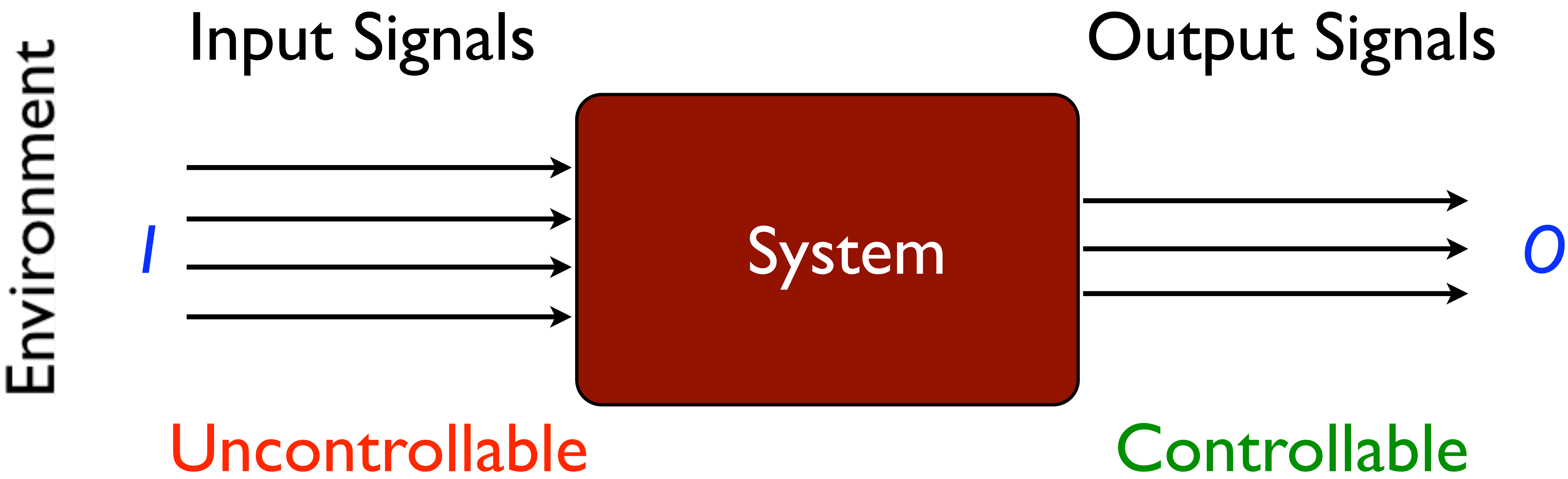
An Antichain Algorithm for LTL Realizability

Emmanuel Filiot
joint with Naiyong Jin and Jean-François Raskin

Université Libre de Bruxelles

GAMES 2009, Udine

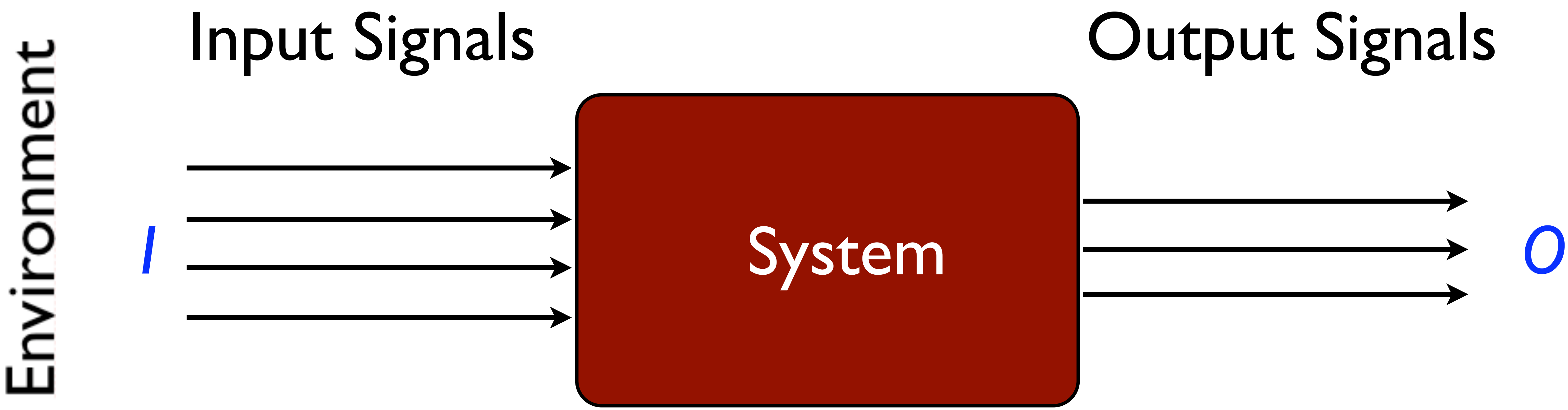
LTL Realizability



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

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

LTL Realizability



Uncontrollable   Controllable

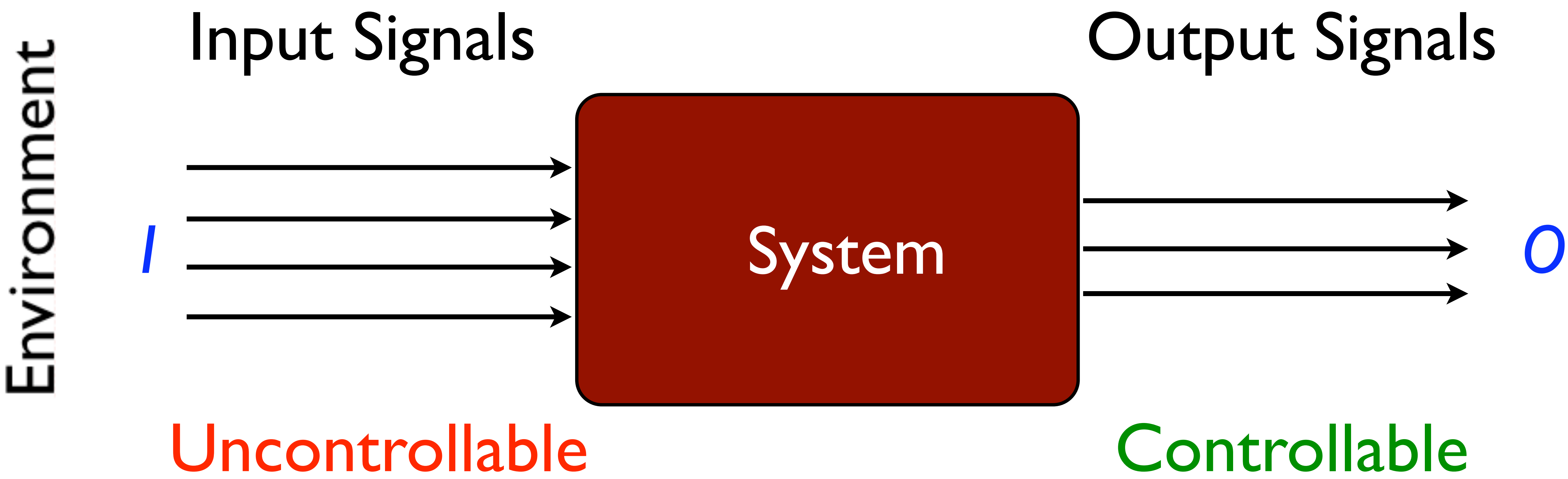
synchronous execution= infinite words over $\Sigma = 2^{I \cup O}$

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

Realizability Problem: Given $\Phi \in$ *LTL on atomic propositions* $I \cup O$

$\exists M \in$ System, $\forall e \in$ Exec, e satisfies Φ ?

LTL Realizability

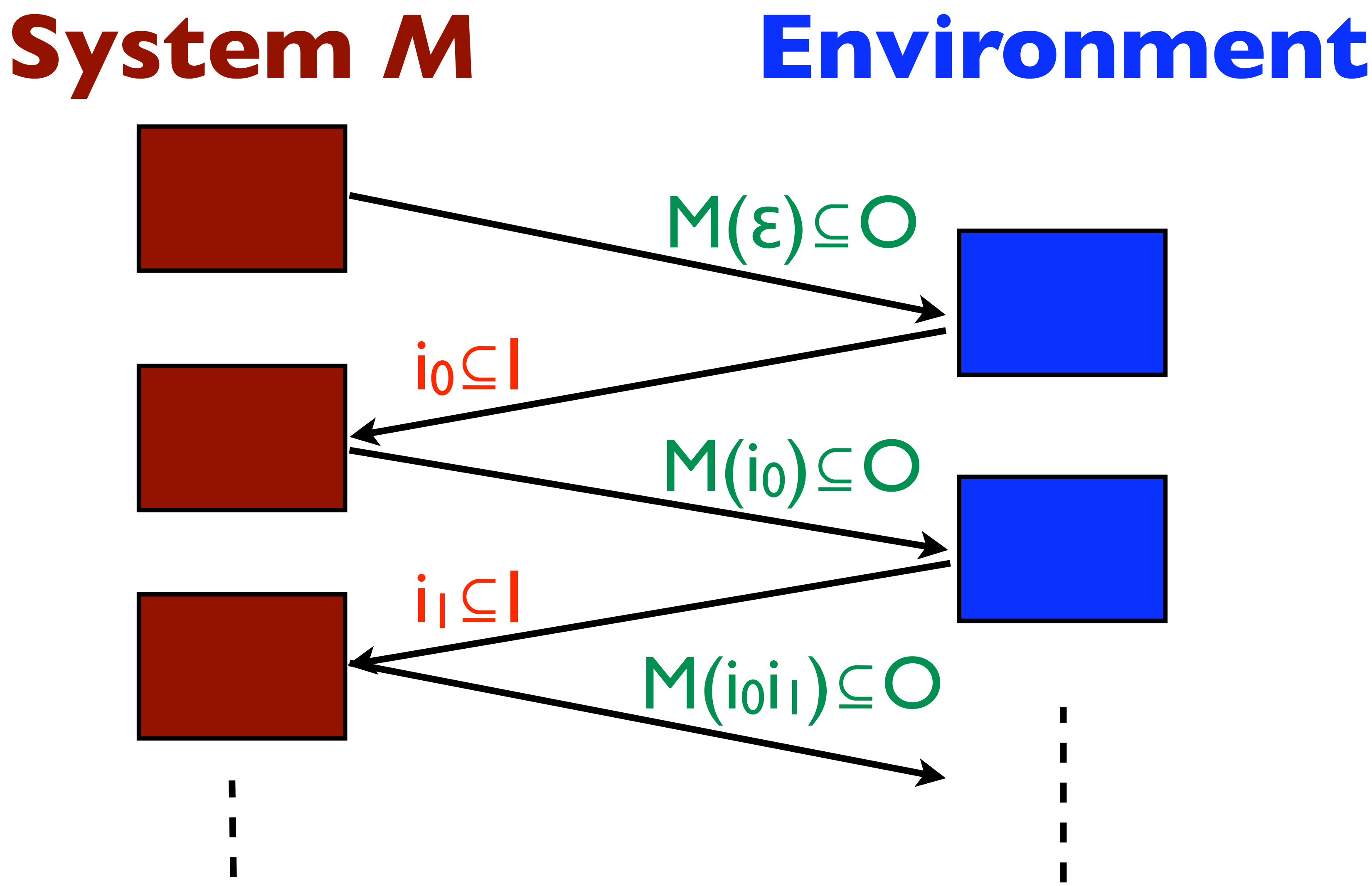


synchronous execution= infinite words over $\Sigma = 2^{I \cup O}$

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

Synthesis Problem: generate such a system

Realizability as an ∞-game



- The system wins the game if the play $(M(\varepsilon) \cup i_0)(M(i_0) \cup i_1)(M(i_0 i_1) \cup i_2)\ldots$ satisfies ϕ
- system ~ *strategy* $(2^I)^* \rightarrow 2^O$

Examples

- $I = \{ i \}$, $O = \{ o \}$

Formula	Satisfiable	Realizable	Strategy
$o \, U \, i$	✔	✘	environment never asserts i
$\Diamond i \rightarrow o \, U \, i$	✔	✔	system always asserts o

Existing Procedures

- 2ExpTime-Complete [Rosner, 92]
- "classical" procedure [Pnueli, Rosner, 89]

LTL $\xrightarrow{2^{O(n)}}$ Büchi Word Automata $\xrightarrow{2^{O(m \log m)}}$ **Det**. Rabin Word Automata

[Safra, 88]

Existing Procedures

- 2ExpTime-Complete [Rosner, 92]
- "classical" procedure [Pnueli, Rosner, 89]

LTL $\xrightarrow{2^{O(n)}}$ Büchi W

Solve a Rabin Game

in Word Automata

Existing Procedures

- 2ExpTime-Complete [Rosner, 92]
- "classical" procedure [Pnueli, Rosner, 89]

LTL $\xrightarrow{2^{O(n)}}$ Büchi W ... in Word Automata

Solve a Rabin Game

- Safraless procedure [Kupferman, Vardi, 05]

LTL → Universal CoBüchi Tree Automata

↓

Büchi Tree Automata ← Alternating Weak Tree Automata

Existing Procedures

- 2ExpTime-Complete [Rosner, 92]
- "classical" procedure [Pnueli, Rosner, 89]

LTL $\xrightarrow{2^{O(n)}}$ Büchi W

Solve a Rabin Game

in Word Automata

- Safraless procedure [Kupferman, Vardi, 05]

LT

ree Automata

Solve a Büchi Game

Büchi Tree Autom

ee Automata

Existing Procedures

- 2ExpTime-Complete [Rosner, 92]
- "classical" procedure [Pnueli, Rosner, 89]

LTL $\xrightarrow{2^{O(n)}}$ Büchi W … bin Word Automata

Solve a Rabin Game

- Safraless procedure [Kupferman, Vardi, 05]

LT … ree Automata

Solve a Büchi Game

Büchi Tree Autom … ee Automata

Implemented in *Lily* [Jobstmann, Bloem,06]

A New Safraless Approach

LTL

↓ $2^{O(n)}$

Universal coBüchi Word automata

↓ $O(1)$

Universal **KcoBüchi** Word automata

↓ $2^{O(m^2)}$

Det. KcoBüchi Word automata

Universal **KcoBüchi** Word: all run visit at most **K** accepting states

A New Safraless Approach

LTL

$2^{O(n)}$ ↓

Un … ata

Solve a Safety Game

Univ … mata

$2^{O(m^2)}$ ↓

Det. KcoBüchi Word automata

Universal **KcoBüchi** Word: all run visit at most **K** accepting states

A New Safraless Approach

LTL

↓ $2^{O(n)}$

Un … ata

Solve a Safety Game

Univ … mata

↓ $2^{O(m^2)}$

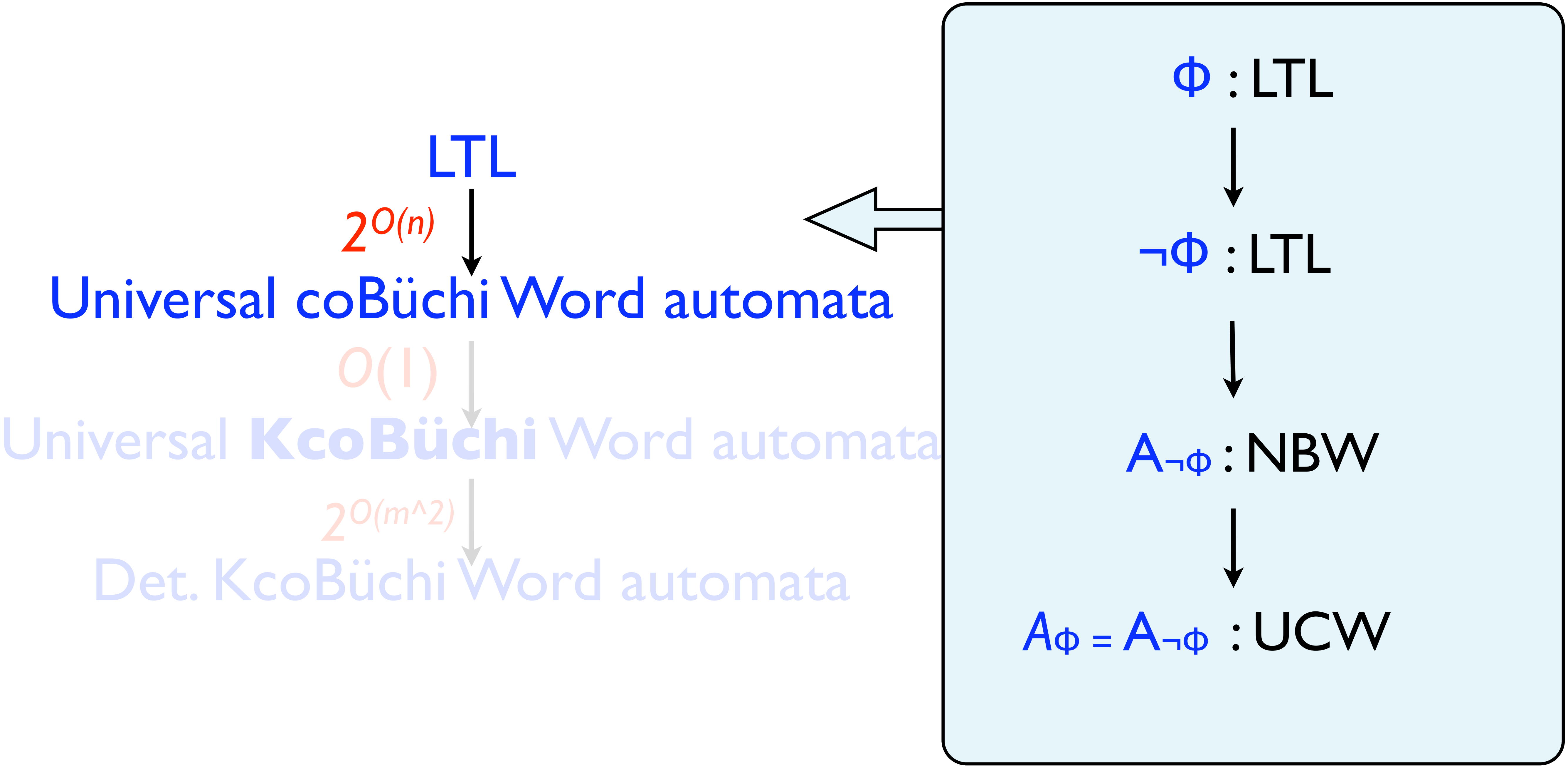
Det. KcoBüchi Word automata

Doubly Exponential Size ...

... Game solved on-the-fly with **antichain** technics

Universal **KcoBüchi** Word: all run visit at most **K** accepting states

A New Safraless Approach



Universal **KcoBüchi** Word: all run visit at most **K** accepting states

A New Safraless Approach

LTL

$2^{O(n)}$

Universal coBüchi Word automata

$O(1)$

Universal **KcoBüchi** Word automata

$2^{O(m^2)}$

Det. KcoBüchi Word automata

Theorem [Safra88,Kupferman-Vardi05]

Let A: UCW with n states,

A is realizable

$\longleftrightarrow$

it is realizable by a finite-state strategy S with at most n^{2n+1} states.

Consequence

the runs of A on words compatible with S visits at most **K**$=n^{2n+2}$ final states

Universal **KcoBüchi** Word: all run visit at most **K** accepting states

A New Safraless Approach

LTL

$2^{O(n)}$

Universal coBüchi Word automata

$O(1)$

Universal **KcoBüchi** Word automata

$2^{O(m^2)}$

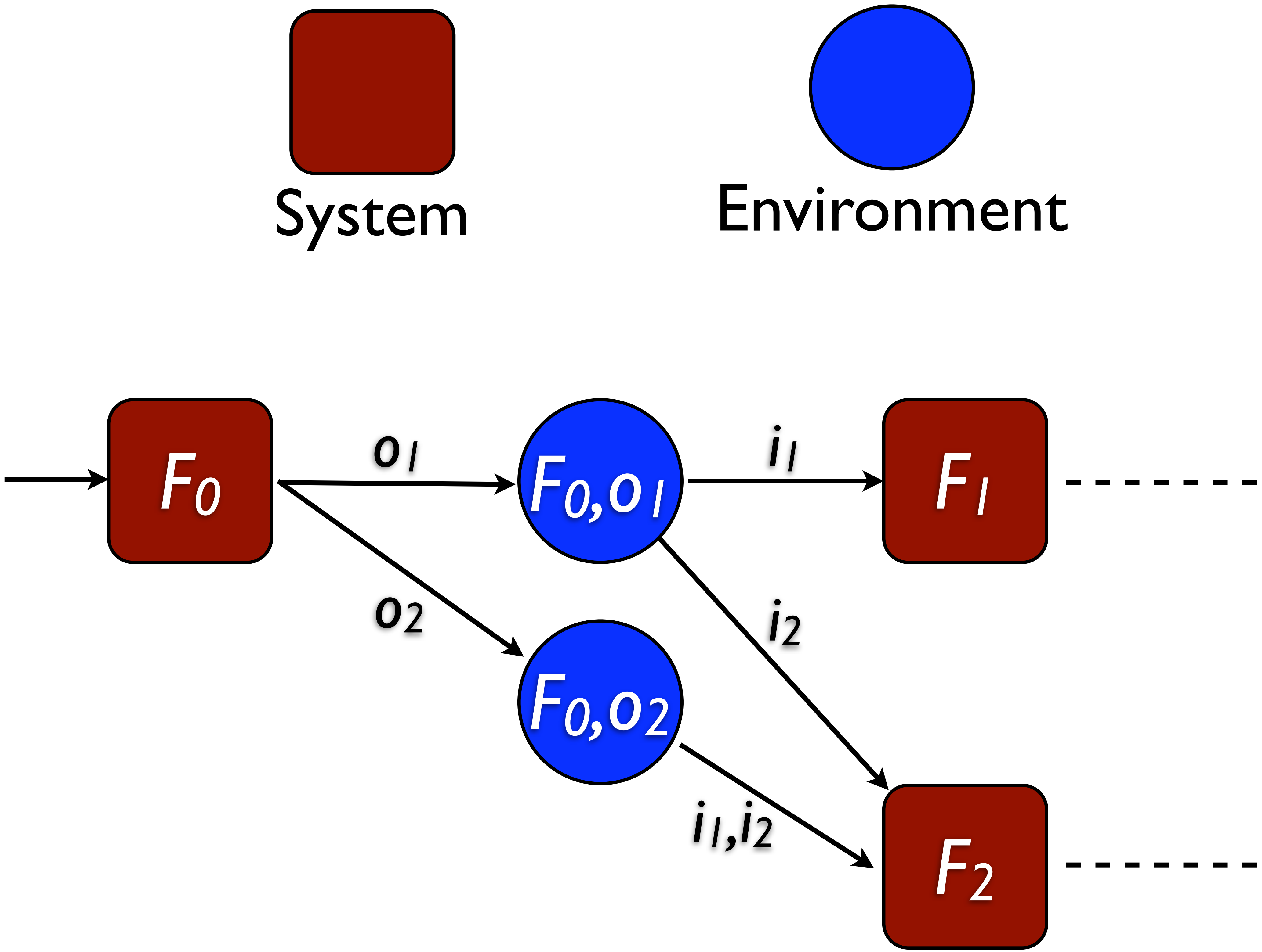
Det. KcoBüchi Word automata

Determinization

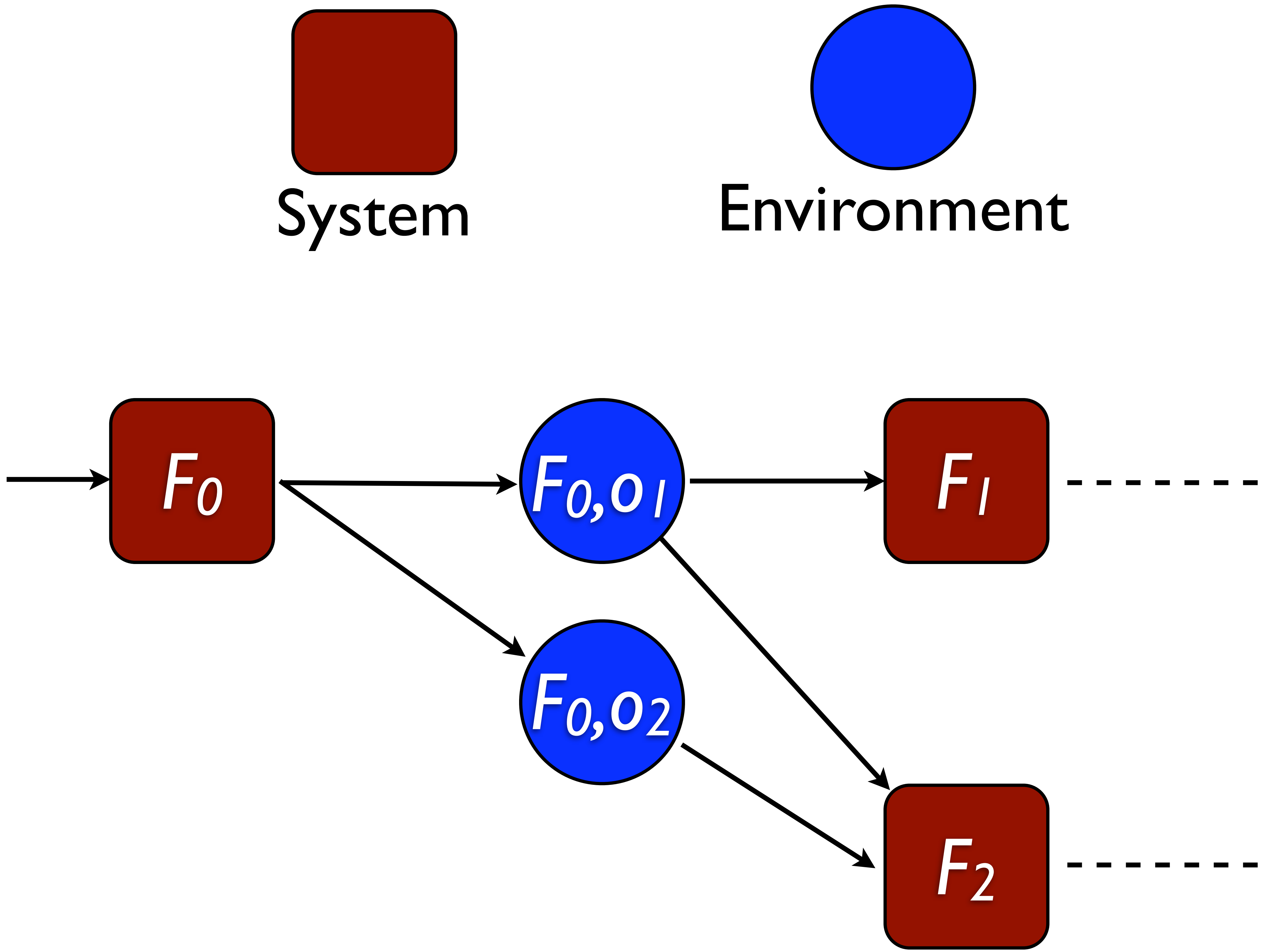
- For each state *q*, count the maximal number of final states visited by runs ending up in *q*
- Set of states: *counting functions F* from *Q* to [-1,0,...,**K**+1]
- Final states are functions *F* such that ∃*q*: *F*(*q*)>**K**
- set the bound to *0*

Universal **KcoBüchi** Word: all run visit at most **K** accepting states

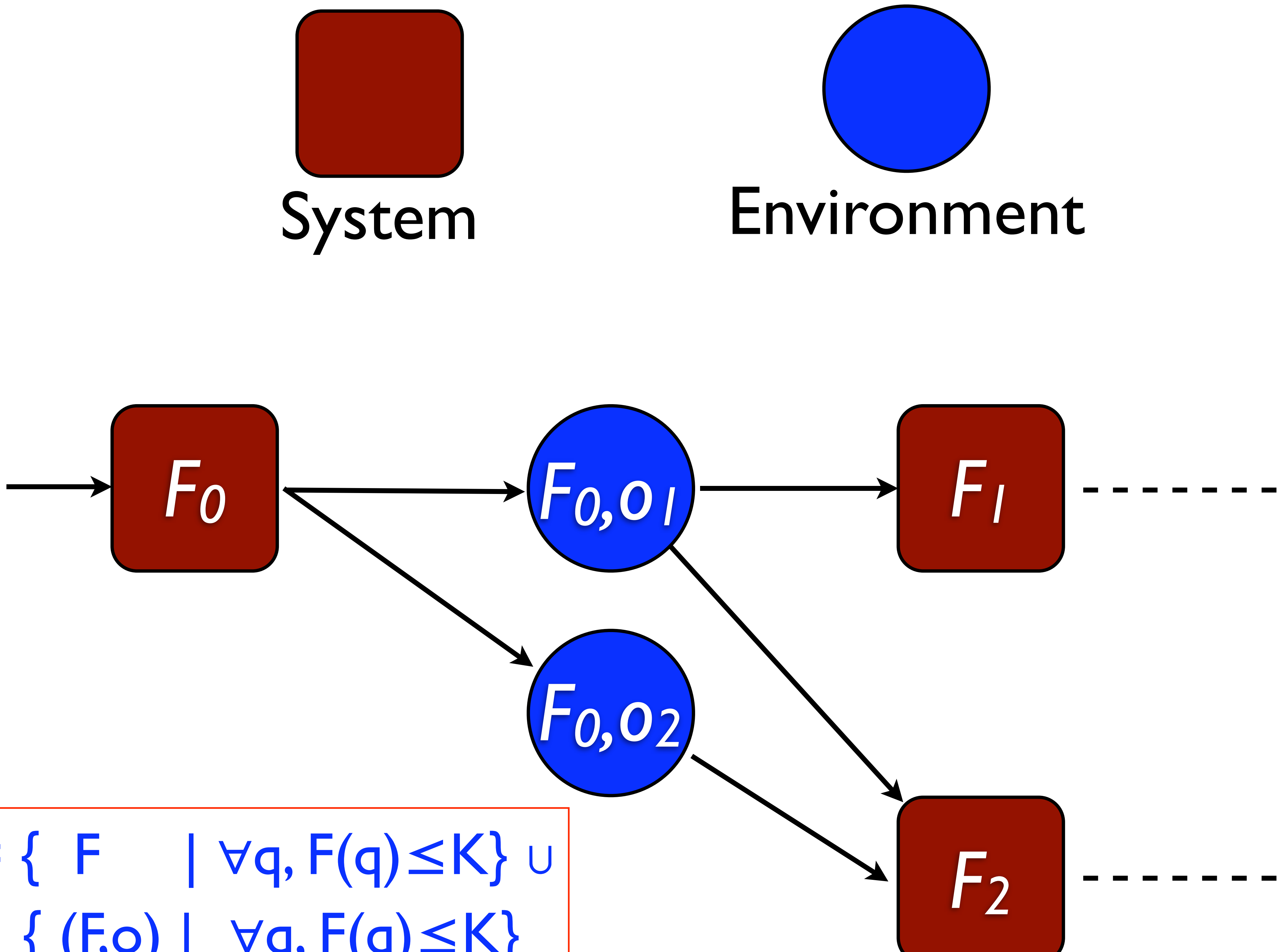
Realizability as a Safety Game



Realizability as a Safety Game



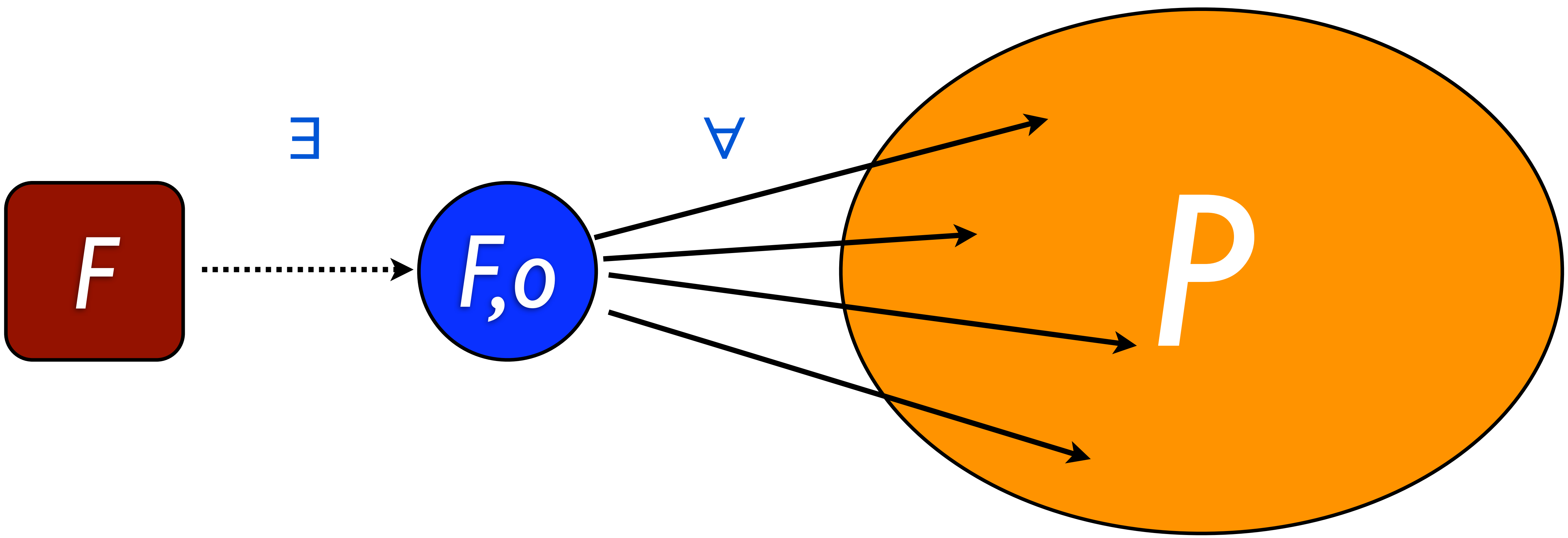
Realizability as a Safety Game



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

Controllable Predecessors

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



- greatest fixpoint Pre^* = *winning region for System*

Controllable Predecessors

1. partial order on counting functions:

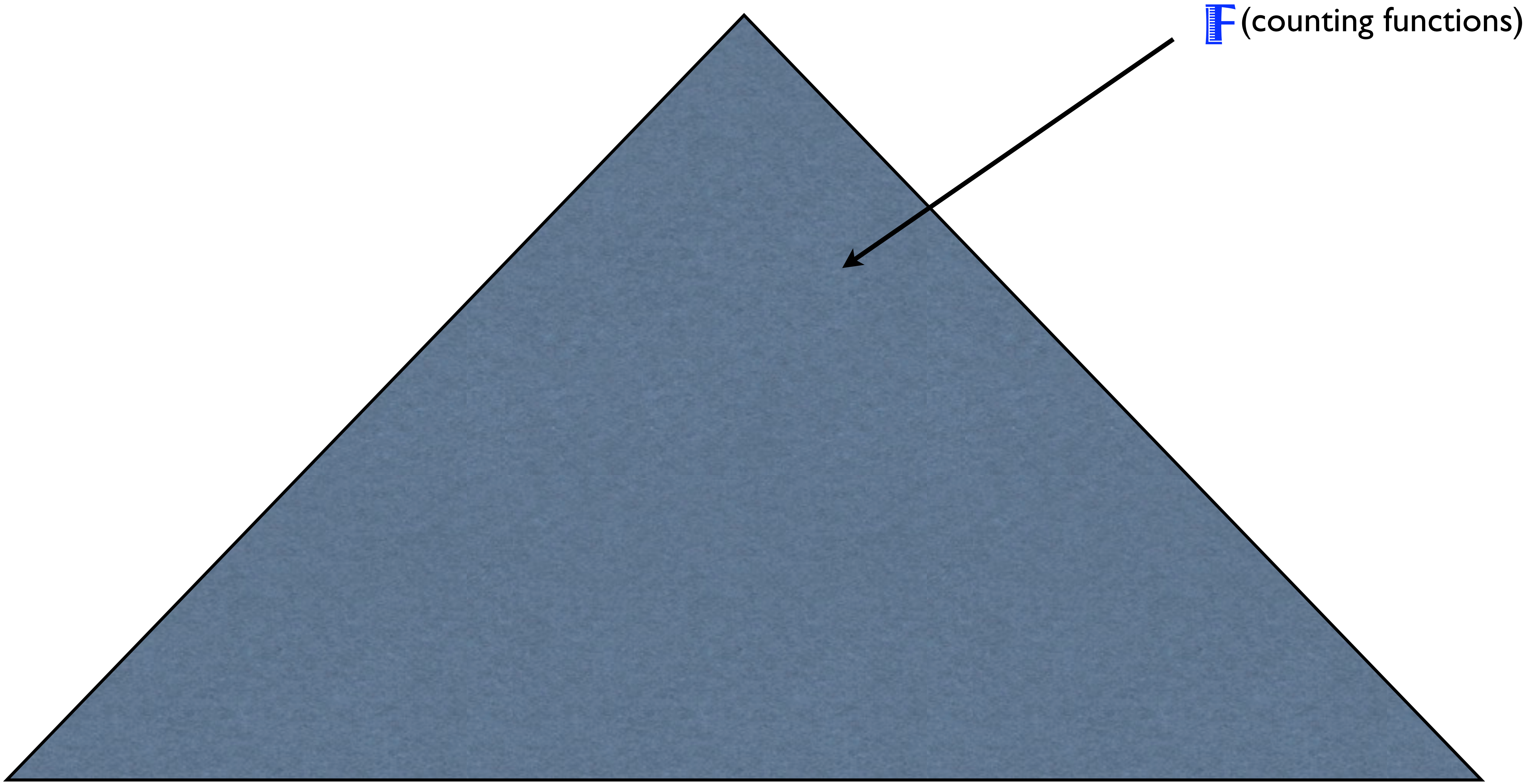
$$F \leq_d F' \text{ if } \forall q\text{: } F(q) \leq F'(q)$$

2. if System wins from *F'*, she also wins from

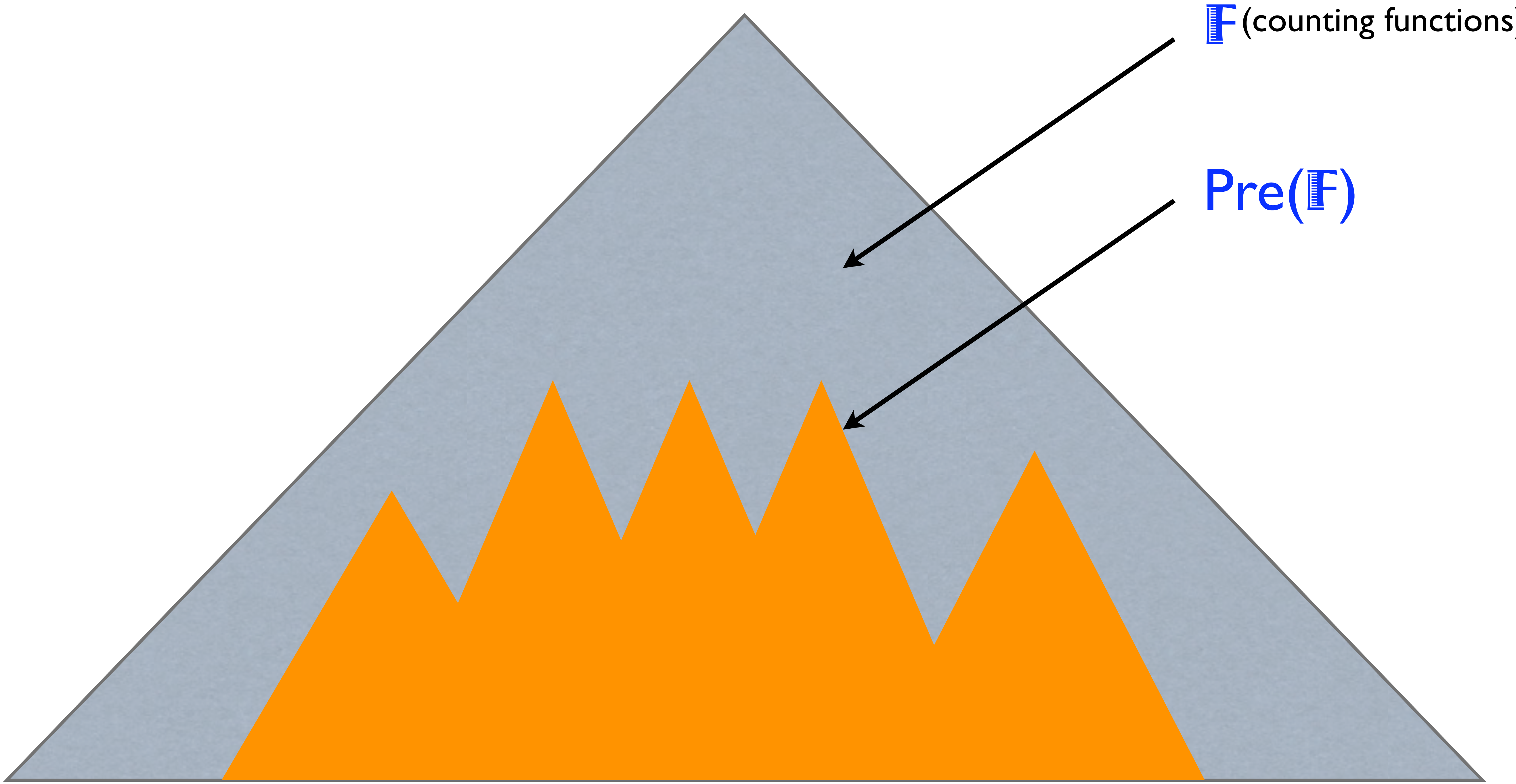
3. *Pre(.)* preserves *downward*-closed sets

4. represent each (downward) set of the fixpoint computation by its maximal elements

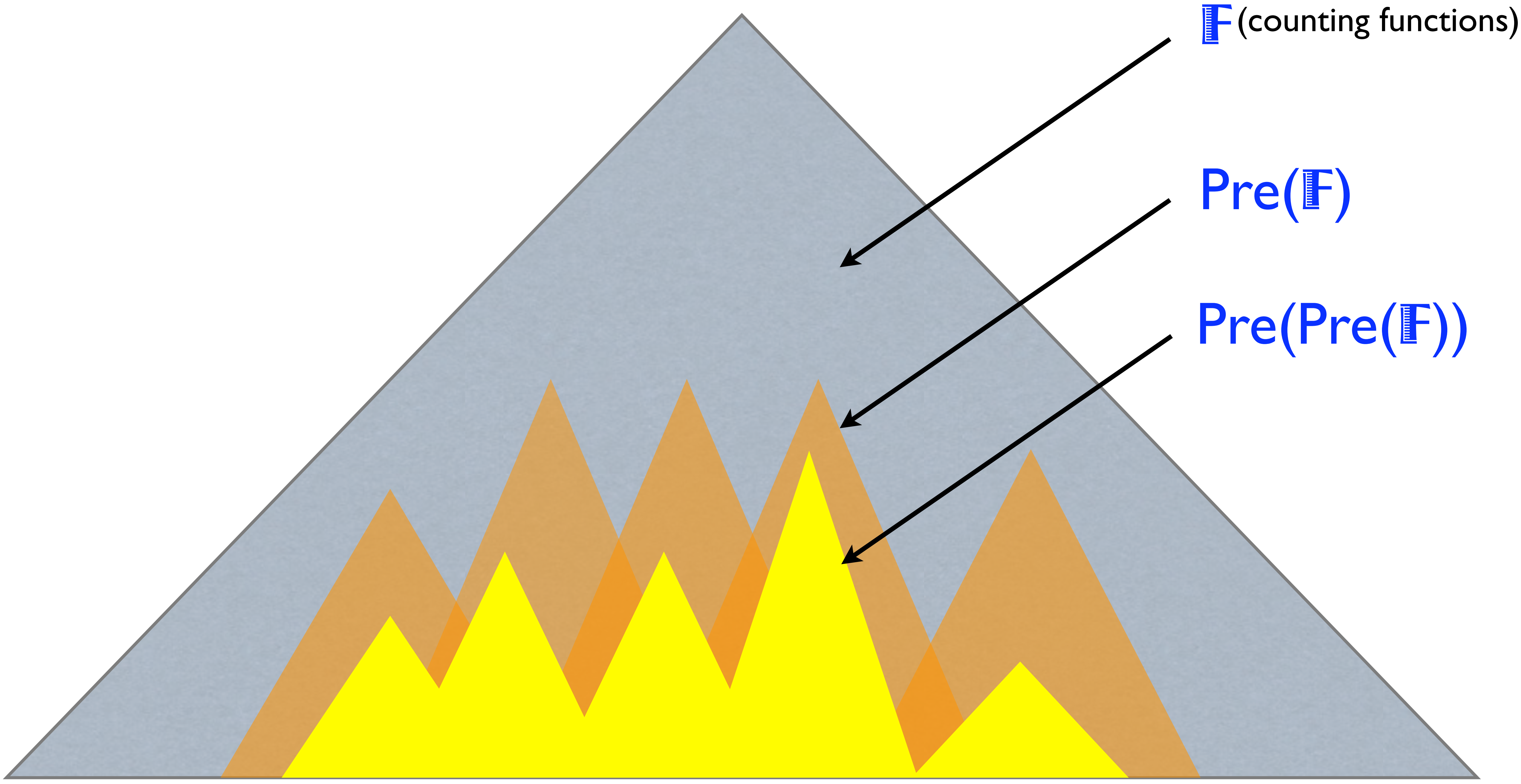
Symbolic Fixpoint Computation



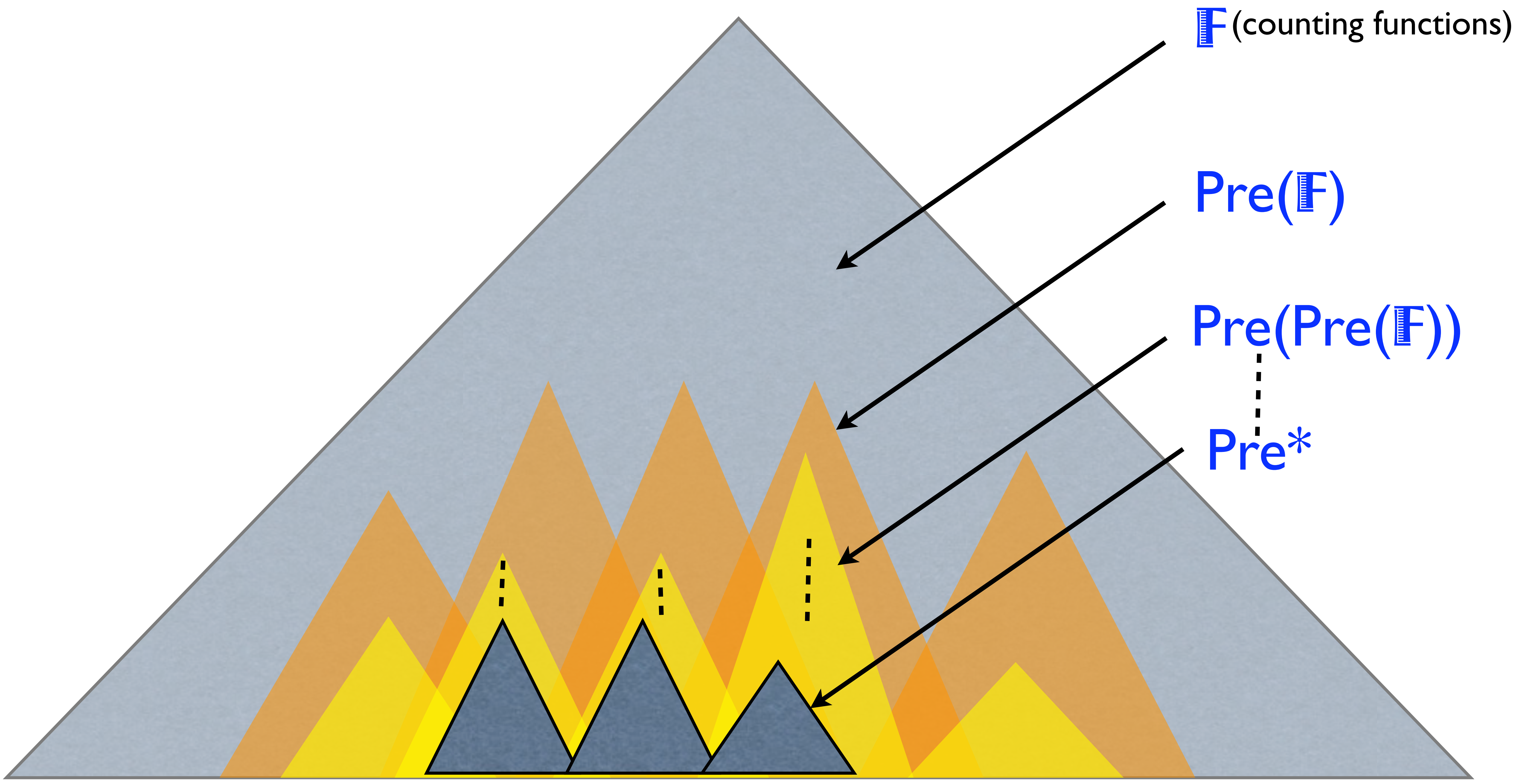
Symbolic Fixpoint Computation



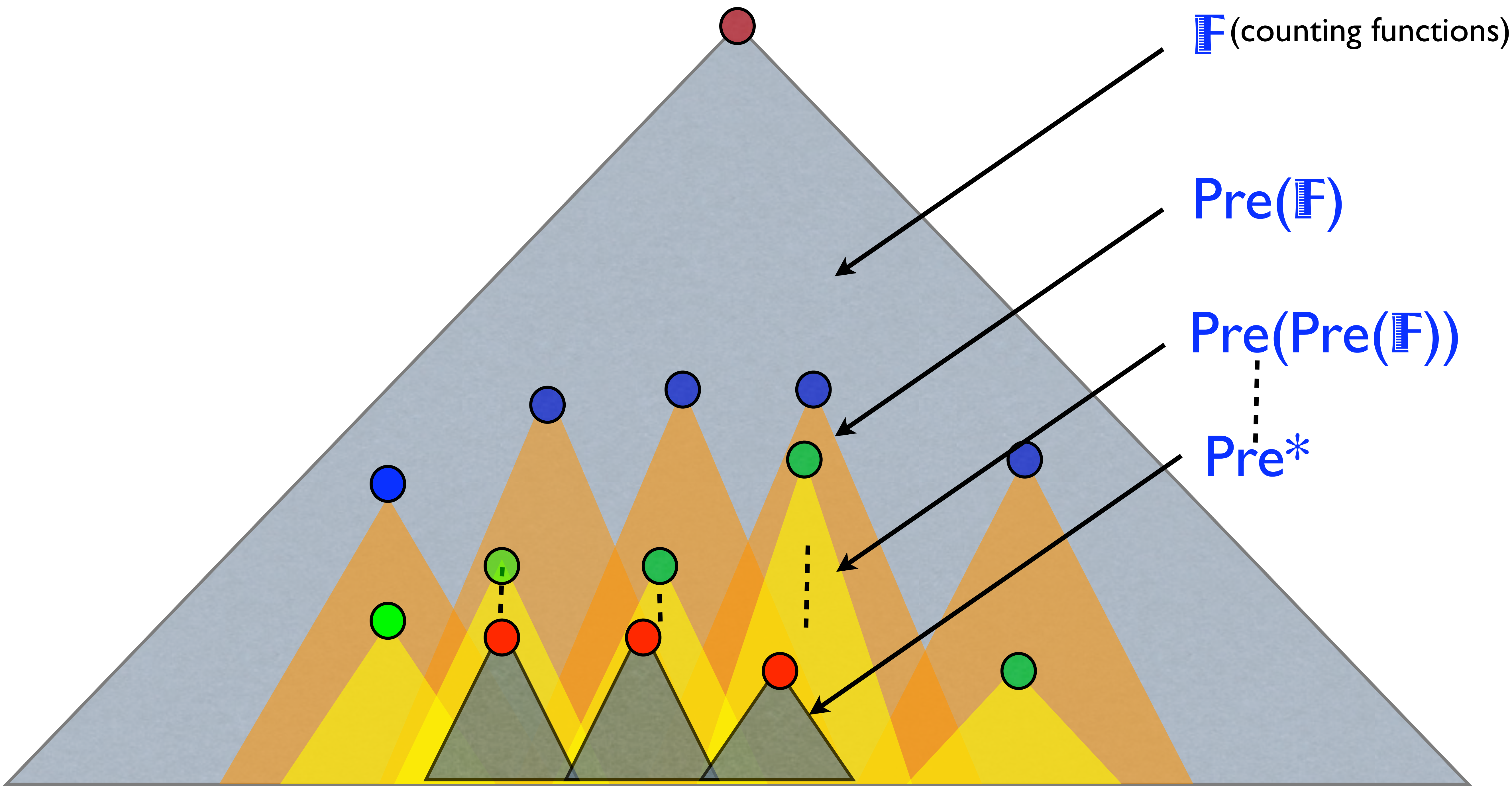
Symbolic Fixpoint Computation



Symbolic Fixpoint Computation



Symbolic Fixpoint Computation



Incremental Algorithm

- the bound **K** is very big (doubly exponential)
- if the spec is realizable with a "small" bound, it is realizable with a "big" bound
- iterate over *k=0,1,...,***K**

Incremental Algorithm

- the bound **K** is very big (doubly exponential)
- if the spec is realizable with a "small" bound, it is realizable with a "big" bound
- iterate over *k=0,1,...,***K**

Not reasonable for unrealizable specifications

Incremental Algorithm

- the
- if t
  rea
- ite

Not

ions

But by Martin’s determination theorem:

ϕ is unrealizable for the System iff $\neg\phi$ is realizable for the Environment.

Experiments

- implementation in Perl (as Lily)
- if the spec is realizable, output a Moore machine that realizes it
- formula to automata construction borrowed from Lily (based on Wring [Somenzi, Bloem])
- significantly faster on all realizable Lily's examples
- bottleneck: formula to automaton construction

Future Work ...

- compositionnality
- avoid automata construction to handle larger formulas

Future Work ...

- compositionnality
- avoid automata construction to handle larger formulas

... Thank You