Don't Trust Your Agents, Verify Them

Angelo Ferrando[1] and Vadim Malvone[2]

[1] Department of Physics, Informatics and Mathematics
University of Modena and Reggio Emilia
[2] INFRES department, LTCI lab, Télécom Paris
IDIA department, Institut Polytechnique de Paris

Outline

Outline

Outline

1. Introduction and Motivation
   - Why strategic verification for MAS?
   - From temporal properties to strategic abilities.
   - Domains and challenges.
2. Background: strategic logics and models
   - CTL/ATL intuition, concurrent game structures, strategies & coalitions, modelling assumptions and limits.
3. Strategic verification in VITAMIN
   - Architecture & components.
   - Supported logics and tasks.
   - Verification workflows.

Outline

Outline

1. Introduction and Motivation
   - Why strategic verification for MAS?
   - From temporal properties to strategic abilities.
   - Domains and challenges.
2. Background: strategic logics and models
   - CTL/ATL intuition, concurrent game structures, strategies & coalitions, modelling assumptions and limits.
3. Strategic verification in VITAMIN
   - Architecture & components.
   - Supported logics and tasks.
   - Verification workflows.
4. Hands-on modelling and verification
   - Build a model, write strategic specifications, execute tasks, analyse and discuss results.
5. Discussion and outlook
   - Open research challenges, industrial relevance, Q&A.

Introduction and Motivation

Preface

System Correctness

- Safety-critical and autonomous systems increasingly rely on **multiple interacting decision-makers**.
- Failures are rarely isolated; they emerge from **coordination** and **interaction**.

Preface

System Correctness

- Safety-critical and autonomous systems increasingly rely on **multiple interacting decision-makers**.
- Failures are rarely isolated; they emerge from **coordination** and **interaction**.

Formal Verification

- Provides mathematically precise models of behaviour.
- Enables reasoning about **all possible executions**.
- Model checking is a key technique for exploring system behaviour.

Model Checking (1)

$M \models \varphi$

There are three fundamental parts:

- M: modeling a multi-agent system;
- φ: specifying a property;
- $\models$: verifying that the model M satisfies the property φ.

Model checking (2)

Model

A Kripke structure is a tuple $K = \langle \mathrm{AP}, \mathrm{St}, S_0, R, L \rangle$, where:

- AP is a set of atomic propositions;
- St is a set of states;
- $S_0 \subseteq \mathrm{St}$ is a set of initial states;
- $R \subseteq \mathrm{St} \times \mathrm{St}$ is a transition relation;
- $L : \mathrm{St} \to 2^{\mathrm{AP}}$ is a labeling function.

Model checking (3)

Specification

- Temporal logics allow to describe the order of the events without define the time explicitly.
- The main temporal logics:
  - LTL $\Rightarrow$ in a computation the events are totally ordered.
    Examples: Xp, Fp, Gp.
  - CTL $\Rightarrow$ in a computation the events are partially ordered.
    Examples: AXp, EFp, AGp.

Model checking (4)

Verification

Given a Kripke structure K and a specification φ, the problem of the model checking consists to verify if $K, s \models \varphi$, for each initial state s.

Model checking (4)

Verification

Given a Kripke structure K and a specification φ, the problem of the model checking consists to verify if $K, s \models \varphi$, for each initial state s.

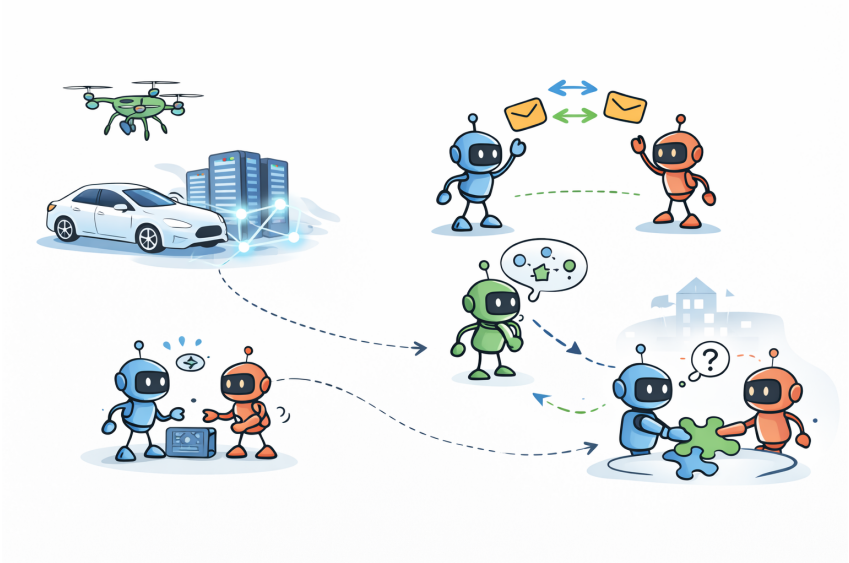
The challenge is scaling correctness to interacting agents.

Why Verifying Multi-Agent Systems Is Hard

- Modern systems are inherently **multi-agent**:
  - autonomous vehicles and robots
  - algorithmic trading systems
  - distributed protocols and platforms

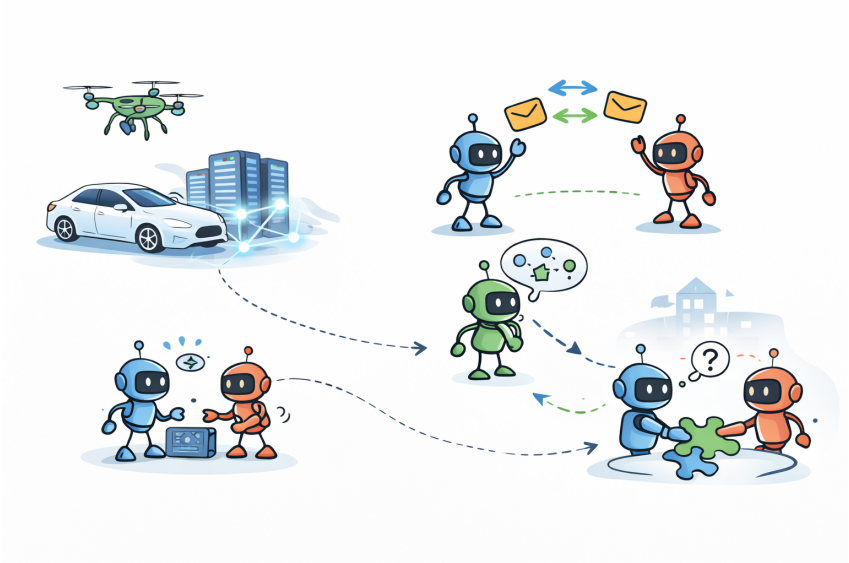
Why Verifying Multi-Agent Systems Is Hard

- Modern systems are inherently **multi-agent**:
  - autonomous vehicles and robots
  - algorithmic trading systems
  - distributed protocols and platforms
- They combine:
  - **concurrency**
  - **partial information**
  - **strategic decision-making**



Why Verifying Multi-Agent Systems Is Hard

- Modern systems are inherently **multi-agent**:
  - autonomous vehicles and robots
  - algorithmic trading systems
  - distributed protocols and platforms
- They combine:
  - **concurrency**
  - **partial information**
  - **strategic decision-making**



Correctness depends on interactions, not just components.

Local Correctness Is Not Enough

Observation

Most failures in distributed and multi-agent systems are not caused by incorrect local behaviour, but by **unexpected interactions**.

Local Correctness Is Not Enough

Observation

Most failures in distributed and multi-agent systems are not caused by incorrect local behaviour, but by **unexpected interactions**.

- each component follows its specification
- assumptions hold locally
- global properties silently break

Local Correctness Is Not Enough

Observation

Most failures in distributed and multi-agent systems are not caused by incorrect local behaviour, but by **unexpected interactions**.

- each component follows its specification
- assumptions hold locally
- global properties silently break

The real difficulty lies in composition.

From Interaction to Strategic Behaviour

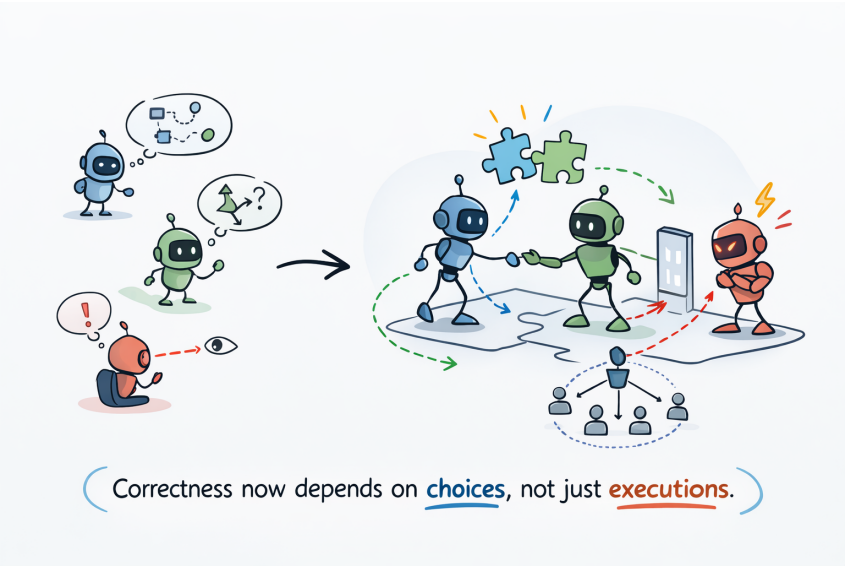
- Many components are not purely reactive:
  - they plan
  - they adapt
  - they anticipate others

From Interaction to Strategic Behaviour

- Many components are not purely reactive:
  - they plan
  - they adapt
  - they anticipate others
- This introduces strategic dimensions:
  - cooperation and competition
  - adversarial behaviour
  - coalitions and authority

From Interaction to Strategic Behaviour

- Many components are not purely reactive:
  - they plan
  - they adapt
  - they anticipate others
- This introduces strategic dimensions:
  - cooperation and competition
  - adversarial behaviour
  - coalitions and authority



Why We Start from Real Failures

- Interaction failures are often:
  - rare
  - emergent
  - invisible to testing

Why We Start from Real Failures

- Interaction failures are often:
  - rare
  - emergent
  - invisible to testing
- They expose:
  - broken assumptions
  - unsafe compositions
  - missing strategic guarantees

Failures reveal what models and specifications must capture.

Policy Composition Failure: BGP Hijack

Domain: Internet routing **Year:** 2008

What happened

A local routing policy accidentally advertised a more specific prefix for YouTube, which propagated globally and redirected traffic worldwide.

Policy Composition Failure: BGP Hijack

Domain: Internet routing **Year:** 2008

What happened

A local routing policy accidentally advertised a more specific prefix for YouTube, which propagated globally and redirected traffic worldwide.

Why this is a MAS failure

Autonomous network agents followed correct local rules whose composition violated global reachability.

Policy Composition Failure: BGP Hijack

Domain: Internet routing **Year:** 2008

What happened

A local routing policy accidentally advertised a more specific prefix for YouTube, which propagated globally and redirected traffic worldwide.

Why this is a MAS failure

Autonomous network agents followed correct local rules whose composition violated global reachability.

Verification lesson

Global safety requires verifying **policy composition** among agents.

Strategic Feedback Loops: Flash Crash

Domain: Algorithmic markets **Year:** 2010

What happened

A large automated sell program interacted with high-frequency trading strategies, triggering a rapid market collapse and rebound within minutes.

Strategic Feedback Loops: Flash Crash

Domain: Algorithmic markets **Year:** 2010

What happened

A large automated sell program interacted with high-frequency trading strategies, triggering a rapid market collapse and rebound within minutes.

Why this is a MAS failure

Strategic agents adapted to each other at machine speed, producing unstable feedback loops.

Strategic Feedback Loops: Flash Crash

Domain: Algorithmic markets **Year:** 2010

What happened

A large automated sell program interacted with high-frequency trading strategies, triggering a rapid market collapse and rebound within minutes.

Why this is a MAS failure

Strategic agents adapted to each other at machine speed, producing unstable feedback loops.

Verification lesson

Safety depends on the **composition of strategies**, not on isolated agent.

Algorithmic Trading: Partial Deployment

Domain: Financial markets **Year:** 2012

What happened

A faulty software update activated obsolete trading logic on one server. Within minutes, millions of unintended orders were issued, causing massive losses.

Algorithmic Trading: Partial Deployment

Domain: Financial markets **Year:** 2012

What happened

A faulty software update activated obsolete trading logic on one server. Within minutes, millions of unintended orders were issued, causing massive losses.

Why this is a MAS failure

Independent trading agents operated with inconsistent internal states due to partial deployment and concurrency.

Algorithmic Trading: Partial Deployment

Domain: Financial markets **Year:** 2012

What happened

A faulty software update activated obsolete trading logic on one server. Within minutes, millions of unintended orders were issued, causing massive losses.

Why this is a MAS failure

Independent trading agents operated with inconsistent internal states due to partial deployment and concurrency.

Verification lesson

Correctness must be verified under **heterogeneous agents and mixed versions**, not idealised uniform deployments.

Smart Contracts: Strategic Exploitation (The DAO)

Domain: Blockchain / smart contracts **Year:** 2016

What happened

An attacker exploited a re-entrancy vulnerability to repeatedly withdraw funds before global balances were updated, draining a large fraction of the contract.

Smart Contracts: Strategic Exploitation (The DAO)

Domain: Blockchain / smart contracts **Year:** 2016

What happened

An attacker exploited a re-entrancy vulnerability to repeatedly withdraw funds before global balances were updated, draining a large fraction of the contract.

Why this is a MAS failure

The contract encoded a multi-agent agreement but failed under adversarial, strategic behaviour exploiting temporal and interaction assumptions.

Smart Contracts: Strategic Exploitation (The DAO)

Domain: Blockchain / smart contracts **Year:** 2016

What happened

An attacker exploited a re-entrancy vulnerability to repeatedly withdraw funds before global balances were updated, draining a large fraction of the contract.

Why this is a MAS failure

The contract encoded a multi-agent agreement but failed under adversarial, strategic behaviour exploiting temporal and interaction assumptions.

Verification lesson

Strategic reasoning is essential to guarantee global invariants against **adversarial strategies**.

Autonomous Driving: Conflicting Agents (Uber)

Domain: Autonomous vehicles **Year:** 2018

What happened

An autonomous vehicle struck a pedestrian while operating in automated mode. The perception system repeatedly misclassified the pedestrian, while emergency braking was disabled, relying on human intervention.

Autonomous Driving: Conflicting Agents (Uber)

Domain: Autonomous vehicles **Year:** 2018

What happened

An autonomous vehicle struck a pedestrian while operating in automated mode. The perception system repeatedly misclassified the pedestrian, while emergency braking was disabled, relying on human intervention.

Why this is a MAS failure

Multiple agents (automation, human supervisor, safety policies) acted with partial and delayed information, leading to uncoordinated responsibility.

Autonomous Driving: Conflicting Agents (Uber)

Domain: Autonomous vehicles **Year:** 2018

What happened

An autonomous vehicle struck a pedestrian while operating in automated mode. The perception system repeatedly misclassified the pedestrian, while emergency braking was disabled, relying on human intervention.

Why this is a MAS failure

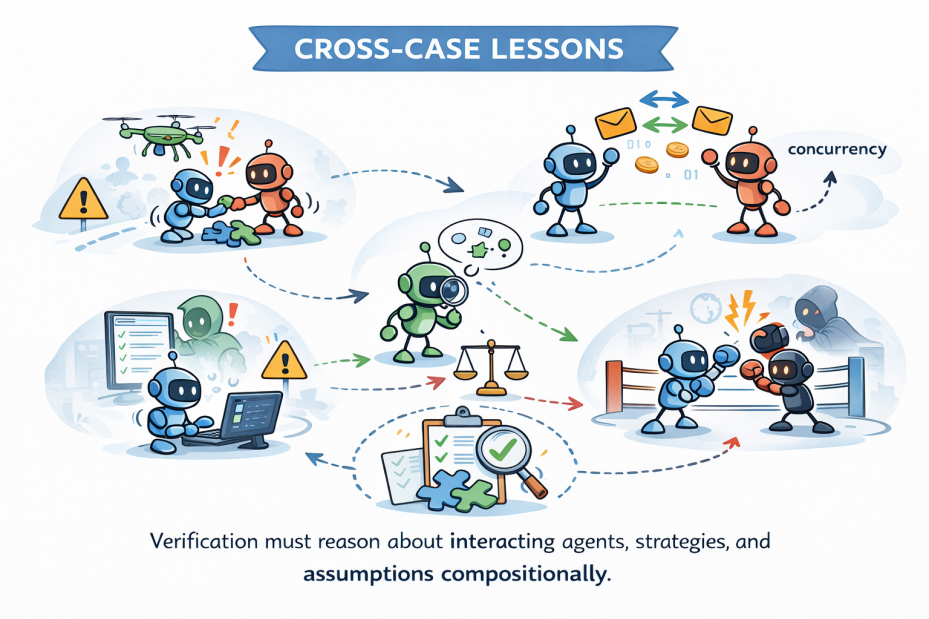
Multiple agents (automation, human supervisor, safety policies) acted with partial and delayed information, leading to uncoordinated responsibility.

Verification lesson

Safety depends on **authority and strategy composition** between human and autonomous agents, not on any single controller.

Cross-Case Lessons

- Failures emerge from **interaction**, not local bugs
- Partial information and concurrency are the norm
- Strategic and adversarial behaviour cannot be ignored
- Assumptions do not compose automatically



From Failures to Formal Reasoning

- The case studies show that failures arise from:
  - interacting components
  - concurrent decisions
  - strategic behaviour under partial information
- Informal reasoning and testing are insufficient to cover all cases.

From Failures to Formal Reasoning

- The case studies show that failures arise from:
  - interacting components
  - concurrent decisions
  - strategic behaviour under partial information
- Informal reasoning and testing are insufficient to cover all cases.

Key question

How can we **systematically reason** about *all possible behaviours* of interacting agents?

From Failures to Formal Reasoning

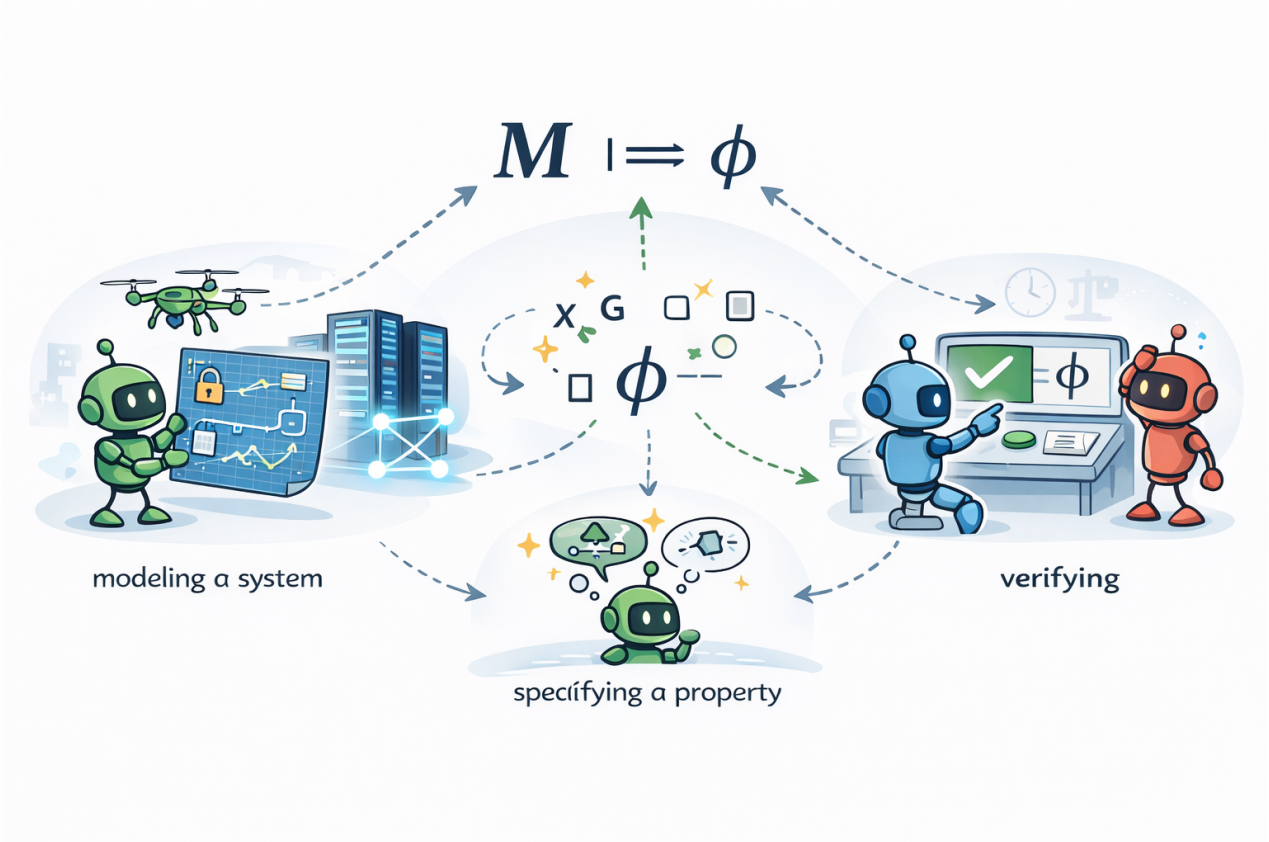
- The case studies show that failures arise from:
  - interacting components
  - concurrent decisions
  - strategic behaviour under partial information
- Informal reasoning and testing are insufficient to cover all cases.

Key question

How can we **systematically reason** about *all possible behaviours* of interacting agents?

This is the role of model checking.

Model Checking



BACKGROUND: STRATEGIC LOGICS AND MODELS

Multi-agent Systems (1)

Key aspects

- There are many agents interacting among them.
- Each agent selects a *strategy*.
- A *strategy* is a conditional plan that prescribes an action at each step of the Multi-Agent System (MAS).
- The composition of strategies induces a computation.

Multi-agent Systems (2)

Model

- A MAS structure is a tuple $M =< \mathrm{AP}, St, s_I, Ac, Ag, L, tr >$, where:
  - AP is a set of atomic propositions;
  - St is a set of states;
  - $s_I \in S$ is a designated initial state;
  - Ac is a set of actions;
  - Ag is a set of agents;
  - L is a labeling function;
  - tr is a transition function.

Multi-agent Systems (2)

Model

- A MAS structure is a tuple $M =< \mathrm{AP}, St, s_I, Ac, Ag, L, tr >$, where:
  - AP is a set of atomic propositions;
  - St is a set of states;
  - $s_I \in S$ is a designated initial state;
  - Ac is a set of actions;
  - Ag is a set of agents;
  - L is a labeling function;
  - tr is a transition function.
- The function tr depends on the interaction between the agents:
  - *Turn-based* $\Rightarrow$ The states of the MAS are partitioned between the agents, then the owner of a state determines the move to take and thus the next state of the MAS.
  - *Concurrent* $\Rightarrow$ The agents choose a move (i.e, actions) simultaneously and independently, and the joint action (i.e. the choices together) determine the next state of the MAS.

Multi-agent Systems (3)

Strategies

Depending on the memory, we distinguish between:

Multi-agent Systems (3)

Strategies

Depending on the memory, we distinguish between:

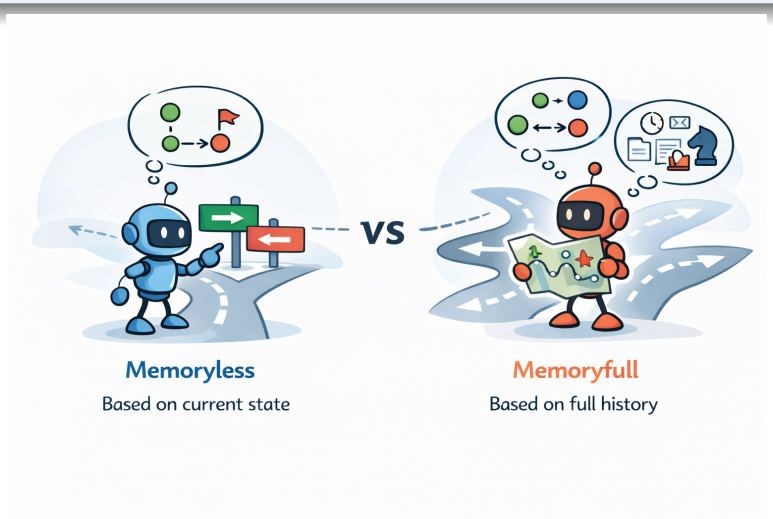
- *Memoryless strategies*
  $\sigma : St \to \mathrm{Ac}$
  Decisions depend only on the current MAS state.

Multi-agent Systems (3)

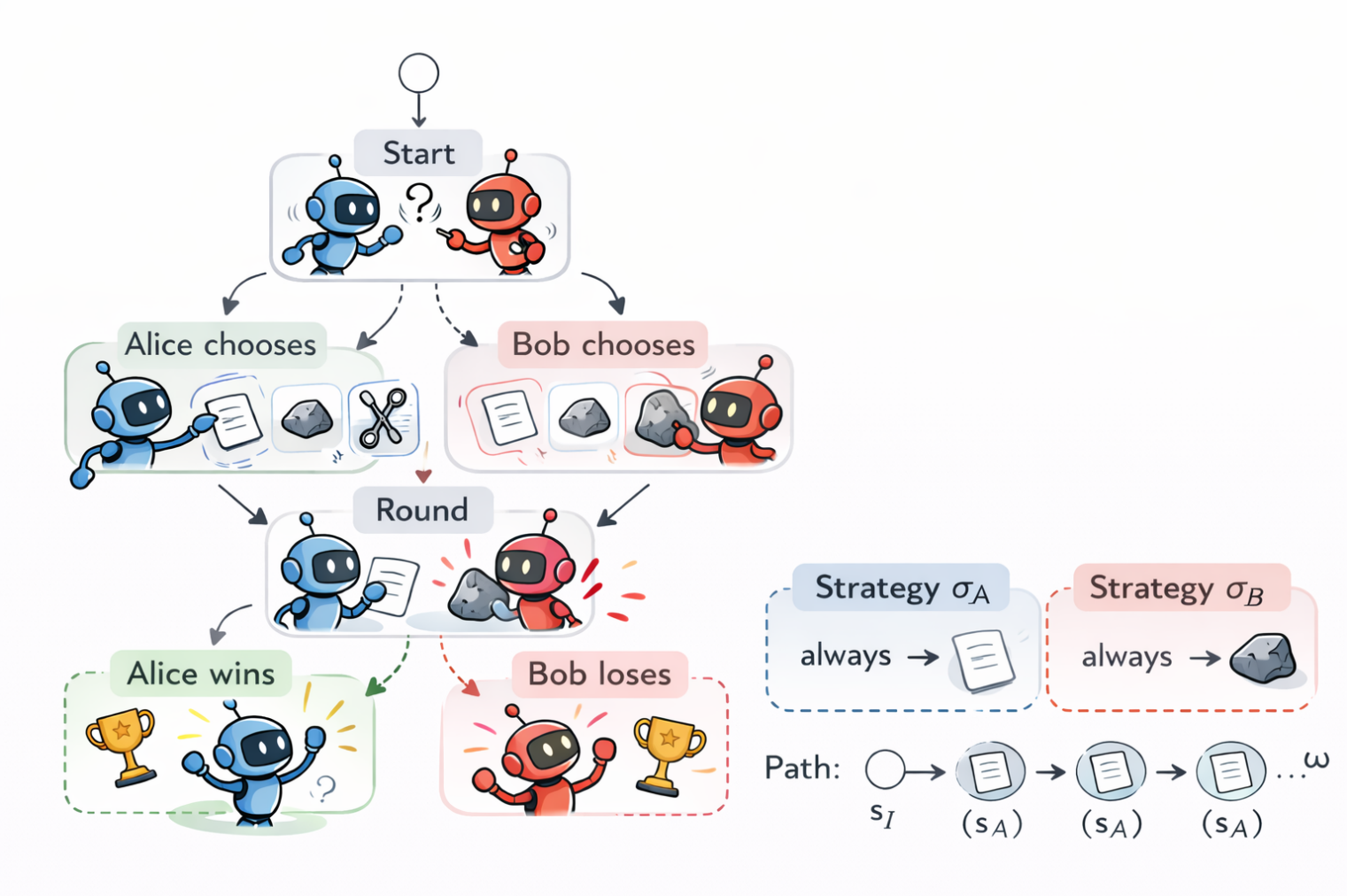
Strategies

Depending on the memory, we distinguish between:

- *Memoryless strategies*
  $\sigma : St \to \mathrm{Ac}$
  Decisions depend only on the current MAS state.
- *Memoryful strategies*
  $\sigma : St^+ \to \mathrm{Ac}$
  Decisions depend on the full history of the MAS.



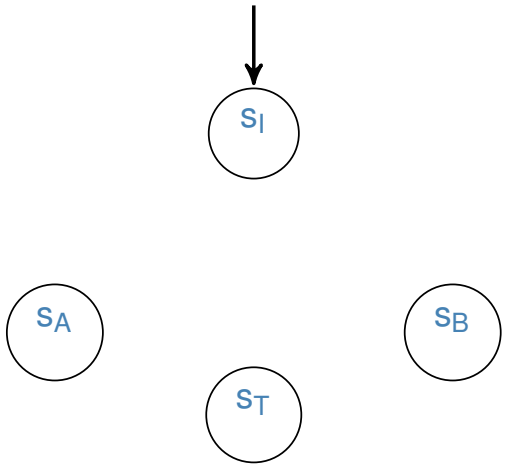
Example: paper, rock, and scissor (1)



Example: paper, rock, and scissor (2)

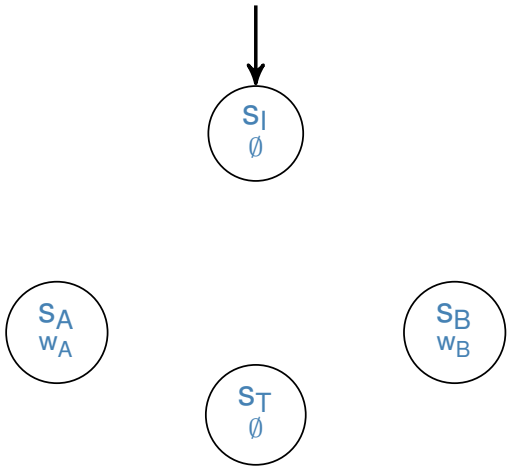
- $\mathrm{Ag} = \{\mathsf{A}$: *Alice*$, \mathsf{B}$: *Bob*$\}$
- $\mathrm{Ac} = \{\mathsf{P}$: *Paper*$, \mathsf{R}$: *Rock*$, \mathsf{S}$: *Scissor*$\}$

Example: paper, rock, and scissor (2)



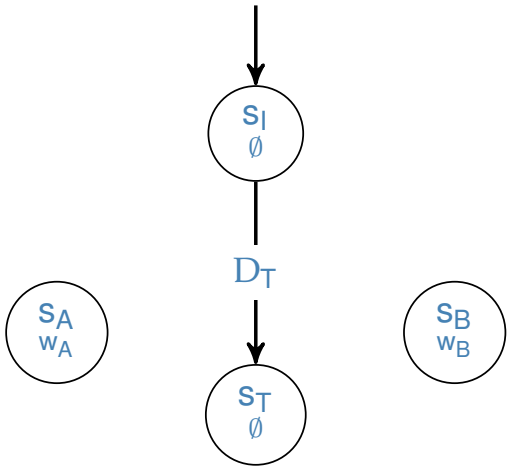
- $\mathrm{Ag} = \{\mathsf{A}: \textit{Alice}, \mathsf{B}: \textit{Bob}\}$
- $\mathrm{Ac} = \{\mathsf{P}: \textit{Paper}, \mathsf{R}: \textit{Rock}, \mathsf{S}: \textit{Scissor}\}$
- $\mathrm{St} = \{s_I, s_A, s_T, s_B\}$
- s_I: *initial state*

Example: paper, rock, and scissor (2)



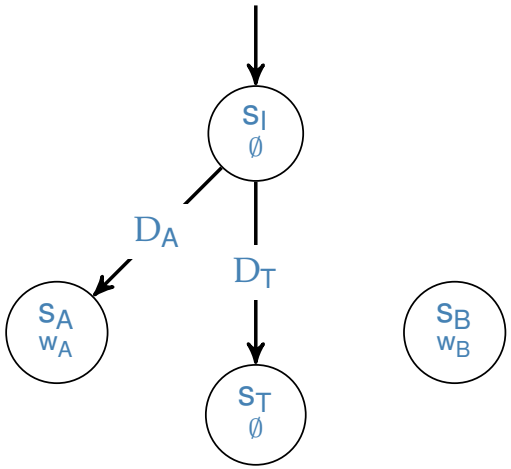
- $\mathrm{Ag} = \{\mathsf{A}: \textit{Alice}, \mathsf{B}: \textit{Bob}\}$
- $\mathrm{Ac} = \{\mathsf{P}: \textit{Paper}, \mathsf{R}: \textit{Rock}, \mathsf{S}: \textit{Scissor}\}$
- $\mathrm{St} = \{s_I, s_A, s_T, s_B\}$
- s_I: *initial state*
- $\mathrm{AP} = \{w_A: \textit{Alice wins}, w_B: \textit{Bob wins}\}$

Example: paper, rock, and scissor (2)



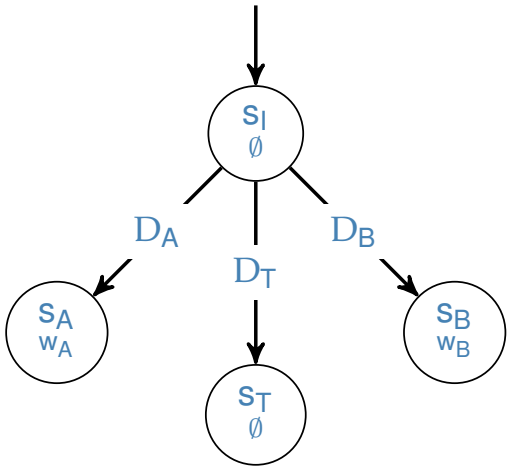
- $\mathrm{Ag} = \{\mathsf{A}: \textit{Alice}, \mathsf{B}: \textit{Bob}\}$
- $\mathrm{Ac} = \{\mathsf{P}: \textit{Paper}, \mathsf{R}: \textit{Rock}, \mathsf{S}: \textit{Scissor}\}$
- $\mathrm{St} = \{s_I, s_A, s_T, s_B\}$
- s_I: *initial state*
- $\mathrm{AP} = \{w_A: \textit{Alice wins}, w_B: \textit{Bob wins}\}$
- $D_T = \{(P, P), (R, R), (S, S)\}$

Example: paper, rock, and scissor (2)



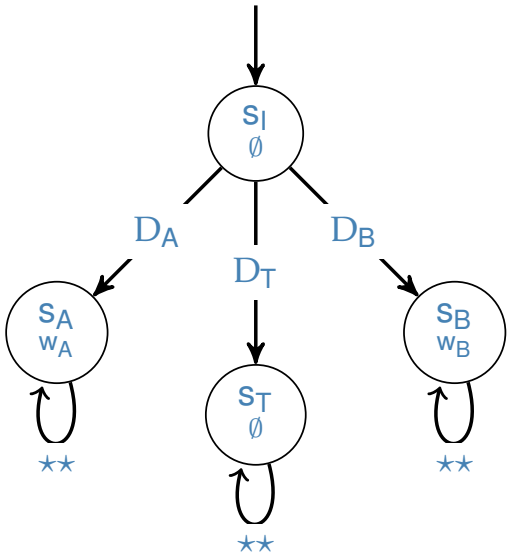
- $\mathrm{Ag} = \{\mathsf{A}: \textit{Alice}, \mathsf{B}: \textit{Bob}\}$
- $\mathrm{Ac} = \{\mathsf{P}: \textit{Paper}, \mathsf{R}: \textit{Rock}, \mathsf{S}: \textit{Scissor}\}$
- $\mathrm{St} = \{s_I, s_A, s_T, s_B\}$
- s_I: *initial state*
- $\mathrm{AP} = \{w_A: \textit{Alice wins}, w_B: \textit{Bob wins}\}$
- $D_T = \{(P, P), (R, R), (S, S)\}$
- $D_A = \{(P, R), (R, S), (S, P)\}$

Example: paper, rock, and scissor (2)



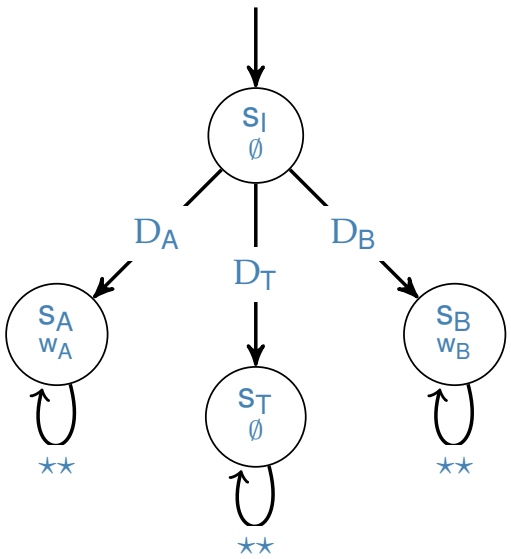
- $\mathrm{Ag} = \{\mathsf{A}: \textit{Alice}, \mathsf{B}: \textit{Bob}\}$
- $\mathrm{Ac} = \{\mathsf{P}: \textit{Paper}, \mathsf{R}: \textit{Rock}, \mathsf{S}: \textit{Scissor}\}$
- $\mathrm{St} = \{s_I, s_A, s_T, s_B\}$
- s_I: *initial state*
- $\mathrm{AP} = \{w_A: \textit{Alice wins}, w_B: \textit{Bob wins}\}$
- $D_T = \{(P,P), (R,R), (S,S)\}$
- $D_A = \{(P,R), (R,S), (S,P)\}$
- $D_B = \{(R,P), (S,R), (P,S)\}$

Example: paper, rock, and scissor (2)



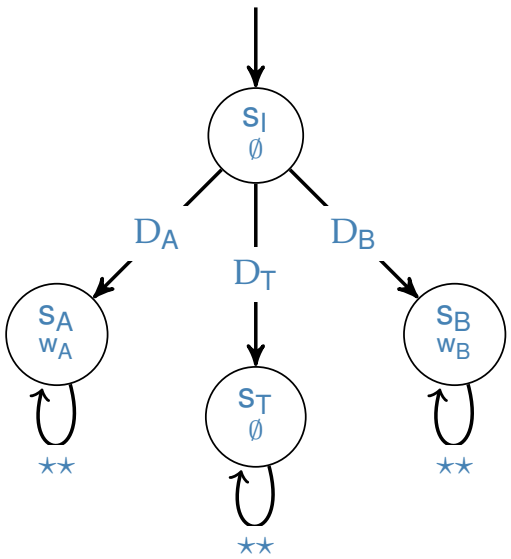
- $\mathrm{Ag} = \{\mathsf{A}: \textit{Alice}, \mathsf{B}: \textit{Bob}\}$
- $\mathrm{Ac} = \{\mathsf{P}: \textit{Paper}, \mathsf{R}: \textit{Rock}, \mathsf{S}: \textit{Scissor}\}$
- $\mathrm{St} = \{s_I, s_A, s_T, s_B\}$
- s_I: *initial state*
- $\mathrm{AP} = \{w_A: \textit{Alice wins}, w_B: \textit{Bob wins}\}$
- $D_T = \{(P,P), (R,R), (S,S)\}$
- $D_A = \{(P,R), (R,S), (S,P)\}$
- $D_B = \{(R,P), (S,R), (P,S)\}$

Example: paper, rock, and scissor (2)



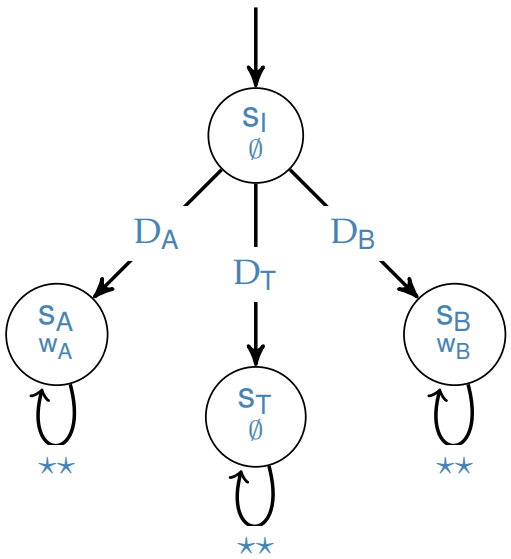
- $\mathrm{Ag} = \{\mathsf{A}: \textit{Alice}, \mathsf{B}: \textit{Bob}\}$
- $\mathrm{Ac} = \{\mathsf{P}: \textit{Paper}, \mathsf{R}: \textit{Rock}, \mathsf{S}: \textit{Scissor}\}$
- $\mathrm{St} = \{s_I, s_A, s_T, s_B\}$
- s_I: *initial state*
- $\mathrm{AP} = \{w_A: \textit{Alice wins}, w_B: \textit{Bob wins}\}$
- $D_T = \{(P,P), (R,R), (S,S)\}$
- $D_A = \{(P,R), (R,S), (S,P)\}$
- $D_B = \{(R,P), (S,R), (P,S)\}$
- A strategy for A is $\sigma_A(h) = \mathsf{P}$, $\forall h \in S^+$

Example: paper, rock, and scissor (2)



- $\mathrm{Ag} = \{\mathsf{A}: \textit{Alice}, \mathsf{B}: \textit{Bob}\}$
- $\mathrm{Ac} = \{\mathsf{P}: \textit{Paper}, \mathsf{R}: \textit{Rock}, \mathsf{S}: \textit{Scissor}\}$
- $\mathrm{St} = \{s_I, s_A, s_T, s_B\}$
- s_I: *initial state*
- $\mathrm{AP} = \{w_A: \textit{Alice wins}, w_B: \textit{Bob wins}\}$
- $D_T = \{(\mathsf{P},\mathsf{P}), (\mathsf{R},\mathsf{R}), (\mathsf{S},\mathsf{S})\}$
- $D_A = \{(\mathsf{P},\mathsf{R}), (\mathsf{R},\mathsf{S}), (\mathsf{S},\mathsf{P})\}$
- $D_B = \{(\mathsf{R},\mathsf{P}), (\mathsf{S},\mathsf{R}), (\mathsf{P},\mathsf{S})\}$
- A strategy for A is $\sigma_A(h) = \mathsf{P}$, $\forall h \in S^+$
- A strategy for B is $\sigma_B(h) = \mathsf{R}$, $\forall h \in S^+$

Example: paper, rock, and scissor (2)



- $\mathrm{Ag} = \{\mathsf{A}: \textit{Alice}, \mathsf{B}: \textit{Bob}\}$
- $\mathrm{Ac} = \{\mathsf{P}: \textit{Paper}, \mathsf{R}: \textit{Rock}, \mathsf{S}: \textit{Scissor}\}$
- $\mathrm{St} = \{s_I, s_A, s_T, s_B\}$
- s_I: *initial state*
- $\mathrm{AP} = \{w_A: \textit{Alice wins}, w_B: \textit{Bob wins}\}$
- $D_T = \{(\mathsf{P},\mathsf{P}), (\mathsf{R},\mathsf{R}), (\mathsf{S},\mathsf{S})\}$
- $D_A = \{(\mathsf{P},\mathsf{R}), (\mathsf{R},\mathsf{S}), (\mathsf{S},\mathsf{P})\}$
- $D_B = \{(\mathsf{R},\mathsf{P}), (\mathsf{S},\mathsf{R}), (\mathsf{P},\mathsf{S})\}$
- A strategy for A is $\sigma_A(h) = \mathsf{P}$, $\forall h \in S^+$
- A strategy for B is $\sigma_B(h) = \mathsf{R}$, $\forall h \in S^+$
- σ_A and σ_B induce the path $s_I \cdot (s_A)^\omega$

Multi-agent Systems (4)

Visibility of the agents

Depending on the visibility of the agents, we distinguish between:

Multi-agent Systems (4)

Visibility of the agents

Depending on the visibility of the agents, we distinguish between:

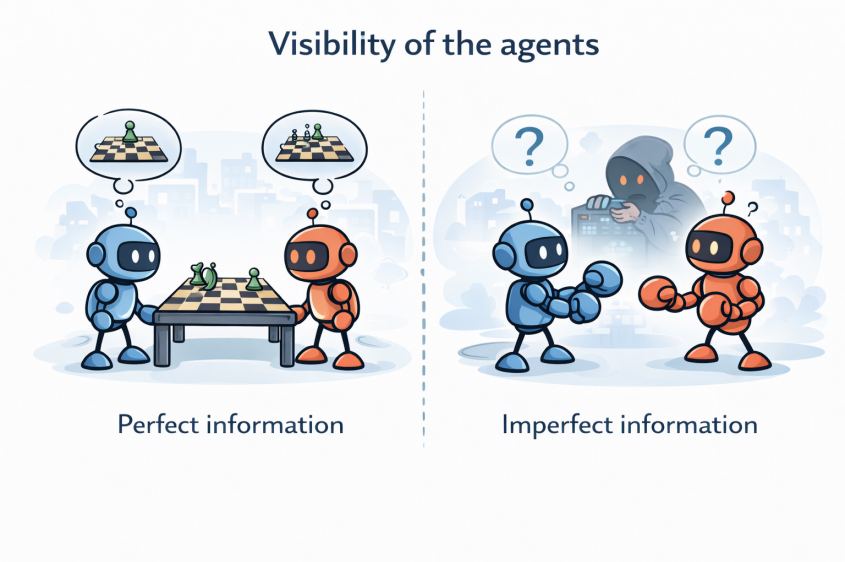
- *Perfect information* MAS
  Agents have full knowledge of the MAS evolution at all times.

Multi-agent Systems (4)

Visibility of the agents

Depending on the visibility of the agents, we distinguish between:

- *Perfect information* MAS
  Agents have full knowledge of the MAS evolution at all times.
- *Imperfect information* MAS
  Agents often have to decide without all relevant information.



Specifications

Alternating-Time Temporal Logic (ATL*) [AHK02]

- ATL* generalizes CTL* by replacing the path quantification with strategy quantification over coalition of agents.
- It uses the strategy modality $\langle\langle A \rangle\rangle$ and $[\![A]\!]$, with A set of agents.
- The strategies are implicit.

SPECIFICATIONS

Alternating-Time Temporal Logic (ATL*) [AHK02]

- ATL* generalizes CTL* by replacing the path quantification with strategy quantification over coalition of agents.
- It uses the strategy modality $\langle\langle A \rangle\rangle$ and $[\![A]\!]$, with A set of agents.
- The strategies are implicit.

Example

$\langle\langle \{\alpha, \beta\} \rangle\rangle \mathsf{G} \neg \mathit{fail}$: agents α and β have a collective strategy that prevents system failure regardless of other agents' strategies.

Verification: Model checking Complexities

Verification: Model checking Complexities

ATL

Strategies \ Information	**perfect**	**imperfect**
memoryless	*PTIME*-complete [AHK02]	P^{NP}-complete [Sch03]
memoryfull	*PTIME*-complete [AHK02]	undecidable [DT11]

Verification: Model checking Complexities

ATL

Strategies \ Information	**perfect**	**imperfect**
memoryless	*PTIME*-complete [AHK02]	P^{NP}-complete [Sch03]
memoryfull	*PTIME*-complete [AHK02]	undecidable [DT11]

ATL*

Strat \ Info	**perfect**	**imperfect**
memoryless	*PSPACE*-complete [Sch03]	*PSPACE*-complete [Sch03]
memoryfull	*2EXPTIME*-complete [AHK02]	undecidable [DT11]

SPECIFICATIONS

Strategy Logic (SL) [MMPV14]

- SL treats the strategies as first-order objects.
- SL extends LTL with:
  - *strategy quantifiers* $\langle\langle x \rangle\rangle$ and $[\![x]\!]$ read respectively as *"there exists a strategy* x*"* and *"for all strategies* x*"*;
  - *agent binding* (a, x) that links an agent a to a strategy associated with a variable x.
- It subsumes other logics for strategic reasoning such as ATL*.
- It allows to represent Nash equilibrium and similar concepts.

Specifications

Strategy Logic (SL) [MMPV14]

- SL treats the strategies as first-order objects.
- SL extends LTL with:
  - *strategy quantifiers* $\langle\langle x \rangle\rangle$ and $[\![x]\!]$ read respectively as *"there exists a strategy* x*"* and *"for all strategies* x*"*;
  - *agent binding* (a, x) that links an agent a to a strategy associated with a variable x.
- It subsumes other logics for strategic reasoning such as ATL*.
- It allows to represent Nash equilibrium and similar concepts.

Example

$\langle\langle \mathsf{x} \rangle\rangle \langle\langle \mathsf{y} \rangle\rangle [\![\mathsf{z}]\!] (\alpha, \mathsf{x})(\beta, \mathsf{y})(\gamma, \mathsf{z})(\mathsf{G} \neg \mathit{fail})$: α and β employ strategies x and y, respectively, ensuring non-failure regardless of γ's strategy z.

Verification: Model Checking complexities

Verification: Model Checking complexities

Strategy Logic

Strat \ Info	**perfect**	**imperfect**
memoryless	*PSPACE*-complete [MMMP21]	*PSPACE*-complete [MMMP21]
memoryfull	non-elementary [MMPV14]	undecidable [DT11]

Verification: Model Checking complexities

Strategy Logic

Strat \ Info	**perfect**	**imperfect**
memoryless	*PSPACE*-complete [MMMP21]	*PSPACE*-complete [MMMP21]
memoryfull	non-elementary [MMPV14]	undecidable [DT11]

Enhance model checking complexity by reducing expressiveness

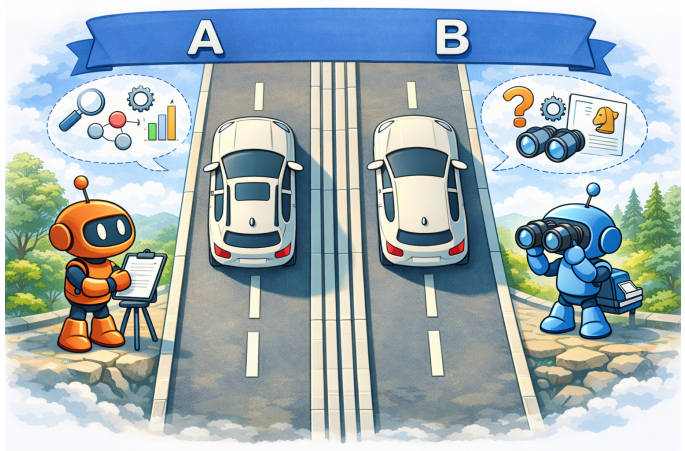
- SL[1G] model checking with perfect information and memoryfull strategies is $2EXPTIME$-complete [MMPV14].
- SL[SG] model checking with perfect information and memoryfull strategies is $PTIME$-complete [BJK$^+$19].

Why do we use logics for strategic reasoning?

WHY DO WE USE LOGICS FOR STRATEGIC REASONING?

Example

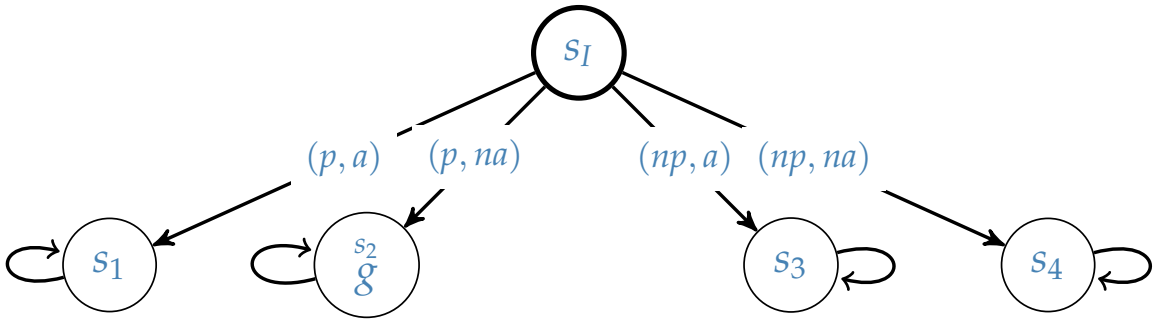
- 2 vehicles: A and B.
- B is in front of A.
- Goal: A wants to pass B (atom g).
- A Actions: pass (p), not pass (np).
- B Actions: accelerate (a), not accelerate (na).



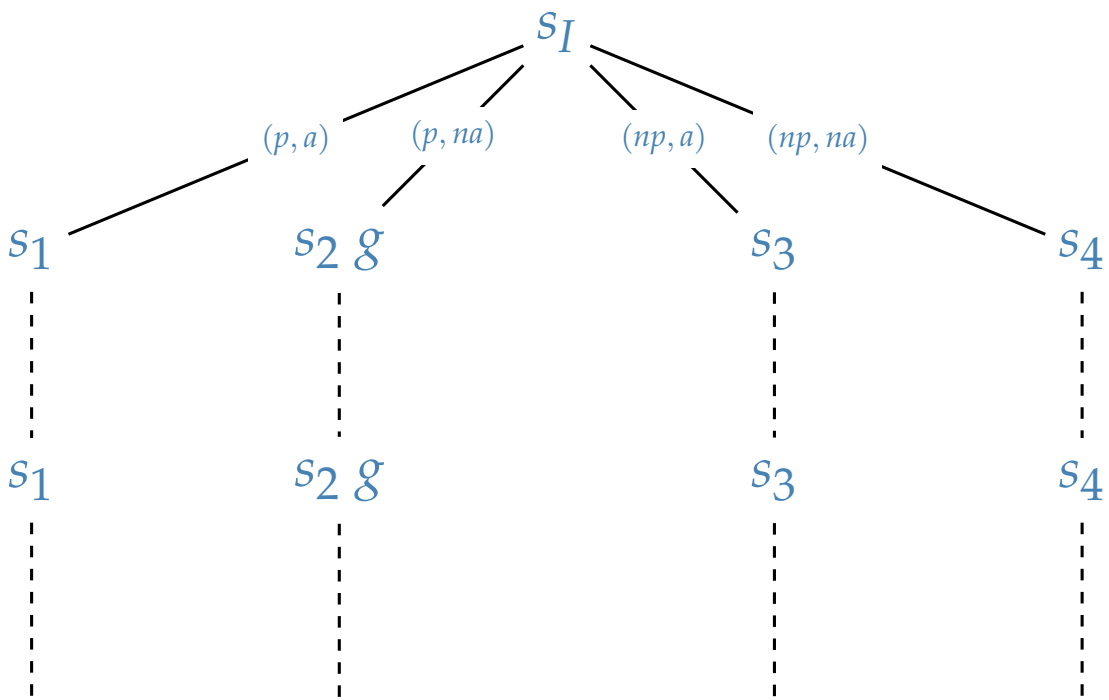
WHY DO WE USE LOGICS FOR STRATEGIC REASONING?

Example

- 2 vehicles: A and B.
- B is in front of A.
- Goal: A wants to pass B (atom g).
- A Actions: pass (p), not pass (np).
- B Actions: accelerate (a), not accelerate (na).

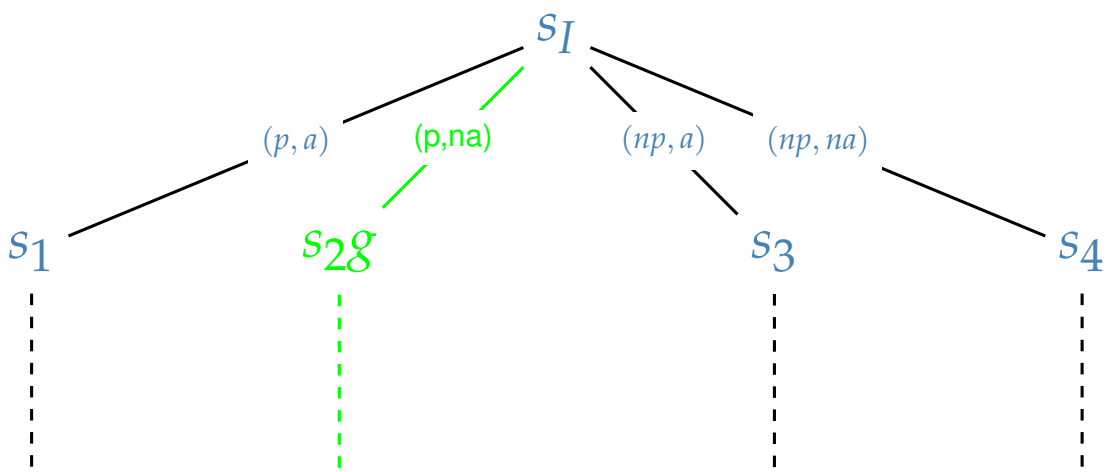


Unwinding of the model



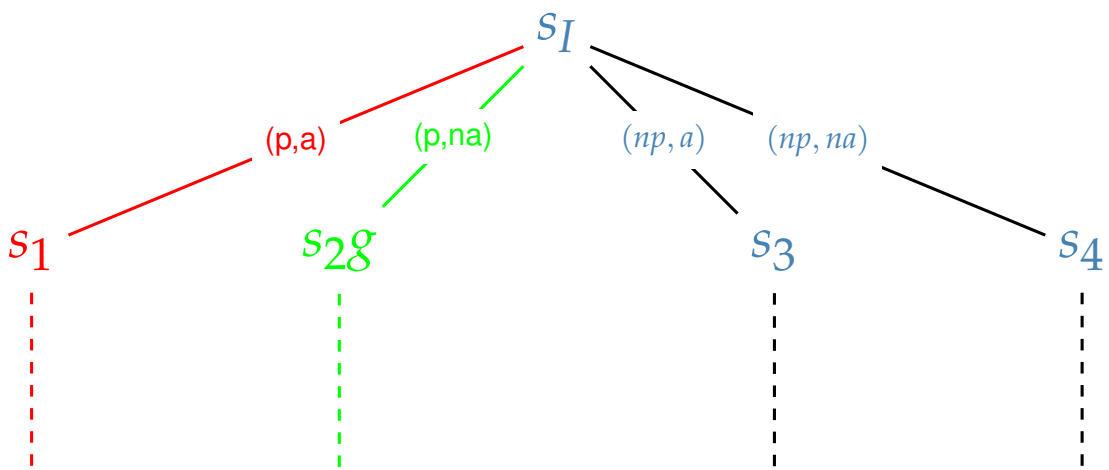
What can we do with CTL?

- The formula $EF\ g$ is verified in the path: $s_I, s_2, \ldots$



What can't we do with CTL?

- Is there a strategy for A to achieve g?
  - if A select the action p then we need to verify the formula in the paths: $s_I, s_1, \ldots$ (false) and $s_I, s_2, \ldots$ (true);



Comparison between CTL and ATL

Summary

- The formula in CTL $EF\, g$ is true.
- The formula in ATL $\langle\langle A \rangle\rangle F\, g$ is false.
- In the model there is a path where A can pass B but there is no strategy for A to pass B!
- With CTL we cannot verify properties on subsets of paths.

STRATEGIC VERIFICATION IN VITAMIN

State of the Art: Tool-Centred Verification

- Formal verification of Multi-Agent Systems (MAS) is supported by tools such as **MCMAS** and **STV**.
- These tools demonstrate that **strategic and epistemic verification is feasible in practice**.

State of the Art: Tool-Centred Verification

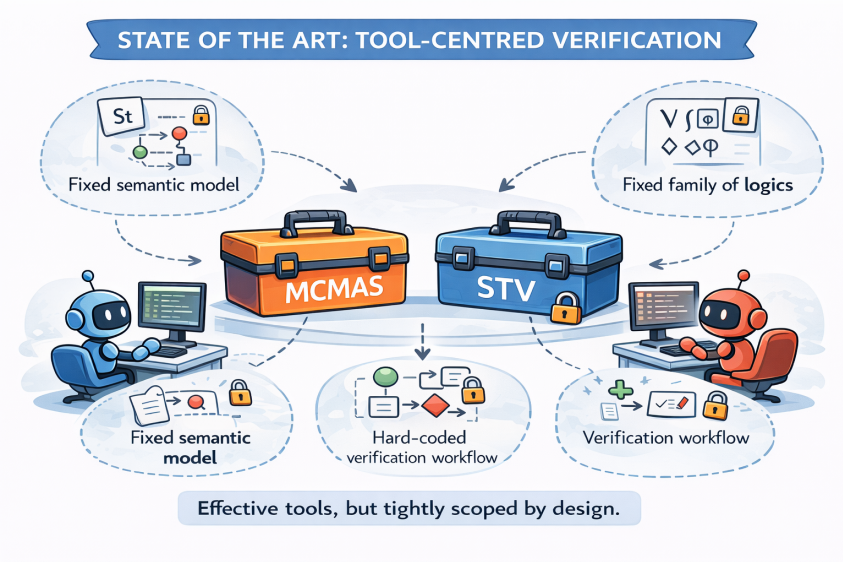
- Formal verification of Multi-Agent Systems (MAS) is supported by tools such as **MCMAS** and **STV**.
- These tools demonstrate that **strategic and epistemic verification is feasible in practice**.
- However, they are typically designed around:
  - a **fixed semantic model**

State of the Art: Tool-Centred Verification

- Formal verification of Multi-Agent Systems (MAS) is supported by tools such as **MCMAS** and **STV**.
- These tools demonstrate that **strategic and epistemic verification is feasible in practice**.
- However, they are typically designed around:
  - a **fixed semantic model**
  - a **fixed family of logics**

State of the Art: Tool-Centred Verification

- Formal verification of Multi-Agent Systems (MAS) is supported by tools such as **MCMAS** and **STV**.
- These tools demonstrate that **strategic and epistemic verification is feasible in practice**.
- However, they are typically designed around:
  - a **fixed semantic model**
  - a **fixed family of logics**
  - a **hard-coded verification workflow**



The Two Representative Tools

MCMAS

- Based on **Local Models**
- Strong support for epistemic reasoning
- Symbolic model checking
- Widely used in the community

The Two Representative Tools

MCMAS

- Based on **Local Models**
- Strong support for epistemic reasoning
- Symbolic model checking
- Widely used in the community

Typical focus

Knowledge, temporal evolution, and strategic abilities.

The Two Representative Tools

MCMAS

- Based on **Local Models**
- Strong support for epistemic reasoning
- Symbolic model checking
- Widely used in the community

STV

- Based on **Local Models**
- Focus on ATL-style strategic reasoning
- GUI-oriented interaction
- Explicit-state verification

Typical focus

Knowledge, temporal evolution, and strategic abilities.

The Two Representative Tools

MCMAS

- Based on **Local Models**
- Strong support for epistemic reasoning
- Symbolic model checking
- Widely used in the community

Typical focus

Knowledge, temporal evolution, and strategic abilities.

STV

- Based on **Local Models**
- Focus on ATL-style strategic reasoning
- GUI-oriented interaction
- Explicit-state verification

Typical focus

Strategic abilities in game-like and decision-making scenarios.

What These Tools Do Well

- Provide **sound and efficient** verification within their scope
- Support widely studied logics (CTL, ATL, epistemic variants)
- Enable practical verification of non-trivial MAS models

What These Tools Do Well

- Provide **sound and efficient** verification within their scope
- Support widely studied logics (CTL, ATL, epistemic variants)
- Enable practical verification of non-trivial MAS models

Key observation

Existing tools are *highly effective* when used as intended.

Evidence: Feature Comparison

Feature	MCMAS	STV	VITAMIN
GUI	(✓)	✓	✓
Internal documentation	(✓)	×	✓
External documentation	×	×	✓
Modular architecture	×	×	✓
Cross-platform	×	✓	✓
API / integration	×	×	✓
Support for non-experts	×	×	✓

Evidence: Feature Comparison

Feature	MCMAS	STV	VITAMIN
GUI	(✓)	✓	✓
Internal documentation	(✓)	×	✓
External documentation	×	×	✓
Modular architecture	×	×	✓
Cross-platform	×	✓	✓
API / integration	×	×	✓
Support for non-experts	×	×	✓

- **Takeaway:** current tools are *not designed* for extensibility and reuse across verification scenarios.

Limitations Emerging from Current Designs

- Strong coupling between:
  - model formalism
  - logic
  - verification engine

Limitations Emerging from Current Designs

- Strong coupling between:
  - model formalism
  - logic
  - verification engine
- Extending a tool with:
  - new models
  - new logics
  - alternative semantics

  typically requires **deep changes to the core**

Limitations Emerging from Current Designs

- Strong coupling between:
  - model formalism
  - logic
  - verification engine
- Extending a tool with:
  - new models
  - new logics
  - alternative semantics

  typically requires **deep changes to the core**
- Limited support for:
  - compositional verification
  - systematic experimentation
  - non-expert users

The Emerging Gap

Observation

Modern MAS research requires rapidly exploring *new combinations* of models, logics, and verification techniques.

The Emerging Gap

Observation

Modern MAS research requires rapidly exploring *new combinations* of models, logics, and verification techniques.

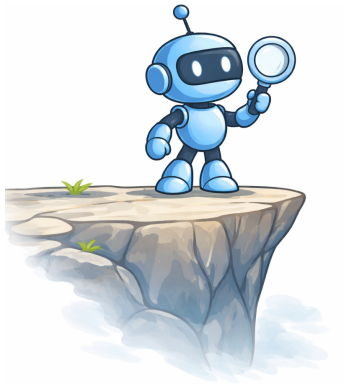
- Strategic reasoning is evolving faster than individual tools
- Rebuilding a verifier for each new logic or semantics is unrealistic
- Verification must scale not only in performance, but in **design flexibility**

The Emerging Gap

Observation

Modern MAS research requires rapidly exploring *new combinations* of models, logics, and verification techniques.

- Strategic reasoning is evolving faster than individual tools
- Rebuilding a verifier for each new logic or semantics is unrealistic
- Verification must scale not only in performance, but in **design flexibility**



This calls for a framework, not another monolithic tool.

From Limitations to Design Goals

Observed problems

- **Hard-coded** verification pipelines
- **Tight** model–logic **coupling**
- **High entry barrier** for non-experts
- Limited **extensibility**

From Limitations to Design Goals

Observed problems

- **Hard-coded** verification pipelines
- **Tight** model–logic **coupling**
- **High entry barrier** for non-experts
- Limited **extensibility**

VITAMIN goals

- **Compositional** architecture
- **Independent** model and logic layers
- **Pluggable** verification engines
- Support for **non-expert** users

From Limitations to Design Goals

Observed problems

- **Hard-coded** verification pipelines
- **Tight** model–logic **coupling**
- **High entry barrier** for non-experts
- Limited **extensibility**

VITAMIN goals

- **Compositional** architecture
- **Independent** model and logic layers
- **Pluggable** verification engines
- Support for **non-expert** users

VITAMIN is designed to decouple verification concerns explicitly.

VITAMIN Architecture: Core Idea

Principle

Separate concerns explicitly: models, logics, and verification techniques should evolve independently.

VITAMIN Architecture: Core Idea

Principle

Separate concerns explicitly: models, logics, and verification techniques should evolve independently.

Main components

- Model layer
- Logic layer
- Model Checker Interface
- GUI

VITAMIN Architecture: Core Idea

Principle

Separate concerns explicitly: models, logics, and verification techniques should evolve independently.

Main components

- Model layer
- Logic layer
- Model Checker Interface
- GUI

User roles

- End-users (expert / non-expert)
- High-level developers (models, logics)
- Low-level developers (optimisation)

Supported Logics: Evidence of Generality

Logic	VITAMIN	MCMAS	STV
CTL	✓	✓	×
ATL	✓	✓	✓
SL	✓$^{\text{lim}}$	✓$^{\text{lim}}$	✓$^{\text{lim}}$
CTLK / ATLK	×	✓	✓$^{\text{lim}}$
ATLF	✓	×	×
RBATL	✓	✓	×
RABATL	✓	×	×
NatATL	✓	×	×
OL	✓	×	×
OATL	✓	×	×
NatOL	✓	×	×
capATL	✓	×	×
NatSL	✓	×	×
WATL	✓	×	×
TOL	✓	×	×
POL	✓	×	×

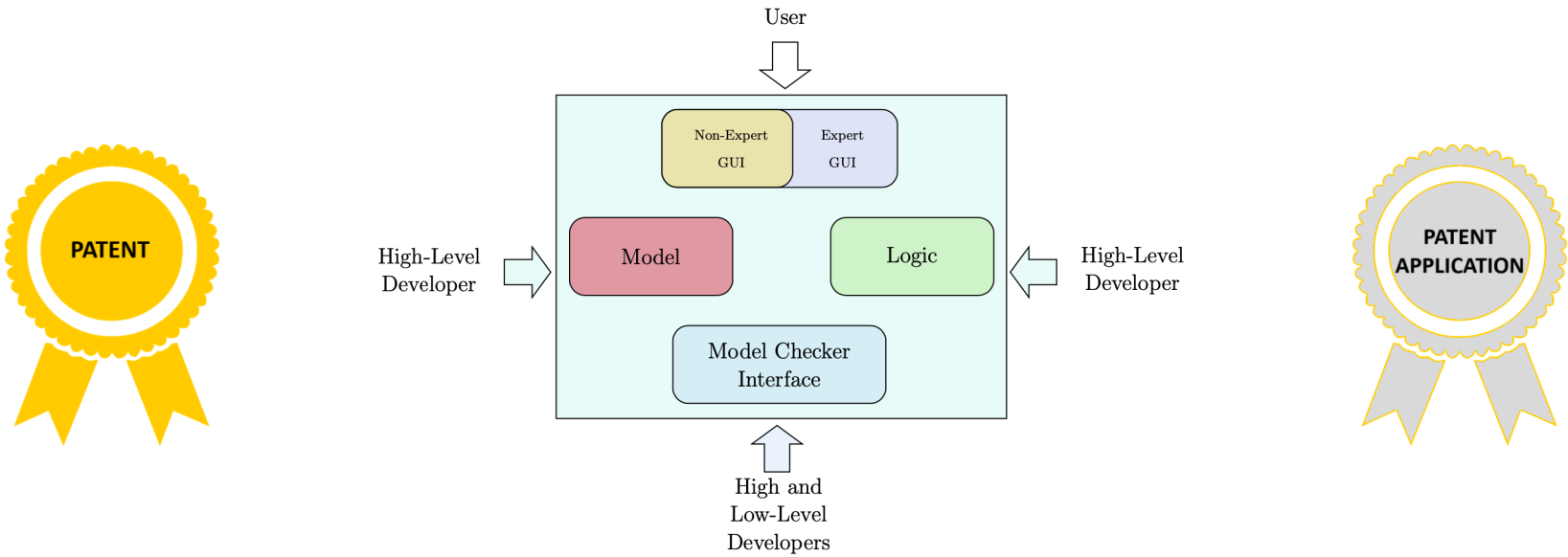
Legend: ✓ = native; ✓$^{\text{lim}}$ = partial/limited; × = not supported.

VITAMIN in One Sentence

VITAMIN is not just a model checker, but a compositional **framework** for **designing**, **experimenting** with, and **deploying** strategic verification solutions for Multi-Agent Systems.

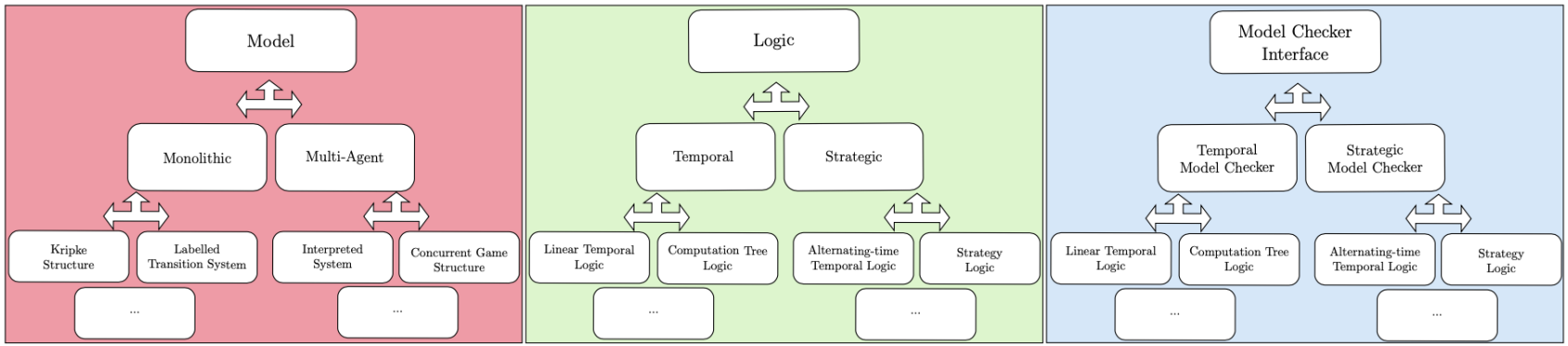
VITAMIN

- VITAMIN (VerIficaTion of A MultI-ageNt system) since **2022**
- Highly compositional and supports various logic and models
- User-friendly experience for both developers and end-users

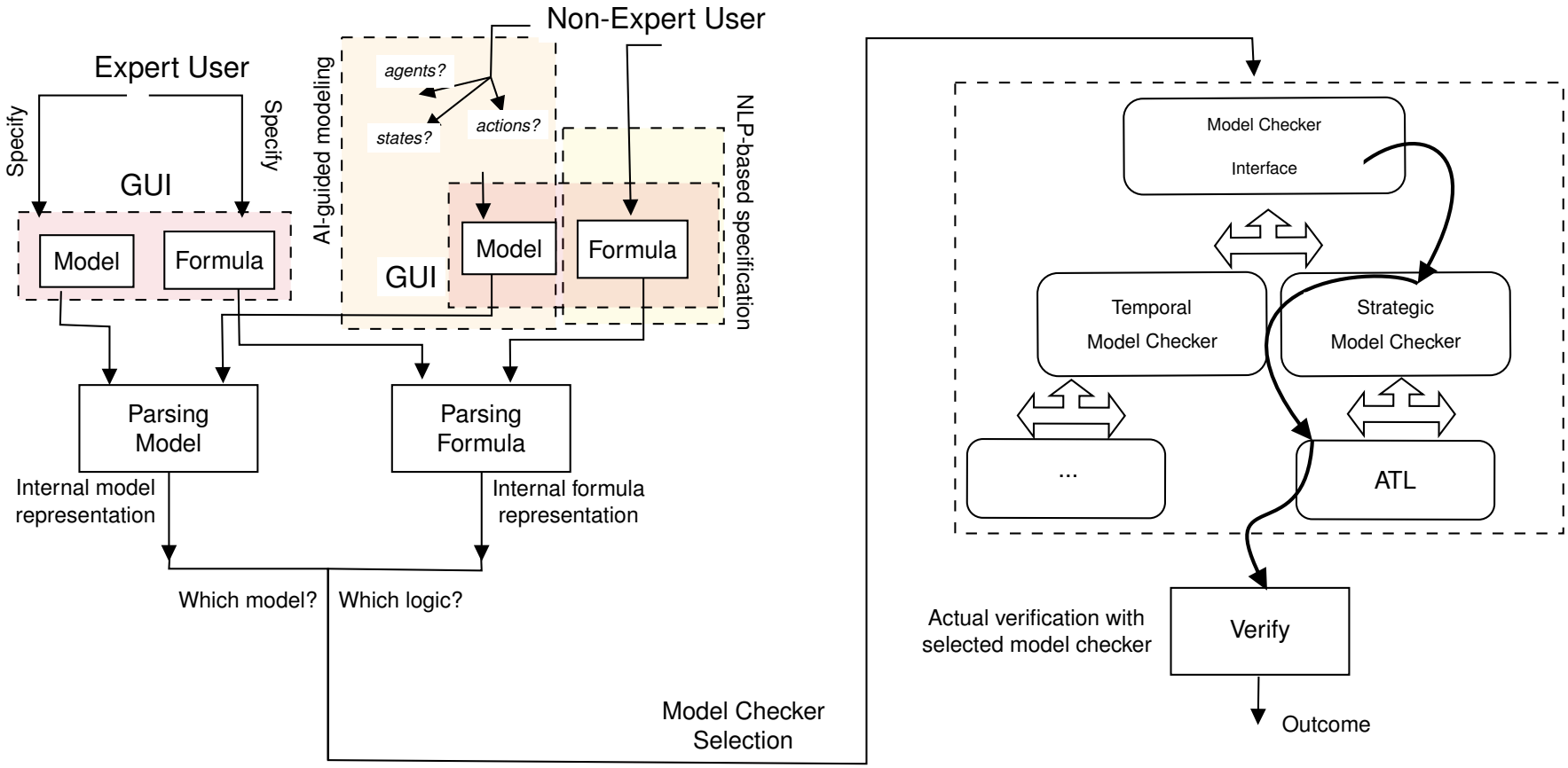


Advantages of VITAMIN

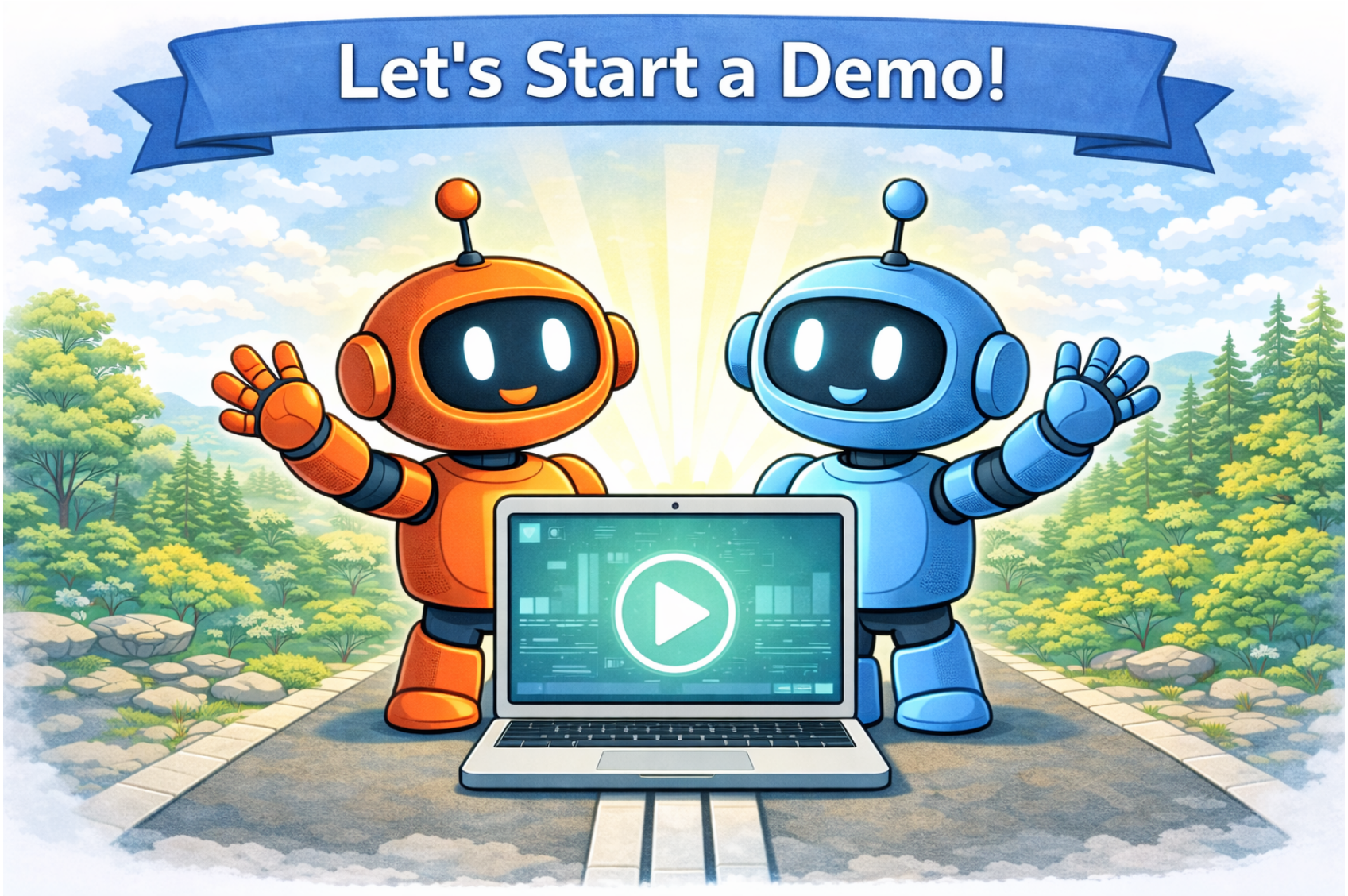
- Generalises MAS verification without being tied to specific logic or model formalisms
- Enhanced user experience guiding the entire verification process
- Compositional nature allows for straightforward extension



Verification Flow



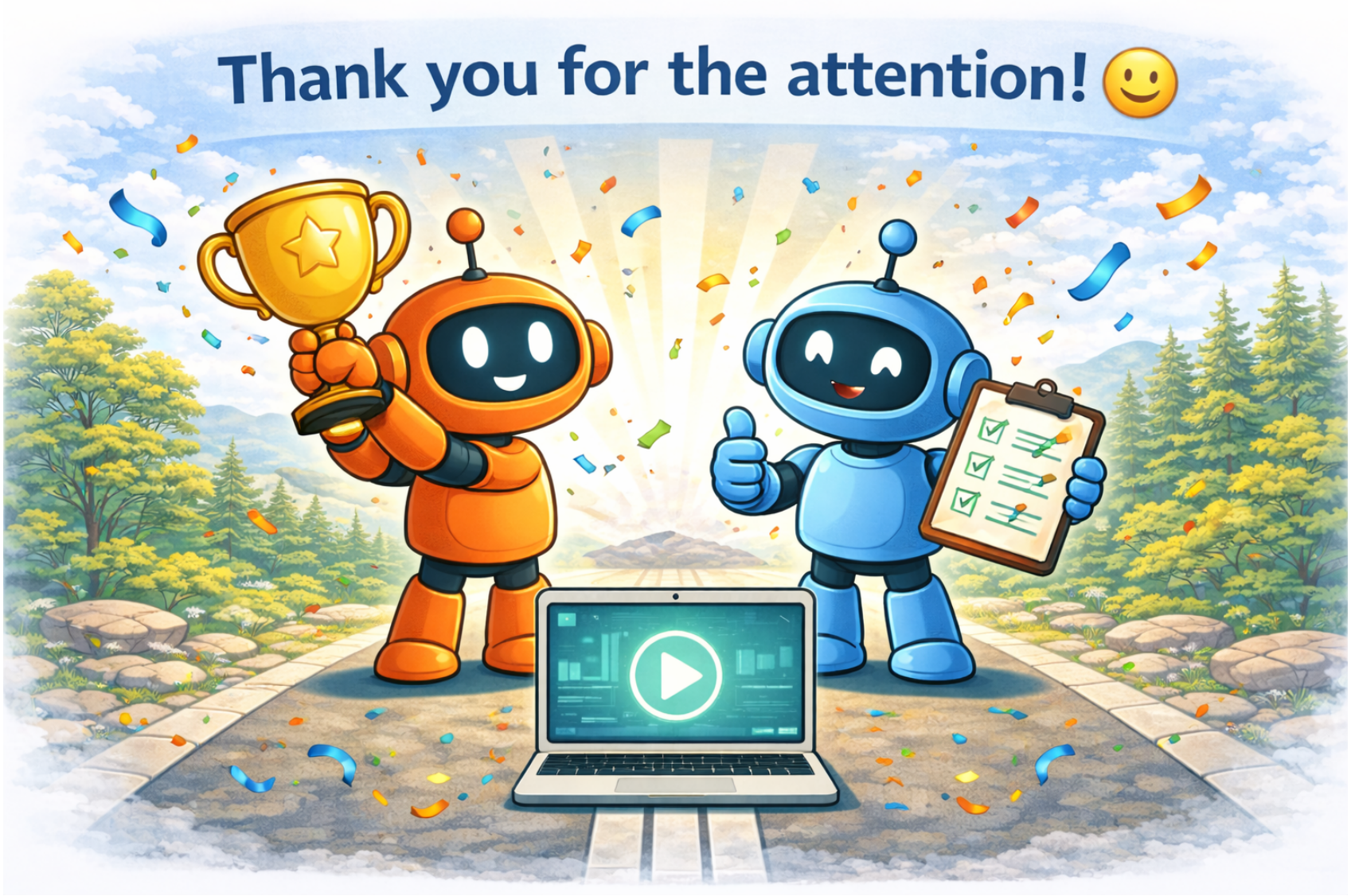
DEMO



Future Directions

Future Directions

- **VITAMIN**, key goals we aim to achieve:
  - Use AI to assist users in generating formal models and formulas.
  - Low-level development with symbolic & abstraction methods.
  - Use AI to optimize the verification process, making it scalable.
  - Ensure the tool is robust, easily extensible, and deployable.


Thank you for the attention!

References I

Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman.

Alternating-time temporal logic.
J. ACM, 49(5):672–713, 2002.

Francesco Belardinelli, Angelo Ferrando, and Vadim Malvone.

An abstraction-refinement framework for verifying strategic properties in multi-agent systems with imperfect information.
Artif. Intell., 316:103847, 2023.

Francesco Belardinelli, Wojciech Jamroga, Damian Kurpiewski, Vadim Malvone, and Aniello Murano.

Strategy logic with simple goals: Tractable reasoning about strategies.
In IJCAI 2019, pages 88–94, 2019.

Francesco Belardinelli, Alessio Lomuscio, Vadim Malvone, and Emily Yu.

Approximating perfect recall when model checking strategic abilities: Theory and applications.
J. Artif. Intell. Res., 73:897–932, 2022.

E.M. Clarke, O. Grumberg, D. Kroening, D. Peled, and H. Veith.

Model Checking, second edition.
Cyber Physical Systems Series. MIT Press, 2018.

Davide Catta, Jean Leneutre, and Vadim Malvone.

Obstruction logic: A strategic temporal logic to reason about dynamic game models.
In ECAI 2023, pages 365–372, 2023.

Davide Catta, Jean Leneutre, Vadim Malvone, and Aniello Murano.

Obstruction alternating-time temporal logic: A strategic logic to reason about dynamic models.
In AAMAS 2024, pages 271–280, 2024.

Catalin Dima and Ferucio Laurentiu Tiplea.

Model-checking ATL under imperfect information and perfect recall semantics is undecidable.
CoRR, abs/1102.4225, 2011.

References II

Angelo Ferrando and Vadim Malvone.

Towards the verification of strategic properties in multi-agent systems with imperfect information.
In AAMAS 2023, pages 793–801, 2023.

Wojciech Jamroga, Vadim Malvone, and Aniello Murano.

Reasoning about natural strategic ability.
In AAMAS 2017, pages 714–722, 2017.

Wojciech Jamroga, Vadim Malvone, and Aniello Murano.

Natural strategic ability.
Artif. Intell., 277, 2019.

Wojciech Jamroga, Vadim Malvone, and Aniello Murano.

Natural strategic ability under imperfect information.
In AAMAS 2019, pages 962–970, 2019.

Bastien Maubert, Munyque Mittelmann, Aniello Murano, and Laurent Perrussel.

Strategic reasoning in automated mechanism design.
In KR 2021, pages 487–496, 2021.

Fabio Mogavero, Aniello Murano, Giuseppe Perelli, and Moshe Y. Vardi.

Reasoning about strategies: On the model-checking problem.
ACM Trans. Comput. Log., 15(4):34:1–34:47, 2014.

Pierre-Yves Schobbens.

Alternating-time logic with imperfect recall.
In LCMAS 2003, pages 82–93, 2003.