Logics for Strategic Reasoning in AI

Francesco Belardinelli

Imperial College London

Spring Term 2019

Course Content

Logics to reason about the behaviour of “rational” agents in multi-agent systems.

- what logics?
- what *reasoning*?
- what *rationality*?
- what multi-agent systems?

Course Outline I

Part I **Introduction to logic-based specification of MAS.**

- Agents and agent systems.
- Why logic? Modal logic.
- Examples: robots on a rescue mission, security of e-voting.
- Formal verification by model checking.

Part II **Reasoning about the evolution of systems.**

- Temporal logic: linear vs. branching time.
- Linear time logic: LTL.
- Branching-time logic: CTL.
- Decision problems: some complexity classes.

Course Outline II

Part IIIa **Specification of individual and coalitional abilities.**

- Temporal logic meets game theory.
- Logics for strategies: alternating-time temporal logic ATL.
- Properties of ATL.
- Agents, systems, games.

Part IIIb **Verification of strategic abilities (I).**

- Algorithms and complexity of verification for standard ATL.
- Some complexity proofs.

Course Outline III

Part IVa **Bringing time, knowledge and games together.**

- Alternating time temporal epistemic logic ATLK.
- Problems with ATLK.
- Imperfect information ATL: Schobbens' version and CSL.
- Levels of strategic ability under uncertainty.

Part IVb **Verification of strategic abilities (II).**

- Imperfect information.
- Taming the complexity.
- Between perception and recall.

Learning Outcomes

- Be able to model examples of multi-agent systems (MAS) into the framework of **concurrent game structures** (CGS).
- Be able to translate informal specifications of strategic abilities of agents in MAS, expressed in the English language, into formulas of **temporal logics** LTL and CTL, and **alternating-time temporal logic** ATL.
- Recognise the differences in modeling agents having perfect/imperfect **information** about the environment, as well as agents having perfect/imperfect **memory** of past events.
- Be able to apply the **model checking algorithms** underpinning the verification of ATL properties in concurrent game structures.

Practical Arrangements

- When? Thursdays 11.00 – 13.00
- Where? room 139, Huxley bld
- How long? 7 weeks until week 8 [Feb 28]
- Week 10 [Mar 14]: revision week
- Week 11 [Mar 21]: exam
- How? 1h lecture $+$ 1h tutorial (including some correction)
- Notes, tutorials, and coursework on CATE
- send me an email: `francesco.belardinelli@imperial.ac.uk`

Useful Reading

The course is self-contained (as possible).

Nonetheless, if you are interested in reading further:

- W. Jamroga (2015); Logical Methods for Specification and Verification of Multi-Agent Systems. Available for free at: `https://home.ipipan.waw.pl/w.jamroga/papers/jamroga15specifmas.pdf`
- Y. Shoham and K. Leyton-Brown (2009); *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press. (freely available)
- K. Baier, J. P. Katoen (2008); *Principles of Model Checking*. MIT Press. (freely available)
- M. Huth, M. Ryan (2004); *Logic in Computer Science: Modelling and Reasoning about Systems*, Cambridge University Press, 2004. (freely available)
- E. M. Clarke, O. Grumberg and D. A. Peled (1999); *Model Checking*. MIT Press.
- R. Alur, T. Henzinger, and O. Kupferman (2002); Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713.

Background

Again, the course is self-contained.

But, it draws on notions from:

- Modal Logic (H499): modal operators, relational (Kripke) structures.
- System Verification (303): temporal logics.
- Complexity (438): complexity classes of decision problems.

Acknowledgements

When preparing the course and notes, I used some materials courtesy of:

- [C. Baier, J. P. Katoen; 2008]: Part I and II.
- [W. Jamroga; 2015]: Part III and IV.

All mistakes are, of course, mine.

Part 1: Reasoning about Systems

Part 1: Reasoning about Systems

1.1 Multi-Agent Systems

Agents and MAS

- **Multi-agent system (MAS):** a system that involves several autonomous entities that act in the same environment
- These entities are called **agents**

- So, what is an agent precisely?
- No commonly accepted definition

For some authors, agents are:

- A paradigm for computation (distributed algorithms/protocols)
- A paradigm for design (agent-based models, interactions simulation)
- A paradigm for programming (agent-oriented programming, software agents: JADE, AgentSpeak, . . .)

Claim:

MAS is a (convenient) metaphor that induces a specific way of seeing the world.

Motivating Example: Rescue Robots

Scenario: Robots on a Rescue Mission

A group of k robots operates in a building on fire to rescue people.
There are n people inside and the building consists of m locations.
The state of each robot can be characterized by its status (alive or dead), current location, and an indication whether the robot is carrying some person (and, if so, which person).
Similarly, a person can be characterized by its current status and location.
Each location can be burning, damaged, or still in a good shape.

Robots and people that are alive can try to move North, South, East or West.
Robots can additionally Pick up a person or Lay it on the ground.
Every agent can also decide to do nothing (action Wait).

Agents and MAS

An agent can possibly be:

- *Reactive*: reacts to changes in the environment;
- *Pro-active*: takes the initiative;
- *Autonomous*: operates without direct intervention of others, has some kind of control over its actions and internal state;
- *Goal-directed*: acts to achieve a goal;
- *Social*: interacts with others (i.e., engages in cooperation, communication, coordination, competition, etc.);
- *Embodied*: has sensors and effectors to read from and make changes to the environment;
- *Intelligent*: ...whatever it means;
- *Rational*: always does the “right” thing.

Agents and MAS

Is there any essential (and commonly accepted) feature of an agent?

An agent acts.

Agents can be described mathematically as a function

$$act : \text{set of percept sequences} \mapsto \text{set of actions}$$

In game theory such a function is called a strategy.
In planning, it is called a conditional plan.

Another 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 are also k coercers who can attempt to bribe or blackmail the voters into voting for a particular candidate.
The coercers can possibly use the services of hackers, capable of intercepting unencrypted messages.

Why MAS are useful

By looking at [Y. Shoham, K. Leyton-Brown; 2009], MAS are being applied to:

- distributed constraint satisfaction
- distributed optimization
- negotiation and auctions
- social laws and conventions
- (non)cooperative game theory
- communication
- social choice
- mechanism design

Claim:

It is useful to reason in terms of agents and MAS.

Part 1: Reasoning about Systems

1.2 The Role of Logics for MAS

Why Logic?

Formal logic can be seen as:

- a framework for reasoning about systems
- makes one realise the implicit assumptions
- . . . and then we can:
  - investigate them, accept or reject them, or
  - relax some of them and still use part of the formal and conceptual machinery.

- reasonably expressive but simpler than the full language of mathematics
  - study of computational aspects, in particular decision problems.

- Multi-agent systems provide a paradigm for modeling the world.
- Logic provides a language to express properties of the models

  . . . reason about them

  . . . and compute answers to (some) questions automatically.

Claim:

Logic and Multi-agent Systems are a good match.

Motivating Example: Rescue Robots

Some desirable **properties** we might want to check in the Rescue Robots scenario:

- Every person in the building is safe.
- Every person will eventually be safe.
- Every person may eventually be safe, provided that they cooperate.

- The robots can rescue all the people in the building.
- The robots can rescue all the people, and they know that they can.
- The robots can rescue all the people, and they know how to do it.

Computational Aspects

- Verification: check specification against implementation (more later on)
- Other decision problems: validity, satisfiability, realizability.
- Executable specifications: specification given directly as tests that can be executed.
- Planning as model checking: verification returns an actual plan.

In the context of MAS:

- Game solving, mechanism design, and reasoning about games have natural interpretation as logical problems.

Major research area:

- How difficult is/what is the complexity of solving these problems?

Another Motivating Example: Security of Voting

Desirable **properties** for the Voting scenario:

Privacy: The system cannot reveal how a particular voter voted. Thus, privacy guarantees that the link between a voter and her vote remains secret.

Receipt-freeness: The voter does not gain any information (a receipt) which can be used to prove to a coercer that she voted in a certain way.

Coercion-resistance: The voter cannot cooperate with a coercer to prove to him that she voted in a certain way. Coercion resistance requires that the coercer cannot become convinced of how the voter has voted, even if the voter cooperates with him.

Hereafter we introduce logics suitable to express (some of) the specifications above.

Part 1: Reasoning about Systems

1.3 Formal Verification

The Verification Problem

Given system S and specification P, does S satisfy P?

- safety-critical systems, security and communication protocols, etc.

Model-checking in a nutshell [Clarke, Emerson, Sifakis]

1. Model system S as some transition system M_S
2. Represent specification P as a formula ϕ_P in some logic-based language
3. Check whether $M_S \models \phi_P$

80's-90's: monolithic systems, systems in isolation: LTL, CTL.

since 2000: systems with several components, multi-agent systems, game structures: ATL, Coalition Logic, Strategy Logic.

- notions of strategies, equilibria from Game Theory $\Rightarrow$ Rational Synthesis

the attacker has a strategy to learn the secret eventually.

$\Rightarrow$ Verification of strategic abilities of autonomous agents.

Famous Software Failures

1984 LSE Taurus (Transfer and Automated Registration of Uncertificated Stock): 500m GBP lost

- *The Sunday Times*: “the beginning of the end for the London Stock Exchange”.

1987 Therac-25 (radiotherapy): 6 reported accidents, 3 people died

- “Reusing software modules does not guarantee safety in the new system".

1990 AT&T: 9h-outage of U.S. telephone network: several 100 million USD.

1992 LASCAD (London Ambulance Service Computer-Aided Dispatch): 11h wait

1993 Denver Airport Baggage Delivery System: USD 1.1m/d during 9 months

1994 Pentium FDIV Bug: 500 million USD

1996 Ariane V Crash: 500 million USD

All these failures were due to software bugs.

The Importance of Software Correctness

Rapidly increasing integration of ICT in different applications:

- embedded systems (automotive)
- communication and security protocols
- transportation systems (autonomous vehicles)

$\Rightarrow$ reliability incrasingly depends on software!

Defects can be **fatal** and **costly**:

- products subject to mass-production
- safety-critical systems

What is Model Checking?

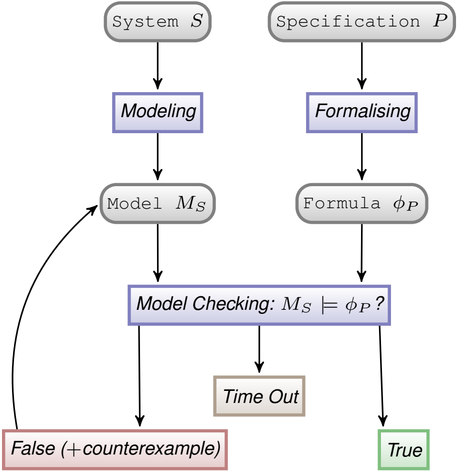
Informal Description

Model checking is an **automated** technique that, given a finite-state model of a system and a formal property, **systematically** checks whether this property holds for (a given state in) that model.

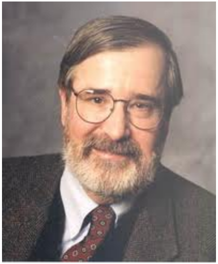
- automated: without intervention from the engineer.
- systematically: all states are checked.
  $\Rightarrow$ models must be finite.

Model Checking

- Given system S and specification P, does S satisfies P?



ACM Turing Award 2007



(a) Edmund Clarke (CMU, USA)

(b) Allen Emerson (U. Texas, USA)

(c) Joseph Sifakis (IMAG Grenoble, F)

- Jury Justification:

  For their roles in developing Model-Checking into a highly effective verification technology, widely adopted in the hardware and software industries.

What Models?

Transition Systems:

- states (labeled with basic propositions).
- transitions between states.

Concurrent Game Structures:

- several agents endowed with local information, actions, protocols.
- action-labeled transitions.

CGS are suitable to represent MAS formally.

What Properties?

Consider the examples given above.

Temporal Logics:

- extensions of propositional logic.
- temporal (modal) operators: G "globally", F "finally", ...
- interpreted over sequences of states (linear) ...
  ... or over infinite trees (branching).

Logics for Strategies:

- modal operator $\langle\langle A \rangle\rangle$: "coalition A has a strategy to achieve ..."
- interpreted over CGS.

References: Multi-Agent Systems

Y. Shoham, K. Leyton-Brown.
Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations.
Cambridge University Press, 2009.

M. Wooldridge.
An Introduction to Multi Agent Systems.
John Wiley & Sons, 2002.

Foundation for Intelligent Physical Agents.
FIPA home page.
`http://www.fipa.org/`.

References: Model Checking

K. Baier, J. P. Katoen (2008);
Principles of Model Checking.
MIT Press, 2008.

Huth, M. & Ryan, M.
Logic in Computer Science: Modeling and reasoning about systems.
Cambridge University Press.

E. M. Clarke, O. Grumberg and D. A. Peled;
Model Checking.
MIT Press, 1999.

Part 2: Reasoning about Time and Change

Part 2: Reasoning about Time and Change

2.1 Temporal Logics

Motivating Example: Rescue Robots

Properties to express

- Every person in the building is safe.
- Every person will eventually be safe.
- Every person may eventually be safe, if everything goes fine.

- Whenever person i gets in trouble, she will eventually be rescued.
- If person i gets outside the building, then she will never be in danger anymore.
- Person i may be rescued without any robot ever entering the building, but guaranteed rescue requires some robots to enter.

Motivating Example: Voting

Properties to express

- The system will not reveal how a particular voter voted (privacy).
- The system does not issue receipts (receipt-freeness).
- The voter can vote, and can refrain from voting.
- The voter can vote, and can refrain from voting. If she votes, the system will not reveal afterwards how she voted (conditional privacy).

Reasoning about Time

Main ideas:

- Temporal (modal) logic extends propositional or predicate logic with modalities to express the behaviour of a reactive system.
- Modal operators refer to dynamics of the system, they are used to specify how the system can/will evolve
- The transition relation is seen as representing time.
- It provides an intuitive but mathematically precise notation for expressing properties about the relation between states in executions.

Beware!

- There are other flavours of temporal logic: Interval Temporal Logic (ITL), First-order Logic with orders, etc.
- These are not covered in this course!

A little bit of history

Temporal reasoning has been studied since ancient times in philosophy.

- **Aristotle**: problem of the future contingents ("there will be a sea-battle tomorrow").
- **Ockham**: branching notion of time.
- **A. N. Prior**: philosopher, interested in free will and predestination.
  - *Time and Modality, 1957*: first modal account of temporal logic, our notation comes from this book.

Late 70's: Pnueli is the first to apply temporal logics to computing.

- **A. Pnueli**: Linear Temporal Logic
- **M. Ben-Ari, Z. Manna and A. Pnueli**: Temporal Logic of Branching Time
- **E. Clarke, A. Emerson**: Computation Tree Logic

No explicit account of time ("I woke up at 8am"), no duration ("I'll be away two days"): only the relative ordering of events is relevant.

- **M. Vardi**: "What on earth does an obscure, old intellectual discipline have to do with the youngest intellectual discipline?".

Temporal Operators

Typical temporal operators:

$\mathrm{X}\varphi$	φ is true in the next moment in time
$\mathrm{G}\varphi$	φ is true in all future moments (globally true)
$\mathrm{F}\varphi$	φ is true in some future moment (finally true)
$\varphi\mathrm{U}\psi$	φ is true until the moment when ψ becomes true

Example formulas:

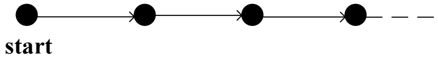
- $\mathrm{G}((\neg\mathsf{passport} \vee \neg\mathsf{ticket}) \rightarrow \mathrm{X}\neg\mathsf{board\_flight})$
- $\mathsf{send}(\mathsf{msg}, \mathsf{rcvr}) \rightarrow \mathrm{F}\,\mathsf{receive}(\mathsf{msg}, \mathsf{rcvr})$

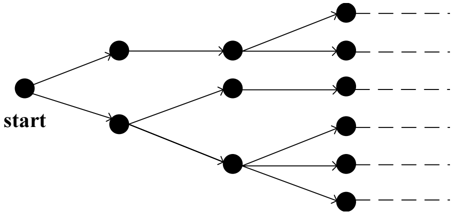
Actually, X and U are enough:

1968 **Kamp**: X and U are sufficient to express all first-order properties over $<$.

Models of Time

- The transition relation represents time.
- Models of time: linear vs. branching.





Where Do Paths Come From?

Definition 2.1 (Unlabelled Transition System)

A (unlabelled) transition system is a pair

$$\langle St, \longrightarrow \rangle$$

where:

- St is a non-empty set of states,
- $\longrightarrow \subseteq St \times St$ is a transition relation.

Note: when we add a valuation $\mathcal{V} : St \to 2^{AP}$ of atoms, we get a **Kripke model**!

Where Do the Paths Come From?

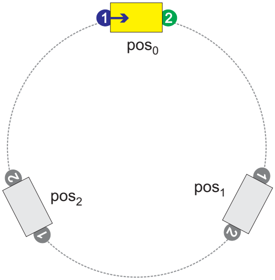
Definition 2.2 (Paths in a transition system)

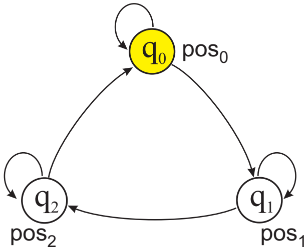
A path λ is an infinite sequence of states that can be effected by subsequent transitions.

A path must be full, i.e., either infinite or ending in a state with no outgoing transition.

Usually, we assume that the transition relation is serial (time flows forever). Then, all paths are infinite.

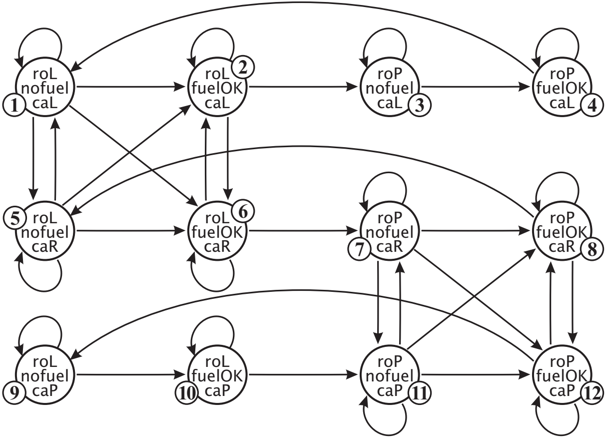
Example: Robots and Carriage





- Robot 1 can push the carriage so that it moves clockwise.
- It can also refrain from pushing, in which case the carriage does not move.
- Robot 2 has no influence on the position of the carriage.
- The carriage can move clockwise, or remain in the same place.
- The carriage can be in 3 different positions (states): q_0, q_1, q_2.
- We label the states by atoms pos_0, pos_1, pos_2.
- Transition system $TS = \langle St, \longrightarrow \rangle$, where
  - $St = \{q_0, q_1, q_2\}$
  - $q_i \longrightarrow q_i$, for $i \in \{0, 1, 2\}$, and $q_0 \longrightarrow q_1$, $q_1 \longrightarrow q_2$, $q_2 \longrightarrow q_0$

Example: Rocket and Cargo



- A rocket can be moved between London (atom roL) and Paris (atom roP).
- The cargo can be in London (caL), Paris (caP), or inside the rocket (caR).
- The rocket can fly only if its fuel tank if full (fuelOK).
- When it flies, it consumes fuel, and nofuel holds after each flight.

Reasoning about Time: Specification Templates

Temporal logic was originally developed to represent tense in natural language.

In Computer Science it 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
- liveness properties
- fairness properties

Reasoning about Time: Safety Properties

Safety/maintenance goals:

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

Typical examples:

> $\mathrm{G}\neg\mathsf{bankrupt}$
>
> $\mathrm{G}(\mathsf{fuelOK} \vee \mathrm{X}\,\mathsf{fuelOK})$
>
> and so on ...

Usually: $\mathrm{G}\neg\ldots$ / $\mathrm{G}\ldots$

Reasoning about Time: Liveness Properties

Liveness/achievement:

"something good will finally happen"

Typical examples:

$\mathrm{F}\,\mathsf{rich}$

$\mathrm{F}\mathrm{G}\,\mathsf{rich}$

$\mathsf{requested} \rightarrow \mathrm{F}\,\mathsf{granted}$

and so on ...

Usually: $\mathrm{F} \ldots$ / $\mathrm{F}\mathrm{G} \ldots$

Reasoning about Time: Fairness Properties

Fairness/service:

> *"whenever something is attempted/requested, then it will be successful/granted"*

Typical examples:

$\mathrm{G}\,\mathrm{F}\,\mathsf{connected}$
$\neg\mathrm{F}\,\mathrm{G}\,\mathsf{down}$
$\mathrm{G}(\mathsf{calling} \rightarrow \mathrm{F}\,\mathsf{answering})$
$(\mathrm{G}\,\mathrm{F}\,\mathsf{attempt}) \rightarrow (\mathrm{G}\,\mathrm{F}\,\mathsf{success})$
and so on . . .

Usually: $\mathrm{G}\,\mathrm{F} \ldots$ / $\neg\mathrm{F}\,\mathrm{G} \ldots$

Fairness properties:

- useful when scheduling processes, responding to messages, etc.
- good for specifying properties of the environment

Reasoning about Time

Wait! What about strategic behaviours?

Temporal logics are the logical basis whereupon strategy logics are built.

Part 2: Reasoning about Time and Change

2.2 Linear Temporal Logic

Linear Time: LTL

- **LTL**: **Linear Temporal Logic**
- Time is linear: just a single path is considered!
- Reasoning about a particular computation of a system
- Model: a path (infinite sequence of states)
- Important distinction: computational vs. behavioral structure

Linear Temporal Logic: Syntax

- Modal logic on infinite paths [Pneuli 1977]

- Propositional logic
  - $a \in AP$ — atomic propositions (atoms)
  - $\neg\phi$ — negation
  - $\phi \wedge \psi$ — conjunction

- Temporal operators
  - $\mathrm{X}\phi$ — *neXt* ϕ
  - $\phi \mathrm{U} \psi$ — ϕ *Until* ψ

LTL is a logic to express properties of linear time.

Derived Operators

$$
\begin{array}{rcll}
\phi \vee \psi & ::= & \neg(\neg\phi \wedge \neg\psi) & \\
\phi \rightarrow \psi & ::= & \neg\phi \vee \psi & \\
\phi \leftrightarrow \psi & ::= & (\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi) & \\
true & ::= & \phi \vee \neg\phi & \\
false & ::= & \phi \wedge \neg\phi = \neg true & \\
\mathrm{F}\phi & ::= & true\mathrm{U}\phi & \text{Finally } \phi \\
\mathrm{G}\phi & = & \neg\mathrm{F}\neg\phi & \text{Globally } \phi
\end{array}
$$

- **Priority order**: unary operators bind more strongly than binary operators; $\neg$ et X bind equally strong; U takes precedence over $\wedge$, $\vee$ and $\rightarrow$.
- Parentheses are omitted whenever appropriate: $\varphi_1 \mathrm{U} \varphi_2 = ((\varphi_1)\mathrm{U}(\varphi_2))$.
- Operator U is right-associative: $\varphi_1 \mathrm{U} \varphi_2 \mathrm{U} \varphi_3 = \varphi_1 \mathrm{U}(\varphi_2 \mathrm{U} \varphi_3)$.
- In some textbooks G and F are also written $\Box$ and $\diamond$.

Linear Temporal Logic: Syntax

Let $AP = \{x = 1, x < 2, x \geq 3\}$.
Then, what about:

1. $\mathrm{X}(x = 1)$?

Linear Temporal Logic: Syntax

Let $AP = \{x = 1, x < 2, x \geq 3\}$.
Then, what about:

1. $\mathrm{X}(x = 1)$ ✓
2. $\mathrm{U}(x = 1)$?

Linear Temporal Logic: Syntax

Let $AP = \{x = 1, x < 2, x \geq 3\}$.
Then, what about:

1. $\mathrm{X}(x = 1)$ ✓
2. $\mathrm{U}(x = 1)$ ✗
3. $(x < 2) \vee \mathrm{G}(x = 1)$?

Linear Temporal Logic: Syntax

Let $AP = \{x = 1, x < 2, x \geq 3\}$.
Then, what about:

1. $\mathrm{X}(x = 1)$ ✓
2. $\mathrm{U}(x = 1)$ ✗
3. $(x < 2) \vee \mathrm{G}(x = 1)$ ✓
4. $(x = 1)\mathrm{F}\mathrm{X}(x \geq 3)$?

Linear Temporal Logic: Syntax

Let $AP = \{x = 1, x < 2, x \geq 3\}$.
Then, what about:

1. $\mathrm{X}(x = 1)$ ✓
2. $\mathrm{U}(x = 1)$ ✗
3. $(x < 2) \vee \mathrm{G}(x = 1)$ ✓
4. $(x = 1)\mathrm{FX}(x \geq 3)$ ✗
5. $\mathrm{X} \to (true\mathrm{U}(x = 1))$?

Linear Temporal Logic: Syntax

Let $AP = \{x = 1, x < 2, x \geq 3\}$.
Then, what about:

1. $\mathrm{X}(x = 1)$ ✓
2. $\mathrm{U}(x = 1)$ ✗
3. $(x < 2) \vee \mathrm{G}(x = 1)$ ✓
4. $(x = 1)\mathrm{FX}(x \geq 3)$ ✗
5. $\mathrm{X} \rightarrow (true\mathrm{U}(x = 1))$ ✗
6. $\mathrm{X}(x = 1 \wedge \mathrm{GX}(x \geq 3))$?

Linear Temporal Logic: Syntax

Let $AP = \{x = 1, x < 2, x \geq 3\}$.
Then, what about:

1. $\mathrm{X}(x = 1)$ ✓
2. $\mathrm{U}(x = 1)$ ✗
3. $(x < 2) \vee \mathrm{G}(x = 1)$ ✓
4. $(x = 1)\mathrm{F}\mathrm{X}(x \geq 3)$ ✗
5. $\mathrm{X} \rightarrow (true\mathrm{U}(x = 1))$ ✗
6. $\mathrm{X}(x = 1 \wedge \mathrm{G}\mathrm{X}(x \geq 3))$ ✓
7. $\mathrm{X}(true\mathrm{U}(x = 1)) \rightarrow \mathrm{G}(x = 1)$?

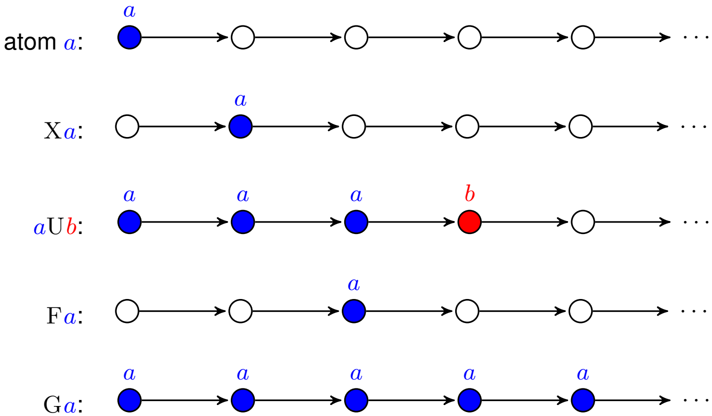
Linear Temporal Logic: Syntax

Let $AP = \{x = 1, x < 2, x \geq 3\}$.
Then, what about:

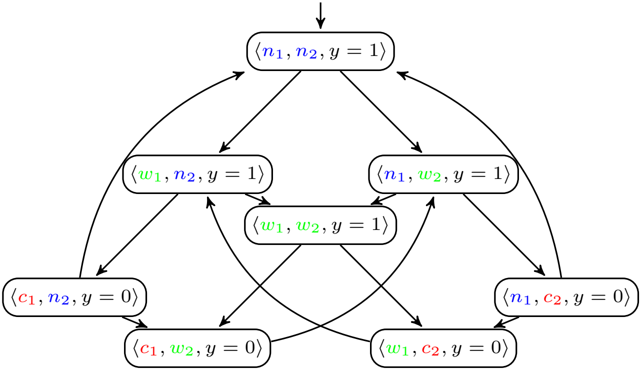
1. $\mathrm{X}(x = 1)$ ✓
2. $\mathrm{U}(x = 1)$ ✗
3. $(x < 2) \vee \mathrm{G}(x = 1)$ ✓
4. $(x = 1)\mathrm{F}\mathrm{X}(x \geq 3)$ ✗
5. $\mathrm{X} \rightarrow (true\mathrm{U}(x = 1))$ ✗
6. $\mathrm{X}(x = 1 \wedge \mathrm{G}\mathrm{X}(x \geq 3))$ ✓
7. $\mathrm{X}(true\mathrm{U}(x = 1)) \rightarrow \mathrm{G}(x = 1)$ ✓

Intuitive Meaning

Discrete account of time: the present moment refers to the current state and the next moment corresponds to the immediate successor state.



Linear-time Properties: Mutual Exclusion



- mutual exclusion: $\mathrm{G}\neg(cr_1 \wedge cr_2)$
- (weak) starvation freedom: $(\mathrm{GF}wa_1 \rightarrow \mathrm{GF}cr_1) \wedge (\mathrm{GF}wa_2 \rightarrow \mathrm{GF}cr_2)$
- (strong) starvation freedom: $\mathrm{GF}cr_1 \wedge \mathrm{GF}cr_2$

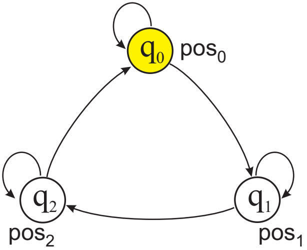
Definition 2.3 (Models of LTL)

A **model of LTL** is a sequence of time moments (states).
We call such models **paths**, and denote them by λ.
Evaluation $\mathcal{V} : St \to 2^{AP}$ of atoms at particular time moments is also needed.

Notation:

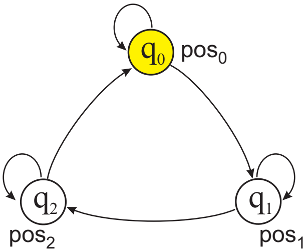
- $\lambda[i]$: ith time moment (starting from 0)
- $\lambda[i \ldots j]$: all time moments between i and j
- $\lambda[i \ldots \infty]$: all timepoints from i on

Example: Robots and Carriage



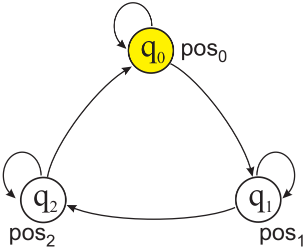
1. $q_0, q_1, q_2, q_2, \ldots$?

Example: Robots and Carriage



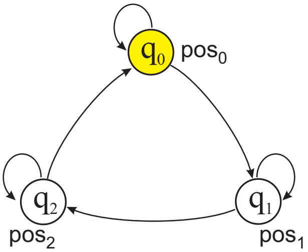
1. $q_0, q_1, q_2, q_2, \ldots = q_0, q_1, (q_2)^\omega$ ✓
2. $q_0, q_1, q_2, q_0, q_1, q_2, \ldots$?

Example: Robots and Carriage



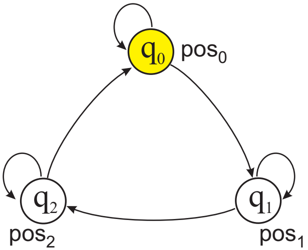
1. $q_0, q_1, q_2, q_2, \ldots = q_0, q_1, (q_2)^\omega$ ✓
2. $q_0, q_1, q_2, q_0, q_1, q_2, \ldots = (q_0, q_1, q_2)^\omega$ ✓
3. $q_0, q_2, q_1, \ldots$?

Example: Robots and Carriage



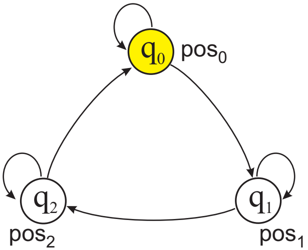
1. $q_0, q_1, q_2, q_2, \ldots = q_0, q_1, (q_2)^\omega$ ✓
2. $q_0, q_1, q_2, q_0, q_1, q_2, \ldots = (q_0, q_1, q_2)^\omega$ ✓
3. $q_0, q_2, q_1, \ldots = (q_0, q_2, q_1) \cdot St^\omega$ ✗
4. $q_0, q_0, \ldots$?

Example: Robots and Carriage



1. $q_0, q_1, q_2, q_2, \ldots = q_0, q_1, (q_2)^\omega$ ✓
2. $q_0, q_1, q_2, q_0, q_1, q_2, \ldots = (q_0, q_1, q_2)^\omega$ ✓
3. $q_0, q_2, q_1, \ldots = (q_0, q_2, q_1) \cdot St^\omega$ ✗
4. $q_0, q_0, \ldots = (q_0)^\omega$ ✓
5. $q_1, q_1, q_2, q_0, q_1, q_0, \ldots$?

Example: Robots and Carriage



1. $q_0, q_1, q_2, q_2, \ldots = q_0, q_1, (q_2)^\omega$ ✓
2. $q_0, q_1, q_2, q_0, q_1, q_2, \ldots = (q_0, q_1, q_2)^\omega$ ✓
3. $q_0, q_2, q_1, \ldots = (q_0, q_2, q_1) \cdot St^\omega$ ✗
4. $q_0, q_0, \ldots = (q_0)^\omega$ ✓
5. $q_1, q_1, q_2, q_0, q_1, q_0, \ldots = q_1, q_1, q_2, q_0, q_1, q_0 \cdot St^\omega$ ✗

Definition 2.4 (Semantics of LTL)

$\lambda \models true$	
$\lambda \models p$	iff p is true at initial moment $\lambda[0]$ (i.e., $p \in \mathcal{V}(\lambda[0])$)
$\lambda \models \neg\varphi$	iff not $\lambda \models \varphi$
$\lambda \models \varphi \wedge \psi$	iff $\lambda \models \varphi$ and $\lambda \models \psi$

Definition 2.4 (Semantics of LTL)

$\lambda \models true$	
$\lambda \models p$	iff p is true at initial moment $\lambda[0]$ (i.e., $p \in \mathcal{V}(\lambda[0])$)
$\lambda \models \neg\varphi$	iff not $\lambda \models \varphi$
$\lambda \models \varphi \wedge \psi$	iff $\lambda \models \varphi$ and $\lambda \models \psi$
$\lambda \models \mathrm{X}\varphi$	iff $\lambda[1] \models \varphi$

Definition 2.4 (Semantics of LTL)

$\lambda \models true$	
$\lambda \models p$	iff p is true at initial moment $\lambda[0]$ (i.e., $p \in \mathcal{V}(\lambda[0])$)
$\lambda \models \neg\varphi$	iff not $\lambda \models \varphi$
$\lambda \models \varphi \wedge \psi$	iff $\lambda \models \varphi$ and $\lambda \models \psi$
$\lambda \models \mathrm{X}\varphi$	iff $\lambda[1] \models \varphi$ No!

Definition 2.4 (Semantics of LTL)

$\lambda \models true$	
$\lambda \models p$	iff p is true at initial moment $\lambda[0]$ (i.e., $p \in \mathcal{V}(\lambda[0])$)
$\lambda \models \neg\varphi$	iff not $\lambda \models \varphi$
$\lambda \models \varphi \wedge \psi$	iff $\lambda \models \varphi$ and $\lambda \models \psi$
$\lambda \models \mathrm{X}\varphi$	iff $\lambda[1..\infty] \models \varphi$

Linear Temporal Logic: Semantics

Definition 2.4 (Semantics of LTL)

$\lambda \models true$	
$\lambda \models p$	iff p is true at initial moment $\lambda[0]$ (i.e., $p \in \mathcal{V}(\lambda[0])$)
$\lambda \models \neg\varphi$	iff not $\lambda \models \varphi$
$\lambda \models \varphi \wedge \psi$	iff $\lambda \models \varphi$ and $\lambda \models \psi$
$\lambda \models \mathrm{X}\varphi$	iff $\lambda[1..\infty] \models \varphi$
$\lambda \models \varphi \mathrm{U} \psi$	iff $\lambda[i..\infty] \models \psi$ for some $i \geq 0$, and $\lambda[j..\infty] \models \varphi$ for all $0 \leq j < i$

The Derived Semantics of G, F, GF, FG

Recall that $\mathrm{F}\varphi \equiv true \mathrm{U} \varphi$ and $\mathrm{G}\varphi \equiv \neg\mathrm{F}\neg\varphi$.

Semantics of derived operators

$\lambda \models \mathrm{F}\varphi$	iff $\lambda[i..\infty] \models \varphi$ for some $i \geq 0$
$\lambda \models \mathrm{G}\varphi$	iff $\lambda[i..\infty] \models \varphi$ for all $i \geq 0$
$\lambda \models \mathrm{GF}\varphi$	iff for all $i \geq 0$, for some $j \geq i$, $\lambda[j..\infty] \models \varphi$ (infinitely often)
$\lambda \models \mathrm{FG}\varphi$	iff for some $i \geq 0$, for all $j \geq i$, $\lambda[j..\infty] \models \varphi$ (persistence)

Note that:

$$\mathrm{G}\varphi \equiv \neg\mathrm{F}\neg\varphi$$

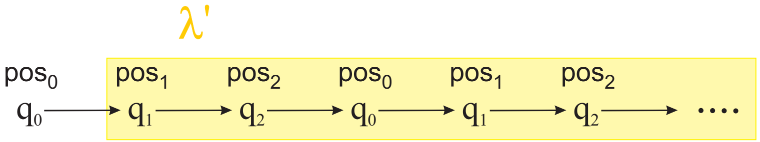
$$\mathrm{F}\varphi \equiv \neg\mathrm{G}\neg\varphi$$

Semantics of LTL: $\mathrm{F}\,\mathsf{pos}_1$

$$\begin{array}{ccccccccccccc}
\lambda & & & & & & & & & & & & \\
\mathsf{pos}_0 & & \mathsf{pos}_1 & & \mathsf{pos}_2 & & \mathsf{pos}_0 & & \mathsf{pos}_1 & & \mathsf{pos}_2 & & \\
q_0 & \longrightarrow & q_1 & \longrightarrow & q_2 & \longrightarrow & q_0 & \longrightarrow & q_1 & \longrightarrow & q_2 & \longrightarrow & \cdots
\end{array}$$

$$\lambda \models \mathrm{F}\,\mathsf{pos}_1$$

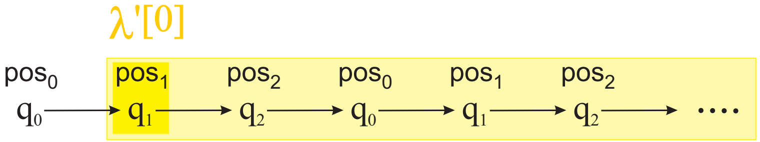
Semantics of LTL: $\mathrm{F}\,\mathsf{pos}_1$



$$\lambda \models \mathrm{F}\,\mathsf{pos}_1$$

$$\lambda' = \lambda[1..\infty] \models \mathsf{pos}_1$$

Semantics of LTL: $\mathrm{F}\,\mathsf{pos}_1$



$$\lambda \models \mathrm{F}\,\mathsf{pos}_1$$

$$\lambda' = \lambda[1..\infty] \models \mathsf{pos}_1$$
$$\mathsf{pos}_1 \in \mathcal{V}(\lambda'[0]) = \mathcal{V}(q_1)$$

Semantics of LTL: $\mathrm{GF}\,\mathsf{pos}_1$

λ

$$\begin{array}{ccccccccccccc} \mathsf{pos}_0 & & \mathsf{pos}_1 & & \mathsf{pos}_2 & & \mathsf{pos}_0 & & \mathsf{pos}_1 & & \mathsf{pos}_2 & & \\ q_0 & \longrightarrow & q_1 & \longrightarrow & q_2 & \longrightarrow & q_0 & \longrightarrow & q_1 & \longrightarrow & q_2 & \longrightarrow & \cdots \end{array}$$

$$\lambda \models \mathrm{GF}\,\mathsf{pos}_1$$

Semantics of LTL: $\mathrm{G}\,\mathsf{F}\,\mathsf{pos}_1$

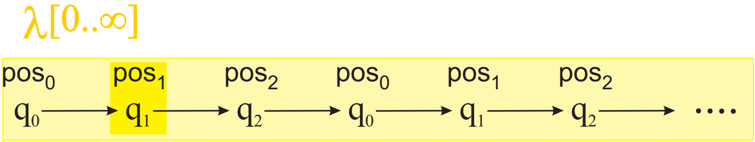
$\lambda[0..\infty]$

$$\begin{array}{ccccccccccccc} \mathsf{pos}_0 & & \mathsf{pos}_1 & & \mathsf{pos}_2 & & \mathsf{pos}_0 & & \mathsf{pos}_1 & & \mathsf{pos}_2 & & \\ q_0 & \longrightarrow & q_1 & \longrightarrow & q_2 & \longrightarrow & q_0 & \longrightarrow & q_1 & \longrightarrow & q_2 & \longrightarrow & \cdots \end{array}$$

$$\lambda \models \mathrm{G}\,\mathrm{F}\,\mathsf{pos}_1$$

$$\lambda[0..\infty] \models \mathrm{F}\,\mathsf{pos}_1$$

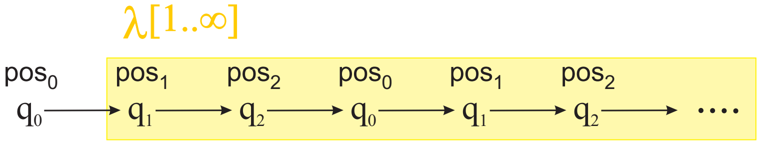
Semantics of LTL: $\mathrm{G}\,\mathrm{F}\,\mathsf{pos}_1$



$$\lambda \models \mathrm{G}\,\mathrm{F}\,\mathsf{pos}_1$$

$$\lambda[0..\infty] \models \mathrm{F}\,\mathsf{pos}_1$$

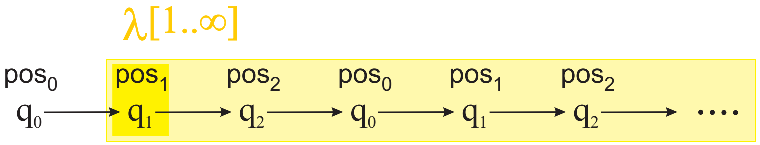
Semantics of LTL: $\mathrm{G}\mathrm{F}\,\mathsf{pos}_1$



$$\lambda \models \mathrm{G}\mathrm{F}\,\mathsf{pos}_1$$

$$\lambda[0..\infty] \models \mathrm{F}\,\mathsf{pos}_1$$
$$\lambda[1..\infty] \models \mathrm{F}\,\mathsf{pos}_1$$

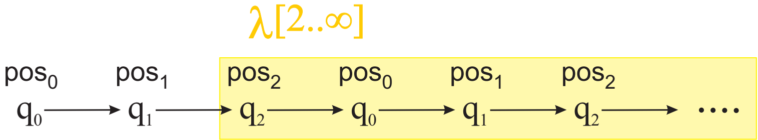
Semantics of LTL: $\mathrm{GF}\,\mathsf{pos}_1$



$$\lambda \models \mathrm{GF}\,\mathsf{pos}_1$$

$$\lambda[0..\infty] \models \mathrm{F}\,\mathsf{pos}_1$$
$$\lambda[1..\infty] \models \mathrm{F}\,\mathsf{pos}_1$$

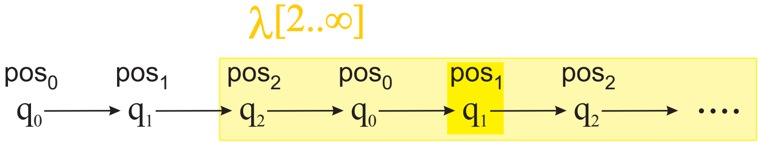
Semantics of LTL: $\mathrm{GF}\,\mathsf{pos}_1$



$$\lambda \models \mathrm{GF}\,\mathsf{pos}_1$$

$$\lambda[0..\infty] \models \mathrm{F}\,\mathsf{pos}_1$$
$$\lambda[1..\infty] \models \mathrm{F}\,\mathsf{pos}_1$$
$$\lambda[2..\infty] \models \mathrm{F}\,\mathsf{pos}_1$$

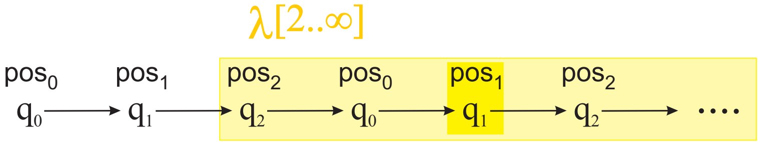
Semantics of LTL: $\mathrm{G}\,\mathrm{F}\,\mathsf{pos}_1$



$$\lambda \models \mathrm{G}\,\mathrm{F}\,\mathsf{pos}_1$$

$$\lambda[0..\infty] \models \mathrm{F}\,\mathsf{pos}_1$$
$$\lambda[1..\infty] \models \mathrm{F}\,\mathsf{pos}_1$$
$$\lambda[2..\infty] \models \mathrm{F}\,\mathsf{pos}_1$$

Semantics of LTL: $\mathrm{G}\,\mathrm{F}\,\mathsf{pos}_1$



$$\lambda \models \mathrm{G}\,\mathrm{F}\,\mathsf{pos}_1$$

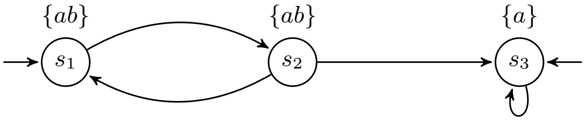
$$\lambda[0..\infty] \models \mathrm{F}\,\mathsf{pos}_1$$
$$\lambda[1..\infty] \models \mathrm{F}\,\mathsf{pos}_1$$
$$\lambda[2..\infty] \models \mathrm{F}\,\mathsf{pos}_1$$
$$\ldots$$

Semantics of LTL in Transition Systems

Definition 2.5 (Semantics of LTL in Transition Systems)

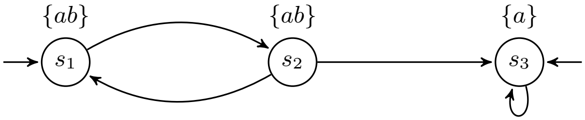
$(M, q) \models \varphi$	iff	$\lambda \models \varphi$ for every path λ in M starting from q.
$M \models \varphi$	iff	$(M, q_0) \models \varphi$ for every initial state q_0 in M.

Example



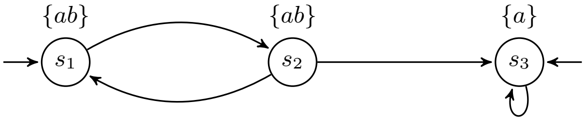
$$M \models \mathrm{G}a?$$

Example



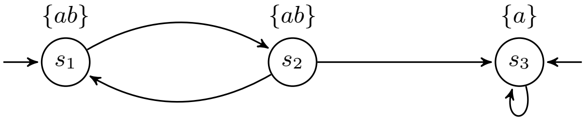
$$M \models \mathrm{G}a$$

Example



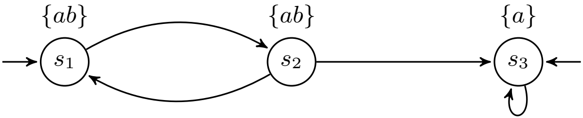
$$
\begin{array}{lcl}
M & \models & \mathrm{G}a \\
M & \models & \mathrm{X}(a \wedge b)?
\end{array}
$$

Example



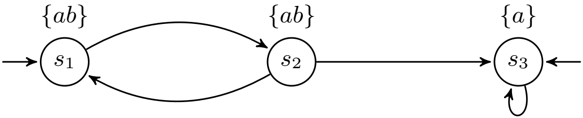
$$
\begin{array}{lcl}
M & \models & \mathrm{G}a \\
M & \not\models & \mathrm{X}(a \wedge b)
\end{array}
$$

Example



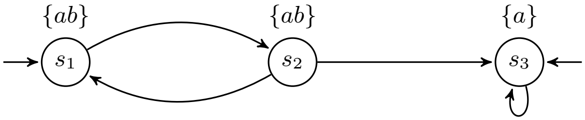
$$
\begin{aligned}
M &\models \mathrm{G}\, a \\
M &\not\models \mathrm{X}\,(a \wedge b) \\
M &\models \mathrm{G}\,(\neg b \rightarrow \mathrm{G}\,(a \wedge \neg b))?
\end{aligned}
$$

Example



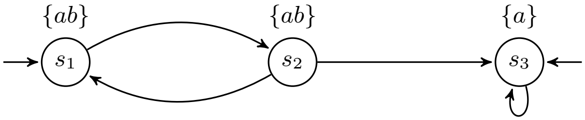
$$
\begin{array}{lcl}
M & \models & \mathrm{G}\,a \\
M & \not\models & \mathrm{X}(a \wedge b) \\
M & \models & \mathrm{G}(\neg b \rightarrow \mathrm{G}(a \wedge \neg b))
\end{array}
$$

Example



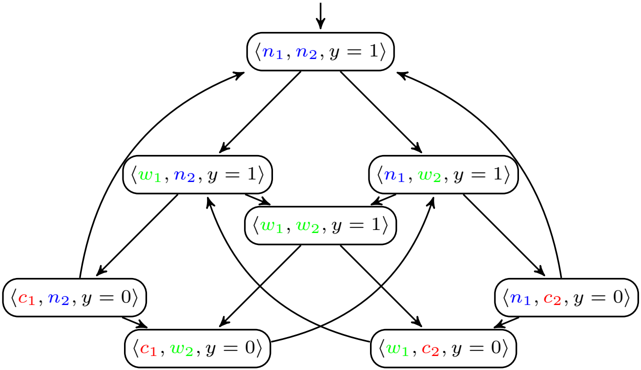
$$
\begin{array}{lll}
M & \models & \mathrm{G}\,a \\
M & \not\models & \mathrm{X}\,(a \wedge b) \\
M & \models & \mathrm{G}\,(\neg b \rightarrow \mathrm{G}\,(a \wedge \neg b)) \\
M & \models & b\mathrm{U}\,(a \wedge \neg b)?
\end{array}
$$

Example



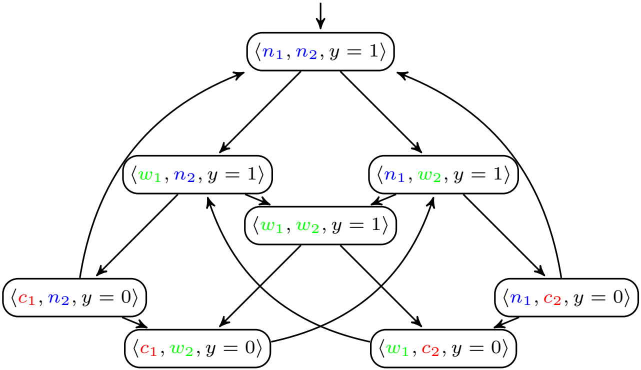
$$
\begin{array}{lcl}
M & \models & \mathrm{G}\,a \\
M & \not\models & \mathrm{X}\,(a \wedge b) \\
M & \models & \mathrm{G}\,(\neg b \rightarrow \mathrm{G}\,(a \wedge \neg b)) \\
M & \models & b\mathrm{U}\,(a \wedge \neg b)
\end{array}
$$

Specifying Properties in LTL: Mutual Exclusion



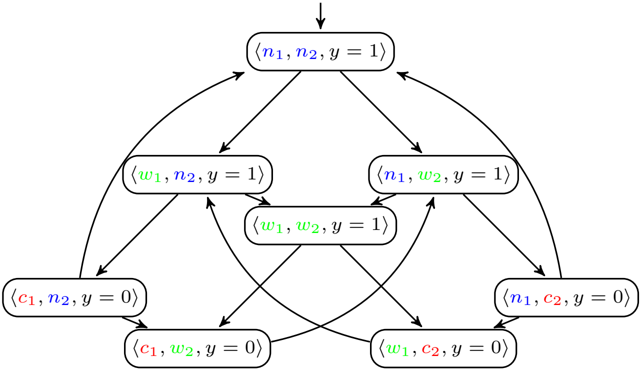
$$M_{Sem} \quad \models \quad \mathrm{G}\neg(cr_1 \wedge cr_2)?$$

Specifying Properties in LTL: Mutual Exclusion



$$M_{Sem} \quad \models \quad \mathrm{G}\neg(cr_1 \wedge cr_2)$$

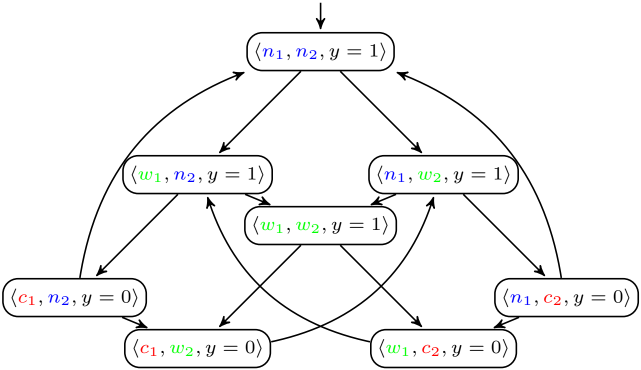
Specifying Properties in LTL: Mutual Exclusion



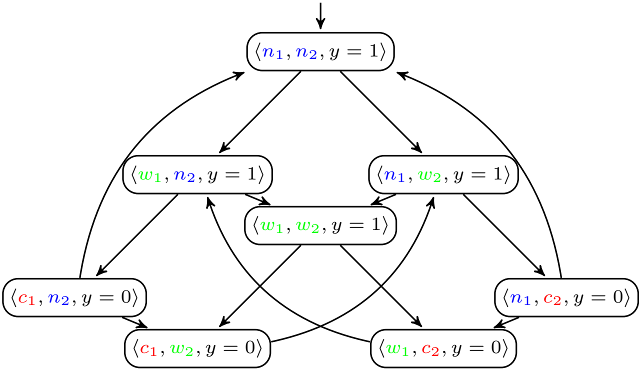
$$M_{Sem} \models G\neg(cr_1 \wedge cr_2)$$

$$M_{Sem} \models GF cr_1 \vee GF cr_2?$$

Specifying Properties in LTL: Mutual Exclusion

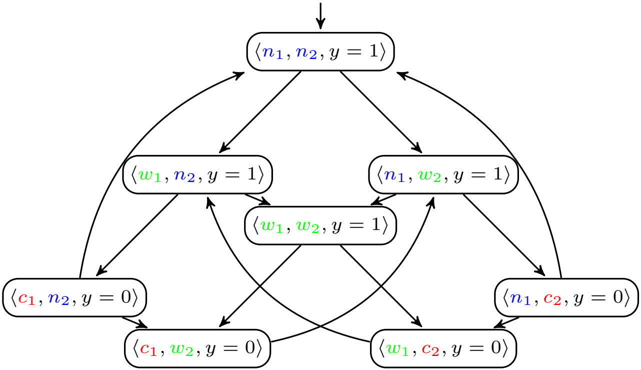


$$M_{Sem} \models G\neg(cr_1 \wedge cr_2)$$
$$M_{Sem} \models GF cr_1 \vee GF cr_2$$



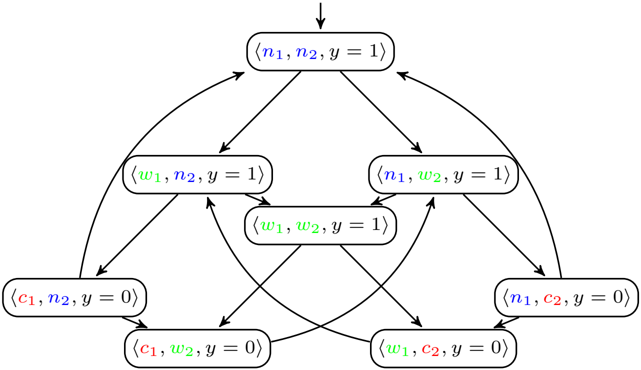
$$M_{Sem} \models G\neg(cr_1 \wedge cr_2)$$
$$M_{Sem} \models GF cr_1 \vee GF cr_2$$
$$M_{Sem} \models GF cr_1 \wedge GF cr_2?$$

Specifying Properties in LTL: Mutual Exclusion



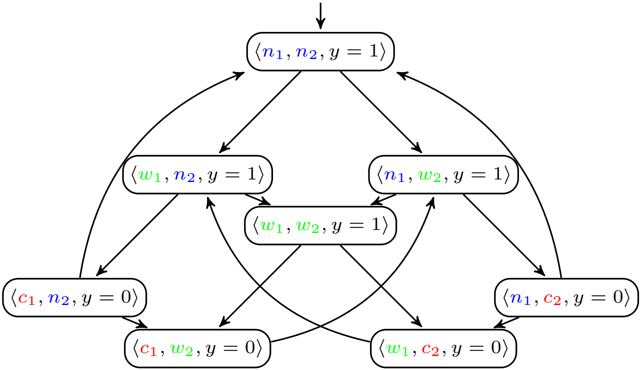
$$
\begin{array}{lcl}
M_{Sem} & \models & G\neg(cr_1 \wedge cr_2) \\
M_{Sem} & \models & GF\,cr_1 \vee GF\,cr_2 \\
M_{Sem} & \not\models & GF\,cr_1 \wedge GF\,cr_2
\end{array}
$$

Specifying Properties in LTL: Mutual Exclusion



$$\begin{array}{lcl} M_{Sem} & \models & \mathrm{G}\neg(cr_1 \wedge cr_2) \\ M_{Sem} & \models & \mathrm{GF}cr_1 \vee \mathrm{GF}cr_2 \\ M_{Sem} & \not\models & \mathrm{GF}cr_1 \wedge \mathrm{GF}cr_2 \\ M_{Sem} & \models & (\mathrm{GF}wa_1 \rightarrow \mathrm{GF}cr_1)? \end{array}$$

Specifying Properties in LTL: Mutual Exclusion

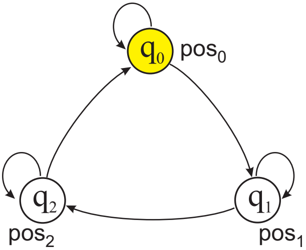


$$
\begin{array}{lll}
M_{Sem} & \models & \mathrm{G}\neg(cr_1 \wedge cr_2) \\
M_{Sem} & \models & \mathrm{GF}cr_1 \vee \mathrm{GF}cr_2 \\
M_{Sem} & \not\models & \mathrm{GF}cr_1 \wedge \mathrm{GF}cr_2 \\
M_{Sem} & \not\models & (\mathrm{GF}wa_1 \rightarrow \mathrm{GF}cr_1)
\end{array}
$$

Properties of LTL

Semantics of Negation

- For paths we have that $\pi \models \phi$ iff $\pi \not\models \neg\phi$
- But it is not the case that $M \not\models \phi$ iff $M \models \neg\phi$
- M does not satisfy either ϕ or $\neg\phi$ if there are (initial) paths π_1 and π_2 such that $\pi_1 \models \phi$ and $\pi_2 \models \neg\phi$



$(M, q_0) \not\models \mathrm{G}\,\mathsf{pos}_0$
(but also: $(M, q_0) \not\models \neg\mathrm{G}\,\mathsf{pos}_0$!)

$(M, q_0) \not\models \mathrm{F}\,\mathsf{pos}_1$
(but also: $(M, q_0) \not\models \neg\mathrm{F}\,\mathsf{pos}_1$!)

$(M, q_0) \models (\neg\mathrm{G}\,\mathsf{pos}_0) \rightarrow \mathrm{F}\,\mathsf{pos}_1$

Definition 2.6 (Equivalence)

Formulas ϕ, ψ are **equivalent**, or $\phi \equiv \psi$, iff for every model M, $M \models \phi$ iff $M \models \psi$.

- Duality

$$\begin{array}{rcl} \neg \mathrm{G}\phi & \equiv & \mathrm{F}\neg\phi \\ \neg \mathrm{F}\phi & \equiv & \mathrm{G}\neg\phi \\ \neg \mathrm{X}\phi & \equiv & \mathrm{X}\neg\phi \end{array}$$

- Idempotency

$$\begin{array}{rcl} \mathrm{GG}\phi & \equiv & \mathrm{G}\phi \\ \mathrm{FF}\phi & \equiv & \mathrm{F}\phi \\ \phi \mathrm{U}(\phi \mathrm{U}\psi) & \equiv & \phi \mathrm{U}\psi \\ (\phi \mathrm{U}\psi)\mathrm{U}\psi & \equiv & \phi \mathrm{U}\psi \end{array}$$

- Absorption

$$\begin{array}{rcl} \mathrm{FGF}\phi & \equiv & \mathrm{GF}\phi \\ \mathrm{GFG}\phi & \equiv & \mathrm{FG}\phi \end{array}$$

In LTL there are only 4 non-equivalent combinations of G and F: G, F, GF, FG.

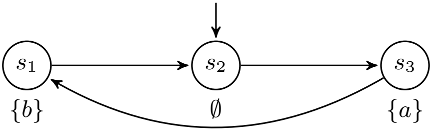
Equivalences

- Distribution

$$\begin{array}{rcl} X(\phi U \psi) & \equiv & (X\phi)U(X\psi) \\ F(\phi \vee \psi) & \equiv & F\phi \vee F\psi \\ G(\phi \wedge \psi) & \equiv & G\phi \wedge G\psi \end{array}$$

- But,

$$\begin{array}{rcl} F(\phi U \psi) & \not\equiv & (F\phi)U(F\psi) \\ G(\phi U \psi) & \not\equiv & (G\phi)U(G\psi) \\ F(\phi \wedge \psi) & \not\equiv & F\phi \wedge F\psi \\ G(\phi \vee \psi) & \not\equiv & G\phi \vee G\psi \end{array}$$



$TS \models Fa \wedge Fb$ but $TS \not\models F(a \wedge b)$

Equivalences

- Expansion

$$
\begin{aligned}
\phi \mathrm{U} \psi &\equiv \psi \lor (\phi \land \mathrm{X}(\phi \mathrm{U} \psi)) \\
\mathrm{F}\psi &\equiv \psi \lor \mathrm{X}\mathrm{F}\psi \\
\mathrm{G}\psi &\equiv \psi \land \mathrm{X}\mathrm{G}\psi
\end{aligned}
$$

$\phi \mathrm{U} \psi$ is the least solution of the expansion law for U:

- if α is such that $\alpha \equiv \psi \lor (\phi \land \mathrm{X}\alpha)$, then $\phi \mathrm{U} \psi \rightarrow \alpha$

Similarly, $\mathrm{G}\psi$ is the greatest solution of the expansion law for G:

- if α is such that $\alpha \equiv \psi \land \mathrm{X}\alpha$, then $\alpha \rightarrow \mathrm{G}\psi$ (e.g., $\alpha = false$)

Motivating Example: Rescue Robots

- Everybody is safe:
  $\bigwedge_{i \in \mathit{People}} \mathsf{safe_i}$
- Everybody will eventually be safe:
  $\bigwedge_{i \in \mathit{People}} \mathrm{F}\, \mathsf{safe_i}$
  different interpretation: $\mathrm{F}\left(\bigwedge_{i \in \mathit{People}} \mathsf{safe_i}\right)$
- Everybody will always be safe, from some moment on:
  $\bigwedge_{i \in \mathit{People}} \mathrm{F}\mathrm{G}\, \mathsf{safe_i}$
  equivalently: $\mathrm{F}\mathrm{G}\left(\bigwedge_{i \in \mathit{People}} \mathsf{safe_i}\right)$
- Everybody may eventually be safe, if everything goes fine:
  Cannot be expressed in LTL!

Motivating Example: Rescue Robots

- Whenever person i gets in trouble, she will eventually be rescued:

  $$G(\neg \mathsf{safe_i} \rightarrow F\, \mathsf{safe_i})$$

- If person i gets outside the building, then she will never be in danger anymore:

  $(F\, \mathsf{outside_i}) \rightarrow (F G\, \mathsf{safe_i})$ $\rightsquigarrow$ not quite what we want!

  $F(\mathsf{outside_i} \rightarrow G\, \mathsf{safe_i})$ $\rightsquigarrow$ not quite right either

  we want to express that *whenever a person gets outside, she will remain safe from then on*:

  $G(\mathsf{outside_i} \rightarrow G\, \mathsf{safe_i})$!

- Person i may be rescued without any robot ever entering the building, but guaranteed rescue requires some robots to enter:

  Cannot be expressed in LTL!

Motivating Example: Voting

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

  $G \neg \mathsf{revealed}_i$

- The system does not issue receipts:

  $G \neg \mathsf{receipt}_i$

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

  Cannot be expressed in LTL

- The voter can vote, and can refrain from voting. If she votes, the system will not reveal afterwards how she voted:

  Cannot be expressed in LTL

What's Missing

To sum up: LTL

- Syntax
- Semantics
- Properties

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 may happen

Part 2: Reasoning about Time and Change

2.3 Computation Tree Logic

Linear- and Branching-time Temporal Logics

Linear-time Temporal Logic : statements on all paths starting from a state.

- $s \models \mathrm{G}(x \leq 20)$ iff for all paths starting from s, always $x \leq 20$
- the universal quantification implicit in the LTL semantics can be made explicit:

$$s \models \mathsf{A}\varphi \quad \text{iff} \quad \pi \models \varphi \text{ for **all** paths } \pi \text{ starting in } s$$

- but what about "for every computation it is always possible to return to the initial state"?
- $\mathrm{AGF}\mathit{start}$ would not do the job. **Why?**

Branching-time Temporal Logic: statements on all paths or some path starting from a state.

- $s \models \mathrm{AG}(x \leq 20)$ iff for **every** path starting from s, always $x \leq 20$
- $s \models \mathrm{EG}(x \leq 20)$ iff for **some** path starting from s, always $x \leq 20$
- alternation of path quantifiers is allowed: $\mathrm{AGEF}\mathit{start}$

Branching Time: CTL

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

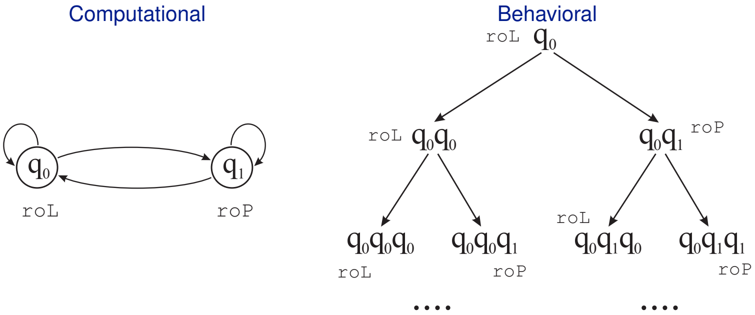
- Path quantifiers: A (for all paths), E (there is a path)
- Temporal operators: X (next), U (until), F (sometime), G (always)
- Two types of formulas: state formulae vs. path formulae

- "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

Computational vs. Behavioral Structures

CTL: semantics based on a notion of branching time:

- infinite tree of states obtained by unfolding the transition system.
- an instant in time can have several following instants.



Incomparable Expressivity:

- some properties are expressible in LTL, but not in CTL.
- some properties are expressible in CTL, but not in LTL.

Different model checking algorithms with different complexities.

Computation Tree Logic: Syntax

- Temporal logic on infinite trees [Clarke & Emerson 1981]

- State formulas Φ, Ψ:
  - $a \in AP$ — atoms
  - $\neg\Phi$ — negation
  - $\Phi \wedge \Psi$ — conjunction
  - Eϕ — for some path ϕ is true
  - Aϕ — for every path ϕ is true

- Path formulas ϕ:
  - $\mathrm{X}\Phi$ — *neXt* Φ
  - $\Phi \mathrm{U} \Psi$ — Φ *Until* Ψ

- Formulas in CTL are all and only the state formulas.

⇒ Notice that X and U alternate with A and E:
  - $\mathrm{AXAX}\Phi$ and $\mathrm{AXEX}\Phi \in CTL$
  - but $\mathrm{AXX}\Phi$ and $\mathrm{AEX}\Phi \notin CTL$

Computation Tree Logic: Syntax

Let $AP = \{x = 1, x < 2, x \geq 3\}$. Then, what about:

1. $\mathrm{EX}(x = 1)$?

Computation Tree Logic: Syntax

Let $AP = \{x = 1, x < 2, x \geq 3\}$. Then, what about:

1. $\mathrm{EX}(x = 1)$ ✓
2. $\mathrm{AX}(x = 1)$?

Computation Tree Logic: Syntax

Let $AP = \{x = 1, x < 2, x \geq 3\}$. Then, what about:

1. $\mathrm{EX}(x = 1)$ ✓
2. $\mathrm{AX}(x = 1)$ ✓
3. $(x < 2) \vee (x = 1)$?

Computation Tree Logic: Syntax

Let $AP = \{x = 1, x < 2, x \geq 3\}$. Then, what about:

1. $\mathsf{EX}(x = 1)$ ✓
2. $\mathsf{AX}(x = 1)$ ✓
3. $(x < 2) \vee (x = 1)$ ✓
4. $\mathsf{E}(x = 1 \wedge \mathsf{AX}(x \geq 3))$?

Computation Tree Logic: Syntax

Let $AP = \{x = 1, x < 2, x \geq 3\}$. Then, what about:

1. $\mathrm{EX}(x = 1)$ ✓
2. $\mathrm{AX}(x = 1)$ ✓
3. $(x < 2) \vee (x = 1)$ ✓
4. $\mathrm{E}(x = 1 \wedge \mathrm{AX}(x \geq 3))$ ✗
5. $\mathrm{EX}(true \mathrm{U}(x = 1))$?

Computation Tree Logic: Syntax

Let $AP = \{x = 1, x < 2, x \geq 3\}$. Then, what about:

1. $\mathsf{EX}(x = 1)$ ✓
2. $\mathsf{AX}(x = 1)$ ✓
3. $(x < 2) \vee (x = 1)$ ✓
4. $\mathsf{E}(x = 1 \wedge \mathsf{AX}(x \geq 3))$ ✗
5. $\mathsf{EX}(true\mathsf{U}(x = 1))$ ✗
6. $\mathsf{EX}(x = 1 \wedge \mathsf{AX}(x \geq 3))$?

Computation Tree Logic: Syntax

Let $AP = \{x = 1, x < 2, x \geq 3\}$. Then, what about:

1. $\mathrm{EX}(x = 1)$ ✓
2. $\mathrm{AX}(x = 1)$ ✓
3. $(x < 2) \vee (x = 1)$ ✓
4. $\mathrm{E}(x = 1 \wedge \mathrm{AX}(x \geq 3))$ ✗
5. $\mathrm{EX}(true\mathrm{U}(x = 1))$ ✗
6. $\mathrm{EX}(x = 1 \wedge \mathrm{AX}(x \geq 3))$ ✓
7. $\mathrm{EXA}(true\mathrm{U}(x = 1))$?

Computation Tree Logic: Syntax

Let $AP = \{x = 1, x < 2, x \geq 3\}$. Then, what about:

1. $\mathsf{EX}(x = 1)$ ✓
2. $\mathsf{AX}(x = 1)$ ✓
3. $(x < 2) \vee (x = 1)$ ✓
4. $\mathsf{E}(x = 1 \wedge \mathsf{AX}(x \geq 3))$ ✗
5. $\mathsf{EX}(true\mathrm{U}(x = 1))$ ✗
6. $\mathsf{EX}(x = 1 \wedge \mathsf{AX}(x \geq 3))$ ✓
7. $\mathsf{EXA}(true\mathrm{U}(x = 1))$ ✓

Derived Operators

Eventually:

$$\begin{array}{lcl} \mathrm{EF}\Phi & = & \mathsf{E}(true\mathrm{U}\Phi) \\ \mathrm{AF}\Phi & = & \mathsf{A}(true\mathrm{U}\Phi) \end{array}$$

potentially Φ

inevitably Φ

Always:

$$\begin{array}{lcl} \mathrm{EG}\Phi & = & \neg\mathrm{AF}\neg\Phi \\ \mathrm{AG}\Phi & = & \neg\mathsf{E}\mathrm{F}\neg\Phi \end{array}$$

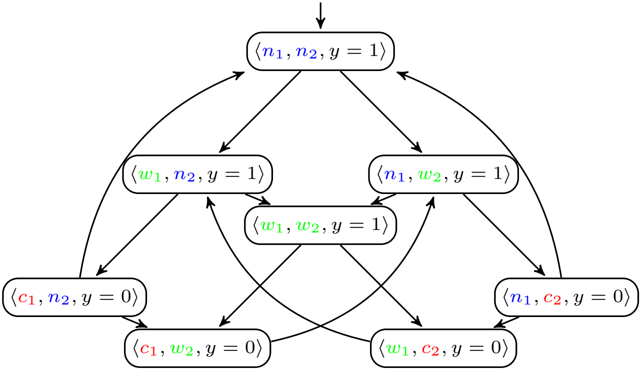
potentially always Φ

invariantly Φ

Alternatively, the syntax of CTL can be given as follows:

$$\Phi \quad ::= \quad a \mid \neg\Phi \mid \Phi \wedge \Phi \mid \mathsf{E}\mathrm{X}\Phi \mid \mathsf{A}\mathrm{X}\Phi \mid \mathsf{E}(\Phi\mathrm{U}\Phi) \mid \mathsf{A}(\Phi\mathrm{U}\Phi)$$

Examples: Mutual Exclusion Problem



1. mutual exclusion: $\mathrm{AG}(\neg crit_1 \vee \neg crit_2)$
2. starvation freedom (strong): $(\mathrm{AGA}F crit_1) \wedge (\mathrm{AGA}F crit_2)$
3. "every request will eventually be granted": $\mathrm{AG}(request \rightarrow \mathrm{AF}\, granted)$
4. "in every state it is possible to return to (one of) the initial state(s)": $\mathrm{AGEF}\, start$

Computation Tree Logic: Semantics

Let $M = \langle St, \longrightarrow, \mathcal{V} \rangle$ be a transition system, Φ, Ψ be state formulas, and γ be a path formula.

Definition 2.7 (Semantics of CTL: state formulas)

$(M, q) \models a$	iff	$a \in \mathcal{V}(q)$
$(M, q) \models \neg\Phi$	iff	$(M, q) \not\models \Phi$
$(M, q) \models \Phi \wedge \Psi$	iff	$(M, q) \models \Phi$ and $(M, q) \models \Psi$
$(M, q) \models \mathsf{E}\gamma$	iff	for some path λ starting from q, $(M, \lambda) \models \gamma$
$(M, q) \models \mathsf{A}\gamma$	iff	for all paths λ starting from q, $(M, \lambda) \models \gamma$

Definition 2.8 (Semantics of CTL: path formulas)

Computation Tree Logic: Semantics

Let $M = \langle St, \longrightarrow, \mathcal{V} \rangle$ be a transition system, Φ, Ψ be state formulas, and γ be a path formula.

Definition 2.7 (Semantics of CTL: state formulas)

$(M, q) \models a$	iff	$a \in \mathcal{V}(q)$
$(M, q) \models \neg\Phi$	iff	$(M, q) \not\models \Phi$
$(M, q) \models \Phi \wedge \Psi$	iff	$(M, q) \models \Phi$ and $(M, q) \models \Psi$
$(M, q) \models \mathsf{E}\gamma$	iff	for some path λ starting from q, $(M, \lambda) \models \gamma$
$(M, q) \models \mathsf{A}\gamma$	iff	for all paths λ starting from q, $(M, \lambda) \models \gamma$

Definition 2.8 (Semantics of CTL: path formulas)

Like in LTL!

Computation Tree Logic: Semantics

Let $M = \langle St, \longrightarrow, \mathcal{V} \rangle$ be a transition system, Φ, Ψ be state formulas, and γ be a path formula.

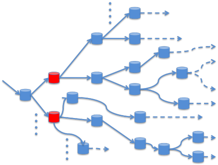
Definition 2.7 (Semantics of CTL: state formulas)

$$
\begin{array}{lll}
(M,q) \models a & \text{iff} & a \in \mathcal{V}(q) \\
(M,q) \models \neg\Phi & \text{iff} & (M,q) \not\models \Phi \\
(M,q) \models \Phi \wedge \Psi & \text{iff} & (M,q) \models \Phi \text{ and } (M,q) \models \Psi \\
(M,q) \models \mathsf{E}\gamma & \text{iff} & \text{for some path } \lambda \text{ starting from } q,\ (M,\lambda) \models \gamma \\
(M,q) \models \mathsf{A}\gamma & \text{iff} & \text{for all paths } \lambda \text{ starting from } q,\ (M,\lambda) \models \gamma
\end{array}
$$

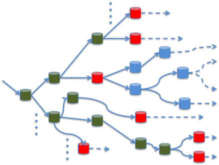
Definition 2.8 (Semantics of CTL: path formulas)

$$
\begin{array}{lll}
(M,\lambda) \models \mathrm{X}\Phi & \text{iff} & (M,\lambda[1]) \models \Phi \\
(M,\lambda) \models \Phi \mathrm{U} \Psi & \text{iff} & (M,\lambda[i]) \models \Psi \text{ for some } i \geq 0, \\
 & & \text{and } (M,\lambda[j]) \models \Phi \text{ for all } 0 \leq j < i
\end{array}
$$

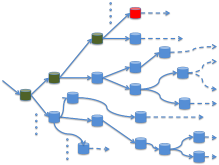
Semantics of CTL: Intuition



(d) $AX\varphi$



(e) $A\varphi U\psi$



(f) $E\varphi U\psi$

Semantics of CTL: Derived Operators

Recall that

$$\begin{aligned}
\mathsf{EF}\Phi &= \mathsf{E}(true \mathrm{U} \Phi)\\
\mathsf{AF}\Phi &= \mathsf{A}(true \mathrm{U} \Phi)\\
\mathsf{EG}\Phi &= \neg\mathsf{AF}\neg\Phi\\
\mathsf{AG}\Phi &= \neg\mathsf{EF}\neg\Phi
\end{aligned}$$

Then, by definition:

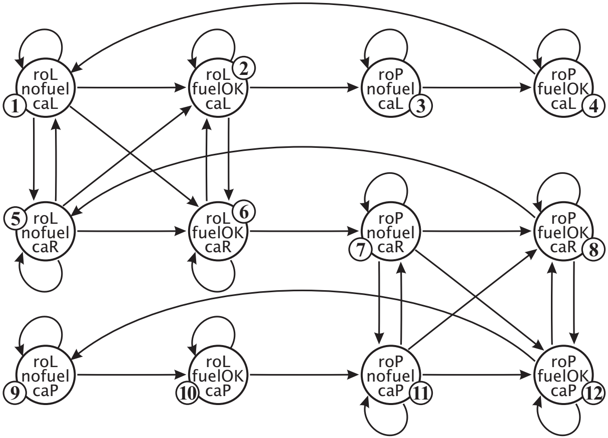
$(M,q) \models \mathsf{EF}\Phi$ iff for some path λ from q, for some $j \geq 0$, $(M,\lambda[j]) \models \Phi$

$(M,q) \models \mathsf{AF}\Phi$ iff for every path λ from q, for some $j \geq 0$, $(M,\lambda[j]) \models \Phi$

$(M,q) \models \mathsf{EG}\Phi$ iff for some path λ from q, for all $j \geq 0$, $(M,\lambda[j]) \models \Phi$

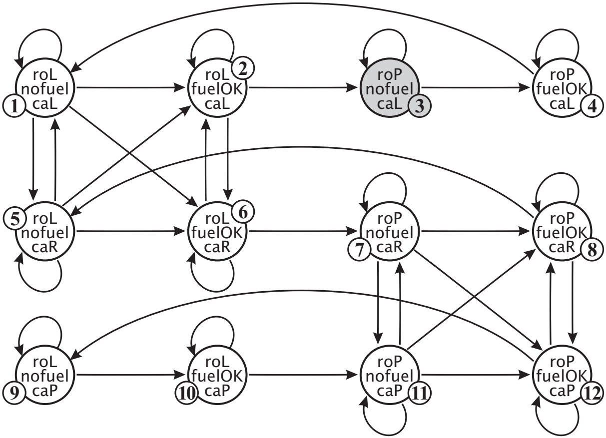
$(M,q) \models \mathsf{AG}\Phi$ iff for every path λ from q, for all $j \geq 0$, $(M,\lambda[j]) \models \Phi$

Example: Rocket and Cargo



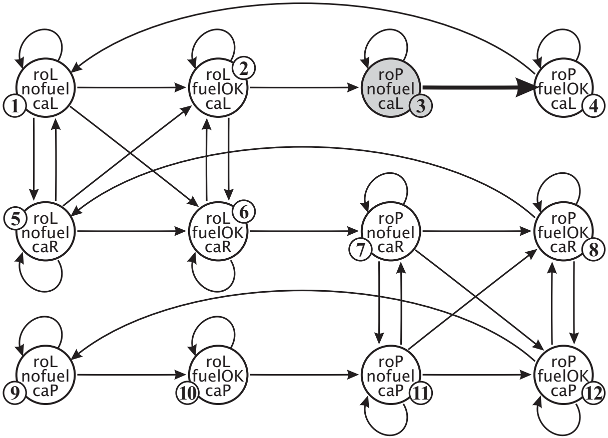
$3 \models \mathrm{EF}\,\mathrm{caP}?$

Example: Rocket and Cargo



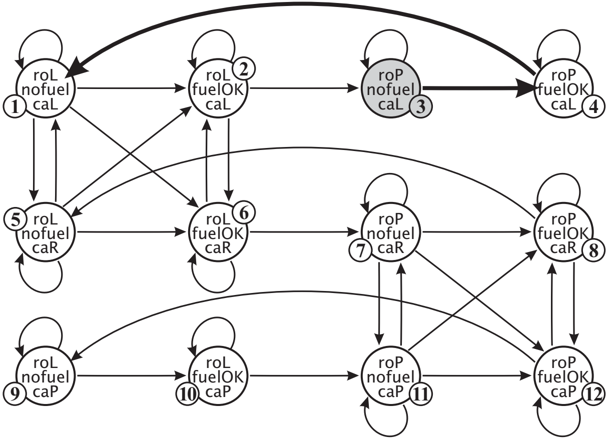
$3 \models \mathrm{EF}\,\mathrm{caP}?$

Example: Rocket and Cargo



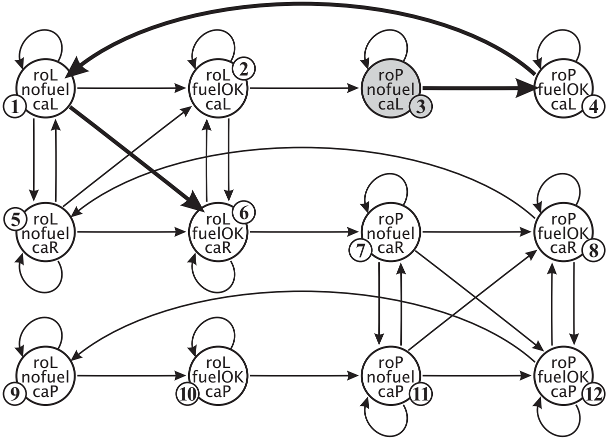
$3 \models \text{EF}\,\text{caP}?$

Example: Rocket and Cargo



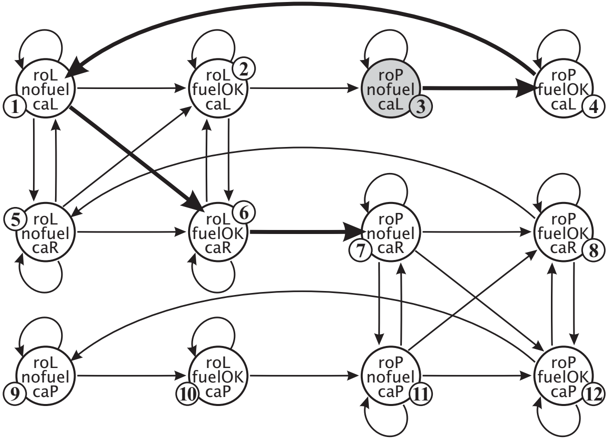
$3 \models \mathrm{EF}\,\mathsf{caP}?$

Example: Rocket and Cargo



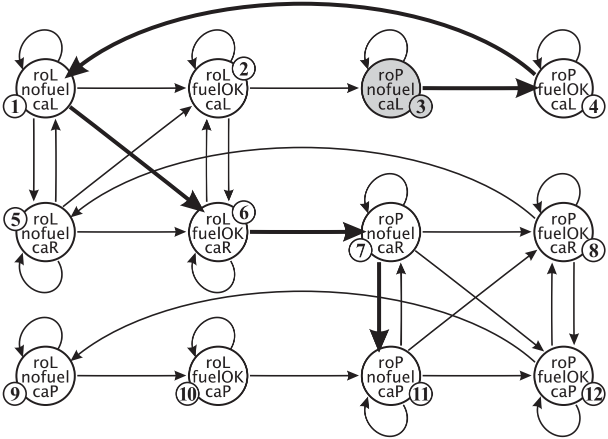
$3 \models \mathrm{EF}\, \mathrm{caP}?$

Example: Rocket and Cargo



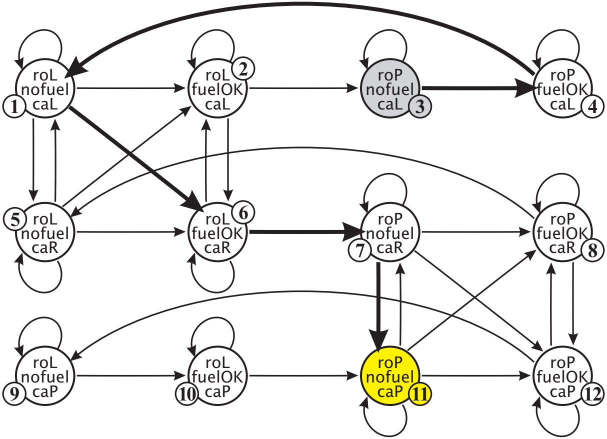
$3 \models \mathrm{EF}\,\mathsf{caP}?$

Example: Rocket and Cargo



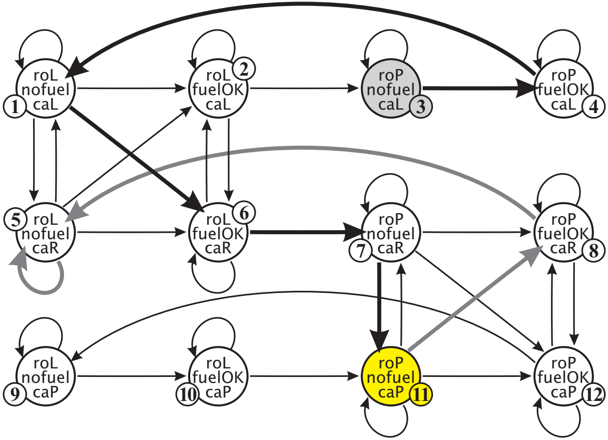
$3 \models \mathrm{EF}\, \mathsf{caP}$?

Example: Rocket and Cargo



$3 \models \mathrm{EF}\,\mathsf{caP}$ ✓

Example: Rocket and Cargo



$3 \models \mathrm{EF}\,\mathsf{caP}$ ✓

Semantics of Transition Systems

- For state formula Φ, the satisfaction set $Sat(\Phi)$ is defined as

$$Sat(\Phi) \quad = \quad \{s \in S \mid s \models \Phi\}$$

- TS **satisfies** formula Φ iff it is satisfied in all initial states:

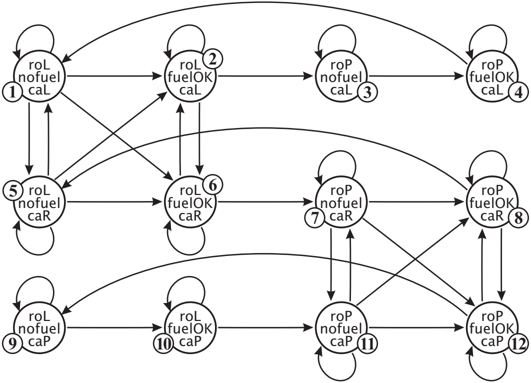
$$TS \models \Phi \quad \text{iff} \quad \text{for every } s_0 \in I, s_0 \models \Phi$$

  or equivalently, $I \subseteq Sat(\Phi)$

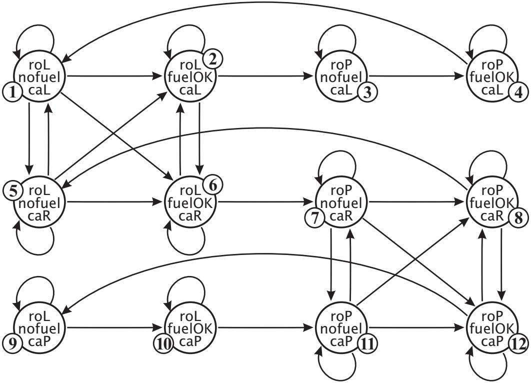
- **Beware**: $TS \not\models \Phi$ and $TS \not\models \neg\Phi$ is consistent!
  because of multiple initial states, for example, $s_0 \models \mathrm{EG}\Phi$ and $s_0' \not\models \mathrm{EG}\Phi$

- A formula Φ is **valid** iff it is true in every transition system.

Rocket and Cargo: More Properties

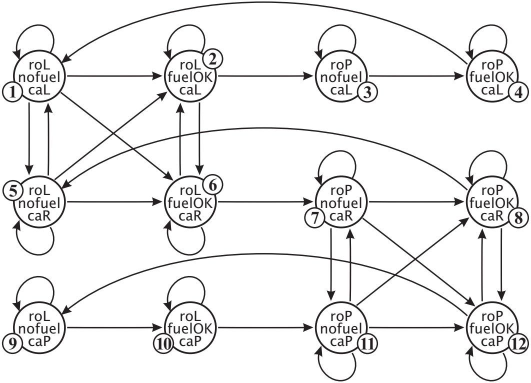


Rocket and Cargo: More Properties



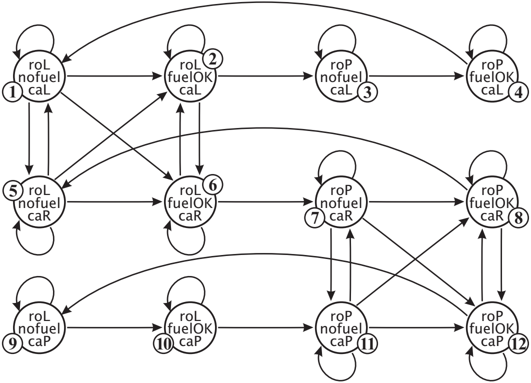
roL $\rightarrow$ EF roP?

Rocket and Cargo: More Properties



$\mathsf{roL} \to \mathrm{EF}\,\mathsf{roP}$ ✓

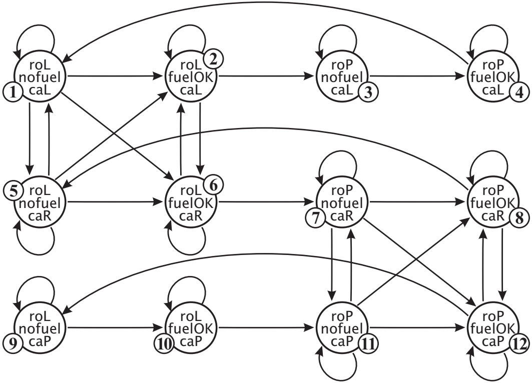
Rocket and Cargo: More Properties



$\mathsf{roL} \rightarrow \mathrm{EF}\,\mathsf{roP}$ ✓

$\mathrm{AG}(\mathsf{roL} \vee \mathsf{roP})$?

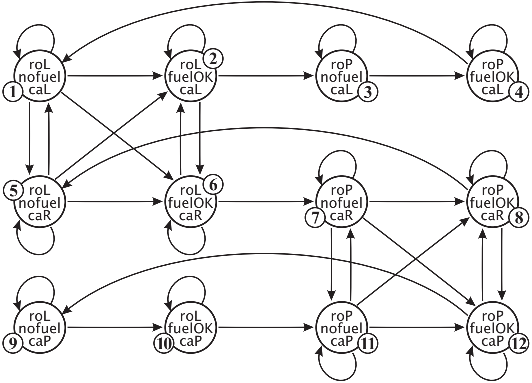
Rocket and Cargo: More Properties



$\mathsf{roL} \rightarrow \mathrm{EF}\,\mathsf{roP}$ ✓

$\mathrm{AG}(\mathsf{roL} \vee \mathsf{roP})$ ✓

Rocket and Cargo: More Properties

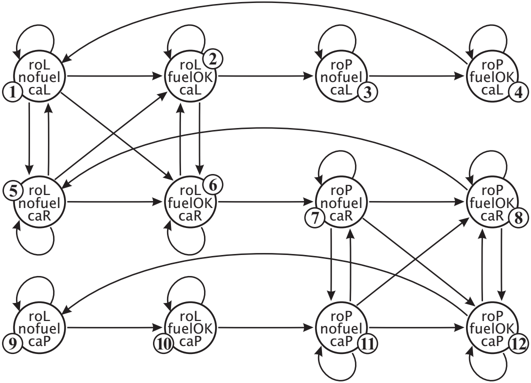


$\mathsf{roL} \rightarrow \mathrm{EF}\,\mathsf{roP}$ ✓

$\mathrm{AG}(\mathsf{roL} \vee \mathsf{roP})$ ✓

$\mathsf{roL} \rightarrow \mathrm{AX}(\mathsf{roP} \rightarrow \mathsf{nofuel})$?

Rocket and Cargo: More Properties

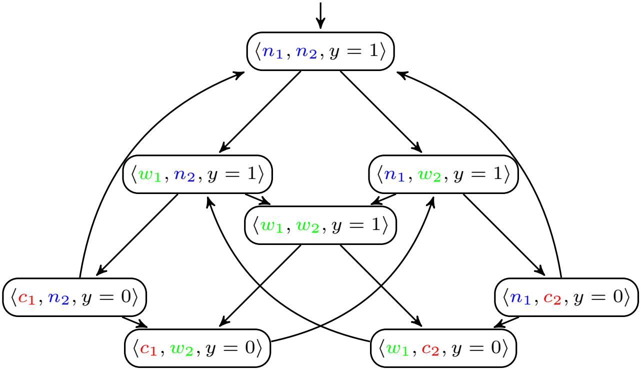


$\mathsf{roL} \to \mathrm{EF}\,\mathsf{roP}$ ✓

$\mathrm{AG}(\mathsf{roL} \vee \mathsf{roP})$ ✓

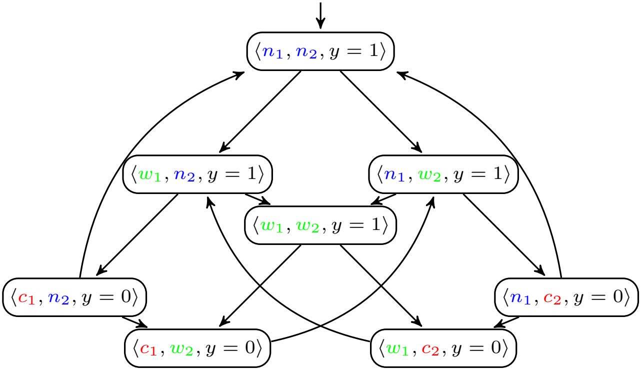
$\mathsf{roL} \to \mathrm{AX}(\mathsf{roP} \to \mathsf{nofuel})$ ✓

Example: Mutual Exclusion Problem



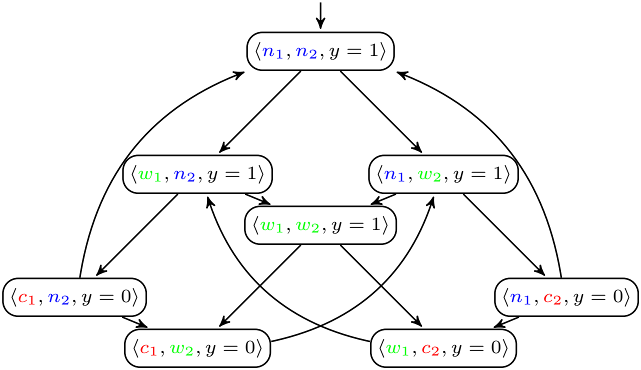
1. $M \models \mathrm{AG}(\neg cr_1 \vee \neg cr_2)$?

Example: Mutual Exclusion Problem



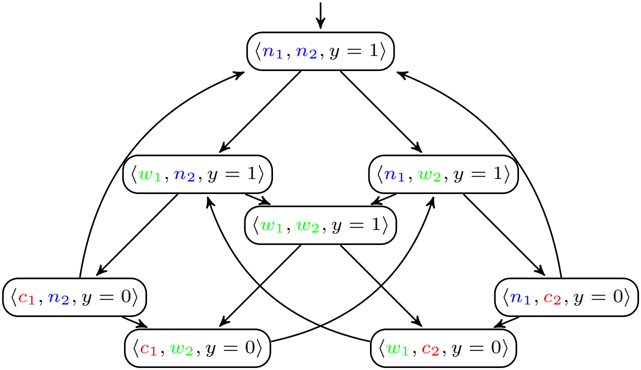
1. $M \models \mathrm{AG}(\neg cr_1 \vee \neg cr_2)$ ✓

Example: Mutual Exclusion Problem



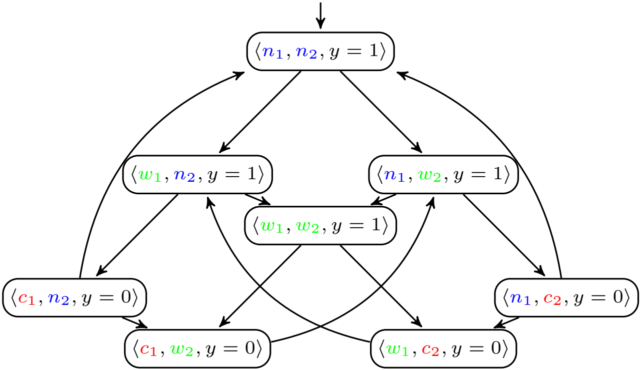
1. $M \models AG(\neg cr_1 \vee \neg cr_2)$ ✓
2. $M \models (AGAF\, cr_1) \wedge (AGAF\, cr_2)$?

Example: Mutual Exclusion Problem



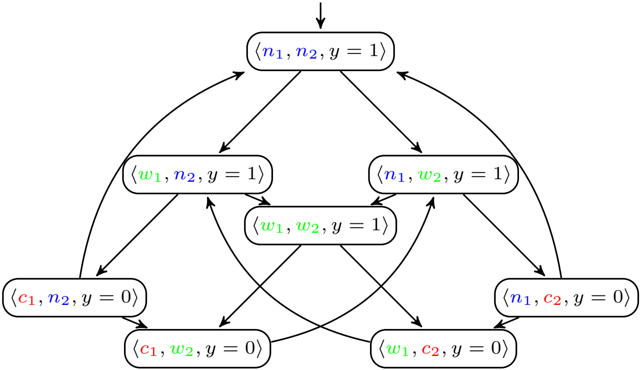
1. $M \models AG(\neg cr_1 \vee \neg cr_2)$ ✓
2. $M \not\models (AGAF\, cr_1) \wedge (AGAF\, cr_2)$ ✗

Example: Mutual Exclusion Problem



1. $M \models \mathrm{AG}(\neg cr_1 \vee \neg cr_2)$ ✓
2. $M \not\models (\mathrm{AGAF}\, cr_1) \wedge (\mathrm{AGAF}\, cr_2)$ ✗
3. $M \models (\mathrm{AGAF}\, wa_1 \rightarrow \mathrm{AGAF}\, cr_1) \wedge (\mathrm{AGAF}\, wa_2 \rightarrow \mathrm{AGAF}\, cr_2)$?

Example: Mutual Exclusion Problem



1. $M \models AG(\neg cr_1 \vee \neg cr_2)$ ✓
2. $M \not\models (AGAF\, cr_1) \wedge (AGAF\, cr_2)$ ✗
3. $M \not\models (AGAF\, wa_1 \rightarrow AGAF\, cr_1) \wedge (AGAF\, wa_2 \rightarrow AGAF\, cr_2)$ ✗

Definition 2.9 (Equivalence)

Two formulas Φ and Ψ are **equivalent**, or $\Phi \equiv \Psi$, iff $Sat(\Phi) = Sat(\Psi)$ for all TS.

$$\Phi \equiv \Psi \quad \text{if and only if} \quad \text{for all } TS, TS \models \Phi \Leftrightarrow TS \models \Psi$$

- Duality:

$$\begin{aligned}
\mathrm{AX}\Phi &\equiv \neg\mathrm{EX}\neg\Phi \\
\mathrm{EX}\Phi &\equiv \neg\mathrm{AX}\neg\Phi \\
\mathrm{AF}\Phi &\equiv \neg\mathrm{EG}\neg\Phi \\
\mathrm{EF}\Phi &\equiv \neg\mathrm{AG}\neg\Phi \\
\mathrm{AG}\Phi &\equiv \neg\mathrm{EF}\neg\Phi \\
\mathrm{EG}\Phi &\equiv \neg\mathrm{AF}\neg\Phi
\end{aligned}$$

Equivalences in CTL: Distributivity

- in LTL we have:

$$\begin{aligned} F(\phi \vee \psi) &\equiv F\phi \vee F\psi \\ G(\phi \wedge \psi) &\equiv G\phi \wedge G\psi \end{aligned}$$

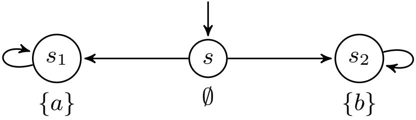
- and in CTL:

$$\begin{aligned} \mathsf{E}F(\Phi \vee \Psi) &\equiv \mathsf{E}F\Phi \vee \mathsf{E}F\Psi \\ AG(\Phi \wedge \Psi) &\equiv AG\Phi \wedge AG\Psi \end{aligned}$$

Equivalences in CTL

- But,

$$\begin{aligned} \mathrm{EG}(\Phi \vee \Psi) &\not\equiv \mathrm{EG}\,\Phi \vee \mathrm{EG}\,\Psi \\ \mathrm{AF}(\Phi \wedge \Psi) &\not\equiv \mathrm{AF}\,\Phi \wedge \mathrm{AF}\,\Psi \end{aligned}$$



- $s \models \mathrm{AF}(a \vee b)$ as for every path π from s, $\pi \models \mathrm{F}(a \vee b)$
- but $s(s_1)^\omega \not\models \mathrm{F}b$. Thus, $s \not\models \mathrm{AF}b$
- a similar line of reasoning shows that $s \not\models \mathrm{AF}a$.
- hence, $s \not\models \mathrm{AF}a \vee \mathrm{AF}b$.

Theorem 2.10 (Fixpoint characterization of branching-time operators)

*The following formulas are **valid** in CTL:*

- $E\varphi_1 U \varphi_2 \quad \leftrightarrow \quad \varphi_2 \vee (\varphi_1 \wedge EX\, E\varphi_1 U \varphi_2)$
- $EF\varphi \quad \leftrightarrow \quad \varphi \vee EX\, EF\varphi$
- $EG\varphi \quad \leftrightarrow \quad \varphi \wedge EX\, EG\varphi$

- $A\varphi_1 U \varphi_2 \quad \leftrightarrow \quad \varphi_2 \vee (\varphi_1 \wedge AX\, A\varphi_1 U \varphi_2)$.
- $AF\varphi \quad \leftrightarrow \quad \varphi \vee AX\, AF\varphi$
- $AG\varphi \quad \leftrightarrow \quad \varphi \wedge AX\, AG\varphi$

- $E\varphi_1 U \varphi_2$ and $A\varphi_1 U \varphi_2$ are the least fixed points.
- $EG\varphi$ and $AG\varphi$ are the greatest fixed points.

Fixpoint Equivalences in CTL

What is the importance of fixpoint equivalences?

- They say that paths satisfying CTL specifications can be constructed incrementally, step by step
- Moreover, solutions to the verification problem can be obtained iteratively
- ...which will be used in most model checking algorithms

Motivating Example: Rescue Robots

- Everybody is safe:

  $\bigwedge_{i \in \mathit{People}} \mathsf{safe_i}$

- Everybody will eventually be safe:

  $\bigwedge_{i \in \mathit{People}} \mathrm{AF}\, \mathsf{safe_i}$

  Another interpretation: $\mathrm{AF}\left(\bigwedge_{i \in \mathit{People}} \mathsf{safe_i}\right)$

- Everybody will always be safe, from some moment on:

  Cannot be expressed in CTL! but in CTL*: $\bigwedge_{i \in \mathit{People}} \mathrm{AFG}\, \mathsf{safe_i}$

  Equivalently: $\mathrm{AFG}\left(\bigwedge_{i \in \mathit{People}} \mathsf{safe_i}\right)$

- Everybody may eventually be safe, if everything goes fine:

  $\bigwedge_{i \in \mathbf{People}} \mathrm{EF}\, \mathsf{safe_i}$

  Another interpretation: $\mathrm{EF}\left(\bigwedge_{i \in \mathbf{People}} \mathsf{safe_i}\right)$

Motivating Example: Rescue Robots

- Whenever person i gets in trouble, she will eventually be rescued:

  $$AG(\neg \mathsf{safe_i} \rightarrow AF\, \mathsf{safe_i})$$

- If person i gets outside the building then she will never be in danger anymore:

  $$AG(\mathsf{outside_i} \rightarrow AG\, \mathsf{safe_i})$$

- Person i may be rescued without any robot ever entering the building, but guaranteed rescue requires some robots to enter:

  $$\mathsf{E}(\bigwedge_{j \in \textbf{\textit{Robots}}} \mathsf{outside_j})\, U\, \mathsf{safe_i} \quad \wedge \quad \neg \mathsf{A}(\bigwedge_{j \in \textbf{\textit{Robots}}} \mathsf{outside_j})\, U\, \mathsf{safe_i}$$

Motivating Example: Voting

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

  $\mathrm{AG}\neg\mathsf{revealed}_i$

- The system does not issue receipts:

  $\mathrm{AG}\neg\mathsf{receipt}_i$

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

  Interpretation 1: The voter may vote, and may refrain from voting:

  $\mathrm{EF}\,\mathsf{voted}_i \wedge \mathrm{EG}\neg\mathsf{voted}_i$

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

  Cannot be expressed in CTL!

- If the voter votes, the system will not reveal afterwards how she voted:

  $\bigwedge_{c\in \mathit{Candidates}} \mathrm{AG}\left(\mathsf{voted}_{i,c} \rightarrow \mathrm{AG}(\neg\mathsf{revealedVote}_{i,c})\right)$

Definition 2.11 (Equivalence)

a CTL formula Φ and an LTL formula ϕ are **equivalent**, or $\Phi \equiv \phi$, iff for every transition system TS,

$$TS \models \Phi \quad \text{iff} \quad TS \models \phi$$

Theorem 2.12 (Clarke & Draghicescu, 1988)

Let Φ be a CTL formula and let ϕ be the LTL formula obtained by deleting all path quantifiers in Φ. Then,

$$\Phi \equiv \phi \textit{ or there is no LTL formula equivalent to } \Phi$$

LTL and CTL are incomparable

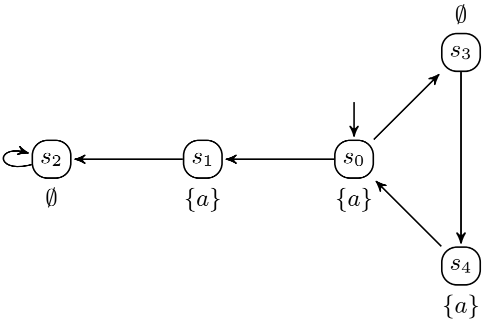
- Some formulas in LTL cannot be expressed in CTL:
  - $\mathrm{FG}a$
  - $\mathrm{F}(a \land \mathrm{X}a)$

- Some formulas in CTL cannot be expressed in LTL:
  - $\mathrm{AFAG}a$
  - $\mathrm{AF}(a \land \mathrm{AX}a)$
  - $\mathrm{AGEF}a$

$\Rightarrow$ cannot be expressed = there is no equivalent formula

Comparison between LTL and CTL

$\mathrm{F}(a \wedge \mathrm{X}a)$ is **not** equivalent to $\mathrm{AF}(a \wedge \mathrm{AX}a)$

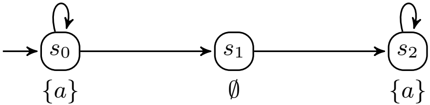


$$s_0 \models \mathrm{F}(a \wedge \mathrm{X}a) \quad \text{but} \quad s_0 \not\models \underbrace{\mathrm{AF}(a \wedge \mathrm{AX}a)}_{\text{consider path } s_0 s_1 (s_2)^\omega}$$

$\Rightarrow$ There is no LTL formula equivalent to $\mathrm{AF}(a \wedge \mathrm{AX}a)$

Actually, there is no CTL formula equivalent to $\mathrm{F}(a \wedge \mathrm{X}a)$.

$\mathrm{AFAG}a$ is **not** equivalent to $\mathrm{FG}a$



$s_0 \models \mathrm{FG}a$ but $s_0 \not\models \underbrace{\mathrm{AFAG}a}_{\text{consider path } (s_0)^\omega}$

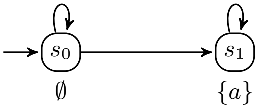
$\Rightarrow$ Again, there is no LTL formula equivalent to $\mathrm{AFAG}a$

Actually, there is no CTL formula equivalent to $\mathrm{FG}a$.

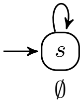
Comparison between LTL and CTL

Formula $\mathrm{AGEF}a$ cannot be expressed in LTL

- proof by contradiction: suppose that for some ϕ in LTL, $\phi \equiv \mathrm{AGEF}a$.
- consider



(g) TS

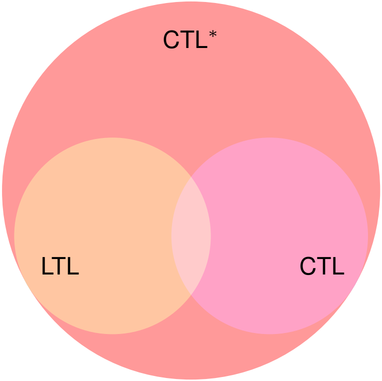


(h) TS'

- $TS \models \mathrm{AGEF}a$. Hence, by assumption, $TS \models \phi$
- $Paths(TS') \subseteq Paths(TS)$. Thus, $TS' \models \phi$
- but, $TS' \not\models \mathrm{AGEF}a$ as $s^\omega \not\models \mathrm{GEF}a$

Beyond LTL and CTL: CTL*

- The incomparability of LTL and CTL motivates their combination: CTL*
- CTL* combines the syntax of LTL and CTL: it strictly extends both.
- The semantics is obtained by extending LTL's with CTL's clauses for path quantifiers.
- Theoretical interest but seldom used in applications.



Conclusions

1. Linear-time Temporal Logic (LTL)
   1. Syntax
   2. Semantics
   3. Expressivity
2. Computation-tree Temporal Logic (CTL)
   1. Syntax
   2. Semantics
   3. Expressivity
3. Comparison between LTL et CTL
   1. CTL*

References

E. A. Emerson (1990).
Temporal and modal logic.
Handbook of Theoretical Computer Science, volume B, pp. 995–1072, Elsevier.

M. Fisher. (2006).
Temporal Logics.
Kluwer.

P. Schnoebelen. (2003).
The complexity of temporal model checking.
Advances in Modal Logics, World Scientific.

Part 3: Verification and Complexity

Part 3: Verification and Complexity

3.1 Decision Problems

What's the Use of MAS Logics?

Modelling & Design

- Modeling systems (the frameworks provide intuitive conceptual structures, and a systematic approach);
- Specifying desirable properties of systems.

Analysis & Verification

- Reasoning about concrete systems;
- Correctness testing.

What's the Use of MAS Logics?

Automatic Generation of Behaviors

- Programming with executable specifications;
- Automatic planning.

Philosophy of Mind and Agency

- Characterization of mental attitudes;
- Discussion of rational agents;
- Testing rationality assumptions.

In this course, we focus on **modeling, specification, and verification**

Motivating Example: Rescue Robots

Tasks

- Check whether everybody will eventually be safe.
- Verify that if person i gets outside the building then she will never be in danger anymore.
- Check if, in all rescue missions, everybody will always be safe, from some moment on.
- Show or disprove that everybody may eventually be safe, if everything goes fine.

Motivating Example: Voting

Tasks

- Verify that the voter can vote, and can refrain from voting.
- Design a system that does not issue receipts.
- Show (or disprove) that no system will ever reveal how a particular voter voted.

Logical Problems

- Decision problem: given representation of an instance, decide whether it belongs to the set of "good" instances.

- Typical logical problems: validity, satisfiability, and model checking:
- **Validity**: given formula φ, determine if φ is valid (true in every model)
- **Satisfiability**: given formula φ, determine if φ is satisfiable (true in some model)
- **Model checking**: given formula φ and model M, determine if φ is true in M

Motivating Example: Rescue Robots

- Check whether everybody will eventually be safe.

  Model checking of $\bigwedge_{i \in People} \mathrm{AF}\, \mathsf{safe_i}$
  in the model of the rescue mission

- Verify that if person i gets outside the building then she will never be in danger anymore.

  Model checking of $\mathrm{AG}(\mathsf{outside_i} \to \mathrm{AG}\, \mathsf{safe_i})$
  in the model of the rescue mission

- Check if, in all rescue missions, everybody will always be safe, from some moment on.

  Validity checking of $\bigwedge_{i \in \mathsf{People}} \mathrm{FG}\, \mathsf{safe_i}$

- Show or disprove that everybody may eventually be safe, if everything goes fine.

  Validity checking of $\bigwedge_{i \in \mathsf{People}} \mathrm{EF}\, \mathsf{safe_i}$

Motivating Example: Voting

- Verify that the voter can vote, and can refrain from voting:

  Model checking of $\mathrm{EF}\,\mathsf{voted_i} \wedge \mathrm{EG}\neg\mathsf{voted_i}$
  in the model of the voting protocol

- Design a system that does not issue receipts:

  Satisfiability checking of $\mathrm{AG}(\bigwedge_{c \in Voters} \neg\mathsf{receipt_i})$

- Show (or disprove) that no system will ever reveal how any voter i voted:

  Validity checking of $\mathrm{AG}(\bigwedge_{i \in Voters} \neg\mathsf{revealed_i})$

Logical Problems

Decision problem can be seen as a Yes/No question.
The answer depends on the input, i.e., the actual parameters.

Algorithmic view:

We want a **machine** (algorithm) to answer the question.
We give the input, the machine returns the answer!

We use Turing machines as models of computation.

A. Turing, *Intelligent Machinery*, 1948:

> *... an unlimited memory capacity obtained in the form of an infinite tape marked out into squares, on each of which a symbol could be printed. At any moment there is one symbol in the machine [...]. The machine can alter the scanned symbol, and its behavior is in part determined by that symbol [...]. [T]he tape can be moved back and forth through the machine, this being one of the elementary operations of the machine.*

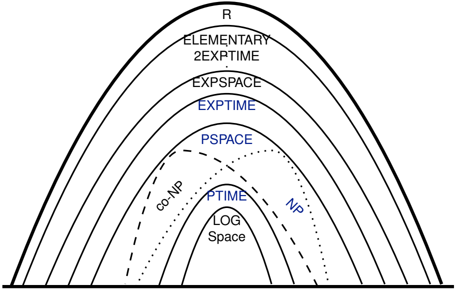
Will that work?

It depends on how difficult the question is
$\rightsquigarrow$ Computational Complexity

Some Complexity Classes

Time and space are the two parameters of the complexity of decision problems.

- PTIME (polynomial time): problems solvable in polynomial time by a deterministic Turing machine.
- NP (polynomial non-deterministic time): problems solvable in polynomial time by a non-deterministic Turing machine.
- PSPACE (polynomial space): problems solvable by a (deterministic) Turing machine that uses only polynomially many memory cells.
- EXPTIME (exponential time): problems solvable in exponential time by a deterministic Turing machine.



Looking for Completeness

Ideally, we would like to characterize precisely the complexity of a decision problem $\rightsquigarrow$ Completeness

A decision problem is complete wrt a complexity class C iff

- it belongs to C: there is a Turing machine to decide the problem in C (membership)
- we cannot do better: there is a reduction to another C-complete problem (hardness)

Church's thesis

If there is an algorithm (in C), then there is a Turing machine (in C).

Complexity

Theoretical complexity has many deficiencies: it refers only to the worst (hardest) instance in the set, neglects coefficients in the function characterizing the complexity, etc.

What is this about?

Scalability!

- The problems in PTIME can in principle be solved efficiently by a brute force approach.
- NP collects problems that can be solved fast if one comes up with the right heuristics.
- Problems in EXPTIME do not scale even with smart heuristics; they are inherently exponential in terms of the time that they demand.

Part 3: Verification and Complexity

3.2 Complexity of Model Checking Temporal Logics

Complexity of CTL Model Checking

Theorem 3.1

Model checking of CTL is PTIME-complete, and can be done in linear time with respect to the size of the model and the length of the formula.

Is that precise enough...?

- What does linear mean precisely?
- And how do we measure the size of the model?

Theorem 3.2

Model checking of CTL is PTIME-complete, and can be done in time $O(m \cdot l)$ where m is the number of transitions in the model and the l is the number of subformulas in the formula.

Complexity of LTL Model Checking

Theorem 3.3

Model checking of LTL is PSPACE-complete, and can be done in linear time with respect to the size of the model and exponential time wrt the length of the formula.

Theorem 3.4

Model checking of LTL is PSPACE-complete, and can be done in time $O(m \cdot 2^l)$ where m is the number of transitions in the model and the l is the number of subformulas in the formula.

Summary of Complexity Results

	Model Cheking	Satisfiability
CTL	PTIME-complete	EXPTIME-complete
LTL	PSPACE-complete	PSPACE-complete
CTL*	PSPACE-complete	2EXPTIME-complete

Take-home message:

Model checking LTL and CTL can be done efficiently.

Several model checking tools available

- CTL: NuSMV, TAPAs, ...
- LTL: Spin, LTSmin, PAT, ...

Check:
`https://en.wikipedia.org/wiki/List_of_model_checking_tools`

References

Edmund M. Clarke, Orna Grumberg and Doron A. Peled (1999),
Model Checking.
MIT Press: Cambridge, MA.

Ph. Schnoebelen (2003),
The Complexity of Temporal Model Checking.
In *Advances in Modal Logics: Proceedings of AiML 2002*, World Scientific.

Part 4: Reasoning about Strategic Abilities

What Agents Can Achieve

- So far, we specified how things must or may evolve.
- In multi-agent systems, it is often very important to know **who** can **make them evolve** in a particular way.

Motivating Example: Rescue Robots

Properties to express

- The robots can rescue all the people in the building.
- If person i gets outside the building then she can stay away from trouble forever.
- Person i may be rescued without any robot ever entering the building, but guaranteed rescue requires some robots to enter.
- The robots can rescue all the people.
- The robots can rescue all the people, and they know that they can.
- The robots can rescue all the people, and they know how to do it.

Motivating Example: Voting

Properties to express

Privacy: The system cannot reveal how a particular voter voted.

Receipt-freeness: The voter cannot gain any information (a receipt) which can be used to prove to a coercer that she voted in a certain way.

Coercion-resistance: The voter cannot cooperate with the coercer to prove to him that she voted in a certain way.

What Agents Can Achieve

McCarthy and Hayes, *Some Philosophical Problems from the Standpoint of Artificial Intelligence*, 1969:

> *We want a computer program that decides what to do by inferring in a formal language [i.e., a logic] that a certain strategy will achieve a certain goal.*
> *This requires formalizing concepts of causality, ability, and knowledge.*

1988 : Belnap and Perloff: logic of "seeing to it that" (STIT)

1985 : Parikh: Game Logic

2000 : Pauly: Coalition Logic

2002 : Alur, Henzinger and Kupferman: Alternating-time Temporal Logic

A word of caution:

- The main logic-based approaches to reasoning about strategic play are weak in game-theoretic sense.
- They are based on the worst case analysis ("surely winning") and binary winning conditions.

  $\leadsto$ roughly correspond to maxmin analysis in two-player zero-sum games with binary payoffs.

Part 4: Reasoning about Strategic Abilities

4.1 Concurrent Game Structures

Concurrent game structures (aka multi-player game frames)

Generalization of repeated games by allowing different strategic games to be played at different stages.
Or, equivalently, generalization of transition systems to a multi-agent setting.

Main features:

- Agents, actions, transitions, atomic propositions
- Atomic propositions + interpretation
- Actions are abstract

Concurrent Game Structures

Fix a set AP of atomic propositions (or atoms).

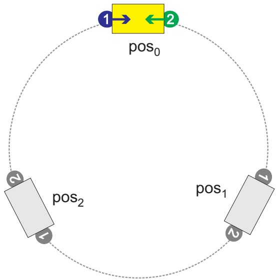
Definition 4.1 (Concurrent Game Structure)

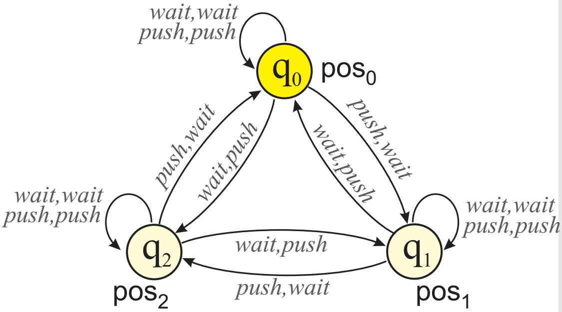
A **concurrent game structure** is a tuple $M = \langle \mathbb{A}\mathrm{gt}, St, \mathcal{V}, Act, d, o \rangle$, where:

- $\mathbb{A}\mathrm{gt} = \{1, \ldots, k\}$ is a finite set of agents
- $St = \{q_1, q_2, \ldots\}$ is a set of states
- $\mathcal{V} : AP \to 2^{St}$ is a valuation of atoms
- $Act = \{\alpha_1, \ldots, \alpha_m\}$ is a finite set of actions
- protocol $d : \mathbb{A}\mathrm{gt} \times St \to 2^{Act}$ defines actions available to an agent in a state
- $o : St \times Act^{\mathbb{A}\mathrm{gt}} \to St$ is a (partial) transition function that assigns outcome states $q' = o(q, \alpha_1, \ldots, \alpha_k)$ to states and joint actions.

- Transitions are deterministic.
- Transition $o(q, \alpha_1, \ldots, \alpha_k)$ is defined iff all α_a are enabled in q, that is, $\alpha_a \in d(a, q)$ for all agents $a \in \mathbb{A}\mathrm{gt}$.

Example: Robots and Carriage





Consider CGS $M = \langle \mathbb{A}\mathrm{gt}, St, \mathcal{V}, Act, d, o \rangle$, where:

- $\mathbb{A}\mathrm{gt} = \{1, 2\}$
- $St = \{q_0, q_1, q_2\}$
- for $i \leq 2$, $\mathcal{V}(\mathsf{pos_i}) = \{q_i\}$
- $Act = \{push, wait\}$ and for every agent a and state q, $d(a, q) = Act$
- $o(q_i, push, push) = o(q_i, wait, wait) = q_i$,
  $o(q_i, push, wait) = q_{i+1} \pmod 3$,
  $o(q_i, wait, push) = q_{i-1} \pmod 3$.

Part 4: Reasoning about Strategic Abilities

4.2 Alternating-Time Temporal Logic

What Agents Can Achieve: ATL

- **ATL**: **Alternating-time Temporal Logic** (Alur et al. 1997-2002)
- Motto: Temporal Logic meets Game Theory
- Overarching Idea: cooperation modalities

$\langle\langle A\rangle\rangle\Phi$: coalition A has a joint strategy to achieve goal Φ (independently of whatever agents in $\mathbb{A}\mathrm{gt} \setminus A$ do.)

Generalization of the branching-time temporal logic CTL* (and therefore both LTL and CTL).

Example Formulas

- $\langle\langle jamesbond \rangle\rangle \mathrm{G}\,(\mathsf{ski} \wedge \neg \mathsf{getBurned})$:
  “James Bond can always go skiing without getting burned”

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

Alternating-time Temporal Logic: Syntax

Definition 4.2 (ATL*)

State (Φ) and path (γ) formulas in ATL* are defined as follows, where $q \in AP$ and $A \subseteq Ag$:

$$\begin{array}{rcl} \Phi & ::= & \mathsf{p} \mid \neg\Phi \mid \Phi \wedge \Phi \mid \langle\!\langle A \rangle\!\rangle\gamma \\ \gamma & ::= & \Phi \mid \neg\gamma \mid \gamma \wedge \gamma \mid \mathrm{X}\gamma \mid \gamma\mathrm{U}\gamma \end{array}$$

Formulas in ATL* are all and only the state formulas.

$\langle\!\langle A \rangle\!\rangle\gamma$: "the agents in coalition A have a strategy to achieve γ" (no matter what the agents in $\overline{A} = \mathbb{A}\mathrm{gt} \setminus A$ do).

As before, *eventually* F and *always* G can be defined from *until* U:

- $\mathrm{F}\gamma \equiv true\mathrm{U}\gamma$
- $\mathrm{G}\gamma \equiv \neg\mathrm{F}\neg\gamma$

$[\![A]\!]\gamma \equiv \neg\langle\!\langle A \rangle\!\rangle\neg\gamma$: no matter what the agents in coalition A do, the outcome γ is unavoidable (agents in A cannot enforce $\neg\gamma$).

Alternating-time Temporal Logic: Syntax

Two main syntactic variants:

- ATL*: no syntactic restrictions
- "Vanilla" ATL: formulas in the ATL fragment of ATL* are obtained from Def. 4.2 by restricting path formulas γ as follows, where Φ is a state formula:

$$\gamma \quad ::= \quad \mathrm{X}\Phi \mid \Phi \mathrm{U} \Phi$$

  Temporal operators apply to cooperation formulas only.

  This is equivalent to the following syntax:

$$\Phi \quad ::= \quad \mathsf{p} \mid \neg\Phi \mid \Phi \wedge \Phi \mid \langle\!\langle A \rangle\!\rangle X \Phi \mid [\![A]\!] X \Phi \mid \langle\!\langle A \rangle\!\rangle (\Phi U \Phi) \mid [\![A]\!] (\Phi U \Phi)$$

Syntax: Examples

Let $AP = \{x = 1, x < 2, x \geq 3\}$ and $\mathbb{A}\mathrm{gt} = \{1, 2, 3\}$. Then, what about:

1. $\langle\langle 1, 2\rangle\rangle \mathrm{X}(x = 1)$?
2.
3.
4.
5.
6.
7.

Syntax: Examples

Let $AP = \{x = 1, x < 2, x \geq 3\}$ and $\mathbb{A}\mathrm{gt} = \{1, 2, 3\}$. Then, what about:

1. $\langle\langle 1, 2\rangle\rangle \mathrm{X}(x = 1)$ ✓
2. $[\![2, 3]\!] \mathrm{X}(x = 1)$?
3.
4.
5.
6.
7.

Syntax: Examples

Let $AP = \{x = 1, x < 2, x \geq 3\}$ and $\mathbb{A}\mathrm{gt} = \{1, 2, 3\}$. Then, what about:

1. $\langle\!\langle 1, 2 \rangle\!\rangle \mathrm{X}(x = 1)$ ✓
2. $[\![2, 3]\!] \mathrm{X}(x = 1)$ ✓
3. $\langle\!\langle 1, 2 \rangle\!\rangle (\mathrm{F}(x < 2) \vee \mathrm{G}(x = 1))$?
4.
5.
6.
7.

Syntax: Examples

Let $AP = \{x = 1, x < 2, x \geq 3\}$ and $\mathbb{A}\mathrm{gt} = \{1, 2, 3\}$. Then, what about:

1. $\langle\langle 1, 2\rangle\rangle \mathrm{X}\, (x = 1)$ ✓
2. $[\![2, 3]\!] \mathrm{X}\, (x = 1)$ ✓
3. $\langle\langle 1, 2\rangle\rangle (\mathrm{F}\, (x < 2) \vee \mathrm{G}\, (x = 1))$ ✓
4. $(x = 1 \wedge [\![1, 2]\!] \mathrm{U}\, (x \geq 3))$?
5.
6.
7.

Syntax: Examples

Let $AP = \{x = 1, x < 2, x \geq 3\}$ and $\mathbb{A}\mathrm{gt} = \{1, 2, 3\}$. Then, what about:

1. $\langle\langle 1, 2\rangle\rangle \mathrm{X}\,(x = 1)$ ✓
2. $[\![2, 3]\!] \mathrm{X}\,(x = 1)$ ✓
3. $\langle\langle 1, 2\rangle\rangle (\mathrm{F}\,(x < 2) \vee \mathrm{G}\,(x = 1))$ ✓
4. $(x = 1 \wedge [\![1, 2]\!] \mathrm{U}\,(x \geq 3))$ ✗
5. $\langle\langle \emptyset \rangle\rangle \mathrm{X}\,(true \mathrm{U}\,(x = 1))$?
6.
7.

Syntax: Examples

Let $AP = \{x = 1, x < 2, x \geq 3\}$ and $\mathbb{A}\mathrm{gt} = \{1, 2, 3\}$. Then, what about:

1. $\langle\langle 1, 2\rangle\rangle \mathrm{X}(x = 1)$ ✓
2. $[\![2, 3]\!] \mathrm{X}(x = 1)$ ✓
3. $\langle\langle 1, 2\rangle\rangle (\mathrm{F}(x < 2) \vee \mathrm{G}(x = 1))$ ✓
4. $(x = 1 \wedge [\![1, 2]\!] \mathrm{U}(x \geq 3))$ ✗
5. $\langle\langle \emptyset\rangle\rangle \mathrm{X}(true \mathrm{U}(x = 1))$ ✓
6. $[\![1, 2]\!] \mathrm{X}(x = 1 \wedge \langle\langle 2, 3\rangle\rangle \mathrm{GX}(x \geq 3))$?
7.

Syntax: Examples

Let $AP = \{x = 1, x < 2, x \geq 3\}$ and $\mathbb{A}\mathrm{gt} = \{1, 2, 3\}$. Then, what about:

1. $\langle\langle 1, 2\rangle\rangle \mathrm{X}(x = 1)$ ✓
2. $[\![2, 3]\!] \mathrm{X}(x = 1)$ ✓
3. $\langle\langle 1, 2\rangle\rangle (\mathrm{F}(x < 2) \vee \mathrm{G}(x = 1))$ ✓
4. $(x = 1 \wedge [\![1, 2]\!] \mathrm{U}(x \geq 3))$ ✗
5. $\langle\langle \emptyset\rangle\rangle \mathrm{X}(true \mathrm{U}(x = 1))$ ✓
6. $[\![1, 2]\!] \mathrm{X}(x = 1 \wedge \langle\langle 2, 3\rangle\rangle \mathrm{G}\mathrm{X}(x \geq 3))$ ✓
7. $\langle\langle 2, 4\rangle\rangle \mathrm{X}[\![2, 3]\!](true \mathrm{U}(x = 1))$?

Syntax: Examples

Let $AP = \{x = 1, x < 2, x \geq 3\}$ and $\mathbb{A}\mathrm{gt} = \{1, 2, 3\}$. Then, what about:

1. $\langle\langle 1, 2\rangle\rangle \mathrm{X}\,(x = 1)$ ✓
2. $[\![2, 3]\!] \mathrm{X}\,(x = 1)$ ✓
3. $\langle\langle 1, 2\rangle\rangle (\mathrm{F}\,(x < 2) \vee \mathrm{G}\,(x = 1))$ ✓
4. $(x = 1 \wedge [\![1, 2]\!] \mathrm{U}\,(x \geq 3))$ ✗
5. $\langle\langle \emptyset\rangle\rangle \mathrm{X}\,(true \mathrm{U}\,(x = 1))$ ✓
6. $[\![1, 2]\!] \mathrm{X}\,(x = 1 \wedge \langle\langle 2, 3\rangle\rangle \mathrm{G}\mathrm{X}\,(x \geq 3))$ ✓
7. $\langle\langle 2, 4\rangle\rangle \mathrm{X} [\![2, 3]\!] (true \mathrm{U}\,(x = 1))$ ✗

Strategies

How to interpret strategic operators?

Definition 4.3 (Strategy)

A **strategy** is a conditional plan.
We represent strategies by functions $s_a : St^+ \to Act$ such that $s_a(q_1, \ldots, q_n) \in d(a, q_n)$.

$\leadsto$ memory-based (perfect recall) strategies

Particular case: memoryless (positional) strategies $s_a : St \to Act$ such that $s_a(q) \in d(a, q)$.
Or, equivalently, for all histories $h, h' \in St^+$, $last(h) = last(h')$ implies $s_a(h) = s_a(h')$.

A joint strategy is a tuple of individual strategies, one for each agent.

- Strategies can freely assign arbitrary choices to histories (resp. states).
- CGS include no semantic means to represent the agents' uncertainty.
  $\leadsto$ CGS can only be used to model agents that have perfect information about the current state of the system.

Definition 4.4 (Outcome of a strategy)

$out(q, s_A)$ is the set of paths that result from coalition A executing joint strategy s_A from state q onward.

For a memoryless strategy the set $out(q, s_A)$ is given as follows:

$$\lambda = q_0, q_1, q_2 \ldots \in out(q, s_A) \quad \text{iff}$$

1. $q_0 = q$
2. for every $i \geq 0$ there exists a joint action $\langle \alpha_1^i, \ldots, \alpha_k^i \rangle$ such that
    1. $\alpha_a^i \in d(a, q_i)$ for every $a \in \mathbb{A}\mathrm{gt}$ (all α_a^i are enabled in each q_i)
    2. $\alpha_a^i = s_A[a](q_i)$ for every $a \in A$
    3. $q_{i+1} = o(q_i, \alpha_1^i, \ldots, \alpha_k^i)$.

For a memory-based strategy the outcome set is given as:

2.2 $\alpha_a^i = s_A[a](q_0, \ldots, q_i)$ for every $a \in A$

Example: Robots and Carriage

Strategies for Robot_1

- memoryless:
  - Robot_1 executes wait no matter what: $s_1(q_0) = s_1(q_1) = s_1(q_2) = wait$
  - Robot_1 waits unless the carriage is in position 2; in that case, he pushes: $s'_1(q_0) = s'_1(q_1) = wait$, $s'_1(q_2) = push$
- memory-based
  - $s''_1(h) = push$ if the length of h is even and it ends with q_0, otherwise $wait$.

Outcomes:

$$\begin{aligned} out(q_0, s_1) \quad = \quad & \{\lambda \in \{q_0, q_1, q_2\}^\omega \mid \lambda[0] = q_0 \text{ and for all } i \geq 0, \\ & \text{if } \lambda[i] = q_j \text{ then } \lambda[i+1] = q_j \text{ or } \lambda[i+1] = q_{(j-1) \bmod 3}\} \end{aligned}$$

$$\begin{aligned} out(q_0, s''_1) \quad = \quad & \{\lambda \in \{q_0, q_1, q_2\}^\omega \mid \lambda[0] = q_0 \text{ and for all } i \geq 0, \\ & \text{if } |\lambda[0..i]| = 2m \text{ and } \lambda[i] = q_0 \text{ then } \lambda[i+1] = q_0 \text{ or } \lambda[i+1] = q_1, \\ & \text{else if } \lambda[i] = q_j \text{ then } \lambda[i+1] = q_j \text{ or } \lambda[i+1] = q_{(j-1) \bmod 3}\} \end{aligned}$$

Semantics of ATL*

Formulas in ATL* are interpreted on Concurrent Game Structures.

Definition 4.5 (Semantics of ATL*: state formulas)

$(M, q) \models \mathsf{p}$ iff q is in $\mathcal{V}(\mathsf{p})$

$(M, q) \models \neg\Phi$ iff $(M, q) \not\models \Phi$

$(M, q) \models \Phi_1 \wedge \Phi_2$ iff $(M, q) \models \Phi_1$ and $(M, q) \models \Phi_2$

$(M, q) \models \langle\langle A \rangle\rangle \gamma$ iff there is a joint strategy s_A such that, for every path $\lambda \in out(q, s_A)$, $(M, \lambda) \models \gamma$

Definition 4.6 (Semantics of ATL*: path formulas)

.

Semantics of ATL*

Formulas in ATL* are interpreted on Concurrent Game Structures.

Definition 4.5 (Semantics of ATL*: state formulas)

$(M, q) \models \mathsf{p}$	iff q is in $\mathcal{V}(\mathsf{p})$
$(M, q) \models \neg\Phi$	iff $(M, q) \not\models \Phi$
$(M, q) \models \Phi_1 \wedge \Phi_2$	iff $(M, q) \models \Phi_1$ and $(M, q) \models \Phi_2$
$(M, q) \models \langle\langle A \rangle\rangle\gamma$	iff there is a joint strategy s_A such that, for every path $\lambda \in out(q, s_A)$, $(M, \lambda) \models \gamma$

Definition 4.6 (Semantics of ATL*: path formulas)

.

Semantics of ATL*

Formulas in ATL* are interpreted on Concurrent Game Structures.

Definition 4.5 (Semantics of ATL*: state formulas)

$(M, q) \models \mathsf{p}$	iff q is in $\mathcal{V}(\mathsf{p})$
$(M, q) \models \neg\Phi$	iff $(M, q) \not\models \Phi$
$(M, q) \models \Phi_1 \wedge \Phi_2$	iff $(M, q) \models \Phi_1$ and $(M, q) \models \Phi_2$
$(M, q) \models \langle\langle A \rangle\rangle \gamma$	iff there is a joint strategy s_A such that, for every path $\lambda \in out(q, s_A)$, $(M, \lambda) \models \gamma$

Definition 4.6 (Semantics of ATL*: path formulas)

The same as LTL!

.

Semantics of ATL*

Formulas in ATL* are interpreted on Concurrent Game Structures.

Definition 4.5 (Semantics of ATL*: state formulas)

$(M, q) \models \mathsf{p}$	iff q is in $\mathcal{V}(\mathsf{p})$
$(M, q) \models \neg\Phi$	iff $(M, q) \not\models \Phi$
$(M, q) \models \Phi_1 \land \Phi_2$	iff $(M, q) \models \Phi_1$ and $(M, q) \models \Phi_2$
$(M, q) \models \langle\langle A \rangle\rangle \gamma$	iff there is a joint strategy s_A such that, for every path $\lambda \in out(q, s_A)$, $(M, \lambda) \models \gamma$

Definition 4.6 (Semantics of ATL*: path formulas)

$(M, \lambda) \models \Phi$	iff $(M, \lambda[0]) \models \Phi$, for a state formula Φ
$(M, \lambda) \models \neg\gamma$	iff $(M, \lambda) \not\models \gamma$
$(M, \lambda) \models \gamma_1 \land \gamma_2$	iff $(M, \lambda) \models \gamma_1$ and $(M, \lambda) \models \gamma_2$
$(M, \lambda) \models \mathrm{X}\gamma$	iff $(M, \lambda[1..\infty]) \models \gamma$
$(M, \lambda) \models \gamma_1 \mathrm{U} \gamma_2$	iff $(M, \lambda[i..\infty]) \models \gamma_2$ for some $i \geq 0$, and $(M, \lambda[j..\infty]) \models \gamma_1$ for all $0 \leq j \leq i$.

Semantics of ATL*: Derived Operators

Recall: $[\![A]\!]\Phi \equiv \neg\langle\!\langle A\rangle\!\rangle\neg\Phi$.

Semantics of ATL*: state formulas

$(M, q) \models [\![A]\!]\Phi$ iff for every joint strategy s_A there exists some path $\lambda \in out(q, s_A)$ such that $(M, \lambda) \models \Phi$

As usual, $\mathrm{F}\gamma \equiv true\mathrm{U}\gamma$ and $\mathrm{G}\gamma \equiv \neg\mathrm{F}\neg\gamma$.

Semantics of ATL*: path formulas

$(M, \lambda) \models \mathrm{F}\gamma$ iff $M, \lambda[i..\infty] \models \gamma$ for some $i \geq 0$;

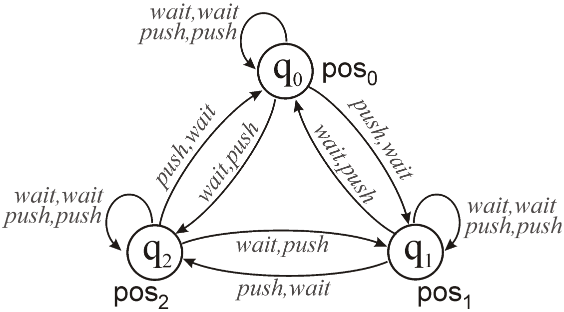
$(M, \lambda) \models \mathrm{G}\gamma$ iff $M, \lambda[i..\infty] \models \gamma$ for all $i \geq 0$;

State-Based Semantics for ATL

The semantics of “vanilla” ATL can be given entirely in terms of states:

$(M, q) \models$ p	iff p is in $\mathcal{V}(q)$
$(M, q) \models \neg\varphi$	iff $(M, q) \not\models \varphi$
$(M, q) \models \varphi_1 \wedge \varphi_2$	iff $(M, q) \models \varphi_1$ and $(M, q) \models \varphi_2$
$(M, q) \models \langle\langle A\rangle\rangle \mathrm{X}\varphi$	iff there is a joint strategy s_A such that, for every path $\lambda \in out(q, s_A)$, $(M, \lambda[1]) \models \varphi$
$(M, q) \models [\![A]\!]\mathrm{X}\varphi$	iff for every joint strategy s_A, there is some path $\lambda \in out(q, s_A)$ such that $(M, \lambda[1]) \models \varphi$
$(M, q) \models \langle\langle A\rangle\rangle \varphi_1 \mathrm{U} \varphi_2$	iff there is s_A such that, for every $\lambda \in out(q, s_A)$, $(M, \lambda[i]) \models \varphi_2$ for some $i \geq 0$ and $(M, \lambda[j]) \models \varphi_1$ for all $0 \leq j \leq i$
$(M, q) \models [\![A]\!]\varphi_1 \mathrm{U} \varphi_2$	iff for every s_A, there exists some $\lambda \in out(q, s_A)$ such that $(M, \lambda[i]) \models \varphi_2$ for some $i \geq 0$ and $(M, \lambda[j]) \models \varphi_1$ for all $0 \leq j \leq i$

Example: Robots and Carriage



$(M, q_0) \models \langle\langle 1 \rangle\rangle \mathrm{G} \neg \mathsf{pos}_1$?

- no agent can enforce the carriage to move to any particular position:

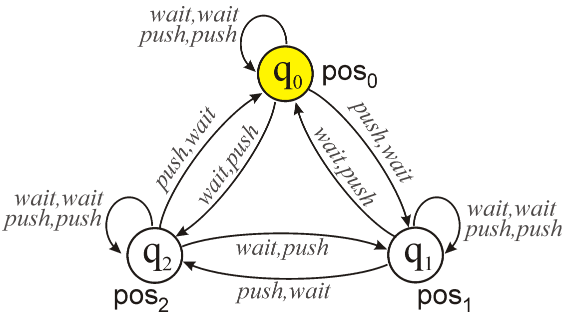
$$(M, q_0) \quad \models \quad \neg\langle\langle 1 \rangle\rangle \mathrm{X}\, \mathsf{pos}_0 \wedge \neg\langle\langle 2 \rangle\rangle \mathrm{X}\, \mathsf{pos}_0$$

- the robots together can move the carriage to any position they like:

$$(M, q_i) \quad \models \quad \langle\langle 1,2 \rangle\rangle \mathrm{F}\, \mathsf{pos}_0 \wedge \langle\langle 1,2 \rangle\rangle \mathrm{F}\, \mathsf{pos}_1 \wedge \langle\langle 1,2 \rangle\rangle \mathrm{F}\, \mathsf{pos}_2$$

$$(M, q_i) \quad \models \quad \langle\langle 1,2 \rangle\rangle (\mathrm{F}\, \mathsf{pos}_0 \wedge \mathrm{F}\, \mathsf{pos}_1 \wedge \mathrm{F}\, \mathsf{pos}_2)$$

Example: Robots and Carriage



$(M, q_0) \models \langle\!\langle 1 \rangle\!\rangle \mathrm{G} \neg \mathsf{pos}_1$?

- no agent can enforce the carriage to move to any particular position:

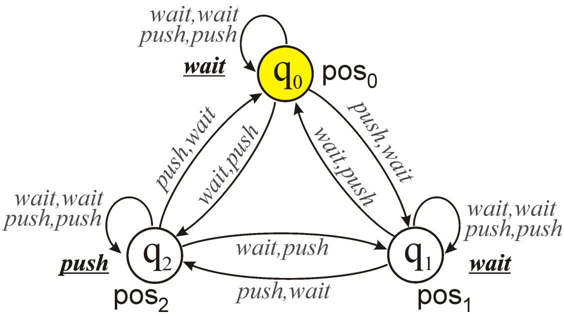
$$(M, q_0) \quad \models \quad \neg\langle\!\langle 1 \rangle\!\rangle \mathrm{X}\,\mathsf{pos}_0 \wedge \neg\langle\!\langle 2 \rangle\!\rangle \mathrm{X}\,\mathsf{pos}_0$$

- the robots together can move the carriage to any position they like:

$$(M, q_i) \quad \models \quad \langle\!\langle 1,2 \rangle\!\rangle \mathrm{F}\,\mathsf{pos}_0 \wedge \langle\!\langle 1,2 \rangle\!\rangle \mathrm{F}\,\mathsf{pos}_1 \wedge \langle\!\langle 1,2 \rangle\!\rangle \mathrm{F}\,\mathsf{pos}_2$$

$$(M, q_i) \quad \models \quad \langle\!\langle 1,2 \rangle\!\rangle (\mathrm{F}\,\mathsf{pos}_0 \wedge \mathrm{F}\,\mathsf{pos}_1 \wedge \mathrm{F}\,\mathsf{pos}_2)$$

Example: Robots and Carriage



$(M, q_0) \models \langle\langle 1 \rangle\rangle \mathrm{G} \neg \mathsf{pos}_1$?

- no agent can enforce the carriage to move to any particular position:

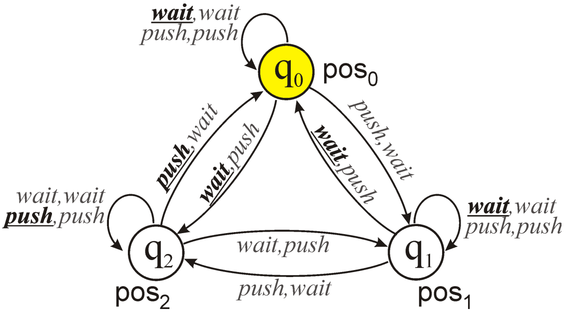
$$(M, q_0) \quad \models \quad \neg\langle\langle 1 \rangle\rangle \mathrm{X}\, \mathsf{pos}_0 \wedge \neg\langle\langle 2 \rangle\rangle \mathrm{X}\, \mathsf{pos}_0$$

- the robots together can move the carriage to any position they like:

$$(M, q_i) \quad \models \quad \langle\langle 1, 2 \rangle\rangle \mathrm{F}\, \mathsf{pos}_0 \wedge \langle\langle 1, 2 \rangle\rangle \mathrm{F}\, \mathsf{pos}_1 \wedge \langle\langle 1, 2 \rangle\rangle \mathrm{F}\, \mathsf{pos}_2$$

$$(M, q_i) \quad \models \quad \langle\langle 1, 2 \rangle\rangle (\mathrm{F}\, \mathsf{pos}_0 \wedge \mathrm{F}\, \mathsf{pos}_1 \wedge \mathrm{F}\, \mathsf{pos}_2)$$

Example: Robots and Carriage



$(M, q_0) \models \langle\langle 1 \rangle\rangle \mathrm{G} \neg \mathsf{pos}_1 ?$

- no agent can enforce the carriage to move to any particular position:

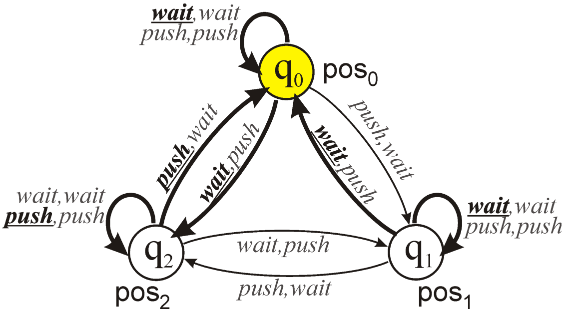
$$(M, q_0) \quad \models \quad \neg\langle\langle 1 \rangle\rangle \mathrm{X}\, \mathsf{pos}_0 \wedge \neg\langle\langle 2 \rangle\rangle \mathrm{X}\, \mathsf{pos}_0$$

- the robots together can move the carriage to any position they like:

$$(M, q_i) \quad \models \quad \langle\langle 1, 2 \rangle\rangle \mathrm{F}\, \mathsf{pos}_0 \wedge \langle\langle 1, 2 \rangle\rangle \mathrm{F}\, \mathsf{pos}_1 \wedge \langle\langle 1, 2 \rangle\rangle \mathrm{F}\, \mathsf{pos}_2$$

$$(M, q_i) \quad \models \quad \langle\langle 1, 2 \rangle\rangle (\mathrm{F}\, \mathsf{pos}_0 \wedge \mathrm{F}\, \mathsf{pos}_1 \wedge \mathrm{F}\, \mathsf{pos}_2)$$

Example: Robots and Carriage



$(M, q_0) \models \langle\langle 1 \rangle\rangle \mathrm{G} \neg \mathsf{pos}_1$?

- no agent can enforce the carriage to move to any particular position:

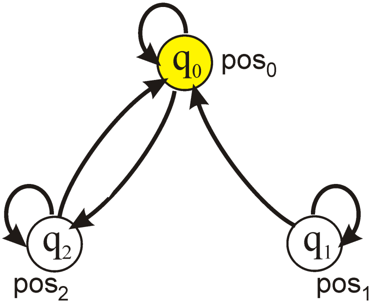
$$(M, q_0) \quad \models \quad \neg\langle\langle 1 \rangle\rangle \mathrm{X}\, \mathsf{pos}_0 \wedge \neg\langle\langle 2 \rangle\rangle \mathrm{X}\, \mathsf{pos}_0$$

- the robots together can move the carriage to any position they like:

$$(M, q_i) \quad \models \quad \langle\langle 1,2 \rangle\rangle \mathrm{F}\, \mathsf{pos}_0 \wedge \langle\langle 1,2 \rangle\rangle \mathrm{F}\, \mathsf{pos}_1 \wedge \langle\langle 1,2 \rangle\rangle \mathrm{F}\, \mathsf{pos}_2$$

$$(M, q_i) \quad \models \quad \langle\langle 1,2 \rangle\rangle (\mathrm{F}\, \mathsf{pos}_0 \wedge \mathrm{F}\, \mathsf{pos}_1 \wedge \mathrm{F}\, \mathsf{pos}_2)$$

Example: Robots and Carriage



$(M, q_0) \models \langle\langle 1 \rangle\rangle \mathrm{G} \neg \mathsf{pos}_1$?

- no agent can enforce the carriage to move to any particular position:

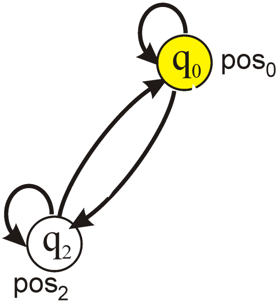
$$(M, q_0) \quad \models \quad \neg\langle\langle 1 \rangle\rangle \mathrm{X}\, \mathsf{pos}_0 \land \neg\langle\langle 2 \rangle\rangle \mathrm{X}\, \mathsf{pos}_0$$

- the robots together can move the carriage to any position they like:

$$(M, q_i) \quad \models \quad \langle\langle 1, 2 \rangle\rangle \mathrm{F}\, \mathsf{pos}_0 \land \langle\langle 1, 2 \rangle\rangle \mathrm{F}\, \mathsf{pos}_1 \land \langle\langle 1, 2 \rangle\rangle \mathrm{F}\, \mathsf{pos}_2$$

$$(M, q_i) \quad \models \quad \langle\langle 1, 2 \rangle\rangle (\mathrm{F}\, \mathsf{pos}_0 \land \mathrm{F}\, \mathsf{pos}_1 \land \mathrm{F}\, \mathsf{pos}_2)$$

Example: Robots and Carriage



$(M, q_0) \models \langle\langle 1 \rangle\rangle \mathrm{G} \neg \mathsf{pos}_1$ ✓

- no agent can enforce the carriage to move to any particular position:

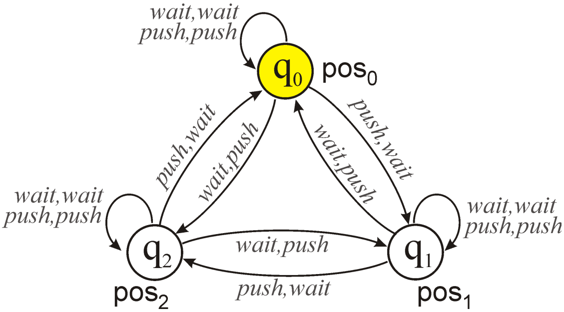
$$(M, q_0) \quad \models \quad \neg\langle\langle 1 \rangle\rangle \mathrm{X}\, \mathsf{pos}_0 \wedge \neg\langle\langle 2 \rangle\rangle \mathrm{X}\, \mathsf{pos}_0$$

- the robots together can move the carriage to any position they like:

$$(M, q_i) \quad \models \quad \langle\langle 1,2 \rangle\rangle \mathrm{F}\, \mathsf{pos}_0 \wedge \langle\langle 1,2 \rangle\rangle \mathrm{F}\, \mathsf{pos}_1 \wedge \langle\langle 1,2 \rangle\rangle \mathrm{F}\, \mathsf{pos}_2$$

$$(M, q_i) \quad \models \quad \langle\langle 1,2 \rangle\rangle (\mathrm{F}\, \mathsf{pos}_0 \wedge \mathrm{F}\, \mathsf{pos}_1 \wedge \mathrm{F}\, \mathsf{pos}_2)$$

Example: Robots and Carriage



$(M, q_0) \models \langle\langle 1 \rangle\rangle \mathrm{G} \neg \mathsf{pos}_1$

- no agent can enforce the carriage to move to any particular position:

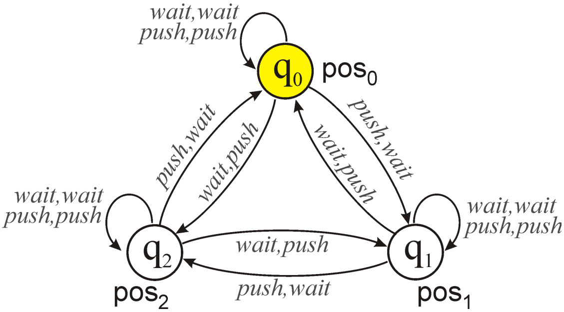
$$(M, q_0) \quad \models \quad \neg\langle\langle 1 \rangle\rangle \mathrm{X}\, \mathsf{pos}_0 \wedge \neg\langle\langle 2 \rangle\rangle \mathrm{X}\, \mathsf{pos}_0$$

- the robots together can move the carriage to any position they like:

$$(M, q_i) \quad \models \quad \langle\langle 1,2 \rangle\rangle \mathrm{F}\, \mathsf{pos}_0 \wedge \langle\langle 1,2 \rangle\rangle \mathrm{F}\, \mathsf{pos}_1 \wedge \langle\langle 1,2 \rangle\rangle \mathrm{F}\, \mathsf{pos}_2$$

$$(M, q_i) \quad \models \quad \langle\langle 1,2 \rangle\rangle (\mathrm{F}\, \mathsf{pos}_0 \wedge \mathrm{F}\, \mathsf{pos}_1 \wedge \mathrm{F}\, \mathsf{pos}_2)$$

Example: Robots and Carriage



$(M, q_0) \models \langle\langle 1 \rangle\rangle \mathrm{G} \neg \mathsf{pos}_1$

- no agent can enforce the carriage to move to any particular position:

$$(M, q_0) \quad \models \quad \neg\langle\langle 1 \rangle\rangle \mathrm{X}\, \mathsf{pos}_0 \wedge \neg\langle\langle 2 \rangle\rangle \mathrm{X}\, \mathsf{pos}_0$$

- the robots together can move the carriage to any position they like:

$$(M, q_i) \quad \models \quad \langle\langle 1, 2 \rangle\rangle \mathrm{F}\, \mathsf{pos}_0 \wedge \langle\langle 1, 2 \rangle\rangle \mathrm{F}\, \mathsf{pos}_1 \wedge \langle\langle 1, 2 \rangle\rangle \mathrm{F}\, \mathsf{pos}_2$$

$$(M, q_i) \quad \models \quad \langle\langle 1, 2 \rangle\rangle (\mathrm{F}\, \mathsf{pos}_0 \wedge \mathrm{F}\, \mathsf{pos}_1 \wedge \mathrm{F}\, \mathsf{pos}_2)$$

Part 4: Reasoning about Strategic Abilities

4.3 Agents, Systems, Games

Connection to Temporal Analysis of Systems

Strategy operators allow a number of useful concepts to be formally specified:

- Safety properties: $\langle\langle os \rangle\rangle \mathrm{G} \neg \mathsf{crash}$
- Liveness properties: $\langle\langle alice, bob \rangle\rangle \mathrm{F}\, \mathsf{paperAccepted}$
- Fairness properties: $\langle\langle prod, dlr \rangle\rangle \mathrm{G}(\mathsf{carRequested} \rightarrow \mathrm{F}\, \mathsf{carDelivered})$
  (Note: this is an ATL* formula!)

Connection to Multi-Agent/Multi-Process Systems

- Validity $\rightleftharpoons$ General properties of systems
- Satisfiability $\rightleftharpoons$ System synthesis
- Model checking $\rightleftharpoons$ Verification

ATL is just another specification language in this context...

Connection to Games

- Concurrent game structure = generalized extensive game

- $\langle\langle A \rangle\rangle\gamma$: operator $\langle\langle A \rangle\rangle$ splits the agents into proponents and opponents
- formula γ defines the winning condition
  $\leadsto$ infinite 2-player, binary, zero-sum game

- Flexible and compact specification of winning conditions

- Solving a game $\approx$ checking if $(M, q) \models \langle\langle A \rangle\rangle\gamma$
- Model checking ATL corresponds to game solving in game theory!

Connection to Games

What about other problems?

- Validity $\rightleftharpoons$ General properties of games
- Satisfiability $\rightleftharpoons$ Game design

  e.g., building a model for $\langle\langle\emptyset\rangle\rangle\gamma_1 \wedge \langle\langle A\rangle\rangle\gamma_2 \rightleftharpoons$ designing a game in which γ_1 is guaranteed and A can achieve γ_2
- (Frame satisfiability $\rightleftharpoons$ Mechanism design)

Motivating Example: Rescue Robots

- The robots can rescue all the people in the building.

  $\bigwedge_{i \in \mathit{People}} \langle\langle \mathit{Robots} \rangle\rangle \mathrm{F}\, \mathsf{safe_i}$

  Alternative formalization:

  $\langle\langle \mathit{Robots} \rangle\rangle \mathrm{F} \left(\bigwedge_{i \in \mathit{People}} \mathsf{safe_i} \right)$

- If person i gets outside the building then she can stay away from trouble forever.

  $\mathrm{AG} (\mathsf{outside_i} \rightarrow \langle\langle i \rangle\rangle \mathrm{G}\, \mathsf{safe_i})$

- Person i may be rescued without any robot ever entering the building, but guaranteed rescue requires some robots to enter.

  $\mathsf{E}\left((\bigwedge_{j \in \mathit{Robots}} \mathsf{outside_j}) \mathrm{U}\, \mathsf{safe_i}\right)$
  $\wedge \quad \neg \langle\langle \mathit{Robots} \rangle\rangle \left((\bigwedge_{j \in \mathit{Robots}} \mathsf{outside_j}) \mathrm{U}\, \mathsf{safe_i}\right)$

Motivating Example: Rescue Robots

- The robots can rescue all the people in the building.

  $$\bigwedge_{i \in \mathit{People}} \langle\langle \mathit{Robots} \rangle\rangle \mathrm{F}\, \mathsf{safe_i}$$

  Alternative formalization:

  $$\langle\langle \mathit{Robots} \rangle\rangle \mathrm{F} \left(\bigwedge_{i \in \mathit{People}} \mathsf{safe_i} \right)$$

- If person i gets outside the building then she can stay away from trouble forever.

  $$\mathrm{AG}\,(\mathsf{outside_i} \rightarrow \langle\langle i \rangle\rangle \mathrm{G}\, \mathsf{safe_i})$$

- Person i may be rescued without any robot ever entering the building, but guaranteed rescue requires some robots to enter.

  $$\begin{aligned} & \mathsf{E}\left((\bigwedge_{j \in \mathit{Robots}} \mathsf{outside_j}) \mathrm{U}\, \mathsf{safe_i} \right) \\ \wedge \quad & \neg \langle\langle \mathit{Robots} \rangle\rangle \left((\bigwedge_{j \in \mathit{Robots}} \mathsf{outside_j}) \mathrm{U}\, \mathsf{safe_i} \right) \end{aligned}$$

Motivating Example: Rescue Robots

- The robots can rescue all the people in the building.

  $\bigwedge_{i \in \mathit{People}} \langle\langle \mathit{Robots} \rangle\rangle \mathrm{F}\, \mathsf{safe_i}$

  Alternative formalization:

  $\langle\langle \mathit{Robots} \rangle\rangle \mathrm{F} \left(\bigwedge_{i \in \mathit{People}} \mathsf{safe_i} \right)$

- If person i gets outside the building then she can stay away from trouble forever.

  $\mathrm{AG}\left(\mathsf{outside_i} \rightarrow \langle\langle i \rangle\rangle \mathrm{G}\, \mathsf{safe_i}\right)$

- Person i may be rescued without any robot ever entering the building, but guaranteed rescue requires some robots to enter.

  $$\begin{aligned} & \mathsf{E}\left(\left(\bigwedge_{j \in \mathit{Robots}} \mathsf{outside_j}\right) \mathrm{U}\, \mathsf{safe_i}\right) \\ \wedge \quad & \neg \langle\langle \mathit{Robots} \rangle\rangle \left(\left(\bigwedge_{j \in \mathit{Robots}} \mathsf{outside_j}\right) \mathrm{U}\, \mathsf{safe_i}\right) \end{aligned}$$

Motivating Example: Rescue Robots

- The robots can rescue all the people in the building.

  $\bigwedge_{i \in \mathit{People}} \langle\!\langle \mathit{Robots} \rangle\!\rangle \mathrm{F}\, \mathsf{safe_i}$

  Alternative formalization:

  $\langle\!\langle \mathit{Robots} \rangle\!\rangle \mathrm{F} \left(\bigwedge_{i \in \mathit{People}} \mathsf{safe_i} \right)$

- If person i gets outside the building then she can stay away from trouble forever.

  $\mathrm{AG} \left(\mathsf{outside_i} \rightarrow \langle\!\langle i \rangle\!\rangle \mathrm{G}\, \mathsf{safe_i} \right)$

- Person i may be rescued without any robot ever entering the building, but guaranteed rescue requires some robots to enter.

  $$\mathsf{E}\left(\left(\bigwedge_{j \in \mathit{Robots}} \mathsf{outside_j} \right) \mathrm{U}\, \mathsf{safe_i} \right)$$
  $$\wedge \quad \neg \langle\!\langle \mathit{Robots} \rangle\!\rangle \left(\left(\bigwedge_{j \in \mathit{Robots}} \mathsf{outside_j} \right) \mathrm{U}\, \mathsf{safe_i} \right)$$

Motivating Example: Rescue Robots

- The robots can rescue all the people in the building.

  $$\bigwedge_{i \in \mathit{People}} \langle\langle \mathit{Robots} \rangle\rangle \mathrm{F}\, \mathsf{safe_i}$$

  Alternative formalization:

  $$\langle\langle \mathit{Robots} \rangle\rangle \mathrm{F} \left(\bigwedge_{i \in \mathit{People}} \mathsf{safe_i} \right)$$

- If person i gets outside the building then she can stay away from trouble forever.

  $$\mathrm{AG} \left(\mathsf{outside_i} \rightarrow \langle\langle i \rangle\rangle \mathrm{G}\, \mathsf{safe_i} \right)$$

- Person i may be rescued without any robot ever entering the building, but guaranteed rescue requires some robots to enter.

  $$\mathsf{E}\left(\left(\bigwedge_{j \in \mathit{Robots}} \mathsf{outside_j} \right) \mathrm{U}\, \mathsf{safe_i} \right)$$
  $$\wedge \quad \neg \langle\langle \mathit{Robots} \rangle\rangle \left(\left(\bigwedge_{j \in \mathit{Robots}} \mathsf{outside_j} \right) \mathrm{U}\, \mathsf{safe_i} \right)$$

Motivating Example: Rescue Robots

- The robots can rescue all the people in the building.

  $$\bigwedge_{i \in \mathit{People}} \langle\langle \mathit{Robots} \rangle\rangle \mathrm{F}\, \mathsf{safe_i}$$

  Alternative formalization:

  $$\langle\langle \mathit{Robots} \rangle\rangle \mathrm{F} \left(\bigwedge_{i \in \mathit{People}} \mathsf{safe_i} \right)$$

- If person i gets outside the building then she can stay away from trouble forever.

  $$\langle\langle \emptyset \rangle\rangle \mathrm{G} \left(\mathsf{outside_i} \rightarrow \langle\langle i \rangle\rangle \mathrm{G}\, \mathsf{safe_i} \right)$$

- Person i may be rescued without any robot ever entering the building, but guaranteed rescue requires some robots to enter.

  $$\mathsf{E}\left(\left(\bigwedge_{j \in \mathit{Robots}} \mathsf{outside_j} \right) \mathrm{U}\, \mathsf{safe_i} \right)$$
  $$\wedge \quad \neg \langle\langle \mathit{Robots} \rangle\rangle \left(\left(\bigwedge_{j \in \mathit{Robots}} \mathsf{outside_j} \right) \mathrm{U}\, \mathsf{safe_i} \right)$$

Motivating Example: Rescue Robots

- The robots can rescue all the people in the building.

  $$\bigwedge_{i \in \mathit{People}} \langle\langle \mathit{Robots} \rangle\rangle \mathrm{F}\, \mathsf{safe_i}$$

  Alternative formalization:

  $$\langle\langle \mathit{Robots} \rangle\rangle \mathrm{F} \left(\bigwedge_{i \in \mathit{People}} \mathsf{safe_i} \right)$$

- If person i gets outside the building then she can stay away from trouble forever.

  $$\langle\langle \emptyset \rangle\rangle \mathrm{G} (\mathsf{outside_i} \rightarrow \langle\langle i \rangle\rangle \mathrm{G}\, \mathsf{safe_i})$$

- Person i may be rescued without any robot ever entering the building, but guaranteed rescue requires some robots to enter.

  $$\mathsf{E}\left((\bigwedge_{j \in \mathit{Robots}} \mathsf{outside_j}) \mathrm{U}\, \mathsf{safe_i} \right) \\ \wedge \quad \neg \langle\langle \mathit{Robots} \rangle\rangle \left((\bigwedge_{j \in \mathit{Robots}} \mathsf{outside_j}) \mathrm{U}\, \mathsf{safe_i} \right)$$

Motivating Example: Rescue Robots

- The robots can rescue all the people in the building.

  $\bigwedge_{i \in \mathit{People}} \langle\langle \mathit{Robots} \rangle\rangle \mathrm{F}\, \mathsf{safe_i}$

  Alternative formalization:

  $\langle\langle \mathit{Robots} \rangle\rangle \mathrm{F} \left(\bigwedge_{i \in \mathit{People}} \mathsf{safe_i} \right)$

- If person i gets outside the building then she can stay away from trouble forever.

  $\langle\langle \emptyset \rangle\rangle \mathrm{G} (\mathsf{outside_i} \rightarrow \langle\langle i \rangle\rangle \mathrm{G}\, \mathsf{safe_i})$

- Person i may be rescued without any robot ever entering the building, but guaranteed rescue requires some robots to enter.

  $\mathsf{E}\left((\bigwedge_{j \in \mathit{Robots}} \mathsf{outside_j}) \mathrm{U}\, \mathsf{safe_i} \right)$
  $\wedge \quad \neg \langle\langle \mathit{Robots} \rangle\rangle \left((\bigwedge_{j \in \mathit{Robots}} \mathsf{outside_j}) \mathrm{U}\, \mathsf{safe_i} \right)$

Motivating Example: Rescue Robots

- The robots can rescue all the people in the building.

  $\bigwedge_{i \in \textit{People}} \langle\langle \textit{Robots} \rangle\rangle \mathrm{F}\, \mathsf{safe_i}$

  Alternative formalization:

  $\langle\langle \textit{Robots} \rangle\rangle \mathrm{F} \left(\bigwedge_{i \in \textit{People}} \mathsf{safe_i} \right)$

- If person i gets outside the building then she can stay away from trouble forever.

  $\langle\langle \emptyset \rangle\rangle \mathrm{G} (\mathsf{outside_i} \rightarrow \langle\langle i \rangle\rangle \mathrm{G}\, \mathsf{safe_i})$

- Person i may be rescued without any robot ever entering the building, but guaranteed rescue requires some robots to enter.

  $\langle\langle \mathbb{A}\mathrm{gt} \rangle\rangle ((\bigwedge_{j \in \textit{Robots}} \mathsf{outside_j}) \mathrm{U}\, \mathsf{safe_i})$
  $\wedge \quad \neg \langle\langle \textit{Robots} \rangle\rangle ((\bigwedge_{j \in \textit{Robots}} \mathsf{outside_j}) \mathrm{U}\, \mathsf{safe_i})$

Motivating Example: Rescue Robots

- The robots can rescue all the people, and they know that they can

  Cannot be expressed in ATL* !

- The robots can rescue all the people, and they know how to do it

  Cannot be expressed in ATL* !

Motivating Example: Voting

- The system cannot reveal how a particular voter voted.

  $\neg\langle\langle system\rangle\rangle \mathrm{F}(\bigvee_{c\in \mathit{Candidates}} \mathsf{revealedVote}_{\mathsf{i,c}})$

- A voter i can gain no receipt which can be used to prove that she voted in a certain way.

  $\neg\langle\langle i\rangle\rangle \mathrm{F}(\bigvee_{c\in \mathit{Candidates}} \mathsf{receiptVote}_{\mathsf{i,c}})$

- A voter i cannot cooperate with the coercer to prove to him that she voted in a certain way.

  $\neg\langle\langle i, coercer\rangle\rangle \mathrm{F} \ldots ?$

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

Beware: even for the first two properties, we need the right modeling of epistemic capabilities in the scenario!

Concluding Remarks

- ATL*/ATL can be seen as a logic for reasoning about agents with perfect information.
- CGS do not allow for a representation of agents' uncertainty.
- It is implicitly assumed that each agent always precisely knows the current state of the system/game.
- The notions of perfect vs. imperfect information will be address in Lecture 6.

References

R. Alur, T. A. Henzinger, and O. Kupferman.
Alternating-time Temporal Logic.
Journal of the ACM, 49:672–713, 2002.

Nils Bulling, Valentin Goranko, and Wojciech Jamroga.
Logics for Reasoning About Strategic Abilities in Multi-Player Games.
In J. van Benthem, S. Ghosh, R. Verbrugge, editors, *Modeling Strategic Reasoning*. Springer, to appear.

Thomas Agotnes, Valentin Goranko, and Wojciech Jamroga.
Alternating-time temporal logic with irrevocable strategies.
In Dov Samet, editor, *Proceedings of the 11th Conference on Theoretical Aspects of Rationality and Knowledge (TARK XI)*, pages 15–24, Brussels, Belgium, June 2007. Presses Universitaires de Louvain/ACM DL.

Part 5: Verification of Strategic Ability

Part 5: Verification of Strategic Ability

5.1 Properties of Alternating-time Temporal Logic

Semantic Embedding of CTL* into ATL*

Temporal reasoning can be **semantically** embedded into strategic reasoning:

- think of a transition system as a concurrent game structure with a single agent ("the system" s)
- transitions are due to actions of agent s.
- $\mathsf{E}\gamma$ ("there is a path on which γ holds") can be then translated as $\langle\langle s \rangle\rangle\gamma$ ("the system can behave in a way that makes γ true").
  Or $[\![\emptyset]\!]\gamma$ equivalently.
- $\mathsf{A}\gamma$ ("for all paths, γ holds") can be translated as $[\![s]\!]\gamma$ ("γ is enforced whatever all the agents – i.e., the system – do").
  Or $\langle\langle \emptyset \rangle\rangle\gamma$ equivalently.

Syntactic Embedding of CTL* into ATL*

Moreover, ATL* extends the branching-time logic CTL* by the following **syntactic** translation:

- $\mathsf{E}\gamma \equiv \langle\!\langle \mathbb{A}\mathrm{gt} \rangle\!\rangle \gamma$ ("there is a path" = outcomes obtainable by grand coalition). Or $[\![\emptyset]\!]\gamma$ equivalently.
- $\mathsf{A}\gamma \equiv [\![\mathbb{A}\mathrm{gt}]\!]\gamma$ ("for all paths" = necessary outcomes). Or $\langle\!\langle\emptyset\rangle\!\rangle\gamma$ equivalently.

In particular,

$$\begin{aligned} \langle\!\langle \mathbb{A}\mathrm{gt} \rangle\!\rangle \gamma &\equiv [\![\emptyset]\!]\gamma \\ [\![\mathbb{A}\mathrm{gt}]\!]\gamma &\equiv \langle\!\langle\emptyset\rangle\!\rangle\gamma \end{aligned}$$

But in general,

$$\langle\!\langle A \rangle\!\rangle \gamma \quad \not\equiv \quad [\![\overline{A}]\!]\gamma$$

Determinacy in Turn-based Games

Definition 5.1 (Determinacy)

In each state, either the players in A can win with objective γ, or the players not in A can win with the complementary objective $\neg\gamma$.

Chess, checkers, etc, are determined.

Theorem 5.2

Turn-based games (of perfect information) are determined.

Corollary 5.3

The following formula is valid in turn-based CGS:

$$\begin{aligned} \langle\langle A \rangle\rangle \gamma & \equiv \neg \langle\langle \overline{A} \rangle\rangle \neg \gamma \\ & \equiv [\![\overline{A}]\!] \gamma \end{aligned}$$

In general, this scheme of formula is only valid for $A = \mathbb{A}\mathrm{gt}$ (see previous slide).

Determinacy in CGS

We have that $\langle\langle A\rangle\rangle\gamma \rightarrow [\![\overline{A}]\!]\gamma$
But the converse is not true in general.

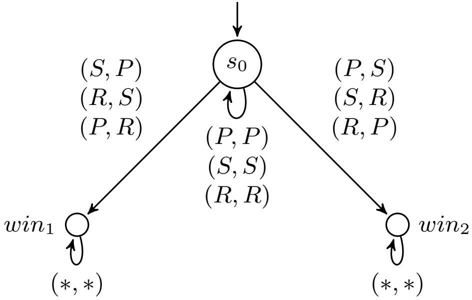


Figure: a CGS for rock-paper-scissors.

In s_0 we have that $[\![2]\!]\mathrm{X}\,\mathsf{win}_1$
But, $s_0 \not\models \langle\langle 1\rangle\rangle\mathrm{X}\,\mathsf{win}_1$

Memory and Strategic Abilities in "Vanilla" ATL

Let us discern between two definitions of the satisfaction relation:

$\models_R$: perfect recall is assumed, strategies are of type $f : St^+ \to Act$

$\models_r$: only memoryless strategies are allowed, i.e., $f : St \to Act$

Theorem 5.4

For any CGS M, state q and ATL formula φ, we have:

$$(M, q) \models_r \varphi \quad \Leftrightarrow \quad (M, q) \models_R \varphi.$$

Memory does not influence strategic abilities in "Vanilla" ATL!

Definition 5.5 (Tree-unfolding)

Given a CGS $M = \langle \mathbb{A}\mathrm{gt}, St, \mathcal{V}, Act, d, o \rangle$ and state $q \in St$, the tree unfolding $T(M, q) = \langle \mathbb{A}\mathrm{gt}, St^*, \mathcal{V}^*, Act, d^*, o^* \rangle$ of M from q is defined as:

- $St^* = hist_M(q)$
- $\mathcal{V}^*(h) = \mathcal{V}(last(h))$
- $d_i^*(h) = d_i(last(h))$
- $o^*(h, \alpha) = h \cdot o(last(h), \alpha)$.

- Perfect recall strategies in M correspond to memoryless strategies in $T(M, q)$: for every q and φ,
$$(T(M, q), q) \models_r \varphi \quad \text{iff} \quad (M, q) \models_R \varphi$$
- For every (M, q), we have $T(M, q) \rightleftarrows_\beta M$ for $\beta = \{(h, last(h)) \mid h \in hist_M(q)\}$.
- Finally, by invariance under bisimulation,
$$(M, q) \models_r \varphi \quad \text{iff } T(M, q), q) \models_r \varphi \quad \text{iff } (M, q) \models_R \varphi$$

For ATL* – contrary to “vanilla” ATL – memory matters:

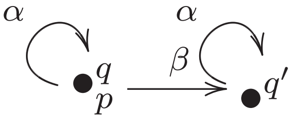
Theorem 5.6

There is a CGS M, a state q in M, and an ATL formula φ, such that*

$$(M, q) \models_r \varphi \quad \not\Leftrightarrow \quad (M, q) \models_R \varphi$$

Counterexample:

M:



$$\varphi = \langle\langle a \rangle\rangle (\mathrm{X} p \land \mathrm{X}\mathrm{X} \neg p)$$

Fixpoint Properties

Theorem 5.7

*The following formulas are **valid** in ATL:*

$$\begin{aligned}
\langle\langle A\rangle\rangle\varphi_1 \mathrm{U}\varphi_2 &\leftrightarrow \varphi_2 \vee (\varphi_1 \wedge \langle\langle A\rangle\rangle \mathrm{X} \langle\langle A\rangle\rangle\varphi_1 \mathrm{U}\varphi_2)\\
\langle\langle A\rangle\rangle \mathrm{F}\varphi &\leftrightarrow \varphi \vee \langle\langle A\rangle\rangle \mathrm{X} \langle\langle A\rangle\rangle \mathrm{F}\varphi\\
\langle\langle A\rangle\rangle \mathrm{G}\varphi &\leftrightarrow \varphi \wedge \langle\langle A\rangle\rangle \mathrm{X} \langle\langle A\rangle\rangle \mathrm{G}\varphi
\end{aligned}$$

Key remark for model checking:

Corollary

Strategies for A that achieve objectives specified in “vanilla” ATL can be **synthesized incrementally** (no backtracking is necessary).

Part 5: Verification of Strategic Ability

5.2 Model Checking ATL

Model Checking ATL

A well-known nice result: model checking ATL is tractable!

Theorem (Alur, Kupferman & Henzinger 1998)

ATL model checking is PTIME-complete, and can be done in linear time.

No worse than CTL! (or at least it seems so)

So... let's model check!

Model Checking ATL: lower bound

Theorem

ATL model checking is (at least) as hard as CTL model checking (which is PTIME-hard).

Check the semantic embedding in the previous part.

Model Checking ATL: upper bound

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

case $\varphi \equiv p$ **:** return $\mathcal{V}(p)$

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 $\varphi \equiv \langle\langle A\rangle\rangle \mathrm{X}\psi$ **:** return $pre(A, mcheck(\mathcal{M}, \psi))$

case $\varphi \equiv \langle\langle A\rangle\rangle \mathrm{G}\psi$ **:** return $mcheckG(A, \mathcal{M}, \psi)$

case $\varphi \equiv \langle\langle A\rangle\rangle \psi_1 \mathrm{U} \psi_2$ **:** return $mcheckU(A, \mathcal{M}, \psi_1, \psi_2)$
end case

$$pre(A, Q) = \{q \mid \text{ there exist } \alpha_A \text{ such that for all } \alpha_{\overline{A}}, o(q, \alpha_A, \alpha_{\overline{A}}) \in Q\}$$

Function $pre()$ returns the set of states Q' such that, when the system is in a state $q \in Q'$, agents in A can enforce the next state to be in Q.

Model Checking ATL: upper bound

To verify a formula of type $\langle\langle A \rangle\rangle\gamma$, the algorithm tries to construct a winning strategy for A, i.e., one that guarantees γ no matter what the other agents do.

function $mcheckG(A, \mathcal{M}, \psi)$.
  Returns the subset of St for which formula $\langle\langle A \rangle\rangle \mathrm{G} \psi$ holds.

$Q_1 := Q$;
$Q_2 := Q_3 := mcheck(\mathcal{M}, \psi)$;
while $Q_1 \not\subseteq Q_2$ **do**
  $Q_1 := Q_2$;
  $Q_2 := pre(A, Q_1) \cap Q_3$
od;
return Q_1

Model Checking ATL: upper bound

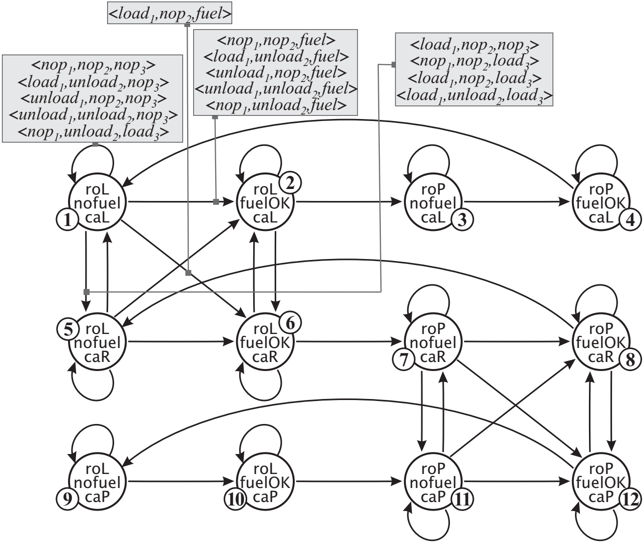
function $mcheckU(A, \mathcal{M}, \psi_1, \psi_2)$.
Returns the subset of St for which formula $\langle\langle A\rangle\rangle\psi_1 \mathrm{U} \psi_2$ holds.

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

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

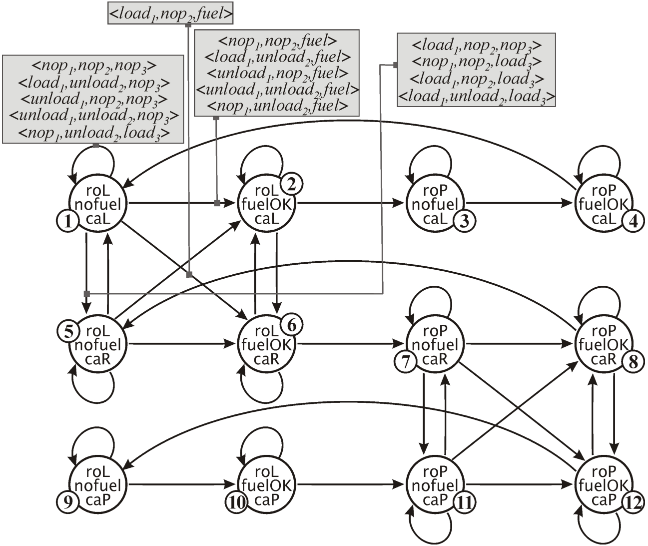


Example: Simple Rocket Domain

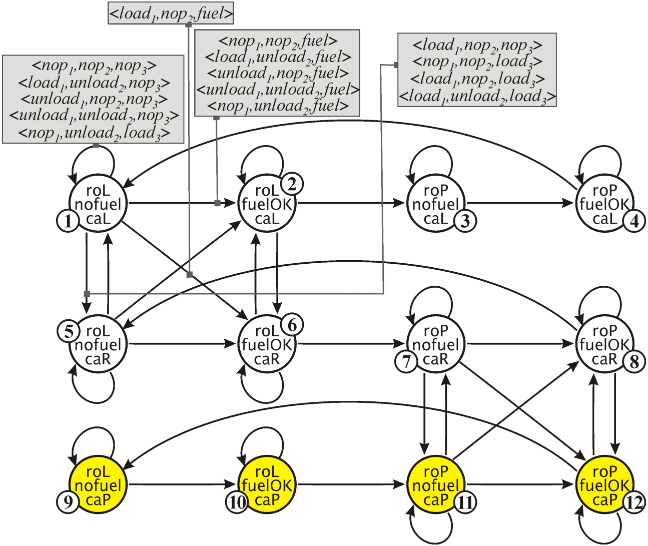
Verification examples

- We want to find the set of states from which agents 1 and 3 can move the cargo to any given location: $\langle\langle 1,3\rangle\rangle \mathrm{F}\,\mathsf{caP} \wedge \langle\langle 1,3\rangle\rangle \mathrm{F}\,\mathsf{caL}$
- How does that work for the coalition of agents 1 and 2: $\langle\langle 1,2\rangle\rangle \mathrm{F}\,\mathsf{caP}$?
- What about a maintenance goal, like agent 3 keeping the cargo in Paris forever: $\langle\langle 3\rangle\rangle \mathrm{G}\,\mathsf{caP}$?

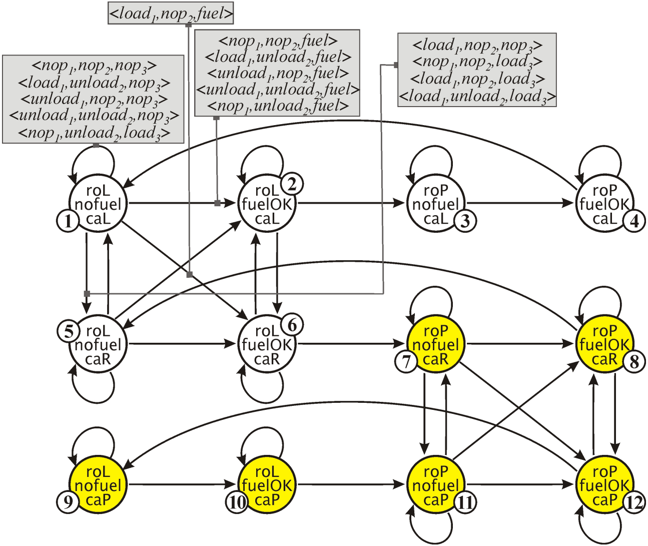
Simple Rocket Domain: Verification of $\langle\langle 1,3\rangle\rangle F\,\mathsf{caP}$



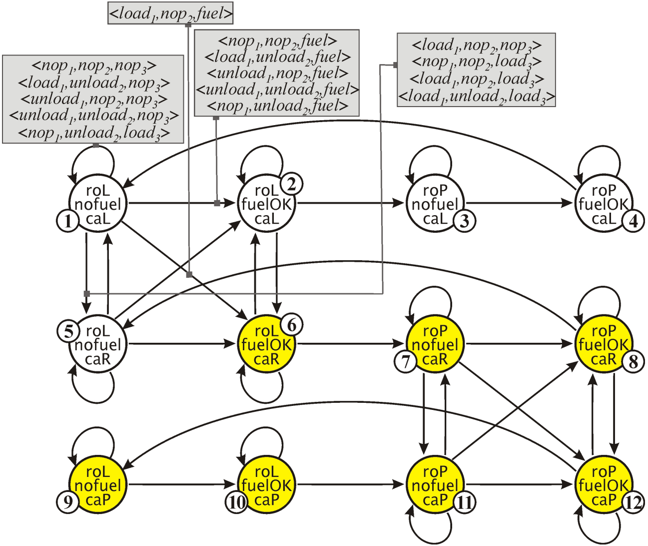
Simple Rocket Domain: Verification of $\langle\langle 1, 3 \rangle\rangle F\, caP$



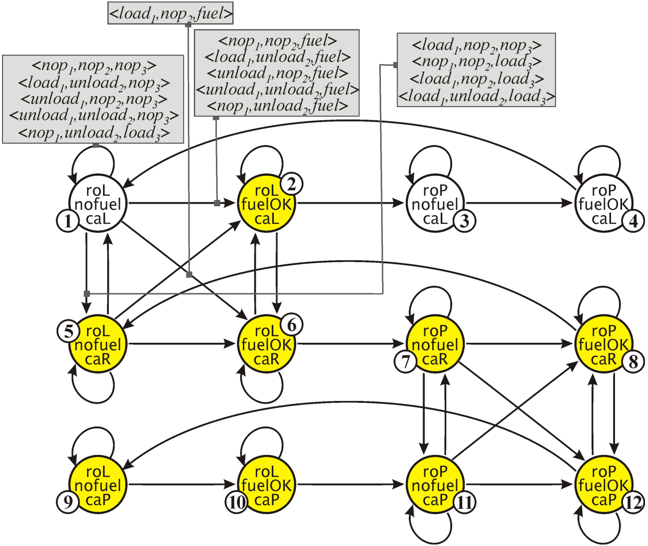
Simple Rocket Domain: Verification of $\langle\langle 1, 3\rangle\rangle F\, \mathrm{caP}$



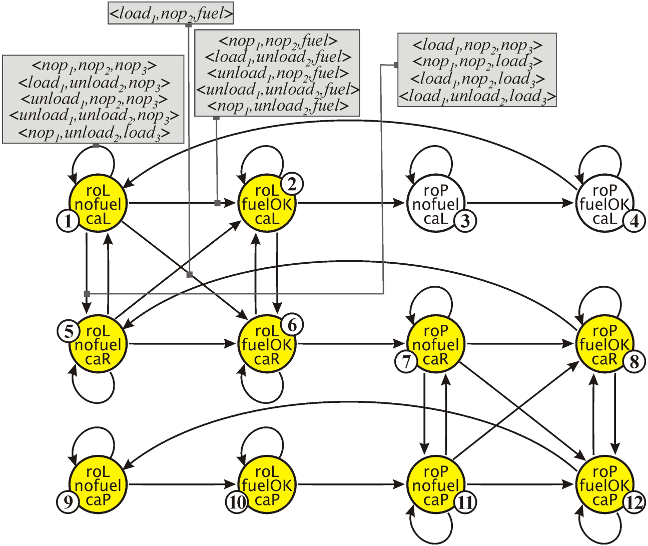
Simple Rocket Domain: Verification of $\langle\langle 1,3 \rangle\rangle F\, caP$



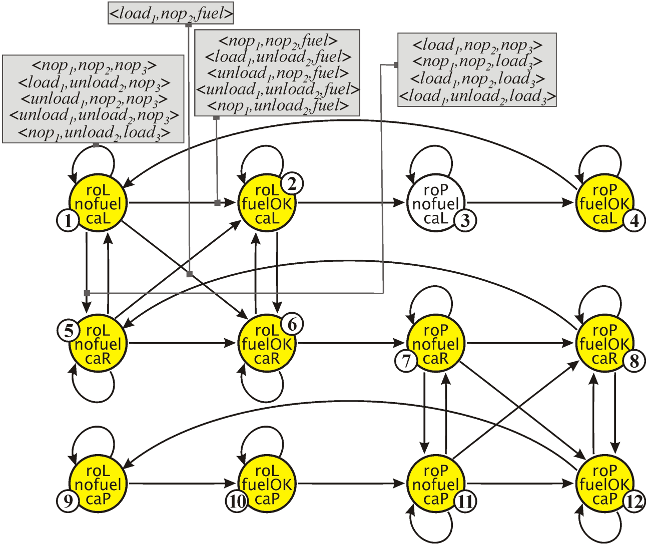
Simple Rocket Domain: Verification of $\langle\langle 1, 3 \rangle\rangle F\, caP$



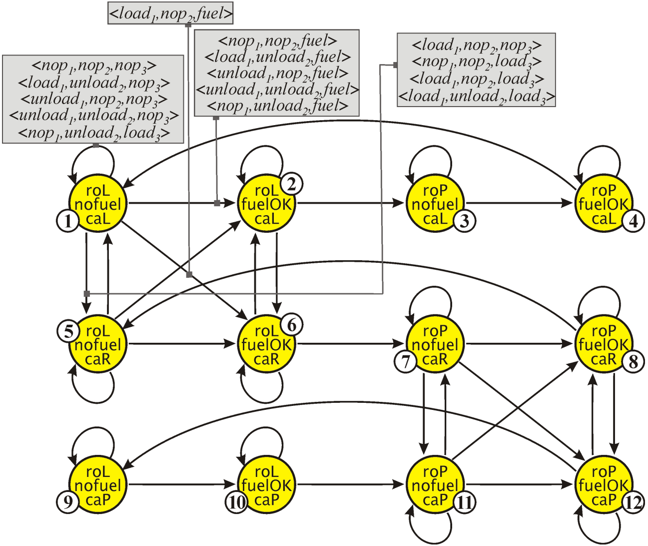
Simple Rocket Domain: Verification of $\langle\langle 1, 3 \rangle\rangle F\, \mathrm{caP}$



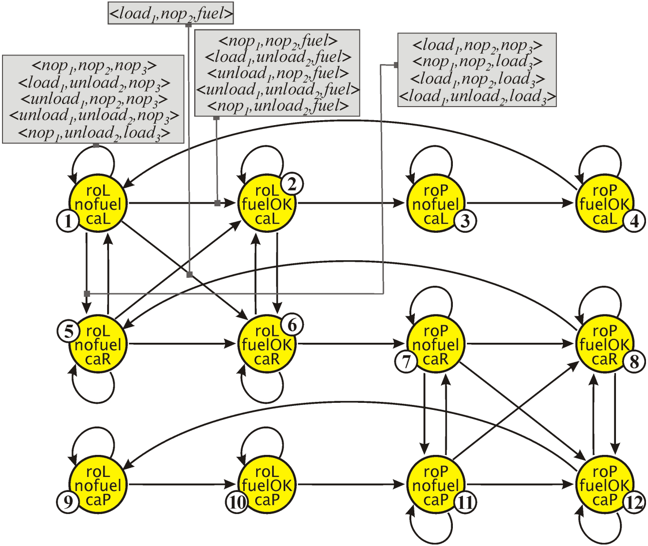
Simple Rocket Domain: Verification of $\langle\langle 1, 3 \rangle\rangle \mathrm{F}\, caP$



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

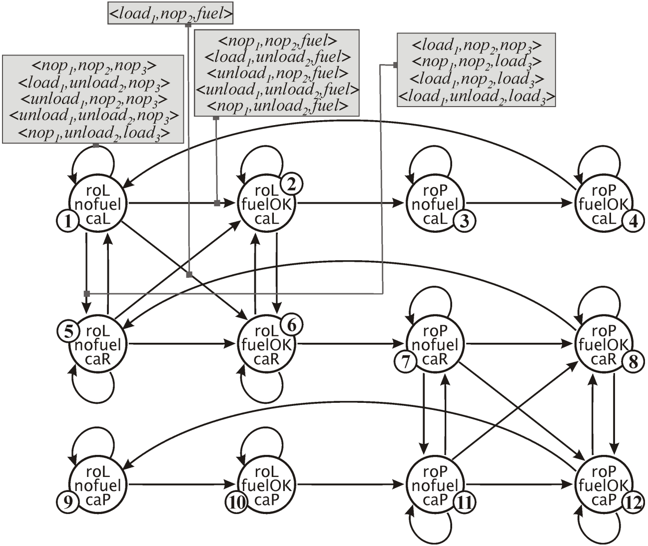


Simple Rocket Domain: Verification of $\langle\langle 1,3\rangle\rangle F\, caP$

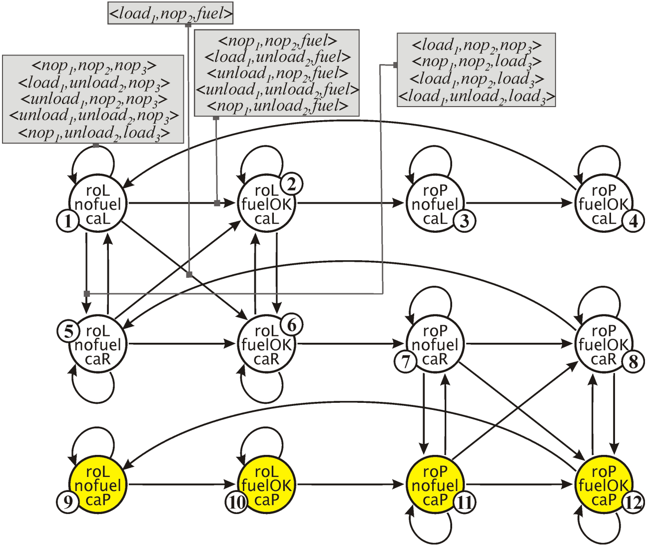


Done!

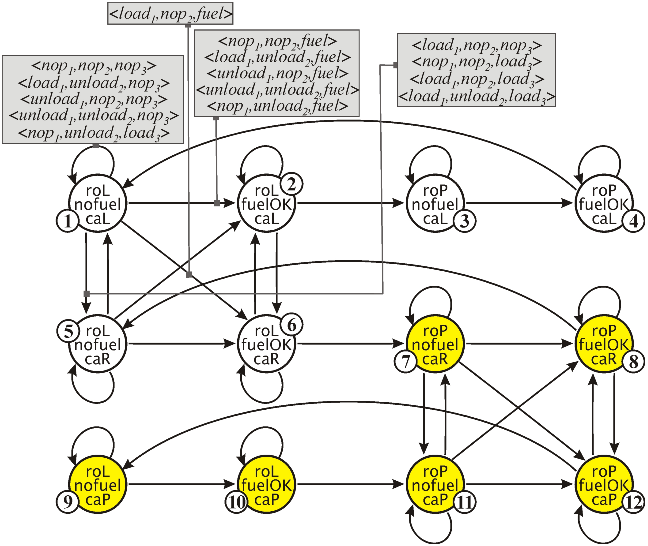
Simple Rocket Domain: Verification of $\langle\langle 1, 2\rangle\rangle F\, caP$



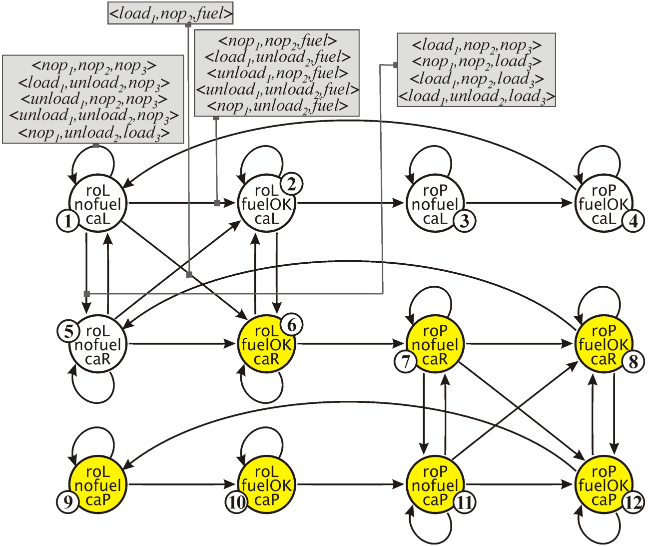
Simple Rocket Domain: Verification of $\langle\langle 1, 2\rangle\rangle F\, caP$



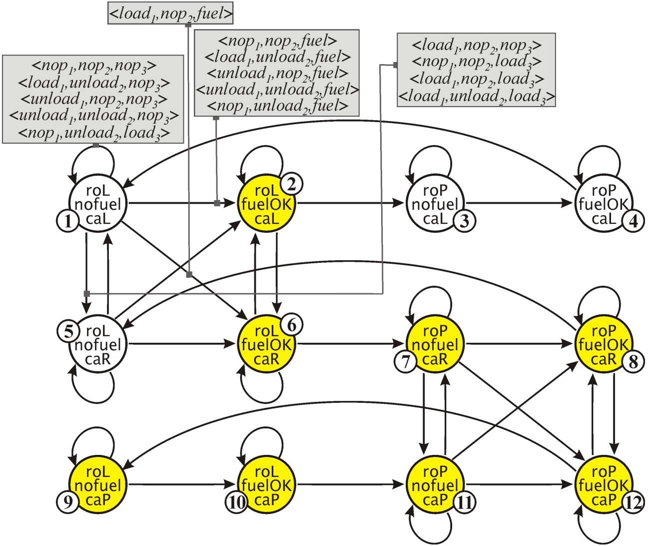
Simple Rocket Domain: Verification of $\langle\langle 1,2\rangle\rangle F\, caP$



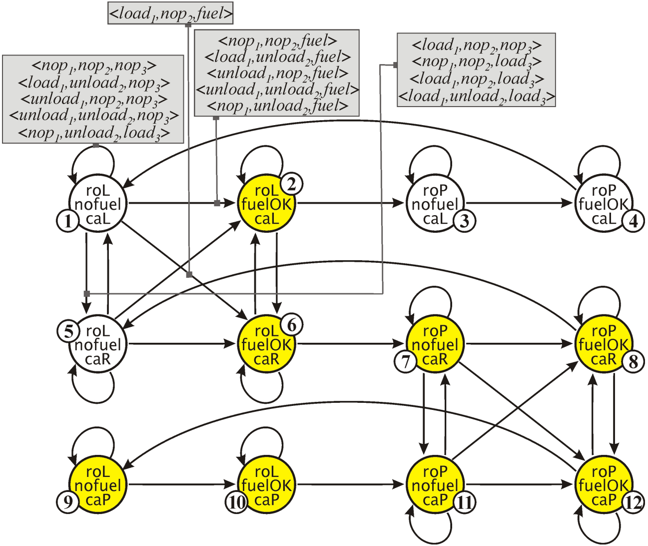
Simple Rocket Domain: Verification of $\langle\langle 1, 2\rangle\rangle \mathrm{F}\, \mathrm{caP}$



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

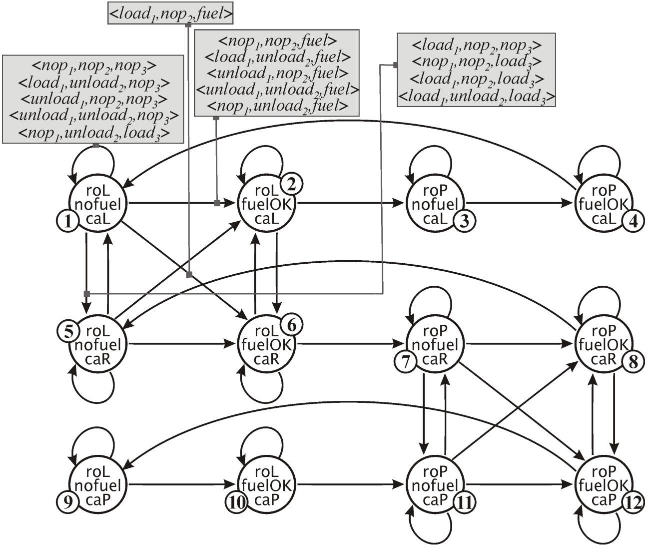


Simple Rocket Domain: Verification of $\langle\langle 1, 2\rangle\rangle F\, caP$

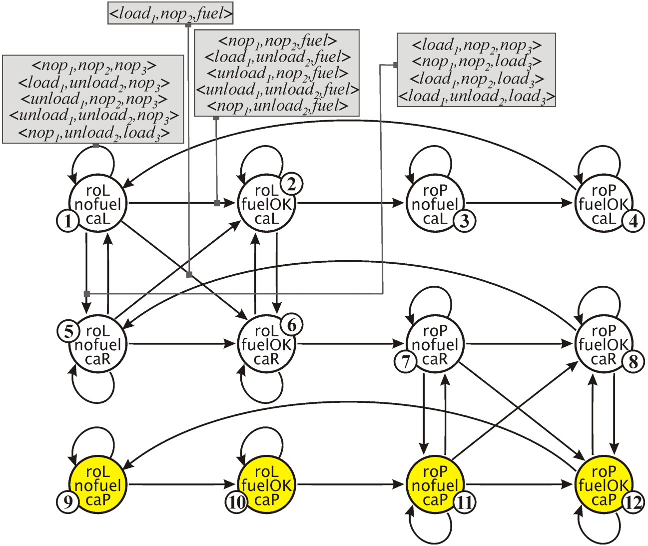


Done!

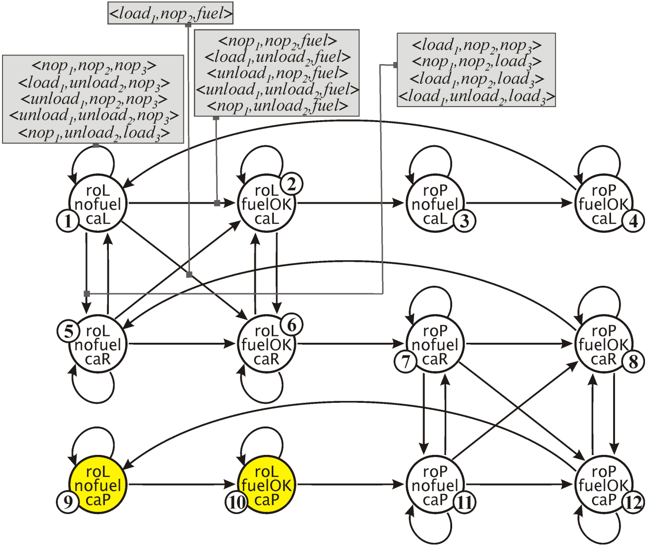
Simple Rocket Domain: Verification of $\langle\langle 3\rangle\rangle \mathrm{G}\, caP$



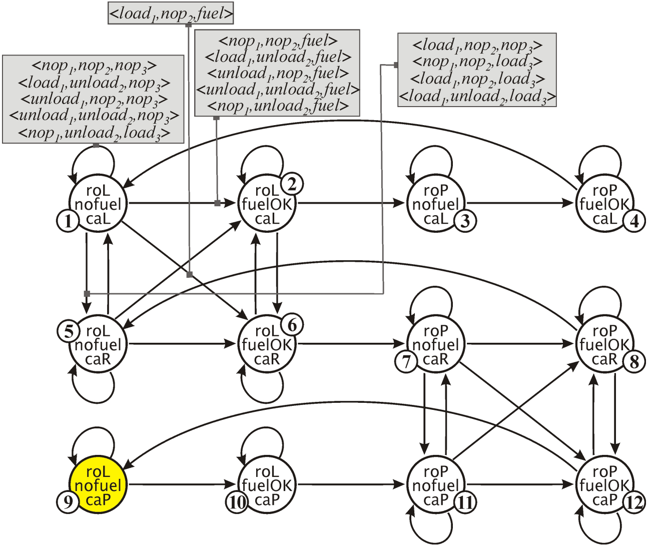
Simple Rocket Domain: Verification of $\langle\langle 3 \rangle\rangle G\, caP$



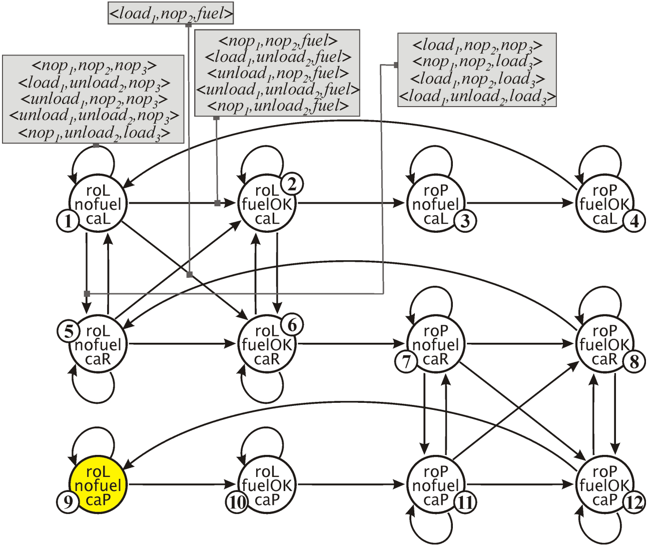
Simple Rocket Domain: Verification of $\langle\langle 3\rangle\rangle G\,\mathsf{caP}$



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



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



Done!

Model Checking ATL: Soundness

- Soundness of the MC algorithm follows from the fixpoint characterizations of strategic modalities.
- Procedures $mcheckG()$ and $mcheckU()$ define functors τ_1, τ_2 on sets:

$$\begin{array}{rcl}\tau_1(Z) = & mcheck(\psi) \cap pre(A, Z) & = \psi \wedge \langle\!\langle A \rangle\!\rangle \mathrm{X} Z \\ \tau_2(Z) = & mcheck(\psi_2) \cup (mcheck(\psi_1) \cap pre(A, Z)) & = \psi_2 \vee (\psi_1 \wedge \langle\!\langle A \rangle\!\rangle \mathrm{X} Z)\end{array}$$

- Loop compute *fixed points*: sets Z such that $\tau(Z) = Z$.
- Recall equivalences:

$$\begin{array}{rcl}\langle\!\langle A \rangle\!\rangle \mathrm{G} \psi & \leftrightarrow & \psi \wedge \langle\!\langle A \rangle\!\rangle \mathrm{X} \langle\!\langle A \rangle\!\rangle \mathrm{G} \psi \\ \langle\!\langle A \rangle\!\rangle \psi_1 \mathrm{U} \psi_2 & \leftrightarrow & \psi_2 \vee (\psi_1 \wedge \langle\!\langle A \rangle\!\rangle \mathrm{X} \langle\!\langle A \rangle\!\rangle \psi_1 \mathrm{U} \psi_2)\end{array}$$

  $\Rightarrow$ $\langle\!\langle A \rangle\!\rangle \mathrm{G} \psi$ and $\langle\!\langle A \rangle\!\rangle \psi_1 \mathrm{U} \psi_2$ are the fixed points computed by $mcheckG(A, \psi)$ and $mcheckU(A, \psi_1, \psi_2)$ respectively.
- It does not matter whether perfect recall or memoryless strategies are used: the algorithm is correct for the R-semantics, but it always finds an r-strategy.
- The algorithm can be adapted for the, seemingly more difficult, task of strategy synthesis for temporal goals.

Model Checking ATL: Complexity

Theorem (Alur, Kupferman & Henzinger 1998/2002)

Model checking ATL is PTIME-complete, and can be done in linear time.

ATL is strictly more expressive than CTL with no computational price to pay.

Model Checking ATL: Complexity

Theorem (Alur, Kupferman & Henzinger 1998/2002)

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

ATL is strictly more expressive than CTL with no computational price to pay.

Model Checking ATL: Complexity

Theorem (Alur, Kupferman & Henzinger 1998/2002)

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

ATL is strictly more expressive than CTL with no computational price to pay.

Part 5: Verification of Strategic Ability

5.3 Model Checking ATL*

Model Checking ATL*: Complexity

Theorem (Alur, Kupferman & Henzinger 1998)

- ATL$^*_R$ model checking is 2EXPTIME-complete in the number of the transitions in the model and the length of the formula.
- ATL$^*_r$ model checking is PSPACE-complete in the number of the transitions in the model and the length of the formula.

Model Checking ATL$^*_R$: lower bound

Theorem 5.8

ATL$^*_R$ model checking is (at least) as hard as LTL realizability (which is 2EXPTIME-hard).

Definition 5.9 (Realizability)

Let X, Y be disjoint, non-empty sets of atoms.

An LTL formula ψ on $X \cup Y$ is realizable iff there exists a function $f : (2^X)^+ \to 2^Y$ such that for every finite sequence $X_0, X_1 \ldots X_n$ with $X_i \subseteq X$, path $(X_0 \cup f(X_0))(X_1 \cup f(X_0 X_1)) \ldots (X_n \cup f(X_0 X_1 \ldots X_n))$ satisfies ψ.

Model Checking ATL$^*_R$: lower bound

Let X, Y be disjoint, non-empty sets of atoms.

Define a 2-player, turn-based CGS $M = \langle \mathbb{A}\mathrm{gt}, St, \mathcal{V}, Act, d, o \rangle$ such that

- $\mathbb{A}\mathrm{gt} = \{1, 2\}$
- $St \rightsquigarrow$ valuations of atoms in $X \cup Y$
- $\mathcal{V} : AP \to 2^{St}$ is the identity function
- $Act \rightsquigarrow$ player 1 controls atoms in X, player 2 atoms in Y
- $d \rightsquigarrow$ agents can change the value of any atom, at any state
- $o \rightsquigarrow$ the value of atoms is updated as determined by agents.

Lemma 5.10

An LTL formula ψ is realizable iff $\langle\langle 2 \rangle\rangle \mathrm{X} \tau(\psi)$ is true in M.

- Translation τ "adds" X operators: it is linear.
- LTL realizability can be reduced to ATL$^*_R$ model checking.

Theorem 5.11

ATL$^*_R$ model checking is in 2EXPTIME.

- We can reuse procedure $mcheck(M, \varphi)$ but for formulas of type $\langle\langle A \rangle\rangle \gamma$.
- We provide a separate procedure to deal with such formulas.

Intuition:

1. Guess a perfect-recall strategy s_A for coalition A.
2. Consider the execution tree T consistent with coalition A following strategy s_A.
3. Check the CTL* formula Aγ on T.

Model Checking ATL$^*_R$: upper bound

1. Given a joint strategy s_A and state q, the CGS M can be unfolded into a (q, A)-execution tree: q-rooted tree representing all possible behaviors consistent with coalition A following strategy s_A.
2. There exists a Büchi tree automaton $\mathcal{A}_{M,q,A}$ that accepts exactly all (q, A)-execution trees.
3. There exists a Rabin tree automaton $\mathcal{A}_\gamma$ that accepts exactly all trees that satisfy the CTL* formula Aγ
4. The product $\mathcal{A}_{M,q,A} \times \mathcal{A}_\gamma$ is a Rabin tree automaton that accepts exactly all (q, A)-execution trees that satisfy Aγ.
5. The language of $\mathcal{A}_{M,q,A} \times \mathcal{A}_\gamma$ is non-empty iff $(M, q) \models \langle\langle A \rangle\rangle \gamma$.

Complexity analysis:

- Notice that $|\mathcal{A}_{M,q,A}| = O(|M|)$ and $|\mathcal{A}_\gamma| = 2^{2^{O(\gamma)}}$.
- Then $|\mathcal{A}_{M,q,A} \times \mathcal{A}_\gamma| = O(|\mathcal{A}_{M,q,A}| \times |\mathcal{A}_\gamma|) = O(|M| \times 2^{2^{O(\gamma)}})$.
- Non-emptyness of $\mathcal{A}_{M,q,A} \times \mathcal{A}_\gamma$ can be checked in polynomial time.

Model Checking ATL$_r^*$: lower bound

Theorem 5.12

ATL$_r^*$ model checking is (at least) as hard as LTL model checking (which is PSPACE-hard).

Reduction: an LTL formula φ is equivalent to the ATL* formula $\langle\langle\emptyset\rangle\rangle\varphi$.

Theorem 5.13

ATL$_r^*$ model checking is in PSPACE.

Again, the case of interest is for formulas of type $\langle\langle A\rangle\rangle\gamma$.

1. Guess a memoryless strategy s_A for coalition A: now this can be done in polynomial space.
2. Trim model M according to s_A: all transitions that cannot occur by following s_A are removed.
3. Check the CTL* formula Aγ in the trimmed model M_{s_A}: this can be done in PSPACE.

Complexity analysis:

- This procedure can be performed in NPSPACE = PSPACE.
- The complexity of the whole procedure is in $\mathrm{P}^{\mathrm{PSPACE}}$ = PSPACE.

Model Checking Temporal and Strategic Logics: Summary

	memoryless strategies	perfect recall
CTL	PTIME-complete	
LTL	PSPACE-complete	
CTL*	PSPACE-complete	
ATL	PTIME-complete	
ATL*	PSPACE-complete	2EXPTIME-complete

ATL : memory does not affect the complexity of model checking.

ATL* : memory makes verification strictly harder.

References

N. Bulling, J. Dix, and W. Jamroga.
Model checking logics of strategic ability: Complexity.
In M. Dastani, K. Hindriks, and J.-J. Meyer, eds.: *Specification and Verification of Multi-Agent Systems*, pages 125–159. Springer, 2010.

N. Bulling and W. Jamroga.
Alternating Epistemic Mu-Calculus.
Proceedings of the 22nd International Joint Conference on Artificial Intelligence IJCAI'11, pages 109–114, 2011.

P. Y. Schobbens.
Alternating-time logic with imperfect recall.
Electronic Notes in Theoretical Computer Science, 85(2), 2004.

Part 6: Abilities under Imperfect Information

Ability and Knowledge

- So far we considered games of perfect information: players are completely aware of the structure of the system as well as the current state of the play.
- In concrete MAS this is seldom the case: usually players have only partial information, both about the general setup and about the specific play.
- Hereafter we try to answer the following question:

What can players achieve in such scenarios?

Ability and Knowledge

Classic AI: ability and knowledge are intimately connected

1969 : McCarthy & Hayes:

> *We want a computer program that decides what to do by inferring in a formal language [i.e., a logic] that a certain strategy will achieve a certain goal. This requires formalizing concepts of causality, ability, and knowledge.*

1981 : R. Moore; *Reasoning about Knowledge and Action*

Ability and Knowledge

[McCarthy & Hayes, 1969]: what does it mean that "a computer program π to be able to achieve a state of affairs ϕ?"

1. objective ability (omniscient external observer): "there is a sub-program σ [...] which would achieve ϕ if it were in memory, and control were transferred to π. No assertion is made that π knows σ or even knows that σ exists."
2. subjective ability: "σ exists as above and that σ will achieve ϕ follows from information in memory according to a proof that π is capable of checking."
3. practical ability (*strategy synthesis*): "π's standard problem-solving procedure will find σ if achieving ϕ is ever accepted as a subgoal."

Motivating Example: Rescue Robots

Properties to express

- The robots can rescue person i
- The robots can rescue person i, and they know that they can
- The robots can rescue person i, and they know how to do it

Ability and Knowledge

[Moore, 1981]: *Reasoning about Knowledge and Action*

Formalisation of ability in terms of knowledge and action:

$$(Can\ \phi) \quad \leftrightarrow \quad \exists\alpha K(Res\ \alpha\ \phi) \vee \exists\alpha K(Res\ \alpha(Can\ \phi))$$

- Agents know the identity of actions.
- Hereafter we focus on *strategies* (i.e., conditional plans) rather than simple actions.

Strategies under Uncertainty

- ATL includes no notion of knowledge (or, dually uncertainty)

  ... which makes reasoning in ATL rather unrealistic for MAS.

- In this lecture, we introduce knowledge and uncertainty into reasoning about strategic abilities.

Note on terminology: in Game Theory we have

- *incomplete information*: uncertainty about the game structure
- *imperfect information*: uncertainty about the current state of the game

We use the two terms interchangeably: our models allow for representing both types of uncertainty uniformly.

Motivating Example: Voting

Properties to express

Privacy: The system cannot reveal how a particular voter voted.

Receipt-freeness: The voter cannot gain any information (a receipt) which can be used to prove to a coercer that she voted in a certain way.

Coercion-resistance: The voter cannot cooperate with the coercer to prove to him that she voted in a certain way.

Part 6: Abilities under Imperfect Information

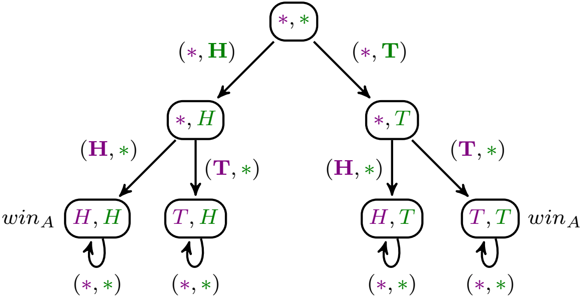
6.1 Strategies and Knowledge

Strategies and Knowledge

How can we reason about **multi-step games with imperfect information**?

- we extend CGS with indistinguishability relations $\sim_a$, one per agent.

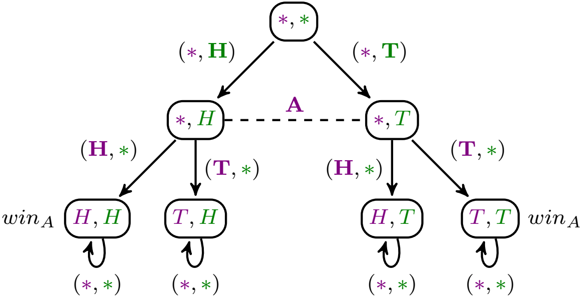
A game of matching pennies:



How can we reason about **multi-step games with imperfect information**?

- we extend CGS with indistinguishability relations $\sim_a$, one per agent.

A game of matching pennies:



Definition 6.1 (iCGS)

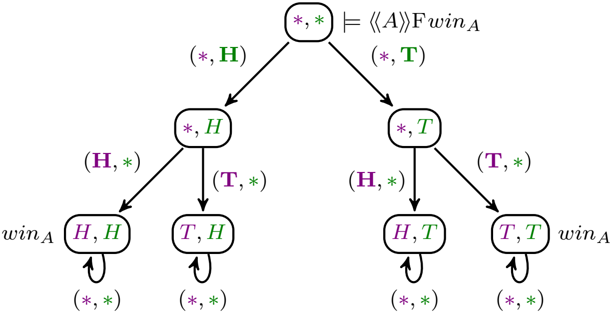
A concurrent game structure with imperfect information is a tuple $M = \langle \mathbb{A}\mathrm{gt}, St, \{\sim_a\}_{a \in \mathbb{A}\mathrm{gt}}, \mathcal{V}, Act, d, o \rangle$, where:

- $\mathbb{A}\mathrm{gt}$, St, $\mathcal{V}$, o are defined as for CGS
- for every $a \in \mathbb{A}\mathrm{gt}$, $\sim_a$ is an equivalence relation on St
- $d : \mathbb{A}\mathrm{gt} \times St \rightarrow 2^{Act}$ defines actions available to an agent in a state such that if $s \sim_a s'$ then $d(s, a) = d(s', a)$.

Standard CSG are iCGS where every $\sim_a$ is the identity relation.

Strategies and Knowledge

Matching pennies again:

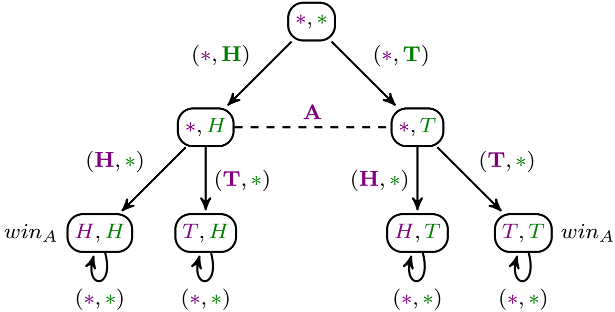


Winning strategy σ such that

- $\sigma(*, *) = *$
- $\sigma(*, H) = H$
- $\sigma(*, T) = T$

Strategies and Knowledge

Matching pennies again:

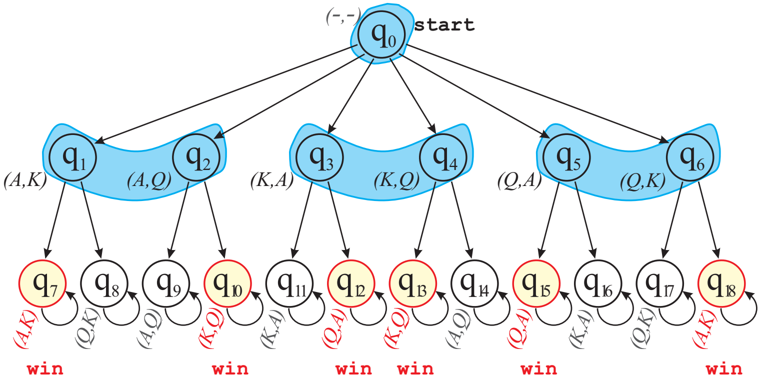


Does strategy σ still make sense?

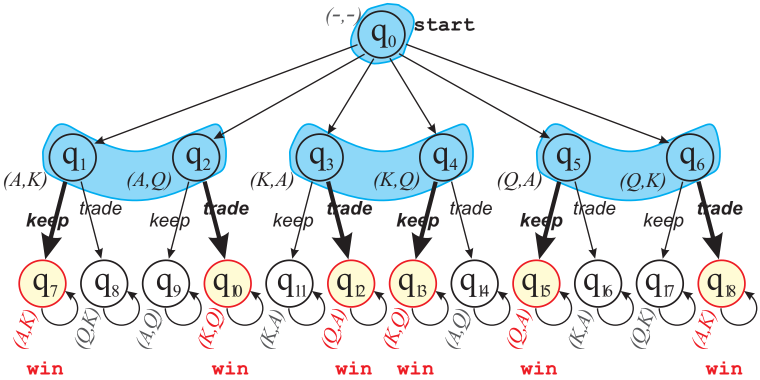
Does $(*, *) \models \langle\langle A \rangle\rangle \mathrm{F}\, win_A$?

Strategies and Knowledge

Kings, Queens, and Aces ($K \geq Q \geq A \geq K$).



Kings, Queens, and Aces ($K \geq Q \geq A \geq K$).

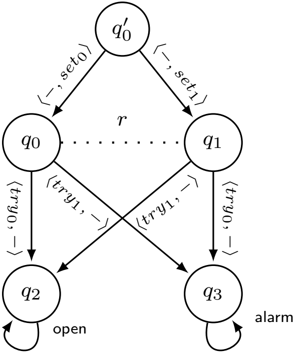


$$(-,-) \models \langle\langle a \rangle\rangle \mathrm{F}\, win$$

Does it make sense?

Schobbens' Robber

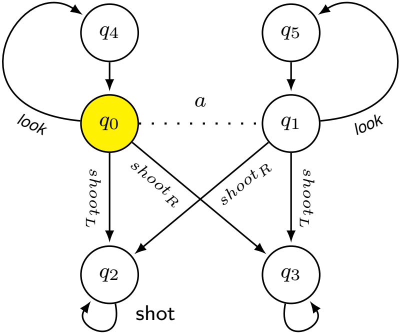
- A vault is protected by a binary code: either 0 or 1.
- The code is set anew every morning by the guard.
- Some time later the robber (r) enters the bank and tries to open the vault.
- However, he doesn't know the current code ($q_0 \sim_r q_1$).



Can we say that the rober has the ability to get access to the vault?

Example: Poor Duck Problem

- A man wants to shoot down a yellow rubber duck in a shooting gallery.
- The duck is in one of two cells, but he does not know in which one.
- The man can either shoot left, right, or reach out to the cells and look.



- The man does not have a (subjective) strategy to shoot the duck in one step.
- He should be able to ensure it in multiple steps if he has a perfect recall.
- The man can shot the duck in one step if he is told the right strategy ...

  ...though he would not be able to come up with it on his own.

Strategies and Knowledge

Problem:

Strategic and epistemic abilities are **not** independent!

$\langle\langle A\rangle\rangle\gamma$ = A can **enforce** γ

It should at least mean that A are able to execute the right strategy!

Executable strategies = uniform strategies

In many cases, we also mean that A are able to identify the strategy...

In order to identify a strategy as successful, the agents must check its outcome paths from indistinguishable states

Definition 6.2 (Uniform strategy)

Strategy s_a is **uniform** iff it returns the same action in indistinguishable states:

- 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.

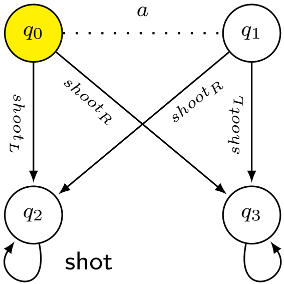
A collective strategy is uniform iff it consists only of uniform individual strategies.

Strategies and Knowledge: Poor Duck Problem

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!



Levels of Strategic Ability

Our cases for $\langle\langle A \rangle\rangle \gamma$ under imperfect information:

1. There is σ (not necessarily executable!) such that, for every execution of σ, γ holds.
2. There is a uniform σ such that, for every execution of σ, γ holds (objective interpretation).
3. A know that there is a uniform σ such that, for every execution of σ, γ holds.
4. There is a uniform σ such that A know that, for every execution of σ, γ holds (subjective interpretation).

Hereafter we focus on 2 and 4 (starting with 2).

Semantics of ATL$_i$*: Objective Interpretation

Definition 6.3 (Semantics of ATL$_i$*)

$(M, q) \models_i \langle\langle A \rangle\rangle \gamma$ iff there is a collective uniform strategy s_A such that, for every path $\lambda \in out(q, s_A)$, $(M, \lambda) \models \gamma$.

Definition 6.4 (Semantics of ATL$_i$*: path formulae)

Same as for CTL* and ATL*

As for ATL*, we can consider both perfect and imperfect recall strategies.

Semantics of ATL$_i$*: Objective Interpretation

Definition 6.3 (Semantics of ATL$_i$*)

$(M, q) \models_i \langle\langle A \rangle\rangle\gamma$ iff there is a collective uniform strategy s_A such that, for every path $\lambda \in out(q, s_A)$, $(M, \lambda) \models \gamma$.

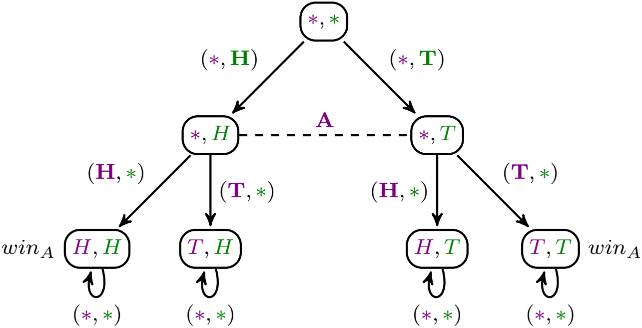
Definition 6.4 (Semantics of ATL$_i$*: path formulae)

$(M, \lambda) \models_i \varphi$	iff $(M, \lambda[0]) \models_i \varphi$, for a state formula φ
$(M, \lambda) \models_i \mathrm{X}\gamma$	iff $(M, \lambda[1..\infty]) \models_i \gamma$
$(M, \lambda) \models_i \gamma_1 \mathrm{U} \gamma_2$	iff $(M, \lambda[k..\infty]) \models_i \gamma_2$ for some $k \geq 0$, and $(M, \lambda[j..\infty]) \models_i \gamma_1$ for all $0 \leq j \leq k$

As for ATL*, we can consider both perfect and imperfect recall strategies.

Strategies and Knowledge

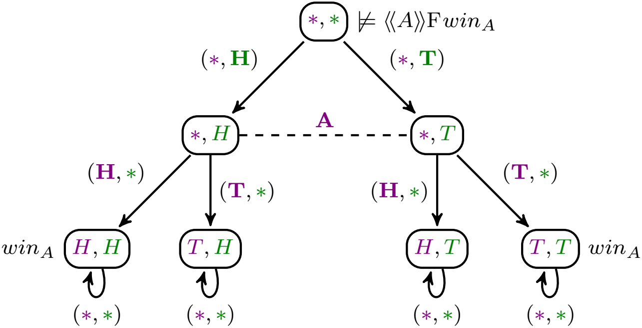
Matching pennies revisited:



Alice no longer has a winning strategy!

Strategies and Knowledge

Matching pennies revisited:



Alice no longer has a winning strategy!

Alternating-time Temporal Logic: Summary

Four variants of ability: IR, Ir, iR, ir (Schobbens 2004)

- memory R/r: perfect/imperfect recall.
- knowledge I/i: perfect/imperfect information.
- r: $s_a : St \to Act$ (memoryless strategies)
- R: $s_a : St^+ \to Act$ (perfect recall strategies)
- i: only uniform strategies,
- I: no restrictions
- r: s_a is uniform iff $q \sim_a q' \Rightarrow s_a(q) = s_a(q')$
- R: s_a is uniform iff $h \approx_a h' \Rightarrow s_a(h) = s_a(h')$
  where $h \approx_a h'$ iff for all i, $h[i] \sim_a h'[i]$

	imperfect recall	perfect recall
perfect information	ATL_{Ir}*	ATL_{IR}*
imperfect information	ATL_{ir}*	ATL_{iR}*

- and similarly for ATL.

Part 6: Abilities under Imperfect Information

6.2 Properties of ATL_i

Fixpoint (Non-)Equivalences

Interesting: $\langle\!\langle A \rangle\!\rangle_i$ **are not fixpoint operators** any more!

Theorem 6.5

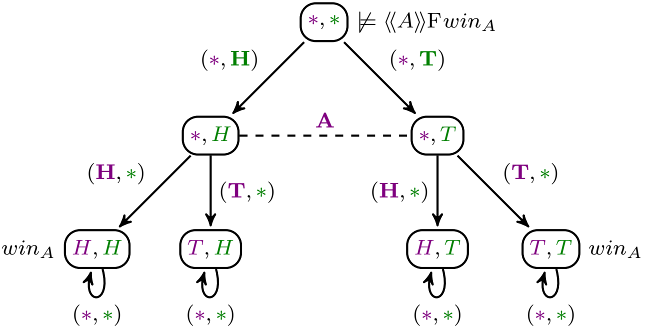
*The following formulas are **not** valid for ATLᵢ:*

$$\begin{aligned}
\langle\!\langle A \rangle\!\rangle_i \varphi_1 \mathrm{U} \varphi_2 &\quad\leftrightarrow\quad \varphi_2 \vee \varphi_1 \wedge \langle\!\langle A \rangle\!\rangle_i \mathrm{X} \langle\!\langle A \rangle\!\rangle_i \varphi_1 \mathrm{U} \varphi_2 \\
\langle\!\langle A \rangle\!\rangle_i \mathrm{F} \varphi &\quad\leftrightarrow\quad \varphi \vee \langle\!\langle A \rangle\!\rangle_i \mathrm{X} \langle\!\langle A \rangle\!\rangle_i \mathrm{F} \varphi \\
\langle\!\langle A \rangle\!\rangle_i \mathrm{G} \varphi &\quad\leftrightarrow\quad \varphi \wedge \langle\!\langle A \rangle\!\rangle_i \mathrm{X} \langle\!\langle A \rangle\!\rangle_i \mathrm{G} \varphi
\end{aligned}$$

What is this about? forgetting and non-composability of strategies

$\rightsquigarrow$ we cannot have incremental model checking algorithms as for ATL$_\mathrm{I}$.

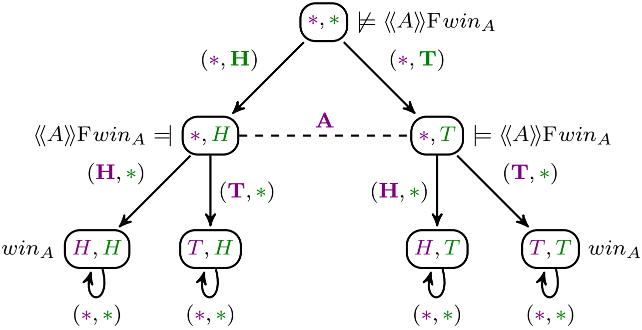
Non-Composability of Strategies: Matching Pennies



Hence,

$$\langle\langle A \rangle\rangle \mathrm{X} \langle\langle A \rangle\rangle \mathrm{F}\, win_A \not\to \langle\langle A \rangle\rangle \mathrm{F}\, win_A$$

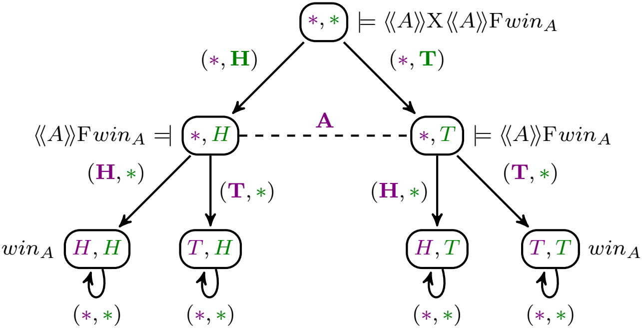
Non-Composability of Strategies: Matching Pennies



Hence,

$$\langle\langle A\rangle\rangle \mathrm{X} \langle\langle A\rangle\rangle \mathrm{F}\, win_A \not\to \langle\langle A\rangle\rangle \mathrm{F}\, win_A$$

Non-Composability of Strategies: Matching Pennies



Hence,

$$\langle\langle A\rangle\rangle \mathrm{X} \langle\langle A\rangle\rangle \mathrm{F}\, win_A \not\to \langle\langle A\rangle\rangle \mathrm{F}\, win_A$$

Non-Composability of Strategies

In general, the following are validities in ATL_{ir}:

$$\begin{aligned}
\langle\!\langle A\rangle\!\rangle_i \varphi_1 \mathrm{U} \varphi_2 &\quad\rightarrow\quad \varphi_2 \vee (\varphi_1 \wedge \langle\!\langle A\rangle\!\rangle_i \mathrm{X} \langle\!\langle A\rangle\!\rangle_i \varphi_1 \mathrm{U} \varphi_2) \\
\langle\!\langle A\rangle\!\rangle_i \mathrm{F} \varphi &\quad\rightarrow\quad \varphi \vee \langle\!\langle A\rangle\!\rangle_i \mathrm{X} \langle\!\langle A\rangle\!\rangle_i \mathrm{F} \varphi \\
\langle\!\langle A\rangle\!\rangle_i \mathrm{G} \varphi &\quad\rightarrow\quad \varphi \wedge \langle\!\langle A\rangle\!\rangle_i \mathrm{X} \langle\!\langle A\rangle\!\rangle_i \mathrm{G} \varphi
\end{aligned}$$

But,

$$\begin{aligned}
\varphi_2 \vee (\varphi_1 \wedge \langle\!\langle A\rangle\!\rangle_i \mathrm{X} \langle\!\langle A\rangle\!\rangle_i \varphi_1 \mathrm{U} \varphi_2) &\quad\not\rightarrow\quad \langle\!\langle A\rangle\!\rangle_i \varphi_1 \mathrm{U} \varphi_2 \\
\varphi \vee \langle\!\langle A\rangle\!\rangle_i \mathrm{X} \langle\!\langle A\rangle\!\rangle_i \mathrm{F} \varphi &\quad\not\rightarrow\quad \langle\!\langle A\rangle\!\rangle_i \mathrm{F} \varphi \\
\varphi \wedge \langle\!\langle A\rangle\!\rangle_i \mathrm{X} \langle\!\langle A\rangle\!\rangle_i \mathrm{G} \varphi &\quad\not\rightarrow\quad \langle\!\langle A\rangle\!\rangle_i \mathrm{G} \varphi
\end{aligned}$$

For ATL_{iR} the opposite is the case.

Fixpoint (Non-)Equivalences

Conjecture

Strategies cannot be synthesized incrementally.

Indeed...

Theorem (Schobbens 2004; Jamroga & Dix 2006)

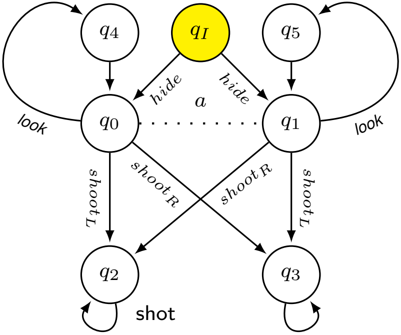
Model checking $\mathrm{ATL}_{\mathrm{ir}}$ is Δ_2^P-complete.

Memory and Imperfect Information

- Under perfect information, memory does not affect the interpretation of ATL.
- But it does affect ATL*.
- What is it like for imperfect information?

Memory and Imperfect Information

- Obviously, $\text{ATL}_{\text{ir}}^* \neq \text{ATL}_{\text{iR}}^*$.
- But also $\text{ATL}_{\text{ir}} \neq \text{ATL}_{\text{iR}}$:



$$q_I \not\models_{ir} \langle\langle a \rangle\rangle \mathrm{F}\,\mathsf{shot}$$

$$q_I \models_{iR} \langle\langle a \rangle\rangle \mathrm{F}\,\mathsf{shot}$$

Motivating Example: Rescue Robots

- The robots can rescue all the people in the building.

  $\bigwedge_{i \in \mathit{People}} \langle\langle \mathit{Robots} \rangle\rangle \mathrm{F}\, \mathsf{safe}_i$

  Alternative formalization:

  $\langle\langle \mathit{Robots} \rangle\rangle \mathrm{F} \left(\bigwedge_{i \in \mathit{People}} \mathsf{safe}_i \right)$

  Note: these look like the specifications in ATL, but a uniform strategy is required for the robots now!

- The robots can rescue all the people, and they know that they can

  Cannot be expressed in $\mathbf{ATL_i^*}$!

- The robots can rescue all the people, and they know how to do it

  Cannot be expressed in $\mathbf{ATL_i^*}$!

Motivating Example: Voting

- The system cannot reveal how a particular voter voted.

  $\neg\langle\langle system\rangle\rangle F(\bigvee_{c \in \mathit{Candidates}} \mathsf{revealedVote_{i,c}})$

- A voter i can gain no receipt which can be used to prove that she voted in a certain way.

  $\neg\langle\langle i\rangle\rangle F(\bigvee_{c \in \mathit{Candidates}} \mathsf{receiptVote_{i,c}})$

- A voter i cannot cooperate with the coercer to prove to him that she voted in a certain way.

  Cannot be expressed in ATL$_i$* !

Note: now for the first two properties we can model the epistemic capabilities in the scenario!

Part 6: Abilities under Imperfect Information

6.3 The Subjective Interpretation

Levels of Strategic Ability

Our cases for $\langle\langle A\rangle\rangle\gamma$ under imperfect information:

1. There is a strategy σ (not necessarily executable!) such that, for every execution of σ, γ holds.
2. There is a uniform strategy σ such that, for every execution of σ, γ holds (objective interpretation).
3. A know that there is a uniform σ such that, for every execution of σ, γ holds.
4. There is a uniform σ such that A know that, for every execution of σ, γ holds (subjective interpretation).

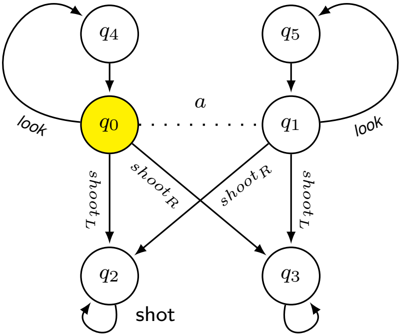
We dealt with 2 in the last lecture

Hereafter we focus on 4. **Why?**

Strategies and Knowledge: Poor Duck Problem

Note:

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



$q_0 \models \langle\langle a \rangle\rangle \mathrm{X}\ shot$ and $q_1 \models \langle\langle a \rangle\rangle \mathrm{X} shot$

But for two different strategies!

Case **[4]**: **knowing how to play**

- Single agent case: we consider all paths starting from indistinguishable states:

$$out_s(q, s_a) \quad = \quad \bigcup_{q' \sim_a q} out_o(q', s_a)$$

- $\langle\langle a \rangle\rangle_{ir}\gamma$: agent a knows how to play to enforce γ from all the states she considers possible
- What about coalitions?

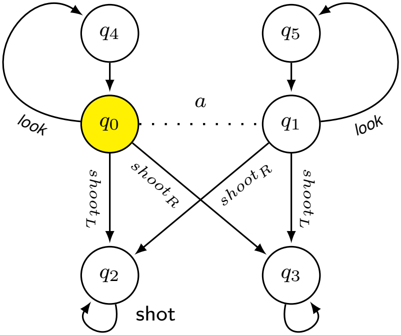
$$out_s(q, s_A) \quad = \quad \bigcup_{a \in A} \bigcup_{q' \sim_a q} out_o(q', s_A)$$

- $\langle\langle A \rangle\rangle_{ir}\gamma$: all agents in A know how to play to enforce γ.

Definition 6.6 (Subjective Semantics of ATL$_i$*)

$(M, q) \models_s \langle\langle A \rangle\rangle\gamma$ iff there is a collective uniform strategy s_A such that, for every path $\lambda \in out_s(q, s_A)$, $(M, \lambda) \models \gamma$.

Strategies and Knowledge: Poor Duck Problem Revisited

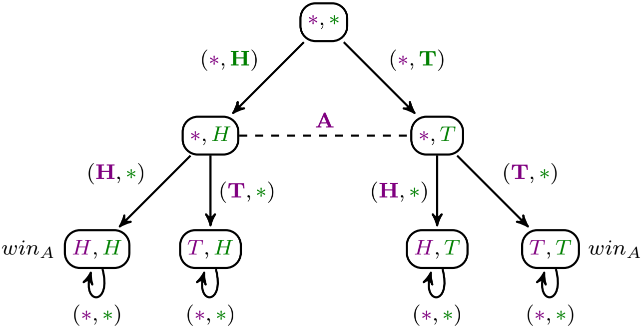


$$q_0 \not\models_r \langle\langle a \rangle\rangle \mathrm{F}\ shot \text{ and } q_1 \not\models_r \langle\langle a \rangle\rangle \mathrm{F}\, shot$$

But,

$$q_0 \models_R \langle\langle a \rangle\rangle \mathrm{F}\ shot \text{ and } q_1 \models_R \langle\langle a \rangle\rangle \mathrm{F}\, shot$$

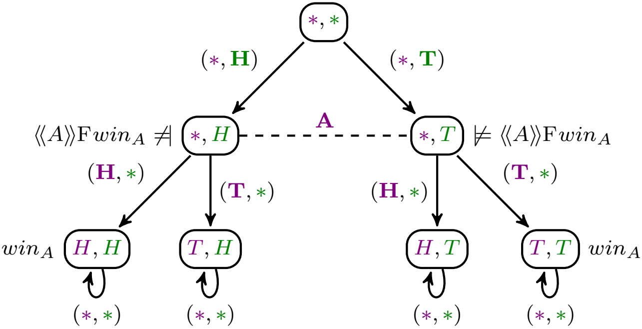
Decomposability of Strategies: Matching Pennies



Hence,

$$\langle\langle A\rangle\rangle \mathrm{X} \langle\langle A\rangle\rangle \mathrm{F}\, win_A \rightarrow \langle\langle A\rangle\rangle \mathrm{F}\, win_A$$

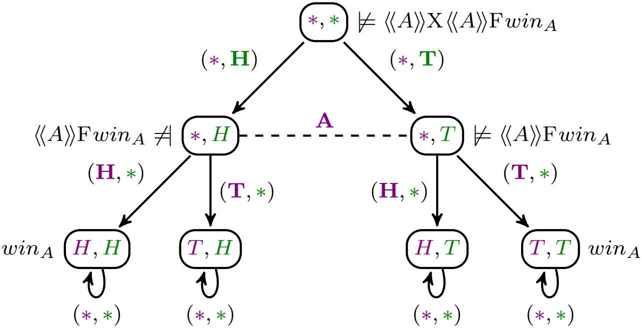
Decomposability of Strategies: Matching Pennies



Hence,

$$\langle\langle A\rangle\rangle \mathrm{X} \langle\langle A\rangle\rangle \mathrm{F}\, win_A \rightarrow \langle\langle A\rangle\rangle \mathrm{F}\, win_A$$

Decomposability of Strategies: Matching Pennies

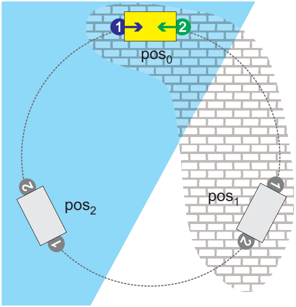


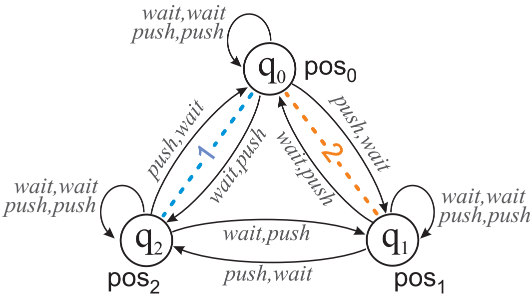
Hence,

$$\langle\langle A\rangle\rangle \mathrm{X} \langle\langle A\rangle\rangle \mathrm{F}\, win_A \to \langle\langle A\rangle\rangle \mathrm{F}\, win_A$$

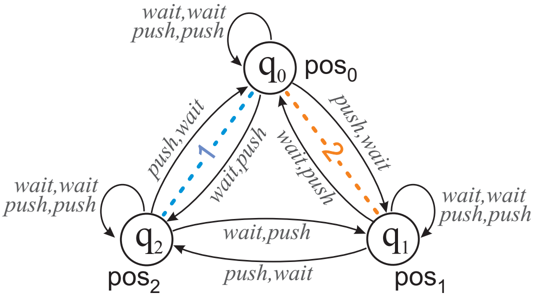
Example: Robots and Carriage

- Robot 1 only perceives the color of the surface;
- Robot 2 only perceives the texture.





Example: Robots and Carriage



$\mathsf{pos}_0 \rightarrow \neg\langle\langle 1 \rangle\rangle_{sr}\, \mathrm{G}\, \neg \mathsf{pos}_1$

$\mathsf{pos}_0 \rightarrow \neg\langle\langle 1, 2 \rangle\rangle_{sr}\, \mathrm{G}\, \neg \mathsf{pos}_1$

$\mathsf{pos}_0 \rightarrow \langle\langle 1, 2 \rangle\rangle_{sr}\, \mathrm{F}\, \mathsf{pos}_1$

Part 7: Model Checking Imperfect Information

Model Checking Imperfect Information

Model Checking under Imperfect Information

Imperfect information makes the model checking problem harder to solve.

ATL	I	i
r	linear time	$\boldsymbol{\Delta}_2^{\mathbf{P}}$-complete
R		undecidable

ATL*	I	i
r	PSPACE-c	
R	2EXPTIME-c	undecidable

The complexity does not change for the objective and subjective interpretation.

We first consider imperfect recall and then perfect recall.

Model Checking ATL_{ir}

Recall: $\langle\langle A \rangle\rangle_i$ are not fixpoint operators any more

Conjecture

Strategies for A cannot be synthesized incrementally.

Indeed,

- the choice of an action at state q has non-local consequences: it fixes agent a's choices at all states q' indistinguishable from q for i.
- for two different members of coalition A, uniformity of their parts of the coalitional strategy imposes different constraints on their choices.
- the agents' ability to *identify* a strategy as winning also varies throughout the game in an arbitrary way (agents can learn as well as forget).

Model Checking ATL_{ir}

Theorem (Schobbens 2004; Jamroga & Dix 2006)

Model checking ATL_{ir} is $\mathbf{\Delta_2^P}$-complete in the number of transitions in the model and the length of the formula.

$\mathbf{\Delta_2^P}$: class of problems solvable in polynomial time by a deterministic Turing machine making calls to an oracle solving NP problems.

How to prove that?

We prove the upper bound by showing a nondeterministic algorithm, and the lower bound by a reduction to an appropriate problem

Model Checking ATL_{ir}: Upper Bound

Checking $(\mathcal{M}, q) \models_{ir} \langle\langle A\rangle\rangle\gamma$ where γ includes no nested cooperation modalities:

1. Guess a uniform, memoryless strategy s_A for coalition A
   - this can be done in polynomial time.
2. Remove from $\mathcal{M}$ all the transitions that are *not* going to be executed according to s_A
3. Model-check the CTL formula Aγ in the resulting model.
   - recall that model-checking CTL is in PTIME.

This procedure is in non-deterministic polynomial time (NP).

For nested cooperation modalities, we proceed recursively (bottom up)

The whole procedure calls a linear number of times ($\mathcal{O}(|\varphi|)$) a procedure in NP: $\mathrm{P}^{\mathrm{NP}} = \mathbf{\Delta_2^P}$.

Model Checking ATL_{ir}: Lower Bound

We prove NP-hardness by reducing model checking ATL_{ir} to the Boolean satisfiability problem (SAT)

Definition (Boolean satisfiability)

Input: Boolean formula $\varphi(x_1, \ldots, x_k)$ in Conjunctive Normal Form (CNF).

Output: True iff $\exists v_1, \ldots, v_k \; \varphi(v_1, \ldots, v_k)$.

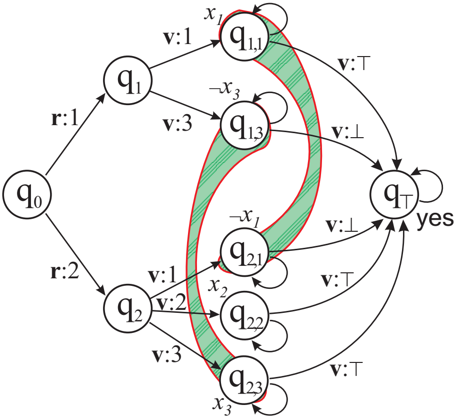
Proposition

SAT is NP-complete.

So:

- If we reduce SAT to our problem, then our problem must be NP-hard.

Model Checking ATL_{ir}: Lower Bound



Model M_φ for $\varphi \equiv (x_1 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$

Proposition

$\exists v_1, \ldots, v_k \,.\, \varphi(v_1, \ldots, v_k)$ iff $(M_\varphi, q_0) \models \langle\langle \mathbf{v} \rangle\rangle \mathrm{F}\,\text{yes}$.

Model Checking ATL_{ir}*

Good News: Model checking ATL_{ir}* is PSPACE-complete (just like perfect information).

Theorem

Model checking ATL_{ir}* is in PSPACE.

Again, the case of interest is for formulas of type $\langle\langle A \rangle\rangle \gamma$.

1. Guess a uniform, memoryless strategy s_A for coalition A
   - this can be done in polynomial time.
2. Trim model M according to s_A: all transitions that cannot occur by following s_A are removed.
3. Check the CTL* formula Aγ in the trimmed model M_{s_A}
   - recall that model-checking CTL* is in PSPACE.

This procedure can be performed in non-deterministic polynomial space NPSPACE = PSPACE.

For nested cooperation modalities, we proceed recursively (bottom up).

The whole procedure calls a linear number of times ($\mathcal{O}(|\varphi|)$) a procedure in PSPACE: P^{PSPACE} = PSPACE.

PSPACE-hardness: model checking ATL_{ir}* is as hard as model checking LTL.

Corollary

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

Imperfect information makes model checking harder!

Model Checking ATL$_{\text{iR}}$

What about agents with perfect recall and imperfect information?

The news are bad...

Theorem [Dima and Tiplea, 2011]

Model checking ATL$_{\text{iR}}$ is **undecidable**.

Proof: by a reduction of the non-halting problem for deterministic Turing machines.

- 3 players suffice (2 proponents + 1 opponent)
- The players play one at a time (taking turns)
- Subsequent configurations of the TM T are represented as levels in the tree unfolding of the iCGS M_T.

Proposition

A deterministic Turing machine T does not halt on the empty word iff $(M_T, s_{init}) \models_{iR} \langle\langle 1, 2 \rangle\rangle \mathrm{G} ok$.

Model Checking Complexity for Variants of ATL: Summary

ATL	I	i
r	linear time	$\mathbf{\Delta_2^P}$-complete
R		undecidable

ATL*	I	i
r	PSPACE-c	
R	2EXPTIME-c	undecidable

References

N. Bulling, J. Dix, and W. Jamroga.
Model checking logics of strategic ability: Complexity.
In M. Dastani, K. Hindriks, and J.-J. Meyer, eds.: *Specification and Verification of Multi-Agent Systems*, pages 125–159. Springer, 2010.

R. Alur, T. A. Henzinger, and O. Kupferman.
Alternating-time Temporal Logic.
Journal of the ACM, 49:672–713, 2002.

P. Y. Schobbens.
Alternating-time logic with imperfect recall.
Electronic Notes in Theoretical Computer Science, 85(2), 2004.