

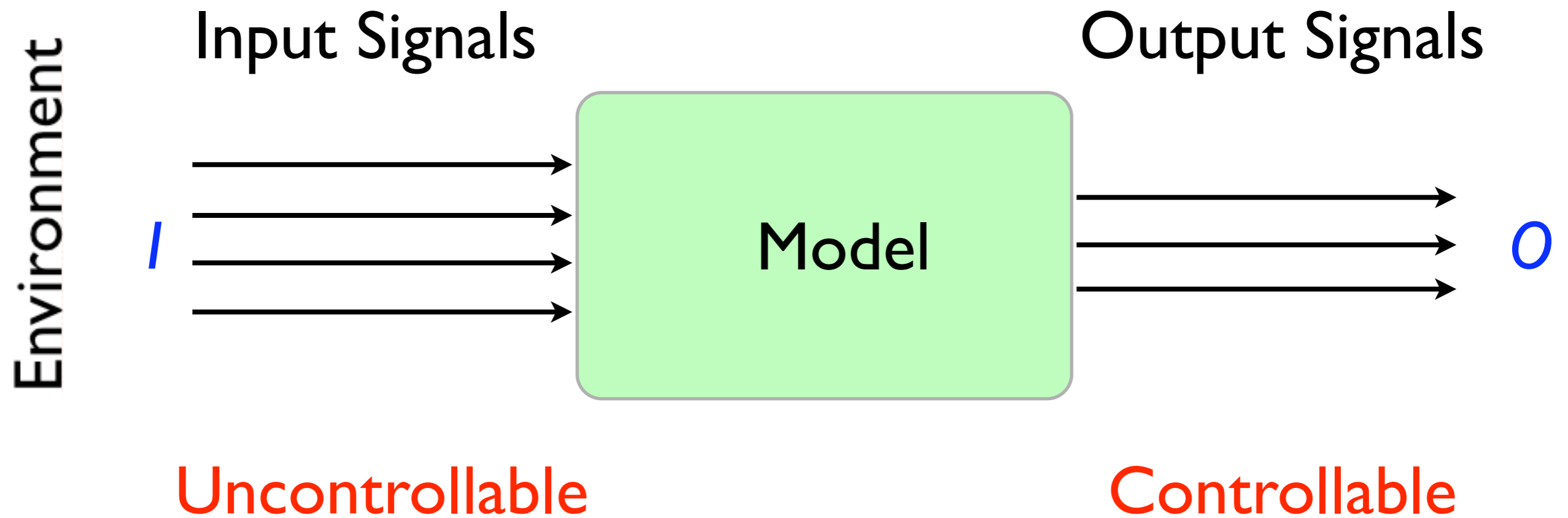
An Antichain Algorithm for LTL Realizability

Emmanuel Filiot Naiyong Jin Jean-François Raskin

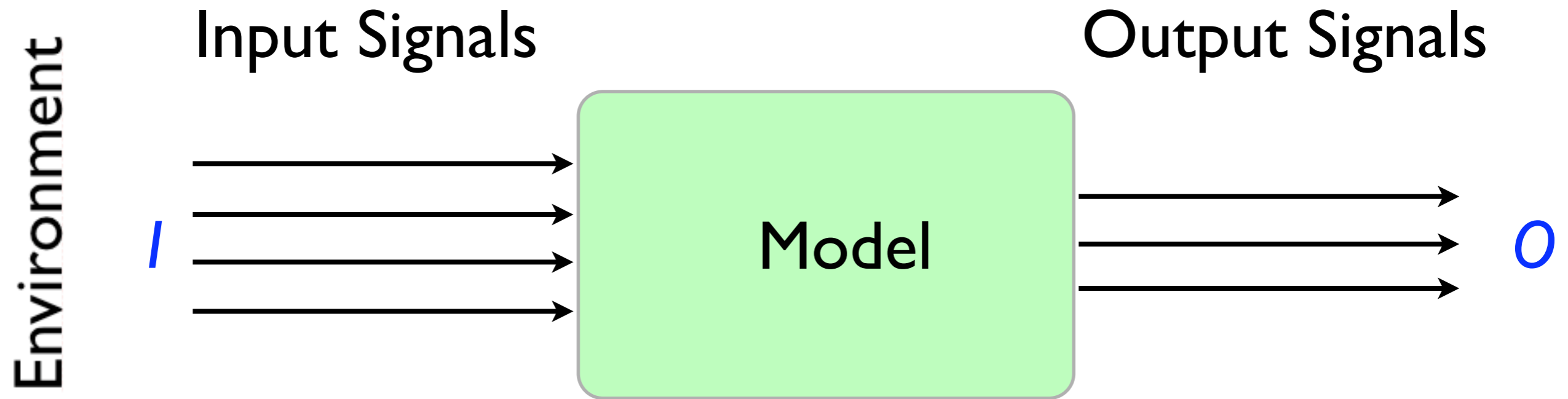
Université Libre de Bruxelles

2009, March, GASICS Meeting

Open Systems



Open Systems



Uncontrollable

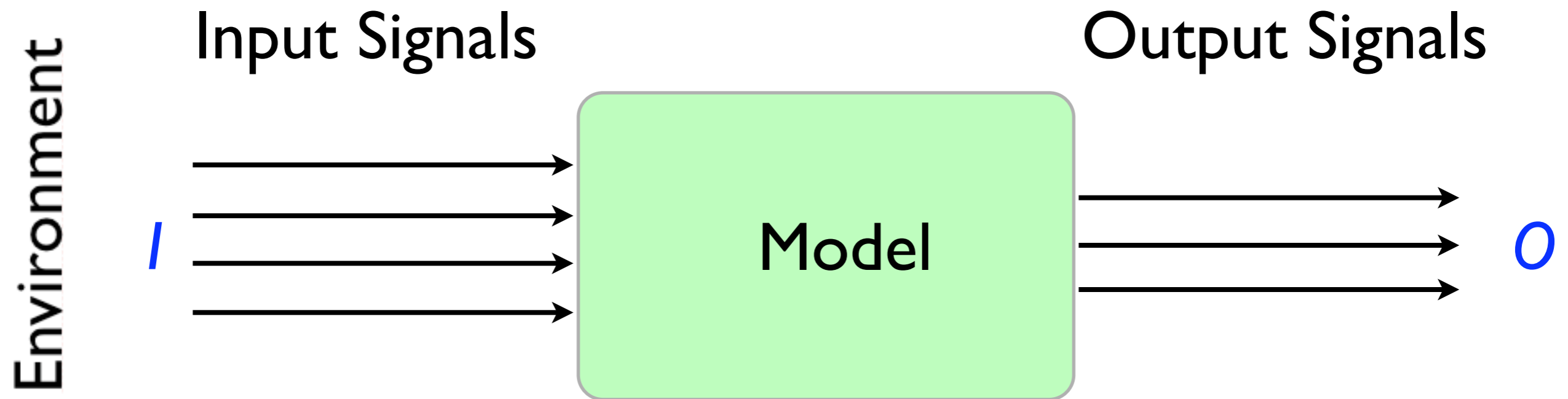
Controllable

synchronous execution = infinite words over $\Sigma = 2^{I \cup O}$

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

$o_j \subseteq O \quad i_j \subseteq I$

Open Systems



Uncontrollable

Controllable

synchronous execution = infinite words over $\Sigma = 2^{I \cup O}$

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

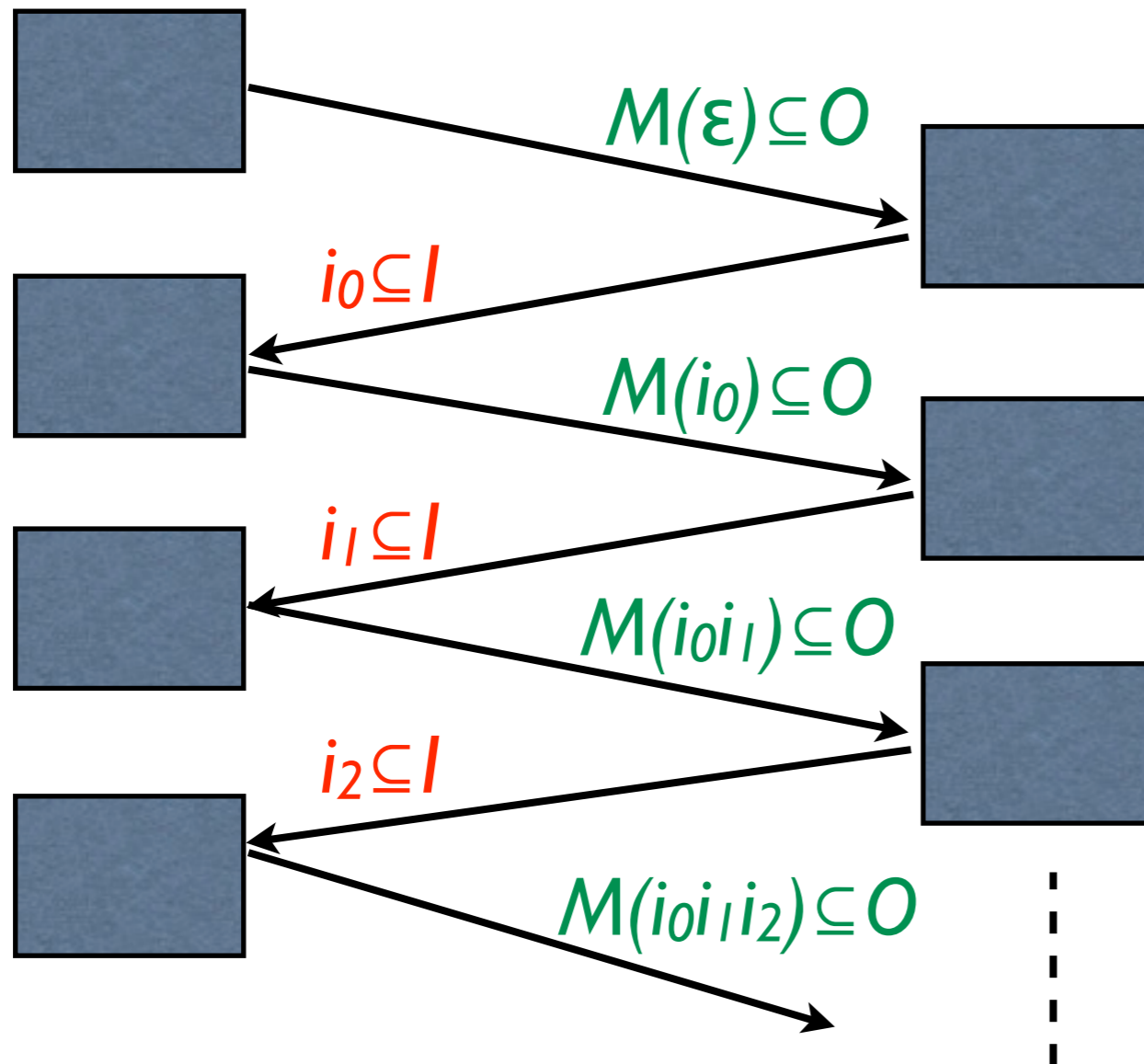
$o_j \subseteq O \quad i_j \subseteq I$

Realizability Problem: given spec Φ , $\exists M \forall E \forall e, e \models \Phi$

Game point of view

Model M

Environment

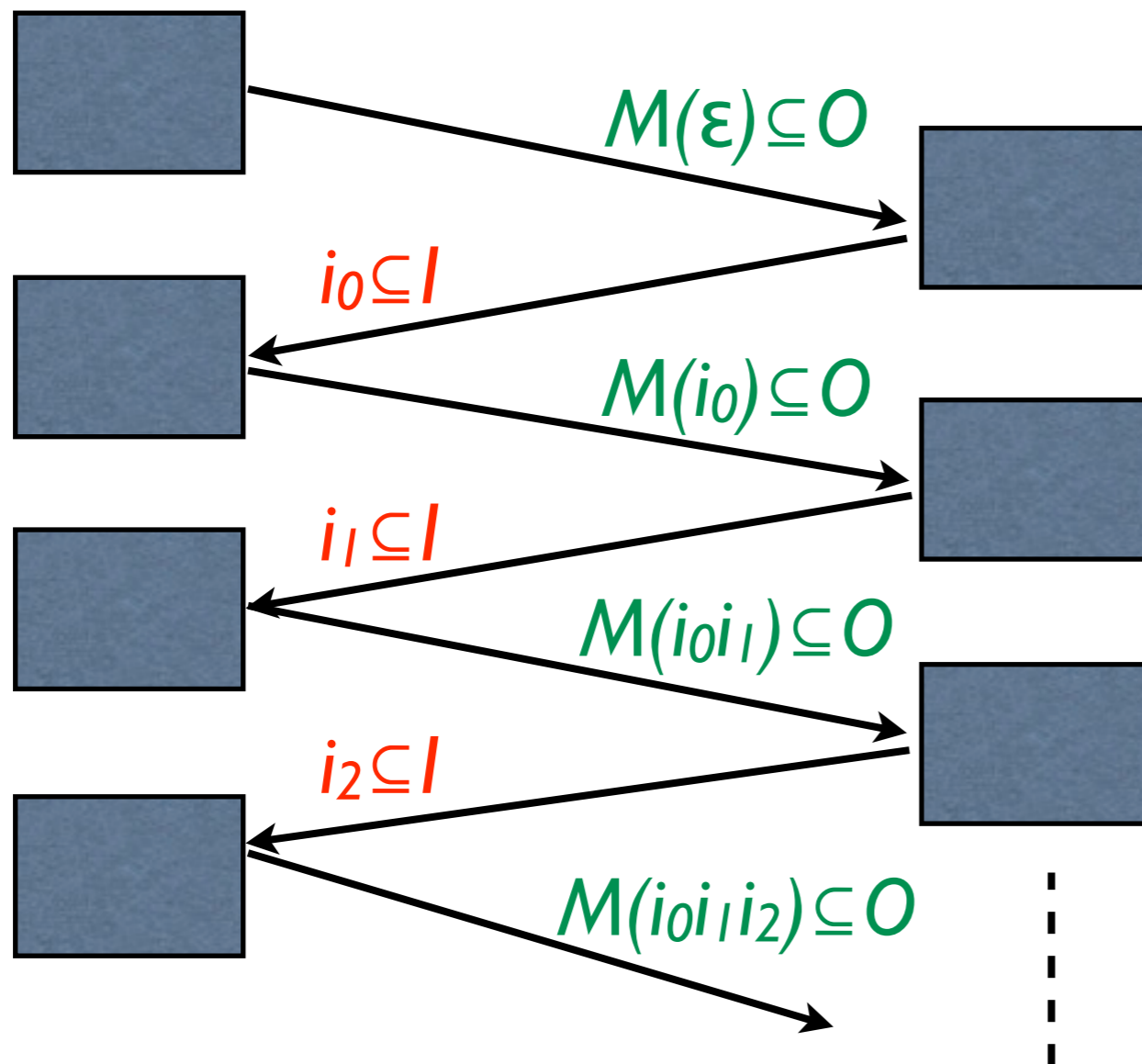


$$(M(\varepsilon) \cup i_0)(M(i_0) \cup i_1)(M(i_0 i_1) \cup i_2) \dots$$

Game point of view

Model M

Environment



How to win the game ?

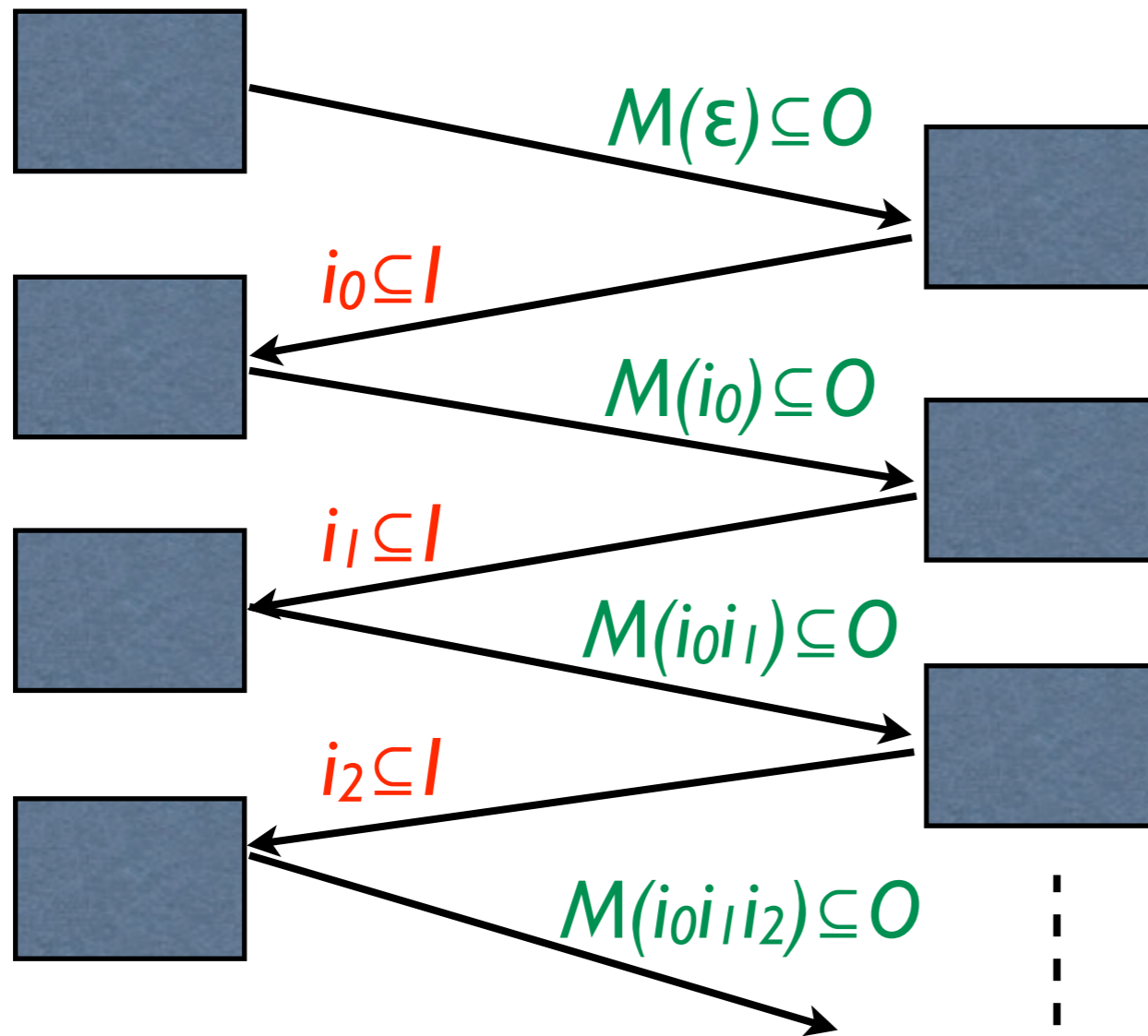
- winning condition
- LTL formula ϕ over atomic propositions $I \cup O$
- models of ϕ are words over $\Sigma = 2^{I \cup O}$

$(M(\varepsilon) \cup i_0)(M(i_0) \cup i_1)(M(i_0i_1) \cup i_2) \dots$

Game point of view

Model M

Environment



How to win the game ?

- winning condition
- LTL formula ϕ over atomic propositions $I \cup O$
- models of ϕ are words over $\Sigma = 2^{I \cup O}$

The model wins the game if

$(M(\epsilon) \cup i_0)(M(i_0) \cup i_1)(M(i_0 i_1) \cup i_2) \dots$ satisfies ϕ

Realizability Problem

- A model M is seen as a strategy $(2^I)^* \rightarrow 2^O$

$$L(M) = \{ (M(\varepsilon) \cup i_0)(M(i_0) \cup i_1)(M(i_0 i_1) \cup i_2) \dots \mid \forall i_0 \forall i_1 \forall i_2 \dots \}$$

- Given two finite sets I, O and an LTL formula ϕ over atomic propositions $I \cup O$, is there a strategy $M: (2^I)^* \rightarrow 2^O$ such that all words of $L(M)$ satisfies ϕ ?

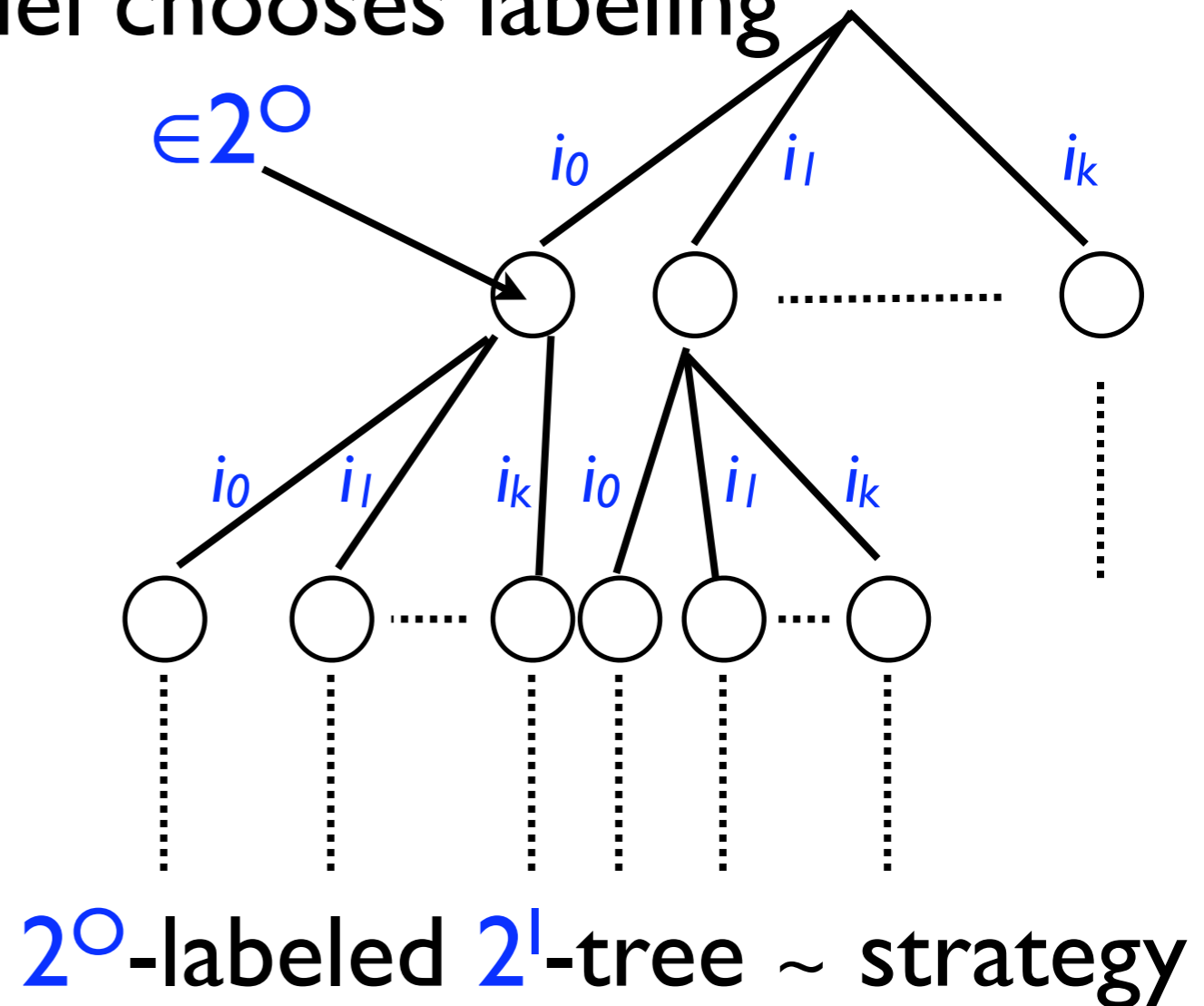
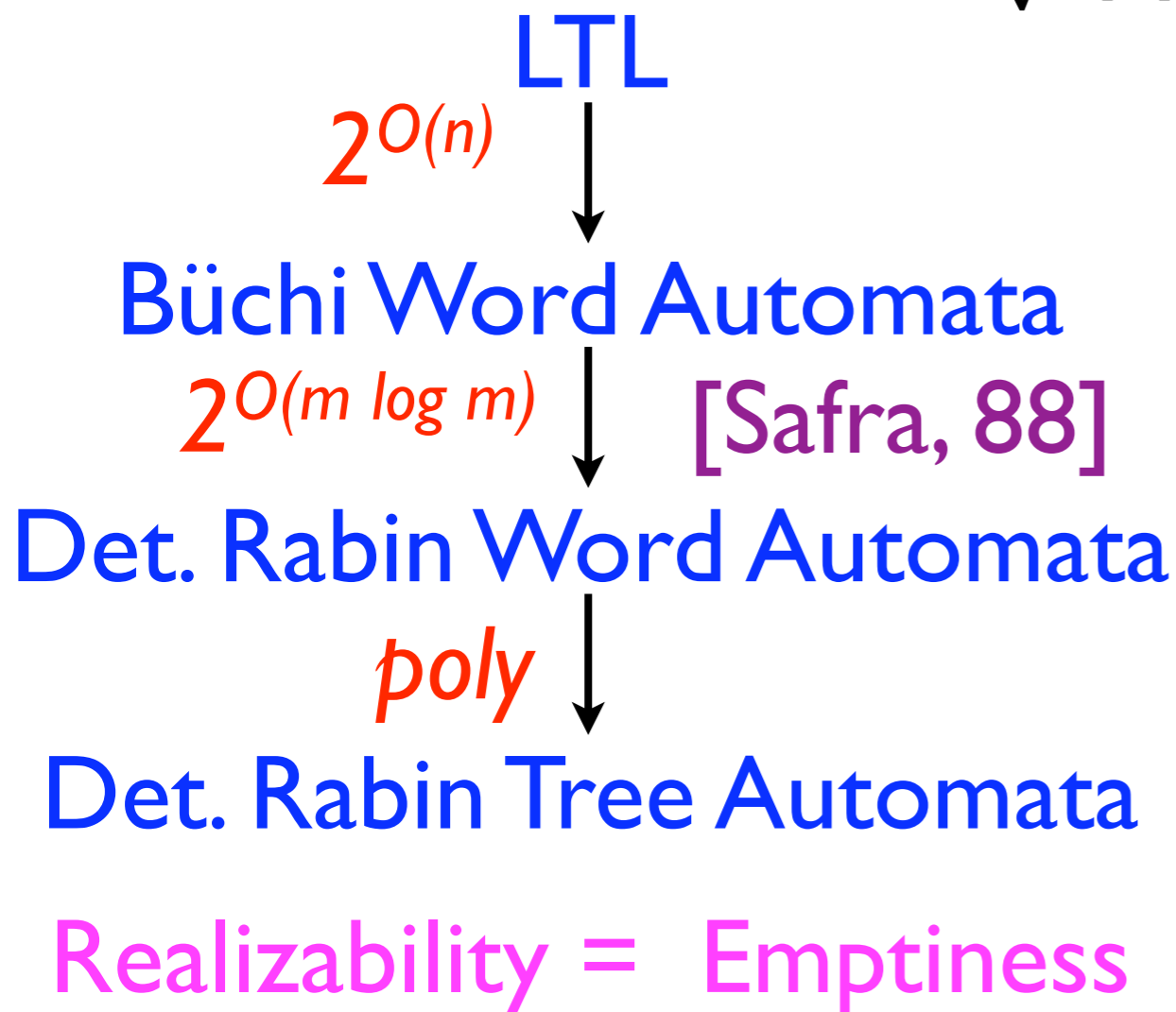
Example

- $I = \{q\}$, $O = \{p\}$, $\Phi = p U q$ is **unrealizable**
(environment never asserts q)
- but $\diamond q \rightarrow p U q$ is **realizable**
(model always asserts p)

Related Work

- LTL realizability introduced by Pnueli and Rosner, 1989, 2-EXPTIME upper-bound:

- ✓ Environment chooses branch
- ✓ Model chooses labeling



Related Work

- 2-Exptime Lower Bound [Rosner,92]
- Existing Safra's determinization implementations do not handle examples of reasonable scales in practice [Althoff,Thomas,Wallmeier,06]
- “Safraless” procedure [Kupferman,Vardi,05]

LTL \rightarrow *Universal Co-Büchi Tree A.* \rightarrow

Weak Alternating Tree A. \rightarrow *Büchi Tree A.*

- All those steps have been implemented and optimized in *Lily* [Jobstmann,Bloem,06]

Antichain Algorithms

- applied to solve automata problems (emptiness, universality, ...)
- exploit the structural properties of automata-based constructions
- syntactic and semantic partial order on states
- successfully applied to solve for instance:
 - Inclusion of Nondeterministic Büchi Automata [Doyen,Raskin,07]
 - LTL satisfiability and model-checking [De Wulf, Doyen, Maquet, Raskin, 08]
 - Inclusion and Universality of Finite Tree Automata [Bouajjani,Habermehl,Holik,Touili,Vojnar,08]

Contribution

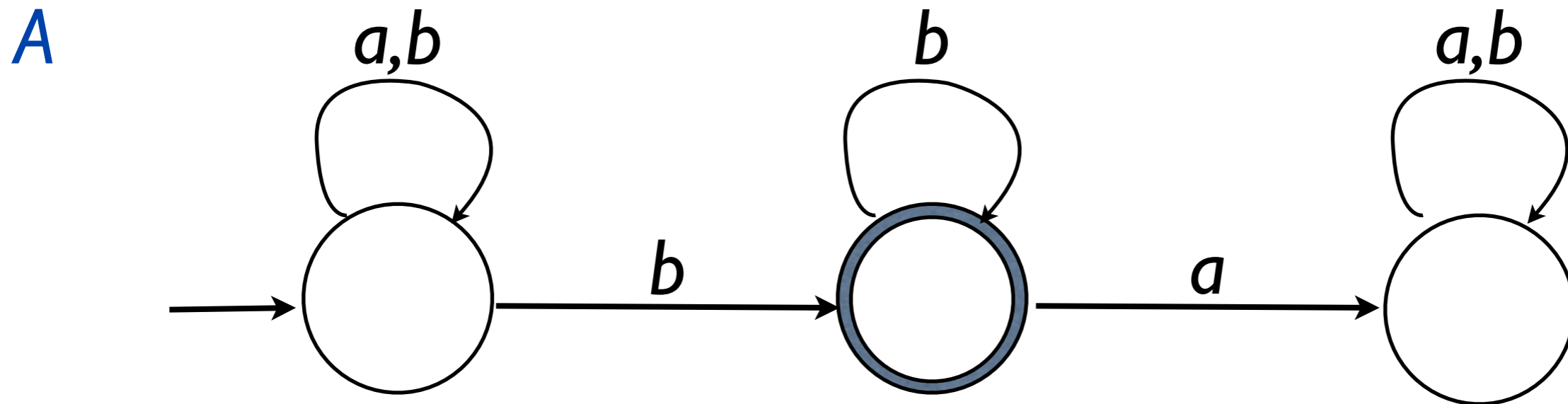
- Simple and direct **Safraless** procedure
- Reduction to a **safety game** via universal co-Büchi word automata
- Game solved by an algorithm that manipulates **antichains of game positions**
- **Implemented** and compared to Lily

Outline

- Reduction to a safety game
- Antichain Algorithm
- Experimental Results

Infinite Word Automata

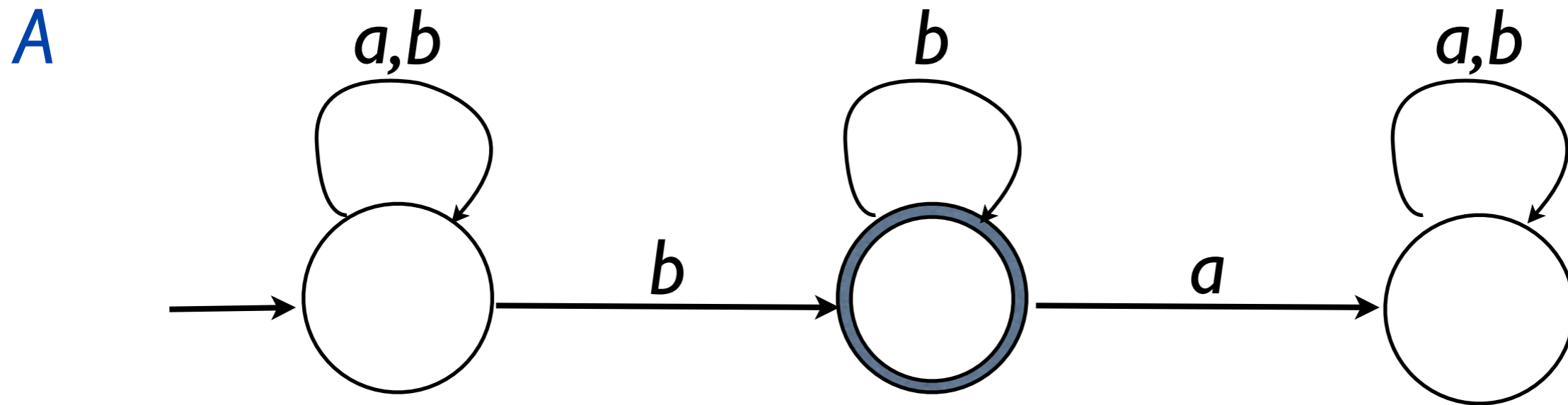
Alphabet: $\Sigma = 2^{I \cup O}$



Acceptance Condition	$w \in \Sigma^\omega$ is accepted if
Büchi	\exists run r on w : r visits final states ∞ often
Universal Co-Büchi	\forall run r on w : r visits final states finitely often

Infinite Word Automata

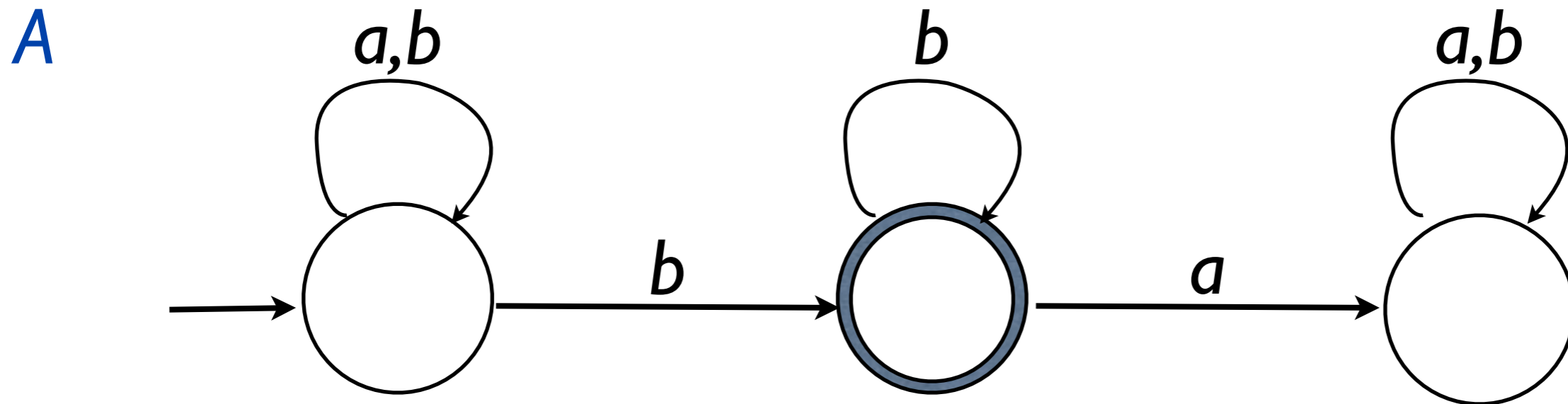
Alphabet: $\Sigma = 2^{I \cup O}$



Acceptance Condition	$w \in L(A)$ if
Büchi	???
Universal Co-Büchi	???

Infinite Word Automata

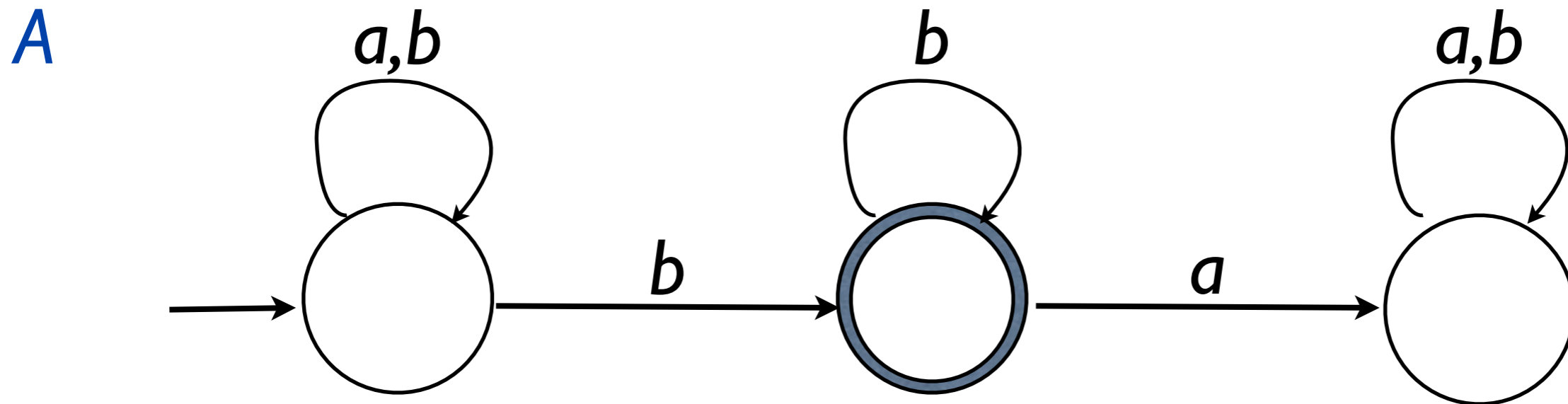
Alphabet: $\Sigma = 2^{I \cup O}$



Acceptance Condition	$w \in L(A)$ if
Büchi	w contains finitely many a
Universal Co-Büchi	???

Infinite Word Automata

Alphabet: $\Sigma = 2^{I \cup O}$



Acceptance Condition	$w \in L(A)$ if
Büchi	w contains finitely many a
Universal Co-Büchi	w contains infinitely many a

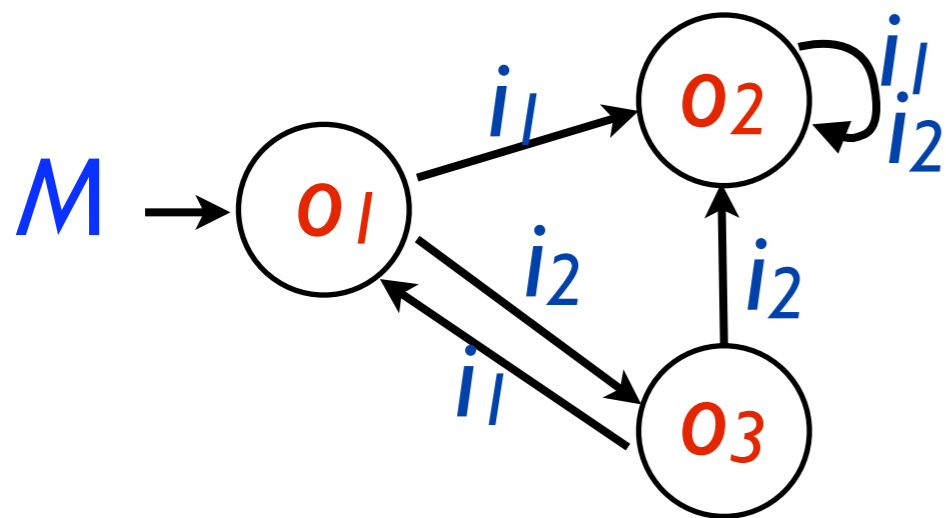
Specification

- LTL formula ϕ over atomic propositions $I \cup O$
- translated into a universal co-Büchi word automaton (UCW)
- remark: $L_{CoB}(A) = \Sigma^\omega \setminus L_B(A)$
- translate $\neg\phi$ into a non-deterministic Büchi $A_{\neg\phi}$
- interpret $A_{\neg\phi}$ as a universal co-Büchi, i.e.

$$L_{CoB}(A_{\neg\phi}) = \{ w \mid w \text{ satisfies } \phi \}$$

Strategies

- if an LTL formula is realizable, there exists a finite-state strategy that realizes it [PR89]
- finite-state strategies are represented as complete Moore machines



$L(M) = \text{traces of infinite paths}$

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

LTL Realizability reduces to:

Input: a UCW A

Output: Is there a non-empty complete Moore machine M such that $L(M) \subseteq L(A)$?

Bounding the number of visited final states

- **Lemma:** given a Moore machine with m states, and a UCW A with n states, if $L(M) \subseteq L(A)$, then all runs on words of $L(M)$ visit accepting states at most nm times.
- **Proof:** there is no reachable cycles containing an accepting state in $M \otimes A$

Reduction to a K -Co-Büchi Specification

- Universal K -Co-Büchi acceptance condition: all runs visit final states at most K times. UKCWs are denoted by (A, K) .

Theorem: Let A a UCW with n states and let $K = n(2^{2n+1} + 1)$. Then A is realizable iff (A, K) is realizable.

Proof: Back direction is trivial. For the converse:

- 1/ UCW $A \rightarrow$ [Safra,88] det Rabin \rightarrow det Rabin game G with $|G| = n^{2^{2n+1} + 1}$
- 2/ Rabin games admit memoryless strategies
- 3/ Therefore A realizable $\Rightarrow \exists M$ with $|G|$ states that realizes it
- 4/ Apply previous Lemma to get the bound on the number of accepting states

Determinization of UKCWs

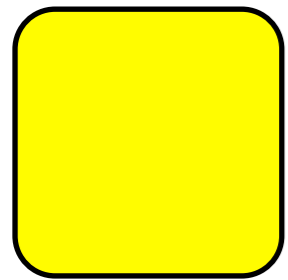
Lemma: UKCWs are determinizable (modulo exptime)

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

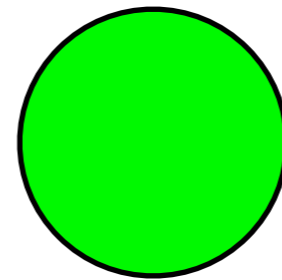
$$\Delta_d(F, \sigma) : q \rightarrow \max_{(q', \sigma, q) \in \Delta} \{ F(q') + (q \in \alpha) \mid F(q') \neq -1 \}$$

Safety Game

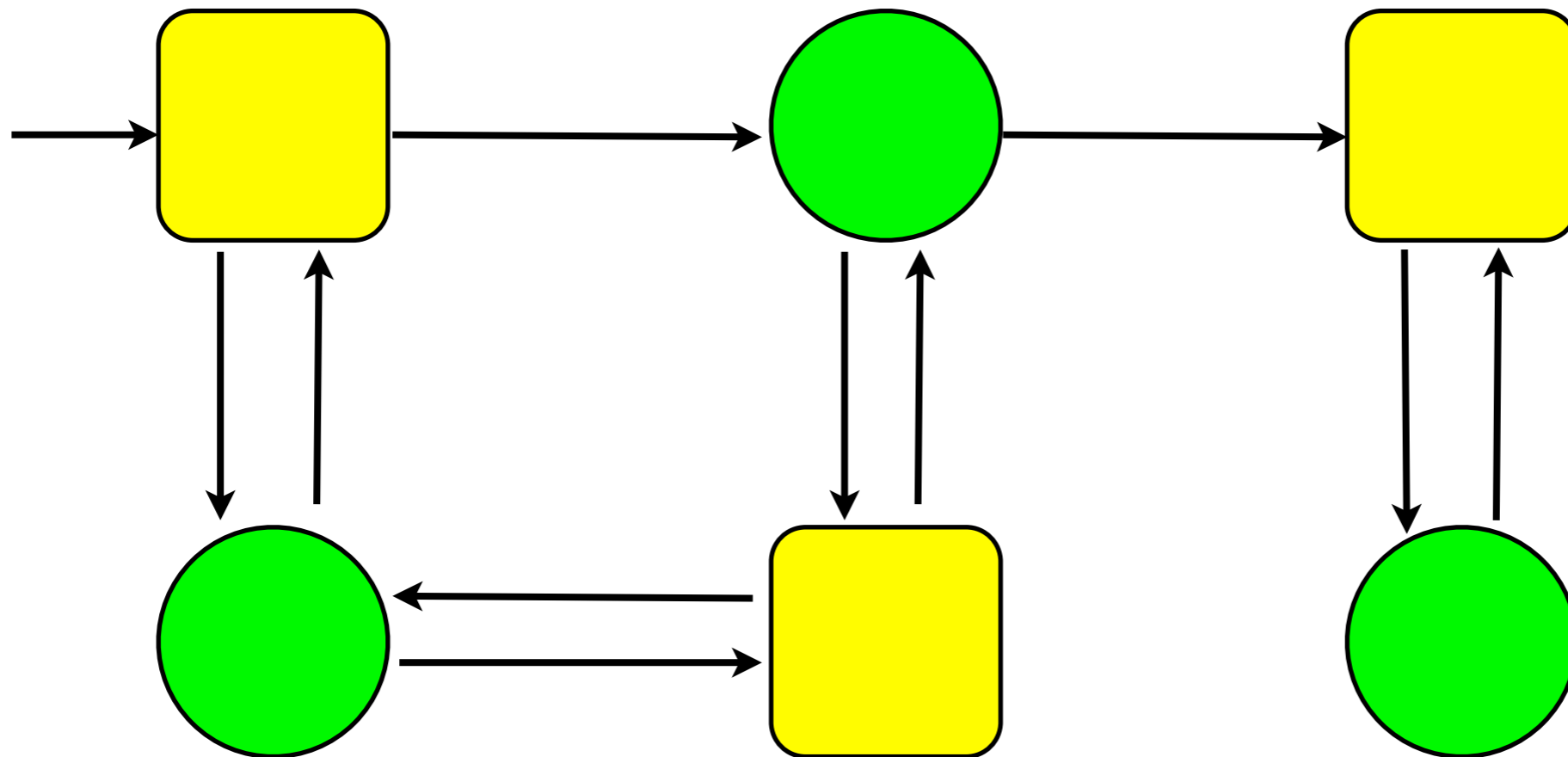
- LTL realizability \rightarrow UKCW realizability \rightarrow safety game



Protagonist's positions

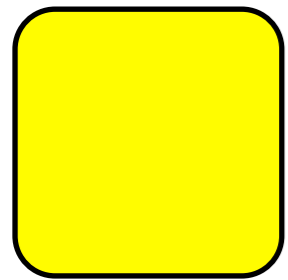


Adversary's positions

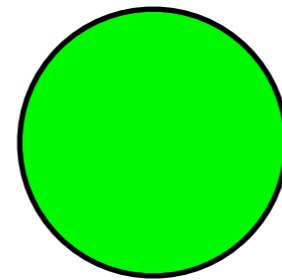


Safety Game

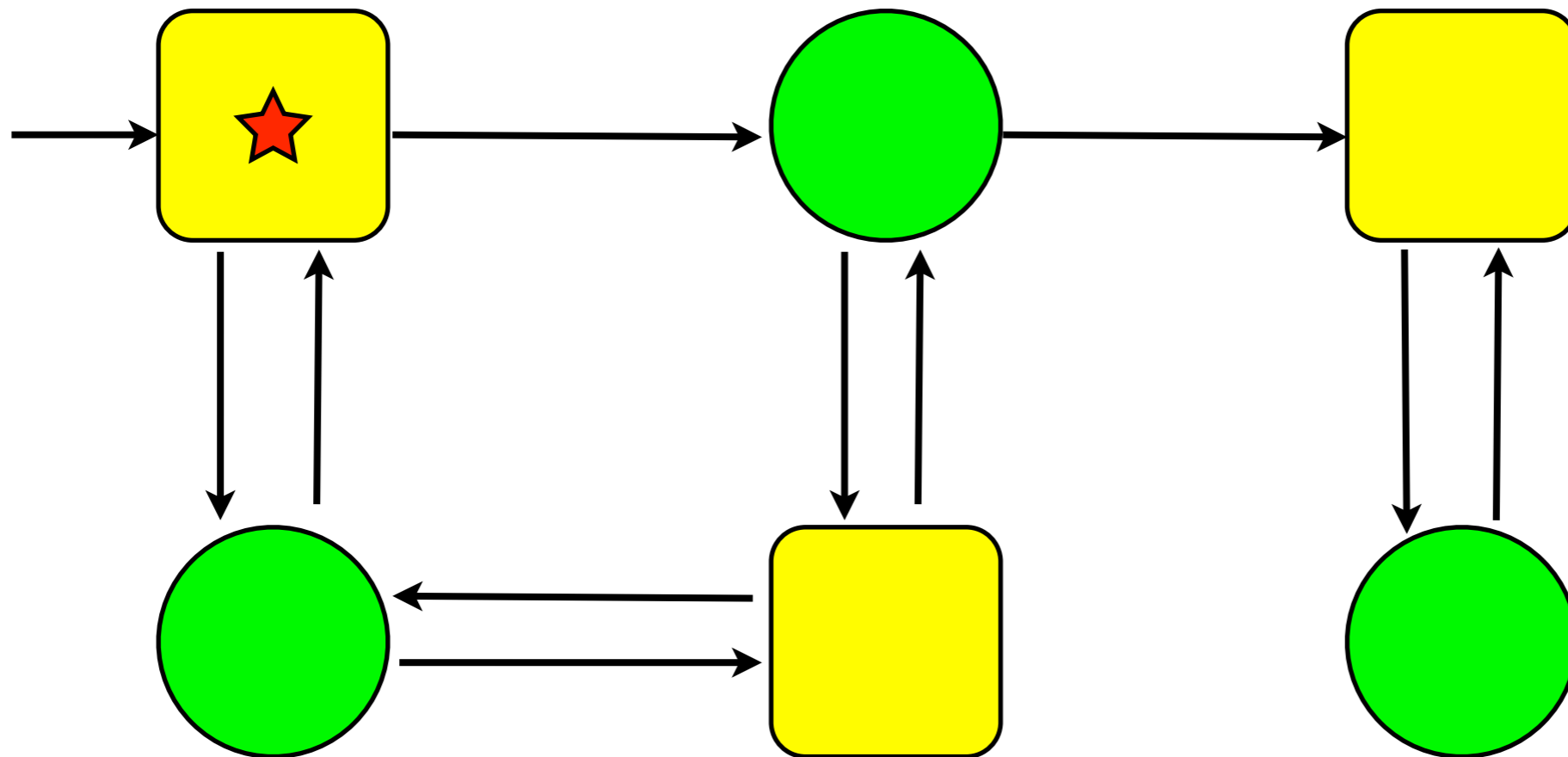
- LTL realizability \rightarrow UKCW realizability \rightarrow safety game



Protagonist's positions

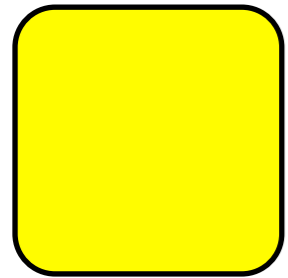


Adversary's positions

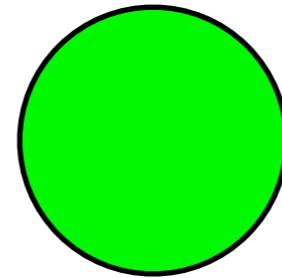


Safety Game

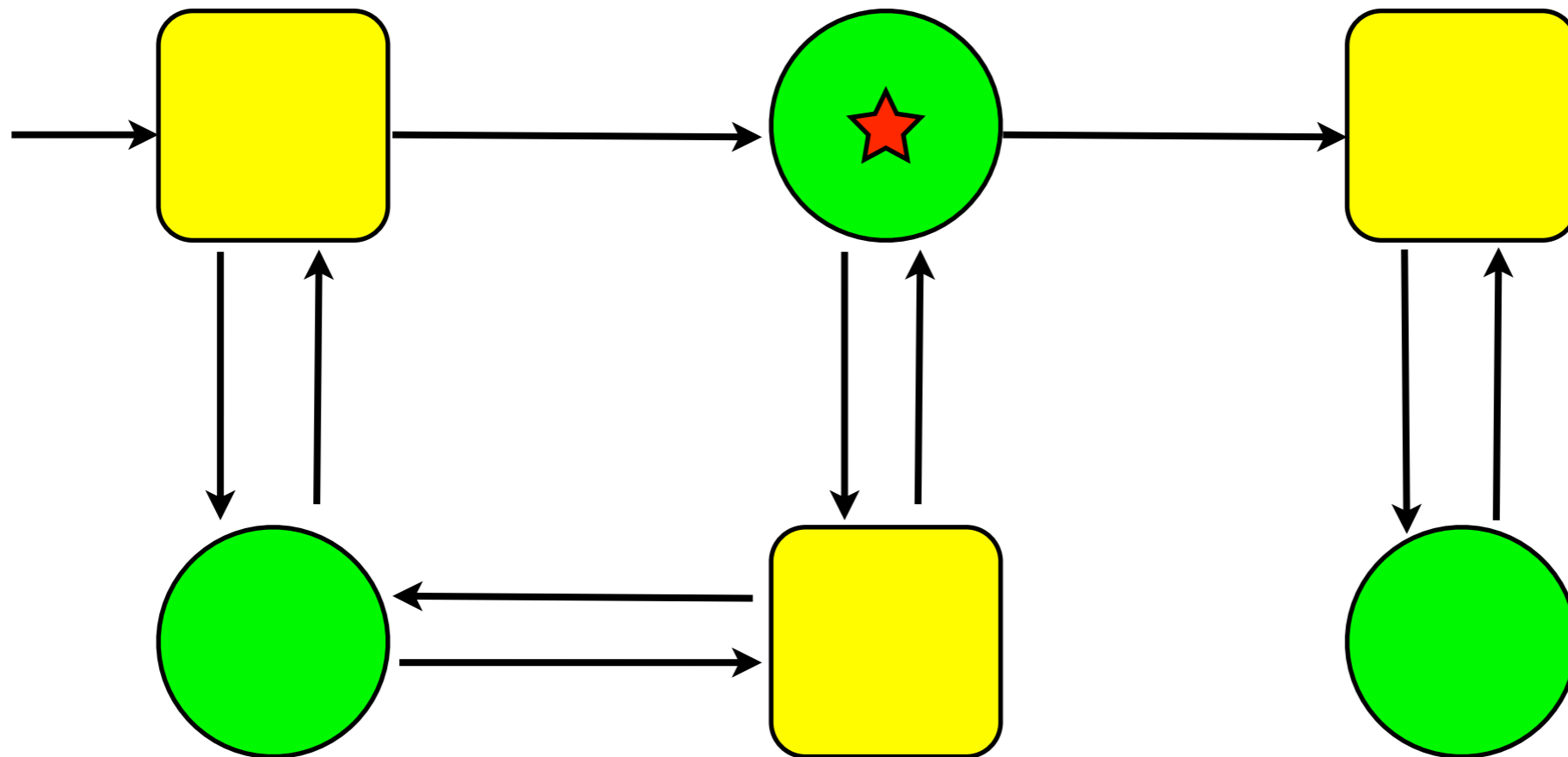
- LTL realizability \rightarrow UKCW realizability \rightarrow safety game



Protagonist's positions

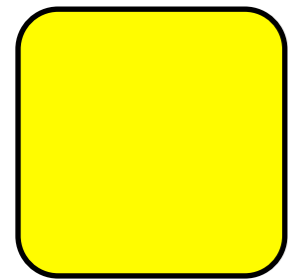


Adversary's positions

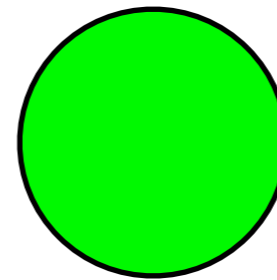


Safety Game

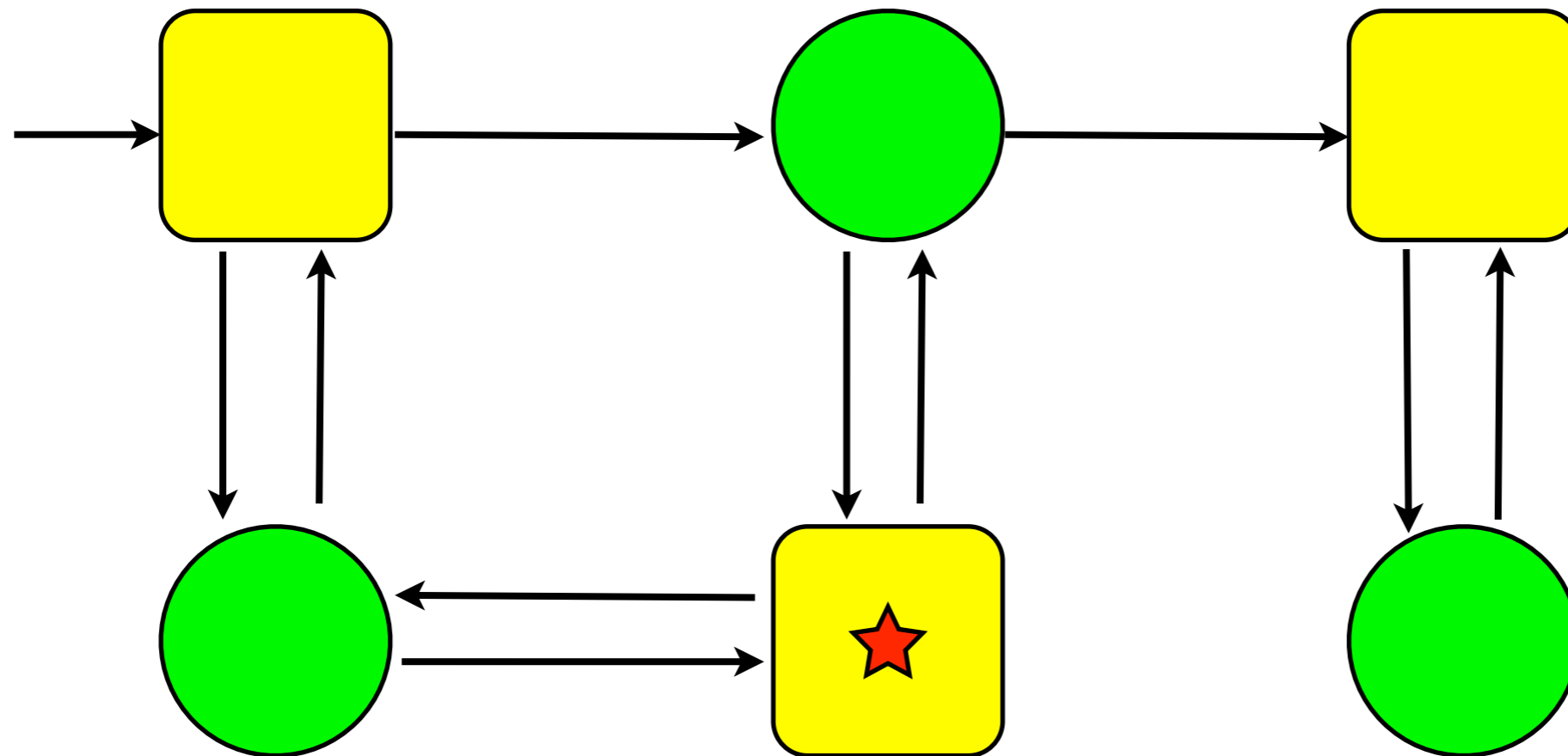
- LTL realizability \rightarrow UKCW realizability \rightarrow safety game



Protagonist's positions

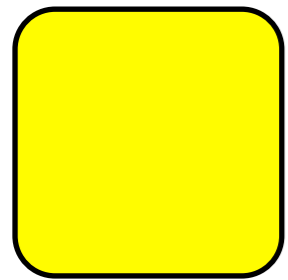


Adversary's positions

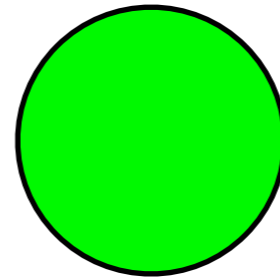


Safety Game

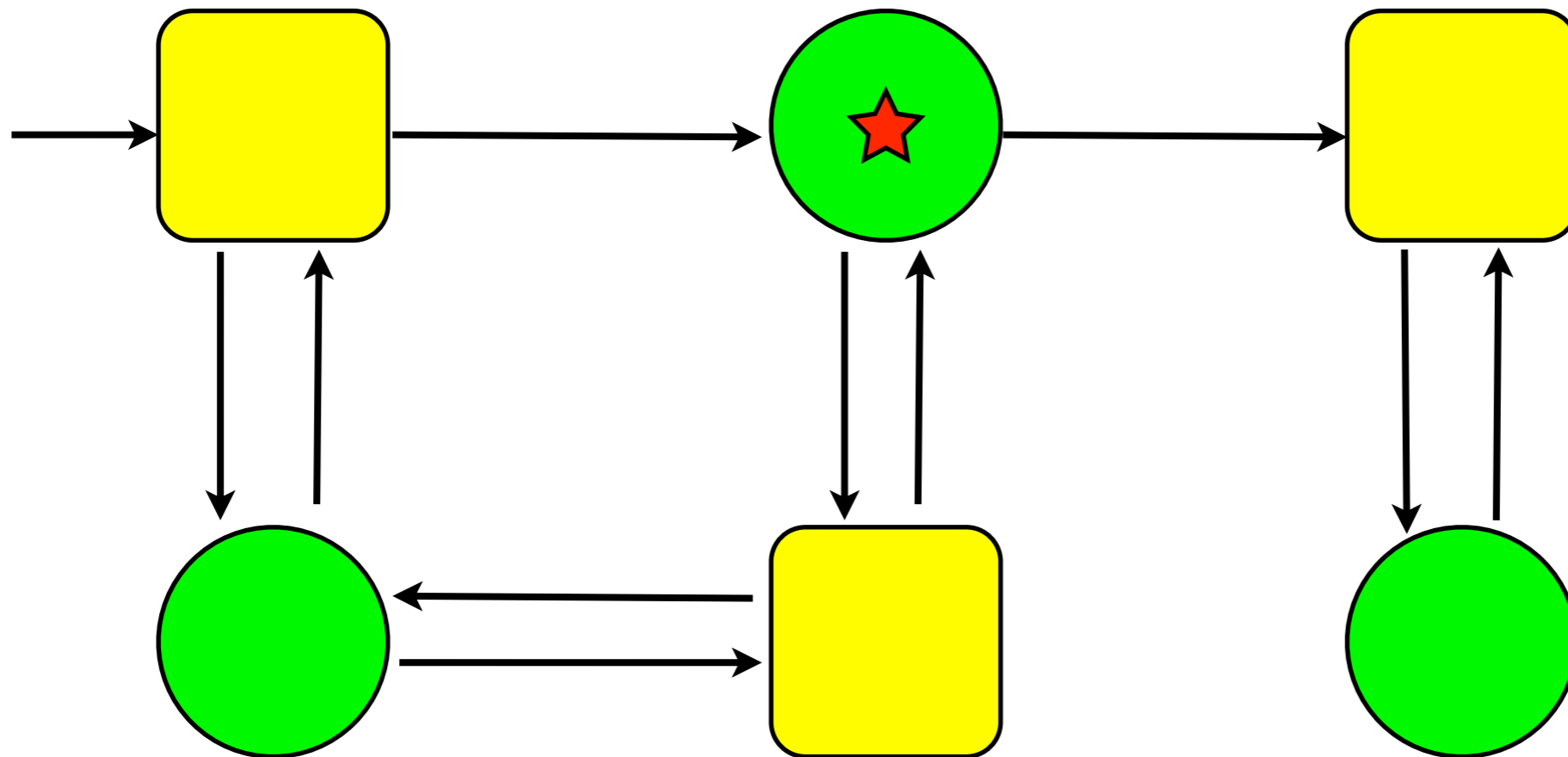
- LTL realizability \rightarrow UKCW realizability \rightarrow safety game



Protagonist's positions

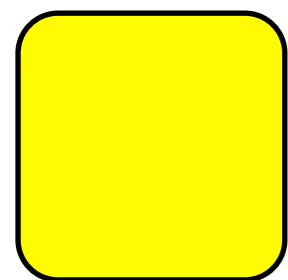


Adversary's positions

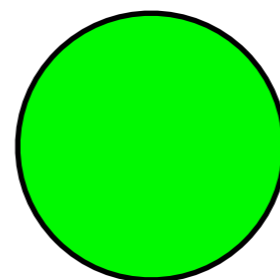


Safety Game

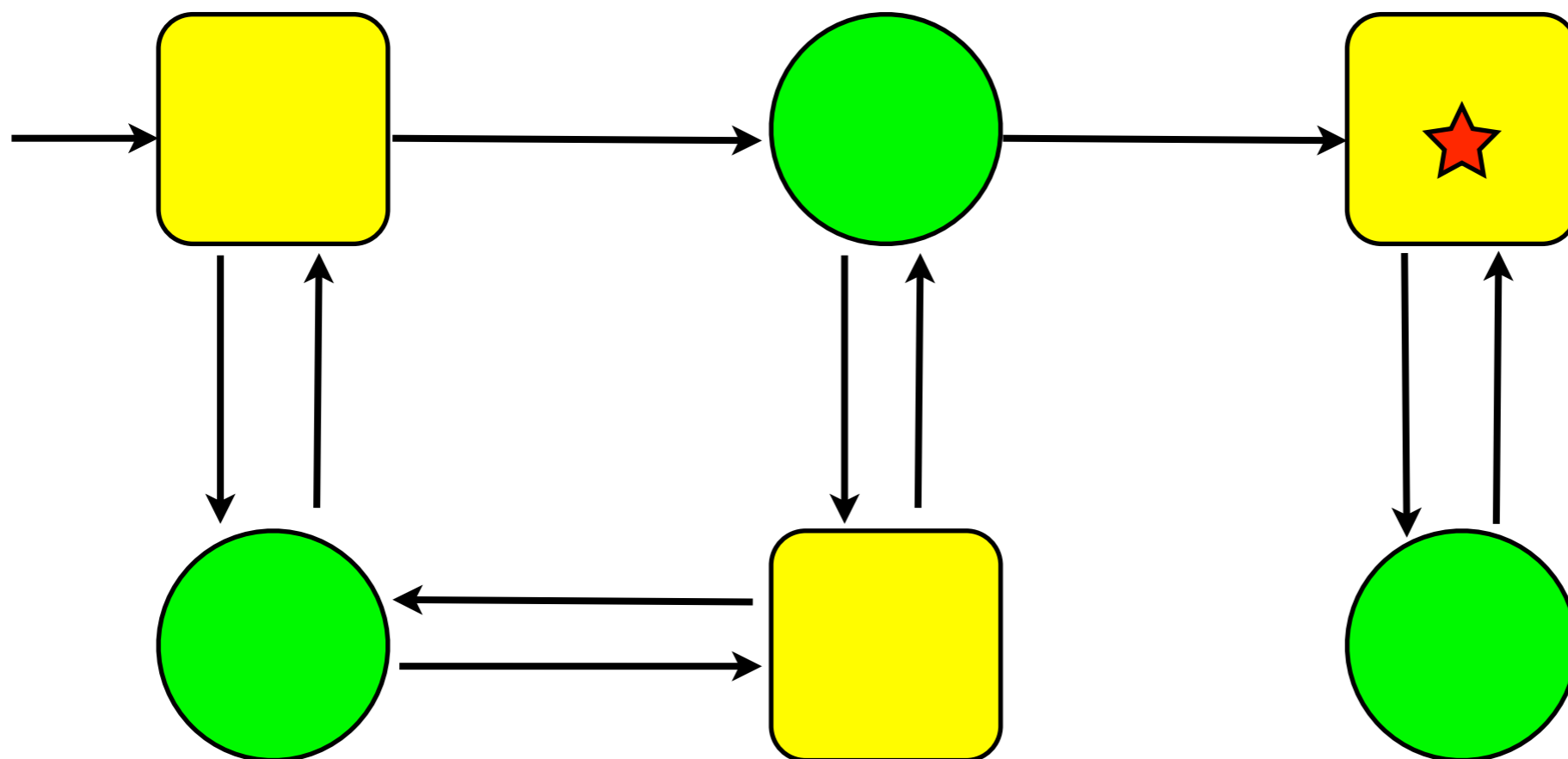
- LTL realizability \rightarrow UKCW realizability \rightarrow safety game



Protagonist's positions

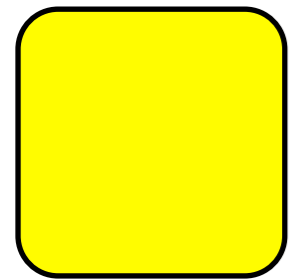


Adversary's positions

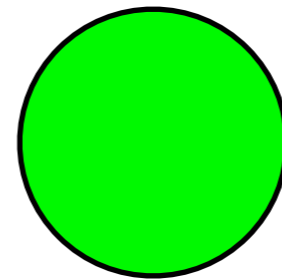


Safety Game

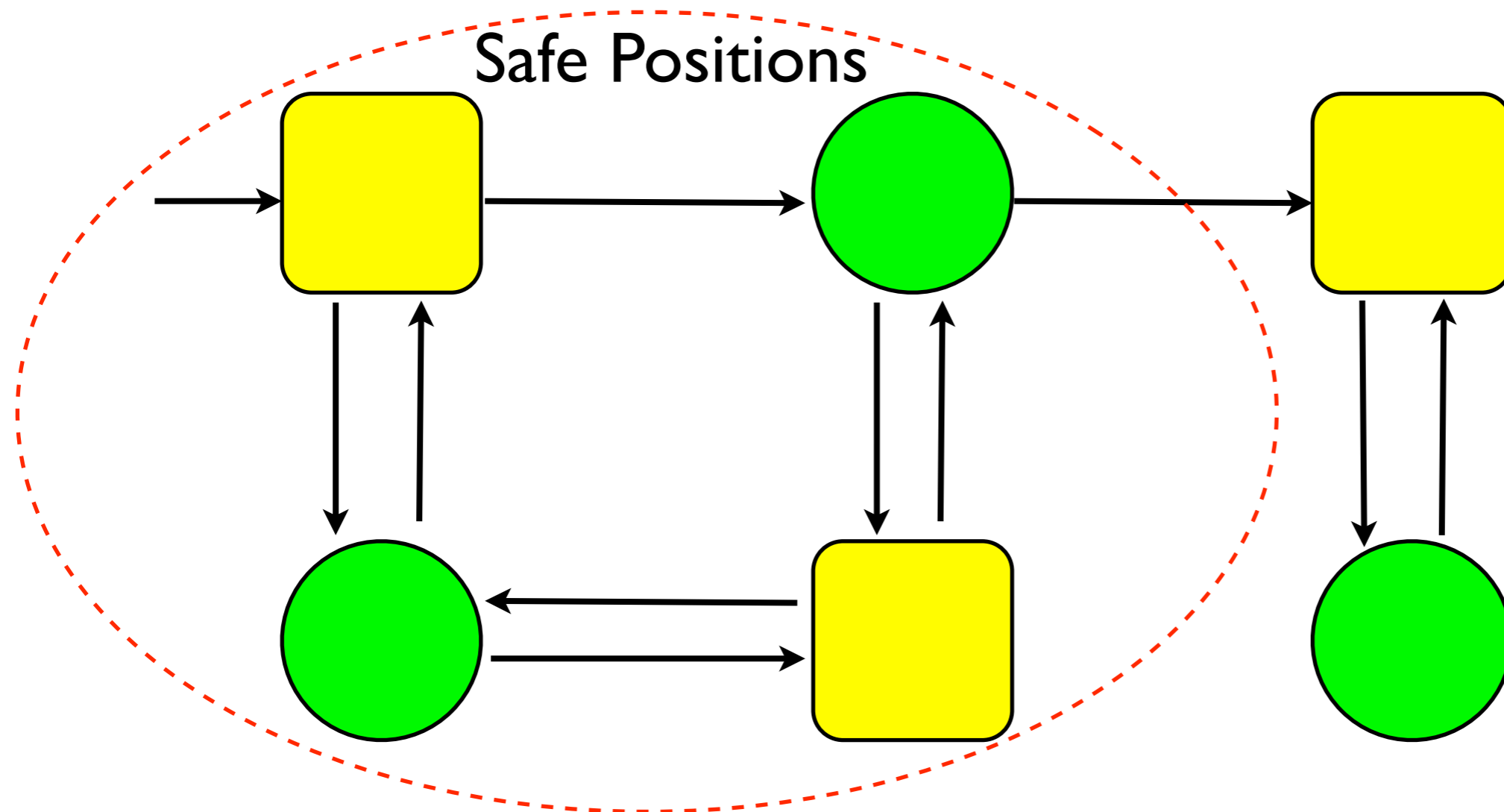
- LTL realizability \rightarrow UKCW realizability \rightarrow safety game



Protagonist's positions



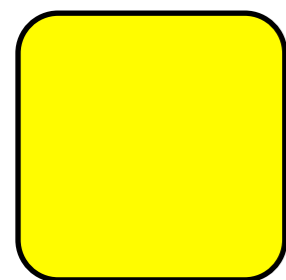
Adversary's positions



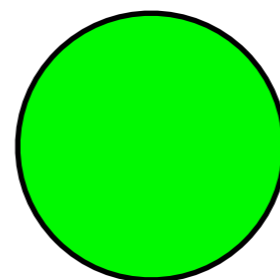
Protagonists wins if she has a strategy to keep the token in safe states

Safety Game

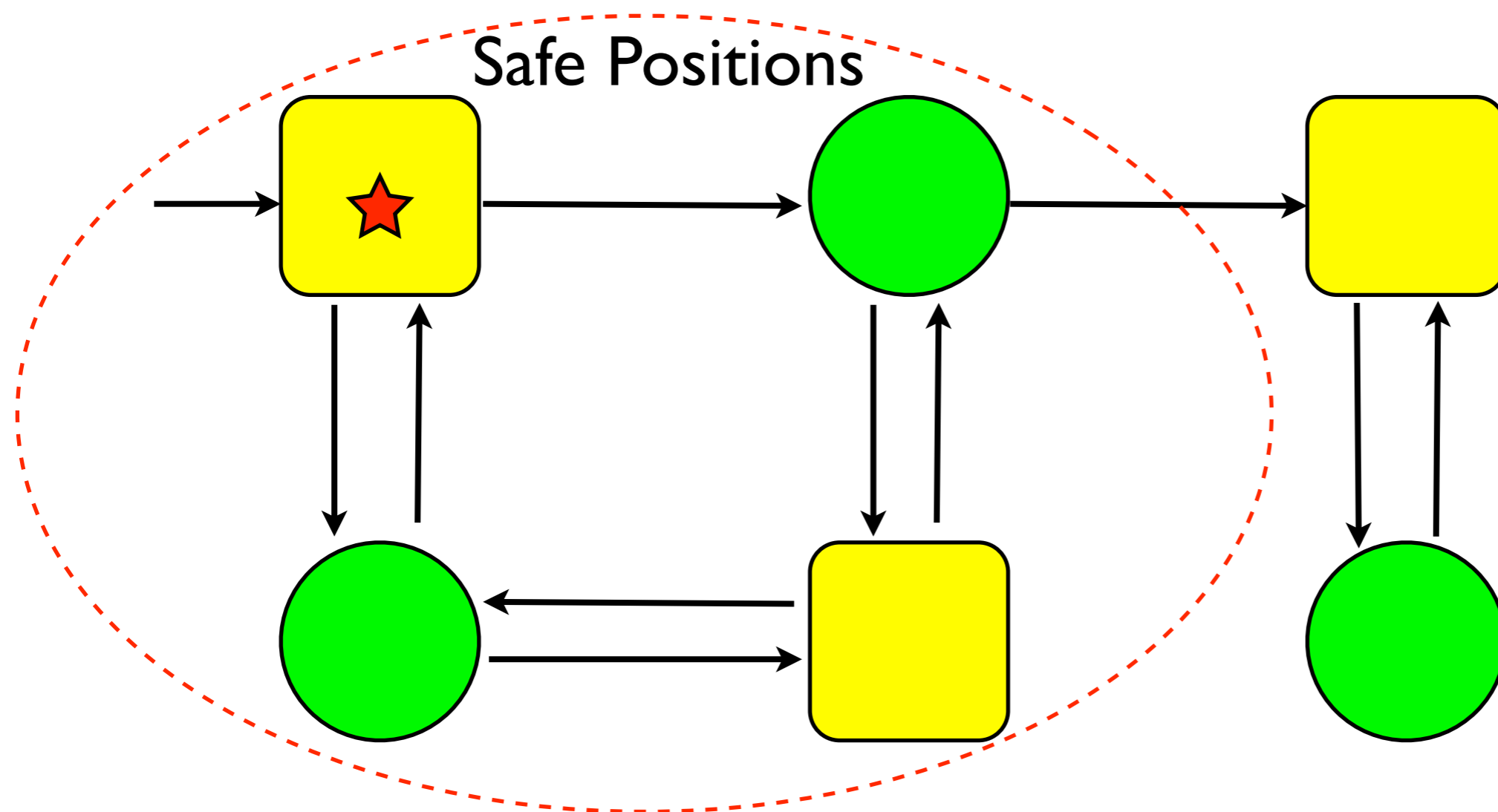
- LTL realizability \rightarrow UKCW realizability \rightarrow safety game



Protagonist's positions



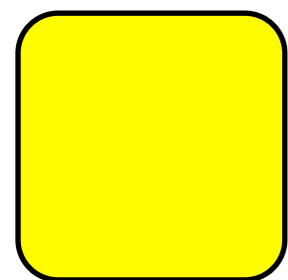
Adversary's positions



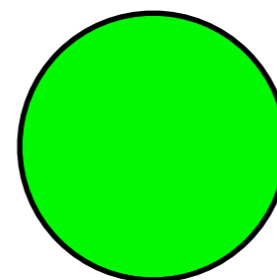
Protagonist wins if she has a strategy to keep the token in safe states

Safety Game

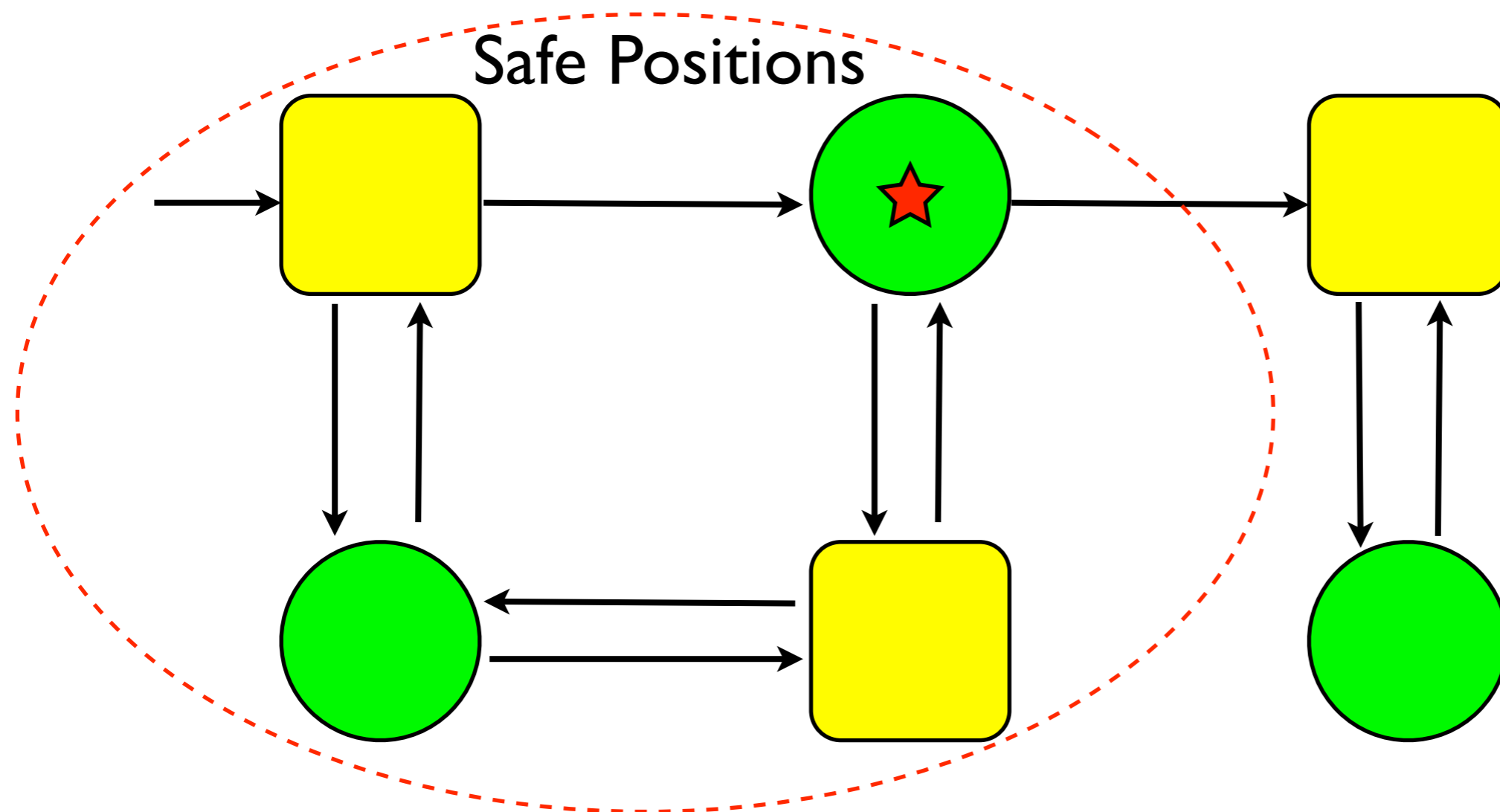
- LTL realizability \rightarrow UKCW realizability \rightarrow safety game



Protagonist's positions



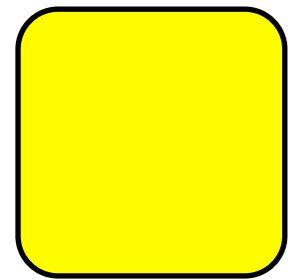
Adversary's positions



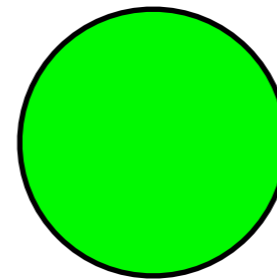
Protagonist wins if she has a strategy to keep the token in safe states

Safety Game

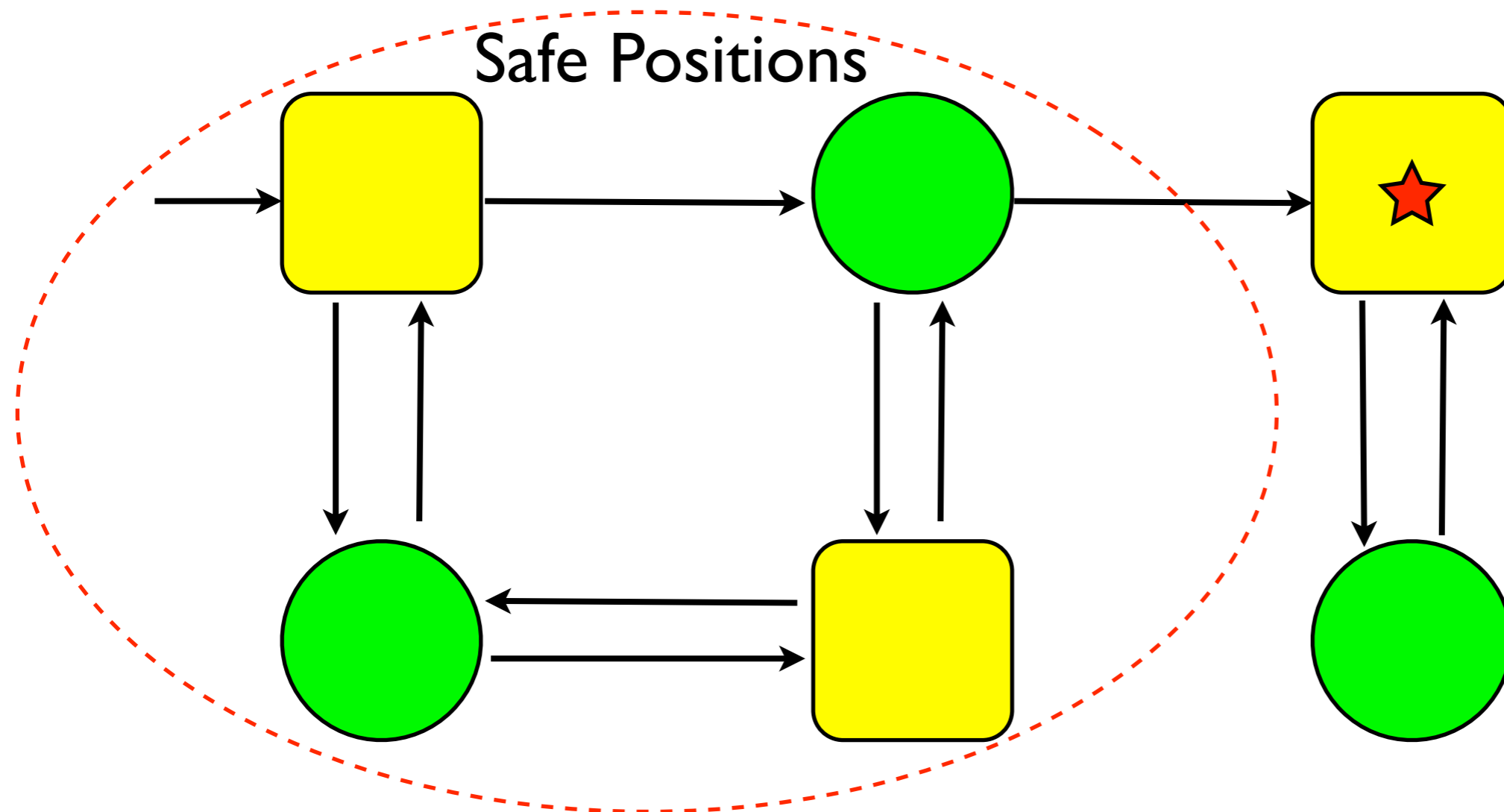
- LTL realizability \rightarrow UKCW realizability \rightarrow safety game



Protagonist's positions



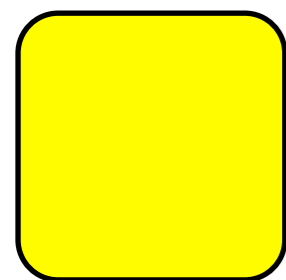
Adversary's positions



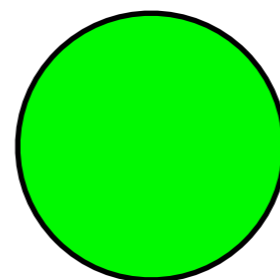
Protagonist wins if she has a strategy to keep the token in safe states

Safety Game

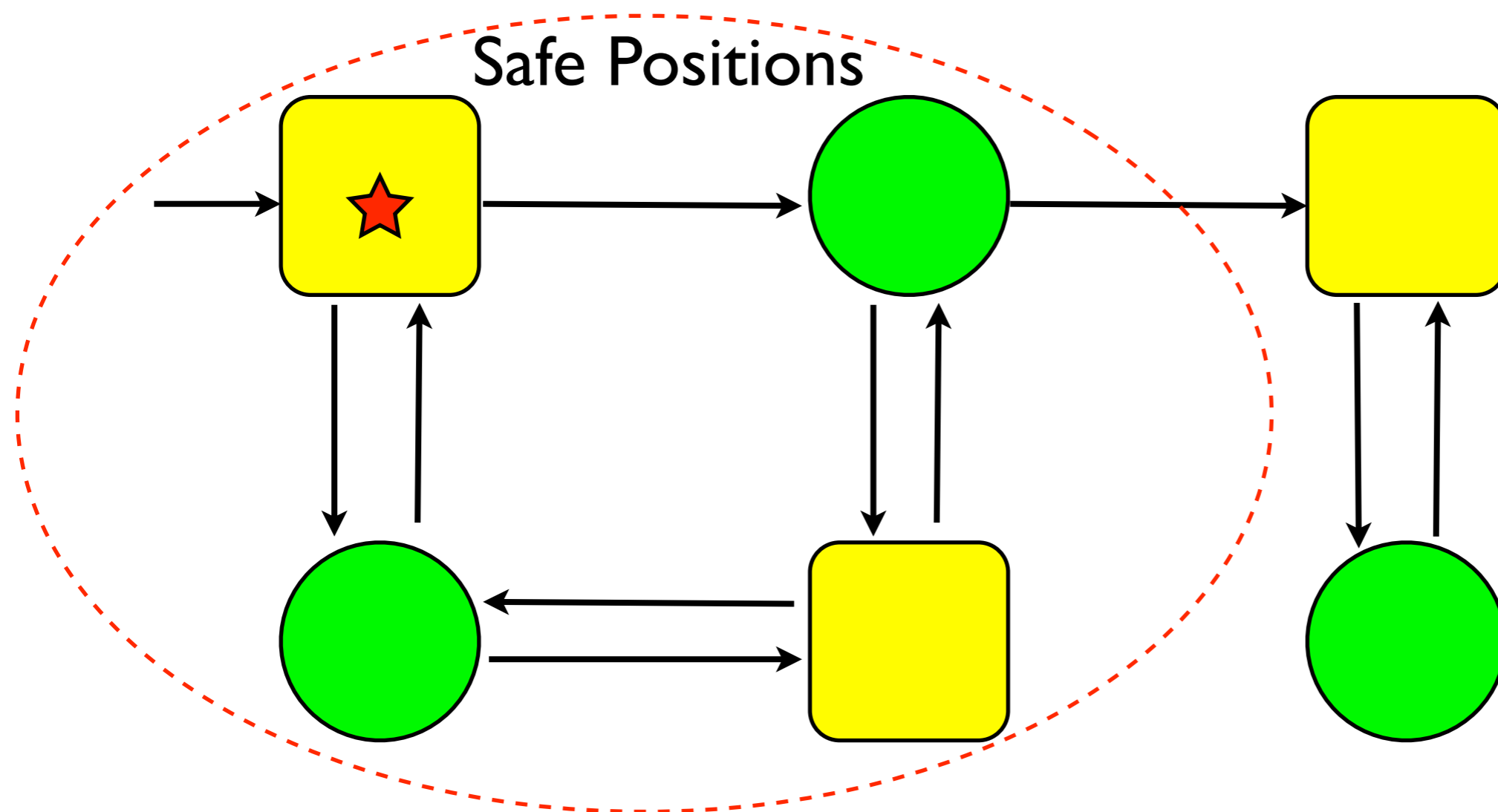
- LTL realizability \rightarrow UKCW realizability \rightarrow safety game



Protagonist's positions



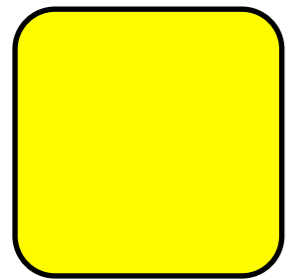
Adversary's positions



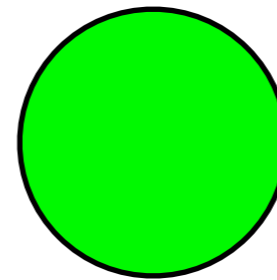
Protagonist wins if she has a strategy to keep the token in safe states

Safety Game

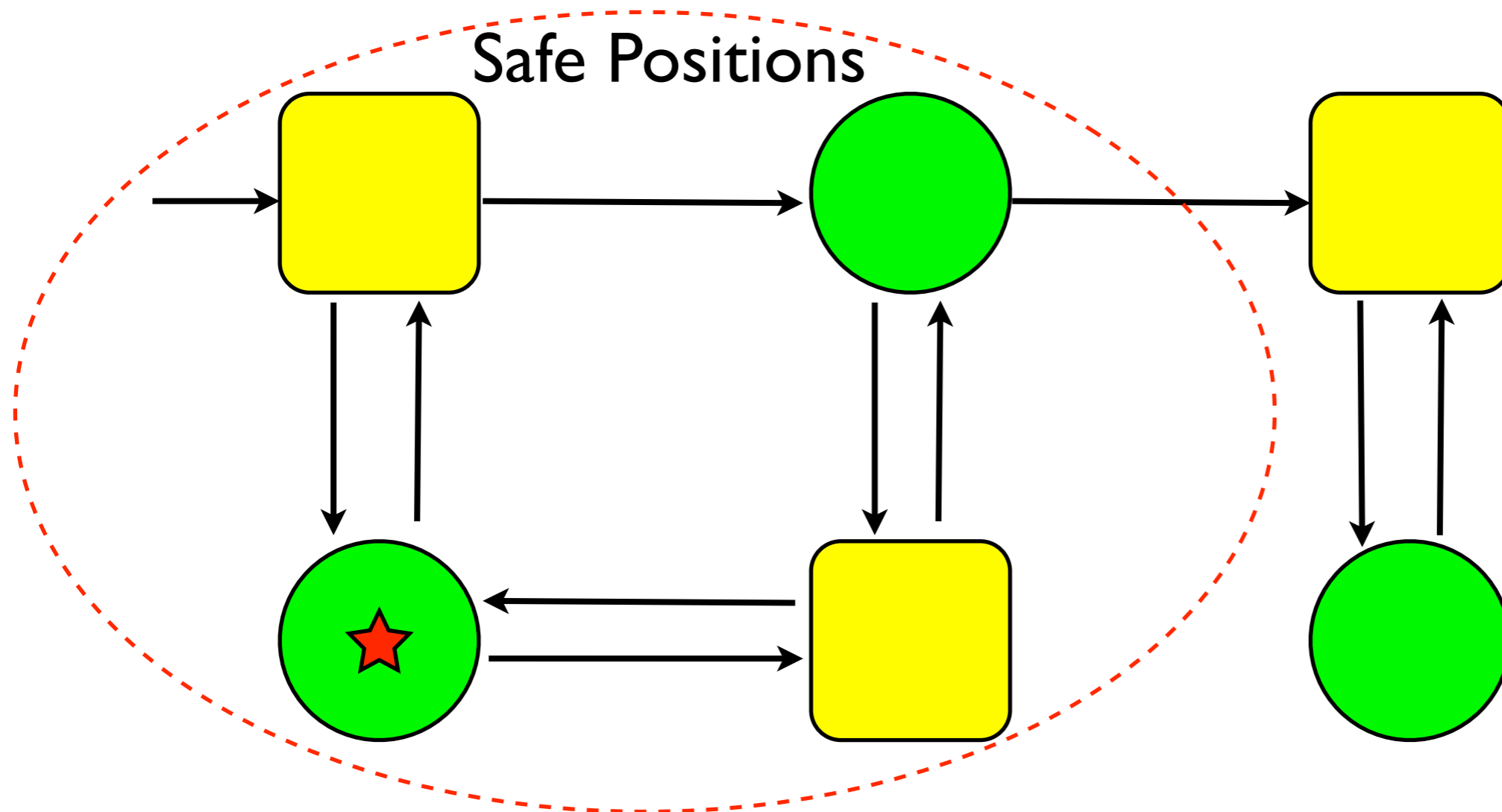
- LTL realizability \rightarrow UKCW realizability \rightarrow safety game



Protagonist's positions



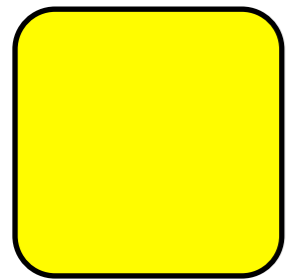
Adversary's positions



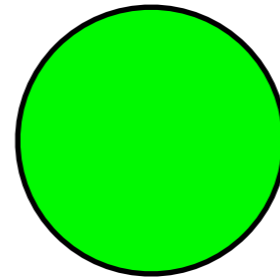
Protagonist wins if she has a strategy to keep the token in safe states

Safety Game

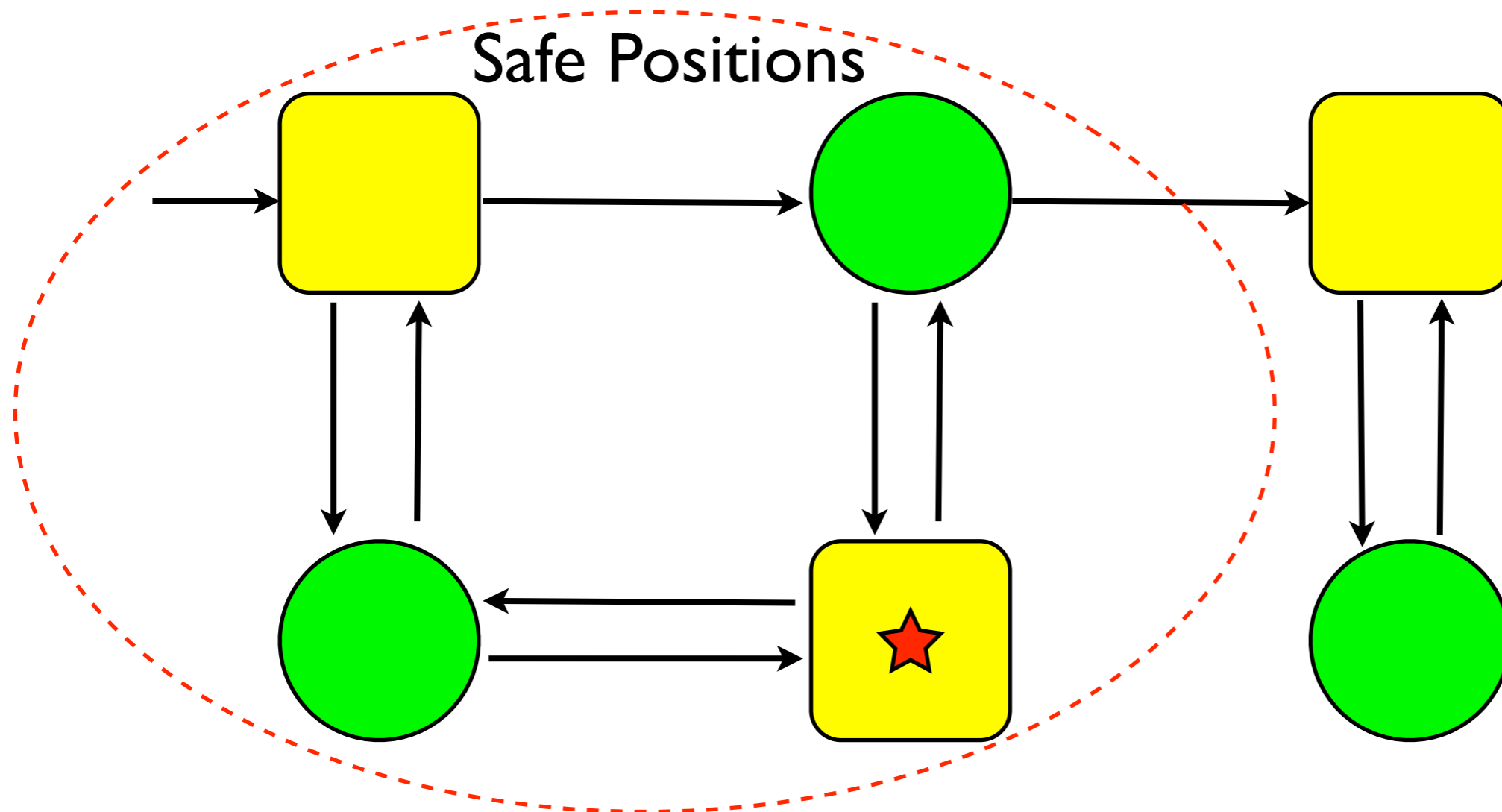
- LTL realizability \rightarrow UKCW realizability \rightarrow safety game



Protagonist's positions



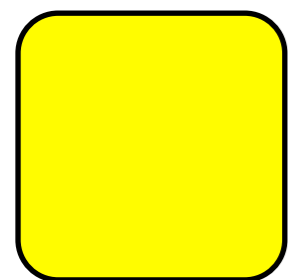
Adversary's positions



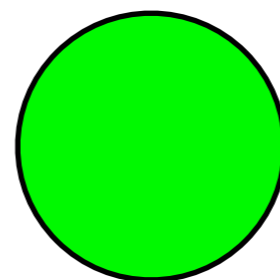
Protagonist wins if she has a strategy to keep the token in safe states

Safety Game

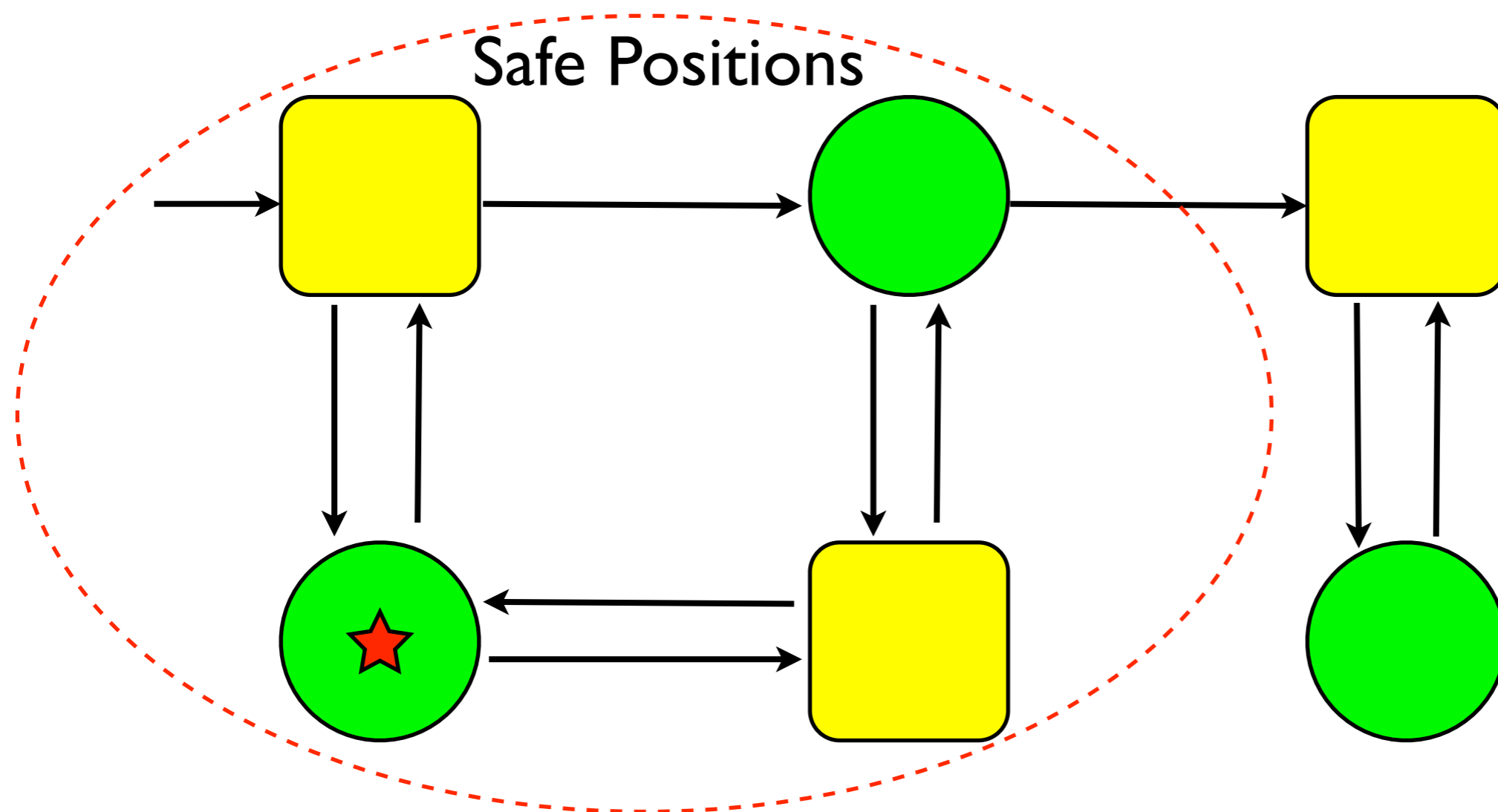
- LTL realizability \rightarrow UKCW realizability \rightarrow safety game



Protagonist's positions



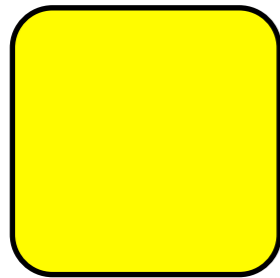
Adversary's positions



Protagonist wins if she has a strategy to keep the token in safe states

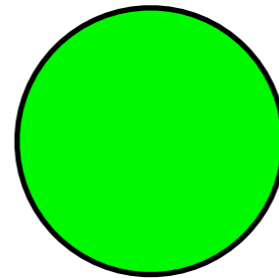
Reduction to a Safety Game

$$(A, K): \text{UKCW} \rightarrow \text{det}(A, K) = (\Sigma, \mathbb{F}, F_0, \mathbb{F}_{>K}, \Delta_d) \rightarrow G(A, K) = (\text{●} \text{●} v_0, T)$$



= \mathbb{F}

Model

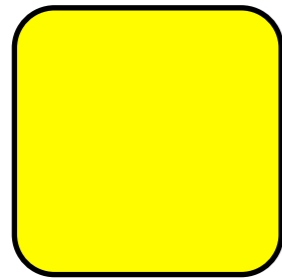


= $\mathbb{F} \times 2^{\circ}$

Environment

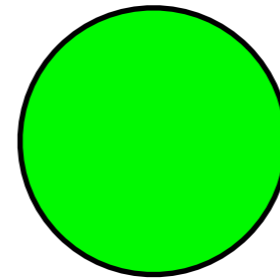
Reduction to a Safety Game

$$(A, K): \text{UKCW} \rightarrow \text{det}(A, K) = (\Sigma, \mathbb{F}, F_0, \mathbb{F}_{>K}, \Delta_d) \rightarrow G(A, K) = (\text{●}, \text{●}, v_0, T)$$



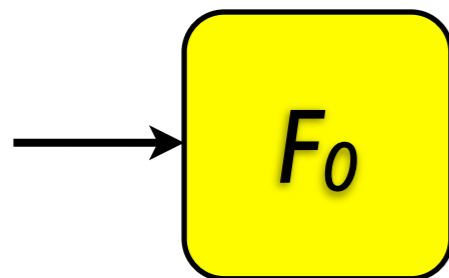
$$= \mathbb{F}$$

Model



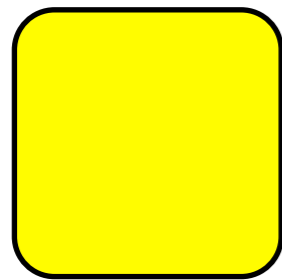
$$= \mathbb{F} \times 2^{\circ}$$

Environment



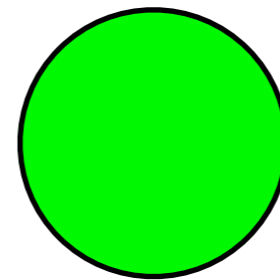
Reduction to a Safety Game

$$(A,K):UKCW \rightarrow det(A,K) = (\Sigma, \mathbb{F}, F_0, \mathbb{F}_{>K}, \Delta_d) \rightarrow G(A,K) = (\text{yellow circle}, \text{green circle}, v_0, T)$$



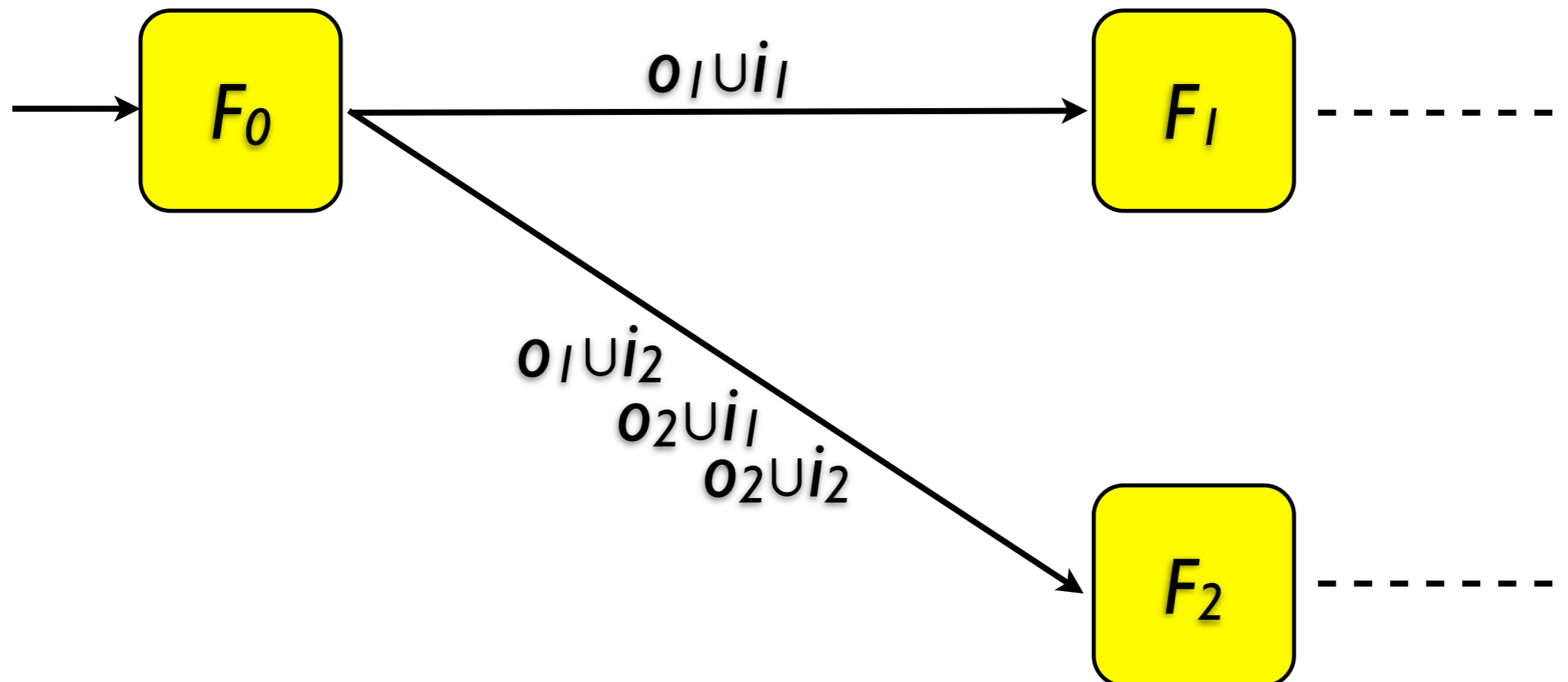
$= \mathbb{F}$

Model



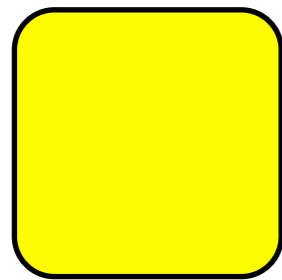
$= \mathbb{F} \times 2^{\mathbb{O}}$

Environment



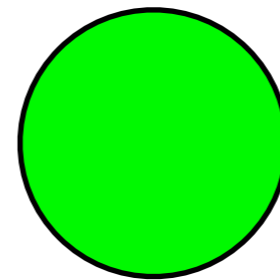
Reduction to a Safety Game

$$(A,K):UKCW \rightarrow det(A,K) = (\Sigma, \mathbb{F}, F_0, \mathbb{F}_{>K}, \Delta_d) \rightarrow G(A,K) = (\text{yellow circle}, \text{green circle}, v_0, T)$$



$= \mathbb{F}$

Model



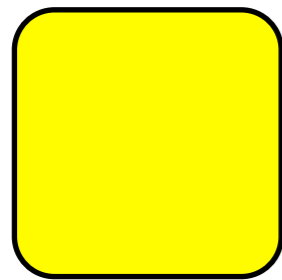
$= \mathbb{F} \times 2^O$

Environment



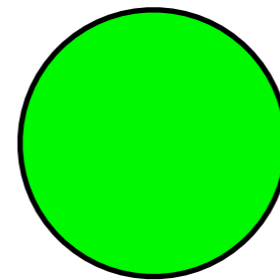
Reduction to a Safety Game

$$(A,K):UKCW \rightarrow det(A,K) = (\Sigma, \mathbb{F}, F_0, \mathbb{F}_{>K}, \Delta_d) \rightarrow G(A,K) = (\text{yellow circle}, \text{green circle}, v_0, T)$$



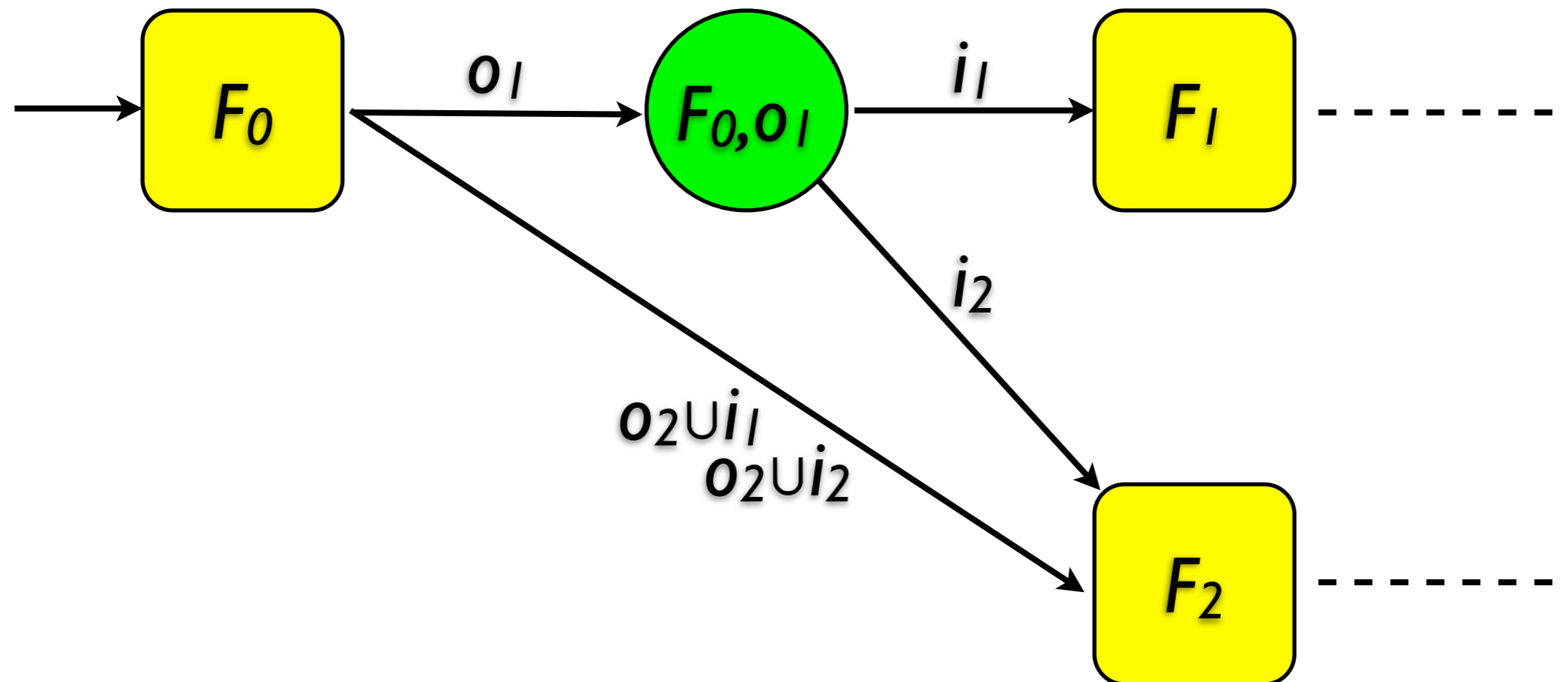
$= \mathbb{F}$

Model



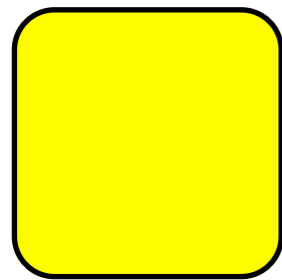
$= \mathbb{F} \times 2^O$

Environment



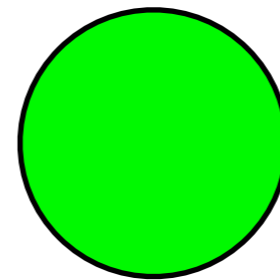
Reduction to a Safety Game

$$(A,K):UKCW \rightarrow det(A,K) = (\Sigma, \mathbb{F}, F_0, \mathbb{F}_{>K}, \Delta_d) \rightarrow G(A,K) = (\text{Model}, \text{Environment}, v_0, T)$$



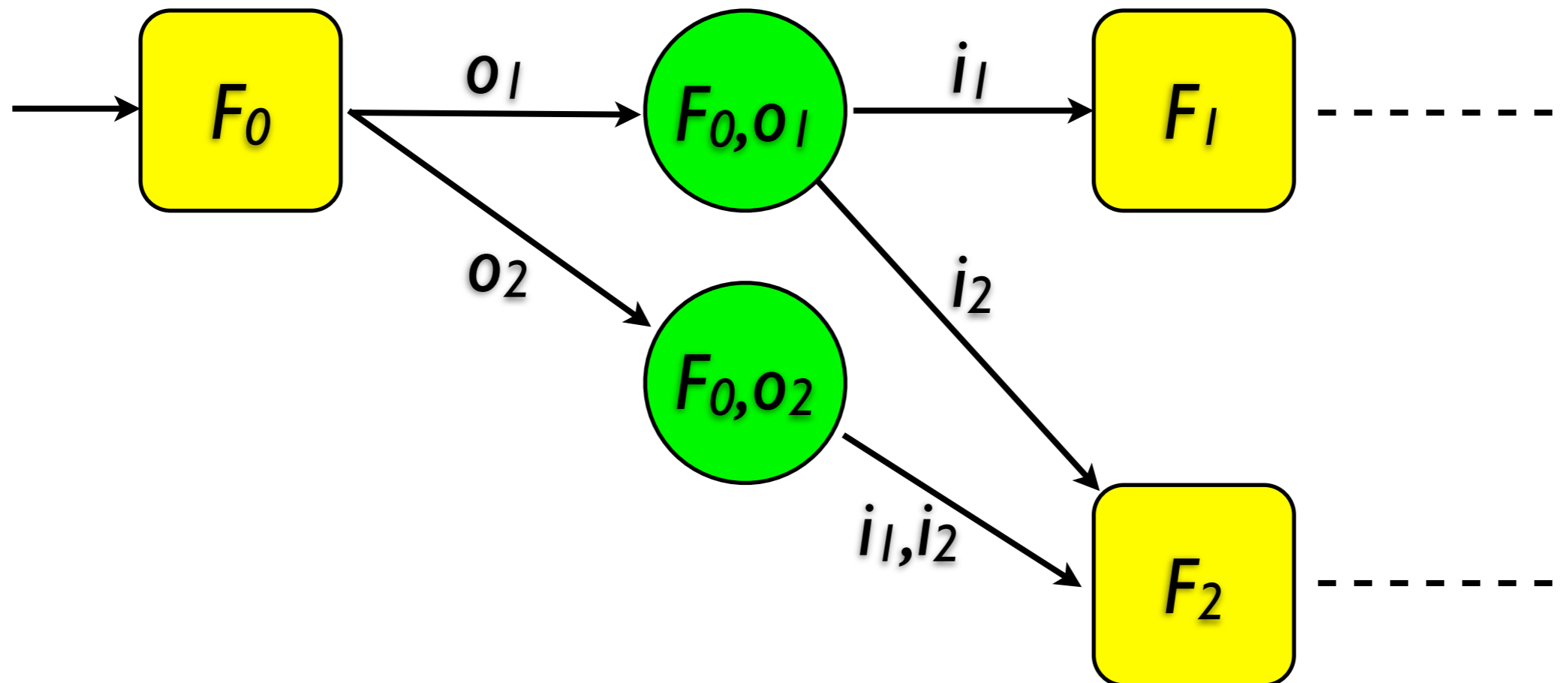
$= \mathbb{F}$

Model



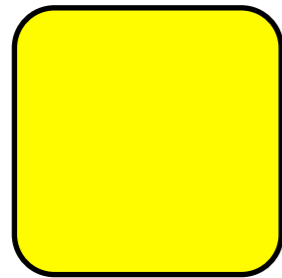
$= \mathbb{F} \times 2^{\mathbb{O}}$

Environment



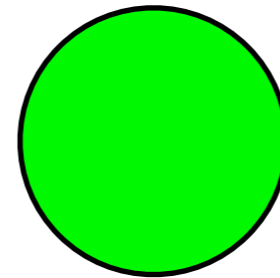
Reduction to a Safety Game

$$(A,K):UKCW \rightarrow det(A,K) = (\Sigma, \mathbb{F}, F_0, \mathbb{F}_{>K}, \Delta_d) \rightarrow G(A,K) = (\text{yellow circle}, \text{green circle}, v_0, T)$$



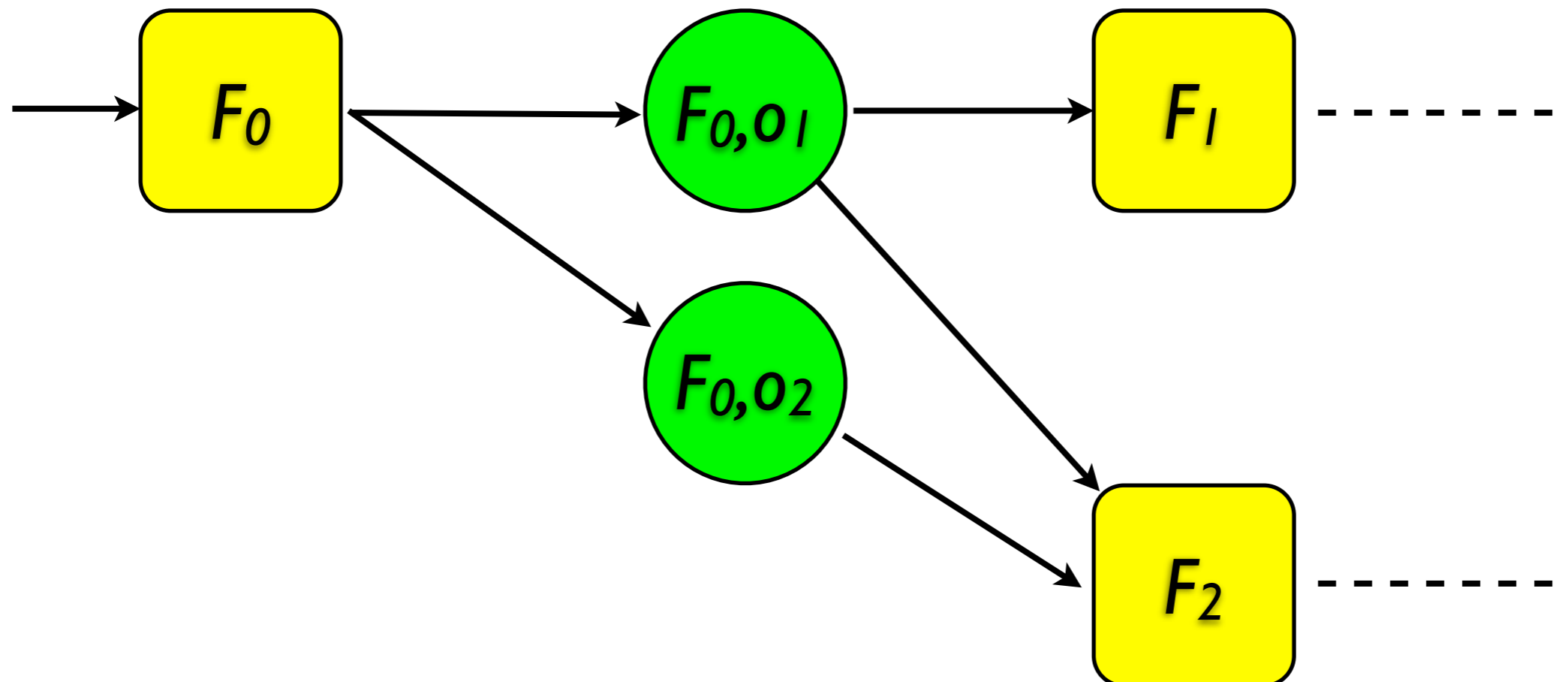
$= \mathbb{F}$

Model



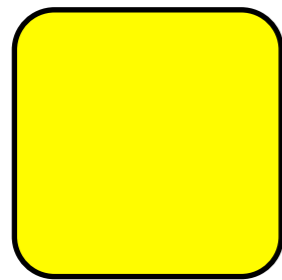
$= \mathbb{F} \times 2^O$

Environment



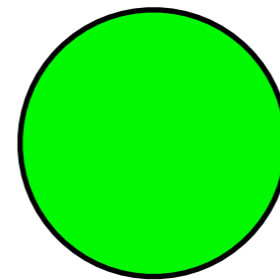
Reduction to a Safety Game

$$(A,K):UKCW \rightarrow det(A,K) = (\Sigma, \mathbb{F}, F_0, \mathbb{F}_{>K}, \Delta_d) \rightarrow G(A,K) = (\text{Model}, \text{Environment}, v_0, T)$$



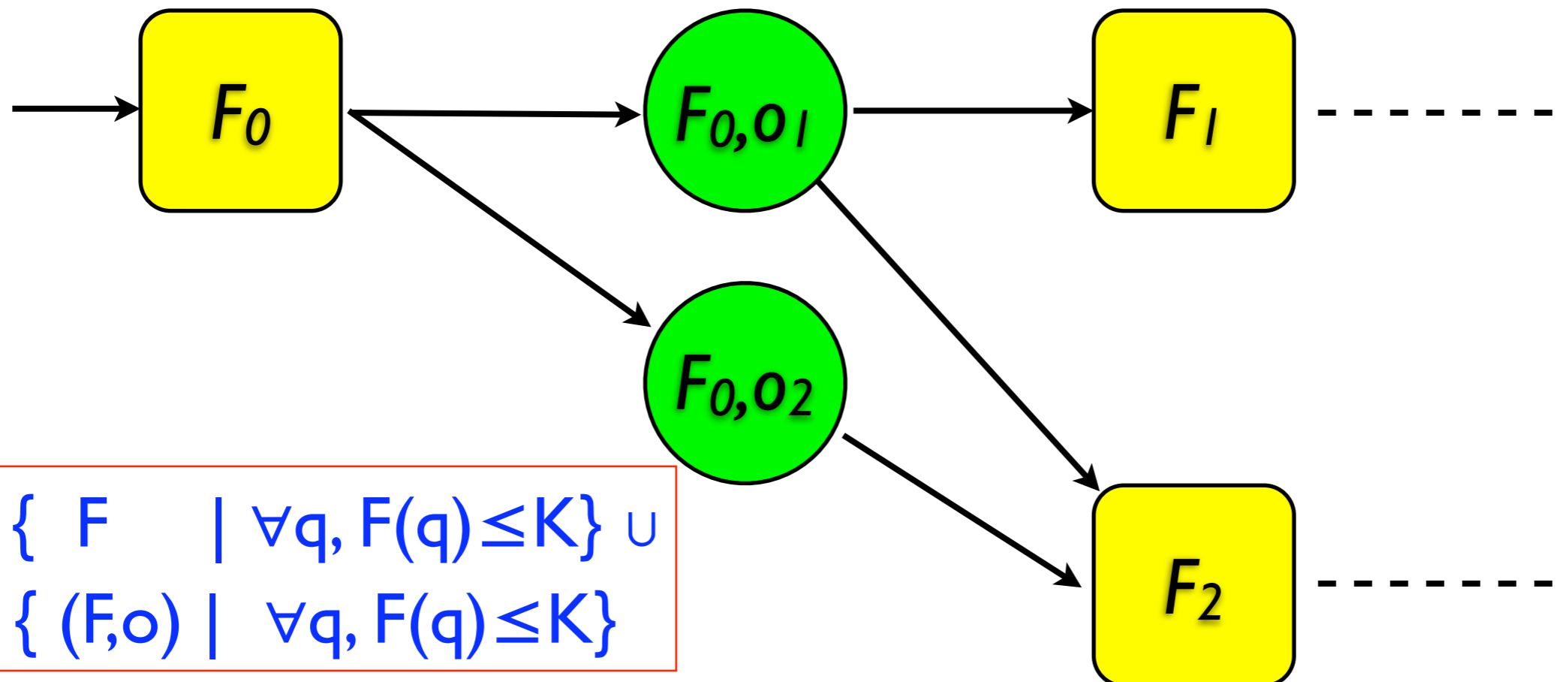
$$= \mathbb{F}$$

Model



$$= \mathbb{F} \times 2^O$$

Environment



$$\text{Safe} = \{ F \mid \forall q, F(q) \leq K \} \cup \{ (F,o) \mid \forall q, F(q) \leq K \}$$

Reduction Theorem

Let A be a UCW and $K = n(n^{2n+1} + 1)$.

A is realizable
iff

Protagonist (Model) has a winning strategy in $G(A, K)$

Reduction Theorem

Let A be a UCW and $K = n(n^{2n+1} + 1)$.

A is realizable

iff

Protagonist (Model) has a winning strategy in $G(A, K)$

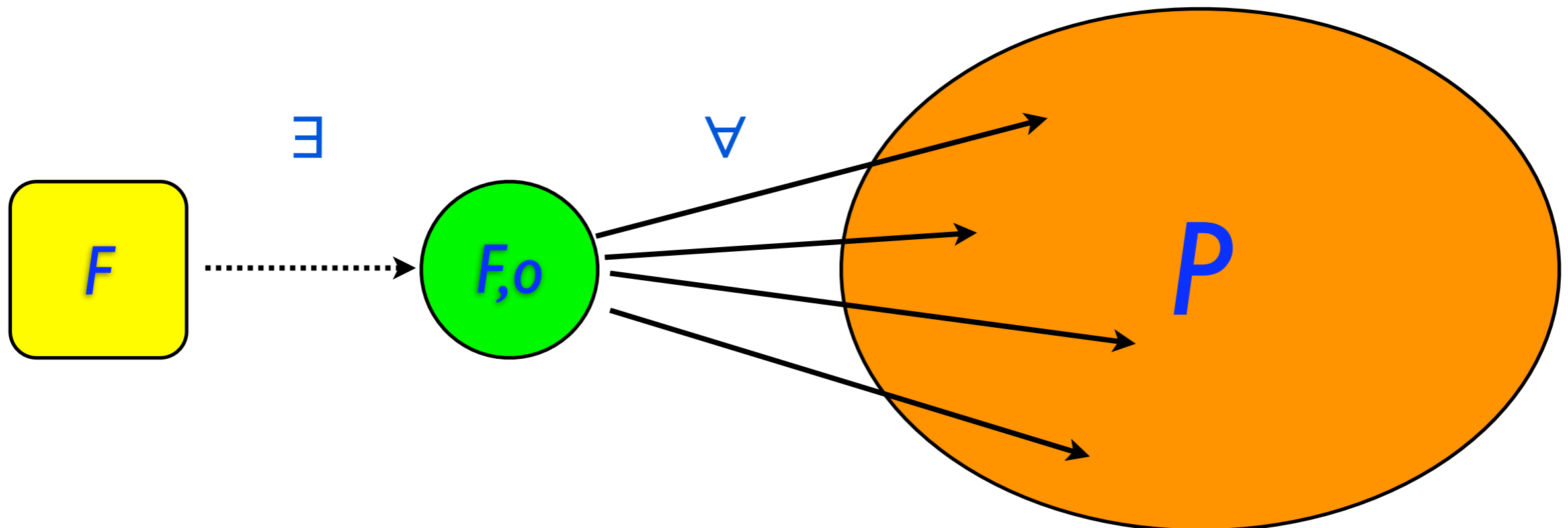
$G(A, K)$ is might be very large ... but we do not construct it

Outline

- Reduction to a safety game
- Antichain Algorithm
- Experimental Results

Controllable Predecessors

- $P \subseteq F$: subset of model positions
- safe controllable predecessors of P
 $Pre(P) = \{ F \mid \exists o \subseteq O, \forall F', ((F,o),F') \in T \Rightarrow F' \in P \} \cap Safe$

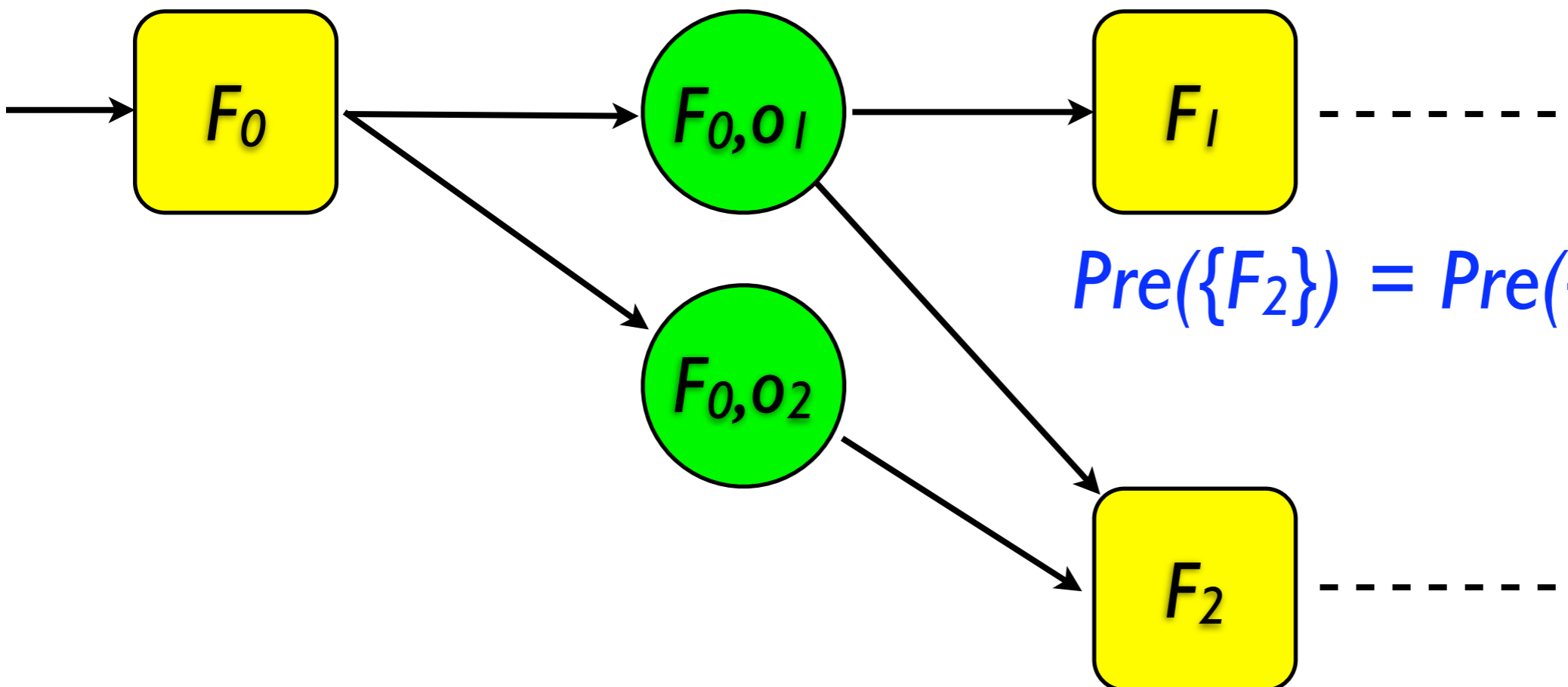


Controllable Predecessors

- $P \subseteq F$: subset of model positions

- safe controllable predecessors of P

$$Pre(P) = \{ F \mid \exists o \subseteq O, \forall F', ((F,o),F') \in T \Rightarrow F' \in P \} \cap Safe$$



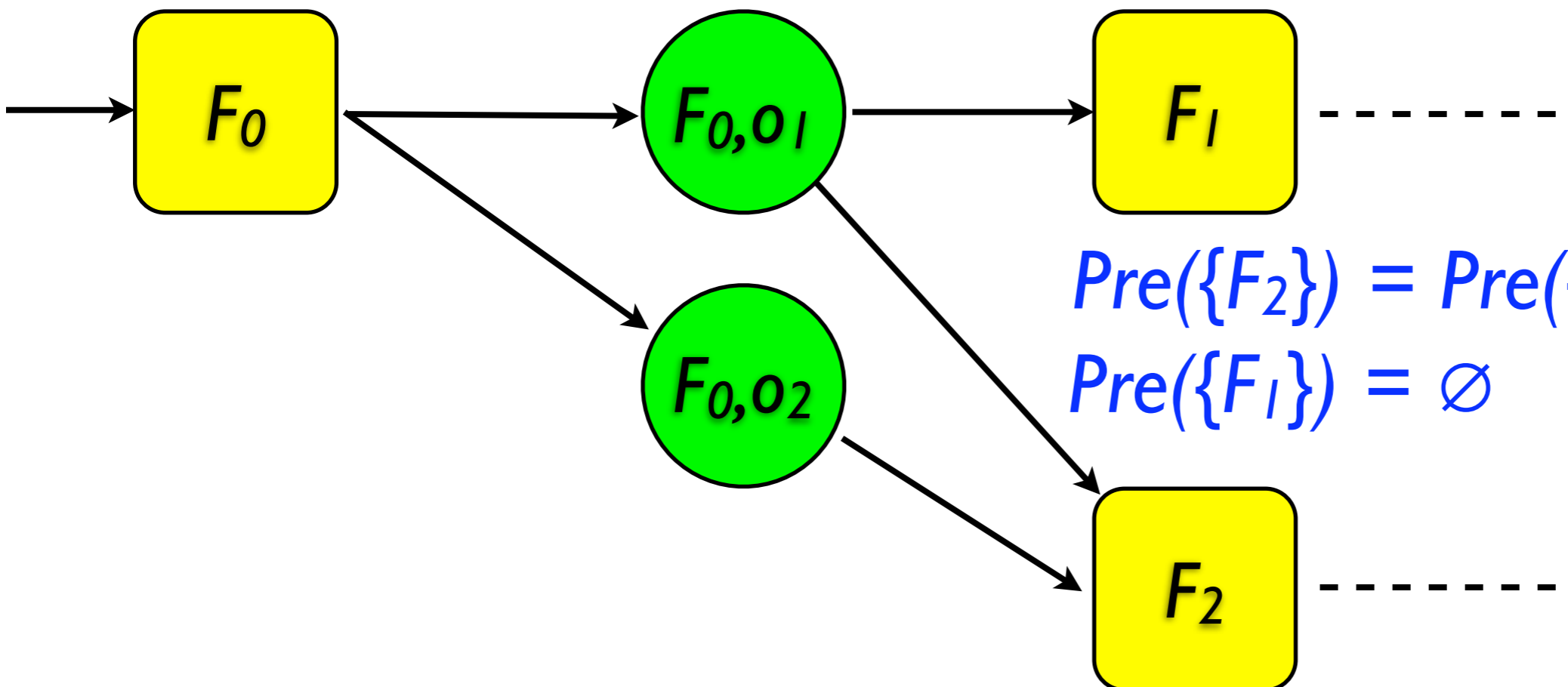
$$Pre(\{F_2\}) = Pre(\{F_1, F_2\}) = \{F_0\}$$

Controllable Predecessors

- $P \subseteq \mathbb{F}$: subset of model positions

- safe controllable predecessors of P

$$Pre(P) = \{ F \mid \exists o \subseteq O, \forall F', ((F, o), F') \in T \Rightarrow F' \in P \} \cap Safe$$



$$Pre(\{F_2\}) = Pre(\{F_1, F_2\}) = \{F_0\}$$

$$Pre(\{F_1\}) = \emptyset$$

Controllable Predecessors

- $P \subseteq \mathbb{F}$: subset of model positions
- safe controllable predecessors of P
 $Pre(P) = \{ F \mid \exists o \subseteq O, \forall F', ((F, o), F') \in T \Rightarrow F' \in P \} \cap Safe$
- $Pre(.)$ is monotonic over the lattice $(2^{\mathbb{F}}, \subseteq)$
- greatest fixpoint $Pre^* = \text{winning region}$

Controllable Predecessors

- $P \subseteq \mathbb{F}$: subset of model positions
- safe controllable predecessors of P
 $Pre(P) = \{ F \mid \exists o \subseteq O, \forall F', ((F, o), F') \in T \Rightarrow F' \in P \} \cap Safe$
- $Pre(\cdot)$ is monotonic over the lattice $(2^{\mathbb{F}}, \subseteq)$
- greatest fixpoint $Pre^* = \text{winning region}$

Model has a winning strategy in $G(A, K)$ iff $F_0 \in Pre^*$

Antichain fixpoint computation

1. F can be partially ordered by $F \leq_d F'$ iff $\forall q: F(q) \leq F'(q)$
2. if Model wins from F' , she also wins from F
3. $Pre(\cdot)$ preserve *downward*-closed sets
4. represent each (downward) set of the fixpoint computation by its maximal elements

Realizability Checking

- **Prop.** for all UCW A , all $K_1 \leq K_2$, $L(A, K_1) \subseteq L(A, K_2) \subseteq L(A)$.
- \Rightarrow Incremental Realizability Checking Algorithm:

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

Realizability Checking

- **Prop.** for all UCW A , all $K_1 \leq K_2$, $L(A, K_1) \subseteq L(A, K_2) \subseteq L(A)$.
- \Rightarrow Incremental Realizability Checking Algorithm:

1. **Input:** an LTL formula Φ , a partition I, O

2. $A \leftarrow$ UCW with n states equivalent to Φ

3. $K \leftarrow n(n^{2n+1} + 1)$

Not reasonable for unrealizable specifications

4. **for** $k=1 \dots K$ **do**
5. **if** Model wins then $G(A, k)$ **return** realizable

6. **endfor**

7. **return** unrealizable

Unrealizability Checking

As a consequence of the determinacy theorem for Borel games [Martin,75]:

ϕ is unrealizable for the Model iff $\neg\phi$ is realizable for the environment.

Consequence: we can adapt the previous algorithm to test unrealizability. In practice, realizability or unrealizability are obtained for small values of k .

Outline

- Reduction to a safety game
- Antichain Algorithm
- Experimental Results



Implementation

- realizability and unrealizability checking
- if the spec is realizable, output a Moore machine that realizes it
- compared to Lily [Jobstmann, Bloem,06]
- implemented in Perl (as Lily)
- formula to automata construction borrowed from Lily (based on Wring [Somenzi, Bloem])

On Realizable Lily's Examples

Φ	Automata Construction (s)	$ \Delta / Q $	Lily (s)	Acacia (s)	k	AC max
F3	0.97	28/10	0.3	0.01	0	2
F6	3.19	63/19	0.91	0.03	0	3
F8	0.07	6/3	0.02	0.01	0	1
F14	0.26	13/5	0.07	0.01	1	3
F16	0.57	26/8	1.45	7.89	3	104
F18	0.92	31/8	2.35	0.89	2	19
F19	0.75	17/7	4.05	0.03	2	5
F21	15.61	72/13	1.88	0.40	0	25
F22	25.28	115/19	1.21	0.1	1	7
E4	0.1	52/5	0.98	0.01	1	1
E5	0.12	117/6	16	0.06	1	1
E6	0.2	262/7	600	0.51	1	1
E7	0.39	583/8	>1h	6	1	1

Conclusion

- novel and direct Safraless approach to LTL realizability
- without optimizations, the performance results are already very encouraging
- available at www.antichains.be/acacia

Future Work

- optimizations (symbolic transitions, compositionnality, pruning)
- avoid automata construction to handle bigger formulas
- Safraless proof

Thank You