

Branching-Time Temporal Logics with Minimal Model Quantifiers

Fabio Mogavero Aniello Murano

Università degli Studi di Napoli "Federico II"

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

13th International Conference on Developments in Language Theory
Stuttgart, Germany, June 30 - July 3, 2009

Systems 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 modelled by a **labeled transition graph** \mathcal{K} (Kripke structure).
- **P** can be expressed as a **temporal logic formula** φ .

Then,

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

Systems 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.
 - Useful for hardware specification.
- **Branching-Time Temporal Logics (PML, CTL, CTL+, and CTL*)**
 - Each moment in time may split into various possible future.
 - Useful for software specification.

CTL*: The classical temporal logic on *computation trees*.

CTL+: CTL* without *nesting* of temporal operators.

CTL: CTL+ without *Boolean combination* of temporal operators.

PML: CTL with *next-time* temporal operators only.

Computational complexity

	M.C. (Formula)	M.C. (Program)	Sat.
LTL	PSPACE-COMPLETE	NLOGSPACE-COMPLETE	PSPACE-COMPLETE
PML	PTIME	NLOGSPACE	PSPACE-COMPLETE
CTL	PTIME-COMPLETE	NLOGSPACE-COMPLETE	EXPTIME-COMPLETE
CTL+	Δ_2^P -COMPLETE	NLOGSPACE-COMPLETE	2EXPTIME-COMPLETE
CTL*	PSPACE-COMPLETE	NLOGSPACE-COMPLETE	2EXPTIME-COMPLETE

Table: Computational complexity of Model Checking and Satisfiability.

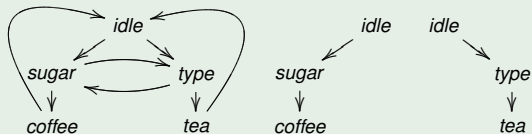
Our motivation

Two very challenging issues with temporal logic:

- To introduce techniques that automatically allow to select small critical parts of the system to be successively verified.
- To extend the expressiveness of classical temporal logics to model more complex specifications.

Example (The Coffee-Tea Machine)

Minimal subsystem: $EF(\text{coffee} \vee \text{tea})$. Property to verify: $EF(\text{sugar})$.



Our proposal

We introduce an extension of CTL* with **Minimal Model Quantifiers**.

We use formulas to both select and verify the system part of interest.

The detection of the subsystem is through a semantics characterization.

We call this idea the **Extract-Verify Paradigm**.

Outline

1 Minimal Model Quantifiers in CTL*

- Syntax and Semantics
- Properties

2 Main results

- Model Checking
- Satisfiability

3 Open problems

4 Conclusion

Syntax of MCTL* MCTL+, MCTL, and MPML

Definition

MCTL* *state* (φ) and *path* (ψ) *formulas* are built inductively as follows:

- 1 $\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \Xi \varphi \mid \varphi \Lambda \varphi \mid E\psi \mid A\psi,$
- 2 $\psi ::= \varphi \mid \neg\psi \mid \psi \wedge \psi \mid \psi \vee \psi \mid X\psi \mid \tilde{X}\psi \mid \psi U \psi \mid \psi R \psi.$

MCTL* extends CTL* by adding the quantifiers Ξ and Λ .

MCTL+: MCTL* without nesting of temporal operators [No: $pU(Xq)$].

MCTL: MCTL+ without comb. of temporal operators [No: $(pUq) \wedge (rRs)$].

MPML: MCTL with next-time temporal operators only [No: pUq].

Informal meaning of Ξ and Λ

Definition

MCTL* *state* (φ) and *path* (ψ) *formulas* are built inductively as follows:

- 1 $\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \Xi \varphi \mid \varphi \Lambda \varphi \mid E\psi \mid A\psi,$
- 2 $\psi ::= \varphi \mid \neg\psi \mid \psi \wedge \psi \mid \psi \vee \psi \mid X\psi \mid \tilde{X}\psi \mid \psi U \psi \mid \psi R \psi.$

Informally, Ξ and Λ can be read as

- $\varphi_1 \Xi \varphi_2$: there is a submodel of φ_2 that satisfies φ_1 ,
- $\varphi_1 \Lambda \varphi_2$: all submodels of φ_2 satisfy φ_1 .
- φ_1 is the *submodel verifier*.
- φ_2 is the *submodel extractor*.

Kripke structures, partial order, and minimality

Definition

A *Kripke structure* (KRIPKE, for short) is a tuple $\mathcal{K} = \langle AP, W, R, L \rangle$ where:

- 1 AP: finite non-empty set of *atomic propositions*;
- 2 W: non-empty set of *worlds*;
- 3 $R \subseteq W \times W$: *transition relation*;
- 4 $L : W \mapsto 2^{AP}$: *labeling function*.

A KRIPKE \mathcal{K}' is a *substructure* of \mathcal{K} , formally $\mathcal{K}' \preceq \mathcal{K}$, iff the related labeled graphs are one a subgraph of the other.

For a set of KRIPKES S , we say that \mathcal{K} is *minimal in S* iff, for all $\mathcal{K}' \in S$, it holds that (i) $\mathcal{K} \preceq \mathcal{K}'$ or (ii) $\mathcal{K}' \not\preceq \mathcal{K}$.

By $\min(S)$ we denote the *antichain* (i.e., the set of minimal structures) of S .

Semantics of MCTL*

Definition

Given a KRIPKE $\mathcal{K} = \langle AP, W, R, L \rangle$, a world $w \in W$, and two MCTL* state formulas φ_1 and φ_2 , it holds that:

- 1 $\mathcal{K}, w \models \varphi_1 \boxplus \varphi_2$ iff there is $\mathcal{K}' \in \min(\mathcal{S}(\mathcal{K}, w, \varphi_2))$ such that $\mathcal{K}', w \models \varphi_1$;
- 2 $\mathcal{K}, w \models \varphi_1 \wedge \varphi_2$ iff for all $\mathcal{K}' \in \min(\mathcal{S}(\mathcal{K}, w, \varphi_2))$ it holds that $\mathcal{K}', w \models \varphi_1$;

where $\mathcal{S}(\mathcal{K}, w, \varphi)$ is the set of $\mathcal{K}' \preceq \mathcal{K}$ rooted in w that are *conservative* w.r.t. φ (i.e., all KRIPKES between \mathcal{K}' and \mathcal{K} behave as \mathcal{K}').

Duality: $\neg(\varphi_1 \boxplus \varphi_2) \equiv (\neg\varphi_1) \wedge \varphi_2$.

Reflexivity of a model

Example (Reflexivity world)

Consider the formula $\varphi = (EX EX p) \exists (EX t)$, where t means true.

φ is Sat, then suppose that $\mathcal{K}, w \models \varphi$.

- 1 The submodel extractor $EX t$ requires that w has an outgoing edge.
- 2 The submodel verifier $EX EX p$ requires that in a minimal and conservative submodel \mathcal{K}' of \mathcal{K} there is a path of length 2 leading to a node in which p holds.

Since φ is built using the existential minimal model quantifiers \exists , we have that \mathcal{K} is necessarily formed by a root world with a self loop.



the world w is labeled with p

Elementary model properties

Hence, we note that

- MPML is not invariant under **unwinding** and **partial unwinding**,
- it does not have the **tree model property**.

Then,

- it is not invariant under **bisimulation**,
- it is **more expressive** than PML.

All the above results also hold for MCTL, MCTL+, and MCTL*, since they subsume MPML.

Succinctness (I)

CTL+ is equivalent to CTL, but exponentially more succinct.

+

MCTL+ is equivalent to MCTL and the translation is polynomial.

=

MCTL is **exponentially more succinct** than CTL.

Succinctness (II)

Consider the CTL+ formula $\varphi = E(F p_1 \wedge F p_2 \wedge F p_3)$.

We translate φ in MCTL with only a polynomial blow-up.

Suppose that there is a path such that $w_0 \rightsquigarrow p_1 \rightsquigarrow p_2 \rightsquigarrow p_3$.

The idea of the translation is to

- **extract** a submodel where each path reaching p_1 or p_2 also reaches p_3 ,
- **verify** that, in such a submodel, there exists a path between p_1 and p_2 .

Outline

1 Minimal Model Quantifiers in CTL*

- Syntax and Semantics
- Properties

2 Main results

- Model Checking
- Satisfiability

3 Open problems

4 Conclusion

Model Checking result

Model Checking for all the introduced logics is decidable.

Idea: use of **oracle machines**.

Bad news:

- M.C. for MPML is Δ_2^P (PML has a PTIME M.C.).
- M.C. for MCTL is Δ_2^P -COMPLETE (CTL has a PTIME-COMPLETE M.C.).

Good news:

- M.C. for MCTL+ is Δ_2^P -COMPLETE (same complexity for CTL+).
- M.C. for MCTL* is PSPACE-COMPLETE (same complexity for CTL*).

However, the program complexity is PSPACE!

Model Checking result

Model Checking for all the introduced logics is decidable.

Idea: use of **oracle machines**.

Bad news:

- M.C. for MPML is Δ_2^P (PML has a PTIME M.C.).
- M.C. for MCTL is Δ_2^P -COMPLETE (CTL has a PTIME-COMPLETE M.C.).

Good news:

- M.C. for MCTL+ is Δ_2^P -COMPLETE (same complexity for CTL+).
- M.C. for MCTL* is PSPACE-COMPLETE (same complexity for CTL*).

However, the program complexity is PSPACE!

Model Checking result

Model Checking for all the introduced logics is decidable.

Idea: use of [oracle machines](#).

Bad news:

- M.C. for MPML is Δ_2^P (PML has a PTIME M.C.).
- M.C. for MCTL is Δ_2^P -COMPLETE (CTL has a PTIME-COMPLETE M.C.).

Good news:

- M.C. for MCTL+ is Δ_2^P -COMPLETE (same complexity for CTL+).
- M.C. for MCTL* is PSPACE-COMPLETE (same complexity for CTL*).

However, the program complexity is PSPACE!

Proof sketch

Basic step of the procedure: $\mathcal{K}, w \models \varphi_1 \boxplus \varphi_2$.

Construction of a **polynomial certificate** \mathcal{K}' of the test $\mathcal{K}, w \models \varphi_1 \boxplus \varphi_2$ that is possible to verify in

- **PTIME** for MCTL,
- **PSPACE** for MCTL*.

Actually, \mathcal{K}' is a minimal substructure of \mathcal{K} w.r.t. φ_2 such that $\mathcal{K}', w \models \varphi_1$.

In particular, we have to verify that \mathcal{K}' is

- **minimal**,
- **conservative**.

Finally, we build a bottom-up algorithm that uses the previous idea as an atomic computation step, through a query to a PTIME or PSPACE oracle.

Proof sketch

Basic step of the procedure: $\mathcal{K}, w \models \varphi_1 \boxplus \varphi_2$.

Construction of a **polynomial certificate** \mathcal{K}' of the test $\mathcal{K}, w \models \varphi_1 \boxplus \varphi_2$ that is possible to verify in

- **PTIME** for MCTL,
- **PSPACE** for MCTL*.

Actually, \mathcal{K}' is a minimal substructure of \mathcal{K} w.r.t. φ_2 such that $\mathcal{K}', w \models \varphi_1$.

In particular, we have to verify that \mathcal{K}' is

- **minimal**,
- **conservative**.

Finally, we build a bottom-up algorithm that uses the previous idea as an atomic computation step, through a query to a PTIME or PSPACE oracle.

Proof sketch

Basic step of the procedure: $\mathcal{K}, w \models \varphi_1 \boxplus \varphi_2$.

Construction of a **polynomial certificate** \mathcal{K}' of the test $\mathcal{K}, w \models \varphi_1 \boxplus \varphi_2$ that is possible to verify in

- **PTIME** for MCTL,
- **PSPACE** for MCTL*.

Actually, \mathcal{K}' is a minimal substructure of \mathcal{K} w.r.t. φ_2 such that $\mathcal{K}', w \models \varphi_1$.

In particular, we have to verify that \mathcal{K}' is

- **minimal**,
- **conservative**.

Finally, we build a bottom-up algorithm that uses the previous idea as an atomic computation step, through a query to a PTIME or PSPACE oracle.

Satisfiability result

Satisfiability for MPML is **decidable**.

Idea: brute force algorithm via **finite model property**.

Sat. for MPML is **NEXPTIME** (PML has a PSPACE-COMPLETE Sat.)

Satisfiability for MCTL, MCTL+, and MCTL* is **highly undecidable**.

Idea: reduction from the **recurrent domino problem**.

Sat. for MCTL is Σ_1^1 -HARD.

Satisfiability result

Satisfiability for MPML is **decidable**.

Idea: brute force algorithm via **finite model property**.

Sat. for MPML is **NEXPTIME** (PML has a PSPACE-COMPLETE Sat.)

Satisfiability for MCTL, MCTL+, and MCTL* is **highly undecidable**.

Idea: reduction from the **recurrent domino problem**.

Sat. for MCTL is **Σ_1^1 -HARD**.

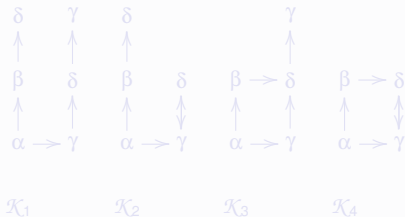
Proof sketch (I)

Let α , β , γ , and δ four incompatible formulas.

E.g., $\alpha = a \wedge b$, $\beta = \neg a \wedge b$, $\gamma = a \wedge \neg b$, and $\delta = \neg a \wedge \neg b$.

Consider the formula $\varphi_e = \alpha \wedge EX(\beta \wedge EX \delta) \wedge EX(\gamma \wedge EX(\delta \wedge EX \gamma))$.

There are only four minimal models (up to isomorphism) of φ_e :



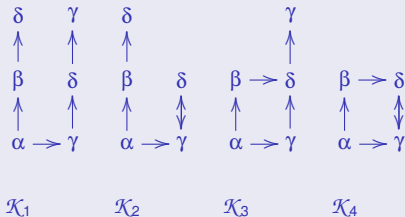
Proof sketch (I)

Let α , β , γ , and δ four incompatible formulas.

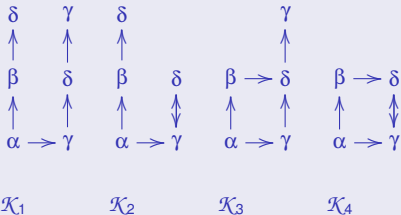
E.g., $\alpha = a \wedge b$, $\beta = \neg a \wedge b$, $\gamma = a \wedge \neg b$, and $\delta = \neg a \wedge \neg b$.

Consider the formula $\varphi_e = \alpha \wedge EX(\beta \wedge EX\delta) \wedge EX(\gamma \wedge EX(\delta \wedge EX\gamma))$.

There are only four minimal models (up to isomorphism) of φ_e :



Proof sketch (II)



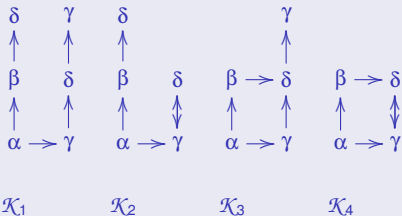
Consider now the formula $\varphi_v = \text{EX}(\beta \wedge \text{EX EX } \gamma)$.

Only \mathcal{K}_3 and \mathcal{K}_4 are models of φ_v .

Hence, $\varphi = \varphi_v \boxplus \varphi_e$ has necessarily a square model.

Using this idea, we are able to reduce the domino problem to MCTL Sat.

Proof sketch (II)



Consider now the formula $\varphi_v = EX(\beta \wedge EX EX \gamma)$.

Only \mathcal{K}_3 and \mathcal{K}_4 are models of φ_v .

Hence, $\varphi = \varphi_v \boxplus \varphi_e$ has necessarily a square model.

Using this idea, we are able to reduce the domino problem to MCTL Sat.

Complexity summary

	M.C. (Formula)	M.C. (Program)	Sat.
LTL	PSPACE-COMPLETE	NLOGSPACE-COMPLETE	PSPACE-COMPLETE
PML	PTIME	NLOGSPACE	PSPACE-COMPLETE
CTL	PTIME-COMPLETE	NLOGSPACE-COMPLETE	EXPTIME-COMPLETE
CTL+	Δ_2^P -COMPLETE	NLOGSPACE-COMPLETE	2EXPTIME-COMPLETE
CTL*	PSPACE-COMPLETE	NLOGSPACE-COMPLETE	2EXPTIME-COMPLETE
MPML	Δ_2^P	PSPACE	NEXPTIME
MCTL	Δ_2^P -COMPLETE	PSPACE	Σ_1^1 -HARD
MCTL+	Δ_2^P -COMPLETE	PSPACE	Σ_1^1 -HARD
MCTL*	PSPACE-COMPLETE	PSPACE	Σ_1^1 -HARD

Table: Computational complexity of Model Checking and Satisfiability.

Outline

1 Minimal Model Quantifiers in CTL*

- Syntax and Semantics
- Properties

2 Main results

- Model Checking
- Satisfiability

3 Open problems

4 Conclusion

Four open questions

Is the formula complexity of model checking for MPML **complete for Δ_2^P** ?

Is the complexities of satisfiability for MPML **complete for NEXPTIME**?

Is the program complexity for all logics **complete for PSPACE**?

Is the bisimulation-invariant fragment of MCTL* **equivalent** to CTL*?

Outline

- 1 Minimal Model Quantifiers in CTL*
 - Syntax and Semantics
 - Properties
- 2 Main results
 - Model Checking
 - Satisfiability
- 3 Open problems
- 4 Conclusion

Conclusion

In this work...

- we introduced MCTL*, i.e., CTL* augmented with Minimal Model Quantifiers (some similarity with *Arbitrary Announcement Logic*^a and *Sabotage Logic*^b),
- we study some elementary model-theoretic properties
 - expressiveness,
 - succinctness,
 - finite model property,
- finally, we show
 - the decidability of M.C. for all the introduced logics,
 - the decidability of Sat. for MPML,
 - the highly undecidability of Sat. for MCTL, MCTL+, and MCTL*.

^a T. French and H.P. van Ditmarsch. Undecidability for Arbitrary Public Announcement Logic, AIML'08.

^b C. Löding and P. Rohde. Model Checking and Satisfiability for Sabotage Modal Logic, FSTTCS'03.

Thank you very much for your attention!
I hope my talk was not too boring.