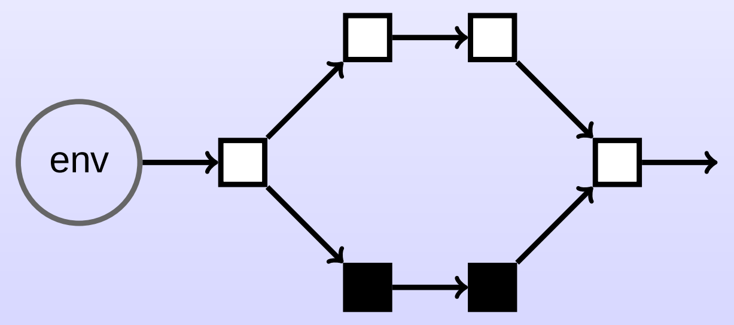
Automatic Abstraction for Complex Partial Designs

Rayna Dimitrova, Saarland University, Germany

Partial Design Verification

partial design

-distributed system
-communicating components
-interacts with environment



☐ – already implemented
■ – no implementation yet

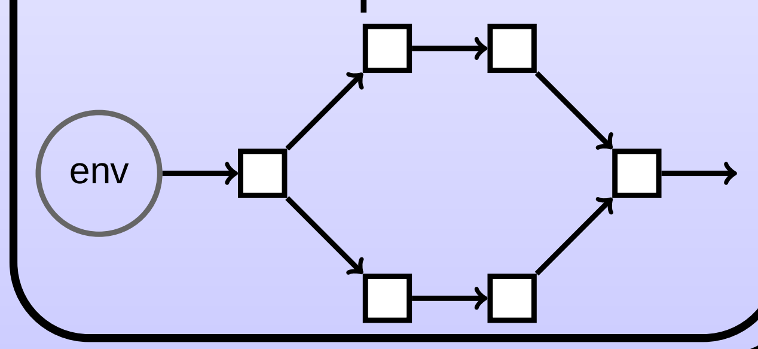
property Φ

?∃ implementations for ■ ■ such that the system satisfies Φ

YES: the implementation can be completed

NO: bug in the implemented components → revise existing implementation

synthesis
derive automatically correct implementation



Motivation & Challenges

Goals of partial-design verification:

- apply verification in early design stages
- reduce development time and costs

Challenges in partial-design verification:

- deal with infinite (or very large) state space
- account for components having incomplete information about the global system state (e.g., private variables of other processes)

Partial Design: Bakery Mutual Exclusion

Process A:

```
l0: ticketA := 0;
l1: while(true){
  l2: ticketA := ticketB + 1;
  l3: await(ticketB = 0 ∨
              ticketA < ticketB);
  l4: critical;
  l5: ticketA := 0;
}
```

Process B:

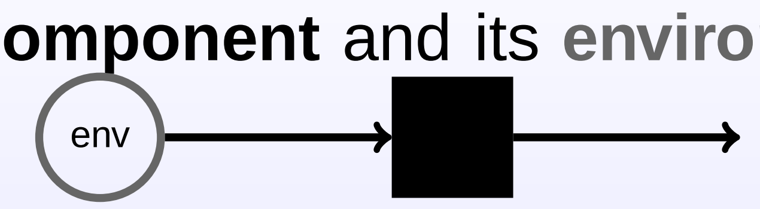
```
m0: ticketB := 0;
m1: while(true){
  m2:  | ticketB := 0;
       | ticketB := ticketA;
       | ticketB := ticketA + 1;
  m3: await(?);
  m4: critical;
  m5: ticketB := 0;
}
```

Property: it is never the case that $pcA = l4 \wedge pcB = m4$ and } **safety property**

(whenever $pcA = l3$, then eventually $pcA = l4$ and whenever $pcB = m3$, then eventually $pcB = m4$) } **strengthen to bounded liveness**

Game Model

infinite turn-based game between a **component** and its **environment**



tries to violate the property Φ

tries to ensure the property Φ

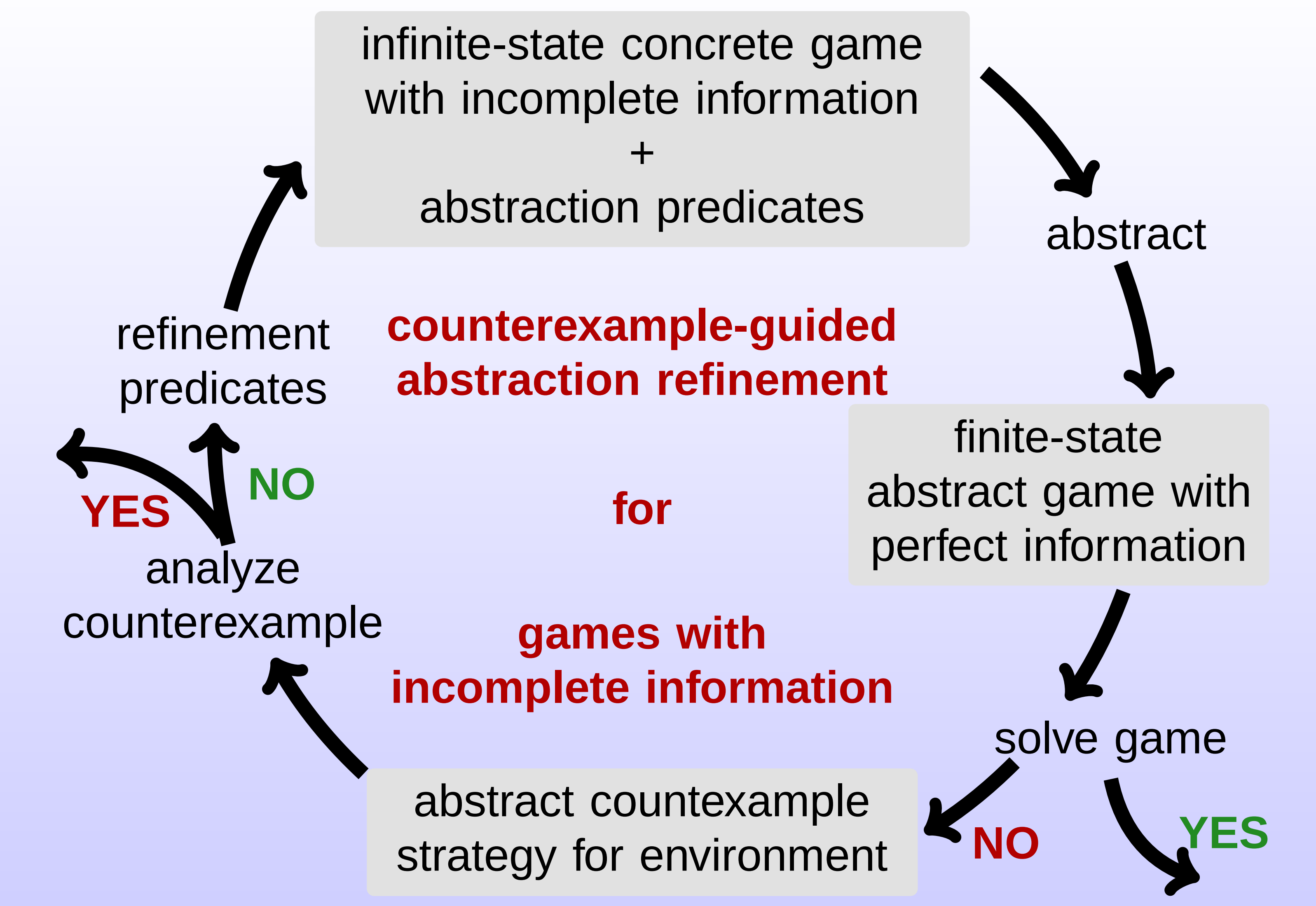
Strategies in the game for the:

- component → **implementation**
- environment → **counterexample**

Informedness of the component

- the component player has **incomplete information** about the global state
- strategy for the component must not depend on information that is not available to it

Results



infinite-state concrete game with incomplete information + abstraction predicates

abstract

finite-state abstract game with perfect information

solve game

YES

NO

abstract counterexample strategy for environment

analyze counterexample

NO

YES

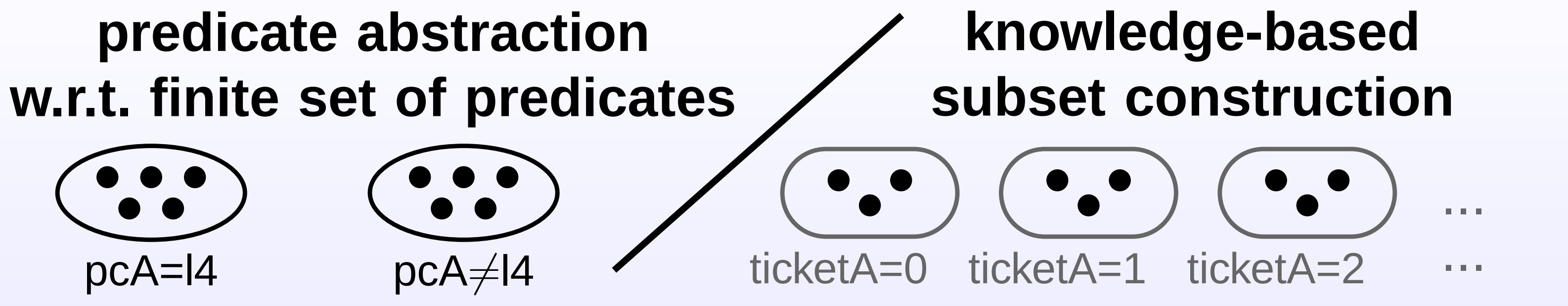
refinement predicates

counterexample-guided abstraction refinement

for

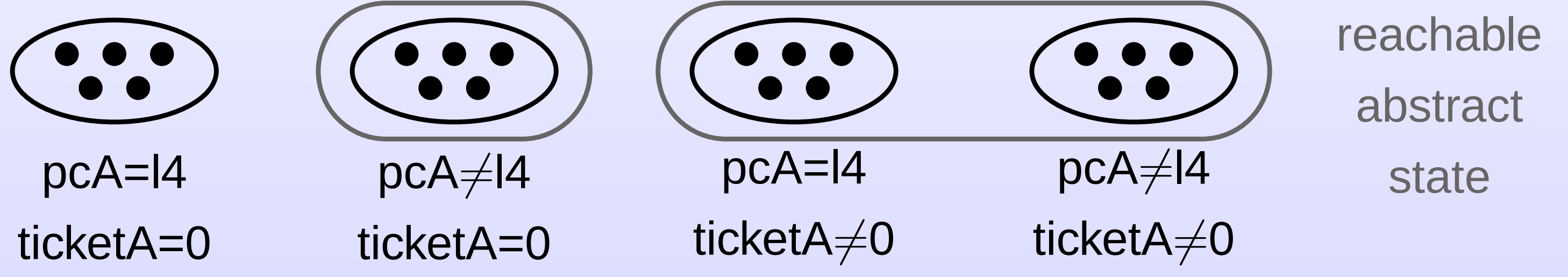
games with incomplete information

Abstraction



predicate abstraction + knowledge-based subset constriction

the predicate pcA=l4 is not observable



Sound abstraction for games under incomplete information

- **overapproximate** the power of the environment player
- **underapproximate** the power of the component player
- the abstract component has less information than the concrete

⇒ abstract implementation → concrete implementation

Refinement

Sound and complete analysis of abstract counterexamples

- **safety properties:** abstract strategy for the environment → **strategy tree**

reduction to satisfiability of a strategy-tree formula

⇒ determine correctly whether an abstract counterexample is concretizable

Refinement procedure for games under incomplete information

interpolant computation based on constraint solving

- impose constraints on the interpolants to obtain suitable predicates

⇒ appropriately refine the abstract informedness when this is necessary

Ongoing & Future Work

Prototype implementation

- optimize interpolation computation
- extend to other logical theories

Application to timed games

- find a suitable symbolic model

Distributed partial designs

- make use of component's locality

[1] Rayna Dimitrova and Bernd Finkbeiner. Abstraction Refinement for Games with Incomplete Information. In *Proc. FSTTCS'08*

UNIVERSITÄT DES SAARLANDES

Microsoft Research