Complexity and Expressivity of Branching- and Alternating-Time Temporal Logics with Finitely Many Variables

Mikhail Rybakov and Dmitry Shkatov

ICTAC 2018
Temporal logics—such as **CLT** (Computational Tree Logic), **CTL***, **ALT** (Alternating-Time Temporal Logic), and **ATL***—are used in formal specification and verification of software and hardware.

In verification, they are used to verify that an implemented system is correct when other verification methods are not guaranteed to succeed (i.e., verification of parallel programs such as operating systems).

In specification, they are used to make sure that a specification is satisfiable and thus a system conforming to a specification can be built.

In this talk, we will be looking at the use of these logics in formal specification.
Temporal logics—such as **CLT** (Computational Tree Logic), **CTL***, **ALT** (Alternating-Time Temporal Logic), and **ATL***—are used in formal specification and verification of software and hardware.

In verification, they are used to verify that an implemented system is correct when other verification methods are not guaranteed to succeed (i.e., verification of parallel programs such as operating systems).

In specification, they are used to make sure that a specification is satisfiable and thus a system conforming to a specification can be built.

In this talk, we will be looking at the use of these logics in formal specification.
Motivation

Temporal logics—such as **CLT** (Computational Tree Logic), **CTL***, **ALT** (Alternating-Time Temporal Logic), and **ATL***—are used in formal specification and verification of software and hardware.

In verification, they are used to verify that an implemented system is correct when other verification methods are not guaranteed to succeed (i.e., verification of parallel programs such as operating systems).

In specification, they are used to make sure that a specification is satisfiable and thus a system conforming to a specification can be built.

In this talk, we will be looking at the use of these logics in formal specification.
Temporal logics—such as CLT (Computational Tree Logic), CTL*, ALT (Alternating-Time Temporal Logic), and ATL*—are used in formal specification and verification of software and hardware.

In verification, they are used to verify that an implemented system is correct when other verification methods are not guaranteed to succeed (i.e., verification of parallel programs such as operating systems).

In specification, they are used to make sure that a specification is satisfiable and thus a system conforming to a specification can be built.

In this talk, we will be looking at the use of these logics in formal specification.
When using temporal logics in specification, we

1. construct a formula, say \( \varphi \), expressing a specification;

2. test that there exists a structure \( M \) modelling a system of the type we are interested in (for programs, this is the graph that models execution paths of the program) such that \( \varphi \) is true in \( M \).

If we have succeeded, then the specification expressed by \( \varphi \) is satisfiable. Moreover, we can use \( M \) in building an implemented system.
The problem with this vision is that testing a temporal formula for satisfiability is hard. Namely,

- for $\text{CTL}$ and $\text{ATL}$, it is EXPTIME-complete.
- for $\text{CTL}^*$ and $\text{ATL}^*$, it is 2EXPTIME-complete.

Therefore, it is interesting to see if the languages of these logics can be restricted so that we obtain an expressive fragment with a more tractable satisfiability problem.

In particular, it has been noticed that most specifications used in practice contain a very small number of primitive propositions (usually, no more than three).
Motivation, continued

The problem with this vision is that testing a temporal formula for satisfiability is hard. Namely,

- for $\mathbf{CTL}$ and $\mathbf{ATL}$, it is EXPTIME-complete.
- for $\mathbf{CTL}^*$ and $\mathbf{ATL}^*$, it is 2EXPTIME-complete.

Therefore, it is interesting to see if the languages of these logics can be restricted so that we obtain an expressive fragment with a more tractable satisfiability problem.

In particular, it has been noticed that most specifications used in practice contain a very small number of primitive propositions (usually, no more than three).
The problem with this vision is that testing a temporal formula for satisfiability is hard. Namely,

- for $\mathsf{CTL}$ and $\mathsf{ATL}$, it is $\text{EXPTIME}$-complete.
- for $\mathsf{CTL}^*$ and $\mathsf{ATL}^*$, it is $2\text{EXPTIME}$-complete.

Therefore, it is interesting to see if the languages of these logics can be restricted so that we obtain an expressive fragment with a more tractable satisfiability problem.

In particular, it has been noticed that most specifications used in practice contain a very small number of primitive propositions (usually, no more than three).
The main question

For any of **CLT**, **CTL***, **ALT**, or **ATL***, can we obtain a fragment with a tractable (or, at least, less hard) satisfiability problem by restricting the number of primitive propositions allowed in the construction of formulas?

For some logics (i.e., the extensions of **K5**), placing a restriction on the number of primitive propositions produces tractable fragments, see [Nagle, Thomason 1975].

For others, a restriction to one or even zero primitive propositions produces fragments as hard as the entire logic, see [Blackburn and Spaan 1993, Halpern 1995, Chagrov and Rybakov 2003].
The main question

For any of **CLT**, **CTL***, **ALT**, or **ATL***, can we obtain a fragment with a tractable (or, at least, less hard) satisfiability problem by restricting the number of primitive propositions allowed in the construction of formulas?

For some logics (i.e., the extensions of **K5**), placing a restriction on the number of primitive propositions produces tractable fragments, see [Nagle, Thomason 1975].

For others, a restriction to one or even zero primitive propositions produces fragments as hard as the entire logic, see [Blackburn and Spaan 1993, Halpern 1995, Chagrov and Rybakov 2003].
For any of $\text{CLT}$, $\text{CTL}^*$, $\text{ALT}$, or $\text{ATL}^*$, can we obtain a fragment with a tractable (or, at least, less hard) satisfiability problem by restricting the number of primitive propositions allowed in the construction of formulas?

For some logics (i.e., the extensions of $\text{K5}$), placing a restriction on the number of primitive propositions produces tractable fragments, see [Nagle, Thomason 1975].

For others, a restriction to one or even zero primitive propositions produces fragments as hard as the entire logic, see [Blackburn and Spaan 1993, Halpern 1995, Chagrov and Rybakov 2003].
Our paper shows that the answer is NO for CLT, CTL*, ALT, and ATL*.

Namely, we show that restricting the languages of these logics to one primitive proposition produces fragments as expressive as the entire logics; therefore, the satisfiability problem for those fragments is as hard as for the entire logics.

While doing so, we present a technique that can be used in other contexts, as well (for example, Propositional Dynamic Logics).

For clarity, in the talk, we only present the details for CTL. The idea for the other logics is similar, and the details can be found in the paper.
The main question, continued

Our paper shows that the answer is NO for CLT, CTL*, ALT, and ATL*.

Namely, we show that restricting the languages of these logics to one primitive proposition produces fragments as expressive as the entire logics; therefore, the satisfiability problem for those fragments is as hard as for the entire logics.

While doing so, we present a technique that can be used in other contexts, as well (for example, Propositional Dynamic Logics).

For clarity, in the talk, we only present the details for CTL. The idea for the other logics is similar, and the details can be found in the paper.
The main question, continued

Our paper shows that the answer is NO for \textit{CLT}, \textit{CTL}\(^*\), \textit{ALT}, and \textit{ATL}\(^*\).

Namely, we show that restricting the languages of these logics to one primitive proposition produces fragments as expressive as the entire logics; therefore, the satisfiability problem for those fragments is as hard as for the entire logics.

While doing so, we present a technique that can be used in other contexts, as well (for example, Propositional Dynamic Logics).

For clarity, in the talk, we only present the details for \textit{CTL}. The idea for the other logics is similar, and the details can be found in the paper.
Formulas of **CTL**

Formulas are defined by the following BNF expression:

\[ \varphi ::= p \mid \bot \mid (\varphi \rightarrow \varphi) \mid \forall X \varphi \mid \forall (\varphi U \varphi) \mid \exists (\varphi U \varphi), \]

where \( p \) is a propositional variable (primitive proposition).

As usual, we use the following abbreviations:

\[ \neg \varphi := (\varphi \rightarrow \bot); \]
\[ (\varphi \land \psi) := \neg (\varphi \rightarrow \neg \psi); \]
\[ (\varphi \lor \psi) := (\neg \varphi \rightarrow \psi); \]
\[ \top = \bot \rightarrow \bot; \]
\[ \exists X \varphi := \neg \forall X \neg \varphi; \]
\[ \exists \diamond \varphi := \exists (\top U \varphi); \]
\[ \forall \Box \varphi := \neg \exists \diamond \neg \varphi. \]
Formulas of CTL

Formulas are defined by the following BNF expression:

\[ \varphi ::= p \mid \bot \mid (\varphi \rightarrow \varphi) \mid \forall X \varphi \mid \forall (\varphi U \varphi) \mid \exists (\varphi U \varphi), \]

where \( p \) is a propositional variable (primitive proposition).

As usual, we use the following abbreviations:

\[
\neg \varphi := (\varphi \rightarrow \bot); \\
(\varphi \land \psi) := \neg(\varphi \rightarrow \neg \psi); \\
(\varphi \lor \psi) := (\neg \varphi \rightarrow \psi); \\
\top = \bot \rightarrow \bot; \\
\exists X \varphi := \neg \forall X \neg \varphi; \\
\exists \lozenge \varphi := \exists (\top U \varphi); \\
\forall \Box \varphi := \neg \exists \lozenge \neg \varphi.
\]
A *Kripke model* is defined as $\mathcal{M} = (\mathcal{S}, \rightarrow, V)$, where $\mathcal{S}$ is a non-empty set (of states), $\rightarrow$ is a binary (transition) relation on $\mathcal{S}$ that is serial (i.e., for every $s \in \mathcal{S}$, there exists $s' \in \mathcal{S}$ such that $s \rightarrow s'$), and $V$ is a (valuation) function $V : \text{Var} \rightarrow 2^\mathcal{S}$.

An infinite sequence $s_0, s_1, \ldots$ of states in $\mathcal{M}$ such that $s_i \rightarrow s_{i+1}$, for every $i \geq 0$, is called a *path*. 

Semantics of **CTL**

Mikhail Rybakov and Dmitry Shkatov

Complexity of Logics with Finitely Many Variables
Satisfaction relation

\[ M, s \models p_i \iff s \in V(p_i); \]

\[ M, s \models \bot \text{ never holds}; \]

\[ M, s \models \varphi_1 \rightarrow \varphi_2 \iff M, s \models \varphi_1 \text{ implies } M, s \models \varphi_2; \]

\[ M, s \models \forall X \varphi_1 \iff M, s' \models \varphi_1 \text{ whenever } s \rightarrow s'. \]

\[ M, s \models \forall (\varphi_1 U \varphi_2) \iff \text{for every path } s_0 \rightarrow s_1 \rightarrow \ldots \text{ with } s_0 = s, \]
\[ M, s_i \models \varphi_2, \text{ for some } i \geq 0, \text{ and } M, s_j \models \varphi_1, \text{ for every } 0 \leq j < i; \]

\[ M, s \models \exists (\varphi_1 U \varphi_2) \iff \text{there exists a path } s_0 \rightarrow s_1 \rightarrow \ldots \text{ with } s_0 = s, \text{ such that } M, s_i \models \varphi_2, \text{ for some } i \geq 0, \text{ and } M, s_j \models \varphi_1, \text{ for every } 0 \leq j < i. \]
A formula $\varphi$ is **satisfiable** if there exists a model $M$ and a state $s$ such that $M, s \models \varphi$.

**Satisfiability problem**

Given a formula $\varphi$, determine whether $\varphi$ is satisfiable.

**Fact**

Satisfiability problem for **CTL** is EXPTIME-complete.
In the rest of the talk, we show that $\text{CTL}$ can be polynomial-time embedded in into its one-variable fragment. That is, given a formula $\varphi$, we construct, in time polynomial in the size of $\varphi$, a formula $\varphi^*$ such that $\varphi^*$ is satisfiable if, and only if, $\varphi$ is satisfiable.

It then follows that satisfiabilty problem for the one-variable fragment of $\text{CTL}$ is $\text{EXPTIME}$-complete.
Construction of \( \varphi^* \), step 1 – constructing \( \hat{\varphi} \)

We assume that \( \varphi \) contains variables \( p_1, \ldots, p_n \). Define the translation \( \cdot' \) as follows:

\[
\begin{align*}
\pi' & = \pi \quad \text{where } i \in \{1, \ldots, n\}; \\
(\bot)' & = \bot; \\
(\phi \rightarrow \psi)' & = \phi' \rightarrow \psi'; \\
(\forall \, \chi \varphi)' & = \forall \, \chi (p_{n+1} \rightarrow \phi'); \\
(\forall (\varphi \cup \psi))' & = \forall (\varphi' \cup (p_{n+1} \land \psi')); \\
(\exists (\varphi \cup \psi))' & = \exists (\varphi' \cup (p_{n+1} \land \psi')).
\end{align*}
\]

Next, let

\[
\Theta = p_{n+1} \land \forall \Box (\exists \, \chi p_{n+1} \leftrightarrow p_{n+1}).
\]

and define

\[
\hat{\varphi} = \Theta \land \varphi'.
\]
Construction of $\varphi^*$, step 1 – constructing $\hat{\varphi}$, continued

**Lemma**

If $\hat{\varphi}$ is satisfiable, then it is satisfied in a model where $p_{n+1}$ is true at every state.

**Proof.**

Let $M, s \models \hat{\varphi}$. Consider the submodel $M'$ of $M$ consisting of the states where $p_{n+1}$ is true. Then, $M', s \models \hat{\varphi}$. □

**Lemma**

Formula $\varphi$ is satisfiable if, and only if, formula $\hat{\varphi}$ is satisfiable.

**Proof.**

First, observe that $\varphi$ is equivalent to $\hat{\varphi}[p_{n+1}/T]$. Then,

$(\Rightarrow)$ From closure of $\text{CTL}$ under substitution.

$(\Leftarrow)$ From the previous lemma. □
**Lemma**

If $\hat{\varphi}$ is satisfiable, then it is satisfied in a model where $p_{n+1}$ is true at every state.

**Proof.**

Let $M, s \models \hat{\varphi}$. Consider the submodel $M'$ of $M$ consisting of the states where $p_{n+1}$ is true. Then, $M', s \models \hat{\varphi}$.

**Lemma**

Formula $\varphi$ is satisfiable if, and only if, formula $\hat{\varphi}$ is satisfiable.

**Proof.**

First, observe that $\varphi$ is equivalent to $\hat{\varphi}[p_{n+1}/\top]$. Then,

(⇒) From closure of $\mathbf{CTL}$ under substitution.

(⇐) From the previous lemma.
Construction of $\varphi^*$, step 2 – modelling of variables of $\hat{\varphi}$

To obtain $\hat{\varphi}$, we model all the variables of $\hat{\varphi}$ using single-variable formulas $A_1, \ldots, A_{n+1}$.

To each formula $A_m$, where $m \in \{1, \ldots, n+1\}$, there corresponds a model $M_m$. 
Construction of $\varphi^*$, step 2 – modelling of variables of $\hat{\varphi}$

\[\begin{align*}
\delta_0 &= \forall \Box p; \\
\delta_{k+1} &= p \land \exists X(\neg p \land \exists X \delta_k). \\
A_m &= \delta_m \land \exists X \forall \Box \neg p.
\end{align*}\]
Construction of $\varphi^*$, step 2 – modelling of variables of $\hat{\varphi}$

$$B_m = \exists x A_m.$$  

$$\varphi^* = \hat{\varphi}[p_i/B_i].$$

**Lemma**

Formula $\varphi^*$ is satisfiable if, and only if, formula $\varphi$ is satisfiable.

**Proof.**

(⇒) From closure of $\text{CTL}$ under substitution.

(⇐) Transform the model satisfying $\varphi$ by attaching to a state satisfying the variable $p_i$ the model $M_i$. 

☐
Main results

Theorem

There exists a polynomial-time computable function \( e \) assigning to every \( \text{CTL} \)-formula \( \varphi \) a single-variable formula \( e(\varphi) \) such that \( e(\varphi) \) is satisfiable if, and only if, \( \varphi \) is satisfiable.

Theorem

The satisfiability problem for the single-variable fragment of \( \text{CTL} \) is \( \text{EXPTIME} \)-complete.

Proof.

The lower bound follows from the previous theorem and \( \text{EXPTIME} \)-hardness of satisfiability for \( \text{CTL} \). The upper bound follows from the \( \text{EXPTIME} \) upper bound for satisfiability for \( \text{CTL} \).
Main results

Theorem
There exists a polynomial-time computable function \( e \) assigning to every \( \text{CTL} \)-formula \( \varphi \) a single-variable formula \( e(\varphi) \) such that \( e(\varphi) \) is satisfiable if, and only if, \( \varphi \) is satisfiable.

Theorem
The satisfiability problem for the single-variable fragment of \( \text{CTL} \) is \( \text{EXPTIME} \)-complete.

Proof.
The lower bound follows from the previous theorem and \( \text{EXPTIME} \)-hardness of satisfiability for \( \text{CTL} \). The upper bound follows from the \( \text{EXPTIME} \) upper bound for satisfiability for \( \text{CTL} \).
The technique we have used for **CTL** can be applied to a wide variety of modal logics used in computer science:

- Other branching- and alternating-time temporal logics, see the paper
- Propositional Dynamic Logics, see [Rybakov, Shkatov 2018]
- Epistemic logics with the common knowledge operator
- Temporal-epistemic logics

The technique is modular with respect to how it handles modalities, so it can be applied to logics combining various modalities.
The main open question

How do we restrict the languages of these logics to obtain fragments with the tractable satisfiability problem?