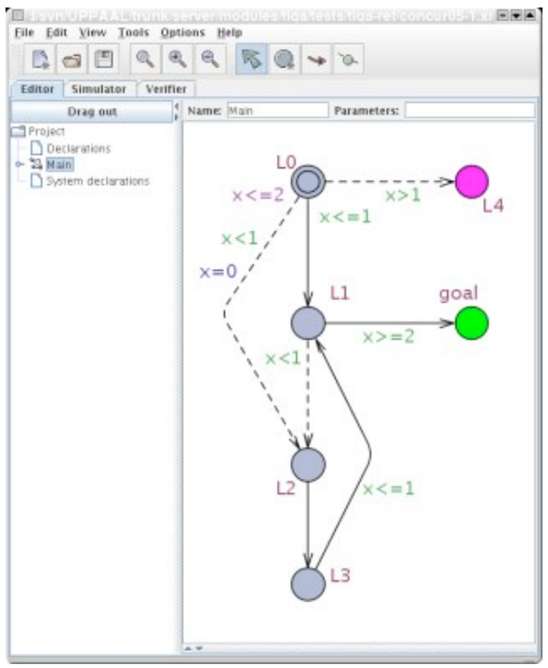
- **Reduce** the realizability problem to a **safety (timed) game**
  - Game played on a graph
  - Goal: avoid bad states
- Not a Büchi condition: avoid Safra !
- Allows incremental procedure
- Tools and algorithms exist to solve safety (timed) games
  - e.g.: UppAal TiGa



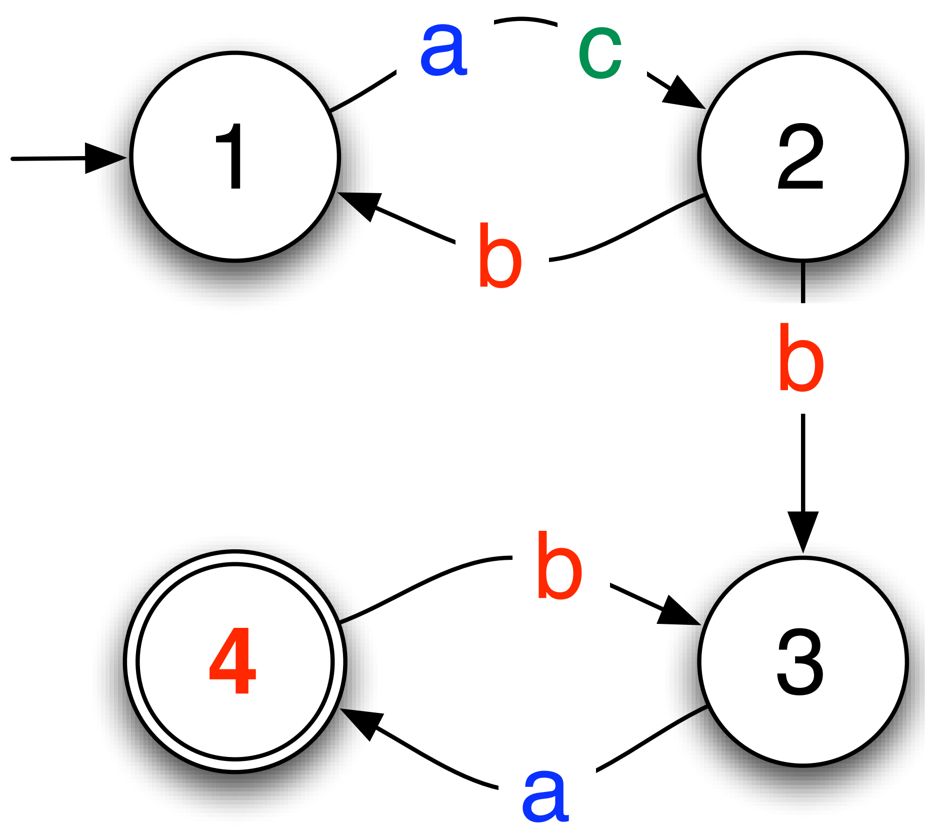
Idea of procedure

- **Reduce** the realizability problem to a **safety (timed) game**
  - Game played on a graph
  - Goal: avoid bad states
- Not a Büchi condition: avoid Safra !
- Allows incremental procedure
- Tools and algorithms exist to solve safety (timed) games
  - e.g.: UppAal TiGa



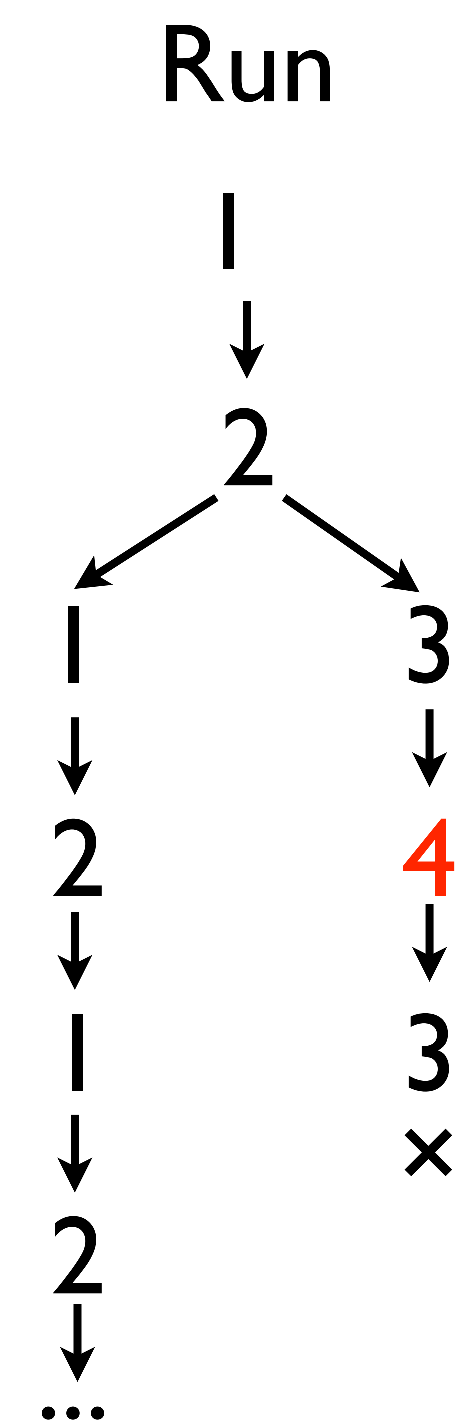


Universal coBüchi **Word** Automata



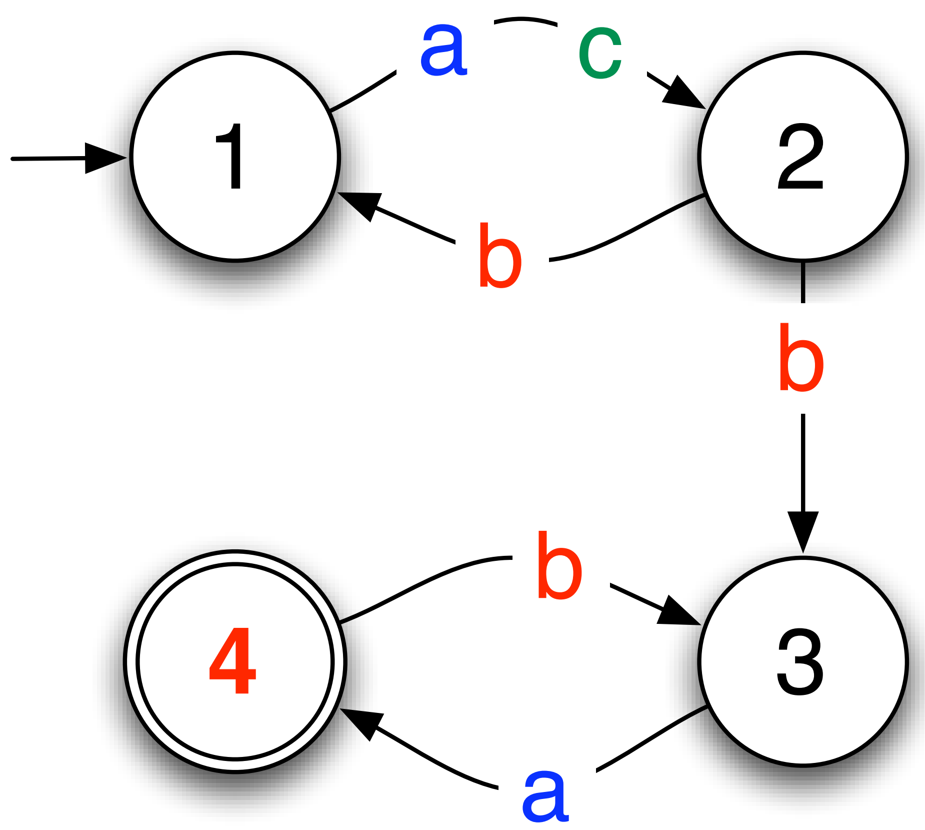
Σ^ω

a
b
a
b
c
...



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

Universal KcoBüchi **Word** Automata



Σ^ω

a

b

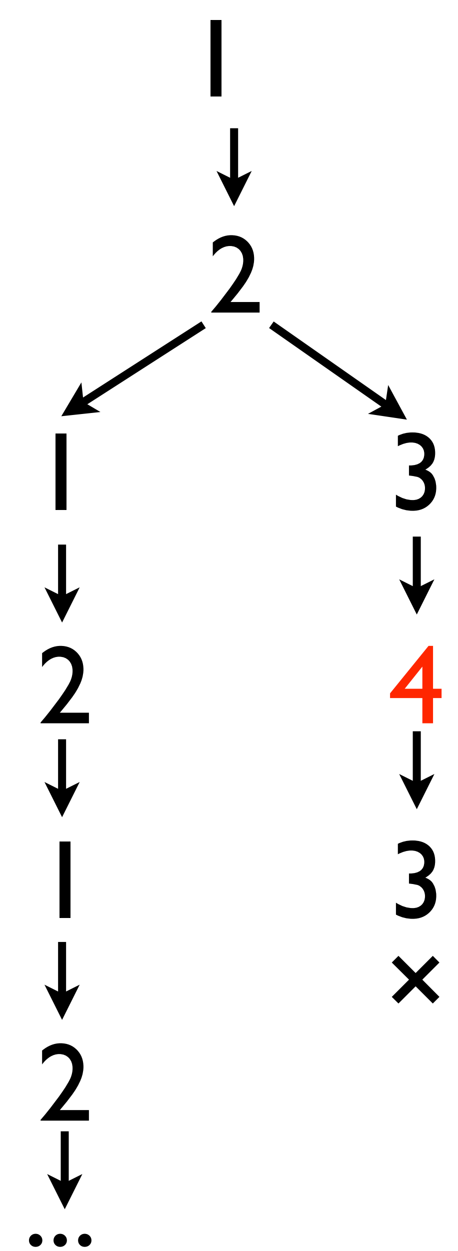
a

b

c

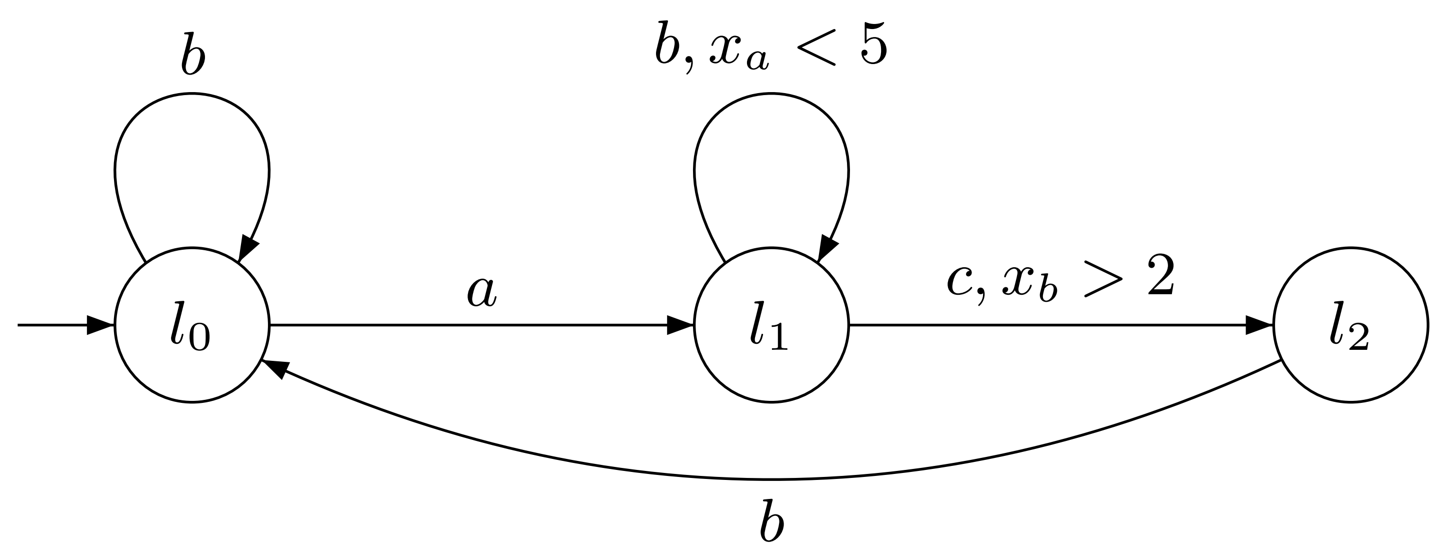
...

Run

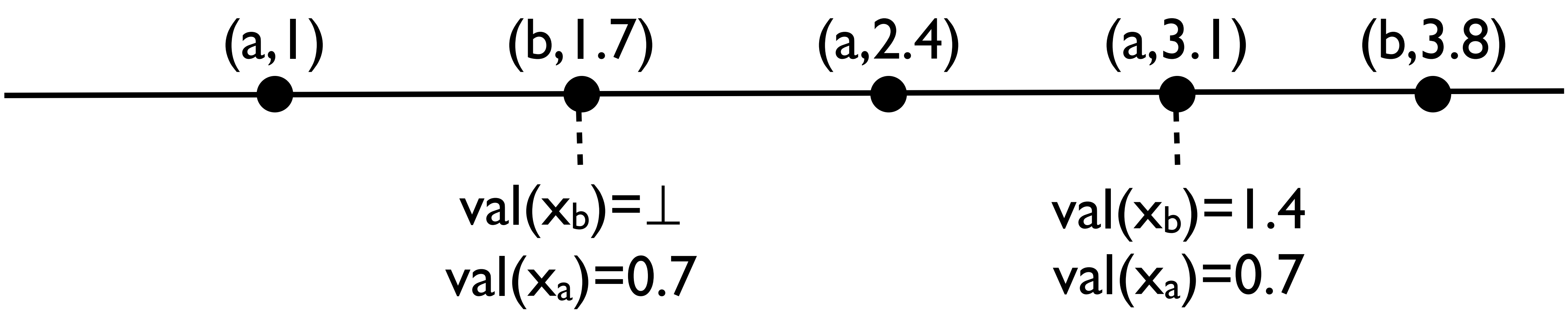


$w \in L_{UKcoB}(A)$
iff
all runs of A on w visit
at most K times α.

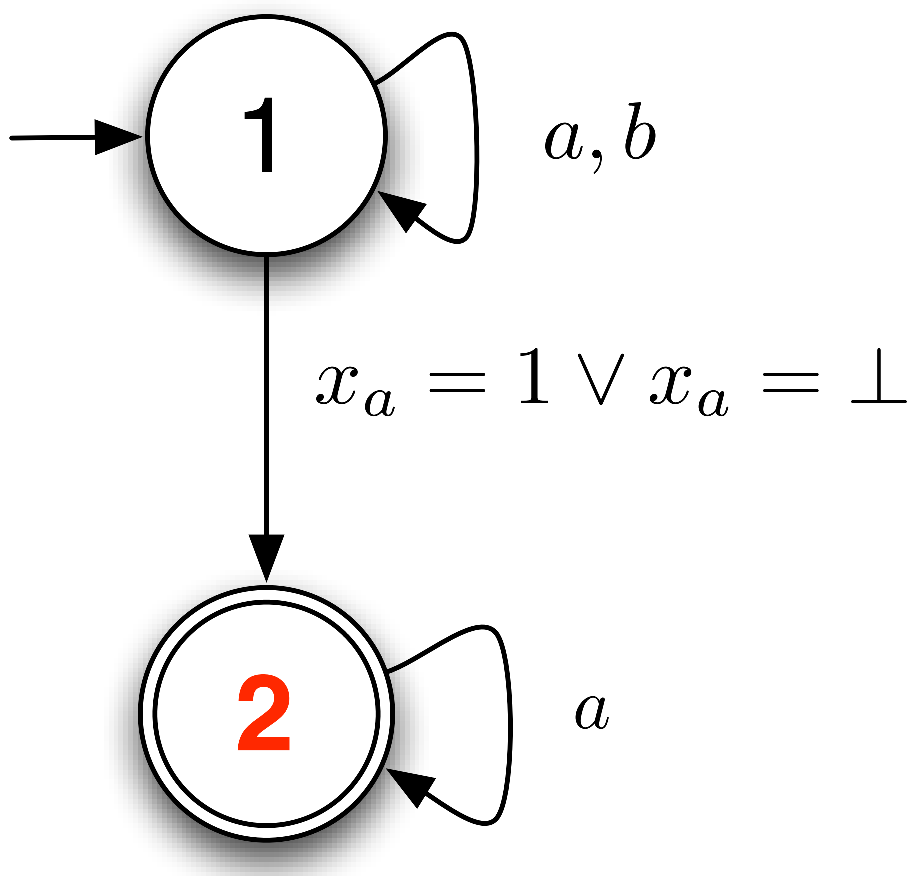
Event-recording automata

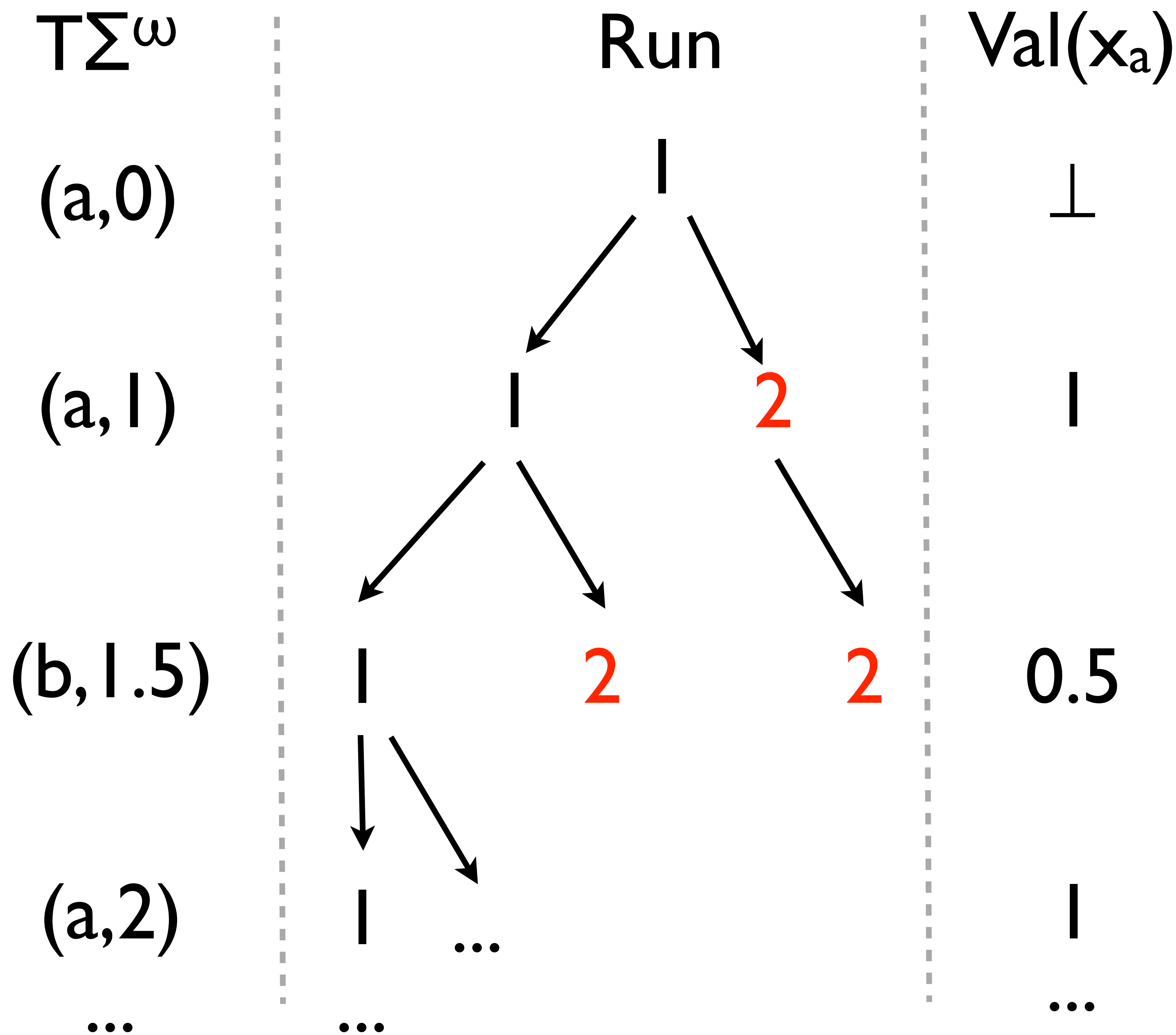


Clock are not reset and are associated to events: $\{ x_\sigma \mid \sigma \in \Sigma \}$
Each clock monitors the last occurence of the associated letter
Values of event-clocks are input determined:

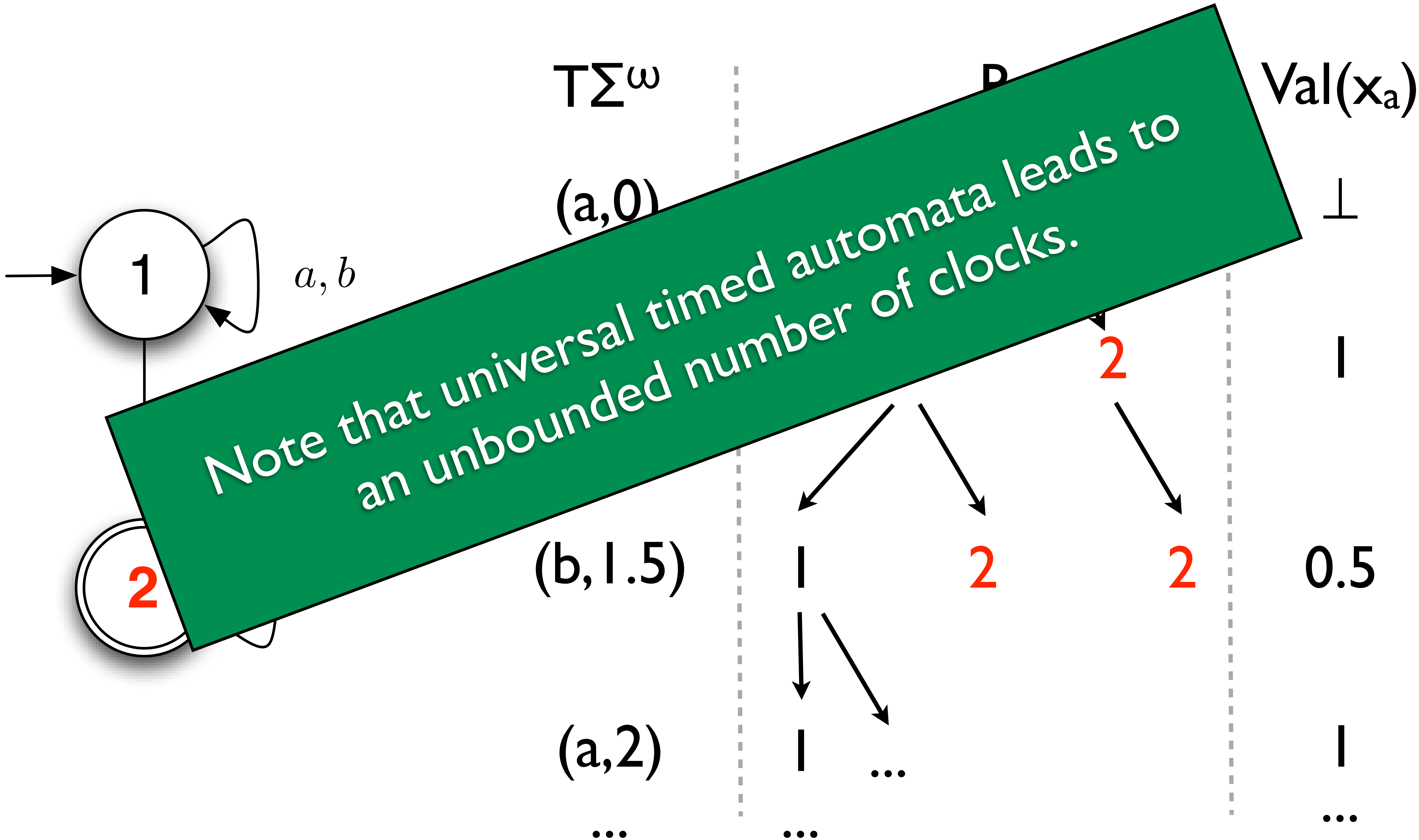


Universal ERA with coBüchi a. c.





Universal ERA with coBüchi a. c.



Back to LTL$_\triangleleft$ realizability

Back to $LTL_{\triangleleft}$ realizability

Theorem: From φ in $LTL_{\triangleleft}$, one can build a Universal co-Büchi ERA A_{φ} **such that** $L_{UcoB}(A_{\varphi}) = [\![\varphi]\!]$

Back to LTL$_\triangleleft$ realizability

Theorem: From φ in LTL$_\triangleleft$, one can build a Universal co-Büchi ERA A_φ **such that** $L_{UcoB}(A_\varphi) = [\![\varphi]\!]$

$\langle \Sigma_1, \Sigma_2, [\![\varphi]\!] \rangle$ becomes $\langle \Sigma_1, \Sigma_2, L_{UcoB}(A_\varphi) \rangle$

Back to LTL$_\triangleleft$ realizability

Theorem: From φ in LTL$_\triangleleft$, one can build a Universal co-Büchi ERA A_φ **such that** $L_{UcoB}(A_\varphi) = [\![\varphi]\!]$

$\langle \Sigma_1, \Sigma_2, [\![\varphi]\!] \rangle$ becomes $\langle \Sigma_1, \Sigma_2, L_{UcoB}(A_\varphi) \rangle$

We are now playing the game on A_φ

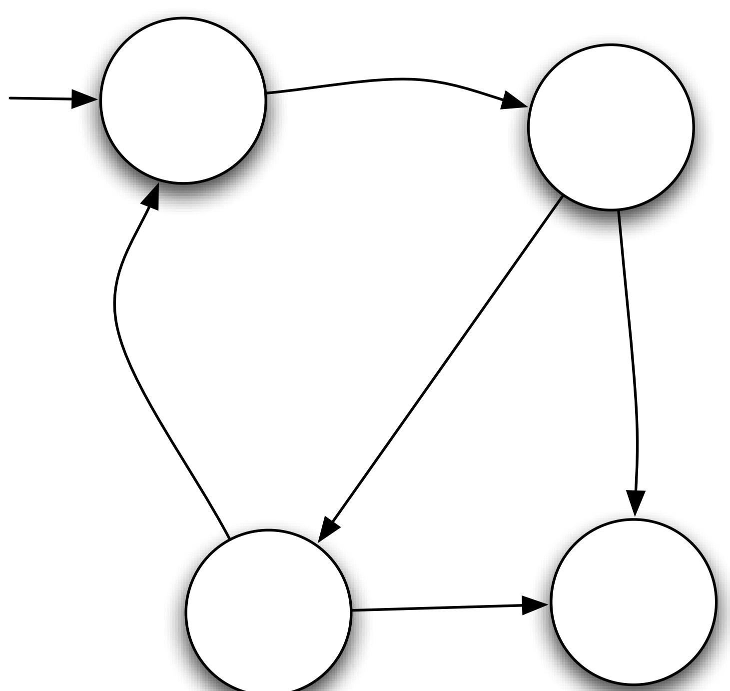
Goal of Player 1: ensure that every run on the outcome visits accepting states **finitely often**

From UCoB to UKCoB

Theorem: Winning strategies of Player 1 on the UCoB automaton can be represented by a finite machine (with m states)

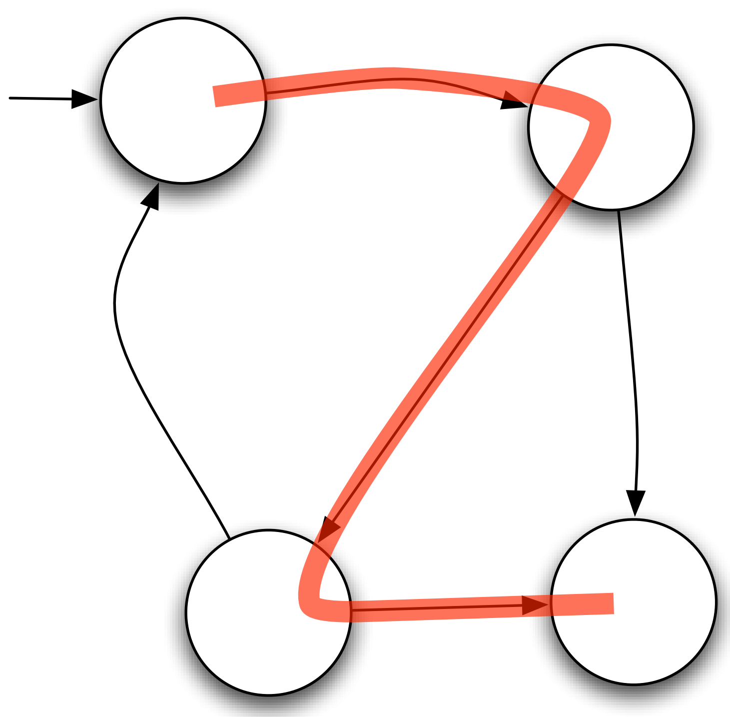
From UCoB to UKCoB

Theorem: Winning strategies of Player 1 on the UCoB automaton can be represented by a finite machine (with m states)



From UCoB to UKCoB

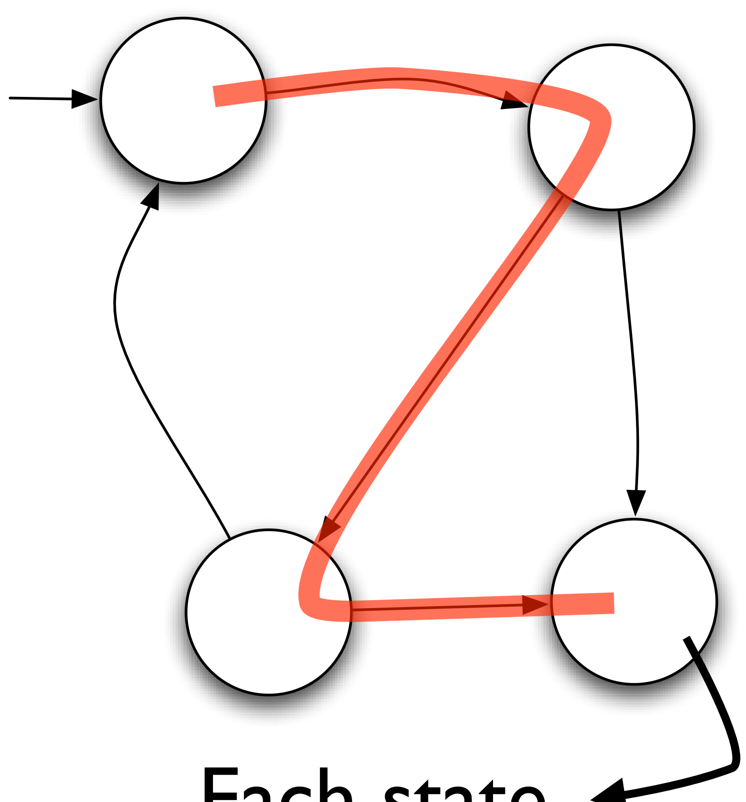
Theorem: Winning strategies of Player 1 on the UCoB automaton can be represented by a finite machine (with m states)



$(\sigma_1, t_1)\ (\sigma_2, t_2)\ (\sigma_3, t_3)\ldots$

From UCoB to UKCoB

Theorem: Winning strategies of Player 1 on the UCoB automaton can be represented by a finite machine (with m states)



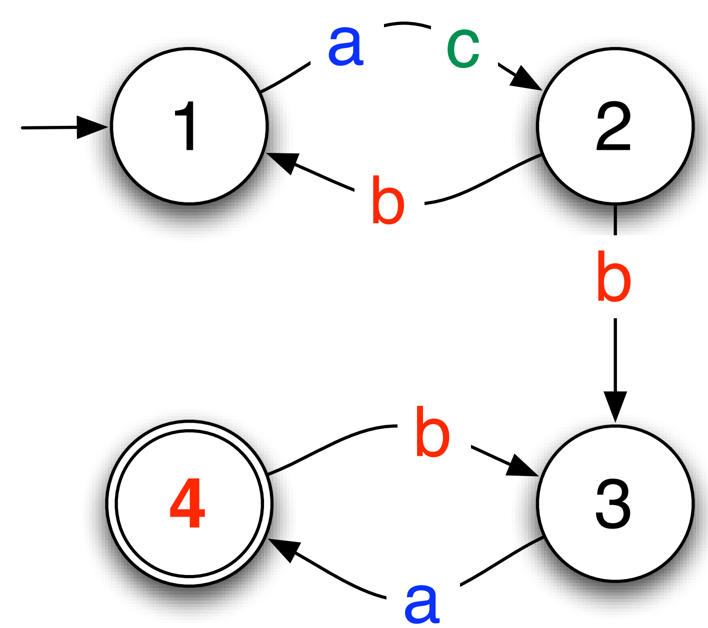
(σ_1, t_1) (σ_2, t_2) (σ_3, t_3)...

From UCoB to UKCoB

Strategy

m states

UCoB

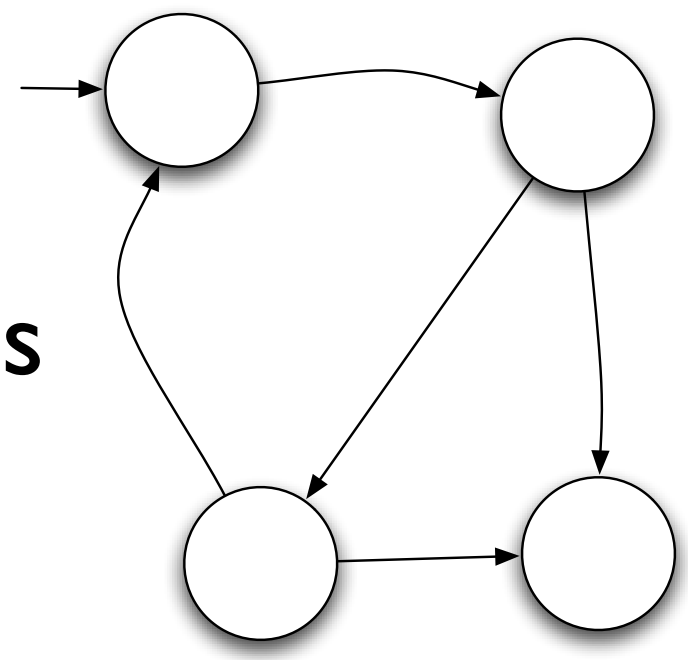


n states

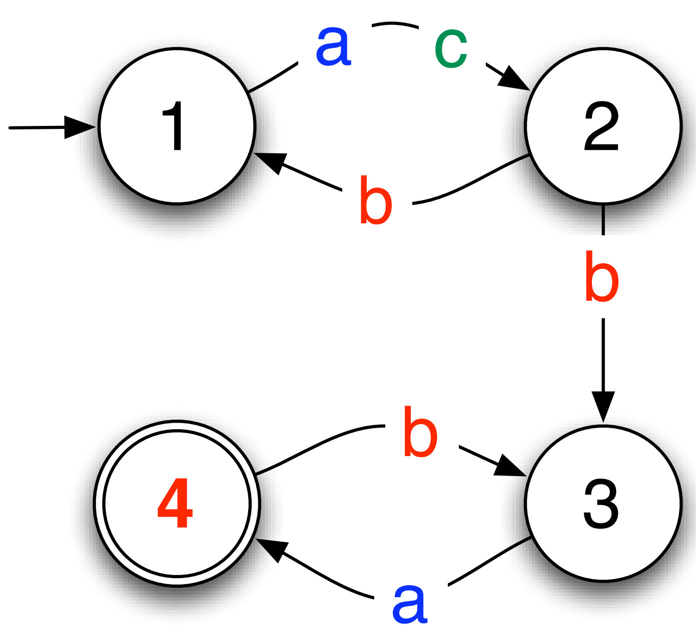
From UCoB to UKCoB

Strategy

m states



UCoB

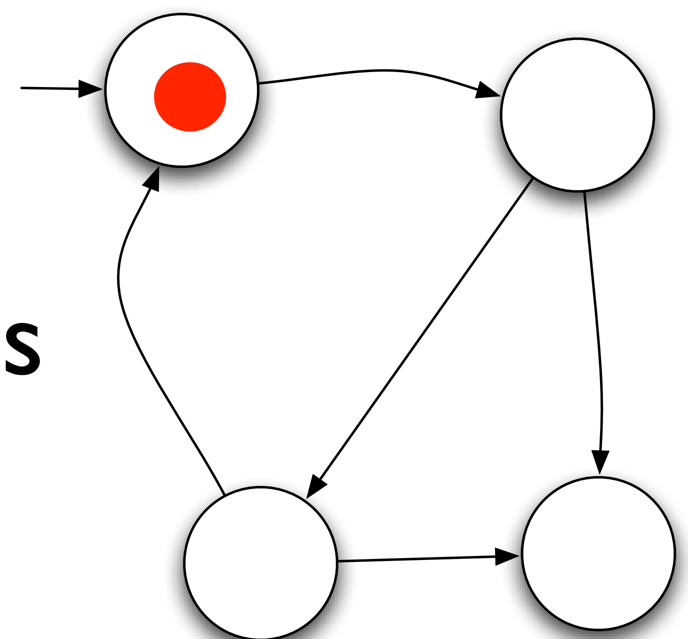


n states

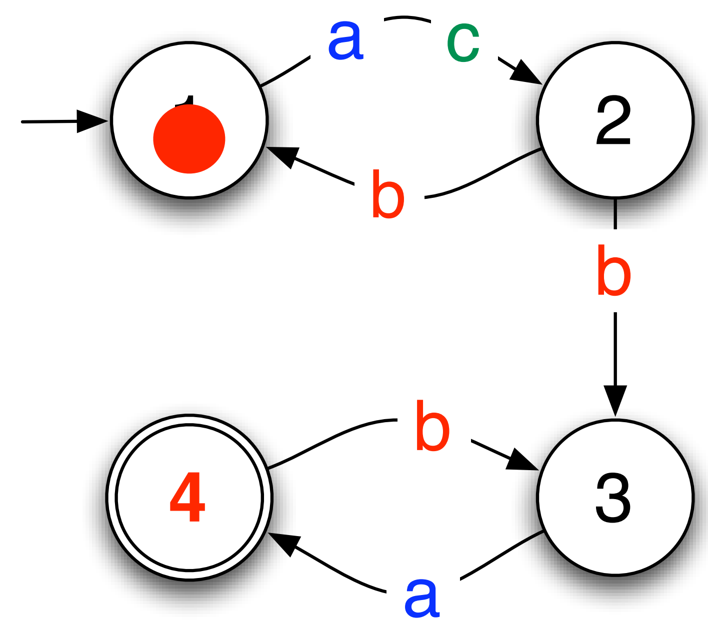
From UCoB to UKCoB

Strategy

m states



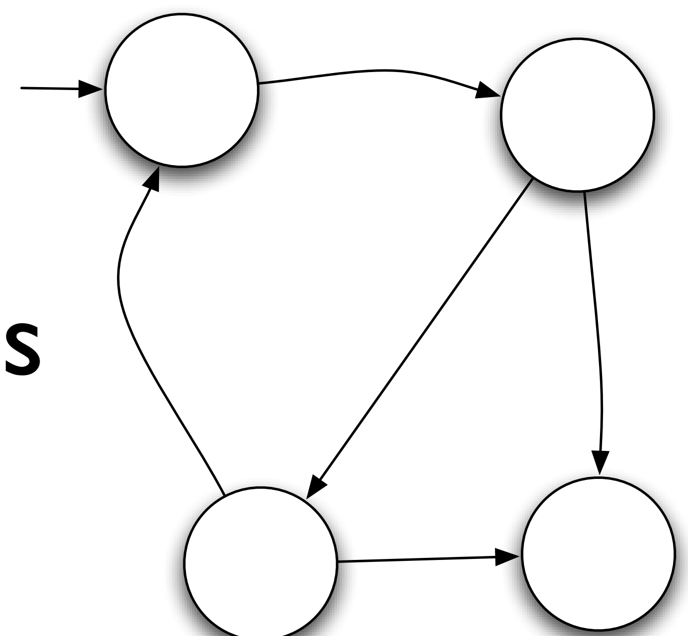
UCoB



n states

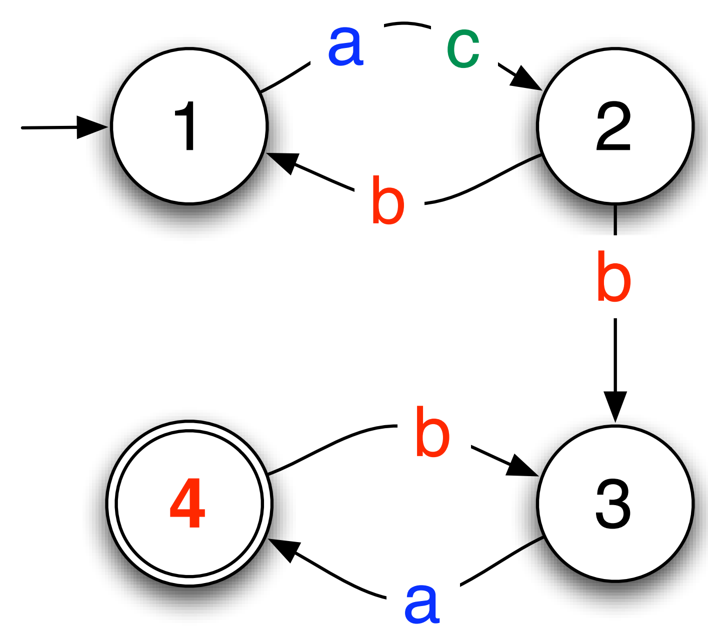
From UCoB to UKCoB

Strategy



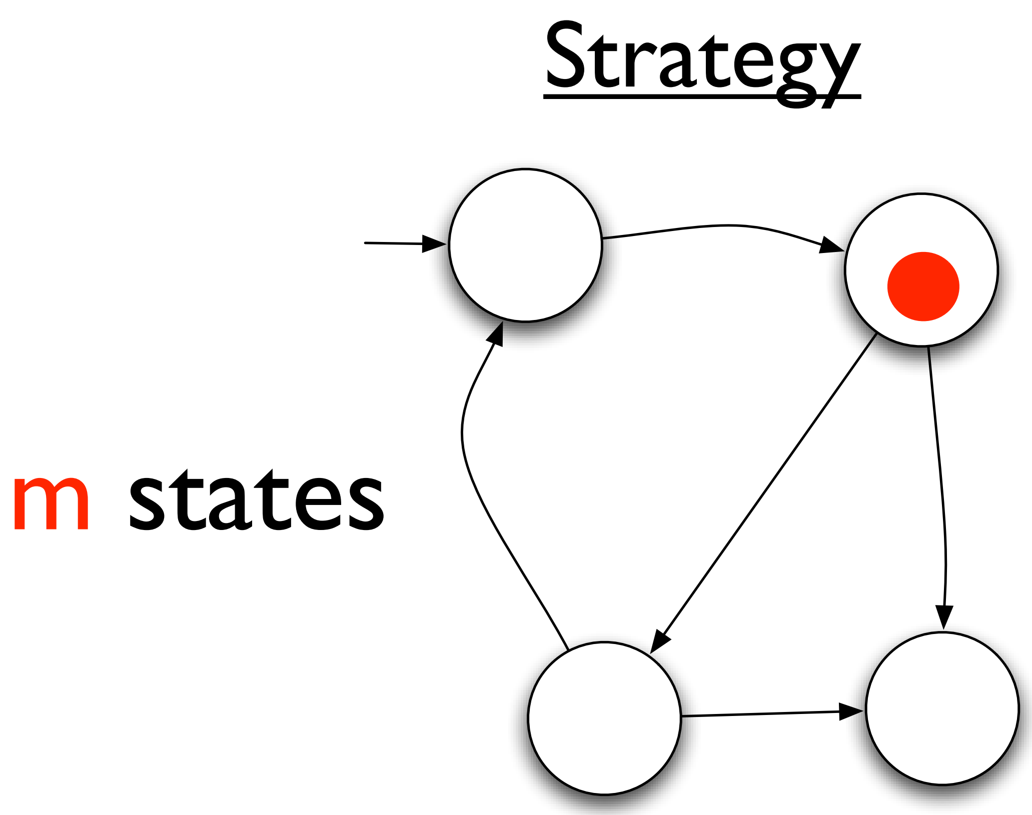
m states

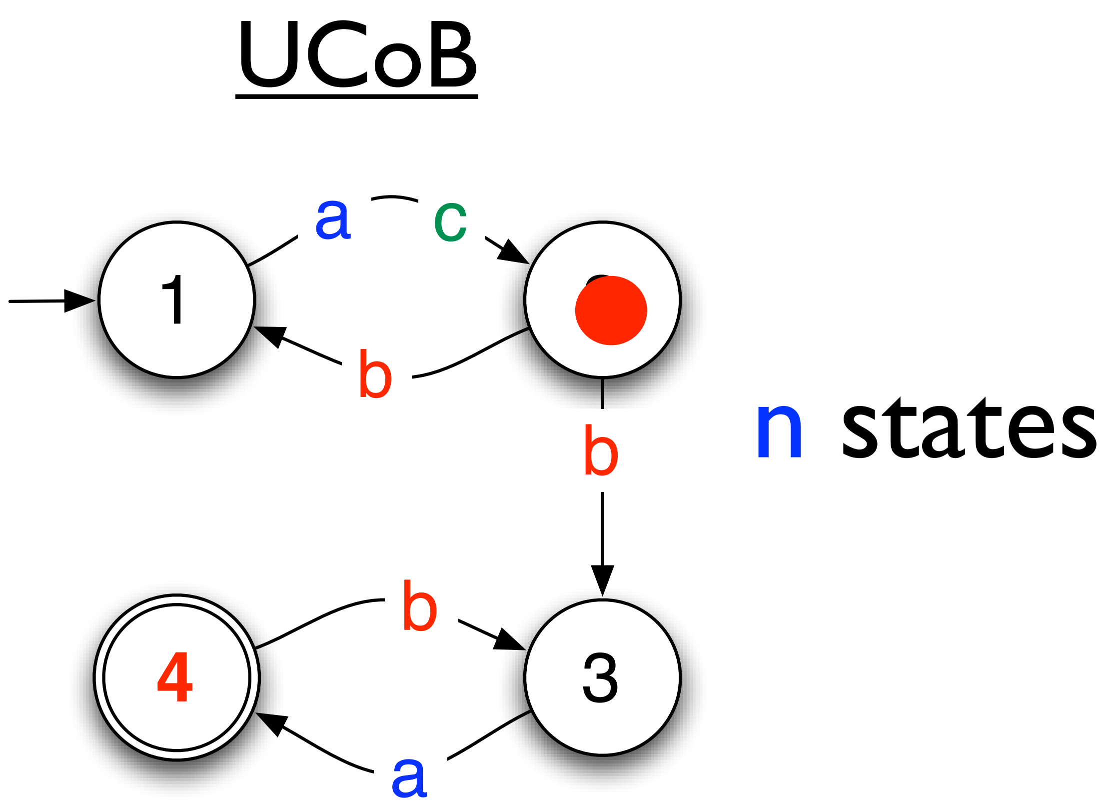
UCoB



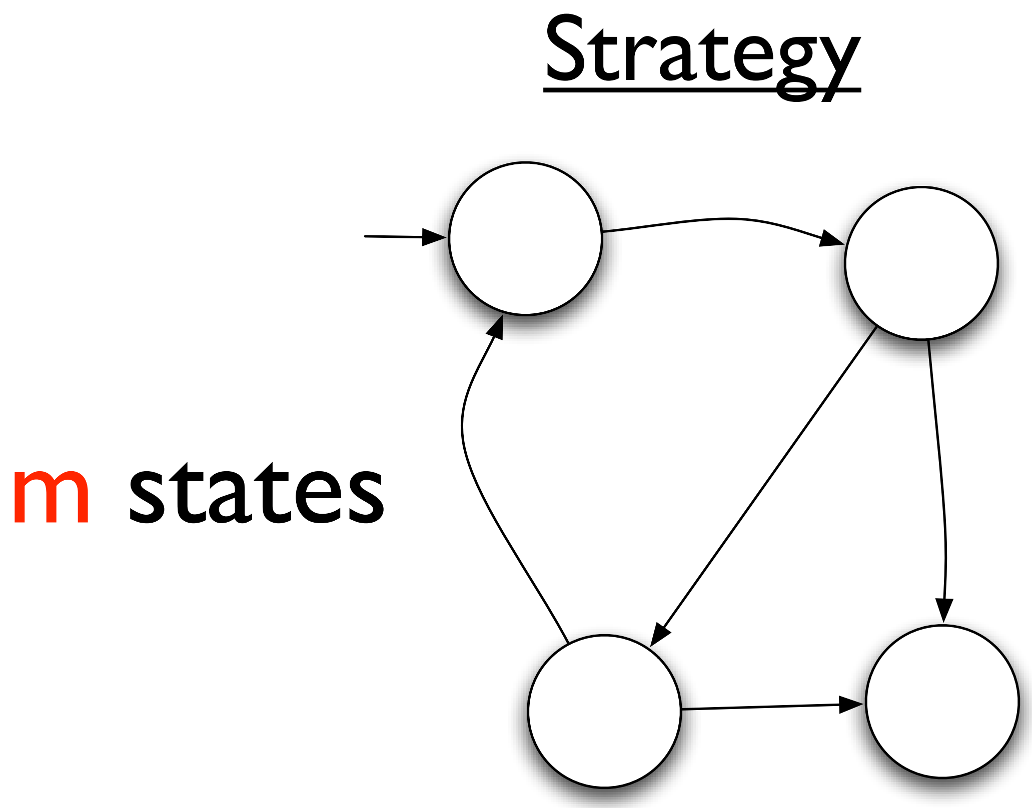
n states

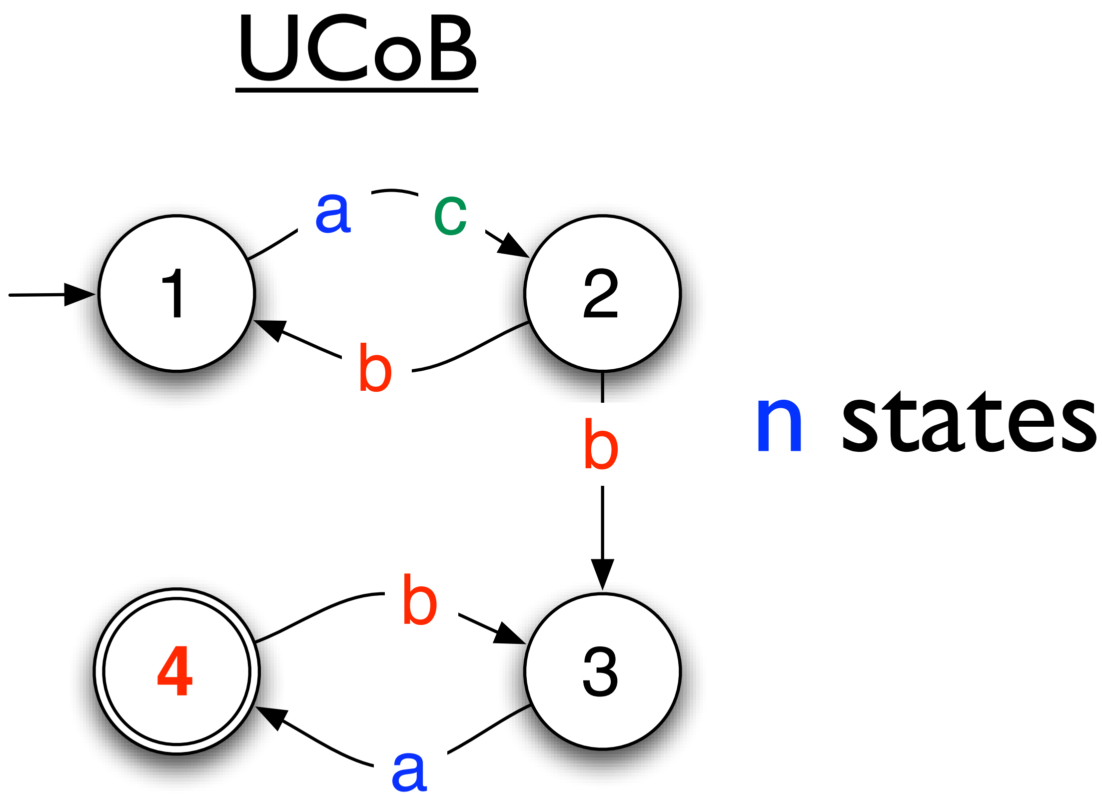
From UCoB to UKCoB





From UCoB to UKCoB

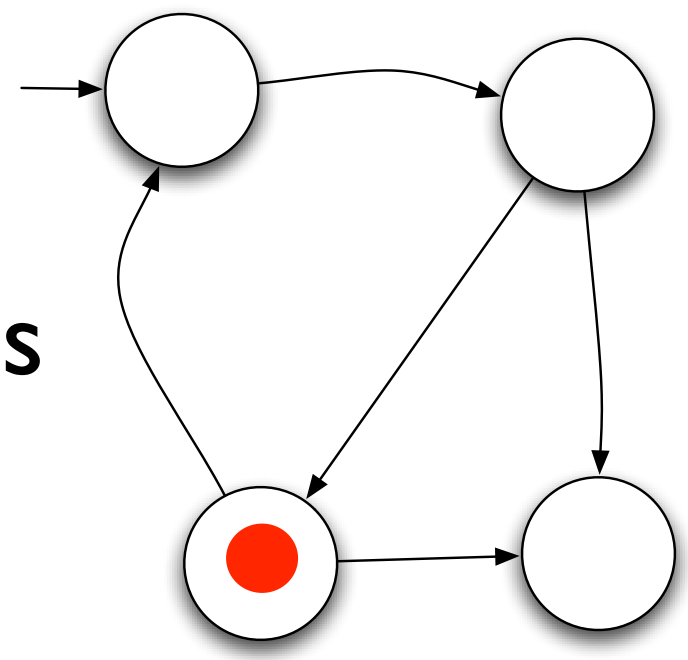




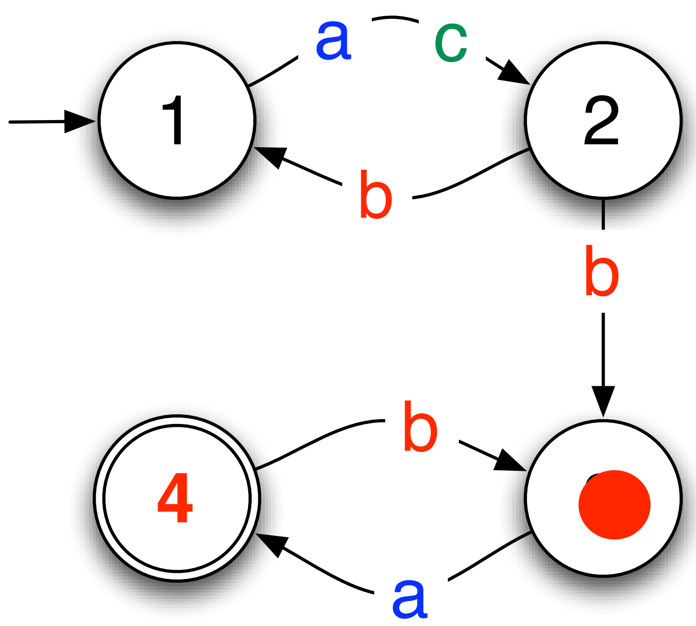
From UCoB to UKCoB

Strategy

m states

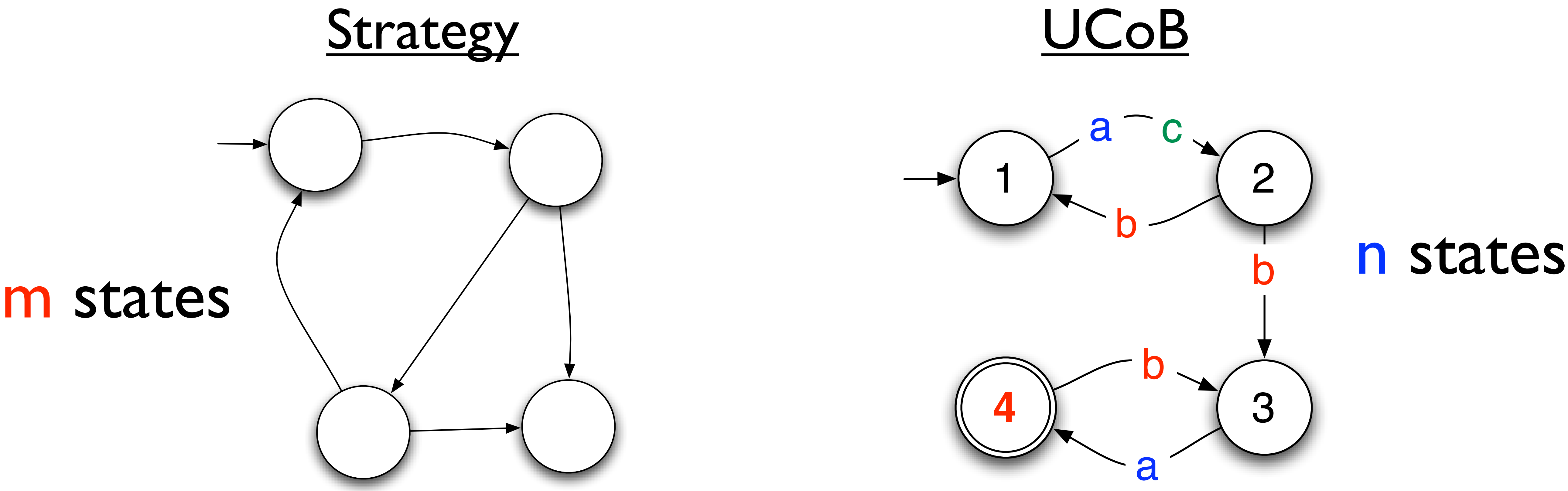


UCoB

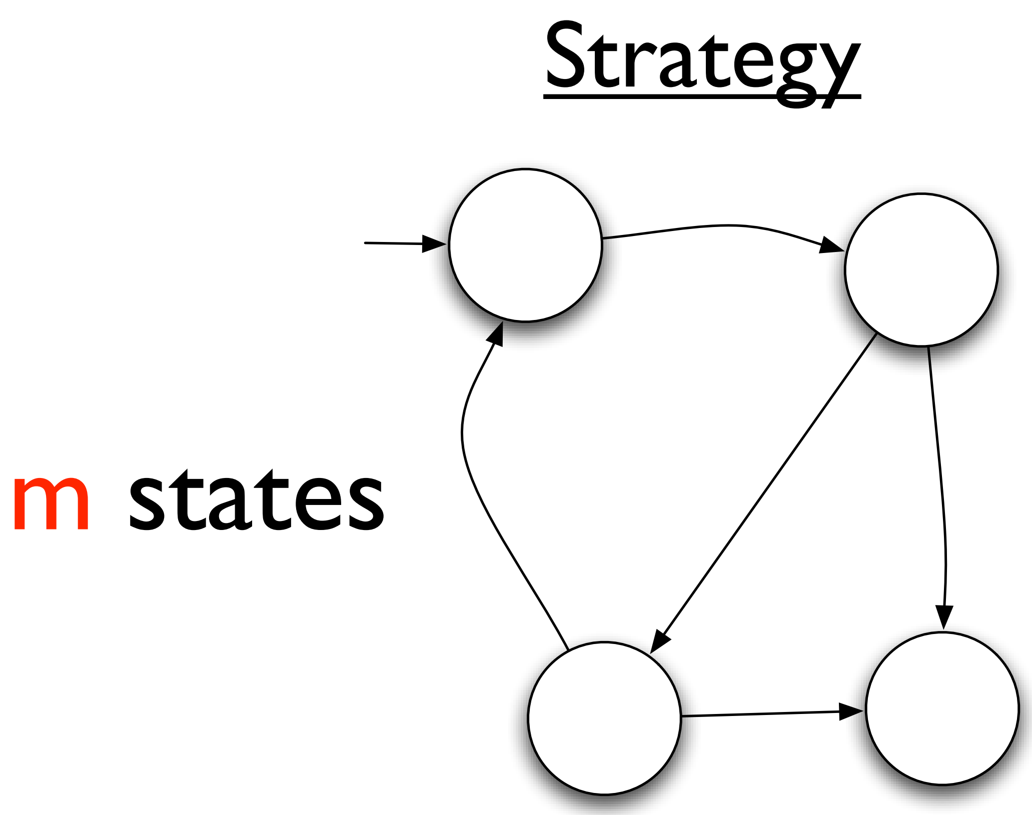


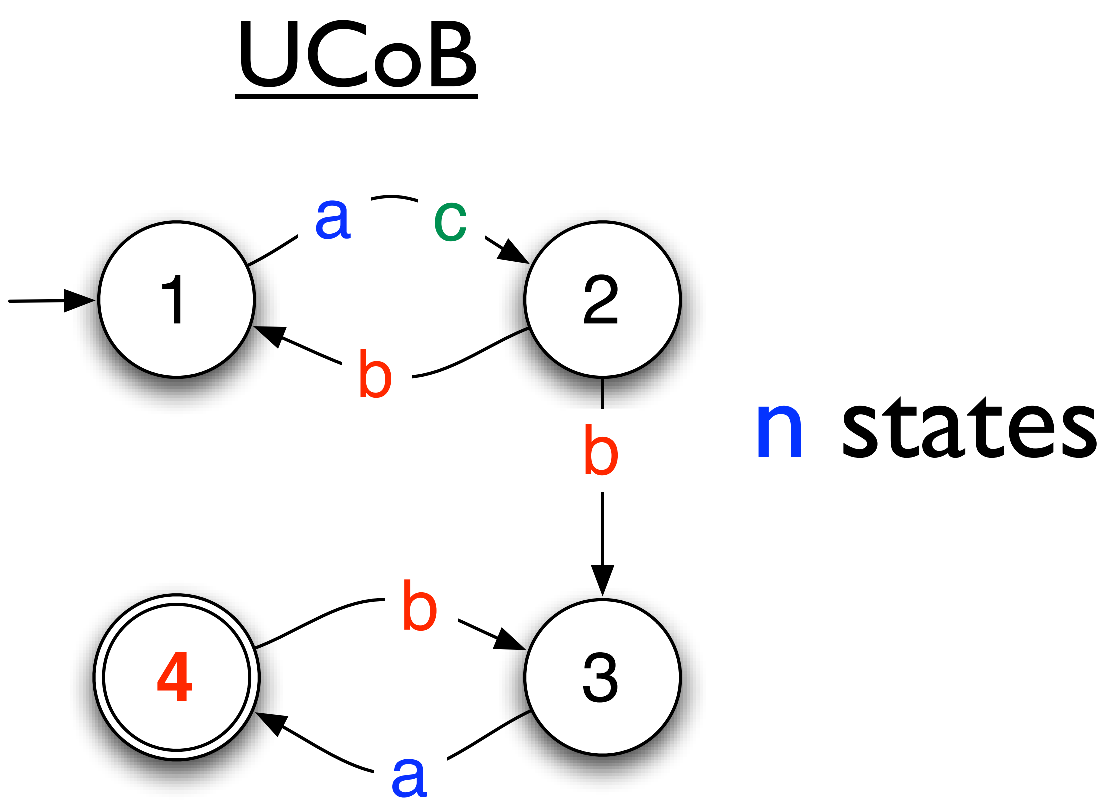
n states

From UCoB to UKCoB



From UCoB to UKCoB



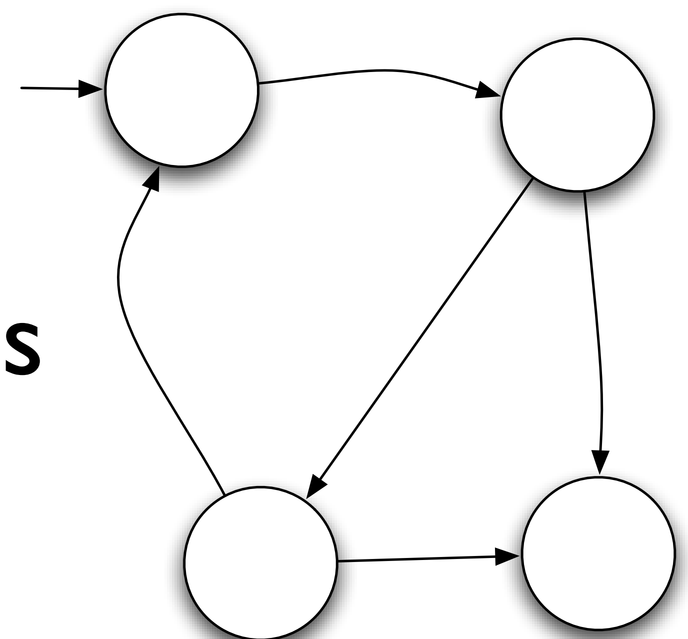


Assume the strategy lets us visit an accepting state more than n×m times

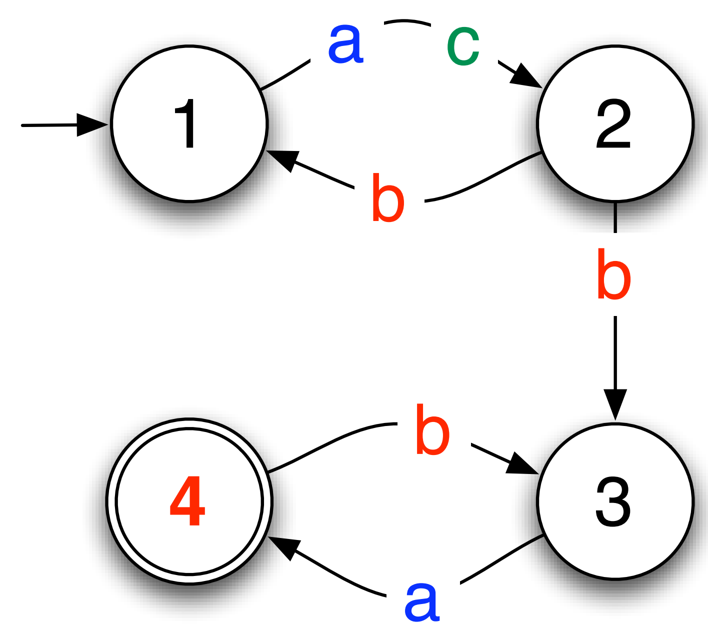
From UCoB to UKCoB

Strategy

m states



UCoB



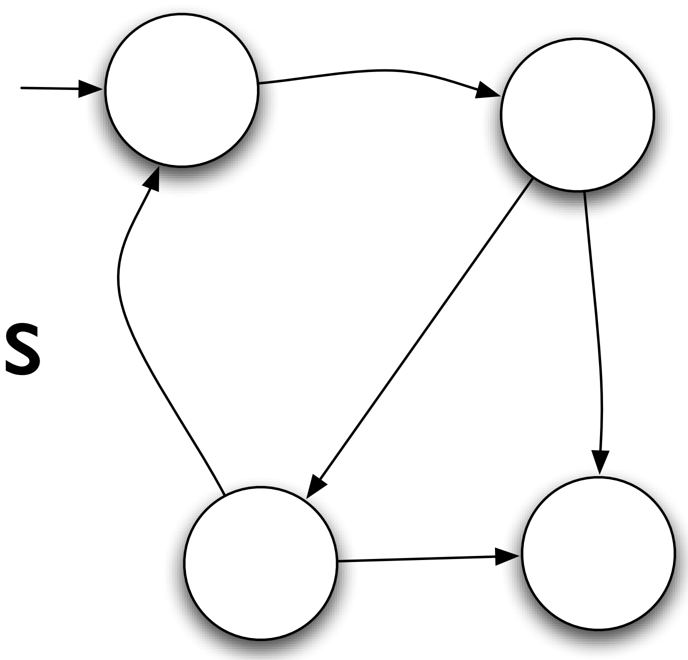
n states

Assume the strategy lets us visit an accepting state more than n×m times

☞ cycle in the product of the strategy and the UCoB

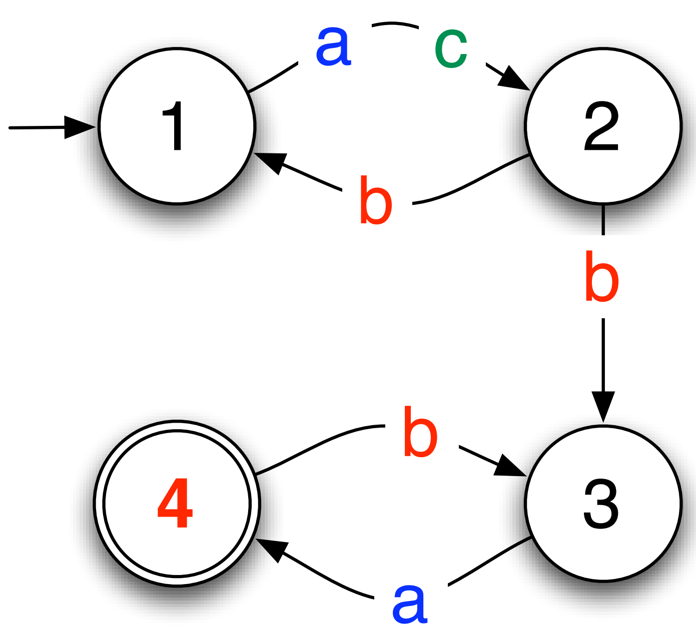
From UCoB to UKCoB

Strategy



m states

UCoB



n states

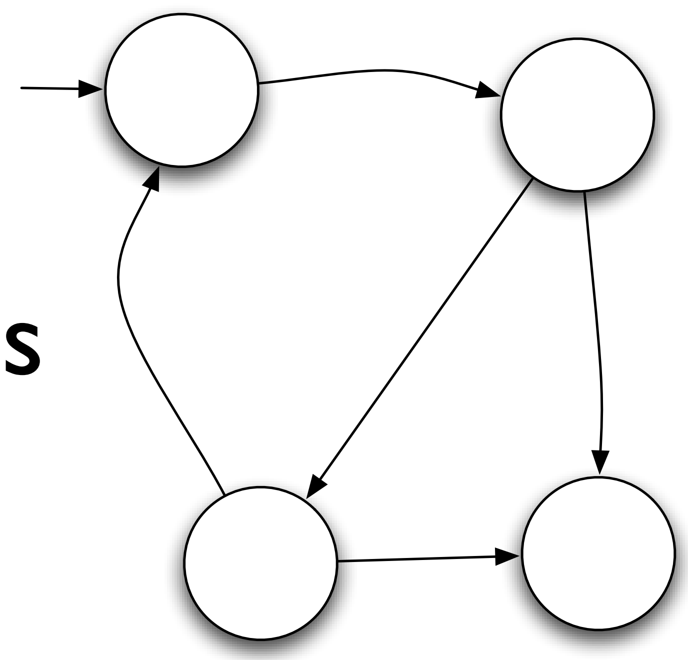
Assume the strategy lets us visit an accepting state more than n×m times

☞ cycle in the product of the strategy and the UCoB

☞ accepting states are visited infinitely often

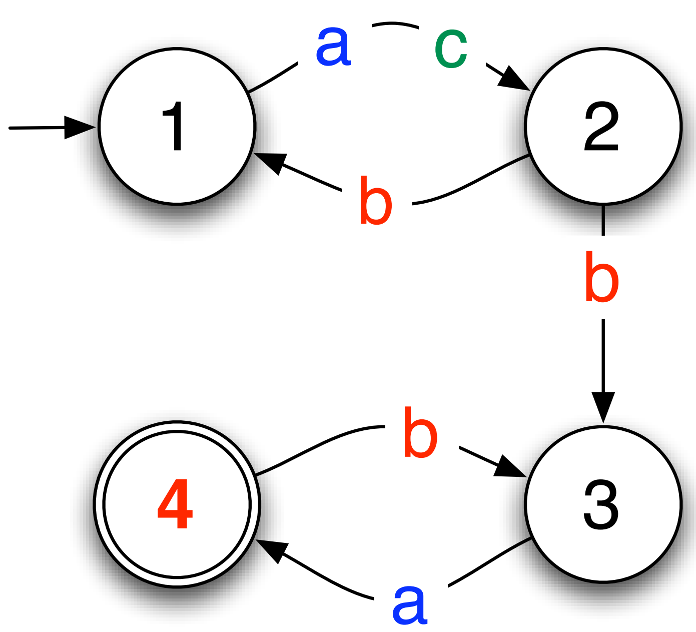
From UCoB to UKCoB

Strategy



m states

UCoB



n states

Assume the strategy lets us visit an accepting state more than n×m times

☞ cycle in the product of the strategy and the UCoB

☞ accepting states are visited infinitely often

☞ the strategy is **not winning**

From UCoB to UKCoB

Theorem: Player 1 has a winning strategy in
$\langle \Sigma_1, \Sigma_2, L_{UcoB}(A_\varphi) \rangle$

iff

she has a winning strategy in
$\langle \Sigma_1, \Sigma_2, L_{UKcoB}(A_\varphi) \rangle$ for $K=n \times m$

We can thus solve the game by playing with the (weaker) **K-Co-Büchi** acceptance condition

K-Co-Büchi = avoid visiting accepting states too often
= **safety condition** !

Incremental procedure

Theorem: **If** Player 1 has a winning strategy in

$\langle \Sigma_1, \Sigma_2, L_{UKcoB}(A_\varphi) \rangle$

then

she has a winning strategy in

$\langle \Sigma_1, \Sigma_2, L_{UK'coB}(A_\varphi) \rangle$ for $K' \geq K$

Incremental procedure

Theorem: **If** Player 1 has a winning strategy in
$\langle \Sigma_1, \Sigma_2, L_{UKcoB}(A_\varphi) \rangle$

then

she has a winning strategy in
$\langle \Sigma_1, \Sigma_2, L_{UK'coB}(A_\varphi) \rangle$ for $K' \geq K$

```
i := 0
While(true)
  If P1 wins on LUicoB(Aφ) return «win»
  Else if P2 wins on LUicoB(A¬φ) return «lose»
  Else i:=i+1
```

Incremental procedure

Theorem: If Player I has a winning strategy in

Each step can be computed by **solving** a safety game

she has a winning strategy in

$\langle \Sigma_1, \Sigma_2, L_{UK'coB}(A_\varphi) \rangle$ for $K' \geq K$

```
i := 0
While(true)
  If P1 wins on LUicoB(Aφ) return «win»
  Else if P2 wins on LUicoB(A¬φ) return «lose»
  Else i:=i+1
```

Incremental procedure

Theorem: **If** Player I has a winning strategy in

she has a winning strategy in

$\langle \Sigma_I,$

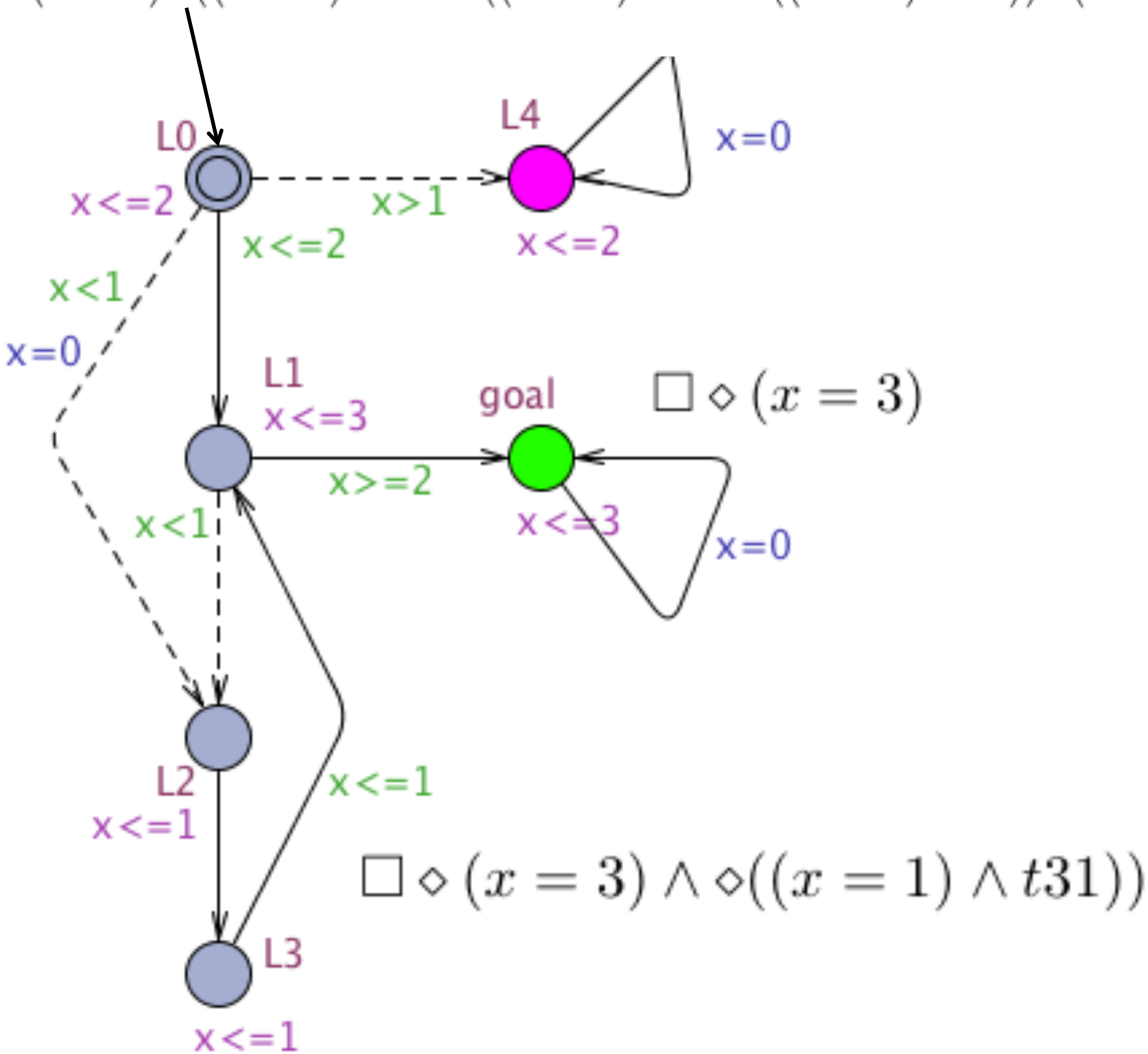
Each step can be computed by **solving** a safety game

In practice this algorithm **might terminate** with **small** values of **i**

```
i := 0
While(true)
  If P1 wins on LUicoB(Aφ) return «win»
  Else if P2 wins on LUicoB(A¬φ) return «lose»
  Else i:=i+1
```

Initial example

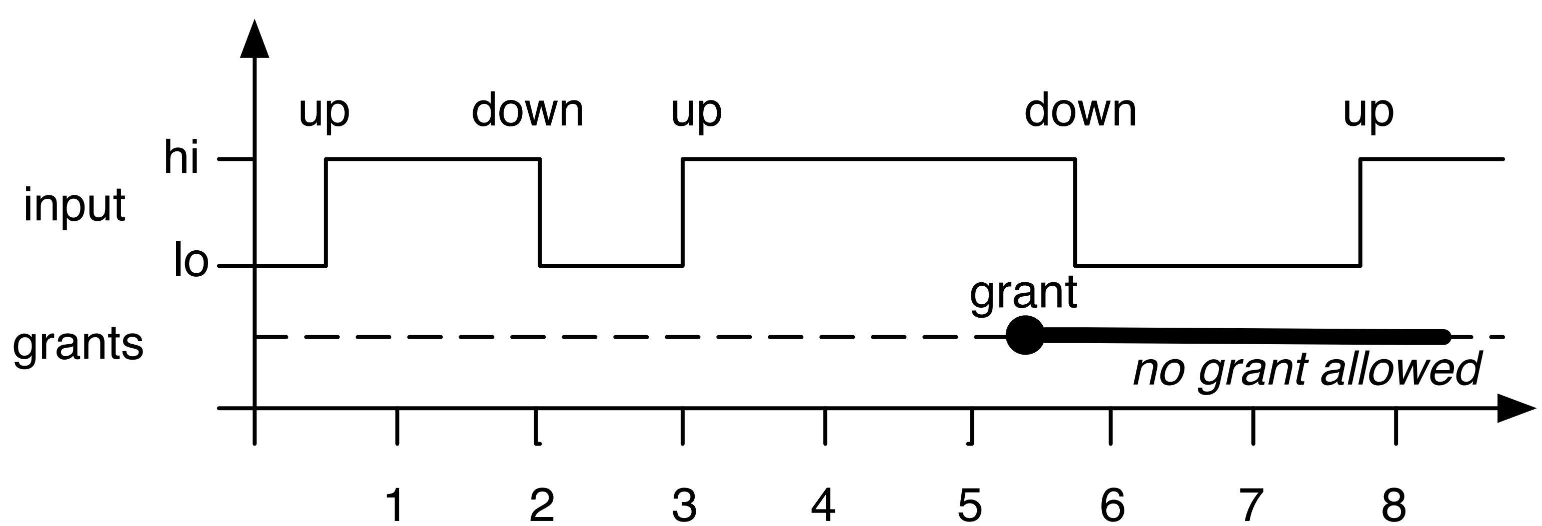
$\Box\Diamond(x=3)\wedge((x<1)\wedge t02\wedge\Diamond((x=1)\wedge t23\wedge\Diamond((x=1)\wedge t31))\vee(t01\wedge(x=1))\wedge\Diamond((x=1)\mathcal{U}(t1g\wedge(x=2))))$



Example

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

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



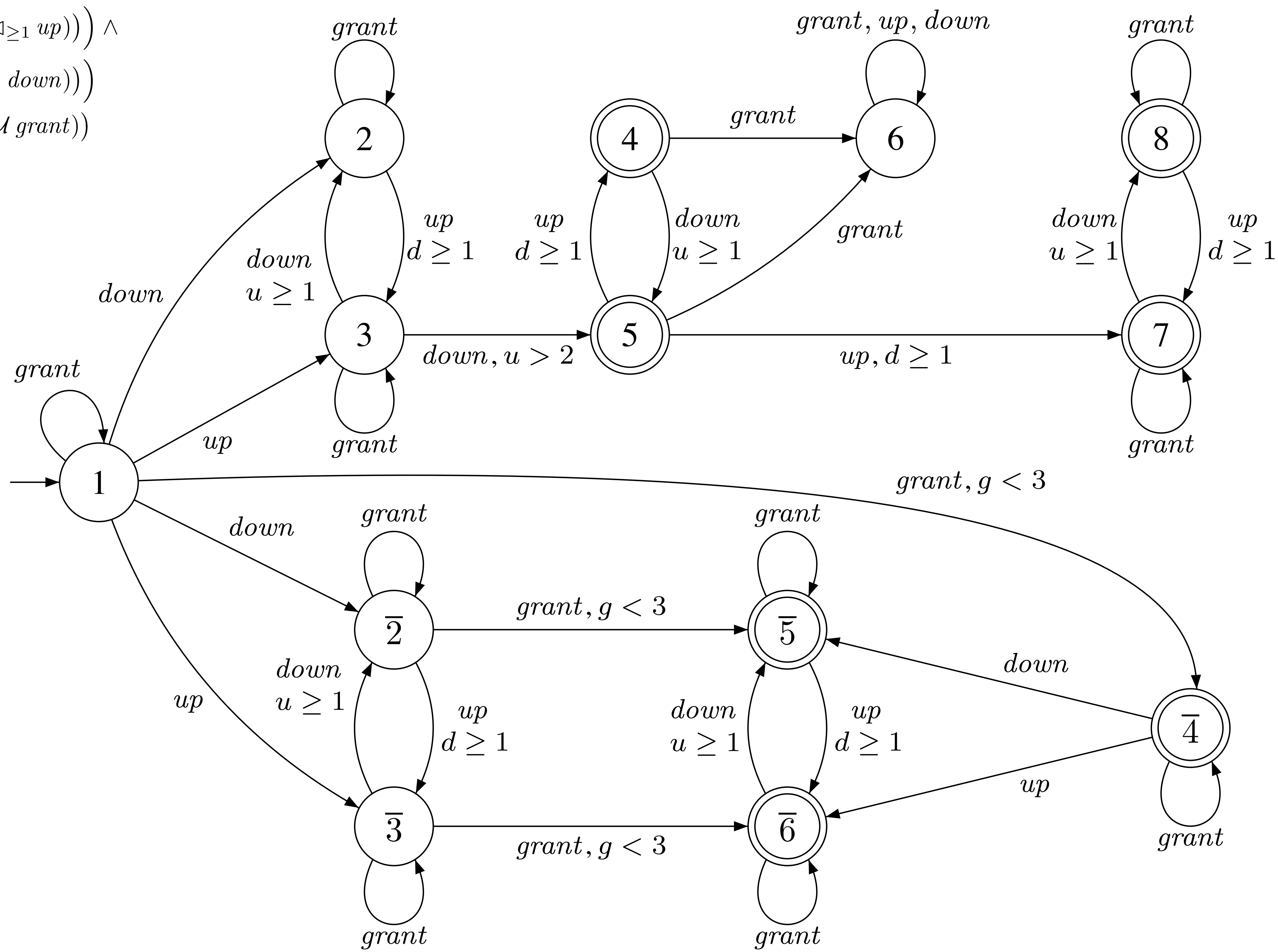
Example

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

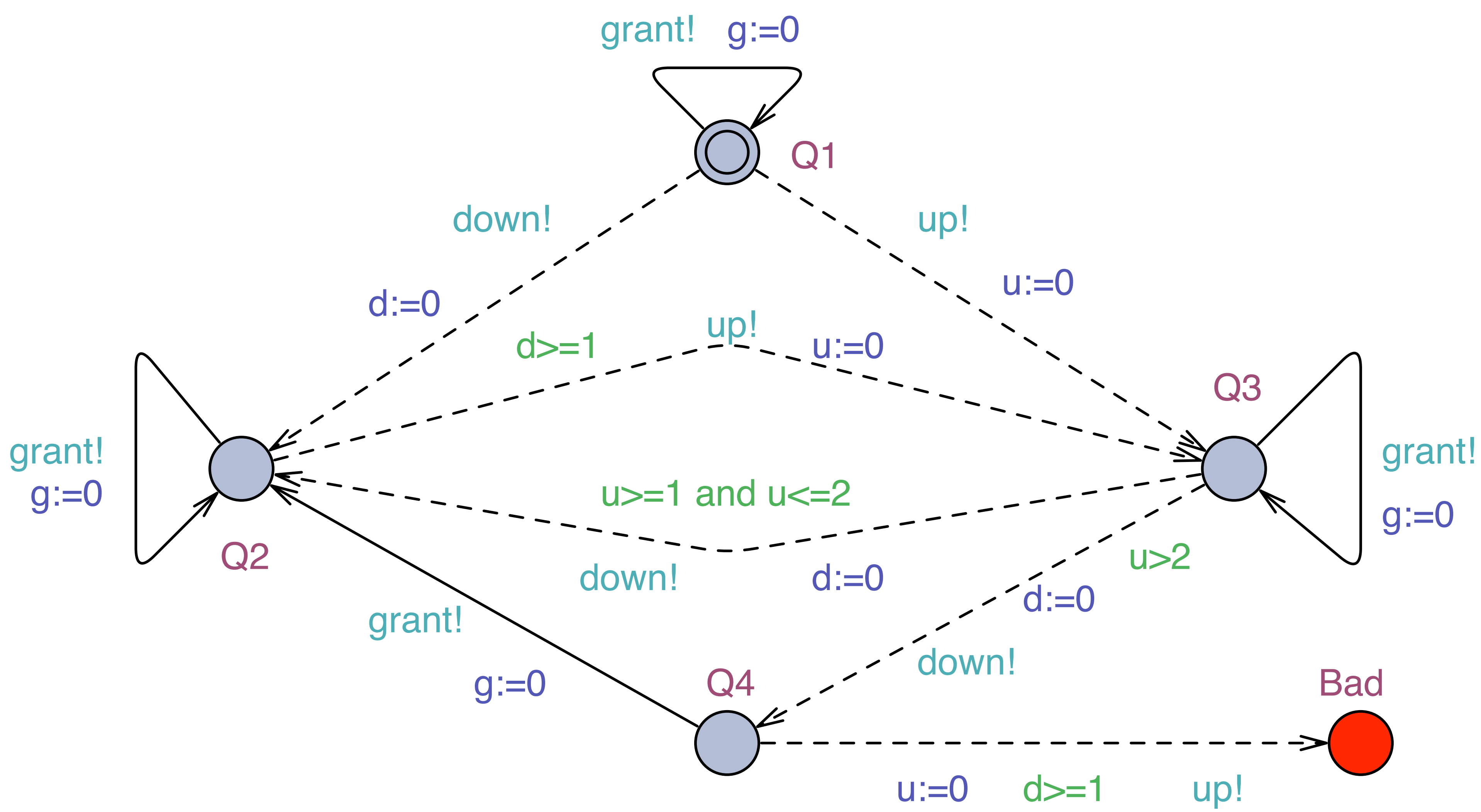
$\mathsf{Req}_1 \equiv \Box\big((down \wedge \lhd_{>2}\, up) \rightarrow (\neg up\, \mathcal{U}\, grant)\big)$

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

using extension of classical constructions

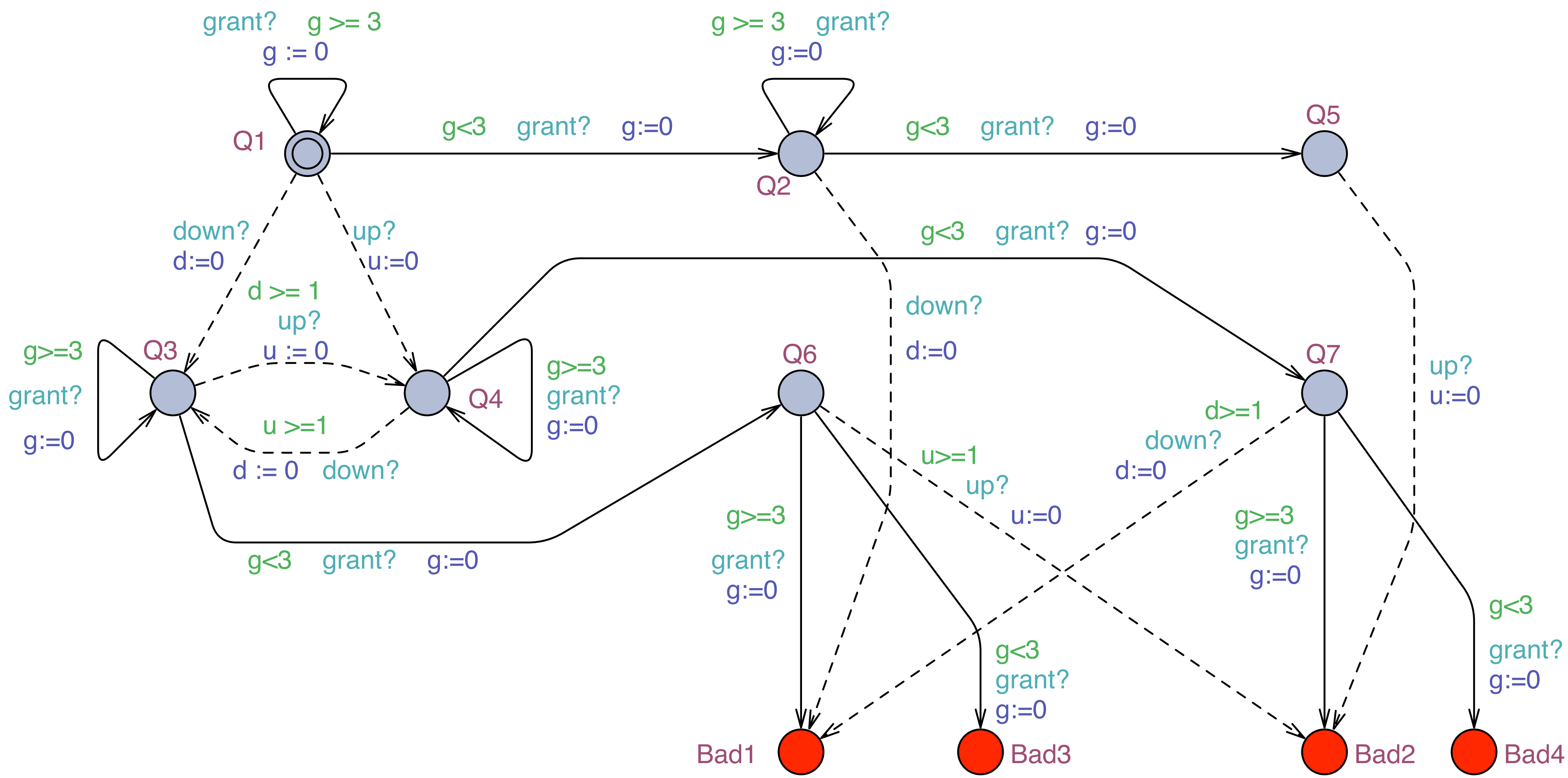


Illustration



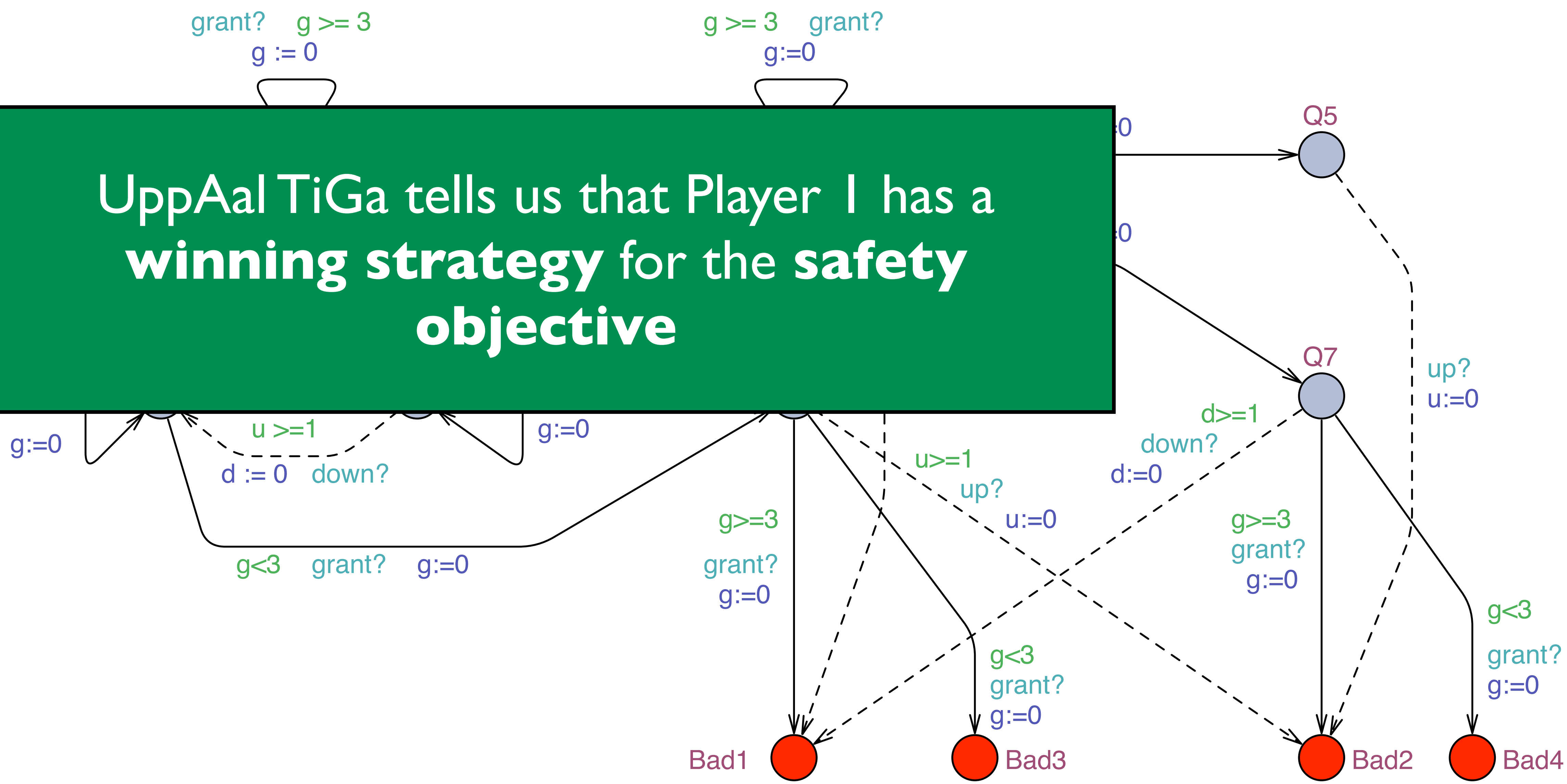
For K=1

Illustration



For K=1

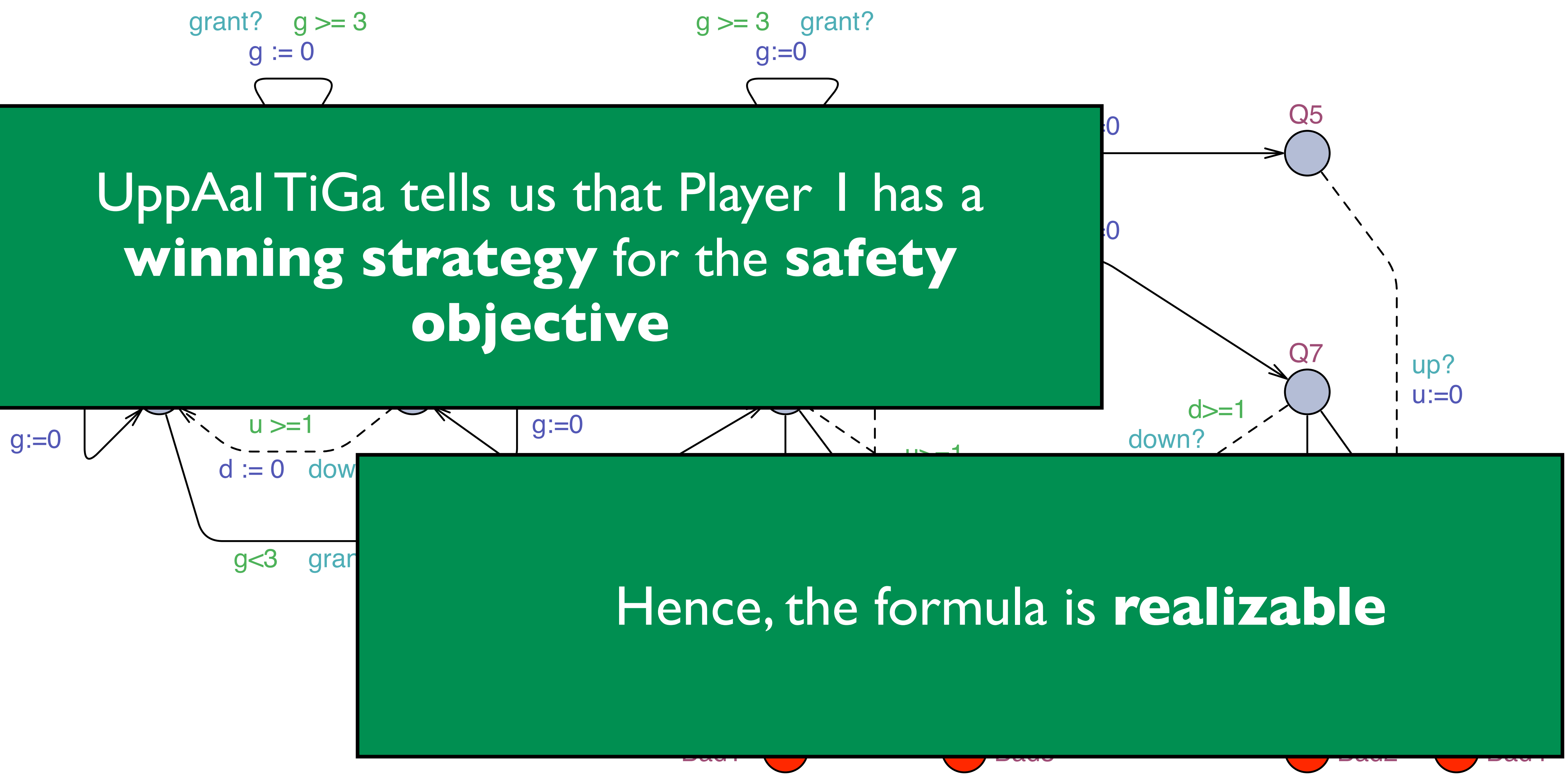
Illustration



UppAal TiGa tells us that Player 1 has a **winning strategy** for the **safety objective**

For K=1

Illustration



UppAal TiGa tells us that Player 1 has a **winning strategy** for the **safety objective**

Hence, the formula is **realizable**

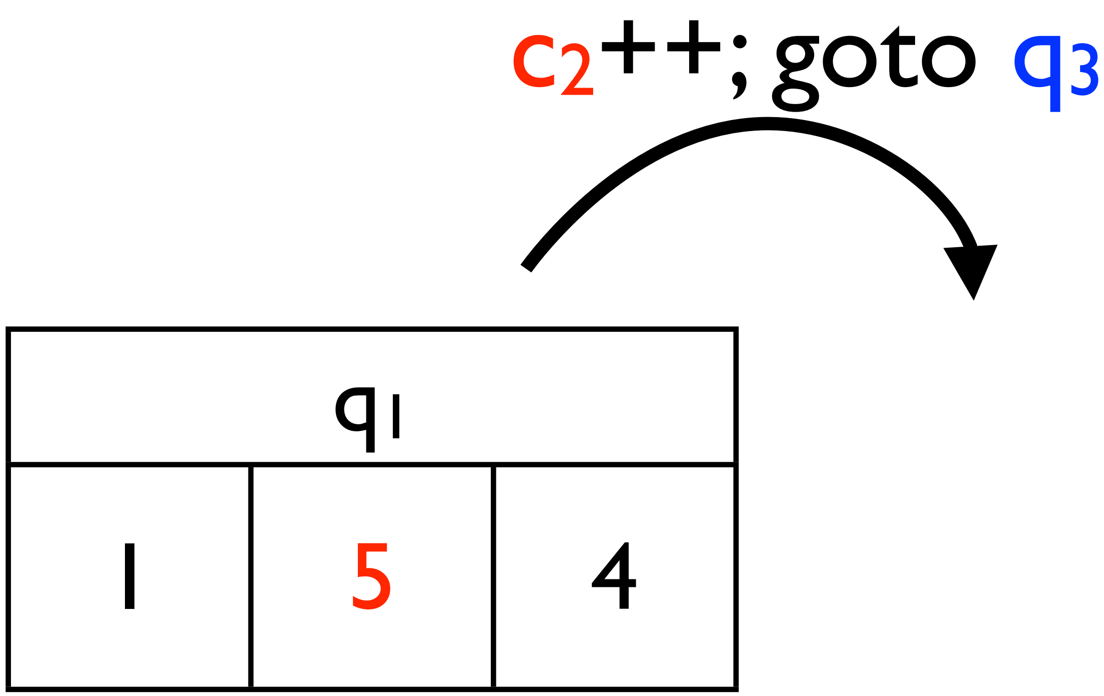
For K=1

Questions ?

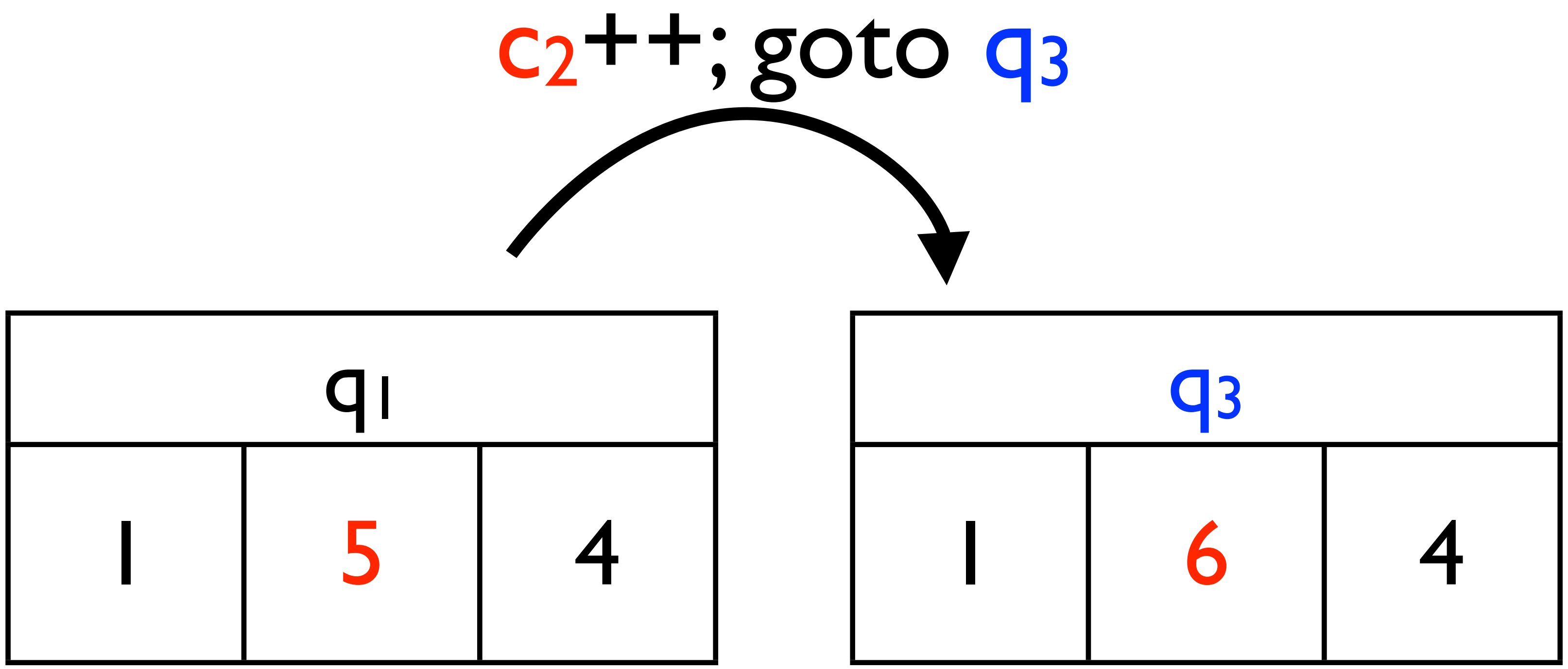
Counter machine - run

q_1		
1	5	4

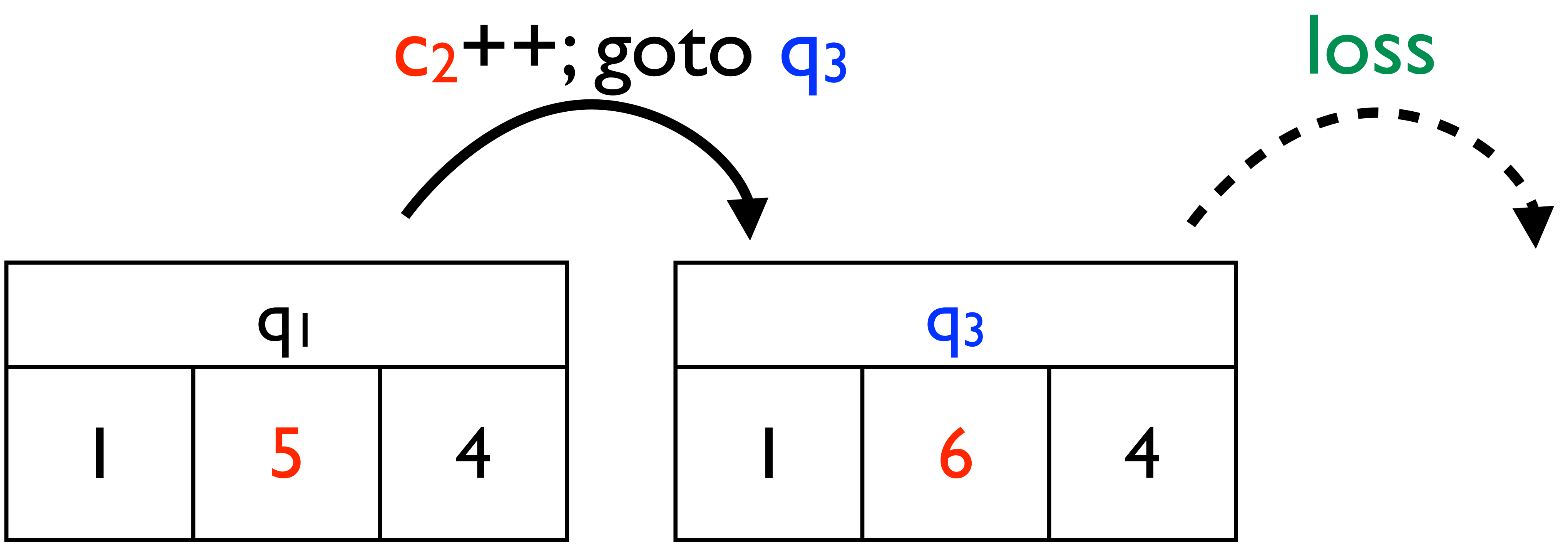
Counter machine - run



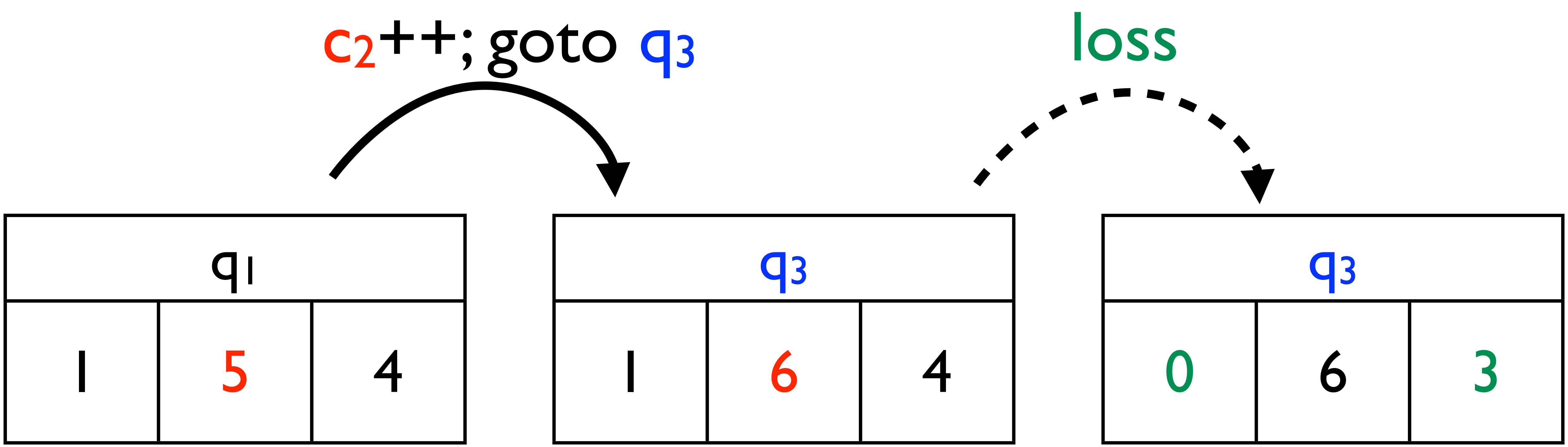
Counter machine - run



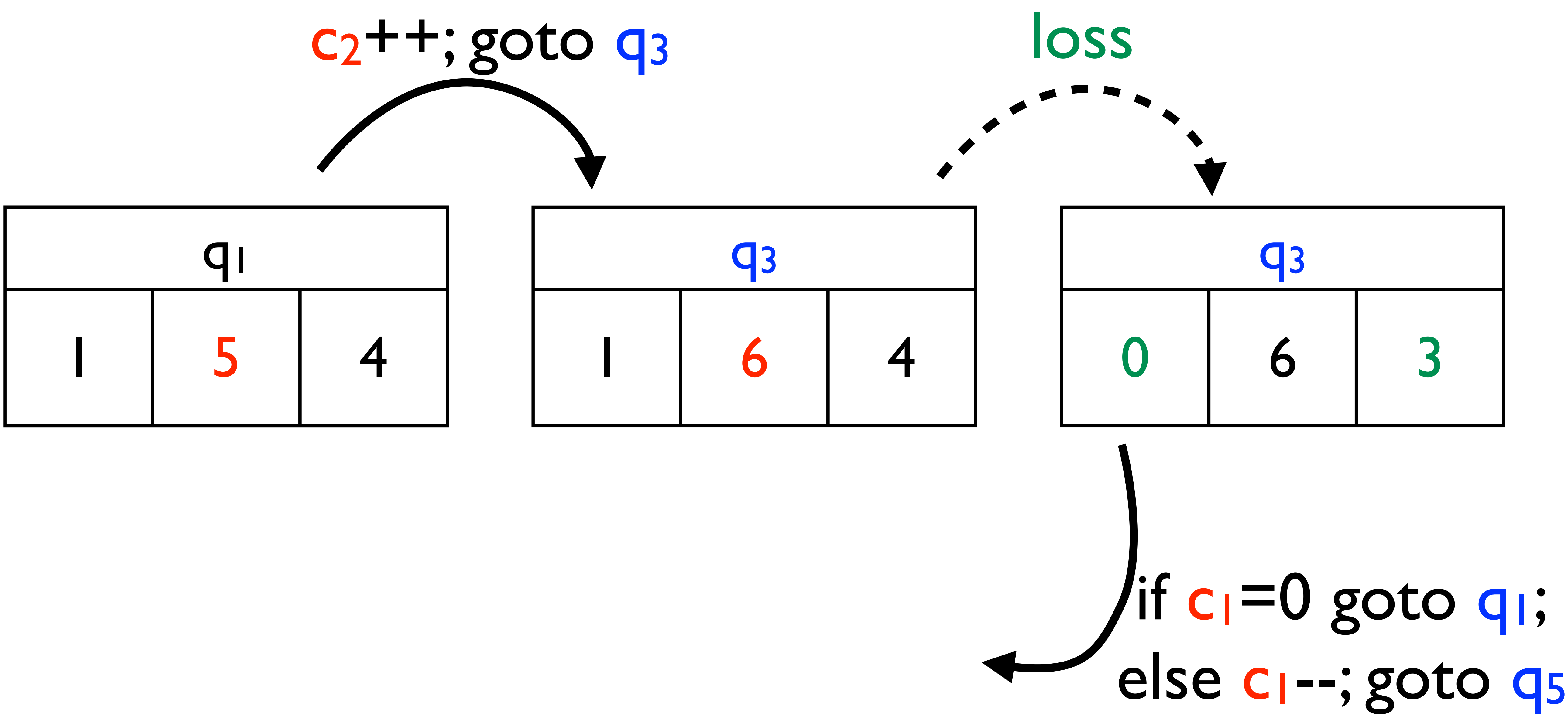
Counter machine - run



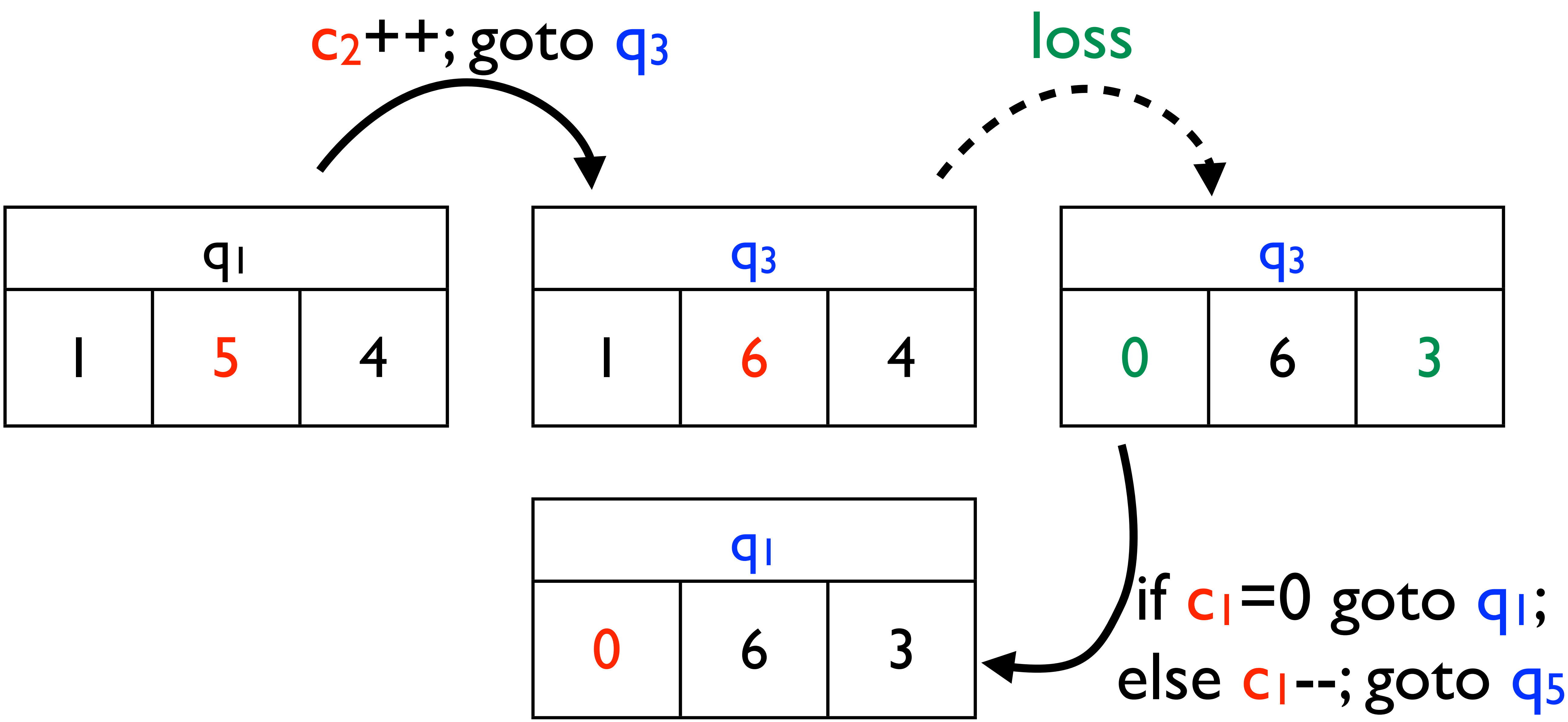
Counter machine - run



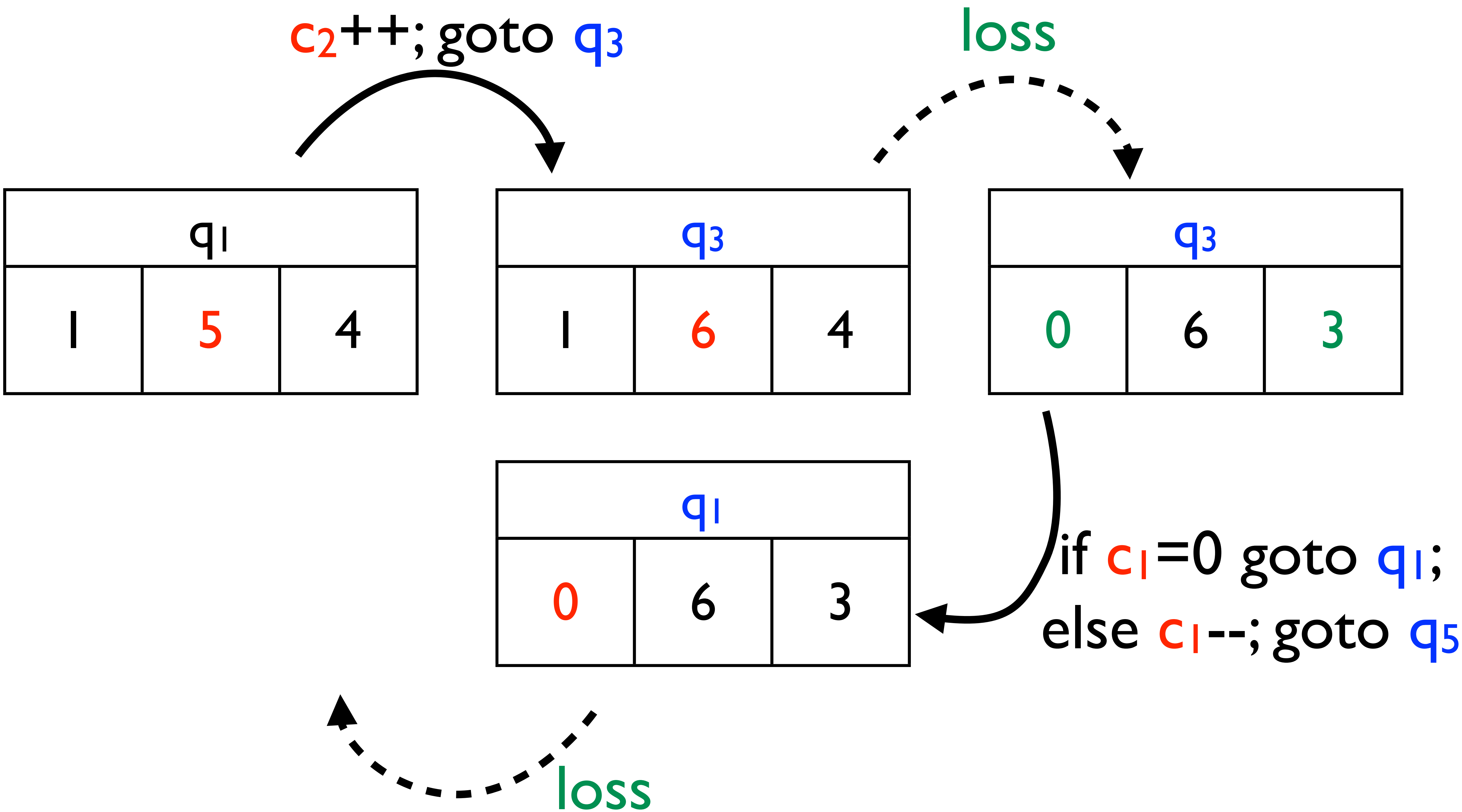
Counter machine - run



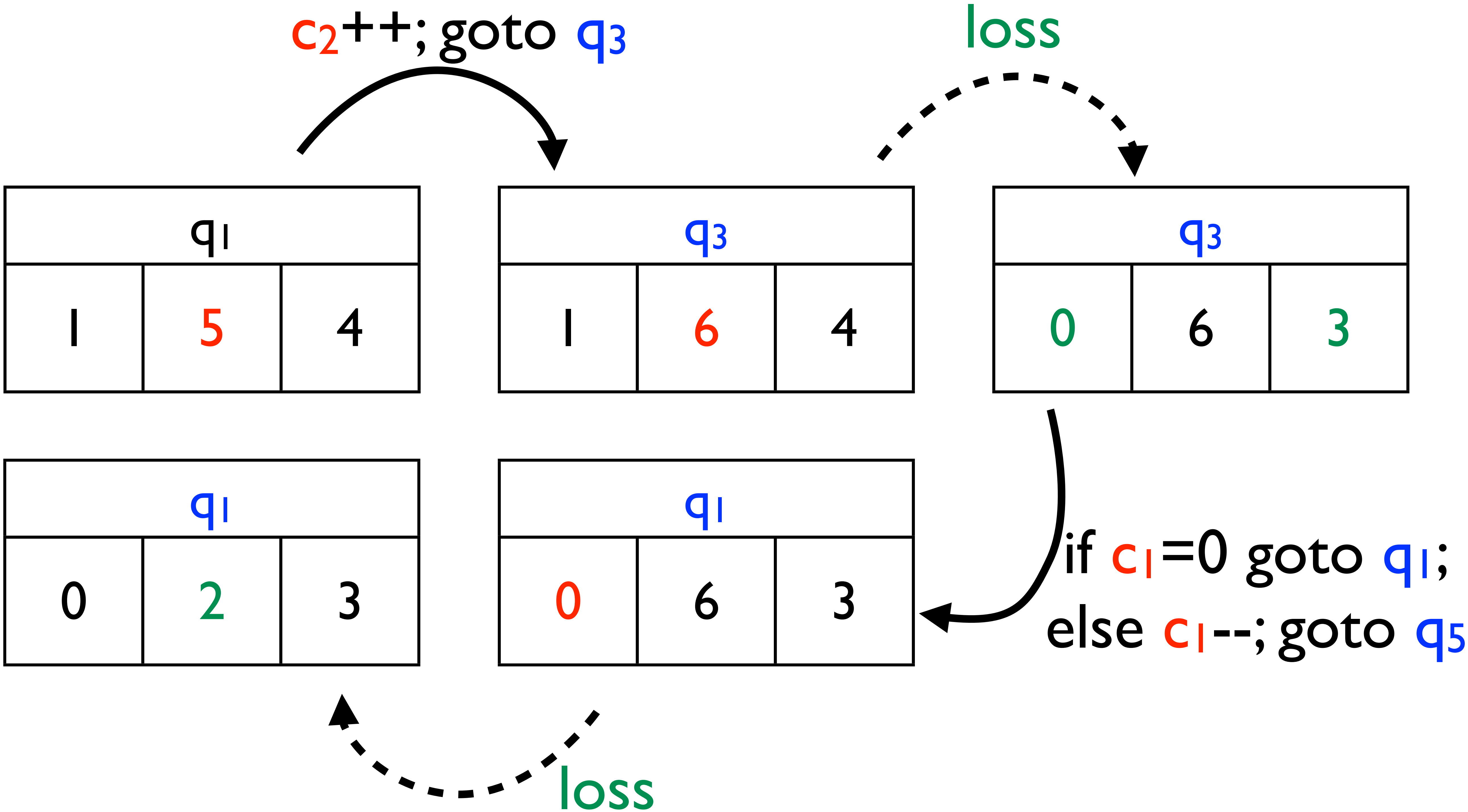
Counter machine - run



Counter machine - run



Counter machine - run

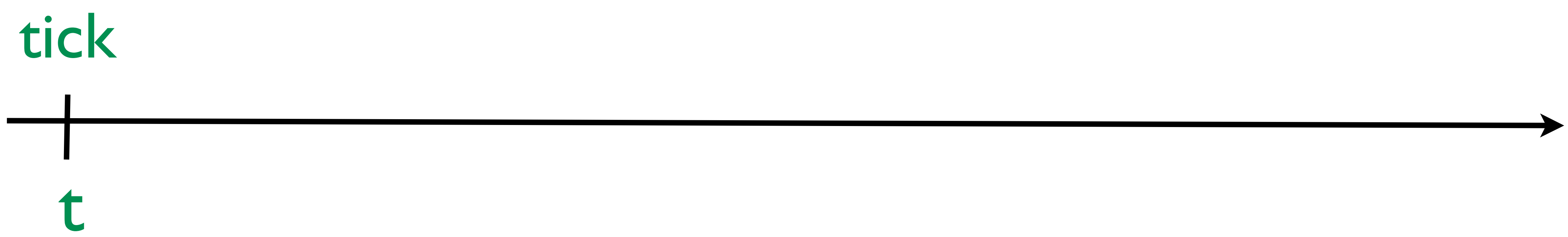


Encoding runs

q		
1	3	0

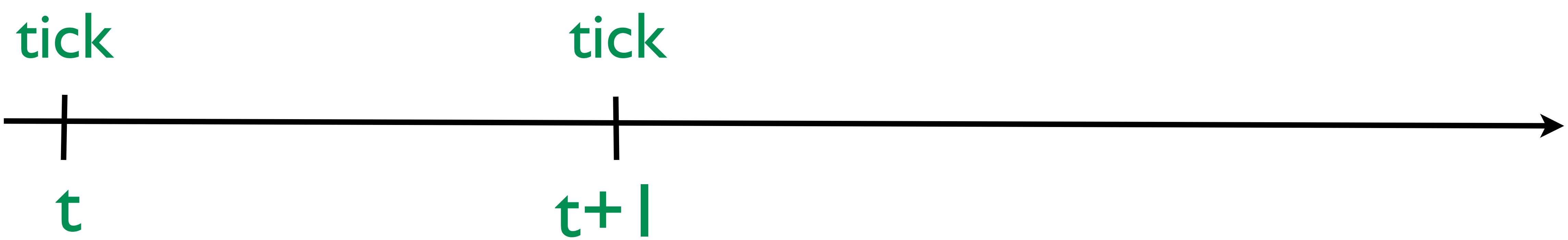
Encoding runs

q		
1	3	0



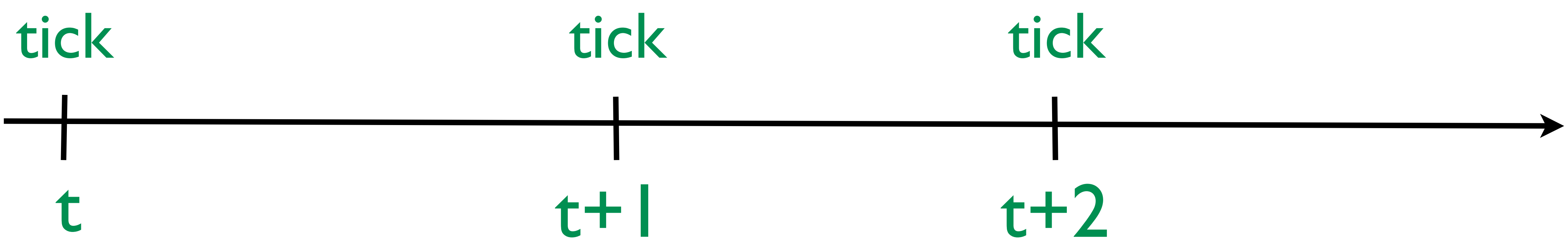
Encoding runs

q		
1	3	0



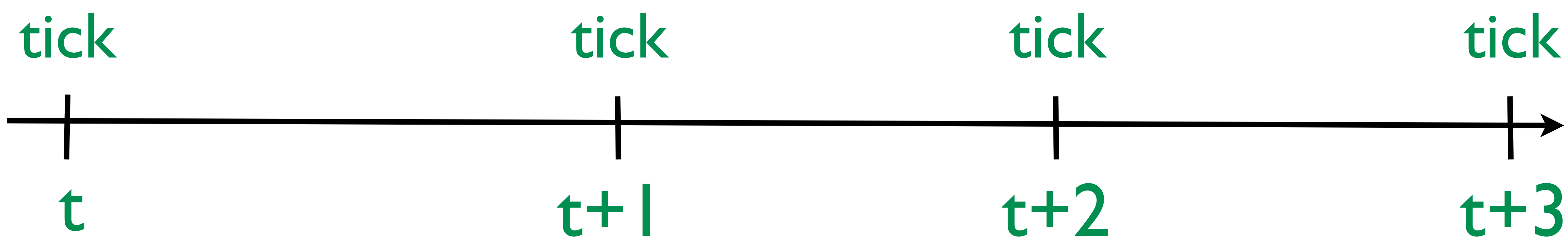
Encoding runs

q		
1	3	0



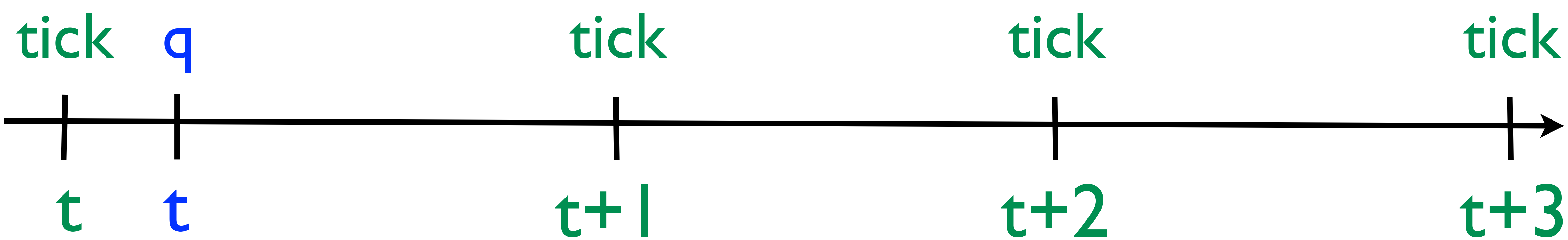
Encoding runs

q		
1	3	0



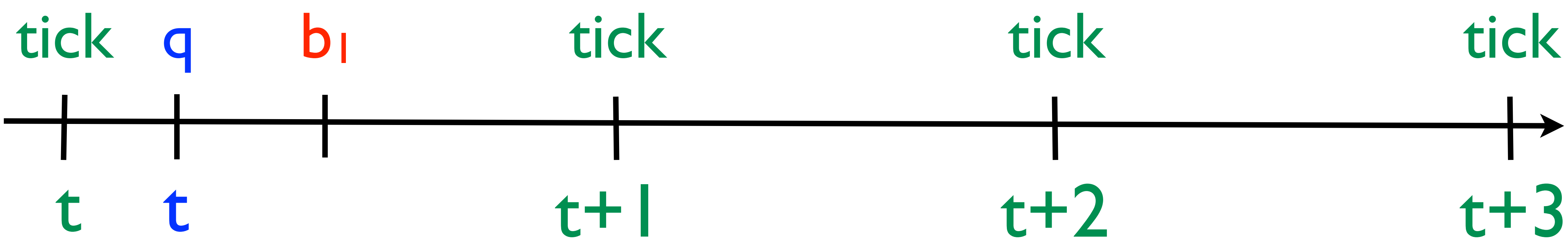
Encoding runs

q		
1	3	0



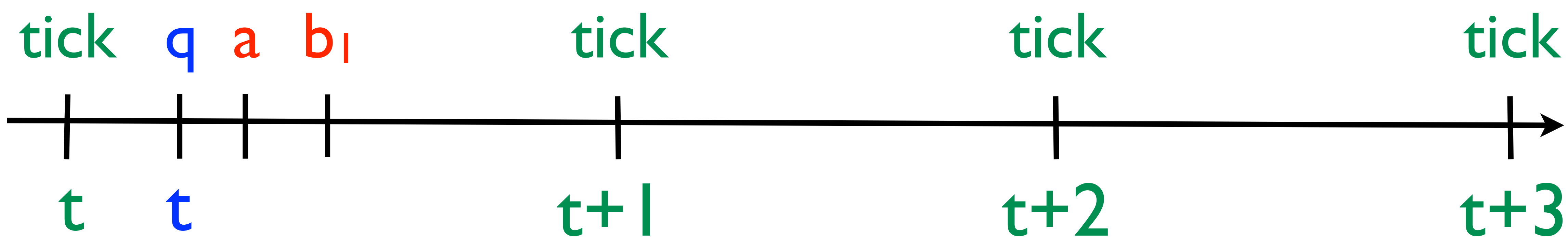
Encoding runs

q		
1	3	0



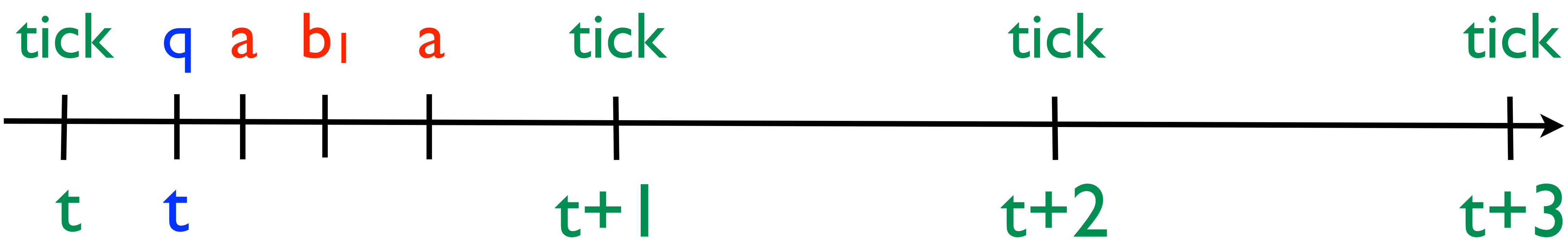
Encoding runs

q		
1	3	0



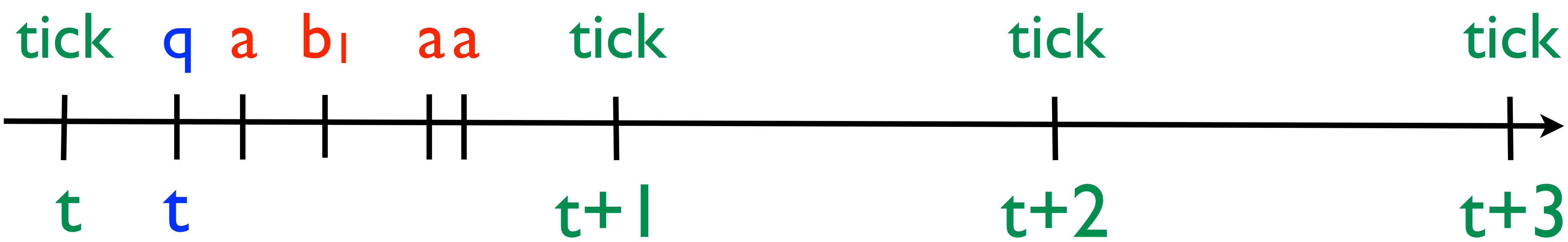
Encoding runs

q		
1	3	0



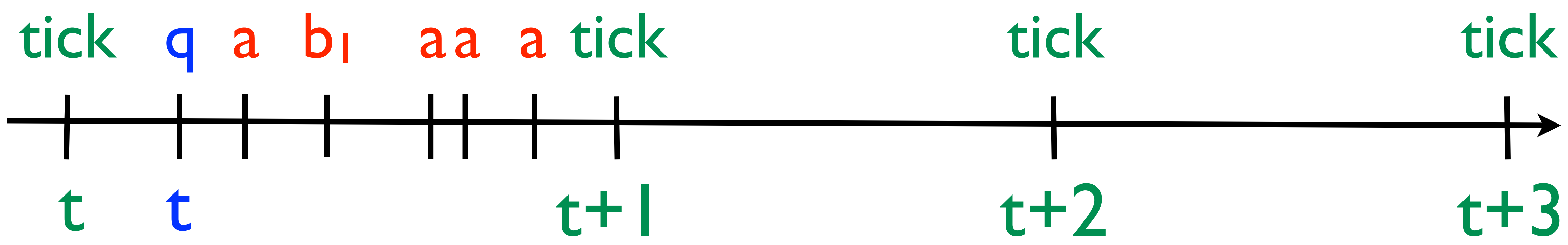
Encoding runs

q		
1	3	0



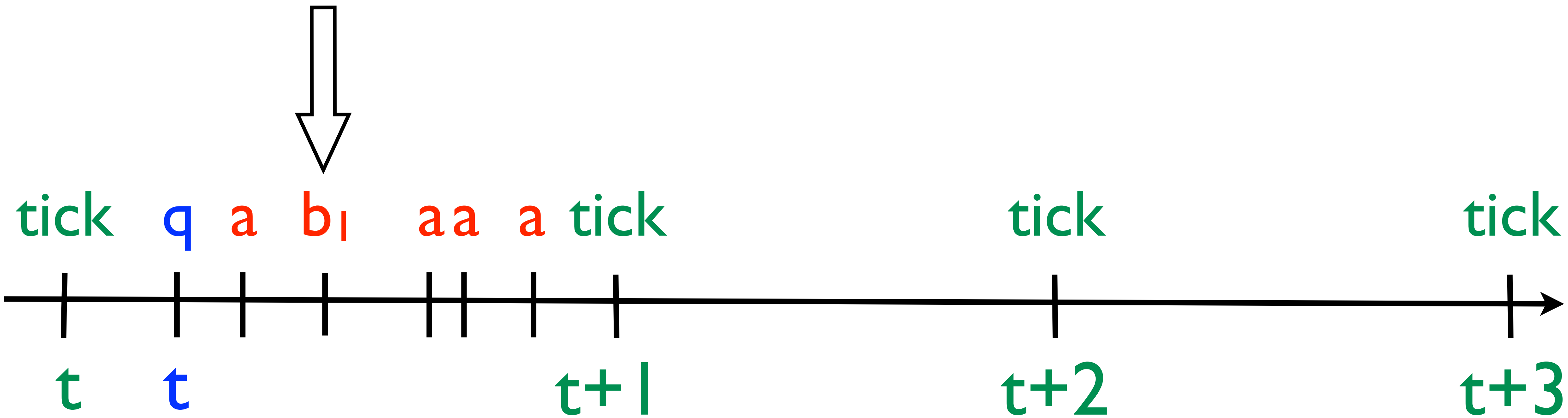
Encoding runs

q		
1	3	0



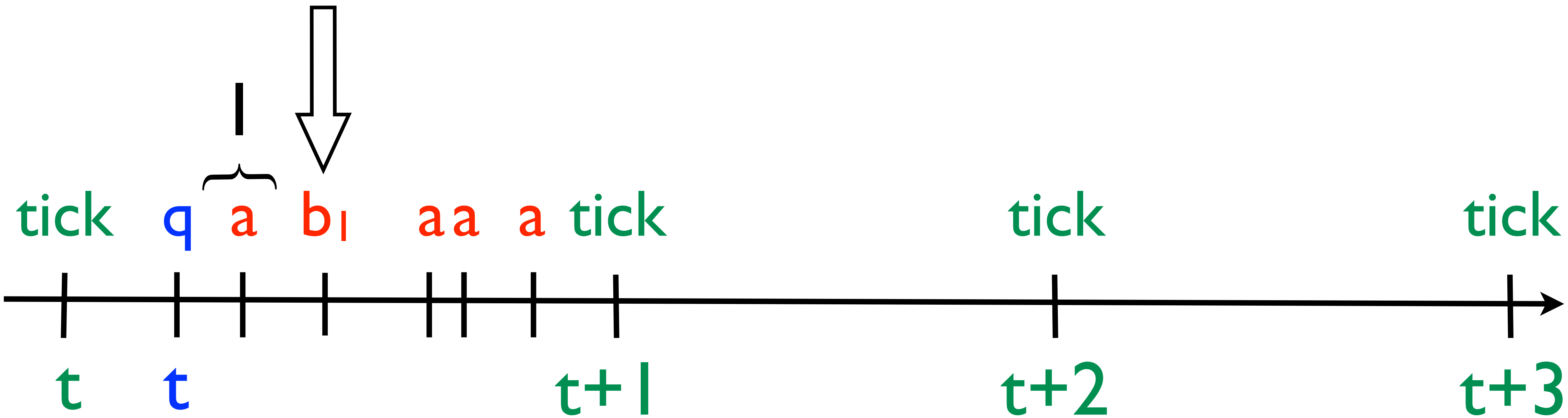
Encoding runs

q		
1	3	0

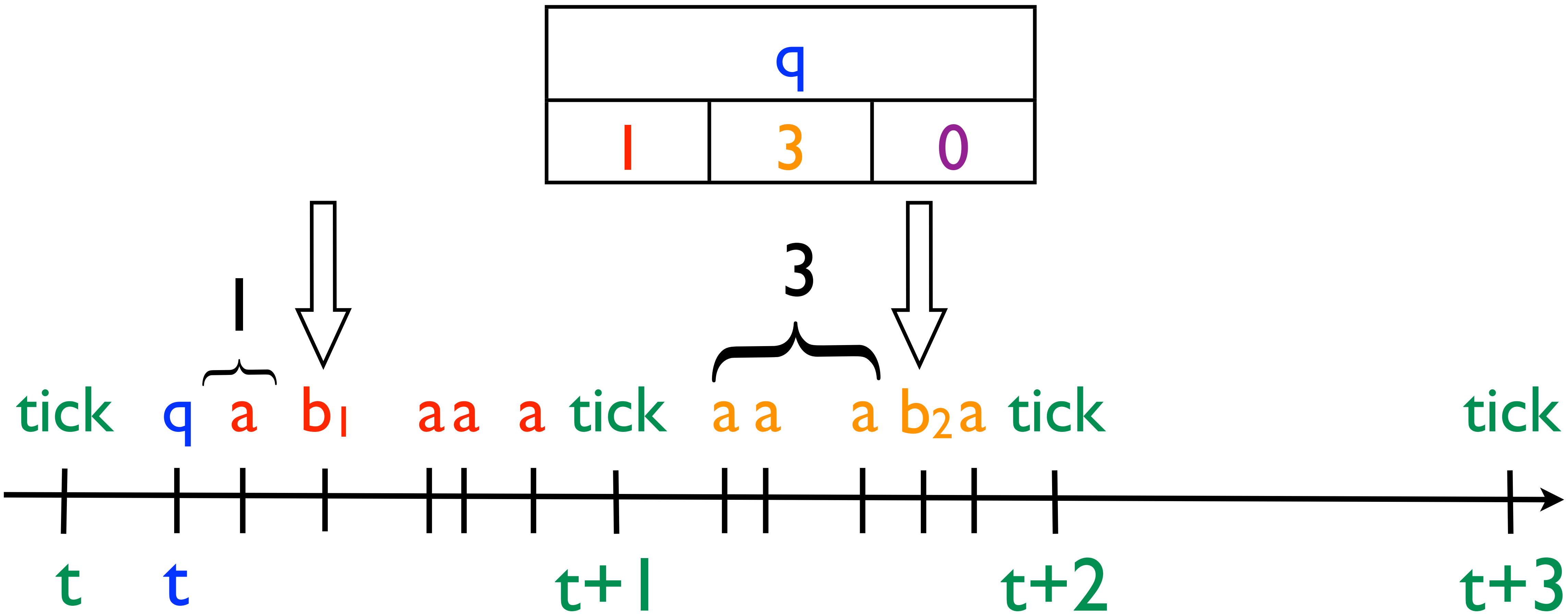


Encoding runs

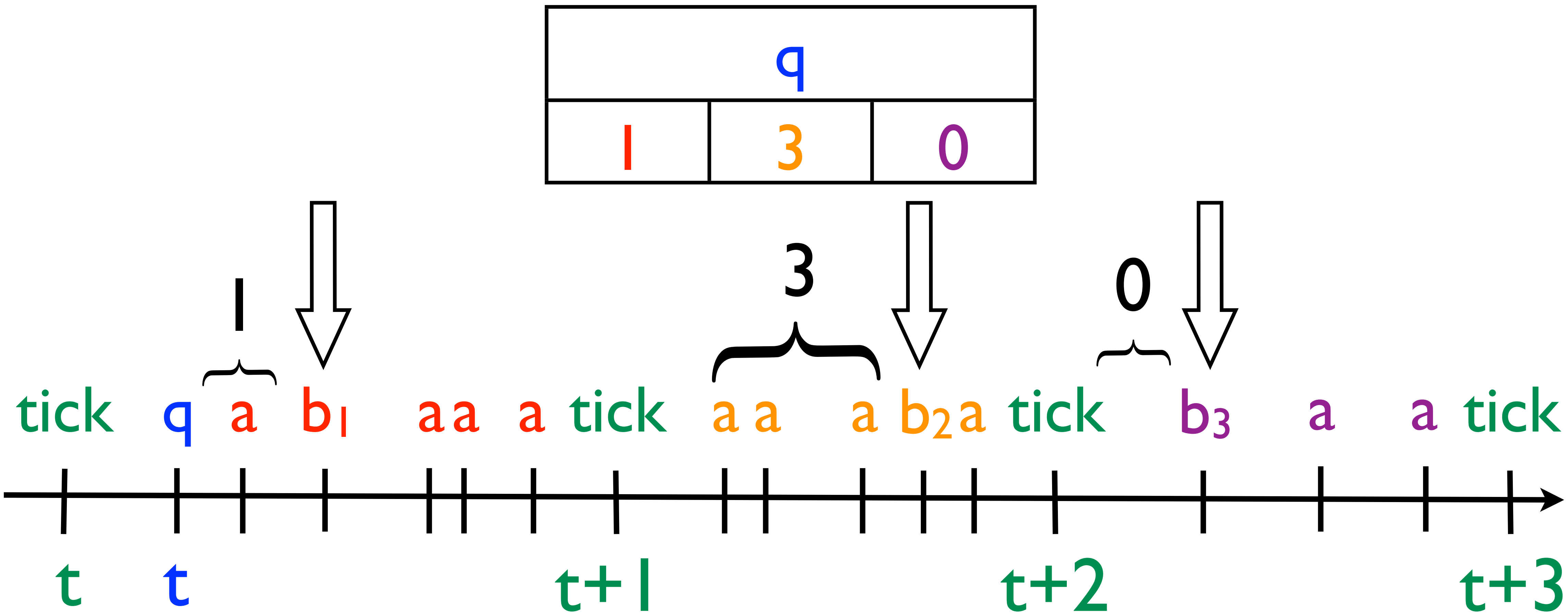
q		
1	3	0



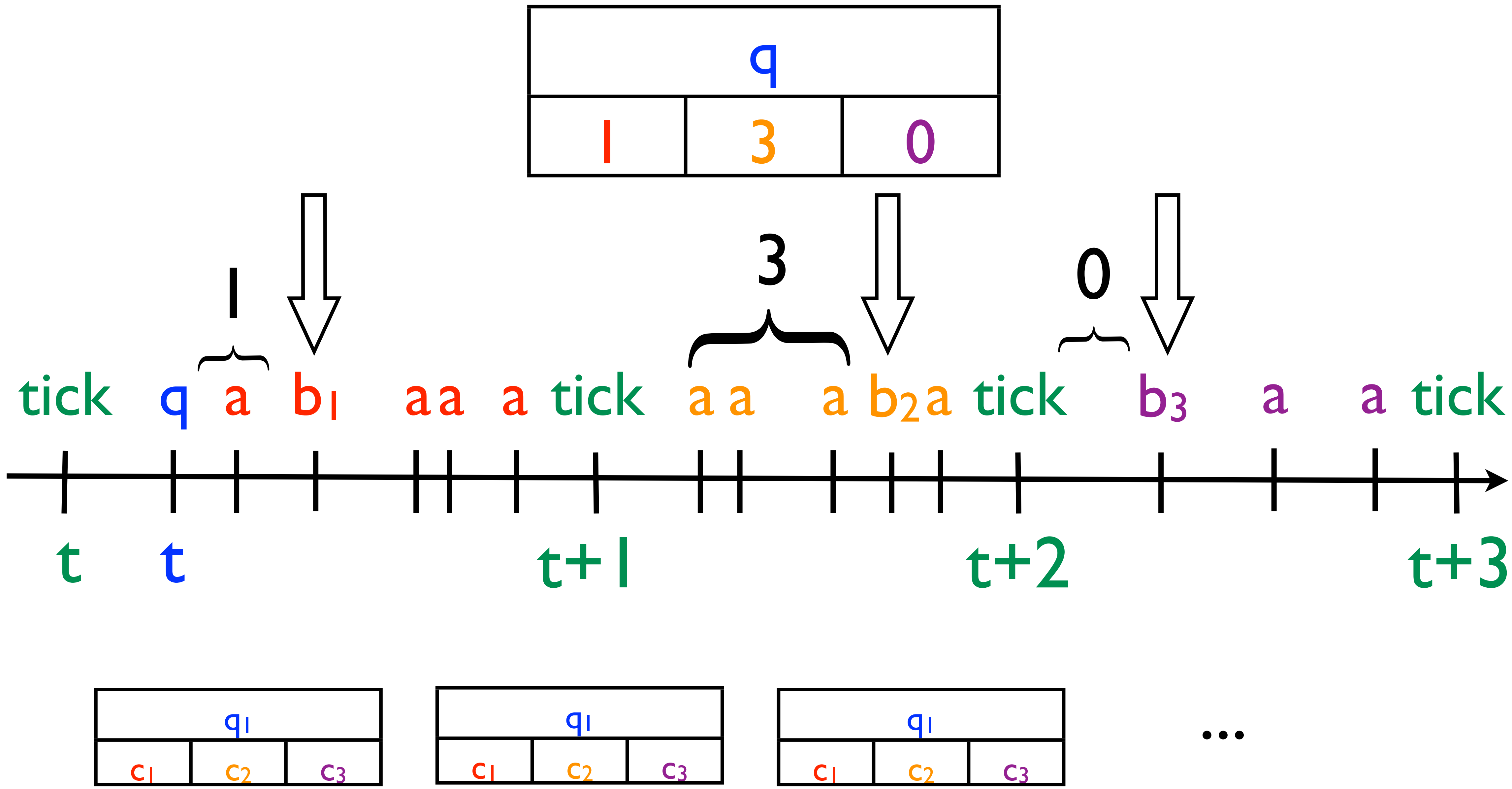
Encoding runs



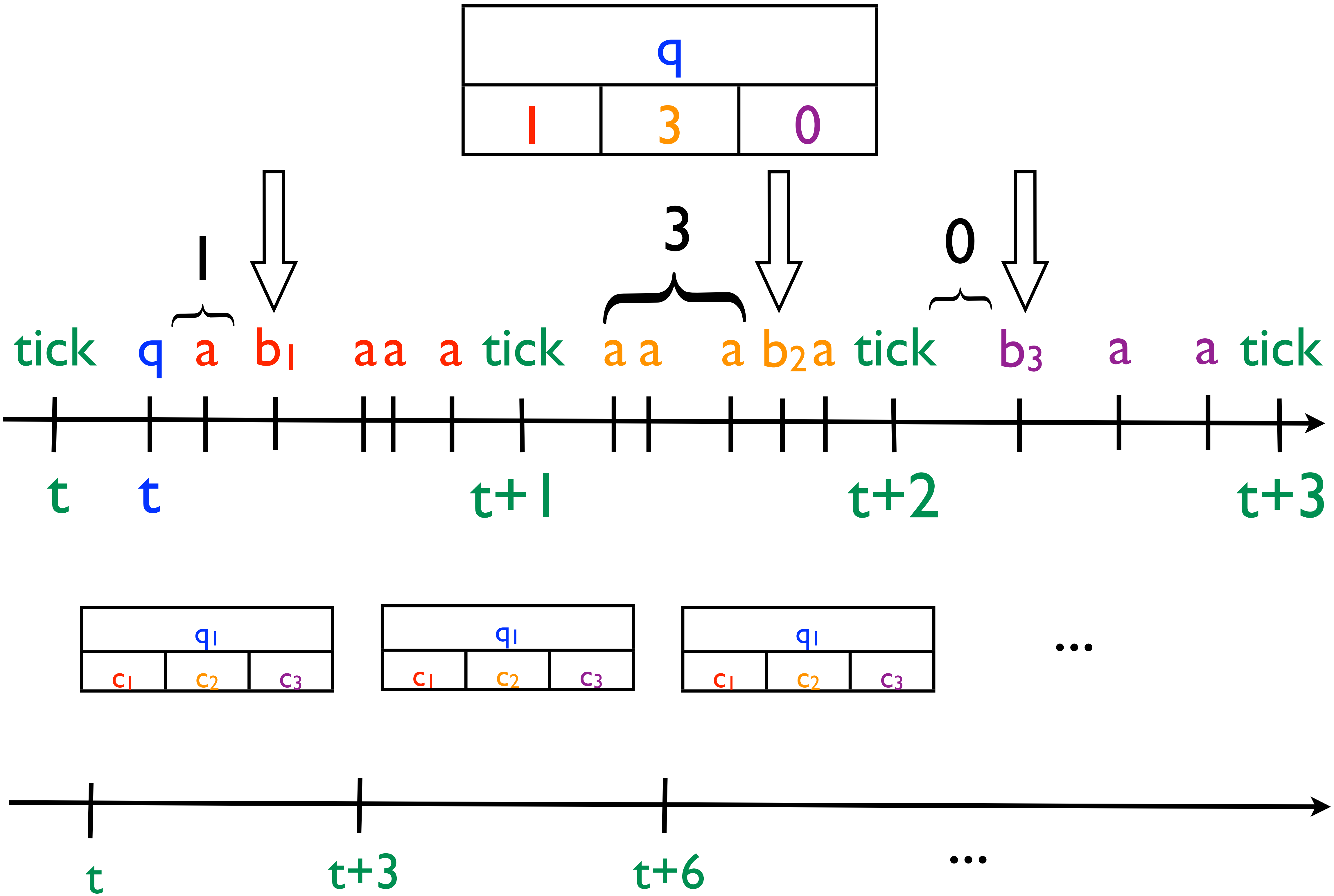
Encoding runs



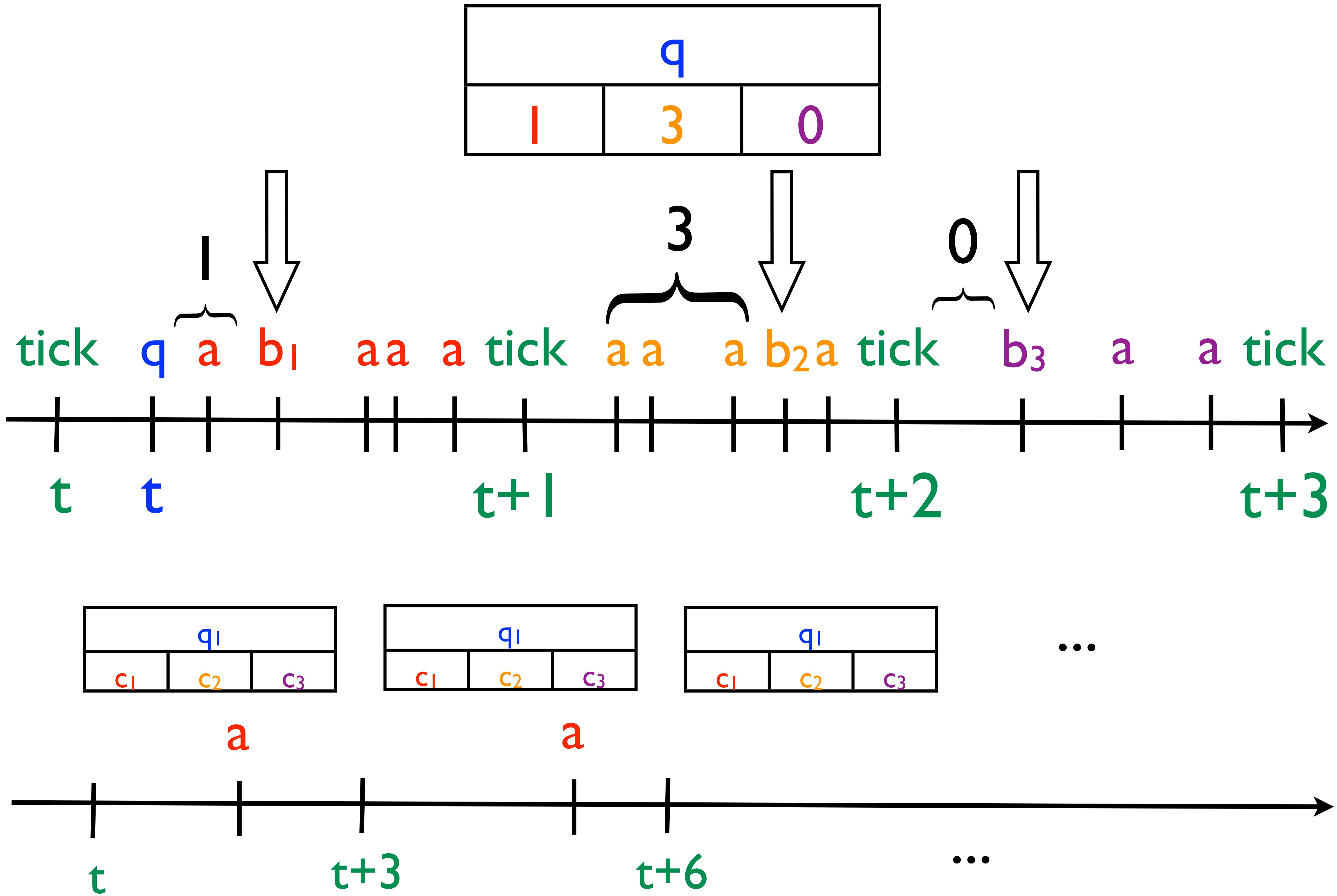
Encoding runs



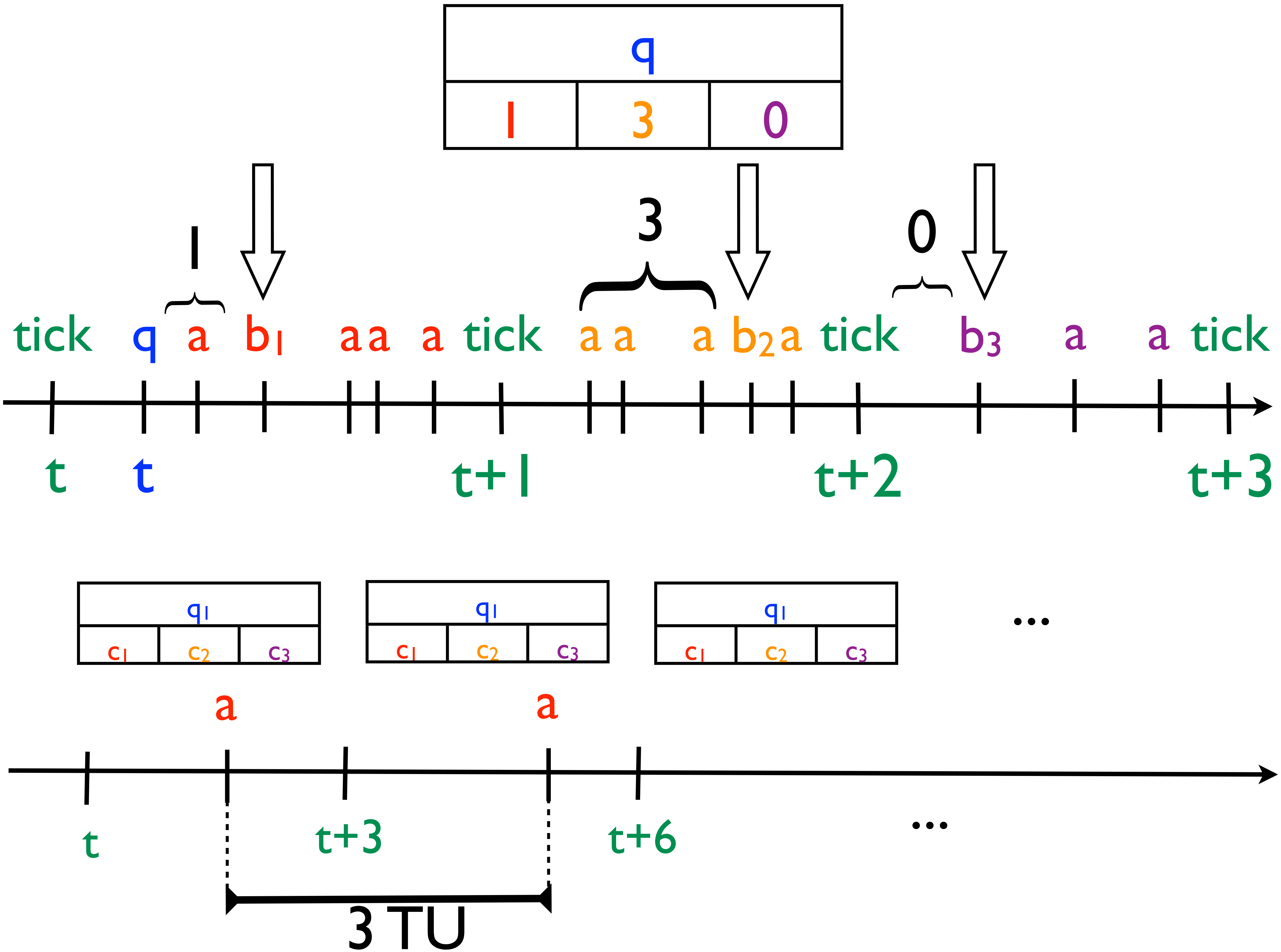
Encoding runs



Encoding runs



Encoding runs

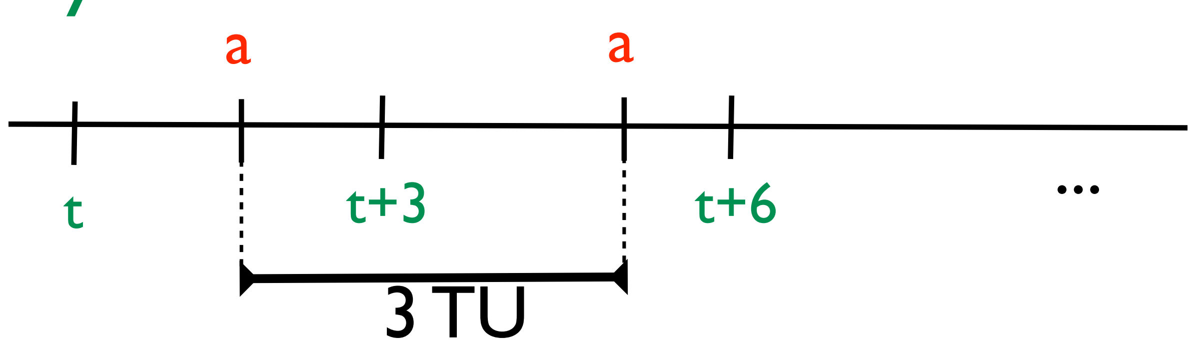


Reduction

- Given a 3CM M
  - Can we devise an ECL formula φ_M s.t.

> φM is satisfiable
> iff
> M admits an infinite bounded run ?

- **NO** !
  - Otherwise ECL satisfiability would be undecidable
  - We can't use ECL to specify that «every a or b should be preceded by an a or b 3 T.U. before» requirement

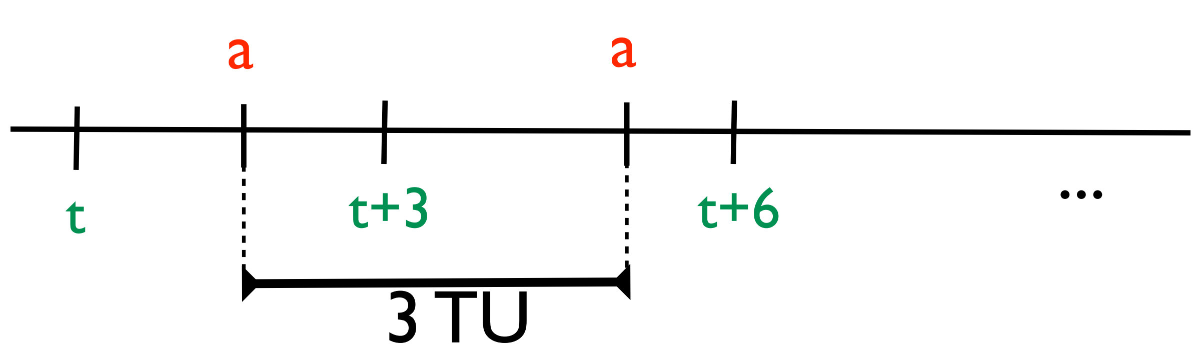


Reduction

- Given a 3CM M
  - Can we devise a timed game $\langle\Sigma_1,\Sigma_2,[\![\varphi_M]\!]\rangle$, where φ_M is an ECL formula s.t.

Player 1 has a winning strategy
iff
M admits an infinite bounded run ?

  - **YES** !
  - Player 1 controls the encoding symbols
  - We use Player 2 as an arbiter to check that Player 1 respects:

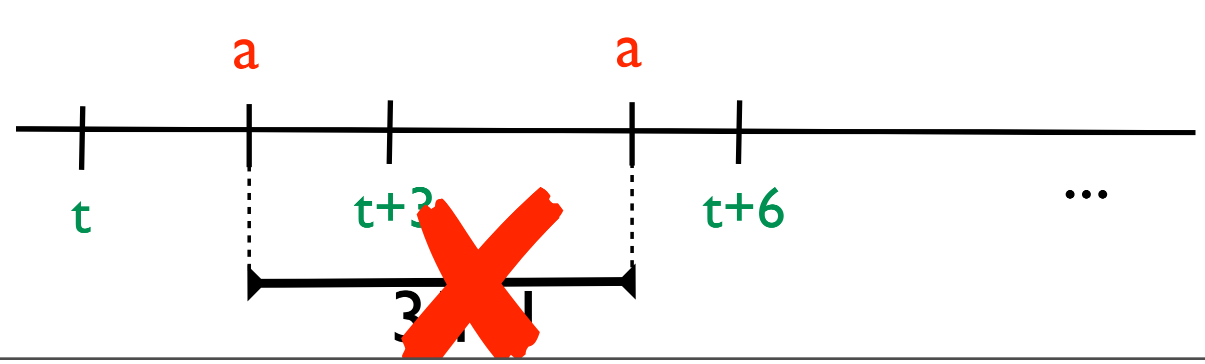


Reduction

- Given a 3CM M
  - Can we devise a timed game $\langle\Sigma_1,\Sigma_2,[\![\varphi_M]\!]\rangle$, where φ_M is an ECL formula s.t.

Player 1 has a winning strategy
iff
M admits an infinite bounded run ?

- **YES** !
- Player 1 controls the encoding symbols
- We use Player 2 as an arbiter to check that Player 1 respects:

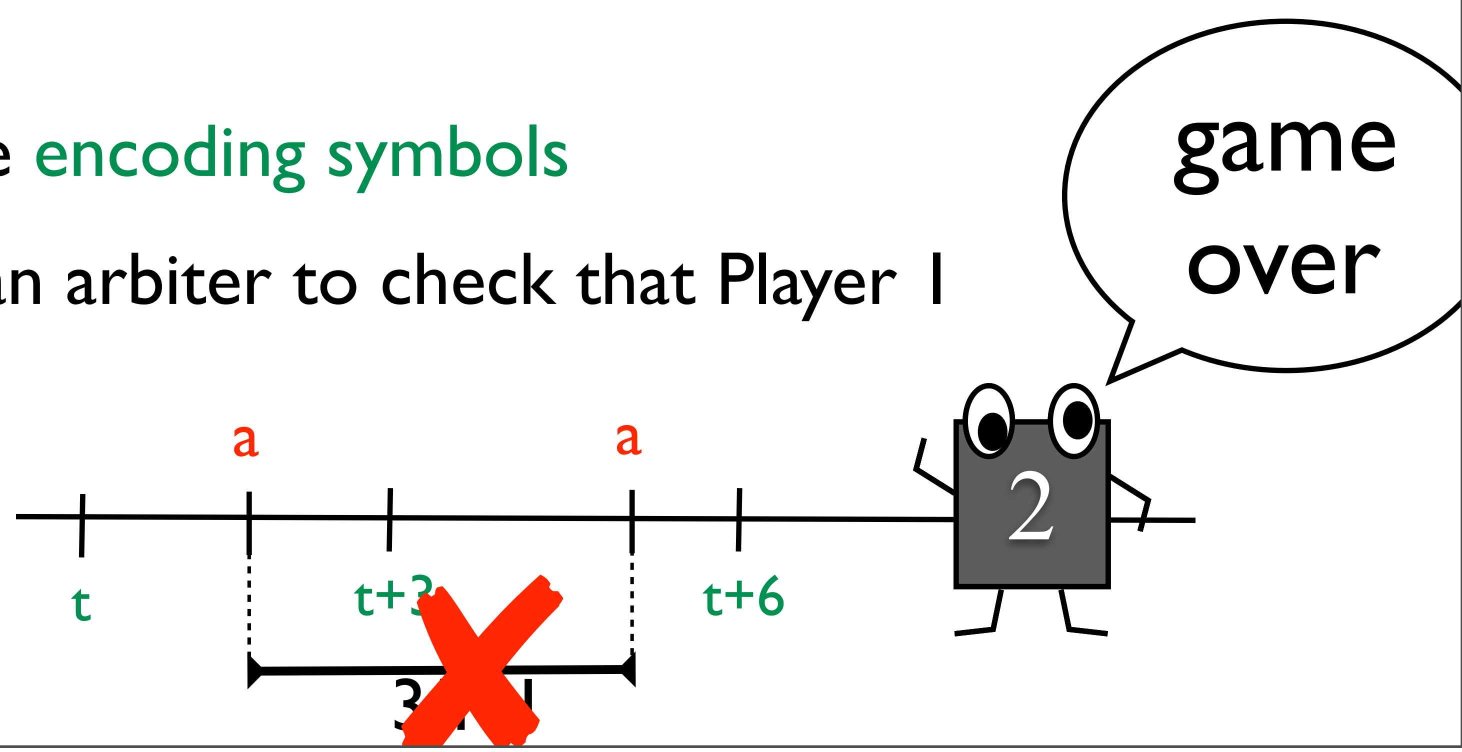


Reduction

- Given a 3CM M
  - Can we devise a timed game $\langle \Sigma_1, \Sigma_2, [\![\varphi_M]\!] \rangle$, where φ_M is an ECL formula s.t.

> Player 1 has a winning strategy
> iff
> M admits an infinite bounded run ?

  - **YES** !
  - Player 1 controls the encoding symbols
  - We use Player 2 as an arbiter to check that Player 1 respects:



Deterministic ?

$\Sigma=\Sigma_1\cup\Sigma_2$

Spec.
Φ
$LTL_\triangleleft$

$?(\Sigma_1) \,||\, Env(\Sigma_2) \models \Phi$

$\exists\lambda_1 \bullet \forall\lambda_2 \bullet$ $\exists$run r of A_Φ $\bullet$ r accepts Outcome(λ_1,λ_2)

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

$\exists\lambda_1 \bullet \forall\lambda_2 \bullet$ unique r of A^d on Outcome(λ_1,λ_2) is accepting

Universal co-Büchi ERA

- Instead of considering classical Büchi condition, we will consider Universal co-Büchi condition
  - Büchi = ∃ a run on w that visits accepting states infinitely often
  - co-Büchi = all runs on w visit accepting states finitely often
- These conditions are dual !

Universal co-Büchi ERA

- Instead of considering classical Büchi condition, we will consider Universal co-Büchi condition
  - Büchi = ∃ a run on w that visits accepting states infinitely often
  - co-Büchi = all runs on w visit accepting states finitely often
- These conditions are dual !

φ

Universal co-Büchi ERA

- Instead of considering classical Büchi condition, we will consider Universal co-Büchi condition
  - Büchi = $\exists$ a run on w that visits accepting states infinitely often
  - co-Büchi = all runs on w visit accepting states finitely often
- These conditions are dual !

$$\varphi \longrightarrow$$

Universal co-Büchi ERA

- Instead of considering classical Büchi condition, we will consider Universal co-Büchi condition
  - Büchi = ∃ a run on w that visits accepting states infinitely often
  - co-Büchi = all runs on w visit accepting states finitely often
- These conditions are dual !

$$\varphi \longrightarrow \neg\varphi$$

Universal co-Büchi ERA

- Instead of considering classical Büchi condition, we will consider Universal co-Büchi condition
  - Büchi = ∃ a run on w that visits accepting states infinitely often
  - co-Büchi = all runs on w visit accepting states finitely often
- These conditions are dual !

$$\varphi \longrightarrow \neg\varphi \longrightarrow$$

Universal co-Büchi ERA

- Instead of considering classical Büchi condition, we will consider Universal co-Büchi condition
  - Büchi = ∃ a run on w that visits accepting states infinitely often
  - co-Büchi = all runs on w visit accepting states finitely often
- These conditions are dual !

$\varphi \longrightarrow \neg\varphi \longrightarrow A_{\neg\varphi}$

Universal co-Büchi ERA

- Instead of considering classical Büchi condition, we will consider Universal co-Büchi condition
  - Büchi = ∃ a run on w that visits accepting states infinitely often
  - co-Büchi = all runs on w visit accepting states finitely often
- These conditions are dual !

$\varphi \longrightarrow \neg\varphi \longrightarrow A_{\neg\varphi}$

«Büchi construction»

Universal co-Büchi ERA

- Instead of considering classical Büchi condition, we will consider Universal co-Büchi condition
  - Büchi = ∃ a run on w that visits accepting states infinitely often
  - co-Büchi = all runs on w visit accepting states finitely often
- These conditions are dual !

$$\varphi \longrightarrow \neg\varphi \longrightarrow A_{\neg\varphi}$$

«Büchi construction»

Then: $L_{UcoB}(A_{\neg\varphi}) = [\![\varphi]\!]$