Automated Verification of Multi-Agent Systems From Theory to Practice, Made Easy

Wojtek Jamroga, Damian Kurpiewski,
and Wojtek Penczek
Polish Academy of Sciences, Warsaw, Poland

Tutorial at ECAI '24 / 20 Nov 2024 @ Santiago de Compostela, Spain

Thank You's

When preparing the course, we used some materials courtesy of:

- Thomas Agotnes, University of Bergen
- Nils Bulling, Clausthal University of Technology
- Juergen Dix, Clausthal University of Technology
- Valentin Goranko, Technical University of Denmark

All the mistakes are, of course, ours.

Basic Reading

- Yoav Shoham and Kevin Leyton-Brown (2009), Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations. Cambridge University Press.
- Edmund M. Clarke, Orna Grumberg and Doron A. Peled (1999), Model Checking. MIT Press.
- Christel Baier, Joost-Pieter Katoen (2008), Principles of model checking. MIT Press.
- Wojciech Jamroga (2015), Logical Methods for Specification and Verification of Multi-Agent Systems. Available for free at: `http://krak.ipipan.waw.pl/~wjamroga/papers/jamroga15specifmas.pdf`

Research supported by: Fonds National de la Recherche Luxembourg, NCBR National Centre for Research and Development, chist-era, National Science Centre Poland

Outline

Part 1: Introduction to Model Checking for MAS

Part 1: Introduction to Model Checking for MAS

1.1 Motivation

Motivation

Formal specification and **verification** helps to:

- Clarify and disambiguate the requirements
- Uncover the implicit assumptions about the system and participants

Motivation

Formal specification and **verification** helps to:

- Clarify and disambiguate the requirements
- Uncover the implicit assumptions about the system and participants
- Sometimes, we can even prove that the requirements hold

Motivation

Formal specification and **verification** helps to:

- Clarify and disambiguate the requirements
- Uncover the implicit assumptions about the system and participants
- Sometimes, we can even prove that the requirements hold wrt the assumptions

Motivation

Formal specification and **verification** helps to:

- Clarify and disambiguate the requirements
- Uncover the implicit assumptions about the system and participants
- Sometimes, we can even prove that the requirements hold wrt the assumptions
- ...or disprove and generate a counterexample

Formal Verification of Strategic Ability

- Many important properties are based on **strategic ability**
- Functionality $\approx$ ability of authorized users to complete some tasks
- Security $\approx$ inability of unauthorized users to complete certain tasks

Formal Verification of Strategic Ability

- Many important properties are based on **strategic ability**
- Functionality $\approx$ ability of authorized users to complete some tasks
- Security $\approx$ inability of unauthorized users to complete certain tasks
- One can try to formalize such properties in modal logics of strategic ability, such as ATL or Strategy Logic

Formal Verification of Strategic Ability

- Many important properties are based on **strategic ability**
- Functionality $\approx$ ability of authorized users to complete some tasks
- Security $\approx$ inability of unauthorized users to complete certain tasks
- One can try to formalize such properties in modal logics of strategic ability, such as ATL or Strategy Logic
- ...and verify them by model checking

Motivating Example: Security of Voting

Voting Scenario

Citizens of Pneumonia are voting in the presidential election.

There are n voters, each of them supposed to enter a voting booth at a polling station, select one of the candidates from the ballot, register their vote, and exit the polling station.

There is also a coercer who can attempt to bribe or blackmail the voters into voting for a particular candidate. The coercer can interact with the voters, e.g., making demands or giving instructions on how to vote. He is also capable of intercepting unencrypted messages.

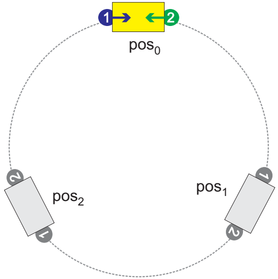
Part 1: Introduction to Model Checking for MAS

1.2 Modeling MAS

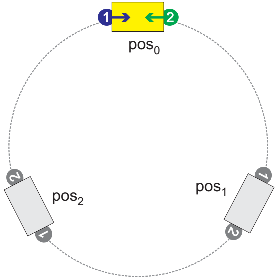
Modeling Multi-Agent Systems

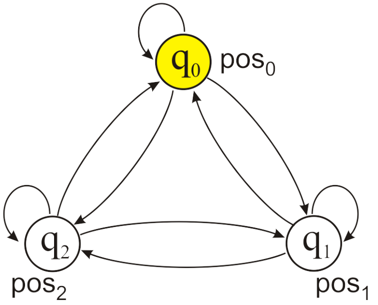
- How to model a distributed system? $\rightsquigarrow$ **transition graph**
- Nodes represent states of the system (or situations)
- Arrows correspond to changes of state

Example: Robots and Carriage

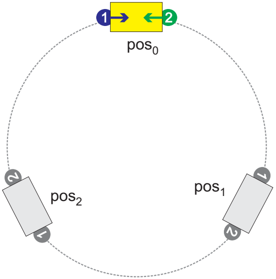


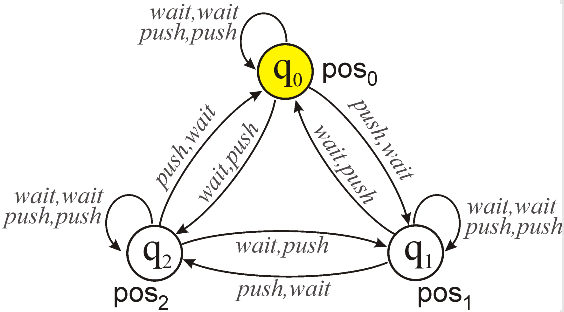
Example: Robots and Carriage



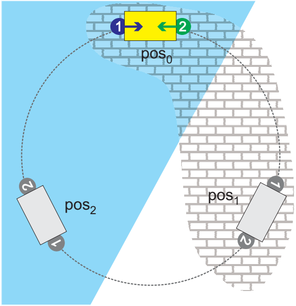


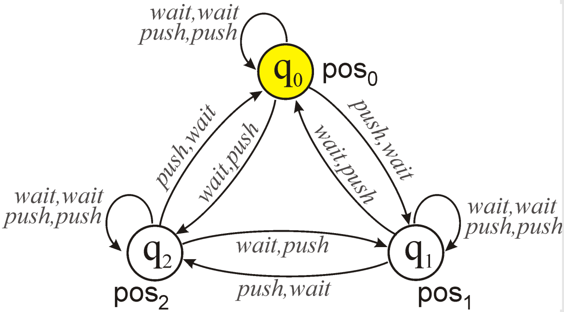
Example: Robots and Carriage



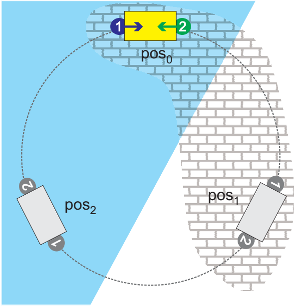


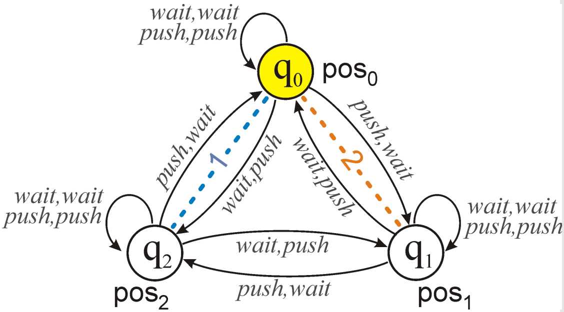
Example: Robots and Carriage





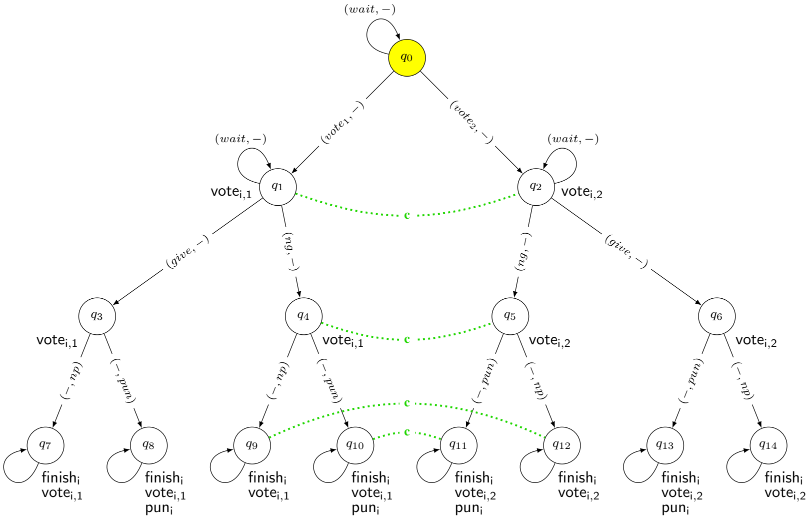
Example: Robots and Carriage



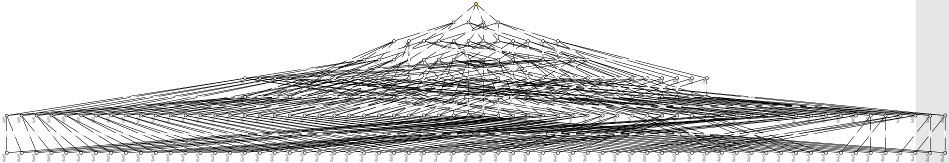


Example: Voting and Coercion

Example: Voting and Coercion



Example: Voting and Coercion



Part 1: Introduction to Model Checking for MAS

1.3 Temporal Properties

Motivating Example: Voting

Properties to express

- The voting system will not reveal how a particular voter voted
- The voter will eventually cast a vote
- The voter can vote, and can refrain from voting

Logical Specification of Properties

Specification of Temporal Properties

Temporal logic: mathematical logic with additional operators to describe how the system will (or may) evolve:

$\bigcirc\varphi$	φ is true in the next moment in time
$\Box\varphi$	φ is true in all future moments
$\Diamond\varphi$	φ is true in some future moment
$\varphi\,\mathrm{U}\,\psi$	φ is true until the moment when ψ becomes true

Specification Templates

Temporal logic has achieved a significant role in the formal specification and verification of concurrent and distributed systems.

Much of the popularity was achieved because some useful concepts can be formally, and concisely, specified using temporal logics, e.g.:

- safety properties
- reachability properties
- fairness properties

Specification Templates: Safety Properties

Safety / maintenance:

"something bad will never happen"
"something good will always hold"

Specification Templates: Safety Properties

Safety / maintenance:

"something bad will never happen"
"something good will always hold"

Example:

$\Box\neg\mathsf{bankrupt}$

Specification Templates: Safety Properties

Safety / maintenance:

"something bad will never happen"
"something good will always hold"

Example:

$\Box \neg \text{bankrupt}$

Usually: $\Box \ldots$

Specification Templates: Reachability Properties

Reachability / achievement / liveness:

“something good will eventually happen”

Specification Templates: Reachability Properties

Reachability / achievement / liveness:

"something good will eventually happen"

Example:

$\Diamond \mathsf{rich}$

Specification Templates: Reachability Properties

Reachability / achievement / liveness:

"something good will eventually happen"

Example:

$\Diamond \mathsf{rich}$

$\Diamond \Box \mathsf{rich}$

Specification Templates: Reachability Properties

Reachability / achievement / liveness:

"something good will eventually happen"

Example:

$\Diamond \mathsf{rich}$

$\Diamond \Box \mathsf{rich}$

Usually: $\Diamond \ldots$

Specification Templates: Fairness Properties

Fairness / service:

“Whenever something is attempted/requested, then it will be successful/granted”

Specification Templates: Fairness Properties

Fairness / service:

"Whenever something is attempted/requested, then it will be successful/granted"

Example:

$\Box(\mathsf{needMoney} \rightarrow \Diamond\mathsf{getMoney})$

Specification Templates: Fairness Properties

Fairness / service:

"Whenever something is attempted/requested, then it will be successful/granted"

Example:

$\Box(\mathrm{needMoney} \rightarrow \Diamond \mathrm{getMoney})$

Usually: $\Box\Diamond \ldots$

Formal Semantics of Linear Time Logic (LTL)

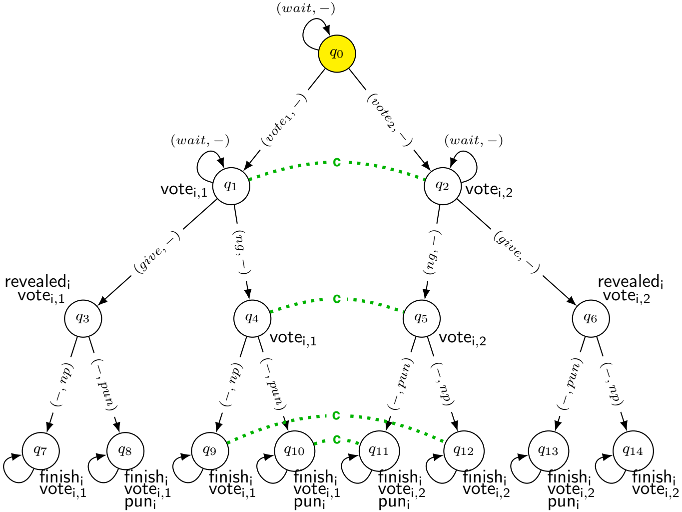
Definition 1.1 (Semantics of LTL)

$M, \lambda \models p$ iff p is true at moment $M, \lambda[0]$;

$M, \lambda \models \bigcirc\varphi$ iff $M, \lambda[1..\infty] \models \varphi$;
$M, \lambda \models \Diamond\varphi$ iff $M, \lambda[i..\infty] \models \varphi$ for some $i \geq 0$;
$M, \lambda \models \Box\varphi$ iff $M, \lambda[i..\infty] \models \varphi$ for all $i \geq 0$;
$M, \lambda \models \varphi \,\mathrm{U}\, \psi$ iff $M, \lambda[i..\infty] \models \psi$ for some $i \geq 0$, and $M, \lambda[j..\infty] \models \varphi$ for all $0 \leq j < i$.

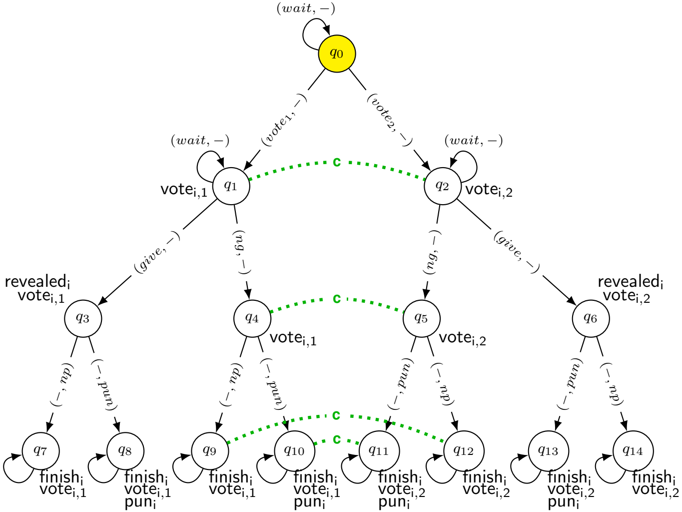
$M, \lambda \models \neg\varphi$ iff not $M, \lambda \models \varphi$;
$M, \lambda \models \varphi \wedge \psi$ iff $M, \lambda \models \varphi$ and $M, \lambda \models \psi$.

Example: Voting and Coercion



$\Diamond(\mathsf{vote}_{i,1} \vee \mathsf{vote}_{i,2})$

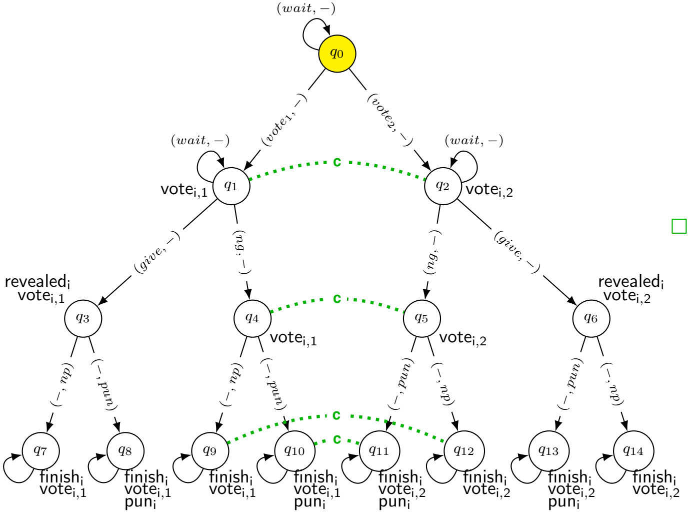
Example: Voting and Coercion



$\Diamond(\mathsf{vote}_{i,1} \vee \mathsf{vote}_{i,2})$

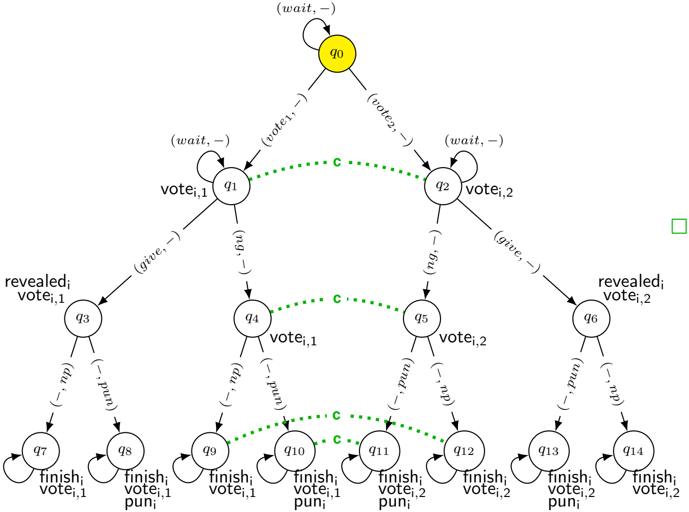
No!

Example: Voting and Coercion



$\Box(\mathsf{vote}_{i,1} \rightarrow \neg\Diamond\mathsf{vote}_{i,2})$

Example: Voting and Coercion



$\Box(\text{vote}_{i,1} \rightarrow \neg\Diamond\text{vote}_{i,2})$

Yes!

Motivating Example: Voting

- The voting system will not reveal how a particular voter voted

Motivating Example: Voting

- The voting system will not reveal how a particular voter voted

  $\Box\neg \mathsf{revealed}_i$

Motivating Example: Voting

- The voting system will not reveal how a particular voter voted

  $\Box\neg \mathsf{revealed}_i$

- The voter will eventually cast a vote

Motivating Example: Voting

- The voting system will not reveal how a particular voter voted

  $\Box \neg \mathsf{revealed}_\mathsf{i}$

- The voter will eventually cast a vote

  $\Diamond \bigvee_{j \in Cand} \mathsf{vote}_{\mathsf{i},\mathsf{j}}$

Motivating Example: Voting

- The voting system will not reveal how a particular voter voted

  $\Box\neg \mathsf{revealed}_i$

- The voter will eventually cast a vote

  $\Diamond \bigvee_{j\in Cand} \mathsf{vote}_{i,j}$

- The voter can vote, and can refrain from voting

Motivating Example: Voting

- The voting system will not reveal how a particular voter voted

  $\Box \neg \mathsf{revealed}_{\mathsf{i}}$

- The voter will eventually cast a vote

  $\Diamond \bigvee_{j \in Cand} \mathsf{vote}_{\mathsf{i,j}}$

- The voter can vote, and can refrain from voting

  Cannot be expressed in LTL

Part 1: Introduction to Model Checking for MAS

1.4 Strategic Abilities

Strategic Abilities

- So far, we have been able to specify how things must go, or how they may evolve

Strategic Abilities

- So far, we have been able to specify how things must go, or how they may evolve
- In multi-agent systems, it is often very important to know **who** can **make them evolve** in a particular way

Motivating Example: Voting

Properties to express

Privacy: The system cannot reveal how a particular voter voted

Enfranchisement: The voter can vote, and can refrain from voting

Coercion-resistance: The voter cannot convince the coercer that she voted in a certain way

Logical Specification of Strategic Abilities

- ATL: Alternating-time Temporal Logic [Alur et al. 1997-2002]
- Temporal logic meets game theory
- Main idea: cooperation modalities

$\langle\langle A\rangle\rangle\Phi$: coalition A has a collective strategy to enforce Φ

$\rightsquigarrow$ Φ can include temporal operators: $\bigcirc$ (next), $\Diamond$ (sometime in the future), $\Box$ (always in the future), U (strong until)

Example Formulas

- $\langle\langle jamesbond\rangle\rangle\Diamond(\mathsf{ski} \wedge \neg\mathsf{getBurned})$:
  "James Bond can go skiing without getting burned"

Example Formulas

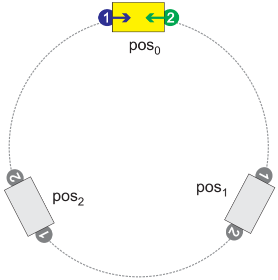
- $\langle\langle jamesbond \rangle\rangle \Diamond (\mathsf{ski} \land \neg \mathsf{getBurned})$:
  "James Bond can go skiing without getting burned"

Example Formulas

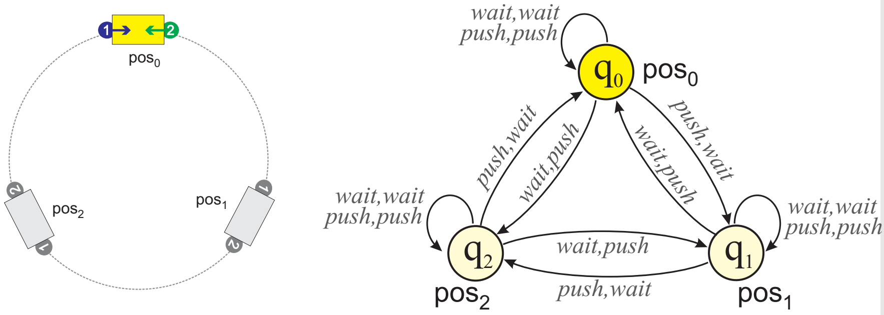
- $\langle\langle jamesbond \rangle\rangle \Diamond (\mathsf{ski} \wedge \neg \mathsf{getBurned})$:
  "James Bond can go skiing without getting burned"

- $\langle\langle jamesbond, bondsgirl \rangle\rangle (\neg \mathsf{destruction}) \, \mathrm{U} \, \mathsf{endOfMovie}$:
  "James Bond and his girlfriend are able to save the world from destruction until the end of the movie"

Example: Robots and Carriage



Example: Robots and Carriage



Strategies and Abilities

Strategy

A **strategy** is a conditional plan.
We represent strategies by functions $s_a : St \to Act$.

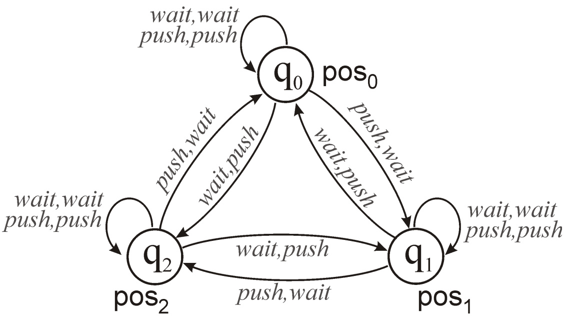
$\leadsto$ memoryless strategies

Alternative: perfect recall strategies $s_a : St^+ \to Act$

Semantics of ATL

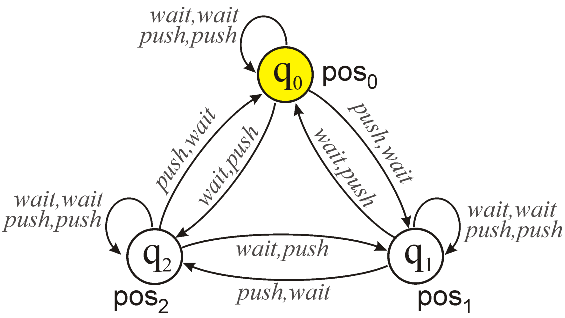
$M, q \models \langle\langle A \rangle\rangle \Phi$ iff there is a collective strategy s_A such that, for every path λ that may result from execution of s_A from q on, we have that $M, \lambda \models \Phi$.

Example: Robots and Carriage



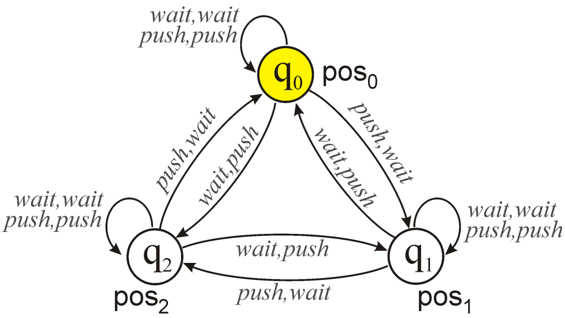
$\mathsf{pos}_0 \rightarrow \langle\langle 1 \rangle\rangle \Diamond \mathsf{pos}_1$

Example: Robots and Carriage



$\mathsf{pos}_0 \rightarrow \langle\langle 1 \rangle\rangle \Diamond \mathsf{pos}_1$

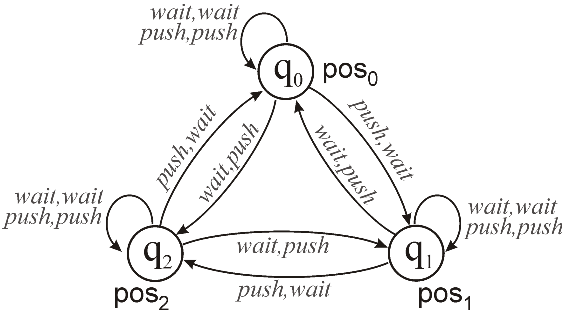
Example: Robots and Carriage



$\mathsf{pos}_0 \to \langle\langle 1 \rangle\rangle \Diamond \mathsf{pos}_1$

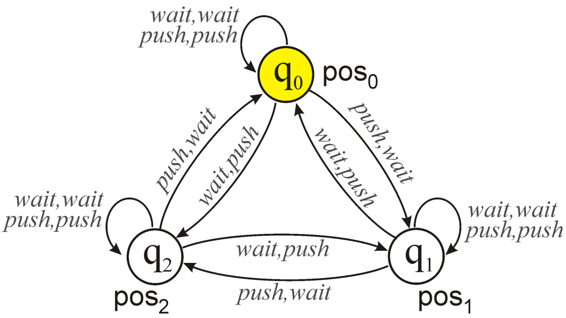
No!

Example: Robots and Carriage



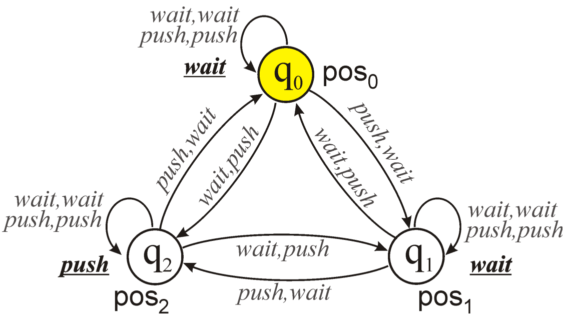
$pos_0 \rightarrow \langle\langle 1 \rangle\rangle \Box \neg pos_1$

Example: Robots and Carriage



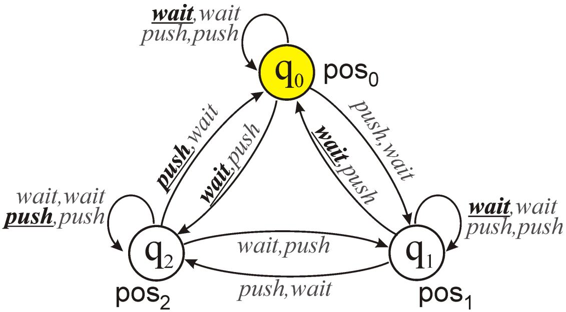
$$pos_0 \rightarrow \langle\langle 1 \rangle\rangle \Box \neg pos_1$$

Example: Robots and Carriage



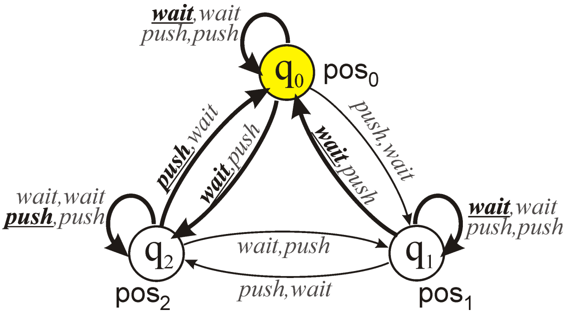
$\mathsf{pos}_0 \rightarrow \langle\langle 1 \rangle\rangle \Box \neg \mathsf{pos}_1$

Example: Robots and Carriage



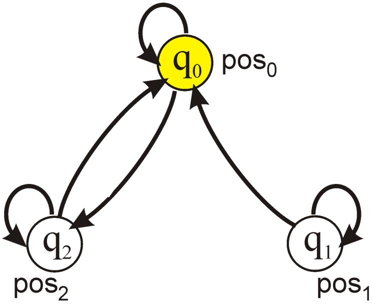
$pos_0 \rightarrow \langle\langle 1 \rangle\rangle \Box \neg pos_1$

Example: Robots and Carriage



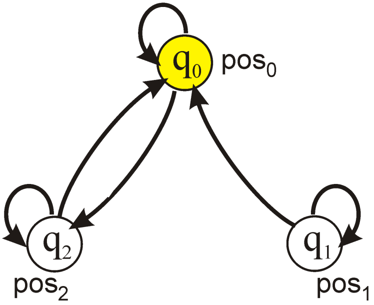
$\mathsf{pos}_0 \rightarrow \langle\langle 1 \rangle\rangle \Box \neg \mathsf{pos}_1$

Example: Robots and Carriage



$\mathsf{pos}_0 \rightarrow \langle\langle 1 \rangle\rangle \Box \neg \mathsf{pos}_1$

Example: Robots and Carriage



$\mathsf{pos}_0 \to \langle\langle 1 \rangle\rangle \Box \neg \mathsf{pos}_1$

Yes!

Part 1: Introduction to Model Checking for MAS

1.5 Imperfect Information

Executable Strategies under Imperfect Information

Strategies under imperfect information must be executable $\rightsquigarrow$ **uniform strategies**

Definition 1.2 (Uniform strategy)

Strategy s_a is **uniform** iff it specifies the same choices for indistinguishable situations:

- (no recall:) if $q \sim_a q'$ then $s_a(q) = s_a(q')$
- (perfect recall:) if $h \approx_a h'$ then $s_a(h) = s_a(h')$
  where $h \approx_a h'$ iff $h[i] \sim_a h'[i]$ for every i.

Executable Strategies under Imperfect Information

Strategies under imperfect information must be executable $\rightsquigarrow$ **uniform strategies**

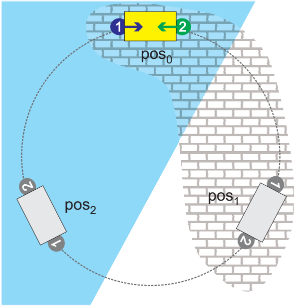
Definition 1.2 (Uniform strategy)

Strategy s_a is **uniform** iff it specifies the same choices for indistinguishable situations:

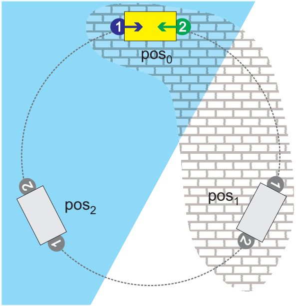
- (no recall:) if $q \sim_a q'$ then $s_a(q) = s_a(q')$
- (perfect recall:) if $h \approx_a h'$ then $s_a(h) = s_a(h')$
  where $h \approx_a h'$ iff $h[i] \sim_a h'[i]$ for every i.

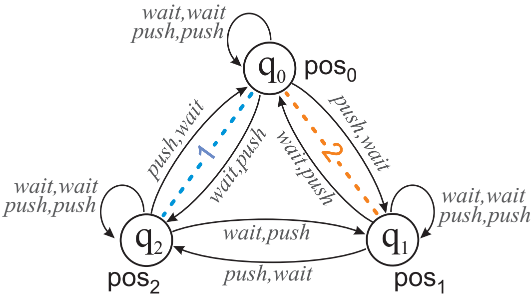
A collective strategy is uniform iff it consists only of uniform individual strategies.

Example: Robots and Carriage

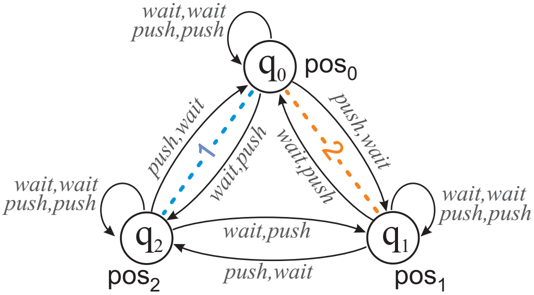


Example: Robots and Carriage

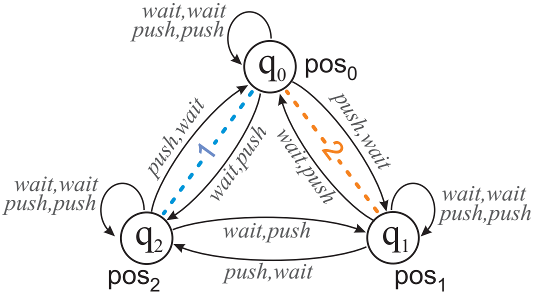




Example: Robots and Carriage

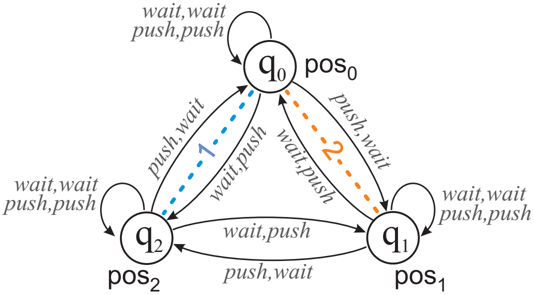


Example: Robots and Carriage



$\mathsf{pos}_0 \rightarrow \neg\langle\langle 1 \rangle\rangle_{ir} \Box \neg \mathsf{pos}_1$

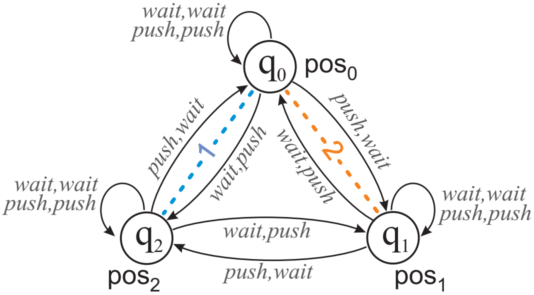
Example: Robots and Carriage



$\mathsf{pos}_0 \rightarrow \neg\langle\langle 1 \rangle\rangle_{ir} \Box \neg \mathsf{pos}_1$

$\mathsf{pos}_0 \rightarrow \neg\langle\langle 1, 2 \rangle\rangle_{ir} \Box \neg \mathsf{pos}_1$

Example: Robots and Carriage



$\mathsf{pos}_0 \to \neg\langle\langle 1\rangle\rangle_{ir}\Box\neg\mathsf{pos}_1$

$\mathsf{pos}_0 \to \neg\langle\langle 1,2\rangle\rangle_{ir}\Box\neg\mathsf{pos}_1$

$\mathsf{pos}_0 \to \langle\langle 1,2\rangle\rangle_{ir}\Diamond\mathsf{pos}_1$

Strategies and Knowledge

Note:

Having a successful strategy does not imply knowing that we have it!

Strategies and Knowledge

Note:

Having a successful strategy does not imply knowing that we have it!

Knowing that a successful strategy exists does not imply knowing the strategy itself!

Motivating Example: Voting

- The system cannot reveal how a particular voter voted

Motivating Example: Voting

- The system cannot reveal how a particular voter voted

  $\neg\langle\langle system\rangle\rangle\Diamond \mathsf{revealed_i}$

Motivating Example: Voting

- The system cannot reveal how a particular voter voted

  $\neg\langle\langle system \rangle\rangle \Diamond \mathsf{revealed_i}$

- The voter can vote, and can refrain from voting

Motivating Example: Voting

- The system cannot reveal how a particular voter voted

  $\neg\langle\langle system \rangle\rangle \Diamond \mathsf{revealed_i}$

- The voter can vote, and can refrain from voting

  $\langle\langle i \rangle\rangle \Diamond (\bigvee_{j \in \mathit{Cand}} \mathsf{vote_{i,j}}) \wedge \langle\langle i \rangle\rangle \Box (\bigwedge_{j \in \mathit{Cand}} \neg \mathsf{vote_{i,j}})$

Motivating Example: Voting

- The system cannot reveal how a particular voter voted

  $$\neg\langle\langle system\rangle\rangle\Diamond \mathsf{revealed_i}$$

- The voter can vote, and can refrain from voting

  $$\langle\langle i\rangle\rangle\Diamond(\bigvee_{j\in \mathit{Cand}} \mathsf{vote_{i,j}}) \wedge \langle\langle i\rangle\rangle\Box(\bigwedge_{j\in \mathit{Cand}} \neg\mathsf{vote_{i,j}})$$

  Stronger variant:

  $$\bigwedge_{j\in \mathit{Cand}}\langle\langle i\rangle\rangle\Diamond \mathsf{vote_{i,j}} \wedge \langle\langle i\rangle\rangle\Box(\bigwedge_{j\in \mathit{Cand}} \neg\mathsf{vote_{i,j}})$$

Motivating Example: Voting

- The system cannot reveal how a particular voter voted

  $\neg\langle\langle system\rangle\rangle\Diamond \mathsf{revealed}_{\mathsf{i}}$

- The voter can vote, and can refrain from voting

  $\langle\langle i\rangle\rangle\Diamond(\bigvee_{j\in \mathit{Cand}} \mathsf{vote}_{\mathsf{i,j}}) \wedge \langle\langle i\rangle\rangle\Box(\bigwedge_{j\in \mathit{Cand}} \neg\mathsf{vote}_{\mathsf{i,j}})$

  Stronger variant:

  $\bigwedge_{j\in \mathit{Cand}}\langle\langle i\rangle\rangle\Diamond \mathsf{vote}_{\mathsf{i,j}} \wedge \langle\langle i\rangle\rangle\Box(\bigwedge_{j\in \mathit{Cand}} \neg\mathsf{vote}_{\mathsf{i,j}})$

- The voter can't convince the coercer that she voted in a certain way

Motivating Example: Voting

- The system cannot reveal how a particular voter voted

  $\neg\langle\langle system\rangle\rangle\Diamond \mathsf{revealed_i}$

- The voter can vote, and can refrain from voting

  $\langle\langle i\rangle\rangle\Diamond(\bigvee_{j\in \mathit{Cand}} \mathsf{vote_{i,j}}) \wedge \langle\langle i\rangle\rangle\Box(\bigwedge_{j\in \mathit{Cand}} \neg\mathsf{vote_{i,j}})$

  Stronger variant:

  $\bigwedge_{j\in \mathit{Cand}}\langle\langle i\rangle\rangle\Diamond \mathsf{vote_{i,j}} \wedge \langle\langle i\rangle\rangle\Box(\bigwedge_{j\in \mathit{Cand}} \neg\mathsf{vote_{i,j}})$

- The voter can't convince the coercer that she voted in a certain way

  $\neg\langle\langle i, c\rangle\rangle\Diamond \ldots ?$

Motivating Example: Voting

- The system cannot reveal how a particular voter voted

  $\neg\langle\langle system\rangle\rangle\Diamond \mathsf{revealed_i}$

- The voter can vote, and can refrain from voting

  $\langle\langle i\rangle\rangle\Diamond(\bigvee_{j\in \mathit{Cand}} \mathsf{vote_{i,j}}) \wedge \langle\langle i\rangle\rangle\Box(\bigwedge_{j\in \mathit{Cand}} \neg\mathsf{vote_{i,j}})$

  Stronger variant:

  $\bigwedge_{j\in \mathit{Cand}}\langle\langle i\rangle\rangle\Diamond \mathsf{vote_{i,j}} \wedge \langle\langle i\rangle\rangle\Box(\bigwedge_{j\in \mathit{Cand}} \neg\mathsf{vote_{i,j}})$

- The voter can't convince the coercer that she voted in a certain way

  $\neg\langle\langle i,c\rangle\rangle\Diamond \ldots ?$

  Cannot be expressed in ATL (we need a notion of knowledge for the coercer) !

Part 1: Introduction to Model Checking for MAS

1.6 Adding Knowledge Operators

Adding Knowledge Operators

- Epistemic operators: $K_i\varphi$ ("i knows that φ")
- Semantics: φ holds in all the states that look the same as the current state to i

$M, q \models K_i\varphi$ iff $M, q' \models \varphi$ for all q' such that $q \sim_i q'$

Motivating Example: Voting

- The voter cannot convince the coercer that she voted in a certain way

Motivating Example: Voting

- The voter cannot convince the coercer that she voted in a certain way

  $\bigwedge_{j \in \mathit{Cand}} \neg \langle\langle i \rangle\rangle \Diamond K_c \mathsf{voted}_{\mathsf{i,j}}$

Motivating Example: Voting

- The voter cannot convince the coercer that she voted in a certain way

  $\bigwedge_{j \in Cand} \neg\langle\langle i \rangle\rangle \Diamond K_c \mathsf{voted}_{i,j}$

  Better specification: $\bigwedge_{j \in Cand} \neg\langle\langle i, c \rangle\rangle \Diamond K_c \mathsf{voted}_{i,j}$

Motivating Example: Voting

- The voter cannot convince the coercer that she voted in a certain way

  $\bigwedge_{j \in \mathit{Cand}} \neg\langle\langle i \rangle\rangle \Diamond K_c \mathsf{voted_{i,j}}$

  Better specification: $\bigwedge_{j \in \mathit{Cand}} \neg\langle\langle i, c \rangle\rangle \Diamond K_c \mathsf{voted_{i,j}}$

  Even better: $\bigwedge_{C \subsetneq \mathit{Cand}} \neg\langle\langle i, c \rangle\rangle \Diamond K_c \left(\bigvee_{j \in C} \mathsf{voted_{i,j}} \right)$

Part 1: Introduction to Model Checking for MAS

1.7 Model Checking

Verification by Model Checking

Model checking problem

$$M, q \overset{?}{\models} \varphi$$

transition system — modal formula

Verification by Model Checking

Model checking problem

$$M, q \overset{?}{\models} \varphi$$

transition system modal formula

That is, we want to implement function $mcheck(M, q, \varphi)$ such that:

$$mcheck(M, q, \varphi) = \begin{cases} \top & \text{if} \quad M, q \models \varphi \\ \bot & \text{else} \end{cases}$$

Verification by Model Checking

Model checking problem

$$M, q \overset{?}{\models} \varphi$$

transition system modal formula

That is, we want to implement function $mcheck(M, q, \varphi)$ such that:

$$mcheck(M, q, \varphi) = \begin{cases} \top & \text{if} \quad M, q \models \varphi \\ \bot & \text{else} \end{cases}$$

This problem is sometimes called **local model checking**

Verification by Model Checking

Local vs. Global Model Checking

Local model checking

We want to implement function

$$mcheck(M, q, \varphi) = \begin{cases} \top & \text{if} \quad M, q \models \varphi \\ \bot & \text{else} \end{cases}$$

Local vs. Global Model Checking

Local model checking

We want to implement function

$$mcheck(M, q, \varphi) = \begin{cases} \top & \text{if} \quad M, q \models \varphi \\ \bot & \text{else} \end{cases}$$

Alternative: ask for the set of states that satisfy φ!

Global model checking

We want to implement function

$$mcheck(M, \varphi) = \{q \in St \mid M, q \models \varphi\}$$

Local vs. Global Model Checking

Local model checking

We want to implement function

$$mcheck(M, q, \varphi) = \begin{cases} \top & \text{if} \quad M, q \models \varphi \\ \bot & \text{else} \end{cases}$$

Alternative: ask for the set of states that satisfy φ!

Global model checking

We want to implement function

$$mcheck(M, \varphi) = \{q \in St \mid M, q \models \varphi\}$$

Often no harder than local model checking...

Part 2: Verification of Strategic Ability

Part 2: Verification of Strategic Ability

2.1 Fixpoint Algorithm

Fixpoint Algorithm for ATL with Perfect Information

A well-known nice result: model checking ATL for agents with perfect information is tractable!

Theorem (Alur, Kupferman & Henzinger 1998/2002)

Model checking ATL with perfect information is P-complete, and can be done in linear time.

Fixpoint Algorithm for ATL with Perfect Information

A well-known nice result: model checking ATL for agents with perfect information is tractable!

Theorem (Alur, Kupferman & Henzinger 1998/2002)

Model checking ATL with perfect information is P-complete, and can be done in time linear wrt the size of the model and the length of the formula.

Fixpoint Algorithm for ATL with Perfect Information

A well-known nice result: model checking ATL for agents with perfect information is tractable!

Theorem (Alur, Kupferman & Henzinger 1998/2002)

Model checking ATL with perfect information is P-complete, and can be done in time $O(ml)$ where $m =$ #transitions in the model and $l =$ #symbols in the formula.

Fixpoint Algorithm for ATL with Perfect Information

The algorithm is based on the following **fixpoint equivalences**:

- $\langle\langle A\rangle\rangle\Box\varphi \quad \leftrightarrow \quad \varphi \land \langle\langle A\rangle\rangle \bigcirc \langle\langle A\rangle\rangle\Box\varphi$

Fixpoint Algorithm for ATL with Perfect Information

The algorithm is based on the following **fixpoint equivalences**:

- $\langle\!\langle A \rangle\!\rangle \Box \varphi \quad \leftrightarrow \quad \varphi \wedge \langle\!\langle A \rangle\!\rangle \bigcirc \langle\!\langle A \rangle\!\rangle \Box \varphi$
- $\langle\!\langle A \rangle\!\rangle \Diamond \varphi \quad \leftrightarrow \quad \varphi \vee \langle\!\langle A \rangle\!\rangle \bigcirc \langle\!\langle A \rangle\!\rangle \Diamond \varphi$

Fixpoint Algorithm for ATL with Perfect Information

The algorithm is based on the following **fixpoint equivalences**:

- $\langle\langle A\rangle\rangle\Box\varphi \quad \leftrightarrow \quad \varphi \wedge \langle\langle A\rangle\rangle \bigcirc \langle\langle A\rangle\rangle\Box\varphi$
- $\langle\langle A\rangle\rangle\Diamond\varphi \quad \leftrightarrow \quad \varphi \vee \langle\langle A\rangle\rangle \bigcirc \langle\langle A\rangle\rangle\Diamond\varphi$
- $\langle\langle A\rangle\rangle\varphi_1 \,\mathrm{U}\, \varphi_2 \quad \leftrightarrow \quad \varphi_2 \vee \varphi_1 \wedge \langle\langle A\rangle\rangle \bigcirc \langle\langle A\rangle\rangle\varphi_1 \,\mathrm{U}\, \varphi_2$.

Fixpoint Algorithm for ATL with Perfect Information

The algorithm is based on the following **fixpoint equivalences**:

- $\langle\langle A\rangle\rangle\Box\varphi \quad\leftrightarrow\quad \varphi \wedge \langle\langle A\rangle\rangle \bigcirc \langle\langle A\rangle\rangle\Box\varphi$
- $\langle\langle A\rangle\rangle\Diamond\varphi \quad\leftrightarrow\quad \varphi \vee \langle\langle A\rangle\rangle \bigcirc \langle\langle A\rangle\rangle\Diamond\varphi$
- $\langle\langle A\rangle\rangle\varphi_1 \,\mathrm{U}\, \varphi_2 \quad\leftrightarrow\quad \varphi_2 \vee \varphi_1 \wedge \langle\langle A\rangle\rangle \bigcirc \langle\langle A\rangle\rangle\varphi_1 \,\mathrm{U}\, \varphi_2$.

Perfect information strategies for reachability/safety objectives can be synthesized incrementally (no backtracking is necessary).

function $mcheck(\mathcal{M}, \varphi)$.
Global model checking formulae of ATL.
Returns the exact subset of St for which formula φ holds.

case $\varphi \equiv p$ **:** return $V(p)$
case $\varphi \equiv \neg\psi$ **:** return $St \setminus mcheck(\mathcal{M}, \psi)$
case $\varphi \equiv \psi_1 \wedge \psi_2$ **:** return $mcheck(\mathcal{M}, \psi_1) \cap mcheck(\mathcal{M}, \psi_2)$
case $\varphi \equiv \langle\langle A\rangle\rangle \bigcirc \psi$ **:** return $pre(A, mcheck(\mathcal{M}, \psi))$
case $\varphi \equiv \langle\langle A\rangle\rangle \Box \psi$ **:**
$Q_1 := Q;\quad Q_2 := Q_3 := mcheck(\mathcal{M}, \psi);$
while $Q_1 \not\subseteq Q_2$ **do** $Q_1 := Q_1 \cap Q_2;\; Q_2 := pre(A, Q_1) \cap Q_3$ **od**;
return Q_1
case $\varphi \equiv \langle\langle A\rangle\rangle \psi_1 \,\mathrm{U}\, \psi_2$ **:**
$Q_1 := \emptyset;\quad Q_2 := mcheck(\mathcal{M}, \psi_2);\quad Q_3 := mcheck(\mathcal{M}, \psi_1);$
while $Q_2 \not\subseteq Q_1$ **do** $Q_1 := Q_1 \cup Q_2;\; Q_2 := pre(A, Q_1) \cap Q_3$ **od**;
return Q_1
end case

$$\text{pre}(A, Q) = \{q \mid \exists\alpha_A \forall\alpha_{\mathbb{A}\mathrm{gt}\setminus A} \,.\, t(q, \alpha_A, \alpha_{\mathbb{A}\mathrm{gt}\setminus A}) \in Q\}$$
where t is the transition function

Example: Simple Rocket Domain

- Assume that there are 3 workers in the rocket (agents 1, 2, and 3)

Example: Simple Rocket Domain

- Assume that there are 3 workers in the rocket (agents 1, 2, and 3)

- Each agent has different capabilities
- Agent 1 can: try to load the cargo, try to unload the cargo, initiate the flight, or do nothing (action nop)
- Agent 2 can do unload or nop
- Agent 3 can do load, refill the fuel tank (action fuel), or do nop

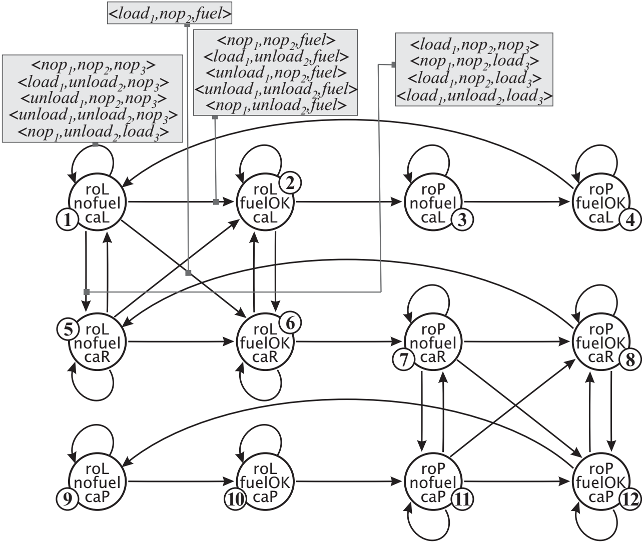
Example: Simple Rocket Domain

- Assume that there are 3 workers in the rocket (agents 1, 2, and 3)

- Each agent has different capabilities
- Agent 1 can: try to load the cargo, try to unload the cargo, initiate the flight, or do nothing (action nop)
- Agent 2 can do unload or nop
- Agent 3 can do load, refill the fuel tank (action fuel), or do nop

- Flying has highest priority: if agent 1 initiates the flight, current actions of the other agents have no effect
- If loading is attempted when the cargo is not around, nothing happens
- Same for unloading when the cargo is not in the rocket, and refilling a full tank
- If different agents try to load and unload at the same time then the majority prevails
- Refilling fuel can be done in parallel with loading/unloading

Example: Simple Rocket Domain



Example: Simple Rocket Domain

- Verification example: we want to find the set of states from which agents 1 and 3 can move the cargo to any given location.
- $\langle\langle 1, 3\rangle\rangle \Diamond \mathsf{caP}$

Example: Simple Rocket Domain

- Verification example: we want to find the set of states from which agents 1 and 3 can move the cargo to any given location.
- $\langle\langle 1, 3\rangle\rangle\Diamond \mathsf{caP} \;\wedge\; \langle\langle 1, 3\rangle\rangle\Diamond \mathsf{caL}$

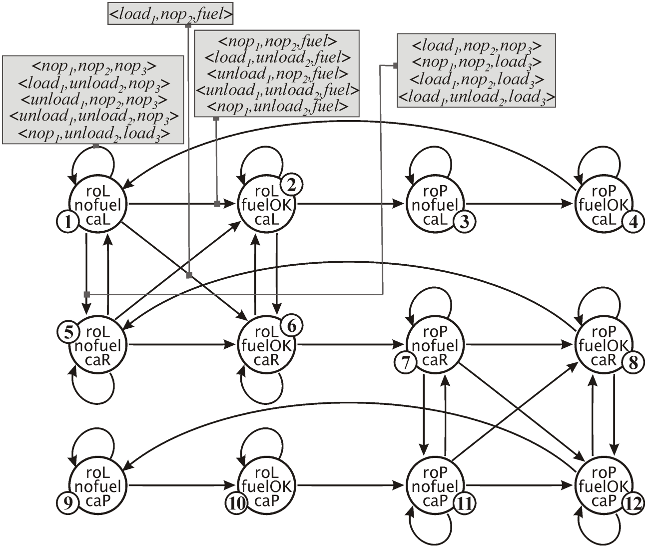
Example: Simple Rocket Domain

- Verification example: we want to find the set of states from which agents 1 and 3 can move the cargo to any given location.
- $\langle\langle 1,3\rangle\rangle\Diamond \mathsf{caP} \ \wedge \langle\langle 1,3\rangle\rangle\Diamond \mathsf{caL}$
- How does that work for the coalition of agents 1 and 2 ($\langle\langle 1,2\rangle\rangle\Diamond \mathsf{caP}$) ?

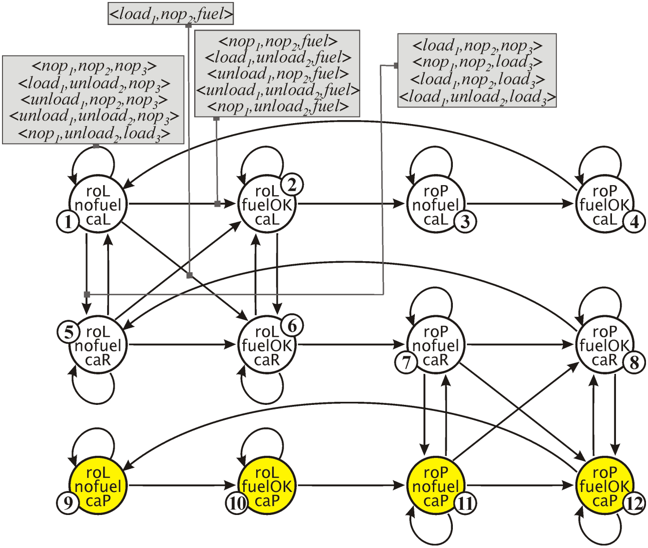
Example: Simple Rocket Domain

- Verification example: we want to find the set of states from which agents 1 and 3 can move the cargo to any given location.
- $\langle\langle 1,3\rangle\rangle\Diamond \mathsf{caP} \;\wedge\; \langle\langle 1,3\rangle\rangle\Diamond \mathsf{caL}$
- How does that work for the coalition of agents 1 and 2 ($\langle\langle 1,2\rangle\rangle\Diamond \mathsf{caP}$) ?
- What about a maintenance goal, like agent 3 keeping the cargo in Paris forever ($\langle\langle 3\rangle\rangle\Box \mathsf{caP}$) ?

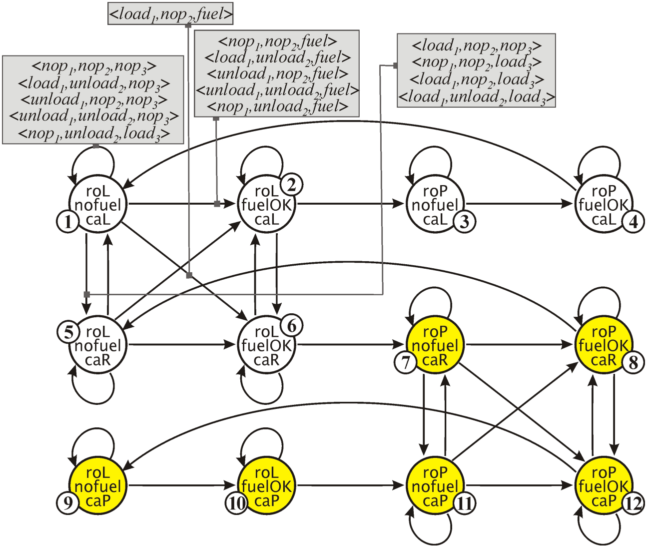
Simple Rocket Domain: Verification of $\langle\langle 1, 3\rangle\rangle \Diamond \mathsf{caP}$



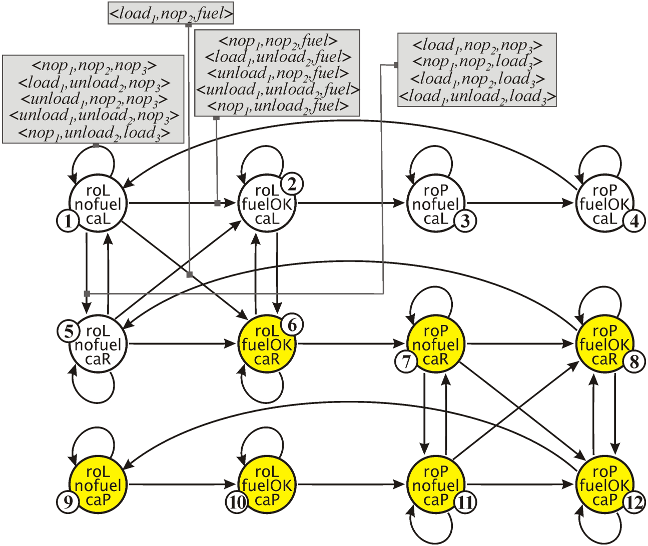
Simple Rocket Domain: Verification of $\langle\langle 1,3\rangle\rangle\Diamond$caP



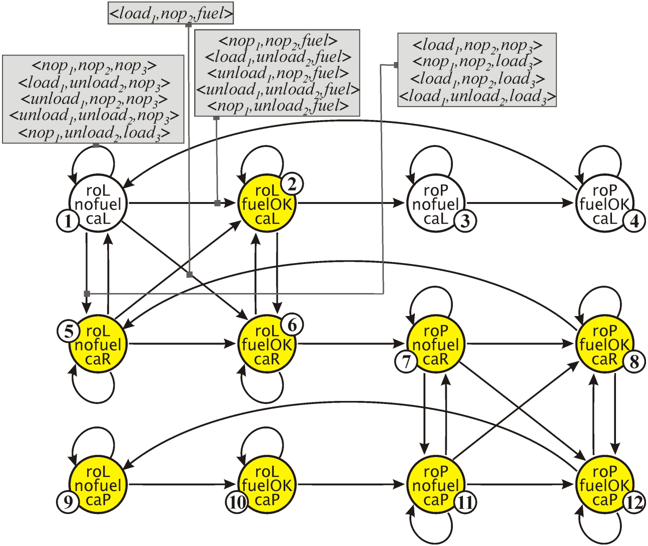
Simple Rocket Domain: Verification of $\langle\langle 1,3\rangle\rangle\Diamond$caP



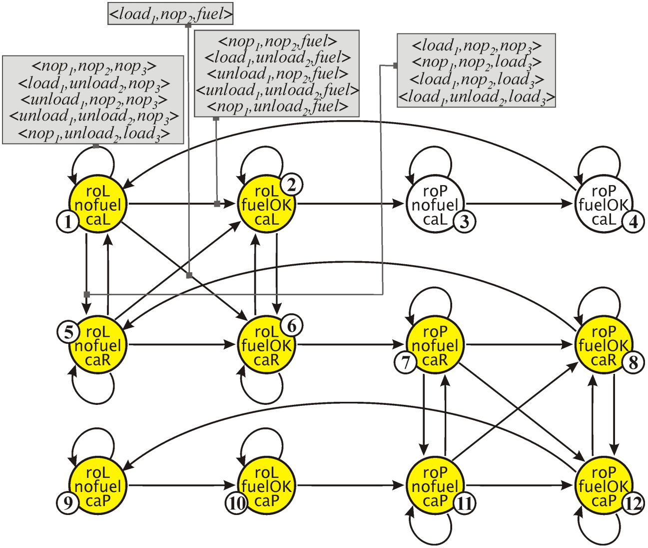
Simple Rocket Domain: Verification of $\langle\langle 1, 3\rangle\rangle \Diamond$caP



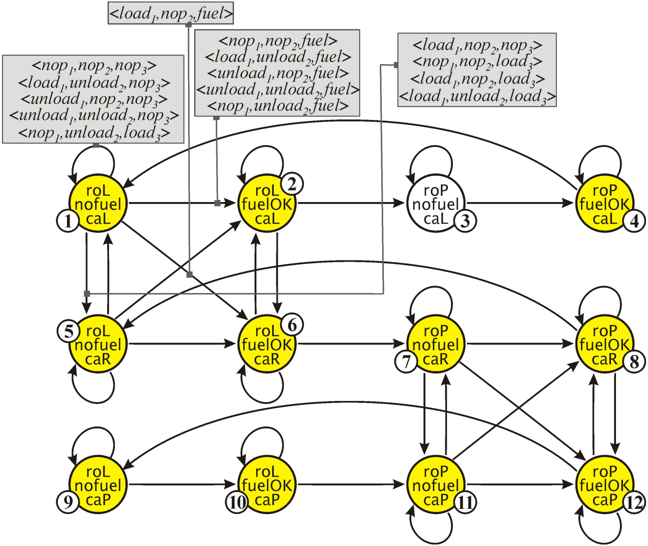
Simple Rocket Domain: Verification of $\langle\langle 1,3 \rangle\rangle \Diamond \mathsf{caP}$



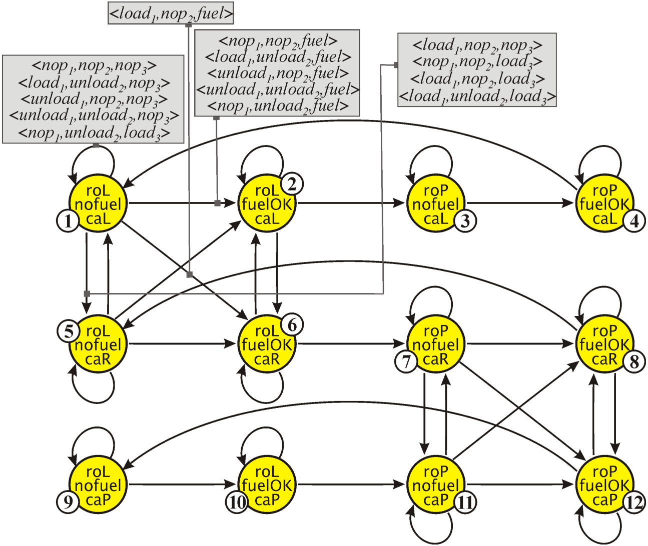
Simple Rocket Domain: Verification of $\langle\langle 1, 3\rangle\rangle\Diamond$caP



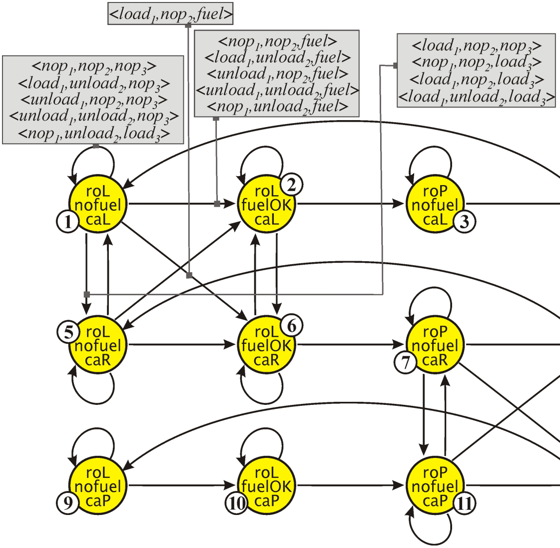
Simple Rocket Domain: Verification of $\langle\langle 1,3\rangle\rangle\Diamond$caP



Simple Rocket Domain: Verification of $\langle\langle 1, 3\rangle\rangle\Diamond$caP

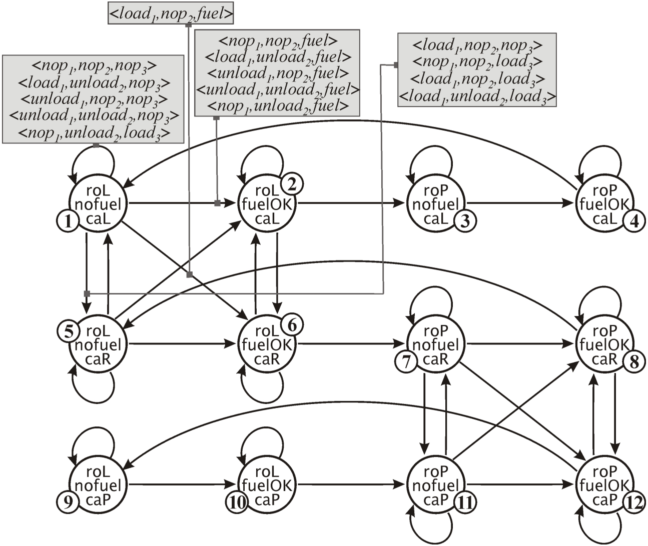


Simple Rocket Domain: Verification of $\langle\langle 1,3\rangle\rangle\Diamond$caP

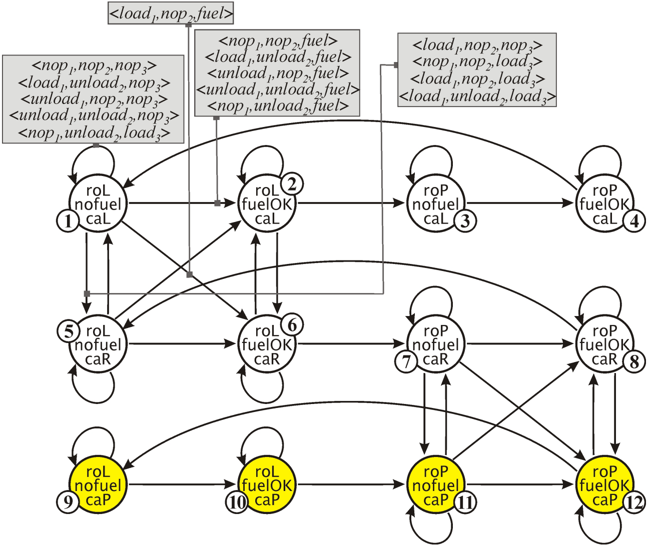


Done!

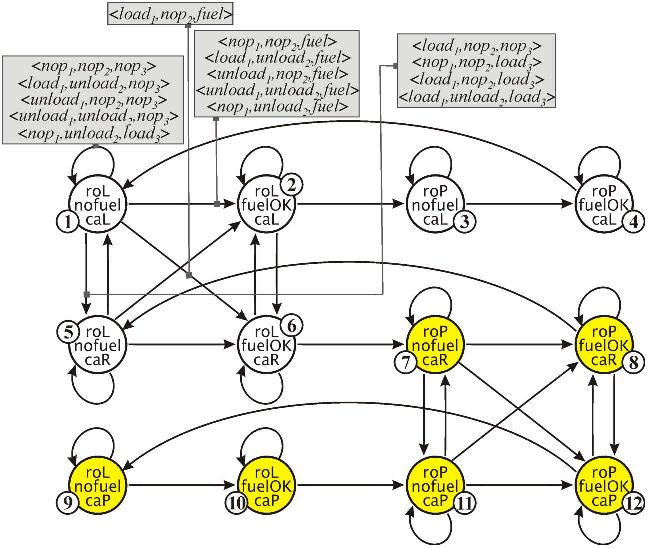
Simple Rocket Domain: Verification of $\langle\langle 1, 2\rangle\rangle\Diamond$caP



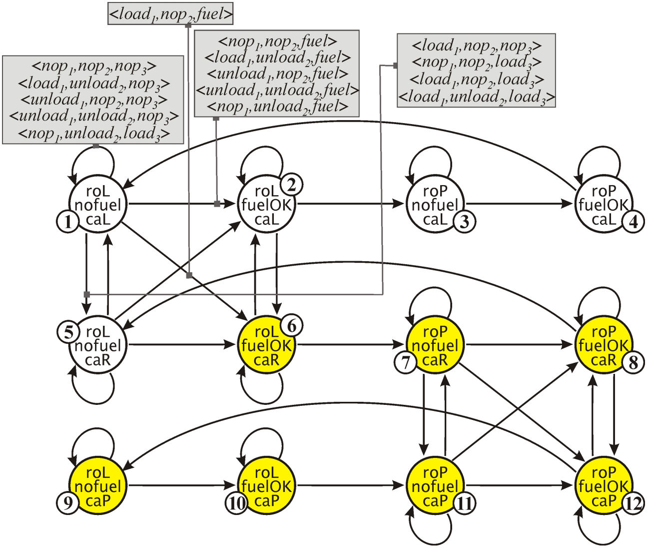
Simple Rocket Domain: Verification of $\langle\langle 1, 2\rangle\rangle\Diamond$caP



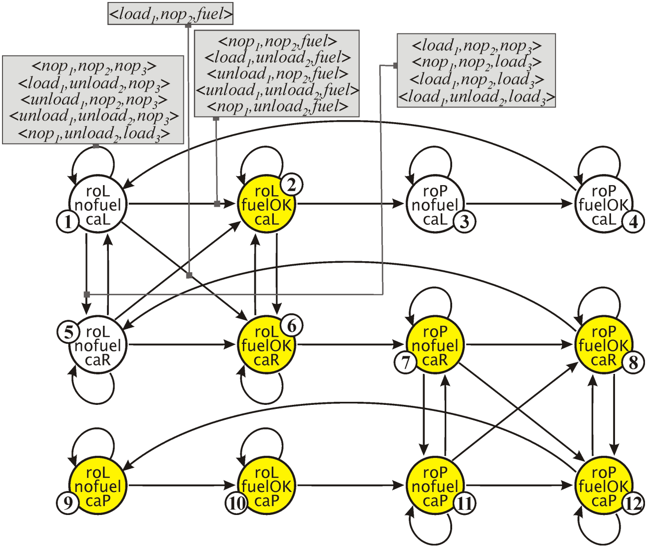
Simple Rocket Domain: Verification of $\langle\langle 1,2\rangle\rangle\Diamond$caP



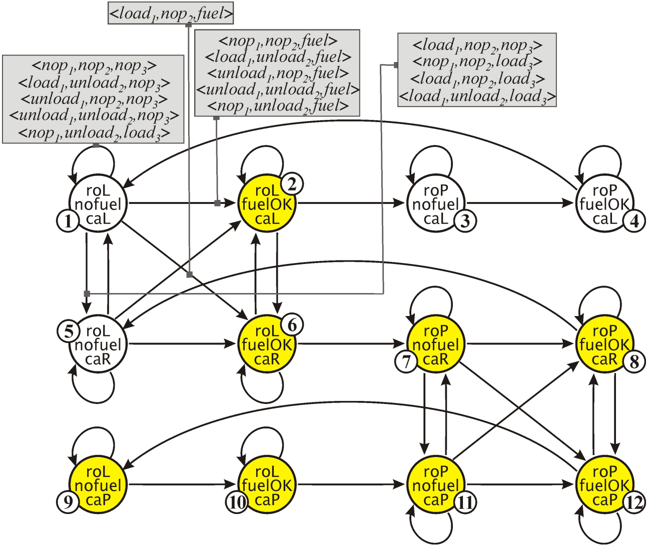
Simple Rocket Domain: Verification of $\langle\langle 1,2\rangle\rangle\Diamond$caP



Simple Rocket Domain: Verification of $\langle\langle 1,2\rangle\rangle \Diamond$caP

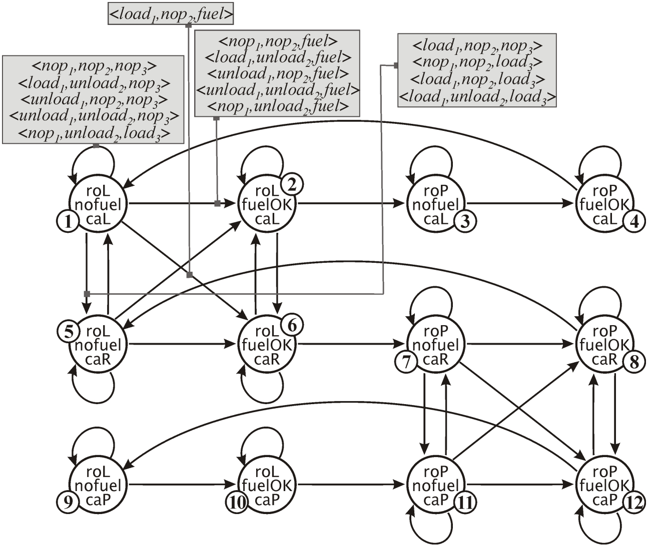


Simple Rocket Domain: Verification of $\langle\langle 1, 2\rangle\rangle\Diamond$caP

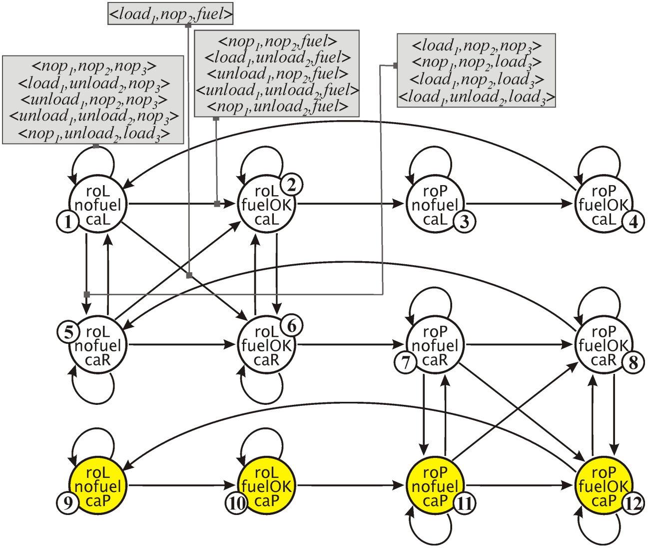


Done!

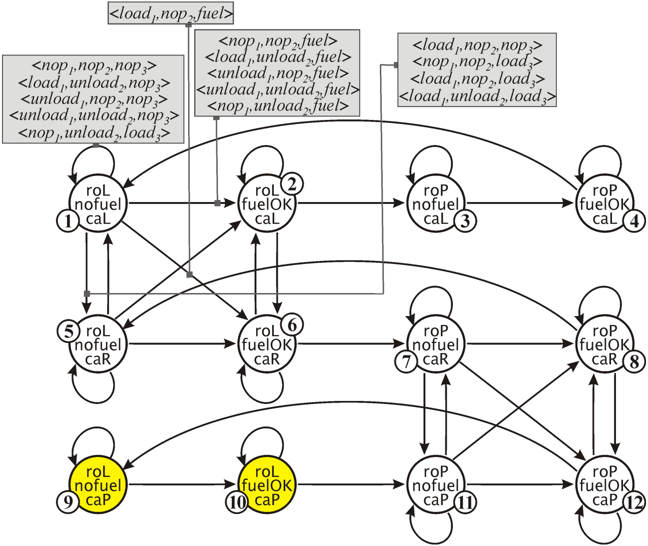
Simple Rocket Domain: Verification of $\langle\langle 3 \rangle\rangle \Box$caP



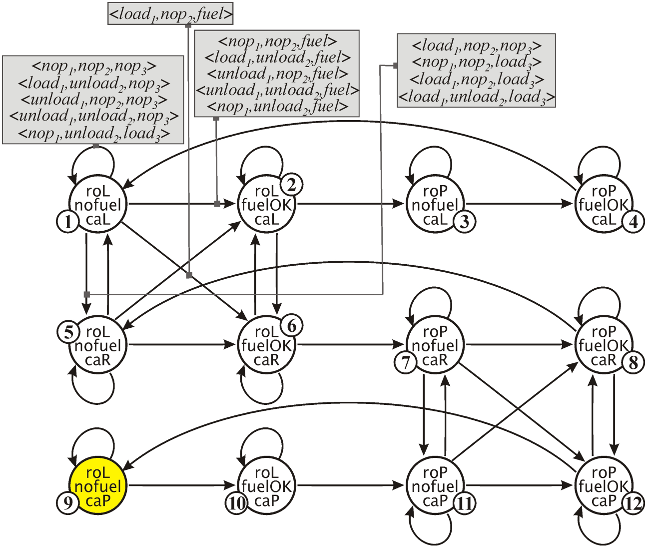
Simple Rocket Domain: Verification of $\langle\langle 3\rangle\rangle\Box$caP



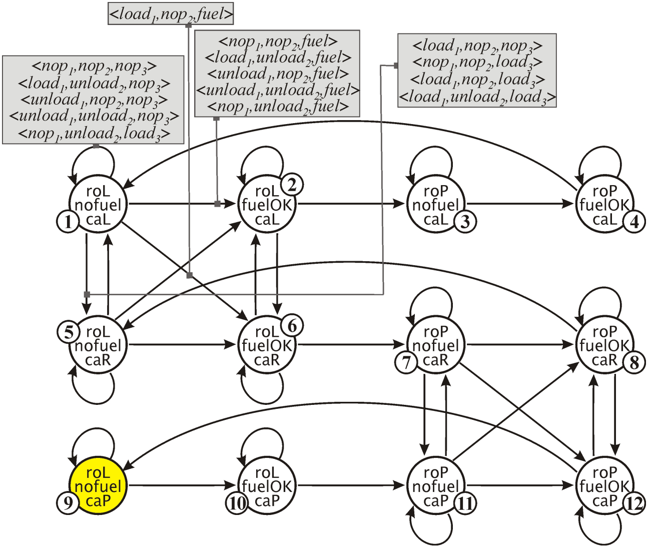
Simple Rocket Domain: Verification of $\langle\langle 3\rangle\rangle\Box$caP



Simple Rocket Domain: Verification of $\langle\langle 3 \rangle\rangle \Box \mathsf{caP}$



Simple Rocket Domain: Verification of $\langle\langle 3\rangle\rangle\Box$caP



Done!

Model Checking ATL for Perfect Information

Theorem (Alur, Kupferman & Henzinger 1998/2002)

ATL model checking for perfect information games is $\mathbf{P}$-complete, and can be done in linear time.

So... let's model check!

Not That Easy...

Challenges:

- Preparing the model $\rightsquigarrow$ socio-technical system!
- Writing formula for the requirement(s)

Not That Easy...

Challenges:

- Preparing the model $\rightsquigarrow$ socio-technical system!
- Writing formula for the requirement(s)

- State- and transition-space explosion
- Invalidity of fixpoint equivalences for imperfect information

Part 2: Verification of Strategic Ability

2.2 Imperfect Information

Model Checking ATL: Imperfect Information

Theorem (Schobbens 2004; Jamroga & Dix 2006)

Model checking ATL for agents with imperfect information playing memoryless strategies is Δ_2-complete in the number of transitions in the model and the length of the formula.

(where $\mathbf{\Delta_2^P}$ is the class of problems solvable in polynomial time by a deterministic Turing machine making adaptive calls to an oracle solving NP problems)

Model Checking ATL: Imperfect Information

Theorem (Schobbens 2004; Jamroga & Dix 2006)

Model checking ATL for agents with imperfect information playing memoryless strategies is Δ_2-complete in the number of transitions in the model and the length of the formula.

(where $\mathbf{\Delta_2^P}$ is the class of problems solvable in polynomial time by a deterministic Turing machine making adaptive calls to an oracle solving NP problems)

Corollary

Imperfect information strategies cannot be synthesized incrementally: we cannot do better than guess the whole strategy and check if it succeeds.

Model Checking ATL: Imperfect Info, Perfect Recall

What about agents with perfect recall and imperfect information?
The news are bad...

Theorem (Dima and Tiplea, 2011)

Model checking ATL for agents with imperfect information and perfect recall is **undecidable**.

The problem is undecidable even for turn-based models with 3 players, and flat formulae with only doubleton coalitions.

It Really Takes Two (To Make Things Undecidable)...

Theorem (Guelev, Dima, and Enea, 2010)

Model checking ATL for singleton coalitions with imperfect information and perfect recall is **EXPTIME-complete**.

Part 3: Practical Model Checking

Part 3: Practical Model Checking

3.1 State-Space Explosion

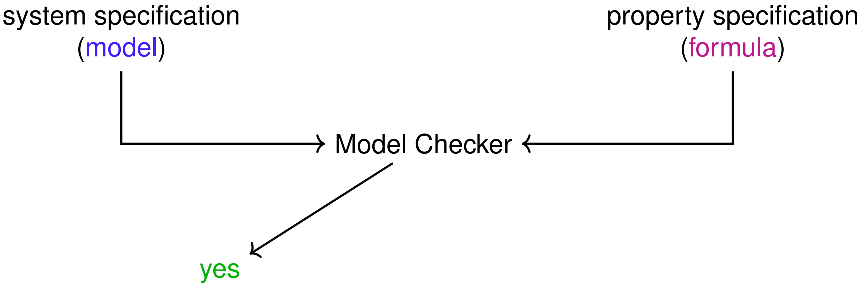
Idea of Model Checking

system specification
(model)

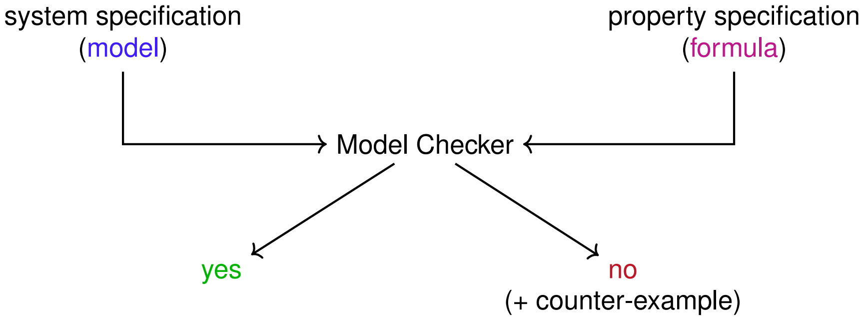
property specification
(formula)

→ Model Checker ←

Idea of Model Checking



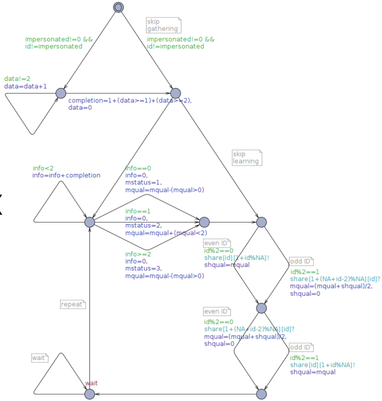
Idea of Model Checking



Model Specification: Example (SAI)

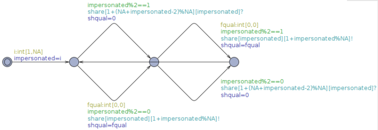
Honest AI

2 x

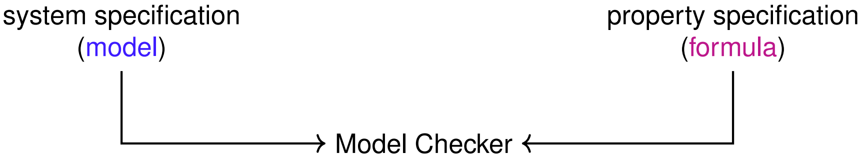


Impersonator

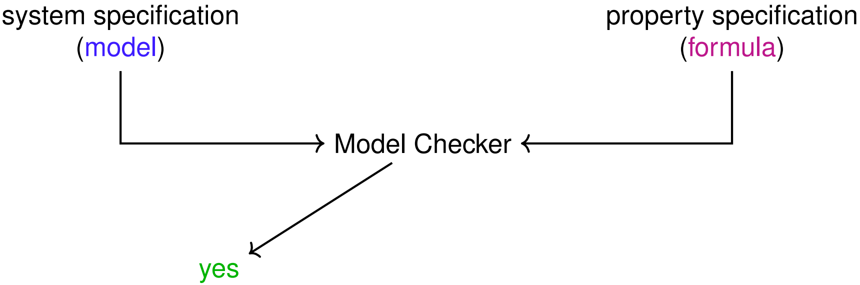
+ 1 x



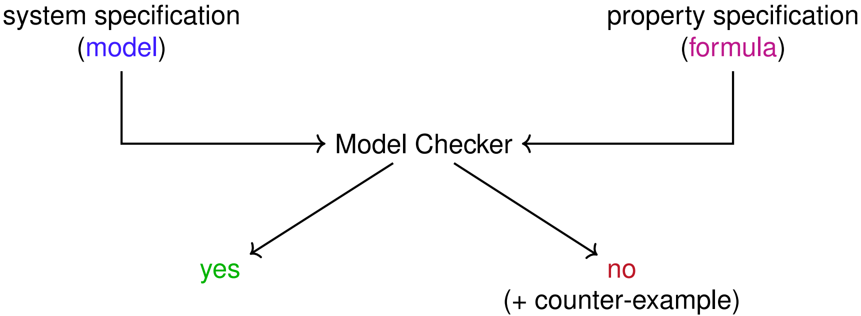
Model Checking in Practice



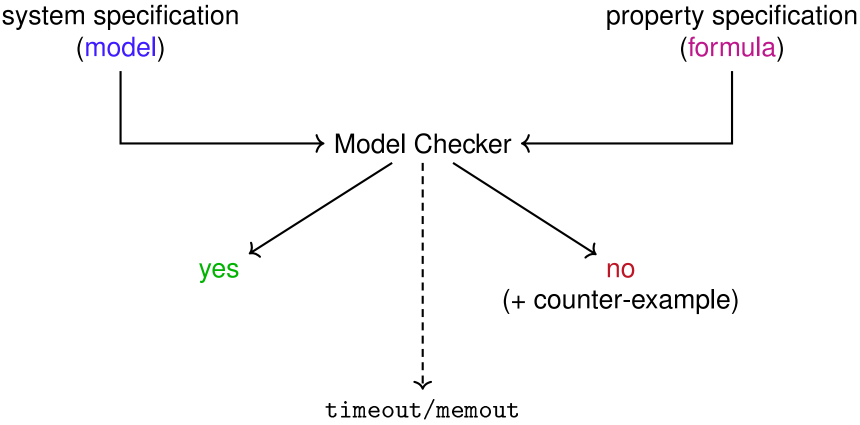
Model Checking in Practice



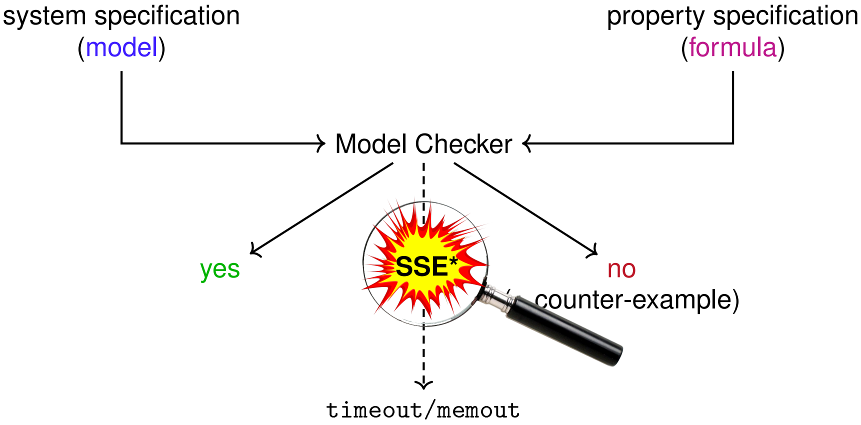
Model Checking in Practice



Model Checking in Practice



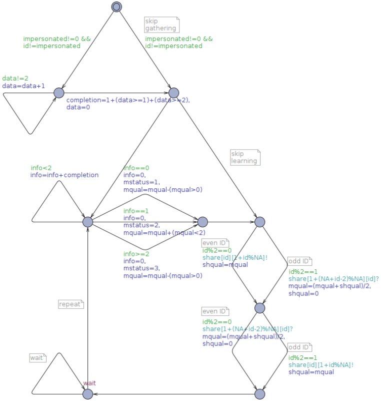
Model Checking in Practice



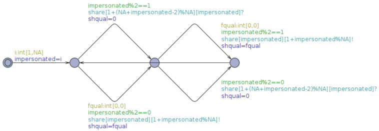
* : State Space Explosion

State Space Explosion: Example (SAI)

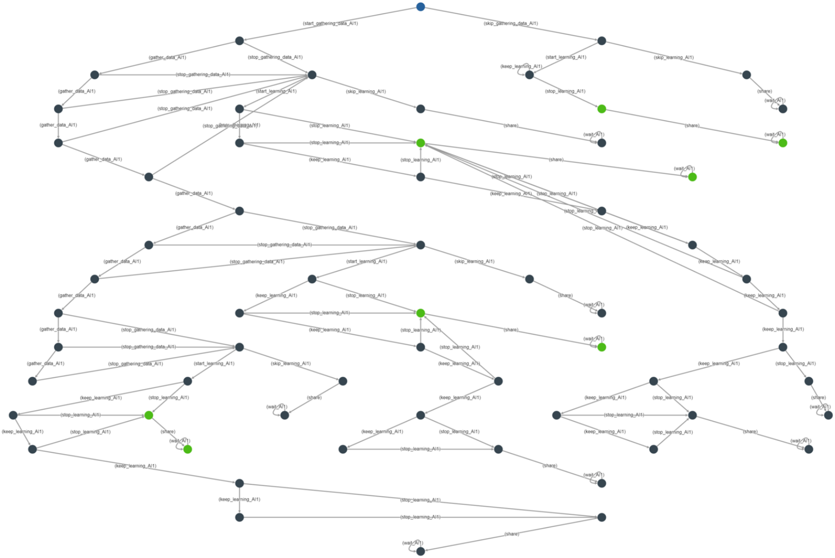
Honest AI



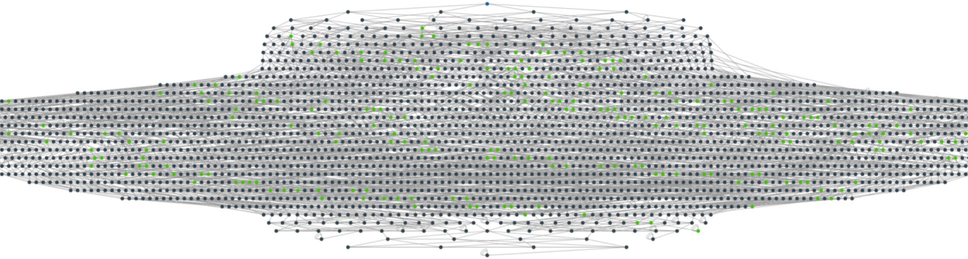
Impersonator



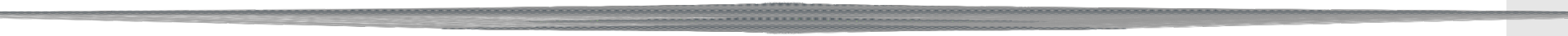
Model of SAI with One Honest Agent



Model of SAI with Two Honest Agents



Model of SAI with Two Honest Agents and One Impersonator



Not Easy Indeed...

- Exact verification of strategic abilities is hard
- Possible way out: **incomplete algorithms**

Not Easy Indeed...

- Exact verification of strategic abilities is hard
- Possible way out: **incomplete algorithms**

- Note: the main source of complexity is the size of the model!
- Possible way out: use smaller models $\rightsquigarrow$ **model reductions**

Not Easy Indeed...

- Exact verification of strategic abilities is hard
- Possible way out: **incomplete algorithms**

- Note: the main source of complexity is the size of the model!
- Possible way out: use smaller models $\rightsquigarrow$ **model reductions**

- Also, we will only look at memoryless strategies from now on

Part 3: Practical Model Checking

3.2 Approximate Model Checking

Approximate Verification of Strategic Ability

- Exact verification of strategic abilities is hard
- Idea: try to find formulae that **approximate the truth value of the given specification** (i.e., upper bound and lower bound)

Approximate Verification of Strategic Ability

- Exact verification of strategic abilities is hard
- Idea: try to find formulae that **approximate the truth value of the given specification** (i.e., upper bound and lower bound)
- ...and which are easier to compute

Approximate Verification of Strategic Ability

- Exact verification of strategic abilities is hard
- Idea: try to find formulae that **approximate the truth value of the given specification** (i.e., upper bound and lower bound)
- ...and which are easier to compute
- If lower bound = upper bound, we get the exact answer!

Approximate Verification of Strategic Ability

Approximation Semantics

$$LB(p) = p,$$
$$LB(\neg\phi) = \neg UB(\phi),$$
$$LB(\phi \land \psi) = LB(\phi) \land LB(\psi),$$
$$LB(\langle A \rangle \phi) = \langle A \rangle LB(\phi),$$
$$LB(\langle\!\langle A \rangle\!\rangle \Box \phi) = \nu Z.(C_A LB(\phi) \land \langle A \rangle^{\bullet} Z),$$
$$LB(\langle\!\langle A \rangle\!\rangle \psi \,\mathrm{U}\, \phi) = \mu Z.\big(E_A LB(\phi) \lor (C_A LB(\psi) \land \langle A \rangle^{\bullet} Z)\big).$$

$$UB(p) = p,$$
$$UB(\neg\phi) = \neg LB(\phi),$$
$$UB(\phi \land \psi) = UB(\phi) \land UB(\psi),$$
$$UB(\langle A \rangle \phi) = E_A \langle\!\langle A \rangle\!\rangle_{\mathrm{Ir}} \bigcirc UB(\phi),$$
$$UB(\langle\!\langle A \rangle\!\rangle \Box \phi) = E_A \langle\!\langle A \rangle\!\rangle_{\mathrm{Ir}} \Box\, UB(\phi),$$
$$UB(\langle\!\langle A \rangle\!\rangle \psi \,\mathrm{U}\, \phi) = E_A \langle\!\langle A \rangle\!\rangle_{\mathrm{Ir}}\, UB(\psi) \,\mathrm{U}\, UB(\phi).$$

Approximate Verification of Strategic Ability

Theorem (Jamroga, Knapik, Kurpiewski, & Mikulski 2019)

For every pointed model M and ATL formula φ:

$$M \models LB(\varphi) \implies M \models \varphi \implies M \models UB(\varphi).$$

Approximate Verification of Strategic Ability

Theorem (Jamroga, Knapik, Kurpiewski, & Mikulski 2019)

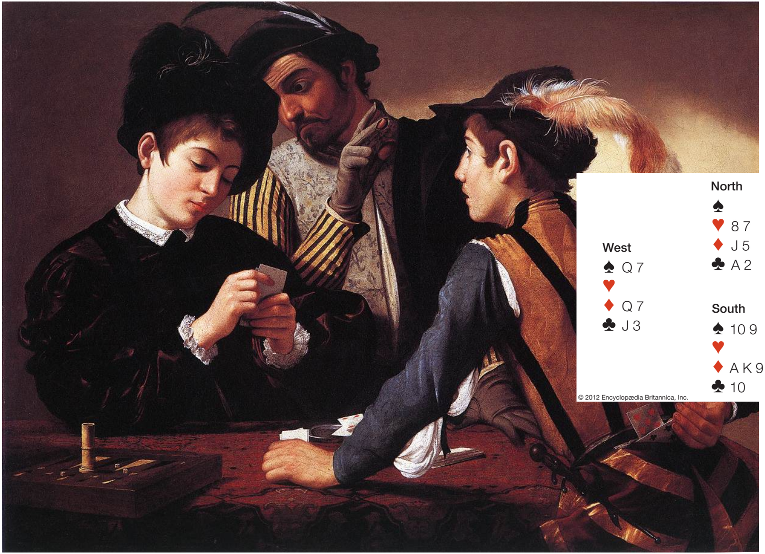
For every pointed model M and ATL formula φ:

$$M \models LB(\varphi) \implies M \models \varphi \implies M \models UB(\varphi).$$

- Benchmark: card play (similar mathematical structure to coercion in a voting protocol!)

Approximate Verification of Strategic Ability

Approximate Verification of Strategic Ability



North
♠
♥ 8 7
♦ J 5
♣ A 2

West
♠ Q 7
♥
♦ Q 7
♣ J 3

East
♠ J 6
♥
♦ 10 8 6
♣ 8

South
♠ 10 9
♥
♦ A K 9
♣ 10

Experimental Results

#cards	#states	Approximate verification				Exact verif.
		tgen	lower	upper	match	
4	11	<0.01	<0.01	<0.01	100%	0.12
8	346	0.01	<0.01	<0.01	100%	2.42 h*
12	12953	0.7	0.07	0.01	100%	timeout
16	617897	35.2	348.4	0.7	100%	timeout
$20^{\star}$	2443467	132.0	8815.7	4.2	100%	timeout

Formula: $\langle\langle \mathbf{S} \rangle\rangle \Diamond \mathsf{win}$

Time in seconds, unless explicitly indicated
timeout $\approx$ 45h

Experimental Results with Optimized Data Structures

#cards	#states	Approximate verification				Exact verif.
		tgen	lower	upper	match	
4	11	$<$0.01	$<$0.01	$<$0.01	100%	0.12
8	346	$<$0.01	$<$0.01	$<$0.01	100%	2.42 h*
12	12953	0.06	$<$0.01	$<$0.01	100%	timeout
16	617897	4.6	0.6	0.3	100%	timeout
$20^{\star}$	2443467	34.0	3.0	2.0	100%	timeout
20	1.5 e7	124.0	8.5	6.0	100%	timeout
$24^{\star}$	7 e7	3779.0	667.0	78.0	100%	timeout

Formula: $\langle\langle \mathbf{S} \rangle\rangle \Diamond \mathsf{win}$

Time in seconds, unless explicitly indicated
timeout $\approx$ 45h

Part 4: Model Reductions

Part 4: Model Reductions

4.1 Idea

Model Reductions

Factors of complexity:

- Size of the model $\rightsquigarrow$ "program complexity"
- Length of the formula $\rightsquigarrow$ "formula complexity"

Model Reductions

Factors of complexity:

- Size of the **model** $\rightsquigarrow$ **"program complexity"**
- Length of the formula $\rightsquigarrow$ "formula complexity"

Model Reductions

Factors of complexity:

- Size of the **model** ⇝ **“program complexity”**
- Length of the formula ⇝ “formula complexity”

Want to make the verification feasible? Use smaller models!

⇝ **Model reductions**

Model Reductions

Model Reductions

Part 4: Model Reductions

4.2 Partial Order Reduction

Partial Order Reduction

- **Partial order reduction (POR)**: a method of generating reduced models that preserve the formulae of logic $\mathcal{L}$
- For each infinite path, the reduced model contains at least one $\mathcal{L}$-equivalent path (but as few as possible!)

Partial Order Reduction

- **Partial order reduction (POR)**: a method of generating reduced models that preserve the formulae of logic $\mathcal{L}$
- For each infinite path, the reduced model contains at least one $\mathcal{L}$-equivalent path (but as few as possible!)
- Idea for $\mathrm{LTL}_{-\bigcirc}$: take only one arbitrary interleaving of independent actions

Partial Order Reduction for LTL_ $\bigcirc$

Algorithm DFS-POR

A stack represents the path $\pi = g_0 a_0 g_1 a_1 \cdots g_n$ currently being visited.
For g_n, the following three operations are executed in a loop:

1. Compute the set $en(g_n) \subseteq Act$ of enabled actions.
2. Select (heuristically) a subset $E(g_n) \subseteq en(g_n)$ of necessary actions.
3. For any action $a \in E(g_n)$, compute the successor state g' of g_n such that $g_n \xrightarrow{a} g'$, and add g' to the stack.
   Recursively proceed to explore the submodel originating at g'.
4. Remove g_n from the stack.

Partial Order Reduction for LTL_ $\bigcirc$

Conditions for selection of $E(g)$

C1 No action $a \in Act \setminus E(g)$ that is dependent on an action in $E(g)$ can be executed before an action in $E(g)$ is executed.

C2 On every cycle in the constructed state graph there is at least one node g for which $E(g) = en(g)$.

C3 Each action in $E(g)$ is invisible, i.e., it does not change $V(g)$.

Partial Order Reduction for LTL$_{-\bigcirc}$

Theorem (Peled 1993)

For every formula φ of LTL$_{-\bigcirc}$:

$$M \models \varphi \quad \text{iff} \quad DFS(M) \models \varphi.$$

Partial Order Reduction for LTL$_{-\bigcirc}$

Theorem (Peled 1993)

For every formula φ of LTL$_{-\bigcirc}$:

$$M \models \varphi \quad \text{iff} \quad DFS(M) \models \varphi.$$

What about **ATL**?

It would seem that a much stronger (and hence less useful) reduction is needed, as ATL is much more expressive than LTL...

Surprise!

Partial Order Reduction for Strategic Abilities

Theorem (Jamroga, Penczek, Sidoruk, Dembinski & Mazurkiewicz 2020)

For every formula φ of **ATL**$_{-\bigcirc}$ without nested strategic operators, interpreted over imperfect information strategies:

$$M \models \varphi \qquad \text{iff} \qquad DFS(M) \models \varphi.$$

Partial Order Reduction for Strategic Abilities

Theorem (Jamroga, Penczek, Sidoruk, Dembinski & Mazurkiewicz 2020)

For every formula φ of **ATL_$\bigcirc$** without nested strategic operators, interpreted over imperfect information strategies:

$$M \models \varphi \qquad \text{iff} \qquad DFS(M) \models \varphi.$$

Note also:

Theorem (Jamroga, Penczek, Sidoruk, Dembinski & Mazurkiewicz 2020)

The same does **not** hold for ATL_$\bigcirc$ interpreted over perfect information strategies.

Partial Order Reduction for Strategic Abilities

Theorem (Jamroga, Penczek, Sidoruk, Dembinski & Mazurkiewicz 2020)

For every formula φ of **ATL_ $\bigcirc$** without nested strategic operators, interpreted over imperfect information strategies:

$$M \models \varphi \quad \text{iff} \quad DFS(M) \models \varphi.$$

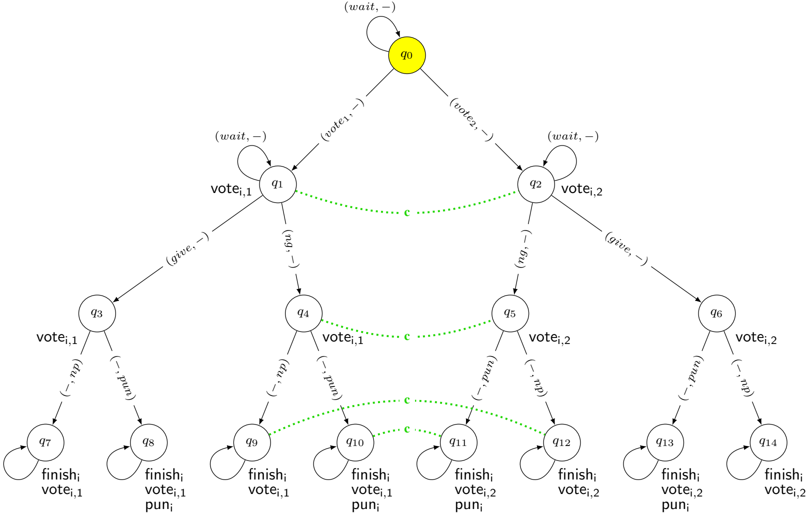
Note also:

Theorem (Jamroga, Penczek, Sidoruk, Dembinski & Mazurkiewicz 2020)

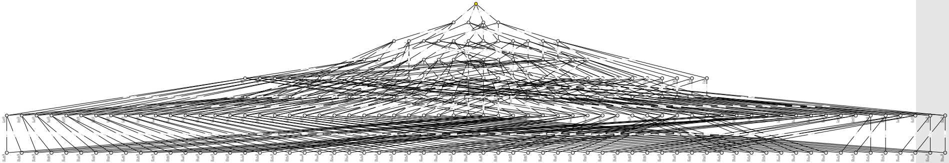
The same does **not** hold for ATL_ $\bigcirc$ interpreted over perfect information strategies.

How good are the reductions in practice?

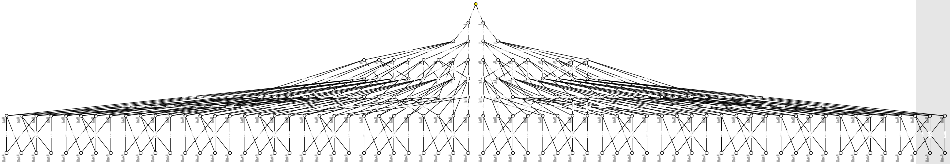
Reduction for Voting and Coercion



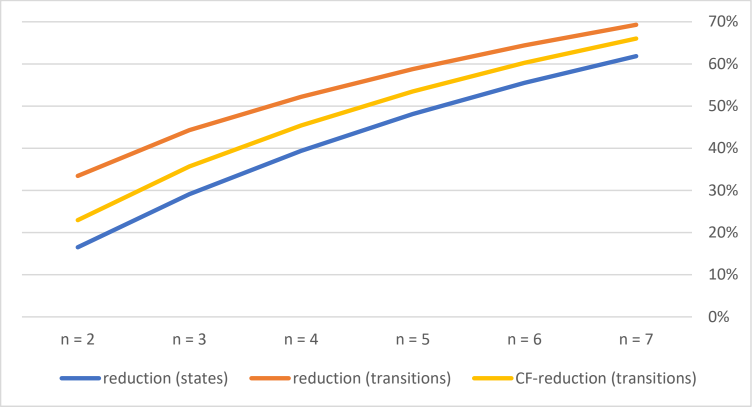
Reduction for Voting and Coercion



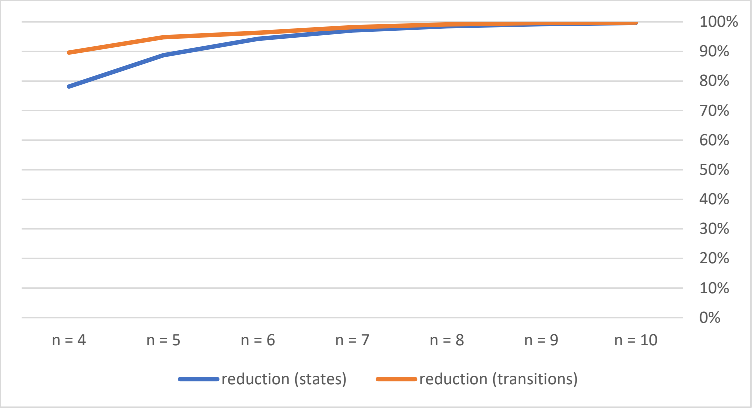
Reduction for Voting and Coercion



Experimental Results: Asynchronous Simple Voting



Experimental Results: Trains, Gate, and Controller



Partial Order Reduction: Summary

- For some strategic abilities, we get an effective automated model reduction off the shelf for free
- There is **free lunch** out there! 

Partial Order Reduction: Summary

- For some strategic abilities, we get an effective automated model reduction off the shelf for free
- There is **free lunch** out there! 
- The trick is... someone must have already paid for the lunch
- What remained was to **prove that we are eligible** to get it (nontrivial!)

Part 5: STrategic Verifier (STV)

Model Checking Tools

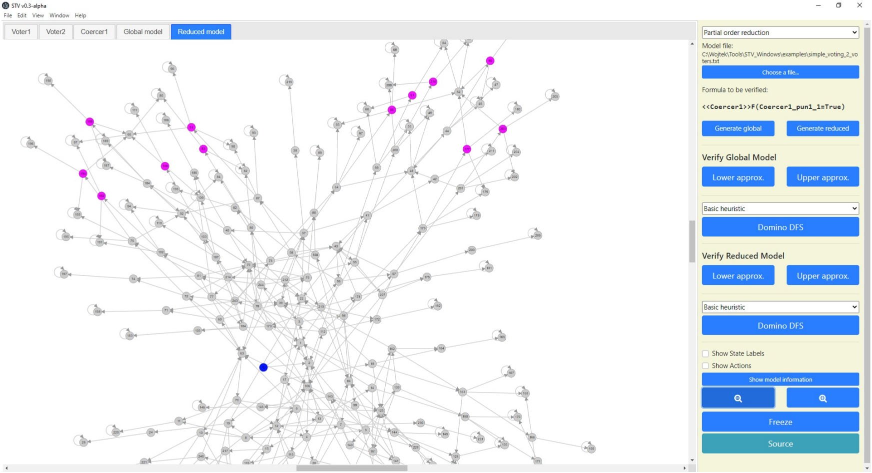
Many model checking tools out there:

- Temporal and timed properties of systems: SPIN, nuSMV, LTSmin, IMITATOR, Uppaal, ...
- Temporal properties of stochastic MAS: PRISM, Storm
- Temporal-epistemic and strategic properties of MAS: MCMAS (and extensions)
- Strategic properties for imperfect information agents: **STV**

STrategic Verifier (STV)

- Experimental model checker developed at ICS PAS
- Implemented techniques:
  1. standard fixpoint algorithm for perfect info games
  2. brute force DFS
  3. parallelized DFS
  4. domination-based strategy search
  5. fixpoint approximation
  6. assume-guarantee reasoning
  7. on the fly model generation
  8. bisimulation checking for bisimulation-based reduction
  9. partial-order reduction
- Includes lightweight GUI and web-based interface
- Includes benchmarks and examples
- Case studies: electronic voting, postal voting, logistic robots

STrategic Verifier (STV)



Links

STV: http://stv.cs-htiew.com/

UPPAAL: https://uppaal.org/

Models: https://tiny.pl/3dp0yzp1


Thank you