Quantified CTL

Nicolas Markey
LSV – ENS Cachan

(joint work with François Laroussinie)

Nord-Pas-de-Calais – Belgium congress of mathematics

Valenciennes, 28 October 2013
Verification of computerised systems

- Computers are everywhere
Verification of computerised systems

- Computers are everywhere

- Bugs are everywhere...

News

Toyota to recall Prius hybrids over ABS software

See video, below

By Martyn Williams
February 1, 2010 01:35 AM ET

IDG News Service - Toyota plans to recall around 400,000 of its Prius hybrid cars to replace software that controls the antilock braking system (ABS), the auto maker said Tuesday.
Verification of computerised systems

- Computers are everywhere

- Bugs are everywhere...

- Verification should be everywhere!
Model checking and synthesis

**system:**

![Diagram of a system with two tanks and a pump](http://www.embedded.com)

**property:**

![Diagram of a property with two tanks and an XOR symbol](http://www.embedded.com)

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

**model-checking algorithm**

**yes/no**
Model checking and synthesis

system:

property:

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

synthesis algorithm
Outline of the presentation

1 Basics about CTL
   - expressing properties of reactive systems
   - efficient verification algorithms

2 Quantified CTL
   - CTL with quantification over atomic propositions
   - model checking and satisfiability are mostly decidable

3 Temporal logics for games: ATL and extensions
   - expressing properties of complex interacting systems
   - QCTL-based decision procedures for ATL_{sc}
Computation-Tree Logic (CTL)

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

- **atomic propositions:** 0, 0, ...

- **boolean combinators:** $\neg \varphi$, $\varphi \lor \psi$, $\varphi \land \psi$, ...

- **path quantifiers:** $E\varphi$, $\varphi$ $A\varphi$

- **temporal modalities:**
  - $X\varphi$, "next $\varphi$"
  - $U\psi$, "until $\psi$"
  - $\text{true}$
  - $F\varphi \equiv \neg \neg \varphi$
  - $G\varphi \equiv \varphi \land F\varphi$
  - $\text{always } \varphi$


Computation-Tree Logic (CTL)

- atomic propositions: $\bigcirc$, $\bigcirc$, ...
- boolean combinators: $\neg \varphi$, $\varphi \lor \psi$, $\varphi \land \psi$, ...
- path quantifiers:

$$
E \varphi \\
A \varphi 
$$

$$
\text{true} U \varphi \equiv \neg F \neg \varphi \\
\varphi U \psi \equiv \varphi U \psi \\
\varphi \text{ until } \psi \\
\varphi \text{ eventually } \psi
$$
Computation-Tree Logic (CTL)

- atomic propositions: , , ...
- boolean combinators: \( \neg \varphi \), \( \varphi \lor \psi \), \( \varphi \land \psi \), ...
- path quantifiers:
  - \( \text{E}\varphi \)
  - \( \text{A}\varphi \)
- temporal modalities:
  - \( \text{X}\varphi \)
  - \( \varphi \text{ U } \psi \)
Computation-Tree Logic (CTL)

- **atomic propositions:** ⊐, ⊐, ...
- **boolean combinators:** ¬ϕ, ϕ ∨ ψ, ϕ ∧ ψ, ...
- **path quantifiers:**

  - Eϕ
  - Aϕ

- **temporal modalities:**

  - Xϕ
  - ϕ U ψ
  - true U ϕ ≡ Fϕ
  - ¬ F ¬ϕ ≡ Gϕ

  “next ϕ”
  “ϕ until ψ”
  “eventually ϕ”
  “always ϕ”
Examples of CTL formulas

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.
Examples of CTL formulas

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

\[ \text{E F} \quad \text{blue is reachable} \]
Examples of CTL formulas

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

\( \text{E G(E F } \text{ ○) } \) there is a path along which ○ is always reachable
Examples of CTL formulas

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

\[ \text{A example: } \Box (\Diamond p) \]

there is a path along which \( p \) is always reachable.
Examples of CTL formulas

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

$$\mathsf{E}\,\mathsf{G}\,(\mathsf{E}\,\mathsf{F} \enspace \bigcirc)$$

there is a path along which $\bigcirc$ is always reachable
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.

[QS82] Queille, Sifakis. Specification and verification of concurrent systems in CESAR. SOP’82.
Examples of CTL formulas

In CTL*, we have no restriction on modalities and quantifiers.
Examples of CTL formulas

In CTL*, we have no restriction on modalities and quantifiers.

$$\exists G F$$ there is a path visiting infinitely many times
Examples of CTL formulas

In CTL*, we have no restriction on modalities and quantifiers.

\[ A(G F \circ \Rightarrow G F \bigcirc) \] any path that visits \( \circ \) infinitely many times, also visits \( \bigcirc \) infinitely many times
Examples of CTL formulas

In CTL*, we have no restriction on modalities and quantifiers.

**Theorem ([EH86])**

*CTL* model checking is PSPACE-complete.

Outline of the presentation

1 Basics about CTL
   - expressing properties of reactive systems
   - efficient verification algorithms

2 Quantified CTL
   - CTL with quantification over atomic propositions
   - model checking and satisfiability are mostly decidable

3 Temporal logics for games: ATL and extensions
   - expressing properties of complex interacting systems
   - QCTL-based decision procedures for ATL_{sc}
Quantified CTL

[Quantified CTL][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

QCTL extends CTL with propositional quantifiers

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

\[
\begin{align*}
\neg \text{EF} \circ \land \forall p. \, [\text{EF}(p \land \circ) \Rightarrow \text{AG} (\circ \Rightarrow p)]
\end{align*}
\]


Quantified CTL

QCTL extends CTL with **propositional quantifiers**

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

- $E F \bigcirc \land \forall p. [E F (p \land \bigcirc) \Rightarrow A G (\bigcirc \Rightarrow p)] \equiv \text{uniq}(\bigcirc)$

---


Quantified CTL

QCTL extends CTL with propositional quantifiers

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

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

\( \rightsquigarrow \) true if we label the Kripke structure;
\( \rightsquigarrow \) 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”:

\[
\begin{align*}
\text{EX}_1 \varphi &\equiv \text{EX} \varphi \land \forall p. \left[ \text{EX} (p \land \varphi) \Rightarrow \text{AX} (\varphi \Rightarrow p) \right] \\
\text{EX}_2 \varphi &\equiv \exists q. \left[ \text{EX}_1 (\varphi \land q) \land \text{EX}_1 (\varphi \land \neg q) \right]
\end{align*}
\]

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:

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

Expressiveness of QCTL

- QCTL can “count”:

  \[
  \text{E}X_1 \varphi \equiv \text{E}X \varphi \land \forall p. \ ([\text{E}X(p \land \varphi) \Rightarrow \text{A}X(\varphi \Rightarrow p)])
  \]

  \[
  \text{E}X_2 \varphi \equiv \exists q. \ ([\text{E}X_1(\varphi \land q) \land \text{E}X_1(\varphi \land \neg q)])
  \]

- QCTL can express (least or greatest) fixpoints:

  \[
  \mu T. \varphi(T) \equiv \exists t. \ ([\text{A}G(t \iff \varphi(t)) \land \forall t'. (\text{A}G(t' \iff \varphi(t')) \Rightarrow \text{A}G(t \Rightarrow t'))])
  \]

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

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

QCTL with structure semantics

<table>
<thead>
<tr>
<th>Theorem</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>QCTL satisfiability for the structure semantics is undecidable.</strong></td>
</tr>
</tbody>
</table>

*Proof*

Encode the problem of tiling finite square grids. Given a set of tiles, whether all finite square grids can be tiled is undecidable.

QCTL with structure semantics

Theorem

QCTL satisfiability for the structure semantics is undecidable.

Proof

Encode the problem of tiling finite square grids.
QCTL with structure semantics

**Theorem**

*QCTL satisfiability for the structure semantics is undecidable.*

**Proof**

Encode the problem of tiling finite square grids.
QCTL with structure semantics

Theorem

QCTL satisfiability for the structure semantics is undecidable.

Proof

Encode the problem of tiling finite square grids.

![Diagram of tiling problem]

QCTL with structure semantics

Theorem

QCTL satisfiability for the structure semantics is undecidable.

Proof

Encode the problem of tiling finite square grids.
QCTL with structure semantics

Theorem

QCTL satisfiability for the structure semantics is undecidable.

Proof

Encode the problem of tiling finite square grids.

Given a set of tiles, whether all finite square grids can be tiled is undecidable.

QCTL with structure semantics

Theorem

QCTL satisfiability for the structure semantics is undecidable.

Proof

Encode the problem of tiling finite square grids.
QCTL with structure semantics

Theorem

QCTL satisfiability for the structure semantics is undecidable.

Proof

Encode the problem of tiling finite square grids.

Given a set of tiles, whether all finite square grids can be tiled is undecidable.

QCTL with structure semantics

**Theorem**

QCTL satisfiability for the structure semantics is undecidable.

**Proof**

Encode the problem of tiling finite square grids.

Given a set of tiles, whether all finite square grids can be tiled is undecidable.
Theorem

QCTL satisfiability for the structure semantics is undecidable.

Proof

Reduction: is there a finite Kripke structure such that

\[\square\]
QCTL with structure semantics

Theorem

QCTL satisfiability for the structure semantics is undecidable.

Proof

Reduction: is there a finite Kripke structure such that

Given a set of tiles, whether all finite square grids can be tiled is undecidable.

QCTL with structure semantics

Theorem

QCTL satisfiability for the structure semantics is undecidable.

Proof

Reduction: is there a finite Kripke structure such that

\[ \text{each state has one or two successors} \]

\[ \text{AG}(\text{EX}_1 \text{true} \lor \text{EX}_2 \text{true}) \]
QCTL with structure semantics

Theorem

QCTL satisfiability for the structure semantics is undecidable.

Proof

Reduction: is there a finite Kripke structure such that

\[ \forall z. (E X E X z \Rightarrow A X E X z) \]

two successors of the same state have a common successor:

\[ A G(\forall z. (E X E X z \Rightarrow A X E X z)) \]
QCTL with structure semantics

Theorem

QCTL satisfiability for the structure semantics is undecidable.

Proof

Reduction: is there a finite Kripke structure such that

\[ ... \text{many more conditions} \ldots \]
QCTL with structure semantics

**Theorem**

QCTL satisfiability for the structure semantics is undecidable.

**Proof**

Reduction: is there a finite Kripke structure such that

- for any tiling, there is a position where the neighbouring tiles do not match

[Diagram of a Kripke structure with transitions labeled 'h']

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 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 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 tree automata:

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

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

\[
\delta(q_0, \bigcirc) = (q_0, q_1) \lor (q_1, q_0)
\]
\[
\delta(q_0, \bigcirc) = (q_1, q_1)
\]
\[
\delta(q_0, \bullet) = (q_2, q_2)
\]
\[
\delta(q_1, \bigstar) = (q_1, q_1)
\]
\[
\delta(q_2, \bigstar) = (q_2, q_2)
\]
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 tree automata:

\[
\begin{align*}
\delta(q_0, \bullet) &= (q_0, q_1) \lor (q_1, q_0) \\
\delta(q_0, \odot) &= (q_1, q_1) \\
\delta(q_0, \ast) &= (q_2, q_2) \\
\delta(q_1, \ast) &= (q_1, q_1) \\
\delta(q_2, \ast) &= (q_2, q_2)
\end{align*}
\]
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 tree automata:

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

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

\[
\delta(q_0, \bullet) = (q_0, q_1) \lor (q_1, q_0)
\]
\[
\delta(q_0, \circ) = (q_1, q_1)
\]
\[
\delta(q_0, \circlearrowleft) = (q_2, q_2)
\]
\[
\delta(q_1, \bigstar) = (q_1, q_1)
\]
\[
\delta(q_2, \bigstar) = (q_2, q_2)
\]
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 tree automata:

- $\delta(q_0, \bigcirc) = (q_0, q_1) \lor (q_1, q_0)$
- $\delta(q_0, \bigotimes) = (q_1, q_1)$
- $\delta(q_0, \blacklozenge) = (q_2, q_2)$
- $\delta(q_1, \blacklozenge) = (q_1, q_1)$
- $\delta(q_2, \blacklozenge) = (q_2, q_2)$
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 tree automata:

\[
\delta(q_0, \bigcirc) = (q_0, q_1) \lor (q_1, q_0)
\]
\[
\delta(q_0, \diamond) = (q_1, q_1)
\]
\[
\delta(q_0, \square) = (q_2, q_2)
\]
\[
\delta(q_1, \otimes) = (q_1, q_1)
\]
\[
\delta(q_2, \otimes) = (q_2, q_2)
\]

This automaton corresponds to \( E \bigcirc U \).
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.
Outline of the presentation

1. Basics about CTL
   - expressing properties of reactive systems
   - efficient verification algorithms

2. Quantified CTL
   - CTL with quantification over atomic propositions
   - model checking and satisfiability are mostly decidable

3. Temporal logics for games: ATL and extensions
   - expressing properties of complex interacting systems
   - QCTL-based decision procedures for $\text{ATL}_{sc}$
Concurrent games

A concurrent game is made of:
- a transition system;

![Diagram of a concurrent game with states q0, q1, q2 and transition arrows]

- player 1
- player 2
Reasoning about multi-agent systems

**Concurrent games**

A concurrent game is made of

- a transition system;
- a set of agents (or players);

![Diagram of concurrent game]

- $q_0$
- $q_1$
- $q_2$
Reasoning about multi-agent 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 multi-agent 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 *strategy* for a given player is a function telling what to play depending on what has happened previously.
Reasoning about open systems

Strategies

A strategy for a given player is a function telling what to play depending on what has happened previously.

Strategy for player □:
alternately go to ● and ○.
A strategy for a given player is a function telling what to play depending on what has happened previously.

Strategy for player ▼: alternately go to ○ and □.
### 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 \( \varphi \).

\[ \langle\langle A \rangle\rangle \varphi \]

\[ \langle\langle A \rangle\rangle F \]

\[ \langle\langle A \rangle\rangle G (\langle\langle A \rangle\rangle F) \]

Theorem
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
Model checking ATL is PTIME-complete.

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 $\varphi$.

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
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 \( \varphi \).

\[ \text{Theorem} \]
Model checking ATL is \text{PTIME}-complete.

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: 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 \( \varphi \).

\[ \langle \Diamond \rangle \text{ F } \langle \Box \rangle \text{ F } \langle \Diamond \rangle \text{ G } (\langle \Box \rangle \text{ F } p) \equiv \langle \Diamond \rangle \text{ G } p \]

Theorem

Model checking ATL is PTIME-complete.

ATL with strategy contexts

Consider the following strategy of Player 0: "always go to 0"; in the remaining tree, Player 0 can always enforce a visit to 0.

\[ \langle \diamond \rangle G(\langle \Box \rangle F 0) \]
consider the following strategy of Player $\bigcirc$: “always go to $\square$";
Consider the following strategy of Player \(\bigcirc\): "always go to \(\square\);"
consider the following strategy of Player \( \bigcirc \): “always go to \( \square \); in the remaining tree, Player \( \square \) can always enforce a visit to \( \bigcirc \).
What ATL$_{sc}$ can express

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

$$
\langle \cdot \text{Server} \cdot \rangle \text{ G} \begin{bmatrix}
\bigwedge_{c \in \text{Clients}} \langle \cdot c \cdot \rangle F \text{ access}_c \\
\neg \bigwedge_{c \neq c'} \text{ access}_c \land \text{ access}_{c'}
\end{bmatrix}
$$
What ATL\textsubscript{sc} can express

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

- **Existence of Nash equilibria**:
  \[
  \langle \cdot A_1, ..., A_n \rangle \ \bigwedge_{i} \left( \langle \cdot A_i \rangle \varphi_{A_i} \Rightarrow \varphi_{A_i} \right)
  \]
What $\text{ATL}_{sc}$ can express

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

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

- **Existence of Nash equilibria**:

$$\langle A_1, ..., A_n \rangle \bigwedge_i (\langle A_i \rangle \varphi_{A_i} \Rightarrow \varphi_{A_i})$$

- **Existence of dominating strategy**:

$$\langle A \rangle [B] (\neg \varphi \Rightarrow [A] \neg \varphi)$$
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 $\text{Next}(\cdot, A, m_i^A)$ of states that can be reached from $\cdot$ when player $A$ plays $m_i^A$.

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 $m_i^A$.

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

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

- this corresponds to a strategy: $A G (m_i^A \Leftrightarrow \bigwedge \neg m_j^A)$;
- the outcomes all satisfy $\varphi$:
\[
A [G(q \land m_i^A \Rightarrow X \text{Next}(q, A, m_i^A)) \Rightarrow \varphi].
\]

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

- player $A$ has moves $m_1^A$, ..., $m_n^A$;
- 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 $m_i^A$.

**Corollary**

$\text{ATL}_{sc}$ model checking is decidable.

**Corollary**

$\text{ATL}_{sc}^0$ (memoryless quantification) model checking is decidable.

What about satisfiability?

**Theorem**

*QCTL satisfiability is decidable.*

But

**Theorem (TW12)**

ATL*sc* satisfiability is undecidable.

Why?

The translation from ATL*sc* to QCTL assumes that the game structure is fixed!

What about satisfiability?

**Theorem**

_\textit{QCTL} satisfiability is decidable._

But

**Theorem (TW12)**

_\textit{ATL}_{sc} satisfiability is undecidable._

What about satisfiability?

Theorem

*QCTL* satisfiability is decidable.

But

Theorem (TW12)

*ATL_{sc}* satisfiability is undecidable.

Why?

The translation from *ATL_{sc}* to QCTL assumes that the game structure is fixed!

Satisfiability for turn-based games

Theorem (LM13b)

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

Satisfiability for turn-based games

Theorem (LM13b)

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

- player $\square$ has moves $\bigcirc$, $\bigcirc$, and $\bigcirc$.
- a strategy can be encoded by marking some of the nodes of the tree with proposition $\text{mov}_A$.

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

$\exists \text{mov}_A$.
- it corresponds to a strategy: $\text{A} \ G(\text{turn}_A \Rightarrow \text{E} \ X_1 \ \text{mov}_A)$;
- the outcomes all satisfy $\varphi$: $\text{A}[G(\text{turn}_A \land X \ \text{mov}_A) \Rightarrow \varphi]$.

# Satisfiability for turn-based games

**Theorem (LM13b)**

*When restricted to turn-based games, $\text{ATL}_{sc}$ satisfiability is decidable.*

**Theorem**

*Model checking $\text{ATL}_{sc}$ with only memoryless quantification is PSPACE-complete.*

What about Strategy Logic? [CHP07,MMV10]

**Strategy logic**

Explicit quantification over strategies + strategy assignment

**Example**

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

Strategy logic can also be translated into QCTL.

**Theorem**

- Strategy-logic satisfiability is decidable when restricted to turn-based games.
- Memoryless strategy-logic satisfiability is undecidable.

Conclusions and future works

Conclusions

- QCTL is a powerful extension of CTL;
- it is equivalent to MSO over finite graphs and regular trees;
- it is a nice tool to understand temporal logics for games (ATL with strategy contexts, Strategy Logic, ...);
Conclusions and future works

**Conclusions**
- QCTL is a powerful extension of CTL;
- it is equivalent to MSO over finite graphs and regular trees;
- it is a nice tool to understand temporal logics for games (ATL with strategy contexts, Strategy Logic, ...);

**Future directions**
- Defining interesting (expressive yet tractable) fragments of those logics;
- Obtaining practicable algorithms.
- Considering randomised strategies.