Symbolic model checking of multi-modal logics: uniform strategies and rich explanations

(Model checking symbolique de logiques multi-modales : stratégies uniformes et explications riches)

Simon Busard

Public defense

UCL, July 3, 2017

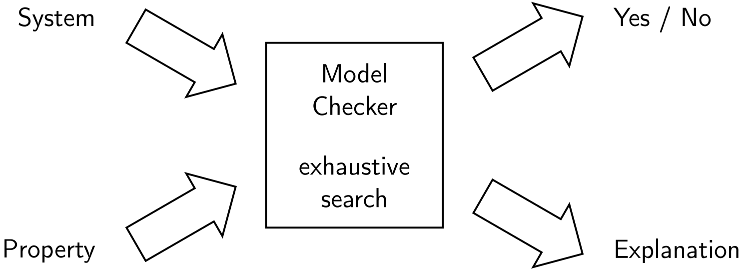
Running example: Mastermind

Simplified Mastermind: 3 colors, 3 turns, 2 pegs (different colors)

Running example: Mastermind

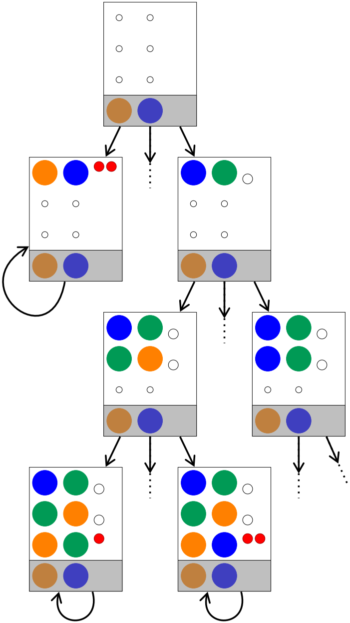
Simplified Mastermind: 3 colors, 3 turns, 2 pegs (different colors)

Model checking = a **verification** technique

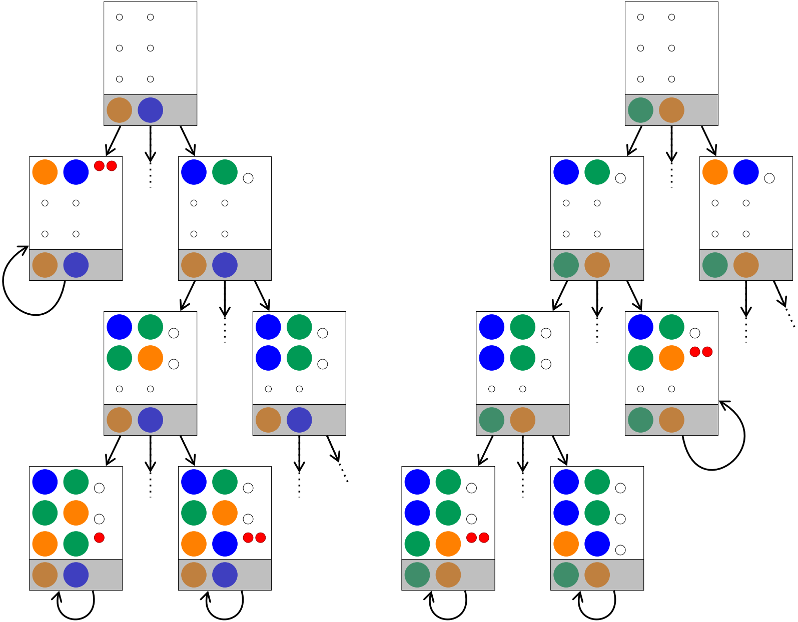


Other verif. techniques: testing, simulation, proof-based approaches

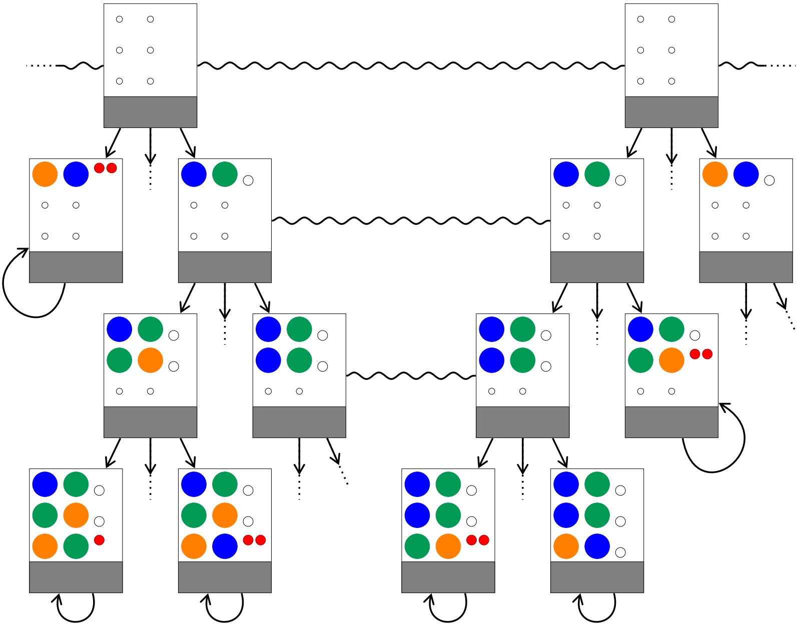
System modeled as a **finite-state machine**



System modeled as a **finite-state machine** (1122 states)



System modeled as a **finite-state machine**



Properties expressed within a **logic**

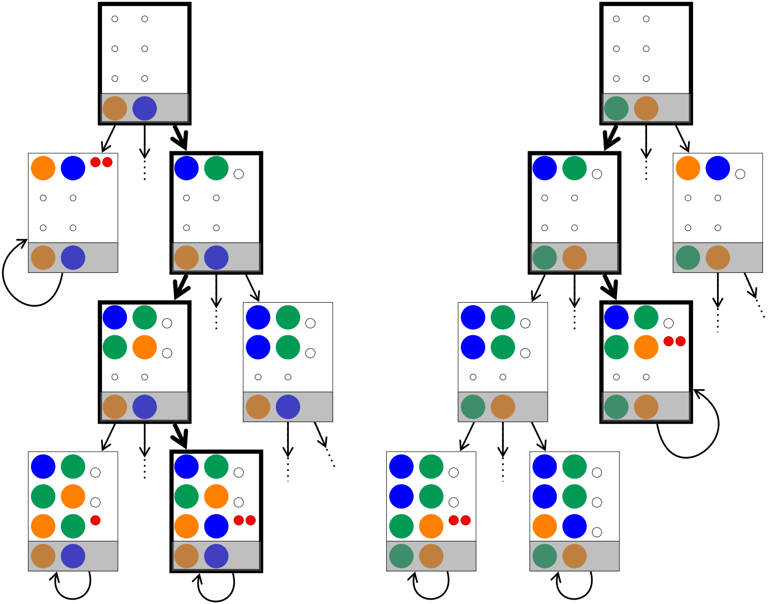
$\Rightarrow$ The logic defines what properties can be expressed

$\Rightarrow$ The properties have a mathematical meaning:
a **formal semantics**

Example in temporal logic:

"it is possible to win the game"
EF *win*

Model checking =
exhaustive search guided by the property



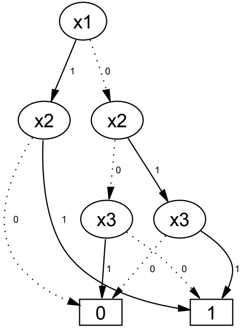
Symbolic model checking

state-space explosion problem:
T turns, C colors, P pegs $\Rightarrow$ up to $C^{P(T+1)}$ states

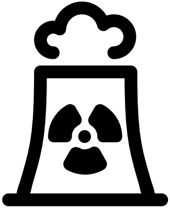
standard Mastermind $\Rightarrow 8^{4\times(12+1)} \approx 9 \times 10^{46}$ states

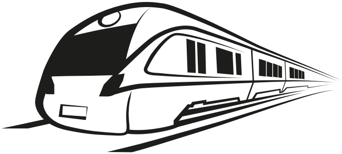
Solution:
use **Binary Decision Diagrams** (BDDs)
to represent and manipulate

- the system
- the transitions
- **sets of states**



Applications: safety-critical systems





Multi-modal logics

Reason about **several aspects** of the model:
time, knowledge, strategies, etc.

"the pilot ***will eventually know*** *that he may land"*

$\mathbf{AF}\ \mathsf{K}_{pilot}\ authorized$

"the doctor is ***always aware*** *that the patient is not dead"*

$\mathbf{AG}\ \mathsf{K}_{doctor}\ \neg dead$

"the power plant controller knows he can avoid explosions"

$\mathsf{K}_{controller}\ \langle\!\langle controller \rangle\!\rangle \mathbf{G}\ \neg explosion$

Thesis contributions

1. Model checking techniques for **uniform strategies**

2. A framework for multi-modal logic **rich explanations** generation and manipulation

Outline

Mastermind: is there a **strategy** to win the game?

How can we find such a strategy?

What is a strategy?

A (general) strategy =
what to do (what **action** to play) in each state

What is a strategy?

A (general) strategy =
what to do (what **action** to play) in each state

A strategy is **winning** for some objective if
all executions of the model following this strategy satisfy the objective

What is a strategy?

A (general) strategy =
what to do (what **action** to play) in each state

A strategy is **winning** for some objective if
all executions of the model following this strategy satisfy the objective

Just play the solution
⇒ unrealistic because the player cannot see the solution

What is a strategy?

A (general) strategy =
what to do (what **action** to play) in each state

A strategy is **winning** for some objective if
all executions of the model following this strategy satisfy the objective

Just play the solution
⇒ unrealistic because the player cannot see the solution

A uniform strategy =
what to do in each **observed** situation =
same actions in **indistinguishable** states

cannot play the solution in the initial state
because the player does not see it

ATL_{ir}: a logic to reason about uniform strategies

Look for uniform strategies to achieve some **objective**

"the player has a strategy to ***eventually*** *win the game"*

$\langle\langle player \rangle\rangle \mathbf{F}\ win$

"the player has a strategy to ***never*** *put a blue peg"*

$\langle\langle player \rangle\rangle \mathbf{G}\ no\ blue\ peg$

"the player has a strategy to play only blue pegs at the next turn"

$\langle\langle player \rangle\rangle \mathbf{X}\ all\ blue$

The same uniform strategy must be winning
for **all states indistinguishable from the states of interest**!

The problem

Checking that there exists a winning uniform strategy
for a given objective

- is difficult (Δ_2^P-complete $= P^{NP}$-complete)
- had no solution since recently

Contributions

Techniques for checking the existence of winning uniform strategies:

1. a **naive** approach
2. an improved approach based on **partial** strategies
3. another approach building winning strategies from target states

They **enumerate** and **check** every uniform strategy of the agents

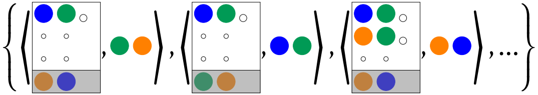
+ a way to remove **surely losing choices**
before enumerating the strategies

Outline

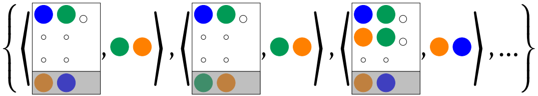
Representing strategies

A strategy = what action to play in each state

⇒ represent a strategy as a set of **state-action pairs** (moves)



A uniform strategy is represented as a set of **non-conflicting** moves



⇒ can be easily represented using binary decision diagrams

Checking one strategy for a given objective

Based on fixpoint computations, easy to compute (PTIME)

$Pre_{\langle\langle\Gamma\rangle\rangle}(Q', f_\Gamma) =$ states in which Γ can enforce to reach a state in Q' by using actions provided by f_Γ

$filter_{\langle\langle\Gamma\rangle\rangle\mathbf{X}}(Q', f_\Gamma) = Pre_{\langle\langle\Gamma\rangle\rangle}(Q', f_\Gamma)$

$=$ states in which Γ can enforce paths with second state in Q' by using actions in f_Γ

$filter_{\langle\langle\Gamma\rangle\rangle\mathbf{F}}(Q', f_\Gamma) = \mu Z.\ Q' \cup Pre_{\langle\langle\Gamma\rangle\rangle}(Z, f_\Gamma)$

$=$ states in which Γ can enforce paths reaching Q' by using actions in f_Γ

$filter_{\langle\langle\Gamma\rangle\rangle\mathbf{G}}(Q', f_\Gamma) = \nu Z.\ Q' \cap Pre_{\langle\langle\Gamma\rangle\rangle}(Z, f_\Gamma)$

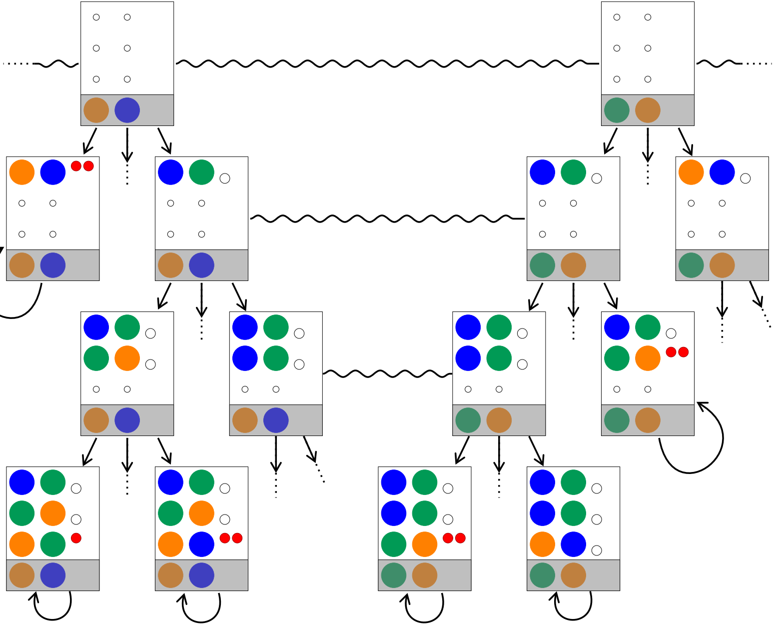
$=$ states in which Γ can enforce paths staying in Q' forever by using actions in f_Γ

The naive approach

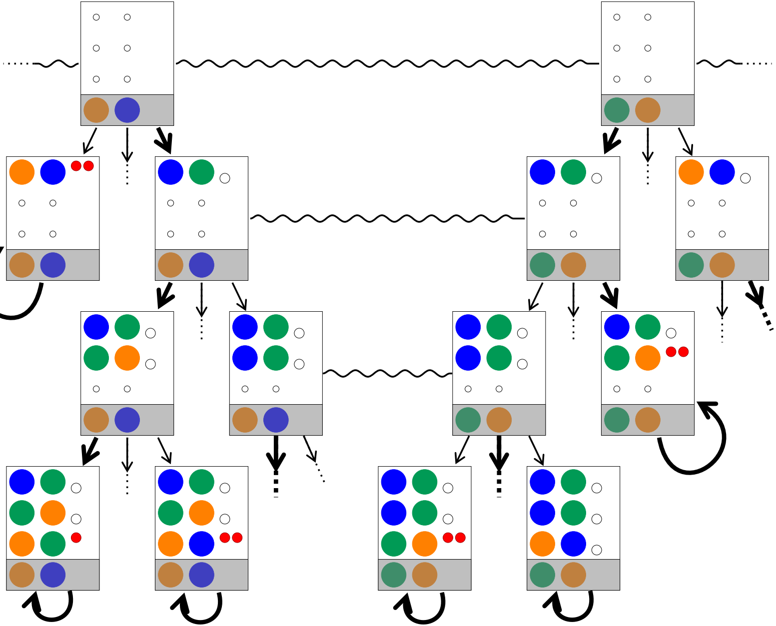
To compute the states for which there exists a winning uniform strategy for some objective

1. split the whole model into **uniform strategies** f_Γ
2. compute the states for which the **strategy is winning** with the corresponding *filter* algorithm
3. keep the states for which the strategy is winning for **all indistinguishable states**

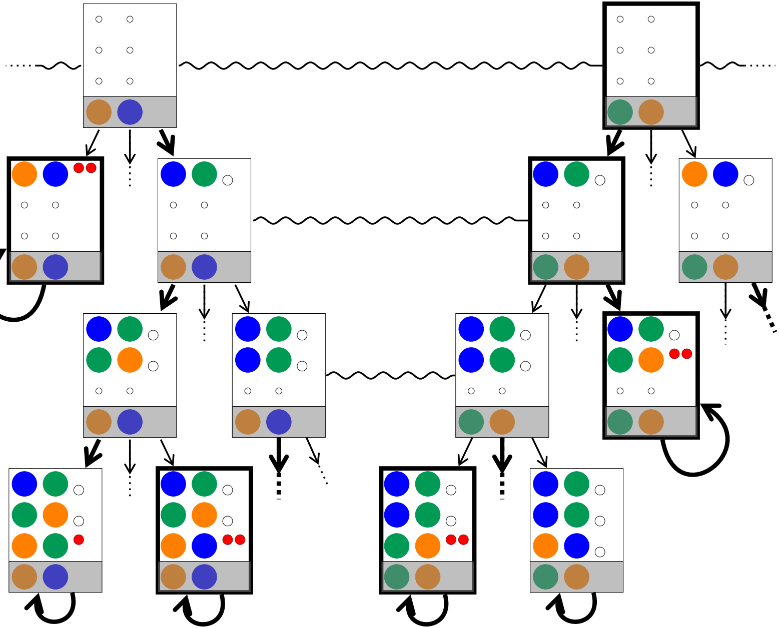
The naive approach



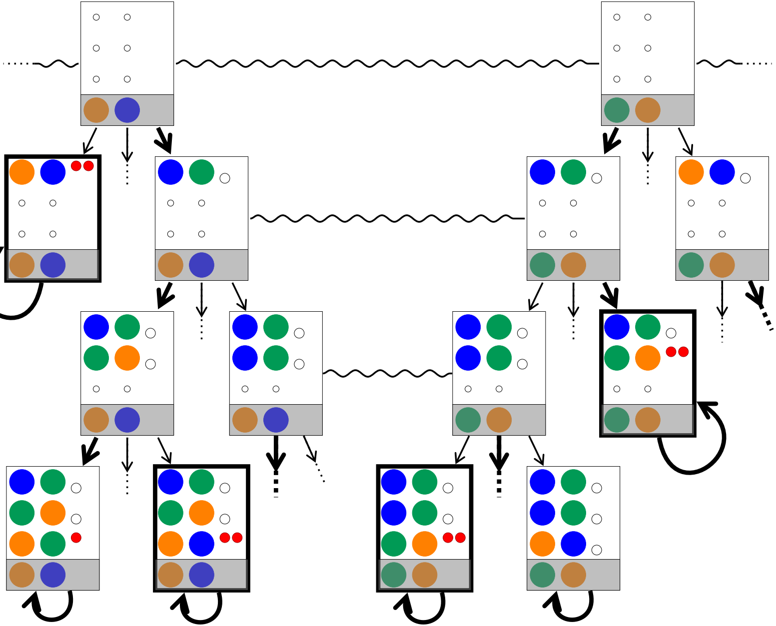
The naive approach



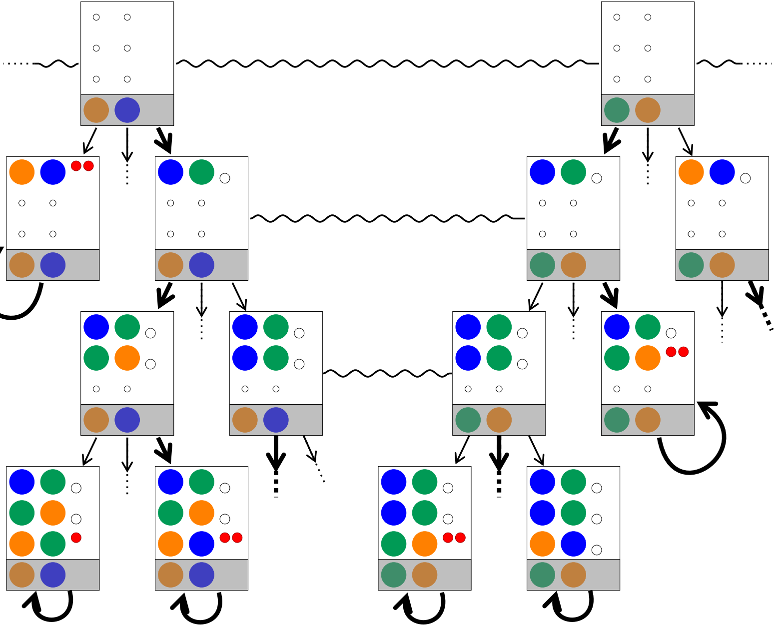
The naive approach



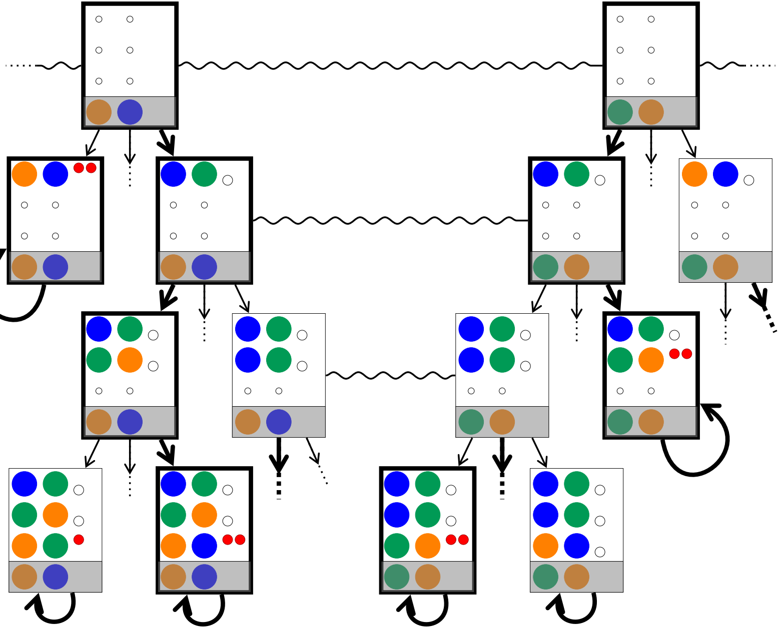
The naive approach



The naive approach



The naive approach



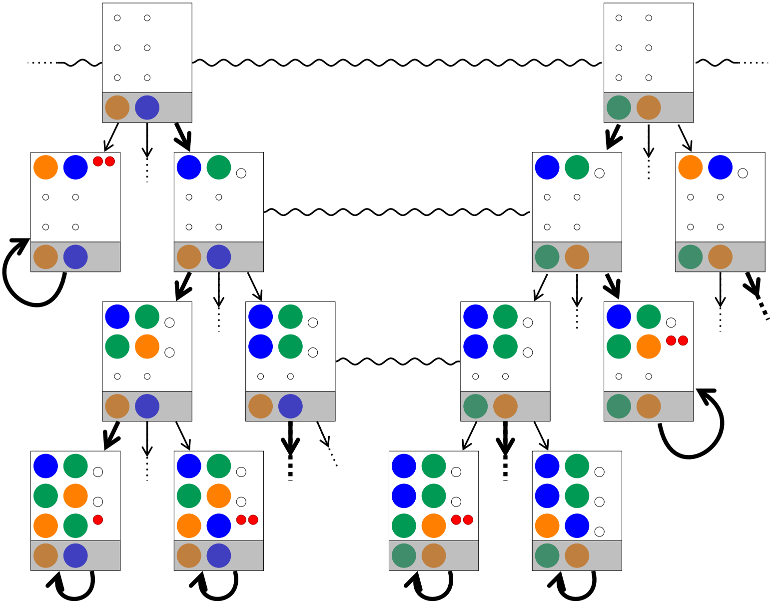
The naive approach is inefficient

The simplified Mastermind has 7×10^{112} **uniform strategies**

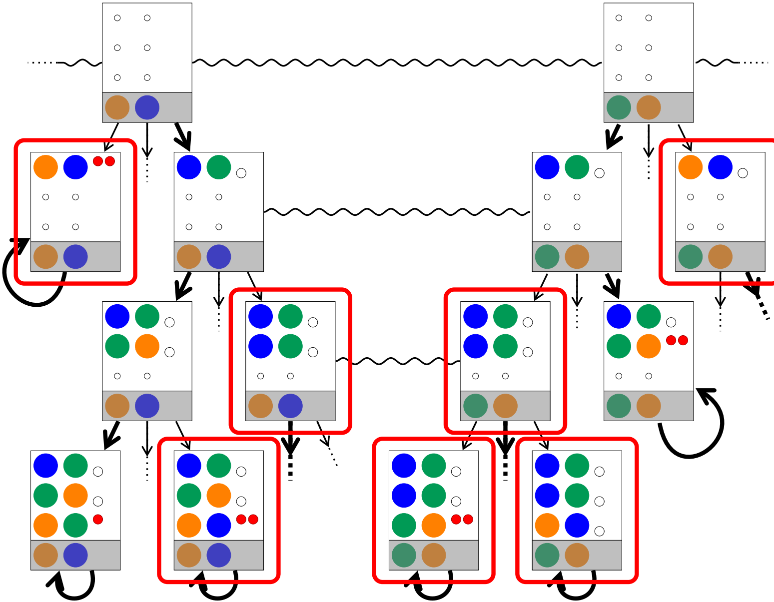
We need ways to **reduce** this number

$\Rightarrow$ **partial strategies**

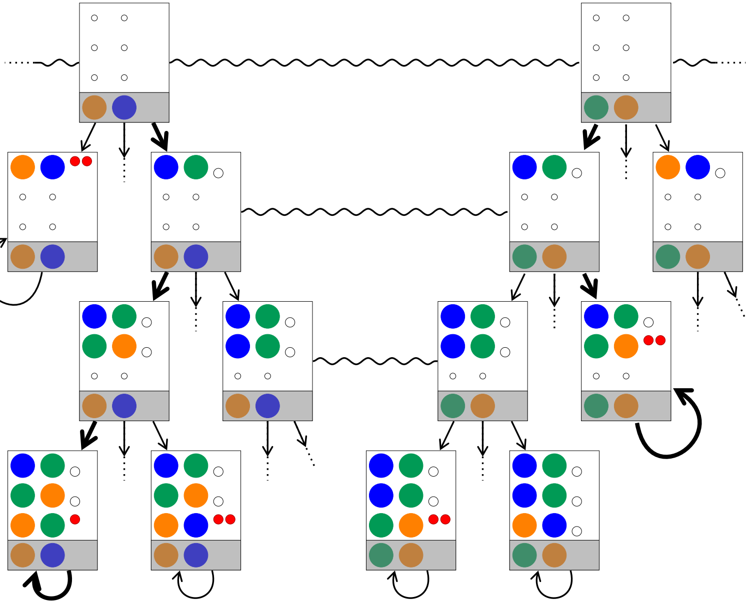
Partial strategies



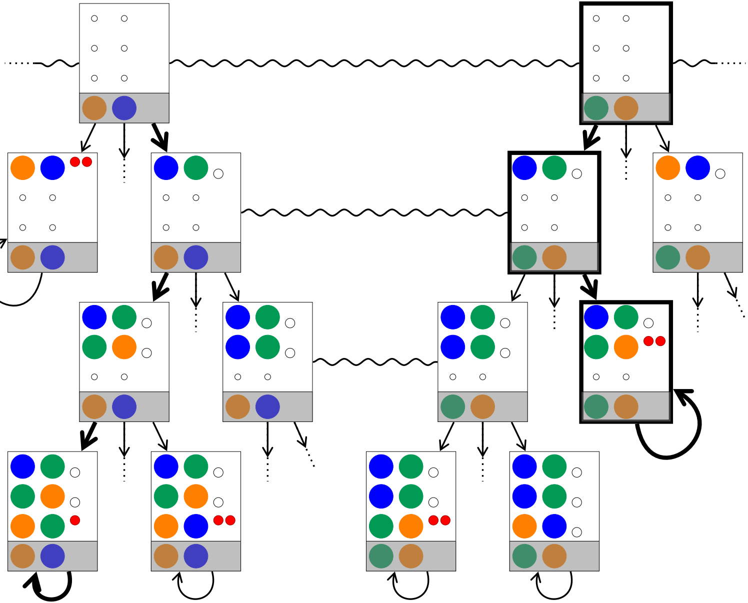
Partial strategies



Partial strategies

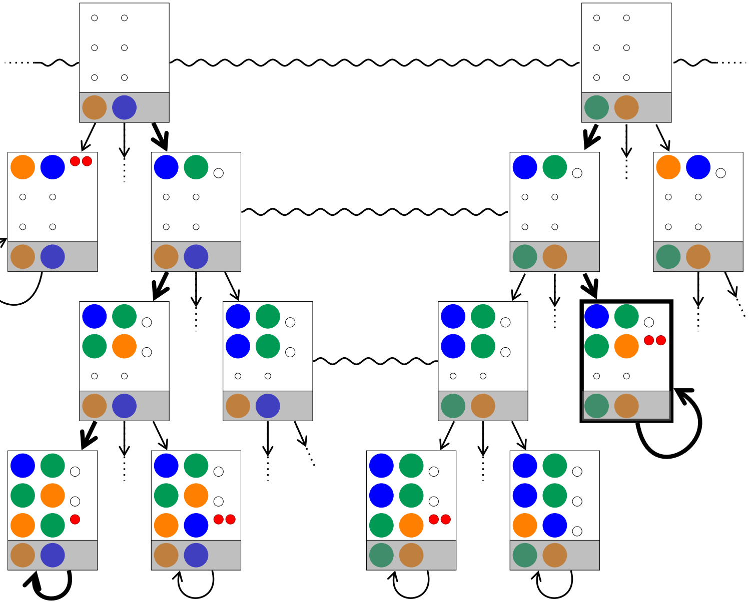


Partial strategies



Partial strategies

From 7×10^{112} to 2×10^{6} strategies



The partial approach

To compute the states of Q' for which there exists a winning uniform strategy for some objective

1. generate each **partial strategy** f_Γ
2. compute the states of Q' for which the **strategy is winning** with the corresponding *filter* algorithm
3. keep the states of Q' for which the strategy is winning for **all indistinguishable states**

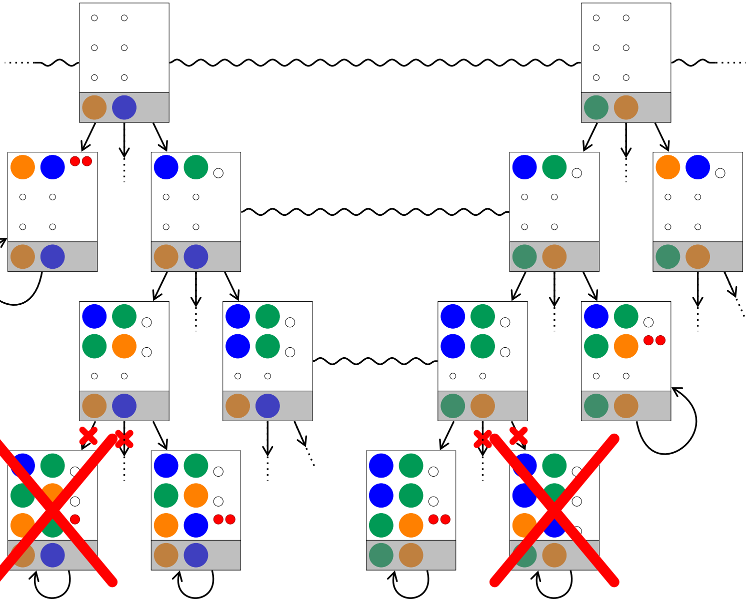
Pre-filtering surely losing moves

1. It is **easy** to compute the moves belonging to a winning **general** strategy (PTIME)

2. If some move does not belong to a winning general strategy, it does not belong to a winning **uniform** one

⇒ We can **remove the losing moves** before enumerating the uniform strategies

Pre-filtering surely losing moves



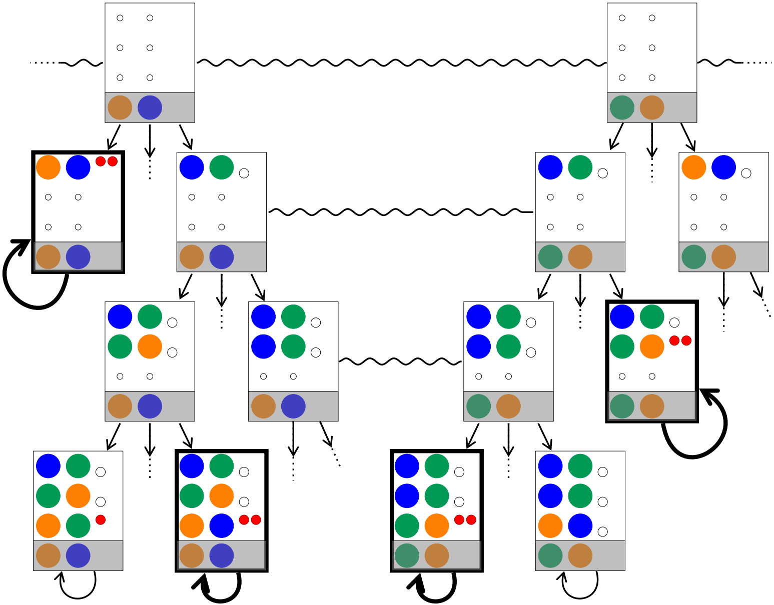
Pre-filtering can help

Pre-filtering can drastically reduce the number of strategies:

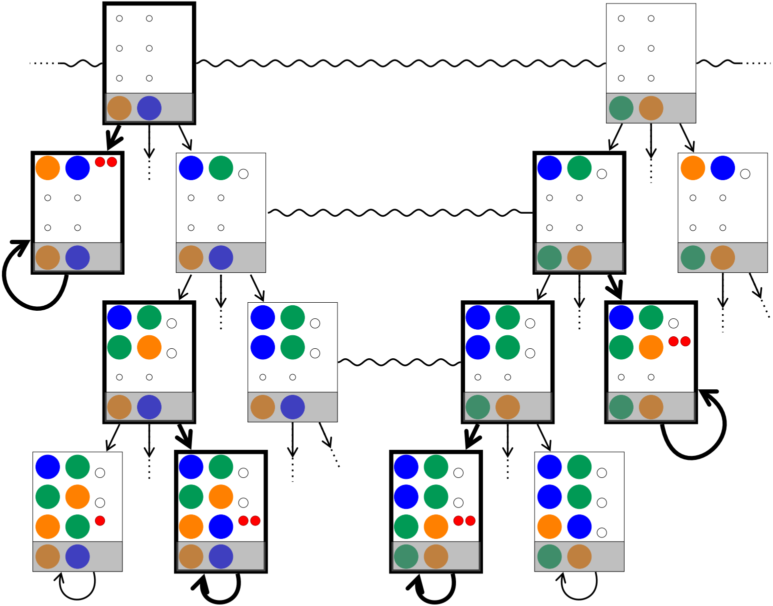
Naive approach: from 7×10^{112} to 10^{22} uniform strategies

Partial approach: from 2×10^6 to 2304 uniform partial strategies

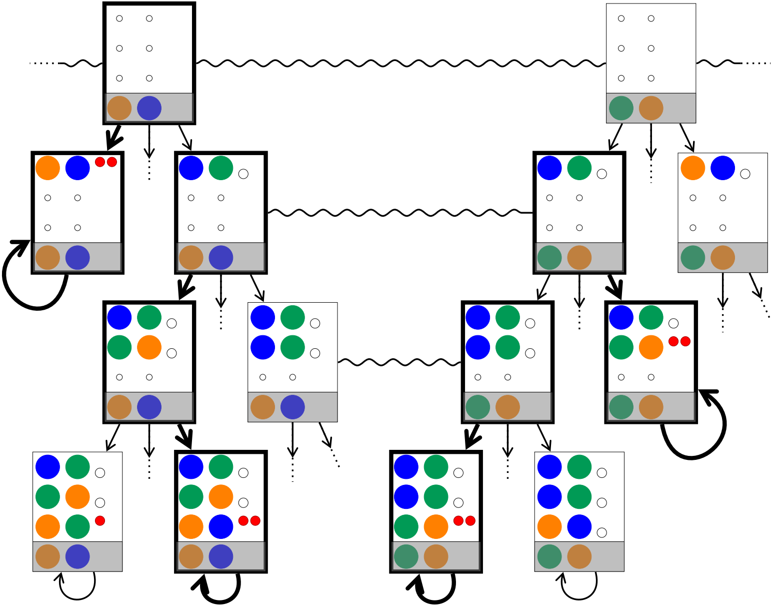
The backward approach



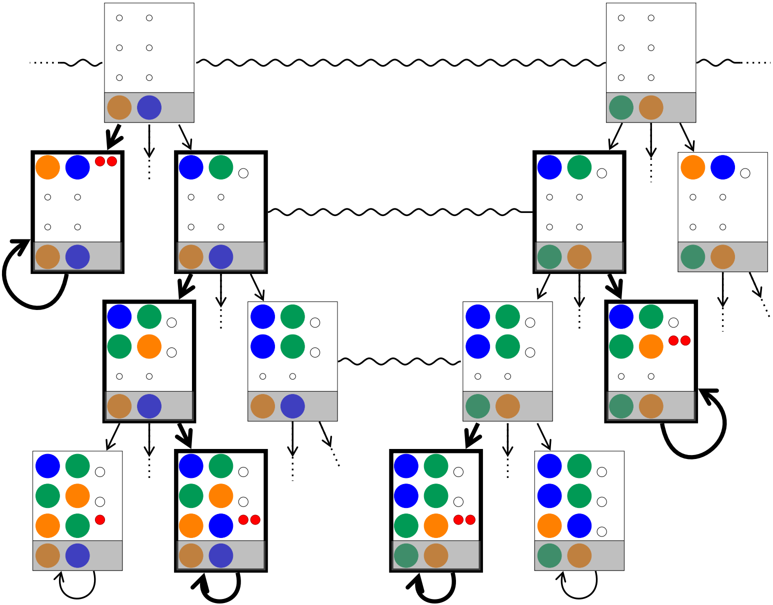
The backward approach



The backward approach



The backward approach



The backward approach

Build (parts of) winning strategies from the ground up

⇒ works for **reachability objectives**
(e.g. reach a winning state)

⇒ does not work for **safety objectives**
(e.g. avoid some losing state forever)

Outline

Comparison with other approaches

Experimentally compared the three approaches with two existing ones:

1. Pilecki et al.
2. Huang and van der Meyden

Enriched with pre-filtering

Comparison with other approaches

Experimentally compared the three approaches
with two existing ones:

1. Pilecki et al.
2. Huang and van der Meyden

Enriched with pre-filtering

Tested on 3 models, 6 properties

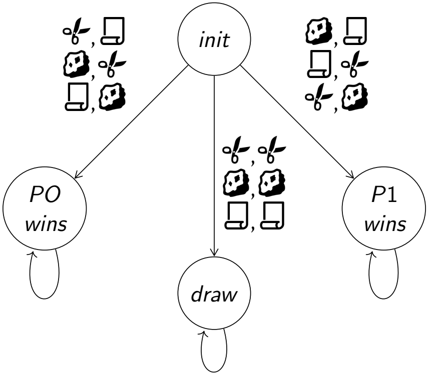
- $\Rightarrow$ the naive approach is inefficient
- $\Rightarrow$ pre-filtering sometimes helps
- $\Rightarrow$ no general winner,
  different approaches are better in different situations

Outline

Model checking uniform strategies: more than winning a game

Coalitions: reason about strategies of **multiple agents**

Concurrent models: agents play **at the same time**



Model checking uniform strategies: more than winning a game

Unconditional fairness constraints:

the player assumes the dealer is **fair**

i.e. if played infinitely often,
all cards a given infinitely often

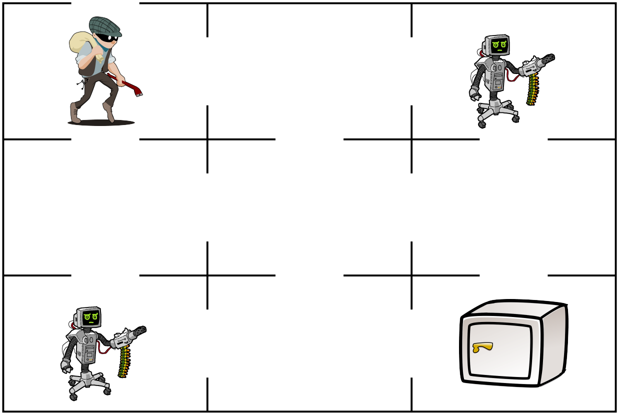
- ⇒ logic to reason
  about uniform strategies
  under fairness constraints
- ⇒ more complicated
  fixpoint computations

Thesis contributions

- A logic to reason about uniform strategies under fairness constraints
- Three techniques to check uniform strategies (naive, partial and backward approaches)
- Pre-filtering surely losing moves (+ application to the approaches)
- An implementation of these approaches with PyNuSMV
- An experimental comparison with existing approaches

Applications

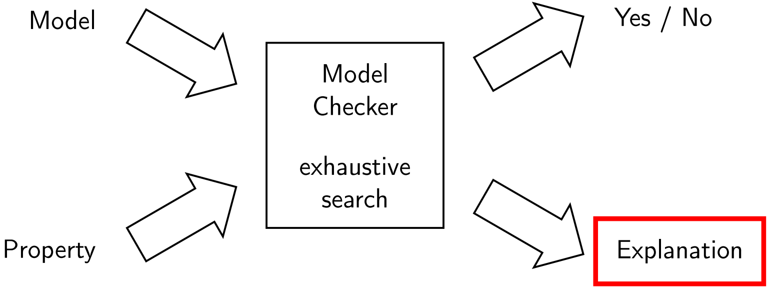
⇒ Security policies



⇒ Networks:
strategies of machines to share data through unreliable links

Outline

Model checking can produce explanations



Multi-modal logics have rich explanations

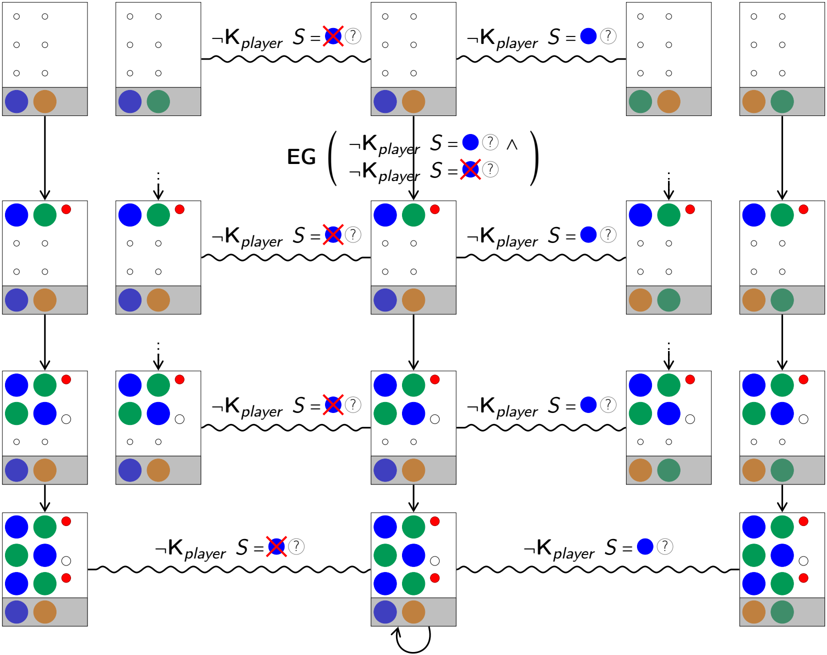
"The player always eventually knows whether the first peg is blue"

$$\mathbf{AF}\ (\mathsf{K}_{player}\ S = \bullet\,\textcircled{?} \vee \mathsf{K}_{player}\ S = \not\bullet\,\textcircled{?})$$

Counter-example =
a part of the model showing why the property is violated

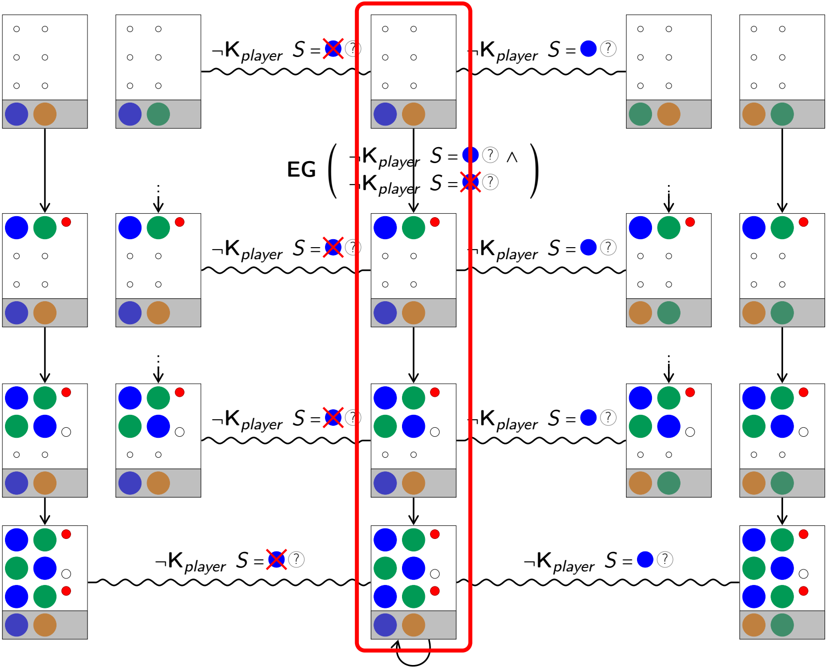
"There is a play along which the player never knows whether the first peg is blue"

$$\mathbf{EG}\ (\neg\mathsf{K}_{player}\ S = \bullet\,\textcircled{?} \wedge \neg\mathsf{K}_{player}\ S = \not\bullet\,\textcircled{?})$$


$\neg K_{player}\ S =$?
EG ($\neg K_{player}\ S =$? ∧ $\neg K_{player}\ S =$?)

The problem

- Such explanations are difficult to generate and manipulate
- State-of-the-art model checkers return partial explanations


¬K_player S = ?
¬K_player S = ?
EG (¬K_player S = ? ∧ ¬K_player S = ?)
¬K_player S = ?
¬K_player S = ?
¬K_player S = ?
¬K_player S = ?
¬K_player S = ?
¬K_player S = ?

Contribution

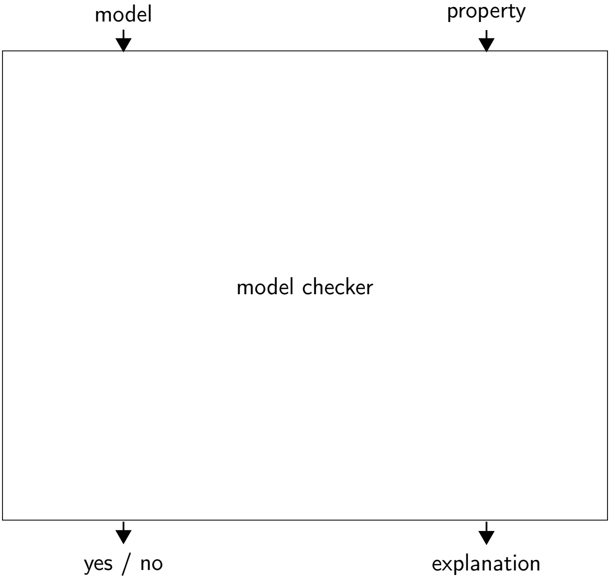
Many multi-modal logics can be translated into the mu-calculus:

(branching) time, knowledge, general strategies, etc.

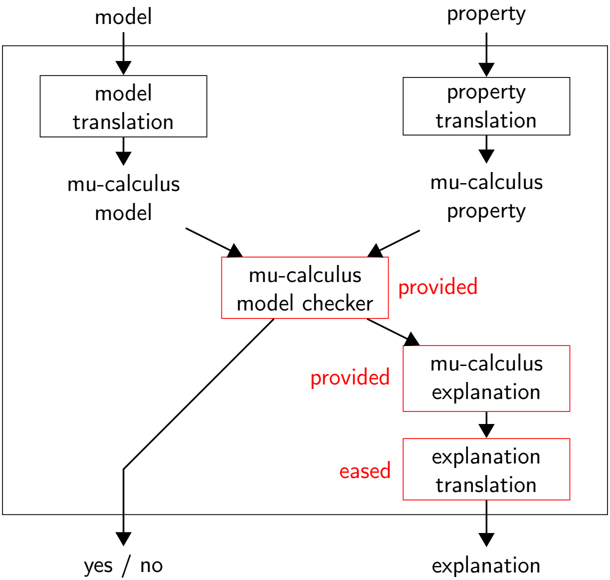
$\Rightarrow$ A mu-calculus-based model checking framework with rich explanations

(mu-calculus = a logic with modal and fixpoint operators)

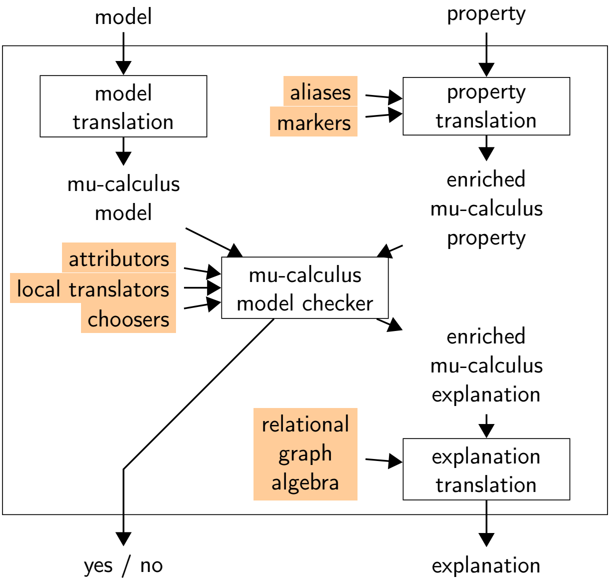
A mu-calculus based framework with rich explanations



A mu-calculus based framework with rich explanations



A mu-calculus based framework with rich explanations



Thesis contributions

- A mu-calculus based **model checker**...
- ...generating **rich explanations**...
- ...with features to **translate** them back into the original logic

- An **implementation** of the framework with PyNuSMV

- A **graphical tool** to visualize and manipulate the explanations

Applications to multi-modal logics: time, knowledge, strategies...

Outline

Conclusion

Two main contributions:

1. techniques to model check uniform strategies under fairness constraints
2. a mu-calculus based framework with rich explanations and translation features

$\Rightarrow$ The framework could be used to manipulate the uniform strategies built by the model checking techniques

Symbolic model checking of multi-modal logics: uniform strategies and rich explanations

(Model checking symbolique de logiques multi-modales : stratégies uniformes et explications riches)

Simon Busard

Public defense

UCL, July 3, 2017