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...

Toyota to recall Prius hybrids over ABS software

IDG News Service - Toyota plans to recall around 400,000 of its Prius hybrid cars to replace software that controls the anti-lock 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:

property:

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

model-checking algorithm

yes/no
Model checking and synthesis

system:

property:

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

Computation-Tree Logic (CTL)

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

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

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

- temporal modalities: $X \varphi$, $\varphi$ until $\psi$, $\varphi$ eventually $\psi$

$\text{true} \equiv \neg F \neg \varphi \equiv G \varphi$
Computation-Tree Logic (CTL)

- atomic propositions: $\circ, \odot, \ldots$
- boolean combinator $\neg \varphi, \varphi \lor \psi, \varphi \land \psi, \ldots$
- path quantifiers: $E\varphi, \varphi, A\varphi$
- temporal modalities: $X\varphi$, $\varphi \text{ until } \psi$, $\varphi \text{ eventually } \psi$, $\neg F \neg \varphi \equiv G \varphi$
Computation-Tree Logic (CTL)

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

  \[ \mathbf{E} \varphi \]

  \[ \mathbf{A} \varphi \]

- temporal modalities:

  \( \mathbf{X} \varphi \)  
  \( \varphi \mathbf{U} \psi \)

  “next \( \varphi \)”
  “\( \varphi \) until \( \psi \)”
Computation-Tree Logic (CTL)

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

\[ E \ F \] is reachable
Examples of CTL formulas

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

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

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

\[ E G (E F \bigcirc) \]

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.

\( E \, G (E, F \, p) \)  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.

\[ \text{E G(E F } \bigcirc \text{)} \]

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.

\[ \neg E(\neg \mathbb{P}) \ U \mathbb{P} \] in order to reach \( \mathbb{P} \), we have to visit \( \mathbb{P} \)
Examples of CTL formulas

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

\[ \neg E(\neg \Diamond) \mathbf{U} \Diamond \]

in order to reach \( \Diamond \), we have to visit \( \bullet \).
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.

\[ \text{E G F} \] there is a path visiting infinitely many times

\[ \begin{array}{c}
\text{E G F} \\
\text{there is a path visiting infinitely many times}
\end{array} \]
Examples of CTL formulas

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

\[ A(\Diamond G F \Rightarrow G F) \]
any path that visits \( \Diamond \) 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 $\text{ATL}_{sc}$
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.

\[ E F \land \forall p. [E F (p \land \varphi) \Rightarrow A G (\varphi \Rightarrow p)] \equiv \text{uniq}(); \text{true if we label the Kripke structure}; \text{false if we label the computation tree}; \]


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.

\[
\bullet \ EF(\bigcirc \land \forall p. [EF(p \land \bigcirc) \Rightarrow AG(\bigcirc \Rightarrow p)])
\]


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.

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

Quantified CTL

[Chu95,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.

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

$\sim$ true if we label the Kripke structure;

$\sim$ 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”:

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

- QCTL can express (least or greatest) fixpoints:

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

Theorem

For both semantics, QCTL, QCTL* and MSO are equally expressive.

QCTL with structure semantics

**Theorem**

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

**Proof**

**Membership:**
- guess a labelling,
- check the subformula.

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

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 any square grid 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.

[Image of tiles and grid]

QCTL with structure semantics

Theorem

$QCTL$ satisfiability for the structure semantics is undecidable.

Proof

Encode the problem of tiling finite square grids.

[Diagram of a grid with tiles representing the problem of tiling.]
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 any square grid 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.

[Diagram showing a grid with different colored tiles]
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 any square grid 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

\[ \text{Given a set of tiles, whether any square grid can be tiled is undecidable.} \]
QCTL with structure semantics

**Theorem**

QCTL satisfiability for the structure semantics is undecidable.

**Proof**

Reduction: is there a finite Kripke structure such that

Given a set of tiles, whether any square grid 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

- each state has one or two successors

[Figure: A finite Kripke structure with each state having one or two successors]
QCTL with structure semantics

Theorem

QCTL satisfiability for the structure semantics is undecidable.

Proof

Reduction: is there a finite Kripke structure such that two successors of the same state have a common successor.
QCTL with structure semantics

Theorem

QCTL satisfiability for the structure semantics is undecidable.

Proof

Reduction: is there a finite Kripke structure such that

![Diagram](image)

- [... many more conditions ...]
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 showing a Kripke structure with transitions labeled with '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.

Proof

Using alternating tree automata:

- polynomial-size automata for CTL;
- boolean combinators can be handled easily;
- quantification is handled by projection, which requires alternation removal (exponential blowup).

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$_{sc}$
Concurrent games

A concurrent game is made of

- a transition system;

![Diagram](image.png)
Concurrent games

A concurrent game is made of
- a transition system;
- a set of agents (or 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.
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 □.
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 ○.
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

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

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

\[ \langle \langle A \rangle \rangle \varphi \] 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 \(\varphi\).

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

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 \( \varphi \).

\[ \langle \langle \varnothing \rangle \rangle F \]
\[ \langle \langle \square \rangle \rangle F \]
\[ \langle \langle \varnothing \rangle \rangle G(\langle \langle \square \rangle \rangle F) \equiv \langle \langle \varnothing \rangle \rangle G p \]

\[ \langle \langle \rangle \rangle \]

---

Temporal logics for games: ATL

ATL extends CTL with strategy quantifiers

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

\[ \langle \bigcirc \rangle \; F \; p \]
\[ \langle \square \rangle \; F \; p \]
\[ \langle \bigcirc \rangle \; G \left( \langle \square \rangle \; F \; p \right) \equiv \langle \bigcirc \rangle \; G \; p \]

Theorem

Model checking ATL is \textit{PTIME}-complete.

consider the following strategy of Player: “always go to..."
Consider the following strategy of Player $\bigcirc$: “always go to $\square$.”

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

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

What $\mathbf{ATL}_{sc}$ can express

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

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

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

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

- **Existence of Nash equilibria**:

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

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

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

- **Existence of dominating strategy**:
  \[
  \langle A\cdot \rangle \ [B] (\neg \varphi \Rightarrow [A\cdot] \neg \varphi)
  \]
Translating ATL_{sc} into QCTL

- player A has moves \( m^A_1, ..., m^A_n \);
- from the transition table, we can compute the set Next(\( \bullet \), A, \( m^A_i \)) of states that can be reached from \( \bullet \) when player A plays \( m^A_i \).

Translating ATL$_{sc}$ into QCTL

- player $A$ has moves $m^A_1, \ldots, m^A_n$;
- from the transition table, we can compute the set $\text{Next}(\bullet, A, m^A_i)$ of states that can be reached from $\bullet$ when player $A$ plays $m^A_i$.

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

$$\exists m^A_1. \exists m^A_2 \ldots \exists m^A_n.$$

- this corresponds to a strategy: $A G (m^A_i \leftrightarrow \land \neg m^A_j)$;
- the outcomes all satisfy $\varphi$:

$$A \left[ G (q \land m^A_i \Rightarrow X \text{Next}(q, A, m^A_i)) \Rightarrow \varphi \right].$$

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

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

**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._
What about satisfiability?

<table>
<thead>
<tr>
<th>Theorem</th>
<th>(\text{QCTL satisfiability is decidable.})</th>
</tr>
</thead>
<tbody>
<tr>
<td>But</td>
<td></td>
</tr>
<tr>
<td>Theorem (TW12)</td>
<td>(\text{ATL}_{sc} \text{ satisfiability is undecidable.})</td>
</tr>
</tbody>
</table>

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, ATL\textsubscript{sc} satisfiability is decidable.

- player $\Box$ 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: $\mathbf{A} \mathbf{G}(\text{turn}_A \Rightarrow \mathbf{E} \mathbf{X}_1 \text{mov}_A);$  
- the outcomes all satisfy $\varphi$: $\mathbf{A}[\mathbf{G}(\text{turn}_A \land \mathbf{X} \text{mov}_A) \Rightarrow \varphi].$

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

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