Reasoning about Action and Planning in LTL action Theories

Diego Calvanese, Giuseppe De Giacomo, Moshe Y. Vardi

Alessio Antonini, Giuseppe Rizzo, Manuel Rodriguez

January 30th, 2014
This paper is about actions and planning with incomplete information in a dynamic system represented with Linear Temporal Logic (LTL) and Quantified Linear Temporal Logic (QLTL).
Introduction

The paper addresses three main issues:

i  reasoning about action effects (i.e., projection, historical queries)

ii  legally execution of actions in a given situation

iii  finding conformant plans for temporally extended goals

For each of these problems authors present techniques and characterize the computational complexity.
Introduction

Settings

- The system is described with a set of atomic facts (fluents)
- Fluents have a truth value which change as result of actions
- The behavior of the system as a set of sequences of situations
- Actions cause transitions from one situation to another
Linear Temporal Logic (LTL) is a linear-time temporal logic used for *specifying and verifying properties* of dynamic systems, such as:

- safety,
- liveness,
- fairness,
- etc.
Formulas of LTL are built from a set $\mathcal{P}$ of propositional symbols and are closed under the boolean operators:

- the unary temporal operators $\bigcirc$, $\Box$ and $\Diamond$
- the binary temporal operator $\mathcal{U}$
Linear Temporal Logic

Operators

\( \diamond \varphi \vdash \varphi \) holds at the next instant

\( \Diamond \varphi \vdash \varphi \) will eventually hold at some future instant

\( \Box \varphi \vdash \) from the current instant on \( \varphi \) will always hold

\( \varphi \mathcal{U} \psi \vdash \) at some future instant \( \psi \) will hold and until that point \( \varphi \) holds

\( \lor \vdash \) or

\( \rightarrow \vdash \) implies

\( \Diamond \varphi \vdash \) true \( \mathcal{U} \varphi \)

\( \Box \varphi \vdash \neg \Diamond \neg \varphi \)
The semantics of LTL is given in terms of interpretations over a linear structure $\mathbb{N}$. Given an instant $i \in \mathbb{N}$, the successive instant is $i + 1$.

An interpretation is a function $\pi : \mathbb{N} \to 2^\mathcal{P}$, which assigns to each element of $\mathcal{P}$ a truth value at each instant $i \in \mathbb{N}$. 
Linear Temporal Logic

Semantic

For an interpretation $\pi$ is inductively defined when an LTL formula $\varphi$ is true at an instant $i \in \mathbb{N}$, in symbols $\pi, i \models \varphi$, as follows:

$$\pi, i \models p, p \in \mathcal{P} \text{ iff } p \in \pi(i)$$
$$\pi, i \models \neg \varphi \text{ iff } \neg \pi, i \models \varphi$$
$$\pi, i \models \varphi \land \varphi' \text{ iff } \pi, i \models \varphi \text{ and } \pi, i \models \varphi'$$
$$\pi, i \models \diamond \varphi \text{ iff } \pi, i + 1 \models \varphi$$
$$\pi, i \models \varphi U \varphi' \text{ iff for some } j \geq i \text{ we have }$$
$$\pi, j \models \varphi' \text{ and for all } k, i \leq k < j, \text{ we have that }$$
$$\pi, k \models \varphi$$

A formula $\varphi$ is true in $\pi$ in notation $\pi \models \varphi$, if $\pi, 0 \models \varphi$. A formula $\varphi$ is satisfiable if it is true in some interpretation, and is valid, if it is true in every interpretation.
Linear Temporal Logic
Büchi automata

LTL is connected to Büchi automata.

- Given $\mathcal{P}$, the set of interpretations $2^\mathcal{P}$ of the propositional variables in $\mathcal{P}$ can be considered the alphabet of a Büchi automaton.
- An infinite word over the alphabet $2^\mathcal{P}$ accepted by an automaton can be viewed as an interpretation of an LTL formula over $\mathcal{P}$. 

Linear Temporal Logic

Büchi automata

LTL formula

- Represents a set of infinite traces which satisfy such formula

Büchi Automaton

- Accepts a set of infinite traces

We can build an automaton which accepts all and only the infinite traces represented by an LTL formula.
Given a finite non empty alphabet $\Sigma$, an *infinite word* is an element of $\Sigma^\omega$. A Büchi automaton is a tuple $\mathcal{A} = (\Sigma, S, S_0, \rho, F)$ where:

- $\Sigma$ is the alphabet of the automaton
- $S$ is the set of finite states
- $S_0$ is the initial state
- $\rho : S \times \Sigma \rightarrow 2^S$ is the transition function
- $F \subseteq S$

The *input words* of $\mathcal{A}$ are infinite words $a_0, a_1, \ldots \in \Sigma^\omega$. 

*Linear Temporal Logic* 

*Büchi automata*
Linear Temporal Logic
Büchi automata

Theorem [1]: for every LTL formula $\varphi$ one can effectively construct a Büchi automaton $A_\varphi$ whose number of states is at most exponential in the length of $\varphi$ and such that $L(A_\varphi)$ is the set of models of $\varphi$. 
Linear Temporal Logic

Büchi automata

\[ \sigma_1 = S_0 S_1 S_2 S_2 S_2 S_2 \ldots \quad \text{ACCEPTED} \]

\[ \sigma_2 = S_0 S_1 S_2 S_1 S_2 S_1 \ldots \quad \text{ACCEPTED} \]

\[ \sigma_3 = S_0 S_1 S_2 S_1 S_1 S_1 \ldots \quad \text{REJECTED} \]
Quantified Linear Temporal Logic (QLTL) is an extension of LTL. QLTL formulas are LTL formulas plus the existential quantifier of propositions

- $\exists p. \varphi(p)$ where $p$ is a proposition variable and $\varphi(p)$ a QLTL formula in which $p$ occurs free
- $\forall p. \varphi(p) = \neg \exists p. \varphi(p)$

The semantic is defined as follows:

- $\pi, i \models \exists p. \varphi$ iff there is some $\pi'$ that agrees with $\pi$ except for the interpretation of proposition $p$, and such that $\pi', i \models \varphi$. 
Theorem [2]: satisfiability for $\sum_{k+1}^{QLTL}$ formulas [...], with $k \geq 1$ is $k$-EXSPACE-complete.
Reasoning about Actions in LTL

General considerations

The behavior of a dynamic system is characterized by a set of evolutions (i.e. sequence of situations)

- Under incomplete information, only a set of possible evolutions can be isolated
- Dynamic system are specified by:
  - a set of fluents $F$, facts concerning the current situation
  - a set of actions $A$, the effects of actions on such a set of facts
A dynamic system can be described as a conjunction of a finite set of LTL formulas.

- add the following LTL formulas to model actions:
  - $\Box(\bigvee_{a \in A} a)$ i.e. an action can be performed at a time
  - $\Box(\bigwedge_{a \in A} (a \rightarrow \bigwedge_{b \in A, b \neq a} \neg b))$ i.e. an action must be performed

In this way an unique proposition $a \in A$ holds in each situation and it specifies the action performed to get to that situation.
Reasoning about Actions in LTL

Specifying effects

In order to specify the effects of an action we need to:

- describe an initial situation by means of a formula $\varphi_{init}$ that involves only fluents
- describe effects of actions by means of formulas of the form

$$\square(\varphi \rightarrow (\circ a \rightarrow \psi))$$

where $\varphi$ and $\psi$ are LTL formulas which involves only fluents
- some contraints can be specified as follows

$$\square(\phi)$$
Using this formalization, there is uncertainty about the initial situation and the effects of the actions

Complete specification using Reiter’s successor state axioms

\[ \circ F \equiv \bigvee_{a \in A} (\varphi_a^+ \land \circ a) \lor (F \land \bigwedge (\neg \varphi_b^- \lor \circ b)) \]

where \( F \) are fluents, \( a \)'s are action that make \( F \) become true under situations described by \( \varphi_a^+ \) and \( b \)'s are actions that makes \( F \) become false under \( \varphi_b^- \)
Reasoning about Actions in LTL

Problem: given a finite sequence of action, determines whether a certain property holds

- Projection problem: does the property $\phi$ hold after the execution of the sequence of actions $a_0 \cdots a_k$?
- Historical queries: does the property $\phi$ always hold (resp. hold at some point) over the duration of the sequence of actions $a_0 \cdots a_k$?

The problem is P-SPACE COMPLETE (due to validity of LTL formula)
The following formula needs to be introduced

\[ \text{Occurs}(a_0, \ldots, a_k) \]
\[ \triangleq (a_0 \land \Diamond (a_1 \land \Diamond (\cdots \Diamond (a_k \land rs) \cdots))) \land \Box (rs \rightarrow \Diamond \Box \neg rs) \]

it represents that the execution of \(a_0, \ldots, a_k\) results in a situation \(rs\) that will be true once
Reasoning about Actions in LTL

Reasoning about Actions Effects in LTL

- Given a system description $\Gamma$:
  - Projection problem can be solved checking that:
    \[
    \Gamma \rightarrow (\text{Occurs}(a_0 \cdots a_k, rs) \rightarrow \Box(rs \rightarrow \phi))
    \]
  - Historical queries can be solved checking that:
    \[
    \Gamma \rightarrow (\text{Occurs}(a_0 \cdots a_k, rs) \rightarrow \Box(\Diamond(rs) \rightarrow \phi))
    \]
    \[
    \Gamma \rightarrow (\text{Occurs}(a_0 \cdots a_k, rs) \rightarrow \Box(\varphi \land \Diamond(rs)))
    \]
Various formalisms for reasoning about actions use a prescriptive approach has been proposed:
- specifying the circumstances for the execution

Instead, LTL uses a non-prescriptive approach:
- it is always possible to execute an action, unless it contradicts the system specification
- due to the partial knowledge ensuring the executability of an action is needed
Reasoning about Actions in LTL

Legal sequences action: dealing with uncomplete knowledge

- **principle of directionality**: information about a given time is deductively independent from information about a later time.
- Instead, LTL introduces the notion of legality of a sequence of actions w.r.t. $\Gamma$. 
Consistency: An infinite sequence of actions $a_0, a_1, \ldots$ is consistent with $\Gamma$ if there exists a model of $\Gamma$ whose interpretation of the actions coincides with $a_0, a_1, \ldots$.

Legality: an action $a_{k+1}$ is legal after the sequence of actions $a_0, \ldots, a_k$ if for all sequences $\sigma_0, \ldots, \sigma_k$ of truth assignments to the fluents in $\mathcal{F}$, if $a_0, \ldots, a_k$ and $\sigma_0, \ldots, \sigma_k$ are consistent with $\Gamma$, then also $a_0, \ldots, a_k, a_{k+1}$ and $\sigma_0, \ldots, \sigma_k$ are consistent with $\Gamma$. 
Legality can be captured in QLTL.

Let $\text{now}$ a proposition acts as marker and $\bar{x}$ and $\bar{y}$ are tuples of variables, one for each fluent, and $\bar{x} \equiv \bar{y}$ stands for the conjunction of equivalences among corresponding components.

The following QLTL formulas can be defined:

$$\text{Point} (\text{now}) \doteq \Diamond (\text{now}) \land \Box (\text{now} \rightarrow \Diamond \Box \neg \text{now})$$

$$\text{EqUntil} (\bar{x}, \bar{y}, \text{now}) \doteq \Box ((\Diamond \text{now}) \rightarrow \bar{x} \equiv \bar{y})$$

$$\text{EqNext} (\bar{x}, \bar{y}, \text{now}) \doteq \Box (\text{now} \rightarrow \Diamond (\bar{x} \equiv \bar{y}))$$
Reasoning about Actions in LTL

Legality in QLTL

- **Point**(now) expresses that now holds at a single time point
- **EqUntil**($\vec{x}$, $\vec{y}$, now) expresses that $\vec{x}$ and $\vec{y}$ coincide at every time point until now holds.
- **EqNext**($\vec{x}$, $\vec{y}$, now) expresses that $\vec{x}$ and $\vec{y}$ coincide at the time point following the one where now holds.
Given a system specification $\Gamma(\tilde{a}, \tilde{f})$ the legality can be defined as:

$$\text{LegalNext}(\Gamma, \tilde{a}, \text{now}) \doteq \text{Point(\text{now})}$$

$$\wedge \forall \tilde{a}_1 \forall \tilde{f}_1 \Gamma(\tilde{a}_1, \tilde{f}_1) \wedge \text{EqUntil}(\tilde{a}_1, \tilde{a}, \text{now})$$

$$\rightarrow \exists \tilde{a}_2 \exists \tilde{f}_2 \Gamma(\tilde{a}_2, \tilde{f}_2) \wedge \text{EqUntil}(\tilde{a}_2, \tilde{a}_1, \text{now})$$

$$\wedge \text{EqUntil}(\tilde{f}_2, \tilde{f}_1, \text{now}) \wedge \text{EqNext}(\tilde{a}_2, \tilde{a}, \text{now})$$

i.e. it exists a further interpretation in accord with the actions and the fluents of another one till now, in which the action performed next is the one selected by $\tilde{a}$. 
Legality for an infinite sequence of actions can be defined as:

$$Legal(\Gamma, \vec{a}) \equiv \exists \vec{f} \Gamma(\vec{a}, \vec{f}) \land \forall \text{now} \ LegalNext(\Gamma, \vec{a}, \text{now})$$

i.e. the sequence of actions resulting from the interpretation of $\vec{a}$ is consistent with $\Gamma$ and every prefix of such a sequence continues next with a legal action.
An infinite sequence of actions $a_0, a_1 \cdots$ is legal wrt an LTL system specification $\Gamma$ iff there exists an interpretation $\pi$ interpreting the actions $\bar{a}$ according to $a_0, a_1 \cdots$ and such that $\pi \models Legal(\Gamma, \bar{a})$.

Checking the existence of an infinite sequence of actions that is legal wrt an LTL system specification $\Gamma$ can be done in 2-EXPSPACE.

- it is a 2-EXPSPACE-hard problem.
Planning
General considerations

- Dynamic properties of LTL, can be verified writing the system specification and used as temporally extended goals to synthesize plans.
- For example:
  - Reachability of the desire state of affairs, is a situation where a given goal $\varphi_{\text{goal}}$ holds reachable? $\Diamond \varphi_{\text{goal}}$.
  - Archiving and maintenance of goals, is it possible to archive a goal $\varphi_{\text{goal}}$ while maintaining another goal $\varphi'_{\text{goal}}$? $\varphi'_{\text{goal}} \cup \varphi_{\text{goal}}$
- Goals can be more sophisticated and similarly, safety, invariance, liveness and fairness properties can be expressed in LTL.
Conformant planning, is the way in the setting for specify by arbitrary formulas of LTL both the Dynamic system and the goal. The problem consist in constructing a plan.

$$plan(\Gamma, \gamma, \bar{a}) \equiv \forall \bar{f}. \Gamma(\bar{a}, \bar{f}) \rightarrow \gamma(\bar{a}, \bar{f})$$

A plan can be infinite but still finitely representable.
To check the existence of a legal plan, we simply need to check the satisfiability of

$$\text{plan}(\Gamma, \gamma, \hat{a}) \land \text{valid}(\Gamma, \hat{a})$$

This can be done building a Büchi automaton and checking the nonemptiness.

*Theorem* [3]: Verifying the validity of a plan in LTL is 2-EXPSPACE-complete.
Conclusions

Those results can be extended to other LTL extensions such as $\mu$ LTL that is a translation of LTL to Buchi automata that is able to represent any property like safety, invariance, liveness and fairness.

Adopting the techniques discussed in this paper, it can be shown:

- that the reasoning on action effects remains PSPACE-complete,
- synthesising non-necessarily legal plans remains EXPSPACE-complete,
- while testing legality becomes EXPSPACE-complete.
Conclusions

Considerations

- Complexity is probably going to increase, by moving to the branching-time setting.
- One should stress that for system specifications of a special form, checking legality of sequences of actions may become much easier.
Finally algorithms for checking nonemptiness of Buchi automata, which are at the base of reasoning procedures for LTL and QLTL, have proved to be well suited for scaling up to very large systems.