Branching-Time Temporal Logics with Minimal Model Quantifiers

Fabio Mogavero Aniello Murano

Universitá degli Studi di Napoli "Federico II", Italy
http://people.na.infn.it/~{mogavero,murano}

DLT 2009
Stuttgart, Germany
June 30 - July 3, 2009

Formal methods for systems correctness

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

Two very challenging 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 $\mathcal{K} \models \varphi$?

Formal methods for systems correctness

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

Two very challenging 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 $\mathcal{K} \models \varphi$?

Formal methods for systems correctness

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

Two very challenging 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 $\mathcal{K} \models \varphi$?

Formal methods for systems correctness

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

Two very challenging 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 $\mathcal{K} \models \varphi$?

Formal languages for 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.
  - Formulas can be interpreted over linear sequences.
  - Useful for hardware specification.
- Branching-Time Temporal Logics (PML, CTL, CTL+, and CTL*)
  - Each moment in time may split into various possible future.
  - Formulas can be interpreted on infinite trees.
  - Useful for software specification.

Formal languages for 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.
  - Formulas can be interpreted over linear sequences.
  - Useful for hardware specification.
- Branching-Time Temporal Logics (PML, CTL, CTL+, and CTL*)
  - Each moment in time may split into various possible future.
  - Formulas can be interpreted on infinite trees.
  - Useful for software specification.

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.

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.

Our proposal is to extend CTL* with Minimal Model Quantifiers.

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

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

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.

Our proposal is to extend CTL* with Minimal Model Quantifiers.

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

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

1. Minimal Model Quantifiers in CTL*
   - Syntax and Semantics
   - Properties

2. Main results
   - Model Checking
   - Satisfiability

3. Open problems

4. Conclusion

5. References

1 Minimal Model Quantifiers in CTL*
- Syntax and Semantics
- Properties

2 Main results
- Model Checking
- Satisfiability

3 Open problems

4 Conclusion

5 References

1 Minimal Model Quantifiers in CTL*
- Syntax and Semantics
- Properties

2 Main results
- Model Checking
- Satisfiability

3 Open problems

4 Conclusion

5 References

1 Minimal Model Quantifiers in CTL*
- Syntax and Semantics
- Properties

2 Main results
- Model Checking
- Satisfiability

3 Open problems

4 Conclusion

5 References

Syntax of MCTL* MCTL+, MCTL, and MPML

Definition

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

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

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

MCTL+: MCTL* without nesting of temporal operators [No: $p \,\mathrm{U}\, (\mathrm{X}\, q)$].

MCTL: MCTL+ without comb. of temporal operators [No: $(p \,\mathrm{U}\, q) \wedge (r \,\mathrm{R}\, s)$].

MPML: MCTL with next-time temporal operators only [No: $p \,\mathrm{U}\, q$].

Syntax of MCTL* MCTL+, MCTL, and MPML

Definition

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

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

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

MCTL+: MCTL* without nesting of temporal operators [No: $p\,\mathsf{U}\,(\mathsf{X}\,q)$].

MCTL: MCTL+ without comb. of temporal operators [No: $(p\,\mathsf{U}\,q) \wedge (r\,\mathsf{R}\,s)$].

MPML: MCTL with next-time temporal operators only [No: $p\,\mathsf{U}\,q$].

Syntax of MCTL* MCTL+, MCTL, and MPML

Definition

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

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

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

MCTL+: MCTL* without nesting of temporal operators [No: $p\,\mathsf{U}\,(\mathsf{X}\,q)$].

MCTL: MCTL+ without comb. of temporal operators [No: $(p\,\mathsf{U}\,q) \wedge (r\,\mathsf{R}\,s)$].

MPML: MCTL with next-time temporal operators only [No: $p\,\mathsf{U}\,q$].

Syntax of MCTL* MCTL+, MCTL, and MPML

Definition

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

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

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

MCTL+: MCTL* without nesting of temporal operators [No: $p\,\mathsf{U}\,(\mathsf{X}\,q)$].

MCTL: MCTL+ without comb. of temporal operators [No: $(p\,\mathsf{U}\,q) \wedge (r\,\mathsf{R}\,s)$].

MPML: MCTL with next-time temporal operators only [No: $p\,\mathsf{U}\,q$].

Syntax of MCTL* MCTL+, MCTL, and MPML

Definition

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

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

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

MCTL+: MCTL* without nesting of temporal operators [No: $p\,\mathsf{U}\,(\mathsf{X}\,q)$].

MCTL: MCTL+ without comb. of temporal operators [No: $(p\,\mathsf{U}\,q) \wedge (r\,\mathsf{R}\,s)$].

MPML: MCTL with next-time temporal operators only [No: $p\,\mathsf{U}\,q$].

Informal meaning of Ξ and Λ

Definition

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

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

Informally, Ξ and Λ can be read as

1. $\varphi_1 \,\Xi\, \varphi_2$: there is a submodel of φ_2 that satisfies φ_1,
2. $\varphi_1 \,\Lambda\, \varphi_2$: all submodels of φ_2 satisfy φ_1.

1. φ_1 is the *submodel verifier*.
2. φ_2 is the *submodel extractor*.

Informal meaning of Ξ and Λ

Definition

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

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

Informally, Ξ and Λ can be read as

1. $\varphi_1\, \Xi\, \varphi_2$: there is a submodel of φ_2 that satisfies φ_1,
2. $\varphi_1\, \Lambda\, \varphi_2$: all submodels of φ_2 satisfy φ_1.

1. φ_1 is the *submodel verifier*.
2. φ_2 is the *submodel extractor*.

Kripke structures, partial order, and minimality

Definition

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

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

A KRIPKE $\mathcal{K}'$ is a *substructure* of $\mathcal{K}$, formally $\mathcal{K}' \preccurlyeq \mathcal{K}$, 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 $\mathcal{K}' \in \mathrm{S}$, it holds that *(i)* $\mathcal{K} \preccurlyeq \mathcal{K}'$ or *(ii)* $\mathcal{K}' \not\preccurlyeq \mathcal{K}$.

By $\min(\mathrm{S})$ we denote the set of minimal structures (*antichain*) of S.

Kripke structures, partial order, and minimality

Definition

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

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

A KRIPKE $\mathcal{K}'$ is a *substructure* of $\mathcal{K}$, formally $\mathcal{K}' \preccurlyeq \mathcal{K}$, 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 $\mathcal{K}' \in \mathrm{S}$, it holds that *(i)* $\mathcal{K} \preccurlyeq \mathcal{K}'$ or *(ii)* $\mathcal{K}' \not\preccurlyeq \mathcal{K}$.

By $\min(\mathrm{S})$ we denote the set of minimal structures (*antichain*) of S.

Kripke structures, partial order, and minimality

Definition

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

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

A KRIPKE $\mathcal{K}'$ is a *substructure* of $\mathcal{K}$, formally $\mathcal{K}' \preccurlyeq \mathcal{K}$, 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 $\mathcal{K}' \in \mathrm{S}$, it holds that *(i)* $\mathcal{K} \preccurlyeq \mathcal{K}'$ or *(ii)* $\mathcal{K}' \not\preccurlyeq \mathcal{K}$.

By $\min(\mathrm{S})$ we denote the set of minimal structures (*antichain*) of S.

Kripke structures, partial order, and minimality

Definition

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

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

A KRIPKE $\mathcal{K}'$ is a *substructure* of $\mathcal{K}$, formally $\mathcal{K}' \preccurlyeq \mathcal{K}$, 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 $\mathcal{K}' \in \mathrm{S}$, it holds that *(i)* $\mathcal{K} \preccurlyeq \mathcal{K}'$ or *(ii)* $\mathcal{K}' \not\preccurlyeq \mathcal{K}$.

By $\min(\mathrm{S})$ we denote the set of minimal structures (*antichain*) of S.

Semantics of MCTL*

Definition

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

1. $\mathcal{K}, w \models \varphi_1 \Xi \varphi_2$ iff there is $\mathcal{K}' \in \min(\mathfrak{S}(\mathcal{K}, w, \varphi_2))$ such that $\mathcal{U}, w \models \varphi_1$;
2. $\mathcal{K}, w \models \varphi_1 \Lambda \varphi_2$ iff for all $\mathcal{K}' \in \min(\mathfrak{S}(\mathcal{K}, w, \varphi_2))$ it holds that $\mathcal{U}, w \models \varphi_1$.

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

An example

Consider the formula $\varphi = (\mathsf{EX}\,\mathsf{EX}\,p) \wedge (\mathsf{EX}\,\mathfrak{t})$, where $\mathfrak{t}$ means true.

φ is Sat!

Suppose that $\mathcal{K}, w \models \varphi$.

1. The submodel extractor $\mathsf{EX}\,\mathfrak{t}$ requires that w has an outcoming edge.
2. The submodel verifier $\mathsf{EX}\,\mathsf{EX}\,p$ 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 universal model quantifiers $\wedge$, we have that $\mathcal{K}$ is necessarily formed by a unique world with a self loop.

$\mathcal{K}$:

the world w is labeled with p

An example

Consider the formula $\varphi = (\mathsf{EX}\,\mathsf{EX}\,p) \curlywedge (\mathsf{EX}\,\mathfrak{t})$, where $\mathfrak{t}$ means true.

φ is Sat!

Suppose that $\mathcal{K}, w \models \varphi$.

1. The submodel extractor $\mathsf{EX}\,\mathfrak{t}$ requires that w has an outcoming edge.
2. The submodel verifier $\mathsf{EX}\,\mathsf{EX}\,p$ 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 universal model quantifiers $\curlywedge$, we have that $\mathcal{K}$ is necessarily formed by a unique world with a self loop.

$\mathcal{K}$:

the world w is labeled with p

An example

Consider the formula $\varphi = (\mathrm{EX}\,\mathrm{EX}\,p)\,\Lambda\,(\mathrm{EX}\,\mathfrak{t})$, where $\mathfrak{t}$ means true.

φ is Sat!

Suppose that $\mathcal{K}, w \models \varphi$.

1. The submodel extractor $\mathrm{EX}\,\mathfrak{t}$ requires that w has an outcoming edge.
2. The submodel verifier $\mathrm{EX}\,\mathrm{EX}\,p$ 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 universal model quantifiers Λ, we have that $\mathcal{K}$ is necessarily formed by a unique world with a self loop.

$\mathcal{K}$:

the world w is labeled with p

An example

Consider the formula $\varphi = (\mathsf{EX}\,\mathsf{EX}\,p) \curlywedge (\mathsf{EX}\,\mathfrak{t})$, where $\mathfrak{t}$ means true.

φ is Sat!

Suppose that $\mathcal{K}, w \models \varphi$.

1. The submodel extractor $\mathsf{EX}\,\mathfrak{t}$ requires that w has an outcoming edge.
2. The submodel verifier $\mathsf{EX}\,\mathsf{EX}\,p$ 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 universal model quantifiers $\curlywedge$, we have that $\mathcal{K}$ is necessarily formed by a unique world with a self loop.

$\mathcal{K}$:

the world w is labeled with p

An example

Consider the formula $\varphi = (\mathsf{EX}\,\mathsf{EX}\,p) \curlywedge (\mathsf{EX}\,\mathfrak{t})$, where $\mathfrak{t}$ means true.

φ is Sat!

Suppose that $\mathcal{K}, w \models \varphi$.

1. The submodel extractor $\mathsf{EX}\,\mathfrak{t}$ requires that w has an outcoming edge.
2. The submodel verifier $\mathsf{EX}\,\mathsf{EX}\,p$ 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 universal model quantifiers $\curlywedge$, we have that $\mathcal{K}$ is necessarily formed by a unique world with a self loop.

$\mathcal{K}$:

the world w is labeled with p

Elementary model properties (1)

Consider again $\varphi = (\mathsf{EX}\,\mathsf{EX}\,p) \land (\mathsf{EX}\,\mathfrak{t})$.

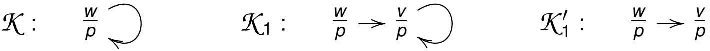
$\mathcal{K}_1$ is the one-step unwinding of $\mathcal{K}$.

- $\{\mathcal{K}\} = \min(\mathfrak{S}(\mathcal{K}, w, \mathsf{EX}\,\mathfrak{t})) = \mathfrak{S}(\mathcal{K}, w, \mathsf{EX}\,\mathfrak{t}) = \{\mathcal{K}\}$;
- $\{\mathcal{K}'_1\} = \min(\mathfrak{S}(\mathcal{K}_1, w, \mathsf{EX}\,\mathfrak{t})) \subset \mathfrak{S}(\mathcal{K}_1, w, \mathsf{EX}\,\mathfrak{t}) = \{\mathcal{K}_1, \mathcal{K}'_1\}$.

- Since $\mathcal{K}, w \models \mathsf{EX}\,\mathsf{EX}\,p$, it holds that $\mathcal{K}, w \models \varphi$;
- Instead, since $\mathcal{K}'_1, w \not\models \mathsf{EX}\,\mathsf{EX}\,p$, it holds that $\mathcal{K}_1, w \not\models \varphi$.

Elementary model properties (1)

Consider again $\varphi = (\mathsf{EX}\,\mathsf{EX}\,p) \wedge (\mathsf{EX}\,\mathfrak{t})$.



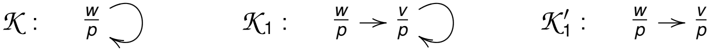
$\mathcal{K}_1$ is the one-step unwinding of $\mathcal{K}$.

- $\{\mathcal{K}\} = \min(\mathfrak{S}(\mathcal{K}, w, \mathsf{EX}\,\mathfrak{t})) = \mathfrak{S}(\mathcal{K}, w, \mathsf{EX}\,\mathfrak{t}) = \{\mathcal{K}\}$;
- $\{\mathcal{K}'_1\} = \min(\mathfrak{S}(\mathcal{K}_1, w, \mathsf{EX}\,\mathfrak{t})) \subset \mathfrak{S}(\mathcal{K}_1, w, \mathsf{EX}\,\mathfrak{t}) = \{\mathcal{K}_1, \mathcal{K}'_1\}$.

- Since $\mathcal{K}, w \models \mathsf{EX}\,\mathsf{EX}\,p$, it holds that $\mathcal{K}, w \models \varphi$;
- Instead, since $\mathcal{K}'_1, w \not\models \mathsf{EX}\,\mathsf{EX}\,p$, it holds that $\mathcal{K}_1, w \not\models \varphi$.

Elementary model properties (1)

Consider again $\varphi = (\mathsf{EX}\,\mathsf{EX}\,p) \wedge (\mathsf{EX}\,\mathfrak{t})$.



$\mathcal{K}_1$ is the one-step unwinding of $\mathcal{K}$.

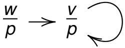
- $\{\mathcal{K}\} = \min(\mathfrak{S}(\mathcal{K}, w, \mathsf{EX}\,\mathfrak{t})) = \mathfrak{S}(\mathcal{K}, w, \mathsf{EX}\,\mathfrak{t}) = \{\mathcal{K}\}$;
- $\{\mathcal{K}'_1\} = \min(\mathfrak{S}(\mathcal{K}_1, w, \mathsf{EX}\,\mathfrak{t})) \subset \mathfrak{S}(\mathcal{K}_1, w, \mathsf{EX}\,\mathfrak{t}) = \{\mathcal{K}_1, \mathcal{K}'_1\}$.

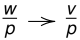
- Since $\mathcal{K}, w \models \mathsf{EX}\,\mathsf{EX}\,p$, it holds that $\mathcal{K}, w \models \varphi$;
- Instead, since $\mathcal{K}'_1, w \not\models \mathsf{EX}\,\mathsf{EX}\,p$, it holds that $\mathcal{K}_1, w \not\models \varphi$.

Elementary model properties (1)

Consider again $\varphi = (\mathsf{EX}\,\mathsf{EX}\,p) \wedge (\mathsf{EX}\,\mathfrak{t})$.

$\mathcal{K}$:


$\mathcal{K}_1$:


$\mathcal{K}'_1$:


$\mathcal{K}_1$ is the one-step unwinding of $\mathcal{K}$.

- $\{\mathcal{K}\} = \min(\mathfrak{S}(\mathcal{K}, w, \mathsf{EX}\,\mathfrak{t})) = \mathfrak{S}(\mathcal{K}, w, \mathsf{EX}\,\mathfrak{t}) = \{\mathcal{K}\}$;
- $\{\mathcal{K}'_1\} = \min(\mathfrak{S}(\mathcal{K}_1, w, \mathsf{EX}\,\mathfrak{t})) \subset \mathfrak{S}(\mathcal{K}_1, w, \mathsf{EX}\,\mathfrak{t}) = \{\mathcal{K}_1, \mathcal{K}'_1\}$.

- Since $\mathcal{K}, w \models \mathsf{EX}\,\mathsf{EX}\,p$, it holds that $\mathcal{K}, w \models \varphi$;
- Instead, since $\mathcal{K}'_1, w \not\models \mathsf{EX}\,\mathsf{EX}\,p$, it holds that $\mathcal{K}_1, w \not\models \varphi$.

Elementary model properties (2)

Hence, it is immediate to 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.

Elementary model properties (2)

Hence, it is immediate to 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.

Elementary model properties (2)

Hence, it is immediate to 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 (1)

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 (1)

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 (1)

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 (2)

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 traslation 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.

Succinctness (2)

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

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 traslation 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.

Succinctness (2)

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

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 traslation 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.

Model Checking result

Model Checking for all the introduced logics is decidable.

Idea: use of oracle machines.

Bad results:

- 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 results:

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

The program complexity is PSPACE.

Model Checking result

Model Checking for all the introduced logics is decidable.

Idea: use of oracle machines.

Bad results:

- 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 results:

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

The program complexity is PSPACE.

Model Checking result

Model Checking for all the introduced logics is decidable.

Idea: use of oracle machines.

Bad results:

- 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 results:

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

The program complexity is PSPACE.

Model Checking result

Model Checking for all the introduced logics is decidable.

Idea: use of oracle machines.

Bad results:

- 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 results:

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

The program complexity is PSPACE.

Proof sketch

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

Construction of a polynomial certificate $\mathcal{K}'$ of the test $\mathcal{K}, w \models \varphi_1 \, \Xi \, \varphi_2$ that is verifiable in

- PTIME for MCTL,
- PSPACE for MCTL*.

$\mathcal{K}', w \models \varphi_1$ and $\mathcal{K}', w \models \varphi_2$.

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 step of an oracle.

Proof sketch

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

Construction of a polynomial certificate $\mathcal{K}'$ of the test $\mathcal{K}, w \models \varphi_1 \, \Xi \, \varphi_2$ that is verifiable in

- PTIME for MCTL,
- PSPACE for MCTL*.

$\mathcal{K}', w \models \varphi_1$ and $\mathcal{K}', w \models \varphi_2$.

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 step of an oracle.

Proof sketch

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

Construction of a polynomial certificate $\mathcal{K}'$ of the test $\mathcal{K}, w \models \varphi_1 \,\Xi\, \varphi_2$ that is verifiable in

- PTIME for MCTL,
- PSPACE for MCTL*.

$\mathcal{K}', w \models \varphi_1$ and $\mathcal{K}', w \models \varphi_2$.

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 step of an oracle.

Proof sketch

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

Construction of a polynomial certificate $\mathcal{K}'$ of the test $\mathcal{K}, w \models \varphi_1 \, \Xi \, \varphi_2$ that is verifiable in

- PTIME for MCTL,
- PSPACE for MCTL*.

$\mathcal{K}', w \models \varphi_1$ and $\mathcal{K}', w \models \varphi_2$.

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 step of an 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 (1)

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 $\varphi_e = \alpha \wedge \mathrm{EX}(\beta \wedge \mathrm{EX}\,\delta) \wedge \mathrm{EX}(\gamma \wedge \mathrm{EX}(\delta \wedge \mathrm{EX}\,\gamma))$.

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

$\mathcal{K}_1$ $\mathcal{K}_2$ $\mathcal{K}_3$ $\mathcal{K}_4$

Proof sketch (1)

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 $\varphi_e = \alpha \wedge \mathrm{EX}(\beta \wedge \mathrm{EX}\,\delta) \wedge \mathrm{EX}(\gamma \wedge \mathrm{EX}(\delta \wedge \mathrm{EX}\,\gamma))$.

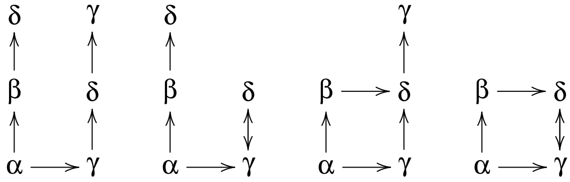
There are only four models (up to isomorphism) of φ_e:

Proof sketch (1)

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 $\delta = \neg a \wedge \neg b$.

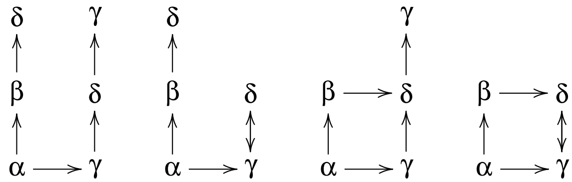
Consider the formula $\varphi_e = \alpha \wedge \mathsf{EX}(\beta \wedge \mathsf{EX}\, \delta) \wedge \mathsf{EX}(\gamma \wedge \mathsf{EX}(\delta \wedge \mathsf{EX}\, \gamma))$.

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



$\mathcal{K}_1$ $\mathcal{K}_2$ $\mathcal{K}_3$ $\mathcal{K}_4$

Proof sketch (2)



$\mathcal{K}_1$ $\mathcal{K}_2$ $\mathcal{K}_3$ $\mathcal{K}_4$

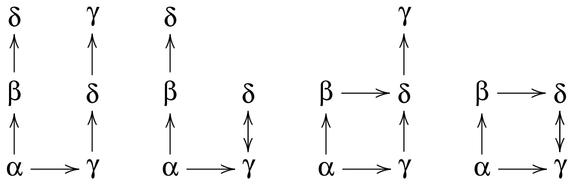
Consider now the formula $\varphi_v = \mathsf{EX}(\beta \wedge \mathsf{EX}\,\mathsf{EX}\,\gamma)$.

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

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

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

Proof sketch (2)



$\mathcal{K}_1$ $\mathcal{K}_2$ $\mathcal{K}_3$ $\mathcal{K}_4$

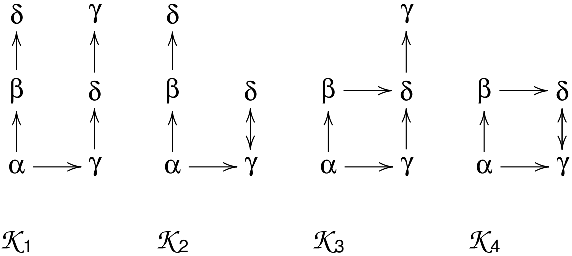
Consider now the formula $\varphi_v = \text{EX}(\beta \wedge \text{EX}\,\text{EX}\,\gamma)$.

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

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

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

Proof sketch (2)



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

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

Hence, $\varphi = \varphi_v \,\Xi\, \varphi_e$ 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.

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*?

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*?

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*?

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*?

Conclusion

In this work...

- we introduced MCTL*, i.e., CTL* augmented with Minimal Model Quantifiers (some similarity with *Arbitrary Announcement Logic*[1] and *Sabotage Logic*[2]),
- 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*.

---

[1] T. French and H.P. van Ditmarsch. Undecidability for Arbitrary Public Announcement Logic

[2] C. Löding and P. Rohde. Model Checking and Satisfiability for Sabotage Modal Logic

Conclusion

In this work...

- we introduced MCTL*, i.e., CTL* augmented with Minimal Model Quantifiers (some similarity with *Arbitrary Announcement Logic*[1] and *Sabotage Logic*[2]),
- 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*.

[1] T. French and H.P. van Ditmarsch. Undecidability for Arbitrary Public Announcement Logic

[2] C. Löding and P. Rohde. Model Checking and Satisfiability for Sabotage Modal Logic

Conclusion

In this work...

- we introduced MCTL*, i.e., CTL* augmented with Minimal Model Quantifiers (some similarity with *Arbitrary Announcement Logic*[1] and *Sabotage Logic*[2]),
- 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*.

---

[1] T. French and H.P. van Ditmarsch. Undecidability for Arbitrary Public Announcement Logic

[2] C. Löding and P. Rohde. Model Checking and Satisfiability for Sabotage Modal Logic

Conclusion

In this work...

- we introduced MCTL*, i.e., CTL* augmented with Minimal Model Quantifiers (some similarity with *Arbitrary Announcement Logic*[1] and *Sabotage Logic*[2]),
- 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*.

---

1 T. French and H.P. van Ditmarsch. Undecidability for Arbitrary Public Announcement Logic

2 C. Löding and P. Rohde. Model Checking and Satisfiability for Sabotage Modal Logic

References

- Martin Abadi, Leslie Lamport: Composing Specifications. ACM Trans. Program. Lang. Syst. 15(1): 73-132 (1993)
- Orna Kupferman, Gila Morgenstern, Aniello Murano: Typeness for omega-regular Automata. Int. J. Found. Comput. Sci. 17(4): 869-884 (2006)
- Orna Kupferman, Moshe Y. Vardi, Pierre Wolper: An automata-theoretic approach to branching-time model checking. J. ACM 47(2): 312-360 (2000)
- Orna Kupferman, Moshe Y. Vardi: An automata-theoretic approach to modular model checking. ACM Trans. Program. Lang. Syst. 22(1): 87-128 (2000)
- Orna Kupferman, Moshe Y. Vardi: Modular Model Checking. COMPOS 1997: 381-401
- Fabio Mogavero, Aniello Murano: Branching-Time Temporal Logics with Minimal Model Quantifiers. Developments in Language Theory 2009: 396-409
- Alessandro Bianco, Fabio Mogavero, Aniello Murano: Graded Computation Tree Logic. LICS 2009: 342-351
- Piero A. Bonatti, Carsten Lutz, Aniello Murano, Moshe Y. Vardi: The Complexity of Enriched Mu-Calculi. Logical Methods in Computer Science 4(3) (2008)