

# Graded Computation Tree Logic

Alessandro Bianco   Fabio Mogavero   Aniello Murano

Università degli Studi di Napoli "Federico II"

<http://people.na.infn.it/~{alessandrobiano,mogavero,murano}>

3rd Workshop of the ESF Networking Programme on  
Games for Design and Verification  
Oxford, England, UK, September 20-23, 2010

# System correctness

Let **S** be a system and **P** a desired behavior (specification).

Two very important problems:

- **Model Checking**: Is **S** correct w.r.t. **P**?
- **Satisfiability**: Is **P** a correct specification?

To answer to these questions, formal methods are used.

- **S** can be modeled by a **labeled transition graph**  $\mathcal{K}$  (Kripke structure).
- **P** can be expressed as a **temporal logic formula**  $\varphi$ .

Then,

- **Model Checking**:  $\mathcal{K} \models \varphi$ ?
- **Satisfiability**: **Is there a  $\mathcal{K}$  such that  $\mathcal{K} \models \varphi$ ?**

# System models

A Kripke structure is a transition graph used to model system behaviours:

- 1 nodes represent system states;
- 2 edges represent system transitions;
- 3 labels represent state properties;
- 4 a path represents a system run.

# System specifications

Temporal logic: description of the temporal ordering of events!

Two 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 the paths state labels.
- Useful for hardware specification.

- **Branching-Time Temporal Logics (CTL, CTL\*, and  $\mu$ -CALCULUS)**

- Each moment in time may split into various possible future.
- CTL\* expresses state properties based on the existence or universality of paths exiting from that state and satisfying LTL-like properties.
- Useful for software specification.

# Graded system specifications

**$G\mu$ -CALCULUS** extends the  $\mu$ -CALCULUS with **graded modalities**:

- "there exists at least  $n$  successors satisfying a given property";
- "all but at most  $n$  successors satisfy a given property".

$\mu$ -CALCULUS: very expressive but too low-level (hard to understand).

LTL, CTL, and CTL\*: less expressive but much more human-friendly.

# Motivations

We ask whether we can extend CTL\* with graded modalities so that:

- there is no extra cost in determining its decision problem;
- the resulting formal language is easy to use and understand.

Why graded modalities on paths?

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

# Outline

1 Graded Computation Tree Logic

2 Satisfiability Resolution

3 Conclusion

# Syntax of GCTL\* and GCTL

**GCTL\*** extends CTL\* with new **graded path quantifiers**:

- "there exists at least  $n$  paths satisfying a given property";
- "all but at most  $n$  paths satisfy a given property".

## Definition

GCTL\* *state* ( $\phi$ ) and *path* ( $\psi$ ) *formulas* are built inductively as follows:

- 1  $\phi ::= p \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid E^{\geq g}\psi \mid A^{<g}\psi,$
- 2  $\psi ::= \phi \mid \neg\psi \mid \psi \wedge \psi \mid \psi \vee \psi \mid X\psi \mid \tilde{X}\psi \mid \psi U\psi \mid \psi R\psi.$

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

# Counting paths

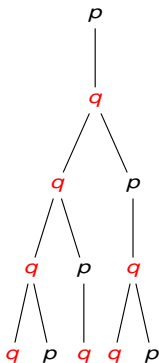
What does counting paths mean?

We can observe the paths starting from a node in the tree unwinding of a Kripke structure.

A property ensured by a common prefix may be satisfied on an infinite number of paths.

We can consider paths with a common prefix satisfying the formula as equivalent.

Example:  $Fq!$



# Equivalence relation

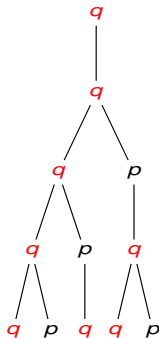
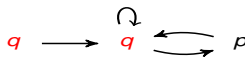
We can define a generic equivalence on paths.

Two paths are equivalent if their common prefix satisfy the formula.

It may happen that the prefix satisfy a formula but a whole path may not.

We need to ask that no matter how the prefix is extended in the structure the path satisfy the formula.

Example:  $Gq!$



# Semantics of GCTL\*

## Definition

Given a Ks  $\mathcal{K} = \langle AP, W, R, L, w_0 \rangle$ , a world  $w \in W$ , and a GCTL\* path formula  $\psi$ , it holds that:

- 1  $\mathcal{K}, w \models E^{\geq g} \psi$  iff  $|\text{Pth}(\mathcal{K}, w, \psi) / \equiv_{\mathcal{K}}^{\psi}| \geq g$ ;
- 2  $\mathcal{K}, w \models A^{<g} \psi$  iff  $|\text{Pth}(\mathcal{K}, w, \neg \psi) / \equiv_{\mathcal{K}}^{\neg \psi}| < g$ ;

where  $\text{Pth}(\mathcal{K}, w, \psi)$  is the set of paths of  $\mathcal{K}$  starting in  $w$  and satisfying  $\psi$ .

For  $g = 1$ , we may write  $E\psi$  and  $A\psi$  instead of  $E^{\geq 1}\psi$  and  $A^{<1}\psi$ .

# Outline

1 Graded Computation Tree Logic

2 Satisfiability Resolution

3 Conclusion

# Previous approach

The previous known algorithm for the satisfiability of GCTL (LICS 2009) is EXPTIME in the size of the formula, when the degree is coded in unary.

Procedure sketch:

- 1 GCTL formula  $\varphi$  ->
- 2 Partitioning Alternating Büchi Tree Automata  $\mathcal{A}_\varphi$  ->
- 3 Nondeterministic Büchi Tree Automata  $\mathcal{N}_\varphi$  ->
- 4 Resolution of the emptiness for  $\mathcal{N}_\varphi$ .

# Problems and solutions

Problems for the binary case:

- 1  $\mathcal{A}_\varphi$  is polynomial both in the length and degree of  $\varphi$  so,  $\mathcal{N}_\varphi$  is exponential in these values.
- 2 The emptiness for  $\mathcal{N}_\varphi$  is exponential in the width of the input trees that is polynomial in the degree of  $\varphi$ .

Solution:

- 1 we non-determinize only part of  $\mathcal{A}_\varphi$  that is polynomial in the length only;
- 2 we use a new encoding for the input trees of  $\mathcal{A}_\varphi$ .

The new algorithm is EXPTIME in the size of the formula when the degree is coded in binary.

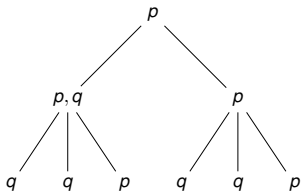
# Binary encoding

We encode the possible models of a GCTL formula in a graded binary tree.

- 1 We transform a Kripke structure in its equivalent tree model.
- 2 We add to the tree nodes degrees representing how many paths satisfy or do not satisfy a given path formula.
- 3 We delay the generation of the successor nodes so that at each step the degree of a node is split only in two parts.
- 4 We add to the tree nodes the informations on how the degrees are split.

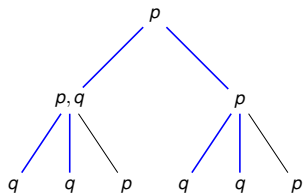
# Graded tree

$$E^{\geq 3} p U q$$



# Graded tree

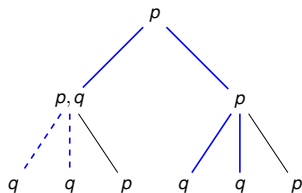
$$E^{\geq 3} p U q$$



- 1 The example tree model contains four paths.

# Graded tree

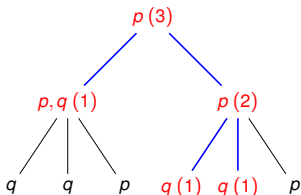
$$E^{\geq 3} p U q$$



- 1 The example tree model contains four paths.
- 2 However, there are only three equivalence classes.

# Graded tree

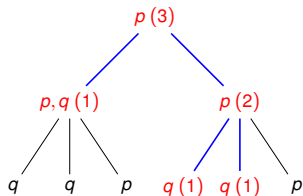
$$E^{\geq 3} p U q$$



- 1 The example tree model contains four paths.
- 2 However, there are only three equivalence classes.
- 3 For each node we add the number of classes passing through it.

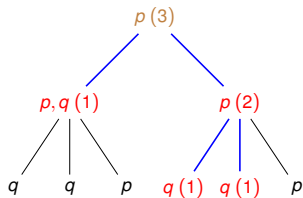
# Delayed generation tree

$$E^{\geq 3} p U q$$

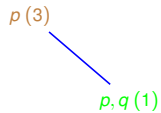
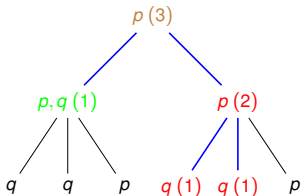


# Delayed generation tree

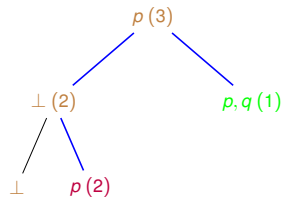
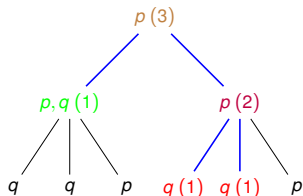
$$E^{\geq 3} p U q$$

$$p(3)$$


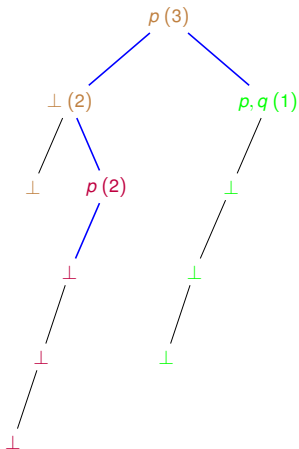
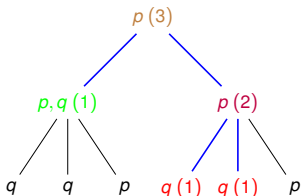
# Delayed generation tree

$$E^{\geq 3} p U q$$


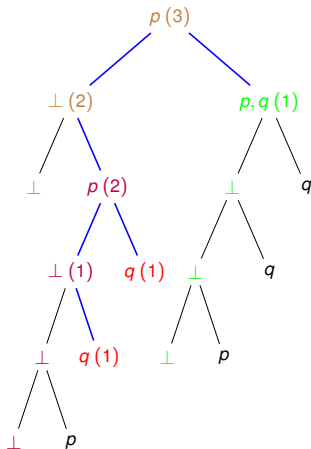
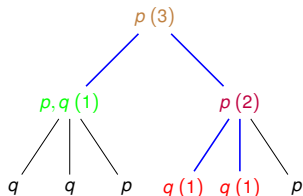
# Delayed generation tree

$$E^{\geq 3} p U q$$


# Delayed generation tree

$$E^{\geq 3} p U q$$


# Delayed generation tree

$$E^{\geq 3} p U q$$


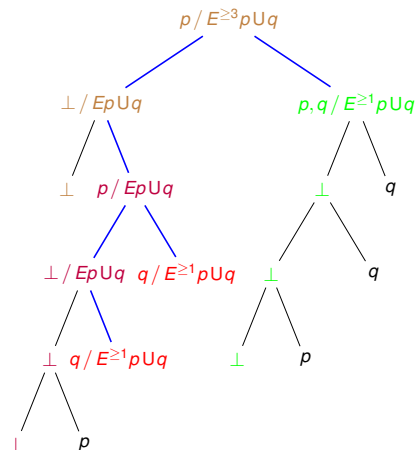
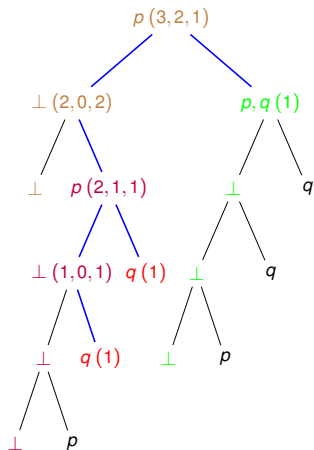


# Automata for satisfiability

The automaton that accepts all the binary models of a formula consists in two parts:

- A coherence satellite checks that the degrees are coherently handed down from a node to its successors.
- An alternating formula automaton checks that the path formulas are satisfied on the paths with non-zero labeling.
- The degree labels are not in the input tree but they are supplied by the coherence satellite (the formula automaton just needs to read the satellite states).
- The automaton with satellite accepts in input non-graded binary trees.

# Formula automaton run



# Satisfiability complexity

- The coherence satellite is a non-deterministic automaton polynomial in the size of the formula but exponential in the degree coded in binary.
- The formula automaton is an alternating automaton polynomial in the length of the formula only.
- We need to non-determinize only the formula automaton through the Miyano-Hayashi reduction.
- The resulting non-deterministic automaton is exponential in the size of the formula, i.e., both in the length and in the degree coded in binary.

# Computational complexity

	Sat.
CTL	EXPTIME-COMPLETE
GCTL	EXPTIME-COMPLETE
$\mu$ -CALCULUS	EXPTIME-COMPLETE
$G\mu$ -CALCULUS <sup>ab</sup>	EXPTIME-COMPLETE

**Table:** Computational complexity of Satisfiability.

<sup>a</sup> O. Kupferman, U. Sattler, and M. Vardi. The Complexity of the  $G\mu$ -CALCULUS, CADE'02.

<sup>b</sup> P. Bonatti, C. Lutz, A. Murano, and M. Vardi. The Complexity of Enriched  $\mu$ -Calculi, ICALP'06 / LMCS'08.

$G\mu$ -CALCULUS subsumes GCTL.

However, GCTL is exponential more succinct than  $G\mu$ -CALCULUS.

# Outline

1 Graded Computation Tree Logic

2 Satisfiability Resolution

3 Conclusion

# Conclusion

In this work...

- we showed how it is possible to count the number of paths satisfying a given formula;
- we introduced a GCTL logic that allow us to use graded quantifiers as properties on states;
- we describe an encoding technique useful to represents and check the distributions of paths with desired properties in a model;
- we solved the GCTL satisfiability problem in time exponential in the size of the formula when the degree is coded in binary.

Thank you very much for your attention!