Partial model checking and partial model synthesis in LTL using a tableau-based approach

Serenella Cerrito ORCID
Université Paris Saclay, Univ EVRY, France

Valentin Goranko ORCID
Stockholm University, Sweden

Sophie Paillocher, ORCID
Université Paris Saclay Univ EVRY, France

Abstract

In the process of designing a computer system S and checking whether an abstract model $\mathscr{M}$ of S verifies a given specification property η, one might have only a partial knowledge of the model, either because $\mathscr{M}$ has not yet been completely defined (constructed) by the designer, or because it is not completely observable by the verifier. This leads to new verification problems, subsuming satisfiability and model checking as special cases. We state and discuss these problems in the case of LTL specifications, and develop a uniform tableau-based approach for their solutions.

Keywords: Linear temporal logic LTL, partial transition systems, partial model checking, partial model synthesis, tableaux.

1 Introduction

In the process of designing a computer system S and checking whether an abstract transition system $\mathscr{M}$ modelling S satisfies a given specification property η, one might have only a partial knowledge of the model, either because $\mathscr{M}$ has not yet been completely defined (constructed) by the designer, or because it is not completely observable by the verifier. Typically, $\mathscr{M}$ is a finite transition system with states labeled by sets of atomic propositions (Boolean variables) with their truth values, and the partial construction or partial observability may be manifested in one or more of the following components: a) the truth values of some propositions at some states; b) whether two given states are connected by a transition; c) whether the observed states are all the existing or intended states, or there are more, not observed or not yet constructed. In a multi-agent context there are additional components, e.g., some agents and/or their actions might also be partly observable or unspecified yet.

Now two natural questions arise:

i) *whether there is at least some way to extend (eventually, complete) the partially constructed transition system in such a way that the property η holds.*

ii) *whether what is already known suffices to verify η, no matter how the partial information might be extended (eventually, completed).*

(Alternatively, one may think that in both cases the model is partially constructed, but in the former case it is up to the verifier to complete the construction, whereas in the latter case it will be completed by another, possibly adversarial, agent.)

These questions have been stated respectively as *partial model synthesis* and *partial model checking* and discussed in general in [15], on which the present work follows up.

In this paper we take the property η to be expressed by an LTL formula and the model $\mathscr{M}$ to be a transition system (Kripke structure), which is incomplete in either of the senses a, b and c described above. Then the two questions above unfold into four decision problems and two associated synthesis problems, precisely described in Section 3. The aim of this work is not only to provide a unified approach for solving the decision problems, but also to provide

constructive methods for their solutions, thus also solving the associated synthesis problems. Thus, when considering, for instance, whether there exists an extension of the partially given model $\mathscr{M}$ assuring the existence of a path verifying η, we are not only interested in providing a correct YES/NO answer, but, in the case where the answer is positive, we want *to build* an extension $\mathscr{M}^c$ of the input $\mathscr{M}$, and a path in it that is a linear model of η.

Furthermore, when extending the model with truth values of Boolean variables at states, we aim to do it in a "minimal" (or, "most general") way, so as to leave to the designer as much freedom as possible for further extensions. That is, if a given Boolean variable p has an unspecified (or, unknown) truth value at a given state s on a path that is to satisfy η, we only force that value to be defined (True or False) when this is really necessary to produce such a path. For these reasons, *inter alia*, our methods and algorithms are inspired by the *tableau methods* for constructively solving the satisfiability problem in LTL, first developed in [23] and further optimised and presented in detail in [12, Chapter 13], that are usually proposed as the most constructive methods for solving the satisfiability problem. Here we will follow the style and technical details of the tableaux construction procedure in the latter.

The decision problems described in Section 3 informally ask respectively whether some (resp., every) admissible extension of the partial model is such that some (resp., every) path in it starting from the given initial state satisfies η. These problems are denoted respectively EE, AA, EA, AE. In this work we first present in full details the solution to EE (and of its dual problem AA) in the case where only the state labels (given by the truth values of the Boolean variables) may be not yet determined or unknown, but the states themselves and the transitions between them are completely specified and observed. Then we show how the proposed algorithm can be extended to deal also with the cases where transitions and/or states may be not yet determined or unknown, partially or completely (the limit case being when nothing is known at all). After that, we show how the proposed methods for solving EE and AA also enable handling of the problems EA and AE. Thus, the algorithm that we initially propose for solving EE is the *core* method of our approach, to which all the other variants of model extension or completion problems are reduced.

Some of the problems that we address here are related to previously published work, esp. on partial model checking of LTL and other logics. For instance, there is a clear connection between our work and [10, 14]. We postpone the discussion of related works to Section 7.2.

The paper is organized as follows. In section 2 we briefly recall some standard notions on LTL and we set some preliminary definitions and notation. In Section 3 we state the main problems. In Section 4 we describe our core algorithm and in Section 5 we present basic results about it. In Section 6 we show how it can be used to solve the other problems we consider. Section 7 discusses related works, points at some perspectives, and concludes. In a technical appendix we include several figures representing different stages of the tableau construction for a running example, plus some additional references on related work.

2 Preliminaries

2.1 Partial transition systems, extensions, and completions

In what follows, Prop is a nonempty finite set of atomic propositions (Boolean variables) and $B = \{\top, ?, \bot\}$ is the set of truth-values, where ? is a third Boolean value intuitively meaning "undetermined" or "unknown". The definition below extends the well-known notion of transition system (aka Kripke structure) to the case of partial construction or incomplete knowledge. It combines some ideas that come, *inter alia*, from [17] (for transitions) and [14] (for three-valued state labels).

▶ **Definition 1** (Partial and complete transition systems). *A **Partial transition system** (also called further a **partial model**) is a tuple $\mathscr{M} = (S, R^{must}, R^{may}, L)$, where: S is a finite set of states, $R^{must} \subseteq S \times S$ and $R^{may} \subseteq S \times S$ are two transition relations such that $R^{must} \subseteq R^{may}$, and $L : S \times \mathrm{Prop} \to B$ is an **interpretation function** associating with each state in S and each atomic proposition in* Prop *a truth value in B. The relation R^{must} corresponds to the already determined, or known, transitions, whereas R^{may} also includes the possible but not yet determined, or not yet known transitions.*

We assume that R^{may} is serial (aka, total); hence, every path in (S, R^{may}, L) is infinite.

*If $R^{may} = R^{must} = R$, we denote the partial transition system simply as (S, R, L) and say that it is **transition-complete**. A transition-complete structure (S, R, L) is **complete**, or just a **transition system**, if all values that L assigns are in $\{\top, \bot\}$. Thus, every complete transition system is also a partial transition system.*

▶ **Definition 2** (Extension and completion of an interpretation function). *Given a partial transition system $\mathscr{M} = (S, R^{must}, R^{may}, L)$, an interpretation function $L' : S \times \mathrm{Prop} \to B$ is said to be an **extension** of L, denoted by $L \preceq L'$, if for each $s \in S$ and $p \in$ Prop, if $L(s,p) \neq ?$ then $L'(s,p) = L(s,p)$.*

*When $L' : S \times \mathrm{Prop} \to \{\top, \bot\}$ (i.e., all values that L' assigns are in $\{\top, \bot\}$), then L' is said to be a **completion**, denoted $L \preceq^c L'$.*

▶ **Definition 3** (Extension and completion of a partial transition system). *A partial transition system $\mathscr{M}' = (S', R'^{must}, R'^{may}, L')$ is an **extension of a partial transition system** $\mathscr{M} = (S, R^{must}, R^{may}, L)$, denoted by $\mathscr{M} \preceq \mathscr{M}'$ if: $S \subseteq S'$, $R^{must} \subseteq R'^{must}$, $R'^{may} \subseteq R^{may}$, and $L \preceq L'$.*

*If R'^{must} is serial, then $\mathscr{M}'$ is a **total extension** of $\mathscr{M}$. A complete transition system $\mathscr{M}' = (S', R', L')$ which is a total extension of $\mathscr{M}'$ is a **completion** of $\mathscr{M}$, denoted by $\mathscr{M} \preceq^c \mathscr{M}'$. Thus, $\mathscr{M} \preceq^c \mathscr{M}'$ iff: $S \subseteq S'$, $R^{must} \subseteq R' \subseteq R^{may}$, R' is serial, and $L \preceq^c L'$.*

2.2 The logic LTL

Here we only fix basic notation and terminology on the Linear Temporal Logic LTL used further in the paper. For further details the reader is referred e.g., to [12].

The formulae of LTL are given by the grammar

$$\varphi, \psi := \top \mid \bot \mid p \mid \neg\varphi \mid (\varphi \wedge \psi) \mid \mathsf{X}\,\varphi \mid \mathsf{G}\,\varphi \mid (\psi\,\mathsf{U}\,\varphi)$$

while $\vee$, $\to$, $\leftrightarrow$, and F are defined in the standard way.

A **linear model** for LTL is an infinite sequence $\sigma : \mathbb{N} \to \mathscr{P}(\mathrm{Prop})$. **Truth (satisfaction) of an LTL formula φ at a position $i \in \mathbb{N}$ of a linear model σ**, denoted $\sigma, i \models \varphi$, is defined as usual. An LTL formula φ is said to be **true in a linear model σ**, denoted $\sigma \models \varphi$, if $\sigma, 0 \models \varphi$; in this case σ is said to be a **linear model of φ**.

LTL formulae are also interpreted at states in transition systems, as follows. Given a transition system $\mathscr{M} = (S, R, L)$ the sequence $\pi = s_0, s_1, s_2, \ldots$ is a **path in $\mathscr{M}$** if $s_0 R s_1 R s_2, \ldots$. The **trace (or, computation) in $\mathscr{M}$ corresponding to** π is the sequence of the sets of true propositions along π: $T(s_0), T(s_1), T(s_2) \ldots, \ldots$, where $T(i) = \{p \in \mathrm{Prop} \mid L(s_i) = \top\}$. Given a state $s \in S$, we denote by $traces_{\mathscr{M}}(s)$ the set of traces in $\mathscr{M}$ with initial state s. Clearly, every trace in $\mathscr{M}$ can be regarded as a linear LTL model, hence the notions $\sigma, i \models \varphi$ and $\sigma \models \varphi$ are readily defined for any trace σ. We say that an LTL formula φ is **universally true** in a transition system $\mathscr{M}$ at a state s if $\sigma \models \varphi$ for all elements σ of $traces_{\mathscr{M}}(s)$. In that case, we also simply say that φ **is true in $\mathscr{M}$ at** s, denoted $\mathscr{M}, s \models_\forall \varphi$

or just $\mathscr{M}, s \models \varphi$. Likewise, we say that φ is **existentially true in** $\mathscr{M}$ **at** s if for some trace $\sigma \in traces_{\mathscr{M}}(s)$ we have $\sigma \models \varphi$. In that case we write $\mathscr{M}, s \models_{\exists} \varphi$.

More notation: given a finite set of formulas $\Gamma = \{\varphi_1, \varphi_2, \ldots, \varphi_n\}$, we write:

$$\bigvee \Gamma := \varphi_1 \vee \varphi_2 \vee \cdots \vee \varphi_n, \qquad \bigwedge \Gamma := \varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_n.$$

We will write $\mathscr{M}, s \models \Gamma$ as a shorthand for $\mathscr{M}, s \models \bigwedge \Gamma$

2.3 Tableau-related preliminaries

The definitions here are borrowed from the presentation of tableaux for LTL in [12, Ch.13.2].

We distinguish four types of formulas : *literals (atoms or negated atoms), conjunctive, disjunctive and successor formulas.* A special type of formulas in LTL are the *eventualities*, i.e. formulas of the type $\mathsf{F}\,\varphi$ or $\psi\,\mathsf{U}\,\varphi$. In both cases, their truth at a time point implies that φ *must be realised at some point in the future*, but is realisation can be procrastinated. So an eventuality can be see as a disjunction, as it is shown Figure 1.

▶ **Definition 4** (Components of a formula). *The* ***components of a formula*** φ *are defined depending on its type. The components of a conjunctive (resp. disjunctive) formula* φ *are formulas, defined further, such that* φ *is equivalent to their conjunction (resp. disjunction). A successor formula* ψ *has only one component, written* $scomp(\psi)$*. Literals do not have components. All types and components of LTL-formulas are summarised in figure 1.*

conjunctive		disjunctive		successor	
formula	components	formula	components	formula	components
$\neg\neg\varphi$	$\varphi,\ \varphi$	$\varphi \vee \psi$	$\varphi,\ \psi$	$\mathsf{X}\,\varphi$	φ
$\varphi \wedge \psi$	$\varphi,\ \psi$	$\varphi \rightarrow \psi$	$\neg\varphi,\ \psi$	$\neg\mathsf{X}\,\varphi$	$\neg\varphi$
$\neg(\varphi \vee \psi)$	$\neg\varphi,\ \neg\psi$	$\neg(\varphi \wedge \psi)$	$\neg\varphi,\ \neg\psi$		
$\neg(\varphi \rightarrow \psi)$	$\varphi,\ \neg\psi$	$\mathsf{F}\,\varphi$	$\varphi,\ \mathsf{X}\,\mathsf{F}\,\varphi$		
$\mathsf{G}\,\varphi$	$\varphi,\ \mathsf{X}\,\mathsf{G}\,\varphi$	$\psi\,\mathsf{U}\,\varphi$	$\varphi,\ \psi \wedge \mathsf{X}\,\psi\,\mathsf{U}\,\varphi$		
$\neg\mathsf{F}\,\varphi$	$\neg\varphi,\ \mathsf{X}\,\neg\mathsf{F}\,\varphi$	$\neg\mathsf{G}\,\varphi$	$\neg\varphi,\ \mathsf{X}\,\neg\mathsf{G}\,\varphi$		
$\neg(\psi\,\mathsf{U}\,\varphi)$	$\neg\psi \vee \mathsf{X}\,\neg(\psi\,\mathsf{U}\,\varphi),\ \neg\varphi$				

Figure 1 Types and components of formulas in LTL

▶ **Definition 5** (Extended closure of a formula). *The extended closure* $\mathsf{ecl}(\varphi)$ *of the formula* φ *is the least set of formulas such that :*

1. $\varphi \in \mathsf{ecl}(\varphi)$,
2. $\mathsf{ecl}(\varphi)$ *is closed under taking all conjunctive, disjunctive, successor components of the respective formulas in* $\mathsf{ecl}(\varphi)$.

▶ **Definition 6** (Extended closure of a set of formulas). *For any set of formulas* Γ *we define*

$$\mathsf{ecl}(\Gamma) := \bigcup\{\mathsf{ecl}(\varphi) | \varphi \in \Gamma\}.$$

A set of formulas Γ *is* ***closed*** *if* $\Gamma = \mathsf{ecl}(\Gamma)$.

▶ **Definition 7** (Patently inconsistent). *A set of formulas is* ***patently inconsistent*** *if it contains* $\bot$*, or* $\neg\top$*, or a contradictory pair of formula* φ *and* $\neg\varphi$.

▶ **Definition 8** (Fully expanded). *A set of formulas* Γ *is* ***fully expanded*** *if and only if*

1. *it is not patently inconsistent,*

2. *for every conjunctive formula in* Γ*, all of its conjunctive components are in* Γ*,*
3. *for every disjunctive formula in* Γ*, at least one of its disjunctive components is in* Γ*.*

Note that the empty set of formulas is vacuously fully expanded.

▶ **Definition 9** (Full expansion of a set of formulas)**.** *A fully expanded set of formulas* Δ *is a* ***full expansion*** *of a set of formulas* Γ *if* Δ *can be obtained from* Γ *by repeated applications of the following rules, where initially no formula is marked as "used":*

- ***(C-comp)*** *for every conjunctive formula* φ *in the current set* Γ *that has not been marked as "used", add all of its conjunctive components to* Γ *and mark* φ *as "used".*
- ***(D-comp)*** *for every disjunctive formula* φ *in the current set* Γ *that has not been marked as "used", add one of its disjunctive components to* Γ *and mark* φ *as "used".*

Note that the rule (D-comp) is non-deterministic, so a set Γ may have several (or no) full expansions. The notation $FE(\Delta)$ means the set of the full expansions of a set of formulas Δ.

Below we also recall some definitions and a theorem about Hintikka traces, that can be found in [12], as we will make use of them in our approach.

▶ **Definition 10.** *Given a closed set of formulas* Γ*, a* ***Hintikka trace*** *(HT) for* Γ *is a mapping* $H : \mathbb{N} \to \mathrm{Prop}(\Gamma)$ *satisfying the following conditions for every* $n \in \mathbb{N}$*:*

1. $H(n)$ *is fully expanded.*
2. *If* $\varphi \in H(n)$ *is a successor formula, then* $scomp(\varphi) \in H(n+1)$*.*
3. *If* $\varphi \,\mathsf{U}\, \psi \in H(n)$*, then there exists* $i \geq n$ *such that* $\psi \in H(n+i)$ *and* $\varphi \in H(n+j)$*for every* j *such that* $0 \leq j < i$*.*

An LTL *formula* φ *is* ***satisfiable in a Hintikka trace*** H *if* $\varphi \in H(n)$ *for some* $n \in \mathbb{N}$*.*

▶ **Theorem 11.** *An LTL formula* η *is satisfiable iff it is satisfiable in some Hintikka trace.*

3 The problems we study

Let $\mathscr{M}$ be a partial transition system, let s_0 be a state in $\mathscr{M}$ and let η be an LTL formula. The following four decision problems naturally arise (cf. Definitions 2 and 3):

- **Existential extension for path existence**
  Are there a total extension $\mathscr{M}^c$ of $\mathscr{M}$ and a path $s_0, s_1, s_2, \ldots$ in $\mathscr{M}^c$ such that its corresponding trace σ is a linear model of η? We denote this decision problem by EE.
- **Existential extension for all paths**
  Is there a total extension $\mathscr{M}^c$ of $\mathscr{M}$ such that for all paths $s_0, s_1, s_2, \ldots$ in $\mathscr{M}^c$ the corresponding traces are linear models of η? We denote this decision problem by EA.
- **Universal extension for path existence**
  Is it true that each total extension $\mathscr{M}^c$ of $\mathscr{M}$ admits some path $s_0, s_1, s_2, \ldots$ such that its corresponding trace is a linear model of η? We denote this decision problem by AE
- **Universal extension for all paths**
  Is it true that each total extension $\mathscr{M}^c$ of $\mathscr{M}$ is such that for each of its paths $s_0, s_1, s_2, \ldots$ the corresponding traces are linear models of η? We denote this decision problem by AA

Clearly, the problems EE and AA are dual: a positive answer to EE with input $(\mathscr{M}, s_0, \eta)$ is a negative answer for EE on input $(\mathscr{M}, s_0, \neg\eta)$, while a negative answer for EE on input $(\mathscr{M}, s_0, \eta)$ is a positive answer for AA on input $(\mathscr{M}, s_0, \neg\eta)$. Likewise, EA and AE are dual.

It is worthwhile observing that when $\mathscr{M}$ is completely unknown then EE coincides with EA and amounts to the LTL satisfiability problem, while each of the problems AA and AE is nothing but the LTL validity problem. On the other hand, when $\mathscr{M}$ is completely known,

both EE and AE turn out to be the existential model checking problem, while EA and AA coincide with the universal model checking problem.

As mentioned in the introduction, we consider three types of possible extensions and completions of partial transition systems. Each case is a special case of an extension in terms of Definition 3, as follows: (i) **label extension**, where the states and transitions are fixed, but the labels of the states are extended by an extension of the interpretation function (cf. Definition 2); (ii) **transition extension**, where the states (and their labels) are fixed, but new transitions are added; (iii) **state extension**, where new states, with partial or complete labels, as well as transitions to and from them can be added. Clearly, (i) and (ii) can be combined, whereas (iii) subsumes (ii) and can naturally subsume (i), too.

Each of these decision problems is easily seen to be PSPACE-complete. In the cases of label extensions and transition extensions, let us non-deterministically choose a completion $\mathscr{M}^c$ and existentially - respectively universally - model-check η on $\mathscr{M}^c$. Since the decision problems for existential and universal model checking of LTL formulae are PSPACE complete ([12]), EE and EA are NPSPACE-easy. On the other hand, they are also NPSPACE-hard, because existential - respectively universal - model checking problems are trivially polynomially reducible to them (as they are the special cases where $\mathscr{M}$ is already complete). Hence they are NPSPACE-complete, so, by Savitch theorem, they are PSPACE-complete. Consequently, their dual problems AE and AA, are also PSPACE-complete. In the case of state extensions, as shown in Section 6.3, the EE problem is easily reducible to the satisfiability of a formula produced from η and the label of the initial state, AA is again its dual, and the problems AE and EA are reducible to repeatedly solving EE, hence they are PSPACE-complete, too.

There are variants of two of the previous problems that are not decision problems but rather *model synthesis problems*. The variant of EE where, on the same input, one does not ask for a YES/NO answer, but rather for an output consisting of a total extension $\mathscr{M}^c$ and a path $\pi = s_0, s_1, s_2, ...$ in $\mathscr{M}^c$ such that its corresponding trace σ is a linear model of η, when these exists (and for a failure message otherwise) is the synthesis problem that we will denote by EE^S. Similarly, the variant of EA where one asks for an extension $\mathscr{M}^c$ where η is universally true is the synthesis problem denoted by EA^S.

4 Tableau-based algorithms for EE and EE^S for label extensions

Here we consider the case of partial transition systems $\mathscr{M} = (S, R^{must}, R^{may}, L)$, where $R^{must} = R^{may} = R$ and S is fixed, and focus on the case of label extensions.

The core algorithm that we propose solves both problems EE and EE^S (modulo some technical variations) as follows. Given as an input a partial transition system $\mathscr{M}$, a distinguished state $s_0 \in \mathscr{M}$ and an LTL formula η, the algorithm enables the computation of *all* the total extensions[1] $\mathscr{M}^c$ where η is true on some path starting from s_0, in a sense explained further.

4.1 The Core Tableau-based Algorithm

We describe the algorithm step-by-step, with the help of a very simple running example.

▶ **Example 12** (Running Example)**.** We have a partial model of an automatic subway and we want to know if it can be extended/completed according to the specification "*the doors*

[1] To be precise, the algorithm itself does not produce the set of all such extensions – including completions of the input model – as it may leave undetermined the truth values of some atoms at states of paths that are not relevant for the existence of a path satisfying η. But, all the extensions of the input model can be trivially computed from the set of the extensions directly built via the tableau based algorithm.

cannot be open while the train is running, and the train will eventually run". For the sake of simplicity, we assume that the partial model consists of only two states and we suppose that the doors are open at the initial state s_0. This partial model is represented in Figure 2. Formally the question is "*Is there an extension $\mathscr{M}'$ of the partial model $\mathscr{M}$ and a trace σ in traces(s_0) such that $\sigma, s_0 \models \eta_0$?*", where $\eta_0 := \mathsf{F}\, r \wedge \mathsf{G}\,(r \rightarrow \neg d)$, with d interpreted as "*the doors are open*" and r as "*the train is running*".

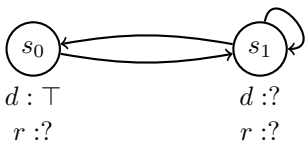

Figure 2 A partial model for Example 12.

The approach that we use to solve this problem is based on the tableau methods for LTL satisfiability, also modified for LTL model-checking, presented in [12, Section 13.2]. Here is an outline of our procedure. First, we construct a tableau, adapted so as to take into account the partially constructed or partially observable input model. Then we use it to:

- conclude that the answer to the decision problem EE is YES, when the tableau is "open" (in a sense that will be specified further), else it is NO;
- in the case of YES, to extract an extension $\mathscr{M}^c$ of $\mathscr{M}$ and a path π in it, starting from s_0, such that the corresponding trace in $\mathscr{M}^c$ satisfies η.

4.1.1 Pretableau Construction Phase

Given a partial model $\mathscr{M} = (S, R, L)$ (i.e., $R = R^{must} = R^{may}$) a state $s_0 \in S$ and an LTL formula η, we construct a graph called **pretableau of** $(\mathscr{M}, s_0, \eta)$ and denoted $\mathscr{P}^{\mathscr{M},s_0,\eta}$. Its nodes are triples consisting of: a state in $\mathscr{M}$, a set of formulas that must be true at that state, and an *annotation function*[2]. There are two types of nodes in the pretableau:

- **tableau states**[3],where the set of formulas is fully expanded;
- **prestates**, that only play an auxiliary and temporary role.

The initialisation of the algorithm produces an initial tableau prestate. Then, tableau states and prestates are built alternatively, in an iterative way described below.

▶ **Notation 13.** *Let Γ be a set of* LTL *formulas, $\mathscr{M} = (S, R, L)$ be a partial model, and let $s \in S$. We denote* Litt(s, Γ, L) *the set of literals (atomic propositions and their negations) appearing in Γ whose truth value at s according to the input interpretation function L is $\top$. Formally:* Litt$(s, \Gamma, L) := \{p | p \in \Gamma, L(s, p) = \top\} \cup \{\neg p | p \in \Gamma, L(s, p) = \bot\}$.

4.1.2 Initialisation of the algorithm

The pretableau construction begins with the creation of the prestate $(s_0, \Gamma_0, A_{\Gamma_0})$ where $\Gamma_0 := \{\eta\} \cup$ Litt$(s_0, ecl(\eta), L_0)$ and A_{Γ_0}, the initial annotation function, coincides with L.

▶ **Example 14** (Running Example continued)**.** In Example 12, the input partial model is such that $L(s_0, d) = \top$ and $L(s_0, r) =?$, so Litt$(s_0, ecl(\eta_0), L) = \{d\}$ and $\Gamma_0 := \{d, \eta_0\}$. Note that the literal r does not appear in Γ_0, as its truth value is undetermined/unknown. The construction of the tableau begins with the creation of the prestate node $(s_0, \Gamma_0, A_{\Gamma_0})$.

---

[2] An annotation function $S \times P \rightarrow B$ should not be confused with the interpretation function L given in the input model; it is used in a tableau node to describe and extend, incrementally, the interpretation function L given in the input model.

[3] Not to be confused with states in a partial transition system.

Note also that any propositional letter $q \in PROP \cap ecl(\eta_0)$ such that neither q nor $\neg q$ is in Γ_0 is either such that $L(s_0, q) =?$ or it does not appear in η_0, in which case neither of q and $\neg q$ is in $ecl(\eta)$, so its truth value does not impact the satisfaction of η_0.

4.1.3 The body of the algorithm

Two rules will be alternatively applied:

- Exp: producing so called **offspring states** of a given prestate,
- Next: producing **successor prestates** of a given state.

Each rule may be applied at most once to each node, to ensure termination.

The rule Exp, defined below, statically analyzes the formulas in a tableau prestate, following their semantics, and generates tableau states.

▶ **Definition 15** (Rule Exp). *Given a prestate (r, Γ, A_Γ), do the following:*

1. *Compute the family $FE(\Gamma)$ of all full expansions of Γ in the sense of Definition 9.*
2. *For each $\Delta \in FE(\Gamma)$, define the new annotation function A_Δ as*

$$A_\Delta(s,p) = \begin{cases} L_\Gamma(s,p) & \text{if } L_\Gamma(s,p) \neq ? \\ \top & \text{if } L_\Gamma(s,p) =? \text{ and } p \in \Delta \\ \bot & \text{if } L_\Gamma(s,p) =? \text{ and } \neg p \in \Delta \\ ? & \text{if } L_\Gamma(s,p) =? \text{ and } p, \neg p \notin \Delta \end{cases}$$

   and add in the current pretableau a new offspring state (s, Δ, A_Δ).
3. *For each newly introduced state (s, Δ, A_Δ), create an edge $(s, \Gamma, A_\Gamma) \dashrightarrow (s, \Delta, A_\Delta)$.*
4. *If the pretableau already contains a state (s, Δ, A_Δ), then do not create a new state but create only an edge $(s, \Gamma, A_\Gamma) \dashrightarrow (s, \Delta, A_\Delta)$ to that state.*

▶ **Notation 16.** *The set $\{(s, \Delta, A_\Delta) | (s, \Gamma, A_\Gamma) \dashrightarrow (s, \Delta, A_\Delta)\}$ of offspring states of (s, Γ, A_Γ) is denoted by $\mathsf{states}(s, \Gamma, A_\Gamma)$;*

▶ **Example 17** (Running example continued). We apply the rule Exp to the prestate $(s_0, \Gamma_0, A_{\Gamma_0})$ obtained in Example 14. The set of the full expansions of Γ_0 contains only Δ_0 where:

$$\Delta_0 := \{d, \neg r, \eta_0, \mathsf{G}(r \to \neg d), \mathsf{F}\, r, r \to \neg d, \mathsf{X}\,\mathsf{F}\, r, \mathsf{X}\,\mathsf{G}(r \to \neg d)\}.$$

Note that $\neg r \in \Delta_0$ while $L(s_0, r) =?$. The new annotation function A_{Δ_0} is defined by:

- $A_{\Delta_0}(s_0, d) = \top$ and $A_{\Delta_0}(s_0, r) = \bot$;
- $A_{\Delta_0}(s_1, d) = A_{\Delta_0}(s_1, r) =?$.

This is the beginning of the construction of a completion of the labels of $\mathscr{M}$.
The currently constructed pretableau is given in the Appendix, Figure 3.

The rule Next creates in the tableau graph successors of any tableau state (s, Δ, A_Δ) corresponding to a state s in input model $\mathscr{M}$. These nodes are tableau prestates; they correspond to not yet fully analyzed descriptions of successors states of s in the model $\mathscr{M}$.

▶ **Definition 18** (Rule Next). *Given a state (s, Δ, A_Δ), do the following:*

- *If there are no R-successors of s in the partial model, remove the state (s, Δ, A_Δ) from the pretableau. Else:*
  - *For every R-successor state t of s in the current partial model, add in the current pretableau a successor prestate $(t, Scomp(\Delta) \cup \mathsf{Litt}(t, ecl(\eta), A_\Delta))$ of (s, Δ, A_Δ), where $Scomp(\Delta) = \{\psi | \mathsf{X}\,\psi \in \Delta\} \cup \{\neg\psi | \neg\mathsf{X}\,\psi \in \Delta\}$.*
  - *If the pretableau already contains a prestate (t, Γ, A_Γ) then do not create a new state but create only an edge $(s, \Delta, A_\Delta) \to (t, \Gamma, A_\Gamma)$ to that prestate.*

▶ **Example 19** (Running example continued)**.** We apply the rule Next to the offspring state $(s_0, \Delta_0, A_{\Delta_0})$ obtained in the example 17. There is only one R-successor of s_0 that is s_1 in our partial model. So, the rule Next generate the only prestate $(s_1, \Gamma_1, A_{\Gamma_1})$, where $\Gamma_1 := \{\mathsf{F}\, r, \mathsf{G}\,(r \to \neg d)\}$. The current pretableau is given in Figure 4 of the Appendix.

▶ **Lemma 20.** *The pretableau construction phase terminates.*

Recall that any node in a pretableau is a triple $\langle s, \Sigma, A\rangle$ where s belongs to the finite set of states S, there are only finitely many annotation functions A (since both S and the set P ere finite) and $\Sigma \subseteq \mathsf{ecl}(\eta)$, where $\mathsf{ecl}(\eta)$ is finite (its cardinality is a polynomial of the size of η). Now, it suffices to note that both the rules Exp and Next avoid duplication of already existing nodes.

▶ **Example 21** (Running example continued)**.** At the end of the construction phase we obtain the pretableau shown in the Appendix in the Figure 5, where prestates are indicated with rectangular boxes and states with oval boxes. Nodes of the pretableau are there enumerated to help reading the whole picture, but we leave to the interested reader the job of spelling out the content of each node.

4.1.4 Prestate and State Elimination Phase

First, we remove all prestates from the pretableau with their incoming and outgoing edges, by applying the following **prestate elimination rule**:

▶ **Definition 22** (Rule PrestateElim)**.** *For every prestate (s, Γ, A_Γ) in the pretableau, do:*

1. *Remove (s, Γ, A_Γ) from the pretableau;*
2. *If there is a state (t, Δ, A_Δ) in the pretableau with $(t, \Delta, A_\Delta) \to (s, \Gamma, A_\Gamma)$, then for every state $(s, \Delta', A'_\Delta) \in \mathsf{states}(s, \Gamma, A_\Gamma)$ create an edge $(t, \Delta, A_\Delta) \to (s, \Delta', A'_\Delta)$.*

*The resulting graph is called the **initial tableau** for $(\mathscr{M}, s_0, \eta)$, denoted by $\mathscr{T}_0^{\mathscr{M}, s_0, \eta}$.*

Once all prestates have been eliminated, we must remove all "wrong" states from the tableau (dead ends and roots of paths that *never realize an eventuality*, which is explained further) by applying the rules StateElim1 and StateElim2, formulated shortly. But first, let us introduce a new notation.

▶ **Notation 23.** *We denote $\mathscr{T}_n^{\mathscr{M}, s_0, \eta}$ the tableau obtained from the initial tableau after n applications of an elimination rule (StateElim1 or StateElim2).*

▶ **Definition 24** (Rule StateElim1)**.** *If a state (r, Δ, A_Δ) has no successor states in the current tableau, then remove (r, Δ, A_Δ) from the tableau.*

In order to formulate StateElim2 we need the following definition.

▶ **Definition 25** (Realization of an eventuality at a tableau state)**.** *An eventuality $\varphi\,\mathsf{U}\,\psi$ (resp. $\mathsf{F}\,\psi$) is realized at the state (s, Δ, A_Δ) in $\mathscr{T}_n^{\mathscr{M}, s_0, \eta}$ if there exists a finite path in $\mathscr{T}_n^{\mathscr{M}, s_0, \eta}$*

$$\Pi = (s, \Delta, A_\Delta), (s_{i_1}, \Delta_{j_1}, A_{\Delta_{j_1}}), \ldots, (s_{i_m}, \Delta_{j_m}, A_{\Delta_{j_m}})$$

where $m \geq 0$, such that

- *$\varphi\,\mathsf{U}\,\psi, \psi \in \Delta_{j_m}$ (resp. $\mathsf{F}\,\psi, \psi \in \Delta_{j_m}$);*
- *$\varphi\,\mathsf{U}\,\psi, \varphi \in \Delta_{j_i}$ (resp. $\mathsf{F}\,\psi \in \Delta_{j_i}$) for every $i = 0, \ldots, m-1$.*

We say that $\varphi\,\mathsf{U}\,\psi$ (resp. $\mathsf{F}\,\psi$) is realized on the path Π, and we call any such path a witness of the realization of the eventuality $\varphi\,\mathsf{U}\,\psi$ (resp. $\mathsf{F}\,\psi$) in Δ. If $\psi \in \Delta$, then we say that $\varphi\,\mathsf{U}\,\psi$ (resp. $\mathsf{F}\,\psi$) is immediately realized at the state (s, Δ, A_Δ) (on the path constituted by the singleton (s, Δ, A_Δ)).

▶ **Definition 26** (Rule StateElim2)**.** *If an eventuality $\alpha \in \Delta$ is not realised on any path starting from (s, Δ, A_Δ) in the current tableau, then remove (s, Δ, A_Δ) from the tableau.*

The state elimination phase is carried out in a sequence of stages, starting at stage 0 with the initial tableau $\mathscr{T}_0^{\mathscr{M},s_0,\eta}$, and eliminating at every stage n at most one state for the current tableau $\mathscr{T}_n^{\mathscr{M},s_0,\eta}$ –by applying one of the state elimination rules– , to produce the new current tableau $\mathscr{T}_{n+1}^{\mathscr{M},s_0,\eta}$. When the state elimination phase reaches a stabilisation point, *i.e.* no more nodes are removed, the current tableau at that stage is called **final tableau** for $(\mathscr{M}, s_0, \eta)$ and denoted by $\mathscr{T}^{\mathscr{M},s_0,\eta}$.

▶ **Example 27** (Running example continued)**.** Let us consider again the initial tableau $\mathscr{T}_0^{\mathscr{M},s_0,\eta}$ shown in the Appendix, Figure 5, where only prestates have been eliminated. All the states have at least one successor state in $\mathscr{T}_0^{\mathscr{M},s_0,\eta}$, so the rule StateElim1 can not be applied. However there is a state with second component $\{d, \neg r, \mathsf{F}\, r, \mathsf{X}\,\mathsf{F}\, r, \mathsf{G}\,(r \to \neg d), r \to \neg d, \mathsf{X}\,\mathsf{G}\,(r \to \neg d)\}$, so it contains the eventuality $\mathsf{F}\, r$ that is not realized in the tableau. So, this state must be eliminated via Rule StateElim2. Thereafter, the rules StateElim1 and StateElim2 are alternatively applied, and the final result of the Elimination phase applied to the initial tableau on Figure 6 is shown in the Appendix in Figure 7.

▶ **Definition 28.** *The final tableau $\mathscr{T}^{\mathscr{M},s_0,\eta}$ is **open** if it contains at least one state $(s_0, \Delta_0, A_{\Delta_0})$ such that $\eta \in \Delta_0$. Otherwise it is **closed**.*

4.2 Specialized algorithms for EE and EE$^{\mathsf{S}}$

The tableau-based core algorithm that we have described in Section 4.1 can be refined and modified so as to obtain two algorithms, solving respectively the problem EE and the problem EE$^{\mathsf{S}}$. In both cases the input is given by a partial model $\mathscr{M} = (S, R, L)$, a distinguished state s_0 and an LTL formula η. However the outputs are different. The initial procedure is always the same: apply the core algorithm to construct a tableau $(T)^{\mathscr{M},s_0,\eta}$.

4.2.1 Algorithm for EE

Since the problem EE is a decision problem, the output here is either *YES* or *NO*. Once applied the core algorithm to construct a tableau $(T)^{\mathscr{M},s_0,\eta}$, the output is *YES* if and only if $(T)^{\mathscr{M},s_0,\eta}$ is open.

4.2.2 Algorithm for EE$^{\mathsf{S}}$

We give two versions of the algorithm. The first version is *non-deterministic*, and outputs just an extension of the input model $\mathscr{M}$ and a path in it (whenever some exist) that satisfies the specification formula η, while the second version (useful in Section 6) is *deterministic* and produces *all possible ways to extend* the input model so as to existentially satisfy η, and thus enables the generation of all such extensions, which we will use in Section 6.

Non-deterministic version ND-EE$^{\mathsf{S}}$. Here the output is either $\emptyset$ (when $\mathscr{T}^{\mathscr{M},s_0,\eta}$ is closed) or else a singleton containing a couple $(\mathscr{M}' = (S, R, L^c), \pi)$ where $\mathscr{M}'$ is an extension of the input model and π a path in it starting at s_0 and satisfying η.

1. Construct the tableau $\mathscr{T}^{\mathscr{M},s_0,\eta}$.
2. If the tableau is closed, then return the empty set. Else:
   a. Non-deterministically choose a path $(s_0, \Delta_{i_0}, A_{\Delta_{i_0}}), (s_{i_1}, \Delta_{i_1}, A_{\Delta_{i_1}}), \ldots$ in $\mathscr{T}^{\mathscr{M},s_0,\eta}$ that starts from a root of the tableau (note that it may have several roots) and such that the trace $\Delta_{i_0}, \Delta_{i_1}, \ldots$ is a Hintikka trace (as in Definition 10).

b. Define L^c as follows:
- if $A_{\Delta_{i_n}}(s,p) = \top$ for some node $(s_{i_n}, \Delta_{i_n}, A_{\Delta_{i_n}})$ on the chosen path, then $L^c(s,p) := \top$;
- else:
  – if $A_{\Delta_{i_n}}(s,p) = \bot$ for some node $(s_{i_n}, \Delta_{i_n}, A_{\Delta_{i_n}})$ on the chosen path then $L^c(s,p) := \bot$;
  – else set $L^c(s,p) := ?$.

c. Return $\mathscr{M}' = (S, R, L^c)$ and the path $s_0, s_{i_1}, s_{i_2}, \ldots$.

Deterministic version D-EE^S. Here the output is either the empty set (when $\mathscr{T}^{\mathscr{M},s_0,\eta}$ is closed) or else a set of couples $(\mathscr{M}', \sigma)$, such that $\mathscr{M}'$ is a total extension of $\mathscr{M}$ and σ is a path in it starting at s_0 and satisfying η. Intuitively, the algorithm can eventually produce all possible ways to extend the input model so as to existentially satisfy η.

1. Construct the tableau $\mathscr{T}^{\mathscr{M},s_0,\eta}$.
2. If the tableau is closed, then return the empty set.
3. Else, let $paths(\mathscr{T}^{\mathscr{M},s_0,\eta})$ be the set of all the tableau paths in $\mathscr{T}^{\mathscr{M},s_0,\eta}$ starting from a root node corresponding to a Hintikka trace :

$$paths(\mathscr{T}^{\mathscr{M},s_0,\eta}) := \{(s_0, \Delta_{i_0}, A_{\Delta_{i_0}}), (s_{i_1}, \Delta_{i_1}, A_{\Delta_{i_1}}), \ldots \text{ is a Hintikka trace}\}$$

For every tableau path π in $paths(\mathscr{T}^{\mathscr{M},s_0,\eta})$, define $\sigma_\pi = s_0, s_{i_1}, s_{i_2}, \ldots$ and the set $LC(\sigma_\pi)$ as the set of *all* the interpretation functions $L^c{}_{\sigma_\pi}$ such that:
- if $A_{\Delta_{i_1}}(s,p) = \top$ for some node $(s_{i_n}, \Delta_{i_n}, A_{\Delta_{i_n}})$ of π, then $L^c{}_{\sigma_\pi}(s,p) = \top$;
- if $A_{\Delta_{i_1}}(s,p) = \bot$ for some node $(s_{i_n}, \Delta_{i_n}, A_{\Delta_{i_n}})$ of π, then $L^c{}_{\sigma_\pi}(s,p) = \bot$.
- If $A_{\Delta_{i_1}}(s,p) = ?$ for every node $(s_{i_n}, \Delta_{i_n}, A_{\Delta_{i_n}})$ of π, then we have two possibilities: $L^c{}_{\sigma_\pi} = \top$ and $L^c{}_{\sigma_\pi} = \bot$, hence here the construction of $L^c{}_{\sigma_\pi}$ branches and two different interpretation functions are produced.

It is now easy to define a sub-procedure that generates, path by path, the elements of the set of all couples formed by a total extension $\mathscr{M}^c$ of the input partial model that is induced by an element of $paths(\mathscr{T}^{\mathscr{M},s_0,\eta})$ and a path in it, namely the set:

$$\{((S, R, L'), \sigma_\pi) \mid \pi \in paths(\mathscr{T}^{\mathscr{M},s_0,\eta}), L' \in LC(\sigma_\pi)\}.$$

Applying such a sub-procedure will eventually generate the full set described above.

▶ **Example 29** (End of the running example)**.** The final tableau for the running example of this section is the one in Figure 7 in the Appendix. This tableau is open, so the partial model $\mathscr{M}$ given in Example 12 can be completed so as to show that a path starting from s_0 and satisfying η exists. For instance, one can take the path $s_0, s_1, s_1, s_1, \ldots$ in the complete transition system (S, R, L') where $L' = A_{\Delta_{1.1}}$ is such that:

$L'(s_0, d) = \top$ $L'(s_0, r) = \bot$ $L'(s_1, d) = \bot$ $L'(s_1, r) = \top$

Indeed the tableau path $(s_0, \Delta_0, A_{\Delta_0}), (s_1, \Delta_{12}, A_{\Delta_{12}}), (s_1, \Delta_{24}, A_{\Delta_{24}}), (s_1, \Delta_{24}, A_{\Delta_{24}}), \ldots$ corresponds to the Hintikka trace $\Delta_0, \Delta_{12}, \Delta_{24}, \Delta_{24}, \ldots$ where
$\Delta_0 := \{d, \neg r, \eta, \mathsf{G}(r \to \neg d), \mathsf{F}\, r, r \to \neg d, \mathsf{X}\,\mathsf{F}\, r, \mathsf{X}\,\mathsf{G}(r \to \neg d)\}$,
$\Delta_{12} := \{\neg d, r, \mathsf{F}\, r\mathsf{G}(r \to \neg d), r \to \neg d, \mathsf{X}\,\mathsf{G}(r \to \neg d)\}$,
$\Delta_{24} := \{\neg d, r, \mathsf{G}(r \to \neg d), r \to \neg d, \mathsf{X}\,\mathsf{G}(r \to \neg d)\}$.

The following result is an immediate consequence of Lemma 20:

▶ **Theorem 30.** *The algorithms for* EE *and for* EE^S *(in both versions) terminate.*

5 Soundness and Completeness Results

▶ **Theorem 31** (Soundness with respect to satisfiability). *Let $\mathscr{M} = \{S, R, L\}$ be a partial model, let $s_0 \in S$ be an initial state and let η be an* LTL *formula. Let $\mathscr{T}^{\mathscr{M},s_0,\eta}$ be an open tableau. Then the following holds:*

1. *The problem* EE *has an answer YES.*
2. *Let $\pi = (s_0, \Delta_{j_0}, A_{j_0}), (s_{i_1}, \Delta_{j_1}, A_{j_1}), (s_{i_2}, \Delta_{j_2}, A_{j_2}), \ldots$ be a path in $\mathscr{T}^{\mathscr{M},s_0,\eta}$ such that $\Delta_{j_0}, \Delta_{j_1}, \Delta_{j_2}, \ldots$ is a Hintikka trace. Then for each total extension $\mathscr{M}^c = \{S, R, L'\}$ such that $A_{j_n} \preceq L'$ for each n, its trace corresponding to $s_0, s_{j_1}, s_{j_2}, \ldots$ satisfies η.*
   This holds, in particular, for each completion $\mathscr{M}^c$.

Sketch of the proof:

1. We construct inductively a path $\pi = (s_0, \Delta_{j_0}, A_{j_0}), (s_{i_1}, \Delta_{j_1}, A_{j_1}), (s_{i_2}, \Delta_{j_2}, A_{j_2}), \ldots$ in the final tableau such that $\Delta_{j_0}, \Delta_{j_1}, \Delta_{j_2}, \ldots$ is a Hintikka trace as follows. If Δ_{j_0} does not contain an eventuality, we choose any successor of $(s_0, \Delta_{j_0}, A_{j_0})$ in the tableau. Else, let $\alpha \in \Delta_{j_0}$ be an eventuality. The formula α is realized (on a path starting from $(s_0, \Delta_{j_0}, A_{j_0})$ as $(s_0, \Delta_{j_0}, A_{j_0})$ has been preserved in the elimination phase. The construction begins by extending $(s_0, \Delta_{j_0}, A_{j_0})$ via a finite path witnessing α. If some other eventuality β is not yet realized at the node $(s_{j_k}, \Delta_{j_k}, A_{j_k})$ ending this path, then by construction $\mathsf{X}\beta \in \Delta_{j_k}$ and β will still occur in the second components of all successors states. We repeat this construction until all pending eventualities are treated. This finite tableau path describes a finite path $s_0, s_{i_1}, s_{i_2}, \ldots$ in a completion $\mathscr{M}$ of $\mathscr{M}$ such that any infinite path extending it will satisfy η.

2. Now, we must prove that, under the given hypothesis, the path $s_0, s_{j_1}, s_{j_2}, \ldots$ satisfies η. This holds because the specification η is satisfied by the Hintikka trace $\Delta_{j_0}, \Delta_{j_1}, \Delta_{j_2}, \ldots$, hence η is satisfied by the linear model $\alpha : n \to P \cap \Delta_{j_n}$ that coincides with the trace in (S, R, L') corresponding to the path $s_0, s_{i_1}, s_{i_2}, \ldots$. □

▶ **Lemma 32.** *No state $(s, \Delta, A_\Delta) \in \mathscr{T}_n^{\mathscr{M},s_0,\eta}$ such that* EE $= YES$ *is removed by any application of the rules StateElim1 or StateElim2.*

The lemma is proved by an easy induction on the number of rounds on the elimination phase. The completeness result below shows not only that when EE is true thanks to some completion $\mathscr{M}^c$ then the corresponding tableau is open (Item 1), but also a stronger result (second item), which intuitively says that each trace in $\mathscr{M}^c$ satisfying η will be described by such a tableau. We make use of this stronger result in Section 6.

▶ **Theorem 33** (Completeness with respect to satisfiability). *Let $\mathscr{M} = \{S, R, L\}$ be a partial model, let $s_0 \in S$ and let η be an* LTL *formula. Let $\mathscr{M}^c = \{S, R, L'\}$ where $L \preceq L'$ be any total extension of $\mathscr{M}$ such that η is existentially true at s_0 in $\mathscr{M}^c$. Then the following hold:*

1. *The final tableau $\mathscr{T}^{\mathscr{M},s_0,\eta}$ is open.*
2. *Let $\sigma = s_0, s_{j_1}, s_{j_2}, \ldots$ be a path in $\mathscr{M}^c$ such that its corresponding trace σ satisfies η. Then any final tableau $\mathscr{T}^{\mathscr{M},s_0,\eta}$ contains a path $\pi = (s_0, \Delta_{j_0}, A_{j_0}), (s_{i_1}, \Delta_{j_1}, A_{j_1}), (s_{i_2}, \Delta_{j_2}, A_{j_2}), \ldots$ such that for each n we have $A_{j_n} \preceq L'$.*

Sketch of proof:

1. Follows by Lemma 32.
2. The path σ satisfies Γ_0 where $\Gamma_0 = \eta \cup \{p | L'(s_{i_0}, p) = \top\} \cup \{\neg p | L'(s_{i_0}, p) = \bot\}$. So there is a Δ_{j_0} in the set $FE(\Gamma_0)$ of the full expansions of Γ_0 such that $\sigma, 0 \models \Delta_{j_0}$. Then $L'(s_{i_0}, p) = \top$ implies that Δ_{j_0} doesn't contain $\neg p$, and $L'(s_{i_0}, p) = \bot$ implies that Δ_{j_0} does not contain p. By the tableau construction, there is a node $(s_0, \Delta_{j_0}, A_{\Delta_{j_0}})$ with $A_{\Delta_{j_0}} \preceq L'$ in the pretableau, and by Lemma 32 this node is still present in the final tableau. The required tableau path π is inductively constructed from this node by using the construction rules and the Lemma 32. □

6 Harvest: solving all problems

So far we have only considered the problems EE and EE^S in the case of label extensions and have developed methods for their solution. Here we will adapt our methods to solve all problems defined in Section 3 for all cases of extensions that we study. For lack of space, here we only outline the adapted procedures.

6.1 The case of label extensions

Since our approach solves EE for label extensions, it clearly also solves the corresponding dual problem AA with input $(\mathscr{M}, s_0, \eta)$, by calling the algorithm for EE on the input $(\mathscr{M}, s_0, \neg\eta)$.

Now, for EA and its dual problem AE, we proceed as follows.

1. First, we apply the algorithm for AA with the input $(\mathscr{M}, s_0, \eta)$. If the answer to AA is YES, then so are also the answers to EA and AE and we are done.
2. In the case the answer to AA is NO, we call the algorithm for EE on $(\mathscr{M}, s_0, \neg\eta)$.
   a. If the returned answer is NO, return YES for both EE and AE.
   b. If the answer is YES, an application of the algorithm D-EE^S from Section 4.2.2 on $(\mathscr{M}, s_0, \neg\eta)$ generates *all* the total extensions where $\neg\eta$ is existentially true at s_0.
      – If there is at least one (amongst the finitely many) total extension $\mathscr{M}^c$ of $\mathscr{M}$ that is not returned by D-EE^S, this means that $\neg\eta$ is not existentially true in $\mathscr{M}^c$ at s_0. Hence, η is universally true in $\mathscr{M}^c$ at s_0, therefore the answer to EA is YES.
      – Else, if every possible total extension of $\mathscr{M}$ is returned, then the answer of AE on $(\mathscr{M}, s_0, \neg\eta)$ is YES and EA returns NO.

6.2 The case of transition extensions

Now, we will adapt our methods to the case of transition extensions. (We can just as well consider the more general case of combined transition and label extensions, but for lack of space we focus on transition extensions only.) First, the tableau based method presented in Section 4.1 is modified to solve EE and EE^S in this case, as follows.

- The pretableau construction phase starts with a partial model $\mathscr{M} = (S, R^{must}, R^{may}, L)$, a state $s_0 \in S$ and an LTL formula η and aims to construct a pretableau $\mathscr{P}^{\mathscr{M}, s_0, \eta}$.
- The main difference is that, whenever the rule Next is applied to a given pretableau state (s, Δ, A_Δ), the first (state) components of the successor prestates are selected amongst all R^{may}-successors of s in $\mathscr{M}$. In the case of the EE problem, one such prestate is guessed, or selected non-deterministically, and added to the pretableau. In the case of the EE^S problem, all such prestates are identified and added to the pretableau.
  (In order to keep the method more economical in terms of the number of added transitions of the produced extension, in case of several possible successor prestates those corresponding to R^{must}-successors of s may be selected with priority.)
- Thereafter, the prestate elimination and state elimination phases, as well as the analysis of the outcome, being the final tableau, proceed essentially the same way as before.

Now, the problem EE^S is solved just like in the case of label extensions, with an algorithm generating all (finitely many) total transition extensions satisfying η. Then, the problems AA, EA, and AE are solved essentially in the same way as for label extensions.

6.3 The case of state extensions

Finally, we consider the most general case, of state extensions. Note that it subsumes the case of transition extensions, and it can also naturally incorporate the case of label extensions, where states with partial labels may be added. The EE problem now takes as an input

$\mathscr{M} = (S, R^{must}, R^{may}, L)$, a state $s_0 \in S$ and an LTL formula η and asks for a total extension of $\mathscr{M}$ with new states and transitions to them that has a path starting at s_0 and satisfying η.

Note that this problem is easily reducible to the LTL satisfiability problem as follows.

Let $Var(\eta)$ be the set of all propositional variables occurring in η, and let $l_{\mathscr{M}}(s_0, \eta) := \bigwedge \mathsf{Litt}(s_0, Var(\eta), L) = \bigwedge\{p | p \in Var(\eta), L(s_0, p) = \top\} \wedge \bigwedge\{\neg p | p \in Var(\eta), L(s_0, p) = \bot\}$.

Then $l_{\mathscr{M}}(s_0, \eta)$ is true at s_0 in $\mathscr{M}$ and in any extension of $\mathscr{M}$ of the most general type. Therefore, a path starting at s_0 in any such extension satisfies η iff it satisfies $\eta \wedge l_{\mathscr{M}}(s_0, \eta)$. On the other hand, any linear LTL model σ that satisfies $\eta \wedge l_{\mathscr{M}}(s_0, \eta)$ can be "grafted" to $\mathscr{M}$ as a path starting at s_0, i.e. all states of σ after s_0 can be added as new states to $\mathscr{M}$, and the resulting state extension of $\mathscr{M}$ will be a solution of the problem $\mathsf{EE}(\mathscr{M}, s_0, \eta)$. Thus, $\mathsf{EE}(\mathscr{M}, s_0, \eta)$ has a solution iff $\eta \wedge l_{\mathscr{M}}(s_0, \eta)$ is satisfiable. Furthermore, it suffices to consider only state extensions of $\mathscr{M}$ obtained by grafting at s_0 "small satisfiability witnesses" of the satisfiability of $\eta \wedge l_{\mathscr{M}}(s_0, \eta)$, *i.e.*, ultimately periodic linear models of size exponentially bounded by the length of $\eta \wedge l_{\mathscr{M}}(s_0, \eta)$ (cf. [12, Section 6.3]). So, the problem is clearly decidable, and can be solved in PSPACE, just like the satisfiability problem for LTL.

In a sense, this observation all but relatively trivialises the problem EE in the case of state extensions. If, however, a "minimal" state extension solving the problem $\mathsf{EE}(\mathscr{M}, s_0, \eta)$ is sought, in sense of adding the least possible number of additional states, then the problem becomes non-trivial again, in terms of its practical complexity. We leave the search for practically optimal algorithms to future work.

Now, the problem AA is again solved as dual to EE. To solve the AE and EA problems, we would need a procedure generating and checking all – infinitely many – state + transition extensions of $\mathscr{M}$ for existential, resp. universal truth of η. That task, however, can be reduced to a finitary one, as follows. Consider the EA problem. To solve it, it suffices to look for a total state extension of η solving the problem $\mathsf{EA}(\mathscr{M}, s_0, \eta)$ which is *minimal* in the sense of only grafting at each existing dead-end state an ultimately periodic path of sufficiently "small size" (lengths of the prefix and of the period, to be guessed, according to the theory) initially with labels assigning ? to all atomic propositions, and then solving the $\mathsf{EA}(\mathscr{M}, s_0, \eta)$ problem for a *label extension* on the resulting state extension. If a solution exists for some suitable guesses of the sizes of the appended ultimately periodic paths, then the initial $\mathsf{EA}(\mathscr{M}, s_0, \eta)$ problem for (total) state extensions has a solution, too. Otherwise, that problem has no solution. Indeed, if there is any such solution, then there would be one of the type described above for suitable choices of the sizes of the appended ultimately periodic paths, because of the "small satisfiability witness" property of LTL (cf. [12, Section 6.3]). The solution of the EA problem also solves the AE problem by duality.

7 Concluding remarks: summary, related work, and future work

7.1 Summary

In this paper we have formulated four decision problems concerning the possibility of extending and completing in three natural possible ways a partially defined or partially known transition system $\mathscr{M}$ which is supposed to satisfy a specification expressed by an LTL formula. We have also considered two variations of the first two problems, that are model synthesis problems. We have proposed tableau based algorithms that provably solve the core problems EE and EE^{S} and we have shown how these algorithms can also be used to solve the remaining problems.

7.2 Related Work

Here we mention and briefly discuss some previous publications that are related, at least in terms of the framework and problems they consider, to our work.

One essentially related work is [22], although the formal tools used there (partially labeled transition systems (PTS) and the logic FLTL) are different. Indeed, in that work PTS are labelled transition systems where states are not labelled by propositions (expressing static atomic properties), but, rather, transitions (edges) are labelled by actions; some actions (resp. transitions), however, are forbidden. As in our work, the authors are concerned with explicitly modelling those aspects of system behaviour that are still unknown, because knowing such gaps can help to improve incremental system modelling. Compared to the notion of PTS in [22], our notion of partial transition system allows for the expression of two distinct kind of incomplete knowledge: of static properties of individual states as well of transitions between states. Another specificity of our approach is that it allows for the precise formulation of six different completion problems and establishes a unified framework where the classical decisions problems of satisfiability, validity, existential model-checking and universal model-checking for LTL formulae are just specific cases.

In [14] two distinct, though related, problems are studied, both named *Generalized Model checking* (LTL GMC). The first problem had already been introduced in [10] and amounts to deciding, given a transition system $\mathscr{M}$ where the state labels are 3-valued, and an LTL formula η, whether there exists a 2-valued partial transition system $\mathscr{M}'$ that is "more complete" than $\mathscr{M}$ and that universally satisfies η, *i.e.* all the paths issued from its initial state satisfy η. The notion "more complete" is based on a completeness pre-order $\preceq$ on states of partial transition systems, that, in its turn, induces a pre-order on partial transition systems. This is closely related to the notion of bisimilarity between transition systems. The complexity of this problem is shown to be 2EXPTIME in the length of the formula and polynomial in the size of the model, by reduction to LTL synthesis. The LTL GMC problem in this version is obviously related to our EA problem, limited to the case where only state labels can be unknown. Yet, the two problems do not coincide. Our problem EA asks whether $\mathscr{M}$ *itself* can be extended (that is: its labels can), and possibly completed so as to get a bi-valued $\mathscr{M}^c$ that universally satisfy η, whereas the above described LTL GMC problem asks for the existence of *some* model $\mathscr{M}'$ satisfying η, where $\mathscr{M}$ and $\mathscr{M}'$ are related via the mentioned pre-order on structures; observe that $\mathscr{M}'$ might have less states than $\mathscr{M}$. The second version of LTL GMC studied in [14]) is based on a simpler pre-order on partial transition systems (previously suggested in [10]). It is less directly related to our work, so we will not discuss it.

For lack of space, we defer further references and details on related work to the Appendix.

7.3 Future Work

We intend to organise implementation of our algorithms and to experimentally test their practical feasibility on some case studies. For example, our approach might be adapted to the formal modelling of ecology systems via transition graphs, proposed in [21]. Clearly, some expert knowledge can be missing in representations based on empirical observations of the environment, and to fill such gaps might turn out useful. In that case, as in other application cases, certainly not every conceivable completion of a partial transition system might be suitable for the intended applications, because of possibly implicit assumptions on which are the "realistic possible completions". It would then be interesting to study also criteria for classifying completions, so as to direct the search in a more realistic way, according to the domain experts.

We also intend to extend the study to the branching time logics CTL and CTL*, and to a multi-agent context, where some agents and/or their actions might also be unspecified or unknown, thus possibly taking branching time and multi-agent logics, such as the alternating time temporal logic ATL [1], to express specification properties.

References

1 Alur, R.; Henzinger, T.A.; Kupferman, O.: Alternating-Time Temporal Logic. *J. ACM* **2002**, *49*, 672–713.

2 Andersen, H.R.: Partial model checking (extended abstract). In: Proc. of LICS 1995. pp. 398–407. IEEE Computer Society (1995)

3 Andersen, H.R., Lind-Nielsen, J.: Partial model checking of modal equations: A survey. Intern. J. on Software Tools for Technology Transfer **2**(3), 242–259 (1999)

4 Areces, C., Fervari, R., Hoffmann, G.: Relation-changing modal operators. Logic Journal of the IGPL **23**(4), 601–627 (2015)

5 Areces, C., Fervari, R., Hoffmann, G., Martel, M.: Undecidability of relation-changing modal logics. In: Proc. of DALI 2017. LNCS, vol. 10669, pp. 1–16. Springer (2017)

6 Areces, C., Fervari, R., Hoffmann, G., Martel, M.: Satisfiability for relation-changing logics. J. Log. Comput. **28**(7), 1443–1470 (2018)

7 Aucher, G., Balbiani, P., del Cerro, L.F., Herzig, A.: Global and local graph modifiers. Electr. Notes Theor. Comput. Sci. **231**, 293–307 (2009)

8 Aucher, G., van Benthem, J., Grossi, D.: Modal logics of sabotage revisited. J. Log. Comput. **28**(2), 269–303 (2018)

9 van Benthem, J.: An essay on sabotage and obstruction. In: Mechanizing Mathematical Reasoning, Essays in Honor of Jörg H. Siekmann on the Occasion of His 60th Birthday. LNCS, vol. 2605, pp. 268–276. Springer (2005)

10 Glenn Bruns and Patrice Godefroid. Generalized model checking: Reasoning about partial state spaces. In Catuscia Palamidessi, editor, *CONCUR 2000 - Concurrency Theory, 11th International Conference, University Park, PA, USA, August 22-25, 2000, Proceedings*, volume 1877 of *Lecture Notes in Computer Science*, pages 168–182. Springer, 2000. URL: `https://doi.org/10.1007/3-540-44618-4_14`, `doi:10.1007/3-540-44618-4\_14`.

11 Costa, G., Basin, D.A., Bodei, C., Degano, P., Galletta, L.: From natural projection to partial model checking and back. In: Proc. of TACAS 2018. LNCS, vol. 10805, pp. 344–361. Springer (2018)

12 Stéphane Demri, Valentin Goranko, and Martin Lange. *Temporal Logics in Computer Science: Finite-State Systems*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2016. `doi:10.1017/CBO9781139236119`.

13 Gabbay, D.M.: Reactive Kripke Semantics. Cognitive Technologies, Springer (2013)

14 Patrice Godefroid and Nir Piterman. LTL generalized model checking revisited. In Neil D. Jones and Markus Müller-Olm, editors, *Verification, Model Checking, and Abstract Interpretation, 10th International Conference, VMCAI 2009, Savannah, GA, USA, January 18-20, 2009. Proceedings*, volume 5403 of *Lecture Notes in Computer Science*, pages 89–104. Springer, 2009. URL: `https://doi.org/10.1007/978-3-540-93900-9_11`, `doi:10.1007/978-3-540-93900-9\_11`.

15 Valentin Goranko. Model checking and model synthesis from partial models: a logic-based perspective. https://arxiv.org/abs/2012.12398, 2020.

16 Kupferman, O., Vardi, M.Y.: Synthesis with incomplete informatio. In: Barringer, H., Fisher, M., Gabbay, D., Gough, G. (eds.) 2nd Intern. Conf. on Advances in Temporal Logic. pp. 109–127. Springer Netherlands, Dordrecht (2000)

17 David A Schmidt. Binary relations for abstraction and refinement. In *Workshop on Refinement and Abstraction, Amagasaki, Japan*. Citeseer, 1999.

18 A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *J. ACM*, 32(3):733–749, July 1985. URL: `https://doi.org/10.1145/3828.3837`, `doi:10.1145/3828.3837`.

19 Staruch, B.: Extensions of partial structures and their application to modelling of multiagent systems. In: Monitoring, Security, and Rescue Techniques in Multiagent Systems, MSRAS 2004, Plock, Poland, June 7-9, 2004. Advances in Soft Computing, vol. 28, pp. 293–304. Springer (2005)

20 Staruch, B., Staruch, B.: First order theories for partial models. Studia Logica **80**(1), 105–120 (2005)
21 Colin Thomas, Maximilien Cosme, Cédric Gaucherel, and Franck Pommereau. Model-checking ecological state-transition graphs. *PLoS Comput. Biol.*, 18(6), 2022. URL: `https://doi.org/10.1371/journal.pcbi.1009657`, `doi:10.1371/journal.pcbi.1009657`.
22 Sebastian Uchitel, Jeff Kramer, and Jeff Magee. Behaviour model elaboration using partial labelled transition systems. *ACM SIGSOFT Software Engineering Notes*, 28(5):19–27, 2003.
23 Pierre Wolper. The tableau method for temporal logic: An overview. *Logique et Analyse*, 28:119–136, 1985.

Appendix: figures on the example and further details on related work

A1. Some figures

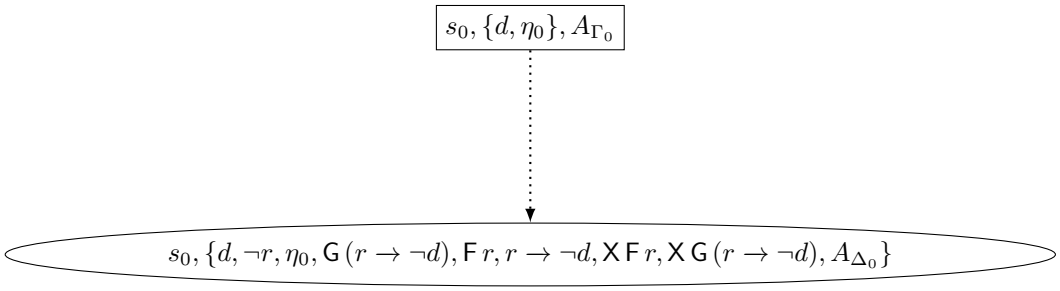

Figure 3 The pretableau construction stage after applying the rule Exp to the initial prestate, for Example 17.

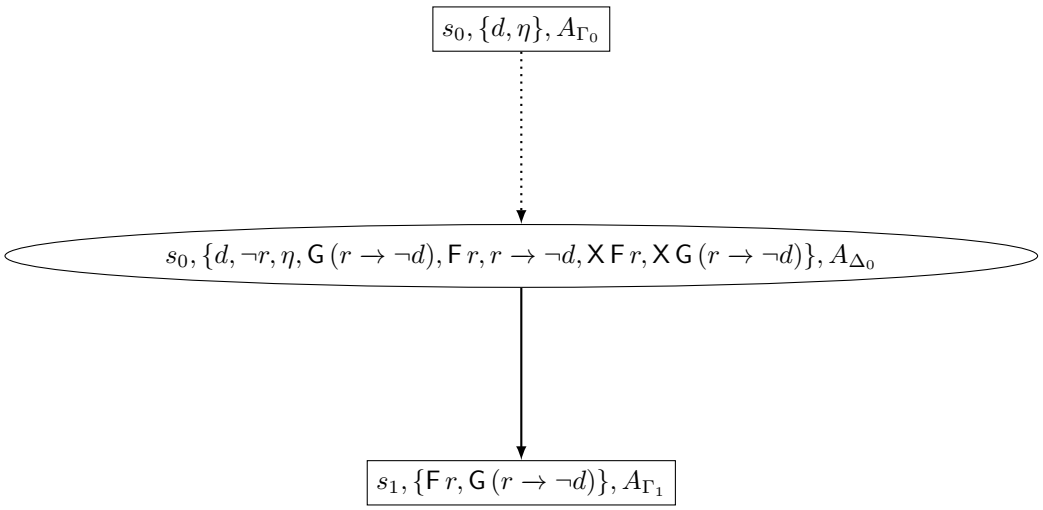

Figure 4 The pretableau construction stage for Example 19.

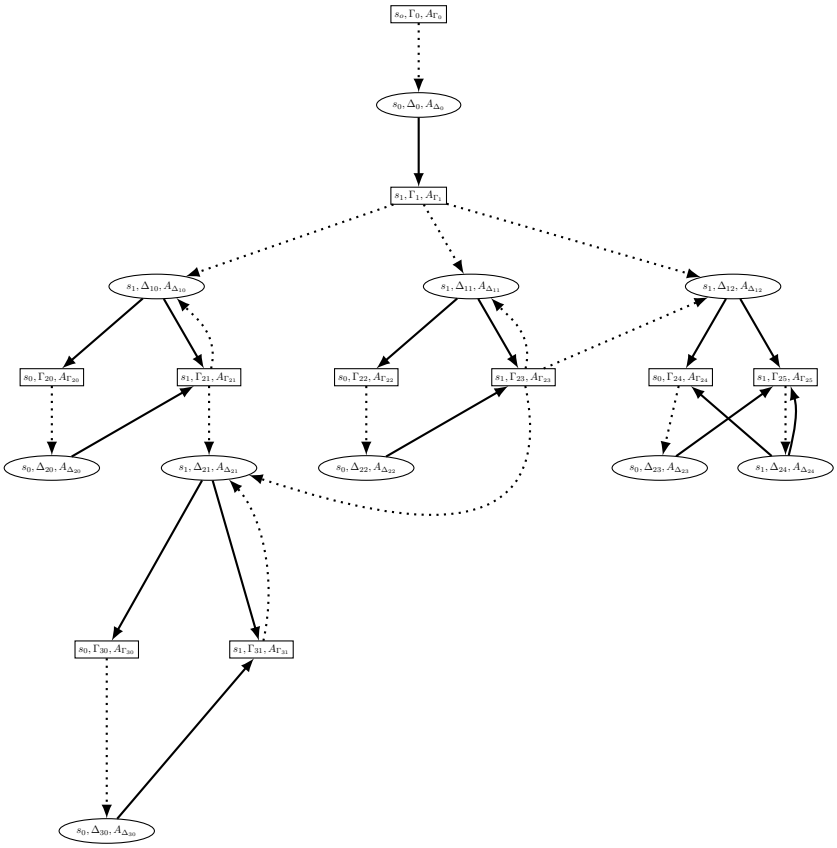

Figure 5 The complete pretableau for the running example 21.

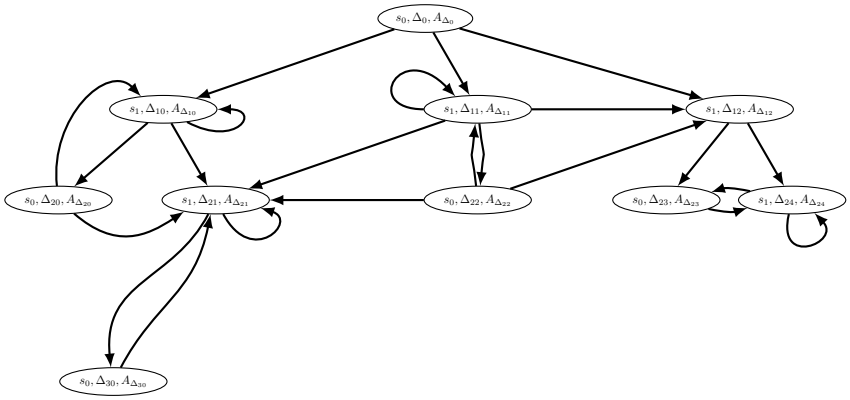

Figure 6 The initial tableau for the running example.

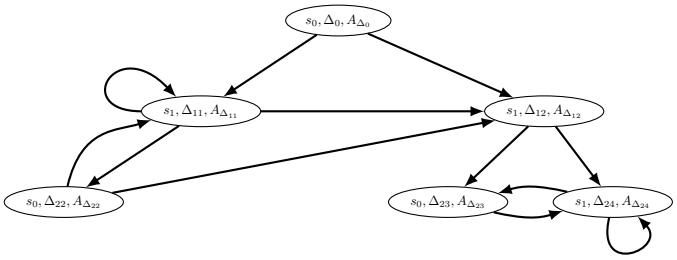

Figure 7 The final tableau for the running example, cf. Example 27.

A2. Further references and details on related work

Other, conceptually or technically related earlier works include:

– [20], taking an algebraic approach to the completion of partial first-order models, again representing partial information about the actual world;

– [2] (see also, [3], [11]) where a technique also called "partial model checking" was introduced for verifying concurrent systems by gradually removing concurrent components and then reducing the specification, until completely checked, in order to avoid the state explosion problem. This idea is only implicitly related to the problems discussed here.

– [16], where synthesis of reactive programs and systems under incomplete information is studied, in the case of an open system must be guaranteed to satisfy a given safety specification but can only partly read the input signals generated by its environment. An important practical case of synthesis from partial models is the problem of *controller synthesis*.

Lastly, another, more technically related to the present work, is the line of research in modal logic involving modal operators that change the models dynamically in the course of formula evaluation. It originates, *inter alia*, with van Benthem's *sabotage logics* [9], [8] and Gabbay's *reactive Kripke semantics* [13]. These were followed by various further proposals for *model update logics*, including enrichments of modal logics with *global and local graph modifiers* [7], and *relation-changing modal operators and logics* [4]. Typically, these approaches involve operations on models that not only extend, but also shrink or modify them in various ways and, moreover, add syntactic instructions for such operations in the language. These features can often lead to undecidability of both model checking and satisfiability of such logics, see [5], [6].