Automated Synthesis of Generalized Invariant Strategies via Counterexample-guided Strategy Refinement

Kailun Luo[1], Yongmei Liu[2]

1.School of Cyberspace Security, Dongguan University of Technology

2.Department of Computer Science, Sun Yat-sen University

AAAI 2022

Motivation

1. Two player games with safety objectives are significant

Motivation

1. Two player games with safety objectives are significant
   - synthesis of reactive systems

Motivation

1. Two player games with safety objectives are significant
   - synthesis of reactive systems
   - first-order model-checking

Motivation

1. Two player games with safety objectives are significant
   - synthesis of reactive systems
   - first-order model-checking
2. Synthesizing strategies has proved to be hard tasks

Motivation

1. Two player games with safety objectives are significant
   - synthesis of reactive systems
   - first-order model-checking
2. Synthesizing strategies has proved to be hard tasks
   - non-elementarily decidable, Strategy Logic

Motivation

1. Two player games with safety objectives are significant
   - synthesis of reactive systems
   - first-order model-checking
2. Synthesizing strategies has proved to be hard tasks
   - non-elementarily decidable, Strategy Logic
   - P-complete, Alternating-time Temporal Logic

Motivation

1. Two player games with safety objectives are significant
   - synthesis of reactive systems
   - first-order model-checking
2. Synthesizing strategies has proved to be hard tasks
   - non-elementarily decidable, Strategy Logic
   - P-complete, Alternating-time Temporal Logic
   - P-complete, computing winning regions for winning memoryless strategies

Motivation

1. Two player games with safety objectives are significant
   - synthesis of reactive systems
   - first-order model-checking
2. Synthesizing strategies has proved to be hard tasks
   - non-elementarily decidable, Strategy Logic
   - P-complete, Alternating-time Temporal Logic
   - P-complete, computing winning regions for winning memoryless strategies
3. We focus on synthesizing generalized strategies to tackle the state-space explosion problem

Generalized Strategy Synthesis

- Multi-agent extensions of generalized planning

Generalized Strategy Synthesis

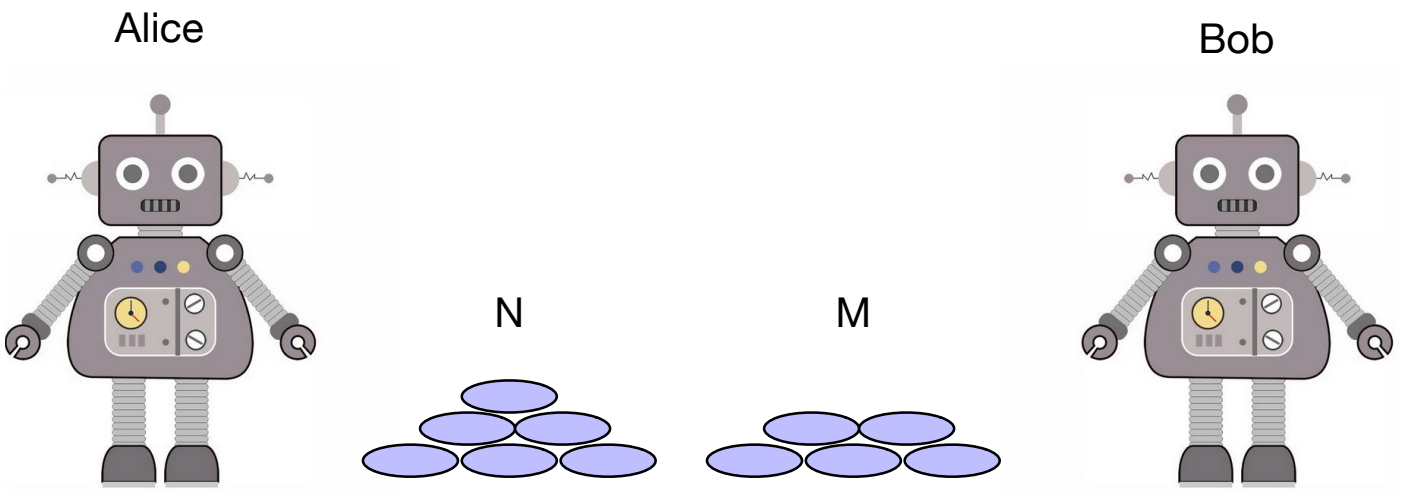
- Multi-agent extensions of generalized planning
- Goal: synthesize one strategy for a set of games with similar structures

Generalized Strategy Synthesis

- Multi-agent extensions of generalized planning
- Goal: synthesize one strategy for a set of games with similar structures
- Generalized strategy $\longrightarrow$ concrete strategy

Generalized Strategy Synthesis

- Multi-agent extensions of generalized planning
- Goal: synthesize one strategy for a set of games with similar structures
- Generalized strategy $\longrightarrow$ concrete strategy
- e.g., the two-pile Nim game



Contribution

- A general representing framework for generalized strategy, based on the situation calculus

Contribution

- A general representing framework for generalized strategy, based on the situation calculus
- A practical synthesis algorithm, based on the idea of invariant and counterexample-guided synthesis

Outline

1. Representation framework
   - how to represent a game problem
   - how to represent a generalized strategy
   - formalize the correctness
2. Automated synthesis algorithm
   - practical verification with invariants
   - counterexample-guided strategy refinement
   - experiments and results

Represent Game Problems: Basic Action Theories

- Initial database:

$$N(S_0) \neq M(S_0)\ ,\ N(S_0) > 0 \vee M(S_0) > 0$$

Represent Game Problems: Basic Action Theories

- Initial database:

$$N(S_0) \neq M(S_0)\ ,\ N(S_0) > 0 \vee M(S_0) > 0$$

- Precondition axioms: $Poss(removeN(x), s)$

$$\equiv N(s) \geq x \wedge x > 0$$

Represent Game Problems: Basic Action Theories

- Initial database:

$$N(S_0) \neq M(S_0)\ , N(S_0) > 0 \vee M(S_0) > 0$$

- Precondition axioms: $Poss(removeN(x), s)$

$$\equiv N(s) \geq x \wedge x > 0$$

- Successor state axioms: $N(do(a, s)) = y$

$$\equiv \exists x.a = removeN(x) \wedge N(s) = x + y$$
$$\vee N(s) = y \wedge \exists x.a = removeM(x)$$

Represent Game Problems: Basic Action Theories

- Initial database:

$$N(S_0) \neq M(S_0)\ , N(S_0) > 0 \vee M(S_0) > 0$$

- Precondition axioms: $Poss(removeN(x), s)$

$$\equiv N(s) \geq x \wedge x > 0$$

- Successor state axioms: $N(do(a, s)) = y$

$$\equiv \exists x.a = removeN(x) \wedge N(s) = x + y$$
$$\vee N(s) = y \wedge \exists x.a = removeM(x)$$

A BAT represents a class (possibly infinitely many) of games

Finite State Assumption

Definition

Given a game BAT $\mathcal{D}$, we define the finite-state axiom $\mathcal{D}_{fs}$ as the formula $\exists k \forall s.exec(s) \supset B(k,s)$, where $B(k,s)$ is the conjunction of the following formulas:

1. $\forall \vec{x}.\vec{x} > k \supset \neg F(\vec{x},s)$, for each relational fluent F;
2. $\forall \vec{x}.\vec{x} > k \supset f(\vec{x},s) = 0$, for each non-unary functional fluent;
3. $\forall \vec{x}.f(\vec{x},s) \leq k$, for each functional fluent f;
4. $\forall \vec{x}.\vec{x} > k \supset \neg Poss(A(\vec{x}),s)$, for each action A.

Finite State Assumption

Definition

Given a game BAT $\mathcal{D}$, we define the finite-state axiom $\mathcal{D}_{fs}$ as the formula $\exists k \forall s.exec(s) \supset B(k,s)$, where $B(k,s)$ is the conjunction of the following formulas:

1. $\forall \vec{x}.\vec{x} > k \supset \neg F(\vec{x},s)$, for each relational fluent F;
2. $\forall \vec{x}.\vec{x} > k \supset f(\vec{x},s) = 0$, for each non-unary functional fluent;
3. $\forall \vec{x}.f(\vec{x},s) \leq k$, for each functional fluent f;
4. $\forall \vec{x}.\vec{x} > k \supset \neg Poss(A(\vec{x}),s)$, for each action A.

Intuition: there is a number k such that all larger numbers are inactive throughout the game

Finite State Assumption

Definition

Given a game BAT $\mathcal{D}$, we define the finite-state axiom $\mathcal{D}_{fs}$ as the formula $\exists k \forall s.exec(s) \supset B(k,s)$, where $B(k,s)$ is the conjunction of the following formulas:

1. $\forall \vec{x}.\vec{x} > k \supset \neg F(\vec{x},s)$, for each relational fluent F;
2. $\forall \vec{x}.\vec{x} > k \supset f(\vec{x},s) = 0$, for each non-unary functional fluent;
3. $\forall \vec{x}.f(\vec{x},s) \leq k$, for each functional fluent f;
4. $\forall \vec{x}.\vec{x} > k \supset \neg Poss(A(\vec{x}),s)$, for each action A.

Intuition: there is a number k such that all larger numbers are inactive throughout the game

We assume that $\mathcal{D} \models \mathcal{D}_{\mathbf{fs}}$

Postdiction Strategies

A **Golog program** of the form $\varphi?; \pi a.a; \psi?$

- where φ and ψ are first-order formulas

Postdiction Strategies

A **Golog program** of the form $\varphi?; \pi a.a; \psi?$

- where φ and ψ are first-order formulas
- $\pi a.a$ denotes that there exists an action

Postdiction Strategies

A **Golog program** of the form $\varphi?; \pi a.a; \psi?$

- where φ and ψ are first-order formulas
- $\pi a.a$ denotes that there exists an action
- Intuition: whenever φ holds, perform any action to make ψ hold

Postdiction Strategies

A **Golog program** of the form $\varphi?; \pi a.a; \psi?$

- where φ and ψ are first-order formulas
- $\pi a.a$ denotes that there exists an action
- Intuition: whenever φ holds, perform any action to make ψ hold
- Advantage: simpler structures, closely related to memoryless strategies.

Postdiction Strategies

A **Golog program** of the form $\varphi?; \pi a.a; \psi?$

- where φ and ψ are first-order formulas
- $\pi a.a$ denotes that there exists an action
- Intuition: whenever φ holds, perform any action to make ψ hold
- Advantage: simpler structures, closely related to memoryless strategies.

Example

$N\%2 = 0?; \pi a.a; N\%2 = 1?$

Safe postdiction Strategies

- Composite strategy $\delta^*_{\mathcal{S}}$: $[turn(p)?; \mathcal{S} \mid \neg turn(p)?; \pi a.a]^*$

Safe postdiction Strategies

- Composite strategy $\delta^*_{\mathcal{S}}$: $[turn(p)?; \mathcal{S} \mid \neg turn(p)?; \pi a.a]^*$

Definition

Given a game problem $P = \langle \mathcal{D}, p, \phi \rangle$ and a postdiction strategy $\mathcal{S}$ for player p, we say that $\mathcal{S}$ is a solution to P if

$$\mathcal{D} \models \forall s \forall \delta. Trans^*(\delta^*_{\mathcal{S}}, S_0, \delta, s) \supset \phi[s] \wedge \exists s'. Trans(\delta_{\mathcal{S}}, s, nil, s').$$

- Configuration: (δ, s)
- $Trans(\delta, s, \delta', s')$: a transition from configuration (δ, s) to (δ', s') in one step
- $Trans^*$: the reflexive transitive closure of $Trans$

Automated Synthesis

Invariant Strategies

1. The definition of safe postdiction strategies involves second-order theorem proving

Invariant Strategies

1. The definition of safe postdiction strategies involves second-order theorem proving
2. Invariant strategies: First-order verifiable

Invariant Strategies

1. The definition of safe postdiction strategies involves second-order theorem proving
2. Invariant strategies: First-order verifiable
   - Intuition: maintain a *strong* first-order property, no matter how the opponent acts throughout the games

Invariant Strategies

1. The definition of safe postdiction strategies involves second-order theorem proving
2. Invariant strategies: First-order verifiable
   - Intuition: maintain a *strong* first-order property, no matter how the opponent acts throughout the games
   - Approximation of a safe postdiction strategy, relaxing the requirement of reachability

Verifying $\varphi?; \pi a.a; \psi?$ Being Invariant Strategies

1. Whenever it's p's turn to move and φ holds, p can execute an action to enforce ψ:

$$\varphi \wedge turn(p) \models \bigvee_{i=1}^{N} \exists \vec{x}.\mathcal{R}[Poss(A_i(\vec{x}), s) \wedge \psi(do(A_i(\vec{x}), s))]_{\downarrow}$$

Verifying $\varphi?; \pi a.a; \psi?$ Being Invariant Strategies

1. Whenever it's p's turn to move and φ holds, p can execute an action to enforce ψ:

$$\varphi \wedge turn(p) \models \bigvee_{i=1}^{N} \exists \vec{x}.\mathcal{R}[Poss(A_i(\vec{x}), s) \wedge \psi(do(A_i(\vec{x}), s))]_{\downarrow}$$

2. Whenever it's the opponent's turn to move and ψ holds, any action the opponent can execute makes φ true:

$$\psi \wedge \neg turn(p) \models \bigwedge_{i=1}^{N} \forall \vec{x}.\mathcal{R}[Poss(A_i(\vec{x}), s) \supset \varphi(do(A_i(\vec{x}), s))]_{\downarrow}$$

Verifying $\varphi?;\pi a.a;\psi?$ Being Invariant Strategies

1. Whenever it's p's turn to move and φ holds, p can execute an action to enforce ψ:

$$\varphi \wedge turn(p) \models \bigvee_{i=1}^{N} \exists \vec{x}.\mathcal{R}[Poss(A_i(\vec{x}), s) \wedge \psi(do(A_i(\vec{x}), s))]_{\downarrow}$$

2. Whenever it's the opponent's turn to move and ψ holds, any action the opponent can execute makes φ true:

$$\psi \wedge \neg turn(p) \models \bigwedge_{i=1}^{N} \forall \vec{x}.\mathcal{R}[Poss(A_i(\vec{x}), s) \supset \varphi(do(A_i(\vec{x}), s))]_{\downarrow}$$

3. Both φ and ψ imply the safety condition ϕ

Verifying $\varphi?; \pi a.a; \psi?$ Being Invariant Strategies

1. Whenever it's p's turn to move and φ holds, p can execute an action to enforce ψ:

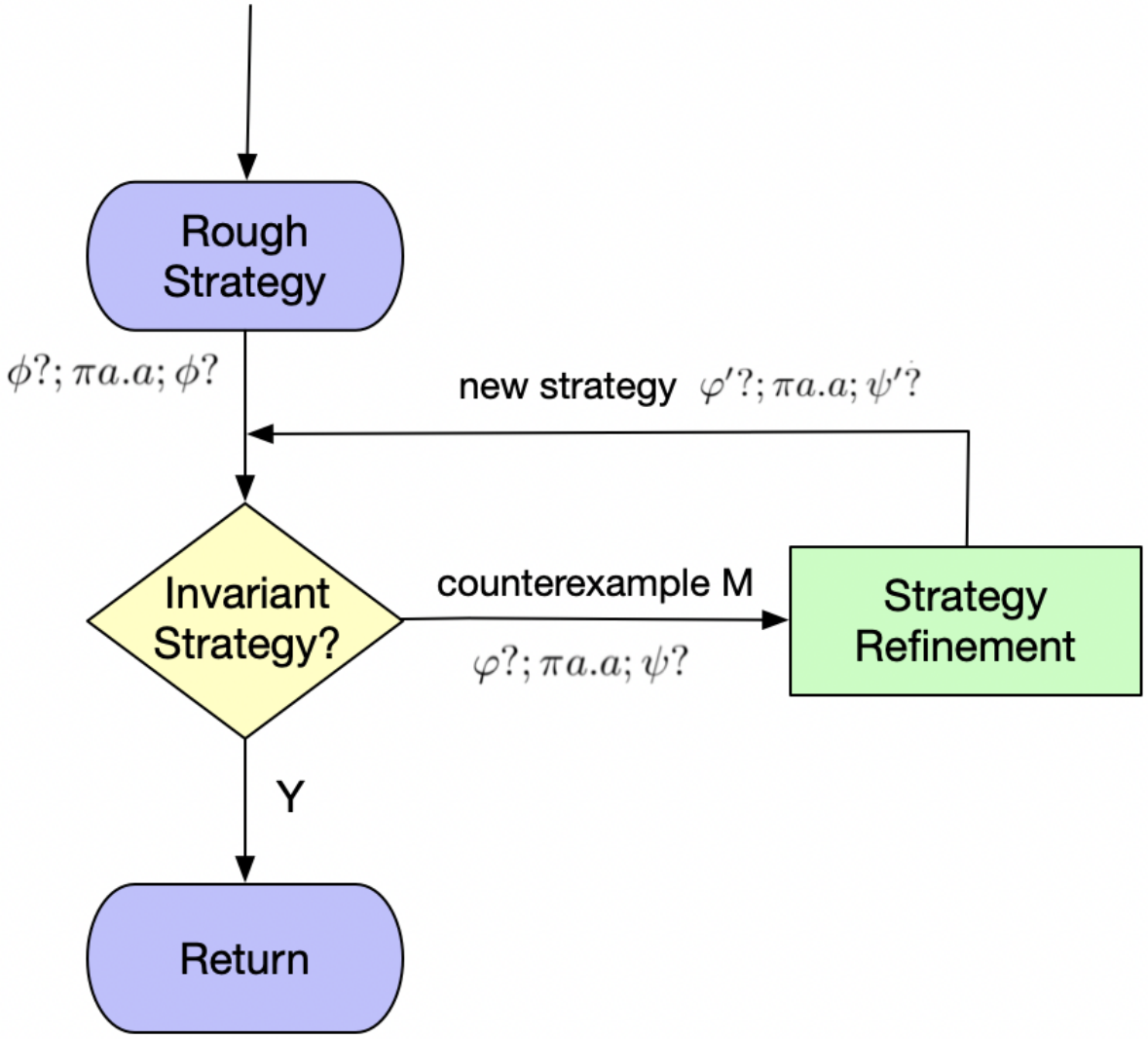
$$\varphi \wedge turn(p) \models \bigvee_{i=1}^{N} \exists \vec{x}.\mathcal{R}[Poss(A_i(\vec{x}), s) \wedge \psi(do(A_i(\vec{x}), s))]_{\downarrow}$$

2. Whenever it's the opponent's turn to move and ψ holds, any action the opponent can execute makes φ true:

$$\psi \wedge \neg turn(p) \models \bigwedge_{i=1}^{N} \forall \vec{x}.\mathcal{R}[Poss(A_i(\vec{x}), s) \supset \varphi(do(A_i(\vec{x}), s))]_{\downarrow}$$

3. Both φ and ψ imply the safety condition ϕ
4. If $p = P_1$, $\mathcal{D}_{S_0\downarrow} \models \varphi$; otherwise $\mathcal{D}_{S_0\downarrow} \models \psi$

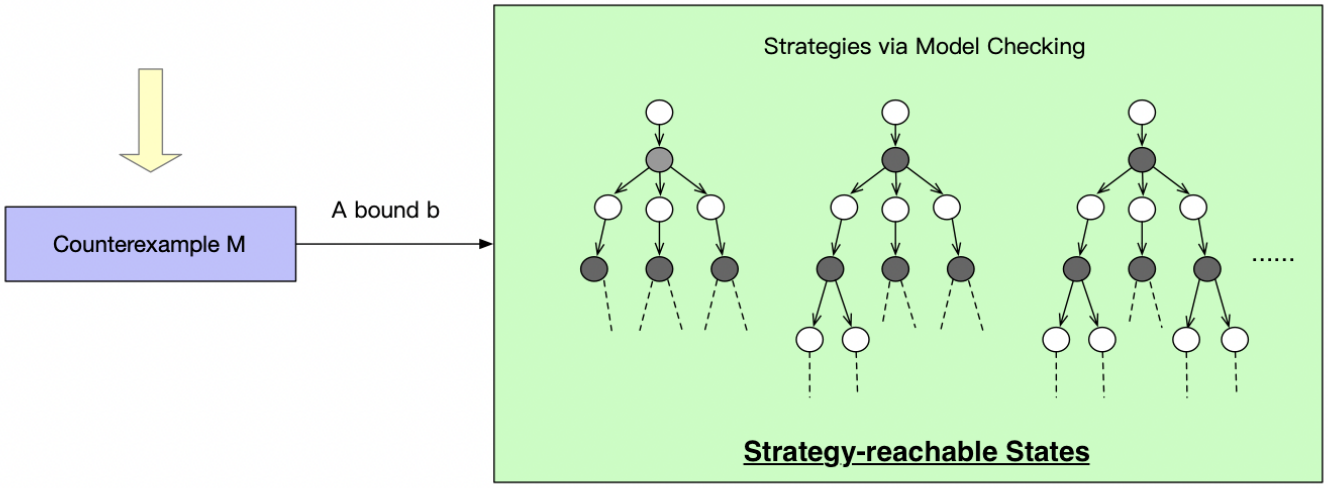
Synthesis Algorithm: General Picture



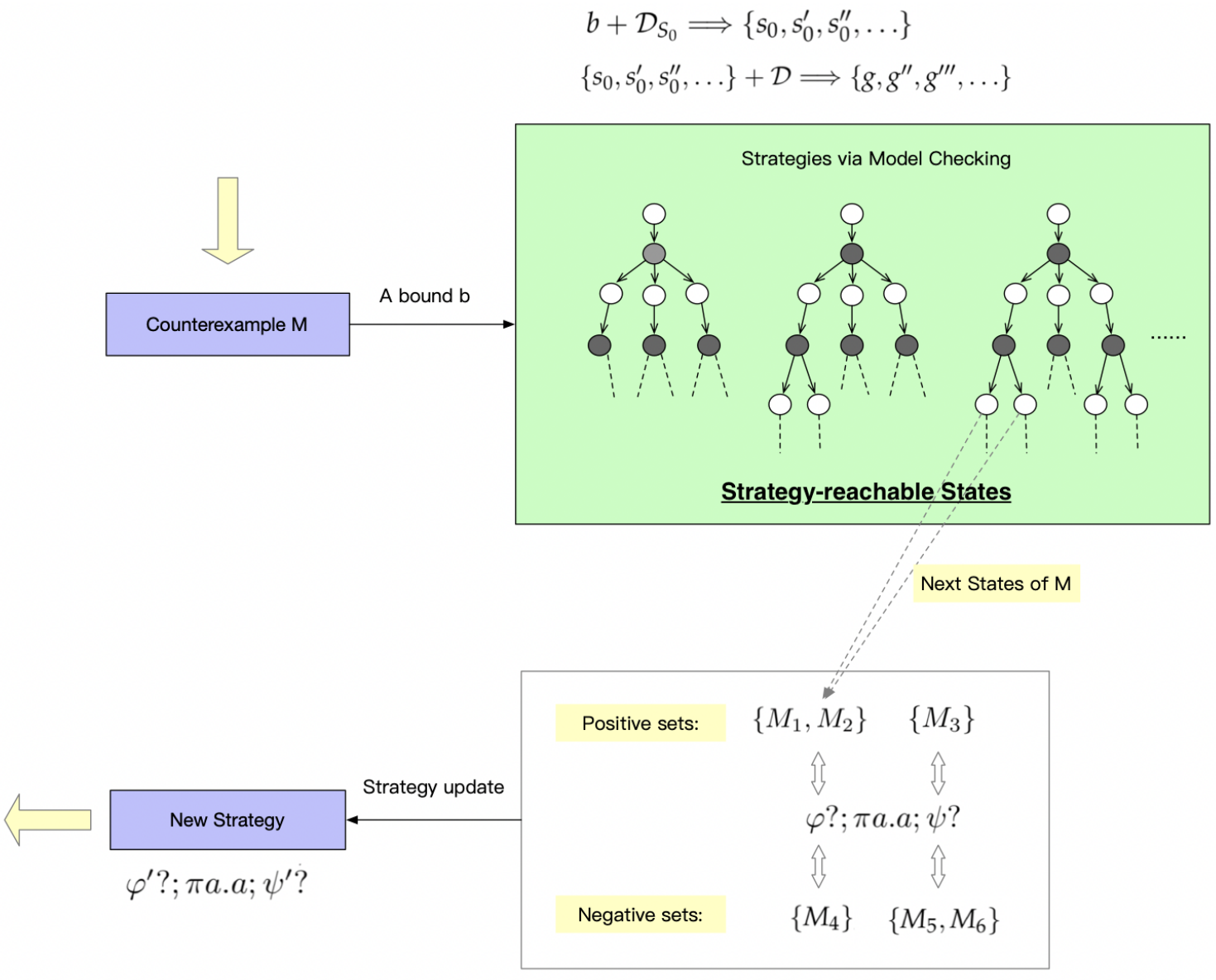
Counterexample-guided Strategy Refinement

$$b + \mathcal{D}_{S_0} \Longrightarrow \{s_0, s_0', s_0'', \ldots\}$$

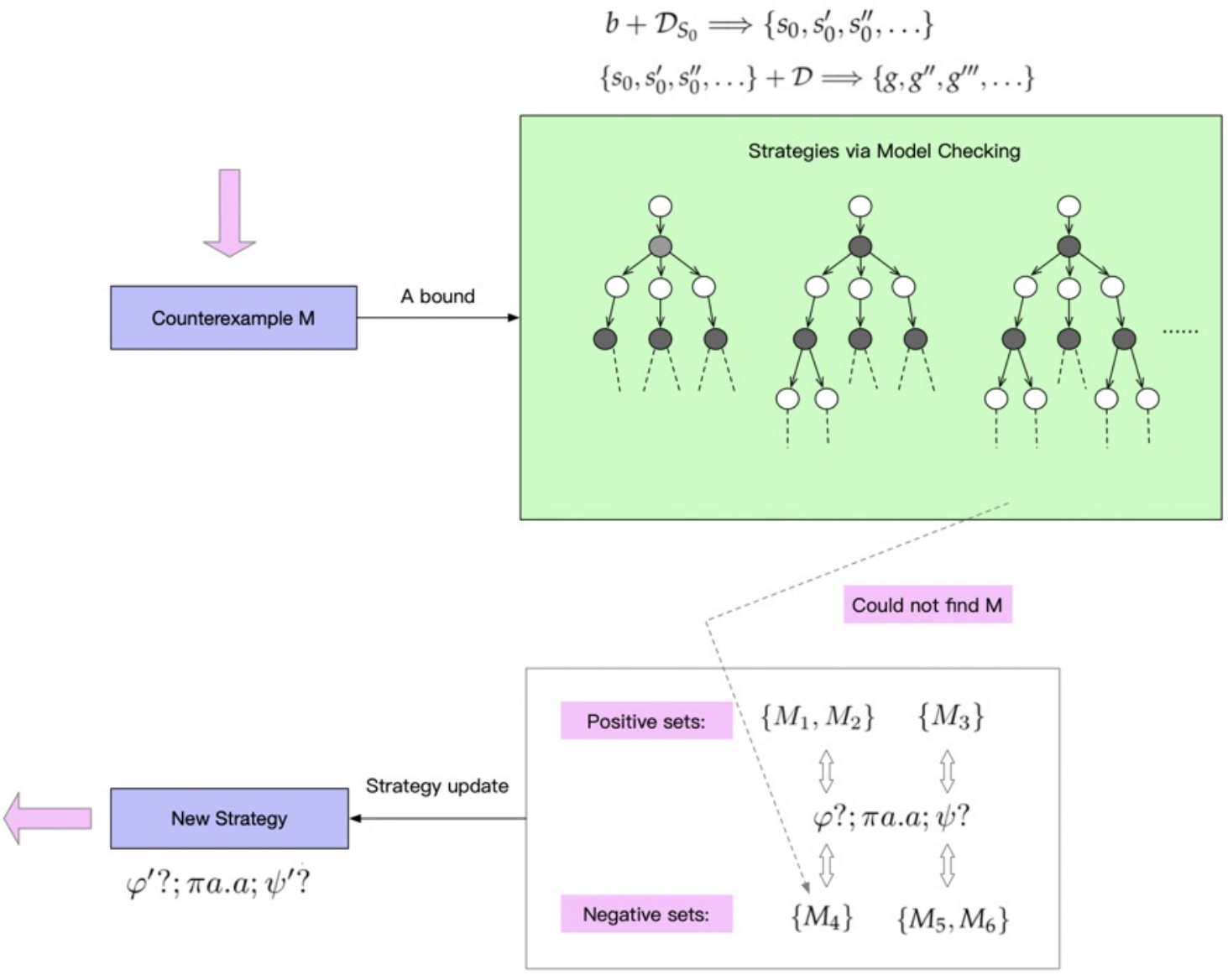
$$\{s_0, s_0', s_0'', \ldots\} + \mathcal{D} \Longrightarrow \{g, g'', g''', \ldots\}$$



Counterexample-guided Strategy Refinement



Counterexample-guided Strategy Refinement

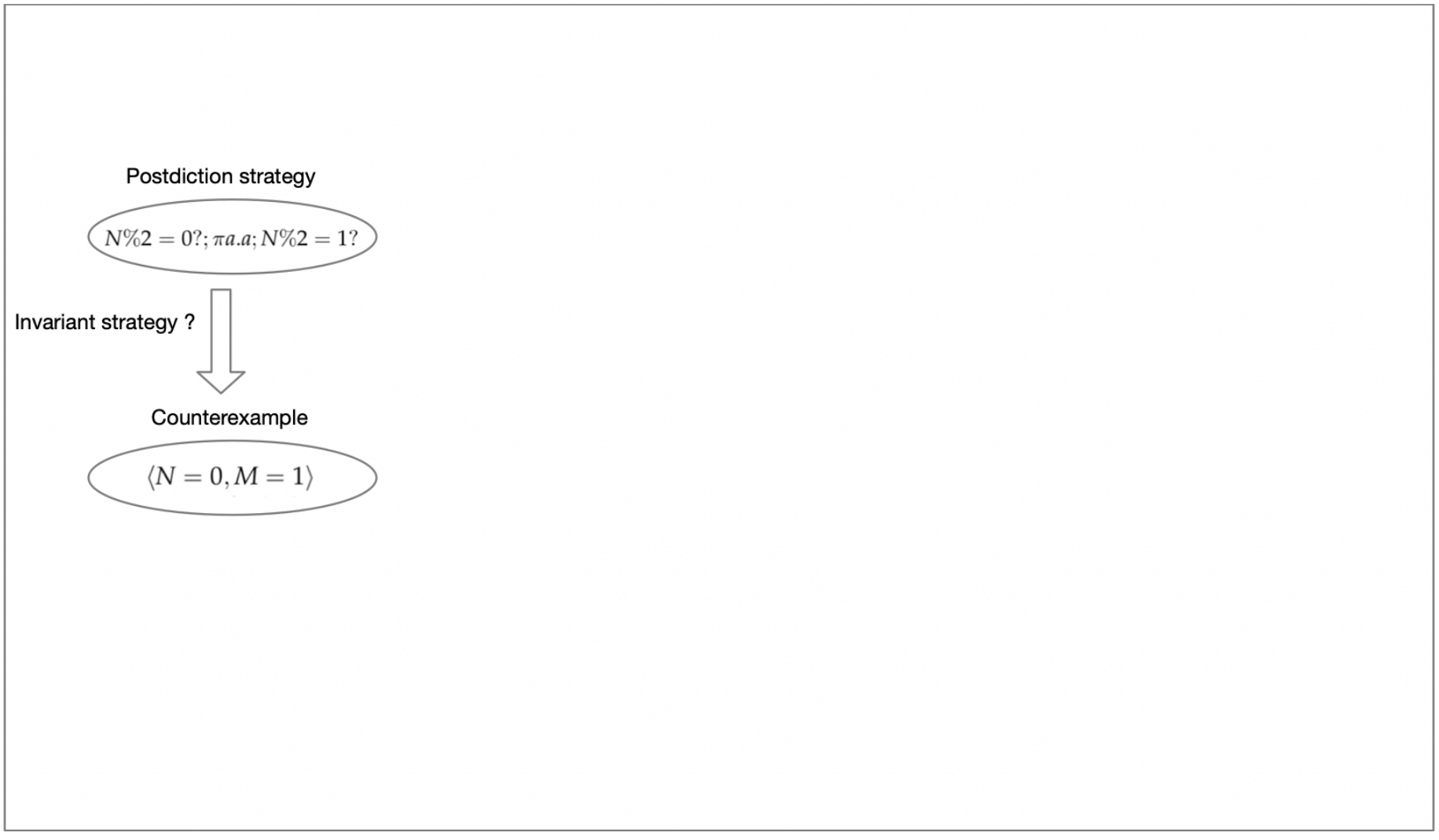


Example

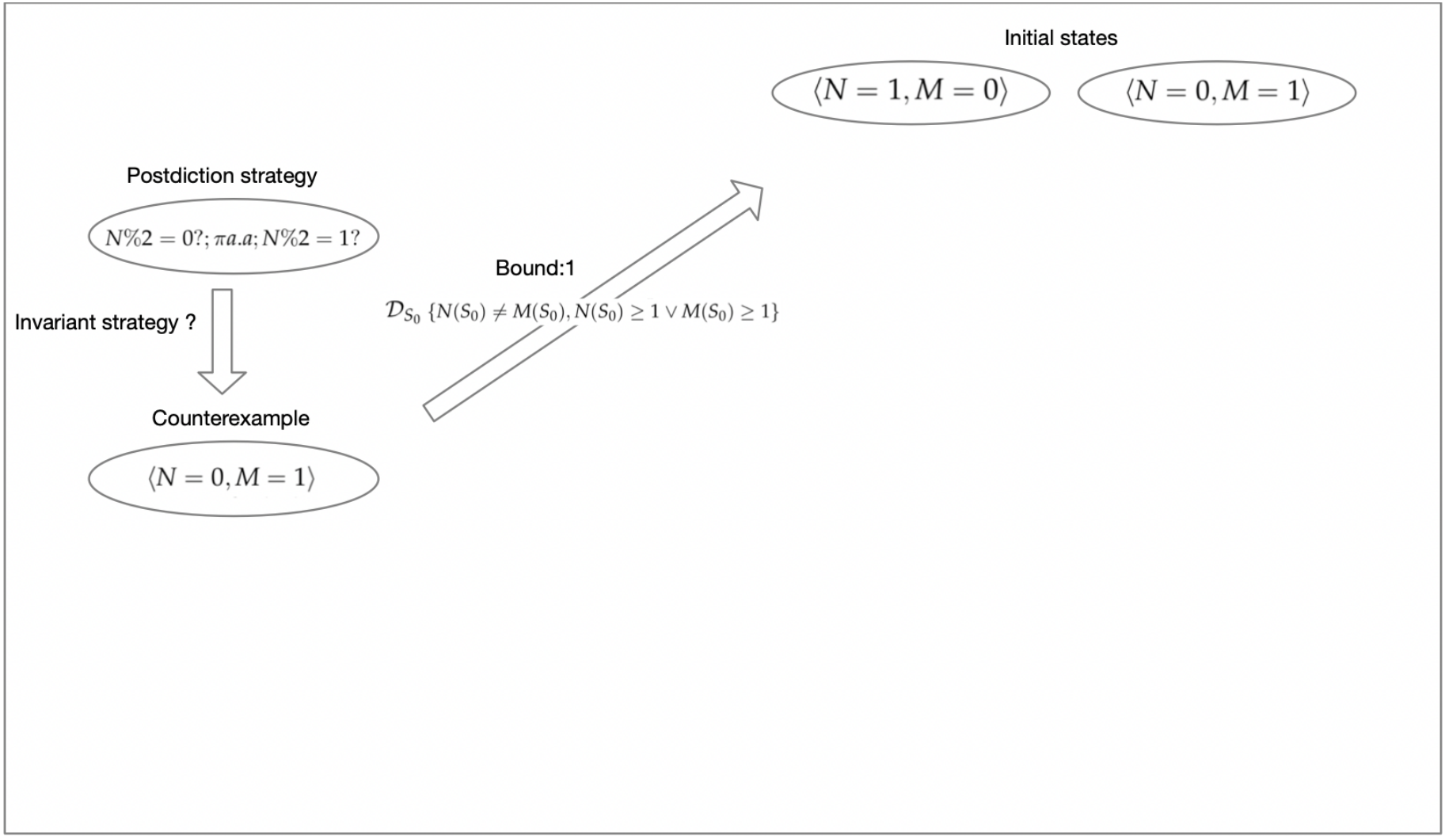
Postdiction strategy

$N\%2 = 0?; \pi a.a; N\%2 = 1?$

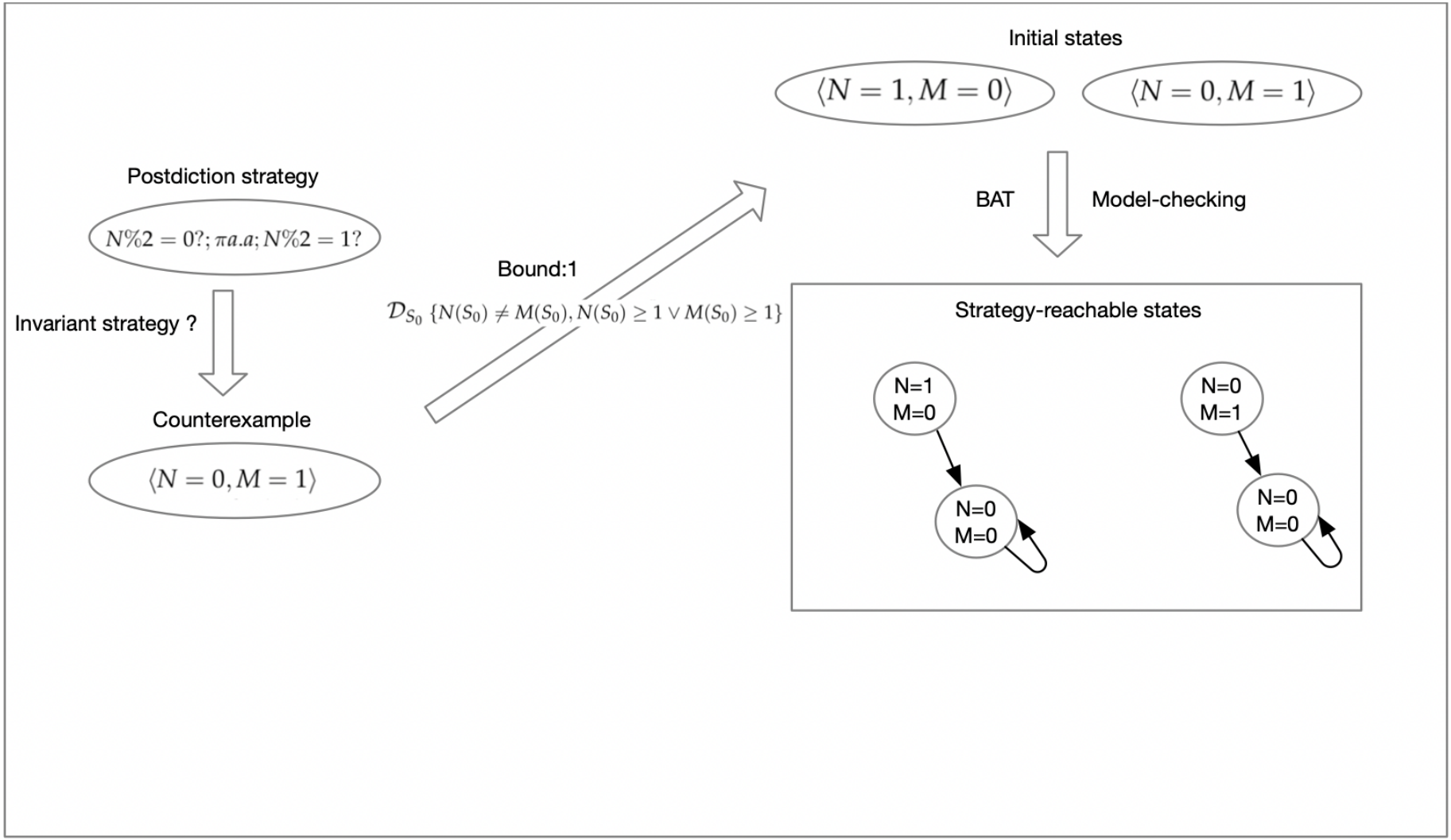
Example


Postdiction strategy
$N\%2 = 0?; \pi a.a; N\%2 = 1?$
Invariant strategy ?
Counterexample
$\langle N = 0, M = 1 \rangle$

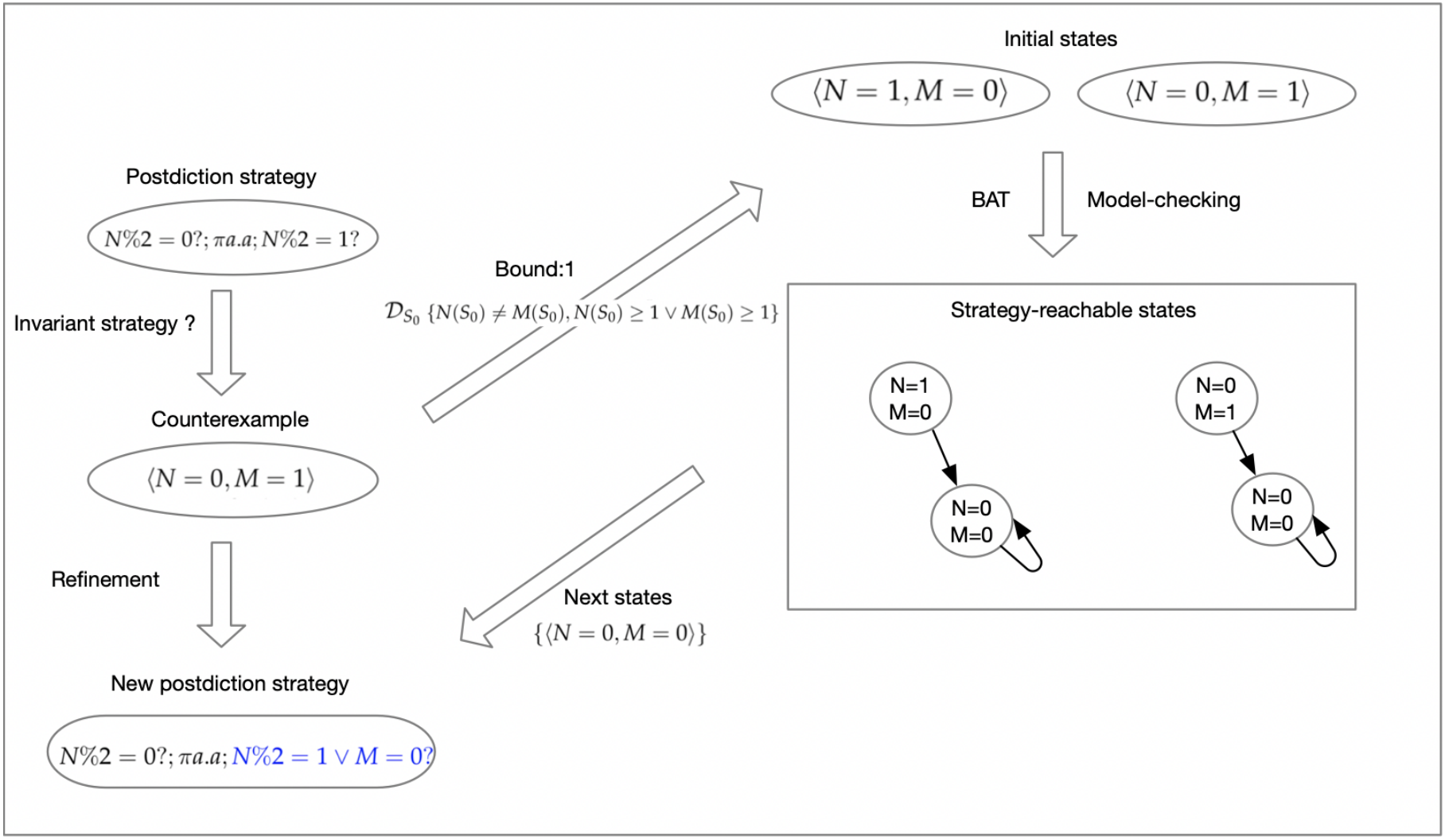
Example


Initial states
$\langle N = 1, M = 0 \rangle$
$\langle N = 0, M = 1 \rangle$
Postdiction strategy
$N\%2 = 0?; \pi a.a; N\%2 = 1?$
Bound:1
$\mathcal{D}_{S_0}\ \{N(S_0) \neq M(S_0), N(S_0) \geq 1 \vee M(S_0) \geq 1\}$
Invariant strategy ?
Counterexample
$\langle N = 0, M = 1 \rangle$

Example


Initial states
$\langle N = 1, M = 0 \rangle$
$\langle N = 0, M = 1 \rangle$
BAT
Model-checking
Strategy-reachable states
N=1
M=0
N=0
M=0
N=0
M=1
N=0
M=0
Postdiction strategy
$N\%2 = 0?; \pi a.a; N\%2 = 1?$
Invariant strategy ?
Counterexample
$\langle N = 0, M = 1 \rangle$
Bound:1
$\mathcal{D}_{S_0}\ \{N(S_0) \neq M(S_0), N(S_0) \geq 1 \vee M(S_0) \geq 1\}$

Example


Initial states
$\langle N = 1, M = 0 \rangle$
$\langle N = 0, M = 1 \rangle$
BAT
Model-checking
Strategy-reachable states
N=1
M=0
N=0
M=0
N=0
M=1
N=0
M=0
Bound:1
$\mathcal{D}_{S_0}\ \{N(S_0) \neq M(S_0), N(S_0) \geq 1 \vee M(S_0) \geq 1\}$
Next states
$\{\langle N = 0, M = 0 \rangle\}$
Postdiction strategy
$N\%2 = 0?; \pi a.a; N\%2 = 1?$
Invariant strategy ?
Counterexample
$\langle N = 0, M = 1 \rangle$
Refinement
New postdiction strategy
$N\%2 = 0?; \pi a.a; N\%2 = 1 \vee M = 0?$

Correctness of the Algorithm

Theorem

Given a game problem, if the algorithm returns a strategy, then it is a safe strategy.

Experimental Results

- Tools: CVC4, Z3, MCMAS

Experimental Results

- Tools: CVC4, Z3, MCMAS
- Tactics: quantifier elimination, different SMT solvers

Experimental Results

- Tools: CVC4, Z3, MCMAS
- Tactics: quantifier elimination, different SMT solvers
- Domains: combinatorial games, grid games, protocol

Experimental Results

- Tools: CVC4, Z3, MCMAS
- Tactics: quantifier elimination, different SMT solvers
- Domains: combinatorial games, grid games, protocol

game	L	R^+	R^-	B	S	T(s)
2-Nim	44	2	5	0	6	14.5
Take-away	94	3	11	1	6	43.2
Sub.	118	7	23	0	4	321.1
E.&D.	64	3	13	0	10	210.8
Mon. 2-Nim	44	1	9	0	6	26.4
Ch.2xN	44	2	12	0	14	35.6
Ch.NxN	58	–	–	–	–	–
Coloring	32	4	11	1	8	303.2
Leader	66	0	8	2	16	367.2

Table 1: Experimental results

Conclusion

- Generalized strategy synthesis problem for a set of games with safety goals

Conclusion

- Generalized strategy synthesis problem for a set of games with safety goals
- Invariant strategies, maintaining invariants no matter how the opponent acts

Conclusion

- Generalized strategy synthesis problem for a set of games with safety goals
- Invariant strategies, maintaining invariants no matter how the opponent acts
- A sound but incomplete method, counterexample-guided strategy refinement, strategies via ATL model-checking

Conclusion

- Generalized strategy synthesis problem for a set of games with safety goals
- Invariant strategies, maintaining invariants no matter how the opponent acts
- A sound but incomplete method, counterexample-guided strategy refinement, strategies via ATL model-checking
- Expressive strategies, to solve richer domains, such as those whose formalizations need quantified formulas

Future Work

1. Non-termination of the algorithm

Future Work

1. Non-termination of the algorithm
   - consider a class of game problems for which our algorithm will terminate

Future Work

1. Non-termination of the algorithm
   - consider a class of game problems for which our algorithm will terminate
2. Invariant strategies, simple structures, whose synthesis relies heavily on the synthesis of formulas

Future Work

1. Non-termination of the algorithm
   - consider a class of game problems for which our algorithm will terminate
2. Invariant strategies, simple structures, whose synthesis relies heavily on the synthesis of formulas
   - consider more complicated structures in strategies, such as finite-state automata (FSA)