Temporal logics for multi-player games

Nicolas Markey
LSV – ENS Cachan

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

French Symposium on Games – Theory and Applications
Paris, May 27, 2015
Verification of computerized systems

- Computers are everywhere
Verification of computerized systems

- Computers are everywhere

- Bugs are everywhere...

News
Toyota to recall Prius hybrids over ABS software
See video, below
By Marilyn Williams
February 9, 2006 04:39 AM ET  Comments (6)  Recommend (15)
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.
Model checking and synthesis

system:

[http://www.embedded.com]

property

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

model-checking algorithm

yes/no
Model checking and synthesis

system:

[http://www.embedded.com]

property

\[ \text{AG}(\neg B.\text{overfull} \land \neg B.\text{dried\_up}) \]
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}
   - 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}
   - algorithms for ATL_{sc}

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

- atomic propositions: $\bigcirc_1$, $\bigcirc_2$, ...

Computation-Tree Logic (CTL)

- atomic propositions: $\Diamond$, $\lozenge$, ...
- boolean combiners: $\neg \varphi$, $\varphi \lor \psi$, $\varphi \land \psi$, ...

- temporal modals: $X \varphi$ ("next \varphi"), $\varphi U \psi$ ("\varphi until \psi"), $\varphi$ ("eventually \varphi"), $\top U \varphi \equiv F \neg \neg \varphi \equiv G \varphi$ ("always \varphi")
Computation-Tree Logic (CTL)

- **atomic propositions:** $\bigcirc$, $\Box$, ...
- **boolean combinators:** $\neg \varphi$, $\varphi \lor \psi$, $\varphi \land \psi$, ...
- **temporal modalities:**
  
  $X \varphi$  
  \[ \begin{array}{cc}
  \text{next } \varphi \end{array} \]

  $\varphi \mathrel{U} \psi$  
  \[ \begin{array}{cc}
  \text{“} \varphi \text{ until } \psi \text{”} \end{array} \]
Computation-Tree Logic (CTL)

- atomic propositions: $\circ$, $\circ$, ...
- boolean combinators: $\neg \varphi$, $\varphi \lor \psi$, $\varphi \land \psi$, ...
- temporal modalities:

  - $X \varphi$
  - $\varphi \mathcal{U} \psi$
  - $\text{true} \mathcal{U} \varphi \equiv F \varphi$
  - $\neg F \neg \varphi \equiv G \varphi$

  "next $\varphi$"
  "$\varphi$ until $\psi$"
  "eventually $\varphi$"
  "always $\varphi$"
Computation-Tree Logic (CTL)

- **atomic propositions:** .setFill, .setFill, ...
- **boolean combinators:** \( \neg \varphi \), \( \varphi \lor \psi \), \( \varphi \land \psi \), ...
- **temporal modalities:**
  - \( X \varphi \)
  - \( \varphi \mathrel{U} \psi \)
  - true \( \mathrel{U} \varphi \equiv F \varphi \)
  - \( \neg F \neg \varphi \equiv G \varphi \)

- **path quantifiers:**
  - \( E \varphi \)
  - \( A \varphi \)
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.

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

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

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

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

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

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

$$\text{EG}(\neg \circ \land \text{EF}_{p} \circ)$$

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

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

\[ \text{EG}(\neg \bigcirc \land \text{EF}_p \bigcirc) \]  

there is a path along which \( \bigcirc \) is always reachable, but never reached

\[ \text{p} \]  

\[ \text{p} \]  

\[ \text{p} \]  

\[ \text{p} \]
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] Queue, Sifakis. Specification and verification of concurrent systems in CESAR. SOP’82.
Reasoning about open systems

A concurrent game on a graph 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

Games on graphs

A concurrent game on a graph is made of

- a transition system;
Reasoning about open systems

Games on graphs

A concurrent game on a graph is made of

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

\[
q_0 \quad q_1 \quad q_2
\]
Reasoning about open systems

Games on graphs

A concurrent game on a graph 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

Games on graphs

A **concurrent game on a graph** 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.

**Example**

Strategy for player □:
alternately go to ⬜ and ⬜️.

![Diagram showing strategy for a given player]
Reasoning about open systems

**Strategies**

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

**Example**

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 \(\varphi\).

ATL extends CTL with strategy quantifiers

⟨⟨A⟩⟩ϕ expresses that A has a strategy to enforce ϕ.

Temporal logics for games: ATL

ATL extends CTL with strategy quantifiers

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

Model checking ATL is PTIME-complete.

Temporal logics for games: ATL

ATL extends CTL with \textbf{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 \( \varphi \).

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 \(\text{PTIME}\)-complete.

Temporal logics for games: ATL

ATL extends CTL with strategy quantifiers

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

\[\langle A \rangle \varphi\]

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

\[\langle\Diamond\rangle F \equiv \langle\Diamond\rangle G p\]

\[\langle\Diamond\rangle p\]

\[\langle\Box\rangle p\]

\[\langle\Diamond\Diamond\rangle F p\]

\[\langle\Diamond\Box\rangle F p\]

\[\langle\Box\Diamond\rangle F p\]

\[\langle\Box\Box\rangle F p\]

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

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.*

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 $\text{ATL}_{sc}$
   - algorithms for $\text{ATL}_{sc}$

4 Conclusions and future works
Consider the following strategy of Player: "always go to ..."
consider the following strategy of Player $\bigcirc$: “always go to $\Box$";
consider the following strategy of Player $\bigcirc$: "always go to $\Box$";

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

consider the following strategy of Player $\bigcirc$: “always go to $\Box$; in the remaining tree, Player $\square$ can always enforce a visit to $\bigcirc$. 

\[ \langle \bigcirc \rangle \text{G}(\langle \Box \rangle \text{F}\bigcirc) \]
ATL with strategy contexts

**Definition**

\( \text{ATL}_{sc} \) has two new strategy quantifiers: \( \langle A \rangle \varphi \) and \( \langle A \mid \varphi \).  
- \( \langle A \rangle \) is similar to \( \langle A \rangle \) but **assigns** the corresponding strategy to \( A \) for evaluating \( \varphi \);
ATL with strategy contexts

Definition

$\text{ATL}_{sc}$ has two new strategy quantifiers: $\langle A \rangle \varphi$ and $\langle A \vert \varphi$. 

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

- $\langle A \vert$ drops the assigned strategies for $A$. 

$\langle A \rangle \varphi \equiv \neg \langle A \rangle \neg \varphi$ 

$\langle A \rangle \varphi$ states that any strategy for $A$ has an outcome along which $\varphi$ holds.
Definition

$\text{ATL}_{sc}$ has two new strategy quantifiers: $\langle A \cdot \rangle \varphi$ and $\langle A \rangle \varphi$.

- $\langle A \cdot \rangle$ is similar to $\langle A \rangle$ but assigns the corresponding strategy to $A$ for evaluating $\varphi$;
- $\langle A \rangle$ drops the assigned strategies for $A$.
- $[A \cdot]$ is dual to $\langle A \cdot \rangle$:

  \[[A \cdot] \varphi \equiv \neg \langle A \cdot \rangle \neg \varphi\]

$[A \cdot] \varphi$ states that any strategy for $A$ has an outcome along which $\varphi$ holds.
What $\text{ATL}_{sc}$ can express

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

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

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

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

- **Existence of Nash equilibria**:

  \[
  \langle \cdot A_1, \ldots, A_n \cdot \rangle \wedge \bigwedge_i \left( \langle \cdot A_i \cdot \rangle \varphi_{A_i} \Rightarrow \varphi_{A_i} \right)
  \]
What $\text{ATL}_{sc}$ can express

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

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

- **Existence of dominating strategy**:
  \[
  \langle A \cdot \rangle [B] (\neg \varphi \Rightarrow [A] \neg \varphi)
  \]
More expressiveness results

Theorem

- $\text{ATL}_{sc}$ is strictly more expressive than $\text{ATL}$;
- The operator $\langle A \rangle$ does not add expressive power.
More expressiveness results

Theorem

- $\text{ATL}_{sc}$ is strictly more expressive than $\text{ATL}$;
- The operator $\langle A \rangle$ does not add expressive power.

Proof

$\langle 1 \rangle (\langle 2 \rangle \ X a \land \langle 2 \rangle \ X b)$ is only true in the second game. But $\text{ATL}$ cannot distinguish between these two games.
Algorithms for checking $\text{ATL}_{sc}$ properties

Tree-automata approach

A strategy is encoded as a labelling of the unwinding tree; we can mark outcomes corresponding to selected strategies; we can build a tree automaton accepting all trees that can be labelled with correct strategies.
A strategy is encoded as a labelling of the unwinding tree;
Algorithms for checking $\text{ATL}_{sc}$ properties

Tree-automata approach

- A strategy is encoded as a labelling of the unwinding tree;
- We can mark outcomes corresponding to selected strategies;
Algorithms for checking $\text{ATL}_{sc}$ properties

Tree-automata approach

A strategy is encoded as a labelling of the unwinding tree;
We can mark outcomes corresponding to selected strategies;
We can build a tree automaton accepting all trees that can be labelled with correct strategies.
Verifying $\text{ATL}_{sc}$ properties

**Theorem**

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

Verifying $\text{ATL}_{sc}$ properties

<table>
<thead>
<tr>
<th>Theorem</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{ATL}_{sc}$ model checking is <strong>decidable</strong> (k-EXPTIME-complete).</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Theorem</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{ATL}_{sc}$ satisfiability checking is <strong>undecidable</strong>.</td>
<td></td>
</tr>
</tbody>
</table>

Verifying $\text{ATL}_{sc}$ properties

**Theorem**

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

**Theorem**

$\text{ATL}_{sc}$ **satisfiability checking** is **undecidable**.

**Theorem**

$\text{ATL}_{sc}$ **satisfiability checking** is **decidable** when restricting to turn-based games.


Conclusions and future works

Conclusions

- ATL mainly expresses properties of zero-sum games;
- ATL\textsubscript{sc} can mix collaboration and antagonism:
  - powerful logic to capture interesting properties;
  - high complexity.

Future directions

- interesting (expressive yet tractable) fragments of the logic;
- practicable algorithms.
- incomplete information, randomised strategies.
Conclusions and future works

Conclusions

- **ATL** mainly expresses properties of zero-sum games;
- **ATL\(_{sc}\)** can mix collaboration and antagonism:
  - powerful logic to capture interesting properties;
  - high complexity.

Future directions

- **interesting** (expressive yet tractable) **fragments** of the logic;
- **practicable** algorithms.

- **incomplete information**, randomised strategies.