Verification of Multi-Agent Systems with ATL

Nicolas Markey
Joint work with François Laroussinie and Ghassan Oreiby

Lab. Specification et Verification
ENS Cachan & CNRS, France

FNRS contact day on “Synthesis and Verification”

October 18, 2006
Principles of model checking

system:

property:
Principles of model checking

system:

property:

\[ G(\text{request} \Rightarrow F\text{grant}) \]
Principles of model checking

system:  ⇒  property:

$G(\text{request} \Rightarrow F\text{grant})$
Principles of model checking

System:

Property:

\[ G(\text{request} \Rightarrow F\text{grant}) \]

Model-checking algorithm

Yes/no
Kripke structures and temporal logics

- The system is modeled (for example) as a Kripke structure:
Kripke structures and temporal logics

- The system is modeled (for example) as a Kripke structure:

Example (of a simplified three-floor lift)

- second floor
- first floor
- ground floor
- cabin
Kripke structures and temporal logics

- The system is modeled (for example) as a Kripke structure:

Example (of a simplified three-floor lift)
Kripke structures and temporal logics

- The system is modeled (for example) as a Kripke structure:

Example (of a simplified three-floor lift)

- Cabin:
  - Second floor
  - First floor
  - Ground floor

- Button:
  - Go 2nd floor
  - Go 1st floor
  - Go gnd floor

- Doors:
  - Is open
  - Is closed

- Request:
  - Open?
  - Close?

- Press:
  - Button at floor

- Controller:
  - Up!
Kripke structures and temporal logics

- The system is modeled (for example) as a **Kripke structure**:

---

**Example (of a simplified three-floor lift)**

- **cabin**
  - second floor
    - down?
    - up?
  - first floor
    - down?
    - up?
  - ground floor
    - down?
    - up?

- **button**
  - go 2nd floor
  - go 1st floor
  - go gnd floor
  - request
  - open?
  - close?
  - press?
  - call
  - idle

- **doors**
  - is open
  - is closed
  - pb
  - ok

---
Kripke structures and temporal logics

- The system is modeled (for example) as a Kripke structure:

Example (of a simplified three-floor lift)

- Cabin
- Button
- Doors
- Controller
The system is modeled (for example) as a Kripke structure:
Kripke structures and temporal logics

- The system is modeled (for example) as a Kripke structure:

- The property is expressed in some (formal) specification language, e.g.:
  - propositional logic for describing an invariant of the system;

**Example**

$$\neg (\text{door}_1.\text{is open} \land \text{door}_2.\text{is open})$$

$$\text{door}_0.\text{is open} \Rightarrow \text{cabin. ground floor}$$
Kripke structures and temporal logics

- The system is modeled (for example) as a Kripke structure:

- The property is expressed in some (formal) specification language, e.g.:
  - propositional logic for describing an invariant of the system;
  - temporal logics, for describing properties of the executions or execution tree of the system;

Example

always (¬(door₁.is open ∧ door₂.is open))

always (button.request₂ ⇒ eventually button.served₂)
Kripke structures and temporal logics

- The system is modeled (for example) as a Kripke structure:

- The property is expressed in some (formal) specification language, e.g.:
  - propositional logic for describing an invariant of the system;
  - temporal logics, for describing properties of the executions or execution tree of the system;
  - $\mu$-calculus, Büchi (tree) automata, ...
Temporal logics: two frameworks

- **linear-time** framework:
  - deals with one single execution at a time;

**Example**

*Any request is eventually granted (along any execution).*
Temporal logics: two frameworks

- **linear-time** framework:
  - deals with one single execution at a time;

  **Example**
  - Any request is eventually granted (along any execution).

- **branching-time** framework:
  - deals with the *computation tree* of the system;
  - quantification on the possible evolutions of the system.

  **Example**
  - *It is always possible to go to the ground floor.*
Outline of the talk

1. Introduction

2. Computation Tree Logic
   - Definition and examples
   - Expressiveness of CTL
   - CTL model-checking

3. Alternating-time Temporal Logic
   - Why multi-agent systems?
   - Modelling multi-agent systems: ATSSs and CGSSs
   - Alternating-time Temporal Logic
   - ATL model-checking
   - Implicit CGSSs

4. Conclusion
Outline of the talk

1. Introduction

2. Computation Tree Logic
   - Definition and examples
   - Expressiveness of CTL
   - CTL model-checking

3. Alternating-time Temporal Logic
   - Why multi-agent systems?
   - Modelling multi-agent systems: ATSSs and CGSSs
   - Alternating-time Temporal Logic
   - ATL model-checking
   - Implicit CGSSs

4. Conclusion
Definition (Syntax of CTL)

\[
\text{CTL} \ni \phi_s ::= p \mid \neg \phi_s \mid \phi_s \lor \phi_s \mid E \phi_p \mid A \phi_p
\]

\[
\phi_p ::= X \phi_s \mid \phi_s U \phi_s
\]

where \( p \) ranges over the set of atomic propositions.
Computation tree logic

**Definition (Syntax of CTL)**

\[ \text{CTL} \ni \phi_s ::= p \mid \neg \phi_s \mid \phi_s \lor \phi_s \mid \text{E}\phi_p \mid \text{A}\phi_p \]

\[ \phi_p ::= \text{X}\phi_s \mid \phi_s \text{U} \phi_s \]

where \( p \) ranges over the set of atomic propositions.

**Definition (Semantics of CTL)**

Given a state \( q \) of a Kripke structure \( \mathcal{K} = \langle Q, \delta, \ell \rangle \), and an execution \( \rho \) of \( \mathcal{K} \) starting in \( q \), we define:

\[ \mathcal{K}, q \models p \iff p \in \ell(q) \]

\[ \mathcal{K}, q \models \neg \phi_s \iff \mathcal{K}, q \not\models \phi_s \]

\[ \mathcal{K}, q \models \phi_s \lor \psi_s \iff \mathcal{K}, q \models \phi_s \text{ or } \mathcal{K}, q \models \psi_s \]
Computation tree logic

**Definition (Syntax of CTL)**

\[
\text{CTL} \ni \phi_s ::= \ p \ | \ \neg \phi_s \ | \ \phi_s \lor \phi_s \ | \ E\phi_p \ | \ A\phi_p \\
\phi_p ::= X\phi_s \ | \ \phi_s U \phi_s
\]

where \( p \) ranges over the set of atomic propositions.

**Definition (Semantics of CTL)**

Given a state \( q \) of a Kripke structure \( K = \langle Q, \delta, \ell \rangle \), and an execution \( \rho \) of \( K \) starting in \( q \), we define:

\[
K, q \models E\phi_p \iff \exists \rho' \text{ starting in } q, \ K, \rho' \models \phi_p \\
K, q \models A\phi_p \iff \forall \rho' \text{ starting in } q, \ K, \rho' \models \phi_p
\]
Computation tree logic

Definition (Syntax of CTL)

\[ \text{CTL} \ni \phi_s ::= p \mid \neg \phi_s \mid \phi_s \lor \phi_s \mid E\phi_p \mid A\phi_p \]

\[ \phi_p ::= X\phi_s \mid \phi_s U \phi_s \]

where \( p \) ranges over the set of atomic propositions.

Definition (Semantics of CTL)

Given a state \( q \) of a Kripke structure \( \mathcal{K} = \langle Q, \delta, \ell \rangle \), and an execution \( \rho \) of \( \mathcal{K} \) starting in \( q \), we define:

\[ \mathcal{K}, \rho \models X\phi_s \iff \mathcal{K}, \rho_1 \models \phi_s \]

\[ \mathcal{K}, \rho \models \phi_s U \psi_s \iff \exists n \geq 0, \mathcal{K}, \rho_n \models \psi_s \]

and \( \forall 0 \leq m < n. \mathcal{K}, \rho_m \models \phi_s \)
Computation tree logic

Definition (Semantics of CTL)

Given a state $q$ of a Kripke structure $\mathcal{K} = \langle Q, \delta, \ell \rangle$, and an execution $\rho$ of $\mathcal{K}$ starting in $q$, we define:

\[ \mathcal{K}, \rho \models X \phi_s \iff \mathcal{K}, \rho_1 \models \phi_s \]
\[ \mathcal{K}, \rho \models \phi_s U \psi_s \iff \exists n \geq 0, \mathcal{K}, \rho_n \models \psi_s \]
and $\forall 0 \leq m < n. \mathcal{K}, \rho_m \models \phi_s$

Definition

\[ F \phi_s \overset{\text{def}}{=} \text{true U } \phi_s \quad ("eventually" \phi_s) \]
\[ G \phi_s \overset{\text{def}}{=} \neg F \neg \phi_s \quad ("always" \phi_s) \]
Examples of CTL properties

Examples

- The cabin is always present when the doors are open:

  \[ \text{AG}(\text{door}_0.\text{is open} \Rightarrow \text{cabin.ground floor}) \]
Examples of CTL properties

Examples

- The cabin is always present when the doors are open:
  \[ \text{AG}(\text{door}_0.\text{is open} \Rightarrow \text{cabin.ground floor}) \]

- Any request is eventually served
  \[ \text{AG}(\text{button}_2.\text{call} \Rightarrow \text{AF door}_2.\text{is open}) \]
Examples of CTL properties

Examples

- The cabin is always present when the doors are open:
  \[ \text{AG}(\text{door}_0.\text{is open} \Rightarrow \text{cabin.ground floor}) \]

- Any request is eventually served
  \[ \text{AG}(\text{button}_2.\text{call} \Rightarrow \text{AF door}_2.\text{is open}) \]

- It is always possible to go to the ground floor
  \[ \text{AG}(\text{EF door}_0.\text{is open}) \]
Examples of CTL properties

Examples

- The cabin is always present when the doors are open:
  \[ \text{AG}(\text{door}_0.\text{is open} \Rightarrow \text{cabin.ground floor}) \]

- Any request is eventually served
  \[ \text{AG}(\text{button}_2.\text{call} \Rightarrow \text{AF door}_2.\text{is open}) \]

- It is always possible to go to the ground floor
  \[ \text{AG}(\neg \text{controller.failure} \Rightarrow \text{E} \left( \neg \text{controller.failure} \mathbin{\text{U}} \left( \text{controller.failure} \mathbin{\lor} \text{door}_0.\text{is open} \right) \right)) \]
Outline of the talk

1. Introduction

2. Computation Tree Logic
   - Definition and examples
   - Expressiveness of CTL
   - CTL model-checking

3. Alternating-time Temporal Logic
   - Why multi-agent systems?
   - Modelling multi-agent systems: ATSSs and CGSSs
   - Alternating-time Temporal Logic
   - ATL model-checking
   - Implicit CGSSs

4. Conclusion
Expressiveness of CTL

Definition

Let $K = (Q, \delta, \ell)$ and $K' = (Q', \delta', \ell')$ be two Kripke structures, and $R \subseteq Q \times Q'$. The relation $R$ is a **bisimulation** if the following conditions hold:

- if $(q, q') \in R$, then $\ell(q) = \ell(q')$;
- for any $(q, q') \in R$, and any $(q, r) \in \delta$, there exists $(q', r') \in \delta'$ s.t. $(r, r') \in R$;
- conversely, for any $(q, q') \in R$, and any $(q', r') \in \delta'$, there exists $(q, r) \in \delta$ s.t. $(r, r') \in R$.

Two states $q$ and $q'$ are bisimilar if there exists a bisimulation $R$ s.t. $(q, q') \in R$. 

Expressiveness of CTL

**Definition**

Let $\mathcal{K} = \langle Q, \delta, \ell \rangle$ and $\mathcal{K'} = \langle Q', \delta', \ell' \rangle$ be two Kripke structures, and $R \subseteq Q \times Q'$. The relation $R$ is a **bisimulation** if the following conditions hold:

- if $(q, q') \in R$, then $\ell(q) = \ell(q')$;
- for any $(q, q') \in R$, and any $(q, r) \in \delta$, there exists $(q', r') \in \delta'$ s.t. $(r, r') \in R$;
- conversely, for any $(q, q') \in R$, and any $(q', r') \in \delta'$, there exists $(q, r) \in \delta$ s.t. $(r, r') \in R$.

**Example**

![Diagram of two Kripke structures](image-url)
Expressiveness of CTL

Definition

Let $\mathcal{K} = \langle Q, \delta, \ell \rangle$ and $\mathcal{K}' = \langle Q', \delta', \ell' \rangle$ be two Kripke structures, and $R \subseteq Q \times Q'$. The relation $R$ is a bisimulation if the following conditions hold:

1. if $(q, q') \in R$, then $\ell(q) = \ell(q')$;
2. for any $(q, q') \in R$, and any $(q, r) \in \delta$, there exists $(q', r') \in \delta'$ s.t. $(r, r') \in R$;
3. conversely, for any $(q, q') \in R$, and any $(q', r') \in \delta'$, there exists $(q, r) \in \delta$ s.t. $(r, r') \in R$.

Counter-example

Diagram: Two Kripke structures $\mathcal{K}$ and $\mathcal{K}'$ with states and transitions labeled. The image shows an example of a bisimulation relation $R$ between the two structures.
Expressiveness of CTL

- CTL characterizes bisimulation:

**Theorem (Hennessy, 1980)**

*Two states of (finitely-branching) Kripke structures are bisimilar iff they satisfy exactly the same CTL formulas.*
Expressiveness of CTL

- CTL characterizes bisimulation:

Theorem (Hennessy, 1980)

Two states of (finitely-branching) Kripke structures are bisimilar iff they satisfy exactly the same CTL formulas.

- CTL can encode the behavior of a (finite) Kripke structure:

Theorem (Browne, 1988)

Given a Kripke structure $\mathcal{K}$, there exists a CTL formula $\Phi_\mathcal{K}$ s.t., for any $\mathcal{K}'$,

$$\mathcal{K}' \models \Phi_\mathcal{K} \iff \mathcal{K}' \text{ and } \mathcal{K} \text{ are bisimilar.}$$
Expressiveness of CTL

- CTL characterizes bisimulation:

  **Theorem (Hennessy, 1980)**

  Two states of (finitely-branching) Kripke structures are bisimilar iff they satisfy exactly the same CTL formulas.

- CTL can encode the behavior of a (finite) Kripke structure:

  **Theorem (Browne, 1988)**

  Given a Kripke structure $\mathcal{K}$, there exists a CTL formula $\Phi_{\mathcal{K}}$ s.t., for any $\mathcal{K}'$,

  $$\mathcal{K}' \models \Phi_{\mathcal{K}} \iff \mathcal{K}' \text{ and } \mathcal{K} \text{ are bisimilar.}$$

But:

- CTL can’t express fairness: “formula $\phi$ holds infinitely often”:

  $$\text{EGF } \phi \neq \text{ EG EF } \phi$$
Outline of the talk

1. Introduction

2. Computation Tree Logic
   - Definition and examples
   - Expressiveness of CTL
   - CTL model-checking

3. Alternating-time Temporal Logic
   - Why multi-agent systems?
   - Modelling multi-agent systems: ATSSs and CGSs
   - Alternating-time Temporal Logic
   - ATL model-checking
   - Implicit CGSs

4. Conclusion
Let $\mathcal{K} = \langle Q, \delta, \ell \rangle$, and $\phi \in \text{CTL}$. Computing the set of states satisfying $\phi$ is PTIME-complete, and can be achieved in time $O(|\phi| \times |\delta|)$. 
Theorem

Let $K = \langle Q, \delta, \ell \rangle$, and $\phi \in CTL$. Computing the set of states satisfying $\phi$ is PTIME-complete, and can be achieved in time $O(|\phi| \times |\delta|)$.

Proof.

- labeling algorithm: we recursively label each state with the set of subformulas it satisfies;
CTL model-checking

Theorem

Let $\mathcal{K} = \langle Q, \delta, \ell \rangle$, and $\phi \in \text{CTL}$. Computing the set of states satisfying $\phi$ is PTIME-complete, and can be achieved in time $O(|\phi| \times |\delta|)$.

Proof.

- labeling algorithm: we recursively label each state with the set of subformulas it satisfies;
- sufficient to consider only modalities $\text{EX}$, $\text{EG}$ and $\text{EU}$:

\[
\begin{align*}
\text{AX} \phi & \equiv \neg \text{EX} \neg \phi \\
\text{A} \phi \text{ U } \psi & \equiv \neg (\text{E} \neg \psi \text{ U } (\neg \phi \land \neg \psi)) \land \neg \text{EG} \neg \psi
\end{align*}
\]

The number of subformulas only increases linearly.
Theorem

Let $\mathcal{K} = \langle Q, \delta, \ell \rangle$, and $\phi \in \text{CTL}$. Computing the set of states satisfying $\phi$ is PTIME-complete, and can be achieved in time $O(|\phi| \times |\delta|)$.

Proof.

- labeling algorithm: we recursively label each state with the set of subformulas it satisfies;
- sufficient to consider only modalities $\text{EX}$, $\text{EG}$ and $\text{EU}$.
- one labelling procedure for each of those modalities.
Outline of the talk

1. Introduction

2. Computation Tree Logic
   - Definition and examples
   - Expressiveness of CTL
   - CTL model-checking

3. Alternating-time Temporal Logic
   - Why multi-agent systems?
   - Modelling multi-agent systems: ATMs and CGSs
   - Alternating-time Temporal Logic
   - ATL model-checking
   - Implicit CGSs

4. Conclusion
Why multi-agent systems?

Example

The fact that it is always possible to reach the ground floor is expressed in CTL as

\[ AG \ EF_{door_0}.is\ open \]

We rather meant

A user has a strategy for reaching the ground floor.
Why multi-agent systems?

Example

The fact that it is always possible to reach the ground floor is expressed in CTL as

\[ AG \ EF_{\text{door}_0 \text{.is open}} \]

We rather meant

A user has a strategy for reaching the ground floor.

We’d like to be able to reason about the ability of an agent to achieve some goal against a (hostile) environment.
Outline of the talk

1. Introduction

2. Computation Tree Logic
   - Definition and examples
   - Expressiveness of CTL
   - CTL model-checking

3. Alternating-time Temporal Logic
   - Why multi-agent systems?
   - Modelling multi-agent systems: ATSSs and CGSSs
   - Alternating-time Temporal Logic
   - ATL model-checking
   - Implicit CGSSs

4. Conclusion
Modelling multi-agent systems: ATSs

Definition (AHK97)

A multi-agent system can be modelled as an alternating transition system (ATS) $\mathcal{A} = \langle Q, \delta, \ell, \mathbb{A}, Mv \rangle$ s.t.:

- $\langle Q, \delta, \ell \rangle$ is a Kripke structure,
- $\mathbb{A} = \{ A_1, ..., A_p \}$ is a finite set of agents.
- $Mv: Q \times A \rightarrow P(P(\delta))$ defines, in each state and for each agent, a set of possible moves, with the following requirements:
  - $\forall q, \forall A_i, \forall m \in Mv(q, A_i). m \subseteq \delta \cap \{q\} \times Q$
  - $\forall q, \forall (m_{A_i})_{i \leq p}. m_{A_i} \in Mv(q, A_i)$ is a singleton.
Modelling multi-agent systems: ATSs

**Definition (AHK97)**

A multi-agent system can be modelled as an alternating transition system (ATS) \( \mathcal{A} = \langle Q, \delta, \ell, A, \text{Mv} \rangle \) s.t.:

- \( \langle Q, \delta, \ell \rangle \) is a Kripke structure,
- \( A = \{ A_1, \ldots, A_p \} \) is a finite set of agents,
- \( \text{Mv}: Q \times A \to \mathcal{P}(\mathcal{P}(\delta)) \) defines, in each state and for each agent, a set of possible moves, with the following requirements:

\[
\forall q, \forall A_i, \forall m \in \text{Mv}(q, A_i). \quad m \subseteq \delta \cap \{ q \} \times Q
\]

\[
\forall q, \forall (m_{A_i}) \text{ s.t. } m_{A_i} \in \text{Mv}(q, A_i). \quad \bigcap_{i \leq p} m_{A_i} \text{ is a singleton.}
\]
Modelling multi-agent systems: ATSSs

Example

\[ \mathbb{A} = \{\text{Prime, Even}\} \]

\[
\begin{align*}
\text{Mv}(q, \text{Prime}) &= \{\{q_2, q_3\}, \{q_1, q_4\}\} \\
\text{Mv}(q, \text{Even}) &= \{\{q_2, q_4\}, \{q_1, q_3\}\}
\end{align*}
\]
Modelling multi-agent systems: ATSS

Example

\[ \mathbb{A} = \{\text{Prime, Even}\} \]

\[ \text{Mv}(q, \text{Prime}) = \{\{q_2, q_3\}, \{q_1, q_4\}\} \]

\[ \text{Mv}(q, \text{Even}) = \{\{q_2, q_4\}, \{q_1, q_3\}\} \]
Modelling multi-agent systems: ATSS

Example

\[ \mathbb{A} = \{\text{Prime, Even}\} \]

\[ Mv(q, \text{Prime}) = \{\{q_2, q_3\}, \{q_1, q_4\}\} \]

\[ Mv(q, \text{Even}) = \{\{q_2, q_4\}, \{q_1, q_3\}\} \]
Strategies and outcomes

**Definition**

- Next\((q) = \{ q' \mid \exists (m_{A_i})_{i \text{ s.t.}} \forall i. \ m_{A_i} \in Mv(q, A_i) \text{ and } \bigcap_i m_{A_i} = \{ q \rightarrow q' \}\};\)
Strategies and outcomes

**Definition**

- \( \text{Next}(q) = \{ q' \mid \exists (m_{A_i}) \text{s.t.} \) \( \forall i. m_{A_i} \in \text{Mv}(q, A_i) \) \( \text{and } \bigcap_i m_{A_i} = \{q \rightarrow q'\}\);

- An **execution** of an ATS is an infinite sequence \( \rho = q_0 \ q_1 \ q_2 \ldots \) s.t. \( q_{j+1} \in \text{Next}(q_j) \) for all \( j \);
Strategies and outcomes

Definition

- \( \text{Next}(q) = \{ q' \mid \exists (m_{A_i}) \text{s.t.} \) \\
  \quad \forall i. \ m_{A_i} \in Mv(q, A_i) \) and \( \bigcap_i m_{A_i} = \{ q \to q' \} \); \\
- An execution of an ATS is an infinite sequence \( \rho = q_0 q_1 q_2 \ldots \) \\
  s.t. \( q_{j+1} \in \text{Next}(q_j) \) for all \( j \); \\
- A strategy for player \( A_i \) is a function \( f_{A_i} : Q^* \to \delta \) s.t. \\
  \forall q_0, q_1, \ldots, q_k. \ f_{A_i}(q_0, q_1, \ldots, q_k) \in Mv(q_k, A_i) \).
Strategies and outcomes

Definition

- **Next**($q$) = \{ $q'$ | \exists (m_{A_i})_i s.t.
  \[ \forall i. \text{ } m_{A_i} \in Mv(q, A_i) \text{ and } \bigcap_i m_{A_i} = \{ q \rightarrow q' \} \}; \]

- An **execution** of an ATS is an infinite sequence $\rho = q_0 q_1 q_2 \ldots$ s.t. $q_{j+1} \in \text{Next}(q_j)$ for all $j$;

- A **strategy** for player $A_i$ is a function $f_{A_i} : Q^* \rightarrow \delta$ s.t.
  \[ \forall q_0, q_1, \ldots, q_k. \text{ } f_{A_i}(q_0, q_1, \ldots, q_k) \in Mv(q_k, A_i). \]

- The **outcomes** of a strategy $f_{A_i}$ from $q$ are the executions $\rho = q_0 q_1 q_2 \ldots$ s.t. $q_0 = q$ and
  \[ \forall j. \text{ } (q_j, q_{j+1}) \in f_{A_i}(q_0, q_1, \ldots, q_j). \]
Strategies and outcomes

Definition

- Next(q) = { q' | ∃(mA_i) s.t.
  \[ \forall i. \ mA_i \in Mv(q, A_i) \text{ and } \bigcap_i mA_i = \{q \rightarrow q'\} \}; \]

- An execution of an ATS is an infinite sequence \( \rho = q_0 q_1 q_2 ... \) s.t. \( q_{j+1} \in \text{Next}(q_j) \) for all \( j \);

- A strategy for player \( A_i \) is a function \( f_{A_i} : Q^* \rightarrow \delta \) s.t.
  \[ \forall q_0, q_1, ..., q_k. \quad f_{A_i}(q_0, q_1, ..., q_k) \in Mv(q_k, A_i). \]

- The outcomes of a strategy \( f_{A_i} \) from \( q \) are the executions \( \rho = q_0 q_1 q_2 ... \) s.t. \( q_0 = q \) and
  \[ \forall j. \quad (q_j, q_{j+1}) \in f_{A_i}(q_0, q_1, ..., q_j). \]

Remark

The notion of strategy extend to coalitions of agent: a strategy for coalition \( \{A_{i_1}, ..., A_{i_n}\} \) is a set of individual strategies \( \{f_{A_{i_1}}, ..., f_{A_{i_n}}\} \).
Modelling multi-agent systems: ATSSs

Example

$\mathbb{A} = \{\text{Prime, Even}\}$

$Mv(q, \text{Prime}) = \{\{q_2, q_3\}, \{q_1, q_4\}\}$

$Mv(q, \text{Even}) = \{\{q_2, q_4\}, \{q_1, q_3\}\}$

$Mv(q_i, \ast) = \{\{q\}\}$
Example

\[ A = \{\text{Prime}, \text{Even}\} \]

\[
\begin{align*}
Mv(q, \text{Prime}) &= \{\{q_2, q_3\}, \{q_1, q_4\}\} \\
Mv(q, \text{Even}) &= \{\{q_2, q_4\}, \{q_1, q_3\}\} \\
Mv(q_i, \ast) &= \{\{q\}\}
\end{align*}
\]

Player Even has a strategy to never the same \(q_i\) twice consecutively:

\[
\begin{align*}
f_{\text{Even}}(\ldots, q_1, q) &= \{q_2, q_4\} \\
f_{\text{Even}}(\ldots, q_2, q) &= \{q_1, q_3\} \\
f_{\text{Even}}(\ldots, q_3, q) &= \{q_2, q_4\} \\
f_{\text{Even}}(\ldots, q_4, q) &= \{q_1, q_3\}
\end{align*}
\]
Modelling multi-agent systems: AT斯

Example

\[ \mathbb{A} = \{ \text{Prime, Even} \} \]

\[
\begin{align*}
\text{Mv}(q, \text{Prime}) &= \{ \{ q_2, q_3 \}, \{ q_1, q_4 \} \} \\
\text{Mv}(q, \text{Even}) &= \{ \{ q_2, q_4 \}, \{ q_1, q_3 \} \} \\
\text{Mv}(q_i, \ast) &= \{ \{ q \} \}
\end{align*}
\]

- Player Even has no strategy to never visit \( q_3 \) and \( q_4 \).
- Player Prime has no strategy to eventually visit \( q_3 \) or \( q_4 \).
Modelling multi-agent systems: ATSs

Example

\[
\mathbb{A} = \{\text{Prime, Even}\}
\]

\[
Mv(q, \text{Prime}) = \{\{q_2, q_3\}, \{q_1, q_4\}\}
\]

\[
Mv(q, \text{Even}) = \{\{q_2, q_4\}, \{q_1, q_3\}\}
\]

\[
Mv(q_i, \ast) = \{\{q\}\}
\]

Remarks

- Games played on ATSs are not determined.
Modelling multi-agent systems: ATSSs

Example

![ATSS Diagram]

$\mathcal{A} = \{\text{Prime, Even}\}$

$\text{Mv}(q, \text{Prime}) = \{\{q_2, q_3\}, \{q_1, q_4\}\}$

$\text{Mv}(q, \text{Even}) = \{\{q_2, q_4\}, \{q_1, q_3\}\}$

$\text{Mv}(q_i, *) = \{\{q\}\}$

Remarks

- Games played on ATSSs are not determined.
- Turn-based ATSSs are a special case of ATSSs where, in each state,
  - the moves of one of the players is a set of singletons,
  - the other players have only one move, containing all the possible transitions from the current state.
Definition (AHK02)

A multi-agent system can be modelled as an concurrent game structure (CGS) $\mathcal{C} = \langle Q, \delta, \ell, \mathbb{A}, M_v, Edg \rangle$ s.t.:

- $\langle Q, \delta, \ell \rangle$ is a Kripke structure;
Modelling multi-agent systems: CGSs

Definition (AHK02)

A multi-agent system can be modelled as an concurrent game structure (CGS) \( C = \langle Q, \delta, \ell, A, Mv, Edg \rangle \) s.t.:

- \( \langle Q, \delta, \ell \rangle \) is a Kripke structure;
- \( A = \{A_1, \ldots, A_p\} \) is a finite set of agents,
- \( Mv: Q \times A \rightarrow \mathcal{P}(\mathbb{Z}^+) \),
- \( Edg: Q \times \mathbb{Z}^+ \rightarrow \delta \) is the transition table.
Modelling multi-agent systems: CGSs

Example

\[
\langle 0, 0 \rangle, \langle 1, 1 \rangle, \langle 1, 0 \rangle, \langle 0, 1 \rangle
\]

<table>
<thead>
<tr>
<th>Even</th>
<th>Prime</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>q₁</td>
</tr>
<tr>
<td>1</td>
<td>q₃</td>
</tr>
</tbody>
</table>
Modelling multi-agent systems: CGSs

Example

- **Graphical Representation**
  - Nodes: \( q, q_1, q_2, q_3, q_4 \)
  - Edges: \( q \) to \( q_1, q_2, q_3, q_4 \)

- **Transition Table**
<table>
<thead>
<tr>
<th>( \text{Even} )</th>
<th>( \text{Prime} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( q_1 )</td>
</tr>
<tr>
<td>1</td>
<td>( q_3 )</td>
</tr>
</tbody>
</table>

\( \langle 0, 0 \rangle, \langle 1, 1 \rangle, \langle 1, 0 \rangle, \langle 0, 1 \rangle \)
Modelling multi-agent systems: CGSs

Example

$$\langle 0, 0 \rangle, \langle 1, 1 \rangle, \langle 1, 0 \rangle, \langle 0, 1 \rangle$$

<table>
<thead>
<tr>
<th>Prime</th>
<th>Even</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>$q_1$</td>
</tr>
<tr>
<td>1</td>
<td>$q_3$</td>
</tr>
<tr>
<td>1</td>
<td>$q_2$</td>
</tr>
</tbody>
</table>

$q_1 \rightarrow q \rightarrow q_2 \rightarrow q_3 \rightarrow q_4$
Modelling multi-agent systems: CGSs

Example

- States: $q_1, q_2, q_3, q_4$
- Transitions:
  - $q_1 \xrightarrow{0,0} q$
  - $q_2 \xrightarrow{1,1} q$
  - $q_3 \xrightarrow{1,0} q$
  - $q_4 \xrightarrow{0,1} q$

- Transition Table:

<table>
<thead>
<tr>
<th>Prime</th>
<th>Even</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>$q_1$</td>
</tr>
<tr>
<td>1</td>
<td>$q_3$</td>
</tr>
</tbody>
</table>
The notions of successor, execution, strategy and outcome are defined similarly to the case of ATSs.
How do ATSSs and CGSSs compare?

**Definition (AHKV98)**

Let $B_1$ and $B_2$ be two models of concurrent games over the same set $\mathbb{A}$ of agents. Then $R \subseteq Q_1 \times Q_2$ is an alternating bisimulation when, for any $(q, q') \in R$, the following conditions hold:

- $\ell_1(q) = \ell_2(q')$;
- for any coalition $A \subseteq \mathbb{A}$, we have
  \[
  \forall m : A \rightarrow Mv_1(q, A). \exists m' : A \rightarrow Mv_2(q', A).
  \]
  \[
  \forall r' \in \text{Next}_2(q', A, m'). \exists r \in \text{Next}_1(q, A, m). (r, r') \in R.
  \]
- symmetrically, for any coalition $A \subseteq \mathbb{A}$, we have
  \[
  \forall m' : A \rightarrow Mv_2(q', A). \exists m : A \rightarrow Mv_1(q, A).
  \]
  \[
  \forall r \in \text{Next}_1(q, A, m). \exists r' \in \text{Next}_2(q', A, m'). (r, r') \in R.
  \]
How do ATSSs and CGSSs compare?

Theorem

ATSs and CGSSs have the same expressive power w.r.t. alternating bisimulation.
How do ATSSs and CGSSs compare?

**Theorem**

ATSs and CGSs have the same expressive power w.r.t. alternating bisimulation.

**Proof.**

- translating an ATS into a CGS is easy.

**Example**

\[
\begin{align*}
\mathcal{M}_v(q, \text{Prime}) &= \left\{ \{q_2, q_3\}, \{q_1, q_4\} \right\} \\
\mathcal{M}_v(q, \text{Even}) &= \left\{ \{q_2, q_4\}, \{q_1, q_3\} \right\}
\end{align*}
\]
How do ATSs and CGSs compare?

**Theorem**

ATSs and CGSs have the same expressive power w.r.t. alternating bisimulation.

**Proof.**

- translating an ATS into a CGS is easy.

**Example**

\[
Mv(q, \text{Prime}) = \{\{q_2, q_3\}, \{q_1, q_4\}\}
\]

\[
Mv(q, \text{Even}) = \{\{q_2, q_4\}, \{q_1, q_3\}\}
\]
How do ATSSs and CGSSs compare?

**Theorem**

ATSs and CGSSs have the same expressive power w.r.t. alternating bisimulation.

**Proof.**

- translating an ATSS into a CGSS is easy.

**Example**

\[
\begin{align*}
Mv(q, \text{Prime}) &= \{\{q_2, q_3\}, \{q_1, q_4\}\} \\
Mv(q, \text{Even}) &= \{\{q_2, q_4\}, \{q_1, q_3\}\}
\end{align*}
\]
How do ATSs and CGSs compare?

**Theorem**

*ATSs and CGSs have the same expressive power w.r.t. alternating bisimulation.*

**Proof.**

- translating an ATS into a CGS is easy.

**Example**

\[
\begin{align*}
Mv(q, \text{Prime}) &= \{ \{q_2, q_3\}, \{q_1, q_4\} \} \\
Mv(q, \text{Even}) &= \{ \{q_2, q_4\}, \{q_1, q_3\} \}
\end{align*}
\]
How do ATSs and CGSs compare?

**Theorem**

ATSs and CGSs have the same expressive power w.r.t. alternating bisimulation.

**Proof.**

- translating an ATS into a CGS is easy.

**Example**

![Diagram of states and transitions](image)

\[ Mv(q, \text{Prime}) = \{\{q_2, q_3\}, \{q_1, q_4\}\} \]

\[ Mv(q, \text{Even}) = \{\{q_2, q_4\}, \{q_1, q_3\}\} \]
How do ATSSs and CGSSs compare?

**Theorem**

ATSs and CGSSs have the same expressive power w.r.t. alternating bisimulation.

**Proof.**

- translating an ATS into a CGS is easy.

**Example**

\[
M_v(q, \text{Prime}) = \{\{q_2, q_3\}, \{q_1, q_4\}\}
\]

\[
M_v(q, \text{Even}) = \{\{q_2, q_4\}, \{q_1, q_3\}\}
\]
How do ATSs and CGSs compare?

**Theorem**

*ATSs and CGSs have the same expressive power w.r.t. alternating bisimulation.*

**Proof.**
- translating an ATS into a CGS is easy.
- the other direction is more involved:

**Example**

[Diagram showing the states A, B, C, D with transitions labeled as pairs of numbers like ⟨1,2⟩, ⟨2,2⟩, etc.]
How do ATSs and CGSs compare?

**Theorem**

ATSs and CGSs have the same expressive power w.r.t. alternating bisimulation.

**Proof.**

- translating an ATS into a CGS is easy.
- the other direction is more involved:

**Example**

**Naive approach**

<table>
<thead>
<tr>
<th>Move</th>
<th>Player 1</th>
<th>Player 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{B, D}</td>
<td>{A, B, D}</td>
</tr>
<tr>
<td>2</td>
<td>{C, D}</td>
<td>{C, D}</td>
</tr>
<tr>
<td>3</td>
<td>{A, D}</td>
<td>{C, D}</td>
</tr>
</tbody>
</table>
How do ATSs and CGSs compare?

**Theorem**

ATSs and CGSs have the same expressive power w.r.t. alternating bisimulation.

**Proof.**

- translating an ATS into a CGS is easy.
- the other direction is more involved:

**Example**

![Diagram](image)

**Correct approach**

<table>
<thead>
<tr>
<th>Move</th>
<th>Player 1</th>
<th>Player 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{B_{1.1}, D_{1.2}, D_{1.3}}</td>
<td>{A_{3.1}, B_{1.1}, D_{2.1}}</td>
</tr>
<tr>
<td>2</td>
<td>{C_{2.2}, C_{2.3}, D_{2.1}}</td>
<td>{C_{2.2}, D_{1.2}, D_{3.2}}</td>
</tr>
<tr>
<td>3</td>
<td>{A_{3.1}, D_{3.2}, D_{3.3}}</td>
<td>{C_{2.3}, D_{1.3}, D_{3.3}}</td>
</tr>
</tbody>
</table>
How do ATSSs and CGSSs compare?

**Theorem**

*ATSs and CGSSs have the same expressive power w.r.t. alternating bisimulation.*

**Proof.**

- translating an ATS into a CGS is easy.
- the other direction is more involved.

**Remark**

This translation of an ATS into a CGS yields an exponential blowup.
This translation of a CGS into an ATS is quadratic.
Outline of the talk

1. Introduction

2. Computation Tree Logic
   - Definition and examples
   - Expressiveness of CTL
   - CTL model-checking

3. Alternating-time Temporal Logic
   - Why multi-agent systems?
   - Modelling multi-agent systems: ATSSs and CGSSs
   - Alternating-time Temporal Logic
   - ATL model-checking
   - Implicit CGSSs

4. Conclusion
Alternating-time Temporal Logic

Definition (Syntax of ATL [AHK97, AHK02])

$$\text{ATL } \ni \phi_s ::= p \mid \neg \phi_s \mid \phi_s \lor \phi_s \mid \langle A \rangle \phi_p$$

$$\phi_p ::= X \phi_s \mid G \phi_s \mid \phi_s U \phi_s$$

where $p$ ranges over $2^{\text{AP}}$, and $A$ ranges over $2^A$. 
Alternating-time Temporal Logic

**Definition (Syntax of ATL [AHK97,AHK02])**

\[
\text{ATL} \ni \phi_s ::= p \mid \neg \phi_s \mid \phi_s \lor \phi_s \mid \langle A \rangle \phi_p
\]

\[
\phi_p ::= X \phi_s \mid G \phi_s \mid \phi_s U \phi_s
\]

where \( p \) ranges over \( 2^{\text{AP}} \), and \( A \) ranges over \( 2^{\text{A}} \).

**Definition (Semantics of ATL)**

The semantics is similar to that of CTL, except:

\[
\mathcal{B},q \models \langle A \rangle \phi_p \iff \exists f_A \in \text{Strategy}(A).
\]

\[
\forall \rho \in \text{Outcomes}(q,f_A). \mathcal{B},\rho \models \phi_p
\]
Alternating-time Temporal Logic

**Definition (Semantics of ATL)**

The semantics is similar to that of CTL, except:

\[
B, q \models \langle A \rangle \phi_p \iff \exists f_A \in \text{Strategy}(A).
\]

\[
\forall \rho \in \text{Outcomes}(q, f_A). B, \rho \models \phi_p
\]

**ATL subsumes CTL:**

\[
E \phi_p \equiv \langle A \rangle \phi_p
\]

\[
A \phi_p \equiv \langle \emptyset \rangle \phi_p
\]
Theorem

\[ \langle A \rangle \phi_s W \psi_s \text{ cannot be expressed in ATL, where } \]
\[ \phi_s W \psi_s \overset{\text{def}}{=} \phi_s U \psi_s \lor G \psi_s. \]
Expressiveness of ATL

**Theorem**

\( \langle A \rangle \phi_s W \psi_s \) cannot be expressed in ATL, where

\[ \phi_s W \psi_s \overset{\text{def}}{=} \phi_s U \psi_s \lor G \psi_s. \]

This is surprising since, in CTL, we have:

\[
E \phi W \psi \equiv E \phi U \psi \lor EG \phi
\]

\[
A \phi W \psi \equiv \neg E(\neg \psi) U (\neg \phi \land \neg \psi).
\]

But:

\( \langle A \rangle (G \phi \lor \phi U \psi) \) is not an ATL formula

\( \langle A \rangle \phi U \psi \lor \langle A \rangle G \phi \not\equiv \langle A \rangle \phi W \psi. \)
Expressiveness of ATL

Theorem

$\langle A \rangle \phi_s W \psi_s$ cannot be expressed in ATL.

Proof.

\[ S_1' \]

\[ (3.1), (4.2) \]

\[ a \]

\[ (1.2), (1.3), (2.1), (3.2), (3.3) \]

\[ b \]

\[ a_i \]

\[ (2.2) \]

\[ (2.3) \]

\[ (4.3) \]

\[ s_{i-1}' \]

\[ (1.1) \]

\[ a \]

\[ (1.1) \]

\[ (2.2) \]

\[ (2.3) \]

\[ (1.1) \]

\[ b \]

\[ a_{i-1} \]

\[ a \]

\[ (2.2) \]

\[ (2.3) \]

\[ (1.1) \]

\[ s_i \]

\[ (3.1), (4.2) \]

\[ a \]

\[ (2.2) \]

\[ (2.3) \]

\[ (1.1) \]

\[ b \]

\[ a_1 \]

\[ (1.2), (1.3), (2.1), (3.2), (3.3) \]

\[ b \]

\[ a_1 \]

\[ (2.2) \]

\[ (2.3) \]

\[ (1.1) \]

\[ a \]

\[ (3.1) \]

\[ s_1 \]

\[ (1.2), (1.3), (2.1), (3.2), (3.3) \]

\[ a \]

\[ (3.1) \]

\[ s_1 \]

\[ (1.2), (1.3), (2.1), (3.2), (3.3) \]

\[ \neg a, \neg b \]
Expressiveness of ATL

Theorem

\[ \langle A \rangle \phi_s \mathbf{W} \psi_s \text{ cannot be expressed in ATL.} \]

In the sequel, we use the following definition of ATL:

Definition

\[
\begin{align*}
\text{ATL} \ni \phi_s & ::= p \mid \neg \phi_s \mid \phi_s \lor \phi_s \mid \langle A \rangle \phi_p \\
\phi_p & ::= X\phi_s \mid \phi_s \mathbf{W} \psi_s \mid \phi_s \mathbf{U} \phi_s
\end{align*}
\]

where \( p \) ranges over \( 2^{\text{AP}} \), and \( A \) ranges over \( 2^{\text{A}} \).

Equivalently, we could have defined and added:

\[
[A] \phi_s \overset{\text{def}}{=} \neg \langle A \rangle \neg \phi_s.
\]
Outline of the talk

1. Introduction

2. Computation Tree Logic
   - Definition and examples
   - Expressiveness of CTL
   - CTL model-checking

3. Alternating-time Temporal Logic
   - Why multi-agent systems?
   - Modelling multi-agent systems: ATSSs and CGSSs
   - Alternating-time Temporal Logic
   - ATL model-checking
   - Implicit CGSSs

4. Conclusion
<table>
<thead>
<tr>
<th>Model-checking ATL</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Theorem (AHK02)</strong></td>
</tr>
<tr>
<td>ATL model-checking over CGSs is PTIME-complete.</td>
</tr>
</tbody>
</table>
Theorem (AHK02)

ATL model-checking over CGSs is PTIME-complete.

Proof.

- recursive labeling algorithm, similar to that of CTL;
Model-checking ATL

Theorem (AHK02)

ATL model-checking over CGSs is PTIME-complete.

Proof.

- recursive labeling algorithm, similar to that of CTL;
- \( \text{CPre}(A, L) = \{ q \mid \exists m_A \cdot \text{Next}(q, A, m_A) \subseteq L \} \)
Model-checking ATL

Theorem (AHK02)

ATL model-checking over CGSs is PTIME-complete.

Proof.

- recursive labeling algorithm, similar to that of CTL;
- \( \text{CPre}(A, L) = \{ q \mid \exists m_A \cdot \text{Next}(q, A, m_A) \subseteq L \} \)
- labeling e.g. with \( \langle A \rangle \phi_s U \psi_s \):
  - first label the states satisfying \( \psi_2 \);
  - then compute the controllable predecessors of the labeled states, and label those satisfying \( \phi_s \), until the fixpoint is reached.
Model-checking ATL

Theorem (AHK02)

ATL model-checking over CGSs is PTIME-complete.

Proof.

- recursive labeling algorithm, similar to that of CTL;
- $\text{CPre}(A, L) = \{ q \mid \exists m_A \cdot \text{Next}(q, A, m_A) \subseteq L \}$
- labeling e.g. with $\langle A \rangle \phi_s U \psi_s$:
  - first label the states satisfying $\psi_2$;
  - then compute the controllable predecessors of the labeled states, and label those satisfying $\phi_s$, until the fixpoint is reached.

Remark

This algorithm runs in time $O(|\phi| \times |\text{Edg}|)$. 
ATL model-checking over ATSs is PTIME-complete.
Model-checking ATL

**Theorem (AHK97)**

*ATL model-checking over ATSs is PTIME-complete.*

This theorem is not correct: the algorithm (similar to the previous one) is linear **only in the size of the underlying CGS**.
Theorem (AHK97)

ATL model-checking over ATSs is PTIME-complete.

This theorem is not correct: the algorithm (similar to the previous one) is linear only in the size of the underlying CGS.

Theorem (AHK97, corrected)

ATL model-checking over ATSs is PTIME-complete in the following cases:

- when the number of player is fixed;
- when the ATS is turn-based.
Theorem

\[ \text{ATL model-checking over ATSSs is } \Delta_2^P\text{-complete}. \]
Model-checking ATL

**Theorem**

*ATL model-checking over ATSs is $\Delta^P_2$-complete.*

\[
\begin{align*}
\Delta^P_3 &= \text{PTIME}^{\Sigma^P_2} \\
\Sigma^P_2 &= \text{NP}^{\Sigma^P_1} \\
\Pi^P_2 &= \text{coNP}^{\Sigma^P_1} \\
\Delta^P_2 &= \text{PTIME}^{\Sigma^P_1} \\
\Sigma^P_1 &= \text{NP} \\
\Pi^P_1 &= \text{coNP}
\end{align*}
\]

\[\text{PTIME}^{\Sigma^P_1} = \text{PTIME}^{\text{NP}} = \text{can be computed in polynomial time by a Turing machine having access to an NP oracle.}\]
Model-checking ATL

**Theorem**

ATL model-checking over ATSs is $\Delta^P_2$-complete.

**Proof.**

- $\Delta^P_2$-algorithm: computing $CPre$ can be achieved in NP.
Model-checking ATL

Theorem

ATL model-checking over ATSs is $\Delta_2^P$-complete.

Proof.

- $\Delta_2^P$-algorithm: computing $\text{CPre}$ can be achieved in NP.
- We sketch the proof of NP-hardness, using 3SAT:
Model-checking ATL

Theorem

ATL model-checking over ATSs is $\Delta_2^P$-complete.

Proof.

- $\Delta_2^P$-algorithm: computing $\text{CPre}$ can be achieved in NP.
- We sketch the proof of NP-hardness, using 3SAT:

\[
C = p \lor \neg q \lor r \quad \leadsto \quad \begin{cases}
    c_0 = \neg p \lor \neg q \lor \neg r \\
    c_1 = \neg p \lor \neg q \lor r \\
    c_2 = \neg p \lor q \lor \neg r \\
    c_3 = \neg p \lor q \lor r \\
    c_4 = p \lor \neg q \lor \neg r \\
    c_5 = p \lor \neg q \lor r \\
    c_6 = p \lor q \lor \neg r \\
    c_7 = p \lor q \lor r
\end{cases}
\]
Model-checking ATL

Theorem

ATL model-checking over ATSs is $\Delta^P_2$-complete.
Model-checking ATL

Theorem

ATL model-checking over ATSs is $\Delta^P_2$-complete.

1 player ($P_1$ to $P_k$) per atomic proposition:

- $P_l \leadsto \{ c^i_j | c^i_j \text{ not made true by } P_l \}$
- $\neg P_l \leadsto \{ c^i_j | c^i_j \text{ not made true by } \neg P_l \}$
Model-checking ATL

Theorem

ATL model-checking over ATSs is $\Delta^P_2$-complete.

1 player ($P_1$ to $P_k$) per atomic proposition:

- $P_l \rightsquigarrow \{ c^i_j \mid c^i_j \text{ not made true by } P_l \}$
- $\neg P_l \rightsquigarrow \{ c^i_j \mid c^i_j \text{ not made true by } \neg P_l \}$

Once those players have chosen their moves, exactly one clause $c^i_j$ per original clause $C^i$ belongs to the intersection of the chosen sets.

\[
\begin{align*}
p &= \top \\
q &= \top \\
r &= \bot
\end{align*}
\]

E.g. $\Rightarrow \neg p \lor \neg q \lor r$

\[\begin{array}{c}
C^1 \\
C^2 \\
\vdots \\
C^n
\end{array} \begin{array}{c}
c^1_0 \\
c^1_1 \\
c^1_2 \\
c^1_3 \\
c^1_4 \\
c^1_5 \text{ yellow} \\
c^1_6 \\
c^1_7 \\
\vdots \\
c^n_0 \\
c^n_1 \\
c^n_2 \\
c^n_3 \text{ yellow} \\
c^n_4 \\
c^n_5 \\
c^n_6 \\
c^n_7 \\
\end{array} \]

(q0)
Model-checking ATL

Theorem

**ATL model-checking over ATSSs is $\Delta^P_2$-complete.**

1 player ($P_1$ to $P_k$) per atomic proposition:
- $P_l \leadsto \{ c^i_j \mid c^i_j \text{ not made true by } P_l \}$
- $\neg P_l \leadsto \{ c^i_j \mid c^i_j \text{ not made true by } \neg P_l \}$

1 extra player chooses one set among \( \{c^1_0, \ldots, c^1_7\} \) to \( \{c^n_0, \ldots, c^n_7\} \)
Model-checking ATL

**Theorem**

ATL model-checking over ATSSs is \(\Delta_2^P\)-complete.

1 player (\(P_1\) to \(P_k\)) per atomic proposition:
- \(P_l \leadsto \{ c^i_j \mid c^i_j \text{ not made true by } P_l \}\)
- \(\neg P_l \leadsto \{ c^i_j \mid c^i_j \text{ not made true by } \neg P_l \}\)

1 extra player chooses one set among \(\{c^1_0, ..., c^1_7\}\) to \(\{c^n_0, ..., c^n_7\}\)

**Lemma**

The 3-SAT instance is true iff

\[ q_0 \models \langle P_1, ..., P_k \rangle \mathbf{X} \neg \]
Outline of the talk

1. Introduction

2. Computation Tree Logic
   - Definition and examples
   - Expressiveness of CTL
   - CTL model-checking

3. Alternating-time Temporal Logic
   - Why multi-agent systems?
   - Modelling multi-agent systems: ATs and CGs
   - Alternating-time Temporal Logic
   - ATL model-checking
   - Implicit CGs

4. Conclusion
Implicit CGSs

Definition

An implicit CGS is a CGS where

- The transition function: in each $q$, it is given
  $$((\phi_0, q_0), \cdots, (\phi_n, q_n))$$ where $q_i \in Q$, $\phi_i$ is a boolean combination of propositions $m_{A_j} = c$, and $\phi_n = \top$;

- $\text{Edg}(q, m_{A_1}, \cdots, m_{A_k}) = q_j$ s.t.
  $$j = \min \{ i \mid \phi_i(q, m_{A_1}, \cdots, m_{A_k}) = \top \}$$
Implicit CGSs

**Definition**

An **implicit CGS** is a CGS where

- The transition function: in each \( q \), it is given 
  \(((\phi_0, q_0), \cdots, (\phi_n, q_n))\) where \( q_i \in Q \), \( \phi_i \) is a boolean combination of propositions \( m_{A_j} = c \), and \( \phi_n = \top \);
- \( \text{Edg}(q, m_{A_1}, \cdots, m_{A_k}) = q_j \) s.t.
  \[
  j = \min\{i \mid \phi_i(q, m_{A_1}, \cdots, m_{A_k}) = \top\}
  \]

**Theorem**

**ATL model-checking on implicit CGSs is \( \Delta^P_3 \)-complete.**
Implicit CGSs

Definition

An **implicit CGS** is a CGS where

- The transition function: in each $q$, it is given
  $((\phi_0, q_0), \cdots, (\phi_n, q_n))$ where $q_i \in Q$, $\phi_i$ is a boolean combination of propositions $m_{A_j} = c$, and $\phi_n = \top$;

- $\text{Edg}(q, m_{A_1}, \cdots, m_{A_k}) = q_j$ s.t.
  $$j = \min \{i \mid \phi_i(q, m_{A_1}, \cdots, m_{A_k}) = \top\}$$

[Diagram showing relationships between explicit and implicit CGSs and ATS with annotations for exponential, quadratic, linear, and cubic.]


**Conclusion**

- **multi-agent systems:**
  - nice framework for modeling the interactions of several agents acting on a system;
  - useful for checking controllability and synthesizing controllers;
  - several different models.

- **alternating-time temporal logic:**
  - extension of CTL for dealing with strategies;
  - reasonable complexity, exponential in the number of agents.
Conclusion

- **multi-agent systems:**
  - nice framework for modeling the interactions of several agents acting on a system;
  - useful for checking controllability and synthesizing controllers;
  - several different models.

- **alternating-time temporal logic:**
  - extension of CTL for dealing with strategies;
  - reasonable complexity, exponential in the number of agents.

**Future works:**
- extending ATL with fairness: EATL, EATL⁺;
- timed models for multi-agent systems;
- timed extensions of ATL.