Third Lecture: Basics of Timed Controller Synthesis

Jean-François Raskin
Université Libre de Bruxelles
Belgium

Artist2 Asian Summer School - Shanghai - July 2008

Goals of the talk

- **Introduction** to basic game technics to solve the controller synthesis problem
- **Timed games** and symbolic technics (sketches)
- Show that the **implementability** of controller **models** is an **important issue**

Goals of the talk

- **Introduction** to basic game technics to solve the controller synthesis problem
- **Timed games** and symbolic technics (sketches)
- Show that the **implementability** of controller **models** is an **important issue**

Give relevant pointers to literature

Context

- Make a model of the environment

  Environment

- Make clear the control objective:

  Bad

- Make a model of your control strategy:

  ControllerMod

- Verify :

  Does Environment || ControllerMod avoid Bad ?

Context

- Make a model of the environment

  Environment

- Make clear the control objective:

  Make the synthesis **d**

- ~~Make a model~~ of your control strategy:

  ControllerMod

- ~~Verify~~ :

  ~~Does Environment || ControllerMod avoid Bad ?~~

- Good, but after ?

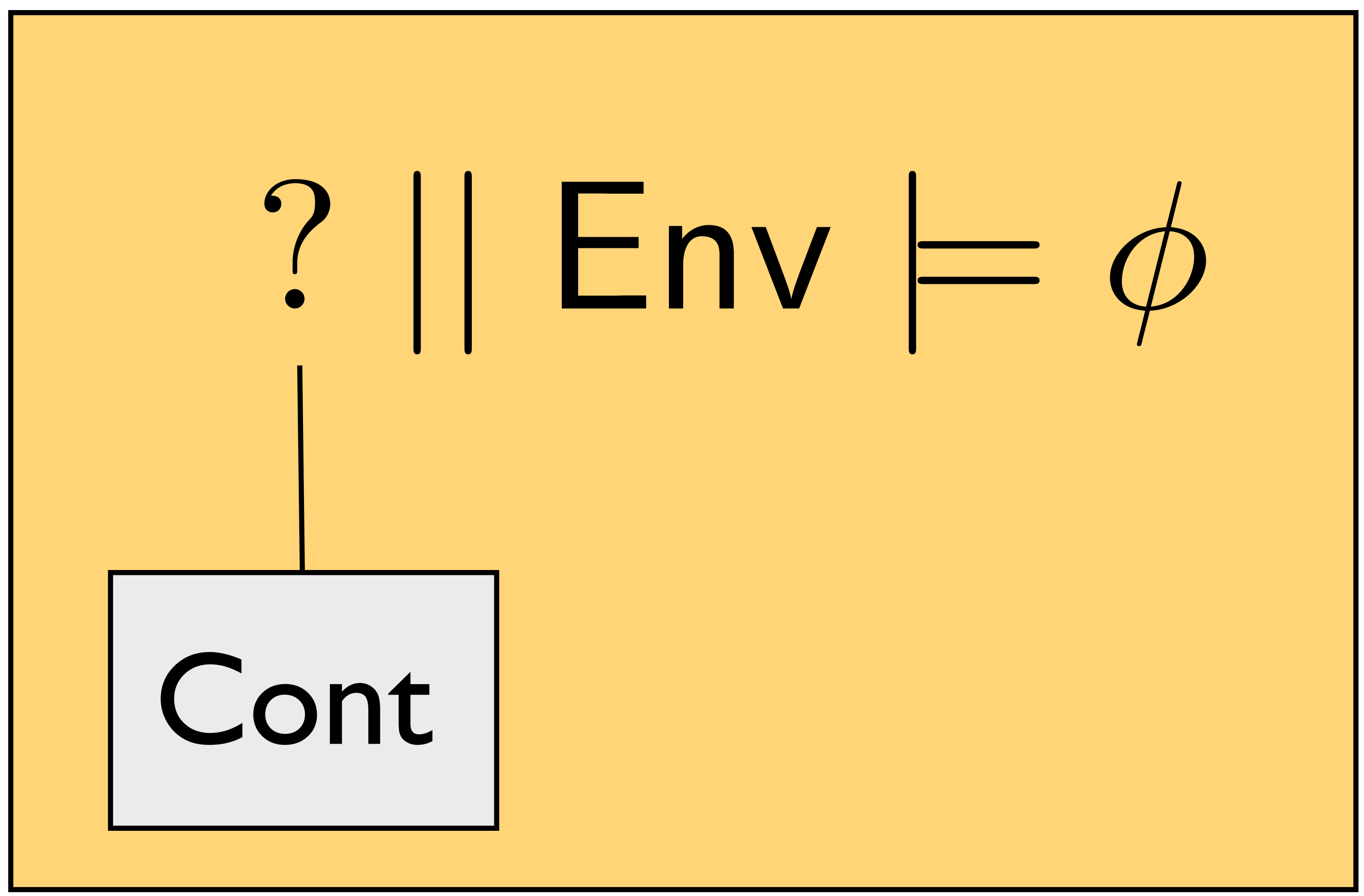
  Is my controller implementable ?

The synthesis problem

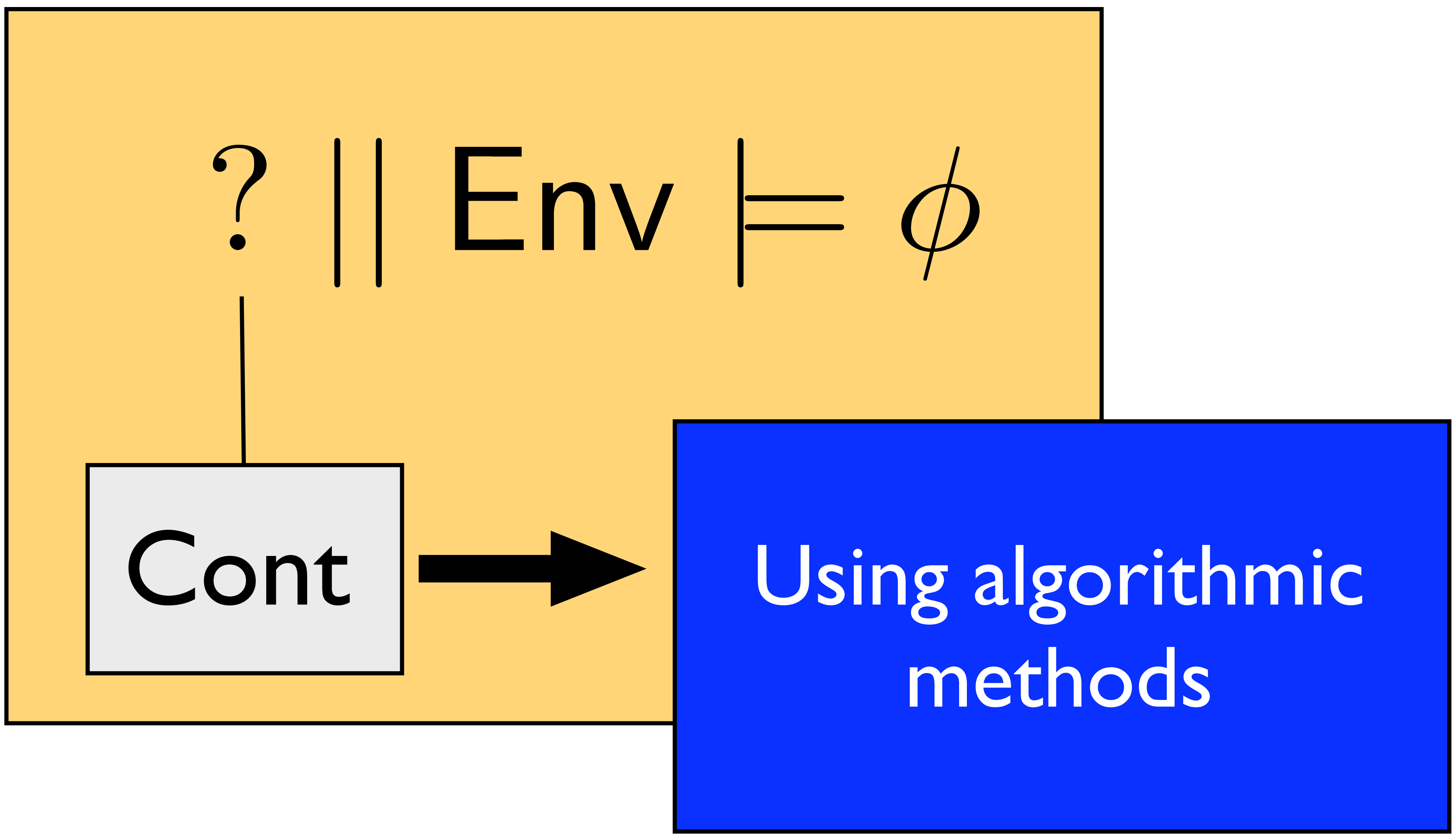
The synthesis problem

$$? \,||\, \mathsf{Env} \models \phi$$

The synthesis problem



The synthesis problem



The synthesis problem

Specialize process A into C such that

$$A \geq C \text{ and } C \mid\mid B \models \phi$$

So, C must refine A and control B to **enforce** ϕ

Basic technics: finite state case

Are transition systems adequate for synthesis ?

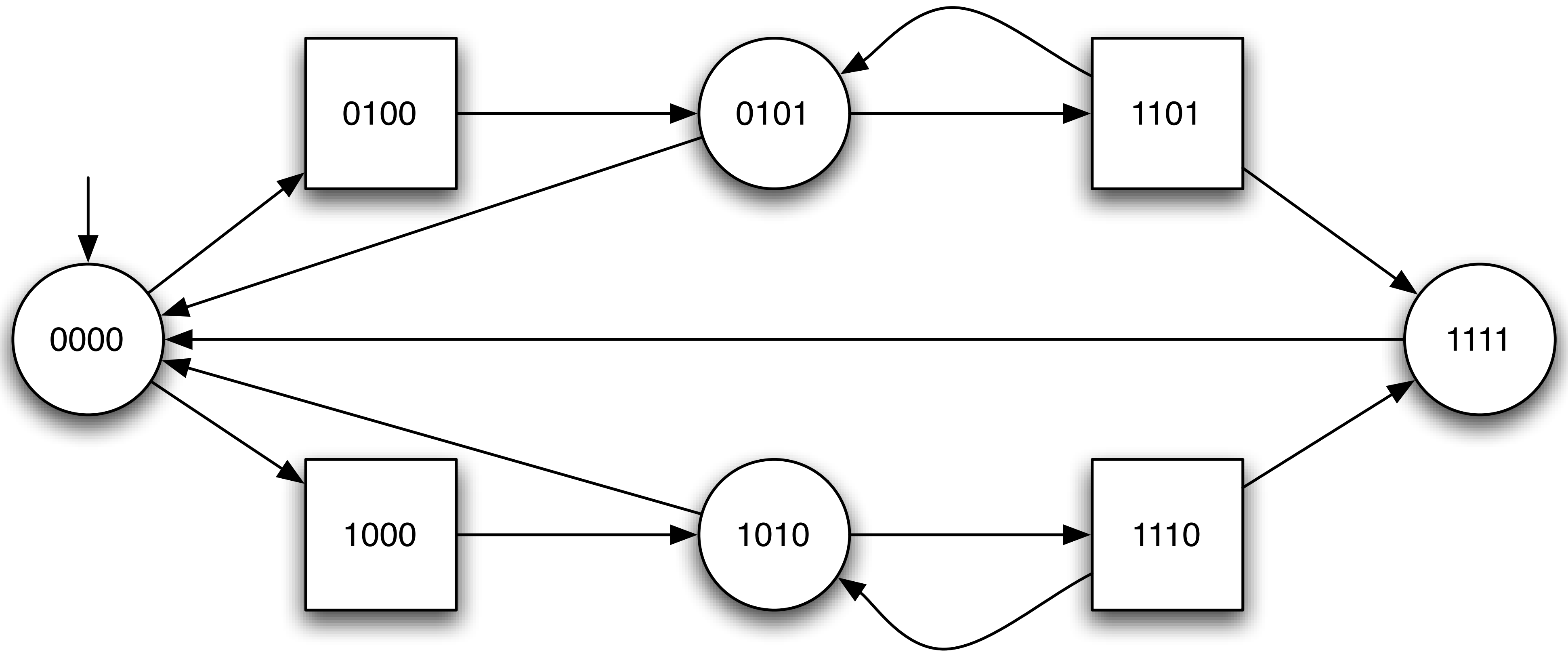
- For the verification problem, the semantics of processes is usually given by **transition systems**
- When we consider the transition system for A || B, we loose the information about the **components**

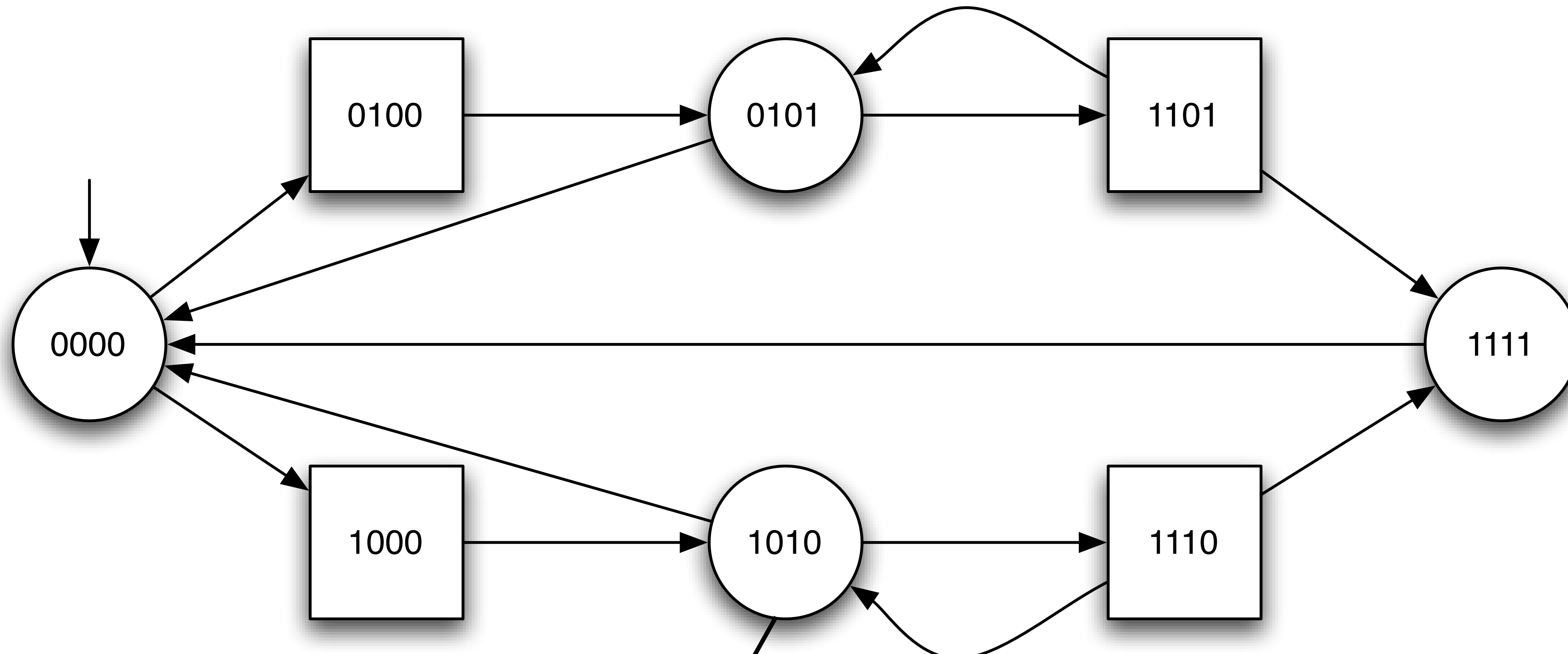
Are transition systems adequate for synthesis ?

- For the verification problem, the semantics of processes is usually given by **transition systems**
- When we consider the transition system for A || B, we loose the information about the **components**

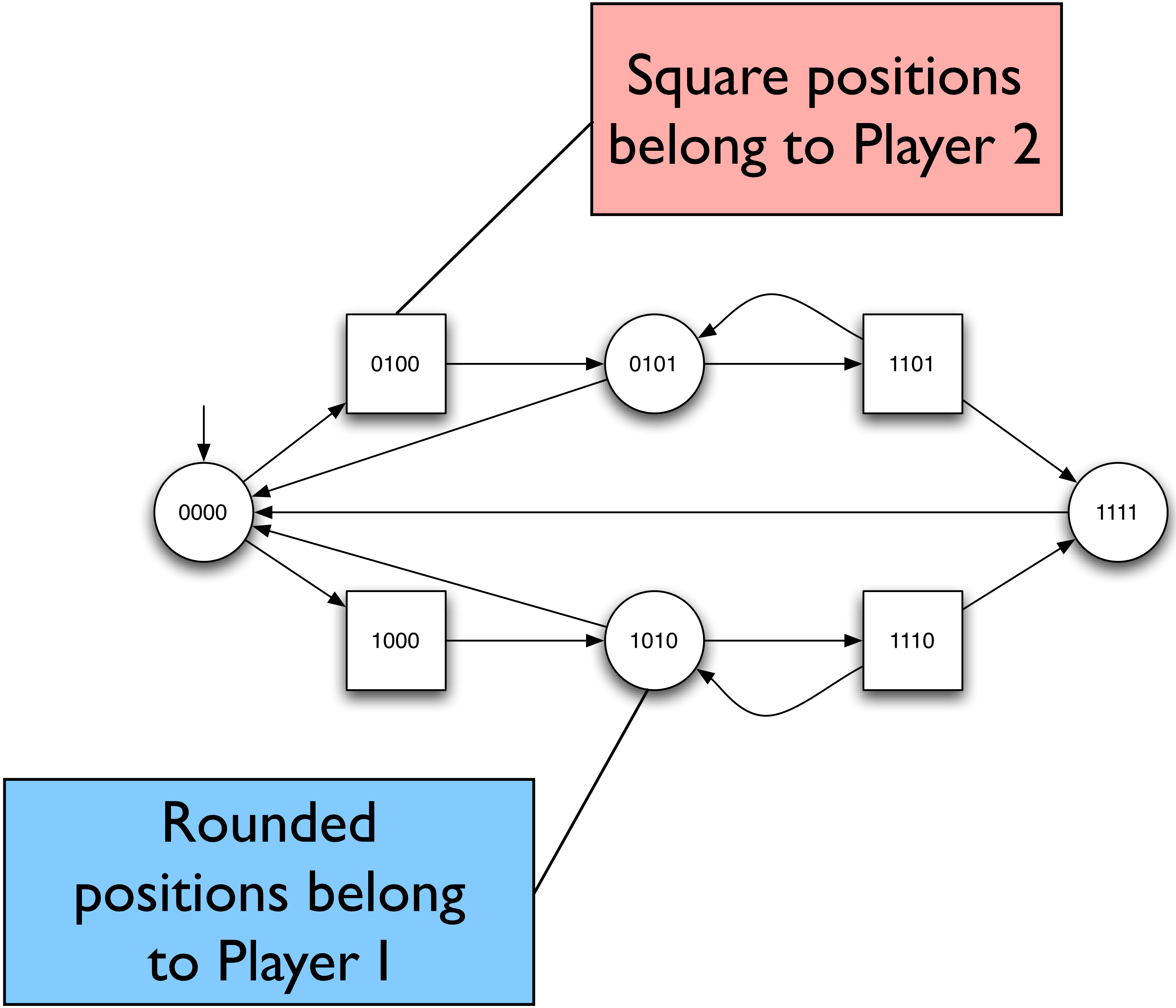
So, we need richer models where **identities** of processes are explicit:
two-player game structures

Two-player game structures

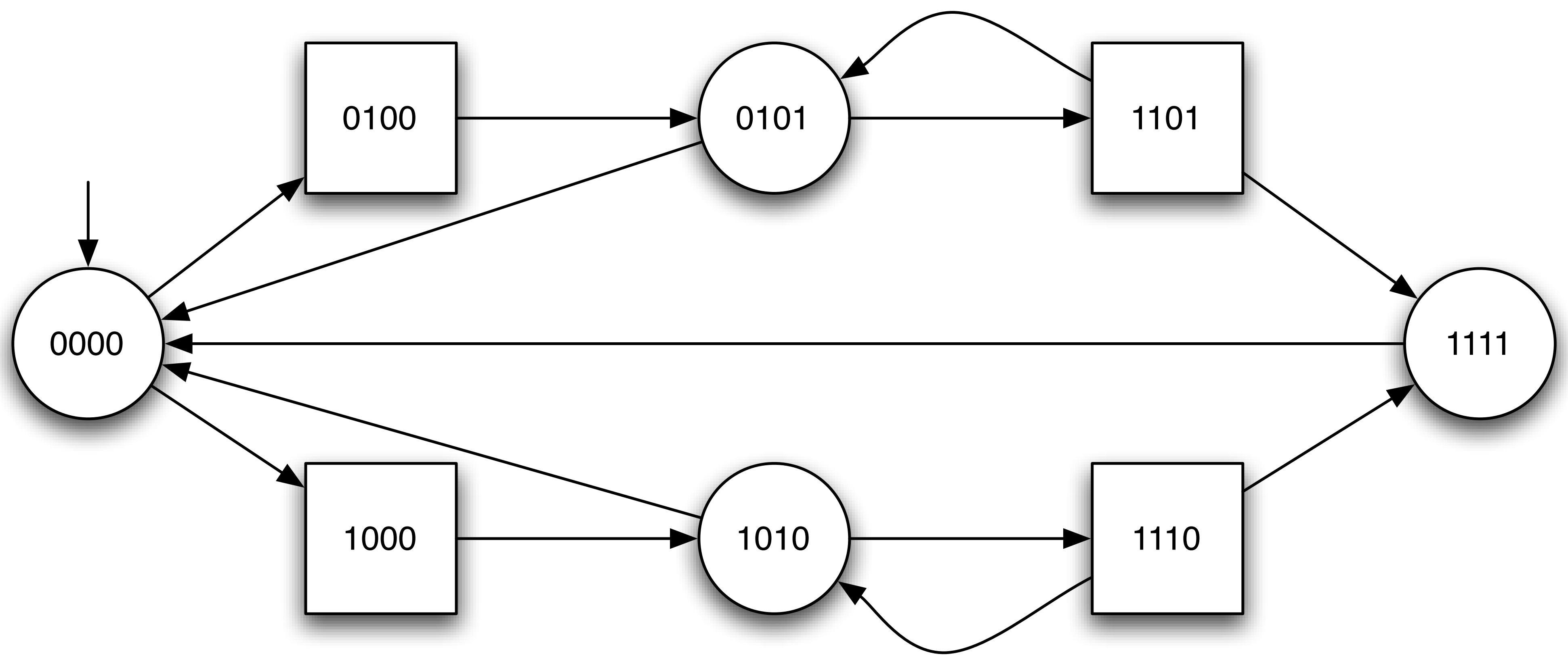

0100
0101
1101
0000
1111
1000
1010
1110



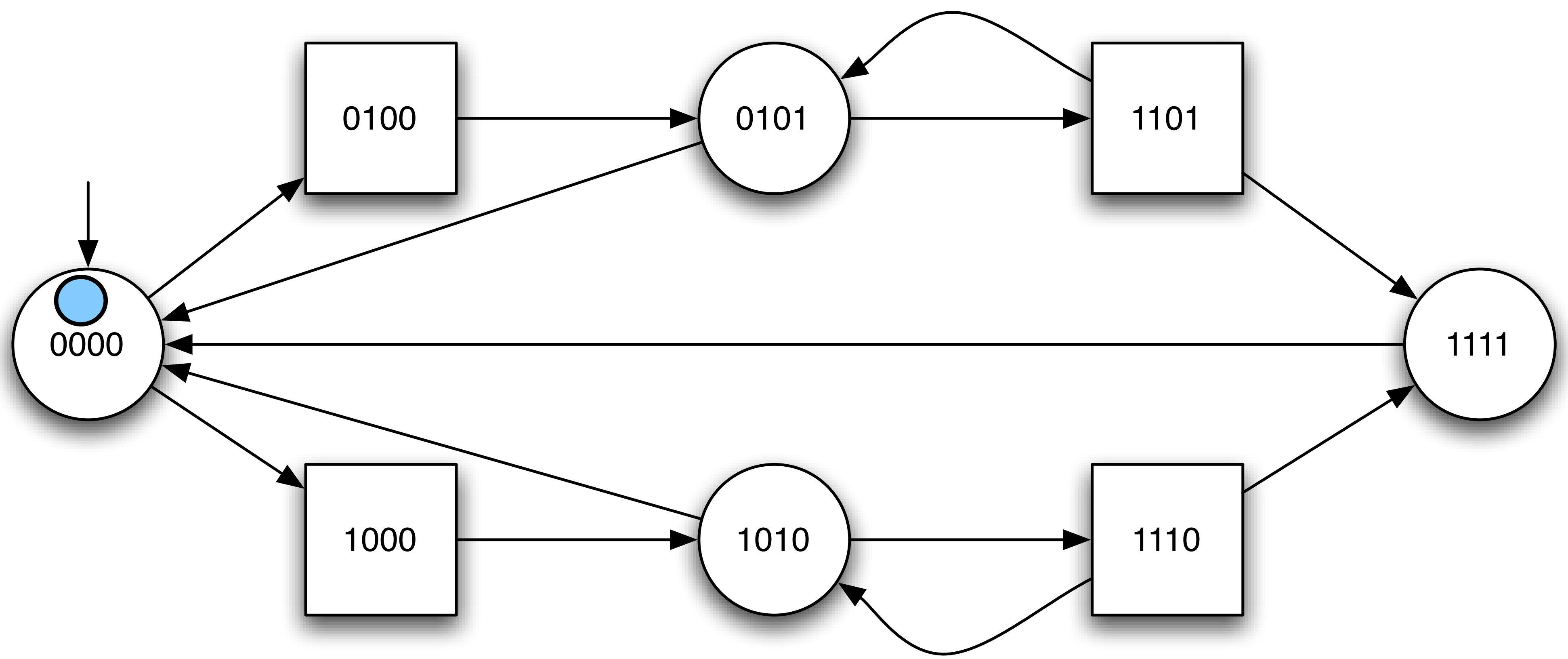
Rounded positions belong to Player I


Square positions belong to Player 2
0100
0101
1101
0000
1111
1000
1010
1110
Rounded positions belong to Player 1

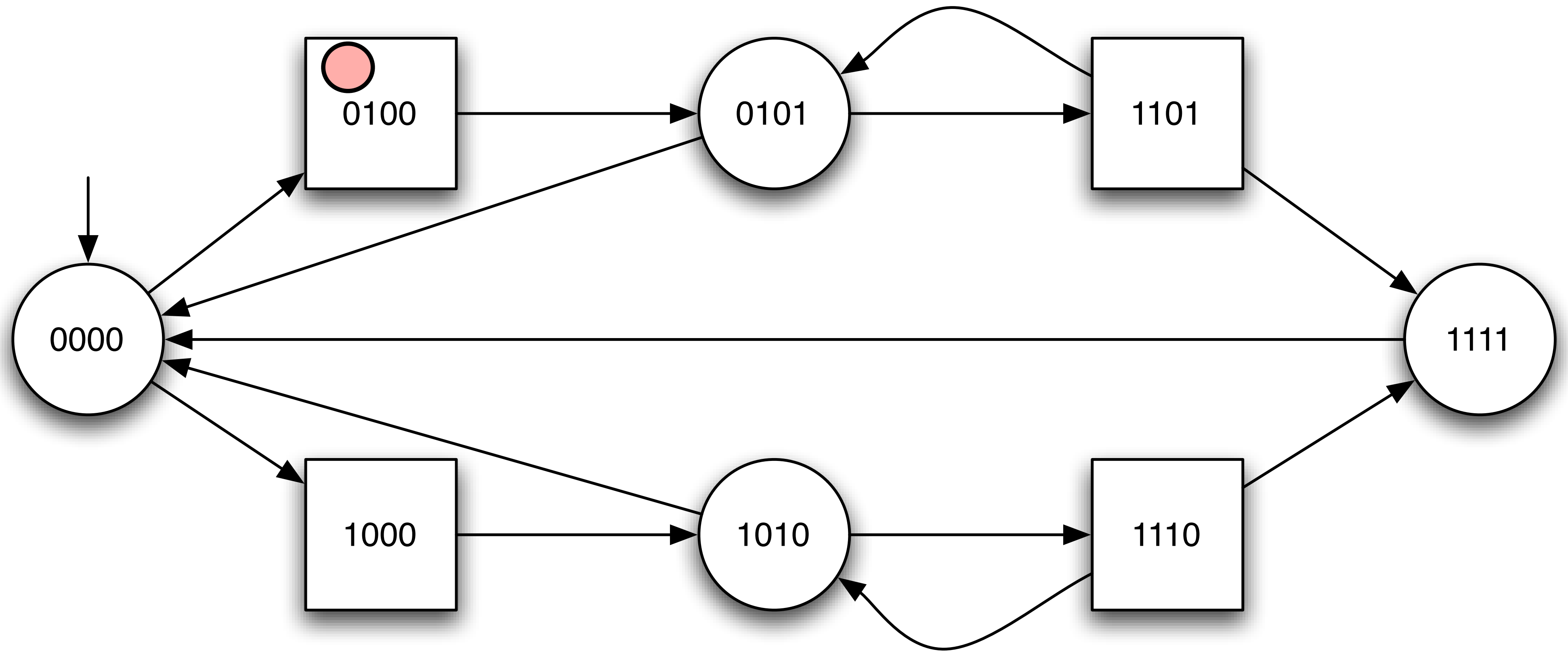
Rounded positions belong to Player 1
Square positions belong to Player 2



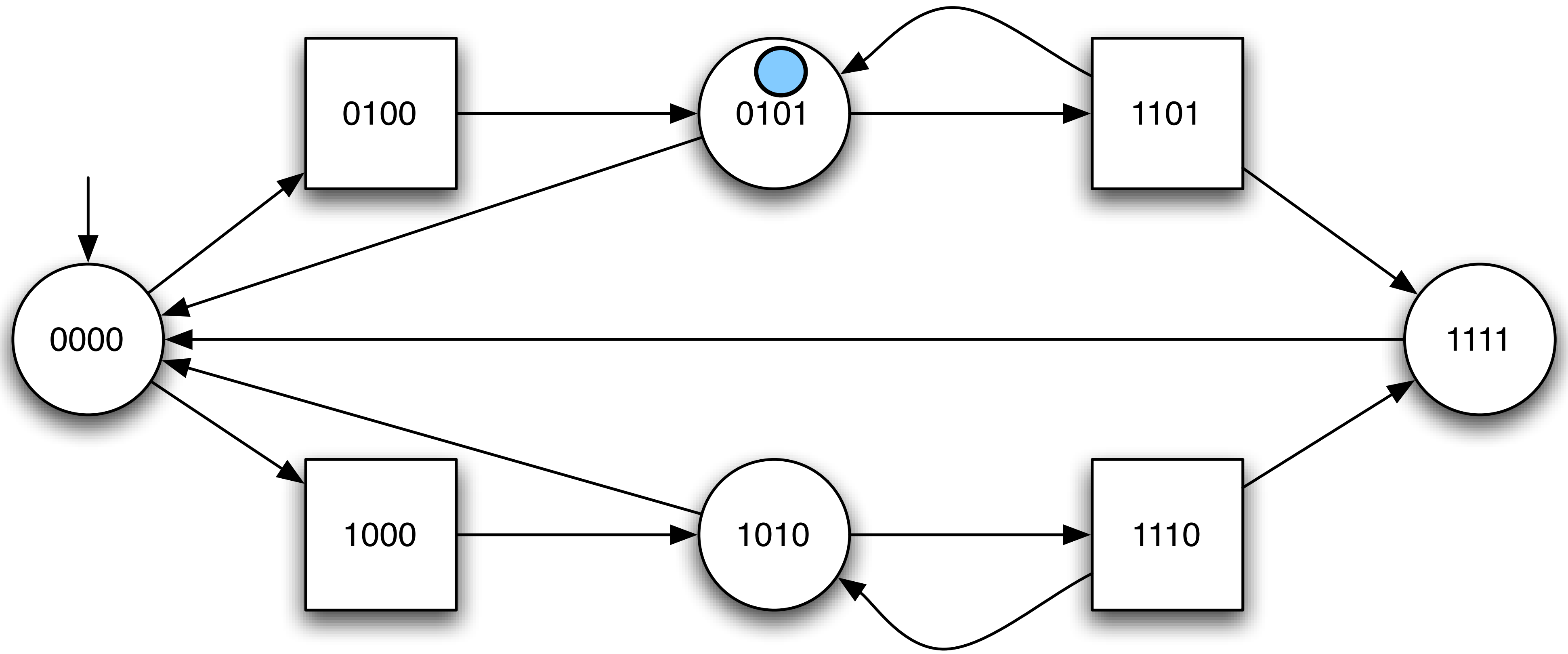
A game is played as follows: in each **round**, the game is in a **position**, if the game is in a rounded position, Player 1 resolves the **choice** for the next state, if the game is in a square position, Play 2 resolves the choice. The game is played for an **infinite number of rounds**.



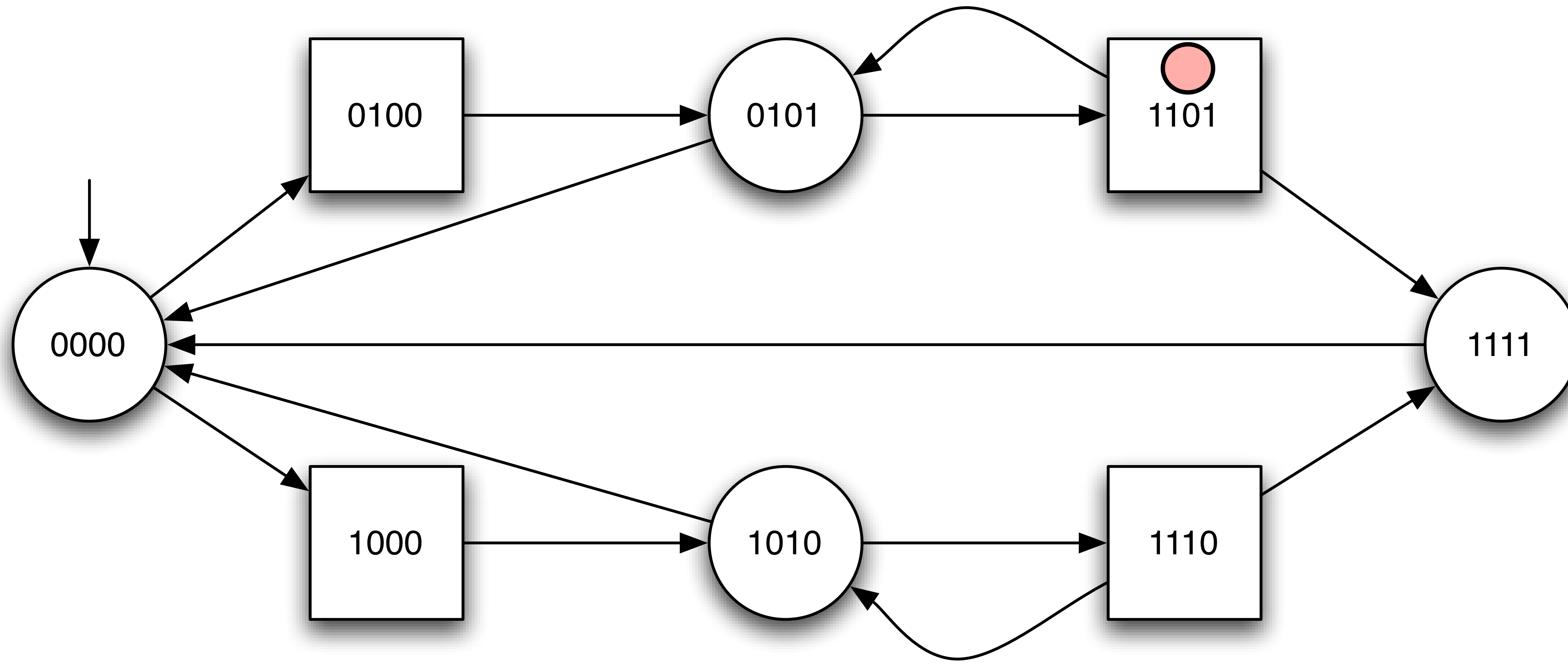
Play : 0000



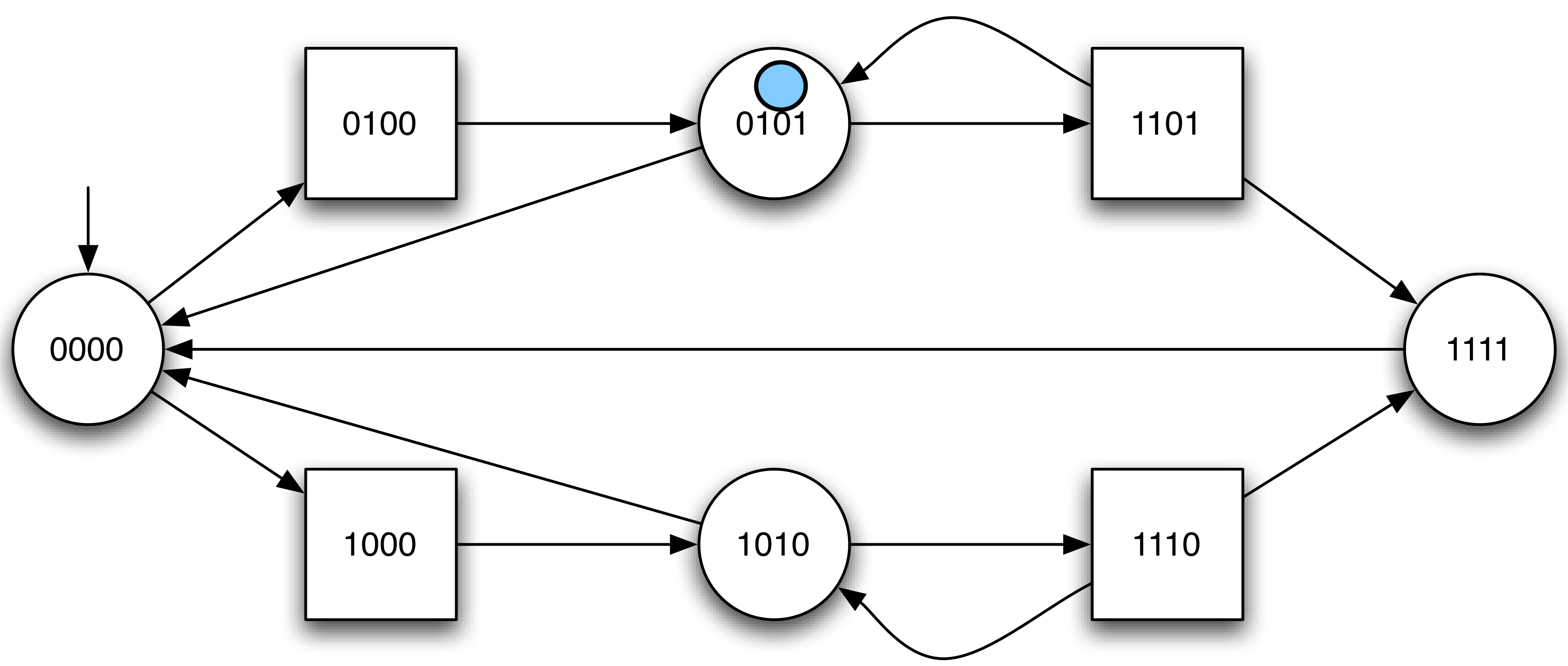
Play : 0000 0100



Play : 0000 0100 0101



Play : 0000 0100 0101 1101



Play : 0000 0100 0101 1101 ...

Two-player Game Structure

A **two-player game structure** is a tuple $G = \langle Q_1, Q_2, \iota, \delta \rangle$ where:

Q_1 and Q_2 are two (finite and) disjoint sets of **positions**

$\iota \in Q_1 \cup Q_2$ is the **initial** position of the game

$\delta \subseteq (Q_1 \cup Q_2) \times (Q_1 \cup Q_2)$ is the **transition relation** of the game

We assume that $\forall q \in Q_1 \cup Q_2 : \exists q' \in Q_1 \cup Q_2 : \delta(q, q')$

Plays, Prefixes of Plays

Let $G = \langle Q_1, Q_2, \iota, \delta \rangle$,

$w = q_0 q_1 \ldots q_n \ldots$ is a **play** in G if

Plays, Prefixes of Plays

Let $G = \langle Q_1, Q_2, \iota, \delta \rangle$,

$w = q_0 q_1 \ldots q_n \ldots$ is a **play** in G if

$$\forall i \geq 0 : q_i \in Q_1 \cup Q_2$$

Plays, Prefixes of Plays

Let $G = \langle Q_1, Q_2, \iota, \delta \rangle$,

$w = q_0 q_1 \dots q_n \dots$ is a **play** in G if

Notations

Let $w = q_0 q_1 \dots q_n \dots$:

$w(i)$ denotes position *i*

$w(0, i)$ denotes the prefix up to position *i*

$last(w(0, i)) = w(i)$

Plays, Prefixes of Plays

Let $G = \langle Q_1, Q_2, \iota, \delta \rangle$,

$w = q_0 q_1 \dots q_n \dots$ is a **play** in G if

1) $w(0) = \iota$
2) $\forall i \geq 0 : \delta(w(i), w(i+1))$

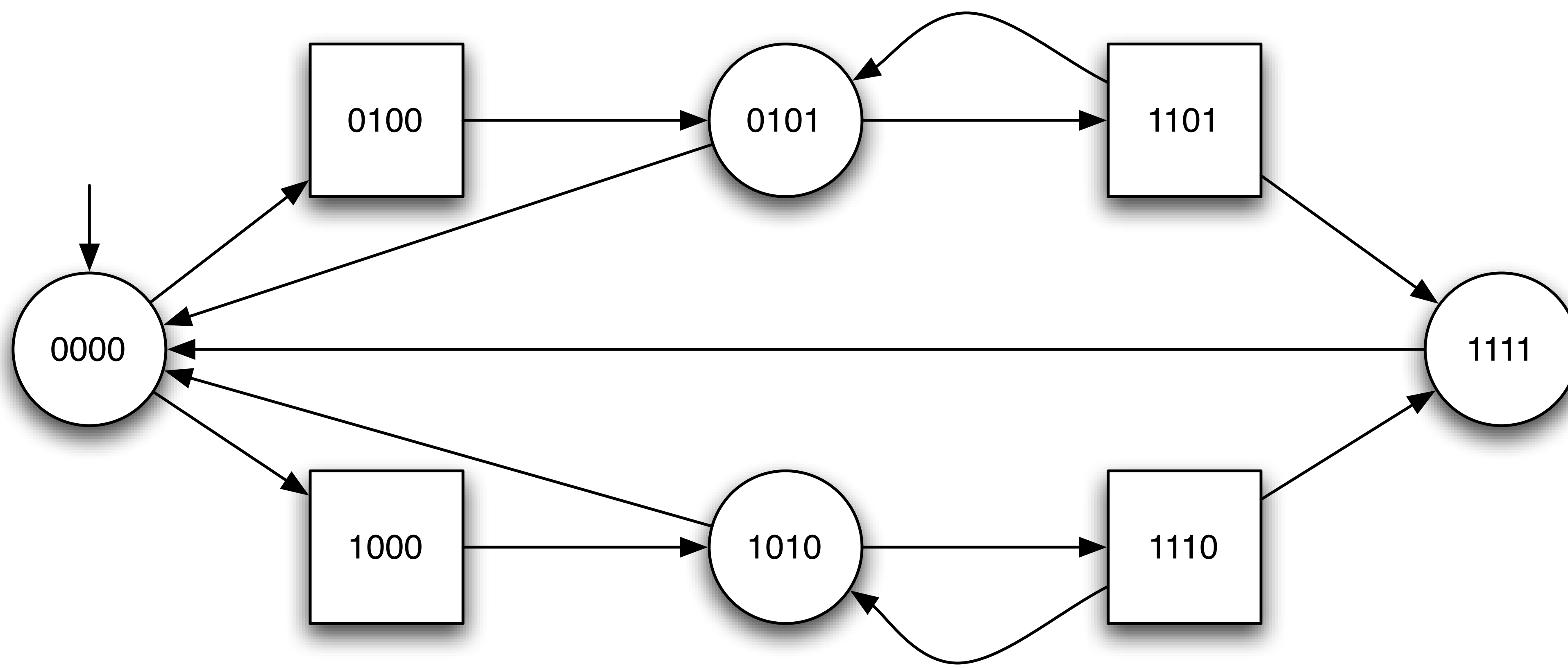
We denote the set of plays in G by : $\mathsf{Plays}(G)$

and

$\mathsf{PrefPlays}(G) = \{q_0 q_1 \dots q_n \mid \exists w \in \mathsf{Plays}(G) \wedge \forall 1 \leq i \leq n : w(i) = q_i\}$

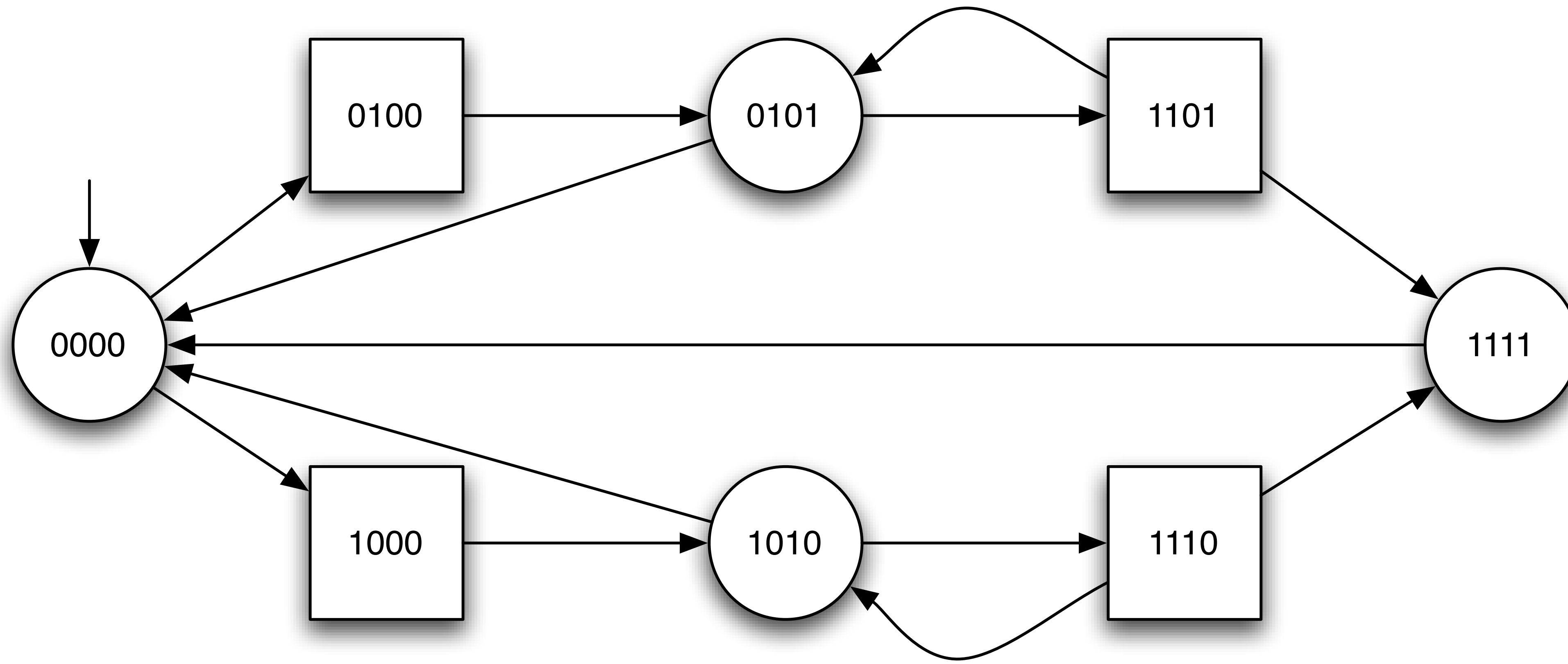
$\mathsf{PrefPlays}_k(G) = \{w \in \mathsf{PrefPlays}(G) \wedge \mathit{last}(w) \in Q_k\}$

Who is winning ?



Play : 0000 0100 0101 1101 ...

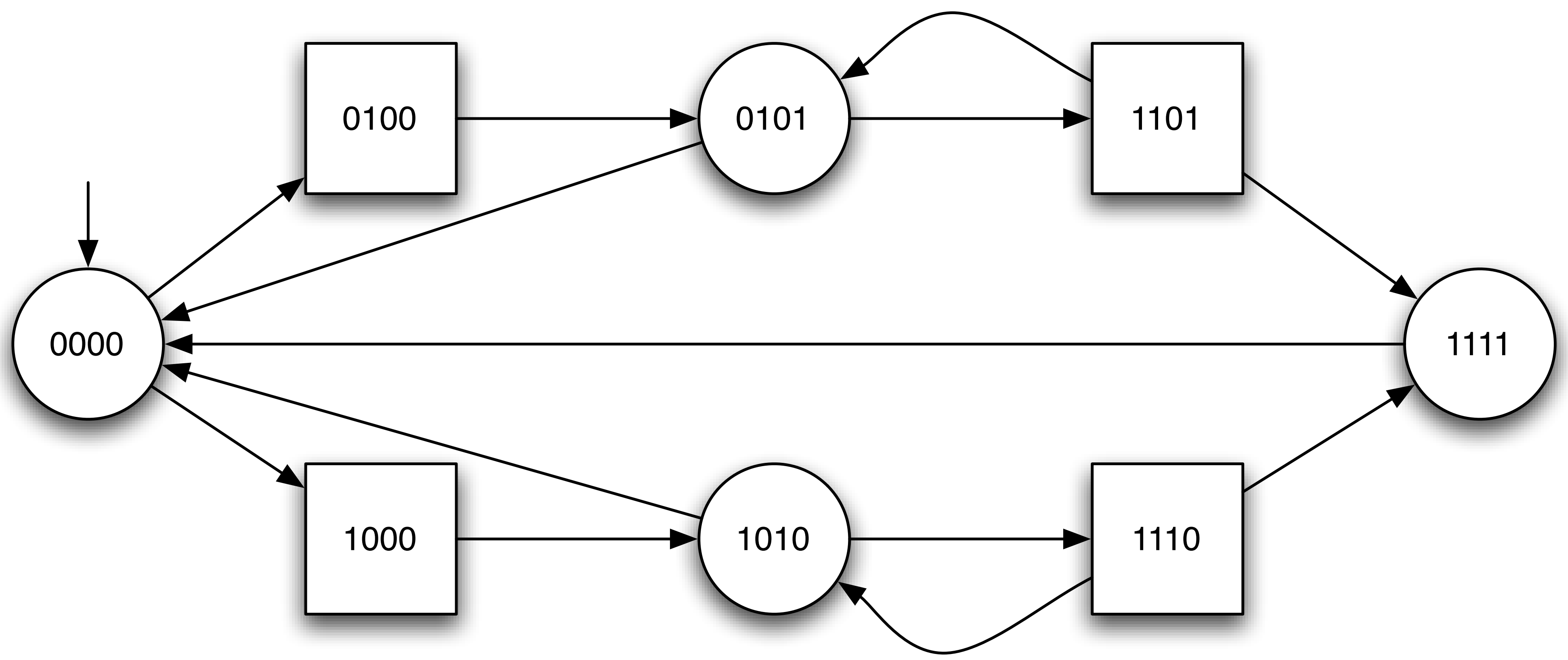
Who is winning ?



Play : 0000 0100 0101 1101 ...

Is this a **good** or a **bad** play for **Player *k*** ?

Who is winning ?



A **winning condition** (for Player *k*)
is a set of plays

$$W \subseteq (Q_1 \cup Q_2)^\omega$$

Game

=

Two-player game structure

+

Winning condition for Player *k*

Strategies

Players are playing **according to strategies.**

A **Player *k* strategy** in *G* is a function:

$$\lambda : \mathsf{PrefPlays}_k(G) \to Q_1 \cup Q_2$$

with the restriction that:

$$\forall w \in \mathsf{PrefPlays}_k(G) : \delta(last(w), \lambda(w))$$

Outcome of a strategy

w is a possible **outcome** of the Player *k* strategy λ if

$$\forall i \geq 0 : w(i) \in Q_k : w(i+1) = \lambda(w(0,i))$$

w is a play where Player k plays according to strategy λ

Outcome of a strategy

w is a possible **outcome** of the Player *k* strategy λ if

$$\forall i \geq 0 : w(i) \in Q_k : w(i+1) = \lambda(w(0,i))$$

The set of plays that have this property is denoted

$$\text{Outcome}_k(G, \lambda)$$

Winning strategy

- Given a pair (G, W)
- We say that Player *k* wins the game (G, W) if and only if:

$$\exists \lambda : \text{Outcome}_k(G, \lambda) \subseteq W$$

Winning strategy

- Given a pair (G, W)
- We say that Player *k* wins the game (G, W) if and only if:

$$\exists \lambda : \text{Outcome}_k(G, \lambda) \subseteq W$$

That is, no matter how the other player resolves his choices, when player *k* plays according to λ , the resulting play belongs to *W*. Player *k* can **force** the play to be in *W*.

Winning strategy

- Given a pair (G, W)
- We say that Player *k* wins the game (G, W) if and only if:

$$\exists\lambda : \text{Outcome}_k(G, \lambda) \subseteq W$$

We say λ that is a **winning strategy** for player k in the game (G, W)

Winning strategies

=

Controllers that enforce winning plays

Winning conditions

- **Not all** winning conditions are reasonable
- One often assumes that the set of winning plays is a **regular set**
- We show here how to solve **reachability** and **safety** games

Reachability Games

Reachability Game

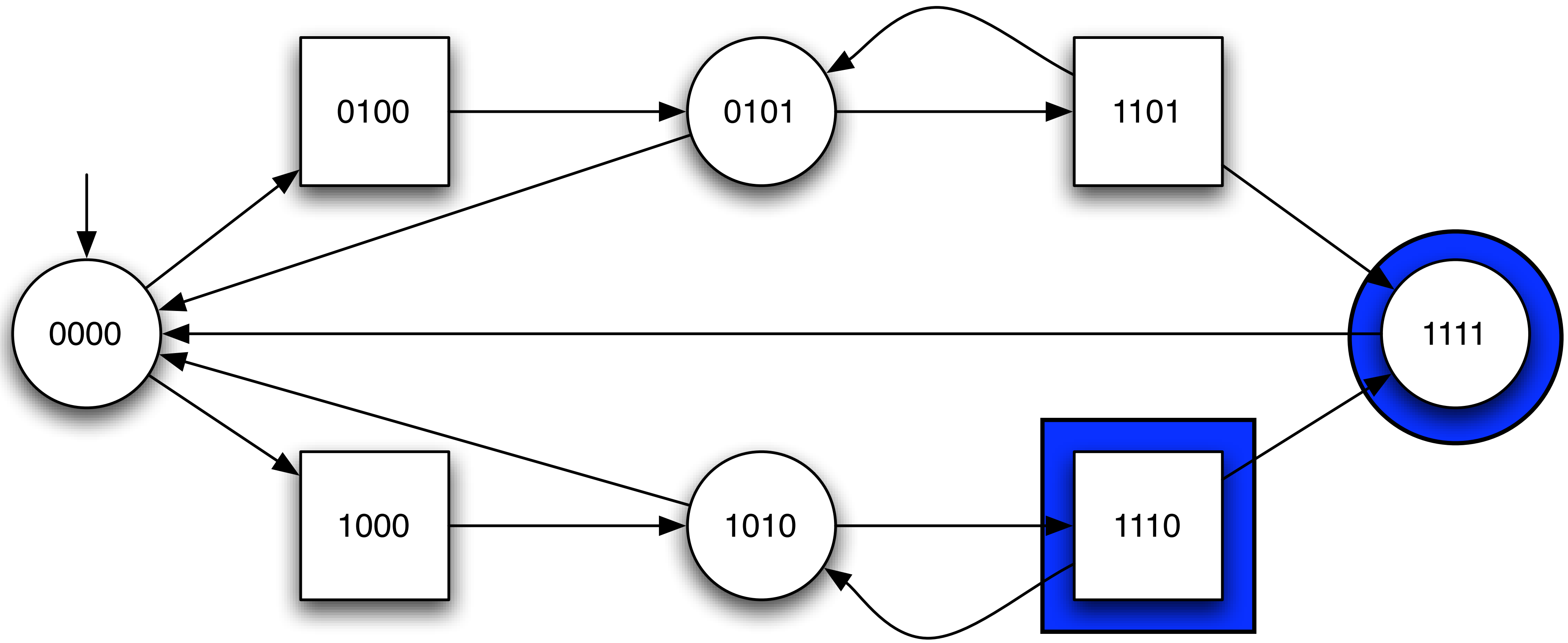
(G, W) is a **reachability game** if

$$\exists Q \subseteq Q_1 \cup Q_2 : W = \{w \in \text{Plays}(G) \mid \exists i : w(i) \in Q\}$$

That is *W* is a set of plays that reaches the set of locations *Q*.

$\text{Reach}(G, Q)$

A Reachability Game



Does Player I, who owns the rounded positions, have a strategy (against any choices of Player II) to reach the set $\{1101, 1111\}$?

Safety Games

Safety Game

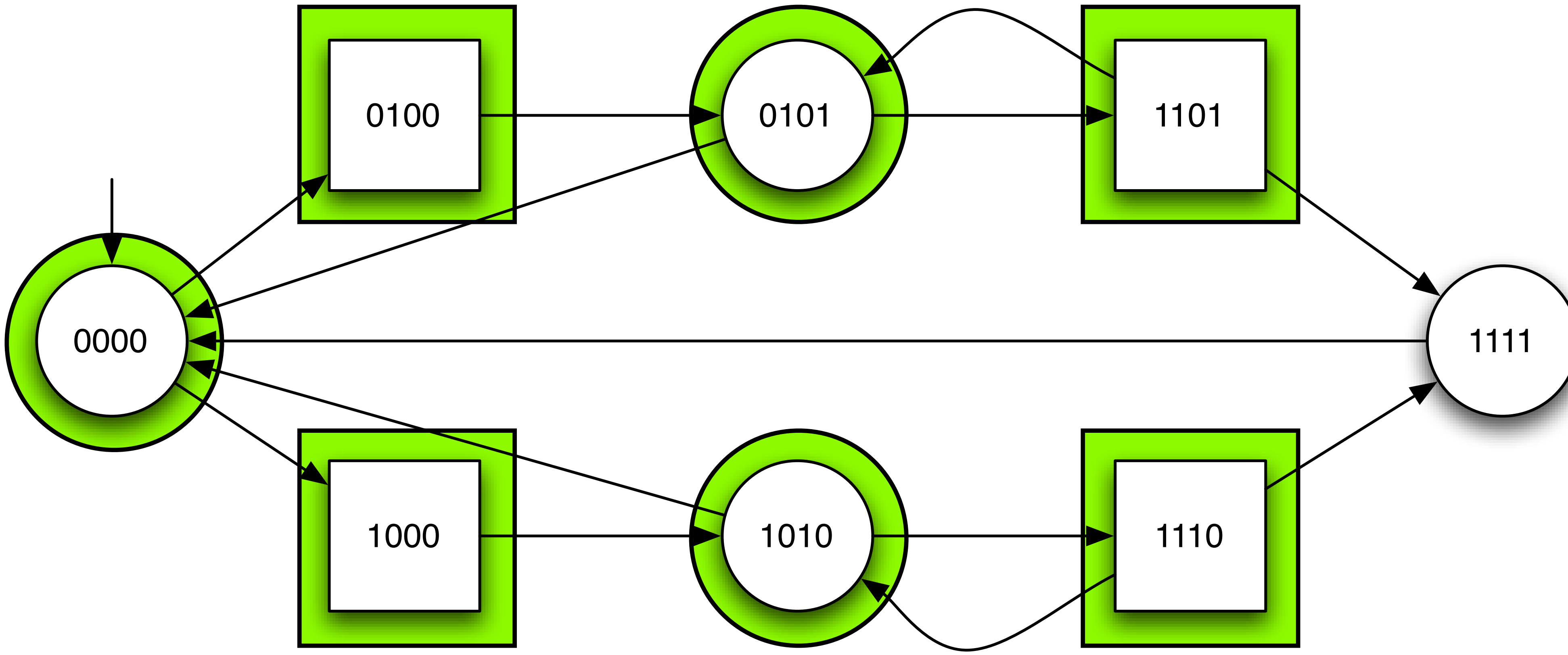
(G, W) is a **safety game** if

$$\exists Q \subseteq Q_1 \cup Q_2 : W = \{w \in \mathsf{Plays}(G) \mid \forall i \geq 0 : w(i) \in Q\}$$

That is *W* is the set of plays that stay within given set of positions *Q*.

$\mathsf{Safe}(G, Q)$

A Safety Game



Does Player I, who owns the rounded positions, have a strategy (against any choices of Player II) to stay within the set of states

$Q \setminus \{1111\}$?

Symbolic algorithms to solve games

Player *k* Controllable Predecessors

X is a set of positions

$$\mathsf{1CPre}_G(X) = \{q \in Q_1 \mid \exists q' : \delta(q,q') \wedge q' \in X\} \cup \{q \in Q_2 \mid \forall q' : \delta(q,q') : q' \in X\}$$

Set of Player I positions where he has a choice of successor that lies in *X*

Set of Player II positions where all her choices for successors lie in *X*

Player *k* Controllable Predecessors

$$\mathsf{1CPre}_G(X) = \{q \in Q_1 \mid \exists q' : \delta(q,q') \wedge q' \in X\} \cup \{q \in Q_2 \mid \forall q' : \delta(q,q') : q' \in X\}$$

Symmetrically

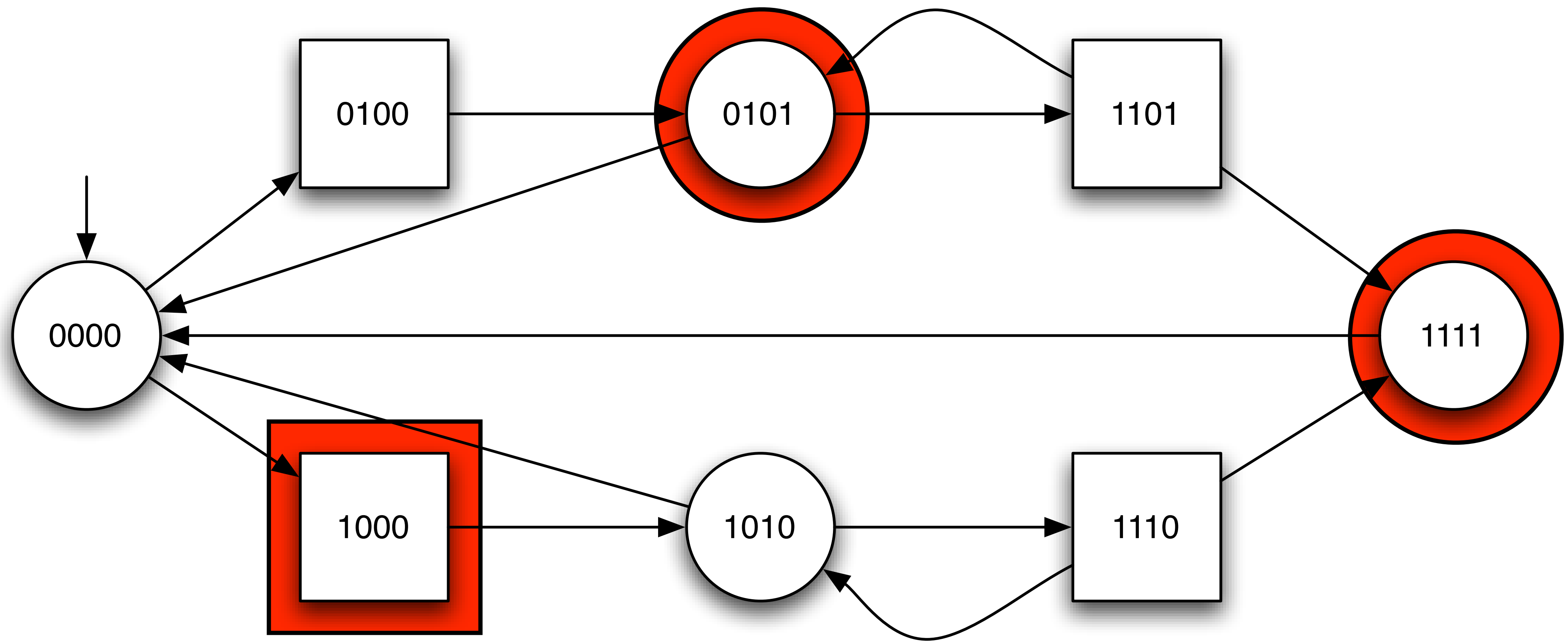
$$\mathsf{2CPre}_G(X) = \{q \in Q_2 \mid \exists q' : \delta(q,q') \wedge q' \in X\} \cup \{q \in Q_1 \mid \forall q' : \delta(q,q') : q' \in X\}$$

Player *k* Controllable Predecessors

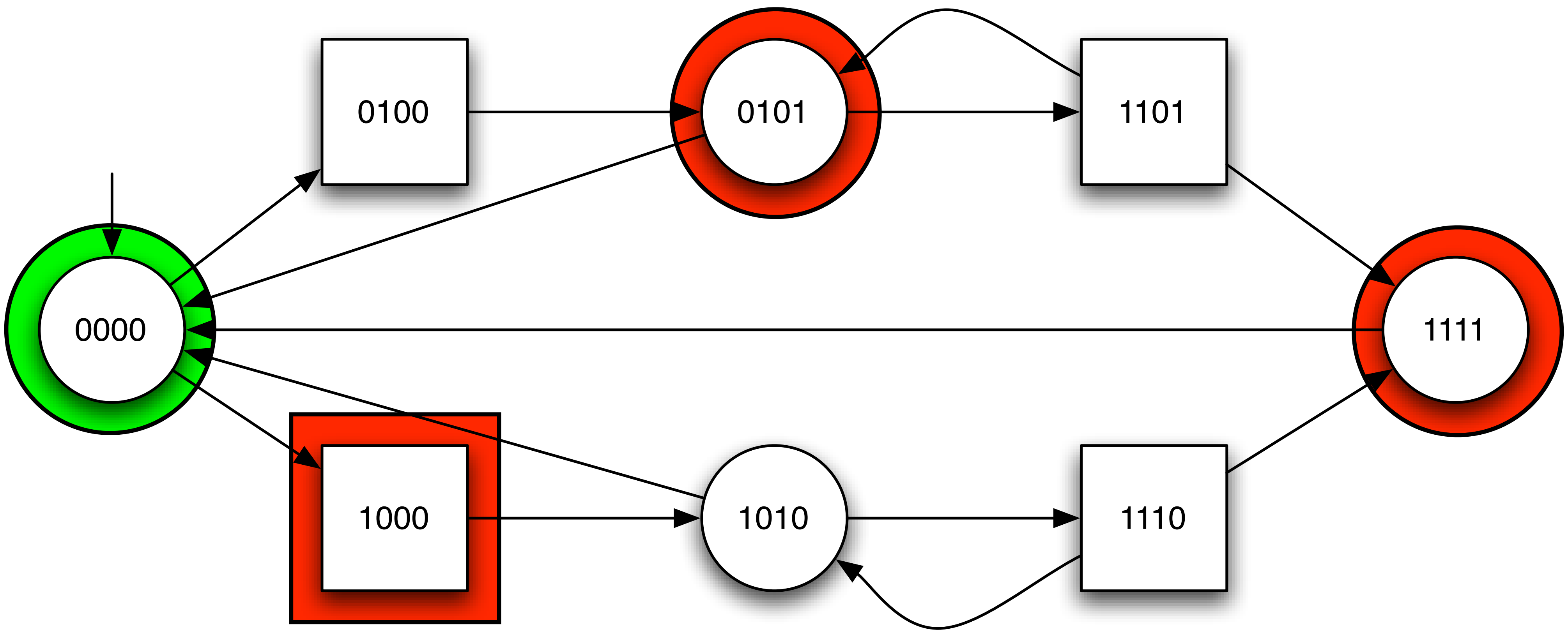
$$\mathsf{1CPre}_G(X) = \{q \in Q_1 \mid \exists q' : \delta(q, q') \wedge q' \in X\} \cup \{q \in Q_2 \mid \forall q' : \delta(q, q') : q' \in X\}$$

Monotonic functions over $\langle 2^{Q_1 \cup Q_2}, \subseteq \rangle$

$$\mathsf{2CPre}_G(X) = \{q \in Q_2 \mid \exists q' : \delta(q, q') \wedge q' \in X\} \cup \{q \in Q_1 \mid \forall q' : \delta(q, q') : q' \in X\}$$



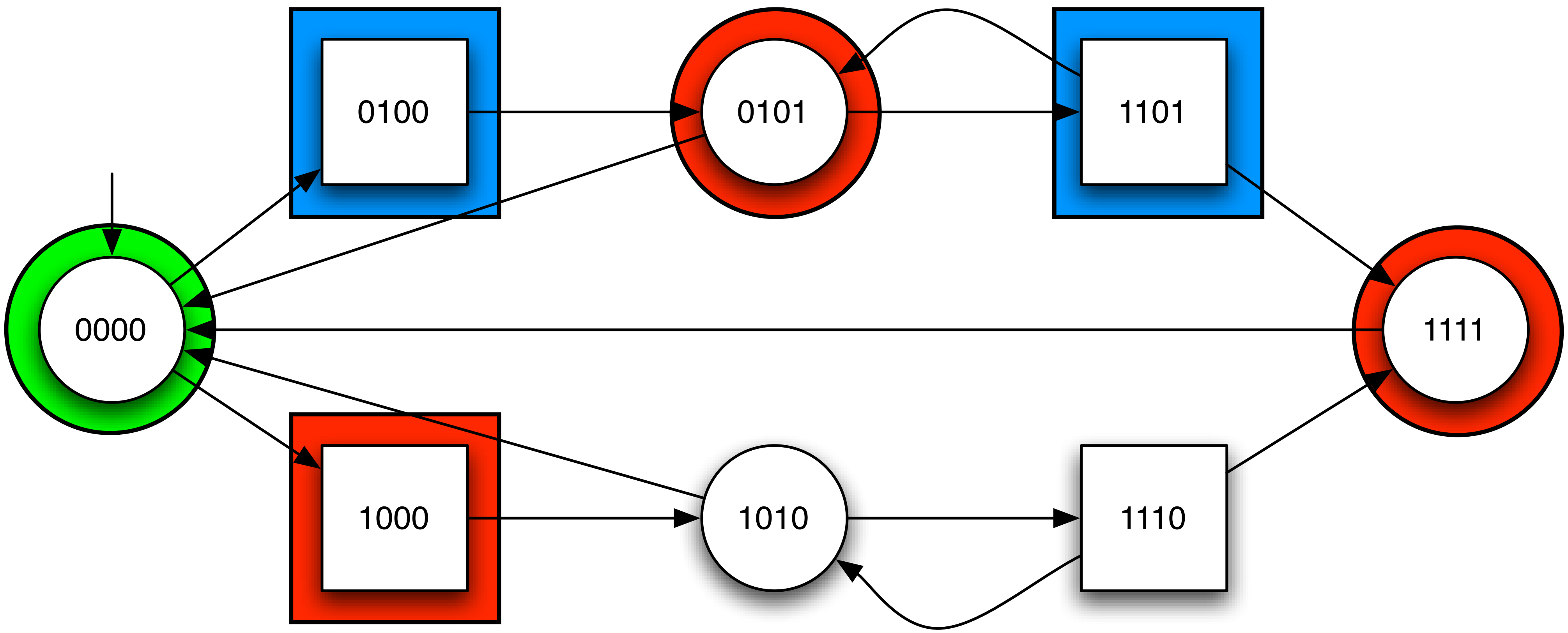
$$X = \{1000, 0101, 1111\}$$



$$X = \{1000, 0101, 1111\}$$

$$\text{1CPre}(X) = \{0000\} \cup \{0100, 1101\}$$

Rounded positions,
there exists a red successor



$$X = \{1000, 0101, 1111\}$$

$$1\mathrm{CPre}(X) = \{0000\} \cup \{0100, 1101\}$$

Rounded positions,
there exists a red successor

Squared positions,
all successors are red

Fixpoints to Solve Games

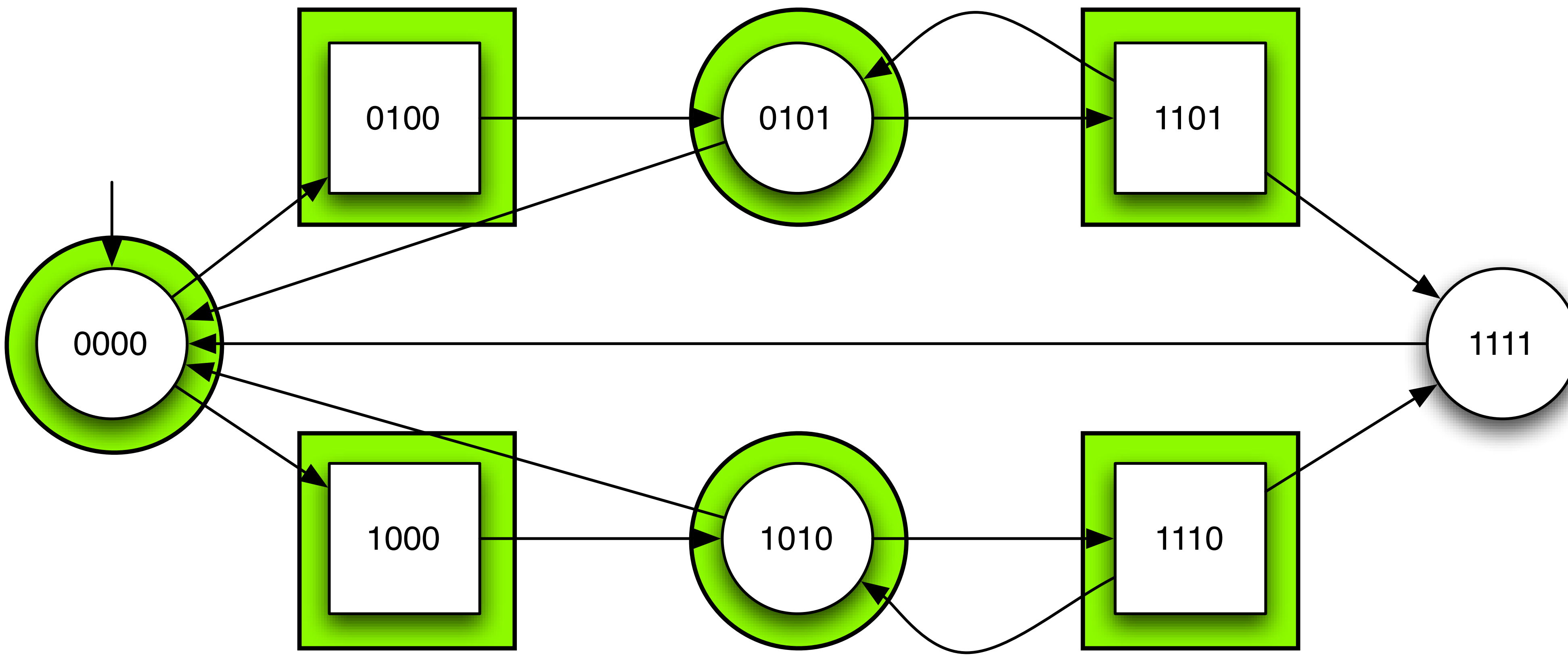
Reachability game for set Q

$$\mu X \cdot Q \cup \mathsf{1CPre}(X)$$

Safety game for set Q

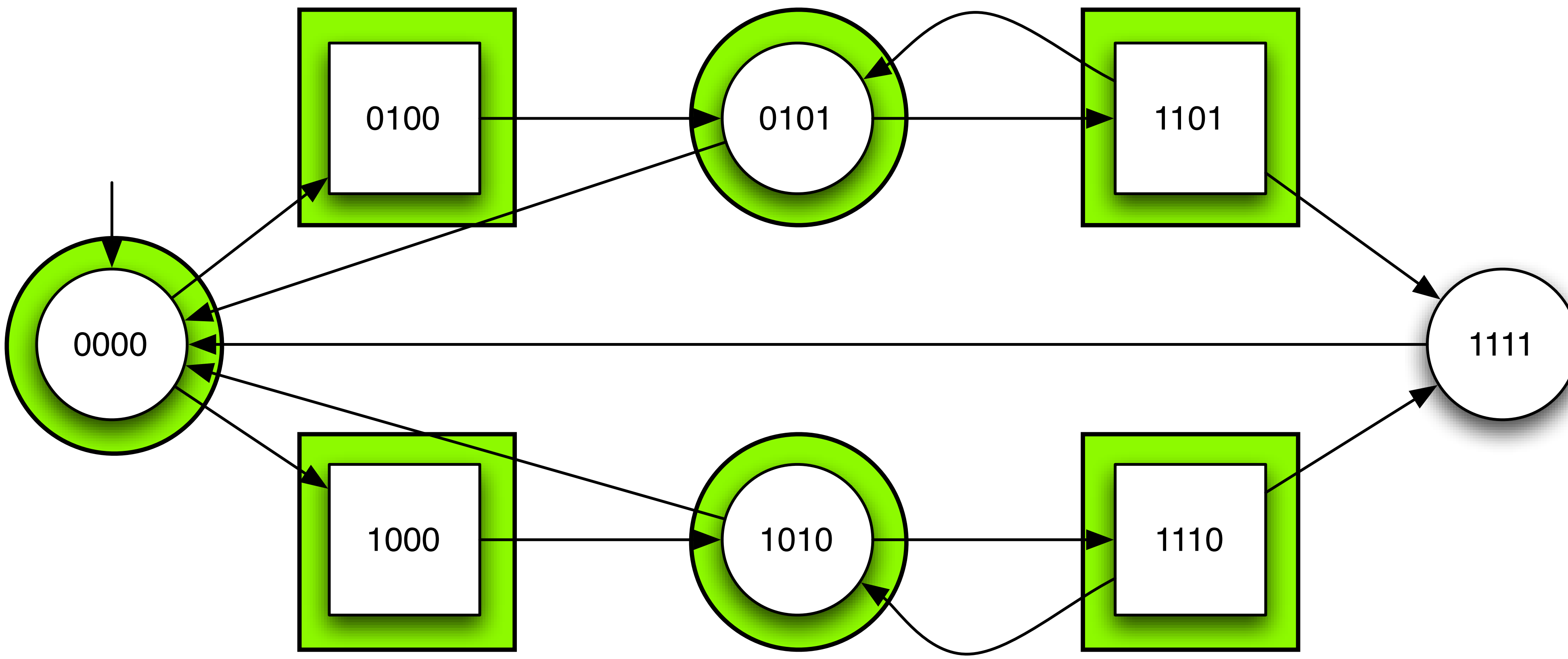
$$\nu X \cdot Q \cap \mathsf{1CPre}(X)$$

Fixpoint for a safety game



Does Player I, who owns the rounded positions, have a strategy to stay within the set of states $Q \setminus \{1111\}$?

Fixpoint for a safety game

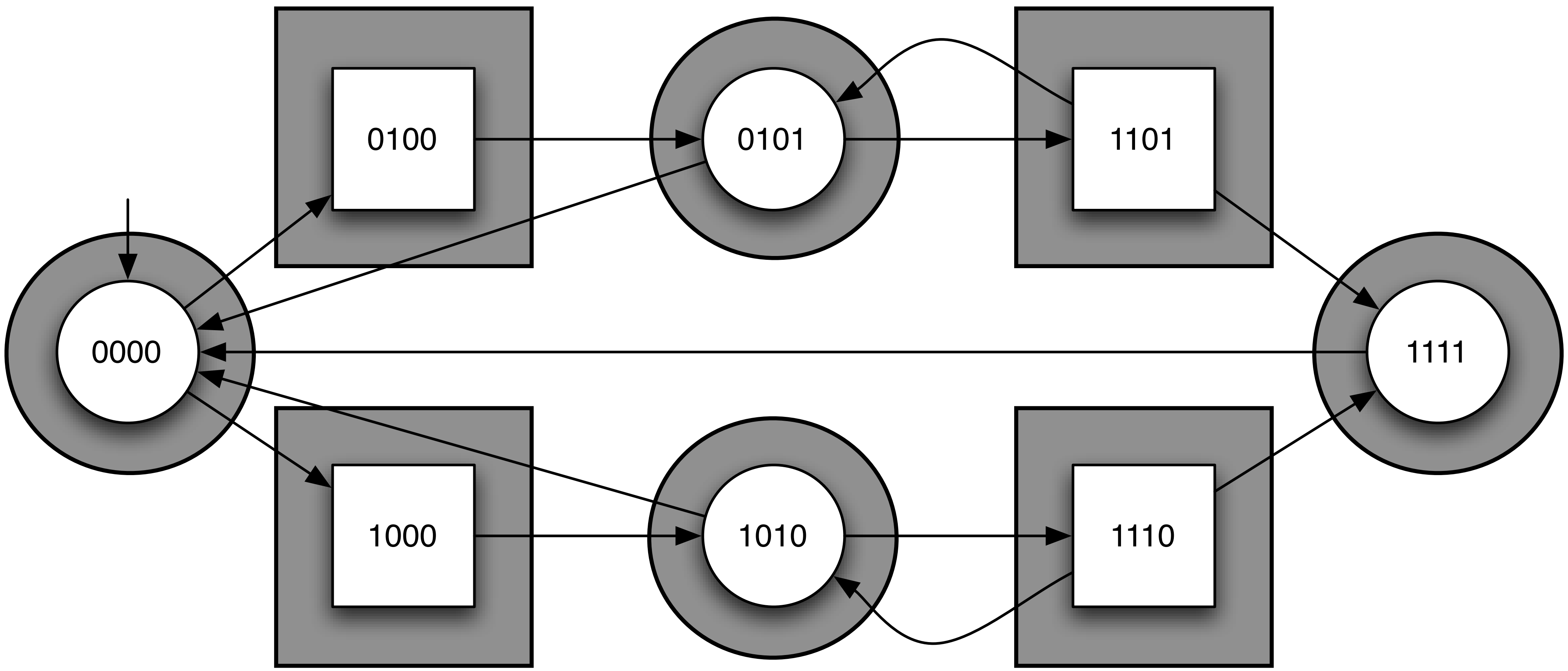


We must compute

$$\nu X \cdot (Q \setminus \{1111\}) \cap \mathsf{1CPre}(X)$$

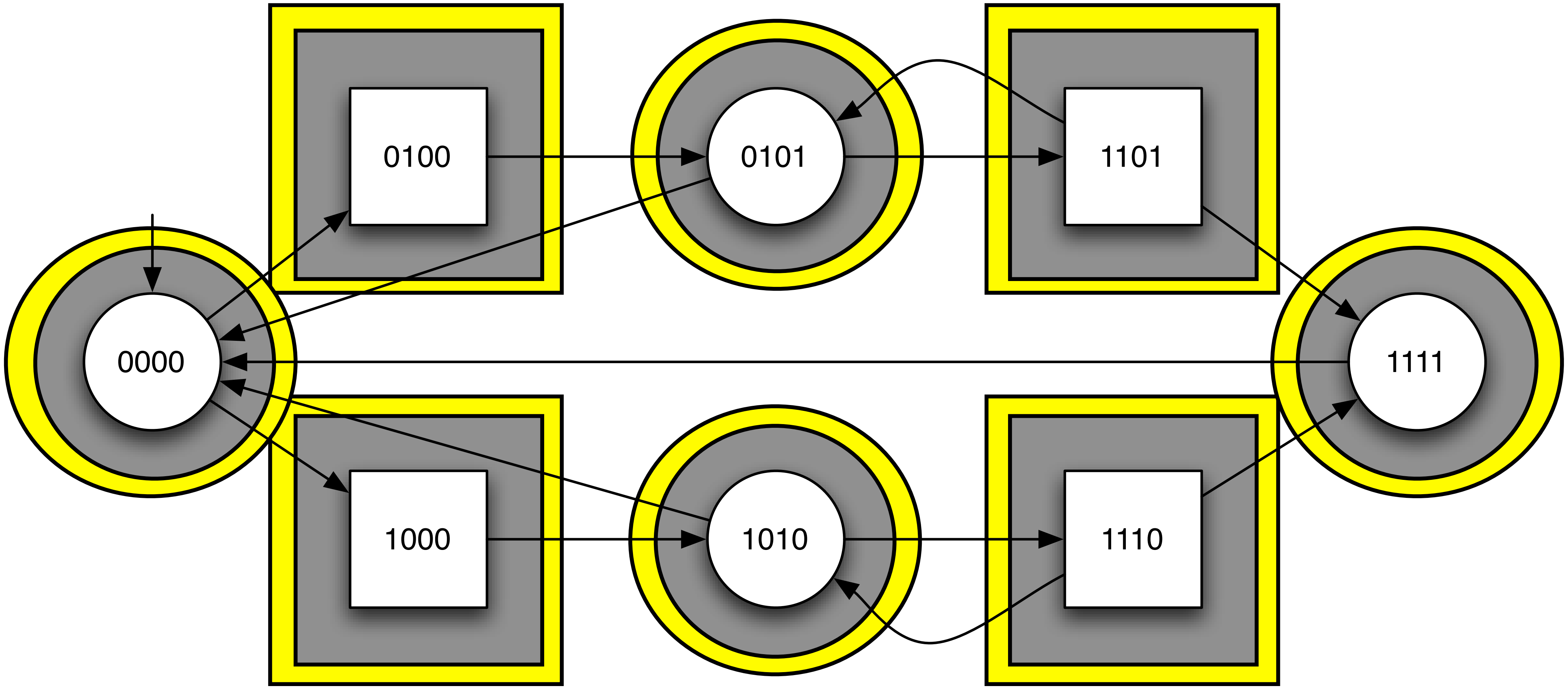
To do that, we use the Tarski fixpoint theorem.

Fixpoint for a safety game



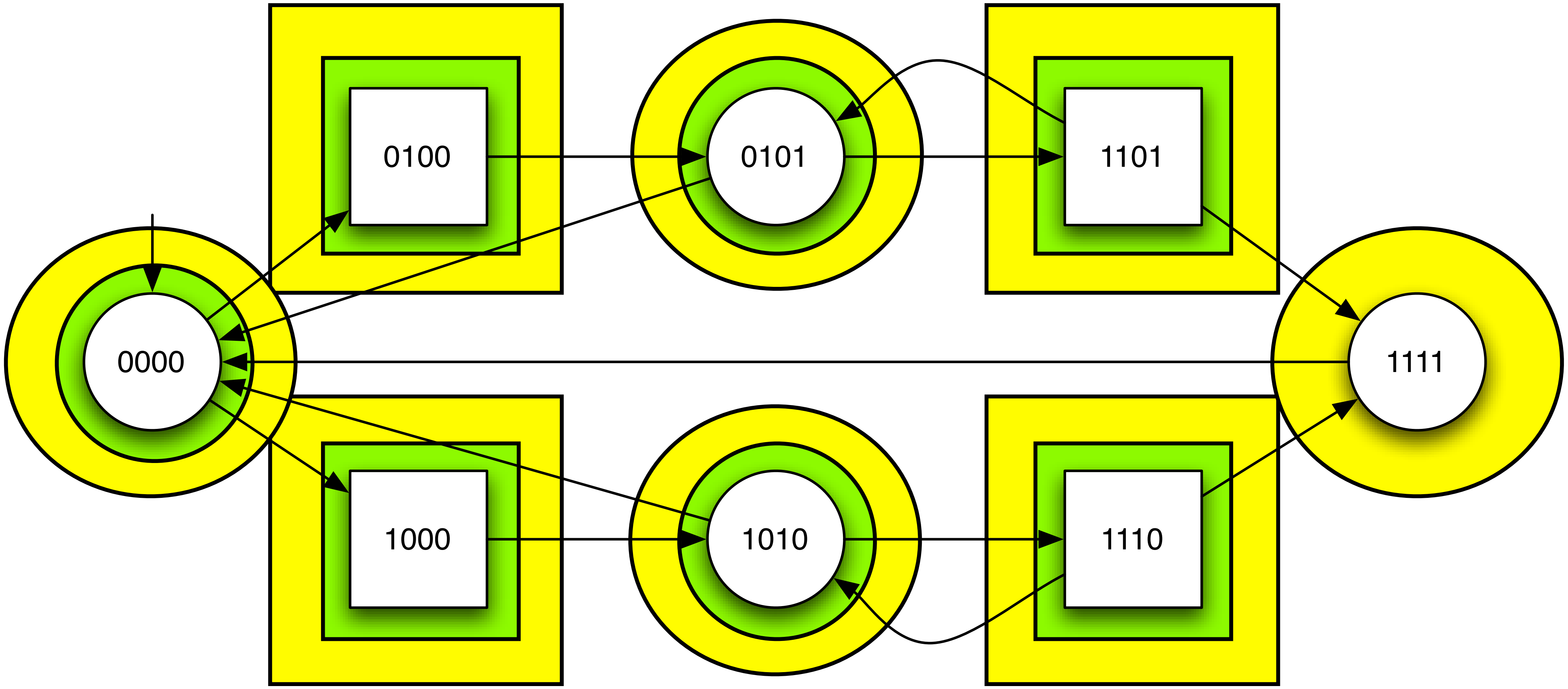
$$X_0 = (Q \setminus \{1111\}) \cap \text{1CPre}(Q)$$

Fixpoint for a safety game



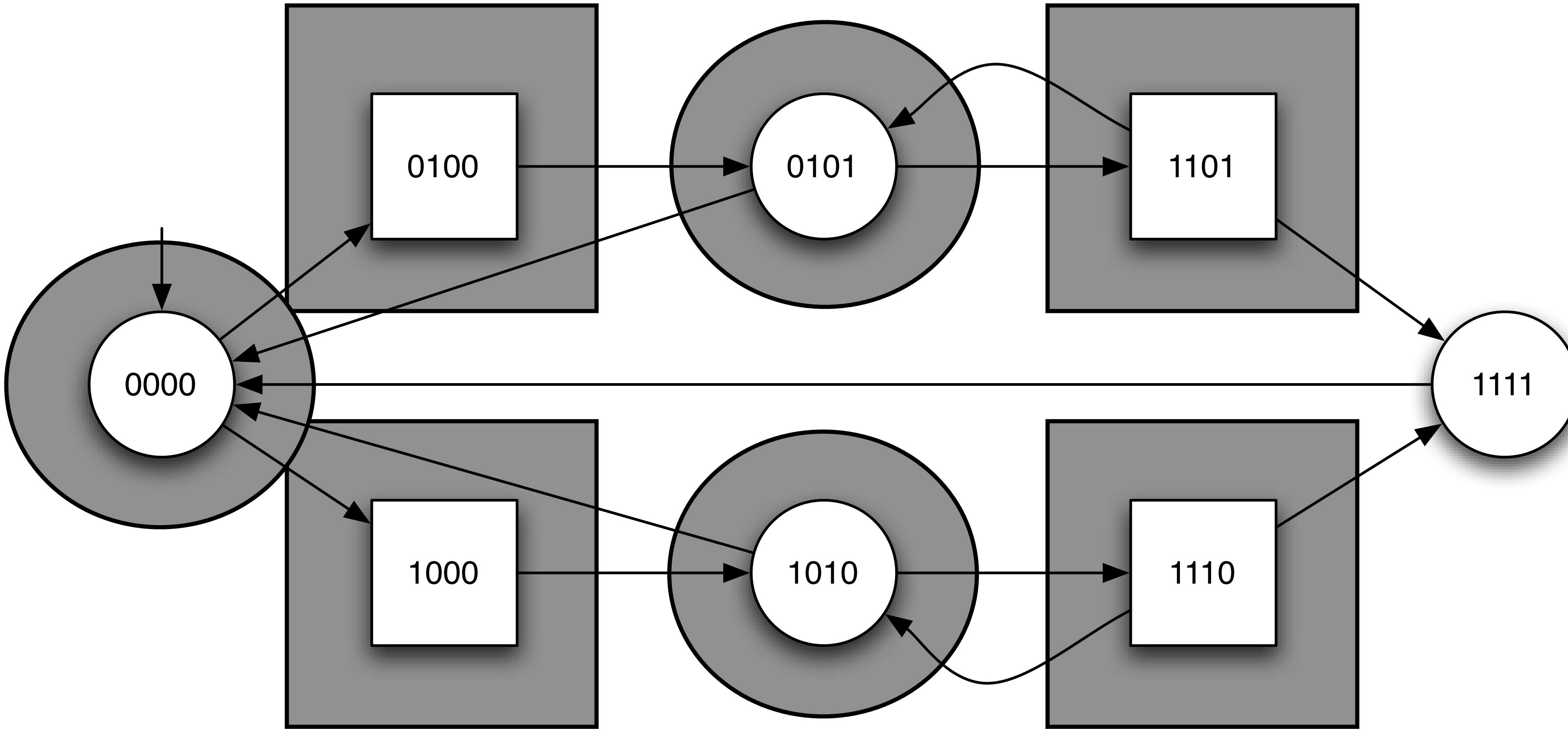
$$X_0 = (Q \setminus \{1111\}) \cap \text{1CPre}(Q)$$

Fixpoint for a safety game



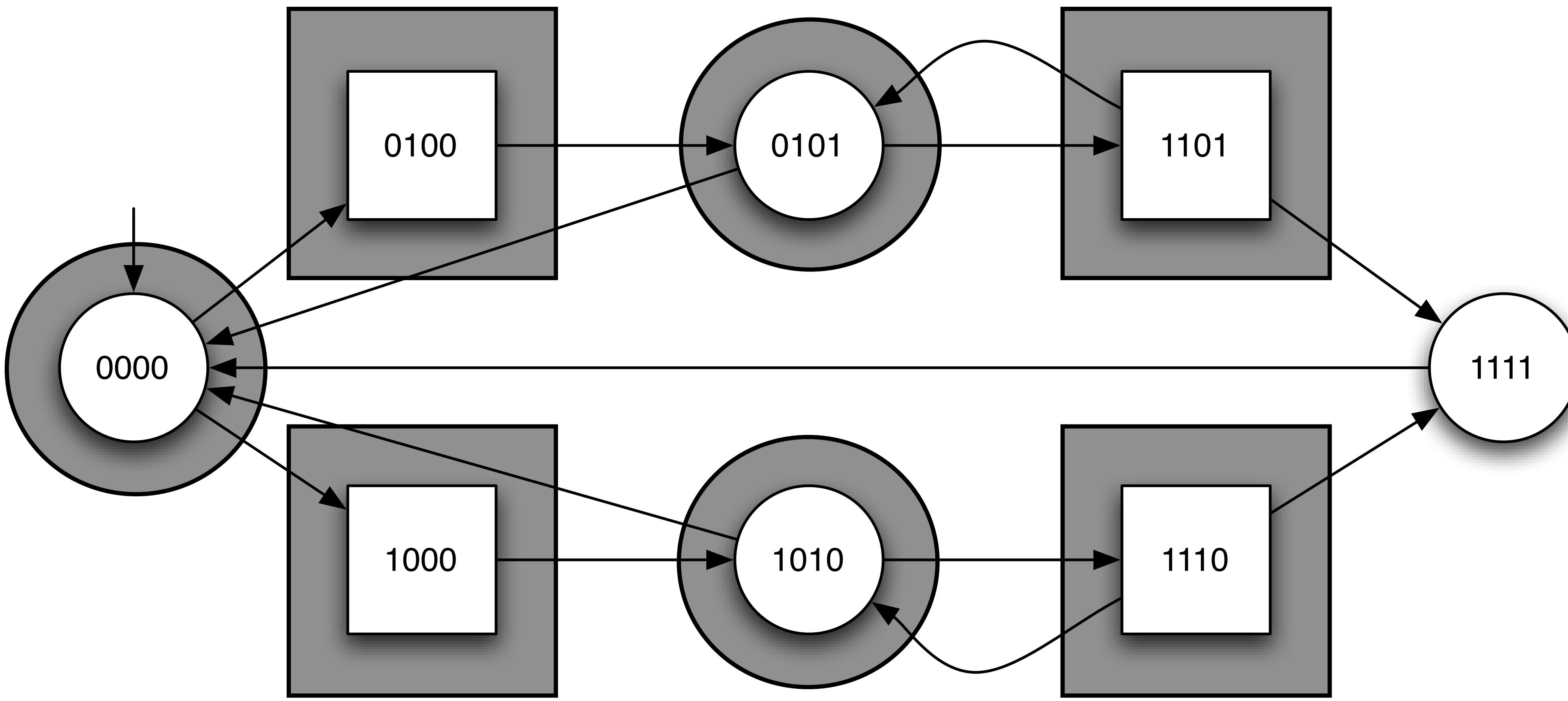
$$X_0 = (Q \setminus \{1111\}) \cap \mathsf{1CPre}(Q)$$

Fixpoint for a safety game



$$X_0 = (Q \setminus \{1111\}) \cap \mathsf{1CPre}(Q)$$

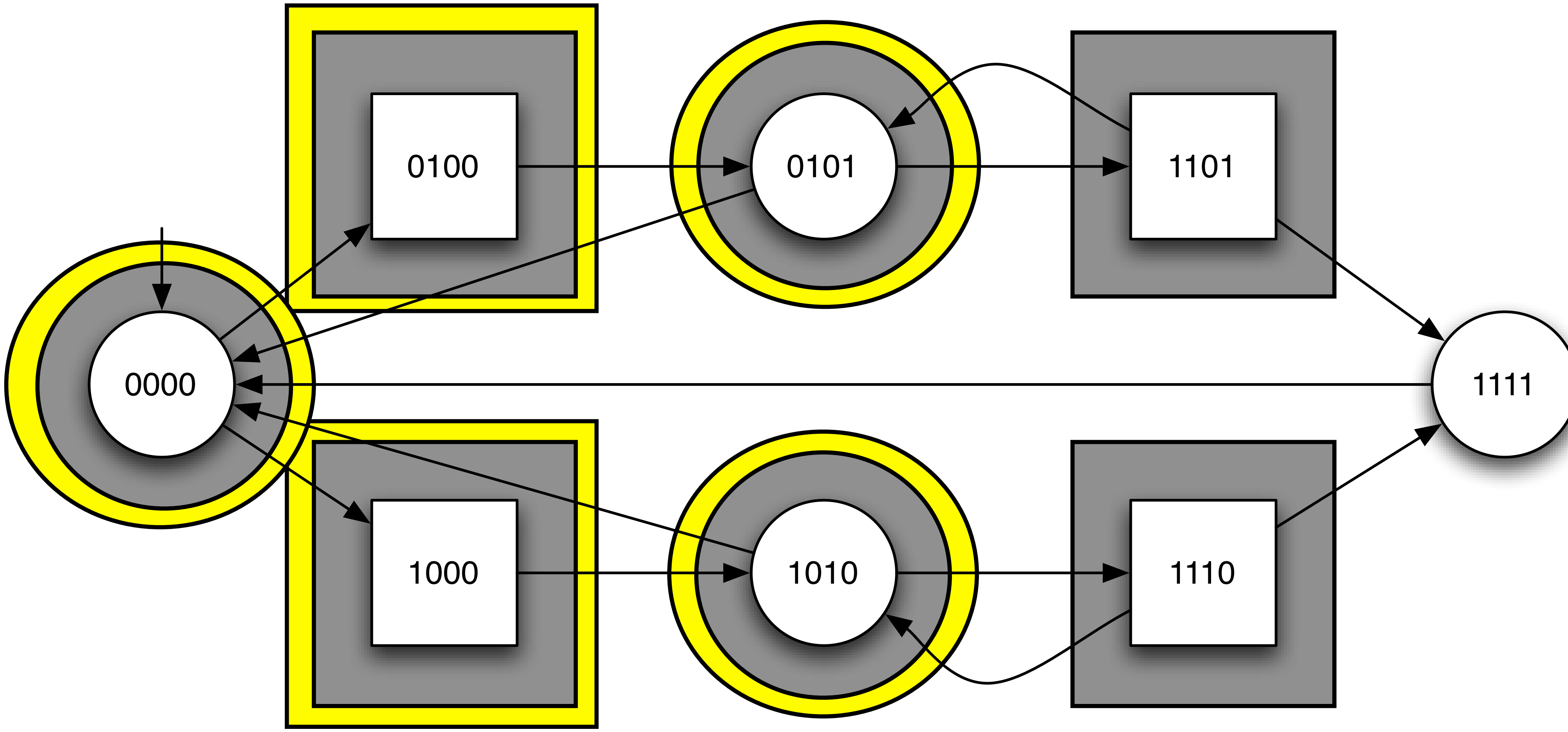
Fixpoint for a safety game



$$X_0 = (Q \setminus \{1111\}) \cap \mathsf{1CPre}(Q)$$

$$X_1 = (Q \setminus \{1111\}) \cap \mathsf{1CPre}(X_0)$$

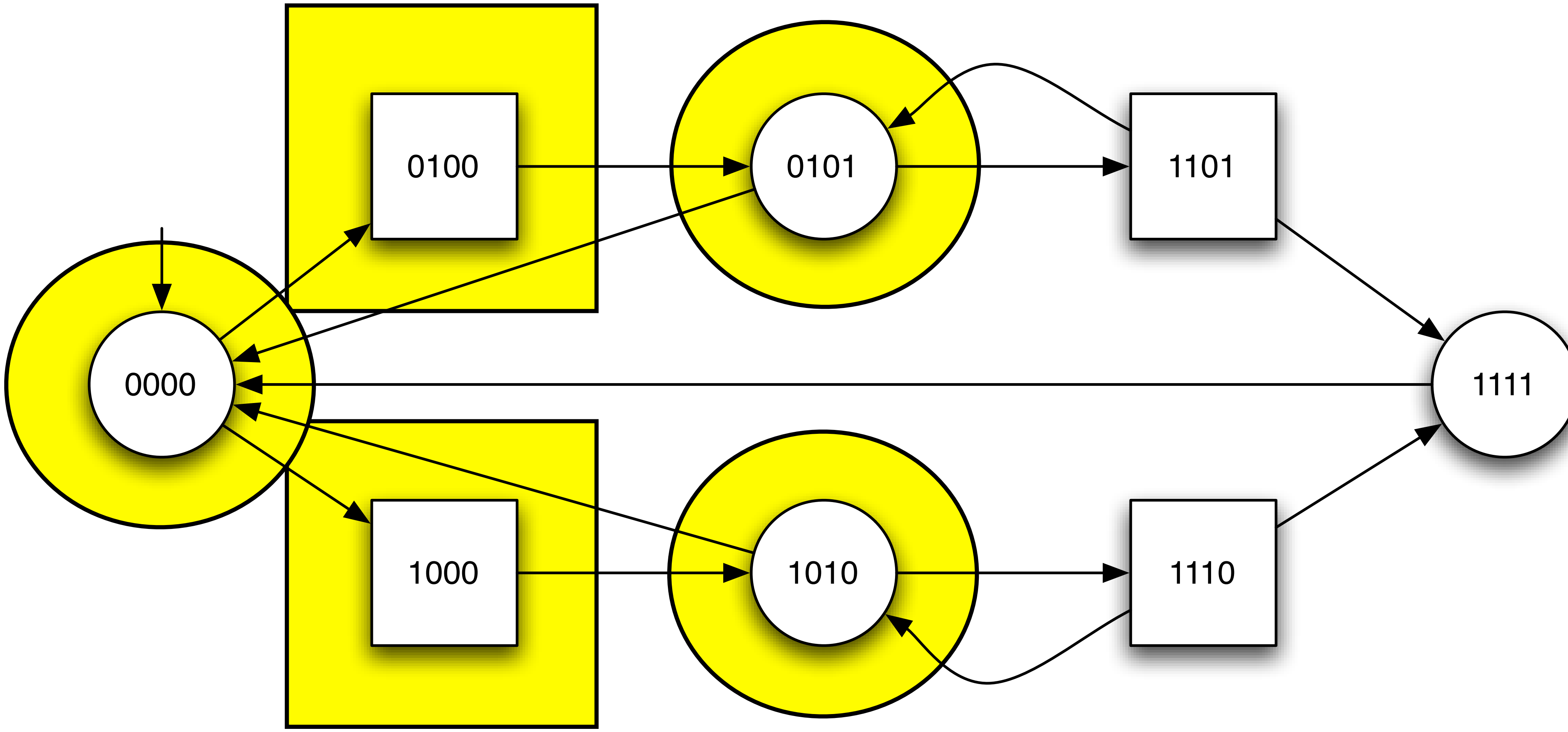
Fixpoint for a safety game



$$X_0 = (Q \setminus \{1111\}) \cap \mathsf{1CPre}(Q)$$

$$X_1 = (Q \setminus \{1111\}) \cap \mathsf{1CPre}(X_0)$$

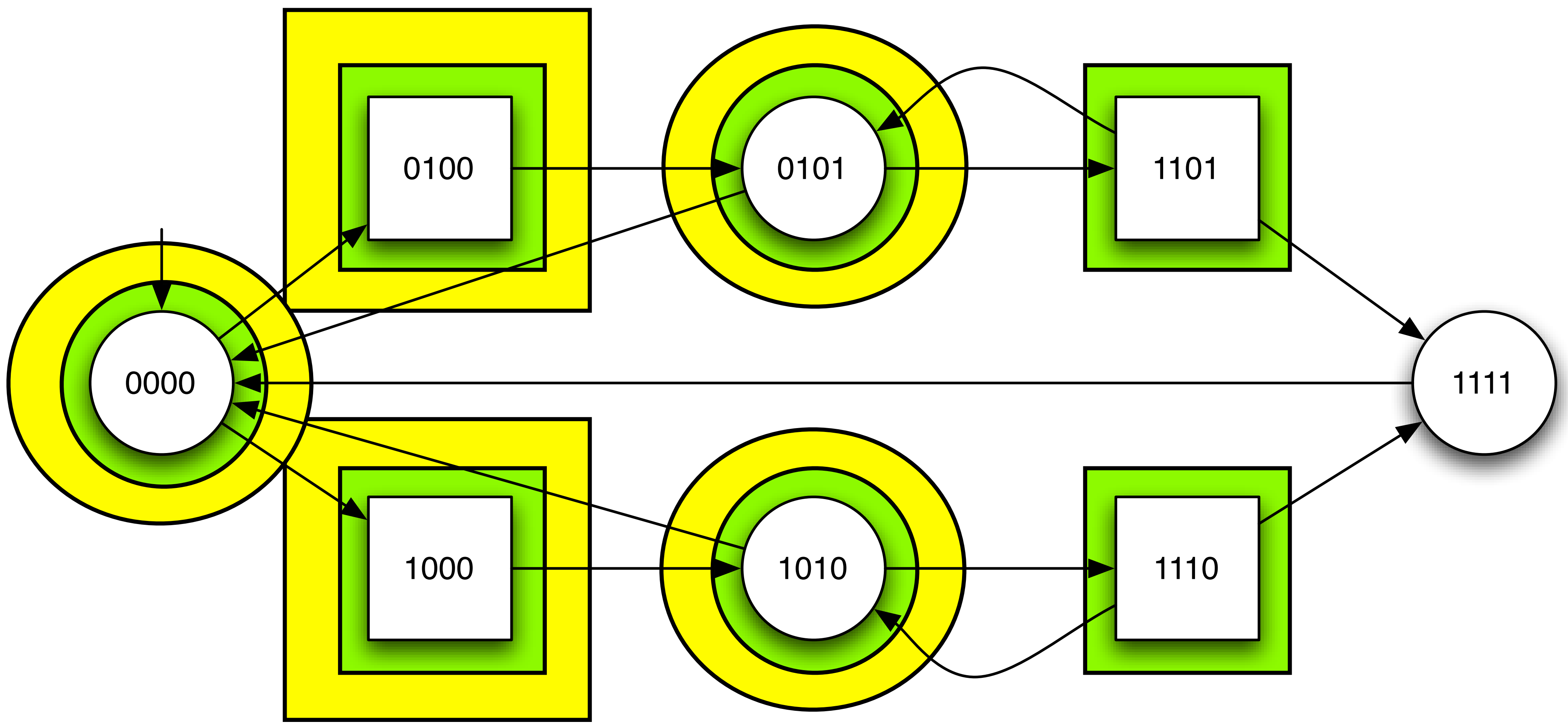
Fixpoint for a safety game



$$X_0 = (Q \setminus \{1111\}) \cap \text{1CPre}(Q)$$

$$X_1 = (Q \setminus \{1111\}) \cap \text{1CPre}(X_0)$$

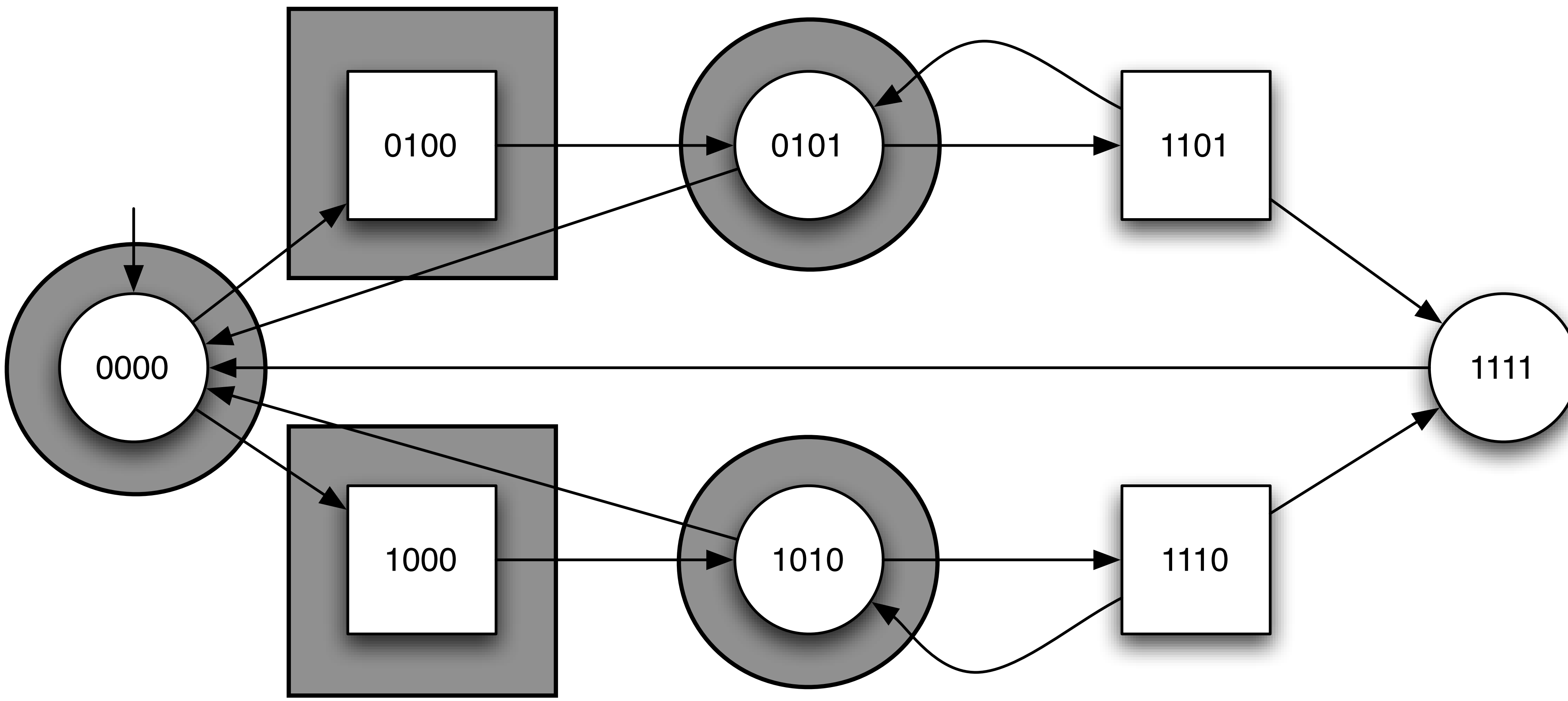
Fixpoint for a safety game



$$X_0 = (Q \setminus \{1111\}) \cap \mathsf{1CPre}(Q)$$

$$X_1 = (Q \setminus \{1111\}) \cap \mathsf{1CPre}(X_0)$$

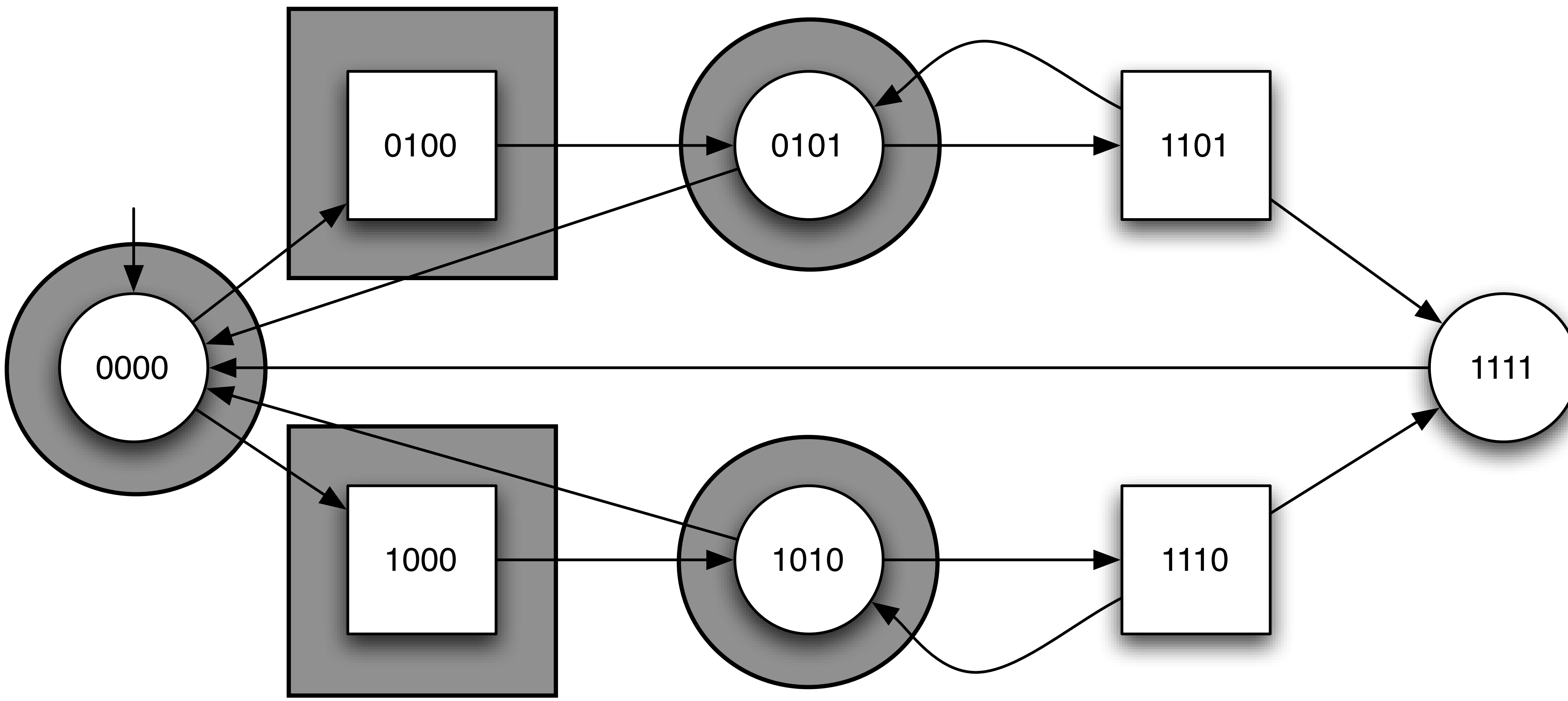
Fixpoint for a safety game



$$X_0 = (Q \setminus \{1111\}) \cap \mathsf{1CPre}(Q)$$

$$X_1 = (Q \setminus \{1111\}) \cap \mathsf{1CPre}(X_0)$$

Fixpoint for a safety game

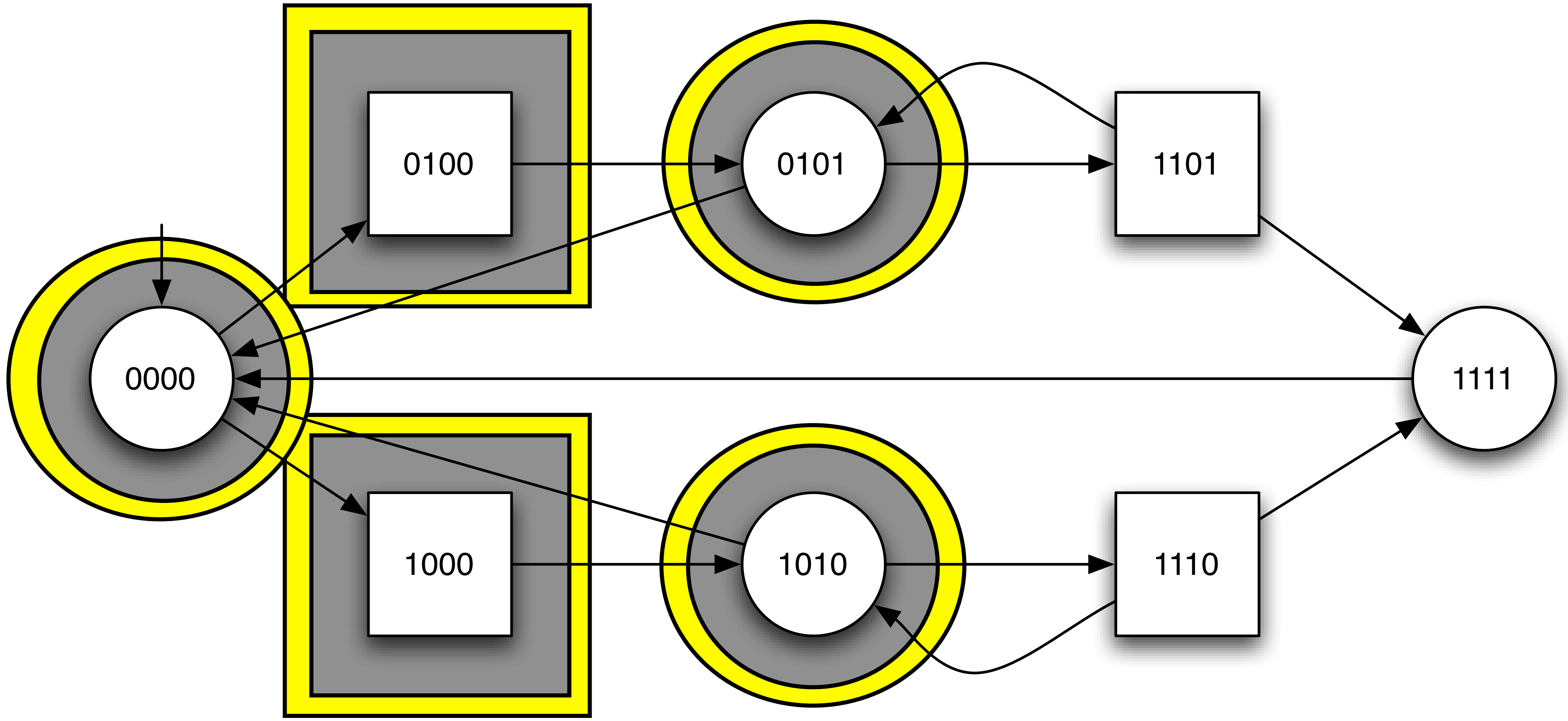


$$X_0 = (Q \setminus \{1111\}) \cap \mathsf{1CPre}(Q)$$

$$X_1 = (Q \setminus \{1111\}) \cap \mathsf{1CPre}(X_0)$$

$$X_2 = (Q \setminus \{1111\}) \cap \mathsf{1CPre}(X_1)$$

Fixpoint for a safety game

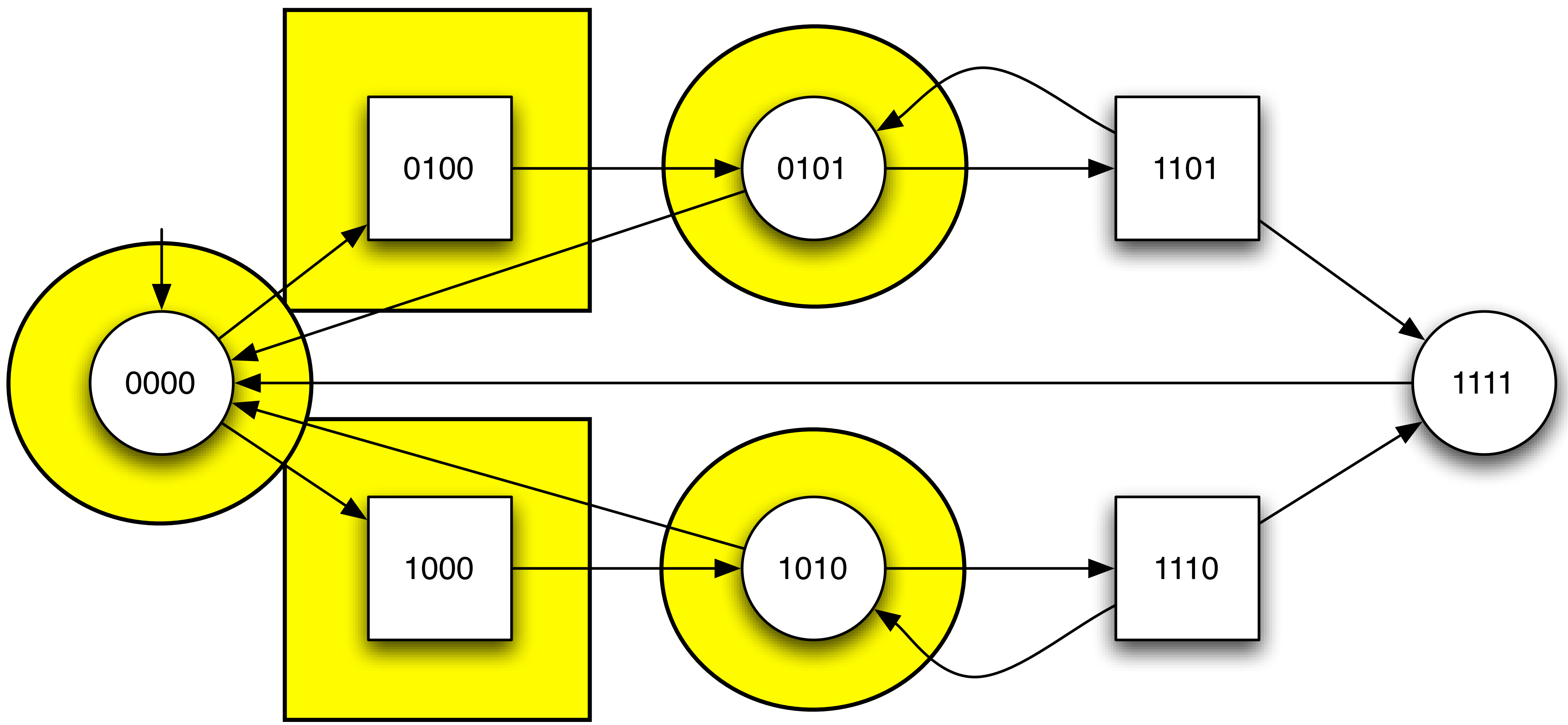


$$X_0 = (Q \setminus \{1111\}) \cap \mathsf{1CPre}(Q)$$

$$X_1 = (Q \setminus \{1111\}) \cap \mathsf{1CPre}(X_0)$$

$$X_2 = (Q \setminus \{1111\}) \cap \mathsf{1CPre}(X_1)$$

Fixpoint for a safety game

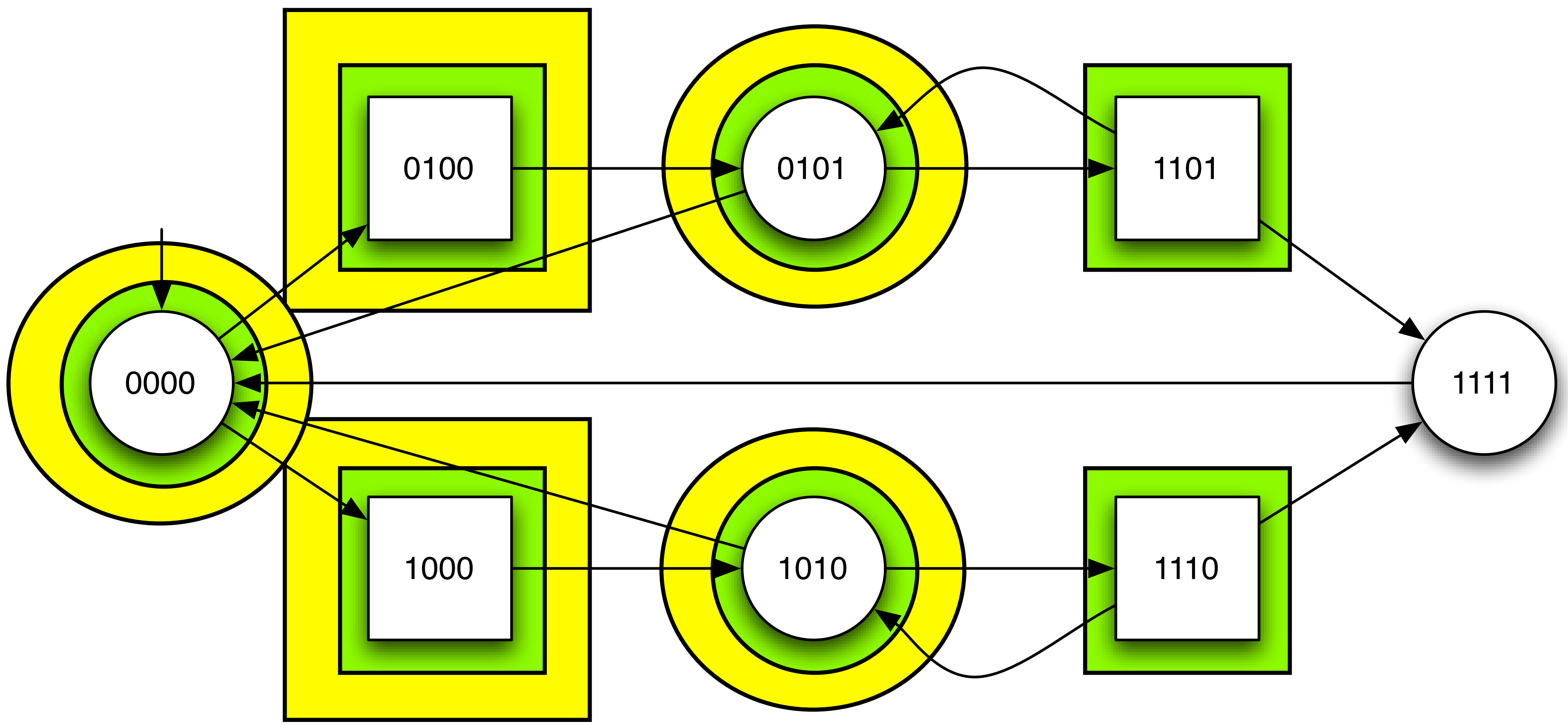


$$X_0 = (Q \setminus \{1111\}) \cap \mathsf{1CPre}(Q)$$

$$X_1 = (Q \setminus \{1111\}) \cap \mathsf{1CPre}(X_0)$$

$$X_2 = (Q \setminus \{1111\}) \cap \mathsf{1CPre}(X_1)$$

Fixpoint for a safety game

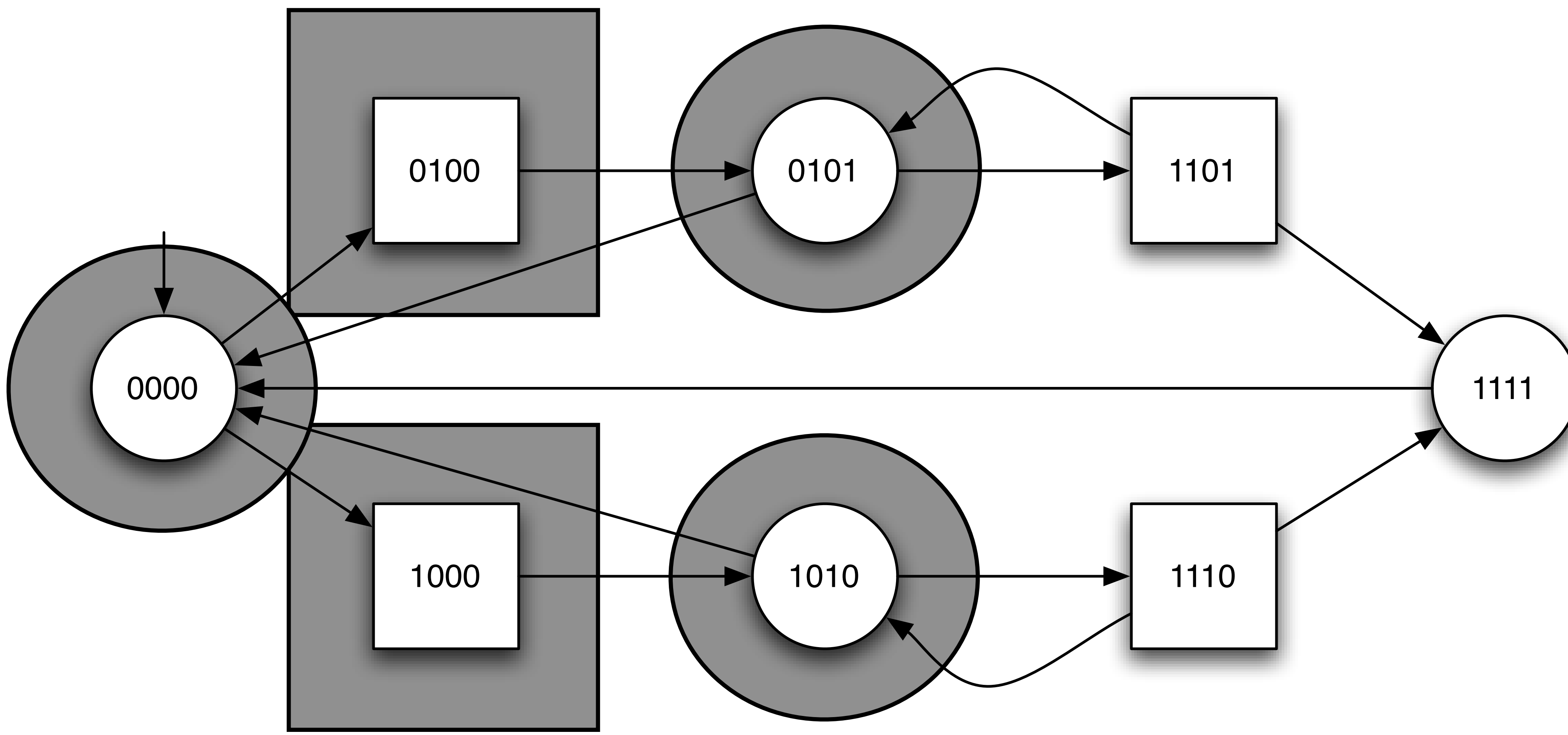


$$X_0 = (Q \setminus \{1111\}) \cap \mathsf{1CPre}(Q)$$

$$X_1 = (Q \setminus \{1111\}) \cap \mathsf{1CPre}(X_0)$$

$$X_2 = (Q \setminus \{1111\}) \cap \mathsf{1CPre}(X_1)$$

Fixpoint for a safety game

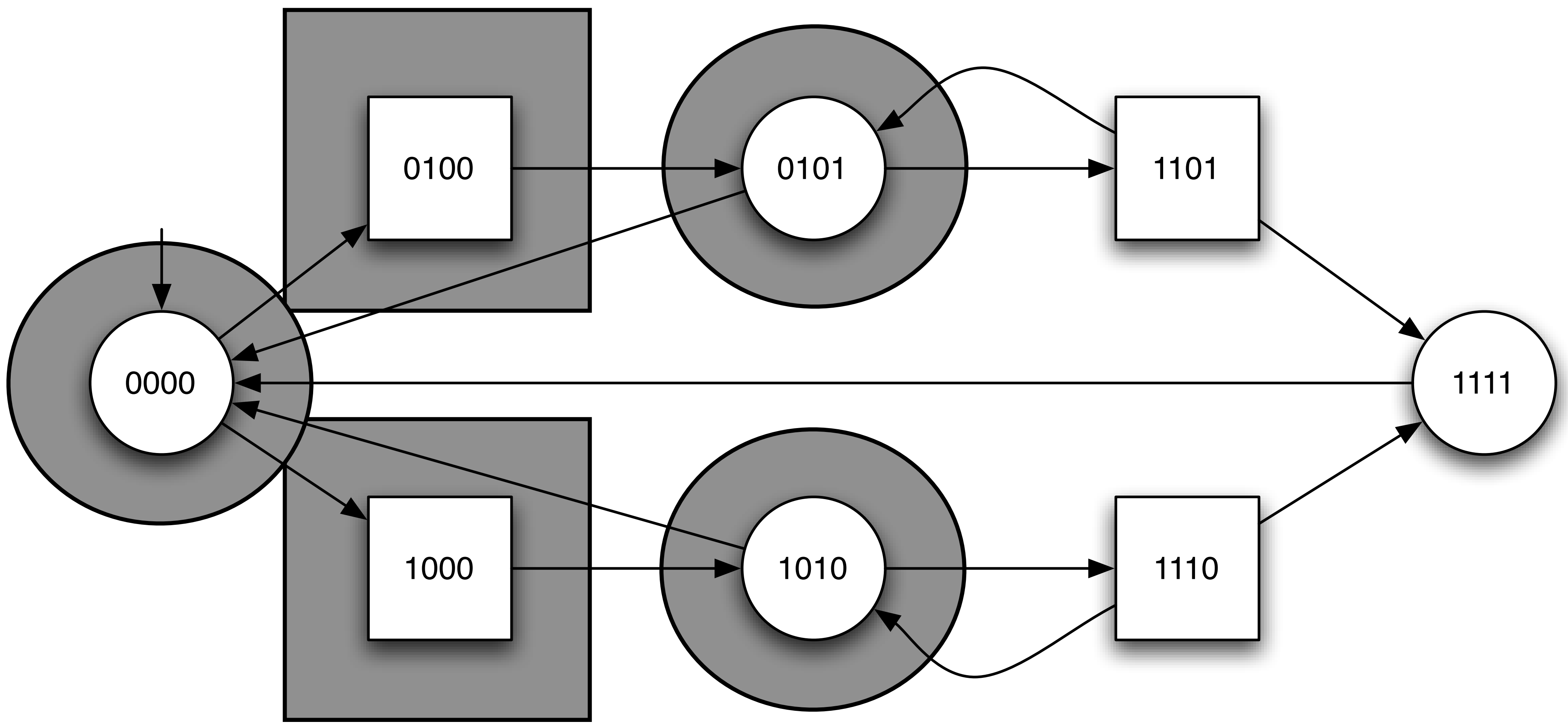


$$X_0 = (Q \setminus \{1111\}) \cap \mathsf{1CPre}(Q)$$

$$X_1 = (Q \setminus \{1111\}) \cap \mathsf{1CPre}(X_0)$$

$$X_2 = (Q \setminus \{1111\}) \cap \mathsf{1CPre}(X_1)$$

Fixpoint for a safety game



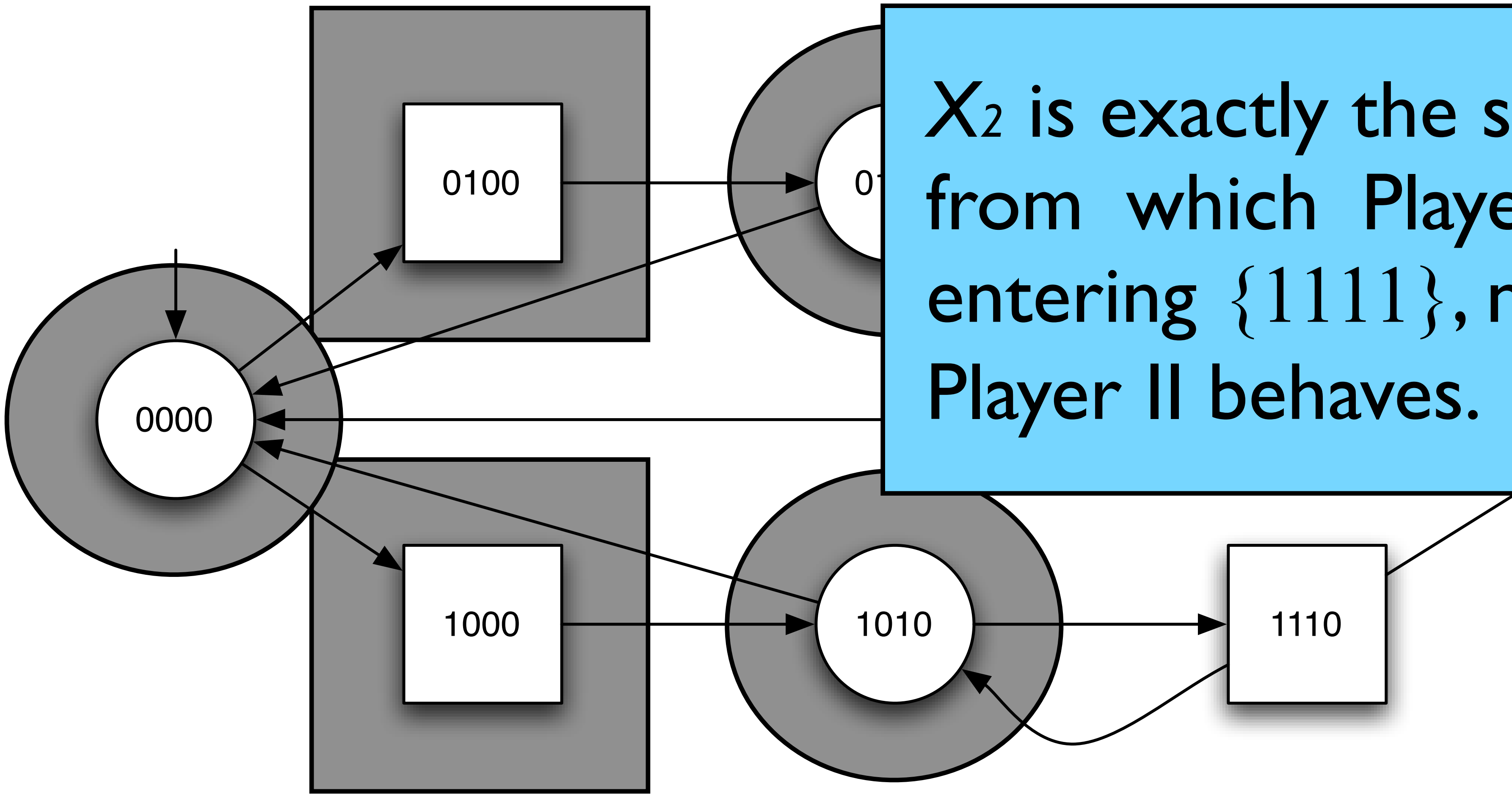
$$X_0 = (Q \setminus \{1111\}) \cap \mathsf{1CPre}(Q)$$

$$X_1 = (Q \setminus \{1111\}) \cap \mathsf{1CPre}(X_0)$$

$$X_2 = \boxed{(Q \setminus \{1111\}) \cap \mathsf{1CPre}(X_1)} = X_1$$

This is the greatest fixpoint

Fixpoint for a safety game



X_2 is exactly the set of positions from which Player I can avoid entering $\{1111\}$, no matter how Player II behaves.

This is the greatest fixpoint

$$X_0 = (Q \setminus \{1111\}) \cap \mathsf{1CPre}(Q)$$

$$X_1 = (Q \setminus \{1111\}) \cap \mathsf{1CPre}(X_0)$$

$$X_2 = (Q \setminus \{1111\}) \cap \mathsf{1CPre}(X_1) = X_1$$

Theorem

Let $G = \langle Q_1, Q_2, \iota, \delta \rangle$ be a TGS, let $\mathsf{Reach}(G, Q)$ be a reachability game defined on *G*, Player I has a winning strategy for this game iff

$$\iota \in \mu X \cdot Q \cup \mathsf{1CPre}(X)$$

Theorem

Let $G = \langle Q_1, Q_2, \iota, \delta \rangle$ be a TGS, let $\mathsf{Safe}(G, Q)$ be a safety game defined on *G*, Player I has a winning strategy for this game iff

$$\iota \in \nu X \cdot Q \cap \mathsf{1CPre}(X)$$

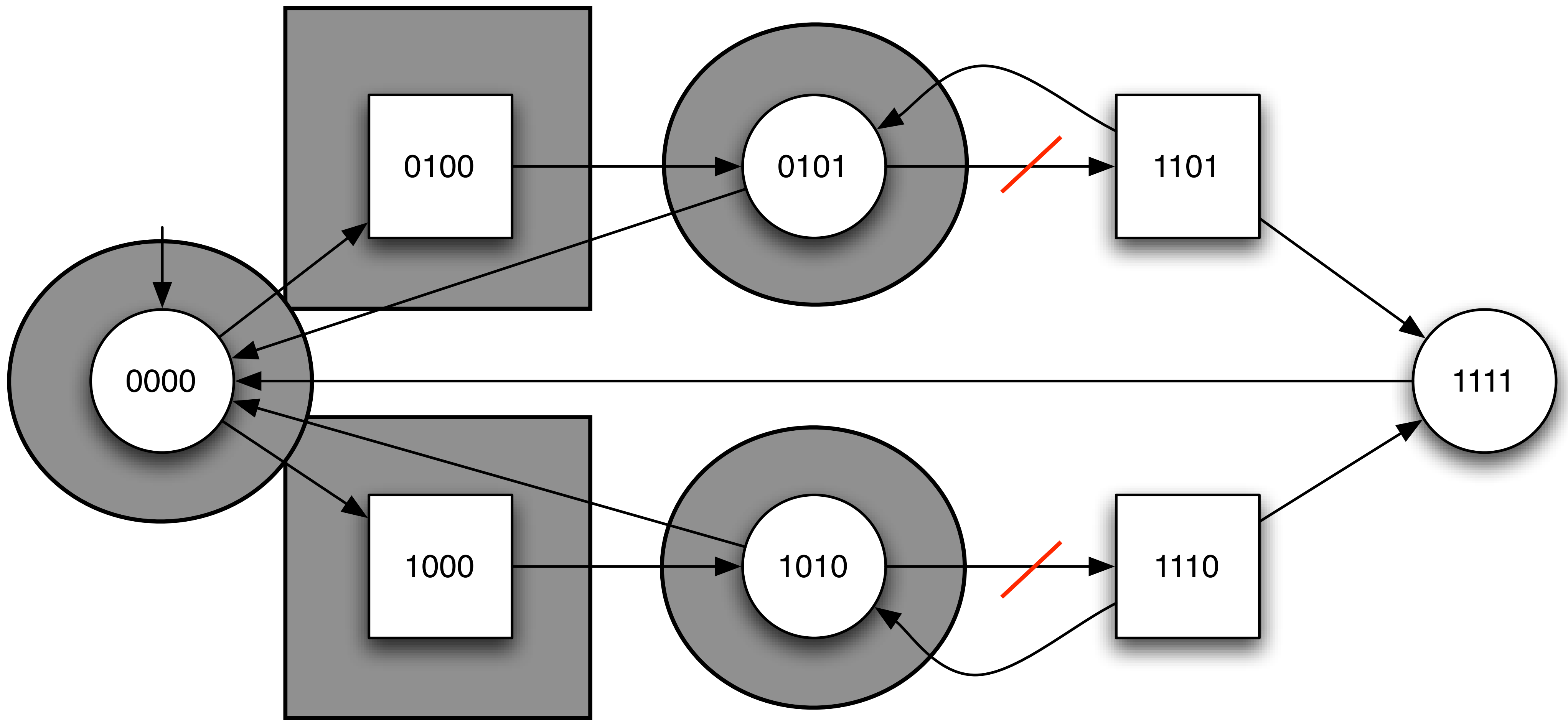
Some more results

Any finite state game with **regular objective** can be solved.

Some more results

Any finite state game with **regular objective** can be solved.

Strategies for safety and reachability games are **positional** (no need for memory).


0100
0101
1101
0000
1111
1000
1010
1110

Some more results

Any finite state game with **regular objective** can be solved.

Strategies for safety and reachability games are **positional** (no need for memory).

For more complicated games, like LTL games, finite **memory** is needed.

Some more results

Any finite state game with **regular objective** can be solved.

Strategies for safety and reachability games are **positional** (no need for memory).

For more complicated games, like LTL games, finite **memory** is needed.

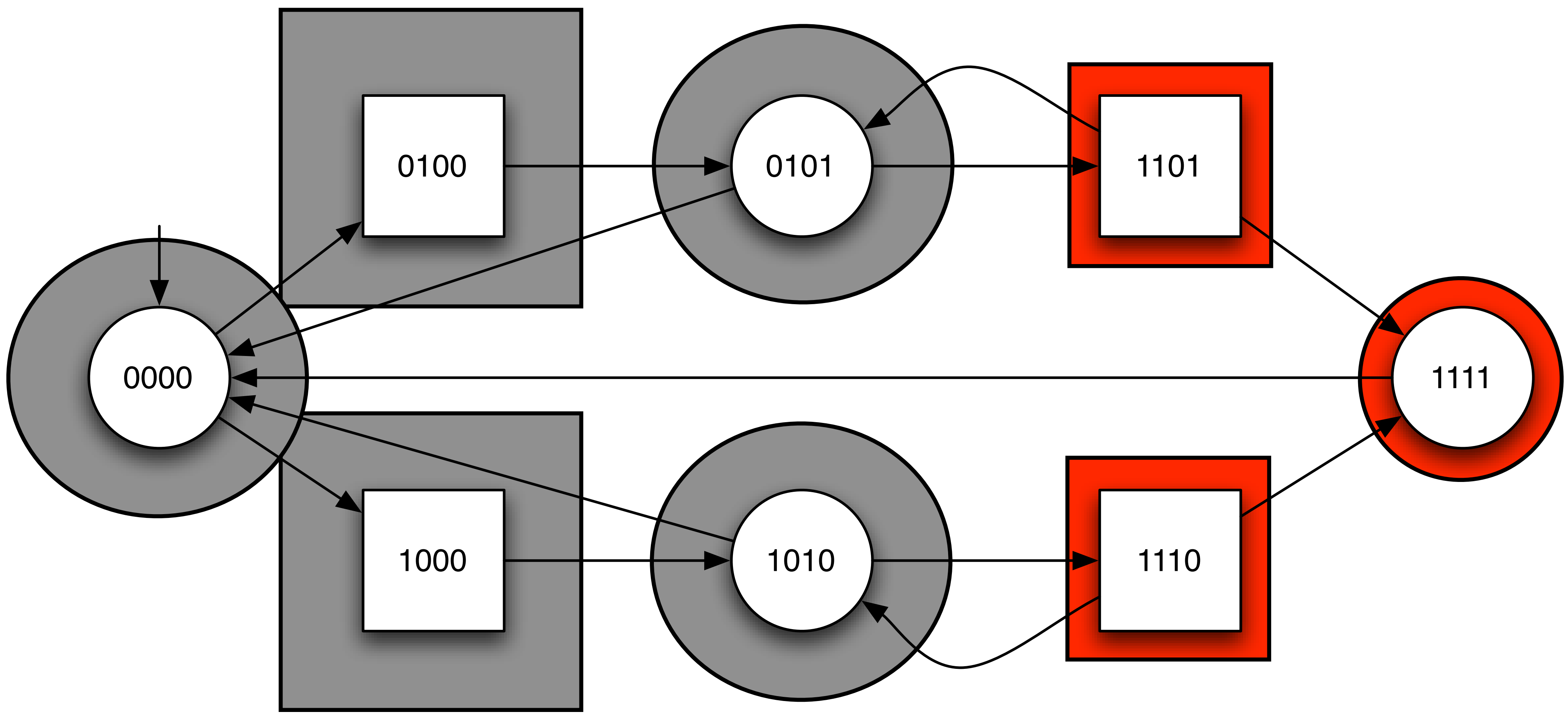
Determinacy theorem: In positional games (where a position is owned by a player), games are determinate in the following sense :

For any regular set of plays *W*,

Player I has a strategy to win (G, W)

iff

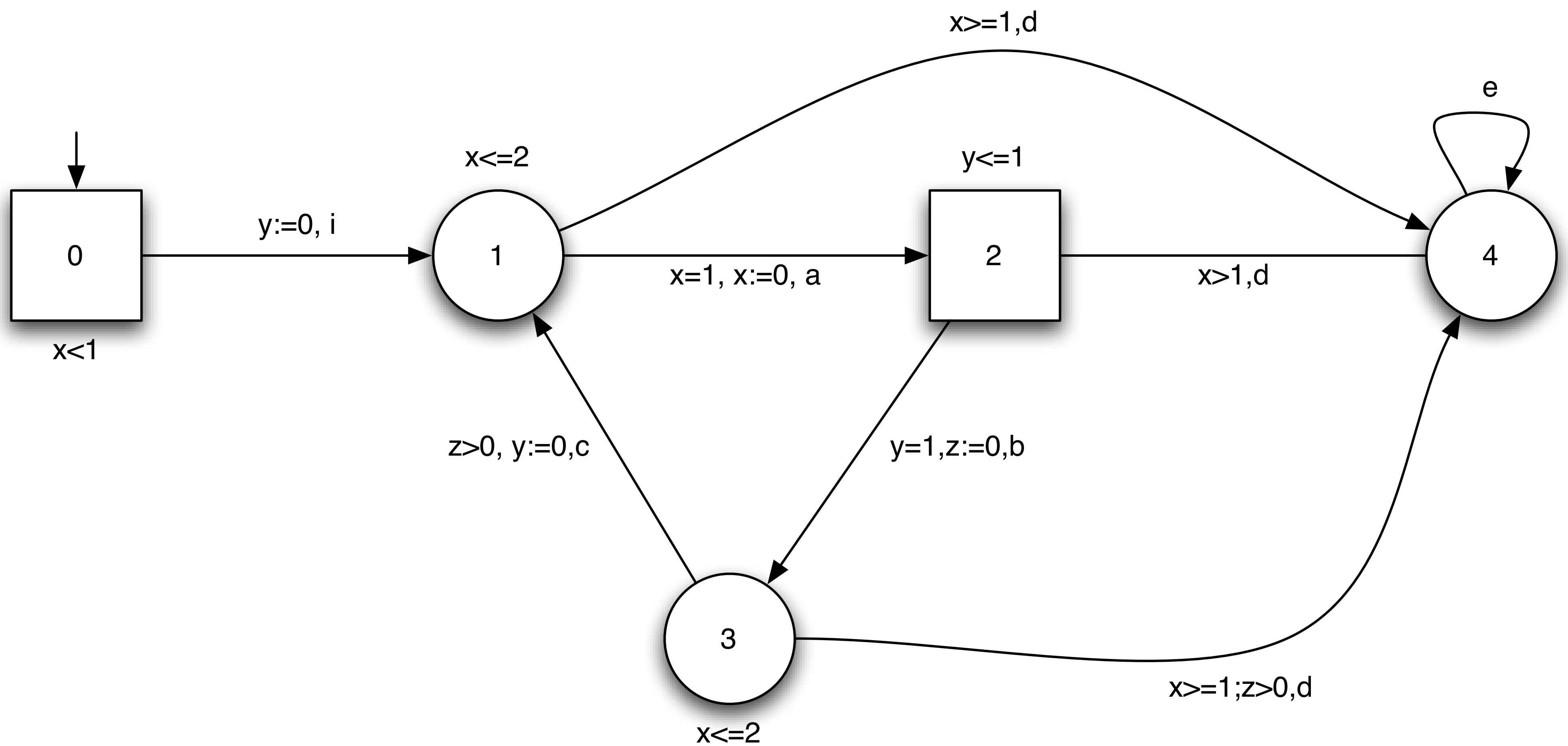
Player II does not have a strategy to win $(G, \overline{W})$



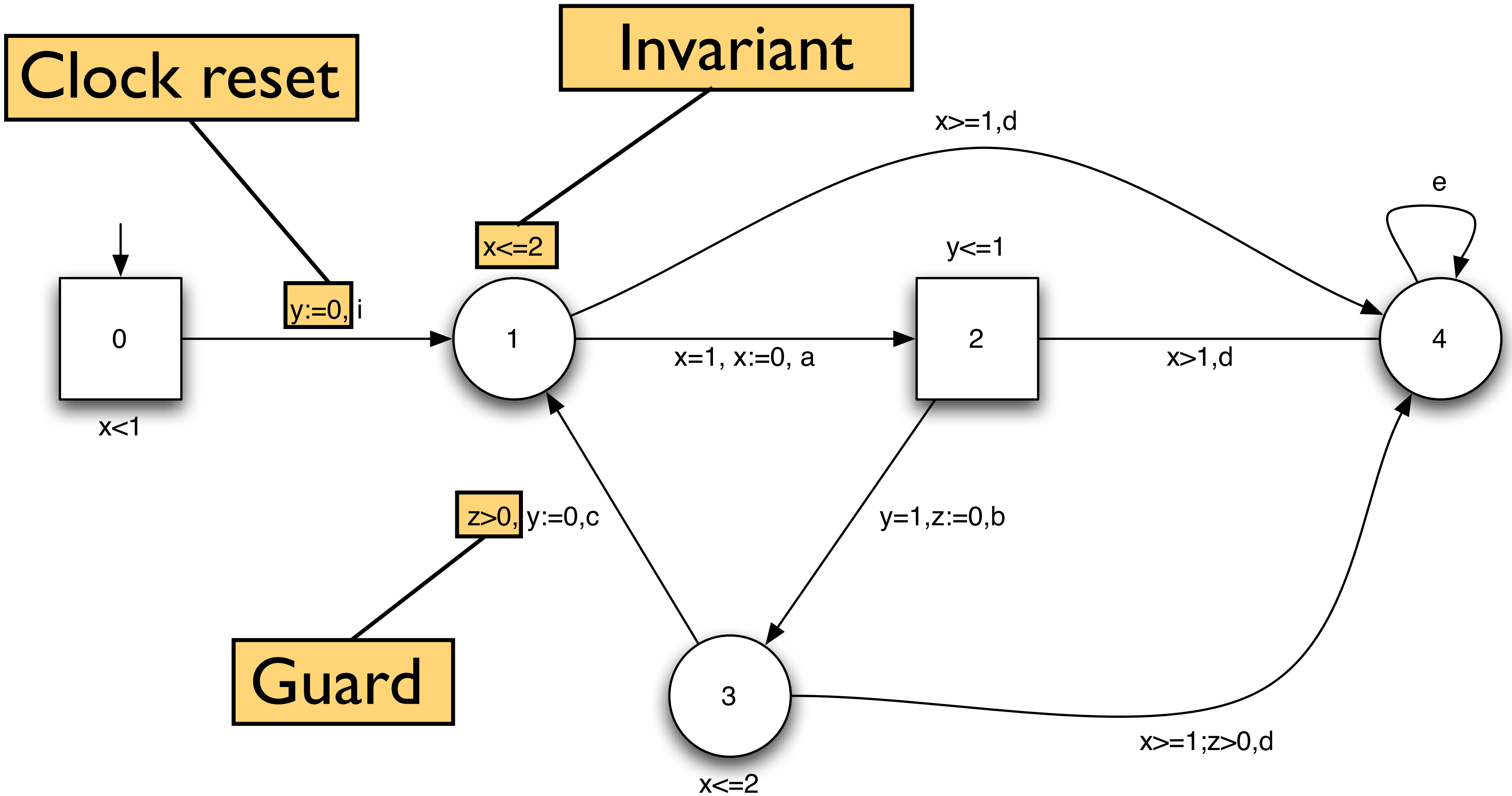
From the red states, and only from those states, Player II has a strategy to reach the state 1111

Timed Controller Synthesis

Timed Automata [AD94]



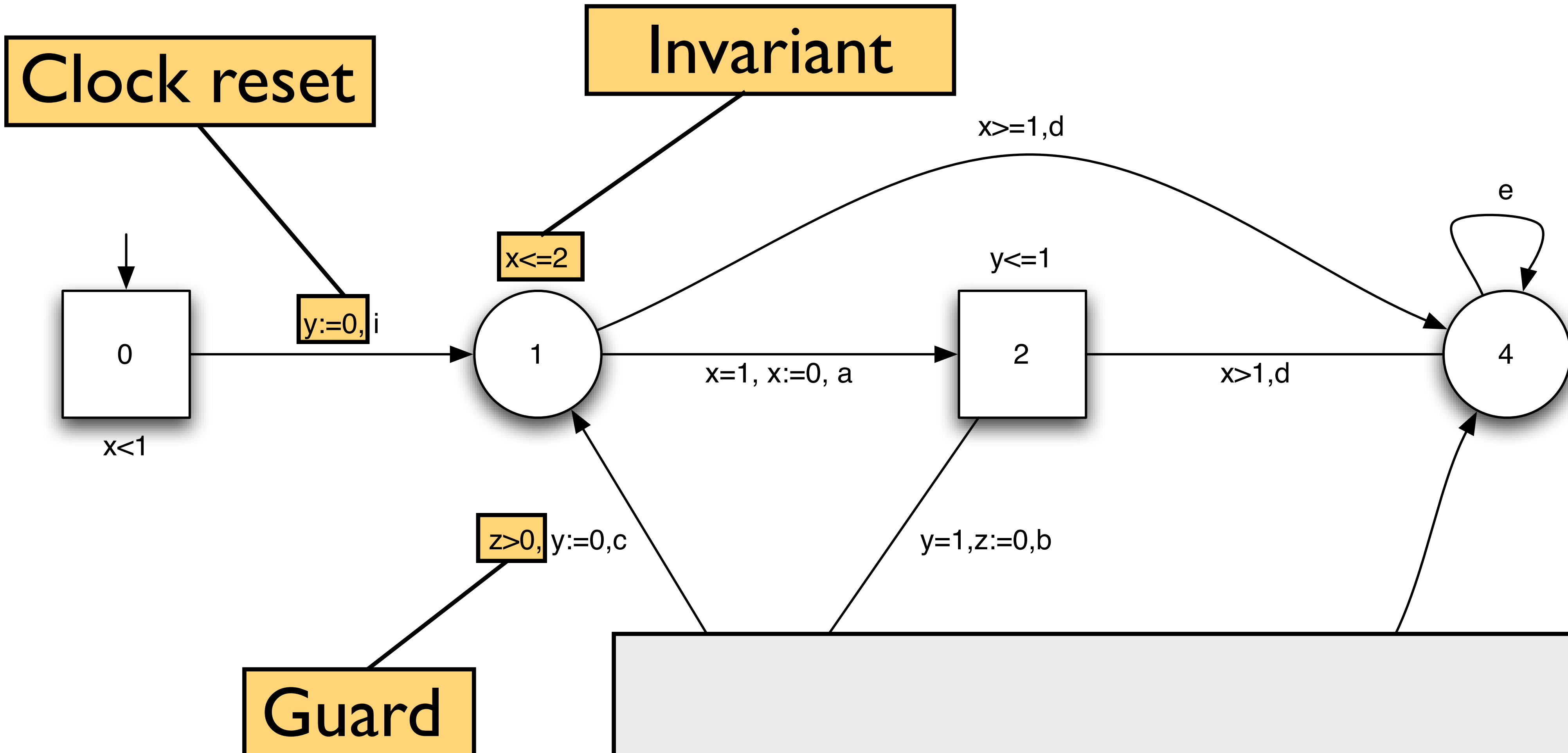
Timed Automata [AD94]



TA=Finite State Automata+Clocks

State of a TA: (*l*,*v*) where *l* is a location and *v* is a valuation of the clocks.

Timed Automata [AD94]

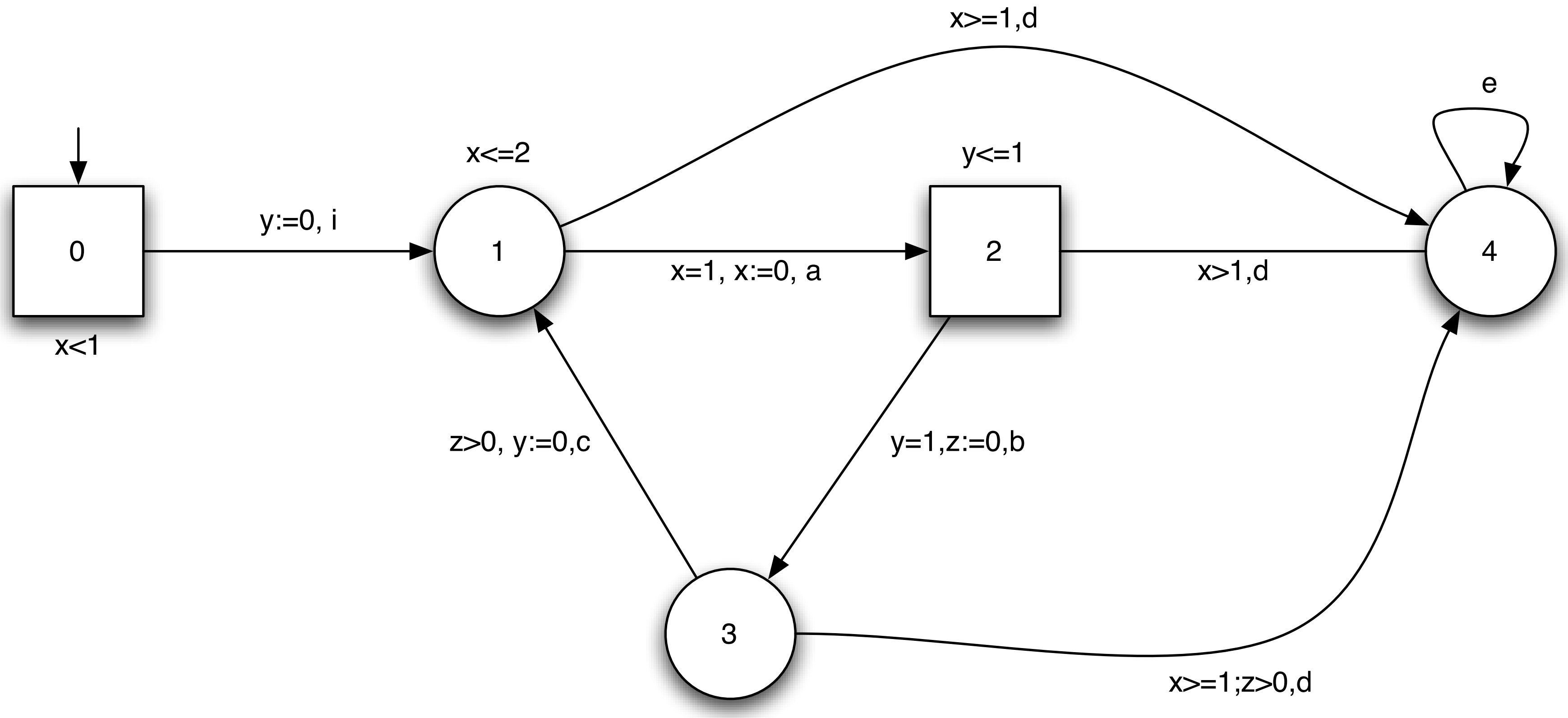


We need a game version

TA=Finite State Automata+Clocks

State of a TA: (*l*,*v*) where *l* is a location and *v* is a valuation of the clocks.

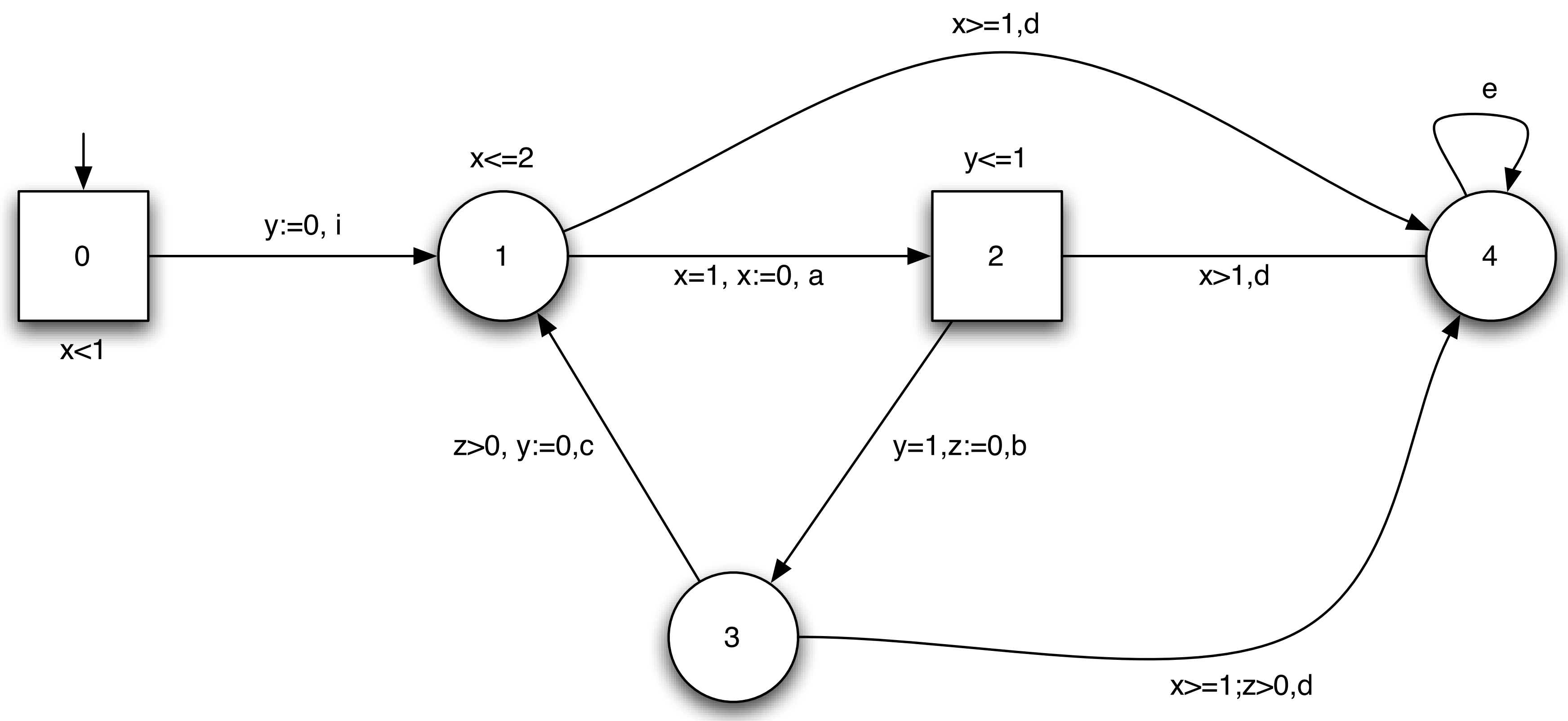
Simple Timed Game Automata



$\langle L_1, L_2, l_0, X, E, Inv \rangle$ where:

- L_1 and L_2 are locations where Player I, respectively Player II, makes choices.
- l_0 is the initial location.

Simple Timed Game Automata



$\langle L_1, L_2, l_0, X, E, Inv \rangle$ where:

- X is a finite set of clocks
- $E \subseteq L_1 \cup L_2 \times 2^X \times 2^{R^n} \times L_1 \cup L_2$, a set of edges
- $Inv : L_1 \cup L_2 \rightarrow 2^{R^n}$, the invariants labeling locations

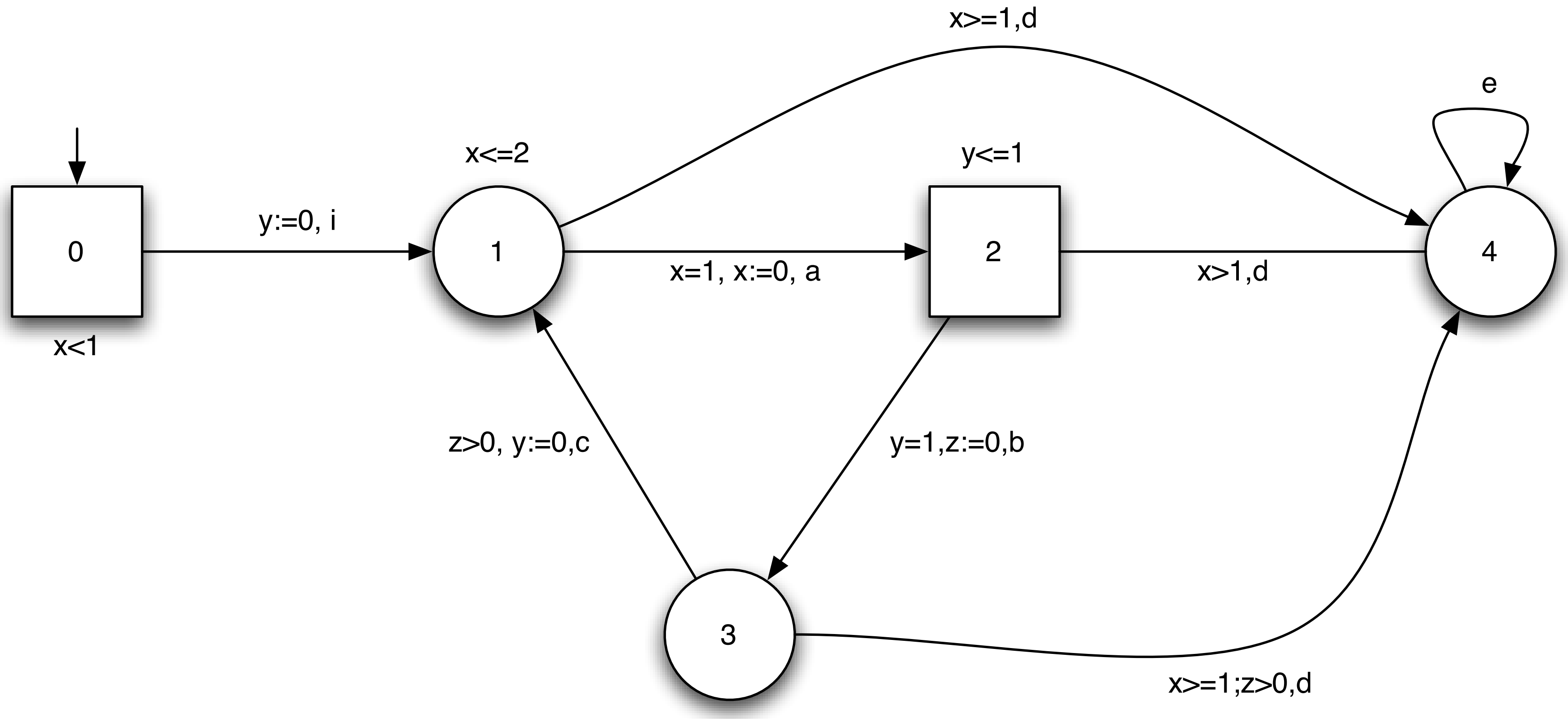
Simple Timed Games

As before, the positions of the games are **partitioned** into positions that belong to Player I and positions that belong to Player II.

Games on STGA are played as follows:

In a Player's *k* position, Player *k* proposes a **time *t*** and an **action a** to be played. This choice must be valid in the sense that it must not violate the **invariant** and the action a must be **enabled** after *t* time units. The game then proceeds to the next position.

Timed Play

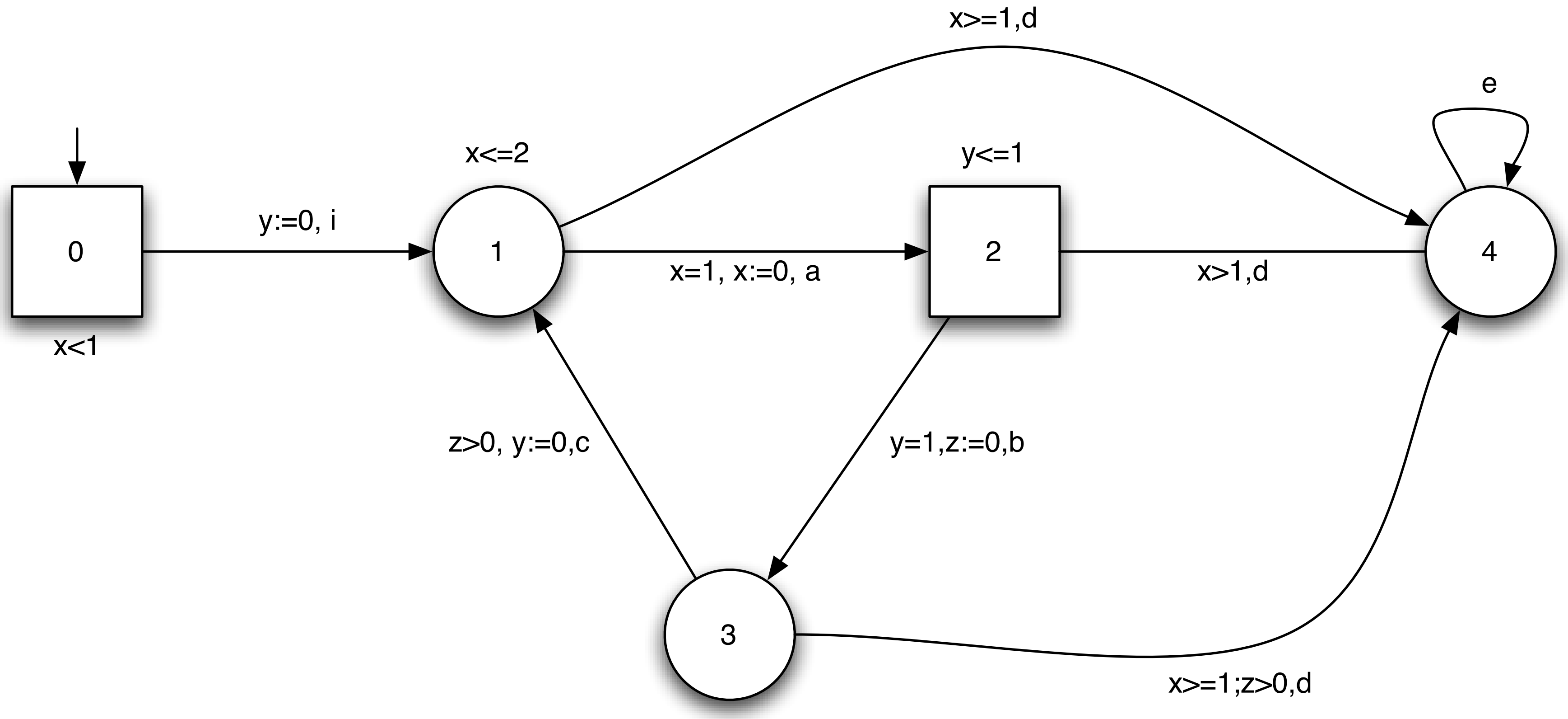


Timed Play :

$$(l_0, \langle 0,0,0 \rangle) \rightarrow_i^{0.5} (l_1, \langle 0.5, 0, 0.5 \rangle)$$

Player II chooses to **wait 0.5** and then to play ***i***

Timed Play



Timed Play :

$$(l_0, \langle 0,0,0 \rangle) \to_i^{0.5} (l_1, \langle 0.5, 0, 0.5 \rangle) \to_a^{0.5} (l_2, \langle 0, 0.5, 1 \rangle)$$

Player I chooses to **wait 0.5** and then to play ***a***

Timed Two-player Game Structure

A timed two-player game structure is a tuple

$G = \langle Q_1, Q_2, \iota, \delta_t \rangle$ where:

Q_1 and Q_2 are two disjoint sets of positions

$\iota \in Q_1 \cup Q_2$ is the initial position

$\delta_t \subseteq (Q_1 \cup Q_2) \times \mathbb{R} \times (Q_1 \cup Q_2)$ is the timed transition relation

We assume that $\forall q \in Q_1 \cup Q_2 : \exists t \in \mathbb{R} : \exists q' \in Q_1 \cup Q_2 : \delta_t(q, t, q')$

From STGA to TTGS

$$\langle L_1, L_2, l_0, X, E, Inv \rangle \longrightarrow G = \langle Q_1, Q_2, \iota, \delta_t \rangle$$

$$Q_1 = \{(l, v) \mid l \in L_1 \wedge v \models Inv(l)\}$$

$$Q_2 = \{(l, v) \mid l \in L_2 \wedge v \models Inv(l)\}$$

$$\iota = (l_0, 0^{|X|})$$

$$\delta((l, v), t, (l', v')) \quad \textbf{iff} \quad \exists \langle l, r, g, l' \rangle \in E :$$

$$\forall t' : 0 \leq t' \leq t : v + t \models Inv(l) \quad \wedge \quad v + t \models g \quad \wedge \quad v' = v + t[r := 0]$$

Timed Play

Let $G = \langle Q_1, Q_2, \iota, \delta_t \rangle$,

$$w = q_0 \rightarrow^{t_0} q_1 \rightarrow^{t_1} q_2 \dots q_n \rightarrow^{t_n} \dots$$

is a **timed play** in *G* if

1) $w(0) = \iota$
2) $\forall i \geq 0 : \delta_t(\ w(i)(q),\ w(i)(t),\ w(i+1)(q)\ )$

The set of timed plays of *G* is noted $\mathsf{Plays}(G)$

$$\mathsf{PrefPlays}(G) = \{q_0 \rightarrow^{t_0} \dots \rightarrow^{t_{n-1}} q_n \mid \exists w \in \mathsf{Plays}(G) \wedge \forall 0 \leq i \leq n : w(i)(q) = q_i \wedge w(i)(t) = t_i\}$$

$$\mathsf{PrefPlays}_k(G) = \{w \in \mathsf{PrefPlays}(G) \wedge \mathit{last}(w) \in Q_k\}$$

Timed Strategy

Players are playing according to **timed strategies**.

A Player *k* strategy in *G* is a function:

$$\lambda : \mathsf{PrefPlays}_k(G) \rightarrow \mathbb{R} \times Q_1 \cup Q_2$$

with the restriction that:

$$\forall w \in \mathsf{PrefPlays}_k(G) : \delta(last(w), \lambda(w)(t), \lambda(w)(q))$$

Outcome of a timed strategy

$$w = q_0 \rightarrow^{t_0} q_1 \rightarrow^{t_1} q_2 \ldots q_n \rightarrow^{t_n} \ldots$$

is a possible **outcome** of the Player *k* timed strategy λ if

$$\forall i \geq 0 : q_i \in Q_k \rightarrow t_i = \lambda(w(0,i))(t) \wedge q_{i+1} = \lambda(w(0,i))(q)$$

The set of timed plays that have this property is denoted

$$\text{Outcome}_k(G, \lambda)$$

Symbolic algorithms to solve timed games

Player k timed controllable predecessors

$$\mathsf{1CPre}_G(X) = \{q \in Q_1 \mid \exists t \in \mathbb{R}, q' : \delta_t(q,t,q') \wedge q' \in X\} \cup \{q \in Q_2 \mid \forall t \in \mathbb{R}, q' : \delta_t(q,t,q') \to q' \in X\}$$

Set of Player I positions where he has a choice of successor that lies in *X*

Set of Player II positions where all her choices for successors lie in *X*

Player *k* timed controllable predecessors

$$\mathsf{1CPre}_G(X) = \{q \in Q_1 \mid \exists t \in \mathbb{R}, q' : \delta_t(q,t,q') \wedge q' \in X\} \cup \{q \in Q_2 \mid \forall t \in \mathbb{R}, q' : \delta_t(q,t,q') \rightarrow q' \in X\}$$

Symmetrically

$$\mathsf{2CPre}_G(X) = \{q \in Q_2 \mid \exists t \in \mathbb{R}, q' : \delta_t(q,t,q') \wedge q' \in X\} \cup \{q \in Q_1 \mid \forall t \in \mathbb{R}, q' : \delta_t(q,t,q') \rightarrow q' \in X\}$$

Player *k* timed controllable predecessors

$$\mathsf{1CPre}_G(X) = \{q \in Q_1 \mid \exists t \in \mathbb{R}, q' : \delta_t(q,t,q') \wedge q' \in X\} \cup \{q \in Q_2 \mid \forall t \in \mathbb{R}, q' : \delta_t(q,t,q') \rightarrow q' \in X\}$$

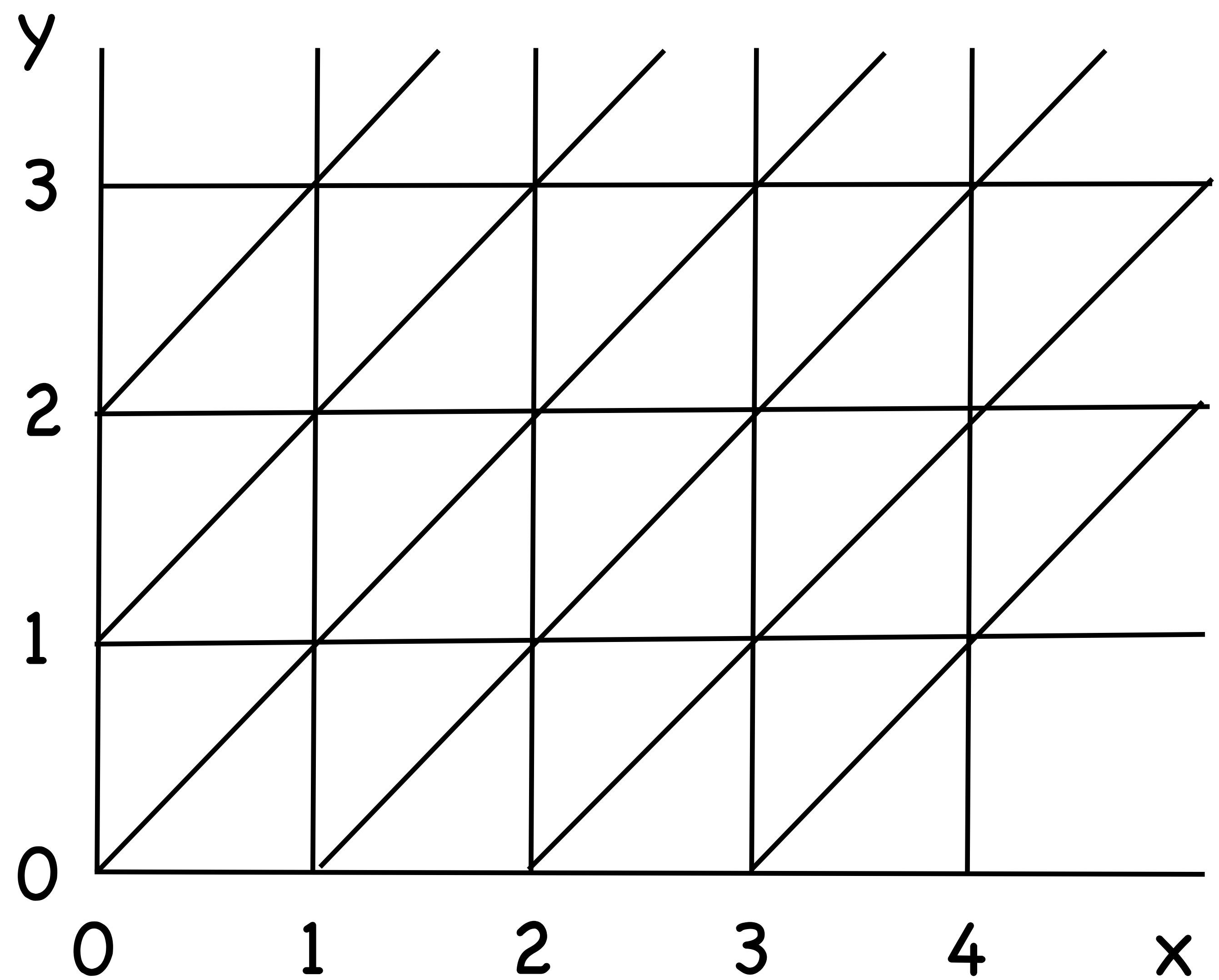
Symmetrically

$$\mathsf{2CPre}_G(X) = \{q \in Q_2 \mid \exists t \in \mathbb{R}, q' : \delta_t(q,t,q') \wedge q' \in X\} \cup \{q \in Q_1 \mid \forall t \in \mathbb{R}, q' : \delta_t(q,t,q') \rightarrow q' \in X\}$$

Difficulty : here *X* ranges over the subsets of an infinite set

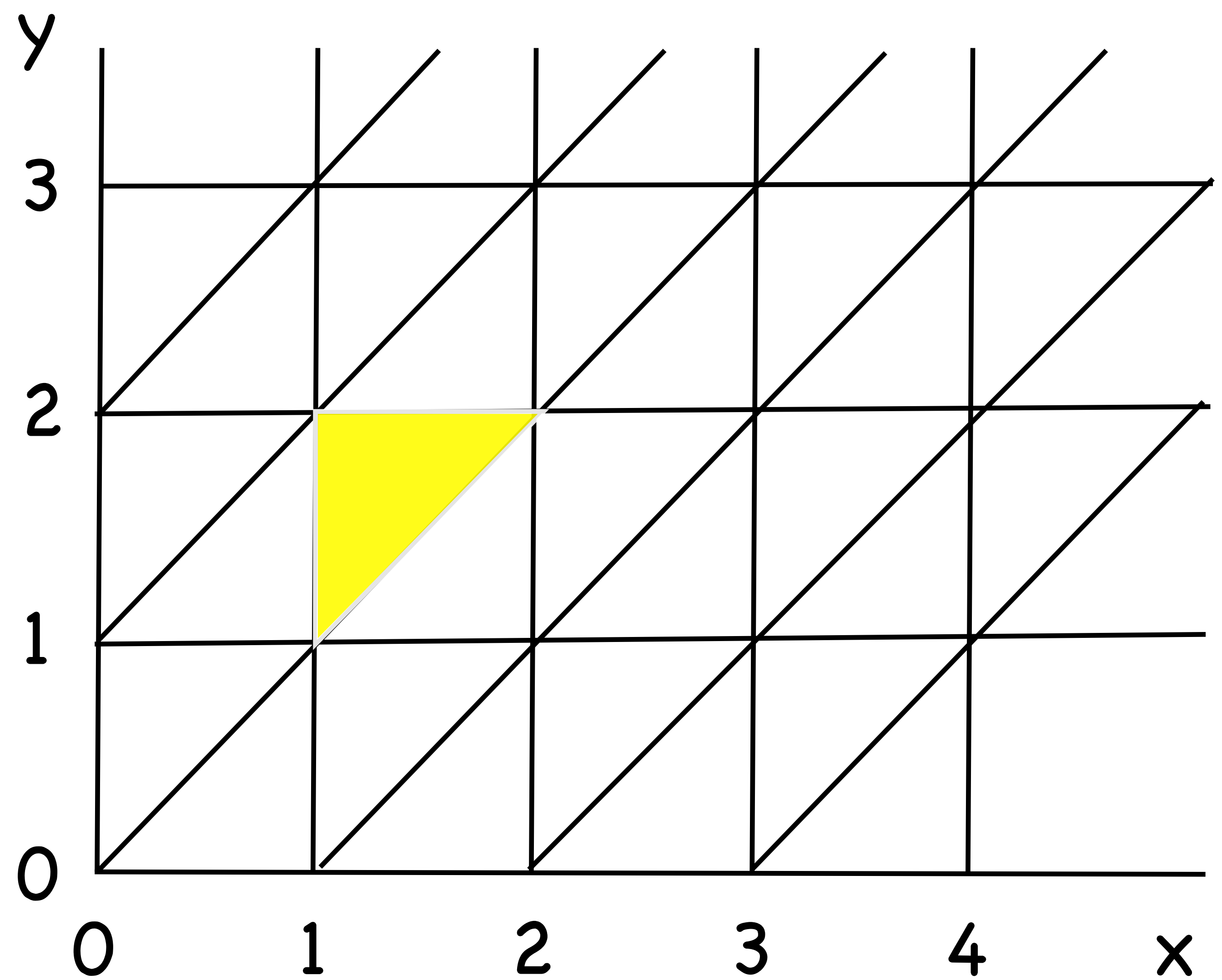
Region equivalence

Region equivalence



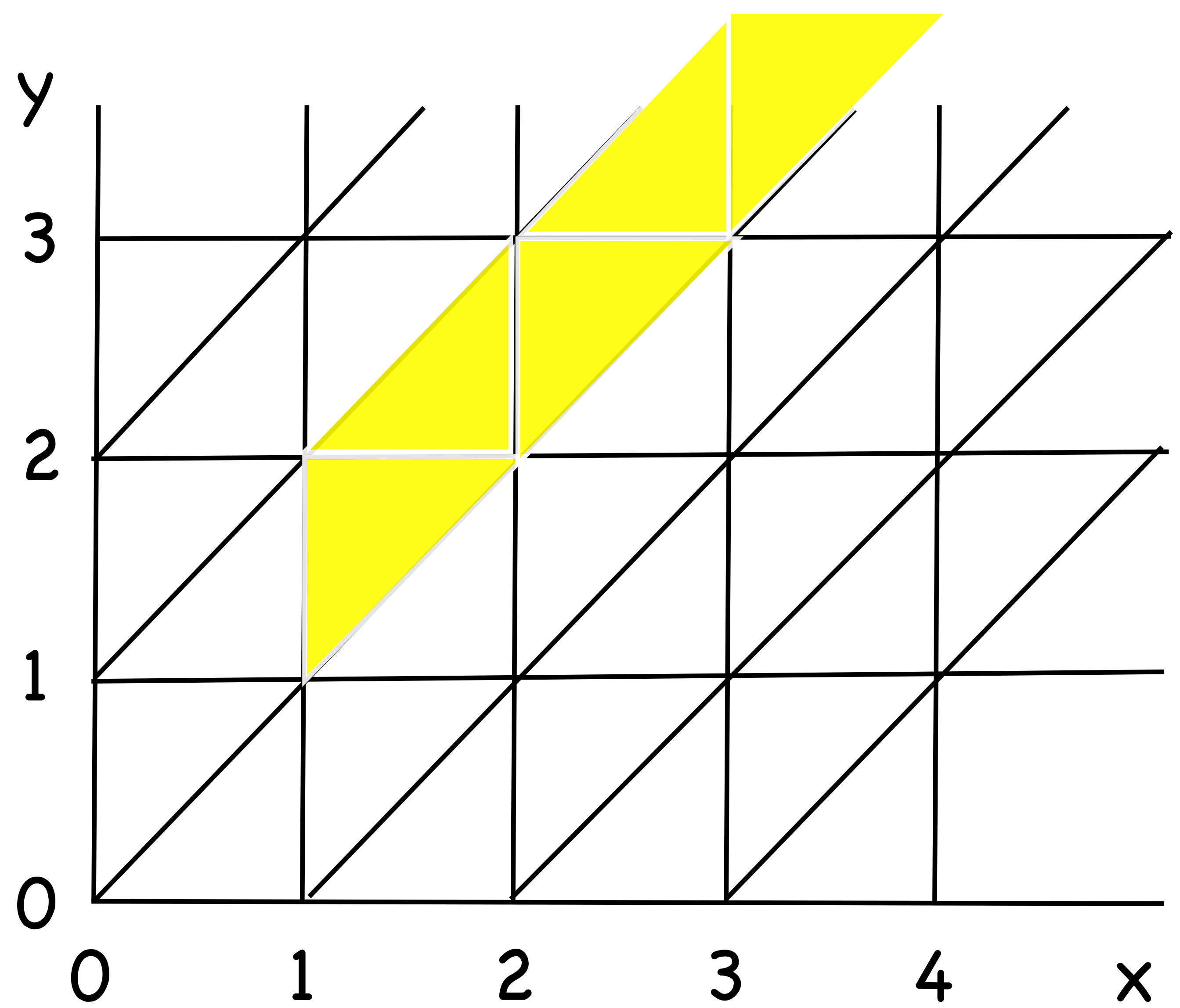
Finite number of equivalence classes

Region equivalence



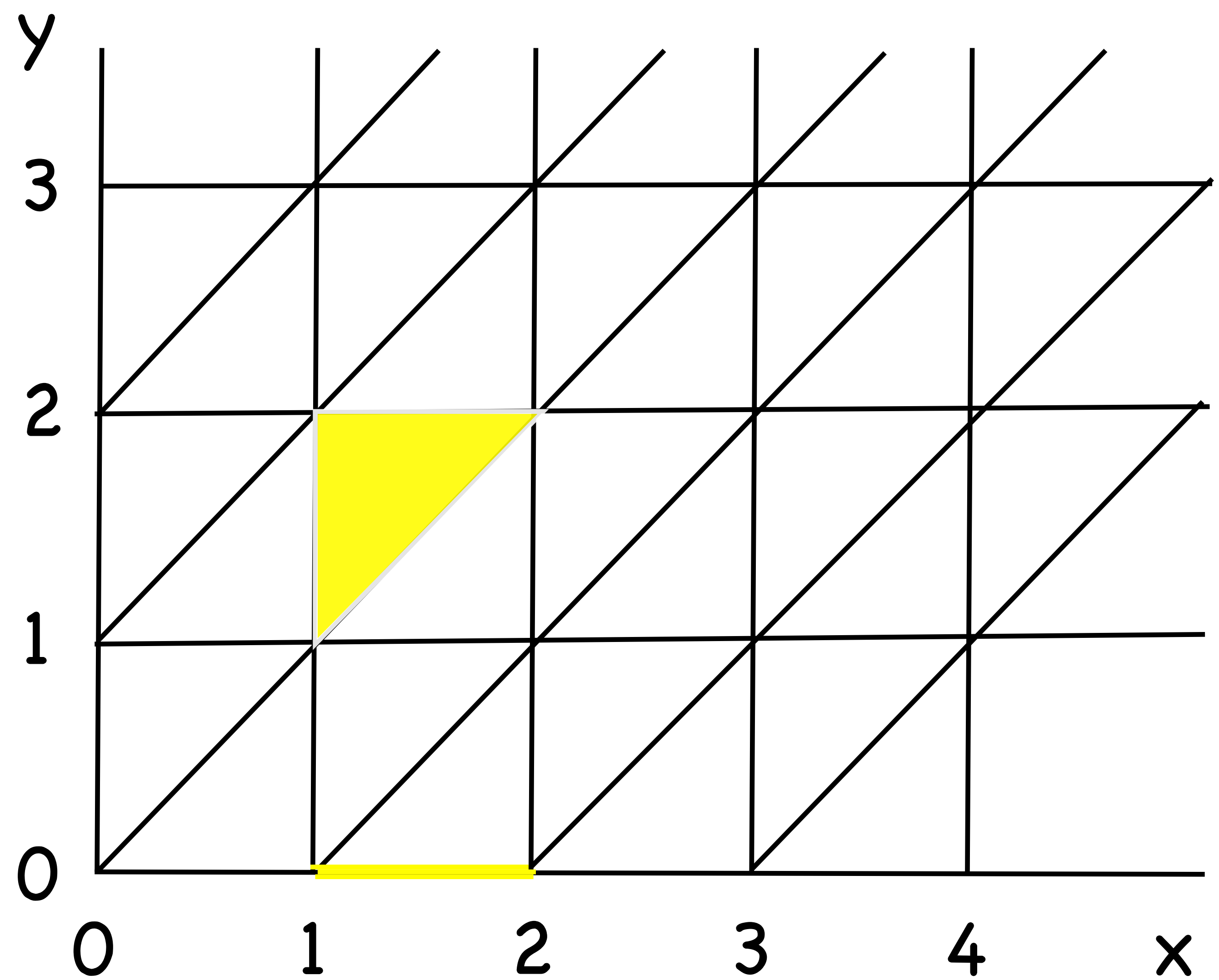
All valuations of a region satisfies the same guards and invariants

Region equivalence



Time elapsing and time predecessors preserve regions

Region equivalence



Reset and inverse reset operations preserve regions

1CPre preserves regions

Theorem. If X is a union of regions then 1CPre(X) is a union of regions.

Corollary. Safety, Reachability and more generally LTL games are decidable on timed game structures generated by timed automata.

Zenoness

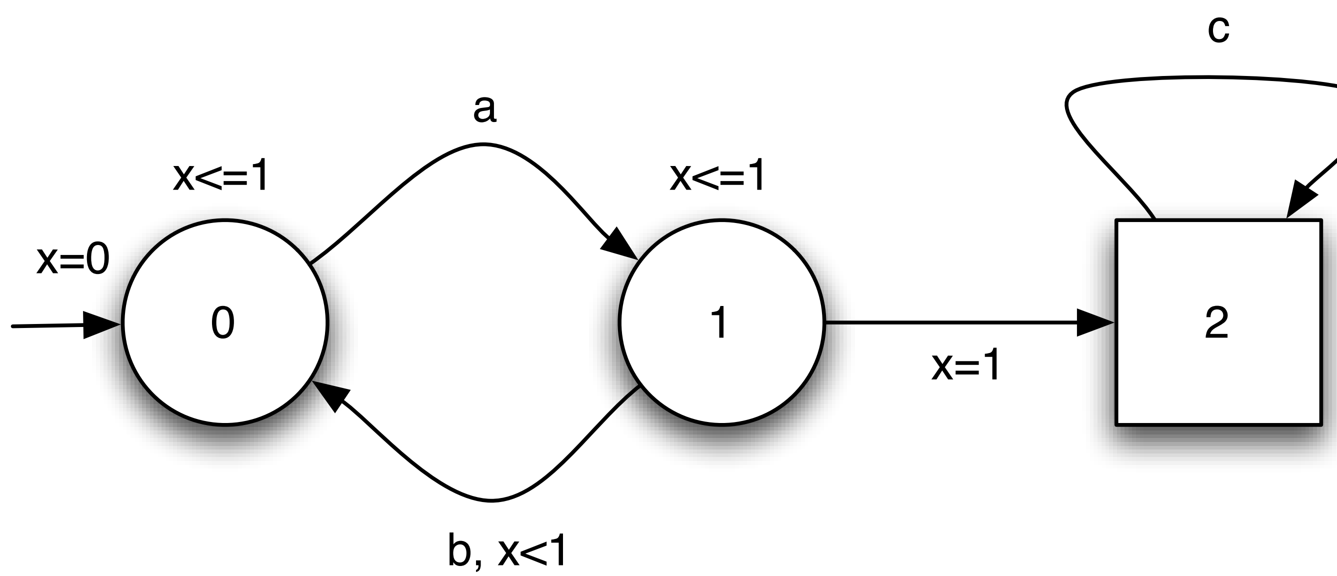
Not all timed strategies are reasonable

A timed play $w = q_0 \rightarrow^{t_0} q_1 \rightarrow^{t_1} q_2 \ldots q_n \rightarrow^{t_n} \ldots$

is **Zeno** if: $\exists t \in \mathbb{R} : \sum_{i=0}^{\infty} t_i \leq t$

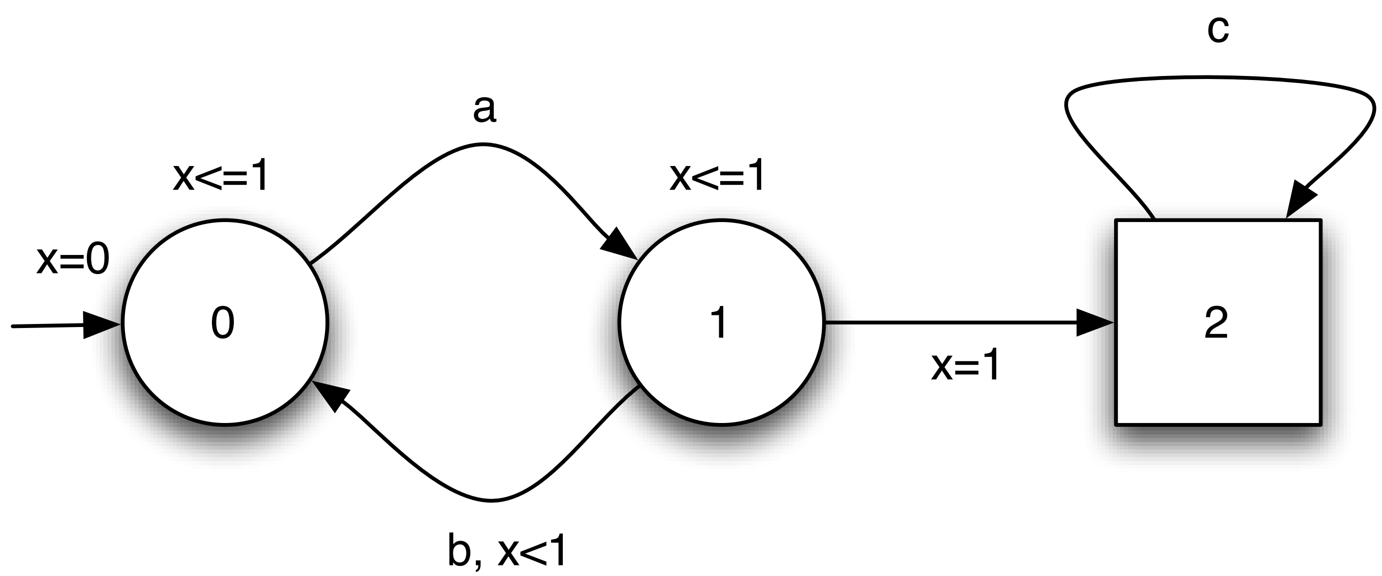
Time does not diverge

Not all timed strategies are reasonable



Does Player I have a timed strategy to avoid entering location l_2 ?

Not all timed strategies are reasonable



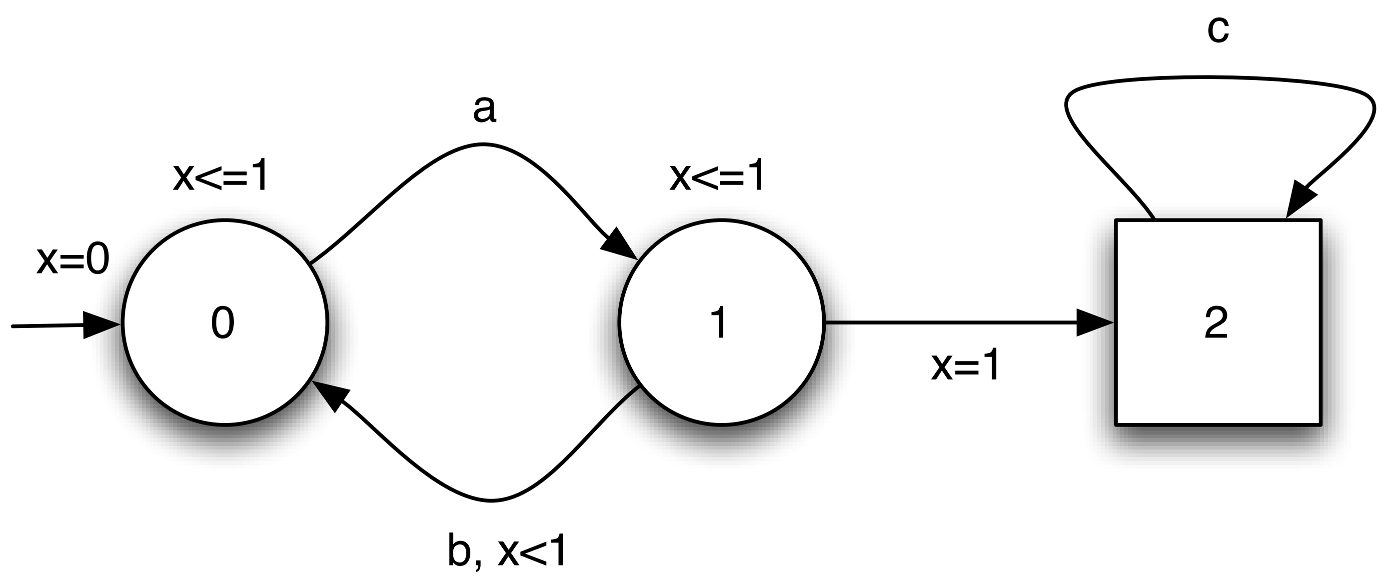
Consider the following timed strategy for Player I:

Let $w \in \mathsf{PrefPlay}_1(G)$:

if $last(w) = (l_0, v)$ then let $t = 1 - \frac{1 - v(x)}{2}$ and $\lambda(w) = (t, (l_1, v(x) + t))$

if $last(w) = (l_1, v)$ then let $t = 1 - \frac{1 - v(x)}{2}$ and $\lambda(w) = (t, (l_0, v(x) + t))$

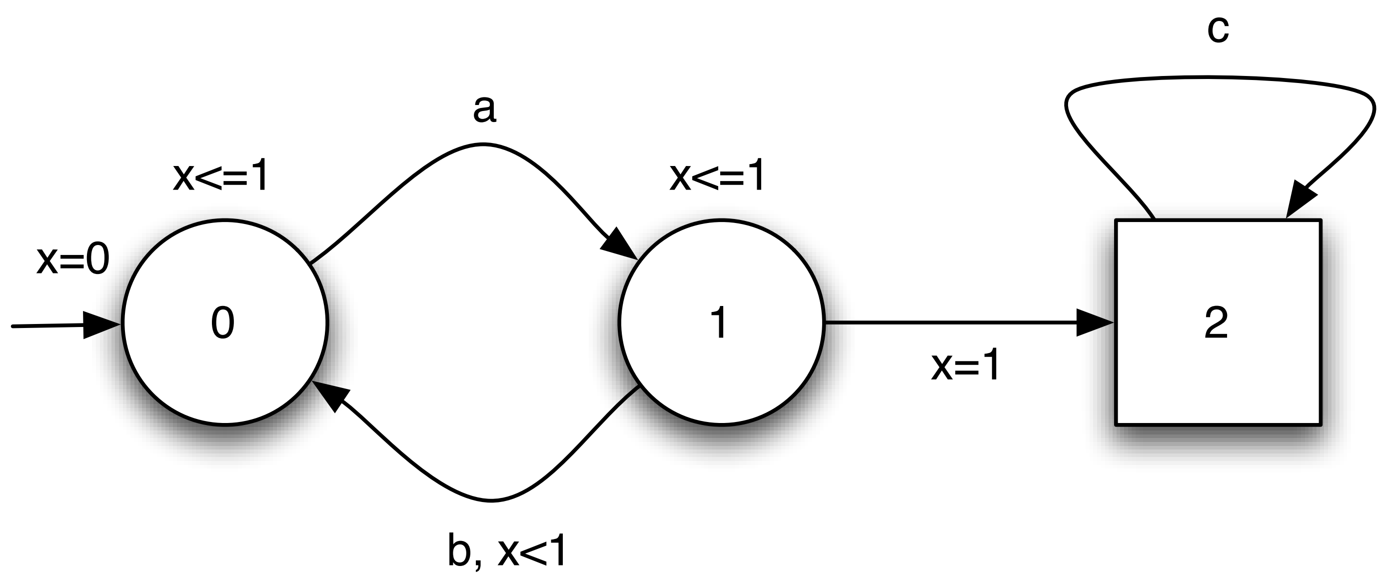
Not all timed strategies are reasonable



When Player I plays this strategy, the only outcome of the games is:

$$(l_0, 0) \rightarrow^{\frac{1}{2}} (l_1, \frac{1}{2}) \rightarrow^{\frac{1}{4}} (l_0, \frac{3}{4}) \rightarrow^{\frac{1}{8}} (l_1, \frac{7}{8}) \ldots$$

Not all timed strategies are reasonable



When Player I plays this strategy, the only outcome of the games is:

$(l_0, 0)$ Clearly, such a strategy can not be implemented $\frac{7}{8})\ldots$

Not all timed strategies are reasonable

They are algorithmic solutions to avoid the synthesis of **zeno strategies**. The correctness of those solutions can be explained using the region graph.

Not all timed strategies are reasonable

They are algorithmic solutions to avoid the synthesis of **zeno strategies**. The correctness of those solutions can be explained using the region graph.

But Zenoness is not the only problem

Implementability issues for timed models

Model-based Development

- Make a model of the environment
  Environment
- Make clear the control objective:
  Bad
- Make a model of your control strategy:
  ControllerMod
- Verify :
  Does Environment || ControllerMod avoid Bad ?
- Good, but after ?

From Correct Models to Correct Implementations

- Should we verify code ?
  - this may be difficult (too much details)
- Can we translate model into code ?

  ... there are tools for that ...
- ... and preserve properties ?

  ... good question...

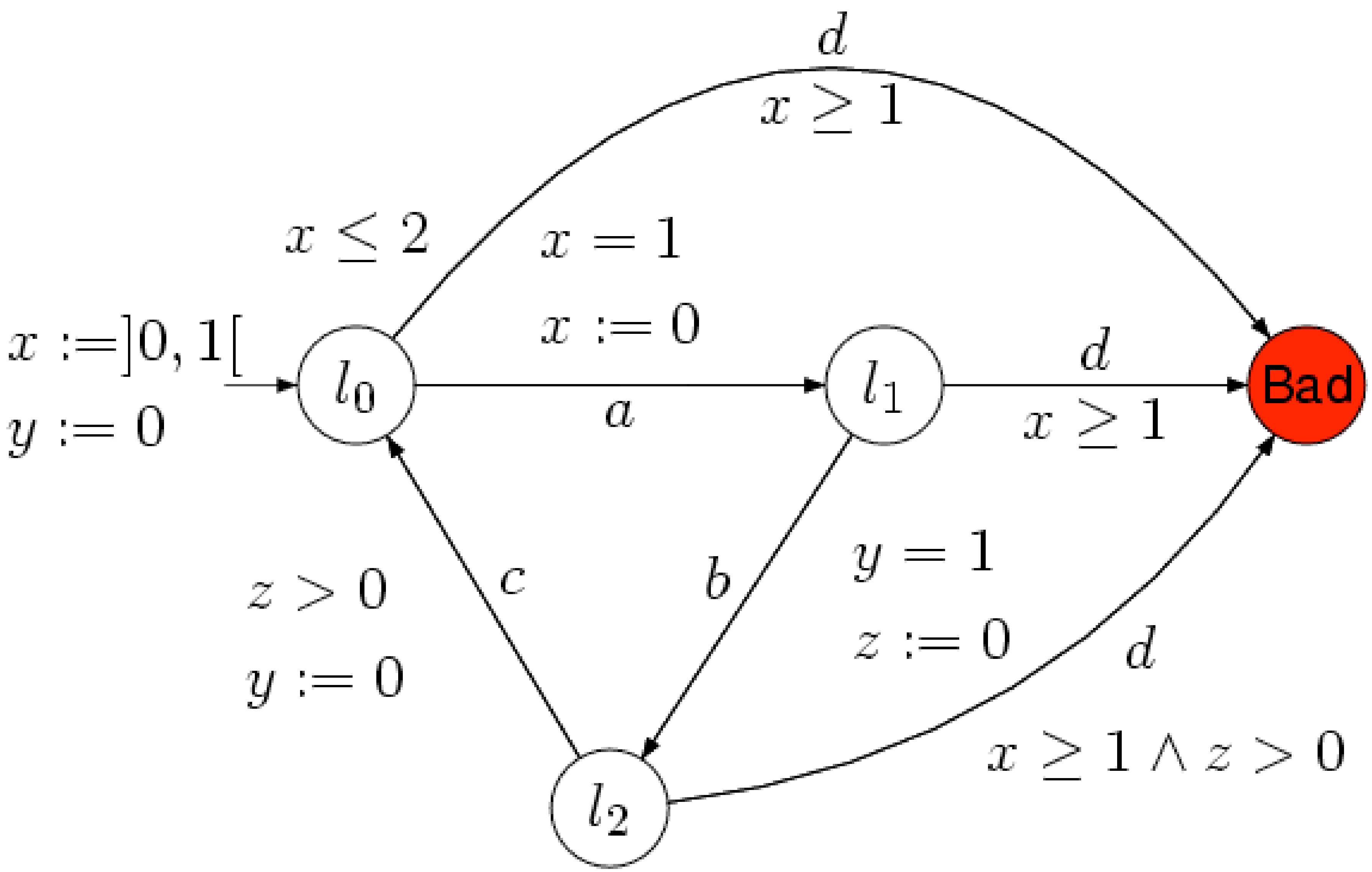
Problem

- Timed automata are (in general) not implementable (in a formal sense)...

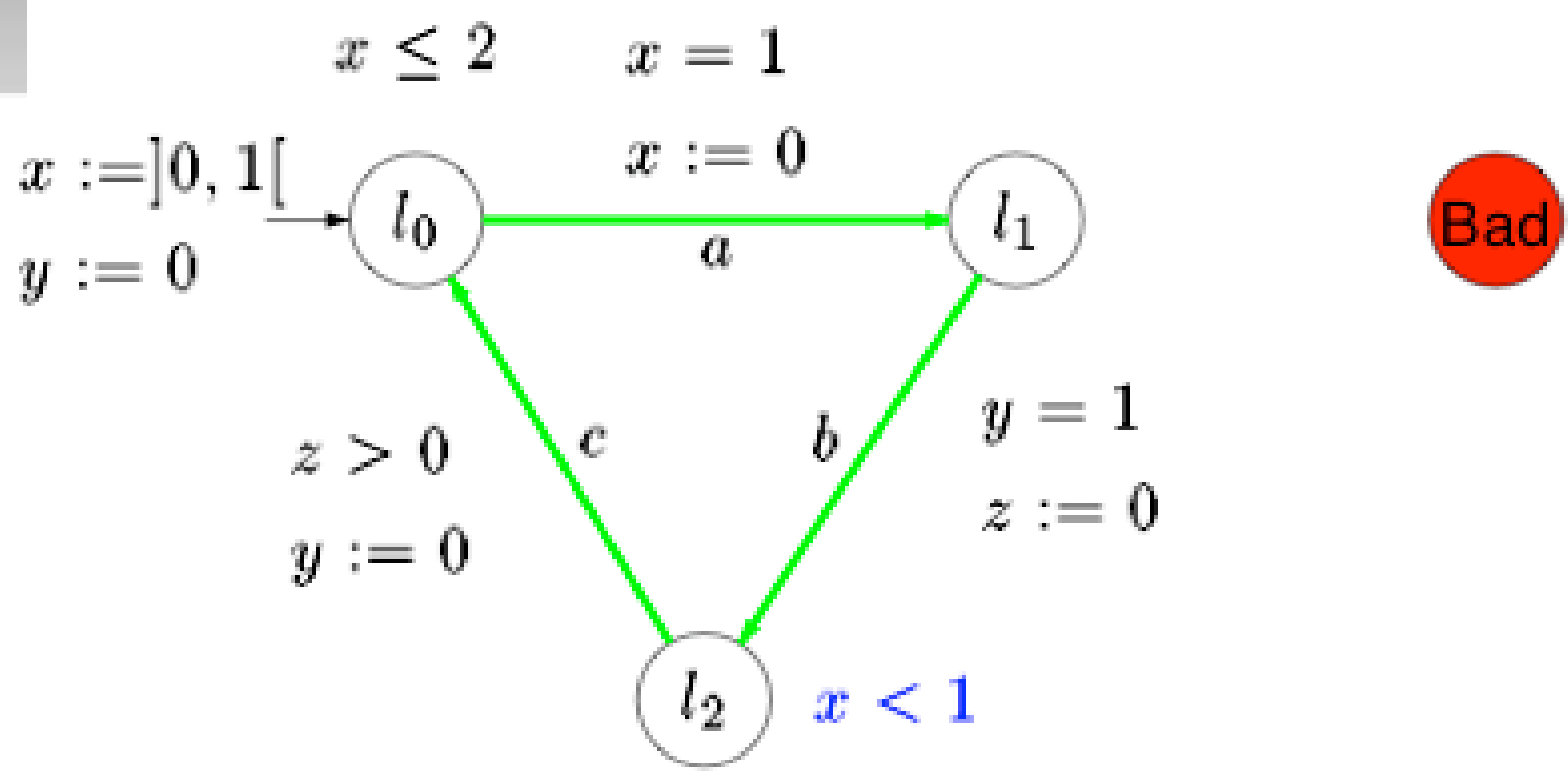
  Why ?

  - Zenoness : 0, 0.5, 0.75, 0.875, ...
  - No minimal bound between two transitions :
    0,0.5,1,1.75,2,2.875,3,...
  - And more ... (robustness)

No Minimal Bound between Two Transitions



It can be controlled

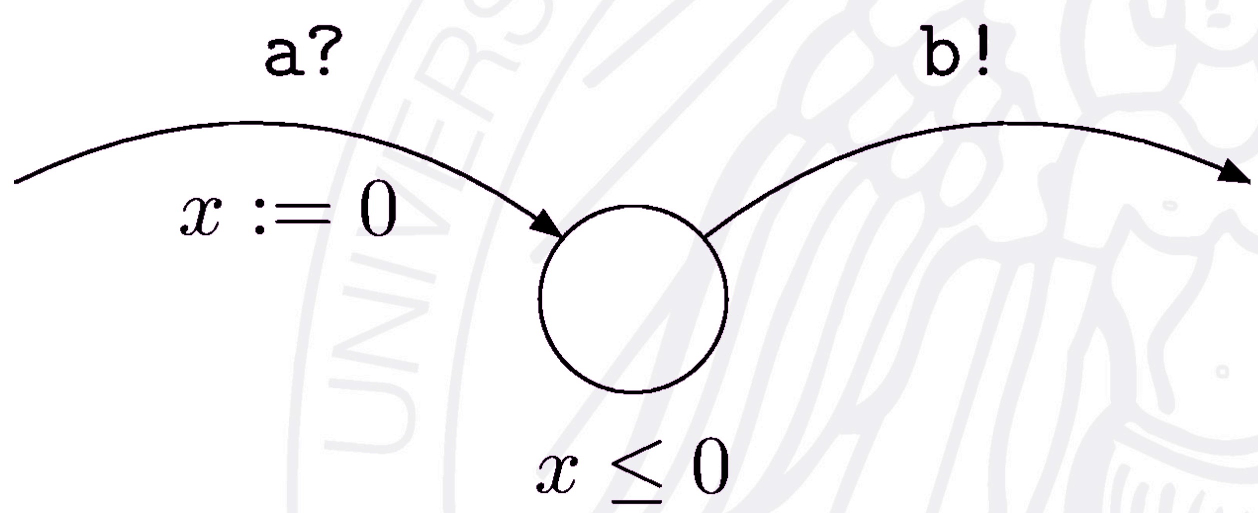


- δ_i : time in l_2 during loop i
- the controller must ensure : $\sum_{i=0}^{i=+\infty} \delta_i < x_0 - y_0$

More...

- One can specify instantaneous responses but not implement them.

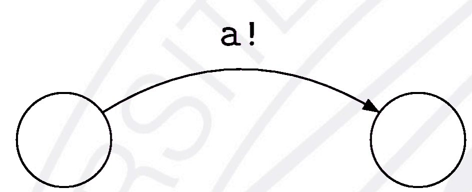
Not implementable



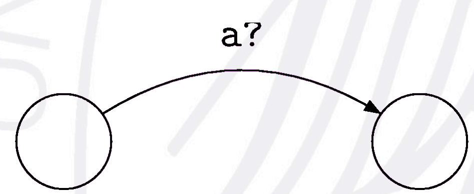
More...

- Instantaneous synchronisations between environment and controller are not implementable.

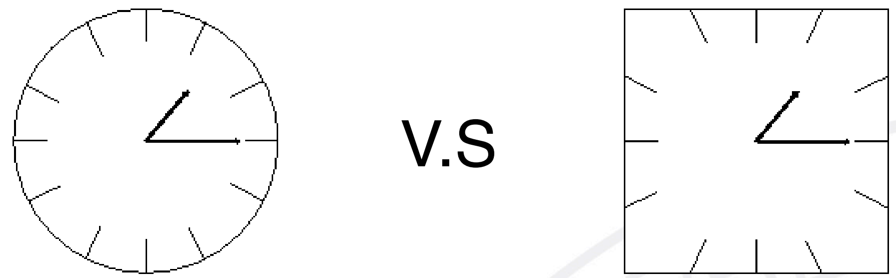
Environment



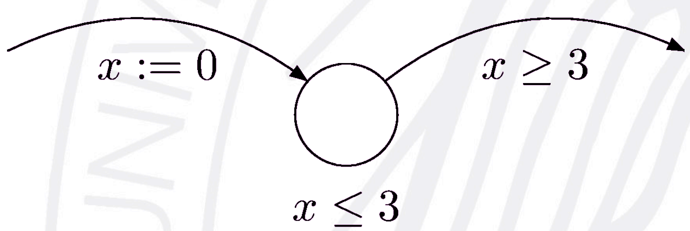
Classical controller
Not implementable



More...



- Models use continuous clocks and implementations use digital clocks with finite precision



Classical controller
Not implementable

Problems : Summary

- My controller stragegy may be correct because of
  - ... it is zeno...
  - ... it acts faster and faster?
  - ... it reacts instanteously to events, timeouts,...? (synchrony hypothesis)
  - ... it uses infinitely precise clocks?

A possible solution…

- Give an alternative semantics to timed automata : Almost ASAP semantics.
  - enabled transitions of the controller become urgent only after Δ time units;
  - events from the environment are received by the controller within Δ time units;
  - truth values of guards are enlarged by **f(Δ)**.

where Δ is a parameter

Definition of the AASAP semantics

Definition 13 [AASAP semantics] Given an ELASTIC controller

$$A = \langle \mathsf{Loc}, l_0, \mathsf{Var}, \mathsf{Lab}, \mathsf{Edg} \rangle$$

and $\Delta \in \mathbb{Q}^{\geq 0}$, the AASAP semantics of A, noted $[\![A]\!]_{\Delta}^{\mathsf{AAsap}}$ is the STTS

$$\mathcal{T} = \langle S, \iota, \Sigma_{\mathsf{in}}, \Sigma_{\mathsf{out}}, \Sigma_{\tau}, \rightarrow \rangle$$

where:

(A1) S is the set of tuples (l, v, I, d) where $l \in \mathsf{Loc}$, $v \in [\mathsf{Var} \rightarrow \mathbb{R}^{\geq 0}]$, $I \in [\Sigma_{\mathsf{in}} \rightarrow \mathbb{R}^{\geq 0} \cup \{\bot\}]$ and $d \in \mathbb{R}^{\geq 0}$;

(A2) $\iota = (l_0, v, I, 0)$ where v is such that for any $x \in \mathsf{Var} : v(x) = 0$, and I is such that for any $\sigma \in \Sigma_{\mathsf{in}}$, $I(\sigma) = \bot$;

(A3) $\Sigma_{\mathsf{in}} = \mathsf{Lab}_{\mathsf{in}}$, $\Sigma_{\mathsf{out}} = \mathsf{Lab}_{\mathsf{out}}$, and $\Sigma_{\tau} = \mathsf{Lab}_{\tau} \cup \mathsf{Lab}_{\mathsf{in}} \cup \{\epsilon\}$;

(A4) The transition relation is defined as follows:

- for the discrete transitions, we distinguish five cases:

(A4.1) let $\sigma \in \mathsf{Lab}_{\mathsf{out}}$. We have $((l, v, I, d), \sigma, (l', v', I, 0)) \in \rightarrow$ iff there exists $(l, l', g, \sigma, R) \in \mathsf{Edg}$ such that $v \models {}_{\Delta}g_{\Delta}$ and $v' = v[R := 0]$;

(A4.2) let $\sigma \in \mathsf{Lab}_{\mathsf{in}}$. We have $((l, v, I, d), \sigma, (l, v, I', d)) \in \rightarrow$ iff $I(\sigma) = \bot$ and $I' = I[\sigma := 0]$;

(A4.3) let $\bar{\sigma} \in \mathsf{Lab}_{\mathsf{in}}$. We have $((l, v, I, d), \bar{\sigma}, (l', v', I', 0)) \in \rightarrow$ iff there exists $(l, l', g, \sigma, R) \in \mathsf{Edg}$, $v \models {}_{\Delta}g_{\Delta}$, $I(\sigma) \neq \bot$, $v' = v[R := 0]$ and $I' = I[\sigma := \bot]$;

(A4.4) let $\sigma \in \mathsf{Lab}_{\tau}$. We have $((l, v, I, d), \sigma, (l', v', I, 0)) \in \rightarrow$ iff there exists $(l, l', g, \sigma, R) \in \mathsf{Edg}$, $v \models {}_{\Delta}g_{\Delta}$, and $v' = v[R := 0]$;

(A4.5) let $\sigma = \epsilon$. We have for any $(l, v, I, d) \in S : ((l, v, I, d), \epsilon, (l, v, I, d)) \in \rightarrow$.

- for the continuous transitions:

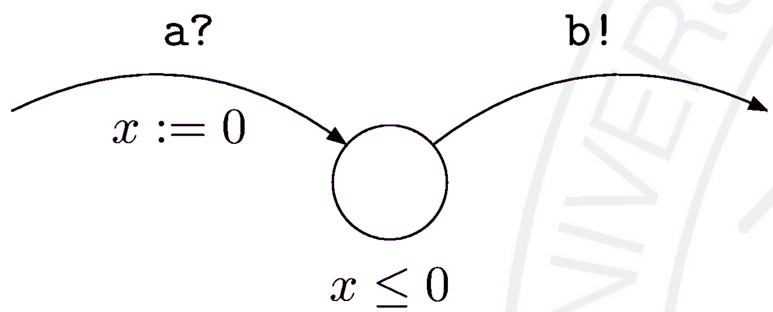
(A4.6) for any $t \in \mathbb{R}^{\geq 0}$, we have $((l, v, I, d), t, (l, v + t, I + t, d + t)) \in \rightarrow$ iff the two following conditions are satisfied:

  - for any edge $(l, l', g, \sigma, R) \in \mathsf{Edg}$ with $\sigma \in \mathsf{Lab}_{\mathsf{out}} \cup \mathsf{Lab}_{\tau}$, we have that:
  $$\forall t' : 0 \leq t' \leq t : (d + t' \leq \Delta \vee \mathsf{TS}(v + t', g) \leq \Delta)$$
  - for any edge $(l, l', g, \sigma, R) \in \mathsf{Edg}$ with $\sigma \in \mathsf{Lab}_{\mathsf{in}}$, we have that:
  $$\forall t' : 0 \leq t' \leq t : (d + t' \leq \Delta \vee \mathsf{TS}(v + t', g) \leq \Delta \vee (I + t')(\sigma) \leq \Delta)$$

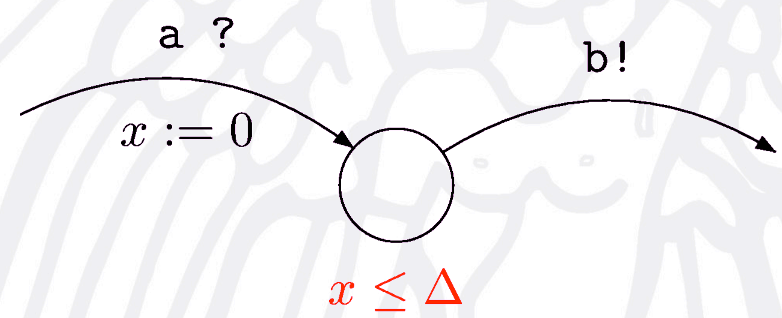
Intuition…

One can specify instantaneous responses
but not implement them.

Not implementable



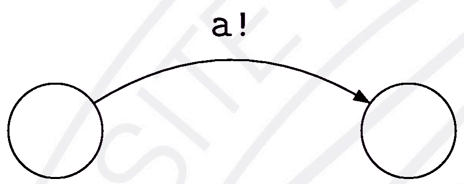
Solution : allow some delay



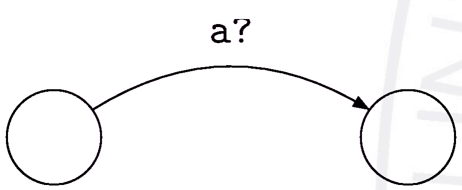
Intuition...

Instantaneous synchronisations between environment and controller are not implementable.

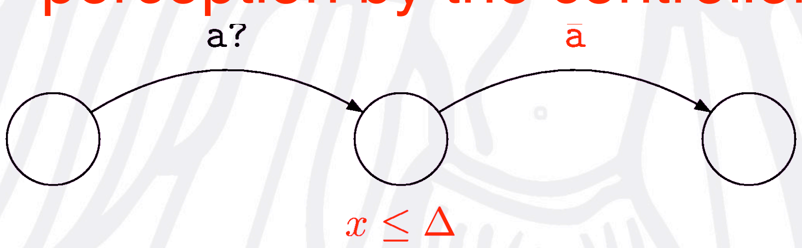
Environment



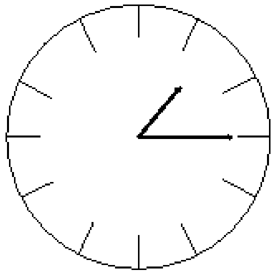
Classical controller
Not implementable



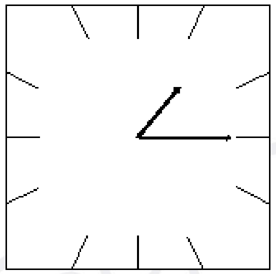
Solution :
Uncouple event from perception by the controller



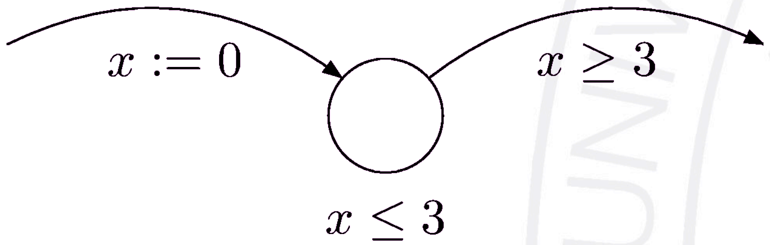
Intuition...



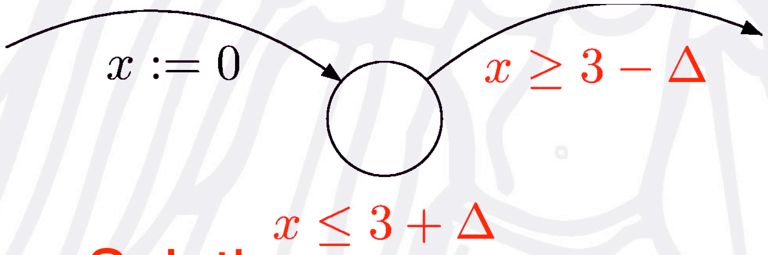
V.S



Models use continuous clocks and implementations use digital clocks with finite precision



Classical controller
Not implementable



Solution :
Slightly relax the constraints

Verification

- The question that we ask when we make verification is no more:

Does Environment || ControllerMod avoid Bad ?

- But:

for which values of Δ,
does Environment || ControllerMod(Δ) avoid Bad ?

Three variations

- Fixed (you know your target platform) :

Given $\Delta > 0$,
does Environment || ControllerMod(Δ) avoid Bad ?

- Existence (is my system implementable ?) :

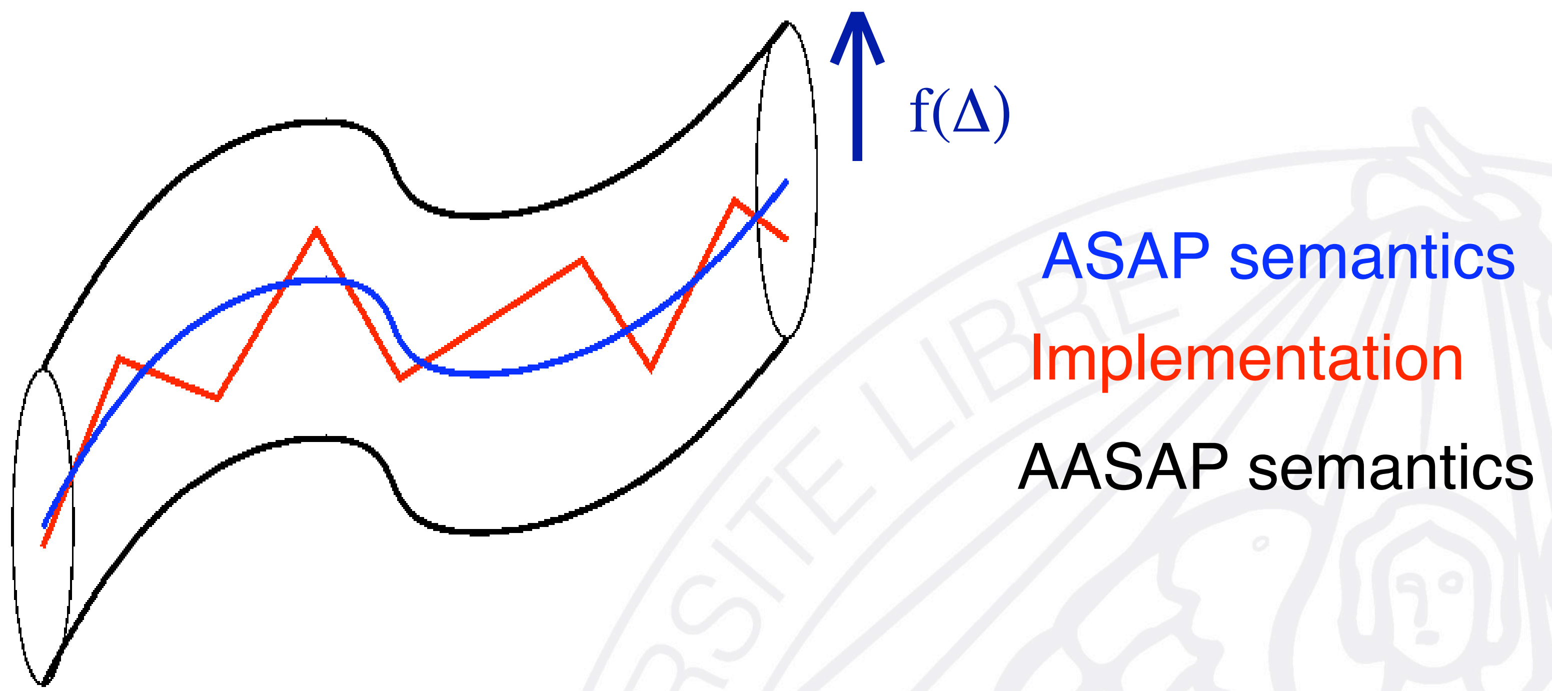
does there exist $\Delta > 0$ such that
Environment || ControllerMod(Δ) avoid Bad ?

- Max (how fast must my controller be ?) :

Max Δ such that

Implementability of the AASAP semantics

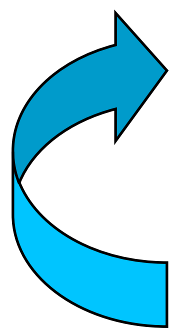
Intuition



- AASAP semantics defines a “tube” of strategies instead of a unique strategy in the ASAP semantics.
- This tube can be refined into an implementation while preserving safety properties verified on the AASAP-sem

Proof of "implementability" ?

- We define an "implementation semantics" based on:



Read System Clock
Update Sensor Values
Check all transitions and fire one if possible

- The timed behaviour of this scheme is determined by two values :
  - Time length of a loop : Δ_L
  - Time between two clock ticks : Δ_P

Program semantics

Definition 15 [Program Semantics] Let A be an ELASTIC controller and $\Delta_L, \Delta_P \in \mathbb{Q}^{>0}$. We define $\Delta_S = \Delta_L + 2\Delta_P$. The (Δ_L, Δ_P) program semantics of A, noted $[\![A]\!]^{\mathsf{Prg}}_{\Delta_L,\Delta_P}$ is the structured timed transition system $\mathcal{T} = \langle S, \iota, \Sigma_{\mathsf{in}}, \Sigma_{\mathsf{out}}, \Sigma_\tau, \rightarrow\rangle$ where:

($P1$) S is the set of tuples (l, r, T, I, u, d, f) such that $l \in \mathsf{Loc}$, r is a function from Var into $\mathbb{R}^{\geq 0}$, $T \in \mathbb{R}^{\geq 0}$, I is a function from $\mathsf{Lab_{in}}$ into $\mathbb{R}^{\geq 0} \cup \{\bot\}$, $u \in \mathbb{R}^{\geq 0}$, $d \in \mathbb{R}^{\geq 0}$, and $f \in \{\top, \bot\}$;

($P2$) $\iota = (l_0, r, 0, I, 0, 0, \bot)$ where r is such that for any $x \in \mathsf{Var}$, $r(x) = 0$, I is such that for any $\sigma \in \mathsf{Lab_{in}}$, $I(\sigma) = \bot$;

($P3$) $\Sigma_{\mathsf{in}} = \mathsf{Lab_{in}}$, $\Sigma_{\mathsf{out}} = \mathsf{Lab_{out}}$, $\Sigma_\tau = \mathsf{Lab}_\tau \cup \overline{\mathsf{Lab_{in}}} \cup \{\epsilon\}$;

($P4$) the transition relation $\rightarrow$ is defined as follows:

- for the discrete transitions:

($P4.1$) let $\sigma \in \mathsf{Lab_{out}}$. $((l, r, T, I, u, d, \bot), \sigma, (l', r', T, I, u, 0, \top)) \in\rightarrow$ iff there exists $(l, l', g, \sigma, R) \in \mathsf{Edg}$ such that $\lfloor T \rfloor_{\Delta_P} - r \models {}_{\Delta_S}g_{\Delta_S}$ and $r' = r[R := \lfloor T \rfloor_{\Delta_P}]$.

($P4.2$) let $\sigma \in \mathsf{Lab_{in}}$. $((l, r, T, I, u, d, f), \sigma, (l, r, T, I', u, d, f)) \in\rightarrow$ iff $I(\sigma) = \bot$ and $I' = I[\sigma := 0]$;

($P4.3$) let $\bar\sigma \in \overline{\mathsf{Lab_{in}}}$. $((l, r, T, I, u, d, \bot), \bar\sigma, (l', r', T, I', u, 0, \top)) \in\rightarrow$ iff there exists $(l, l', g, \sigma, R) \in \mathsf{Edg}$ such that $\lfloor T \rfloor_{\Delta_P} - r \models {}_{\Delta_S}g_{\Delta_S}$, $I(\sigma) > u$, $r' = r[R := \lfloor T \rfloor_{\Delta_P}]$ and $I' = I[\sigma := \bot]$;

($P4.4$) let $\sigma \in \mathsf{Lab}_\tau$. $((l, r, T, I, u, d, \bot), \sigma, (l', r', T, I, u, 0, \top)) \in\rightarrow$ iff there exists $(l, l', g, \sigma, R) \in \mathsf{Edg}$ such that $\lfloor T \rfloor_{\Delta_P} - r \models {}_{\Delta_S}g_{\Delta_S}$ and $r' = r[R := \lfloor T \rfloor_{\Delta_P}]$.

($P4.5$) let $\sigma = \epsilon$. $((l, r, T, I, u, d, f), \sigma, (l, r, T + u, I, 0, d, \bot)) \in\rightarrow$ iff either $f = \top$ or the two following conditions hold:

- for any $\bar\sigma$ such that $\sigma \in \mathsf{Lab_{in}}$, for any $(l, l', g, \sigma, R) \in \mathsf{Edg}$, we have that either $\lfloor T \rfloor_{\Delta_P} - r \not\models {}_{\Delta_S}g_{\Delta_S}$ or $I(\sigma) \leq u$
- for any $\sigma \in \mathsf{Lab_{out}} \cup \mathsf{Lab}_\tau$, for any $(l, l', g, \sigma, R) \in \mathsf{Edg}$, we have that $\lfloor T \rfloor_{\Delta_P} - r \not\models {}_{\Delta_S}g_{\Delta_S}$

- for the continuous transitions:

($P4.6$) $((l, r, T, I, u, d, f), t, (l, r, T, I + t, u + t, d + t, f)) \in\rightarrow$ iff $u + t \leq \Delta_L$.

Proof of "implementability" ?

Theorem :

For any timed controller, its AASAP semantics simulates (in the formal sense) its implementation semantics, provided that :

$$\Delta > 3\Delta_L + 4\Delta_P$$

In this case, the implementation is guaranteed to preserve verified properties of the model, that is:

Environment || ControllerMod(Δ) avoid Bad

implies

Environment || ControllerImpl(Δ_L,Δ_P) avoid Bad

Properties of the AASAP Semantics

- Faster is better !

For any Δ_1, Δ_2 such that $\Delta_1 < \Delta_2$:

if

Environment || ControllerMod(Δ_2) avoid Bad

then

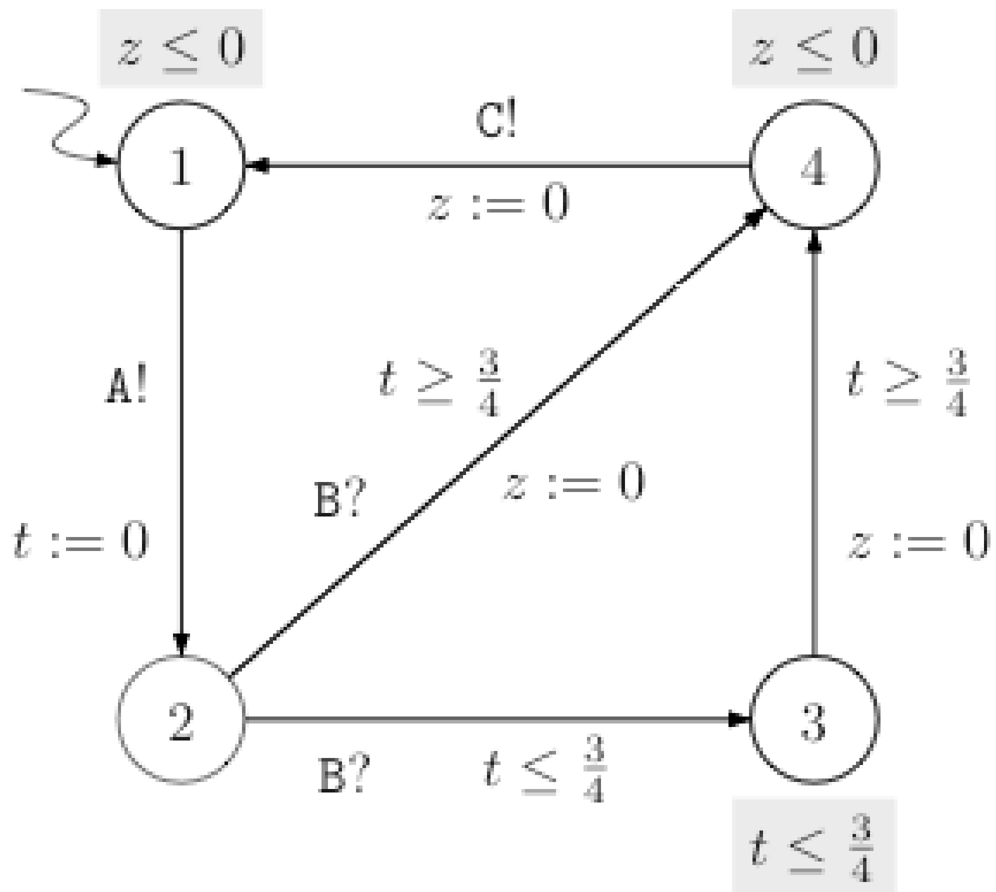
Environment || ControllerMod(Δ_1) avoid Bad

ULB UNIVERSITÉ LIBRE DE BRUXELLES

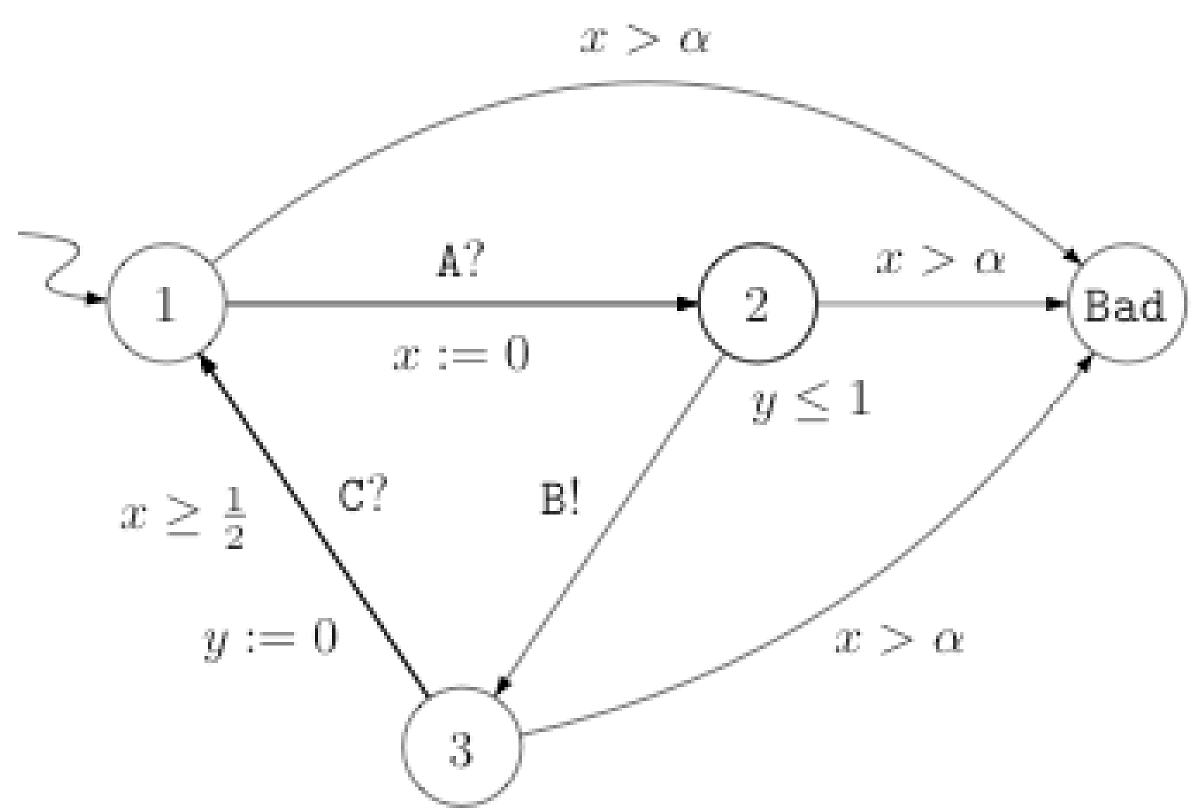
Properties of the AASAP Semantics

- If $\Delta>0$, we get for free a proof that strategies:
  - are nonzeno
  - are such that transitions does not need to be taken faster and faster
- If only $\Delta=0$ guarantees some reachability property, then the control strategy is not implementable

An example



(a) The ASAP controller



(b) The environment

If $\alpha=1$ then the system is safe if and only if $\Delta=0$

If $\alpha=2$ then the system is safe if and only if $\Delta<0.25$

In practice ?

- The AASAP semantics can be coded into a parametric timed automata with only one clock compared to the parameter $\Delta \in Q$.
- Unfortunately, the reachability problem for that class of timed automata is undecidable… Direct corollary of [CHR02].
- Hytech implements a semi-decision procedure for that problem.
- Does there exist $\Delta > 0$ such that
  Environment || ControllerMod(Δ) avoid Bad ?

Tool Set

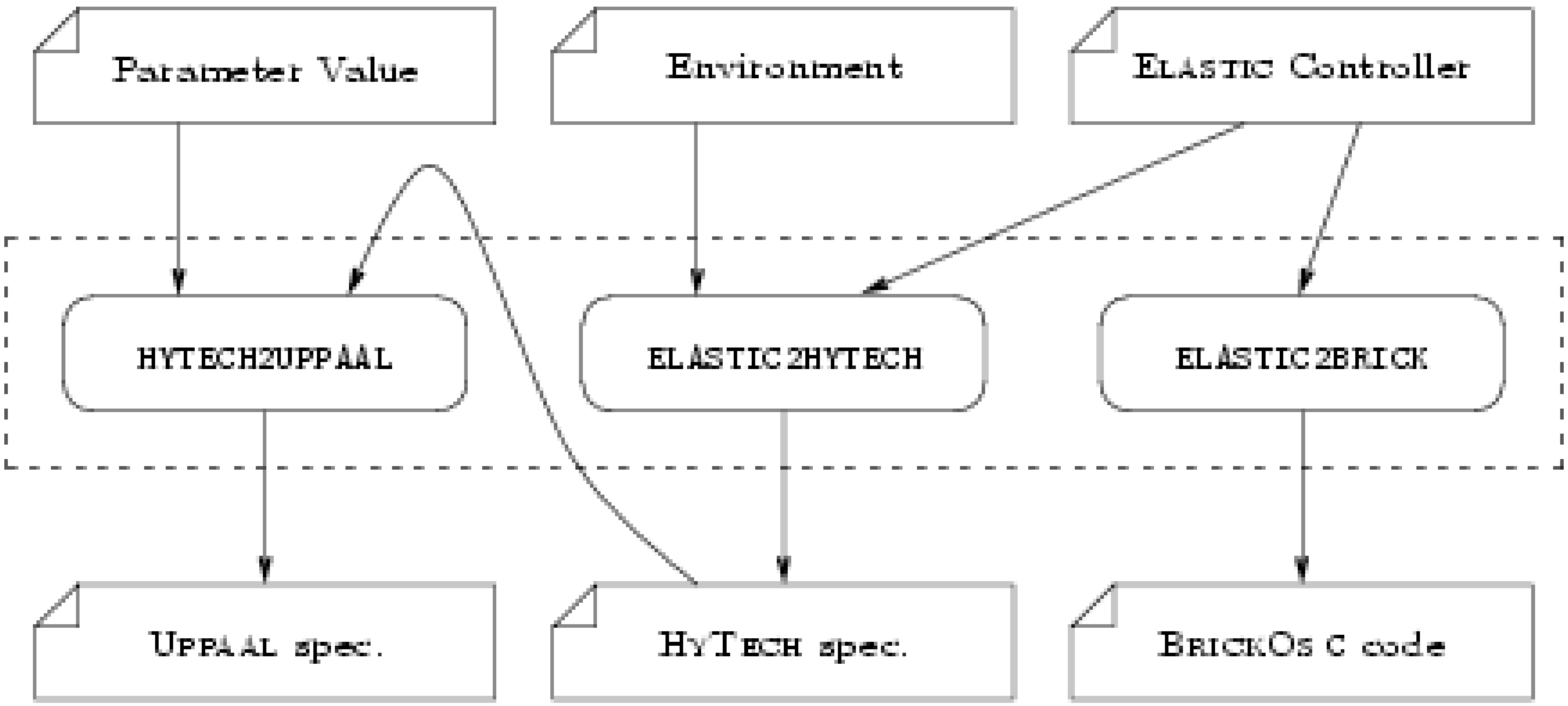


Fig. 5. Structure of our tool set.

Methodology to develop controllers

❶ Models using synchrony hypothesis
Environment || ControllerMod

❷ Check
Does Environment || ControllerMod(0) avoid Bad ?

❸ Compute the largest Δ_1 such that
Environment || ControllerMod(Δ_1) avoid Bad

❹ if $\Delta_1 > 3\ \Delta_L + 4\ \Delta_P$

❺ Generate code
This code will enforce the safety property

Conclusion

- Two player games are natural theoretical model to study the synthesis problem
- There exist elegant algorithms to solve general games
- The step to go from a model to a correct implementation needs more investigations

Bibliography

General references on games and synthesis

[AHK02] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *J. ACM*, 49:672–713, 2002.

[GH82] Y. Gurevich and L. Harrington. Trees, automata, and games. STOC 1982: In *Proceedings of the 14th International Symposium on Theory of Computing*, pages 60–65, ACM Press, 1984.

[PR90] A. Pnueli and R. Rosner. Distributed reactive systems are hard to synthesize. In *Proceedings of the 31st International Symposium on Foundations of Computer Science*, pages 746–757. IEEE Computer Society Press, 1990.

[Rei84] J.H. Reif. The complexity of two-player games of incomplete information. *Journal on Computer and System Sciences*, 29:274–301, 1984.

[Tho95] W. Thomas. On the synthesis of strategies in infinite games. In *Proceedings of the 12th International Symposium on Theoretical Aspects of Computer Science*, volume 900 of Lecture Notes in Computer Science, pages 1–13. Springer-Verlag, 1995.

[RW89] P.J.G. Ramadge and W.M. Wonham. The control of discrete-event systems. *IEEE Transactions on Control Theory*, 77:81–98, 1989.

References on timed and hybrid games

[AMPS98] E. Asarin, O. Maler, A. Pnueli, and J. Sifakis. Controller synthesis for timed automata. In *Proc. IFAC Symp. System Structure and Control*, pages 469–474. Elsevier, 1998.

[BDMP02] P. Bouyer, D. D'Souza, P. Madhusudan, and A. Petit. Timed control with partial observability. Research Report LSV-02-5, LSV, ENS de Cachan, France, 2002.

[CHR02] F. Cassez, T.A. Henzinger, and J.-F. Raskin. A comparison of control problems for timed and hybrid systems. In *Proc. 5th Int. Works. Hybrid Systems: Computation and Control (HSCC'02)*, volume 2289 of *LNCS*, pages 134–148. Springer, 2002.

[HHM99] T.A. Henzinger, B. Horowitz, and R. Majumdar. Rectangular hybrid games. In *Concurrency Theory*, Lect. Notes in Comp. Sci. 1664, pages 320–335. Springer, 1999.

[HK99] T.A. Henzinger and P.W. Kopke. Discrete-time control for rectangular hybrid automata. *Theor. Comp. Sci.*, 221:369–392, 1999.

References on implementability issues and robustness

[DDR04] M. De Wulf, L. Doyen, and J.-F. Raskin. Almost ASAP semantics: From timed models to timed implementations. In *HSCC 04: Hybrid Systems—Computation and Control*, Lecture Notes in Computer Science 2993, pages 296–310. Springer-Verlag, 2004.

Martin De Wulf, Laurent Doyen, Nicolas Markey, and Jean-François Raskin. Robustness and Implementability of Timed Automata. In FORMATS'04, Lecture Notes in Computer Science, 3253, pp. 118-133, Springer Verlag, 2004.

Martin De Wulf, Laurent Doyen, Jean-François Raskin. Systematic Implementations of Timed Models. In Formal Methods Europe'05, LNCS 3582, pp. 139-156, Springer Verlag, 2005.

K. Altisen and S. Tripakis. Implementation of timed automata: an issue of semantics or modeling?. In FORMATS'05 (to appear). A previous version of this paper is available as VERIMAG Technical Report TR-2005-12.

[Pur98] Anuj Puri. Dynamical properties of timed automata. In *Proceedings of Formal Techniques in Real-Time and Fault-Tolerant Systems, 5th International Symposium, FTRTFT'98, Lyngby, Denmark, September 14-18, 1998*, volume 1486 of *Lecture Notes in Computer Science*, pages 210–227. Springer, 1998.

[GHJ97] V. Gupta, T.A. Henzinger, and R. Jagadeesan. Robust timed automata, *HART 97: Hybrid and Real-time Systems*. LNCS 1201, Springer-Verlag, 331–345, 1997.