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

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.
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:


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


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

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

Reasoning about Systems
1.1 Multi-Agent Systems
1.2 The Role of Logics for MAS
1.3 Formal Verification
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

\[ \text{act} : \text{set of percept sequences} \rightarrow \text{set of actions} \]

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 \textit{reasoning} about systems
- makes one \textit{realise} the implicit \textit{assumptions}
- ... and then we can:
  - \textit{investigate} them, \textit{accept or reject} them, or
  - \textit{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 $\phi_P$ 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 ⇒ Rational Synthesis

  the attacker has a strategy to learn the secret eventually.

⇒ 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.


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.
Rapidly increasing integration of ICT in different applications:
- embedded systems (automotive)
- communication and security protocols
- transportation systems (autonomous vehicles)
⇒ reliability increasingly 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.  
  ⇒ models must be finite.
Model Checking

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

System $S$ \[\rightarrow\] Modeling \[\rightarrow\] Model $M_S$

Specification $P$ \[\rightarrow\] Formalising \[\rightarrow\] Formula $\phi_P$

Model Checking: $M_S \models \phi_P$?

- True
- False (+ counterexample)

Time Out
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.
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 $\langle A \rangle$: “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.*  

M. Wooldridge.  
*An Introduction to Multi Agent Systems.*  

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.*

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.*
Part 2: Reasoning about Time and Change

Reasoning about Time and Change
2.1 Temporal Logics
2.2 Linear Temporal Logic
2.3 Computation Tree Logic
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:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X \varphi$</td>
<td>$\varphi$ is true in the next moment in time</td>
</tr>
<tr>
<td>$G \varphi$</td>
<td>$\varphi$ is true in all future moments (globally true)</td>
</tr>
<tr>
<td>$F \varphi$</td>
<td>$\varphi$ is true in some future moment (finally true)</td>
</tr>
<tr>
<td>$\varphi U \psi$</td>
<td>$\varphi$ is true until the moment when $\psi$ becomes true</td>
</tr>
</tbody>
</table>

Example formulas:

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

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

\[
\langle St, \rightarrow \rangle
\]

where:

- \( St \) is a non-empty set of states,
- \( \rightarrow \subseteq St \times St \) is a transition relation.

**Note**: when we add a valuation \( \nu : St \rightarrow 2^{AP} \) of atoms, we get a **Kripke model**!
Definition 2.2 (Paths in a transition system)

A path $\lambda$ 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 $\text{pos}_0$, $\text{pos}_1$, $\text{pos}_2$.
- Transition system $TS = \langle St, \rightarrow \rangle$, where
  - $St = \{ q_0, q_1, q_2 \}$
  - $q_i \rightarrow q_i$, for $i \in \{0, 1, 2\}$, and $q_0 \rightarrow q_1$, $q_1 \rightarrow q_2$, $q_2 \rightarrow q_0$
A rocket can be moved between London (atom \textit{roL}) and Paris (atom \textit{roP}).

The cargo can be in London (\textit{caL}), Paris (\textit{caP}), or inside the rocket (\textit{caR}).

The rocket can fly only if its fuel tank if full (\textit{fuelOK}).

When it flies, it consumes fuel, and \textit{nofuel} holds after each flight.
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:

\[ G \neg \text{bankrupt} \]
\[ G (\text{fuelOK} \lor X \text{fuelOK}) \]
and so on . . .

Usually: \[ G \neg \ldots / G \ldots \]
Liveness/achievement:

“something good will finally happen”

Typical examples:

\[ F \text{ rich} \]
\[ FG \text{ rich} \]
requested \( \rightarrow \) F granted
and so on . . .

Usually: \( F . . . / FG . . . \)
Reasoning about Time: Fairness Properties

Fairness/service:

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

Typical examples:

- GF connected
- \( \neg FG \text{ down} \)
- \( G(\text{calling} \rightarrow F\text{answering}) \)
- \( (GF \text{ attempt}) \rightarrow (GF \text{ success}) \)
and so on . . .

Usually: GF . . . / \( \neg FG . . . \)

Fairness properties:

- useful when scheduling processes, responding to messages, etc.
- good for specifying properties of the environment
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
  - $\phi \land \psi$ conjunction

- Temporal operators
  - $X\phi$ neXt $\phi$
  - $\phi U \psi$ $\phi$ Until $\psi$

LTL is a logic to express properties of linear time.
Derived Operators

\[
\begin{align*}
\phi \lor \psi & ::= \neg(\neg \phi \land \neg \psi) \\
\phi \rightarrow \psi & ::= \neg \phi \lor \psi \\
\phi \leftrightarrow \psi & ::= (\phi \rightarrow \psi) \land (\psi \rightarrow \phi) \\
true & ::= \phi \lor \neg \phi \\
false & ::= \phi \land \neg \phi = \neg true \\
F \phi & ::= trueU \phi & \text{Finally } \phi \\
G \phi & = \neg F \neg \phi & \text{Globally } \phi
\end{align*}
\]

- **Priority order**: unary operators bind more strongly than binary operators; \(\neg\) et \(X\) bind equally strong; \(U\) takes precedence over \(\land\), \(\lor\) and \(\rightarrow\).
- Parentheses are omitted whenever appropriate: \(\varphi_1 U \varphi_2 = ((\varphi_1)U(\varphi_2))\).
- Operator \(U\) is right-associative: \(\varphi_1 U \varphi_2 U \varphi_3 = \varphi_1 U (\varphi_2 U \varphi_3)\).
- 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. $X(x = 1)$?
Linear Temporal Logic: Syntax

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

1. $X(x = 1)$ ✓

2. $U(x = 1)$?
Linear Temporal Logic: Syntax

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

1. $X(x = 1)$ ✓
2. $U(x = 1)$ ✗
3. $(x < 2) \lor G(x = 1)$ ?
Linear Temporal Logic: Syntax

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

1. $X(x = 1)$ ✓
2. $U(x = 1)$ ✗
3. $(x < 2) \lor G(x = 1)$ ✓
4. $(x = 1)FX(x \geq 3)$?
Linear Temporal Logic: Syntax

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

1. $X(x = 1)$ ✓
2. $U(x = 1)$ ×
3. $(x < 2) \lor G(x = 1)$ ✓
4. $(x = 1)F X(x \geq 3)$ ×
5. $X \rightarrow (true U(x = 1))$?
Linear Temporal Logic: Syntax

Let \( AP = \{x = 1, x < 2, x \geq 3\} \).

Then, what about:

1. \( X(x = 1) \) ✓
2. \( U(x = 1) \) ✗
3. \( (x < 2) \lor G(x = 1) \) ✓
4. \( (x = 1) FX(x \geq 3) \) ✗
5. \( X \rightarrow (true U(x = 1)) \) ✗
6. \( X(x = 1 \land GX(x \geq 3)) \)?
Let $AP = \{x = 1, x < 2, x \geq 3\}$. Then, what about:

1. $X(x = 1)$ ✓
2. $U(x = 1)$ ✗
3. $(x < 2) \lor G(x = 1)$ ✓
4. $(x = 1)FX(x \geq 3)$ ✗
5. $X \rightarrow (trueU(x = 1))$ ✗
6. $X(x = 1 \land GX(x \geq 3))$ ✓
7. $X(trueU(x = 1)) \rightarrow G(x = 1)$?
Linear Temporal Logic: Syntax

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

1. \( X(x = 1) \) ✓
2. \( U(x = 1) \) ❌
3. \( (x < 2) \lor G(x = 1) \) ✓
4. \( (x = 1)FX(x \geq 3) \) ❌
5. \( X \rightarrow (trueU(x = 1)) \) ❌
6. \( X(x = 1 \land GX(x \geq 3)) \) ✓
7. \( X(trueU(x = 1)) \rightarrow G(x = 1) \) ✓
Intuitive Meaning

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

atom $a$: 

$Xa$: 

$aUb$: 

$Fa$: 

$Ga$: 

Linear-time Properties: Mutual Exclusion

- **mutual exclusion**: $G \models (\neg cr_1 \land \neg cr_2)$
- **(weak) starvation freedom**: $(GF_{wa_1} \rightarrow GF cr_1) \land (GF_{wa_2} \rightarrow GF cr_2)$
- **(strong) starvation freedom**: $GF cr_1 \land GF cr_2$
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 $\lambda$.

Evaluation $\mathcal{V} : St \rightarrow 2^{AP}$ of atoms at particular time moments is also needed.

Notation:
- $\lambda[i]$: $i$th 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

$q_0, q_1, q_2, q_2, \ldots$?
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. $q_0, q_1, q_2, q_2, \ldots = q_0, q_1, (q_2)^\omega \checkmark$

2. $q_0, q_1, q_2, q_0, q_1, q_2, \ldots = (q_0, q_1, q_2)^\omega \checkmark$

3. $q_0, q_2, q_1, \ldots ?$
Example: Robots and Carriage

1. $q_0, q_1, q_2, q_2, \ldots = q_0, q_1, (q_2)^\omega \checkmark$
2. $q_0, q_1, q_2, q_0, q_1, q_2, \ldots = (q_0, q_1, q_2)^\omega \checkmark$
3. $q_0, q_2, q_1, \ldots = (q_0, q_2, q_1) \cdot St^\omega \times$
4. $q_0, q_0, \ldots$?
Example: Robots and Carriage

\[ q_0, q_1, q_2, q_2, \ldots = q_0, q_1, (q_2)^\omega \checkmark \]

\[ q_0, q_1, q_2, q_0, q_1, q_2, \ldots = (q_0, q_1, q_2)^\omega \checkmark \]

\[ q_0, q_2, q_1, \ldots = (q_0, q_2, q_1) \cdot St^\omega \times \]

\[ q_0, q_0, \ldots = (q_0)^\omega \checkmark \]

\[ q_1, q_1, q_2, q_0, q_1, q_0, \ldots ? \]
Example: Robots and Carriage

1. $q_0, q_1, q_2, q_2, \ldots = q_0, q_1, (q_2)^\omega \quad \checkmark$
2. $q_0, q_1, q_2, q_0, q_1, q_2, \ldots = (q_0, q_1, q_2)^\omega \quad \checkmark$
3. $q_0, q_2, q_1, \ldots = (q_0, q_2, q_1) \cdot St^\omega \quad \times$
4. $q_0, q_0, \ldots = (q_0)^\omega \quad \checkmark$
5. $q_1, q_1, q_2, q_0, q_1, q_0, \ldots = q_1, q_1, q_2, q_0, q_1, q_0 \cdot St^\omega \quad \times$
Definition 2.4 (Semantics of LTL)

\[ \lambda \models true \]
\[ \lambda \models p \quad \text{iff} \quad p \text{ is true at initial moment } \lambda[0] \text{ (i.e., } p \in \mathcal{V}(\lambda[0])) \]
\[ \lambda \models \neg \varphi \quad \text{iff not } \lambda \models \varphi \]
\[ \lambda \models \varphi \land \psi \quad \text{iff } \lambda \models \varphi \text{ and } \lambda \models \psi \]
**Linear Temporal Logic: Semantics**

<table>
<thead>
<tr>
<th><strong>Definition 2.4 (Semantics of LTL)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda \models true$</td>
</tr>
<tr>
<td>$\lambda \models p$</td>
</tr>
<tr>
<td>$\lambda \models \neg \varphi$</td>
</tr>
<tr>
<td>$\lambda \models \varphi \land \psi$</td>
</tr>
<tr>
<td>$\lambda \models X \varphi$</td>
</tr>
</tbody>
</table>
Definition 2.4 (Semantics of LTL)

\[
\lambda \models true \\
\lambda \models p \quad \text{iff } p \text{ is true at initial moment } \lambda[0] \text{ (i.e., } p \in V(\lambda[0])) \\
\lambda \models \neg \varphi \quad \text{iff not } \lambda \models \varphi \\
\lambda \models \varphi \land \psi \quad \text{iff } \lambda \models \varphi \text{ and } \lambda \models \psi \\
\lambda \models X \varphi \quad \text{iff } \lambda[1] \models \varphi \quad \text{No!}
\]
Linear Temporal Logic: Semantics

Definition 2.4 (Semantics of LTL)

\[ \lambda \models true \]
\[ \lambda \models p \quad \text{iff} \quad p \text{ is true at initial moment } \lambda[0] \text{ (i.e., } p \in V(\lambda[0])) \]
\[ \lambda \models \neg \varphi \quad \text{iff not } \lambda \models \varphi \]
\[ \lambda \models \varphi \land \psi \quad \text{iff } \lambda \models \varphi \text{ and } \lambda \models \psi \]
\[ \lambda \models X \varphi \quad \text{iff } \lambda[1..\infty] \models \varphi \]
Linear Temporal Logic: Semantics

Definition 2.4 (Semantics of LTL)

| $\lambda \models true$ | iff $p$ is true at initial moment $\lambda[0]$ (i.e., $p \in V(\lambda[0])$) |
| $\lambda \models p$ | iff not $\lambda \models \varphi$ |
| $\lambda \models \varphi \land \psi$ | iff $\lambda \models \varphi$ and $\lambda \models \psi$ |
| $\lambda \models X \varphi$ | iff $\lambda[1..\infty] \models \varphi$ |
| $\lambda \models \varphi U \psi$ | 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 $F\varphi \equiv true U \varphi$ and $G\varphi \equiv \neg F \neg \varphi$.

### Semantics of derived operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda \models F\varphi$</td>
<td>iff $\lambda[i..\infty] \models \varphi$ for some $i \geq 0$</td>
</tr>
<tr>
<td>$\lambda \models G\varphi$</td>
<td>iff $\lambda[i..\infty] \models \varphi$ for all $i \geq 0$</td>
</tr>
<tr>
<td>$\lambda \models GF\varphi$</td>
<td>iff for all $i \geq 0$, for some $j \geq i$, $\lambda[j..\infty] \models \varphi$ (infinitely often)</td>
</tr>
<tr>
<td>$\lambda \models FG\varphi$</td>
<td>iff for some $i \geq 0$, for all $j \geq i$, $\lambda[j..\infty] \models \varphi$ (persistence)</td>
</tr>
</tbody>
</table>

Note that:

- $G\varphi \equiv \neg F \neg \varphi$
- $F\varphi \equiv \neg G \neg \varphi$
Semantics of LTL: $F_{pos_1}$

\[ \lambda \]

\[ \text{pos}_0 \rightarrow \text{pos}_1 \rightarrow \text{pos}_2 \rightarrow \text{pos}_0 \rightarrow \text{pos}_1 \rightarrow \text{pos}_2 \rightarrow \cdots \]

\[ \lambda \models F_{pos_1} \]
Semantics of LTL: $F_{\text{pos}_1}$

$\lambda^'$

$\lambda \models F_{\text{pos}_1}$

$\lambda^' = \lambda[1..\infty] \models \text{pos}_1$
Semantics of LTL: $F^{pos_1}$

\[ \lambda'[0] \]

\[
\begin{array}{cccccc}
\text{pos}_0 & \text{pos}_1 & \text{pos}_2 & \text{pos}_0 & \text{pos}_1 & \text{pos}_2 \\
q_0 & q_1 & q_2 & q_0 & q_1 & q_2 & \cdots
\end{array}
\]

\[
\lambda \models F^{pos_1}
\]

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

\[
\text{pos}_1 \in V(\lambda'[0]) = V(q_1)
\]
Semantics of LTL: $GF_{pos_1}$

\[\lambda\]

$pos_0$ $pos_1$ $pos_2$ $pos_0$ $pos_1$ $pos_2$

$q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow \cdots$

$\lambda \models GF_{pos_1}$
Semantics of LTL: $GF_{pos_1}$

$\lambda[0..\infty]$  

$q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow \cdots$

$\lambda \models GF_{pos_1}$

$\lambda[0..\infty] \models F_{pos_1}$
Semantics of LTL: $GF_{pos_1}$

$\lambda[0..\infty]$

\[
\begin{array}{c}
pos_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow \ldots \end{array}
\]

$\lambda \models GF_{pos_1}$

$\lambda[0..\infty] \models F_{pos_1}$
Semantics of LTL: $GF_{pos_1}$

$\lambda[1..\infty]$ 

\begin{align*}
\lambda &\models GF_{pos_1} \\
\lambda[0..\infty] &\models F_{pos_1} \\
\lambda[1..\infty] &\models F_{pos_1}
\end{align*}
Semantics of LTL: $GF_{pos_1}$

$\lambda[1..\infty]$

$\lambda \models GF_{pos_1}$

$\lambda[0..\infty] \models F_{pos_1}$

$\lambda[1..\infty] \models F_{pos_1}$
Semantics of LTL: $GF_{pos_1}$

$$\lambda[2..\infty]$$

$\lambda \models GF_{pos_1}$

$\lambda[0..\infty] \models F_{pos_1}$

$\lambda[1..\infty] \models F_{pos_1}$

$\lambda[2..\infty] \models F_{pos_1}$
Semantics of LTL: $\text{GF}^{\text{pos}_1}$

\[ \lambda [2..\infty] \]

\[ \lambda |\Rightarrow \text{GF}^{\text{pos}_1} \]

\[ \lambda [0..\infty] |\Rightarrow \text{F}^{\text{pos}_1} \]

\[ \lambda [1..\infty] |\Rightarrow \text{F}^{\text{pos}_1} \]

\[ \lambda [2..\infty] |\Rightarrow \text{F}^{\text{pos}_1} \]
Semantics of LTL: $\text{GF}_{\text{pos}_1}$

$$\lambda[2..\infty]$$

$q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow \ldots$

$\lambda \models \text{GF}_{\text{pos}_1}$

$\lambda[0..\infty] \models F_{\text{pos}_1}$

$\lambda[1..\infty] \models F_{\text{pos}_1}$

$\lambda[2..\infty] \models F_{\text{pos}_1}$

$\ldots$
Definition 2.5 (Semantics of LTL in Transition Systems)

\[(M, q) \models \varphi \quad \text{iff} \quad \lambda \models \varphi \quad \text{for every path} \ \lambda \ \text{in} \ M \ \text{starting from} \ q.\]

\[M \models \varphi \quad \text{iff} \quad (M, q_0) \models \varphi \quad \text{for every initial state} \ q_0 \ \text{in} \ M.\]
Example

\[ M \models Ga? \]
Example

\[
M \models Ga
\]
Example

\[ M \models \text{Ga} \]

\[ M \models X(a \land b) ? \]
Example

\[ M \models Ga \]
\[ M \not\models X(a \land b) \]
Example

\begin{align*}
M & \models Ga \\
M & \not\models X(a \land b) \\
M & \models G(\neg b \rightarrow G(a \land \neg b))?
\end{align*}
Example

\[
\begin{align*}
M & \models Ga \\
M & \not\models X(a \land b) \\
M & \models G(\neg b \rightarrow G(a \land \neg b))
\end{align*}
\]
Example

\[
\begin{align*}
M &\models Ga \\
M &\not\models X(a \land b) \\
M &\models G(\neg b \rightarrow G(a \land \neg b)) \\
M &\models bU(a \land \neg b)\
\end{align*}
\]
Example

\[
\begin{align*}
M & \models Ga \\
M & \not\models X(a \land b) \\
M & \models G(\neg b \rightarrow G(a \land \neg b)) \\
M & \models bU(a \land \neg b)
\end{align*}
\]
Specifying Properties in LTL: Mutual Exclusion

\[ M_{Sem} \models G \neg (cr_1 \land cr_2) \]
Specifying Properties in LTL: Mutual Exclusion

\[ M_{Sem} \models G \neg (cr_1 \land cr_2) \]
Specifying Properties in LTL: Mutual Exclusion

\[\langle n_1, n_2, y = 1 \rangle\]

\[\langle w_1, n_2, y = 1 \rangle\]

\[\langle n_1, w_2, y = 1 \rangle\]

\[\langle w_1, w_2, y = 1 \rangle\]

\[\langle c_1, n_2, y = 0 \rangle\]

\[\langle c_1, w_2, y = 0 \rangle\]

\[\langle w_1, c_2, y = 0 \rangle\]

\[\langle n_1, c_2, y = 0 \rangle\]

\[
M_{Sem} \models G \neg (cr_1 \land cr_2)
\]

\[
M_{Sem} \models GF cr_1 \lor GF cr_2
\]
Specifying Properties in LTL: Mutual Exclusion

\[ \langle n_1, n_2, y = 1 \rangle \]
\[ \langle w_1, n_2, y = 1 \rangle \]
\[ \langle n_1, w_2, y = 1 \rangle \]
\[ \langle w_1, w_2, y = 1 \rangle \]
\[ \langle c_1, w_2, y = 0 \rangle \]
\[ \langle c_1, n_2, y = 0 \rangle \]
\[ \langle w_1, c_2, y = 0 \rangle \]
\[ \langle n_1, c_2, y = 0 \rangle \]
\[ \langle c_1, w_2, y = 0 \rangle \]

\[ M_{Sem} \models G \neg (cr_1 \land cr_2) \]
\[ M_{Sem} \models GF cr_1 \lor GF cr_2 \]
Specifying Properties in LTL: Mutual Exclusion

\[ M_{Sem} \models G \neg (cr_1 \land cr_2) \]
\[ M_{Sem} \models GF cr_1 \lor GF cr_2 \]
\[ M_{Sem} \models GF cr_1 \land GF cr_2? \]
Specifying Properties in LTL: Mutual Exclusion

\[ \langle n_1, n_2, y = 1 \rangle \]
\[ \langle w_1, n_2, y = 1 \rangle \]
\[ \langle n_1, w_2, y = 1 \rangle \]
\[ \langle w_1, w_2, y = 1 \rangle \]
\[ \langle c_1, n_2, y = 0 \rangle \]
\[ \langle c_1, w_2, y = 0 \rangle \]
\[ \langle w_1, c_2, y = 0 \rangle \]
\[ \langle n_1, c_2, y = 0 \rangle \]

\[ M_{Sem} \models \neg (cr_1 \land cr_2) \]
\[ M_{Sem} \models GFcr_1 \lor GFcr_2 \]
\[ M_{Sem} \not\models GFcr_1 \land GFcr_2 \]
Specifying Properties in LTL: Mutual Exclusion

\[ \langle n_1, n_2, y = 1 \rangle \]

\[ \langle w_1, n_2, y = 1 \rangle \]

\[ \langle n_1, w_2, y = 1 \rangle \]

\[ \langle w_1, w_2, y = 1 \rangle \]

\[ \langle n_1, c_2, y = 0 \rangle \]

\[ \langle c_1, n_2, y = 0 \rangle \]

\[ \langle c_1, w_2, y = 0 \rangle \]

\[ \langle w_1, c_2, y = 0 \rangle \]

\[ M_{Sem} \models \text{G} \neg (cr_1 \land cr_2) \]

\[ M_{Sem} \models \text{GF} cr_1 \lor \text{GF} cr_2 \]

\[ M_{Sem} \not\models \text{GF} cr_1 \land \text{GF} cr_2 \]

\[ M_{Sem} \models (\text{GF} wa_1 \rightarrow \text{GF} cr_1)? \]
Specifying Properties in LTL: Mutual Exclusion

\[ M_{Sem} \models G \neg (cr_1 \land cr_2) \]
\[ M_{Sem} \models GF cr_1 \lor GF cr_2 \]
\[ M_{Sem} \not\models GF cr_1 \land GF cr_2 \]
\[ M_{Sem} \not\models (GF wa_1 \rightarrow GF cr_1) \]
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 \( \phi \) or \( \neg \phi \) if there are (initial) paths \( \pi_1 \) and \( \pi_2 \) such that \( \pi_1 \models \phi \) and \( \pi_2 \models \neg \phi \)

\[
(M, q_0) \not\models G \text{pos}_0 \\
\text{(but also: } (M, q_0) \not\models \neg G \text{pos}_0 !)\\
(M, q_0) \not\models F \text{pos}_1 \\
\text{(but also: } (M, q_0) \not\models \neg F \text{pos}_1 !)\\
(M, q_0) \models (\neg G \text{pos}_0) \rightarrow F \text{pos}_1
\]
**Equivalences**

**Definition 2.6 (Equivalence)**

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

- **Duality**

  $\neg G \phi \equiv F \neg \phi$

  $\neg F \phi \equiv G \neg \phi$

  $\neg X \phi \equiv X \neg \phi$

- **Idempotency**

  $GG \phi \equiv G \phi$

  $FF \phi \equiv F \phi$

  $\phi U (\phi U \psi) \equiv \phi U \psi$

  $(\phi U \psi) U \psi \equiv \phi U \psi$

- **Absorption**

  $FGF \phi \equiv GF \phi$

  $GFG \phi \equiv FG \phi$

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

---

F. Belardinelli · Logics for Strategic Reasoning in AI

Imperial College London – Spring Term 2019
Equivalences

- Distribution

\[
X(\phi U \psi) \equiv (X\phi) U (X\psi)
\]
\[
F(\phi \lor \psi) \equiv F\phi \lor F\psi
\]
\[
G(\phi \land \psi) \equiv G\phi \land G\psi
\]

- But,

\[
F(\phi U \psi) \not\equiv (F\phi) U (F\psi)
\]
\[
G(\phi U \psi) \not\equiv (G\phi) U (G\psi)
\]
\[
F(\phi \land \psi) \not\equiv F\phi \land F\psi
\]
\[
G(\phi \lor \psi) \not\equiv G\phi \lor G\psi
\]

\[
TS \models Fa \land Fb \text{ but } TS \not\models F(a \land b)
\]
**Equivalences**

- **Expansion**

  \[ \phi U \psi \equiv \psi \lor (\phi \land X(\phi U \psi)) \]
  \[ F \psi \equiv \psi \lor XF \psi \]
  \[ G \psi \equiv \psi \land XG \psi \]

\( \phi U \psi \) is the **least** solution of the expansion law for \( U \):
- if \( \alpha \) is such that \( \alpha \equiv \psi \lor (\phi \land X \alpha) \), then \( \phi U \psi \rightarrow \alpha \)

Similarly, \( G \psi \) is the **greatest** solution of the expansion law for \( G \):
- if \( \alpha \) is such that \( \alpha \equiv \psi \land X \alpha \), then \( \alpha \rightarrow G \psi \) (e.g., \( \alpha = false \))
Motivating Example: Rescue Robots

- Everybody is safe:
  \[ \wedge_{i \in \text{People}} \text{safe}_i \]

- Everybody will eventually be safe:
  \[ \wedge_{i \in \text{People}} F\text{safe}_i \]
  different interpretation: \[ F(\wedge_{i \in \text{People}} \text{safe}_i) \]

- Everybody will always be safe, from some moment on:
  \[ \wedge_{i \in \text{People}} FG\text{safe}_i \]
  equivalently: \[ FG(\wedge_{i \in \text{People}} \text{safe}_i) \]

- 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:
  \[ G(\neg \text{safe}_i \rightarrow F \text{safe}_i) \]

- If person \(i\) gets outside the building, then she will never be in danger anymore:
  \[ (F \text{outside}_i) \rightarrow (FG \text{safe}_i) \quad \sim \text{not quite what we want!} \]
  \[ F(\text{outside}_i \rightarrow G \text{safe}_i) \quad \sim \text{not quite right either} \]

we want to express that \textit{whenever a person gets outside, she will remain safe from then on}:
\[ G(\text{outside}_i \rightarrow G \text{safe}_i) ! \]

- Person \(i\) may be rescued without any robot ever entering the building, but guaranteed rescue requires some robots to enter:
  \textbf{Cannot be expressed in LTL!}
Motivating Example: Voting

- The system will not reveal how a particular voter voted:
  \( G \neg \text{revealed}_i \)

- The system does not issue receipts:
  \( G \neg \text{receipt}_i \)

- 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.

- $s \models G(x \leq 20)$ iff for all paths starting from $s$, always $x \leq 20$
- the universal quantification implicit in the LTL semantics can be made explicit:
  $$s \models A\varphi \iff \pi \models \varphi \text{ for all paths } \pi \text{ starting in } s$$
- but what about “for every computation it is always possible to return to the initial state”?  
- $AGF \text{start}$ would not do the job. Why?

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

- $s \models AG(x \leq 20)$ iff for every path starting from $s$, always $x \leq 20$
- $s \models EG(x \leq 20)$ iff for some path starting from $s$, always $x \leq 20$
- alternation of path quantifiers is allowed: $AGEF \text{start}$
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 $\Phi$, $\Psi$:
  - $a \in AP$ atoms
  - $\neg \Phi$ negation
  - $\Phi \land \Psi$ conjunction
  - $E\phi$ for some path $\phi$ is true
  - $A\phi$ for every path $\phi$ is true

- Path formulas $\phi$:
  - $X\Phi$ $neXt \Phi$
  - $\Phi U \Psi$ $\Phi Until \Psi$

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

$\Rightarrow$ Notice that $X$ and $U$ alternate with $A$ and $E$:
  - $AXAX\Phi$ and $AXEX\Phi \in CTL$
  - but $AXX\Phi$ and $AEX\Phi \notin CTL$
Let $AP = \{x = 1, x < 2, x \geq 3\}$. Then, what about:

$\Diamond \ EX(x = 1)$?
Let $AP = \{ x = 1, x < 2, x \geq 3 \}$. Then, what about:

1. $EX(x = 1)$
   - ✔

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

1. $\text{EX}(x = 1)$ ✓
2. $\text{AX}(x = 1)$ ✓
3. $(x < 2) \lor (x = 1)$ ?
Computation Tree Logic: Syntax

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

1. $\text{EX}(x = 1)$ ✓
2. $\text{AX}(x = 1)$ ✓
3. $(x < 2) \lor (x = 1)$ ✓
4. $\text{E}(x = 1 \land \text{AX}(x \geq 3))$?
Let $AP = \{ x = 1, x < 2, x \geq 3 \}$. Then, what about:

1. $EX(x = 1)$ ✓
2. $AX(x = 1)$ ✓
3. $(x < 2) \lor (x = 1)$ ✓
4. $E(x = 1 \land AX(x \geq 3))$ ×
5. $EX(trueU(x = 1))$?
Computation Tree Logic: Syntax

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

1. $EX(x = 1)$ ✓
2. $AX(x = 1)$ ✓
3. $(x < 2) \lor (x = 1)$ ✓
4. $E(x = 1 \land AX(x \geq 3))$ ✗
5. $EX(trueU(x = 1))$ ✗
6. $EX(x = 1 \land AX(x \geq 3))$?
Computation Tree Logic: Syntax

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

1. $EX(x = 1)$ ✓
2. $AX(x = 1)$ ✓
3. $(x < 2) \lor (x = 1)$ ✓
4. $E(x = 1 \land AX(x \geq 3))$ ✗
5. $EX(trueU(x = 1))$ ✗
6. $EX(x = 1 \land AX(x \geq 3))$ ✓
7. $EXA(trueU(x = 1))$?
Computation Tree Logic: Syntax

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

1. \( \text{EX}(x = 1) \) ✓
2. \( \text{AX}(x = 1) \) ✓
3. \( (x < 2) \lor (x = 1) \) ✓
4. \( \text{E}(x = 1 \land \text{AX}(x \geq 3)) \) ✗
5. \( \text{EX}(true \text{U}(x = 1)) \) ✗
6. \( \text{EX}(x = 1 \land \text{AX}(x \geq 3)) \) ✓
7. \( \text{EXA}(true \text{U}(x = 1)) \) ✓
Derived Operators

Eventually:

\[
\begin{align*}
EF\Phi & = \ E(trueU\Phi) \quad \text{potentially } \Phi \\
AF\Phi & = \ A(trueU\Phi) \quad \text{inevitably } \Phi
\end{align*}
\]

Always:

\[
\begin{align*}
EG\Phi & = \neg AF\neg \Phi \quad \text{potentially always } \Phi \\
AG\Phi & = \neg EF\neg \Phi \quad \text{invariantly } \Phi
\end{align*}
\]

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

\[
\Phi \ ::= \ a \ | \ \neg \Phi \ | \ \Phi \land \Phi \ | \ EX\Phi \ | \ AX\Phi \ | \ E(\Phi U\Phi) \ | \ A(\Phi U\Phi)
\]
Examples: Mutual Exclusion Problem

1. mutual exclusion: $AG(\neg crit_1 \lor \neg crit_2)$
2. starvation freedom (strong): $(AGAF crit_1) \land (AGAF crit_2)$
3. “every request will eventually be granted”: $AG(request \rightarrow AF granted)$
4. “in every state it is possible to return to (one of) the initial state(s)”: $AGEF start$
Let $M = \langle St, \rightarrow, \mathcal{V} \rangle$ be a transition system, $\Phi, \Psi$ be state formulas, and $\gamma$ 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 $(M, q) \not\models \Phi$
- $(M, q) \models \Phi \land \Psi$ iff $(M, q) \models \Phi$ and $(M, q) \models \Psi$
- $(M, q) \models E\gamma$ iff for some path $\lambda$ starting from $q$, $(M, \lambda) \models \gamma$
- $(M, q) \models A\gamma$ iff for all paths $\lambda$ starting from $q$, $(M, \lambda) \models \gamma$

**Definition 2.8 (Semantics of CTL: path formulas)**
Let $M = \langle St, \rightarrow, \mathcal{V} \rangle$ be a transition system, $\Phi, \Psi$ be state formulas, and $\gamma$ 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 $(M, q) \not\models \Phi$
- $(M, q) \models \Phi \land \Psi$ iff $(M, q) \models \Phi$ and $(M, q) \models \Psi$
- $(M, q) \models E\gamma$ iff for some path $\lambda$ starting from $q$, $(M, \lambda) \models \gamma$
- $(M, q) \models A\gamma$ iff for all paths $\lambda$ starting from $q$, $(M, \lambda) \models \gamma$

**Definition 2.8 (Semantics of CTL: path formulas)**

Like in LTL!
Computation Tree Logic: Semantics

Let $M = \langle St, \rightarrow, \mathcal{V} \rangle$ be a transition system, $\Phi, \Psi$ be state formulas, and $\gamma$ be a path formula.

Definition 2.7 (Semantics of CTL: state formulas)

$$(M, q) \models a \quad \text{iff} \quad a \in \mathcal{V}(q)$$

$$(M, q) \models \neg \Phi \quad \text{iff} \quad (M, q) \not\models \Phi$$

$$(M, q) \models \Phi \land \Psi \quad \text{iff} \quad (M, q) \models \Phi \text{ and } (M, q) \models \Psi$$

$$(M, q) \models E\gamma \quad \text{iff} \quad \text{for some path } \lambda \text{ starting from } q, \ (M, \lambda) \models \gamma$$

$$(M, q) \models A\gamma \quad \text{iff} \quad \text{for all paths } \lambda \text{ starting from } q, \ (M, \lambda) \models \gamma$$

Definition 2.8 (Semantics of CTL: path formulas)

$$(M, \lambda) \models X\Phi \quad \text{iff} \quad (M, \lambda[1]) \models \Phi$$

$$(M, \lambda) \models \Phi U \Psi \quad \text{iff} \quad \text{for some } i \geq 0,$$

and $(M, \lambda[j]) \models \Phi \text{ for all } 0 \leq j < i$
Semantics of CTL: Intuition

(d) $AX\varphi$

(e) $A\varphi U\psi$

(f) $E\varphi U\psi$
Semantics of CTL: Derived Operators

Recall that

\[
\begin{align*}
\text{EF} \Phi &= \mathcal{E}(true \mathcal{U} \Phi) \\
\text{AF} \Phi &= \mathcal{A}(true \mathcal{U} \Phi) \\
\text{EG} \Phi &= \neg \text{AF} \neg \Phi \\
\text{AG} \Phi &= \neg \text{EF} \neg \Phi
\end{align*}
\]

Then, by definition:

\[
\begin{align*}
(M, q) \models \text{EF} \Phi & \iff \text{for some path } \lambda \text{ from } q, \text{ for some } j \geq 0, (M, \lambda[j]) \models \Phi \\
(M, q) \models \text{AF} \Phi & \iff \text{for every path } \lambda \text{ from } q, \text{ for some } j \geq 0, (M, \lambda[j]) \models \Phi \\
(M, q) \models \text{EG} \Phi & \iff \text{for some path } \lambda \text{ from } q, \text{ for all } j \geq 0, (M, \lambda[j]) \models \Phi \\
(M, q) \models \text{AG} \Phi & \iff \text{for every path } \lambda \text{ from } q, \text{ for all } j \geq 0, (M, \lambda[j]) \models \Phi
\end{align*}
\]
Example: Rocket and Cargo

3 \models EF\text{caP}?
Example: Rocket and Cargo

3 |= EF caP?
Example: Rocket and Cargo

3 \models EF\text{caP}?
Example: Rocket and Cargo

\[ 3 \models EF\text{caP?} \]
Example: Rocket and Cargo

3 \models EF \neg caP?
Example: Rocket and Cargo

\[ 3 \models EF\text{cap}? \]
Example: Rocket and Cargo

$3 \models EF \text{caP}$?
Example: Rocket and Cargo

\[ 3 \models EF caP \checkmark \]
Example: Rocket and Cargo

3 \models EF \text{caP} \checkmark
Semantics of Transition Systems

- For state formula $\Phi$, the satisfaction set $Sat(\Phi)$ is defined as
  
  $$Sat(\Phi) = \{ s \in S \mid s \models \Phi\}$$

- $TS$ satisfies formula $\Phi$ iff it is satisfied in all initial states:
  
  $$TS \models \Phi \iff \text{for every } s_0 \in I, s_0 \models \Phi$$

  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, $s_0 \models EG\Phi$ and $s'_0 \not\models EG\Phi$

- A formula $\Phi$ is **valid** iff it is true in every transition system.
Rocket and Cargo: More Properties

\[\text{rocket}\] \text{and} \text{cargo: more properties}\]
Rocket and Cargo: More Properties

\[ \text{roL} \rightarrow \text{EF roP?} \]
Rocket and Cargo: More Properties

roL → EF roP ✓
Rocket and Cargo: More Properties

\[ \text{roL} \rightarrow \text{EF} \text{roP} \checkmark \]

AG(\text{roL} \lor \text{roP})?
Rocket and Cargo: More Properties

roL → EF roP ✓

AG (roL ∨ roP) ✓
Rocket and Cargo: More Properties

\[ \text{roL} \rightarrow \text{EF roP} \checkmark \]

\[ \text{AG}(\text{roL} \lor \text{roP}) \checkmark \]

\[ \text{roL} \rightarrow \text{AX}(\text{roP} \rightarrow \text{nofuel})? \]
Rocket and Cargo: More Properties

\[ \text{roL} \rightarrow \text{EF}\text{roP} \checkmark \]

\[ \text{AG}(\text{roL} \lor \text{roP}) \checkmark \]

\[ \text{roL} \rightarrow \text{AX}(\text{roP} \rightarrow \text{nofuel}) \checkmark \]
Example: Mutual Exclusion Problem

\[
\begin{align*}
M & \models AG(\neg cr_1 \lor \neg cr_2)
\end{align*}
\]
Example: Mutual Exclusion Problem

\[ M \models AG(\neg cr_1 \lor \neg cr_2) \checkmark \]
Example: Mutual Exclusion Problem

\[ M \models AG(\neg cr_1 \lor \neg cr_2) \checkmark \]

\[ M \not\models (AG AF cr_1) \land (AG AF cr_2) \]

\[ 1 \]

\[ 2 \]
Example: Mutual Exclusion Problem

1. $M \models AG(\neg cr_1 \lor \neg cr_2)$ ✓
2. $M \not\models (AG AF cr_1) \land (AG AF cr_2)$ ✗
Example: Mutual Exclusion Problem

1. $M \models AG(\neg cr_1 \lor \neg cr_2)$ ✓
2. $M \not\models (AGAF cr_1) \land (AGAF cr_2)$ ✗
3. $M \models (AGAF wa_1 \rightarrow AGAF cr_1) \land (AGAF wa_2 \rightarrow AGAF cr_2)$ ？
Example: Mutual Exclusion Problem

1. $M \models AG(\neg cr_1 \lor \neg cr_2)$ ✓
2. $M \not\models (AGAF cr_1) \land (AGAF cr_2)$ ✗
3. $M \not\models (AGAF wa_1 \rightarrow AGAF cr_1) \land (AGAF wa_2 \rightarrow AGAF cr_2)$ ✗
Definition 2.9 (Equivalence)

Two formulas $\Phi$ and $\Psi$ are equivalent, or $\Phi \equiv \Psi$, iff $\text{Sat}(\Phi) = \text{Sat}(\Psi)$ for all TS.

$\Phi \equiv \Psi$ if and only if for all $TS$, $TS \models \Phi \iff TS \models \Psi$

Duality:

- $\text{AX} \Phi \equiv \neg \text{EX} \neg \Phi$
- $\text{EX} \Phi \equiv \neg \text{AX} \neg \Phi$
- $\text{AF} \Phi \equiv \neg \text{EG} \neg \Phi$
- $\text{EF} \Phi \equiv \neg \text{AG} \neg \Phi$
- $\text{AG} \Phi \equiv \neg \text{EF} \neg \Phi$
- $\text{EG} \Phi \equiv \neg \text{AF} \neg \Phi$
Equivalences in CTL: Distributivity

- in LTL we have:

\[ F(\phi \lor \psi) \equiv F\phi \lor F\psi \]
\[ G(\phi \land \psi) \equiv G\phi \land G\psi \]

- and in CTL:

\[ EF(\Phi \lor \Psi) \equiv EF\Phi \lor EF\Psi \]
\[ AG(\Phi \land \Psi) \equiv AG\Phi \land AG\Psi \]
But,

\[
EG(\Phi \lor \Psi) \not\equiv EG\Phi \lor EG\Psi \\
AF(\Phi \land \Psi) \not\equiv AF\Phi \land AF\Psi
\]

\[
\begin{array}{c}
s_1 \quad s \quad s_2 \\
\{a\} \quad \emptyset \quad \{b\}
\end{array}
\]

- \(s \models AF(a \lor b)\) as for every path \(\pi\) from \(s\), \(\pi \models F(a \lor b)\)
- but \(s(s_1)^\omega \not\models Fb\). Thus, \(s \not\models AFb\)
- a similar line of reasoning shows that \(s \not\models AFa\).
- hence, \(s \not\models AFa \lor AFb\).
Theorem 2.10 (Fixpoint characterization of branching-time operators)

The following formulas are valid in CTL:

- \( E\varphi_1 U \varphi_2 \leftrightarrow \varphi_2 \lor (\varphi_1 \land EX E\varphi_1 U \varphi_2) \)
- \( EF \varphi \leftrightarrow \varphi \lor EX EF \varphi \)
- \( EG \varphi \leftrightarrow \varphi \land EX EG \varphi \)

- \( A\varphi_1 U \varphi_2 \leftrightarrow \varphi_2 \lor (\varphi_1 \land AX A\varphi_1 U \varphi_2) \).
- \( AF \varphi \leftrightarrow \varphi \lor AX AF \varphi \)
- \( AG \varphi \leftrightarrow \varphi \land AX AG \varphi \)

- \( E\varphi_1 U \varphi_2 \) 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: 
  \[ \bigwedge_{i \in \text{People}} \text{safe}_i \]

- Everybody will eventually be safe: 
  \[ \bigwedge_{i \in \text{People}} \text{AF}\text{safe}_i \]
  Another interpretation: \[ \text{AF}\left( \bigwedge_{i \in \text{People}} \text{safe}_i \right) \]

- Everybody will always be safe, from some moment on: 
  **Cannot be expressed in CTL! but in CTL*:** 
  \[ \bigwedge_{i \in \text{People}} \text{AFG}\text{safe}_i \]
  Equivalently: \[ \text{AFG}\left( \bigwedge_{i \in \text{People}} \text{safe}_i \right) \]

- Everybody may eventually be safe, if everything goes fine: 
  \[ \bigwedge_{i \in \text{People}} \text{EF}\text{safe}_i \]
  Another interpretation: \[ \text{EF}\left( \bigwedge_{i \in \text{People}} \text{safe}_i \right) \]
Motivating Example: Rescue Robots

- Whenever person $i$ gets in trouble, she will eventually be rescued:
  \[ AG(\neg \text{safe}_i \rightarrow AF \text{safe}_i) \]

- If person $i$ gets outside the building then she will never be in danger anymore:
  \[ AG(\text{outside}_i \rightarrow AG \text{safe}_i) \]

- Person $i$ may be rescued without any robot ever entering the building, but guaranteed rescue requires some robots to enter:
  \[ E(\bigwedge_{j \in \text{Robots}} \text{outside}_j) U \text{safe}_i \land \neg A(\bigwedge_{j \in \text{Robots}} \text{outside}_j) U \text{safe}_i \]
Motivating Example: Voting

- The system will not reveal how a particular voter voted:
  \[ \text{AG} \neg \text{revealed}_i \]

- The system does not issue receipts:
  \[ \text{AG} \neg \text{receipt}_i \]

- The voter can vote, and can refrain from voting \( \rightsquigarrow \text{ambiguous} \ldots \)
  
  **Interpretation 1:** The voter may vote, and may refrain from voting:
  \[ \text{EF voted}_i \land \text{EG} \neg \text{voted}_i \]

  **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:
  \[ \bigwedge_{c \in \text{Candidates}} \text{AG} \left( \text{voted}_{i,c} \rightarrow \text{AG} \left( \neg \text{revealedVote}_{i,c} \right) \right) \]
Equivalence of Formulas in LTL and CTL

**Definition 2.11 (Equivalence)**

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

$$TS \models \Phi \iff TS \models \phi$$

**Theorem 2.12 (Clarke & Draghicescu, 1988)**

Let $\Phi$ be a CTL formula and let $\phi$ be the LTL formula obtained by deleting all path quantifiers in $\Phi$. Then,

$$\Phi \equiv \phi \text{ or there is no LTL formula equivalent to } \Phi$$
LTL and CTL are incomparable

- Some formulas in LTL cannot be expressed in CTL:
  - $FGa$
  - $F(a \land Xa)$

- Some formulas in CTL cannot be expressed in LTL:
  - $AFAGa$
  - $AF(a \land AXa)$
  - $AGEFa$

$\Rightarrow$ cannot be expressed $=$ there is no equivalent formula
Comparison between LTL and CTL

\[ F(a \land Xa) \text{ is not equivalent to } AF(a \land AXa) \]

\[
\begin{align*}
s_0 \models F(a \land Xa) & \quad \text{but} \quad s_0 \not\models \underbrace{AF(a \land AXa)}_{\text{consider path } s_0 s_1 (s_2)\omega} \\
\Rightarrow \text{There is no LTL formula equivalent to } AF(a \land AXa) \\
\text{Actually, there is no CTL formula equivalent to } F(a \land Xa).
\end{align*}
\]
Comparison between LTL and CTL

\[ \text{AFAG}_a \text{ is not equivalent to } \text{FG}_a \]

\[
\begin{array}{c}
\text{s}_0 \\
\{a\}
\end{array} \quad \rightarrow \quad 
\begin{array}{c}
\text{s}_1 \\
\emptyset
\end{array} \quad \rightarrow \quad 
\begin{array}{c}
\text{s}_2 \\
\{a\}
\end{array}
\]

\[
s_0 \models \text{FG}_a \quad \text{but} \quad s_0 \nmid \text{AFAG}_a
\]

consider path \((s_0)^\omega\)

⇒ Again, there is no LTL formula equivalent to AFAG\(_a\)

Actually, there is no CTL formula equivalent to FG\(_a\).
Comparison between LTL and CTL

Formula $AGEFa$ cannot be expressed in LTL

- proof by contradiction: suppose that for some $\phi$ in LTL, $\phi \equiv AGEFa$.
- consider

\[ s_0 \xrightarrow{} s_1 \]

\[ s_0 \xrightarrow{} s \]

\[ \emptyset \xrightarrow{} \{a\} \]

\[ \emptyset \]

\( g \) TS

\( h \) TS'

- $TS \models AGEFa$. Hence, by assumption, $TS \models \phi$
- $Paths(TS') \subseteq Paths(TS)$. Thus, $TS' \models \phi$
- but, $TS' \not\models AGEFa$ as $s^\omega \not\models GEFa$
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

Temporal and modal logic.

*Temporal Logics*.
Kluwer.

The complexity of temporal model checking.
*Advances in Modal Logics*, World Scientific.
Part 3: Verification and Complexity

Verification and Complexity

3.1 Decision Problems
3.2 Complexity of Model Checking Temporal Logics
3.1 Decision Problems
# What's the Use of MAS Logics?

<table>
<thead>
<tr>
<th>Modelling &amp; Design</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modeling systems (the frameworks provide intuitive conceptual structures, and a systematic approach);</td>
</tr>
<tr>
<td>Specifying desirable properties of systems.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Analysis &amp; Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reasoning about concrete systems;</td>
</tr>
<tr>
<td>Correctness testing.</td>
</tr>
</tbody>
</table>
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 $\varphi$, determine if $\varphi$ is valid (true in every model)
  - **Satisfiability**: given formula $\varphi$, determine if $\varphi$ is satisfiable (true in some model)
  - **Model checking**: given formula $\varphi$ and model $M$, determine if $\varphi$ is true in $M$
Motivating Example: Rescue Robots

- Check whether everybody will eventually be safe.
  
  Model checking of $\bigwedge_{i \in \text{People}} A F \text{safe}_i$
  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 $A G (\text{outside}_i \rightarrow A G \text{safe}_i)$
  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 $\bigwedge_{i \in \text{People}} F G \text{safe}_i$

- Show or disprove that everybody may eventually be safe, if everything goes fine.
  
  Validity checking of $\bigwedge_{i \in \text{People}} E F \text{safe}_i$
Motivating Example: Voting

- Verify that the voter can vote, and can refrain from voting:
  
  **Model checking** of $\text{EF} \text{voted}_i \land \text{EG} \neg \text{voted}_i$
  
in the model of the voting protocol

- Design a system that does not issue receipts:
  
  **Satisfiability checking** of $\text{AG}(\bigwedge_{c \in \text{Voters}} \neg \text{receipt}_i)$

- Show (or disprove) that no system will ever reveal how any voter $i$ voted:
  
  **Validity checking** of $\text{AG}(\bigwedge_{i \in \text{Voters}} \neg \text{revealed}_i)$
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

$\sim$ **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 ↪ 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 \textit{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 \textit{linear} mean precisely?
- And how do we measure the size of the model?

**Theorem 3.2**

Model checking of CTL is \textit{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.
Theorem 3.3

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

Theorem 3.4

Model checking of LTL is \textit{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

<table>
<thead>
<tr>
<th></th>
<th>Model Checking</th>
<th>Satisfiability</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTL</td>
<td>PTIME-complete</td>
<td>EXPTIME-complete</td>
</tr>
<tr>
<td>LTL</td>
<td>PSPACE-complete</td>
<td>PSPACE-complete</td>
</tr>
<tr>
<td>CTL*</td>
<td>PSPACE-complete</td>
<td>2EXPTIME-complete</td>
</tr>
</tbody>
</table>

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:

References


Part 4: Reasoning about Strategic Abilities

Reasoning about Strategic Abilities

4.1 Concurrent Game Structures
4.2 Alternating-Time Temporal Logic
4.3 Agents, Systems, Games
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.
  - 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

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

**Definition 4.1 (Concurrent Game Structure)**

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

- $\text{Agt} = \{1, \ldots, k\}$ is a finite set of **agents**
- $St = \{q_1, q_2, \ldots\}$ is a set of **states**
- $V: AP \to 2^{St}$ is a **valuation** of atoms
- $Act = \{\alpha_1, \ldots, \alpha_m\}$ is a finite set of **actions**
- protocol $d: \text{Agt} \times St \to 2^{Act}$ defines actions **available** to an agent in a state
- $o: St \times Act^{\text{Agt}} \to St$ 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 $\alpha_a$ are **enabled** in $q$, that is, $\alpha_a \in d(a, q)$ for all agents $a \in \text{Agt}$. 
Consider CGS $M = \langle \text{Agt}, \text{St}, \mathcal{V}, \text{Act}, d, o \rangle$, where:

- $\text{Agt} = \{1, 2\}$
- $\text{St} = \{q_0, q_1, q_2\}$
- for $i \leq 2$, $\mathcal{V}(\text{pos}_i) = \{q_i\}$
- $\text{Act} = \{\text{push, wait}\}$ and for every agent $a$ and state $q$, $d(a, q) = \text{Act}$
- $o(q_i, \text{push}, \text{push}) = o(q_i, \text{wait}, \text{wait}) = q_i,
  o(q_i, \text{push}, \text{wait}) = q_{i+1} \pmod{3},
  o(q_i, \text{wait}, \text{push}) = q_{i-1} \pmod{3}$. 
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

\[ \langle A \rangle \Phi: \text{coalition } A \text{ has a joint strategy to achieve goal } \Phi \text{ (independently of whatever agents in } \text{Ag}_\text{t} \setminus A \text{ do.)} \]

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

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

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

  “James Bond and his girlfriend are able to save the world from destruction until the end of the movie”
Definition 4.2 (ATL*)

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

$$
\begin{align*}
\Phi & ::= p | \neg \Phi | \Phi \land \Phi | \langle A \rangle \gamma \\
\gamma & ::= \Phi | \neg \gamma | \gamma \land \gamma | X \gamma | \gamma U \gamma
\end{align*}
$$

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

$\langle A \rangle \gamma$: “the agents in coalition $A$ have a strategy to achieve $\gamma$” (no matter what the agents in $\overline{A} = Agt \setminus A$ do).

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

- $F \gamma \equiv true U \gamma$
- $G \gamma \equiv \neg F \neg \gamma$

$[A] \gamma \equiv \neg \langle A \rangle \neg \gamma$: no matter what the agents in coalition $A$ do, the outcome $\gamma$ 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 $\gamma$ as follows, where $\Phi$ is a state formula:

$$
\gamma ::= X\Phi | \Phi U \Phi
$$

Temporal operators apply to cooperation formulas only.

This is equivalent to the following syntax:

$$
\Phi ::= p | \neg \Phi | \Phi \wedge \Phi | \langle A \rangle X \Phi | [A] X \Phi | \langle A \rangle (\Phi U \Phi) | [A] (\Phi U \Phi)
$$
Let \( AP = \{ x = 1, x < 2, x \geq 3 \} \) and \( \text{Agt} = \{1, 2, 3\} \). Then, what about: 

1. \( \langle 1, 2 \rangle X (x = 1) \)?
Let $AP = \{x = 1, x < 2, x \geq 3\}$ and $\text{Agt} = \{1, 2, 3\}$. Then, what about:

1. $\langle 1, 2 \rangle X (x = 1)$

2. $[2, 3] X (x = 1)$?
Syntax: Examples

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

1. $\langle 1, 2 \rangle X(x = 1)$ ✓
2. $[2, 3] X(x = 1)$ ✓
3. $\langle 1, 2 \rangle (F(x < 2) \lor G(x = 1))$ ?
Syntax: Examples

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

1. $\langle 1, 2 \rangle X (x = 1)$ ✓
2. $[2, 3] X (x = 1)$ ✓
3. $\langle 1, 2 \rangle (F(x < 2) \lor G(x = 1))$ ✓
4. $(x = 1 \land [1, 2] U (x \geq 3))$?
Let $AP = \{x = 1, x < 2, x \geq 3\}$ and $Agt = \{1, 2, 3\}$. Then, what about:

1. $\langle 1, 2 \rangle X(x = 1)$ ✓
2. $[2, 3] X(x = 1)$ ✓
3. $\langle 1, 2 \rangle (F(x < 2) \lor G(x = 1))$ ✓
4. $(x = 1 \land [1, 2] U(x \geq 3))$ ✗
5. $\langle \emptyset \rangle X(true U(x = 1))$?
Let $AP = \{x = 1, x < 2, x \geq 3\}$ and $\text{Agt} = \{1, 2, 3\}$. Then, what about:

1. $\langle 1, 2 \rangle X(x = 1)$
2. $[2, 3]X(x = 1)$
3. $\langle 1, 2 \rangle (F(x < 2) \lor G(x = 1))$
4. $(x = 1 \land [1, 2]U(x \geq 3))$
5. $\langle 0 \rangle X(\text{true}U(x = 1))$
6. $[1, 2]X(x = 1 \land \langle 2, 3 \rangle GX(x \geq 3))$
Syntax: Examples

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

1. $\langle 1, 2 \rangle X(x = 1)$ ✓
2. $[2, 3] X(x = 1)$ ✓
3. $\langle 1, 2 \rangle (F(x < 2) \lor G(x = 1))$ ✓
4. $(x = 1 \land [1, 2] U(x \geq 3)) \times$
5. $\langle \emptyset \rangle X(true U(x = 1))$ ✓
6. $[1, 2] X(x = 1 \land \langle 2, 3 \rangle GX(x \geq 3))$ ✓
7. $\langle 2, 4 \rangle X[2, 3](true U(x = 1))$?
Syntax: Examples

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

1. $\langle 1, 2 \rangle X (x = 1)$ ✓
2. $[2, 3] X (x = 1)$ ✓
3. $\langle 1, 2 \rangle (F (x < 2) \lor G (x = 1))$ ✓
4. $(x = 1 \land [1, 2] U (x \geq 3))$ ×
5. $\langle \emptyset \rangle X (true U (x = 1))$ ✓
6. $[1, 2] X (x = 1 \land \langle 2, 3 \rangle GX (x \geq 3))$ ✓
7. $\langle 2, 4 \rangle X [2, 3] (true U (x = 1))$ ×
Strategies

How to interpret strategic operators?

**Definition 4.3 (Strategy)**

A **strategy** is a conditional plan.
We represent strategies by functions $s_a : St^+ \rightarrow Act$ such that

$s_a(q_1, \ldots, q_n) \in d(a, q_n)$.

→ memory-based (perfect recall) strategies

Particular case: **memoryless (positional) strategies** $s_a : St \rightarrow Act$ 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.

→ 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)**

\( \text{out}(q, s_A) \) 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 \( \text{out}(q, s_A) \) is given as follows:

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

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 \( a \in A \text{ gt} \) (all \( \alpha_a^i \) 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:

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

\[
\text{out}(q_0, s''_1) = \{ \lambda \in \{q_0, q_1, q_2\}^\omega \mid \lambda[0] = q_0 \text{ and for all } i \geq 0, \\
\text{if } |\lambda[0..i]| = 2m \text{ and } \lambda[i] = q_0 \text{ then } \lambda[i + 1] = q_0 \text{ or } \lambda[i + 1] = q_1, \\
\text{else if } \lambda[i] = q_j \text{ then } \lambda[i + 1] = q_j \text{ or } \lambda[i + 1] = q_{(j-1) \mod 3} \}
\]
Formulas in ATL* are interpreted on Concurrent Game Structures.

Definition 4.5 (Semantics of ATL*: state formulas)

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

Definition 4.6 (Semantics of ATL*: path formulas)

\[(M, \lambda) \models X \gamma \iff (M, \lambda[1..\infty]) \models \gamma\]
\[(M, \lambda) \models \gamma_1 U \gamma_2 \iff (M, \lambda[i..\infty]) \models \gamma_2 \text{ for some } i \geq 0, \text{ and } (M, \lambda[j..\infty]) \models \gamma_1 \text{ for all } 0 \leq j \leq i\]
Formulas in ATL* are interpreted on Concurrent Game Structures.

**Definition 4.5 (Semantics of ATL*: state formulas)**

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

**Definition 4.6 (Semantics of ATL*: path formulas)**

\[(M, \lambda) \models X\gamma \quad \text{iff} \quad (M, \lambda[1..\infty]) \models \gamma\]
\[(M, \lambda) \models \Phi_1 \mathcal{U} \Phi_2 \quad \text{iff} \quad (M, \lambda[i..\infty]) \models \gamma_2 \text{ for some } i \geq 0, \text{ and } (M, \lambda[j..\infty]) \models \gamma_1 \text{ for all } 0 \leq j \leq i\]
Semantics of ATL*

Formulas in ATL* are interpreted on Concurrent Game Structures.

**Definition 4.5 (Semantics of ATL*: state formulas)**

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

**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)

\((M, q) \models p\) iff \(q\) is in \(V(p)\)

\((M, q) \models \neg \Phi\) 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 \(\Phi\)

\((M, \lambda) \models \neg \gamma\) iff \((M, \lambda) \not\models \gamma\)

\((M, \lambda) \models \gamma_1 \land \gamma_2\) iff \((M, \lambda) \models \gamma_1\) and \((M, \lambda) \models \gamma_2\)

\((M, \lambda) \models X \gamma\) iff \((M, \lambda[1..\infty]) \models \gamma\)

\((M, \lambda) \models \gamma_1 U \gamma_2\) iff \((M, \lambda[i..\infty]) \models \gamma_2\) for some \(i \geq 0\), and \((M, \lambda[j..\infty]) \models \gamma_1\) for all \(0 \leq j \leq i\).
Recall: $[A] \Phi \equiv \neg \langle A \rangle \neg \Phi$.

**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 $(M, \lambda) \models \Phi$

**Semantics of ATL*: path formulas**

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

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

As usual, $F \gamma \equiv true U \gamma$ and $G \gamma \equiv \neg F \neg \gamma$. 
State-Based Semantics for ATL

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

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

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

\[(M, q_0) \models \neg \langle 1 \rangle X pos_0 \land \neg \langle 2 \rangle X pos_0\]

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

\[(M, q_i) \models \langle 1, 2 \rangle F pos_0 \land \langle 1, 2 \rangle F pos_1 \land \langle 1, 2 \rangle F pos_2\]

\[(M, q_i) \models \langle 1, 2 \rangle (F pos_0 \land F pos_1 \land F pos_2)\]
Example: Robots and Carriage

- no agent can enforce the carriage to move to any particular position:
  
  \[(M, q_0) \models \neg \langle 1 \rangle X \text{pos}_0 \land \neg \langle 2 \rangle X \text{pos}_0\]

- the robots together can move the carriage to any position they like:
  
  \[(M, q_i) \models \langle 1, 2 \rangle F \text{pos}_0 \land \langle 1, 2 \rangle F \text{pos}_1 \land \langle 1, 2 \rangle F \text{pos}_2\]
  \[(M, q_i) \models \langle 1, 2 \rangle (F \text{pos}_0 \land F \text{pos}_1 \land F \text{pos}_2)\]
Example: Robots and Carriage

- no agent can enforce the carriage to move to any particular position:
  \((M, q_0) \models \neg \langle 1 \rangle X \text{pos}_0 \land \neg \langle 2 \rangle X \text{pos}_0\)

- the robots together can move the carriage to any position they like:
  \((M, q_i) \models \langle 1, 2 \rangle F \text{pos}_0 \land \langle 1, 2 \rangle F \text{pos}_1 \land \langle 1, 2 \rangle F \text{pos}_2\)

\((M, q_0) \models \langle 1 \rangle G \neg \text{pos}_1?\)
Example: Robots and Carriage

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

  \[(M, q_0) \models \neg(1)X pos_0 \land \neg(2)X pos_0\]

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

  \[
  (M, q_i) \models (1, 2)F pos_0 \land (1, 2)F pos_1 \land (1, 2)F pos_2
  \]
  \[
  (M, q_i) \models (1, 2)(F pos_0 \land F pos_1 \land F pos_2)
  \]
Example: Robots and Carriage

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

\[(M, q_0) \models \neg \langle 1 \rangle X pos_0 \land \neg \langle 2 \rangle X pos_0\]

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

\[
(M, q_i) \models \langle 1, 2 \rangle F pos_0 \land \langle 1, 2 \rangle F pos_1 \land \langle 1, 2 \rangle F pos_2
\]

\[
(M, q_i) \models \langle 1, 2 \rangle (F pos_0 \land F pos_1 \land F pos_2)
\]
Example: Robots and Carriage

- no agent can enforce the carriage to move to any particular position:
  \[(M, q_0) \models \neg \langle 1 \rangle X \text{pos}_0 \land \neg \langle 2 \rangle X \text{pos}_0\]

- the robots together can move the carriage to any position they like:
  \[
  (M, q_i) \models \langle 1, 2 \rangle F \text{pos}_0 \land \langle 1, 2 \rangle F \text{pos}_1 \land \langle 1, 2 \rangle F \text{pos}_2
  
  (M, q_i) \models \langle 1, 2 \rangle (F \text{pos}_0 \land F \text{pos}_1 \land F \text{pos}_2)
  \]
Example: Robots and Carriage

- no agent can enforce the carriage to move to any particular position:
  \[(M, q_0) \models \langle\langle 1\rangle\rangle \neg G \neg pos_1\]

- the robots together can move the carriage to any position they like:
  \[(M, q_i) \models \langle\langle 1, 2\rangle\rangle F pos_0 \land \langle\langle 1, 2\rangle\rangle F pos_1 \land \langle\langle 1, 2\rangle\rangle F pos_2\]
Example: Robots and Carriage

- no agent can enforce the carriage to move to any particular position:
  \[(M, q_0) \models \langle 1 \rangle \neg \Diamond pos_1 \land \langle 2 \rangle \neg \Diamond pos_0\]

- the robots together can move the carriage to any position they like:
  \[(M, q_i) \models \langle 1, 2 \rangle F pos_0 \land \langle 1, 2 \rangle F pos_1 \land \langle 1, 2 \rangle F pos_2\]
  \[(M, q_i) \models \langle 1, 2 \rangle (F pos_0 \land F pos_1 \land F pos_2)\]
Example: Robots and Carriage

- no agent can enforce the carriage to move to any particular position:
  \[(M, q_0) \models \neg \langle 1 \rangle X pos_0 \land \neg \langle 2 \rangle X pos_0\]

- the robots together can move the carriage to any position they like:
  \[(M, q_i) \models \langle 1, 2 \rangle F pos_0 \land \langle 1, 2 \rangle F pos_1 \land \langle 1, 2 \rangle F pos_2\]
  \[(M, q_i) \models \langle 1, 2 \rangle (F pos_0 \land F pos_1 \land F pos_2)\]
Part 4: Reasoning about Strategic Abilities

4.3 Agents, Systems, Games
Strategy operators allow a number of useful concepts to be formally specified:

- **Safety properties**: $\langle os \rangle G \neg \text{crash}$
- **Liveness properties**: $\langle alice, bob \rangle F \text{paperAccepted}$
- **Fairness properties**: $\langle prod, dlr \rangle G (\text{carRequested} \rightarrow F \text{carDelivered})$

(Note: this is an ATL* formula!)
Connection to Multi-Agent/Multi-Process Systems

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

ATL is just another specification language in this context...
Connection to Games

- Concurrent game structure = generalized extensive game

- $\langle A \rangle \gamma$: operator $\langle A \rangle$ splits the agents into proponents and opponents

- formula $\gamma$ defines the winning condition
  \(\sim\) infinite 2-player, binary, zero-sum game

- Flexible and compact specification of winning conditions

- Solving a game $\approx$ checking if \((M, q) \models \langle A \rangle \gamma\)

- Model checking ATL corresponds to game solving in game theory!
Connection to Games

What about other problems?

- **Validity** ⇔ General properties of games
- **Satisfiability** ⇔ Game design
  - e.g., building a model for $\langle\emptyset\rangle\gamma_1 \land \langle A\rangle\gamma_2$ ⇔ designing a game in which $\gamma_1$ is guaranteed and $A$ can achieve $\gamma_2$
- (Frame satisfiability ⇔ Mechanism design)
Motivating Example: Rescue Robots

- The robots can rescue all the people in the building.
  \[ \bigwedge_{i \in \text{People}} \langle \langle Robots \rangle \rangle F \text{safe}_i \]
  Alternative formalization:
  \[ \langle \langle Robots \rangle \rangle F \left( \bigwedge_{i \in \text{People}} \text{safe}_i \right) \]

- If person \( i \) gets outside the building then she can stay away from trouble forever.
  \[ A G ( \text{outside}_i \rightarrow \langle \langle i \rangle \rangle G \text{safe}_i) \]

- Person \( i \) may be rescued without any robot ever entering the building, but guaranteed rescue requires some robots to enter.
  \[ E \left( \left( \bigwedge_{j \in \text{Robots}} \text{outside}_j \right) \cup \text{safe}_i \right) \]
  \[ \land \land \neg \langle \langle Robots \rangle \rangle \left( \left( \bigwedge_{j \in \text{Robots}} \text{outside}_j \right) \cup \text{safe}_i \right) \]
Motivating Example: Rescue Robots

- The robots can rescue all the people in the building.
  \[ \bigwedge_{i \in \text{People}} \langle\langle \text{Robots} \rangle\rangle F \text{ safe}_i \]
  Alternative formalization:
  \[ \langle\langle \text{Robots} \rangle\rangle F \left( \bigwedge_{i \in \text{People}} \text{ safe}_i \right) \]

- If person \( i \) gets outside the building then she can stay away from trouble forever.
  \[ AG(\text{ outside}_i \rightarrow \langle\langle i \rangle\rangle G \text{ safe}_i) \]

- Person \( i \) may be rescued without any robot ever entering the building, but guaranteed rescue requires some robots to enter.
  \[ E\left( \left( \bigwedge_{j \in \text{Robots}} \text{ outside}_j \right) U \text{ safe}_i \right) \]
  \[ \land \quad \neg \langle\langle \text{Robots} \rangle\rangle \left( \left( \bigwedge_{j \in \text{Robots}} \text{ outside}_j \right) U \text{ safe}_i \right) \]
The robots can rescue all the people in the building.

\[ \land_{i \in \text{People}} \langle\langle \text{Robots}\rangle\rangle F \text{safe}_i \]

Alternative formalization:

\[ \langle\langle \text{Robots}\rangle\rangle F (\land_{i \in \text{People}} \text{safe}_i) \]

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

\[ AG(\text{outside}_i \rightarrow \langle\langle i \rangle\rangle G \text{safe}_i) \]

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

\[ E((\land_{j \in \text{Robots}} \text{outside}_j) \cup \text{safe}_i) \]
\[ \land \neg \langle\langle \text{Robots}\rangle\rangle ((\land_{j \in \text{Robots}} \text{outside}_j) \cup \text{safe}_i) \]
Motivating Example: Rescue Robots

- The robots can rescue all the people in the building.
  \[ \bigwedge_{i \in \text{People}} \langle \langle \text{Robots} \rangle \rangle F \text{safe}_i \]
  Alternative formalization:
  \[ \langle \langle \text{Robots} \rangle \rangle F \left( \bigwedge_{i \in \text{People}} \text{safe}_i \right) \]

- If person \( i \) gets outside the building then she can stay away from trouble forever.
  \[ \mathcal{A} \mathcal{G} (\text{outside}_i \rightarrow \langle \langle i \rangle \rangle \mathcal{G} \text{safe}_i) \]

- Person \( i \) may be rescued without any robot ever entering the building, but guaranteed rescue requires some robots to enter.
  \[ \mathcal{E} \left( \left( \bigwedge_{j \in \text{Robots}} \text{outside}_j \right) \cup \text{safe}_i \right) \]
  \[ \land \neg \langle \langle \text{Robots} \rangle \rangle \left( \left( \bigwedge_{j \in \text{Robots}} \text{outside}_j \right) \cup \text{safe}_i \right) \]
Motivating Example: Rescue Robots

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

\[ \bigwedge_{i \in \text{People}} (\langle Robots \rangle F \text{safe}_i) \]

Alternative formalization:

\[ \langle Robots \rangle F \left( \bigwedge_{i \in \text{People}} \text{safe}_i \right) \]

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

\[ AG(\text{outside}_i \rightarrow \langle \{i\} \rangle G \text{safe}_i) \]

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

\[ E\left( (\bigwedge_{j \in \text{Robots}} \text{outside}_j) U \text{safe}_i \right) \]

\[ \land \neg \langle \langle \text{Robots} \rangle \rangle \left( (\bigwedge_{j \in \text{Robots}} \text{outside}_j) U \text{safe}_i \right) \]
Motivating Example: Rescue Robots

- The robots can rescue all the people in the building.
  \[ \bigwedge_{i \in \text{People}} \langle \langle \text{Robots} \rangle \rangle F \text{safe}_i \]
  Alternative formalization:
  \[ \langle \langle \text{Robots} \rangle \rangle F \left( \bigwedge_{i \in \text{People}} \text{safe}_i \right) \]

- If person \( i \) gets outside the building then she can stay away from trouble forever.
  \[ \langle \langle \emptyset \rangle \rangle G (\text{outside}_i \rightarrow \langle \langle i \rangle \rangle G \text{safe}_i) \]

- Person \( i \) may be rescued without any robot ever entering the building, but guaranteed rescue requires some robots to enter.
  \[ E \left( \left( \bigwedge_{j \in \text{Robots}} \text{outside}_j \right) \cup \text{safe}_i \right) \]
  \[ \wedge \neg \langle \langle \text{Robots} \rangle \rangle \left( \left( \bigwedge_{j \in \text{Robots}} \text{outside}_j \right) \cup \text{safe}_i \right) \]
Motivating Example: Rescue Robots

- The robots can rescue all the people in the building.
  \[ \bigwedge_{i \in \text{People}} \langle\langle \text{Robots} \rangle\rangle F \text{safe}_i \]

  Alternative formalization:
  \[ \langle\langle \text{Robots} \rangle\rangle F \left( \bigwedge_{i \in \text{People}} \text{safe}_i \right) \]

- If person \( i \) gets outside the building then she can stay away from trouble forever.
  \[ \langle\langle \emptyset \rangle\rangle G (\text{outside}_i \rightarrow \langle\langle \langle i \rangle\rangle G \text{safe}_i \rangle) \]

- Person \( i \) may be rescued without any robot ever entering the building, but guaranteed rescue requires some robots to enter.
  \[ E \left( \left( \bigwedge_{j \in \text{Robots}} \text{outside}_j \right) U \text{safe}_i \right) \]
  \[ \wedge \neg \langle\langle \text{Robots} \rangle\rangle \left( \left( \bigwedge_{j \in \text{Robots}} \text{outside}_j \right) U \text{safe}_i \right) \]
Motivating Example: Rescue Robots

The robots can rescue all the people in the building.

\[ \bigwedge_{i \in \text{People}} \langle\langle \text{Robots}\rangle\rangle F \text{safe}_i \]

Alternative formalization:

\[ \langle\langle \text{Robots}\rangle\rangle F \left( \bigwedge_{i \in \text{People}} \text{safe}_i \right) \]

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

\[ \langle\langle \emptyset \rangle\rangle G (\text{outside}_i \rightarrow \langle\langle i \rangle\rangle G \text{safe}_i) \]

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

\[ E \left( \left( \bigwedge_{j \in \text{Robots}} \text{outside}_j \right) \cup \text{safe}_i \right) \]
\[ \land \neg \langle\langle \text{Robots}\rangle\rangle \left( \left( \bigwedge_{j \in \text{Robots}} \text{outside}_j \right) \cup \text{safe}_i \right) \]
Motivating Example: Rescue Robots

- The robots can rescue all the people in the building.
  \[ \bigwedge_{i \in \text{People}} \langle\langle \text{Robots} \rangle\rangle F \text{safe}_i \]
  Alternative formalization:
  \[ \langle\langle \text{Robots} \rangle\rangle F \left( \bigwedge_{i \in \text{People}} \text{safe}_i \right) \]

- If person \( i \) gets outside the building then she can stay away from trouble forever.
  \[ \langle\langle \emptyset \rangle\rangle G (\text{outside}_i \rightarrow \langle\langle i \rangle\rangle G \text{safe}_i) \]

- Person \( i \) may be rescued without any robot ever entering the building, but guaranteed rescue requires some robots to enter.
  \[ \langle\langle \text{Agt} \rangle\rangle \left( \left( \bigwedge_{j \in \text{Robots}} \text{outside}_j \right) \cup \text{safe}_i \right) \]
  \[ \land \quad \neg \langle\langle \text{Robots} \rangle\rangle \left( \left( \bigwedge_{j \in \text{Robots}} \text{outside}_j \right) \cup \text{safe}_i \right) \]
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.
  \[ \neg \langle \langle \text{system} \rangle \rangle F (\bigvee_{c \in \text{Candidates}} \text{revealedVote}_{i,c}) \]

- A voter \( i \) can gain no receipt which can be used to prove that she voted in a certain way.
  \[ \neg \langle \langle i \rangle \rangle F (\bigvee_{c \in \text{Candidates}} \text{receiptVote}_{i,c}) \]

- A voter \( i \) cannot cooperate with the coercer to prove to him that she voted in a certain way.
  \[ \neg \langle \langle i, \text{coercer} \rangle \rangle F \ldots ? \]

  **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.
Alternating-time Temporal Logic.

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.

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
Presses Universitaires de Louvain/ACM DL.
Part 5: Verification of Strategic Ability

Verification of Strategic Ability

5.1 Properties of Alternating-time Temporal Logic
5.2 Model Checking ATL
5.3 Model Checking ATL*
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 \( s \).
- \( E \gamma \) ("there is a path on which \( \gamma \) holds") can be then translated as \( \langle \langle s \rangle \rangle \gamma \) ("the system can behave in a way that makes \( \gamma \) true"). Or \( [\emptyset] \gamma \) equivalently.
- \( A \gamma \) ("for all paths, \( \gamma \) holds") can be translated as \( [s] \gamma \) ("\( \gamma \) 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:

- \( E \gamma \equiv \langle \langle A \rangle \rangle \gamma \) ("there is a path" = outcomes obtainable by grand coalition).
  Or \([\emptyset] \gamma\) equivalently.

- \( A \gamma \equiv [A] \gamma \) ("for all paths" = necessary outcomes).
  Or \( \langle \langle \emptyset \rangle \rangle \gamma\) equivalently.

In particular,

\[
\langle A \rangle \gamma \equiv [\emptyset] \gamma
\]

\[
[ A ] \gamma \equiv \langle \langle \emptyset \rangle \rangle \gamma
\]

But in general,

\[
\langle A \rangle \gamma \neq [ \overline{A} ] \gamma
\]
**Definition 5.1 (Determinacy)**

In each state, either the players in $A$ can win with objective $\gamma$, 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:

$$\langle \langle A \rangle \rangle \gamma \equiv \neg \langle \langle A \rangle \rangle \neg \gamma \equiv [A] \gamma$$

In general, this scheme of formula is only valid for $A = \texttt{Agt}$ (see previous slide).
Determinacy in CGS

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

![Diagram of a CGS for rock-paper-scissors]

**Figure:** a CGS for rock-paper-scissors.

In $s_0$ we have that $[2]X\text{win}_1$
But, $s_0 \not\models \langle 1 \rangle X \text{win}_1$
Memory and Strategic Abilities in “Vanilla” ATL

Let us discern between two definitions of the satisfaction relation:

\[ \models_R : \text{perfect recall is assumed, strategies are of type } f : St^+ \to Act \]

\[ \models_r : \text{only memoryless strategies are allowed, i.e., } f : St \to Act \]

**Theorem 5.4**

For any CGS \( M \), state \( q \) and ATL formula \( \varphi \), we have:

\[(M, q) \models_r \varphi \iff (M, q) \models_R \varphi.\]

Memory does not influence strategic abilities in “Vanilla” ATL!
Definition 5.5 (Tree-unfolding)

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

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

Perfect recall strategies in $M$ correspond to memoryless strategies in $T(M, q)$: for every $q$ and $\varphi$,

$$(T(M, q), q) \models_r \varphi \quad \text{iff} \quad (M, q) \models_R \varphi$$

For every $(M, q)$, we have $T(M, q) \Leftrightarrow_\beta M$ for $\beta = \{ (h, last(h)) \mid h \in hist_M(q) \}$.

Finally, by invariance under bisimulation,

$$(M, q) \models_r \varphi \quad \text{iff} \quad T(M, q), q) \models_r \varphi \quad \text{iff} \quad (M, q) \models_R \varphi$$
ATL* and Memory

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 $\varphi$, such that

$$(M, q) \models_r \varphi \iff (M, q) \models_R \varphi$$

Counterexample:

$M$:

$$\begin{align*}
&\alpha \\
&\beta \\
&\varphi = \langle\langle a \rangle\rangle (Xp \land XX\neg p)
\end{align*}$$
The following formulas are valid in ATL:

\[ \langle\langle A\rangle\rangle \varphi_1 U \varphi_2 \iff \varphi_2 \lor (\varphi_1 \land \langle\langle A\rangle\rangle X \langle\langle A\rangle\rangle \varphi_1 U \varphi_2) \]
\[ \langle\langle A\rangle\rangle F \varphi \iff \varphi \lor \langle\langle A\rangle\rangle X \langle\langle A\rangle\rangle F \varphi \]
\[ \langle\langle A\rangle\rangle G \varphi \iff \varphi \land \langle\langle A\rangle\rangle X \langle\langle A\rangle\rangle G \varphi \]

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
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.
### Procedure $mcheck(\mathcal{M}, \varphi)$.

Global model checking formulas of ATL.

Returns the exact subset of $St$ for which formula $\varphi$ holds.

<table>
<thead>
<tr>
<th>Case</th>
<th>Formula Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varphi \equiv p$</td>
<td>return $\mathcal{V}(p)$</td>
<td></td>
</tr>
<tr>
<td>$\varphi \equiv \neg \psi$</td>
<td>return $St \setminus mcheck(\mathcal{M}, \psi)$</td>
<td></td>
</tr>
<tr>
<td>$\varphi \equiv \psi_1 \land \psi_2$</td>
<td>return $mcheck(\mathcal{M}, \psi_1) \cap mcheck(\mathcal{M}, \psi_2)$</td>
<td></td>
</tr>
<tr>
<td>$\varphi \equiv \langle A \rangle X \psi$</td>
<td>return $pre(A, mcheck(\mathcal{M}, \psi))$</td>
<td></td>
</tr>
<tr>
<td>$\varphi \equiv \langle A \rangle G \psi$</td>
<td>return $mcheckG(A, \mathcal{M}, \psi)$</td>
<td></td>
</tr>
<tr>
<td>$\varphi \equiv \langle A \rangle \psi_1 U \psi_2$</td>
<td>return $mcheckU(A, \mathcal{M}, \psi_1, \psi_2)$</td>
<td></td>
</tr>
</tbody>
</table>

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

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 $Q$. 
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 $\gamma$ no matter what the other agents do.

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

    $Q_1 := Q$;
    $Q_2 := Q_3 := mcheck(M, \psi)$;
    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** \( \text{mcheckU}(A, \mathcal{M}, \psi_1, \psi_2) \).

Returns the subset of \( St \) for which formula \( \langle\langle A\rangle\rangle \psi_1 U \psi_2 \) holds.

\[
\begin{align*}
Q_1 & := \emptyset; \\
Q_2 & := \text{mcheck}(\mathcal{M}, \psi_2); \\
Q_3 & := \text{mcheck}(\mathcal{M}, \psi_1); \\
\textbf{while} & \ Q_2 \not\subseteq Q_1 \ \textbf{do} \\
& \quad Q_1 := Q_1 \cup Q_2; \\
& \quad Q_2 := \text{pre}(A, Q_1) \cap Q_3 \\
\textbf{od}; \\
\text{return} & \ Q_1
\end{align*}
\]
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
Verification examples

- We want to find the set of states from which agents 1 and 3 can move the cargo to any given location: $\langle \langle 1, 3 \rangle \rangle F_{caP} \land \langle \langle 1, 3 \rangle \rangle F_{caL}$

- How does that work for the coalition of agents 1 and 2: $\langle \langle 1, 2 \rangle \rangle F_{caP}$?

- What about a maintenance goal, like agent 3 keeping the cargo in Paris forever: $\langle \langle 3 \rangle \rangle G_{caP}$?
Simple Rocket Domain: Verification of $\langle 1, 3 \rangle F_{caP}$
Simple Rocket Domain: Verification of $\langle 1, 3 \rangle F \text{caP}$
Simple Rocket Domain: Verification of $\langle 1, 3 \rangle F_{caP}$
Simple Rocket Domain: Verification of $\langle 1, 3 \rangle F caP$
Simple Rocket Domain: Verification of $\langle 1, 3 \rangle F^{caP}$
Simple Rocket Domain: Verification of $\langle 1, 3 \rangle F caP$
Simple Rocket Domain: Verification of $\langle 1, 3 \rangle F_{\text{caP}}$
Simple Rocket Domain: Verification of $\langle 1, 3 \rangle F_{\text{caP}}$

Diagram showing transitions between states such as `rol_nofuel_cal`, `rol_fuelOK_cal`, `nop_nofuel_cal`, and others, with labels like `<load, nop, fuel>`, `<nop, nop, fuel>`, etc., indicating actions and transitions.
Simple Rocket Domain: Verification of $\langle 1, 3 \rangle F c_{\text{aP}}$

Done!
Simple Rocket Domain: Verification of $\langle 1, 2 \rangle F caP$
Simple Rocket Domain: Verification of $\langle 1, 2 \rangle F caP$
Simple Rocket Domain: Verification of $\langle 1, 2 \rangle F \text{caP}$
Simple Rocket Domain: Verification of \( \langle 1, 2 \rangle F \text{caP} \)
Simple Rocket Domain: Verification of $\langle 1, 2 \rangle F \text{caP}$
Simple Rocket Domain: Verification of $\langle 1, 2 \rangle F c a P$

Done!
Simple Rocket Domain: Verification of \( \langle 3 \rangle G \text{caP} \)
Simple Rocket Domain: Verification of $\langle 3 \rangle G \text{caP}$
Simple Rocket Domain: Verification of $\langle 3 \rangle \mathcal{G}_{\text{caP}}$
Simple Rocket Domain: Verification of $\langle 3 \rangle G caP$
Simple Rocket Domain: Verification of $\langle 3 \rangle G caP$

Done!
Model Checking ATL: Soundness

- Soundness of the MC algorithm follows from the fixpoint characterizations of strategic modalities.
- Procedures \(m\text{check}\_G()\) and \(m\text{check}\_U()\) define functors \(\tau_1, \tau_2\) on sets:
  \[
  \tau_1(Z) = m\text{check}(\psi) \cap \text{pre}(A, Z) = \psi \land \langle\langle A\rangle\rangle X Z \\
  \tau_2(Z) = m\text{check}(\psi_2) \cup (m\text{check}(\psi_1) \cap \text{pre}(A, Z)) = \psi_2 \lor (\psi_1 \land \langle\langle A\rangle\rangle X Z)
  \]
- Loop compute fixed points: sets \(Z\) such that \(\tau(Z) = Z\).
- Recall equivalences:
  \[
  \langle\langle A\rangle\rangle G \psi \iff \psi \land \langle\langle A\rangle\rangle X \langle\langle A\rangle\rangle G \psi \\
  \langle\langle A\rangle\rangle \psi_1 U \psi_2 \iff \psi_2 \lor (\psi_1 \land \langle\langle A\rangle\rangle X \langle\langle A\rangle\rangle \psi_1 U \psi_2)
  \]
  \[
  \Rightarrow \langle\langle A\rangle\rangle G \psi \text{ and } \langle\langle A\rangle\rangle \psi_1 U \psi_2 \text{ are the fixed points computed by } m\text{check}\_G(A, \psi) \text{ and } m\text{check}\_U(A, \psi_1, \psi_2) \text{ respectively.}
  \]
- 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


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


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


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

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\textsuperscript{*} model checking is 2EXPTIME-complete in the number of the transitions in the model and the length of the formula.
- ATL\textsubscript{r}\textsuperscript{*} model checking is PSPACE-complete in the number of the transitions in the model and the length of the formula.
Model Checking $\text{ATL}_R^*$: lower bound

**Theorem 5.8**

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

**Definition 5.9 (Realizability)**

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

An LTL formula $\psi$ 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 $(X_0 \cup f(X_0))(X_1 \cup f(X_0X_1)) \ldots (X_n \cup f(X_0X_1 \ldots X_n))$ satisfies $\psi$. 
Let $X$, $Y$ be disjoint, non-empty sets of atoms.

Define a 2-player, turn-based CGS $M = \langle \text{Agt}, St, V, Act, d, o \rangle$ such that

- $\text{Agt} = \{1, 2\}$
- $St \leadsto$ valuations of atoms in $X \cup Y$
- $V : AP \rightarrow 2^{St}$ is the identity function
- $Act \leadsto$ player 1 controls atoms in $X$, player 2 atoms in $Y$
- $d \leadsto$ agents can change the value of any atom, at any state
- $o \leadsto$ the value of atoms is updated as determined by agents.

**Lemma 5.10**

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

- Translation $\tau$ “adds” $X$ operators: it is linear.
- LTL realizability can be reduced to $\text{ATL}^*_R$ model checking.
Model Checking $\text{ATL}^*_R$: upper bound

**Theorem 5.11**

$\text{ATL}^*_R$ model checking is in $\text{2EXPTIME}$.

- We can reuse procedure $\text{mcheck}(M, \varphi)$ but for formulas of type $\langle A \rangle \gamma$.
- 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 $A \gamma$ on $T$. 
Model Checking $\text{ATL}_R^*$: 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 $A\gamma$.

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 $A\gamma$.

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 $\text{ATL}^*_{\tau}$: lower bound

**Theorem 5.12**

$\text{ATL}^*_{\tau}$ model checking is (at least) as hard as LTL model checking (which is PSPACE-hard).

Reduction: an LTL formula $\varphi$ is equivalent to the $\text{ATL}^*$ formula $\langle\langle\emptyset\rangle\rangle \varphi$. 
Theorem 5.13

\[ \text{ATL}^* \text{ model checking is in PSPACE.} \]

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 \( A \gamma \) in the trimmed model \( M_{s_A} \): this can be done in PSPACE.

Complexity analysis:

- This procedure can be performed in \( \text{NPSPACE} = \text{PSPACE} \).
- The complexity of the whole procedure is in \( \text{P}^{\text{PSPACE}} = \text{PSPACE} \).
# Model Checking Temporal and Strategic Logics: Summary

<table>
<thead>
<tr>
<th>Logic</th>
<th>Memoryless Strategies</th>
<th>Perfect Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTL</td>
<td>PTIME-complete</td>
<td></td>
</tr>
<tr>
<td>LTL</td>
<td>PSPACE-complete</td>
<td></td>
</tr>
<tr>
<td>CTL*</td>
<td>PSPACE-complete</td>
<td></td>
</tr>
<tr>
<td>ATL</td>
<td>PTIME-complete</td>
<td></td>
</tr>
<tr>
<td>ATL*</td>
<td>PSPACE-complete</td>
<td>2EXPTIME-complete</td>
</tr>
</tbody>
</table>

**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.

N. Bulling and W. Jamroga.
Alternating Epistemic Mu-Calculus.

P. Y. Schobbens.
Alternating-time logic with imperfect recall.
Part 6: Abilities under Imperfect Information

Abilities under Imperfect Information

6.1 Strategies and Knowledge
6.2 Properties of $\text{ATL}_i$
6.3 The Subjective Interpretation
So far we considered games of \textit{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:

\begin{quote}
What can players achieve in such scenarios?
\end{quote}
Ability and Knowledge

Classic AI: ability and knowledge are intimately connected

1969 : McCarthy & Hayes:

\textit{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; \textit{Reasoning about Knowledge and Action}
Ability and Knowledge

[McCarthy & Hayes, 1969]: what does it mean that “a computer program $\pi$ to be able to achieve a state of affairs $\phi$?”

1 objective ability (omniscient external observer): “there is a sub-program $\sigma$ […] which would achieve $\phi$ if it were in memory, and control were transferred to $\pi$. No assertion is made that $\pi$ knows $\sigma$ or even knows that $\sigma$ exists.”

2 subjective ability: “$\sigma$ exists as above and that $\sigma$ will achieve $\phi$ follows from information in memory according to a proof that $\pi$ is capable of checking.”

3 practical ability (strategy synthesis): “$\pi$’s standard problem-solving procedure will find $\sigma$ if achieving $\phi$ 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
[Moore, 1981]: *Reasoning about Knowledge and Action*

Formalisation of ability in terms of knowledge and action:

\[(Can \phi) \iff \exists \alpha K(Res \alpha \phi) \lor \exists \alpha K(Res \alpha(Can \phi))\]

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

```
\begin{align*}
(*, H) &\rightarrow (*, H) & (*, T) \\
(*, T) &\rightarrow (H, *) & (T, *) \\
(H, H) &\rightarrow (*, *) & (T, H) \\
(T, H) &\rightarrow (*, *) & (H, T) \\
(H, T) &\rightarrow (*, *) & (T, T) \\
(T, T) &\rightarrow (*, *)
\end{align*}
```

$\text{win}_A$
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:

```
A

(*, *)

(H, *)

(T, *)

(*, H)

(*, T)

(A, *)

win_A

H, H

T, H

H, T

T, T
```

F. Belardinelli · Logics for Strategic Reasoning in AI Imperial College London – Spring Term 2019
Concurrent Game Structures with Imperfect Information

Definition 6.1 (iCGS)

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

- $\text{Agt}, \text{St}, \mathcal{V}, o$ are defined as for CGS
- for every $a \in \text{Agt}$, $\sim_a$ is an equivalence relation on $\text{St}$
- $d : \text{Agt} \times \text{St} \rightarrow 2^{\text{Act}}$ 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.
Winning strategy $\sigma$ such that

- $\sigma(\ast, \ast) = \ast$
- $\sigma(\ast, H) = H$
- $\sigma(\ast, T) = T$
Matching pennies again:

\[ \begin{align*}
\text{win}_A &\quad H, H \\
&\quad T, H \\
&\quad H, T \\
&\quad T, T
\end{align*} \]

Does strategy \( \sigma \) still make sense?
Does \( (\ast, \ast) \models \langle A \rangle F \text{win}_A \)?
Strategies and Knowledge

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

Does it make sense?
Strategies and Knowledge

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

\[ (-,-) \models \langle \langle a \rangle \rangle F \text{win} \]

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 robber 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 shoot 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!

\[
\langle\langle A\rangle\rangle_\gamma = A \text{ can enforce } \gamma
\]

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**
Executable Strategies

**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.
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 $\langle A \rangle \gamma$ under imperfect information:

1. There is $\sigma$ (not necessarily executable!) such that, for every execution of $\sigma$, $\gamma$ holds.

2. There is a uniform $\sigma$ such that, for every execution of $\sigma$, $\gamma$ holds (objective interpretation).

3. $A$ know that there is a uniform $\sigma$ such that, for every execution of $\sigma$, $\gamma$ holds.

4. There is a uniform $\sigma$ such that $A$ know that, for every execution of $\sigma$, $\gamma$ holds (subjective interpretation).

Hereafter we focus on 2 and 4 (starting with 2).
**Definition 6.3 (Semantics of ATL$_i$*)**

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

**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.
**Definition 6.3 (Semantics of ATL\textsubscript{i}*)**

\[(M, q) \models_{i} \langle A \rangle \gamma \iff \text{there is a collective uniform strategy } s_{A} \text{ such that, for every path } \lambda \in \text{out}(q, s_{A}), (M, \lambda) \models \gamma.\]

**Definition 6.4 (Semantics of ATL\textsubscript{i}*: path formulae)**

\[(M, \lambda) \models_{i} \varphi \quad \text{iff} \quad (M, \lambda[0]) \models_{i} \varphi, \text{ for a state formula } \varphi\]

\[(M, \lambda) \models_{i} X \gamma \quad \text{iff} \quad (M, \lambda[1..\infty]) \models_{i} \gamma\]

\[(M, \lambda) \models_{i} \gamma_{1} \cup \gamma_{2} \quad \text{iff} \quad (M, \lambda[k..\infty]) \models_{i} \gamma_{2} \text{ for some } k \geq 0, \text{ and } (M, \lambda[j..\infty]) \models_{i} \gamma_{1} \text{ for all } 0 \leq j \leq k\]

As for ATL\textsuperscript{*}, we can consider both perfect and imperfect recall strategies.
Strategies and Knowledge

Matching pennies revisited:

Alice no longer has a winning strategy!
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: $s_a : St \rightarrow Act$ (memoryless strategies)
- R: $s_a : St^+ \rightarrow Act$ (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 $i$, $h[i] \sim_a h'[i]$

<table>
<thead>
<tr>
<th></th>
<th>imperfect recall</th>
<th>perfect recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>perfect information</td>
<td>$\text{ATL}_{ir}^*$</td>
<td>$\text{ATL}_{IR}^*$</td>
</tr>
<tr>
<td>imperfect information</td>
<td>$\text{ATL}_{ir}^*$</td>
<td>$\text{ATL}_{iR}^*$</td>
</tr>
</tbody>
</table>

- 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 ATL$_i$:

\[
\begin{align*}
\langle\langle A \rangle\rangle_i \varphi_1 U \varphi_2 & \iff \varphi_2 \lor \varphi_1 \land \langle\langle A \rangle\rangle_i X \langle\langle A \rangle\rangle_i \varphi_1 U \varphi_2 \\
\langle\langle A \rangle\rangle_i F \varphi & \iff \varphi \lor \langle\langle A \rangle\rangle_i X \langle\langle A \rangle\rangle_i F \varphi \\
\langle\langle A \rangle\rangle_i G \varphi & \iff \varphi \land \langle\langle A \rangle\rangle_i X \langle\langle A \rangle\rangle_i G \varphi
\end{align*}
\]

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

$\sim$ we cannot have incremental model checking algorithms as for ATL$_i$. 
Non-Composability of Strategies: Matching Pennies

\[
\langle \langle A \rangle \rangle X \langle \langle A \rangle \rangle F_{\text{win}_A} \not\rightarrow \langle \langle A \rangle \rangle F_{\text{win}_A}
\]

Hence,

\[
\langle \langle A \rangle \rangle X \langle \langle A \rangle \rangle F_{\text{win}_A} \not\rightarrow \langle \langle A \rangle \rangle F_{\text{win}_A}
\]
Non-Composability of Strategies: Matching Pennies

\[
\langle\langle A \rangle\rangle F\text{\textit{win}}_A \not\models \langle\langle A \rangle\rangle F\text{\textit{win}}_A
\]

\[
\langle\langle A \rangle\rangle F\text{\textit{win}}_A \models A \models \langle\langle A \rangle\rangle F\text{\textit{win}}_A
\]

Hence,

\[
\langle\langle A \rangle\rangle X \langle\langle A \rangle\rangle F\text{\textit{win}}_A \not\models \langle\langle A \rangle\rangle F\text{\textit{win}}_A
\]
Non-Composability of Strategies: Matching Pennies

Hence,

\[ \langle A \rangle X \langle A \rangle F^{win_A} \not \rightarrow \langle A \rangle F^{win_A} \]
Non-Composability of Strategies

In general, the following are validities in $\text{ATL}_i$:

\[
\langle\langle A \rangle\rangle_i \varphi_1 U \varphi_2 \rightarrow \varphi_2 \lor (\varphi_1 \land \langle\langle A \rangle\rangle_i X \langle\langle A \rangle\rangle_i \varphi_1 U \varphi_2)
\]

\[
\langle\langle A \rangle\rangle_i F \varphi \rightarrow \varphi \lor \langle\langle A \rangle\rangle_i X \langle\langle A \rangle\rangle_i F \varphi
\]

\[
\langle\langle A \rangle\rangle_i G \varphi \rightarrow \varphi \land \langle\langle A \rangle\rangle_i X \langle\langle A \rangle\rangle_i G \varphi
\]

But,

\[
\varphi_2 \lor (\varphi_1 \land \langle\langle A \rangle\rangle_i X \langle\langle A \rangle\rangle_i \varphi_1 U \varphi_2) \nleftrightarrow \langle\langle A \rangle\rangle_i \varphi_1 U \varphi_2
\]

\[
\varphi \lor \langle\langle A \rangle\rangle_i X \langle\langle A \rangle\rangle_i F \varphi \nleftrightarrow \langle\langle A \rangle\rangle_i F \varphi
\]

\[
\varphi \land \langle\langle A \rangle\rangle_i X \langle\langle A \rangle\rangle_i G \varphi \nleftrightarrow \langle\langle A \rangle\rangle_i G \varphi
\]

For $\text{ATL}_{iR}$ the opposite is the case.
Conjecture

Strategies cannot be synthesized incrementally.

Indeed...

Theorem (Schobbens 2004; Jamroga & Dix 2006)

Model checking $\text{ATL}_{ir}$ is $\Delta^P_2$-complete.
Under perfect information, memory does not affect the interpretation of ATL.
But it does affect ATL*.
What is it like for imperfect information?
- Obviously, $\text{ATL}_\text{ir}^* \neq \text{ATL}_\text{iR}^*$.
- But also $\text{ATL}_\text{ir} \neq \text{ATL}_\text{iR}$:

\[
q_1 \not\models_{\text{ir}} \langle\langle a\rangle\rangle F \text{shot}
\]
\[
q_1 \models_{\text{iR}} \langle\langle a\rangle\rangle F \text{shot}
\]
Motivating Example: Rescue Robots

- The robots can rescue all the people in the building.
  \[ \bigwedge_{i \in \text{People}} \langle \langle \text{Robots} \rangle \rangle F \text{safe}_i \]
  Alternative formalization:
  \[ \langle \langle \text{Robots} \rangle \rangle F \left( \bigwedge_{i \in \text{People}} \text{safe}_i \right) \]
  **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 ATL^i* !

- The robots can rescue all the people, and they **know how to do it**
  Cannot be expressed in ATL^i* !
Motivating Example: Voting

- The system cannot reveal how a particular voter voted.
  \[ \neg \langle \langle \text{system} \rangle \rangle F (\bigvee_{c \in \text{Candidates}} \text{revealedVote}_{i,c}) \]

- A voter \( i \) can gain no receipt which can be used to prove that she voted in a certain way.
  \[ \neg \langle \langle i \rangle \rangle F (\bigvee_{c \in \text{Candidates}} \text{receiptVote}_{i,c}) \]

- 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 $\langle A \rangle \gamma$ under imperfect information:

1. There is a strategy $\sigma$ (not necessarily executable!) such that, for every execution of $\sigma$, $\gamma$ holds.

2. There is a uniform strategy $\sigma$ such that, for every execution of $\sigma$, $\gamma$ holds (objective interpretation).

3. $A$ know that there is a uniform $\sigma$ such that, for every execution of $\sigma$, $\gamma$ holds.

4. There is a uniform $\sigma$ such that $A$ know that, for every execution of $\sigma$, $\gamma$ holds (subjective interpretation).

We dealt with 2 in the last lecture

Hereafter we focus on 4. Why?
Having a successful strategy does not imply knowing that we have it!

\[ q_0 = \langle\langle a \rangle\rangle X \text{ shot} \text{ and } q_1 = \langle\langle a \rangle\rangle X \text{ shot} \]

But for two different strategies!
Knowing How to Play: Subjective Interpretation

Case [4]: knowing how to play

- Single agent case: we consider all paths starting from indistinguishable states:

\[
\text{out}_s(q, s_a) = \bigcup_{q' \sim_a q} \text{out}_o(q', s_a)
\]

- \( \langle \langle a \rangle \rangle_{ir} \gamma \): agent \( a \) knows how to play to enforce \( \gamma \) from all the states she considers possible

- What about coalitions?

\[
\text{out}_s(q, s_A) = \bigcup_{a \in A} \bigcup_{q' \sim_a q} \text{out}_o(q', s_A)
\]

- \( \langle \langle A \rangle \rangle_{ir} \gamma \): all agents in \( A \) know how to play to enforce \( \gamma \).

**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 \( \lambda \in \text{out}_s(q, s_A) \), \( (M, \lambda) \models \gamma \).
But,

\[ q_0 \not\models_r \langle a \rangle F \text{ shot} \quad \text{and} \quad q_1 \not\models_r \langle a \rangle F \text{ shot} \]

But,

\[ q_0 \models_R \langle a \rangle F \text{ shot} \quad \text{and} \quad q_1 \models_R \langle a \rangle F \text{ shot} \]
Decomposability of Strategies: Matching Pennies

\[
\langle \langle A \rangle \rangle X \langle \langle A \rangle \rangle F \text{win}_A \rightarrow \langle \langle A \rangle \rangle F \text{win}_A
\]

Hence,
Decomposability of Strategies: Matching Pennies

Hence,

\[ \langle \langle A \rangle \rangle X \langle \langle A \rangle \rangle F_{\text{win}_A} \rightarrow \langle \langle A \rangle \rangle F_{\text{win}_A} \]
Decomposability of Strategies: Matching Pennies

\[ \langle \langle A \rangle \rangle X \langle \langle A \rangle \rangle F_{\text{win}_A} \not\models \langle \langle A \rangle \rangle F_{\text{win}_A} \]

\[ \begin{align*}
\langle \langle A \rangle \rangle F_{\text{win}_A} & \not\models \langle \langle A \rangle \rangle F_{\text{win}_A} \\
* , H & \not\models \langle \langle A \rangle \rangle F_{\text{win}_A} \\
\langle \langle A \rangle \rangle F_{\text{win}_A} & \not\models \langle \langle A \rangle \rangle F_{\text{win}_A} \\
* , T & \not\models \langle \langle A \rangle \rangle F_{\text{win}_A} \\
\langle \langle A \rangle \rangle F_{\text{win}_A} & \not\models \langle \langle A \rangle \rangle F_{\text{win}_A}
\end{align*} \]

\[ \begin{align*}
\langle \langle A \rangle \rangle F_{\text{win}_A} & \not\models \langle \langle A \rangle \rangle F_{\text{win}_A} \\
* , H & \not\models \langle \langle A \rangle \rangle F_{\text{win}_A} \\
\langle \langle A \rangle \rangle F_{\text{win}_A} & \not\models \langle \langle A \rangle \rangle F_{\text{win}_A} \\
* , T & \not\models \langle \langle A \rangle \rangle F_{\text{win}_A} \\
\langle \langle A \rangle \rangle F_{\text{win}_A} & \not\models \langle \langle A \rangle \rangle F_{\text{win}_A}
\end{align*} \]

Hence,

\[ \langle \langle A \rangle \rangle X \langle \langle A \rangle \rangle F_{\text{win}_A} \rightarrow \langle \langle A \rangle \rangle F_{\text{win}_A} \]
Example: Robots and Carriage

- Robot 1 only perceives the color of the surface;
- Robot 2 only perceives the texture.
Example: Robots and Carriage

\[
\begin{align*}
\text{pos}_0 & \rightarrow \neg \langle 1 \rangle_{sr} G \neg \text{pos}_1 \\
\text{pos}_0 & \rightarrow \neg \langle 1, 2 \rangle_{sr} G \neg \text{pos}_1 \\
\text{pos}_0 & \rightarrow \langle 1, 2 \rangle_{sr} F \text{ pos}_1
\end{align*}
\]
Part 7: Model Checking Imperfect Information

Model Checking Imperfect Information
Imperfect information makes the model checking problem harder to solve.

<table>
<thead>
<tr>
<th>ATL</th>
<th>$I$</th>
<th>$i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r$</td>
<td>linear time</td>
<td>$\Delta^P_2$-complete</td>
</tr>
<tr>
<td>$R$</td>
<td></td>
<td>undecidable</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ATL*</th>
<th>$I$</th>
<th>$i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r$</td>
<td></td>
<td>PSPACE-c</td>
</tr>
<tr>
<td>$R$</td>
<td>2EXPTIME-c</td>
<td>undecidable</td>
</tr>
</tbody>
</table>

The complexity does not change for the objective and subjective interpretation.

We first consider imperfect recall and then perfect recall.
Recall: $\langle A \rangle_i$ 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).
Theorem (Schobbens 2004; Jamroga & Dix 2006)

Model checking ATL$_{ir}$ is $\Delta^P_2$-complete in the number of transitions in the model and the length of the formula.

$\Delta^P_2$: 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 $\text{ATL}_{ir}$: Upper Bound

Checking $(\mathcal{M}, q) \models_{ir} \langle A \rangle \gamma$ where $\gamma$ 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 $A\gamma$ 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: $P^{NP} = \Delta^P_2$. 
Model Checking $\text{ATL}_{ir}$: Lower Bound

We prove NP-hardness by reducing model checking $\text{ATL}_{ir}$ 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 \( M_{\varphi} \) for \( \varphi \equiv (x_1 \lor \neg x_3) \land (\neg x_1 \lor x_2 \lor x_3) \)

**Proposition**

\[ \exists v_1, \ldots, v_k . \varphi(v_1, \ldots, v_k) \iff (M_{\varphi}, q_0) \models \langle v \rangle F \text{yes}. \]
Model Checking $\text{ATL}_{ir}^*$

**Good News**: Model checking $\text{ATL}_{ir}^*$ is PSPACE-complete (just like perfect information).

**Theorem**

Model checking $\text{ATL}_{ir}^*$ 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 $A \gamma$ 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 ($O(|\varphi|)$) a procedure in PSPACE: $P^{PSPACE} = PSPACE$.

**PSPACE-hardness**: model checking $\text{ATL}_{ir}^*$ 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!
What about agents with perfect recall and imperfect information?

The news are bad...

**Theorem [Dima and Tiplea, 2011]**

Model checking $\text{ATL}_{iR}$ 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

$$(M_T, s_{init}) \models_{iR} \langle\langle 1, 2 \rangle\rangle \text{Gok}.$$
### Model Checking Complexity for Variants of ATL: Summary

<table>
<thead>
<tr>
<th>ATL</th>
<th>$I$</th>
<th>$i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r$</td>
<td>linear time</td>
<td>$\Delta^P_2$-complete</td>
</tr>
<tr>
<td>$R$</td>
<td></td>
<td>undecidable</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ATL*</th>
<th>$I$</th>
<th>$i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r$</td>
<td></td>
<td>PSPACE-c</td>
</tr>
<tr>
<td>$R$</td>
<td>2EXPTIME-c</td>
<td>undecidable</td>
</tr>
</tbody>
</table>
References

N. Bulling, J. Dix, and W. Jamroga.
Model checking logics of strategic ability: Complexity.

Alternating-time Temporal Logic.

P. Y. Schobbens.
Alternating-time logic with imperfect recall.