Temporal logics for multi-agent systems

Nicolas Markey
CNRS – IRISA (Rennes, France)

(based on joint works with Patricia Bouyer, Patrick Gardy, and François Laroussinie)

MFCS’17 – August 23, 2017
Model checking and synthesis

system:

property:

model-checking algorithm

yes/no

\[ AG(\neg B.\text{overfull} \land \neg B.\text{dried\_up}) \]

[http://www.embedded.com]
Model checking and synthesis

system:

[http://www.embedded.com]

property:

\[ A \Box (\neg B.\text{overfull} \land \neg B.\text{dried\_up}) \]

synthesis algorithm

\[ b! \rightarrow a? \]

\[ a! \rightarrow b? \]
Computation-Tree Logic (CTL)

- atomic propositions: 0, 1, ...

Examples

- safety: $\text{AG safe}$ states will be reached

- fairness: $\text{AG AF}$ states will be visited infinitely many times

- request-response: $\text{AG (request } \Rightarrow \text{EF grant)}$ any request can eventually be granted
Computation-Tree Logic (CTL)

- atomic propositions: \( \bigcirc, \bigcirc, \ldots \)
- boolean combinators: \( \neg \varphi, \varphi \lor \psi, \varphi \land \psi, \ldots \)

Examples

- safety: \( \text{AG} \) safe \\
  only safe states will be reached

- fairness: \( \text{AG} \, \text{AF} \) fair \\
  fair states will be visited infinitely many times

- request-response: \( \text{AG} \, (\text{request} \rightarrow \text{EF} \, \text{grant}) \) \\
  any request can eventually be granted
Computation-Tree Logic (CTL)

- atomic propositions: "", "", ...
- boolean combinators: \( \neg \varphi \), \( \varphi \lor \psi \), \( \varphi \land \psi \), ...
- temporal modalities:
  - \( X \varphi \)
  - \( \varphi U \psi \)

```
safety: AG safe only safe states will be reached

fairness: AG AF fair states will be visited infinitely many times

request-response: AG (request \( \Rightarrow \) EF grant)
```

"next \( \varphi \)"

"\( \varphi \) until \( \psi \)"
Computation-Tree Logic (CTL)

- atomic propositions: \( \bigcirc, \bigcirc, \ldots \)
- boolean combinators: \( \neg \varphi, \varphi \lor \psi, \varphi \land \psi, \ldots \)
- temporal modalities:
  - \( X \varphi \) \( \bigcirc \rightarrow \varphi \rightarrow \varphi \rightarrow \varphi \rightarrow \bigcirc \) \( \ldots \) “next \( \varphi \)”
  - \( \varphi \mathbf{U} \psi \) \( \varphi \rightarrow \varphi \rightarrow \psi \rightarrow \varphi \rightarrow \bigcirc \) \( \ldots \) “\( \varphi \) until \( \psi \)”
  - \( \text{true} \mathbf{U} \varphi \equiv \mathbf{F} \varphi \) \( \bigcirc \rightarrow \varphi \rightarrow \varphi \rightarrow \varphi \rightarrow \varphi \rightarrow \bigcirc \) \( \ldots \) “eventually \( \varphi \)”
  - \( \neg \mathbf{F} \neg \varphi \equiv \mathbf{G} \varphi \) \( \varphi \rightarrow \varphi \rightarrow \varphi \rightarrow \varphi \rightarrow \varphi \rightarrow \bigcirc \) \( \ldots \) “always \( \varphi \)”

Examples
- safety: \( \mathbf{A}G \text{safe} \) only safe states will be reached
- fairness: \( \mathbf{A}G \mathbf{A}F \) fair states will be visited infinitely many times
- request-response: \( \mathbf{A}R \Rightarrow \mathbf{E}F \) any request can eventually be granted
Computation-Tree Logic (CTL)

- **atomic propositions:** \( \bigcirc, \bigcirc, \ldots \)
- **boolean combinators:** \( \neg \varphi, \varphi \lor \psi, \varphi \land \psi, \ldots \)
- **temporal modalities:**
  - \( X \varphi \) - “next \( \varphi \)”
  - \( \varphi \mathcal{U} \psi \) - “\( \varphi \) until \( \psi \)”
  - \( \text{true} \mathcal{U} \varphi \equiv \mathcal{F} \varphi \) - “eventually \( \varphi \)”
  - \( \neg \mathcal{F} \neg \varphi \equiv \mathcal{G} \varphi \) - “always \( \varphi \)”

- **path quantifiers:**
Computation-Tree Logic (CTL)

- atomic propositions: \( \bigcirc, \bigcirc, \ldots \)
- boolean combinators: \( \neg \varphi, \varphi \lor \psi, \varphi \land \psi, \ldots \)
- temporal modalities:
  - \( X \varphi \) "next \( \varphi \)"
  - \( \varphi U \psi \) "\( \varphi \) until \( \psi \)"
  - \( \varphi \) "eventually \( \varphi \)"
  - \( \neg F \neg \varphi \) \( \equiv \) \( G \varphi \) "always \( \varphi \)"

Path quantifiers:
- \( E \varphi \)
- \( A \varphi \)

Examples

- **safety**: \( AG \text{ safe} \)
  - only safe states will be reached
- **fairness**: \( AG AF \text{ fair} \)
  - fair states will be visited infinitely many times
- **request-response**: \( AG(request \Rightarrow EF \text{ grant}) \)
  - any request can eventually be granted
In CTL, each temporal modality is in the immediate scope of a path quantifier.
In CTL, each temporal modality is in the immediate scope of a path quantifier.

\( EF \) is reachable
In CTL, each temporal modality is in the immediate scope of a path quantifier.

\[ EF \] is reachable
CTL model checking

In CTL, each temporal modality is in the immediate scope of a path quantifier.

\[ \text{EG}(\neg \quad \land \quad \text{EF} \quad ) \] there is a path along which \( \quad \) is always reachable, but never reached
In CTL, each temporal modality is in the immediate scope of a path quantifier.

$$\text{EG}(\neg \square_p \land \text{EF} \bigcirc_{p})$$

there is a path along which $\bigcirc_{p}$ is always reachable, but never reached.
In CTL, each temporal modality is in the immediate scope of a path quantifier.

\[ \text{EG}(\neg p \land \text{EF} p) \] there is a path along which \( p \) is always reachable, but never reached.
CTL model checking

In CTL, each temporal modality is in the immediate scope of a path quantifier.

**Theorem ([CE81, QS82])**

*CTL model checking is PTIME-complete.*

[CE81] Clarke, Emerson. Design and Synthesis of Synchronization Skeletons... LOP, 1981.
CTL model checking

In CTL, each temporal modality is in the immediate scope of a path quantifier.

Theorem ([CE81,QS82])

CTL model checking is PTIME-complete.

Remark ([KVW94])

CTL model checking on product structures is PSPACE-complete.

[CE81] Clarke, Emerson. Design and Synthesis of Synchronization Skeletons... LOP, 1981.
Outline of the presentation

1. Introduction

2. Alternating-time Temporal Logic
   - From ATL to ATL with strategy contexts
   - Model checking ATL_{sc}

3. Strategy Logic
   - Explicitly handling strategies
   - Semantics of SL

4. Conclusions and future works
Outline of the presentation

1. Introduction

2. Alternating-time Temporal Logic
   • From ATL to ATL with strategy contexts
   • Model checking ATL_{sc}

3. Strategy Logic
   • Explicitly handling strategies
   • Semantics of SL

4. Conclusions and future works
Games on graphs

Concurrent games

A concurrent game is made of

- a transition system;
Games on graphs

**Concurrent games**

A concurrent game is made of
- a transition system;
- a set of agents (or players);

---

![Diagram of a concurrent game](image.png)
Games on graphs

Concurrent games

A concurrent game is made of:

- a transition system;
- a set of agents (or players);
- a table indicating the transition to be taken given the actions of the players.

\[
\begin{array}{c|c|c}
\text{player 1} & q_0 & q_2 & q_1 \\
\hline
q_1 & \begin{array}{c}
\text{player 2}
\end{array} & q_0 & q_2 \\
q_0 & q_1 & q_0 & q_2 \\
q_2 & q_1 & q_0 & q_0 \\
\end{array}
\]
Games on graphs

Concurrent games

A concurrent game is made of

- a transition system;
- a set of agents (or players);
- a table indicating the transition to be taken given the actions of the players.

Turn-based games

A turn-based game is a game where only one agent plays at a time.
**Strategies**

A **(pure) strategy** for a given player is a function telling which action to play depending on what has happened previously.
Reasoning about open systems

Strategies

A (pure) strategy for a given player is a function telling which action to play depending on what has happened previously.

Example
Reasoning about open systems

**Strategies**

A *(pure) strategy* for a given player is a function telling which action to play depending on what has happened previously.

**Example**

Strategy for player 🟢
Reasoning about open systems

Strategies

A (pure) strategy for a given player is a function telling which action to play depending on what has happened previously.

Example

Strategy for player

alternately go to \( \bigcirc \) and \( \bigcirc \) (starting with \( \bigcirc \)).
Reasoning about open systems

**Strategies**

A **(pure) strategy** for a given player is a function telling which action to play depending on what has happened previously.

**Example**

Strategy for player

Alternately go to blue and green (starting with blue).
Reasoning about open systems

**Strategies**

A (pure) strategy for a given player is a function telling which action to play depending on what has happened previously.

**Example**

Strategy for player alternately go to \(\square\) and \(\bigcirc\) (starting with \(\bigcirc\)).
Reasoning about open systems

**Strategies**

A (pure) strategy for a given player is a function telling which action to play depending on what has happened previously.

**Example**

Strategy for player

alternately go to \(\bullet\) and \(\circ\) (starting with \(\bullet\)).
Temporal logics for games: ATL [AHK02]

<table>
<thead>
<tr>
<th>ATL extends CTL with <strong>strategy quantifiers</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\langle A \rangle \varphi$ expresses that $A$ has a strategy to enforce $\varphi$.</td>
</tr>
</tbody>
</table>

Temporal logics for games: ATL [AHK02]

ATL extends CTL with strategy quantifiers

\[ \langle A \rangle \varphi \] expresses that \( A \) has a strategy to enforce \( \varphi \).

Semantics of \( \langle A \rangle \varphi \)

Existential quantification (over strategies) implicitly includes a universal quantification (over outcomes):

\[ G, \bigcirc \models \langle A \rangle \varphi \iff \exists \sigma_A. \forall \pi \in \text{Out}(\bigcirc, \sigma_A). \pi \models \varphi. \]
Temporal logics for games: ATL [AHK02]

ATL extends CTL with strategy quantifiers

\(\langle A \rangle \varphi\) expresses that A has a strategy to enforce \(\varphi\).

Temporal logics for games: ATL [AHK02]

ATL extends CTL with strategy quantifiers

$\langle A \rangle \varphi$ expresses that $A$ has a strategy to enforce $\varphi$. 

Model checking ATL is PTIME-complete.

Temporal logics for games: ATL [AHK02]

ATL extends CTL with **strategy quantifiers**

\[\langle A \rangle \varphi\] expresses that \(A\) has a strategy to enforce \(\varphi\).

Temporal logics for games: ATL [AHK02]

ATL extends CTL with strategy quantifiers

\(\langle A \rangle \varphi\) expresses that A has a strategy to enforce \(\varphi\).
Temporal logics for games: ATL [AHK02]

ATL extends CTL with strategy quantifiers

\( \langle A \rangle \varphi \) expresses that A has a strategy to enforce \( \varphi \).

\begin{itemize}
  \item \( \langle \bigcirc \rangle F \)
  \item \( \langle \square \rangle F \)
  \item \( \langle \bigcirc \rangle G(\langle \square \rangle F) \)
\end{itemize}

Temporal logics for games: ATL [AHK02]

ATL extends CTL with strategy quantifiers

\[ \langle A \rangle \varphi \] expresses that A has a strategy to enforce \( \varphi \).

\[ \langle \bigcirc \rangle F \]
\[ \langle \Box \rangle F \]
\[ \langle \bigcirc \rangle G \langle \Box \rangle F \equiv \langle \bigcirc \rangle G p \]

Temporal logics for games: ATL \cite{AHK02}

ATL extends CTL with strategy quantifiers

$$\langle A \rangle \varphi$$ expresses that \( A \) has a strategy to enforce \( \varphi \).

\[\begin{align*}
\langle 0 \rangle \ F \ \circ \\
\langle [] \rangle \ F \ \bigcirc \\
\langle 0 \rangle \ G (\langle [] \rangle \ F \ \bigcirc) \equiv \langle 0 \rangle \ G \ p
\end{align*}\]

\cite{AHK02} Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.
Temporal logics for games: ATL [AHK02]

ATL extends CTL with strategy quantifiers

\( \langle A \rangle \varphi \) expresses that A has a strategy to enforce \( \varphi \).

Theorem ([AHK02])

Model checking ATL is PTIME-complete.

Temporal logics for games: ATL [AHK02]

ATL extends CTL with strategy quantifiers

$\langle A \rangle \varphi$ expresses that $A$ has a strategy to enforce $\varphi$.

Theorem ([AHK02])

Model checking ATL is PTIME-complete.

Remark ([LMO08])

In PTIME only if the transition table is given explicitly
(size $|\text{Moves}| |\text{Agt}|$)

ATL with strategy contexts [DLM10]

Example

\[ (\circ)^{\square} G (\square F \circ) \]
ATL with strategy contexts [DLM10]

Example

[Da Costa, Laroussinie, Markey. ATL with strategy contexts: expressiveness and ... FSTTCS, 2010]
Player \( \bigcirc \) in \( \bigcirc \) always plays to \( \square \).

\[
\langle \square \rangle G(\langle \square \rangle F \bigcirc)
\]

[Da Costa, Laroussinie, Markey. ATL with strategy contexts: expressiveness and ... FSTTCS, 2010]
ATL with strategy contexts [DLM10]

Example

Player \(\bigcirc\) in \(\bigcirc\) always plays to \(\square\).

\(\langle \bigcirc \rangle G (\langle \Box \rangle F \bigcirc)\)

Player \( \bigcirc \) in \( \bigcirc \) always plays to \( \square \).

Example

- Player $\bigcirc$ always plays to $\square$;
- Player $\square$ then plays to $\bigcirc$.

[DaCosta, Laroussinie, Markey. ATL with strategy contexts: expressiveness and ... FSTTCS, 2010]
ATL with strategy contexts [DLM10]

Definition

$\text{ATL}_{sc}$ has a new strategy quantifier:

- $\langle \cdot A \cdot \rangle \varphi$ is similar to $\langle A \rangle \varphi$ but assigns the corresponding strategy to $A$ for evaluating $\varphi$;

ATL with strategy contexts [DLM10]

Definition

\( \text{ATL}_{sc} \) has a **new strategy quantifier**:

- \( \langle \cdot A \cdot \rangle \varphi \) is similar to \( \langle A \rangle \varphi \) but **assigns** the corresponding strategy to \( A \) for evaluating \( \varphi \).

Definition

**Semantics of ATL strategy quantifier:**

\[ G, \bigcirc \models \langle A \rangle \varphi \iff \exists \sigma_A. \forall \pi \in \text{Out}(\bigcirc, \sigma_A). \pi \models \varphi \]
Definition

ATL\textsubscript{sc} has a new strategy quantifier:

- \langle \cdot A \cdot \rangle \varphi \text{ is similar to } \langle A \rangle \varphi \text{ but assigns the corresponding strategy to } A \text{ for evaluating } \varphi; 

Definition

Semantics of ATL strategy quantifier:
\[ G, \circ \models \langle A \rangle \varphi \iff \exists \sigma_A. \forall \pi \in \text{Out}(\circ, \sigma_A). \pi \models \varphi \]

Semantics of ATL\textsubscript{sc} strategy quantifier:
\[ G, \circ \models_{\sigma_B} \langle A \rangle \varphi \iff \exists \sigma_A. \forall \pi \in \text{Out}(\circ, \sigma_A \cup \sigma_B). \pi \models_{\sigma_A \cup \sigma_B} \varphi \]

Expressiveness of $\text{ATL}_{sc}$

- **Client-server interactions** for accessing a shared resource:

$$
\langle \cdot \text{Server} \cdot \rangle \quad G \quad \land \quad \bigwedge_{c \in \text{Clients}} \langle \cdot c \cdot \rangle \quad F \quad \text{access}_c \\
\neg \bigwedge_{c \neq c'} \text{access}_c \land \text{access}_{c'}
$$
Expressiveness of ATL\(_{sc}\)

- **Client-server interactions** for accessing a shared resource:

\[
\langle \text{Server} \rangle \ G \left[ \bigwedge_{c \in \text{Clients}} \langle c \rangle F \text{access}_c \land \neg \bigwedge_{c \neq c'} \text{access}_c \land \text{access}_{c'} \right]
\]

- **Existence of Nash equilibria**:

\[
\langle A_1, \ldots, A_n \rangle \bigwedge_i \left( [\langle A_i \rangle \varphi_{A_i}] \Rightarrow \varphi_{A_i} \right)
\]
Expressiveness of $\text{ATL}_{sc}$

- **Client-server interactions** for accessing a shared resource:

$$\langle \text{Server} \rangle \ G \left[ \bigwedge_{c \in \text{Clients}} \langle c \rangle \ F \ \text{access}_c \right]$$

- **Existence of Nash equilibria**:

$$\langle A_1, \ldots, A_n \rangle \bigwedge_i \left( \left[ \langle A_i \rangle \varphi_{A_i} \right] \Rightarrow \varphi_{A_i} \right)$$

- **Existence of dominating strategy**:

$$\langle A \rangle \ [B] \left( \neg \varphi \Rightarrow [A] \neg \varphi \right)$$
Outline of the presentation

1. Introduction

2. Alternating-time Temporal Logic
   - From ATL to ATL with strategy contexts
   - Model checking $\text{ATL}_{sc}$

3. Strategy Logic
   - Explicitly handling strategies
   - Semantics of SL

4. Conclusions and future works
Quantified CTL [ES84,Kup95,Fre01,DLM12]

QCTL extends CTL with propositional quantifiers

\[ \exists p. \varphi \] means that there exists a labelling of the model with \( p \) under which \( \varphi \) holds.

Quantified CTL [ES84,Kup95,Fre01,DLM12]

QCTL extends CTL with propositional quantifiers

\( \exists p. \varphi \) means that there exists a labelling of the model with \( p \) under which \( \varphi \) holds.

\[
EF \Diamond \land \neg \left( \exists p. \left[ EF(\Diamond \land p) \land EF(\Diamond \land \neg p) \right] \right)
\]

Quantified CTL \[\text{[ES84,Kup95,Fre01,DLM12]}\]

QCTL extends CTL with propositional quantifiers

\[\exists p. \varphi\] means that there exists a labelling of the model with \(p\) under which \(\varphi\) holds.

\[\text{EF} \bigcirc \land \neg \left( \exists p. \left[ \text{EF} (\bigcirc \land p) \land \text{EF} (\bigcirc \land \neg p) \right]\right) \equiv \text{uniq}(\bigcirc)\]

Quantified CTL [ES84,Kup95,Fre01,DLM12]

QCTL extends CTL with propositional quantifiers

\[ \exists p. \varphi \] means that there exists a labelling of the model with \( p \) under which \( \varphi \) holds.

\[ EF \bigcirc \land \neg \left( \exists p. \ [EF(\bigcirc \land p) \land EF(\bigcirc \land \neg p)] \right) \equiv \text{uniq}(\bigcirc) \]

\[ \rightsquigarrow \text{true if we label the Kripke structure}; \]
\[ \rightsquigarrow \text{false if we label the computation tree}; \]

Semantics of QCTL

- structure semantics:

\[ \models_s \exists p . \varphi \iff \models \varphi \]
Semantics of QCTL

- **structure semantics:**

\[ \models_s \exists p. \varphi \iff \models \varphi \]

- **tree semantics:**

\[ \models_t \exists p. \varphi \iff \models \varphi \]
Expressiveness of QCTL

QCTL can “count”:

\[ EX_1 \varphi \equiv EX \varphi \land \forall p. [EX(p \land \varphi) \Rightarrow AX(\varphi \Rightarrow p)] \]
\[ EX_2 \varphi \equiv \exists q. [EX_1(\varphi \land q) \land EX_1(\varphi \land \neg q)] \]

QCTL can express (least or greatest) fixpoints:

\[ \mu T. \varphi(T) \equiv \exists t. [AG(t \iff \varphi(t)) \land (\forall t. \prime(AG(t \iff \varphi(t)) \Rightarrow AG(t \Rightarrow t')))] \]

Theorem

QCTL and MSO are equally expressive.
Expressiveness of QCTL

- QCTL can “count”:

\[ EX_1 \varphi \equiv EX \varphi \land \forall p. [EX(p \land \varphi) \Rightarrow AX(\varphi \Rightarrow p)] \]

\[ EX_2 \varphi \equiv \exists q. [EX_1(\varphi \land q) \land EX_1(\varphi \land \neg q)] \]

- QCTL can express (least or greatest) fixpoints:

\[ \mu T. \varphi(T) \equiv \exists t. [AG(t \iff \varphi(t)) \land \\
(\forall t'. (AG(t' \iff \varphi(t')) \Rightarrow AG(t \Rightarrow t')))] \]
Expressiveness of QCTL

- QCTL can “count”:
  \[ \text{EX}_1 \varphi \equiv \text{EX} \varphi \land \forall p. \ [\text{EX}(p \land \varphi) \Rightarrow \text{AX}(\varphi \Rightarrow p)] \]
  \[ \text{EX}_2 \varphi \equiv \exists q. \ [\text{EX}_1(\varphi \land q) \land \text{EX}_1(\varphi \land \neg q)] \]

- QCTL can express (least or greatest) fixpoints:
  \[ \mu T. \varphi(T) \equiv \exists t. \ [\text{AG}(t \iff \varphi(t)) \land \forall t'. (\text{AG}(t' \iff \varphi(t')) \Rightarrow \text{AG}(t \Rightarrow t'))] \]

Theorem

QCTL and MSO are equally expressive.
Decision problems for QCTL [LM14]

Theorem

Under the structure semantics:

- model checking is PSPACE-complete;
- satisfiability is undecidable.

Decision problems for QCTL [LM14]

Theorem

Under the structure semantics:
- model checking is \( \text{PSPACE-} \)complete;
- satisfiability is undecidable.

Under the tree semantics: for formulas with \( k \) nested quantifiers,
- model checking is \( k\text{-EXPTIME-} \)complete;
- satisfiability is \( (k+1)\text{-EXPTIME-} \)complete.

QCTL with structure semantics

**Theorem**

*Model checking QCTL for the structure semantics is PSPACE-complete.*
Theorem

Model checking QCTL for the structure semantics is PSPACE-complete.

Proof

Membership:
- (nondeterministically) pick a labelling,
- check the subformula.

Hardness:
QBF is a special case (without even using temporal modalities).
QCTL with structure semantics

Theorem
Model checking QCTL for the structure semantics is PSPACE-complete.

Proof
Membership:
Iteratively (nondeterministically) pick a labelling,
check the subformula.

Hardness:
QBF is a special case (without even using temporal modalities).

Theorem
QCTL satisfiability for the structure semantics is undecidable.
### Theorem

- Model checking QCTL with $k$ quantifiers in the tree semantics is $k$-EXPTIME-complete.
- Satisfiability of QCTL with $k$ quantifiers in the tree semantics is $(k+1)$-EXPTIME-complete.
QCTL with tree semantics

**Theorem**

- Model checking QCTL with $k$ quantifiers in the tree semantics is $k$-EXPTIME-complete.
- Satisfiability of QCTL with $k$ quantifiers in the tree semantics is $(k+1)$-EXPTIME-complete.

**Proof**

Using (alternating) parity tree automata:
QCTL with tree semantics

Theorem

- Model checking QCTL with k quantifiers in the tree semantics is k-EXPTIME-complete.
- Satisfiability of QCTL with k quantifiers in the tree semantics is (k+1)-EXPTIME-complete.

Proof

Using (alternating) parity tree automata:

![Parity Tree Automaton Diagram]

1. $\delta(q_0, \delta) = (q_1, q_1)$
2. $\delta(q_0, \star) = (q_2, q_2)$
3. $\delta(q_1, \star) = (q_1, q_1)$
4. $\delta(q_2, \star) = (q_2, q_2)$
**QCTL with tree semantics**

**Theorem**
- Model checking QCTL with $k$ quantifiers in the tree semantics is $k$-EXPTIME-complete.
- Satisfiability of QCTL with $k$ quantifiers in the tree semantics is $(k+1)$-EXPTIME-complete.

**Proof**
Using (alternating) parity tree automata:

\[
\begin{align*}
\delta(q_0, \square) &= (q_0, q_1) \lor (q_1, q_0) \\
\delta(q_0, \circ) &= (q_1, q_1) \\
\delta(q_0, \blacklozenge) &= (q_2, q_2) \\
\delta(q_1, \blacklozenge) &= (q_1, q_1) \\
\delta(q_2, \blacklozenge) &= (q_2, q_2)
\end{align*}
\]
QCTL with tree semantics

**Theorem**

- Model checking QCTL with $k$ quantifiers in the tree semantics is $k$-EXPTIME-complete.
- Satisfiability of QCTL with $k$ quantifiers in the tree semantics is $(k+1)$-EXPTIME-complete.

**Proof**

Using (alternating) parity tree automata:

\[
\begin{align*}
\delta(q_0, \bigcirc) &= (q_0, q_1) \lor (q_1, q_0) \\
\delta(q_0, \bigodot) &= (q_1, q_1) \\
\delta(q_0, \bigtriangledown) &= (q_2, q_2) \\
\delta(q_1, \bigstar) &= (q_1, q_1) \\
\delta(q_2, \bigstar) &= (q_2, q_2)
\end{align*}
\]
QCTL with tree semantics

**Theorem**

- Model checking QCTL with \( k \) quantifiers in the tree semantics is \( k \)-EXPTIME-complete.
- Satisfiability of QCTL with \( k \) quantifiers in the tree semantics is \( (k+1) \)-EXPTIME-complete.

**Proof**

Using (alternating) parity tree automata:

\[
\delta(q_0, \bigcirc) = (q_0, q_1) \lor (q_1, q_0)
\]

\[
\delta(q_0, \bigcirc) = (q_1, q_1)
\]

\[
\delta(q_0, \bullet) = (q_2, q_2)
\]

\[
\delta(q_1, \bigstar) = (q_1, q_1)
\]

\[
\delta(q_2, \bigstar) = (q_2, q_2)
\]
Model checking QCTL with $k$ quantifiers in the tree semantics is $k$-EXPTIME-complete.

Satisfiability of QCTL with $k$ quantifiers in the tree semantics is $(k+1)$-EXPTIME-complete.

Proof

Using (alternating) parity tree automata:

\[
\begin{align*}
\delta(q_0, \bullet) &= (q_0, q_1) \lor (q_1, q_0) \\
\delta(q_0, \circ) &= (q_1, q_1) \\
\delta(q_0, \Box) &= (q_2, q_2) \\
\delta(q_1, \star) &= (q_1, q_1) \\
\delta(q_2, \star) &= (q_2, q_2)
\end{align*}
\]
QCTL with tree semantics

**Theorem**
- Model checking QCTL with $k$ quantifiers in the tree semantics is $k$-EXPTIME-complete.
- Satisfiability of QCTL with $k$ quantifiers in the tree semantics is $(k+1)$-EXPTIME-complete.

**Proof**

Using (alternating) parity tree automata:

\[
\delta(q_0, \bullet) = (q_0, q_1) \vee (q_1, q_0)
\]

\[
\delta(q_0, \bigcirc) = (q_1, q_1)
\]

\[
\delta(q_0, \bigotimes) = (q_2, q_2)
\]

\[
\delta(q_1, \bigstar) = (q_1, q_1)
\]

\[
\delta(q_2, \bigstar) = (q_2, q_2)
\]
QCTL with tree semantics

Theorem

- Model checking QCTL with $k$ quantifiers in the tree semantics is $k$-EXPTIME-complete.
- Satisfiability of QCTL with $k$ quantifiers in the tree semantics is $(k+1)$-EXPTIME-complete.

Proof

Using (alternating) parity tree automata:

\[
\begin{align*}
\delta(q_0, \bigcirc) &= (q_0, q_1) \lor (q_1, q_0) \\
\delta(q_0, \bigcirc) &= (q_1, q_1) \\
\delta(q_0, \bullet) &= (q_2, q_2) \\
\delta(q_1, \ast) &= (q_1, q_1) \\
\delta(q_2, \ast) &= (q_2, q_2)
\end{align*}
\]
QCTL with tree semantics

Theorem

- Model checking QCTL with $k$ quantifiers in the tree semantics is $k$-EXPTIME-complete.
- Satisfiability of QCTL with $k$ quantifiers in the tree semantics is $(k+1)$-EXPTIME-complete.

Proof

Using (alternating) parity tree automata:

\[
\begin{align*}
\delta(q_0, \bigcirc) &= (q_0, q_1) \lor (q_1, q_0) \\
\delta(q_0, \bigcirc) &= (q_1, q_1) \\
\delta(q_0, \Box) &= (q_2, q_2) \\
\delta(q_1, \Diamond) &= (q_1, q_1) \\
\delta(q_2, \Diamond) &= (q_2, q_2)
\end{align*}
\]
QCTL with tree semantics

Theorem

- Model checking QCTL with \( k \) quantifiers in the tree semantics is \( k \)-EXPTIME-complete.
- Satisfiability of QCTL with \( k \) quantifiers in the tree semantics is \( (k+1) \)-EXPTIME-complete.

Proof

Using (alternating) parity tree automata:

\[
\begin{align*}
\delta(q_0, \bullet) &= (q_0, q_1) \lor (q_1, q_0) \\
\delta(q_0, \circ) &= (q_1, q_1) \\
\delta(q_0, \star) &= (q_2, q_2) \\
\delta(q_1, \star) &= (q_1, q_1) \\
\delta(q_2, \star) &= (q_2, q_2)
\end{align*}
\]
QCTL with tree semantics

Theorem

- Model checking QCTL with \( k \) quantifiers in the tree semantics is \( k \)-\text{EXPTIME}-complete.
- Satisfiability of QCTL with \( k \) quantifiers in the tree semantics is \( (k+1) \)-\text{EXPTIME}-complete.

Proof

Using (alternating) parity tree automata:

\[
\begin{align*}
\delta(q_0, \bullet) &= (q_0, q_1) \lor (q_1, q_0) \\
\delta(q_0, \circ) &= (q_1, q_1) \\
\delta(q_0, \star) &= (q_2, q_2) \\
\delta(q_1, \circ) &= (q_1, q_1) \\
\delta(q_2, \star) &= (q_2, q_2)
\end{align*}
\]

This automaton corresponds to \( \mathbf{E} \mathbf{U} \)
QCTL with tree semantics

Theorem

- Model checking QCTL with $k$ quantifiers in the tree semantics is $k$-EXPTIME-complete.
- Satisfiability of QCTL with $k$ quantifiers in the tree semantics is $(k+1)$-EXPTIME-complete.

Proof

- polynomial-size automata for CTL;
- quantification is handled by projection, which first requires removing alternation (exponential blowup);
- an automaton equivalent to a QCTL formula can be built inductively;
- emptiness of an alternating parity tree automaton can be decided in exponential time.
Translating ATL\textsubscript{sc} into QCTL [DLM12]

- player $A$ has moves $m_1^A$, ..., $m_n^A$;
- from the transition table, we can compute the set $\text{Next}(\bigcirc, A, m_i^A)$ of states that can be reached from $\bigcirc$ when player $A$ plays $m_i^A$.

Translating $\text{ATL}_{sc}$ into QCTL [DLM12]

- player $A$ has moves $m_1^A$, ..., $m_n^A$;
- from the transition table, we can compute the set $\text{Next}(\bullet, A, m_i^A)$ of states that can be reached from $\bullet$ when player $A$ plays $m_i^A$.

$\langle \cdot A \cdot \rangle \varphi$ can be encoded as follows:

\[ \exists m_1^A. \exists m_2^A \ldots \exists m_n^A. \]

- this corresponds to a strategy: $A \, G(m_i^A \iff \bigwedge \neg m_j^A)$;
- the outcomes all satisfy $\varphi$:

\[ A [G(q \land m_i^C \Rightarrow X \, \text{Next}(q, C, m_i^C)) \Rightarrow \varphi]. \]
Translating $\text{ATL}_{sc}$ into QCTL [DLM12]

- player $A$ has moves $m_1^A$, ..., $m_n^A$;
- from the transition table, we can compute the set $\text{Next}(0, A, m_i^A)$ of states that can be reached from $0$ when player $A$ plays $m_i^A$.

**Corollary**

$\text{ATL}_{sc}$ model checking is decidable (TOWER-complete).

Outline of the presentation

1 Introduction

2 Alternating-time Temporal Logic
   - From ATL to ATL with strategy contexts
   - Model checking $\text{ATL}_{sc}$

3 Strategy Logic
   - Explicitly handling strategies
   - Semantics of SL

4 Conclusions and future works
Strategy Logic [CHP07,MMV10]

Strategy logic

Explicit quantification over strategies + strategy assignment:

$$\exists \sigma_A. \forall \sigma_B. \text{assign}(A \rightarrow \sigma_A, B \rightarrow \sigma_B). \varphi$$

Strategy Logic [CHP07,MMV10]

Strategy logic

Explicit quantification over strategies + strategy assignment:

$$\exists \sigma_A. \forall \sigma_B. \text{assign}(A \rightarrow \sigma_A, B \rightarrow \sigma_B). \varphi$$

- Quantification over strategies
- Assignment of strategies to players
- Path property

Strategy Logic [CHP07, MMV10]

Strategy logic

Explicit quantification over strategies + strategy assignment:

$$\exists \sigma_A. \forall \sigma_B. \text{assign}(A \rightarrow \sigma_A, B \rightarrow \sigma_B). \varphi$$

- Quantification over strategies
- Assignment of strategies to players
- Path property

Theorem

*SL is at least as expressive as ATL*$_{sc}$.


Strategy Logic [CHP07,MMV10]

Strategy logic

Explicit quantification over strategies + strategy assignment:

$$\exists \sigma_A. \forall \sigma_B. \text{assign}(A \rightarrow \sigma_A, B \rightarrow \sigma_B). \varphi$$

- Quantification over strategies
- Assignment of strategies to players
- Path property

Theorem

SL is at least as expressive as ATL_{sc}.

Theorem

SL model-checking is decidable (TOWER-complete).

Outline of the presentation

1. Introduction

2. Alternating-time Temporal Logic
   - From ATL to ATL with strategy contexts
   - Model checking ATL_{sc}

3. Strategy Logic
   - Explicitly handling strategies
   - Semantics of SL

4. Conclusions and future works
What is the exact semantics? [BGM16]

$$\exists \sigma. \mathbf{AG}(\text{assign}(A \rightarrow \sigma). \varphi)$$

What is the exact semantics? [BGM16]

\[ \exists \sigma. \text{AG}(\text{assign}(A \rightarrow \sigma). \varphi) \]

Strategy $\sigma$ is selected here

What is the exact semantics? [BGM16]

\[ \exists \sigma. \text{AG}(\text{assign}(A \rightarrow \sigma). \varphi) \]

Strategy \( \sigma \) is selected here

If strategy \( \sigma \) is assigned here, is the prefix part of the history?

What is the exact semantics? [BGM16]

\[ \exists \sigma. \ AG(\text{assign}(A \rightarrow \sigma). \ \varphi) \]

Example

If \( \sigma \) prescribes to alternate between \( m_0 \) and \( m_1 \), starting with \( m_0 \):

[Diagram of a tree structure with nodes labeled \( m_0 \) and \( m_1 \) at various levels, illustrating the alternation.

What is the exact semantics? [BGM16]

\[ \exists \sigma. \ AG(\text{assign}(A \rightarrow \sigma). \varphi) \]

Example

If \( \sigma \) prescribes to alternate between \( m_0 \) and \( m_1 \), starting with \( m_0 \):

What is the exact semantics? [BGM16]

$$\exists \sigma. \textbf{AG}(\text{assign}(A \rightarrow \sigma). \varphi)$$

**Example**

If \(\sigma\) prescribes to alternate between \(m_0\) and \(m_1\), starting with \(m_0\):

History starts when strategy is selected.

What is the exact semantics? [BGM16]

$$\exists \sigma. \ AG(\text{assign}(A \rightarrow \sigma). \ \varphi)$$

Example

If $\sigma$ prescribes to alternate between $m_0$ and $m_1$, starting with $m_0$:

- History starts when strategy is selected.
- History starts when strategy is assigned.

What is the exact semantics? [BGM16]

$$\exists \sigma. \text{AG}(\text{assign}(A \to \sigma). \varphi)$$

Example

If $\sigma$ prescribes to alternate between $m_0$ and $m_1$, starting with $m_0$:

This is the semantics of Strategy Logic

History starts when strategy is assigned.

History starts when strategy is selected.

What is the exact semantics? [BGM16]

\[ \exists \sigma. \ AG(\text{assign}(A \rightarrow \sigma). \ \varphi) \]

Example

If \( \sigma \) prescribes to alternate between \( m_0 \) and \( m_1 \), starting with \( m_0 \):

- History starts when strategy is selected.
- This is the semantics of Strategy Logic

This makes SL model checking undecidable

- History starts when strategy is assigned.
- This makes SL model checking undecidable

Encoding of a 2-counter machine

Strategy $\sigma$ encodes counter value $k$ if it plays $k$ times to $a$ before playing to $\bot$ (similarly for $\sigma'$).

Lemma

Two strategies $\sigma$ and $\sigma'$ encode the same counter value iff, when entering the game above and applying these strategies, it holds

$$G(a \Rightarrow Xa) \land F\left[(a \land X\bot) \land \neg\langle\cdot\rangle\neg XX\bot\right] (\phi = \cdot)$$
Encoding of a 2-counter machine

Strategy $\sigma_\bigcirc$ encodes counter value $k$ if it plays $k$ times to $[a]$ before playing to $\bot$ (similarly for $\sigma_{\square}$).
Strategy $\sigma_\bigcirc$ encodes counter value $k$ if it plays $k$ times to $a$ before playing to $\bot$ (similarly for $\sigma_\blacksquare$).

**Lemma**

Two strategies $\sigma_\bigcirc$ and $\sigma_\blacksquare$ encode the same counter value iff, when entering the game above and applying these strategies, it holds

$$G(a \Rightarrow Xa) \land F[(a \land X\bot) \land \neg \langle \diamond \rangle \neg XX\bot] \quad (\varphi=)$$
A strategy for $\bigcirc$ encodes a run of the 2-counter machine;

- At each step, Player $\square$ can check correct values of counters (this is where we use our special semantics).
Strategy dependences (ongoing work) [MMPV11, GBM17]

From this point on, we only consider formulas in SL\([BG]\):

\[
\text{assign} (A_j \mapsto \sigma_i) \quad \phi_{LTL}
\]

Strategy dependences (ongoing work) [MMPV11, GBM17]

Note

From this point on, we only consider formulas in SL[BG]:

\[ (Q; \sigma_i). \text{assign}(A_j \mapsto \sigma_i). \varphi_{\text{LTL}} \]

quantifier block

goal

Strategy dependences (ongoing work) \[\text{[MMPV11,GBM17]}\]

**Note**

From this point on, we only consider formulas in SL[BG]:

\[
\text{SL} \quad \text{SL}\lbrack\text{BG}\rbrack \quad \text{SL}\lbrack\text{DG}\rbrack \quad \text{SL}\lbrack\text{CG}\rbrack \quad \text{SL}\lbrack1G\rbrack
\]

- **quantifier block**
- **goal**

\[(Q_i \sigma_i). \quad \text{assign}(A_j \rightarrow \sigma_i). \quad \varphi_{\text{LTL}}\]

**disjunction/conjunction of goals**

---


Strategy dependences (ongoing work) [MMPV11, GBM17]

Note

From this point on, we only consider formulas in SL[BG]:

\[ (Q; \sigma_i). \text{assign}(A_j \rightarrow \sigma_i). \varphi_{\text{LTL}} \]

quantifier block  goal

SL

SL[BG]  boolean combination of goals

SL[DG]  SL[CG]  disjunction/conjunction of goals

SL[1G]
Strategy dependences (ongoing work) [MMPV11, GBM17]

∀σ_A ∃σ_B. assign(A ↦ σ_A; B ↦ σ_B). ϕ

Strategy dependences (ongoing work) [MMPV11, GBM17]

∀σ_A∃σ_B. assign(A ↦ σ_A; B ↦ σ_B). ϕ

Strategy σ_B at node n may depend on the whole strategy σ_A.

Strategy dependences (ongoing work) \cite{MMPV11,GBM17}

\[ \forall \sigma_A \exists \sigma_B. \text{assign}(A \rightarrow \sigma_A; B \rightarrow \sigma_B). \varphi \]

Strategy \( \sigma_B \) at node \( n \) may depend on the whole strategy \( \sigma_A \).

Strategy \( \sigma_B \) at node \( n \) only depends on \( \sigma_A \) along the branch to \( n \).

\cite{MMPV11} Mogavero, Murano, Perelli, Vardi. Reasoning about strategies: ... J. ACM, 2011.
\cite{GBM17} Gardy, Bouyer, Markey. Dependences in Strategy Logic. Submitted (arXiv 1708.05849).
Dependence maps (Skolemization)

Example

\[ \varphi = \forall y. \exists z. \forall x_A. \forall x_B. \]

\[ \bigvee \left\{ \begin{array}{l}
    \text{assign}(\square \mapsto x_A; \bigcirc \mapsto y; \blacklozenge \mapsto z). \quad \text{F } p_1 \\
    \text{assign}(\square \mapsto x_B; \bigcirc \mapsto y; \blacklozenge \mapsto z). \quad \text{F } p_2
\end{array} \right\} \]

Then \( q_0 \models \varphi \) is (a priori) not equivalent to \( q_0 \models \neg \varphi \).
Dependence maps (Skolemization)

Example

$\varphi = \forall y. \exists z. \forall x_A. \forall x_B.$

$\bigvee \begin{cases} 
\text{assign}(\square \mapsto x_A; \bigcirc \mapsto y; \diamond \mapsto z). & \mathbf{F} p_1 \\
\text{assign}(\square \mapsto x_B; \bigcirc \mapsto y; \diamond \mapsto z). & \mathbf{F} p_2 
\end{cases}$

Dependence maps

Dependence maps select $\exists$-strategies from $\forall$-strategies:

$\theta: (\mathcal{V}^\forall \to \text{Strats}) \to (\mathcal{V}^\exists \to \text{Strats})$

$w \mapsto \theta(w)$
Dependence maps (Skolemization)

Example

\[ \varphi = \forall y. \exists z. \forall x_A. \forall x_B. \]
\[
\bigvee \left\{ \text{assign}(\square \mapsto x_A; \bigcirc \mapsto y; \Diamond \mapsto z). \quad \text{F } p_1 \right. \\
\text{assign}(\square \mapsto x_B; \bigcirc \mapsto y; \Diamond \mapsto z). \quad \text{F } p_2 \right. \]

Dependence maps

Dependence maps select \( \exists \)-strategies from \( \forall \)-strategies:

\[ \theta : (\forall \rightarrow \text{Strats}) \rightarrow (\exists \rightarrow \text{Strats}) \]
\[ w \mapsto \theta(w) \]

Then

\[ q_0 \models (Q_i \sigma_i). \Phi \iff \exists \theta. \forall w. \ q_0 \models_{\theta(w) \cup w} \Phi. \]
Dependence maps (Skolemization)

Example

\[ \varphi = \forall y. \exists z. \forall x_A. \forall x_B. \]

\[ \bigvee \left\{ \begin{array}{l}
\text{assign}(\square \mapsto x_A; \bigcirc \mapsto y; \Diamond \mapsto z). \quad \mathsf{F} \ p_1 \\
\text{assign}(\square \mapsto x_B; \bigcirc \mapsto y; \Diamond \mapsto z). \quad \mathsf{F} \ p_2
\end{array} \right. \]

Dependence maps

Dependence maps select \( \exists \)-strategies from \( \forall \)-strategies:

\[ \theta : (\mathcal{V}^\forall \rightarrow \text{Strats}) \rightarrow (\mathcal{V}^\exists \rightarrow \text{Strats}) \]

Then

\[ q_0 \models (Q_i \sigma_i). \Phi \quad \Leftrightarrow \quad \exists \theta. \forall w. \; q_0 \models \theta(w) \cup w \; \Phi. \]

\[ q_0 \not\models \varphi \; \text{is (a priori) not equivalent to} \; q_0 \models \neg \varphi. \]
Dependence maps (Skolemization)

Example

\[ \varphi = \forall y. \exists z. \forall x_A. \forall x_B. \]

\[ \bigvee \left\{ \begin{array}{l}
\text{assign}(\square \mapsto x_A; \bigcirc \mapsto y; \lozenge \mapsto z). \quad \mathbf{F} \ p_1 \\
\text{assign}(\square \mapsto x_B; \bigcirc \mapsto y; \lozenge \mapsto z). \quad \mathbf{F} \ p_2
\end{array} \right. \]

Dependence maps

- classical SL semantics (\(\models^C\)):
  - only depend on previously-quantified variables:
    - if \(w(y) = w'(y)\), then \(\theta(w)(z) = \theta(w')(z)\)
Dependence maps (Skolemization)

Example

\[ \varphi = \forall y. \exists z. \forall x_A. \forall x_B. \]

\[ \bigvee \left\{ \begin{array}{l}
\text{assign}(\square \mapsto x_A; \bigcirc \mapsto y; \Diamond \mapsto z). \quad \mathbf{F} \ p_1 \\
\text{assign}(\square \mapsto x_B; \bigcirc \mapsto y; \Diamond \mapsto z). \quad \mathbf{F} \ p_2
\end{array} \right. \]

Dependence maps

- **classical SL semantics** (\(\models^C\)):
  - only depend on previously-quantified variables:
    
    \[ \text{if } w(y) = w'(y), \text{ then } \theta(w)(z) = \theta(w')(z) \]

- **elementary semantics** (\(\models^E\)):
  - only depend on previously-quantified variables along current prefix:

    \[ \text{for all } n, \text{ if } w(y)(m) = w'(y)(m) \text{ for all } m \preceq n, \text{ then } \theta(w)(z)(n) = \theta(w')(z)(n) \]
Dependence maps (Skolemization)

Example

\[ \varphi = \forall y. \exists z. \forall x_A. \forall x_B. \]

\[
\bigvee \begin{cases}
\text{assign}(\square \mapsto x_A; \bigcirc \mapsto y; \diamond \mapsto z). & \text{F } p_1 \\
\text{assign}(\square \mapsto x_B; \bigcirc \mapsto y; \diamond \mapsto z). & \text{F } p_2
\end{cases}
\]

Theorem

For any \( \varphi \in SL[BG] \):

\[ q_0 \models^C \varphi \iff q_0 \models \varphi \]

\[ q_0 \models^E \varphi \Rightarrow q_0 \models^C \varphi \]
Dependence maps (Skolemization)

**Example**

\[ \varphi = \forall y. \exists z. \forall x_A. \forall x_B. \]
\[ \bigvee \left\{ \text{assign}(\square \mapsto x_A; \bigcirc \mapsto y; \Diamond \mapsto z) \right. \bigwedge F p_1 \]
\[ \left. \text{assign}(\square \mapsto x_B; \bigcirc \mapsto y; \Diamond \mapsto z) \bigwedge F p_2 \right\} \]

- \( q_0 \models^C \varphi \): z plays to \( p_2 \) iff y plays to \( p_1 \).

**Theorem**

*For any \( \varphi \in SL[BG] \):

\[ q_0 \models^C \varphi \iff q_0 \models \varphi \]
\[ q_0 \models^E \varphi \Rightarrow q_0 \models^C \varphi \]
Dependence maps (Skolemization)

**Example**

\[ \varphi = \forall y. \exists z. \forall x_A. \forall x_B. \]

\[ \bigvee \left\{ \begin{array}{l}
\text{assign(□} \mapsto x_A; \bigcirc \mapsto y; \spadesuit \mapsto z). \ F p_1 \\
\text{assign(□} \mapsto x_B; \bigcirc \mapsto y; \spadesuit \mapsto z). \ F p_2
\end{array} \right\} \]

- \( q_0 \models^C \varphi \): \( z \) plays to \( p_2 \) iff \( y \) plays to \( p_1 \).
- \( q_0 \not\models^E \varphi \): \( y \) only plays in \( q_1 \), so \( z \) will not depend on \( y \).

**Theorem**

For any \( \varphi \in SL[BG] \):

\[ q_0 \models^C \varphi \iff q_0 \models \varphi \quad q_0 \models^E \varphi \Rightarrow q_0 \models^C \varphi \]
Dependence maps (Skolemization)

Example

\[ \varphi = \forall y. \exists z. \forall x_A. \forall x_B. \]

\[ \bigvee \left\{ \text{assign}([\square \mapsto x_A; \bigcirc \mapsto y; \Diamond \mapsto z]). \ F p_1 \right\} \]

\[ \text{assign}([\square \mapsto x_B; \bigcirc \mapsto y; \Diamond \mapsto z]). \ F p_2 \]

- \( q_0 \models^C \varphi \): z plays to \( p_2 \) iff y plays to \( p_1 \).
- \( q_0 \not\models^E \varphi \): y only plays in \( q_1 \), so z will not depend on y.
- \( q_0 \not\models^E \neg \varphi \): because otherwise \( q_0 \models^C \neg \varphi \).

Theorem

For any \( \varphi \in SL[BG] \):

\[ q_0 \models^C \varphi \iff q_0 \models \varphi \]

\[ q_0 \models^E \varphi \Rightarrow q_0 \models^C \varphi \]
Beyond elementary dependences [GBM17]

Elementary semantics

Strategy $\sigma_B$ at node $n$ only depends on $\sigma_A$ along the branch to $n$. 

Beyond elementary dependences [GBM17]

Elementary semantics
Strategy $\sigma_B$ at node $n$ only depends on $\sigma_A$ along the branch to $n$.

Timeline semantics
Strategy $\sigma_B$ also depends on $\sigma_C$ along strict prefixes of $n$.

Beyond elementary dependences [GBM17]

∀σ_A
∃σ_B
∀σ_C

Timeline semantics
Strategy σ_B also depends on σ_C along strict prefixes of n.

Theorem
For any ϕ ∈ SL[BG],

q_0 ≡^E ϕ \Rightarrow q_0 ≡^T ϕ

Beyond elementary dependences [GBM17]

Timeline semantics
Strategy $\sigma_B$ also depends on $\sigma_C$ along strict prefixes of $n$.

Theorem
For any $\varphi \in SL[BG]$, 

$q_0 \models^E \varphi \Rightarrow q_0 \models^T \varphi$

Theorem
For a large fragment of $SL[BG]$ (containing $SL[CG]$ and $SL[DG]$),

$q_0 \models^T \varphi \iff q_0 \not\models^T \neg \varphi$

Model checking this fragment is 2EXPTIME-complete (for $\models^T$).

Outline of the presentation

1. Introduction

2. Alternating-time Temporal Logic
   - From ATL to ATL with strategy contexts
   - Model checking ATL_{sc}

3. Strategy Logic
   - Explicitly handling strategies
   - Semantics of SL

4. Conclusions and future works
Conclusions and future works

Conclusions

- Temporal logics have been **successful** for model checking “closed” systems;
- For multi-agent systems, **ATL is not expressive enough**;
- But **ATL_{sc}** and **Strategy Logic** are too expressive:
  - model checking is very hard;
  - does the semantics correspond to what we expect/need?

Many open questions:
- better understand the semantics
- identify expressive fragments with efficient algorithms
- extend with stochastic strategies, quantities, partial observation...
Conclusions and future works

**Conclusions**

- Temporal logics have been successful for model checking “closed” systems;
- For multi-agent systems, ATL is not expressive enough;
- But $\text{ATL}_{sc}$ and Strategy Logic are too expressive:
  - model checking is very hard;
  - does the semantics correspond to what we expect/need?

**Conclusions**

- Many open questions:
  - better understand the semantics
  - identify expressive fragments with efficient algorithms
  - extend with stochastic strategies, quantities, partial observation…