Temporal logics for multi-agent systems

Nicolas Markey
LSV – ENS Cachan

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

Centre Fédéré en Vérification
Brussels, January 29, 2016
Model checking and synthesis

system:

property

A G (¬B. overfull ∧ ¬B. dried_up)

model-checking algorithm

yes/no
Model checking and synthesis

system:

[http://www.embedded.com]

property

AG(¬B.overfull ∧ ¬B.dried_up)
Outline of the presentation

1. Introduction

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

3. ATL with strategy contexts
   - specifying properties of complex interacting systems
   - expressive power of ATL_{sc}
   - translation into Quantified CTL (QCTL)
   - algorithms for ATL_{sc}

4. Strategy Logic

5. 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. ATL with strategy contexts
   - specifying properties of complex interacting systems
   - expressive power of ATL$_{sc}$
   - translation into Quantified CTL (QCTL)
   - algorithms for ATL$_{sc}$

4. Strategy Logic

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

- atomic propositions: $\bigcirc$, $\bigotimes$, ...

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

- temporal modalities: $X \phi$, $\phi U \psi$, $\phi U \psi$...

- path quantifiers: $E \phi$, $A \phi$ ...

$\phi U \psi \equiv \neg F \neg \phi \equiv G \phi \equiv \psi$
Computation-Tree Logic (CTL)

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

- atomic propositions: \( \text{true}, \text{false}, \ldots \)
- boolean combinators: \( \neg \varphi, \varphi \lor \psi, \varphi \land \psi, \ldots \)
- temporal modalities:
  \[
  X \varphi \quad \varphi \text{ U } \psi \quad \text{“next } \varphi\text{”} \quad \text{“} \varphi \text{ until } \psi\text{”}
  \]
Computation-Tree Logic (CTL)

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

\[ EF \text{ is reachable} \]
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.

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

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

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

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.

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

There is a path along which \( p \) is always reachable, but never reached.

Diagram:  
- Blue circle: \( p \) is always reachable.  
- Green circle: \( p \) is never reached.

Graph:  
- Red circle with checkmark: \( p \) is true.  
- Orange circle with checkmark: \( p \) is true.

Arrow:  
- Blue arrow: \( p \) is always reachable.

Closed loop:  
- Green circle: \( p \) is 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

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.
Concurrent games

A concurrent game is made of

- a transition system;

\[
\begin{array}{ccc}
q_0 & \rightarrow & q_1 \\
\uparrow & & \downarrow \\
q_2 & \rightarrow & q_1 \\
\end{array}
\]
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.

\[
\begin{array}{c}
q_0 \\
q_1 \\
q_2
\end{array}
\]

\[
\begin{array}{c|c|c|c}
& \text{player 1} & \text{player 2} \\
\hline
q_0 & \text{\footnotesize\includegraphics{hand1}} & \text{\footnotesize\includegraphics{hand2}} \\
q_2 & \text{\footnotesize\includegraphics{hand3}} & \text{\footnotesize\includegraphics{hand4}} \\
q_1 & \text{\footnotesize\includegraphics{hand5}} & \text{\footnotesize\includegraphics{hand6}}
\end{array}
\]
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.
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](image-url)
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 ).
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 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 ⬇️ 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**

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 🟦.
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 □.
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 🟥.

Diagram showing a tree structure with nodes and edges, illustrating the strategy.
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):

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

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.

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

\[ \langle A \rangle \varphi \] 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 $\varphi$.

$\langle A \rangle \varphi$ 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 $\varphi$.

$$p$$

MyGraph

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.

Theorem ([LMO08])

In PTIME only if the transition table is given explicitly

(size \( |\text{Moves}| \cdot |\text{Agt}| \))

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.

Theorem ([LMO08])

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

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. ATL with strategy contexts
   - specifying properties of complex interacting systems
   - expressive power of ATL_{sc}
   - translation into Quantified CTL (QCTL)
   - algorithms for ATL_{sc}

4. Strategy Logic

5. Conclusions and future works
Example

\[ \langle \square \rangle \mathsf{G} (\langle \square \rangle \mathsf{F} \top) \]

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

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 in always plays to .

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

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

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

- Player \( \Diamond \) in \( \Box \) 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.
ATL with strategy contexts [BDLM09, DLM10]

Example

Player $\bigcirc$ in $\Box$ 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.
ATL with strategy contexts \([\text{BDLM09,DLM10}]\)

Example

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

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.
Definition

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

- $⟨·A·⟩ \varphi$ is similar to $⟨⟨A⟩⟩ \varphi$ 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

\( \text{ATL}_{sc} \) 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 \( \varphi \);

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

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

- $\langle \cdot \overline{A} \cdot \rangle \varphi \equiv \langle \cdot \text{Agt} \setminus A \cdot \rangle \varphi$
  (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**

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

- \( \langle A \rangle \varphi \) 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;

- \( \langle A \rangle \varphi \) drops the assigned strategies for \( A \).

- \( [A] \varphi \) is dual to \( \langle A \rangle \varphi \):

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

Definition

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

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

Definition

Semantics of ATL strategy quantifier:

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

Definition

ATL<sub>sc</sub> 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<sub>sc</sub> 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:

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

newly selected strategies added to the context:

$$\sigma_A \circ \sigma_B : \begin{align*}
a &\mapsto \sigma_A(a) & \text{if } a \in A \setminus B \\
b &\mapsto \sigma_B(b) & \text{if } b \in B \setminus A \\
c &\mapsto \sigma_A(c) & \text{if } c \in B \cap A
\end{align*}$$
What \( \text{ATL}_{sc} \) can express

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

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

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

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

- **Existence of Nash equilibria**:

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

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

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

- **Existence of Nash equilibria**:

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

- **Existence of dominating strategy**:

  $$
  \langle \cdot A \cdot \rangle \ [B] \ (\neg \varphi \Rightarrow [A] \neg \varphi)
  $$
Expressiveness of $\text{ATL}_{sc}$

**Theorem**

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

**Theorem**

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

**Proof**

$$\langle A \rangle \varphi \equiv (\{\emptyset\} \langle A \rangle \hat{\varphi})$$
Theorem

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

Proof

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

1. Introduction

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

3. ATL with strategy contexts
   - specifying properties of complex interacting systems
   - expressive power of ATL_{sc}
   - translation into Quantified CTL (QCTL)
   - algorithms for ATL_{sc}

4. Strategy Logic

5. 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 \cite{ES84,Kup95,Fre01}

QCTL extends CTL with propositional quantifiers

\[ \exists p. \varphi \text{ means that there exists a labelling of the model with } p \text{ under which } \varphi \text{ holds.} \]

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

\cite{ES84} Emerson and Sistla. Deciding Full Branching Time Logic. Information & Control, 1984.
\cite{Kup95} Kupferman. Augmenting Branching Temporal Logics with Existential Quantification... CAV, 1995.
\cite{Fre01} French. Decidability of Quantified Propositional Branching Time Logics. AJCAI, 2001.
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.

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

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

\( \Rightarrow \) true if we label the Kripke structure;

\( \Rightarrow \) 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:

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

\[ p \]

\[ \models \varphi \]
Expressiveness of QCTL

- QCTL can “count”:

\[
\begin{align*}
\mathbf{E} \mathbf{X}_1 \varphi & \equiv \mathbf{E} \mathbf{X} \varphi \land \forall p. \ [\mathbf{E} \mathbf{X}(p \land \varphi) \Rightarrow \mathbf{A} \mathbf{X}(\varphi \Rightarrow p)] \\
\mathbf{E} \mathbf{X}_2 \varphi & \equiv \exists q. \ [\mathbf{E} \mathbf{X}_1(\varphi \land q) \land \mathbf{E} \mathbf{X}_1(\varphi \land \neg q)]
\end{align*}
\]

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”:

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

- QCTL can express (least or greatest) fixpoints:

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

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.
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:
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:

$$\delta(q_0, \bigcirc) = (q_0, q_1) \lor (q_1, q_0)$$
$$\delta(q_0, \bigcirc) = (q_1, q_1)$$
$$\delta(q_0, \bullet) = (q_2, q_2)$$
$$\delta(q_1, \bigstar) = (q_1, q_1)$$
$$\delta(q_2, \bigstar) = (q_2, q_2)$$
Theorem

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

Proof

Using (alternating) parity tree automata:

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

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

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

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

- \( \delta(q_0, \text{□}) = (q_0, q_1) \lor (q_1, q_0) \)
- \( \delta(q_0, \text{●}) = (q_1, q_1) \)
- \( \delta(q_0, \text{□}) = (q_2, q_2) \)
- \( \delta(q_1, \text{□}) = (q_1, q_1) \)
- \( \delta(q_2, \text{□}) = (q_2, q_2) \)
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:

\[
\delta(q_0, \bullet) = (q_0, q_1) \lor (q_1, q_0)
\]
\[
\delta(q_0, \circ) = (q_1, q_1)
\]
\[
\delta(q_0, \blacklozenge) = (q_2, q_2)
\]
\[
\delta(q_1, \bigodot) = (q_1, q_1)
\]
\[
\delta(q_2, \bigodot) = (q_2, q_2)
\]
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:

\[
\begin{align*}
\delta(q_0, \bullet) &= (q_0, q_1) \lor (q_1, q_0) \\
\delta(q_0, 0) &= (q_1, q_1) \\
\delta(q_0, 1) &= (q_2, q_2) \\
\delta(q_1, 0) &= (q_1, q_1) \\
\delta(q_2, 0) &= (q_2, q_2)
\end{align*}
\]
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:

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

This automaton corresponds to \(E \bullet U \circ\)
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 $\text{ATL}_{sc}$ 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 $m_i^A$.

Translating ATL\textsubscript{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 $m_i^A$.

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

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

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

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

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 $\text{Next}(\circ, A, m_i^A)$ of states that can be reached from $\circ$ when player $A$ plays $m_i^A$.

**Corollary**

$\text{ATL}_{sc}$ model checking is decidable, with non-elementary complexity.

**Corollary**

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

Hardness of model checking $\text{ATL}_{sc}$

**Encode QLTL satisfiability**

Example: $\Phi = \forall p_1. \exists p_2. \ G(p_2 \iff X p_1).$
Hardness of model checking \( \text{ATL}_{sc} \)

Encode QLTL satisfiability

Example:

\[ \Phi = \forall p_1. \exists p_2. \ G(p_2 \iff X p_1). \]
Encode QLTL satisfiability

Example: \( \Phi = \forall p_1. \exists p_2. \ G(p_2 \iff X p_1). \)
Encode QLTL satisfiability

Example: \[ \Phi = \forall p_1. \exists p_2. \ G(p_2 \iff X p_1). \]
Hardness of model checking $\text{ATL}_{sc}$

Encode QLTL satisfiability

Example: $\Phi = \forall p_1. \exists p_2. \ G(p_2 \iff X p_1).$
Encode QLTL satisfiability

Example: \( \Phi = \forall p_1. \exists p_2. \ G(p_2 \iff X p_1). \)
What about satisfiability?

Theorem

\textit{QCTL satisfiability is decidable (for the tree semantics).}
What about satisfiability?

**Theorem**

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

But

**Theorem ([TW12])**

*ATL_{sc} satisfiability is undecidable.*

---

What about satisfiability?

**Theorem**

QCTL satisfiability is decidable (for the tree semantics).

But

**Theorem ([TW12])**

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

Why?

The translation from $\text{ATL}_{sc}$ 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 movA.

⟨·A·⟩ φ can be encoded as follows:

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

- it corresponds to a strategy: \[ AG(\text{turn}_A \Rightarrow EX_1 \text{mov}_A); \]
- the outcomes all satisfy φ: \[ A[G(\text{turn}_A \land X \text{mov}_A) \Rightarrow \varphi]. \]

Outline of the presentation

1. Introduction

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

3. ATL with strategy contexts
   - specifying properties of complex interacting systems
   - expressive power of ATL\textsubscript{sc}
   - translation into Quantified CTL (QCTL)
   - algorithms for ATL\textsubscript{sc}

4. Strategy Logic

5. Conclusions and future works
Strategy Logic [CHP07, MMV10]

Strategy logic

Explicit quantification and binding of strategies

Strategy Logic [CHP07, MMV10]

**Strategy logic**

Explicit quantification and binding of strategies

**Definition**

Strategy Logic (SL) formulas are built using:

- strategy quantifications: $\exists \sigma. \psi$;


Strategy Logic [CHP07, MMV10]

Strategy logic

Explicit quantification and binding of strategies

Definition

Strategy Logic (SL) formulas are built using:

- strategy quantifications: $\exists \sigma. \psi$;
- strategy bindings: $\text{bind}(A \mapsto \sigma). \varphi$;

Strategy Logic [CHP07,MMV10]

**Strategy logic**

Explicit quantification and binding of strategies

**Definition**

Strategy Logic (SL) formulas are built using:

- strategy quantifications: $\exists \sigma. \psi$;
- strategy bindings: $\text{bind}(A \mapsto \sigma). \varphi$;
- **LTL** to express properties of paths (outcomes);

Strategy Logic [CHP07,MMV10]

**Definition**

Strategy Logic (SL) formulas are built using:

- **strategy quantifications**: \( \exists \sigma. \psi; \)
- **strategy bindings**: \( \text{bind}(A \mapsto \sigma). \varphi; \)
- **LTL** to express properties of paths (outcomes);

**Example**

\[ \exists \sigma. \text{bind}(A \mapsto \sigma). \varphi \]
Strategy Logic [CHP07,MMV10]

Definition

Strategy Logic (SL) formulas are built using:
- strategy quantifications: \( \exists \sigma. \psi \);
- strategy bindings: \( \text{bind}(A \mapsto \sigma). \varphi \);
- LTL to express properties of paths (outcomes);

Example

\[
\exists \sigma. \text{bind}(A \mapsto \sigma). \varphi
\]

\[
\exists \sigma. \forall \sigma'. \text{bind}(A \mapsto \sigma). \text{bind}(B \mapsto \sigma'). \varphi
\]
Strategy Logic [CHP07,MMV10]

Definition

Strategy Logic (SL) formulas are built using:

- strategy quantifications: \( \exists \sigma. \psi \);
- strategy bindings: \( \text{bind}(A \mapsto \sigma). \varphi \);
- LTL to express properties of paths (outcomes);

Example

\[ \exists \sigma. \text{bind}(A \mapsto \sigma). \varphi \]
\[ \exists \sigma. \forall \sigma'. \text{bind}(A \mapsto \sigma). \text{bind}(B \mapsto \sigma'). \varphi \equiv \langle \cdot A \cdot \rangle \varphi \]
Strategy Logic [CHP07, MMV10]

Definition

Strategy Logic (SL) formulas are built using:
- strategy quantifications: $\exists \sigma. \psi$;
- strategy bindings: $\text{bind}(A \mapsto \sigma). \varphi$;
- LTL to express properties of paths (outcomes);

Example

$$\exists \sigma. \text{bind}(A \mapsto \sigma). \varphi$$
$$\exists \sigma. \forall \sigma'. \text{bind}(A \mapsto \sigma). \text{bind}(B \mapsto \sigma'). \varphi \equiv \langle \cdot A \rangle \varphi$$
$$\exists \sigma. \text{bind}(A \mapsto \sigma). \text{bind}(B \mapsto \sigma). \varphi$$

Strategy Logic [CHP07, MMV10]

Definition

Strategy Logic (SL) formulas are built using:

- **Strategy quantifications**: \( \exists \sigma. \psi \);
- **Strategy bindings**: \( \text{bind}(A \mapsto \sigma). \varphi \);
- **LTL** to express properties of paths (outcomes);

Example

\[
\exists \sigma. \text{bind}(A \mapsto \sigma). \varphi \\
\exists \sigma. \forall \sigma'. \text{bind}(A \mapsto \sigma). \text{bind}(B \mapsto \sigma'). \varphi \equiv \langle A \rangle \varphi \\
\exists \sigma. \text{bind}(A \mapsto \sigma). \text{bind}(B \mapsto \sigma). \varphi \\
\exists \sigma. A \mathbf{G}(\text{bind}(A \mapsto \sigma). \varphi)
\]


What does $\exists \sigma. \mathbf{AG}(\text{bind}(A \mapsto \sigma). \varphi)$ mean?
Semantics of SL [BGM16]

What does $\exists \sigma. \text{AG}(\text{bind}(A \mapsto \sigma). \varphi)$ mean?

$\sigma$ is selected here...

What does $\exists \sigma. \ AG(\text{bind}(A \rightarrow \sigma). \ \varphi)$ mean?

$\sigma$ is selected here...

... but is applied there

What does $\exists \sigma. \text{A G}(\text{bind}(A \mapsto \sigma). \varphi)$ mean?

What is the history of $\sigma$ when a player starts applying it?
If history starts when selecting strategies...

Theorem

*Strategy logic can be translated into QCTL.*
If history starts when selecting strategies...

**Theorem**

*Strategy logic can be translated into QCTL.*

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

**SL can be translated as follows:**

- encoding of $\exists \sigma. \psi$:
  \[
  \exists m_1^\sigma \exists m_2^\sigma \ldots \exists m_k^\sigma. \ A G (m_i^\sigma \iff \bigwedge \neg m_j^\sigma)
  \]

- encoding of $\varphi \in \text{LTL}$ (under **full binding** $\alpha: \text{Ag} \rightarrow \text{Strat}$):
  \[
  A \left[ G (q \land m_i^{\alpha(A)} \Rightarrow X \text{Next}(q, A, m_i^{\alpha(A)})) \Rightarrow \varphi \right]
  \]
If history starts when selecting strategies...

**Theorem**

Strategy logic can be translated into QCTL.

**Theorem ([CHP07, MMV10, DLM12, LM13])**

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


If history starts when applying strategies...

... then strategies cannot easily be stored on the execution tree
If history starts when applying strategies...

... then strategies cannot easily be stored on the execution tree

Theorem

SL model checking is **undecidable** in floating semantics.
If history starts when applying strategies...

... then strategies cannot easily be stored on the execution tree

**Theorem**

*SL model checking is *undecidable* in *floating semantics.*
If history starts when applying strategies...

... then strategies cannot easily be stored on the execution tree

**Theorem**

*SL model checking is undecidable in floating semantics.*

Strategies for \( \bigcirc \) and \( \square \) characterized by the integer representing the first time they play to \( \bot \).
If history starts when applying strategies...

... then strategies cannot easily be stored on the execution tree.

**Theorem**

*SL model checking is undecidable in floating semantics.*

Strategies for $\bigcirc$ and $\blacksquare$ characterized by the integer representing the first time they play to $\bot$.

Checking that two strategies $\sigma_\bigcirc$ and $\sigma_\blacksquare$ represent the same integer:

$$G(a \Rightarrow X \bigcirc a) \land F[(\bigcirc \land X \bot) \land [\bigcirc \cdot X X \bot]] \quad (\varphi_=)$$
If history starts when applying strategies...

Encode run of a deterministic 2-counter machine.
If history starts when applying strategies...

Encode run of a deterministic 2-counter machine $\mathcal{M}$:

- Player $\bigcirc$ plays a strategy that mimics the run of $\mathcal{M}$;
- Player $\blacksquare$ checks validity of simulation.
If history starts when applying strategies...

Encode run of a deterministic 2-counter machine $M$:

- $s$: if $c==0$ then goto $s'$ else goto $s''$

$$
\exists G \left[ \bigwedge_{s \text{ s.t. } \delta(s)=(c,s',s'')} s \Rightarrow \left( \left[ \langle \square \rangle (X \circ X \land X X \, \bot) \right] \iff \left( \langle \square \rangle \, X \, X \, s' \right) \right) \right]
$$
If history starts when applying strategies...

Encode run of a deterministic 2-counter machine $M$:

- $s$: if $c==0$ then goto $s'$ else goto $s''$
- $s$: $c++$; goto $s'$

$$\square \text{G} \lbrack \square s \Rightarrow \exists \sigma_{\text{count}}. \langle \cdot \rangle X (\mathbb{C} \land \text{bind}(\square \mapsto \sigma_{\text{count}}). \varphi_\equiv) \land \langle \cdot \rangle X X (\square s' \land X (\mathbb{C} \land \text{bind}(\square \mapsto \sigma_{\text{count}}). \varphi_{+1}))$$
Conclusions and future works

Conclusions

- **ATL_{sc}** 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_{sc}, Strategy Logic, ...**);
Conclusions and future works

Conclusions

- $\text{ATL}_{sc}$ 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 ($\text{ATL}_{sc}$, Strategy Logic, ...);

Future directions

- Defining and studying \textit{symmetric automata} for QCTL;
- Defining \textit{interesting fragments} of those logics;
- Considering \textit{partial observation};
- Considering \textit{randomised strategies}.
MOVEP 2016

- 12th Summer School MOVEP
- Genoa, Italy
- 27 June - 1 July
MOVEP 2016

- 12th Summer School MOVEP
- Genoa, Italy
- 27 June - 1 July

FORMATS 2016

- 14th Int. Conference FORMATS
- colocated with CONCUR and QEST
- Quebec City, Canada
- 24-26 August