ATL with strategy contexts

Nicolas Markey
LSV – ENS Cachan

(based on joint works with Thomas Brihaye, Arnaud Da Costa-Lopes, François Laroussinie)

Laboratoire d’Algorithmique, Complexité et Logique
Créteil, November 9, 2015
Model checking and synthesis

**system:**

[Diagram of two tanks with a pump connecting them.]

[Reference: http://www.embedded.com]

**property**

[Diagram of a tank with a checkmark indicating fullness or emptiness.]

**model-checking algorithm**

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

**yes/no**
Model checking and synthesis

system:

property

synthesis algorithm

\[
AG(\neg B.\text{overfull} \land \neg B.\text{dried}_\text{up})
\]
Outline of the presentation

1. Introduction

2. Basics of CTL and ATL
   - expressing properties of reactive systems
   - efficient verification algorithms

3. Temporal logics for multi-agent systems
   - specifying properties of complex interacting systems
   - expressive power of ATL_{sc}
   - translation into Quantified CTL (QCTL)
   - algorithms for ATL_{sc}

4. Conclusions and future works
Outline of the presentation

1. Introduction

2. Basics of CTL and ATL
   - expressing properties of reactive systems
   - efficient verification algorithms

3. Temporal logics for multi-agent systems
   - specifying properties of complex interacting systems
   - expressive power of ATL_{sc}
   - translation into Quantified CTL (QCTL)
   - algorithms for ATL_{sc}

4. Conclusions and future works
Computation-Tree Logic (CTL)

- atomic propositions: \(\bigcirc, \bigcirc, \ldots\)
Computation-Tree Logic (CTL)

- atomic propositions: \( \bigcirc, \bigcirc, \ldots \)

- boolean combinators: \( \neg \varphi, \varphi \lor \psi, \varphi \land \psi, \ldots \)
Computation-Tree Logic (CTL)

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

  - \( X \varphi \)  
    \[ \begin{array}{c}
    \bigcirc \rightarrow \varphi \rightarrow \bigcirc \rightarrow \bigcirc \rightarrow \bigcirc \\
    \end{array} \]  
    “next \( \varphi \)”

  - \( \varphi \mathbf{U} \psi \)  
    \[ \begin{array}{c}
    \varphi \rightarrow \varphi \rightarrow \psi \rightarrow \bigcirc \\
    \end{array} \]  
    “\( \varphi \) until \( \psi \)”
Computation-Tree Logic (CTL)

- atomic propositions: $\Diamond$, $\Box$, ...
- boolean combinators: $\neg \varphi$, $\varphi \lor \psi$, $\varphi \land \psi$, ...
- temporal modalities:
  - $\text{X} \varphi$
  - $\varphi \text{U} \psi$
  - $\text{true \ U} \varphi \equiv \text{F} \varphi$
  - $\neg \text{F} \neg \varphi \equiv \text{G} \varphi$

\text{“next $\varphi$”}  \text{“$\varphi$ until $\psi$”}  \text{“eventually $\varphi$”}  \text{“always $\varphi$”}
Computation-Tree Logic (CTL)

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

- path quantifiers:
Examples of CTL formulas

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

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

\( EF \) is reachable
Examples of CTL formulas

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

\[ EF \quad \text{is reachable} \]
Examples of CTL formulas

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

$$EG(\neg \bullet \land EF \bullet)$$

there is a path along which $\bullet$ is always reachable, but never reached
Examples of CTL formulas

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

\[ \text{EG}(\neg \Diamond \land \text{EF} \Diamond) \] there is a path along which \( \Diamond \) is always reachable, but never reached
Examples of CTL formulas

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

\[ EG(\neg \Box p \land EF \diamond p) \]

there is a path along which \( \Box \) is always reachable, but never reached
Examples of CTL formulas

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.
Examples of CTL formulas

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

Theorem ([CE81, QS82])

*CTL model checking is PTIME-complete.*

Theorem ([KVW94])

*CTL model checking on product structures is PSPACE-complete.*

[CE81] Clarke, Emerson. Design and Synthesis of Synchronization Skeletons... LOP, 1981.
Reasoning about open systems

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.
Reasoning about open systems

**Concurrent games**

A concurrent game is made of

- a transition system;

![Diagram of a concurrent game](image)
Concurrent games

A concurrent game is made of

- a transition system;
- a set of agents (or players);
Reasoning about open systems

**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.
Reasoning about open systems

**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.
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.
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

![Diagram of a strategy for a 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 \( ● \) and \( ○ \)
(starting with \( ○ \)).
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 and (starting with ).
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).
A (pure) strategy for a given player is a function telling which action to play depending on what has happened previously.

Example

Memoryless 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**

Memoryless strategy for player

always go to 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**

Memoryless strategy for player 

always go to ○.
Strategies

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

Example

Memoryless strategy for player
always go to  

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

Memoryless strategy for player always go to ⬇️.
Temporal logics for games: ATL

ATL extends CTL with strategy quantifiers

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

Temporal logics for games: ATL

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):

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

Temporal logics for games: ATL

ATL extends CTL with **strategy quantifiers**

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

Temporal logics for games: ATL

ATL extends CTL with strategy quantifiers

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

Model checking ATL is \( \text{PTIME} \)-complete.

Temporal logics for games: ATL

ATL extends CTL with strategy quantifiers

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

Temporal logics for games: ATL

ATL extends CTL with strategy quantifiers

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

Theorem (\[AHK02\])

Model checking ATL is \( \text{PTIME} \)-complete.

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

ATL extends CTL with **strategy quantifiers**

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

Temporal logics for games: ATL

ATL extends CTL with strategy quantifiers

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

Temporal logics for games: ATL

ATL extends CTL with strategy quantifiers

\( ⟨⟨A⟩⟩ \varphi \) expresses that \( A \) has a strategy to enforce \( \varphi \).

Theorem ([AHK02])

Model checking ATL is PTIME-complete.

Temporal logics for games: ATL

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

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.

For concurrent games, this assumes that the transition table is given explicitly (size \(|\text{Moves}|\cdot|\text{Agt}|\)).

Temporal logics for games: ATL

ATL extends CTL with strategy quantifiers

\[⟨⟨A⟩⟩ \varphi \] expresses that A has a strategy to enforce \( \varphi \).

Theorem ([AHK02])

*Model checking ATL is PTIME-complete.*

For concurrent games, this assumes that the transition table is given explicitly (size \(|\text{Moves}|\times|\text{Agt}|\)).

Memoryless strategies are sufficient for ATL.

Outline of the presentation

1. Introduction

2. Basics of CTL and ATL
   - expressing properties of reactive systems
   - efficient verification algorithms

3. Temporal logics for multi-agent systems
   - specifying properties of complex interacting systems
   - expressive power of $\text{ATL}_{sc}$
   - translation into Quantified CTL (QCTL)
   - algorithms for $\text{ATL}_{sc}$

4. Conclusions and future works
ATL with strategy contexts [BDLM09,DLM10]

Example

\[\langle \Box \rangle G(\langle \square \rangle F \Diamond)\]

Brihaye, Da Costa, Laroussinie, Markey. ATL with strategy contexts and bounded memory. LFCS, 2009.
Da Costa, Laroussinie, Markey. ATL with strategy contexts: expressiveness and ... FSTTCS, 2010.
Example

\[
(\langle \text{O} \rangle \ G( \langle \Box \rangle \ F \text{O})
\]

Brihaye, Da Costa, Laroussinie, Markey. ATL with strategy contexts and bounded memory. LFCS, 2009.
Da Costa, Laroussinie, Markey. ATL with strategy contexts: expressiveness and ... FSTTCS, 2010.
ATL with strategy contexts \([\text{BDLM09, DLM10}]\)

Example

\[\langle\Box\rangle \mathbf{G} (\langle\square\rangle \mathbf{F} \Diamond)\]

- Player \(\bigcirc\) in \(\bigcirc\) always plays to \(\Box\).

Brihaye, Da Costa, Laroussinie, Markey. ATL with strategy contexts and bounded memory. LFCS, 2009.
ATL with strategy contexts \([\text{BDLM09, DLM10}]\)

Example

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

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

Brihaye, Da Costa, Laroussinie, Markey. ATL with strategy contexts and bounded memory. LFCS, 2009.
Da Costa, Laroussinie, Markey. ATL with strategy contexts: expressiveness and ... FSTTCS, 2010.
Example

Player 0 in 1 always plays to 2.

Brihaye, Da Costa, Laroussinie, Markey. ATL with strategy contexts and bounded memory. LFCS, 2009.
Da Costa, Laroussinie, Markey. ATL with strategy contexts: expressiveness and ... FSTTCS, 2010.
ATL with strategy contexts [BDLM09,DLM10]

Example

- Player $\Diamond$ in $\Box$ always plays to $\Box$;
- Player $\Box$ in $\Box$ then plays to $\Diamond$.

Brihaye, Da Costa, Laroussinie, Markey. ATL with strategy contexts and bounded memory. LFCS, 2009.
Da Costa, Laroussinie, Markey. ATL with strategy contexts: expressiveness and ... FSTTCS, 2010.
ATL with strategy contexts

**Definition**

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

- \( \langle \cdot A \cdot \rangle \varphi \) is similar to \( \langle \langle A \rangle \rangle \varphi \) but assigns the corresponding strategy to \( A \) for evaluating \( \varphi \);
ATL with strategy contexts

Definition

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

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

- $\langle \overline{A} \rangle \varphi \equiv \langle \text{Agt} \setminus A \rangle \varphi$
  (useful for getting formulas that do not depend on Agt);
ATL with strategy contexts

Definition

ATL$_{sc}$ has new strategy quantifiers:

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

- $\langle \overline{A} \rangle \varphi \equiv \langle \text{Agt} \setminus A \rangle \varphi$
  (useful for getting formulas that do not depend on Agt);

- $\langle A \rangle_0 \varphi$ is similar to $\langle A \rangle \varphi$ but quantifies over memoryless strategies;
ATL with strategy contexts

Definition

\( \text{ATL}_{sc} \) has new strategy quantifiers:

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

- \( \langle \cdot \overline{A} \cdot \rangle \varphi \equiv \langle \text{Agt} \setminus A \cdot \rangle \varphi \)
  (useful for getting formulas that do not depend on Agt);

- \( \langle \cdot A \cdot \rangle_0 \varphi \) is similar to \( \langle \cdot A \cdot \rangle \varphi \) but quantifies over memoryless strategies;

- \( \langle A \rangle \varphi \) drops the assigned strategies for \( A \).
ATL with strategy contexts

Definition

ATL\textsubscript{sc} has new strategy quantifiers:

- \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;

- \langle \overline{A} \rangle \varphi \equiv \langle \text{Agt} \setminus A \rangle \varphi
  \text{ (useful for getting formulas that do not depend on Agt);}

- \langle A \rangle_0 \varphi \text{ is similar to } \langle A \rangle \varphi \text{ but quantifies over memoryless strategies;}

- \langle A \rangle \varphi \text{ drops the assigned strategies for } A.

- \langle A \rangle \varphi \text{ is dual to } \langle A \rangle \varphi:

  \[
  \langle A \rangle \varphi \equiv \neg \langle A \rangle \neg \varphi
  \]
ATL with strategy contexts

Definition

ATL$_{sc}$ has new strategy quantifiers:

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

Definition

Semantics of ATL strategy quantifier:

$\mathcal{G}, \bigcirc \models \langle A \rangle \varphi \iff \exists \sigma_A. \forall \pi \in \text{Out}(\bigcirc, \sigma_A). \pi \models \varphi$
ATL with strategy contexts

**Definition**

ATL$_{sc}$ has **new strategy quantifiers**:
- $\langle \cdot A \cdot \rangle \varphi$ is similar to $\langle A \rangle \varphi$ but **assigns** the corresponding strategy to $A$ for evaluating $\varphi$;

**Definition**

**Semantics of ATL strategy quantifier:**

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

**Semantics of ATL$_{sc}$ strategy quantifier:**

$G, \bigcirc \models_{\sigma_B} \langle \cdot A \cdot \rangle \varphi \iff \exists \sigma_A. \forall \pi \in \text{Out}(\bigcirc, \sigma_A \circ \sigma_B). \pi \models_{\sigma_A \circ \sigma_B} \varphi$
ATL with strategy contexts

**Definition**

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

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

**Definition**

**Semantics of $\text{ATL}_{sc}$ strategy quantifier:**

$\mathcal{G}, \bigcirc \models_{\sigma_B} \langle \cdot A \cdot \rangle \varphi \iff \exists \sigma_A. \forall \pi \in \text{Out}(\bigcirc, \sigma_A \circ \sigma_B). \pi \models_{\sigma_A \circ \sigma_B} \varphi$

~ newly selected strategies added to the context:

$\sigma_A \circ \sigma_B :$

- $a \mapsto \sigma_A(a)$ if $a \in A \setminus B$
- $b \mapsto \sigma_B(b)$ if $b \in B \setminus A$
- $c \mapsto \sigma_A(c)$ if $c \in B \cap A$
What $\text{ATL}_{sc}$ can express

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

$$\langle \cdot \text{Server} \rangle \; G \; \left[ \bigwedge_{c \in \text{Clients}} \langle \cdot \cdot \cdot \rangle \; F \; \text{access}_c \right] \wedge \left[ \neg \bigwedge_{c \neq c'} \text{access}_c \wedge \text{access}_{c'} \right]$$
What ATL$_{sc}$ can express

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

  $\langle \cdot \rangle \text{G} \left[ \bigwedge_{c \in \text{Clients}} \langle \cdot c \cdot \rangle F \text{access}_c \right. \quad \left. \bigwedge \neg \bigwedge_{c \neq c'} \text{access}_c \land \text{access}_{c'} \right]$

- **Existence of Nash equilibria**:

  $\langle \cdot A_1, ..., A_n \cdot \rangle \bigwedge_i (\langle \cdot A_i \cdot \rangle \varphi_{A_i} \Rightarrow \varphi_{A_i})$
What $\text{ATL}_{sc}$ can express

- **Client-server interactions** for accessing a shared resource:
  \[
  \langle \cdot \rangle \text{Server} \land G \left( \bigwedge_{c \in \text{Clients}} \langle \cdot \rangle F \text{access}_c \land \neg \bigwedge_{c \neq c'} \text{access}_c \land \text{access}_{c'} \right)
  \]

- **Existence of Nash equilibria**:
  \[
  \langle \cdot \rangle A_1, \ldots, A_n \land \bigwedge_i (\langle \cdot \rangle A_i \varphi_{A_i} \Rightarrow \varphi_{A_i})
  \]

- **Existence of dominating strategy**:
  \[
  \langle \cdot \rangle A \land [B] (\neg \varphi \Rightarrow [A] \neg \varphi)
  \]
Expressiveness of $\text{ATL}_{sc}$

**Theorem**

$\text{ATL}_{sc}$ is strictly more expressive than ATL
Expressiveness of $\text{ATL}_{sc}$

**Theorem**

$\text{ATL}_{sc}$ is strictly more expressive than $\text{ATL}$

**Proof**

\[
\langle A \rangle \varphi \equiv (\langle \emptyset \rangle \langle A \rangle \hat{\varphi})
\]
Expressiveness of $\text{ATL}_{sc}$

**Theorem**

$\text{ATL}_{sc}$ is strictly more expressive than $\text{ATL}$

**Proof**

$\langle 1 \cdot \rangle (\langle 2 \cdot \rangle \mathcal{X} a \land \langle 2 \cdot \rangle \mathcal{X} b)$ is only true in the second game. But $\text{ATL}$ cannot distinguish between these two games.

---

[Diagram showing two game graphs with states and transitions]
Outline of the presentation

1. Introduction

2. Basics of CTL and ATL
   - expressing properties of reactive systems
   - efficient verification algorithms

3. Temporal logics for multi-agent systems
   - specifying properties of complex interacting systems
   - expressive power of $\text{ATL}_{sc}$
   - translation into Quantified CTL (QCTL)
   - algorithms for $\text{ATL}_{sc}$

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

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]

QCTL extends CTL with propositional quantifiers

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

\[ \text{EF} \bigcirc \land \forall p. \left[ \text{EF}(p \land \bigcirc) \Rightarrow \text{AG} (\bigcirc \Rightarrow p) \right] \]

Quantified CTL [ES84, Kup95, Fre01]

QCTL extends CTL with propositional quantifiers

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

$$\bullet \ EF \bigcirc \land \forall p. \ [EF(p \land \bigcirc) \Rightarrow AG(\bigcirc \Rightarrow p)] \equiv \text{uniq(\bigcirc)}$$

Quantified CTL [ES84,Kup95,Fre01]

QCTL extends CTL with propositional quantifiers

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

$$\bullet \text{ EF } \bigcirc \land \forall p. \left[ \text{ EF } (p \land \bigcirc) \Rightarrow \text{ AG } (\bigcirc \Rightarrow p) \right] \equiv \text{ uniq } (\bigcirc)$$

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

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”:

\( \text{EX}_1 \varphi \equiv \text{EX} \varphi \land \forall p. [\text{EX}(p \land \varphi) \Rightarrow \text{AX} (\varphi \Rightarrow p)] \)

\( \text{EX}_2 \varphi \equiv \exists q. [\text{EX}_1 (q \land \varphi) \land \text{EX}_1 (q \land \neg q)] \)

Expressiveness of QCTL

- QCTL can “count”:

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

- QCTL can express (least or greatest) fixpoints:

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

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:

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

Theorem

QCTL, QCTL* and MSO are equally expressive (under both semantics).

QCTL with structure semantics

**Theorem**

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

QCTL with structure semantics

**Theorem**

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

**Proof**

**Membership:** labelling algorithm.
- (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:** labelling algorithm.

- (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.*

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.

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:

[Diagram of a parity tree automaton]

```
q_0
q_1
q_0
q_1
q_0
q_1
q_1
q_1
q_1
q_1
```
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.

Using (alternating) parity tree automata:

\[
\delta(q_0, \bigcirc) = (q_0, q_1) \lor (q_1, q_0)
\]
\[
\delta(q_0, \bigcirc) = (q_1, q_1)
\]
\[
\delta(q_0, \bullet) = (q_2, q_2)
\]
\[
\delta(q_1, \star) = (q_1, q_1)
\]
\[
\delta(q_2, \star) = (q_2, q_2)
\]
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:

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

$$
\begin{align*}
\delta(q_0, \bigcirc) &= (q_0, q_1) \lor (q_1, q_0) \\
\delta(q_0, \blackcirc) &= (q_1, q_1) \\
\delta(q_0, \bullet) &= (q_2, q_2) \\
\delta(q_1, \bigstar) &= (q_1, q_1) \\
\delta(q_2, \bigstar) &= (q_2, q_2)
\end{align*}
$$
**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:

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

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

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

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

- \(\delta(q_0, \bigcirc) = (q_0, q_1) \lor (q_1, q_0)\)
- \(\delta(q_0, \bigodot) = (q_1, q_1)\)
- \(\delta(q_0, \bigotimes) = (q_2, q_2)\)
- \(\delta(q_1, \bigstar) = (q_1, q_1)\)
- \(\delta(q_2, \bigstar) = (q_2, q_2)\)
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:

\[
\delta(q_0, \bullet) = (q_0, q_1) \vee (q_1, q_0)
\]
\[
\delta(q_0, \bigcirc) = (q_1, q_1)
\]
\[
\delta(q_0, \bigotimes) = (q_2, q_2)
\]
\[
\delta(q_1, \bigstar) = (q_1, q_1)
\]
\[
\delta(q_2, \bigstar) = (q_2, q_2)
\]

This automaton corresponds to E\(\bigcirc\) U \(\bigcirc\)
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 **tree 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$_{sc}$ into QCTL

- 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

- 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 $m_i^A$.

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

$$\exists m_1^A. \exists m_2^A \ldots \exists m_n^A.$$ 

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

$$A [ \Box (q \land m_i^A \Rightarrow X \ \text{Next}(q, A, m_i^A)) \Rightarrow \varphi ].$$

Translating $\text{ATL}_{sc}$ into QCTL

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

**Corollary**

\textit{ATL}_{sc} model checking is decidable, with non-elementary complexity (TOWER-complete).

**Corollary**

\textit{ATL}_{sc}^0 (quantification restricted to memoryless strategies) model checking is PSPACE-complete.

What about satisfiability?

**Theorem**

QCTL satisfiability is decidable (for the tree semantics).
What about satisfiability?

**Theorem**

QCTL satisfiability is decidable (for the tree semantics).

But

**Theorem ([TW12])**

$\text{ATL}^\text{sc}$ satisfiability is undecidable.

---

What about satisfiability?

**Theorem**

*QCTL satisfiability is decidable (for the tree semantics).*

But

**Theorem ([TW12])**

*ATL<sub>sc</sub> satisfiability is undecidable.*

Why?

The translation from ATL<sub>sc</sub> to QCTL assumes that the game structure is given!

Satisfiability for turn-based games

Theorem ([LM13])
When restricted to turn-based games, $\text{ATL}_{sc}$ satisfiability is decidable.

Satisfiability for turn-based games

Theorem ([LM13])

When restricted to turn-based games, ATL\(_{sc}\) satisfiability is decidable.

- player □ has moves □, ○, and ▢.
- a strategy can be encoded by marking some of the nodes of the tree with proposition \(\text{mov}_A\).

\[\langle \cdot A \rangle \varphi \text{ can be encoded as follows:} \]

\[\exists \text{mov}_A.\]

- it corresponds to a strategy: \(\text{AG}(\text{turn}_A \Rightarrow \text{EX}_1 \text{mov}_A)\);
- the outcomes all satisfy \(\varphi\): \(\text{A}[\text{G}(\text{turn}_A \land \text{X mov}_A) \Rightarrow \varphi]\).

What about Strategy Logic? [CHP07, MMV10]

**Strategy logic**
Explicit quantification over strategies + strategy assignment

**Example**
\[
\langle \cdot A \rangle \varphi \equiv \exists \sigma_1. \text{assign}(\sigma_1, A). \varphi
\]

Strategy logic can also be translated into QCTL.

**Theorem**
- *Strategy-logic model-checking is decidable.*
- *Strategy-logic satisfiability is decidable when restricted to turn-based games.*

Conclusions and future works

Conclusions

- ATL\textsubscript{sc} is a very expressive, yet \textit{decidable} extension of ATL;
- QCTL is a powerful extension of CTL;
- it is a \textit{nice tool to understand temporal logics for games} (ATL\textsubscript{sc}, Strategy Logic, ...);
- Defining interesting fragments of those logics;
- Obtaining practicable algorithms.
- Considering partial observation;
- Considering randomised strategies.
## Conclusions

- **ATL<sub>sc</sub>** is a very expressive, yet **decidable** extension of **ATL**;
- **QCTL** is a powerful extension of **CTL**;
- it is a **nice tool to understand temporal logics for games** (**ATL<sub>sc</sub>**, **Strategy Logic**, ...);

## Future directions

- Defining interesting fragments of those logics;
- Obtaining practicable algorithms.
- Considering partial observation;
- Considering randomised strategies.