



Formal Verification of Multi-Agent Systems Why, What, and Especially: How?

Catalin Dima and Wojtek Jamroga

ESSLLI 2026 / 3–7 August 2026 @ Prague, Czechia



Course website:

<https://home.ipipan.waw.pl/w.jamroga/courses/Verification2026ESSLLI/>



Basic Reading

- 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>
- Yoav Shoham and Kevin Leyton-Brown (2009), **Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations**. Cambridge University Press.
- Christel Baier, Joost-Pieter Katoen (2008), **Principles of Model Checking**. MIT Press.
- Edmund M. Clarke, Orna Grumberg and Doron A. Peled (1999), **Model Checking**. MIT Press.



Complementary Courses

Related courses on at ESSLLI:

- François Schwarzentruher, [Verifying graph neural networks](#) (advanced, week 1)
- Sujata Ghosh and Dazhu Li, [Reasoning in graph games: A modal logic study](#) (advanced, week 1)
- Valentin Goranko and Dmitry Shkatov, [Tableaux-based decision methods for modal, temporal, and epistemic logics](#) (introductory, week 1)
- Karel Chvalovský and Mikoláš Janota, [Introduction to SAT and SMT solving](#) (introductory, week 2)
- Valentin Goranko, [Coordination games, rationality, and logic](#) (foundational, week 2)

Recommended!



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
- [Wojciech Penczek](#), Polish Academy of Sciences

All the mistakes are, of course, ours.



Research supported by:

 Fonds National de la
Recherche Luxembourg

 **NCBR**
National Centre for Research
and Development

 **chist-era**

 **NATIONAL SCIENCE CENTRE**
POLAND

 **anr**
agence nationale
de la recherche
AU SERVICE DE LA SCIENCE

 **RFSI**
réseau francilien
en sciences informatiques

 **Île de France**



Outline

- 1 Introduction to Formal Specification for MAS
- 2 Specification and Verification of Strategic Abilities
- 3 Practical Model Checking
- 4 Strategy Logic and Rational Verification
- 5 Model Reductions



Part 1: Formal Specification for MAS

- 1.1 Motivation
- 1.2 Modeling MAS
- 1.3 Temporal Properties
- 1.4 Branching Time
- 1.5 Adding Knowledge Operators
- 1.6 Knowledge and Time
- 1.7 Model Checking
- 1.8 Tool Presentation: Uppaal



Part 1: Formal Specification 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: Formal Specification for MAS

1.2 Modeling MAS

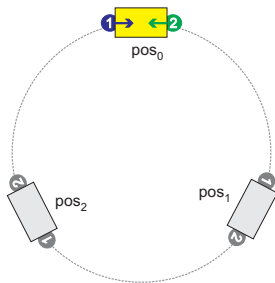


Modeling Multi-Agent Systems

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

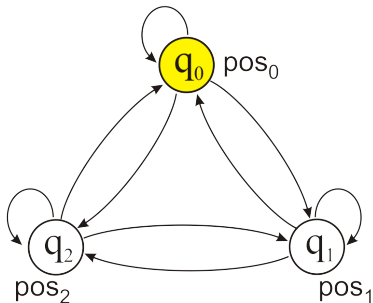
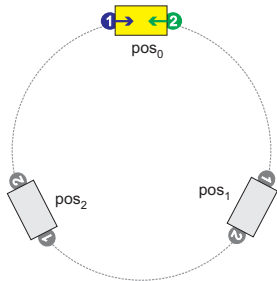


Example: Robots and Carriage



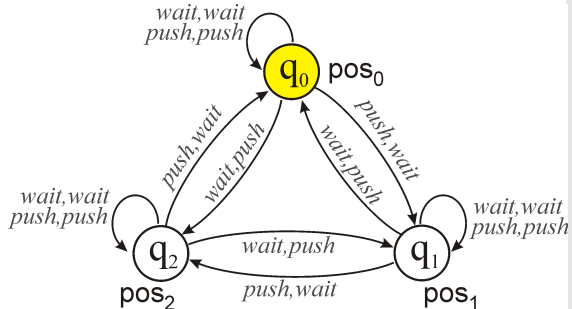
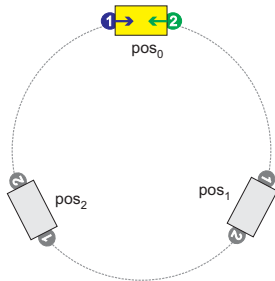


Example: Robots and Carriage



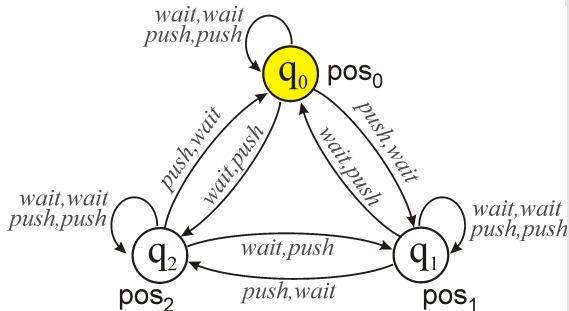
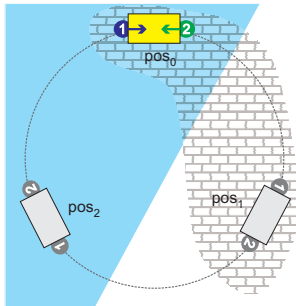


Example: Robots and Carriage



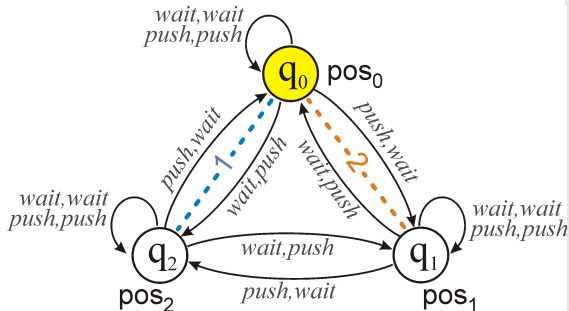
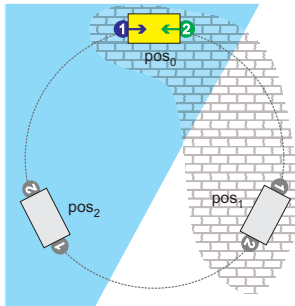


Example: Robots and Carriage





Example: Robots and Carriage



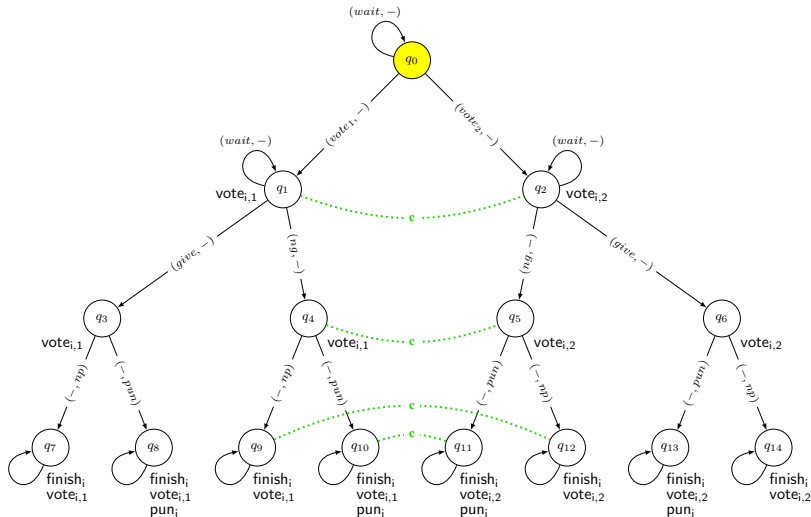


Example: Voting and Coercion



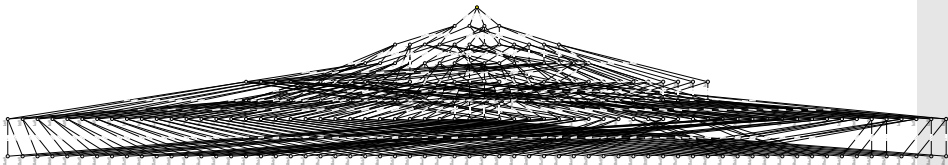


Example: Voting and Coercion





Example: Voting and Coercion





Part 1: Formal Specification for MAS

1.3 Temporal Properties



Motivating Example: Voting

Properties to express

- The voter **will eventually** cast a vote
- The voting system **will not** reveal how a particular voter voted
- 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
$\square \varphi$	φ is true in all future moments
$\diamond \varphi$	φ is true in some future moment
$\varphi \mathbf{U} \psi$	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:

$\Box \neg \text{bankrupt}$



Specification Templates: Safety Properties

Safety / maintenance:

“something bad will never happen”

“something good will always hold”

Example:

$\Box \neg \text{bankrupt}$

Usually: $\Box \dots$



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:

◇rich



Specification Templates: Reachability Properties

Reachability / achievement / liveness:

“something good will eventually happen”

Example:

$\Diamond \text{rich}$

$\Diamond \Box \text{rich}$



Specification Templates: Reachability Properties

Reachability / achievement / liveness:

“something good will eventually happen”

Example:

$\Diamond$ rich

$\Diamond \Box$ rich

Usually: $\Diamond \dots$



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:

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



Specification Templates: Fairness Properties

Fairness / service:

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

Example:

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

Usually: $\Box \Diamond \dots$



Formal Semantics of Linear Time Logic (LTL)

Definition 1.1 (Semantics of LTL)

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

$M, \lambda \models \neg\varphi$ iff not $M, \lambda \models \varphi$;

$M, \lambda \models \varphi \wedge \psi$ iff $M, \lambda \models \varphi$ and $M, \lambda \models \psi$.

$M, \lambda \models \bigcirc\varphi$ iff $M, \lambda[1..\infty] \models \varphi$;

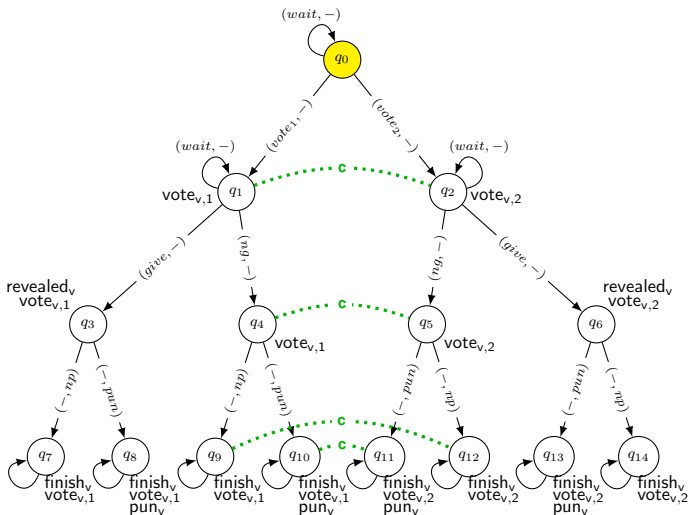
$M, \lambda \models \Diamond\varphi$ iff $M, \lambda[i..\infty] \models \varphi$ for some $i \geq 0$;

$M, \lambda \models \Box\varphi$ iff $M, \lambda[i..\infty] \models \varphi$ for all $i \geq 0$;

$M, \lambda \models \varphi \cup \psi$ iff $M, \lambda[i..\infty] \models \psi$ for some $i \geq 0$, and $M, \lambda[j..\infty] \models \varphi$ for all $0 \leq j < i$.



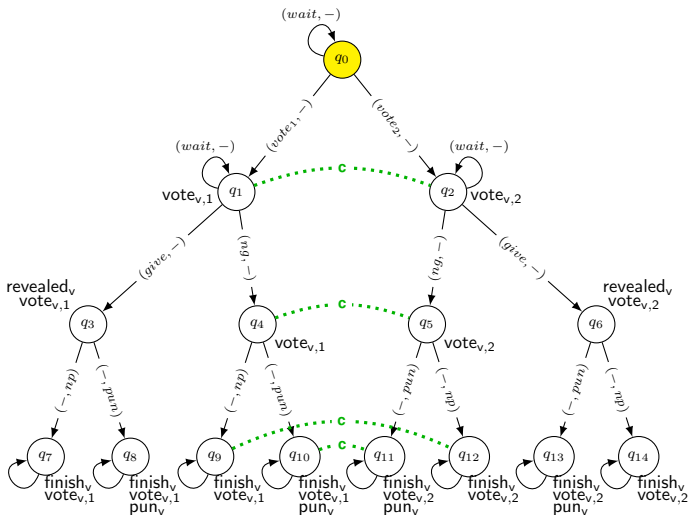
Example: Voting and Coercion



$\Diamond (vote_{v,1} \vee vote_{v,2})$

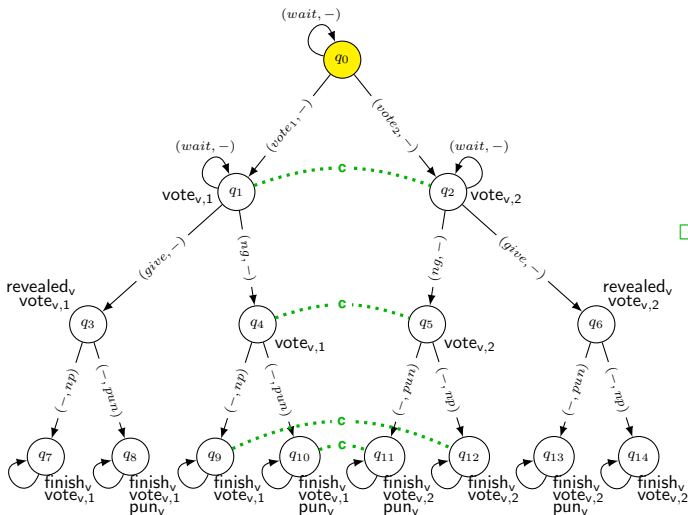


Example: Voting and Coercion

 $\Diamond (vote_{v,1} \vee vote_{v,2})$ **No!**



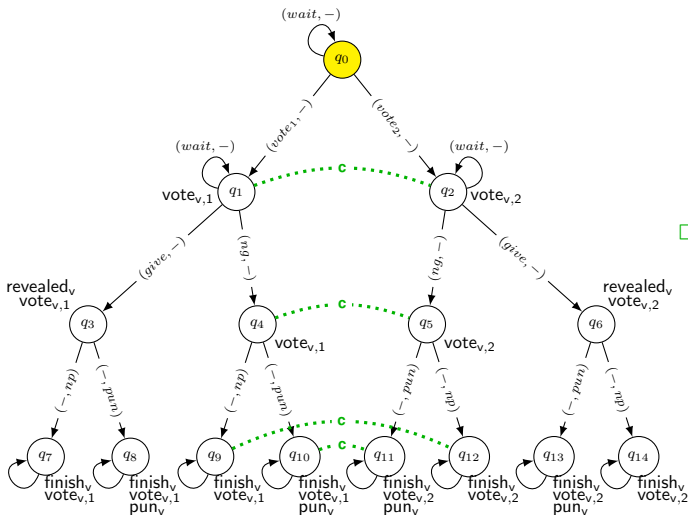
Example: Voting and Coercion



$\Box (vote_{v,1} \rightarrow \neg \Diamond vote_{v,2})$



Example: Voting and Coercion


$$\Box (vote_{v,1} \rightarrow \neg \Diamond vote_{v,2})$$
Yes!



Motivating Example: Voting

- The voter will eventually cast a vote



Motivating Example: Voting

- The voter will eventually cast a vote

$$\diamond \bigvee_{j \in \text{Cand}} \text{vote}_{v,j}$$



Motivating Example: Voting

- The voter will eventually cast a vote

$$\diamond \bigvee_{j \in \text{Cand}} \text{vote}_{v,j}$$

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



Motivating Example: Voting

- The voter will eventually cast a vote

$$\diamond \bigvee_{j \in \text{Cand}} \text{vote}_{v,j}$$

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

$$\square \neg \text{revealed}_v$$



Motivating Example: Voting

- The voter will eventually cast a vote

$$\diamond \bigvee_{j \in \text{Cand}} \text{vote}_{v,j}$$

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

$$\square \neg \text{revealed}_v$$

- More precise specification: Nobody will ever know for whom the voter has voted



Motivating Example: Voting

- The voter will eventually cast a vote

$$\diamond \bigvee_{j \in \text{Cand}} \text{vote}_{v,j}$$

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

$$\square \neg \text{revealed}_v$$

- More precise specification: Nobody will ever know for whom the voter has voted

Cannot be expressed in LTL



Motivating Example: Voting

- The voter will eventually cast a vote

$$\diamond \bigvee_{j \in \text{Cand}} \text{vote}_{v,j}$$

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

$$\square \neg \text{revealed}_v$$

- More precise specification: Nobody will ever know for whom the voter has voted

Cannot be expressed in LTL

- The voter can vote, and can refrain from voting



Motivating Example: Voting

- The voter will eventually cast a vote

$$\diamond \bigvee_{j \in \text{Cand}} \text{vote}_{v,j}$$

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

$$\square \neg \text{revealed}_v$$

- More precise specification: Nobody will ever know for whom the voter has voted

Cannot be expressed in LTL

- The voter can vote, and can refrain from voting

Cannot be expressed in LTL



What's Missing

What's missing in LTL?



What's Missing

What's missing in LTL?

- The ability to distinguish between **necessary and possible courses of action**
- What **will** happen = what **must** happen
- Sometimes we want to express that something **might** happen



Part 1: Formal Specification for MAS

1.4 Branching Time



Branching Time: CTL

- **CTL: Computation Tree Logic**
- Reasoning about **all** possible computations of a system



Branching Time: CTL

- **CTL: Computation Tree Logic**
- Reasoning about **all** possible computations of a system
- **Path quantifiers**: A (for all paths), E (there is a path)
- **Temporal operators**: $\bigcirc$ (next), $\Diamond$ (sometime), $\Box$ (always), U (until)
- Two kinds of formulae: **state formulae** vs. **path formulae**



Branching Time: CTL

- **CTL: Computation Tree Logic**
- Reasoning about **all** possible computations of a system
- **Path quantifiers**: A (for all paths), E (there is a path)
- **Temporal operators**: $\bigcirc$ (next), $\Diamond$ (sometime), $\Box$ (always), U (until)
- Two kinds of formulae: **state formulae** vs. **path formulae**
- “Vanilla” CTL: every temporal operator must be immediately preceded by exactly one path quantifier
- CTL*: no syntactic restrictions



Branching Time: CTL

- **CTL: Computation Tree Logic**
- Reasoning about **all** possible computations of a system
- **Path quantifiers**: **A** (for all paths), **E** (there is a path)
- **Temporal operators**: **O** (next), **◇** (sometime), **□** (always), **U** (until)
- Two kinds of formulae: **state formulae** vs. **path formulae**
- “Vanilla” CTL: every temporal operator must be immediately preceded by exactly one path quantifier
- CTL*: no syntactic restrictions
- Reasoning in “vanilla” CTL is easier to automatize



Branching Time: CTL

Definition 1.2 (Semantics of CTL*: state formulae)

- $M, q \models E\gamma$ iff there is a path λ , starting from q , such that $M, \lambda \models \gamma$;
 $M, q \models A\gamma$ iff for all paths λ , starting from q , we have $M, \lambda \models \gamma$.



Branching Time: CTL

Definition 1.2 (Semantics of CTL*: state formulae)

$M, q \models E\gamma$ iff there is a path λ , starting from q , such that $M, \lambda \models \gamma$;
 $M, q \models A\gamma$ iff for all paths λ , starting from q , we have $M, \lambda \models \gamma$.

Definition 1.3 (Semantics of CTL*: path formulae)

Like in LTL!



Branching Time: CTL

Definition 1.2 (Semantics of CTL*: state formulae)

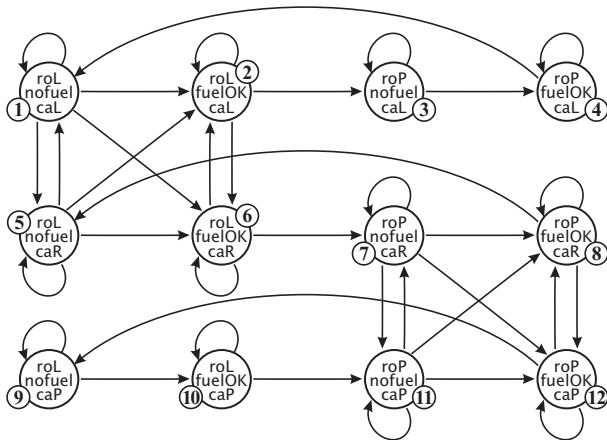
- $M, q \models E\gamma$ iff there is a path λ , starting from q , such that $M, \lambda \models \gamma$;
 $M, q \models A\gamma$ iff for all paths λ , starting from q , we have $M, \lambda \models \gamma$.

Definition 1.3 (Semantics of CTL*: path formulae)

- $M, \lambda \models \varphi$ iff $M, \lambda[0] \models \varphi$, for a state formula φ ;
 $M, \lambda \models \bigcirc\varphi$ 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 $i \geq 0$;
 $M, \lambda \models \varphi \cup \psi$ iff $M, \lambda[i..\infty] \models \psi$ for some $i \geq 0$, and $M, \lambda[j..\infty] \models \varphi$ for all $0 \leq j < i$.

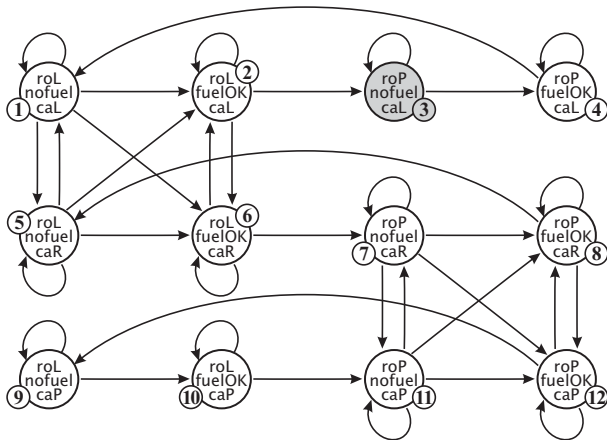


Example: Rocket and Cargo

 $E \diamond \text{caP}$

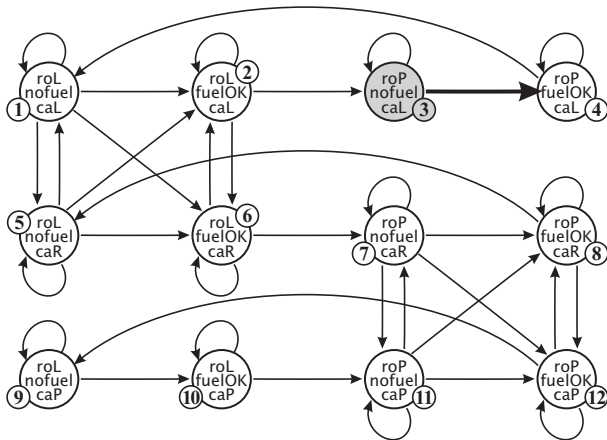


Example: Rocket and Cargo

 $E \diamond \text{caP}$

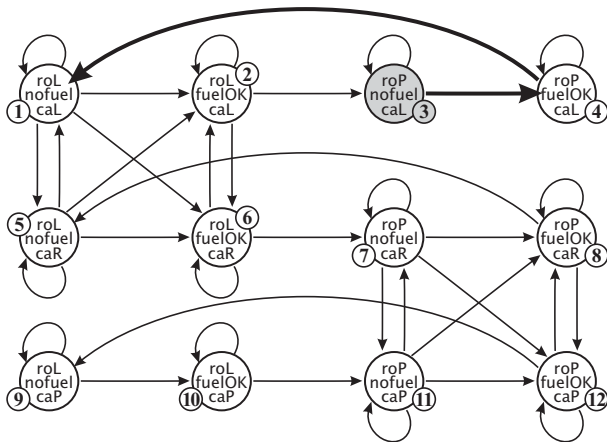


Example: Rocket and Cargo

 $E \diamond caP$

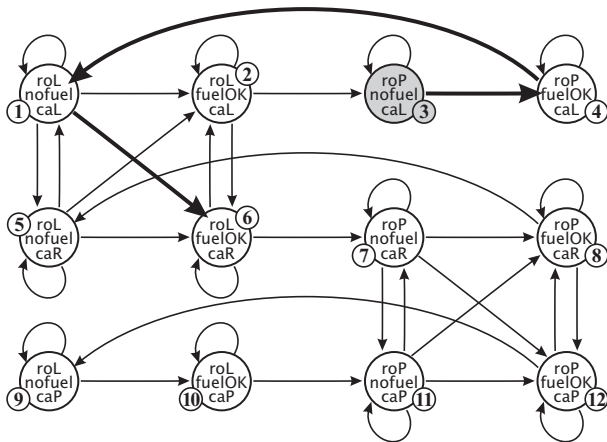


Example: Rocket and Cargo

 $E \diamond \text{caP}$

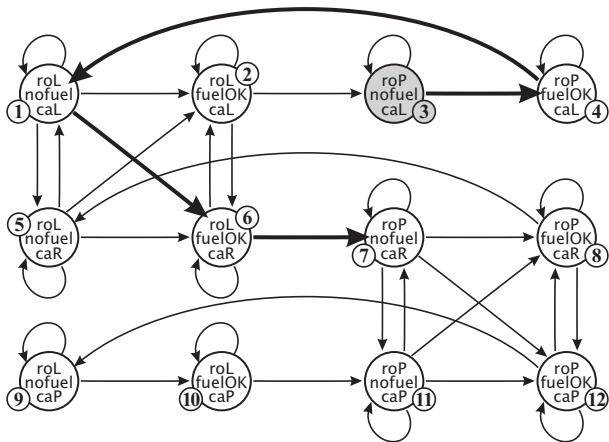


Example: Rocket and Cargo

 $E \diamond \text{caP}$

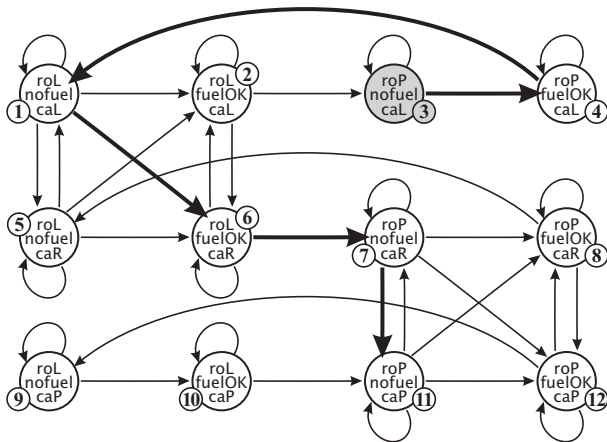


Example: Rocket and Cargo

 $E \diamond \text{caP}$

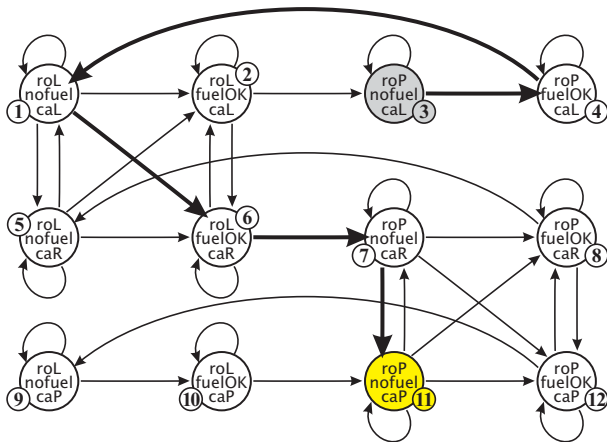


Example: Rocket and Cargo

 $E \diamond caP$

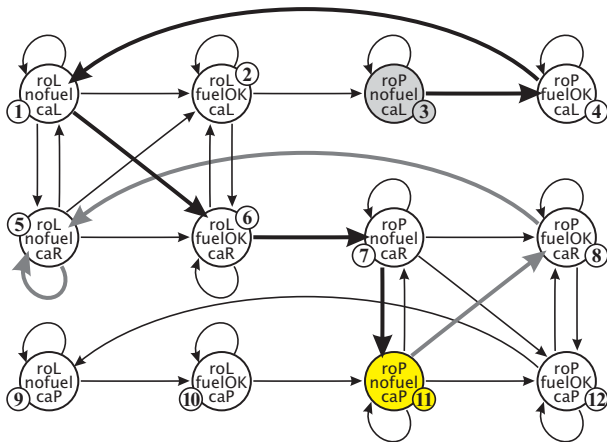


Example: Rocket and Cargo

 $E \diamond caP$

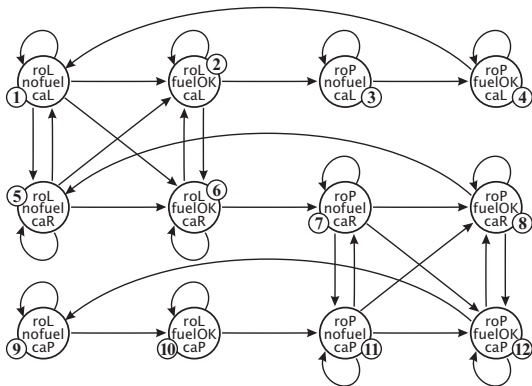


Example: Rocket and Cargo

 $E \diamond caP$

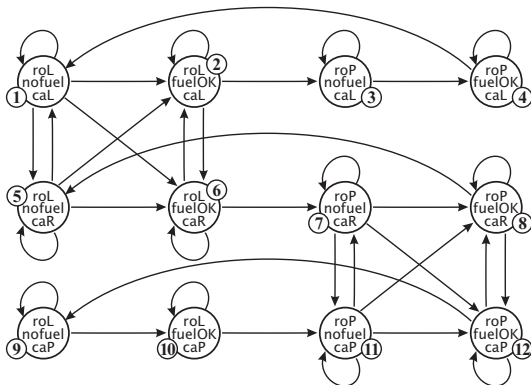


Rocket and Cargo: More Properties





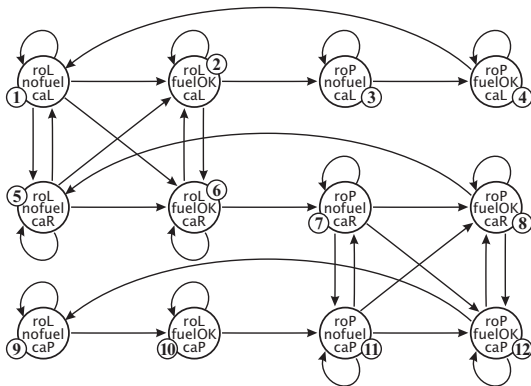
Rocket and Cargo: More Properties



$roL \rightarrow E \Diamond roP$

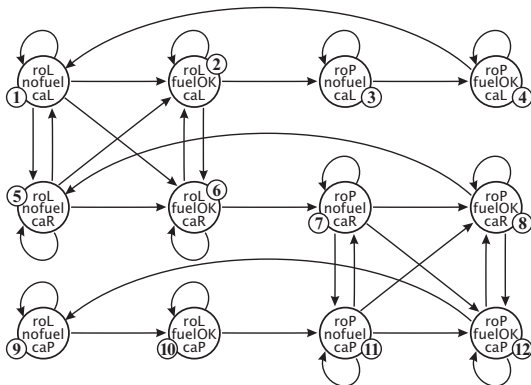


Rocket and Cargo: More Properties


$$\text{roL} \rightarrow E\Diamond \text{roP}$$
$$A\Box(\text{roL} \vee \text{roP})$$



Rocket and Cargo: More Properties


$$\text{roL} \rightarrow E\Diamond \text{roP}$$
$$A\Box(\text{roL} \vee \text{roP})$$
$$\text{roL} \rightarrow A\bigcirc(\text{roP} \rightarrow \text{nofuel})$$



Motivating Example: Voting

- The voter can vote, and can refrain from voting



Motivating Example: Voting

- The voter can vote, and can refrain from voting $\leadsto$ ambiguous...



Motivating Example: Voting

- The voter can vote, and can refrain from voting $\leadsto$ ambiguous...

Interpretation 1: The voter **might** vote, and **might** refrain from voting

$$E\Diamond \text{vote}_v \wedge E\Box \neg \text{vote}_v$$



Motivating Example: Voting

- The voter can vote, and can refrain from voting $\leadsto$ ambiguous...

Interpretation 1: The voter **might** vote, and **might** refrain from voting

$$E\Diamond \text{vote}_v \wedge E\Box \neg \text{vote}_v$$

More refined specification:

$$E\Diamond (\bigvee_{j \in \text{Cand}} \text{vote}_{v,j}) \wedge E\Box (\bigwedge_{j \in \text{Cand}} \neg \text{vote}_{v,j})$$



Motivating Example: Voting

- The voter can vote, and can refrain from voting $\leadsto$ ambiguous...

Interpretation 1: The voter **might** vote, and **might** refrain from voting

$$E\Diamond \text{vote}_v \wedge E\Box \neg \text{vote}_v$$

More refined specification:

$$E\Diamond (\bigvee_{j \in \text{Cand}} \text{vote}_{v,j}) \wedge E\Box (\bigwedge_{j \in \text{Cand}} \neg \text{vote}_{v,j})$$

Interpretation 2: She **is able** to vote, and **able** to refrain from voting



Motivating Example: Voting

- The voter can vote, and can refrain from voting $\leadsto$ **ambiguous...**

Interpretation 1: The voter **might** vote, and **might** refrain from voting

$$E\Diamond \text{vote}_v \wedge E\Box \neg \text{vote}_v$$

More refined specification:

$$E\Diamond (\bigvee_{j \in \text{Cand}} \text{vote}_{v,j}) \wedge E\Box (\bigwedge_{j \in \text{Cand}} \neg \text{vote}_{v,j})$$

Interpretation 2: She **is able** to vote, and **able** to refrain from voting

Cannot be expressed in CTL*



Motivating Example: Voting

- The voter can vote, and can refrain from voting $\leadsto$ **ambiguous...**

Interpretation 1: The voter **might** vote, and **might** refrain from voting

$$E\Diamond \text{vote}_v \wedge E\Box \neg \text{vote}_v$$

More refined specification:

$$E\Diamond (\bigvee_{j \in \text{Cand}} \text{vote}_{v,j}) \wedge E\Box (\bigwedge_{j \in \text{Cand}} \neg \text{vote}_{v,j})$$

Interpretation 2: She **is able** to vote, and **able** to refrain from voting

Cannot be expressed in CTL*

- Nobody will ever know for whom the voter has voted



Motivating Example: Voting

- The voter can vote, and can refrain from voting $\leadsto$ **ambiguous...**

Interpretation 1: The voter **might** vote, and **might** refrain from voting

$$E\Diamond \text{vote}_v \wedge E\Box \neg \text{vote}_v$$

More refined specification:

$$E\Diamond (\bigvee_{j \in \text{Cand}} \text{vote}_{v,j}) \wedge E\Box (\bigwedge_{j \in \text{Cand}} \neg \text{vote}_{v,j})$$

Interpretation 2: She **is able** to vote, and **able** to refrain from voting

Cannot be expressed in CTL*

- Nobody will ever know for whom the voter has voted

Cannot be expressed in CTL*



Part 1: Formal Specification for MAS

1.5 Adding Knowledge Operators



Motivating Example: Voting

Properties to express

- The coercer does not know how a particular voter has voted (so he doesn't have an effective threat)
- The voter knows that the coercer does not know (so that she doesn't have to worry about blackmail)
- The coercer knows that the voter knows that the coercer does not know (hence he knows that blackmail doesn't make sense)
- ...and so on, ad infinitum



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 $q \sim_i q'$



Collective Knowledge

A group of agents A can **know that** φ in several different epistemic modes:



Collective Knowledge

A group of agents A can **know that** φ in several different epistemic modes:

- $E_A\varphi$: **everybody** in A **knows** that φ (or: A have **mutual knowledge** that φ)



Collective Knowledge

A group of agents A can **know that** φ in several different epistemic modes:

- $E_A\varphi$: **everybody** in A **knows** that φ (or: A have **mutual knowledge** that φ)
- $C_A\varphi$: it is a **common knowledge** among A that φ



Collective Knowledge

A group of agents A can **know that** φ in several different epistemic modes:

- $E_A\varphi$: **everybody** in A **knows** that φ (or: A have **mutual knowledge** that φ)
- $C_A\varphi$: it is a **common knowledge** among A that φ
- $D_A\varphi$: A have **distributed knowledge** that φ



Collective Knowledge: Semantics

- $M, q \models E_A \varphi$ iff $M, q' \models \varphi$ for every q' such that $q \sim_A^E q'$, where $\sim_A^E = \bigcup_{i \in A} \sim_i$



Collective Knowledge: Semantics

- $M, q \models E_A \varphi$ iff $M, q' \models \varphi$ for every q' such that $q \sim_A^E q'$, where $\sim_A^E = \bigcup_{i \in A} \sim_i$
- $M, q \models C_A \varphi$ iff $M, q' \models \varphi$ for every q' such that $q \sim_A^C q'$, where $\sim_A^C$ is the transitive closure of $\sim_A^E$

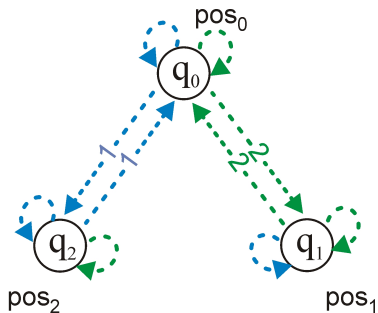


Collective Knowledge: Semantics

- $M, q \models E_A \varphi$ iff $M, q' \models \varphi$ for every q' such that $q \sim_A^E q'$, where $\sim_A^E = \bigcup_{i \in A} \sim_i$
- $M, q \models C_A \varphi$ iff $M, q' \models \varphi$ for every q' such that $q \sim_A^C q'$, where $\sim_A^C$ is the transitive closure of $\sim_A^E$
- $M, q \models D_A \varphi$ iff $M, q' \models \varphi$ for every q' such that $q \sim_A^D q'$, where $\sim_A^D = \bigcap_{i \in A} \sim_i$

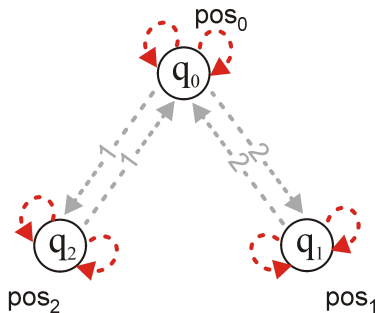


Example: Distributed Knowledge



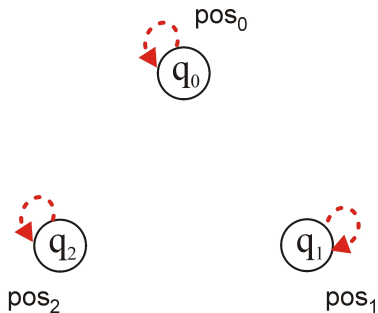


Example: Distributed Knowledge



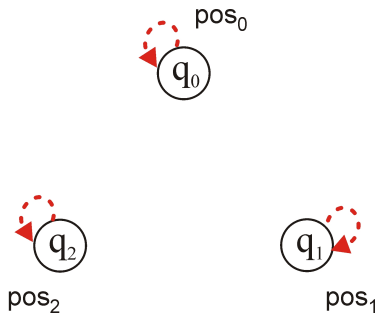


Example: Distributed Knowledge





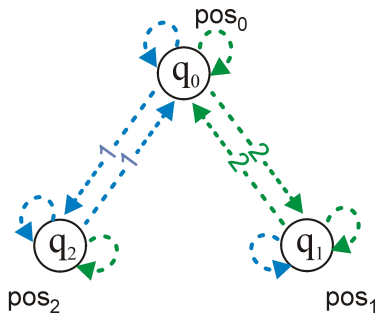
Example: Distributed Knowledge



$$pos_2 \rightarrow D_{\{1,2\}} pos_2$$

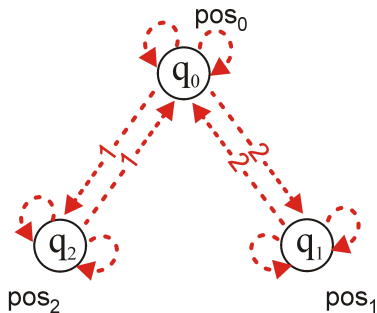


Example: Mutual Knowledge (“Everybody Knows”)



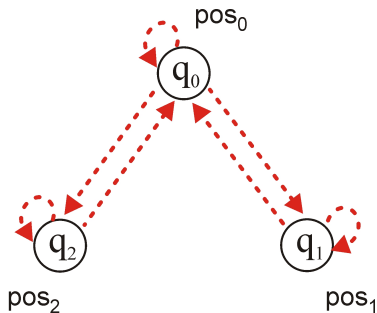


Example: Mutual Knowledge (“Everybody Knows”)



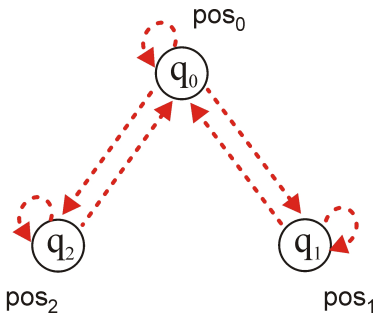


Example: Mutual Knowledge (“Everybody Knows”)





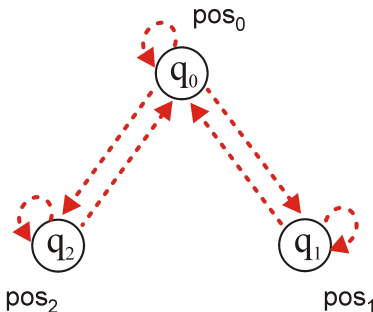
Example: Mutual Knowledge (“Everybody Knows”)



$$pos_2 \rightarrow \neg E_{\{1,2\}} pos_2$$



Example: Mutual Knowledge (“Everybody Knows”)

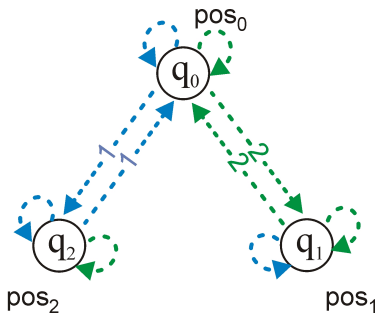


$$pos_2 \rightarrow \neg E_{\{1,2\}} pos_2$$

$$pos_2 \rightarrow E_{\{1,2\}} \neg pos_1$$

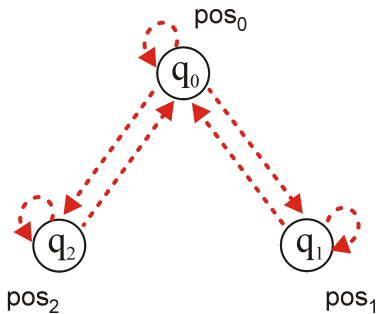


Example: Common Knowledge



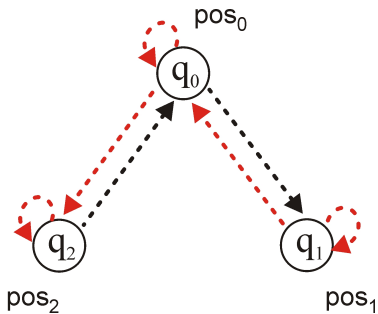


Example: Common Knowledge



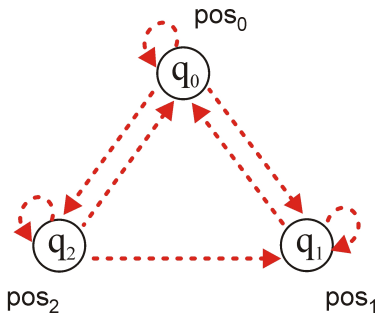


Example: Common Knowledge



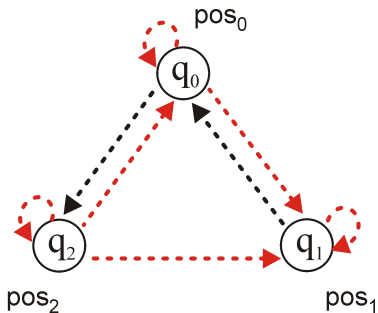


Example: Common Knowledge



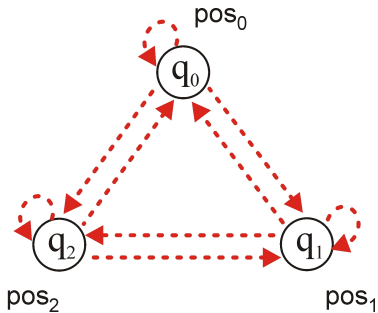


Example: Common Knowledge



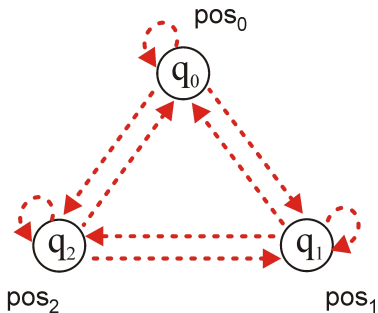


Example: Common Knowledge





Example: Common Knowledge



$$\begin{aligned} pos_2 &\rightarrow \neg C_{\{1,2\}} pos_2 \\ pos_2 &\rightarrow \neg C_{\{1,2\}} \neg pos_1 \end{aligned}$$



Motivating Example: Voting

- The coercer does not know how a particular voter has voted (so he doesn't have an effective threat)



Motivating Example: Voting

- The coercer does not know how a particular voter has voted (so he doesn't have an effective threat)

$$\bigwedge_{j \in \text{Cand}} \neg K_c \text{voted}_{v,j}$$



Motivating Example: Voting

- The coercer does not know how a particular voter has voted (so he doesn't have an effective threat)

$$\bigwedge_{j \in \text{Cand}} \neg K_c \text{voted}_{v,j}$$

- The voter knows that the coercer does not know (so that she doesn't have to worry about blackmail)



Motivating Example: Voting

- The coercer does not know how a particular voter has voted (so he doesn't have an effective threat)

$$\bigwedge_{j \in \text{Cand}} \neg K_c \text{voted}_{v,j}$$

- The voter knows that the coercer does not know (so that she doesn't have to worry about blackmail)

$$K_v \left(\bigwedge_{j \in \text{Cand}} \neg K_c \text{voted}_{v,j} \right)$$



Motivating Example: Voting

- The coercer does not know how a particular voter has voted (so he doesn't have an effective threat)

$$\bigwedge_{j \in \text{Cand}} \neg K_c \text{voted}_{v,j}$$

- The voter knows that the coercer does not know (so that she doesn't have to worry about blackmail)

$$K_v \left(\bigwedge_{j \in \text{Cand}} \neg K_c \text{voted}_{v,j} \right)$$

- The coercer knows that the voter knows that the coercer does not know (hence he knows that blackmail doesn't make sense)



Motivating Example: Voting

- The coercer does not know how a particular voter has voted (so he doesn't have an effective threat)

$$\bigwedge_{j \in \text{Cand}} \neg K_c \text{voted}_{v,j}$$

- The voter knows that the coercer does not know (so that she doesn't have to worry about blackmail)

$$K_v \left(\bigwedge_{j \in \text{Cand}} \neg K_c \text{voted}_{v,j} \right)$$

- The coercer knows that the voter knows that the coercer does not know (hence he knows that blackmail doesn't make sense)

$$K_c K_v \left(\bigwedge_{j \in \text{Cand}} \neg K_c \text{voted}_{v,j} \right)$$



Motivating Example: Voting

- The coercer does not know how a particular voter has voted (so he doesn't have an effective threat)

$$\bigwedge_{j \in \text{Cand}} \neg K_c \text{voted}_{v,j}$$

- The voter knows that the coercer does not know (so that she doesn't have to worry about blackmail)

$$K_v \left(\bigwedge_{j \in \text{Cand}} \neg K_c \text{voted}_{v,j} \right)$$

- The coercer knows that the voter knows that the coercer does not know (hence he knows that blackmail doesn't make sense)

$$K_c K_v \left(\bigwedge_{j \in \text{Cand}} \neg K_c \text{voted}_{v,j} \right)$$

- ...and so on, ad infinitum



Motivating Example: Voting

- The coercer does not know how a particular voter has voted (so he doesn't have an effective threat)

$$\bigwedge_{j \in \text{Cand}} \neg K_c \text{voted}_{v,j}$$

- The voter knows that the coercer does not know (so that she doesn't have to worry about blackmail)

$$K_v \left(\bigwedge_{j \in \text{Cand}} \neg K_c \text{voted}_{v,j} \right)$$

- The coercer knows that the voter knows that the coercer does not know (hence he knows that blackmail doesn't make sense)

$$K_c K_v \left(\bigwedge_{j \in \text{Cand}} \neg K_c \text{voted}_{v,j} \right)$$

- ...and so on, ad infinitum

$$C_{c,v} \left(\bigwedge_{j \in \text{Cand}} \neg K_c \text{voted}_{v,j} \right)$$



Part 1: Formal Specification for MAS

1.6 Knowledge and Time



Combining Knowledge and Time

$$\text{LTLK} = \text{LTL} + \text{K}$$

$$\text{CTLK}^* = \text{CTL}^* + \text{K}$$

Syntax & semantics: freely combine temporal and epistemic operators



Motivating Example: Voting

Properties to express

- Nobody will ever know for whom the voter has voted
- The voter can vote, and can refrain from voting



Motivating Example: Voting

- Nobody will ever know for whom the voter has voted



Motivating Example: Voting

- Nobody will ever know for whom the voter has voted

$$\bigwedge_{a \in \mathbb{A}gt} \bigwedge_{j \in Cand} A \Box \neg K_a \text{vote}_{v,j}$$



Motivating Example: Voting

- Nobody will ever know for whom the voter has voted

$$\bigwedge_{a \in \text{Agt}} \bigwedge_{j \in \text{Cand}} A \Box \neg K_a \text{vote}_{v,j}$$

- Better specification: Nobody will ever know for whom the voter has voted, and for whom she hasn't



Motivating Example: Voting

- Nobody will ever know for whom the voter has voted

$$\bigwedge_{a \in \text{Agt}} \bigwedge_{j \in \text{Cand}} A \Box \neg K_a \text{vote}_{v,j}$$

- Better specification: Nobody will ever know for whom the voter has voted, and for whom she hasn't

$$\bigwedge_{a \in \text{Agt}} \bigwedge_{j \in \text{Cand}} A \Box (\neg K_a \text{vote}_{v,j} \wedge \neg K_a \neg \text{vote}_{v,j})$$



Motivating Example: Voting

- Nobody will ever know for whom the voter has voted

$$\bigwedge_{a \in \mathbb{A}gt} \bigwedge_{j \in Cand} A\Box \neg K_a \text{vote}_{v,j}$$

- Better specification: Nobody will ever know for whom the voter has voted, and for whom she hasn't

$$\bigwedge_{a \in \mathbb{A}gt} \bigwedge_{j \in Cand} A\Box (\neg K_a \text{vote}_{v,j} \wedge \neg K_a \neg \text{vote}_{v,j})$$

- The voter is able to vote, and to refrain from voting



Motivating Example: Voting

- Nobody will ever know for whom the voter has voted

$$\bigwedge_{a \in \text{Agt}} \bigwedge_{j \in \text{Cand}} A \Box \neg K_a \text{vote}_{v,j}$$

- Better specification: Nobody will ever know for whom the voter has voted, and for whom she hasn't

$$\bigwedge_{a \in \text{Agt}} \bigwedge_{j \in \text{Cand}} A \Box (\neg K_a \text{vote}_{v,j} \wedge \neg K_a \neg \text{vote}_{v,j})$$

- The voter is able to vote, and to refrain from voting

Cannot be expressed in CTLK*



Part 1: Formal Specification for MAS

1.7 Model Checking



Verification by Model Checking

Model checking problem

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

transition system

modal formula



Verification by Model Checking

Model checking problem

$$\begin{array}{ccc} M, q & \stackrel{?}{\models} & \varphi \\ \text{transition system} & & \text{modal formula} \end{array}$$

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

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



Verification by Model Checking

Model checking problem

$$\begin{array}{ccc} M, q & \stackrel{?}{\models} & \varphi \\ \text{transition system} & & \text{modal formula} \end{array}$$

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

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

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



Local vs. Global Model Checking

Local model checking

We want to implement function

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



Local vs. Global Model Checking

Local model checking

We want to implement function

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

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

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

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 1: Formal Specification for MAS

1.8 Tool Presentation: Uppaal



Verification of Temporal Properties with Uppaal

Tool presentation: **Uppaal**



Part 2: Verification of Strategic Abilities

- 2.1 What Agents Can Achieve
- 2.2 Model Checking for ATL
- 2.3 Imperfect Information
- 2.4 Problems with Complexity
- 2.5 Tool Presentation: MCMAS



Part 2: Verification of Strategic Abilities

2.1 What Agents Can Achieve



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**

$\rightsquigarrow \Phi$ can include temporal operators: $\bigcirc$ (next), $\diamond$ (sometime in the future), $\square$ (always in the future), $\bigcup$ (strong until)



Example Formulas

- $\langle\langle jamesbond \rangle\rangle \Diamond (ski \wedge \neg getBurned)$:
“James Bond can go skiing without getting burned”



Example Formulas

- $\langle\langle jamesbond \rangle\rangle \Diamond (ski \wedge \neg getBurned)$:
“James Bond can go skiing without getting burned”





Example Formulas

- $\langle\langle jamesbond \rangle\rangle \Diamond (ski \wedge \neg getBurned)$:

“James Bond can go skiing without getting burned”



- $\langle\langle jamesbond, bondsgirl \rangle\rangle (\neg destruction) \cup endOfMovie$:

“James Bond and his girlfriend are able to save the world from destruction until the end of the movie”



ATL* and ATL

ATL allows only elementary temporal operators inside a strategy operator.

- ATL formula: $\langle\langle jamesbond, bondsgirl \rangle\rangle (\neg destruction) \cup endOfMovie$.





ATL* and ATL

ATL allows only elementary temporal operators inside a strategy operator.

- ATL formula: $\langle\langle jamesbond, bondsgirl \rangle\rangle (\neg \text{destruction}) \text{ U } \text{endOfMovie}$.



- ATL formula with nesting:

$$\langle\langle jamesbond \rangle\rangle (\neg \text{destruction} \text{ U } (\langle\langle bondsgirl \rangle\rangle \Diamond \text{alive})).$$



ATL* and ATL

ATL allows only elementary temporal operators inside a strategy operator.

- ATL formula: $\langle\langle jamesbond, bondsgirl \rangle\rangle (\neg \text{destruction}) \text{ U } \text{endOfMovie}$.



- ATL formula with nesting:

$\langle\langle jamesbond \rangle\rangle (\neg \text{destruction} \text{ U } (\langle\langle bondsgirl \rangle\rangle \Diamond \text{alive}))$.

- ATL* formula: $\langle\langle jamesbond \rangle\rangle (\neg \text{destruction} \text{ U } \text{endOfMovie} \wedge \Diamond \text{alive})$.



Strategies

Definition 2.1 (Strategy)

A **strategy** is a **conditional plan**.



Strategies

Definition 2.1 (Strategy)

A **strategy** is a **conditional plan**.

We represent strategies by functions $s_a : St \rightarrow Act$.

$\leadsto$ **memoryless agents**



Strategies

Definition 2.1 (Strategy)

A **strategy** is a **conditional plan**.

We represent strategies by functions $s_a : St \rightarrow Act$.

$\leadsto$ **memoryless agents**

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



Strategies

Definition 2.1 (Strategy)

A **strategy** is a **conditional plan**.

We represent strategies by functions $s_a : St \rightarrow Act$.

$\leadsto$ **memoryless agents**

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

A **collective strategy** is simply a **tuple** of individual strategies.



Strategies

Definition 2.1 (Strategy)

A **strategy** is a **conditional plan**.

We represent strategies by functions $s_a : St \rightarrow Act$.

$\leadsto$ **memoryless agents**

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

A **collective strategy** is simply a **tuple** of individual strategies.

Definition 2.2 (Outcome of a strategy)

Function $out(q, s_A)$ returns the **set of all paths that may result from agents A executing strategy s_A from state q onward**.



Semantics of ATL*

Definition 2.3 (Semantics of ATL*: state formulae)

$M, q \models \langle\langle A \rangle\rangle \Phi$ iff there is a collective strategy s_A such that, for every path $\lambda \in \text{out}(q, s_A)$, we have $M, \lambda \models \Phi$;



Semantics of ATL*

Definition 2.3 (Semantics of ATL*: state formulae)

$M, q \models \langle\langle A \rangle\rangle \Phi$ iff there is a collective strategy s_A such that, for every path $\lambda \in out(q, s_A)$, we have $M, \lambda \models \Phi$;

Definition 2.4 (Semantics of ATL*: path formulae)

Analogous to LTL and CTL* !



Semantics of ATL*

Definition 2.3 (Semantics of ATL*: state formulae)

$M, q \models \langle\langle A \rangle\rangle \Phi$ iff there is a collective strategy s_A such that, for every path $\lambda \in \text{out}(q, s_A)$, we have $M, \lambda \models \Phi$;

Definition 2.4 (Semantics of ATL*: path formulae)

$M, \lambda \models \varphi$ iff $M, \lambda[0] \models \varphi$, for a state formula φ ;

$M, \lambda \models \bigcirc \gamma$ iff $M, \lambda[1..\infty] \models \gamma$;

$M, \lambda \models \Diamond \gamma$ iff $M, \lambda[i..\infty] \models \gamma$ for some $i \geq 0$;

$M, \lambda \models \Box \gamma$ iff $M, \lambda[i..\infty] \models \gamma$ for all $i \geq 0$;

$M, \lambda \models \gamma_1 \mathbf{U} \gamma_2$ iff $M, \lambda[i..\infty] \models \gamma_2$ for some $i \geq 0$, and $M, \lambda[j..\infty] \models \gamma_1$ for all $0 \leq j \leq i$;



CTL as a particular case of ATL

- $E\Diamond\varphi = \langle\langle dummy \rangle\rangle\Diamond\varphi.$
- $A\Diamond\varphi = \llbracket dummy \rrbracket\Diamond\varphi.$



CTL as a particular case of ATL

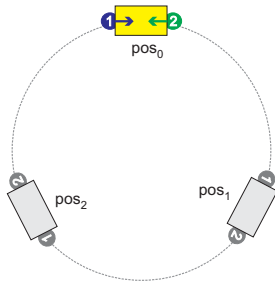
- $E\Diamond\varphi = \langle\langle dummy \rangle\rangle\Diamond\varphi.$
- $A\Diamond\varphi = \llbracket dummy \rrbracket\Diamond\varphi.$

State-based semantics for ATL (in the perfect information case):

$M, q \models \varphi$ instead of $M, \lambda \models \varphi$, where $q = \lambda[0]$

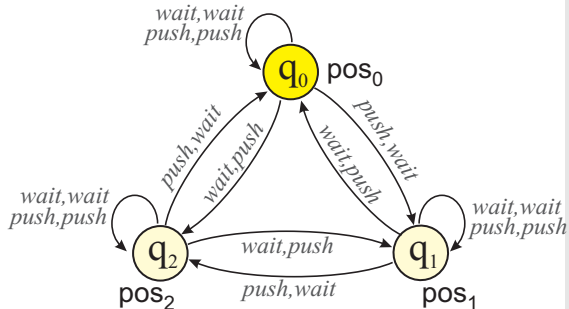
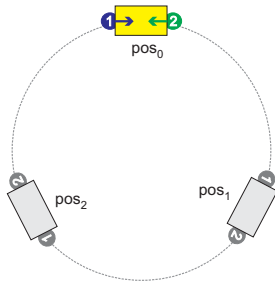


Example: Robots and Carriage



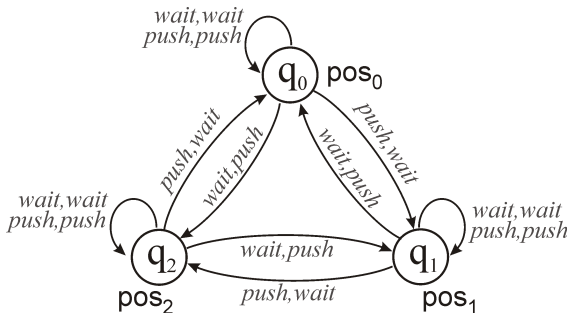


Example: Robots and Carriage





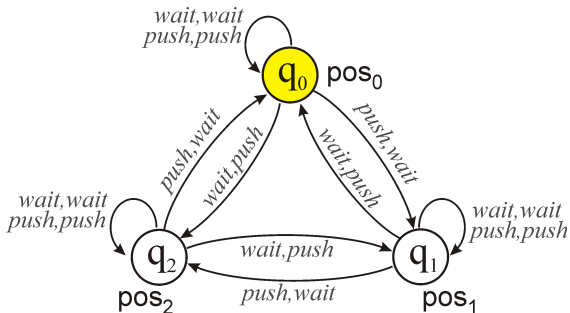
Example: Robots and Carriage



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



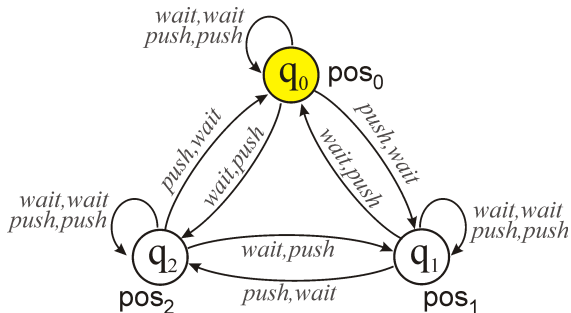
Example: Robots and Carriage



$$pos_0 \rightarrow \langle\langle 1 \rangle\rangle \Diamond pos_1$$



Example: Robots and Carriage

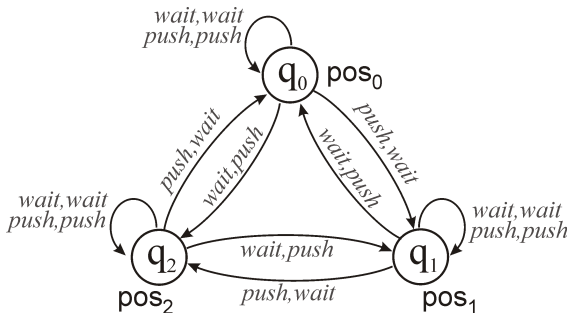


$pos_0 \rightarrow \langle\langle 1 \rangle\rangle \Diamond pos_1$

No!



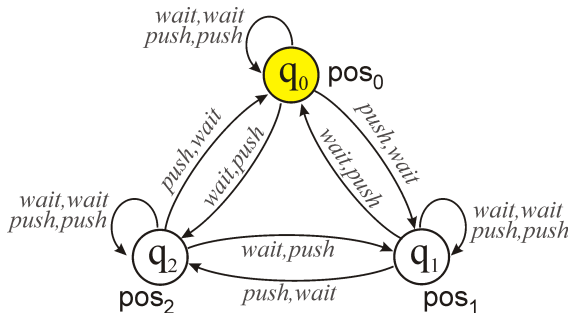
Example: Robots and Carriage



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



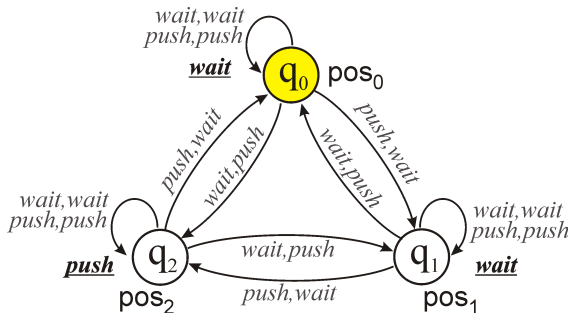
Example: Robots and Carriage



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



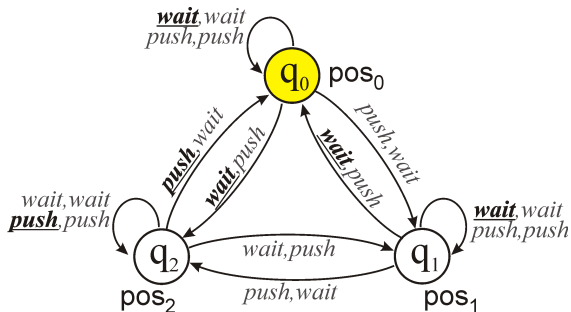
Example: Robots and Carriage



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



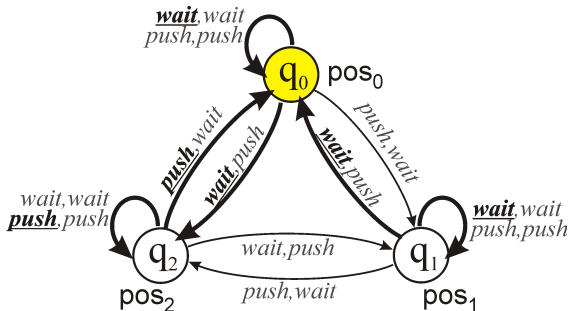
Example: Robots and Carriage



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



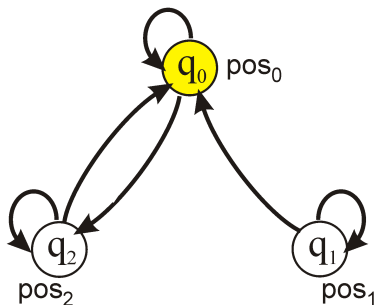
Example: Robots and Carriage



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



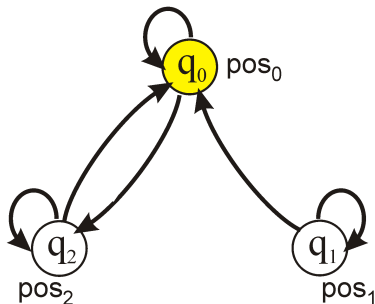
Example: Robots and Carriage



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



Example: Robots and Carriage



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

Yes!



Motivating Example: Voting

- The system cannot reveal how a particular voter voted



Motivating Example: Voting

- The system cannot reveal how a particular voter voted

$\neg \langle\langle system \rangle\rangle \Diamond revealed_v$



Motivating Example: Voting

- The system cannot reveal how a particular voter voted
 $\neg \langle\langle system \rangle\rangle \Diamond revealed_v$
- The voter can vote, and can refrain from voting



Motivating Example: Voting

- The system cannot reveal how a particular voter voted

$$\neg \langle\langle system \rangle\rangle \Diamond \text{revealed}_v$$

- The voter can vote, and can refrain from voting

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



Motivating Example: Voting

- The system cannot reveal how a particular voter voted

$$\neg \langle\langle system \rangle\rangle \Diamond revealed_v$$

- The voter can vote, and can refrain from voting

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

Stronger variant:

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



Motivating Example: Voting

- The system cannot reveal how a particular voter voted

$$\neg \langle\langle system \rangle\rangle \Diamond revealed_v$$

- The voter can vote, and can refrain from voting

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

Stronger variant:

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

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



Motivating Example: Voting

- The system cannot reveal how a particular voter voted

$$\neg \langle\langle system \rangle\rangle \Diamond \text{revealed}_v$$

- The voter can vote, and can refrain from voting

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

Stronger variant:

$$\bigwedge_{j \in Cand} \langle\langle v \rangle\rangle \Diamond \text{vote}_{v,j} \wedge \langle\langle v \rangle\rangle \Box (\bigwedge_{j \in Cand} \neg \text{vote}_{v,j})$$

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

We must refer to knowledge of the coercer!



Motivating Example: Strategic-Epistemic Specifications

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



Motivating Example: Strategic-Epistemic Specifications

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

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



Motivating Example: Strategic-Epistemic Specifications

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

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

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



Motivating Example: Strategic-Epistemic Specifications

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

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

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

Even better: $\bigwedge_{C \subsetneq \text{Cand}} \neg \langle \langle v, c \rangle \rangle \Diamond K_c \left(\bigvee_{j \in C} \text{voted}_{v,j} \right)$



Part 2: Verification of Strategic Abilities

2.2 Model Checking for ATL



Model Checking ATL





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 $m = \text{\#transitions in the model}$ and $l = \text{\#symbols in the formula}$.



Fixpoint Algorithm for ATL with Perfect Information

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

$$\blacksquare \langle\langle A \rangle\rangle \Box \varphi \quad \leftrightarrow \quad \varphi \wedge \langle\langle A \rangle\rangle \bigcirc \langle\langle A \rangle\rangle \Box \varphi$$



Fixpoint Algorithm for ATL with Perfect Information

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

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



Fixpoint Algorithm for ATL with Perfect Information

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

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



Fixpoint Algorithm for ATL with Perfect Information

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

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

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 $\varphi \equiv \langle\langle A \rangle\rangle \Box \psi$:

$Q_1 := Q$; $Q_2 := Q_3 := mcheck(\mathcal{M}, \psi)$;

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 $\varphi \equiv \langle\langle A \rangle\rangle \psi_1 \cup \psi_2$:

$Q_1 := \emptyset$; $Q_2 := mcheck(\mathcal{M}, \psi_2)$; $Q_3 := mcheck(\mathcal{M}, \psi_1)$;

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

$$pre(A, Q) = \{q \mid \exists \alpha_A \forall \alpha_{\text{Agt} \setminus A} . t(q, \alpha_A, \alpha_{\text{Agt} \setminus A}) \in Q\}$$

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**

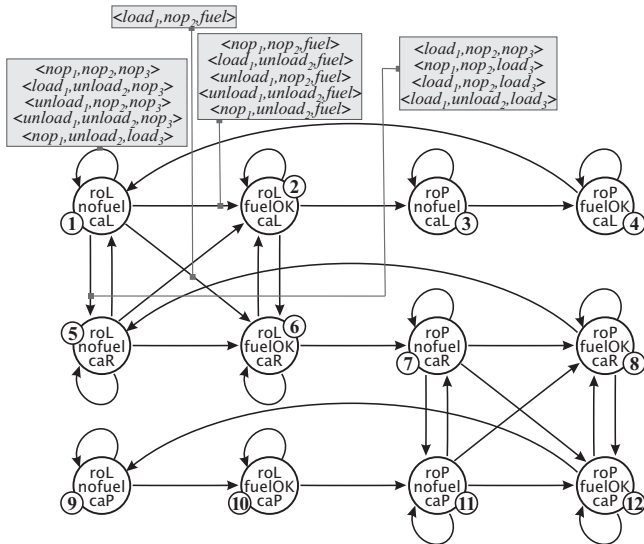


Example: Simple Rocket Domain

- 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.
- $\langle\langle 1, 3 \rangle\rangle \Diamond \text{caP}$



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.
- $\langle\langle 1, 3 \rangle\rangle \Diamond \text{caP} \wedge \langle\langle 1, 3 \rangle\rangle \Diamond \text{caL}$



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.
- $\langle\langle 1, 3 \rangle\rangle \Diamond \text{caP} \wedge \langle\langle 1, 3 \rangle\rangle \Diamond \text{caL}$
- How does that work for the coalition of agents 1 and 2
($\langle\langle 1, 2 \rangle\rangle \Diamond \text{caP}$) ?

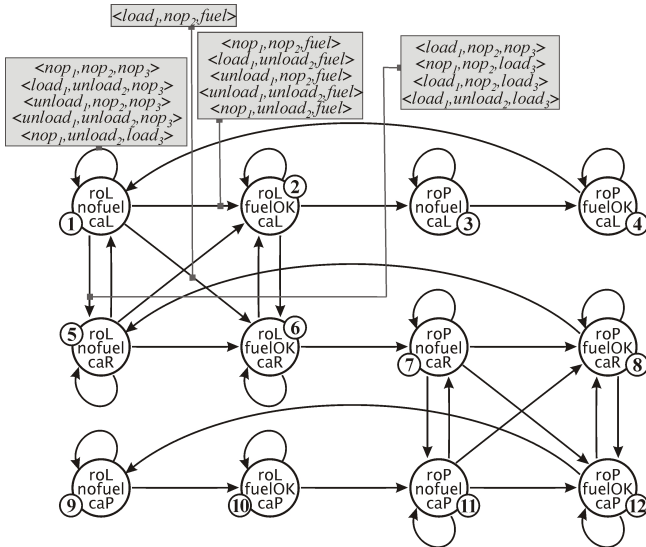


Example: Simple Rocket Domain

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

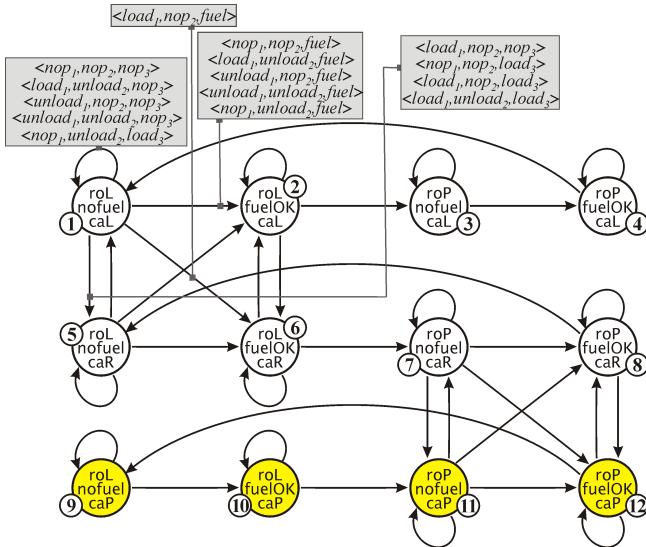


Simple Rocket Domain: Verification of $\langle\langle 1, 3 \rangle\rangle \diamond \text{caP}$



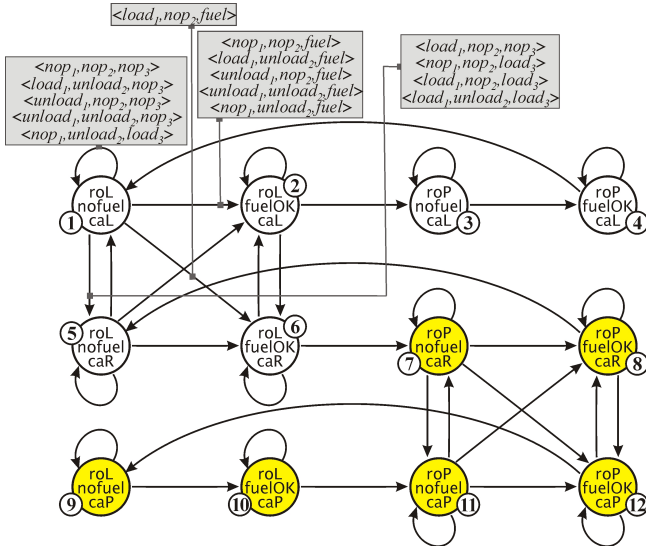


Simple Rocket Domain: Verification of $\langle\langle 1, 3 \rangle\rangle \diamond \text{caP}$



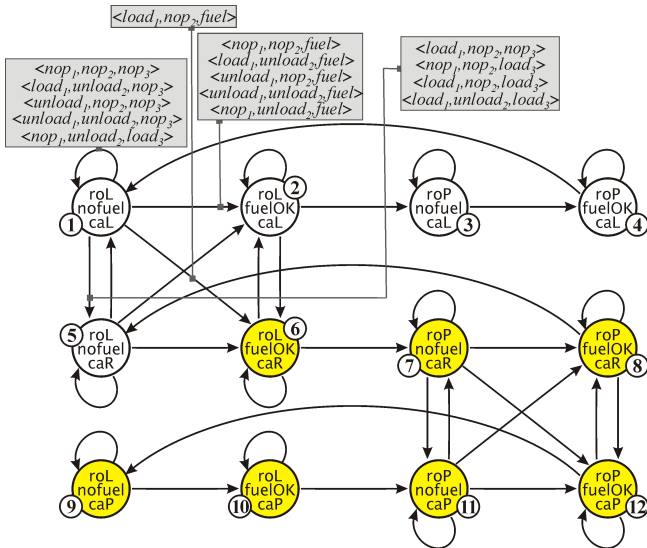


Simple Rocket Domain: Verification of $\langle\langle 1, 3 \rangle\rangle \diamond \text{caP}$



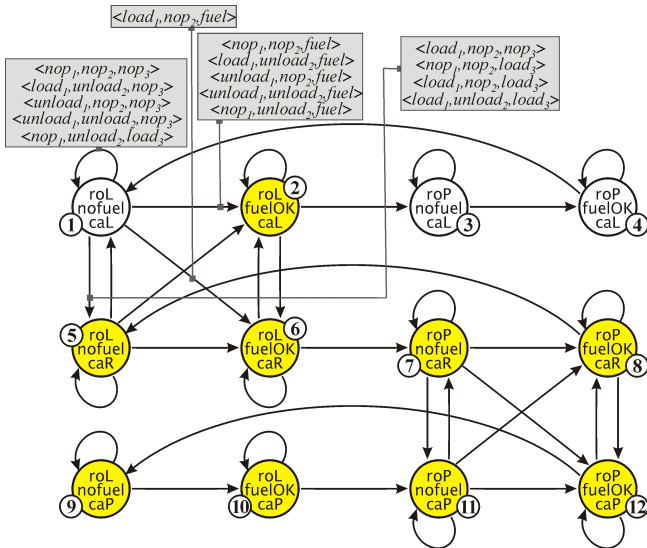


Simple Rocket Domain: Verification of $\langle\langle 1, 3 \rangle\rangle \diamond \text{caP}$



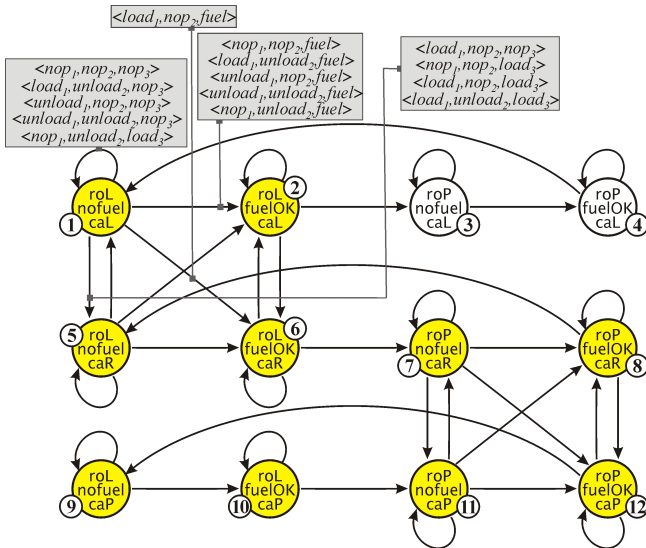


Simple Rocket Domain: Verification of $\langle\langle 1, 3 \rangle\rangle \diamond \text{caP}$



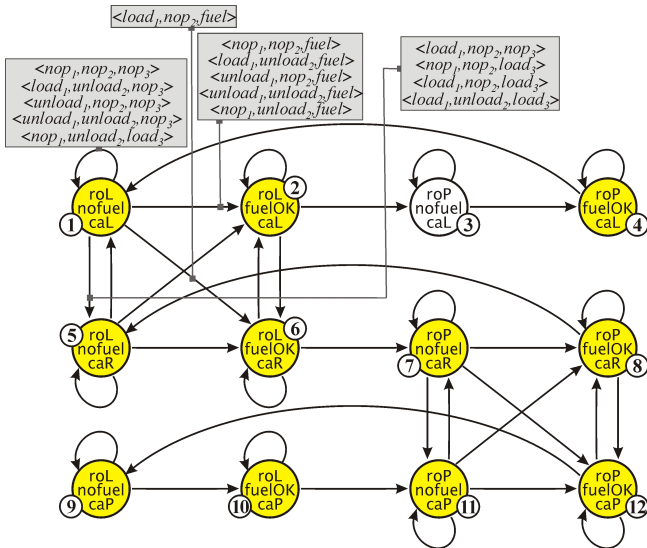


Simple Rocket Domain: Verification of $\langle\langle 1, 3 \rangle\rangle \diamond \text{caP}$



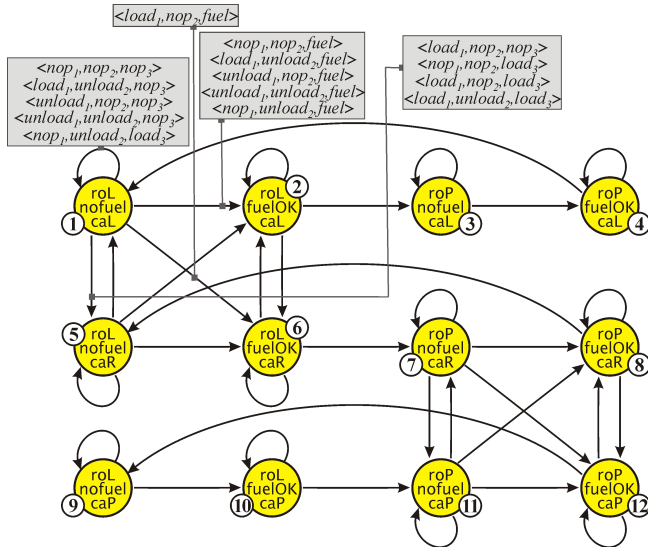


Simple Rocket Domain: Verification of $\langle\langle 1, 3 \rangle\rangle \diamond \text{caP}$



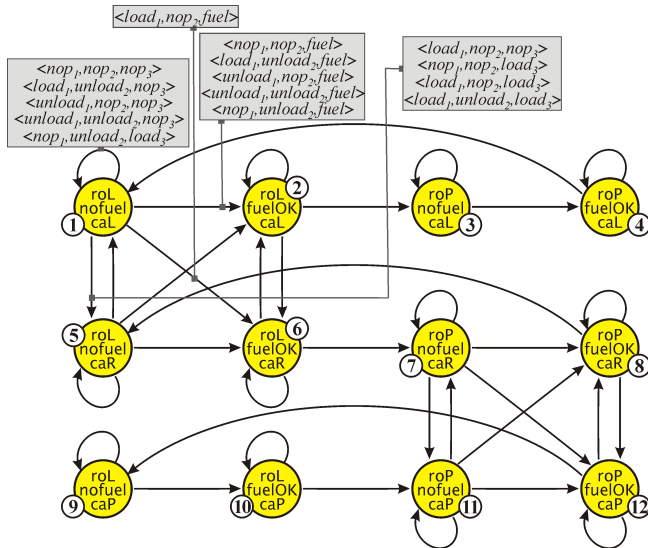


Simple Rocket Domain: Verification of $\langle\langle 1, 3 \rangle\rangle \diamond \text{caP}$





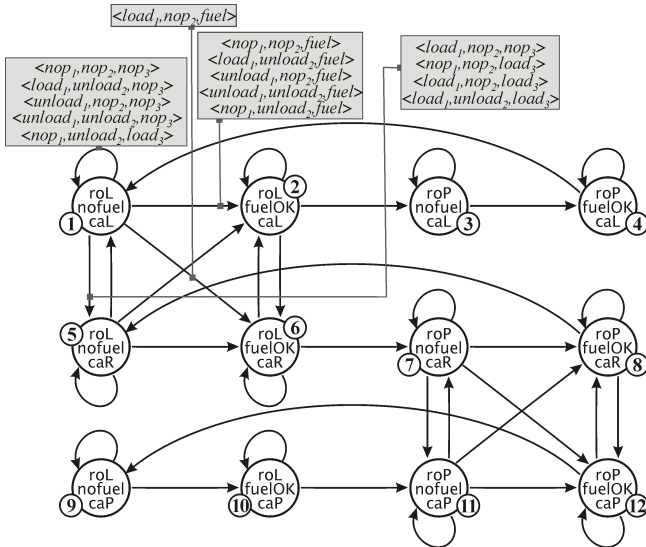
Simple Rocket Domain: Verification of $\langle\langle 1, 3 \rangle\rangle \diamond \text{caP}$



Done!

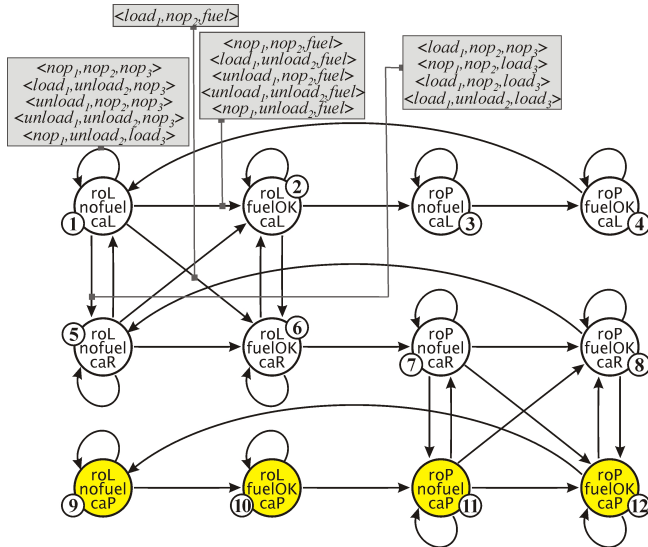


Simple Rocket Domain: Verification of $\langle\langle 1, 2 \rangle\rangle \diamond \text{caP}$



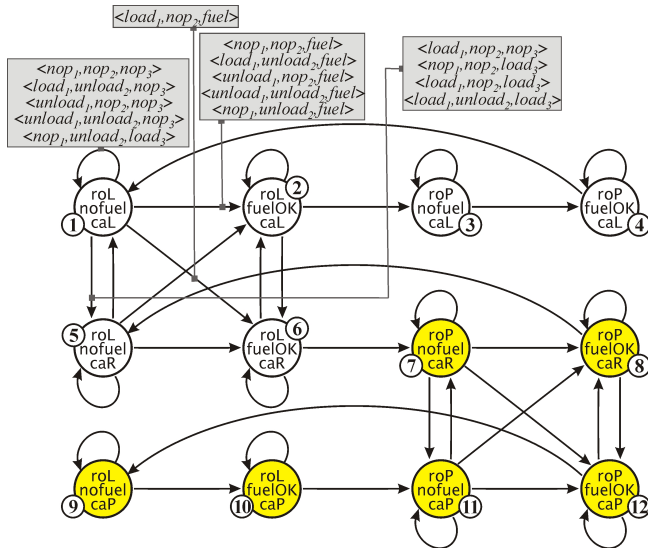


Simple Rocket Domain: Verification of $\langle\langle 1, 2 \rangle\rangle \diamond \text{caP}$



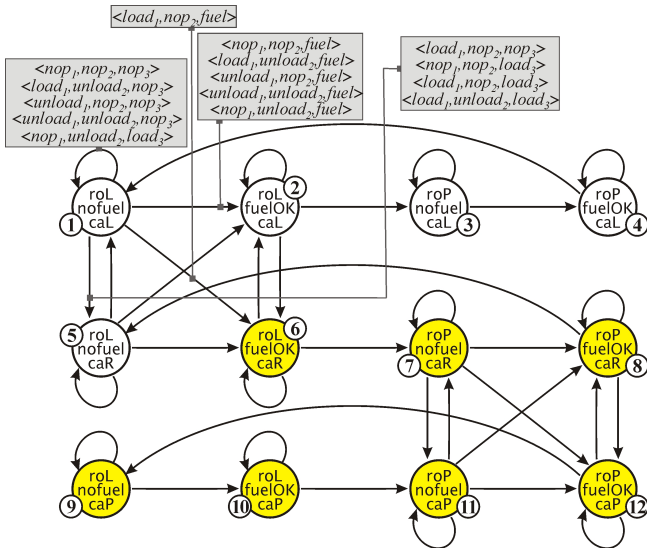


Simple Rocket Domain: Verification of $\langle\langle 1, 2 \rangle\rangle \diamond \text{caP}$



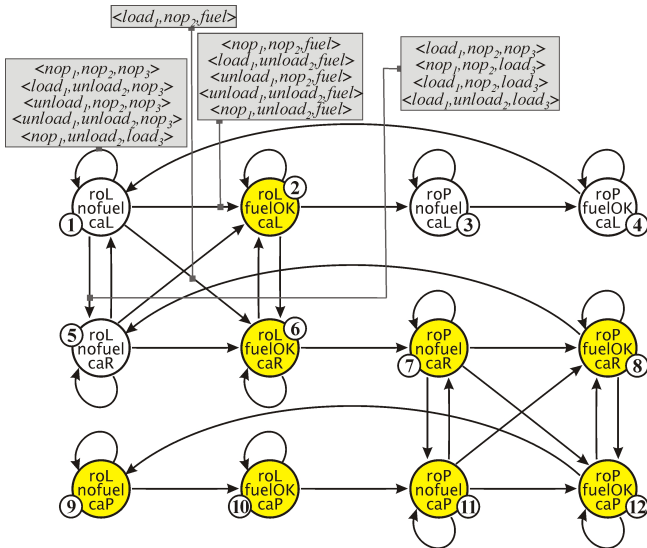


Simple Rocket Domain: Verification of $\langle\langle 1, 2 \rangle\rangle \diamond \text{caP}$



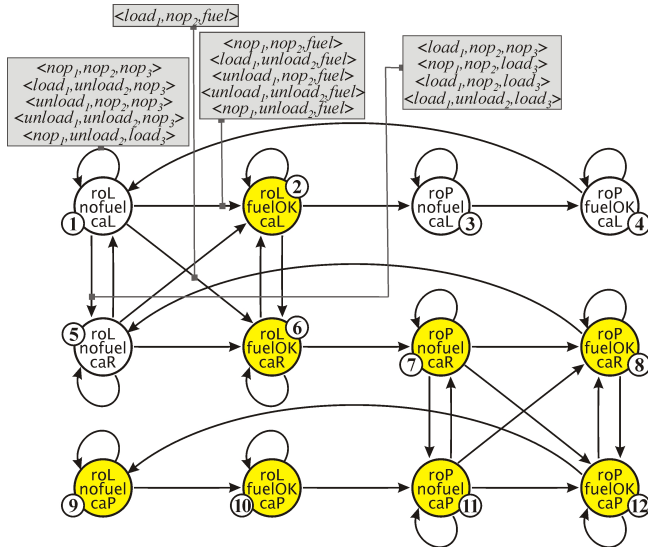


Simple Rocket Domain: Verification of $\langle\langle 1, 2 \rangle\rangle \diamond \text{caP}$





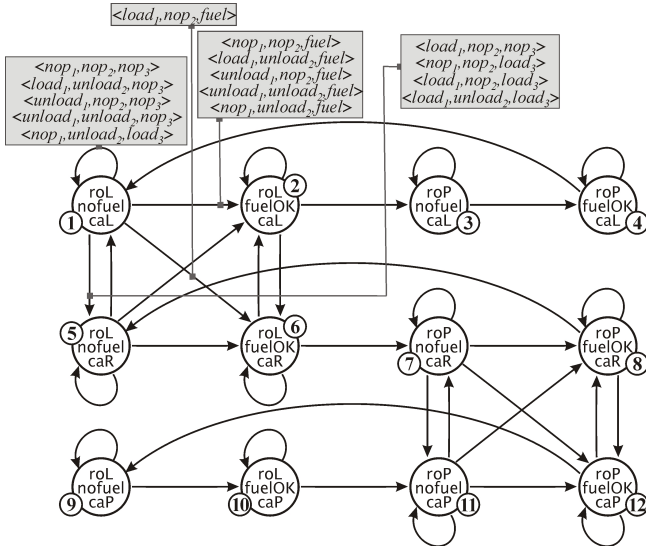
Simple Rocket Domain: Verification of $\langle\langle 1, 2 \rangle\rangle \diamond \text{caP}$



Done!

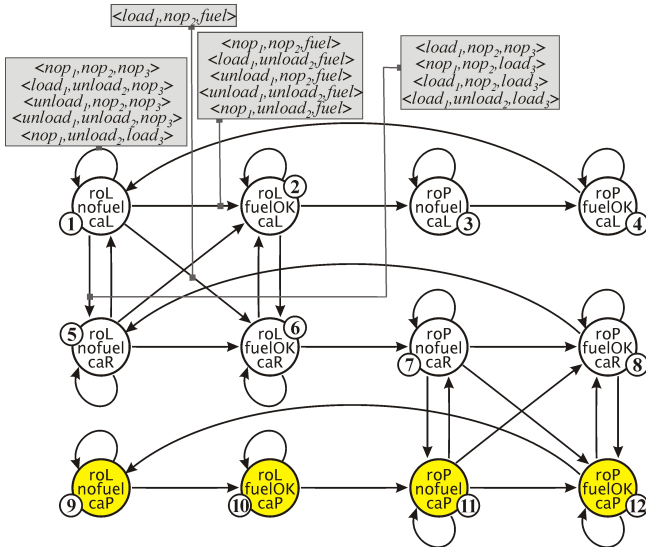


Simple Rocket Domain: Verification of $\langle\langle 3 \rangle\rangle \Box \text{caP}$



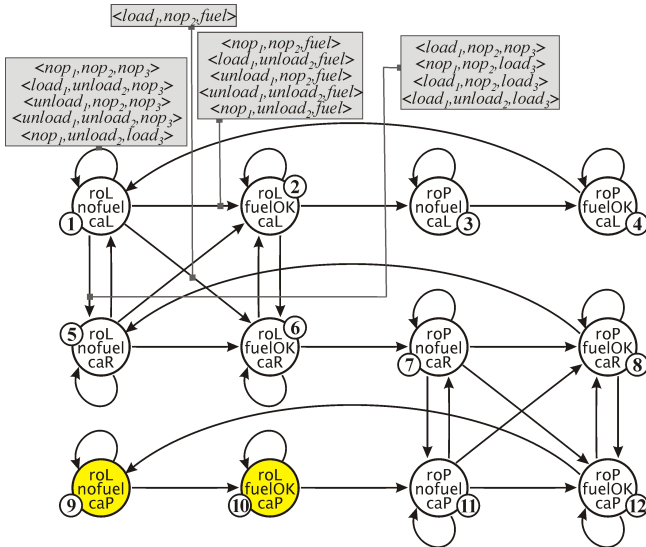


Simple Rocket Domain: Verification of $\langle\langle 3 \rangle\rangle \Box \text{caP}$



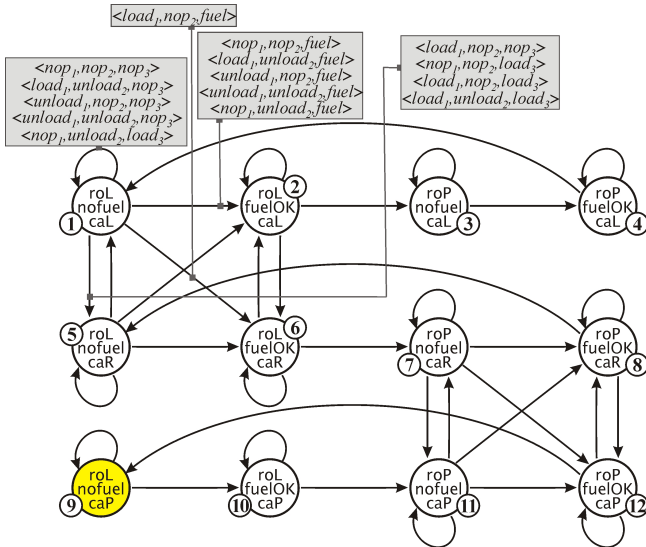


Simple Rocket Domain: Verification of $\langle\langle 3 \rangle\rangle \Box \text{caP}$

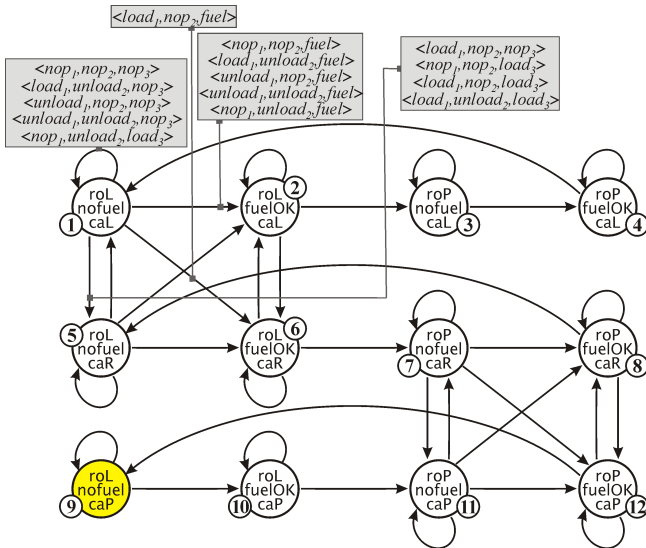




Simple Rocket Domain: Verification of $\langle\langle 3 \rangle\rangle \Box \text{caP}$



Simple Rocket Domain: Verification of $\llbracket 3 \rrbracket \Box \text{caP}$



Done!



Model Checking ATL for Perfect Information

Theorem (Alur, Kupferman & Henzinger 1998/2002)

ATL model checking for perfect information games is **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 Abilities

2.3 Imperfect Information



Executable Strategies under Imperfect Information

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

Definition 2.5 (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** $\leadsto$ **uniform strategies**

Definition 2.5 (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.



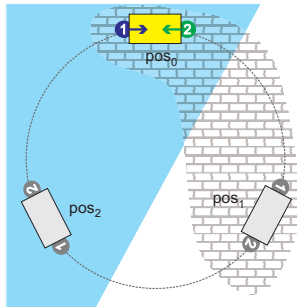
Semantics with imperfect information

Definition 2.6

$M, q \models \langle\langle A \rangle\rangle \Phi$ iff there is a collective strategy s_A such that,
for every $q' \sim_a q$ for some $a \in A$ and
for every path $\lambda \in out(q', s_A)$, we have $M, \lambda \models \Phi$;

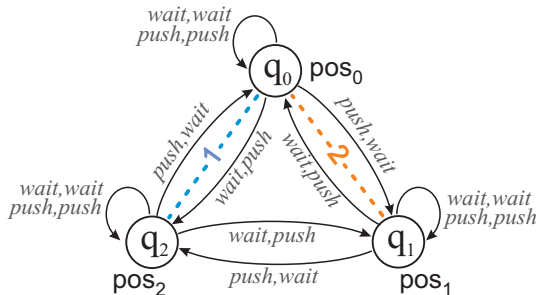
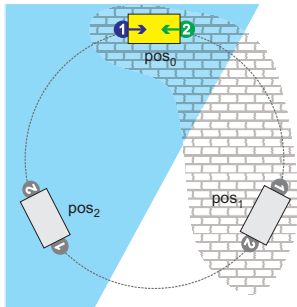


Example: Robots and Carriage



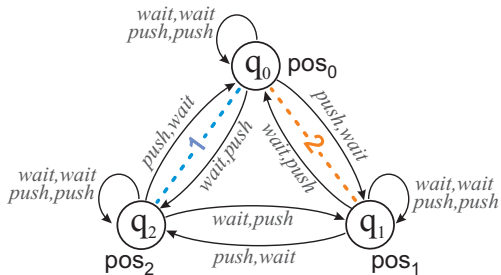


Example: Robots and Carriage



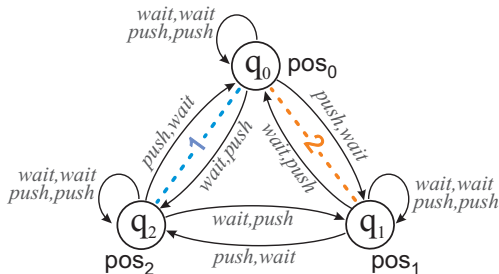


Example: Robots and Carriage





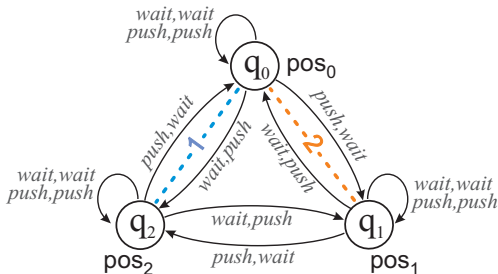
Example: Robots and Carriage



$$pos_0 \models \langle\langle 1 \rangle\rangle_{ir} \Box \neg pos_1$$



Example: Robots and Carriage



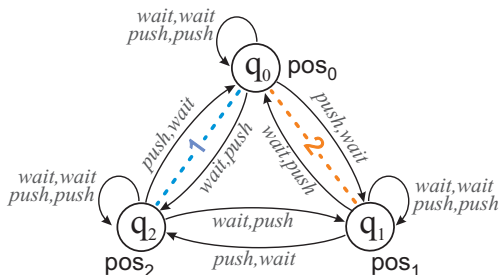
$$pos_0 \models \langle\langle 1 \rangle\rangle_{ir} \Box \neg pos_1$$

No

$$pos_0 \models \langle\langle 1, 2 \rangle\rangle_{ir} \Box \neg pos_1$$



Example: Robots and Carriage



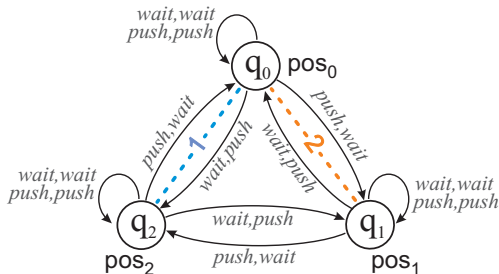
$\text{pos}_0 \models \langle\langle 1 \rangle\rangle_{ir} \Box \neg \text{pos}_1$ **No**

$\text{pos}_0 \models \langle\langle 1, 2 \rangle\rangle_{ir} \Box \neg \text{pos}_1$ **No**

$\text{pos}_0 \models \langle\langle 1, 2 \rangle\rangle_{ir} \Diamond \text{pos}_1$



Example: Robots and Carriage



$pos_0 \models \langle\langle 1 \rangle\rangle_{ir} \Box \neg pos_1$ No

$pos_0 \models \langle\langle 1, 2 \rangle\rangle_{ir} \Box \neg pos_1$ No

$pos_0 \models \langle\langle 1, 2 \rangle\rangle_{ir} \Diamond pos_1$ Yes



Semantic Variants of ATL

Memory of agents:

- **Perfect recall** (R) vs. **imperfect recall** strategies (r)

Available information:

- **Perfect information** (I) vs. **imperfect information** strategies (i)

Success of strategies:

- **Objectively** vs. **subjectively successful** strategies



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!



Part 2: Verification of Strategic Abilities

2.4 Problems with Complexity



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 Δ_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 Δ_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**.



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**



Part 2: Verification of Strategic Abilities

2.5 Tool Presentation: MCMAS



Interpreted systems programming language (ISPL)

- Each agent owns a set of variables.
- Local state is characterized by local variable values.



Interpreted systems programming language (ISPL)

- Each agent owns a set of variables.
- Local state is characterized by local variable values.
- Special "Environment" agent.
- Some **Environment** variables may be read by all agents ("Obsvars").
- Some **Environment** variables may be read by specific agents ("Lobsvars").

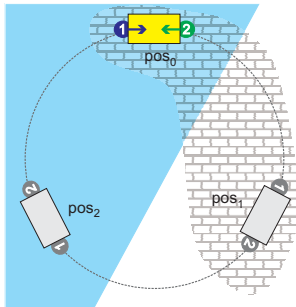


Interpreted systems programming language (ISPL)

- Each agent owns a set of variables.
- Local state is characterized by local variable values.
- Special "Environment" agent.
- Some **Environment** variables may be read by all agents ("Obsvars").
- Some **Environment** variables may be read by specific agents ("Lobsvars").
- Each agent may issue a set of actions.
- Different actions may be available at different states ("Protocol").

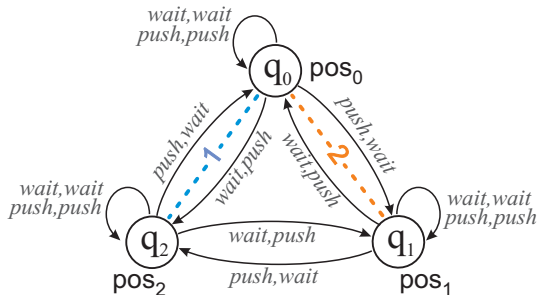
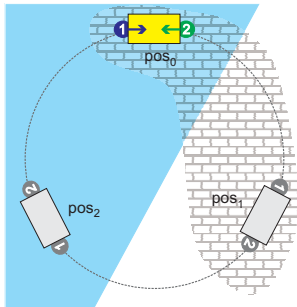


Example: Robots and Carriage





Example: Robots and Carriage





ISPL syntax: local variables and protocol

Agent Environment

Vars:

state : {pos0,pos1,pos2};

vis1 : {p01, p2};

vis2 : {p0, p12};

end Vars

Actions = {none};

Protocol:

Other : {none};

end Protocol

...



ISPL syntax: local variables and protocol

```
Agent Ag1
```

```
  Lobsvars = {vis1};
```

```
  Vars:
```

```
    state : {none};
```

```
  end Vars
```

```
  Actions = {push, wait};
```

```
  Protocol:
```

```
    Other : {push,wait};
```

```
  end Protocol
```

```
...
```

```
end Agent
```



Transition specification in ISPL

- **Guarded commands** specify the **possible** updates of local variables.



Transition specification in ISPL

- **Guarded commands** specify the **possible** updates of local variables.
- A global transition is obtained by simultaneously executing several guarded commands.



Transition specification in ISPL

- **Guarded commands** specify the **possible** updates of local variables.
- A global transition is obtained by simultaneously executing several guarded commands.
- Guarded commands for the same local variable are exclusive.



Transition specification in ISPL

- **Guarded commands** specify the **possible** updates of local variables.
- A global transition is obtained by simultaneously executing several guarded commands.
- Guarded commands for the same local variable are exclusive.
- Guarded commands controlled by enabling conditions.
- Enabling conditions may check the last action issued by any agent.



Transition specification in ISPL

- **Guarded commands** specify the **possible** updates of local variables.
- A global transition is obtained by simultaneously executing several guarded commands.
- Guarded commands for the same local variable are exclusive.
- Guarded commands controlled by enabling conditions.
- Enabling conditions may check the last action issued by any agent.
- Initial states specified globally.
- Multiple initial states possible (constraint on initial variable assignment).



ISPL syntax: guarded commands and evolution

Evolution:

```
state = pos0 if (state=pos0 and Ag1.Action = push
                  and Ag2.Action = push)
or (state=pos0 and Ag1.Action = wait and Ag2.Action = wait)
or (state=pos1 and Ag1.Action = wait and Ag2.Action = push)
or (state=pos2 and Ag1.Action = push and Ag2.Action = wait);
```



ISPL syntax: guarded commands and evolution

Evolution:

```
state = pos0 if (state=pos0 and Ag1.Action = push
                  and Ag2.Action = push)
  or (state=pos0 and Ag1.Action = wait and Ag2.Action = wait)
  or (state=pos1 and Ag1.Action = wait and Ag2.Action = push)
  or (state=pos2 and Ag1.Action = push and Ag2.Action = wait);

state = pos1 if (state=pos1 and Ag1.Action = push
                  and Ag2.Action = push)
  or (state=pos1 and Ag1.Action = wait and Ag2.Action = wait)
  or (state=pos2 and Ag1.Action = wait and Ag2.Action = push)
  or (state=pos0 and Ag1.Action = push and Ag2.Action = wait);
```




ISPL syntax: guarded commands and evolution

Evolution:

```
state = pos0 if (state=pos0 and Ag1.Action = push
                  and Ag2.Action = push)
  or (state=pos0 and Ag1.Action = wait and Ag2.Action = wait)
  or (state=pos1 and Ag1.Action = wait and Ag2.Action = push)
  or (state=pos2 and Ag1.Action = push and Ag2.Action = wait);

state = pos1 if (state=pos1 and Ag1.Action = push
                  and Ag2.Action = push)
  or (state=pos1 and Ag1.Action = wait and Ag2.Action = wait)
  or (state=pos2 and Ag1.Action = wait and Ag2.Action = push)
  or (state=pos0 and Ag1.Action = push and Ag2.Action = wait);

vis1 = p01 if (state=pos0 and Ag1.Action = push
               and Ag2.Action = push)
  or (state=pos0 and Ag1.Action = wait and Ag2.Action = wait)
  or (state=pos1 and Ag1.Action = wait and Ag2.Action = push)
  or (state=pos2 and Ag1.Action = push and Ag2.Action = wait)
  or (state=pos1 and Ag1.Action = push and Ag2.Action = push)
  or (state=pos1 and Ag1.Action = wait and Ag2.Action = wait)
  or (state=pos2 and Ag1.Action = wait and Ag2.Action = push)
  or (state=pos0 and Ag1.Action = push and Ag2.Action = wait);
end Evolution
```



Initial states

InitStates

```
(Environment.state = pos2 and Environment.vis1=p2
    and Environment.vis2=p12)
or (Environment.state = pos0 and Environment.vis1=p01
    and Environment.vis2=p0);
```

end InitStates



Formulas in MCMAS

- Atomic propositions different from state variables.
- Truth of atomic proposition defined by (global) constraint on state variables.



Formulas in MCMAS

- Atomic propositions different from state variables.
- Truth of atomic proposition defined by (global) constraint on state variables.
- CTL operators.
- Group definitions (needed for epistemic and coalition operators).
- Epistemic operators: $K_{group}, D_{group}, C_{group}$.
- Coalition operators: $\langle group \rangle$.



Formulas in MCMAS

Evaluation

```
at0 if Environment.state = pos0;  
at1 if Environment.state = pos1;  
at2 if Environment.state = pos2;
```

end Evaluation



Formulas in MCMAS

Groups

```
alice = {Ag1};  
all = {Ag1, Ag2};
```

end Groups

Formulae

```
<alice> G ! at1;  
<all> G (at0 or at2);
```

end Formulae



Majority voting example

Consider a majority election system:

- 3 candidates: Macron, Mélenchon, Philippe.
- Parliament composed of **2 seats**.
- 8 voters.
- Lexicographic order in case of ties.

Can voters 1-3 make Macron get a seat?



Majority voting example

Consider a majority election system:

- 3 candidates: Macron, Mélenchon, Philippe.
- Parliament composed of **2 seats**.
- 8 voters.
- Lexicographic order in case of ties.

Can voters 1-3 make Macron get a seat?

Can voters 1-5 achieve the objective of Macron being not elected?



Majority voting example

Consider a majority election system:

- 3 candidates: Macron, Mélenchon, Philippe.
- Parliament composed of **2 seats**.
- 8 voters.
- Lexicographic order in case of ties.

Can voters 1-3 make Macron get a seat?

Can voters 1-5 achieve the objective of Macron being not elected?

Can voters 1-6 achieve the objective of Macron being not elected?



The art of modeling = the art of programming

- Each voter allowed to cast their ballot only once, Environment is the polling station: boolean `vote1, ..., vote8`.
- Voters may enter the booth in arbitrary order
`turn : {1, 2, 3, 4, 5, 6, 7, 8}`, nondeterministically assigned new value.
- Agent 3 votes by issuing action: `voteMac, voteMel, votePhi`, allowed when `Environment.vote3 = true`.
- Environment "captures" agent action and updates counters
`countMac, countMel, countPhi`.
- Check winner after all voters cast their ballots.



Part 3: Practical Model Checking

- 3.1 Towards Practical Model Checking
- 3.2 Approximate Model Checking
- 3.3 DominoDFS
- 3.4 Case Study: Verification of Social AI
- 3.5 Tool Presentation: STV



Part 3: Practical Model Checking

3.1 Towards Practical Model Checking



Towards Practical Model Checking for Strategies





Not That Easy...

Caveat: there are serious **complexity obstacles**:

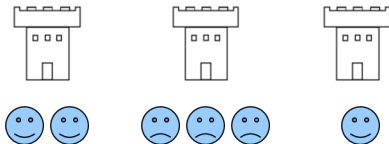
- **State-space** explosion
- **Transition-space** explosion
- **Strategy-space** explosion
- Invalidity of **fixpoint equivalences** for imperfect information

Model checking agent logics for agents with **imperfect information** ranges from **NP-complete** to **undecidable**, depending on the exact syntax, semantics, and representation of models.



Model Checking for Strategies

- Model checking for strategies under partial observability is hard
- But... **how hard** is it in practice?
- We will check it experimentally on a simple class of models
 $\leadsto$ **Castles**





Benchmark: Models

Castles

The model consists of one agent called **Environment** that keeps track of the health points of three castles, plus a number of agents called **Workers** each of whom is assigned to a castle.

Health points (ranging from 0 to 3) represent the current condition of the castle; 0 HP means that the castle is **defeated**.

Workers can execute the following actions:

- 1 attack a castle they do not work for,
- 2 defend the castle they do work for, or
- 3 do nothing.

No agent can defend its castle twice in a row. A castle gets damaged if the number of attackers is greater than the number of defenders, and the damage is equal to the difference.

Every Worker knows for each castle if it is defeated or not.



Performance Results: Perfect vs. Imperfect Information

N	perfect info (MCMAS)	imperfect info (SMC)	imperfect info (MCMAS)
4 (1 1 1)	43	130	72 000
5 (1 1 2)	70	6 686	timeout
6 (2 1 2)	250	4 508	timeout
7 (2 2 2)	954	3 366	timeout
8 (3 2 2)	1 996	255 549	timeout

Formula: $\varphi_1 \equiv \langle\langle c12 \rangle\rangle \Diamond \text{castle3Defeated}$

Time in ms

timeout = 2h



Not Easy Indeed...

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



Not Easy Indeed...

- Exact verification of strategic abilities is hard
- Possible way out: **incomplete verification**
- Note: the main source of complexity is the **size of the model!**
- Possible way out: use smaller models $\rightsquigarrow$ **model reductions**



Incomplete Model Checking for Strategies



Two ideas:

- Approximate model checking
- Brute force search with local optimization



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

$$LB(p) = p,$$

$$LB(\neg\phi) = \neg UB(\phi),$$

$$LB(\phi \wedge \psi) = LB(\phi) \wedge LB(\psi),$$

$$LB(\langle A \rangle \phi) = \langle A \rangle LB(\phi),$$

$$LB(\langle\langle A \rangle\rangle \Box \phi) = \nu Z. (C_A LB(\phi) \wedge \langle A \rangle \bullet Z),$$

$$LB(\langle\langle A \rangle\rangle \psi \cup \phi) = \mu Z. (E_A LB(\phi) \vee (C_A LB(\psi) \wedge \langle A \rangle \bullet Z)).$$

$$UB(p) = p,$$

$$UB(\neg\phi) = \neg LB(\phi),$$

$$UB(\phi \wedge \psi) = UB(\phi) \wedge UB(\psi),$$

$$UB(\langle A \rangle \phi) = E_A \langle\langle A \rangle\rangle_{Ir} \circ UB(\phi),$$

$$UB(\langle\langle A \rangle\rangle \Box \phi) = E_A \langle\langle A \rangle\rangle_{Ir} \Box UB(\phi),$$

$$UB(\langle\langle A \rangle\rangle \psi \cup \phi) = E_A \langle\langle A \rangle\rangle_{Ir} UB(\psi) \cup UB(\phi).$$



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

- Approximations are bound to work only on a subset of cases
- In particular, our approximation will work only on models where the participants have **sufficiently good memory**
- E.g., bound to fail for **Castles** (the agents forget too much...)
- However, in **security analysis** one usually assumes that the attacker remembers all the **relevant** observations



Approximate Verification of Strategic Ability

- Approximations are bound to work only on a subset of cases
- In particular, our approximation will work only on models where the participants have **sufficiently good memory**
- E.g., bound to fail for **Castles** (the agents forget too much...)
- However, in **security analysis** one usually assumes that the attacker remembers all the **relevant** observations
- Benchmark: **card play** (similar mathematical structure to coercion in a voting protocol!)

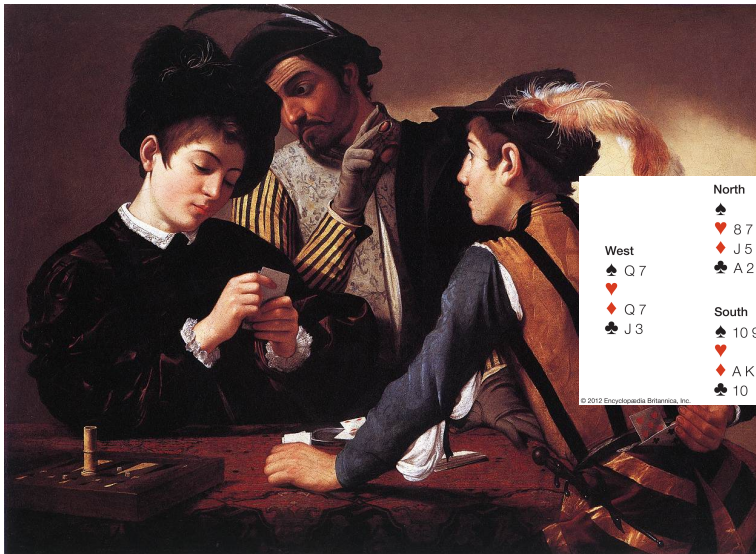


Approximate Verification of Strategic Ability





Approximate Verification of Strategic Ability



West		North	East
♠ Q 7	♥	♠	♠ J 6
♦ Q 7	♦	♥ 8 7	♥
♣ J 3	♣	♦ J 5	♦ 10 8 6
		♣ A 2	♣ 8
		South	
		♠ 10 9	
		♥	
		♦ A K 9	
		♣ 10	

© 2012 Encyclopædia Britannica, Inc.



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*	2443467	132.0	8815.7	4.2	100%	timeout

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

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*	2443467	34.0	3.0	2.0	100%	timeout
20	1.5 e7	124.0	8.5	6.0	100%	timeout
24*	7 e7	3779.0	667.0	78.0	100%	timeout

Formula: $\langle\langle S \rangle\rangle \Diamond \text{win}$

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



Experimental Results for Absent-Minded Declarer

#cards	#states	Approximate verification				Exact verification
		tgen	lower	upper	match	
4	19	<0.01	<0.01	<0.01	100%	9.68 h*
8	713	0.04	0.01	<0.01	100%	timeout
12	52843	5.2	18.6	0.6	80%	timeout
16	memout					timeout

Formula: $\langle\langle S \rangle\rangle \Diamond \text{win}$

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



Fixpoint Approximation with Local Model Generation

Main bottlenecks:

- 1 Generation of global state/transition space (memory & time)
- 2 Verification of the global model (time)



Fixpoint Approximation with Local Model Generation

Main bottlenecks:

- 1 Generation of global state/transition space (memory & time)
- 2 Verification of the global model (time)

For some instances, the **local projection** of the global model suffices for **point 2**!



Fixpoint Approximation with Local Model Generation

#V	Model generation			Verification		
	Global	Approx. Standard	Approx. Optimized	Global	Approx.	Result
2	0.04	6.60	6.54	<0.01	<0.01	TRUE
3	0.10	6.62	6.60	0.29	<0.01	TRUE
4	1.22	6.93	6.91	30.15	<0.01	TRUE
5	35.80	8.71	8.70	2659	<0.01	TRUE
6	1206	36.95	29.42	timeout	<0.01	TRUE
7	timeout	282.48	280.62	——	<0.01	TRUE
8	timeout	5539	4046	——	<0.01	TRUE
9	timeout			——	——	——

Table 1: Results for Asynchronous Simple Voting



Fixpoint Approximation with Local Model Generation

#V	Model generation			Verification		
	Global	Approx. Standard	Approx. Optimized	Global	Approx.	Result
2	0.82	19.43	19.27	8.20	<0.01	TRUE
3	131.61	26.44	19.28	timeout	<0.01	TRUE
4	timeout	524.93	19.25	—	<0.01	TRUE
5	timeout		19.34	—	<0.01	TRUE
6	timeout		19.40	—	<0.01	TRUE
7	timeout		19.41	—	<0.01	TRUE
8	timeout		19.43	—	<0.01	TRUE
9	timeout		19.44	—	<0.01	TRUE

Table 2: Results for Asynchronous Simple Voting with Revoting



Part 3: Practical Model Checking

3.3 DominoDFS



DominoDFS

- When no better idea, try **brute-force search** for a winning strategy
- Give luck a chance $\leadsto$ **Depth-First Search (DFS)**



DominoDFS

- When no better idea, try **brute-force search** for a winning strategy
- Give luck a chance $\leadsto$ **Depth-First Search (DFS)**
- Idea: optimize the search by discarding **dominated strategies**



DominoDFS

- When no better idea, try **brute-force search** for a winning strategy
- Give luck a chance $\leadsto$ **Depth-First Search (DFS)**
- Idea: optimize the search by discarding **dominated partial strategies**



DominoDFS

- When no better idea, try **brute-force search** for a winning strategy
- Give luck a chance $\leadsto$ **Depth-First Search (DFS)**
- Idea: optimize the search by discarding **dominated partial strategies**
- Additional advantage: might work where fixpoint approximation is guaranteed to fail



Strategic Domination

- Consider a **partial strategy** σ_a defined in an epistemic class $[q]_{\sim_a}$
- The **context** of σ_a is given by a partial (possibly nondeterministic) strategy σ_a^C defined everywhere **outside** $[q]_{\sim_a}$
- Then, (σ_a, σ_a^C) specify a full (possibly nondeterministic) strategy of agent a



Strategic Domination

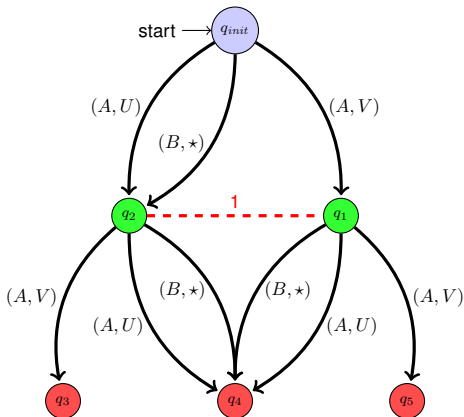
- Consider a **partial strategy** σ_a defined in an epistemic class $[q]_{\sim_a}$
- The **context** of σ_a is given by a partial (possibly nondeterministic) strategy σ_a^C defined everywhere **outside** $[q]_{\sim_a}$
- Then, (σ_a, σ_a^C) specify a full (possibly nondeterministic) strategy of agent a

Strategic Domination

Partial strategy σ_a **dominates** σ'_a with respect to context σ_a^C iff the outcome paths of (σ'_a, σ_a^C) **strictly subsume** those of (σ_a, σ_a^C) .

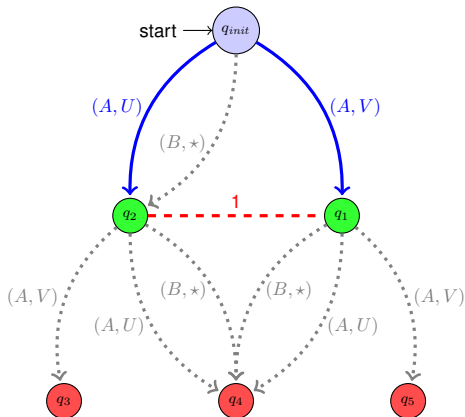


Domination-Based Depth-First Strategy Search



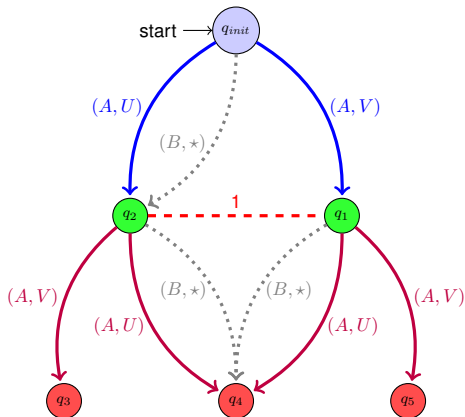


Domination-Based Depth-First Strategy Search



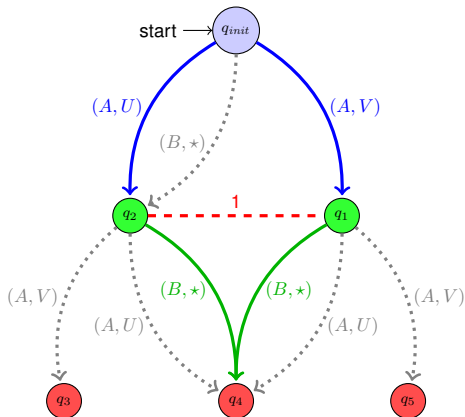


Domination-Based Depth-First Strategy Search



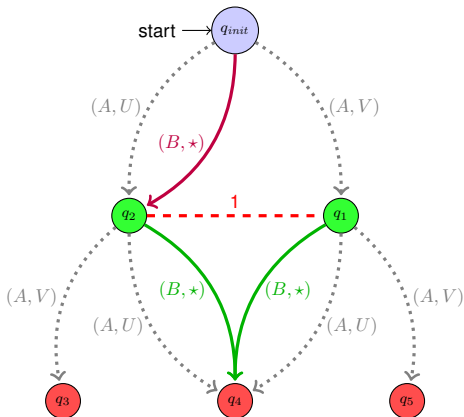


Domination-Based Depth-First Strategy Search





Domination-Based Depth-First Strategy Search





Experimental Results: Bridge Endplay

#cards	DominoDFS	MCMAS	Approx.	Approx. opt.
4	0.0006	0.12	0.0008	< 0.0001
8	0.01	8712	0.01	< 0.0001
12	0.8	timeout	0.8	0.06
16	160	timeout	384	5.5
20	1373	timeout	8951	39
24	memout	timeout	memout	4524



Experimental Results: Castles

#agents	DominoDFS	MCMAS	brute force (SMC)
(1, 1, 1)	0.3	65	63
(2, 1, 1)	1.5	12898	184
(2, 2, 1)	25	timeout	4923
(2, 2, 2)	160	timeout	timeout
(3, 2, 2)	2688	timeout	timeout
(3, 3, 2)	timeout	timeout	timeout

Fixpoint approximation bound to be **inconclusive!**



Part 3: Practical Model Checking

3.4 Case Study: Verification of Social AI



Social Explainable AI

- **Social Explainable AI (SAI)**: a new direction in artificial intelligence
- Emphasis on decentralisation, human-centricity, and explainability
- Aim: put the human again in the spotlight



Social Explainable AI

- **Social Explainable AI (SAI)**: a new direction in artificial intelligence
- Emphasis on decentralisation, human-centricity, and explainability
- Aim: put the human again in the spotlight

- Current work on SAI: development of concepts and solutions
- Focus on its intended functionalities and properties
- Missing: analysis of unintended interactions
- Adversarial play!



Formal Models of SAI

- We focus on the **learning phase** of the SAI protocol
- Each agent represents a machine equipped with an AI component
- The **local model** of an AI agent consists of three phases: the gathering data phase, the learning phase and the sharing phase.



Phase 1: Gathering Data

- The agent is able to gather the data required for the learning phase.
- The corresponding action can be performed multiple times.
- Each time the the local variable that represents the amount of gathered data is increased.
- After the end of the phase the amount of gathered data is analyzed.
- Depending on the exact value, the agent's preparation is marked as incomplete, complete or excessive.
- After this phase the agent proceeds to the learning phase.



Specification of Phase 1 (Gathering Data)

```
start_gathering_data: start -> gather
gather_data:
    gather -[AI1_data<2]> gather
                [AI1_data+=1]
%% 1: Incomplete data
stop_gathering_data:
    gather -[AI1_data < 1]> data_ready
                [AI1_data=0, AI1_data_completion=1]
%% 2: Complete data
stop_gathering_data:
    gather -[1 <= AI1_data < 2]> data_ready
                AI1_data=0, AI1_data_completion=2]
%% 3: Excessive data
stop_gathering_data:
    gather -[2 <= AI1_data]> data_ready
                [AI1_data=0, AI1_data_completion=3]
skip_gathering_data: start -> data_ready
```



Phase 2: Learning

- The agent uses the previously gathered data to train its local AI model.
- The effectiveness of the training depends on the amount of gathered data.
- Excessive data means that the model can be easily overtrained.
- Insufficient data may lead to more iterations required to properly train the model.
- The training action can be performed multiple times.
- Each time the local variable related to the quality of the model is increased.
- After the end of the phase the internal AI model can be overtrained, undertrained or properly trained.
- After this phase the agent is required to share its model with other agents.



Specification of Phase 2 (Learning)

```
start_learning: data_ready -> learn
keep_learning:
    learn -[AI1_information<2]> learn
        [AI1_information+=AI1_data_completion]
%% 1: Incomplete model
stop_learning:
    learn -[AI1_information<1 and AI1_model_quality>0]> educated
        [AI1_information=0, AI1_model_status=1, AI1_model_quality -=1]
stop_learning:
    learn -[AI1_information<1 and AI1_model_quality<=0]> educated
        [AI1_information=0, AI1_model_status=1]
%% 2: Complete model
stop_learning:
    learn -[1<=AI1_information<2 and AI1_model_quality<2]> educated
        [AI1_information=0, AI1_model_status=2, AI1_model_quality +=1]
stop_learning:
    learn -[1<=AI1_information<2 and AI1_model_quality>=2]> educated
        [AI1_information=0, AI1_model_status=2]
%% 3: Overtrained model
stop_learning:
    learn -[2<=AI1_information and AI1_model_quality>0]> educated
        [AI1_information=0, AI1_model_status=3, AI1_model_quality -=1]
stop_learning:
    learn -[2<=AI1_information and AI1_model_quality <= 0]> educated
        [AI1_information=0, AI1_model_status=3]
skip_learning: data_ready -> sharing
```



Phase 3: Sharing

- Agents share their local AI models with each other following a simple sharing protocol.
- The protocol is based on packet traversal in the ring topology.
- Each agent receives the model from the agent with previous ID and sends its model to the agent with next ID.
- The last agent shares its model with the first agent to close the ring.
- To avoid deadlocks, agents with odd IDs first receive the models and then send their own, and agents with even IDs first send their own models and then receive the models from the agents before them.
- When receiving the model, the agent can either accept it or reject it, and its decision is based on the quality of the model being shared.
- After accepting the model, the agent merges it with its own and the resulting model quality is the maximum of the two.
- After the sharing phase, the agent can go back to the learning phase to further train its model.



Specification of Phase 3 (Sharing)

```
start_sharing: educated -> sharing

%% Share local model and get the maximum quality two

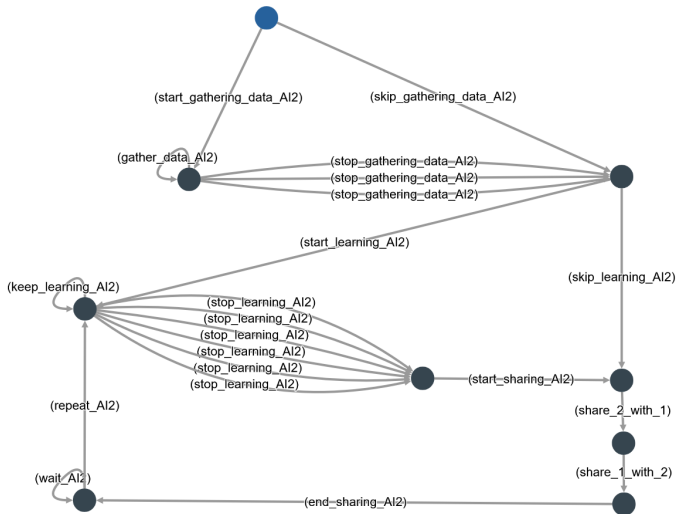
%% receive left
shared share_2_with_1:
    sharing -> sharing2 [ A11_model_quality=^A12_model_quality ]

%% send right
shared share_1_with_2:
    sharing2 -> sharing3 [ A12_model_quality=^A11_model_quality ]

end_sharing: sharing3 -> end
```

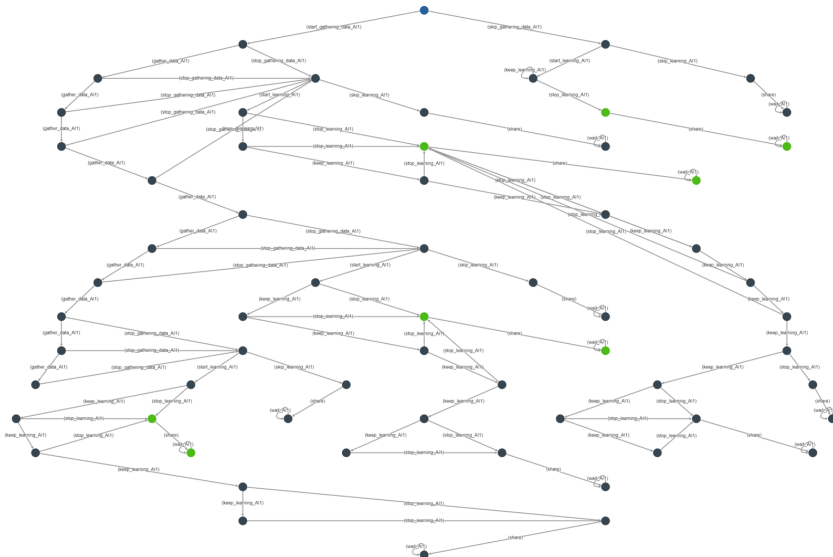


Visualization of Honest AI in STV



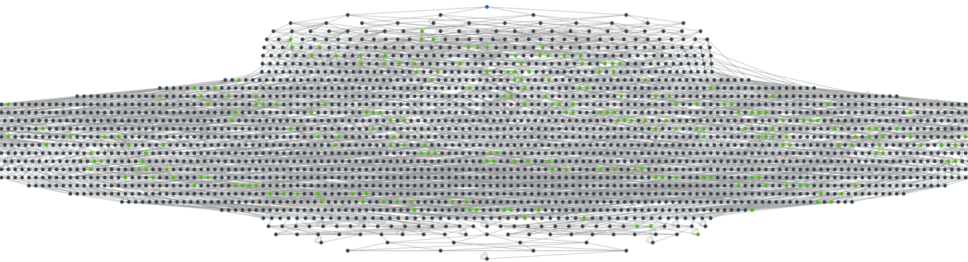


Model of SAI with One Honest Agent





Model of SAI with Two Honest Agents





Modeling Possible Attacks

We consider two attack scenarios:

- Man in the middle
- Impersonator



Man in the Middle Attack

Man in the Middle:

- Does not participate in the data gathering and learning phases
- Participates in the sharing phase
- Can intercept any model that is being sent by one of the honest agents and then pass it to any other agent



Impersonator Attack

Impersonator:

- One of the AI agents is infected with malicious code that results in unwanted behavior
- The agent cannot participate in the data gathering or learning phases
- Can share its model with others following the sharing protocol
- Can fake the quality of its local AI model, hence tricking the next agent into accepting it

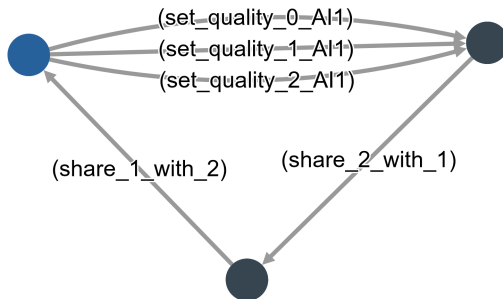


Specification of Impersonator

```
Agent AI2:  
init: start  
set_quality_0: start -> set_quality  
                [AI2_model_quality=0]  
  
set_quality_1: start -> set_quality  
                [AI2_model_quality=1]  
  
set_quality_2: start -> set_quality  
                [AI2_model_quality=2]  
  
shared share_2_with_3: set_quality -> sharing  
shared share_1_with_2: sharing -> start
```

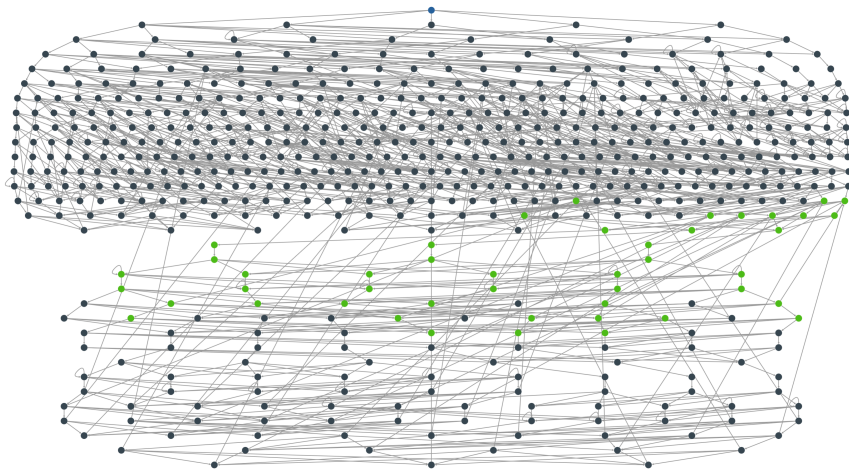


Graphical Representation of Impersonator in STV





Model of SAI with One Honest Agent and One Impersonator





Model of SAI with Two Honest Agents and One Impersonator



Model of SAI with Two Honest Agents and One Impersonator

Manual scrutiny impossible!

~> automated verification



Verification Experiments

Models: n honest AI agents + 1 attacker



Verification Experiments

Models: n honest AI agents + 1 attacker

Formulas:

■ $\phi_1 \equiv \langle\langle A \rangle\rangle \Box (shared_p \rightarrow (\bigwedge_{i \in [1, n]} mqual_i \leq k))$

(Attacker has a strategy to ensure that **all** honest agents will not achieve quality greater than k)



Verification Experiments

Models: n honest AI agents + 1 attacker

Formulas:

- $\phi_1 \equiv \langle\langle A \rangle\rangle \Box (shared_p \rightarrow (\bigwedge_{i \in [1, n]} mqual_i \leq k))$
(Attacker has a strategy to ensure that **all** honest agents will not achieve quality greater than k)
- $\phi_2 \equiv \langle\langle A \rangle\rangle \Box (shared_p \rightarrow (\bigvee_{i \in [1, n]} mqual_i \leq k))$
(Attacker has a strategy to ensure that **some** honest agents will not achieve quality greater than k)



Verification Experiments

Models: n honest AI agents + 1 attacker

Formulas:

- $\phi_1 \equiv \langle\langle A \rangle\rangle \Box (shared_p \rightarrow (\bigwedge_{i \in [1, n]} mqual_i \leq k))$
(Attacker has a strategy to ensure that **all** honest agents will not achieve quality greater than k)
- $\phi_2 \equiv \langle\langle A \rangle\rangle \Box (shared_p \rightarrow (\bigvee_{i \in [1, n]} mqual_i \leq k))$
(Attacker has a strategy to ensure that **some** honest agents will not achieve quality greater than k)

Model checker: STV
(technique: fixpoint approximation)



Model Checking SAI: Impersonator

#Ag	#st	#tr	Gen	Verif ϕ_1	Verif ϕ_2
2	886	2007	< 0.1	< 0.1/False	< 0.1/True
3	79806	273548	28	151/False	202/True
4	6538103	29471247	1284	5061/False	5102/True
5	93581930	623680431	7845	25828/False	25916/True
6	timeout				

All times in seconds

timeout = 8h



Model Checking SAI: Man in the Middle

#Ag	#st	#tr	Gen	Verif ϕ_1	Verif ϕ_2
3	23966	67666	12	21/False	33/True
4	4798302	20257664	875	3810/False	3882/True
5	71529973	503249452	5688	19074/False	20103/True
6	timeout				

All times in seconds

timeout = 8h



Summary

- First attempt at formal analysis of Social Explainable AI
- Work in progress!



Summary

- First attempt at **formal analysis** of **Social Explainable AI**
- Work in progress!
- Very simple models and only some aspects of the procedure were modeled, but...
- ...we successfully model-checked for strategies in systems of up to almost a **billion transitions**
- Considerable achievement for an **NP-hard verification problem!**



Part 3: Practical Model Checking

3.5 Tool Presentation: STV



Strategic Verifier (STV)

- Experimental model checker developed at ICS PAS
- Implemented techniques:
 - 1 standard **fixpoint algorithm** for perfect info games
 - 2 brute force **Depth-First Strategy synthesis**
 - 3 **domination-based strategy search**
 - 4 **fixpoint approximation**
 - 5 distributed strategy synthesis
 - 6 DFS with **on-the-fly model generation**
 - 7 verification of **Natural ATL**
 - 8 model checking for **epistemic** and **Hartley uncertainty operators**
 - 9 DFS for **Probabilistic ATL with imperfect information**
 - 10 **bisimulation checking** for bisimulation-based reduction
 - 11 **partial-order reduction**
 - 12 **assume-guarantee verification**
 - 13 benchmarks and examples
- Includes lightweight GUI and web-based interface



Strategic Verifier (STV)

STV v0.3-alpha

File Edit View Window Help

Voter1 Voter2 Coercer1 Global model **Reduced model**

Partial order reduction

Model file:
C:\Web\Tools\STV_Windows\examples\simple_voting_2.vot
TMS.E

Choose a file...

Formula to be verified:
 $\langle\langle \text{Coercer1} \rangle \rangle F(\text{Coercer1_pun1_isTrue})$

Generate global Generate reduced

Verify Global Model

Lower approx. Upper approx.

Basic heuristic

Domino DFS

Verify Reduced Model

Lower approx. Upper approx.

Basic heuristic

Domino DFS

☐ Show State Labels
☐ Show Actions

Show model information

Q Q

Freeze

Source



STrategic Verifier (STV)

Current build: <https://github.com/blackbat13/stv>

Desktop version for Windows:

<https://github.com/blackbat13/stv/releases/download/v0.3.1-alpha/stv-v0.3-alpha-win32-x64.zip>

Web version: <http://stv.cs-htiew.com/>



STrategic Verifier (STV)

Tool presentation: **STV**



Part 4: Strategy Logic and Rational Verification

- 4.1 Why going beyond ATL?
- 4.2 Some examples
- 4.3 Model-checking SL
- 4.4 Imperfect information
- 4.5 MCMAS-SLK



Part 4: Strategy Logic and Rational Verification

4.1 Why going beyond ATL?



ATL $\simeq$ zero-sum games

ATL (or ATL^{*}) setting is adversarial:

$$\langle\langle \textit{alice} \rangle\rangle\varphi = \exists\sigma_{\textit{alice}}\forall\sigma_{\textit{others}}\varphi$$

- φ : Alice's objective, $\neg\varphi$, is the objective of the others.
- Same for coalition operators $\langle\langle \textit{alice}, \textit{bob} \rangle\rangle\varphi$: φ is a single joint objective for *alice* and *bob*.



ATL $\simeq$ zero-sum games

ATL (or ATL^{*}) setting is adversarial:

$$\langle\langle \textit{alice} \rangle\rangle \varphi = \exists \sigma_{\textit{alice}} \forall \sigma_{\textit{others}} \varphi$$

- φ : Alice's objective, $\neg\varphi$, is the objective of the others.
- Same for coalition operators $\langle\langle \textit{alice}, \textit{bob} \rangle\rangle \varphi$: φ is a single joint objective for *alice* and *bob*.

$M \models \langle\langle \textit{alice} \rangle\rangle \varphi$ implies solving a two-player zero-sum game with the "protagonist" having the LTL objective φ .



ATL $\simeq$ zero-sum games

ATL (or ATL^{*}) setting is adversarial:

$$\langle\langle \textit{alice} \rangle\rangle \varphi = \exists \sigma_{\textit{alice}} \forall \sigma_{\textit{others}} \varphi$$

- φ : Alice's objective, $\neg\varphi$, is the objective of the others.
- Same for coalition operators $\langle\langle \textit{alice}, \textit{bob} \rangle\rangle \varphi$: φ is a single joint objective for *alice* and *bob*.

$M \models \langle\langle \textit{alice} \rangle\rangle \varphi$ implies solving a two-player zero-sum game with the "protagonist" having the LTL objective φ .

- 2-EXPTIME-complete problem.



Beyond zero-sum games

- Game theory is not always about one agent losing = the other agent winning.
- **Nash Equilibrium**: a strategy profile in which individual deviations cannot help improve the outcome.
- First-order formulation: $(\sigma_1, \sigma_2, \dots, \sigma_k)$ is a NE if
 - For any agent i , replacing σ_i with σ'_i produces a **play** with a lower outcome for agent i :

$$\exists \sigma'_i. outcome_i(\sigma_1, \dots, \sigma_k) \geq outcome_i(\sigma_1, \dots, \sigma'_i, \dots, \sigma_k)$$



Beyond zero-sum games

- Game theory is not always about one agent losing = the other agent winning.
- **Nash Equilibrium**: a strategy profile in which individual deviations cannot help improve the outcome.
- First-order formulation: $(\sigma_1, \sigma_2, \dots, \sigma_k)$ is a NE if
 - For any agent i , replacing σ_i with σ'_i produces a **play** with a lower outcome for agent i :

$$\exists \sigma'_i. \text{outcome}_i(\sigma_1, \dots, \sigma_k) \geq \text{outcome}_i(\sigma_1, \dots, \sigma'_i, \dots, \sigma_k)$$

- "Logical" NE problems: how to express/check that some/all NE satisfy some property?



Qualitative Nash Equilibria

- Assume n agents, each having an **LTL objective** φ_i .
- $(\sigma_1, \dots, \sigma_n)$ is a NE for objectives $(\varphi_1, \dots, \varphi_n)$ if each agent cannot improve satisfiability of her objective:
 - Either the (unique!) run **satisfies** φ_i .
 - Or for any other σ'_i , the unique run $(\sigma_1, \dots, \sigma'_i, \dots, \sigma_n)$ **does not satisfy** φ_i .



Qualitative Nash Equilibria

- Assume n agents, each having an **LTL objective** φ_i .
- $(\sigma_1, \dots, \sigma_n)$ is a NE for objectives $(\varphi_1, \dots, \varphi_n)$ if each agent cannot improve satisfiability of her objective:
 - Either the (unique!) run **satisfies** φ_i .
 - Or for any other σ'_i , the unique run $(\sigma_1, \dots, \sigma'_i, \dots, \sigma_n)$ **does not satisfy** φ_i .

Folk fact

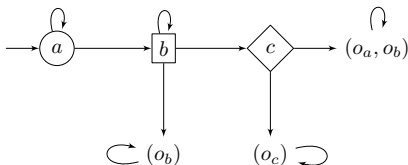
ATL* is unable to express NE.

Some adaptations exist, but...



Nash Equilibrium with reachability objectives

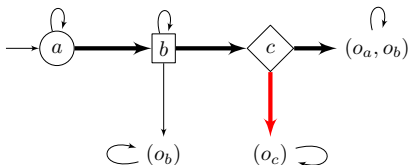
- Three players: *alice*, *bob*, and *charlie*.
- Each agent controls the state with her name.
- Reachability objectives: $\diamond o_a$, $\diamond o_b$, $\diamond o_c$.
- Which strategy profile is a NE?





Nash Equilibrium with reachability objectives

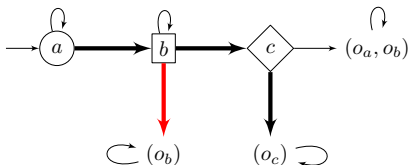
- Three players: *alice*, *bob*, and *charlie*.
- Each agent controls the state with her name.
- Reachability objectives: $\diamond o_a$, $\diamond o_b$, $\diamond o_c$.
- Which strategy profile is a NE?





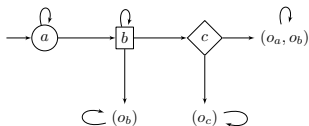
Nash Equilibrium with reachability objectives

- Three players: *alice*, *bob*, and *charlie*.
- Each agent controls the state with her name.
- Reachability objectives: $\diamond o_a$, $\diamond o_b$, $\diamond o_c$.
- Which strategy profile is a NE?





Expressing NE with LTL objectives in some first-order logic



$$\begin{aligned}
 & \exists \sigma_a \exists \sigma_b \exists \sigma_c \Big(\\
 & \quad out(\sigma_a, \sigma_b, \sigma_c) \not\models \Diamond o_a \longrightarrow \forall \sigma'_a out(\sigma'_a, \sigma_b, \sigma_c) \not\models \Diamond o_a \\
 & \quad \wedge \\
 & \quad out(\sigma_a, \sigma_b, \sigma_c) \not\models \Diamond o_b \longrightarrow \forall \sigma'_b out(\sigma_a, \sigma'_b, \sigma_c) \not\models \Diamond o_b \\
 & \quad \wedge \\
 & \quad out(\sigma_a, \sigma_b, \sigma_c) \not\models \Diamond o_c \longrightarrow \forall \sigma'_c out(\sigma_a, \sigma_b, \sigma'_c) \not\models \Diamond o_c \\
 & \quad \Big)
 \end{aligned}$$



Strategy Logic (SL)

Linear Temporal Logic operators + strategy quantifiers + **strategy binders**:

$\exists x$	Variable x is assigned some (new) strategy.
$b(x, a)$	Agent a is assigned the strategy carried by variable x .



Strategy Logic (SL)

Linear Temporal Logic operators + strategy quantifiers + **strategy binders**:

$\exists x$	Variable x is assigned some (new) strategy.
$b(x, a)$	Agent a is assigned the strategy carried by variable x .

$$\begin{aligned} \exists x_1 \exists x_2 \exists x_3 & b(a, x_1) b(b, x_2) b(c, x_3) \Big((\neg \Diamond o_a \longrightarrow \forall y \, b(y, a) \neg \Diamond o_a) \bigwedge \\ & (\neg \Diamond o_b \longrightarrow \forall y \, b(y, b) \neg \Diamond o_b) \bigwedge \\ & (\neg \Diamond o_c \longrightarrow \forall y \, b(y, c) \neg \Diamond o_c) \Big) \end{aligned}$$



Semantics of SL

Semantic context:

- $\xi_V : Var \longrightarrow Strat.$
- $\xi_{Ag} : Ag \longrightarrow Strat.$
- $out(\xi_V, \xi_{Ag})$: the (unique) outcome of the strategies.

$(\xi_V, \xi_{Ag}) \models \Diamond \phi$	if $out(\xi_{Ag}) \models \Diamond \phi$
$(\xi_V, \xi_{Ag}) \models \exists x \phi$	if $\exists \sigma \in Strat$ with $(\xi_V[x \rightarrow \sigma], \xi_{Ag}) \models \phi$
$(\xi_V, \xi_{Ag}) \models b(a, x) \phi$	if $(\xi_V, \xi_{Ag}[a \rightarrow \xi_V(x)]) \models \phi$



Tricks with SL

- Same strategy can be reused by several agents.

$$\exists x \forall y (b(a, x) b(b, y) \Diamond win \wedge b(b, x) b(a, y) \neg win)$$



Tricks with SL

- Same strategy can be reused by several agents.

$$\exists x \forall y (b(a, x) b(b, y) \Diamond win \wedge b(b, x) b(a, y) \neg win)$$

- ... and even at several time instances.
- Requires some symmetry of the "protocol" (actions available for each agent at each state).



Part 4: Strategy Logic and Rational Verification

4.2 Some examples



Majority voting example – qualitative NE

Consider a majority election system:

- 3 candidates: Macron, Mélenchon, Philippe.
- Parliament composed of 2 seats.
- 8 voters.
- Lexicographic order in case of ties.

Individual **objectives**:

- 5 voters want Mélenchon to win.
- 2 voters want Macron to win.
- 1 voter hates Macron.

Is there a NE in which Macron does not get elected?



Qualitative NE example

Is there a NE in which Macron does not get elected?

$$\exists x_1 \dots \exists x_8 \models (a_1, x_1) \dots \models (a_8, x_8)$$

$$\left(\Diamond \neg MacElected \wedge \right. \\ \left. (\neg \Diamond \neg MacElected \longrightarrow \forall y \models (y, a_1) \neg \Diamond \neg MacElected) \wedge \right. \\ \left. \right)$$



Qualitative NE example

Is there a NE in which Macron does not get elected?

$$\exists x_1 \dots \exists x_8 \, b(a_1, x_1) \dots b(a_8, x_8)$$

$$\begin{aligned} & \left(\Diamond \neg MacElected \bigwedge \right. \\ & \quad \left(\neg \Diamond \neg MacElected \longrightarrow \forall y \, b(y, a_1) \neg \Diamond \neg MacElected \right) \bigwedge \\ & \quad \left(\neg \Diamond MelElected \longrightarrow \forall y \, b(y, a_2) \neg \Diamond MelElected \right) \bigwedge \\ & \quad \dots \\ & \quad \left(\neg \Diamond MelElected \longrightarrow \forall y \, b(y, a_6) \neg \Diamond MelElected \right) \bigwedge \\ & \left. \right) \end{aligned}$$



Qualitative NE example

Is there a NE in which Macron does not get elected?

$$\exists x_1 \dots \exists x_8 \models (a_1, x_1) \dots \models (a_8, x_8)$$

$$\begin{aligned}
& \left(\Diamond \neg MacElected \bigwedge \right. \\
& \quad \left(\neg \Diamond \neg MacElected \longrightarrow \forall y \models (y, a_1) \neg \Diamond \neg MacElected \right) \bigwedge \\
& \quad \left(\neg \Diamond MelElected \longrightarrow \forall y \models (y, a_2) \neg \Diamond MelElected \right) \bigwedge \\
& \quad \dots \\
& \quad \left(\neg \Diamond MelElected \longrightarrow \forall y \models (y, a_6) \neg \Diamond MelElected \right) \bigwedge \\
& \quad \left(\neg \Diamond MacElected \longrightarrow \forall y \models (y, a_7) \neg \Diamond MacElected \right) \bigwedge \\
& \quad \left(\neg \Diamond MacElected \longrightarrow \forall y \models (y, a_8) \neg \Diamond MacElected \right) \\
& \left. \right)
\end{aligned}$$



(Slightly) quantitative NE example

Prisoner's Dilemma:



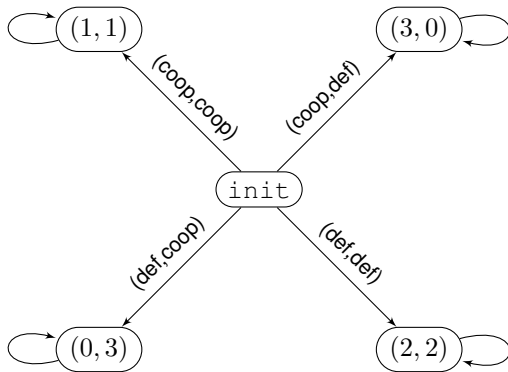
(Slightly) quantitative NE example

Prisoner's Dilemma:

- Two thugs are imprisoned and questioned separately.
- Police don't have enough evidence to convict them of serious crime, only 1 year lesser charge possible.
- If any of them confesses the more serious crime and the other doesn't, the defecting testifying prisoner goes free and the other gets 3 years sentence.
- But if both testify, each gets a 2 years sentence.

	B cooperates (with A)	B defects
A cooperates	(1, 1)	(3, 0)
A defects	(0, 3)	(2, 2)

Unique (Pure) Nash equilibrium: both defect.



Need boolean encoding of the payoff function.



Encoding NE for the Prisoners' Dilemma

	B cooperates (with B)	B defects
A cooperates	(1, 1)	(3, 0)
A defects	(3, 0)	(2, 2)

$$\varphi_{good} = \exists x_1 \exists x_2 \mathbf{b}(a_1, x_1) \mathbf{b}(a_2, x_2)$$

$$\begin{aligned}
 & \left(\Diamond (ag2pen1y \wedge ag2pen1y) \bigwedge \right. \\
 & (\Diamond ag1pen1y \longrightarrow \forall y \mathbf{b}(a_1, y) \Diamond (ag1pen1y \vee ag1pen2y \vee ag1pen3y)) \bigwedge \\
 & (\Diamond ag1pen2y \longrightarrow \forall y \mathbf{b}(a_1, y) \Diamond (ag1pen2y \vee ag1pen3y)) \bigwedge \\
 & (\Diamond ag1pen3y \longrightarrow \forall y \mathbf{b}(a_1, y) \Diamond ag1pen3y) \bigwedge \\
 & (\Diamond ag2pen1y \longrightarrow \forall y \mathbf{b}(a_2, y) \Diamond (ag2pen1y \vee ag2pen2y \vee ag2pen3y)) \bigwedge \\
 & (\Diamond ag2pen2y \longrightarrow \forall y \mathbf{b}(a_2, y) \Diamond (ag2pen2y \vee ag2pen3y)) \bigwedge \\
 & \left. (\Diamond ag2pen3y \longrightarrow \forall y \mathbf{b}(a_2, y) \Diamond ag2pen3y) \bigwedge \right)
 \end{aligned}$$

$$\Gamma \not\models \varphi_{good}$$



Dominant strategies

- Strategy σ_i is **dominant** for agent i if she gets a better outcome by playing σ_i than by playing any other strategy.

$$\forall \sigma' (out(\sigma') \leq out(\sigma'_{-i}, \sigma_i))$$

- $(\text{defect}, \text{defect})$ is a dominant strategy profile for Prisoner's Dilemma.



Dominant strategies

- Strategy σ_i is **dominant** for agent i if she gets a better outcome by playing σ_i than by playing any other strategy.

$$\forall \sigma' (out(\sigma') \leq out(\sigma'_{-i}, \sigma_i))$$

- $(\text{defect}, \text{defect})$ is a dominant strategy profile for Prisoner's Dilemma.
- Qualitative variant with LTL objectives $\varphi_1, \dots, \varphi_n$:

$$\varphi(x) := \bigwedge_{i \in Ag} \forall \bar{y} (b(\bar{y}, Ag) \varphi_i \rightarrow b(\bar{y}_{-i}, Ag \setminus i) b(x_i, i) \varphi_i)$$



Part 4: Strategy Logic and Rational Verification

4.3 Model-checking SL



Model-checking SL

Theorem

Model-checking SL with memoryful strategies is decidable in nonelementary time.

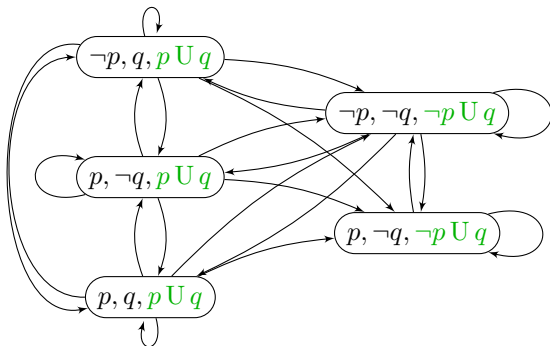
- Proof via encoding into MSO over trees.
- Each strategy variable can be encoded as a 2nd order variable.
- Then translated into (tree) automata.
- Negation requires determinization = exponential explosion.
- Model-checking formula with n quantifier alternations over a game structure Γ produces an automaton with size $\underbrace{2 \dots 2^{|\Gamma|}}_{n+1 \text{ times}}$.
- Result holds even for formulas with "immediate assignment" of strategies to agents (and no strategy reuse).



Automata for formulas

Automaton for $p \mathbf{U} q$:

- Fixpoint expansion $p \mathbf{U} q = q \vee (p \wedge \bigcirc p \mathbf{U} q)$:



Need a **fairness** condition: Büchi, Muller, parity, etc.



Memoryless SL

Theorem

Model-checking SL with memoryless strategies is PSPACE-complete.

- Same complexity as CTLK, ATLK* (with memoryless strategies), etc.
- Algorithm: perform strategy (re)assignment and backtrack in case of failure.



Relevant subproblems of SL model-checking

Rational verification

Some (qualitative) NE satisfy some desired property:

- E-Nash: $\exists x_1 \dots \exists x_n \, b(a_1, x_1) \dots b(a_n, x_n) \, (\varphi_0 \wedge NE(\varphi_1, \dots, \varphi_n))$.
 - A-Nash: $\forall x_1 \dots \forall x_n \, b(a_1, x_1) \dots b(a_n, x_n) \, (NE(\varphi_1, \dots, \varphi_n) \rightarrow \varphi_0)$.
-
- Bypass the MSO technique by encoding into zero-sum two-player games ("suspect game").
 - 2-EXPTIME-complete.



Relevant subproblems: Rational Synthesis

One agent is the "controller" and her objective φ_0 is essential, the others behave rationally.



Relevant subproblems: Rational Synthesis

One agent is the "controller" and her objective φ_0 is essential, the others behave rationally.

Cooperative Rational Synthesis

Synthesize a strategy for 0 such that one of the NE for the remaining agents satisfy φ_0 :

$$\exists x_0 \exists x_1 \dots \exists x_n \ b(a_0, x_0) b(a_1, x_1) \dots b(a_n, x_n) \ (\varphi_0 \wedge NE(\varphi_1, \dots, \varphi_n))$$

Almost the same as E-Nash.



Relevant subproblems: Rational Synthesis

One agent is the "controller" and her objective φ_0 is essential, the others behave rationally.

Cooperative Rational Synthesis

Synthesize a strategy for 0 such that one of the NE for the remaining agents satisfy φ_0 :

$$\exists x_0 \exists x_1 \dots \exists x_n \ b(a_0, x_0) b(a_1, x_1) \dots b(a_n, x_n) \ (\varphi_0 \wedge NE(\varphi_1, \dots, \varphi_n))$$

Almost the same as E-Nash.

Non-cooperative Rational Synthesis

Synthesize a strategy for 0 such that **all** the NE for the remaining agents satisfy φ_0 :

$$\exists x_0 \forall x_1 \dots \forall x_n \ b(a_0, x_0) b(a_1, x_1) \dots b(a_n, x_n) \ (NE(\varphi_1, \dots, \varphi_n) \rightarrow \varphi_0)$$

- $NE(\varphi_1, \dots, \varphi_n)$ is played in the $\xi(x_0)$ -fixed game.



Relevant subproblems: Rational Synthesis

One agent is the "controller" and her objective φ_0 is essential, the others behave rationally.

Cooperative Rational Synthesis

Synthesize a strategy for 0 such that one of the NE for the remaining agents satisfy φ_0 :

$$\exists x_0 \exists x_1 \dots \exists x_n \ b(a_0, x_0) b(a_1, x_1) \dots b(a_n, x_n) \ (\varphi_0 \wedge NE(\varphi_1, \dots, \varphi_n))$$

Almost the same as E-Nash.

Non-cooperative Rational Synthesis

Synthesize a strategy for 0 such that **all** the NE for the remaining agents satisfy φ_0 :

$$\exists x_0 \forall x_1 \dots \forall x_n \ b(a_0, x_0) b(a_1, x_1) \dots b(a_n, x_n) \ (NE(\varphi_1, \dots, \varphi_n) \rightarrow \varphi_0)$$

- $NE(\varphi_1, \dots, \varphi_n)$ is played in the $\xi(x_0)$ -fixed game.

Both 2-EXPTIME-complete problems.



Part 4: Strategy Logic and Rational Verification

4.4 Imperfect information



SL with imperfect information

- Agentless **observation relations**.
- Uniform strategy w.r.t. some observation relation.
- Strategy assignment to an agent assigns at the same time the observation relation.



Decidable cases for ATL_i/SL_i with perfect recall

- **Memoryless** variant: model-checking is PSPACE-complete.



Decidable cases for ATLi/SLi with perfect recall

- **Memoryless** variant: model-checking is PSPACE-complete.
- SLi model-checking for agents with perfect recall is undecidable (as for ATLi).



Decidable cases for ATLi/SLi with perfect recall

- **Memoryless** variant: model-checking is PSPACE-complete.
- SLi model-checking for agents with perfect recall is undecidable (as for ATLi).
- Broadcast: all agents observe the same set of environment variables (obsvars): ATLi and SLi.



Decidable cases for ATLi/SLi with perfect recall

- **Memoryless** variant: model-checking is PSPACE-complete.
- SLi model-checking for agents with perfect recall is undecidable (as for ATLi).
- Broadcast: all agents observe the same set of environment variables (obsvars): ATLi and SLi.
- Hierarchical knowledge: if $q \sim_a q'$ then $q \sim_b q'$: ATLi only.



Decidable cases for ATLi/SLi with perfect recall

- **Memoryless** variant: model-checking is PSPACE-complete.
- SLi model-checking for agents with perfect recall is undecidable (as for ATLi).
- Broadcast: all agents observe the same set of environment variables (obsvars): ATLi and SLi.
- Hierarchical knowledge: if $q \sim_a q'$ then $q \sim_b q'$: ATLi only.
- Finite knowledge gaps (between reaching common knowledge) : LTL synthesis (non-nesting ATLi).



Decidable cases for ATLi/SLi with perfect recall

- **Memoryless** variant: model-checking is PSPACE-complete.
- SLi model-checking for agents with perfect recall is undecidable (as for ATLi).
- Broadcast: all agents observe the same set of environment variables (obsvars): ATLi and SLi.
- Hierarchical knowledge: if $q \sim_a q'$ then $q \sim_b q'$: ATLi only.
- Finite knowledge gaps (between reaching common knowledge) : LTL synthesis (non-nesting ATLi).
- Absence of information forks, coalition-cast: all communications from adversaries to coalition members are observed by everyone in the coalition (ATLi only).



Part 4: Strategy Logic and Rational Verification

4.5 MCMAS-SLK



MCMAS adaptation for SL: MCMAS-SLK

- Same syntax as MCMAS for game structures.
- Specific syntax for formulas:

```
«x0» «x1» «x2» (Environment,x0) (Ag1,x1) (Ag2,x2)
( (F (ag1pen2yrs and ag2pen2yrs))
and
(F ag1pen1yr -> [[y1]] (Ag1,y1)
F (ag1pen1yr or ag1pen2yrs or ag1pen3yrs)) and
(F ag1pen2yrs -> [[y1]] (Ag1,y1)
F (ag1pen2yrs or ag1pen3yrs)) and
(F ag1pen3yrs -> [[y1]] (Ag1,y1) F (ag1pen3yrs)) and

(F ag2pen1yr -> [[y2]] (Ag2,y2)
F (ag2pen1yr or ag2pen2yrs or ag2pen3yrs)) and
(F ag2pen2yrs -> [[y2]] (Ag2,y2)
F (ag2pen2yrs or ag2pen3yrs)) and
(F ag2pen3yrs -> [[y2]] (Ag2,y2) F (ag2pen3yrs))
);
```



Part 5: Model Reductions

- 5.1 Explosions and Reductions
- 5.2 Bisimulation-Based Reduction
- 5.3 Partial Order Reduction
- 5.4 May/Must Abstraction
- 5.5 Practical May/Must Abstraction
- 5.6 Tool Presentation: STV+Reductions

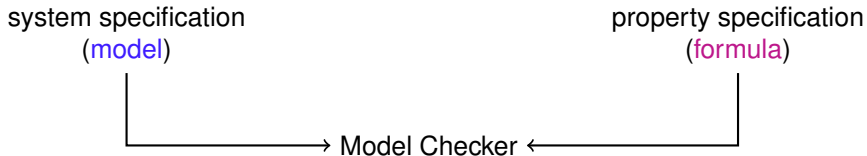


Part 5: Model Reductions

5.1 Explosions and Reductions

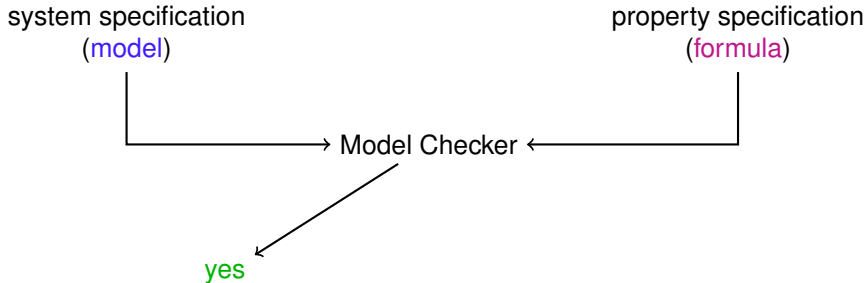


Idea of Model Checking



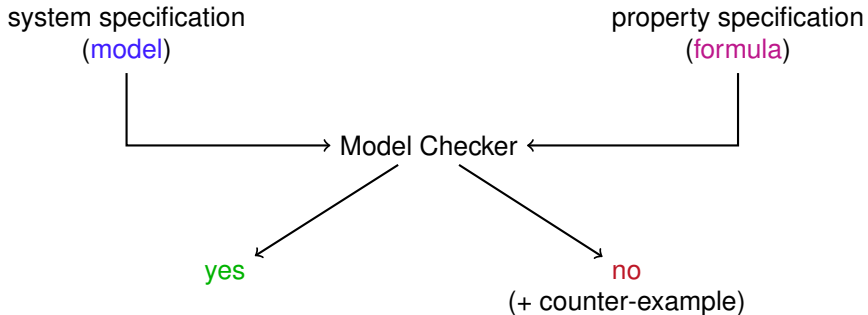


Idea of Model Checking





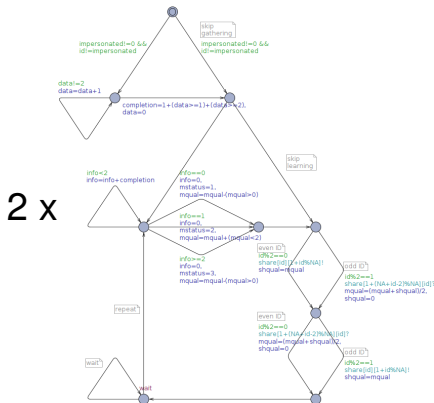
Idea of Model Checking





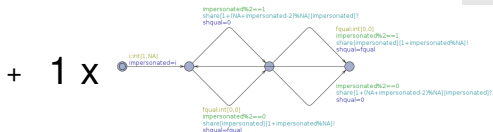
Model Specification: Example (SAI)

Honest AI



2 x

Impersonator



+ 1 x



Model Checking in Practice

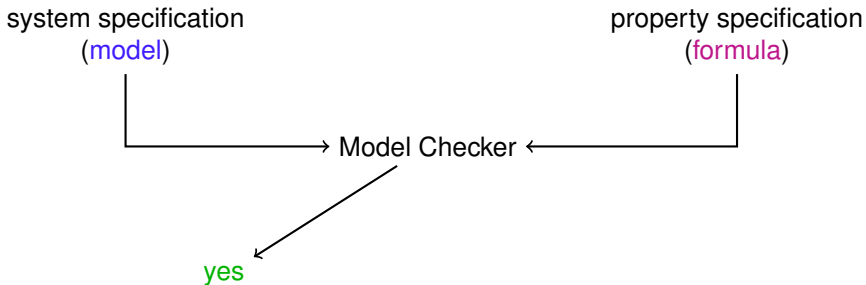
system specification
(model)

property specification
(formula)



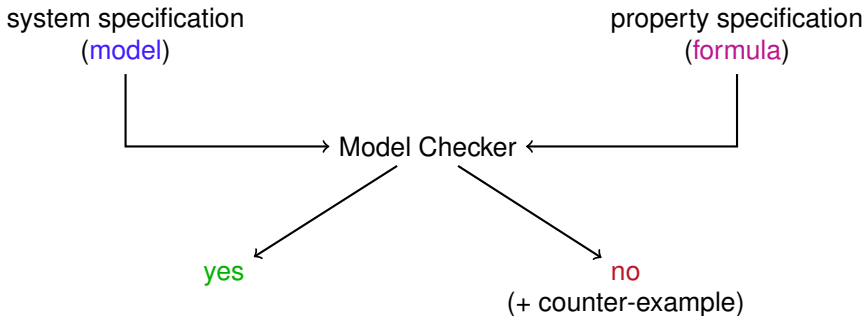


Model Checking in Practice



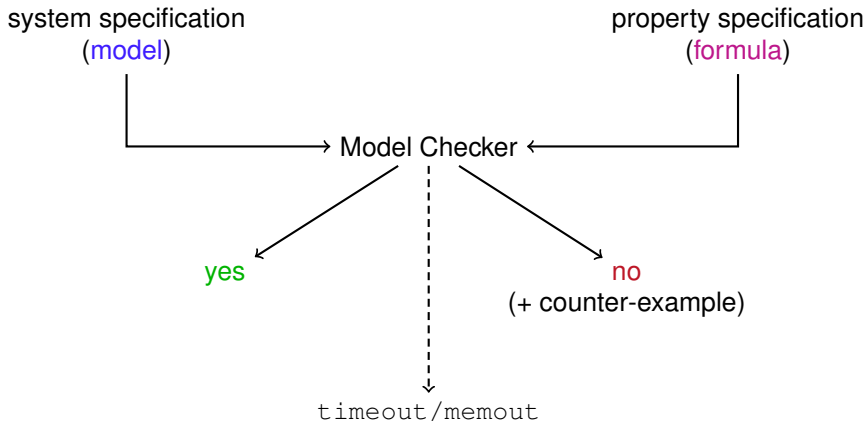


Model Checking in Practice



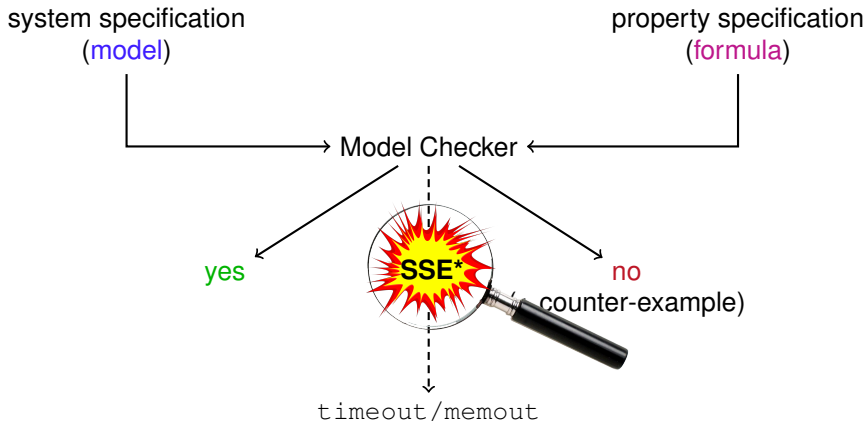


Model Checking in Practice



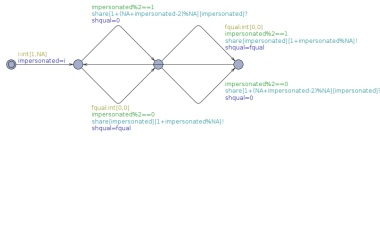


Model Checking in Practice



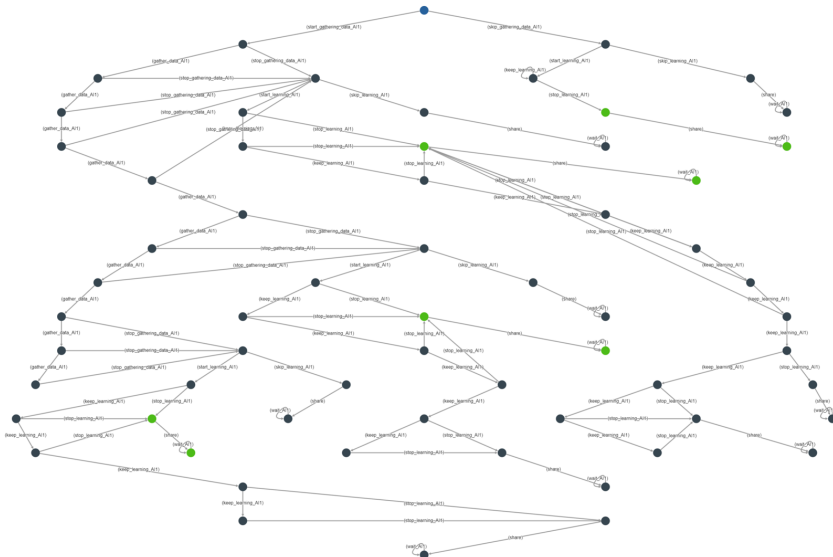
* : State/Transition/Strategy Space Explosion

Impersonator



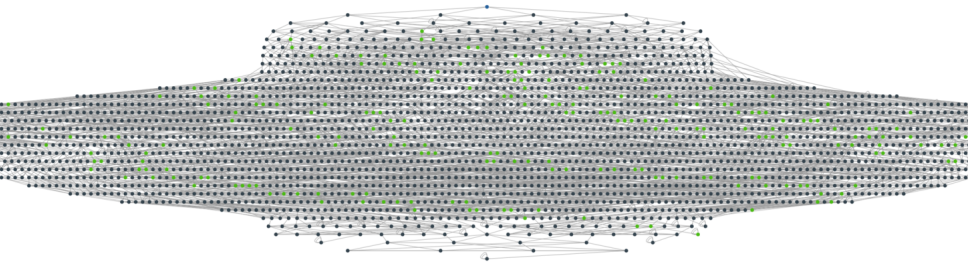


Model of SAI with One Honest Agent





Model of SAI with Two Honest Agents





Model of SAI with Two Honest Agents and One Impersonator



Model Reductions

Factors of complexity:

- Size of the model $\leadsto$ “program complexity”
- Length of the formula $\leadsto$ “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** $\leadsto$ “**program complexity**”
- Length of the formula $\leadsto$ “**formula complexity**”

Want to make the verification feasible? Use smaller models!

$\leadsto$ **Model reductions**



Model Reductions





Model Reductions





Model Reductions

Main idea

Given model M and a class of properties/formulas Φ , come up with a **smaller model** M' that is **equivalent to** M with respect to Φ (in a suitable, precisely defined sense).



Model Reductions

Main idea

Given model M and a class of properties/formulas Φ , come up with a **smaller model** M' that is **equivalent to** M with respect to Φ (in a suitable, precisely defined sense).

Ideally, M' should be generated **directly from the model specification**.



Part 5: Model Reductions

5.2 Bisimulation-Based Reduction



Attacking the complexity by modifying the system

- Are different models equally expressive?



Attacking the complexity by modifying the system

- Are different models equally expressive?
- How to modify a model while keeping some formula satisfied?

Variant 1 : **(bi)simulation**.



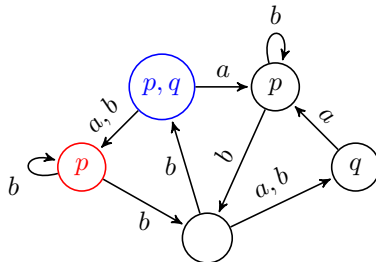
Modal logics and labeled transition systems

- A basic modal language: $\phi ::= p \mid \neg\phi \mid \phi \wedge \phi \mid \langle a \rangle \phi$
with $a \in \Sigma$, $p \in \Pi$.
- Dual operator $[a]\phi = \neg\langle a \rangle\neg\phi$.



Modal logics and labeled transition systems

- A basic modal language: $\phi ::= p \mid \neg\phi \mid \phi \wedge \phi \mid \langle a \rangle \phi$
with $a \in \Sigma$, $p \in \Pi$.
- Dual operator $[a]\phi = \neg\langle a \rangle\neg\phi$.
- Semantics : labeled transition systems.



-  $\models \langle b \rangle p$.
-  $\models [a][b]\neg q$.

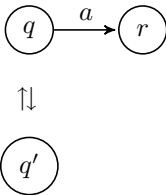


Modal logic and Bisimulations

Definition 5.1 (Bisimulation)

Consider two transition systems $T = (Q, \Sigma, \xrightarrow{\cdot}, \pi)$ and $T' = (Q', \Sigma, \xrightarrow{\cdot}, \pi')$. A relation $\rightleftharpoons \subseteq Q \times Q'$ is a **bisimulation** if for every $q \rightleftharpoons q'$,

- 1 [Label] $\pi(q) = \pi'(q')$
- 2 [Zig] $\forall a \in \Sigma, \forall q' \in Q, \text{ if } q \xrightarrow{a} r \text{ then } \exists q' \xrightarrow{a} r' \text{ with } r \rightleftharpoons r'.$



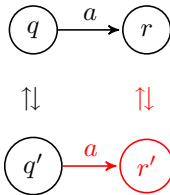


Modal logic and Bisimulations

Definition 5.1 (Bisimulation)

Consider two transition systems $T = (Q, \Sigma, \dot{\rightarrow}, \pi)$ and $T' = (Q', \Sigma, \dot{\rightarrow}, \pi')$. A relation $\rightleftharpoons \subseteq Q \times Q'$ is a **bisimulation** if for every $q \rightleftharpoons q'$,

- 1 [Label] $\pi(q) = \pi'(q')$
- 2 [Zig] $\forall a \in \Sigma, \forall q' \in Q, \text{ if } q \xrightarrow{a} r \text{ then } \exists q' \xrightarrow{a} r' \text{ with } r \rightleftharpoons r'.$



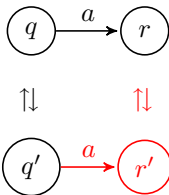


Modal logic and Bisimulations

Definition 5.1 (Bisimulation)

Consider two transition systems $T = (Q, \Sigma, \xrightarrow{\cdot}, \pi)$ and $T' = (Q', \Sigma, \xrightarrow{\cdot}, \pi')$. A relation $\rightleftharpoons \subseteq Q \times Q'$ is a **bisimulation** if for every $q \rightleftharpoons q'$,

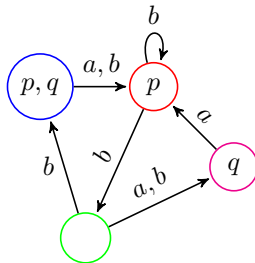
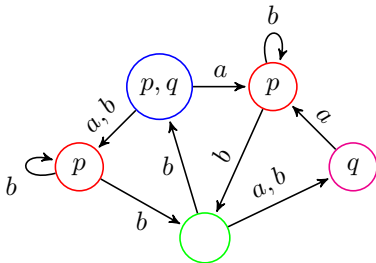
- 1 [Label] $\pi(q) = \pi'(q')$
- 2 [Zig] $\forall a \in \Sigma, \forall q' \in Q', \text{ if } q \xrightarrow{a} r \text{ then } \exists q' \xrightarrow{a} r' \text{ with } r \rightleftharpoons r'.$
- 3 [Zag] $\forall a \in \Sigma, \forall r' \in Q', \text{ if } q' \xrightarrow{a} r' \text{ then } \exists q \xrightarrow{a} r \text{ with } r \rightleftharpoons r'.$



When we only have "Label" and "Zig", we speak of **simulation**.



Labeled bisimulation





Computing the coarsest simulation

Greatest fixpoint computation:

$$\begin{aligned} B_0 &= \{(q, q') \in Q \times Q' \mid \pi(q) = \pi(q')\} \\ B_{n+1} &= B_n \setminus \{(q, q') \mid \exists a \in Act, r \in Q \text{ with } q \xrightarrow{a} r \\ &\quad \text{but } \forall r' \in Q' \text{ with } q' \xrightarrow{a} r' \text{ we have } (r, r') \notin B_n\} \end{aligned}$$

Symmetric variant for bisimulation.



Hennessy-Milner Theorem

Theorem 5.2

- Given two labeled transition systems $T = (Q, \Sigma, \dot{\rightarrow}, \pi)$ and $T' = (Q', \Sigma, \dot{\rightarrow}, \pi')$ and a bisimulation $\rightleftharpoons \subseteq Q \times Q'$, for any formula φ and pair of states $q \in Q, q' \in Q'$,

$$q \models \varphi \text{ if and only if } q' \models \varphi$$



Hennessy-Milner Theorem

Theorem 5.2

- Given two labeled transition systems $T = (Q, \Sigma, \dot{\rightarrow}, \pi)$ and $T' = (Q', \Sigma, \dot{\rightarrow}, \pi')$ and a bisimulation $\rightleftharpoons \subseteq Q \times Q'$, for any formula φ and pair of states $q \in Q, q' \in Q'$,

$$q \models \varphi \text{ if and only if } q' \models \varphi$$

- Given a labeled transition system $T = (Q, \Sigma, \dot{\rightarrow}, \pi)$, consider the relation $\rightleftharpoons \subseteq Q \times Q$ defined by

$$q \rightleftharpoons r \text{ if and only if for all formula } \varphi, q \models \varphi \text{ iff } r \models \varphi$$

Then $\rightleftharpoons$ is a bisimulation (the coarsest).



The simulation game

A safety game between Duplicator and Spoiler:

- Spoiler positions: $(q, q') \in Q \times Q'$.
- Spoiler chooses an action a , state r and transition $q \xrightarrow{a} r$.
- Game position switches to a Duplicator position (q, q', a, r) .
- Duplicator must answer with state r' and transition $q' \xrightarrow{a} r'$.
- ... in which case game advances to location (r, r') .
- Failure to produce this transition = Spoiler wins.
- Locations (q, q') with $\pi(q) \neq \pi(q')$ are also winning for Spoiler.
- **Safety** game: Duplicator must avoid bad states.



The simulation game

A safety game between Duplicator and Spoiler:

- Spoiler positions: $(q, q') \in Q \times Q'$.
- Spoiler chooses an action a , state r and transition $q \xrightarrow{a} r$.
- Game position switches to a Duplicator position (q, q', a, r) .
- Duplicator must answer with state r' and transition $q' \xrightarrow{a} r'$.
- ... in which case game advances to location (r, r') .
- Failure to produce this transition = Spoiler wins.
- Locations (q, q') with $\pi(q) \neq \pi(q')$ are also winning for Spoiler.
- **Safety** game: Duplicator must avoid bad states.

Bisimulation variant: Spoiler may choose state in Q' instead of Q .



The simulation game and Hennessy-Milner theorem

Theorem

- Duplicator wins the simulation game if and only if there exists a simulation between the two given LTS.
- Spoiler wins the simulation game starting from location (q, r) if and only if there exists a formula φ which is satisfied in q but not in r .



Other instances of a Hennessy-Milner theorem

- CTL, unlabeled transition systems and unlabeled bisimulation.



Other instances of a Hennessy-Milner theorem

- CTL, unlabeled transition systems and unlabeled bisimulation.
- Modal mu-calculus and labeled bisimulation.



Other instances of a Hennessy-Milner theorem

- CTL, unlabeled transition systems and unlabeled bisimulation.
- Modal mu-calculus and labeled bisimulation.
- Epistemic logic.



Other instances of a Hennessy-Milner theorem

- CTL, unlabeled transition systems and unlabeled bisimulation.
- Modal mu-calculus and labeled bisimulation.
- Epistemic logic.
- Dynamic logic.

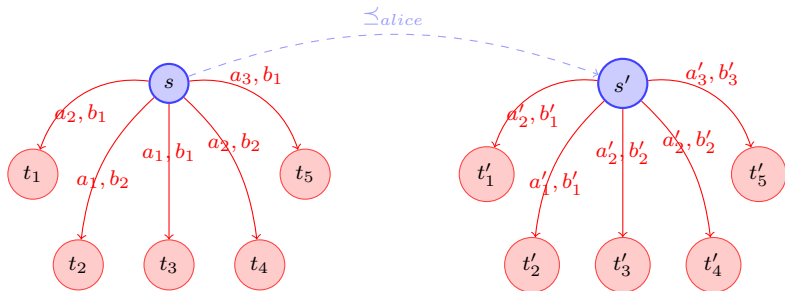


Other instances of a Hennessy-Milner theorem

- CTL, unlabeled transition systems and unlabeled bisimulation.
- Modal mu-calculus and labeled bisimulation.
- Epistemic logic.
- Dynamic logic.
- Alternating temporal logic, game structures and **alternating refinement relations**.
- **ATL with imperfect information**.
- Fragments of Strategy logic.

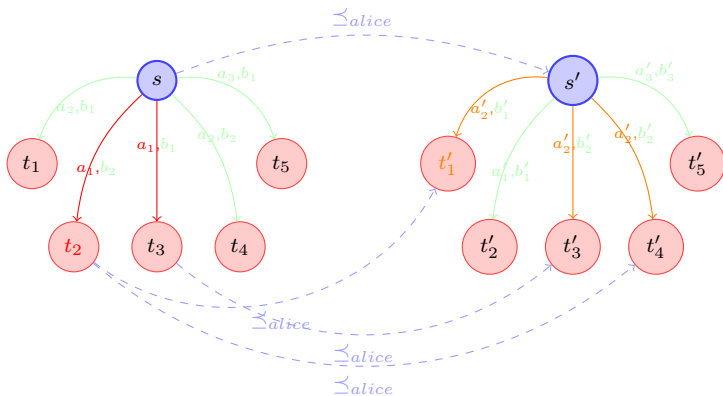


Alternating simulation





Alternating simulation



$$\forall \alpha \exists \alpha' \forall s' \xrightarrow{\alpha'} t' \exists s \xrightarrow{\alpha} t \text{ with } t \preceq_{alice} t'$$



Alternating simulations and ATL

Theorem (ATL validity preservation)

Given two game structures Γ, Γ' , assume there exists a simulation $\preceq_A$ with $q \preceq_A q'$ for some $q \in Q, q' \in Q'$. Then for each formula φ employing only $\langle\langle A \rangle\rangle$,

$$\Gamma, q \models \varphi \text{ implies } \Gamma', q' \models \varphi$$



Alternating simulations and ATL

Theorem (ATL validity preservation)

Given two game structures Γ, Γ' , assume there exists a simulation $\preceq_A$ with $q \preceq_A q'$ for some $q \in Q, q' \in Q'$. Then for each formula φ employing only $\langle\langle A \rangle\rangle$,

$$\Gamma, q \models \varphi \text{ implies } \Gamma', q' \models \varphi$$

Based on the following:

Lemma (Strategy transfer)

In the above settings:

$$\forall \sigma_A \exists \sigma'_A \pi_{Atoms}out(q', \sigma'_A) \subseteq \pi_{Atoms}out(q, \sigma_A)$$



(Bi)simulations for ATL

- Variant for turn-based (two-player) games.



(Bi)simulations for ATL

- Variant for turn-based (two-player) games.
- Variant for concurrent game structures and **A-restricted formulas** available (actually the original one!).



(Bi)simulations for ATL

- Variant for turn-based (two-player) games.
- Variant for concurrent game structures and **A-restricted formulas** available (actually the original one!).
 - Full Hennessy-Milner theorem w.r.t. formulas containing only the coalition operator $\langle\langle A \rangle\rangle$ (or its dual).



(Bi)simulations for ATL

- Variant for turn-based (two-player) games.
- Variant for concurrent game structures and **A-restricted formulas** available (actually the original one!).
 - Full Hennessy-Milner theorem w.r.t. formulas containing only the coalition operator $\langle\langle A \rangle\rangle$ (or its dual).
- Variant for any subset of protagonists – **too restrictive.**



(Bi)simulations for ATL

- Variant for turn-based (two-player) games.
- Variant for concurrent game structures and **A-restricted formulas** available (actually the original one!).
 - Full Hennessy-Milner theorem w.r.t. formulas containing only the coalition operator $\langle\langle A \rangle\rangle$ (or its dual).
- Variant for any subset of protagonists – **too restrictive**.
 - Full Hennessy-Milner theorem for full ATL.



(Bi)simulations for ATL

- Variant for turn-based (two-player) games.
- Variant for concurrent game structures and **A-restricted formulas** available (actually the original one!).
 - Full Hennessy-Milner theorem w.r.t. formulas containing only the coalition operator $\langle\langle A \rangle\rangle$ (or its dual).
- Variant for any subset of protagonists – **too restrictive**.
 - Full Hennessy-Milner theorem for full ATL.
- Variant for $SL[1G]$ (perfect information).
 - Half Hennessy-Milner theorem available.



(Bi)simulations for ATL with imperfect information

Given are:

- M, M' : two iCGS's, sharing the set of agents $\mathbb{A}_{\text{gt}}$ and the set of atoms AP
- coalition $A \subseteq \mathbb{A}_{\text{gt}}$
- relation $\Rightarrow_A \subseteq S \times S'$ between the states of M and M' .



(Bi)simulations for ATL with imperfect information

Given are:

- M, M' : two iCGS's, sharing the set of agents $\mathbb{A}_{gt}$ and the set of atoms AP
- coalition $A \subseteq \mathbb{A}_{gt}$
- relation $\Rightarrow_A \subseteq S \times S'$ between the states of M and M' .

Strategy simulator

A **simulator of partial strategies for coalition A with respect to $\Rightarrow_A$** is any family of functions

$$ST = \{ST_{C_A(q), C_A(q')} : PStr_A(C_A(q)) \rightarrow PStr_A(C_A(q')) \mid q \Rightarrow_A q'\}.$$

The idea is that $ST_{C_A(q), C_A(q')}$ “transforms” each partial strategy σ_A that works on the neighborhood of q in model M into a corresponding strategy σ'_A that works on the neighborhood of q' in model M' .



(Bi)simulation for ATL with imperfect information

Simulation for ATL_{ir}

$\Rightarrow_A \subseteq S \times S'$ is a **simulation for A** iff there exists a simulator of partial strategies ST such that $q \Rightarrow_A q'$ implies the following:

- 1 $\pi(q) = \pi'(q')$;
- 2 For every $i \in A$ and $r' \in S'$, if $q' \sim'_i r'$ then for some $r \in S$ we have that $q \sim_i r$ and $r \Rightarrow_A r'$.
- 3 For any states $r \in C_A(q)$ and $r' \in C'_A(q')$ such that $r \Rightarrow_A r'$, every partial strategy $\sigma_A \in PStr_A(C_A(q))$, and every state $s' \in succ(r', ST(\sigma_A))$, there exists a state $s \in succ(r, \sigma_A)$ such that $s \Rightarrow_A s'$.

Bisimulation for ATL_{ir}

A relation $\Leftrightarrow_A$ is a **bisimulation for A** iff both $\Rightarrow_A$ and $\Rightarrow_A^{-1} = \{(q', q) \mid q \Rightarrow_A q'\}$ are simulations.



(Bi)simulations for ATL with imperfect information

Preservation Theorem for ATL_{ir} (Belardinelli, Condurache, Dima, Jamroga, Knapik 2021)

If $\Leftrightarrow_A$ is a bisimulation for A and $q \Leftrightarrow_A q'$, then for every A -formula φ ,

$$(M, q) \models \varphi \quad \text{if and only if} \quad (M', q') \models \varphi.$$



(Bi)simulations for ATL with imperfect information

Preservation Theorem for ATL_{ir} (Belardinelli, Condurache, Dima, Jamroga, Knapik 2021)

If $\Leftrightarrow_A$ is a bisimulation for A and $q \Leftrightarrow_A q'$, then for every A -formula φ ,

$$(M, q) \models \varphi \quad \text{if and only if} \quad (M', q') \models \varphi.$$

Corollary

If $\Leftrightarrow$ is a bisimulation for every $A \subseteq \mathbb{A}gt$, and $q \Leftrightarrow q'$, then for every formula φ ,

$$(M, q) \models \varphi \quad \text{if and only if} \quad (M', q') \models \varphi.$$



(Bi)simulations for ATL with imperfect information

Preservation Theorem for ATL_{ir} (Belardinelli, Condurache, Dima, Jamroga, Knapik 2021)

If $\Leftrightarrow_A$ is a bisimulation for A and $q \Leftrightarrow_A q'$, then for every A -formula φ ,

$$(M, q) \models \varphi \quad \text{if and only if} \quad (M', q') \models \varphi.$$

Corollary

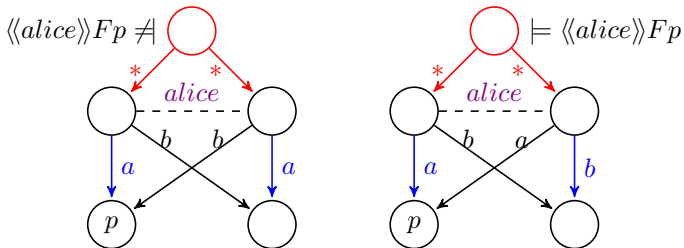
If $\Leftrightarrow$ is a bisimulation for every $A \subseteq \text{Agt}$, and $q \Leftrightarrow q'$, then for every formula φ ,

$$(M, q) \models \varphi \quad \text{if and only if} \quad (M', q') \models \varphi.$$

Full Hennessy-Milner theorem available for ATL with imperfect information, perfect recall and semantics based on common knowledge at the "starting point" (Belardinelli, Dima, Malvone, Tiplea, 2020).



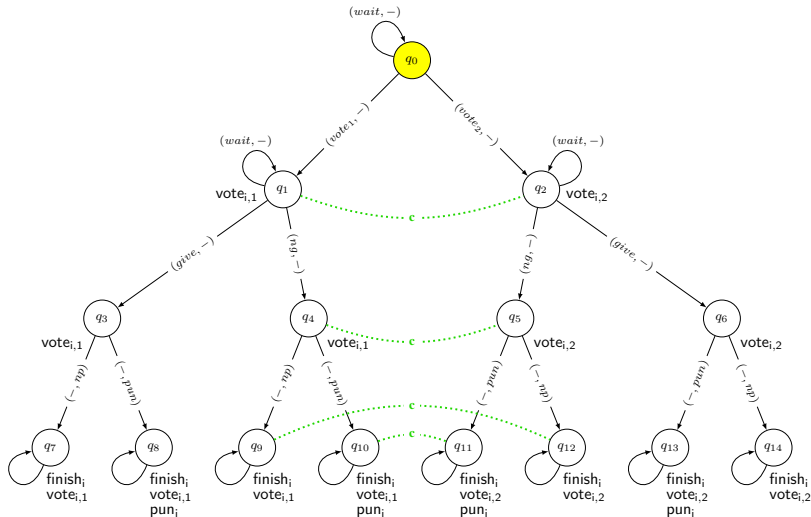
Alternating bisimulations for ATL not good for ATL_i



Note the two systems are even labeled bisimilar and epistemic-bisimilar!

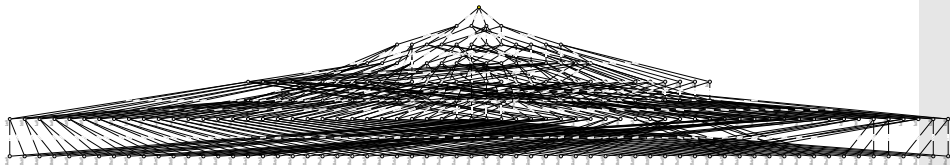


Reduction for Voting and Coercion



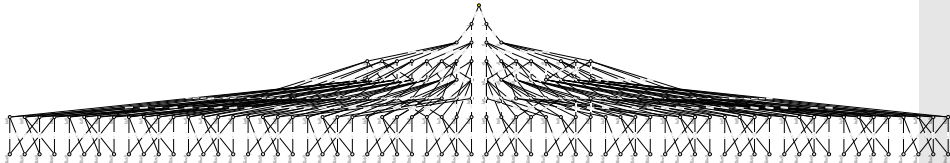


Reduction for Voting and Coercion





Reduction for Voting and Coercion





Bisimulation-Based Model Reduction: Summary

- Strong preservation result for the bisimulation (preserves the truth of all $ATL_{i,T}$ formulae)
- Can provide very significant reduction



Bisimulation-Based Model Reduction: Summary

- Strong preservation result for the bisimulation (preserves the truth of all $ATL_{i,T}$ formulae)
- Can provide very significant reduction **when you know where to look**



Bisimulation-Based Model Reduction: Summary

- Strong preservation result for the bisimulation (preserves the truth of all $ATL_{i,T}$ formulae)
- Can provide very significant reduction **when you know where to look**
- ...But: the conditions are also very strong $\leadsto$ limited applicability
- ...And the reduced model + bisimulation must be crafted by hand



Bisimulation-Based Model Reduction: Summary

- Strong preservation result for the bisimulation (preserves the truth of all $ATL_{i,T}$ formulae)
- Can provide very significant reduction **when you know where to look**
- ...But: the conditions are also very strong $\leadsto$ limited applicability
- ...And the reduced model + bisimulation must be crafted by hand
- No methodology/algorithm for **automated reduction**
 - But software development methodology based on refinement do exist.



Part 5: Model Reductions

5.3 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 **LTL**_○: take only **one** arbitrary interleaving of **independent** actions



Partial Order Reduction for LTL_ ○

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_

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 $LTL_{\circ}$

Theorem (Peled 1993)

For every formula φ of $LTL_{\circ}$:

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



Partial Order Reduction for $LTL_{\circ}$

Theorem (Peled 1993)

For every formula φ of $LTL_{\circ}$:

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

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**_○ without nested strategic operators, interpreted over imperfect information strategies:

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



Partial Order Reduction for Strategic Abilities

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

For every formula φ of **ATL**_○ 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_○ interpreted over **perfect information strategies**.



Partial Order Reduction for Strategic Abilities

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

For every formula φ of **ATL**_○ without nested strategic operators, interpreted over **imperfect information strategies**:

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

Note also:

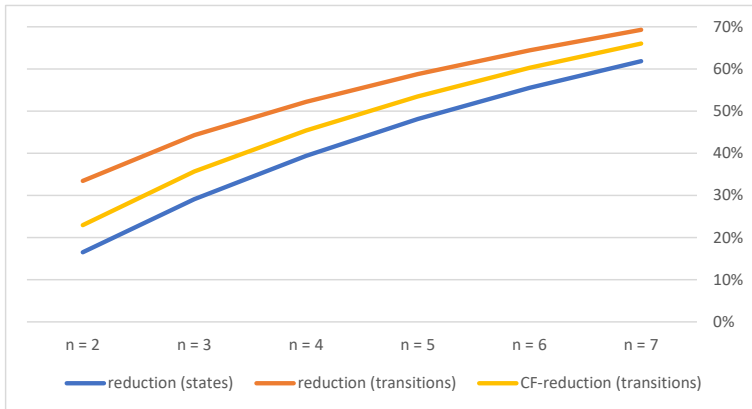
Theorem (Jamroga, Penczek, Sidoruk, Dembinski & Mazurkiewicz 2020)

The same does **not** hold for ATL_○ interpreted over **perfect information strategies**.

How good are the reductions in practice?

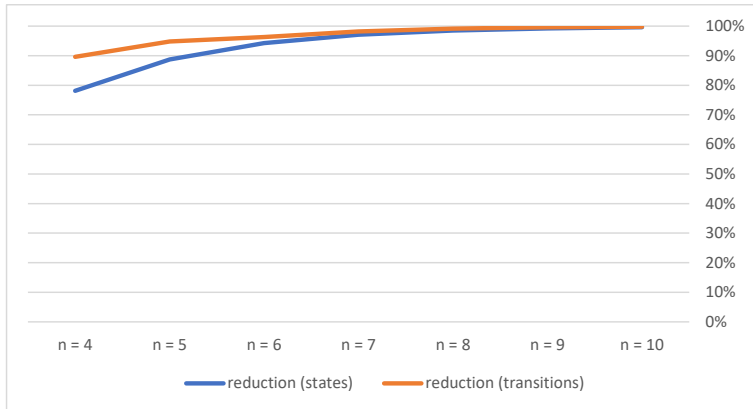


Experimental Results: Asynchronous Simple Voting





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: Model Reductions

5.4 May/Must Abstraction



State Abstraction

- Idea: cluster similar **concrete states** into **abstract states**



State Abstraction

- Idea: cluster similar **concrete states** into **abstract states**
- Formally: take **concrete model** M_C and define **equivalence relation** $\approx \subseteq St_C \times St_C$ that partitions the states in M_C
- Then, compute the abstract states as **equivalence classes of $\approx$**
- ...and fix the transitions in the abstract model in the right way



State Abstraction

- Idea: cluster similar **concrete states** into **abstract states**
- Formally: take **concrete model** M_C and define **equivalence relation** $\approx \subseteq St_C \times St_C$ that partitions the states in M_C
- Then, compute the abstract states as **equivalence classes of $\approx$**
- ...and fix the transitions in the abstract model in the right way (whatever it means)



State Abstraction

- Ideally, the resulting **abstract model** M_A should satisfy **exactly the same formulas** as M_C



State Abstraction

- Ideally, the resulting **abstract model** M_A should satisfy **exactly the same formulas** as M_C
- Lossless abstraction $\leadsto$ **abstraction refinement**
- Lossy abstraction $\leadsto$ **may/must abstraction**
- Lossy abstraction with refinement $\leadsto$ **CEGAR**



Abstraction Refinement for CTL*

We construct the **abstract model** M_A as follows



Abstraction Refinement for CTL*

We construct the **abstract model** M_A as follows

- 1 Abstract states $St_A =$ **equivalence classes of** $\approx$



Abstraction Refinement for CTL*

We construct the **abstract model** M_A as follows

- 1 Abstract states $St_A =$ **equivalence classes of $\approx$**
- 2 May transitions $\rightarrow^{may} =$ **transitions of type EE** (exists/exists)



Abstraction Refinement for CTL*

We construct the **abstract model** M_A as follows

- 1 Abstract states $St_A =$ **equivalence classes of $\approx$**
- 2 May transitions $\rightarrow^{may} =$ **transitions of type EE (exists/exists)**
- 3 Must transitions $\rightarrow^{must} =$ **transitions of type AE (forall/exists)**



Abstraction Refinement for CTL*

We construct the **abstract model** M_A as follows

- 1 Abstract states $St_A =$ **equivalence classes of $\approx$**
- 2 May transitions $\rightarrow^{may} =$ **transitions of type EE** (exists/exists)
- 3 Must transitions $\rightarrow^{must} =$ **transitions of type AE** (forall/exists)
- 4 If $\rightarrow^{may} = \rightarrow^{must}$ then return the abstract model, otherwise **split** the states in St_A on which $\rightarrow^{may} \neq \rightarrow^{must}$, and repeat from 2



Abstraction Refinement

Theorem 5.3

*The resulting set of abstract states St_A corresponds to the **largest bisimulation** on St_C .*



Abstraction Refinement

Theorem 5.3

*The resulting set of abstract states St_A corresponds to the **largest bisimulation** on St_C .*

Theorem 5.4

For every $\varphi \in \text{CTL}^$: $M_C, q_C \models \varphi$ **iff** $M_A, q_A \models \varphi$.*



Abstraction Refinement

Theorem 5.3

*The resulting set of abstract states St_A corresponds to the **largest bisimulation** on St_C .*

Theorem 5.4

For every $\varphi \in \text{CTL}^$: $M_C, q_C \models \varphi$ **iff** $M_A, q_A \models \varphi$.*

Problem: even the largest bisimulation is usually **too small!**
 $\leadsto$ insufficient reductions



May/Must Abstraction for CTL*

We construct **two** abstract models M_{may}, M_{must} :



May/Must Abstraction for CTL*

We construct **two** abstract models M_{may}, M_{must} :

- 1 M_{may} : abstract states St_A + may transitions $\rightarrow^{may}$ (i.e., transitions of type EE)
- 2 M_{must} : abstract states St_A + must transitions $\rightarrow^{must}$ (i.e., transitions of type AE)



May/Must Abstraction for CTL*

We construct **two** abstract models M_{may}, M_{must} :

- 1 M_{may} : abstract states St_A + may transitions $\rightarrow^{may}$ (i.e., transitions of type EE)
- 2 M_{must} : abstract states St_A + must transitions $\rightarrow^{must}$ (i.e., transitions of type AE)
- 3 M_{may} **overapproximates** the paths in M_C
- 4 M_{must} **underapproximates** the paths in M_C



May/Must Abstraction for CTL*

Theorem 5.5 (Preservation for ACTL*)

$$M_{may}, [q]_{\approx} \models \varphi \Rightarrow M_C, q_C \models \varphi \Rightarrow M_{must}, [q]_{\approx} \models \varphi$$



May/Must Abstraction for CTL*

Theorem 5.5 (Preservation for ACTL*)

$$M_{may}, [q]_{\approx} \models \varphi \Rightarrow M_C, q_C \models \varphi \Rightarrow M_{must}, [q]_{\approx} \models \varphi$$

Theorem 5.6 (Preservation for ECTL*)

$$M_{must}, [q]_{\approx} \models \varphi \Rightarrow M_C, q_C \models \varphi \Rightarrow M_{may}, [q]_{\approx} \models \varphi$$



May/Must Abstraction for CTL*

Theorem 5.5 (Preservation for ACTL*)

$$M_{may}, [q]_{\approx} \models \varphi \Rightarrow M_C, q_C \models \varphi \Rightarrow M_{must}, [q]_{\approx} \models \varphi$$

Theorem 5.6 (Preservation for ECTL*)

$$M_{must}, [q]_{\approx} \models \varphi \Rightarrow M_C, q_C \models \varphi \Rightarrow M_{may}, [q]_{\approx} \models \varphi$$

Note: de facto **3-valued interpretation** of transitions, strategies and formulas (**cannot/may/must**)



May/Must Abstraction for Strategy Logic

Definition 1.1 (Abstraction). Let $G = \langle Ag, St, s_0, Act, \tau, AP, V \rangle$ be a CGS, and $\approx \subset St \times St$ an equivalence relation. We write $[s]$ for the equivalence class of $\approx$ that contains s . The abstract model of G w.r.t. $\approx$ is defined as the 3-valued CGS $\mathcal{A}(G) = \langle \mathcal{A}(Ag), \mathcal{A}(St), \mathcal{A}(s_0), \mathcal{A}^{may}(Act), \mathcal{A}^{must}(Act), \mathcal{A}^{may}(\tau), \mathcal{A}^{must}(\tau), \mathcal{A}(AP), \mathcal{A}(V) \rangle$, with:

- $\mathcal{A}(Ag) = Ag$ and $\mathcal{A}(AP) = AP$.
- $\mathcal{A}(St) = \{[s] \mid s \in St\}$ with $\mathcal{A}(s_0) = [s_0]$.
- $\mathcal{A}^{may}(Act) = Act$.
- $\mathcal{A}^{may}(\tau) = \tau^{may} : \mathcal{A}(St) \times (\mathcal{A}^{may}(Act))^{| \mathcal{A}(Ag) |} \rightarrow 2^{\mathcal{A}(St)}$ such that $\tau^{may}([s], \vec{\alpha}) = \{[s_{succ}] \mid \exists s' \in [s] \exists s'_{succ} \in [s_{succ}] . s'_{succ} \in \tau(s', \vec{\alpha})\}$.
- $\mathcal{A}^{must}(\tau) = \tau^{must} : \mathcal{A}(St) \times (\mathcal{A}^{may}(Act))^{| \mathcal{A}(Ag) |} \rightarrow 2^{\mathcal{A}(St)}$ such that $\tau^{must}([s], \vec{\alpha}) = \{[s_{succ}] \mid \forall s' \in [s] \exists s'_{succ} \in [s_{succ}] . s'_{succ} \in \tau(s', \vec{\alpha})\}$.
- $\mathcal{A}^{must}(Act)$ is a maximal¹ set $Act^{must} \subseteq Act$ such that $\forall s \in St \forall \vec{\alpha} \in (Act^{must})^{| \mathcal{A}(Ag) |} . \tau^{must}([s], \vec{\alpha}) \neq \emptyset$. Note that a unique maximal set does not always exist. In such cases, a natural heuristic would be to choose the maximal subset of actions with the largest cardinality, breaking ties lexicographically in case there are still multiple solutions.
- $\mathcal{A}(V)([s], p) = \begin{cases} \top & \text{if } V(s', p) = \top \text{ for all } s' \in [s] \\ \perp & \text{if } V(s', p) = \perp \text{ for all } s' \in [s] \\ \mathbf{u} & \text{otherwise.} \end{cases}$



May/Must Abstraction for Strategy Logic

Theorem 5.7 (Preservation)

Let M be a concrete model and $\mathcal{A}(M)$ its abstraction induced by equivalence relation $\approx$. Then, for every formula φ in SL:

$$\begin{aligned}(\mathcal{A}(M) \models^3 \varphi) = \top &\Rightarrow (M \models^2 \varphi) = \top \\(\mathcal{A}(M), \models^3 \varphi) = \perp &\Rightarrow (M \models^2 \varphi) = \perp\end{aligned}$$

So, whenever verification of the abstract model returns a conclusive truth value, **it must be also correct for the concrete model.**



Part 5: Model Reductions

5.5 Practical May/Must Abstraction



Abstraction by Removal of Variables

- How to cluster the states? $\rightsquigarrow$ Art, not science...



Abstraction by Removal of Variables

- How to cluster the states? $\leadsto$ Art, not science...
- Domain expert needed!



Abstraction by Removal of Variables

- How to cluster the states? $\leadsto$ Art, not science...
- Domain expert needed!
- Thus, the procedure must be understandable and easy to use for people outside formal methods



Abstraction by Removal of Variables

- How to cluster the states? $\leadsto$ Art, not science...
- **Domain expert** needed!
- Thus, the procedure must be **understandable** and **easy to use** for people outside formal methods
- Additionally: generation of abstract model **without generating the concrete one!**



Abstraction by Removal of Variables

Idea:



Abstraction by Removal of Variables

Idea:

- The user selects a subset of **model variables** to be removed



Abstraction by Removal of Variables

Idea:

- The user selects a subset of **model variables** to be removed
- The algorithm computes **over-** and **underapproximation** of their **local domains**



Abstraction by Removal of Variables

Idea:

- The user selects a subset of **model variables** to be removed
- The algorithm computes **over-** and **underapproximation** of their **local domains**
- The variables are removed **from the representation**, and replaced by their (possibly multiple) values from the domain approximation



Abstraction by Removal of Variables

Theorem 5.8 (Preservation)

Let M be a concrete model. Then, for every formula φ in $ACTL^$:*

$$MAY(M) \models \varphi \quad \Rightarrow \quad M \models \varphi \quad \Rightarrow \quad MUST(M) \models \varphi$$



Part 5: Model Reductions

5.6 Tool Presentation: STV+Reductions



Model Reductions for MAS: STV+Reductions

Tool presentation: **STV+Reductions**



Part 6: Summary



Summary (very brief)

- We have tried to present a quick overview of specification and verification in **modal logics of strategic ability**
- Many issues left out... (feel free to ask!)
- Current focus: how to **use** (and expand) existing techniques
- ...e.g., for specification and verification of properties in e-voting protocols
- Interestingly, it triggers the invention and development of new theoretical results and algorithmic techniques!



THANK YOU