Logics in Computer Science

Fabio Mogavero

Università degli Studi di Napoli Federico II

13th Italian Conference on Theoretical Computer Science
Varese, Italy, September 19-21, 2012
Idea behind the thesis

Main idea

We looked for logics extending some of the classic temporal logics used in computer science, with the aim to augment their expressive power without increasing the complexity of their decision problems.

Application

Some of the introduced logics can be effectively used as specification languages for the formal verification and synthesis of systems.
Areas of research

Topic areas

Our research fall in the two, so called, areas of:

- logics for computations;
- logics for strategies.

The thesis contains four different works, two for each area of research!
In this talk

Logics for computations

*Graded Computation Tree Logic*

Logics for strategies

*Reasoning About Strategies*
System verification, design, and synthesis

The framework

Let $S$ represent a system and $P$ a desired behavior.

There are three very important problems!

- **Verification**: Is $S$ correct w.r.t. $P$?
- **Design**: Is $P$ a correct specification?
- **Synthesis**: Build an $S$ satisfying $P$. 
Formal methods

Formalization
To answer to these questions, formal methods are used.
- \( S \) can be modeled by a labeled transition graph \( \mathcal{K} \).
- \( P \) can be expressed as a temporal logic formula \( \varphi \).

Decision procedures
- Verification -> Model Checking: Is \( \mathcal{K} \) a model of \( \varphi \) (\( \mathcal{K} \models \varphi \))? 
- Design/Synthesis -> Satisfiability: Is there a \( \mathcal{K} \) such that \( \mathcal{K} \models \varphi \)?
Verification

**Verification as debugging:** failure of verification identifies bugs.

- Both specifications and programs formalize informal requirements.
- Verification contrasts two independent formalizations.
- Failure of verification reveals inconsistency between formalizations.

**Model checking:** uncommonly effective debugging tool.

- Systematic exploration of the design state space.
- Good at catching difficult “limit cases”. 
Design and synthesis

Satisfiability: useful to verify...
- the realizability of a specification;
- the non-triviality of a specification.

Satisfiability witness: useful to synthesize the model of a correct system.
Part I

Logics for Computations
A *Kripke structure* is a labeled transition graph \( \mathcal{K} = \langle \mathcal{AP}, \mathcal{W}, R, L, w_0 \rangle \) used to model system behaviors in a monolithic way:

1. \( \mathcal{AP} \): *atomic propositions* represents *system properties*;
2. \( \mathcal{W} \): *worlds* represent *system states*;
3. \( R \subseteq \mathcal{W} \times \mathcal{W} \): *edges* represent *system transitions*;
4. \( L : \mathcal{W} \to 2^{\mathcal{AP}} \): *labels* represent *state properties*;
5. \( w_0 \in \mathcal{W} \): *designated initial world*. 
Classic temporal logics

Main families of temporal logics

- **Linear-Time Temporal Logics** (LTL)
  - Each moment in time has a unique possible future.
  - LTL expresses path properties based on path state labels.
  - Useful for hardware specification.

- **Branching-Time Temporal Logics** (CTL, CTL*, and μCALCULUS)
  - Each moment in time may split into various possible future.
  - CTL* expresses state properties based on the existence of paths satisfying LTL-like properties.
  - Useful for software specification.
Linear-time Temporal Logic [Pnueli, ’79]

**LTL syntax**

\[ \varphi ::= p \mid \neg \varphi \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid X \varphi \mid \varphi U \varphi \mid \varphi R \varphi. \]

**Informal semantics**

- **X \varphi**: \( \varphi \) holds in the next state;
- **\( \varphi_1 U \varphi_2 \)**: \( \varphi_1 \) holds until \( \varphi_2 \) holds;
- **\( \varphi_1 R \varphi_2 \)**: \( \varphi_2 \) holds forever or until \( \varphi_1 \) holds;
- **F \varphi**: \( \varphi \) holds eventually;
- **G \varphi**: \( \varphi \) holds forever.
Full Computation Tree Logic [Emerson & Halpern, ’86]

**CTL* syntax**

1. $\phi ::= p \mid \neg \phi \mid \phi \land \phi \mid \phi \lor \phi \mid E\psi \mid A\psi,$
2. $\psi ::= LTL(\phi).$

**Example**

- **EG EF $\psi$:** there is a computation from which, in each moment of its time, it starts another computation satisfying eventually $\psi$;
- **AFG $\psi$:** all computations eventually reach a point in their time from which $\psi$ holds forever.
## Computational complexities

<table>
<thead>
<tr>
<th>Logics</th>
<th>Model Checking</th>
<th>Satisfiability</th>
</tr>
</thead>
<tbody>
<tr>
<td>LTL</td>
<td>PSPACE-COMPLETE</td>
<td>PSPACE-COMPLETE</td>
</tr>
<tr>
<td>CTL</td>
<td>PTIME-COMPLETE</td>
<td>EXPSPACE-COMPLETE</td>
</tr>
<tr>
<td>CTL*</td>
<td>PSPACE-COMPLETE</td>
<td>2EXPTIME-COMPLETE</td>
</tr>
<tr>
<td>μCALCULUS</td>
<td>UPTIME ∩ COUPTIME</td>
<td>EXPSPACE-COMPLETE</td>
</tr>
</tbody>
</table>

**Table**: Computational complexity of model-checking and satisfiability problems.
Graded Computation Tree Logic
Facts

The ultimate temporal logic

The $\mu$CALCULUS subsumes many logics, in particular, LTL, CTL, and CTL$^*$. Several extension of $\mu$CALCULUS have been considered. One among all: $G\mu$CALCULUS, i.e., the $\mu$CALCULUS extended with graded modalities (“there are at least $n$ successors such that...”).

Expressiveness vs simplicity

$\mu$CALCULUS: very expressive but too low-level (hard to understand and use). LTL, CTL, and CTL$^*$: less expressive but much more human-friendly.
**Motivations**

**A natural question**

How could logics that allow to reason about paths be affected by considering graded modalities?

**Why graded modalities on paths?**

- XML query languages.
- Cyclomatic complexity.
- Redundancy in a system.
- Counting error counterexamples.
Our contribution

We investigate an extension of $\text{CTL}$ with graded modalities ($\text{GCTL}$, for short).

There is a technical challenge involved with such an extension:

- the counting concept have to relapse both on states and paths;
- it is easy to have structures with an infinite number of paths satisfying a given property, so the counting concept becomes easily useless.

To solve the problem we use the concepts of minimality and conservativeness.
**Syntax of GCTL***

GCTL* is a syntactic variant of CTL* in which classic path quantifiers are replaced by their **graded** version, the existential \( E^{\geq g} \) and the universal \( A^{< g} \).

**Definition**

GCTL* **state** (\( \varphi \)) and **path** (\( \psi \)) **formulas** are built inductively in the following way, where \( p \) is an atomic proposition:

1. \( \varphi ::= p \mid \neg \varphi \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid E^{\geq g} \psi \mid A^{< g} \psi \),
2. \( \psi ::= LTL(\varphi) \).

The simpler class of GCTL formulas is obtained by forcing each temporal operator, occurring in a formula, to be coupled with a graded path quantifier.
Fundamental notions

Domain of quantification

The domain on which quantifiers range is not simply the set of paths, but that of equivalence classes on paths w.r.t. an appropriate equivalence relation.

An useful equivalence relation

Two paths are prefix equivalent iff

1. their common prefix satisfy the formula (minimality);
2. no matter how the prefix is extended in the structure, the resulting path satisfies the formula (conservativeness).
Semantics of \text{GCTL}^*$

**Graded quantifications**

- \text{E} \geq g \psi: "there exist at least } g \text{ equivalence classes of paths satisfying } \psi".
- \text{A} < g \psi: "all but less than } g \text{ equivalence classes of paths satisfy } \psi".

**CTL$^*$ sublogic**

- Boolean and temporal operators have classic semantics.
Model-theoretic properties

Positive model-theoretic properties

- **Invariance** under unwinding;
- **Tree** and **finite** model property.

Negative model-theoretic property

**Non-invariance** under bisimulation.
Expressiveness

Theorem

- $\text{GCTL}^*$ is strictly more expressive than $\text{CTL}^*$.
- $\text{GCTL}$ is strictly more expressive than $\text{CTL}$. 
Succinctness

A simple property

In a tree, there are just \( g \) grandchildren of the root labeled with \( p \), while all other nodes are not.

GCTL formalization

It is possible to express such a property with the following GCTL formula of size logarithmic in \( g \):

\[
\phi = (E^{=g} F p) \land (\neg p \land AX (\neg p \land AX (p \land AX AG \neg p))).
\]

Observation

Each G\(\mu\)CALCULUS formulas equivalent to \( \phi \) has to have size at least polynomial in \( g \).
A natural question

A question
Do the additional expressiveness and succinctness of GCTL imply an increasing of the computational-complexity cost of deciding its satisfiability problem?

The answer
No! It remains EXP\text{TIME}-\text{COMPLETE}.
Satisfiability problem

Satisfiability via tree automata

General procedure: \( \exists T . T \models \varphi \iff L(\mathcal{A}_\varphi) \neq \emptyset \)

1. Given a specification \( \varphi \), construct a tree automaton \( \mathcal{A}_\varphi \) recognizing all the tree models of \( \varphi \) itself.

2. Check for the non-emptiness of \( \mathcal{A}_\varphi \).

Challenge: The automaton \( \mathcal{A}_\varphi \) has to deal with the degree of \( \varphi \).

Solution: Every original tree model is encoded into a binary tree having an extra labeling used to manage the splitting of the degrees among paths.
Part II

Logics for Strategies
From monolithic to multi-agent systems

<table>
<thead>
<tr>
<th>Historical development</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Model checking</strong>: analyzes systems monolithically (system components plus environment) [Clarke &amp; Emerson, Queille &amp; Sifakis, '81].</td>
</tr>
<tr>
<td><strong>Module checking</strong>: separates out the environment from the system components, but still views the system monolithically; two-player game between system and environment [Kupferman &amp; Vardi, '96].</td>
</tr>
<tr>
<td><strong>Alternating temporal reasoning</strong>: multi-agent systems (components individually considered), playing strategically [Alur et al., '02].</td>
</tr>
<tr>
<td><strong>Strategic logic reasoning</strong>: two-player turn-based games verified by considering strategies as first order objects [Chatterjee et al., '07].</td>
</tr>
</tbody>
</table>
Strategic reasoning

Example (Reactive synthesis)

Synthesize an interactive system that satisfies a given specification, independently of the possible sequences of inputs.

Example (Nash equilibrium)

Verify that all players of a game have optimal strategies (each player has a strategy such that it is rational for him to adhere to it assuming that all the other players also do so).
A *concurrent game structure* is a tuple $G = \langle \text{AP}, \text{Ag}, \text{Ac}, \text{St}, \lambda, \tau, s_0 \rangle$ where:

1. $\text{AP}$: finite set of *atomic propositions*;
2. $\text{Ag}$: finite set of *agents*;
3. $\text{Ac}$: set of *actions*;
4. $\text{St}$: set of *states*;
5. $s_0 \in \text{St}$: *designated initial state*;
6. $\lambda : \text{St} \to 2^{\text{AP}}$: *labeling function*;
7. $\tau : \text{St} \times \text{Ac}^{\text{Ag}} \to \text{St}$: *transition function* mapping a state and a *decision* (i.e., a function from $\text{Ag}$ to $\text{Ac}$) to a new state.
**Concurrent game structures (II)**

$S_t$ is the arena of the game in which the agents operate.

$A_c$ consists of local actions of all the agents.

$A_c^{Ag}$ represents the set of choices of an action for each agent.
Alternating-time Temporal Logic [Alur et al., ’02]

$\langle\langle A \rangle\rangle \psi$: There is a strategy for the agents in $A$ enforcing the property $\psi$, independently of what the agents not in $A$ can do.

Example

$\langle\langle \{\alpha, \beta\} \rangle\rangle G \neg fail$: “Agents $\alpha$ and $\beta$ cooperate to ensure that a system (having possibly more than two processes (agents)) never enters a fail state”.

- Strategies are treated only implicitly.
- Quantifier alternation fixed to 1.
Strategy Logic [Chatterjee et al., ’07]

Example

$\exists x_1, x_2. \forall y. (\psi(x_1, y) \lor \neg\psi(x_2, y))$: “There are two strategies for the first player, $x_1$ and $x_2$ such that, independently of the strategy $y$ of the second player, it holds that $x_1$ enforces $\psi$ or $x_2$ enforces $\neg\psi$.”

- Restricted framework of two-players turn-based games.
- Non-elementary model checking without matching lower-bound.
- Open satisfiability problem.
Reasoning About Strategies
Facts

Logics for strategies

In the literature there are no logics for the concurrent multi-player framework in which strategies are treated as explicit first-order objects.
Motivations

Extending the framework

We are looking for a logic in which we can talk about the strategic behavior of agents in generic multi-player concurrent games.

Application

It can be used as a specification language for the formal verification and synthesis of modular and interactive systems.
We introduce a new *Strategy Logic* (SL), as a more general framework (both in its syntax and semantics), for explicit reasoning about strategies in multi-player concurrent games.

We also study a chain of more tractable syntactic fragments which result to be strictly more expressive than ATL*.
Syntax of $\text{SL}$

$\text{SL}$ syntactically extends $\text{LTL}$ by means of strategy quantifiers, the existential $\langle\langle x \rangle\rangle$ and the universal $\llbracket [x] \rrbracket$, and agent binding $(a, x)$.

**Definition**

$\text{SL}$ formulas are built inductively in the following way, where $x$ is a variable and $a$ an agent.

$$ \varphi ::= \text{LTL}(\varphi) \mid \langle\langle x \rangle\rangle \varphi \mid \llbracket [x] \rrbracket \varphi \mid (a, x) \varphi. $$
Fundamental notions

A *strategy* is a function mapping each *history* of the game to an *action*.

When all the agents have associated strategies, they identify a unique path named *play* of the game.
Semantics of $\mathcal{SL}$

Strategy quantifications

- $\langle x \rangle \varphi$: "there exists a strategy $x$ for which $\varphi$ is true".
- $[x] \varphi$: "for all strategies $x$, it holds that $\varphi$ is true".

Agent binding

- $(a, x) \varphi$: "$\varphi$ holds, when the agent $a$ uses the strategy $x$".

$LTL$ sublogic

- Boolean and temporal operators have classic semantics.
Failure is not an option

Example (No failure property)

“In a system $S$ built on three processes, $\alpha$, $\beta$, and $\gamma$, the first two have to cooperate in order to ensure that $S$ never enters a failure state”.

Three different formalization in SL.

1. $\langle\langle x \rangle\rangle\langle\langle y \rangle\rangle[\langle z \rangle(\langle \alpha, x \rangle(\langle \beta, y \rangle(\langle \gamma, z \rangle(G \neg\text{fail}))))]:$ $\alpha$ and $\beta$ have two strategies, $x$ and $y$, respectively, that, independently of what $\gamma$ decides, ensure that a failure state is never reached.

2. $\langle\langle x \rangle\rangle[\langle z \rangle(\langle y \rangle(\langle \alpha, x \rangle(\langle \beta, y \rangle(\langle \gamma, z \rangle(G \neg\text{fail}))))):$ $\beta$ can choose his strategy $y$ dependently of that one chosen by $\gamma$.

3. $\langle\langle x \rangle\rangle[\langle z \rangle(\langle \alpha, x \rangle(\langle \beta, x \rangle(\langle \gamma, z \rangle(G \neg\text{fail})))):]$ $\alpha$ and $\beta$ have a common strategy $x$ to ensure the required property.
Equilibria

Example (Nash equilibrium)
If each player has chosen a strategy and no player can benefit by changing his strategy while the other players keep theirs unchanged, then the current strategy profile constitute a Nash equilibrium [Nash, ’50].

Example (\(k\)-resilient equilibrium)
Up to \(k\) players can deviate without improving their payoff [Aumann, ’59].

Example (\(k\)-immune equilibrium)
The payoff of a non-deviating player does not degrade if up to \(k\) players deviate [Abraham et al., ’06].
ATL* model-theoretic properties

Positive model-theoretic properties

- Invariance under bisimulation.
- Invariance under decision-unwinding.
- Bounded decision-tree model property.
SL model-theoretic properties

Negative model-theoretic properties

- Non-invariance under bisimulation.
- Non-invariance under decision-unwinding.
- Unbounded model property.

Positive model-theoretic properties

- Invariance under local isomorphism.
- Invariance under state-unwinding.
- State-tree model property.
Expressiveness

Theorem

$\mathcal{L}_c$ is strictly more expressive than $\text{ATL}^*$.  

Explanation

- Unbounded quantifier alternation.
- More than one temporal goal at a time.
- Agents can be forced to share the same strategy.
Model checking (I)

Upper bound

- We reduce the solution of the model-checking problem to the checking of the non-emptiness of a suitable tree-automaton.

- The automaton construction is an evolution and merging of the translations from QPTL into nondeterministic Büchi automata [Sistla et al., ’87] and from LTL into alternating Büchi automata [Vardi, ’94].

Lower bound

- We reduce the satisfiability problem of QPTL to the model-checking problem of SL on a fixed one-agent game.

- The reduction is inspired by the hardness proof of ATL* with strategy contexts [Costa et al., ’10].
**Theorem**

$\text{SL model checking problem has a } \text{NONELEMENTARYTIME-COMPLETE}$

$\text{formula complexity and } \text{PTIME data complexity}.$
Satisfiability

Lower bound

- We reduce the recurrent domino problem to the satisfiability of \( \text{SL} \).
- The reduction is strongly-based on the fact that \( \text{SL} \) does not have the bounded model property.

Theorem

\( \text{SL} \) has a \textit{highly undecidable} (\( \Sigma^1_1 \)-HARD) satisfiability problem.
## An overview

<table>
<thead>
<tr>
<th></th>
<th>Model checking</th>
<th>Satisfiability</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SL</strong></td>
<td>NONELEMENTARY-COMPLETE</td>
<td>$\Sigma_1^1$-HARD</td>
</tr>
<tr>
<td><strong>SL[NG]</strong></td>
<td>NONELEMENTARY-COMPLETE</td>
<td>$\Sigma_1^1$-HARD</td>
</tr>
<tr>
<td><strong>SL[BG]</strong></td>
<td>?</td>
<td>$\Sigma_1^1$-HARD</td>
</tr>
<tr>
<td><strong>SL[CG]</strong></td>
<td>2ExpTime-complete</td>
<td>?</td>
</tr>
<tr>
<td><strong>SL[1G]</strong></td>
<td>2ExpTime-complete</td>
<td>2ExpTime-complete</td>
</tr>
<tr>
<td><strong>ATL</strong></td>
<td>2ExpTime-complete</td>
<td>2ExpTime-complete</td>
</tr>
</tbody>
</table>
Thank you very much for your kind attention! I hope my talk was enough logic!