# Partial Model Checking and Partial Model Synthesis in LTL Using a Tableau-Based Approach

**Serenella Cerrito**
Université Paris Saclay, Univ EVRY, France

**Valentin Goranko**
Stockholm University, Sweden
and University of the Witwatersrand, Johannesburg, South Africa (visiting professorship)

**Sophie Paillocher**
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 $\eta$, 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.

**2012 ACM Subject Classification** Theory of computation → Logic and verification; Theory of computation → Modal and temporal logics; Theory of computation → Verification by model checking

**Keywords and phrases** Linear temporal logic LTL, partial transition systems, partial model checking, partial model synthesis, tableaux

**Digital Object Identifier** 10.4230/LIPIcs.FSCD.2023.23

**Acknowledgements** We thank the anonymous reviewers for their detailed and helpful comments.

## 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 $\eta$, 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 $\eta$ holds.*

**ii)** *whether what is already known suffices to verify $\eta$, 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 [14], on which the present work follows up.

In this paper we take the property $\eta$ 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, by constructive methods that are neither non-deterministic nor brute-force. Thus, when considering, for instance, whether there exists an extension of the partially given model $\mathscr{M}$ assuring the existence of a path verifying $\eta$, 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 $\mathscr{M}'$ of the input $\mathscr{M}$, and a path in it that is a linear model of $\eta$.

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 $\eta$, 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 [22] and further optimised and presented in detail in [11, 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 $\eta$. These problems are denoted respectively EE, AA, EA, AE. In this work we first present in full details the solution to EE (and to 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 [9, 13]. 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.

## 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 [16] (for transitions) and [13] (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 $p \in \mathrm{Prop}$, 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 [11].

The formulae of LTL are given by the grammar

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

where $p \in \mathrm{Prop}$. The operators $\vee$, $\to$, $\leftrightarrow$, and $\mathsf{F}$ 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** $\varphi$ **at a position** $i \in \mathbb{N}$ **of a linear model** $\sigma$, denoted $\sigma, i \models \varphi$, is defined as usual. An LTL formula $\varphi$ is said to be **true in a linear model** $\sigma$, denoted $\sigma \models \varphi$, if $\sigma, 0 \models \varphi$; in this case $\sigma$ is said to be a **linear model of** $\varphi$.

LTL formulae are also interpreted at states in transition systems, as follows. Given a transition system $\mathscr{M} = (S, R, L)$ the sequence $\pi = s_0, s_1, s_2, ...$ is a **path in** $\mathscr{M}$ if $s_0 R s_1 R s_2, ...$. Every finite sequence $s_0, s_1, ..., s_n$ is called an **initial sub-path of** $\pi$. The **computation in** $\mathscr{M}$ **corresponding to** $\pi$ is the sequence of labels (sets of true propositions) along $\pi$: $T(s_0), T(s_1), T(s_2) \dots, ...$, where $T(i) = \{p \in \text{Prop} \mid L(s_i, p) = \top\}$. Given a state $s \in S$, we denote by $comput_{\mathscr{M}}(s)$ the set of computations in $\mathscr{M}$ with initial state $s$.

Every computation 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 computation $\sigma$. We say that an LTL formula $\varphi$ is **universally true** in a transition system $\mathscr{M}$ at a state $s$ if $\sigma \models \varphi$ for all elements $\sigma$ of $comput_{\mathscr{M}}(s)$. In that case, we also simply say that $\varphi$ **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 $\varphi$ is **existentially true in** $\mathscr{M}$ **at** $s$ if for some computation $\sigma \in comput_{\mathscr{M}}(s)$ we have $\sigma \models \varphi$. In that case we write $\mathscr{M}, s \models_\exists \varphi$.

Given a finite set of formulas $\Gamma = \{\varphi_1, \dots, \varphi_n\}$, where $n \geq 0$, we denote $\bigvee \Gamma := \varphi_1 \vee \dots \vee \varphi_n$, and $\bigwedge \Gamma := \varphi_1 \wedge \dots \wedge \varphi_n$. When $\Gamma = \emptyset$, $\bigvee \Gamma := \bot$ and $\bigwedge \Gamma := \top$.
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 [11, Ch.13.2].

We distinguish four types of formulas in LTL: *literals (atoms or negated atoms), conjunctive, disjunctive and successor formulas*, and *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 $\varphi$ *must be realised at some point in the future*, but its realisation can be procrastinated. By the well known fixed-point LTL equivalences $\mathsf{F}\,\varphi \equiv \varphi \vee \mathsf{X}\,\mathsf{F}\,\varphi$ and $\psi \,\mathsf{U}\, \varphi \equiv \varphi \vee (\psi \wedge \mathsf{X}\,(\psi \,\mathsf{U}\, \varphi))$, an eventuality can be seen as a disjunction, as it is shown Figure 1.

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

**Table 1** Types and components of formulas in LTL.

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

When $\Gamma$ is a set of formulae, the notation $Scomp(\Gamma)$ will denote $\{scomp(\psi) | \psi \in \Gamma\}$.

▶ **Definition 5** (Extended closure of a formula). *The extended closure $\mathsf{ecl}(\varphi)$ of the formula $\varphi$ 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* $\Gamma$ *we define*

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

*A set of formulas* $\Gamma$ *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* $\varphi$ *and* $\neg\varphi$*.*

▶ **Definition 8** (Fully expanded). *A set of formulas* $\Gamma$ *is* ***fully expanded*** *if and only if*
1. *it is not patently inconsistent,*
2. *for every conjunctive formula in* $\Gamma$*, all of its conjunctive components are in* $\Gamma$*,*
3. *for every disjunctive formula in* $\Gamma$*, at least one of its disjunctive components is in* $\Gamma$*.*

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* $\Delta$ *is a* ***full expansion*** *of a set of formulas* $\Gamma$ *if* $\Delta$ *can be obtained from* $\Gamma$ *by repeated applications of the following rules, where initially no formula is marked as "used":*
- ***(C-comp)*** *for every conjunctive formula* $\varphi$ *in the current set* $\Gamma$ *that has not been marked as "used", add all of its conjunctive components to* $\Gamma$ *and mark* $\varphi$ *as "used".*
- ***(D-comp)*** *for every disjunctive formula* $\varphi$ *in the current set* $\Gamma$ *that has not been marked as "used", add one of its disjunctive components to* $\Gamma$ *and mark* $\varphi$ *as "used".*

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

Below we also recall some definitions and a theorem about Hintikka traces, that can be found in [11], as we will make use of them in our approach.

▶ **Definition 10.** *Given a closed set of formulas* $\Gamma$*, a* ***Hintikka trace*** *(HT) for* $\Gamma$ *is a mapping* $H : \mathbb{N} \to \mathscr{P}(\Gamma)$ *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* $\varphi$ *is* ***satisfiable in a Hintikka trace*** $H$ *if* $\varphi \in H(n)$ *for some* $n \in \mathbb{N}$*.*

▶ **Theorem 11.** *An LTL formula* $\eta$ *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 $\eta$ 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 $\mathscr{M}'$ of $\mathscr{M}$ and a path $s_0, s_1, s_2, \ldots$ in $\mathscr{M}'$ such that its corresponding computation $\sigma$ is a linear model of $\eta$? We denote this decision problem by EE.
- **Existential extension for all paths**
  Is there a total extension $\mathscr{M}'$ of $\mathscr{M}$, so that for all paths $s_0, s_1, s_2, \ldots$ in $\mathscr{M}'$ the corresponding computations are linear models of $\eta$? We denote this decision problem by EA.

- **Universal extension for path existence**
  Does every total extension $\mathscr{M}'$ of $\mathscr{M}$ has a path $s_0, s_1, s_2, \ldots$ such that its corresponding computation is a linear model of $\eta$? We denote this decision problem by AE.
- **Universal extension for all paths**
  Is every total extension $\mathscr{M}'$ of $\mathscr{M}$ such that for every path $s_0, s_1, s_2, \ldots$ in it the corresponding computation is a linear model of $\eta$? 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 AA 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 $\mathscr{M}'$ and existentially - respectively universally - model-check $\eta$ on $\mathscr{M}'$. Since the decision problems for existential and universal model checking of LTL formulae are PSPACE complete ([17, 11]), 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 $\eta$ 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. Still, let us note that the decision methods developed here are practically much more efficient than brute-force, generic PSPACE-complete algorithms.

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 $\mathscr{M}'$ and a path $\pi = s_0, s_1, s_2, \ldots$ in $\mathscr{M}'$ such that its corresponding trace $\sigma$ is a linear model of $\eta$, when these exists (and for a failure message otherwise) is the synthesis problem that we will denote by $\mathsf{EE}^{\mathsf{S}}$. Similarly, the variant of EA where one asks for an extension $\mathscr{M}^c$ where $\eta$ is universally true is the synthesis problem denoted by $\mathsf{EA}^{\mathsf{S}}$.

## 4 Tableau-based algorithms for EE and $\mathsf{EE}^{\mathsf{S}}$ for label extensions

Usually, tableau methods are used to determine the satisfiability of an input formula $\eta$ by trying systematically to build from scratch possible models of it. Here, we adapt the method to take into account also a partial Kripke structure $\mathscr{M}$ as an additional input, the goal being to possibly construct a model of $\eta$ extending $\mathscr{M}$.

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 $\mathsf{EE}$ and $\mathsf{EE}^{\mathsf{S}}$ (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 $\mathsf{LTL}$ formula $\eta$, the algorithm enables the computation of *all* the total extensions[1] $\mathscr{M}'$ where $\eta$ 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 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 1. Formally the question is "*Is there an extension $\mathscr{M}'$ of the partial model $\mathscr{M}$ and a computation $\sigma$ in $comput_{\mathscr{M}}(s_0)$ such that $\sigma, s_0 \models \eta_0$?*", where $\eta_0 := \mathsf{F}\, r \wedge \mathsf{G}\,(r \to \neg d)$, with $d$ interpreted as "*the doors are open*" and $r$ as "*the train is running*".

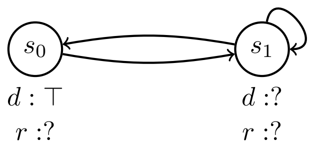

**Figure 1** A partial model for Example 12.

The approach that we use to solve this problem is based on the tableau methods for $\mathsf{LTL}$ satisfiability, also modified for $\mathsf{LTL}$ model-checking, outlined in [11, 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 $\mathsf{EE}$ 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 $\mathscr{M}'$ of $\mathscr{M}$ and a path $\pi$ in it, starting from $s_0$, such that the corresponding computation in $\mathscr{M}^c$ satisfies $\eta$.

### 4.1.1 Pretableau Construction Phase

Here, as in Section 2, Prop is a set of atomic propositions. Given a partial model $\mathscr{M} = (S, R, L)$ (i.e., $R = R^{must} = R^{may}$) a state $s_0 \in S$ and an $\mathsf{LTL}$ formula $\eta$, we construct a graph called **pretableau of** $(\mathscr{M}, s_0, \eta)$ and denoted $\mathscr{P}^{\mathscr{M}, s_0, \eta}$. The pretableau, and then the tableaus, are directed graphs consisting of **nodes** and arrows between them. The nodes are triples

[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 $\eta$. 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.

$(s, \Gamma, A_\Gamma)$ consisting of: (the name of) a state $s$ in $\mathscr{M}$, a set of formulas $\Gamma$ that must be true at that state, which we call its **description**, and an **annotation function** $A_\Gamma : S \times \mathrm{Prop} \to B$, explained further[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.

As explained in [11, Chapter 13], the sets of formulae labelling a tableau prestate are not yet fully processed. For example, a prestate component $\Gamma$ may declare that a formula $p \vee q$ is true in some state $s$ in $\mathscr{M}$ without indicating which of $p$ and $q$ must be true. Thus, in order to get a precise description of $s$, $\Gamma$ needs to be expanded (via the rule later called $\mathsf{Exp}$) into two alternative sets, by explicitly adding respectively $p$ and $q$ to them, and possibly leading to two "offspring states".

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

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

## 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 $\Gamma_0 := \{\eta\} \cup \mathsf{Litt}(s_0, ecl(\eta))$ and the initial annotation function $A_{\Gamma_0}$ coincides with $L$. Note that, in general, any propositional letter $q \in PROP \cap \mathsf{ecl}(\eta)$ such that $r$ does not occur in $\eta$ is such that neither of $q$ and $\neg q$ is in $ecl(\eta)$, so $r$'s truth value does not impact the satisfaction of $\eta_0$.

▶ **Example 14** (Running Example continued)**.** In Example 12 with $\eta_0 = \mathsf{F}\, r \wedge \mathsf{G}\,(r \to \neg d)$, we obtain $\mathsf{ecl}(\eta_0) = \{\eta_0, \mathsf{F}\, r, r, \mathsf{X}\,\mathsf{F}\, r, \mathsf{G}\,(r \to \neg d), r \to \neg d, \mathsf{X}\,\mathsf{G}\,(r \to \neg d), \neg r, \neg d\}$. Further, the input partial model is such that $L(s_0, d) = \top$ and $L(s_0, r) =?$, so $\mathsf{Litt}(s_0, ecl(\eta_0), L) = \emptyset$ and $\Gamma_0 := \{\eta_0\}$. Note that although $\neg r \in \mathsf{ecl}(\eta_0)$ neither $r$ nor $\neg r$ appears in $\Gamma_0$, as its truth value is undetermined/unknown. The construction of the tableau begins with the creation of the prestate node $(s_0, \Gamma_0, A_{\Gamma_0})$ where $A_{\Gamma_0} = L$

## 4.1.3 The body of the algorithm

Two rules will be alternatively applied:

- $\mathsf{Exp}$: producing so called **offspring states** of a given prestate,
- $\mathsf{Next}$: producing **successor prestates** of a given state.

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

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

[2] An annotation function 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.

▶ **Definition 15** (Rule Exp). *Given a prestate $(s, \Gamma, A_\Gamma)$, do the following:*

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

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

   *and add in the current pretableau a new offspring state $(s, \Delta, A_\Delta)$.*
3. *For each newly introduced state $(s, \Delta, A_\Delta)$, create an edge $(s, \Gamma, A_\Gamma) \dashrightarrow (s, \Delta, A_\Delta)$.*
4. *If the pretableau already contains a state $(s, \Delta, A_\Delta)$, 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 $(s, \Gamma, A_\Gamma)$ is denoted by* 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 $\Gamma_0$ contains only one state, $\Delta_0$, because a choice of $r$ when expanding $\Gamma_0$ would lead to a patently inconsistent set of formulae. We have:

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

Note that $\neg r \in \Delta_0$ while $L(s_0, r) = ?$.
The new annotation function $A_{\Delta_0}$ is defined by: $A_{\Delta_0}(s_0, d) = \top$ and $A_{\Delta_0}(s_0, r) = \bot$.
This is the beginning of the update of the input interpretation function in $\mathscr{M}$.

The currently constructed pretableau is given in Figure 2.

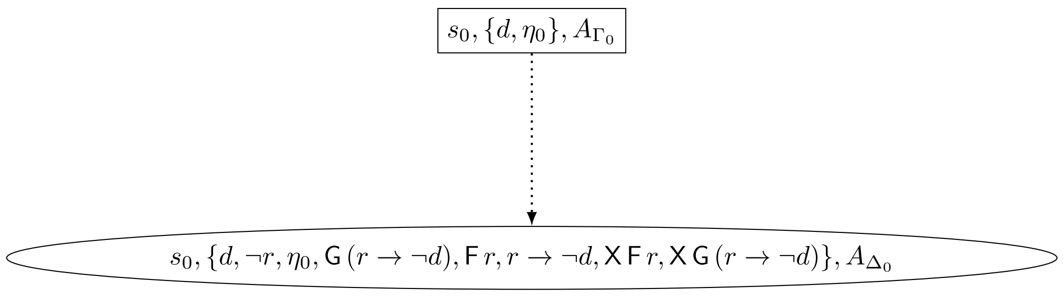
$s_0, \{d, \eta_0\}, A_{\Gamma_0}$
$s_0, \{d, \neg r, \eta_0, \mathsf{G}\,(r \to \neg d), \mathsf{F}\,r, r \to \neg d, \mathsf{X}\,\mathsf{F}\,r, \mathsf{X}\,\mathsf{G}\,(r \to \neg d)\}, A_{\Delta_0}$

**Figure 2** The pretableau construction stage after applying the rule Exp to the initial prestate, for Example 17.

The rule Next creates in the tableau graph successors of any *tableau state* $(s, \Delta, A_\Delta)$ corresponding to a *state* $s$ in input model $\mathscr{M}$. (NB the distinction between states in a model, which are semantic objects, and tableau states, which are syntactic objects.)

▶ **Definition 18** (Rule Next). *Given a state $(s, \Delta, A_\Delta)$, do the following:*

- *If there are no $R$-successors of $s$ in the partial model, remove the state $(s, \Delta, A_\Delta)$ 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 $(t, Scomp(\Delta) \cup$ Litt$(t, ecl(\eta), A_\Delta))$ of $(s, \Delta, A_\Delta)$*
  - *If the pretableau-graph already contains a node $V = (t, \Gamma, A_\Gamma)$ that is then do not create a new state but create only an edge $(s, \Delta, A_\Delta) \to (t, \Gamma, A_\Gamma)$ to $V$.*

▶ **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 $\Gamma_1 := \{\mathsf{F}\, r, \mathsf{G}\,(r \to \neg d)\}$. We have that $A_{\Gamma_1} = A_{\Delta_0}$(only the rule Exp can modify an annotation function). The current pretableau is given in Figure 3.

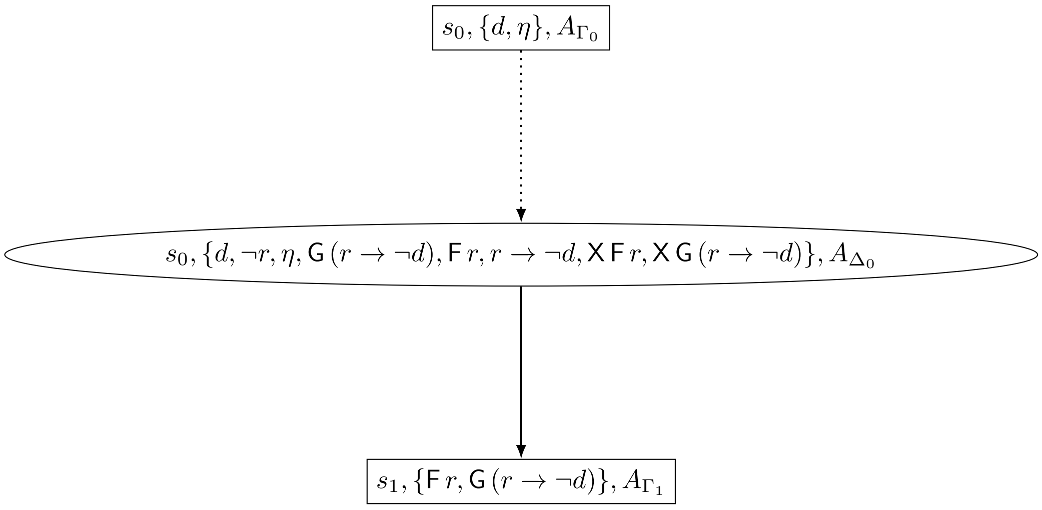
$$\begin{tikzcd}
\boxed{s_0, \{d, \eta\}, A_{\Gamma_0}} \arrow[d, dotted] \\
s_0, \{d, \neg r, \eta, \mathsf{G}\,(r \to \neg d), \mathsf{F}\, r, r \to \neg d, \mathsf{X}\,\mathsf{F}\, r, \mathsf{X}\,\mathsf{G}\,(r \to \neg d)\}, A_{\Delta_0} \arrow[d] \\
\boxed{s_1, \{\mathsf{F}\, r, \mathsf{G}\,(r \to \neg d)\}, A_{\Gamma_1}}
\end{tikzcd}$$

**Figure 3** The pretableau construction stage for Example 19.

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

**Proof.** For any pretableau node $\langle s, \Sigma, A\rangle$ there are only finitely many annotation functions $A$ (since both $S$ and $P$ are finite) and $\Sigma \subseteq \mathsf{ecl}(\eta)$, where $\mathsf{ecl}(\eta)$ is finite. Now, it suffices to note that both the rules Exp and Next disallow duplication of already existing nodes. ◀

▶ **Example 21** (Running example continued). At the end of the construction phase we obtain the pretableau shown in the in the Figure 4, where prestates are indicated with rectangular boxes and states with oval boxes. Nodes of the pretableau are enumerated to help reading the picture, but we leave it to the interested reader to flesh 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 $(s, \Gamma, A_\Gamma)$ in the pretableau, do:*
1. *Remove $(s, \Gamma, A_\Gamma)$ from the pretableau;*
2. *If there is a state $(t, \Delta, A_\Delta)$ 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.

In what follow, 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.

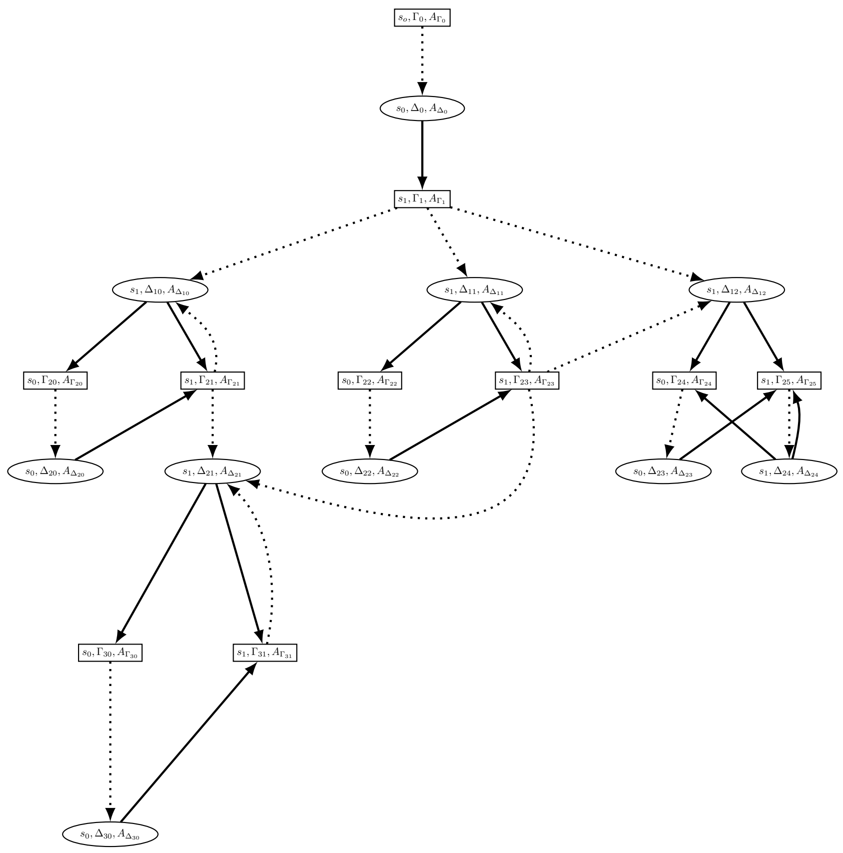
$s_0, \Gamma_0, A_{\Gamma_0}$
$s_0, \Delta_0, A_{\Delta_0}$
$s_1, \Gamma_1, A_{\Gamma_1}$
$s_1, \Delta_{10}, A_{\Delta_{10}}$
$s_1, \Delta_{11}, A_{\Delta_{11}}$
$s_1, \Delta_{12}, A_{\Delta_{12}}$
$s_0, \Gamma_{20}, A_{\Gamma_{20}}$
$s_1, \Gamma_{21}, A_{\Gamma_{21}}$
$s_0, \Gamma_{22}, A_{\Gamma_{22}}$
$s_1, \Gamma_{23}, A_{\Gamma_{23}}$
$s_0, \Gamma_{24}, A_{\Gamma_{24}}$
$s_1, \Gamma_{25}, A_{\Gamma_{25}}$
$s_0, \Delta_{20}, A_{\Delta_{20}}$
$s_1, \Delta_{21}, A_{\Delta_{21}}$
$s_0, \Delta_{22}, A_{\Delta_{22}}$
$s_0, \Delta_{23}, A_{\Delta_{23}}$
$s_1, \Delta_{24}, A_{\Delta_{24}}$
$s_0, \Gamma_{30}, A_{\Gamma_{30}}$
$s_1, \Gamma_{31}, A_{\Gamma_{31}}$
$s_0, \Delta_{30}, A_{\Delta_{30}}$

**Figure 4** The complete pretableau for the running example 21.

▶ **Definition 23** (Rule StateElim1). *If a state $(r, \Delta, A_\Delta)$ has no successor states in the current tableau, then remove $(r, \Delta, A_\Delta)$ from the tableau.*

In order to formulate StateElim2 we need the following definition.

▶ **Definition 24** (Realization of an eventuality at a tableau state). *An eventuality $\varphi \,\mathsf{U}\, \psi$ (resp. $\mathsf{F}\,\psi$) is realized at the state $(s, \Delta, A_\Delta)$ in $\mathcal{T}_n^{\mathcal{M}, s_0, \eta}$ if there exists a finite path in $\mathcal{T}_n^{\mathcal{M}, s_0, \eta}$*

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

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

▶ **Definition 25** (Rule StateElim2). *If an eventuality $\alpha \in \Delta$ is not realised on any path starting from $(s, \Delta, A_\Delta)$ in the current tableau, then remove $(s, \Delta, A_\Delta)$ from the tableau.*

The state elimination phase is carried out in a sequence of stages, starting at stage 0 with the initial tableau $\mathcal{T}_0^{\mathcal{M}, s_0, \eta}$, and eliminating at every stage $n$ at most one state for the current tableau $\mathcal{T}_n^{\mathcal{M}, s_0, \eta}$ – 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 26** (Running example continued)**.** Figure 5 shows the initial tableau, where only prestates have been eliminated. All the states here have at least one successor state so the rule StateElim1 can not be applied. However there is a state with description component $\{d, \neg r, \mathsf{F}\, r, \mathsf{X}\,\mathsf{F}\, r, \mathsf{G}\,(r \rightarrow \neg d), r \rightarrow \neg d, \mathsf{X}\,\mathsf{G}\,(r \rightarrow \neg d\})$ shown in Figure 6.

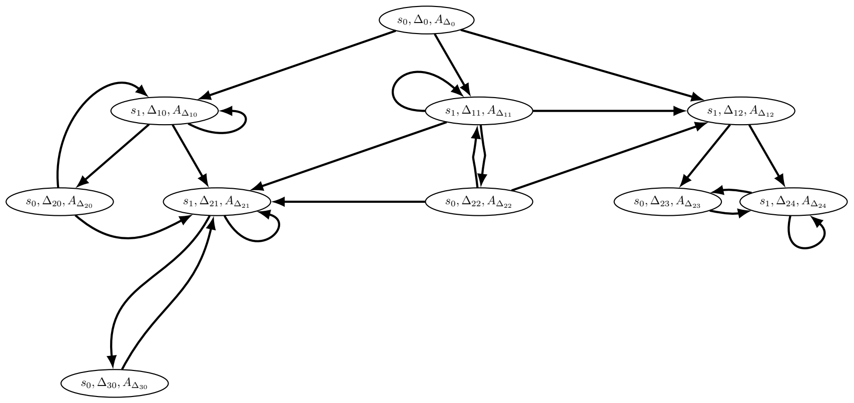
$s_0, \Delta_0, A_{\Delta_0}$
$s_1, \Delta_{10}, A_{\Delta_{10}}$
$s_1, \Delta_{11}, A_{\Delta_{11}}$
$s_1, \Delta_{12}, A_{\Delta_{12}}$
$s_0, \Delta_{20}, A_{\Delta_{20}}$
$s_1, \Delta_{21}, A_{\Delta_{21}}$
$s_0, \Delta_{22}, A_{\Delta_{22}}$
$s_0, \Delta_{23}, A_{\Delta_{23}}$
$s_1, \Delta_{24}, A_{\Delta_{24}}$
$s_0, \Delta_{30}, A_{\Delta_{30}}$

**Figure 5** The initial tableau for the running example.

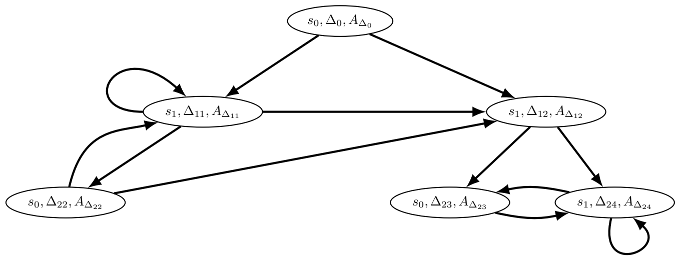
$s_0, \Delta_0, A_{\Delta_0}$
$s_1, \Delta_{11}, A_{\Delta_{11}}$
$s_1, \Delta_{12}, A_{\Delta_{12}}$
$s_0, \Delta_{22}, A_{\Delta_{22}}$
$s_0, \Delta_{23}, A_{\Delta_{23}}$
$s_1, \Delta_{24}, A_{\Delta_{24}}$

**Figure 6** The final tableau for the running example 26.

▶ **Definition 27.** *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 $\eta$. 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

The output for the decision problem EE 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 $\mathsf{EE}^\mathsf{S}$

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 $\eta$. The second version (useful in Section 6) is *deterministic* and produces *all possible ways to extend* the input model so as to existentially satisfy $\eta$, and thus enables the generation of all such extensions, which we will use in Section 6.

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

1. Construct the tableau $\mathscr{T}^{\mathscr{M},s_0,\eta}$.
2. If the final tableau is closed, then return the empty set. Else:
   - **a.** Non-deterministically choose a path $(s_0, \Delta_0, A_{\Delta_0}), (s_1, \Delta_1, A_{\Delta_1}), \ldots$ 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 $\Delta_0, \Delta_1, \ldots$ is a Hintikka trace (as in Definition 10).
     (When the final tableau is open such a Hintikka trace exists, as we prove in Section 5.)
   - **b.** Define $L'$ as follows:
     - if $A_{\Delta_n}(s,p) = \top$ for some node $(s_n, \Delta_n, A_{\Delta_n})$ on the chosen path, then $L'(s,p) := \top$;
     - else:
       - if $A_{\Delta_n}(s,p) = \bot$ for some node $(s_n, \Delta_n, A_{\Delta_n})$ on the chosen path then $L'(s,p) := \bot$;
       - else set $L'(s,p) := ?$.
   - **c.** Return $\mathscr{M}' = (S, R, L')$ and the path $s_0, s_1, s_2, \ldots$.

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

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 paths in $\mathscr{T}^{\mathscr{M},s_0,\eta}$ starting from a root node such that the sequence of their description components is a Hintikka trace (Def. 10):

   $$paths(\mathscr{T}^{\mathscr{M},s_0,\eta}) := \{(s_0, \Delta_0, A_{\Delta_0}), (s_1, \Delta_1, A_{\Delta_1}), \cdots \mid \Delta_0, \Delta_1, \ldots \text{ is a Hintikka trace}\}$$

   For every tableau path $\pi$ in $paths(\mathscr{T}^{\mathscr{M},s_0,\eta})$, define $\sigma_\pi = s_0, s_1, s_2, \ldots$ and the set $LC(\sigma_\pi)$ as the set of *all* the interpretation functions $L^c{}_{\sigma_\pi}$ such that:
   - if $A_{\Delta_n}(s,p) = \top$ for some node $(s_n, \Delta_n, A_{\Delta_n})$ of $\pi$, then $L^c{}_{\sigma_\pi}(s,p) = \top$;
   - if $A_{\Delta_n}(s,p) = \bot$ for some node $(s_n, \Delta_n, A_{\Delta_n})$ of $\pi$, then $L^c{}_{\sigma_\pi}(s,p) = \bot$.
   - If $A_{\Delta_n}(s,p) = ?$ for every node $(s_n, \Delta_n, A_{\Delta_n})$ of $\pi$, 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 28** (End of the running example)**.** The final tableau for the running example of this section is the one in Figure 6. 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 $\eta$ exists. For instance, one can take the path $s_0, s_1, s_1, s_1, \dots$ 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 $(s_0, \Delta_0, A_{\Delta_0}), (s_1, \Delta_{12}, A_{\Delta_{12}}), (s_1, \Delta_{24}, A_{\Delta_{24}}), (s_1, \Delta_{24}, A_{\Delta_{24}}), \dots$ corresponds to the Hintikka trace $\Delta_0, \Delta_{12}, \Delta_{24}, \Delta_{24}, \dots$ where
$\Delta_0 := \{d, \neg r, \eta, \mathsf{G}\,(r \to \neg d), \mathsf{F}\,r, r \to \neg d, \mathsf{X}\,\mathsf{F}\,r, \mathsf{X}\,\mathsf{G}\,(r \to \neg d)\}$,
$\Delta_{12} := \{\neg d, r, \mathsf{F}\,r, \mathsf{G}\,(r \to \neg d), r \to \neg d, \mathsf{X}\,\mathsf{G}\,(r \to \neg d)\}$,
$\Delta_{24} := \{\neg d, r, \mathsf{G}\,(r \to \neg d), r \to \neg d, \neg d, \mathsf{X}\,\mathsf{G}\,(r \to \neg d))\}$.

The following result is an immediate consequence of Lemma 20:

▶ **Theorem 29.** *The algorithms for* EE *and for* $\mathsf{EE}^{\mathsf{S}}$ *(in both versions) terminate.*

## 5 Soundness and Completeness Results

▶ **Theorem 30.** *Let $\mathscr{M} = (S, R, L)$ be a partial model, let $s_0 \in S$ be an initial state and let $\eta$ be an* LTL *formula, such that $\mathscr{T}^{\mathscr{M}, s_0, \eta}$ is an open tableau. Then the following hold:*

1. Soundness with respect the existence of a solution for EE.
   *The problem* EE *has an answer YES.*
2. Appropriate tableau paths describe solutions for EE.
   *Let $\pi = (s_0, \Delta_0, A_0), (s_1, \Delta_1, A_1), (s_2, \Delta_2, A_2), \dots$ be $\underline{\text{any}}$ path in $\mathscr{T}^{\mathscr{M}, s_0, \eta}$ such that $\Delta_0, \Delta_1, \Delta_2, \dots$ is a Hintikka trace. Then for any extension $\mathscr{M}' = (S, R, L')$ of $\mathscr{M}$ such that $A_n \preceq L'$ for each $n$, the computation $\sigma$ in $\mathscr{M}' = (S, R, L')$ corresponding to $s_0, s_1, s_2, \dots$ satisfies $\eta$.*

**Proof.**

1. Here, intuitively we show that *any* open tableau $\mathscr{T}^{\mathscr{M}, s_0, \eta}$ necessarily contains at least *some path* $\pi$ that "describes" *some* solution to EE.
   First, we inductively construct an infinite chain $\mathscr{S}$ of initial sub-paths in the tableau:
   $\pi_0 = (s_0, \Delta_0, A_0)$,
   $\pi_1 = (s_0, \Delta_0, A_0), (s_1 \Delta_1, A_1)$,
   $\pi_2 = (s_0, \Delta_0, A_0), (s_1, \Delta_1, A_1), (s_2, \Delta_2, A_2)$...
   where each $\pi_k$ is a finite initial sub-path of the next one, as follows:
   - $\pi_0$ consists of the initial state node $(s_0, \Delta_0, A_0)$, where $\eta \in \Delta_0$ and $A_0 = L$.
   - Let $\pi_k$ be the last constructed sub-path and let $V = (s_k, \Delta_k, A_k)$ be its last node. Let $E$ be the set of eventualities in $\Delta_k$.
     - If $E$ is non-empty, let $e = \varphi \,\mathsf{U}\, \psi$ be one of them. Since rule StateElim2 has not removed $V$, there is a finite sub-path $\pi_{sub}$ in the tableau which starts at $V$ and ends at a node $(s_{k+m}, \Delta_{k+m}, A_{k+m})$, for some $m \geq 0$, such that $\varphi \,\mathsf{U}\, \psi, \psi \in \Delta_{k+m}$. Let us observe that if $e' = \varphi' \,\mathsf{U}\, \psi'$ is another eventuality in $E$, and $\psi'$ does not appear in (the description component of) any node of $\pi_{sub}$, then each node in such a sub-path will be such that its $\Delta_l$ component has $\varphi' \,\mathsf{U}\, \psi'$ as an element (by tableau construction). We then build the new sub-path $\pi_{k+1}$ by appending $\pi_{sub}$ to $\pi_k$.

- If $E$ is empty, then let $V'$ be any successor of $V$ in the tableau $\mathcal{T}^{\mathcal{M},s_0,\eta}$ (by rule StateElim1 each tableau path is infinite). We extend $\pi_k$ by appending $V'$ to it, thereby obtaining $\pi_{k+1}$. Let us observe that this case necessarily eventually occurs, because eventuality formulae in $\mathcal{T}^{\mathcal{M},s_0,\eta}$ are elements of $\mathsf{ecl}(\eta)$, that is finite.

Let $\pi$ be the limit tableau path of the sequence $\mathcal{S}$. The sequence of the description components of the vertices in $\pi$ is a Hintikka trace (see Definition 10). By construction, each node $(s_i, \Delta_i, A_i) \in \pi$ is such that $L \preceq A_i$, and for some $n \geq 0$ the annotation function becomes stable, that is, for any $m \geq n$, $A_m = A_n$.

Now let $\mathcal{M}'$ be the extension of $\mathcal{M}$ obtained by replacing $L$ by $A_n$. It is easy to show that every state $s_i$ in $\mathcal{M}'$ satisfies all the formulae in $\Delta_i$. In particular $s_0$ satisfies all formulae in $\Delta_0$, in particular $\eta$. Therefore the computation corresponding to the path $s_0, s_1, s_2, ...$ generated by $\pi$ is a linear model of $\eta$, and the answer to EE is YES.

2. Now we must show that *any* tableau path $\pi = (s_0, \Delta_0, A_0), (s_1, \Delta_1, A_1), (s_2, \Delta_2, A_2), \ldots$ where the corresponding description components constitute a Hintikka trace describes a linear model $\sigma$ of $\eta$ for some extension $\mathcal{M}' = (S, R, L')$ of $\mathcal{M}$ (actually for a class of extensions). A computation $\sigma$ in $\mathcal{M}' = (S, R, L')$ corresponding to $s_0, s_1, s_2, ...$ is defined so that, for any $i \geq 0$, $L'(s_i) = \mathrm{Prop} \cap \Delta_i$. For any $i$, we have $\sigma, i \models \Delta_i$, thus, in particular, we have $\sigma, 0 \models \Delta_0$, and $\eta \in \Delta_0$, which implies that $\sigma$ is a linear model of $\eta$. ◀

The completeness result below shows not only that when the answer to EE is YES due to some extension $\mathcal{M}'$ then the corresponding tableau is open, but a stronger result, which intuitively says that each computation in $\mathcal{M}'$ satisfying $\eta$ is described by a path of such a tableau. We make use of this stronger result in Section 6.

▶ **Theorem 31** (Completeness, stronger version).

1. All solutions to EE are described by tableau paths.
   *Let $\mathcal{M} = (S, R, L)$ be a partial transition system that is transition-complete, let $\eta$ be an* LTL *formula and let $s_0 \in S$. Let $\mathcal{M}' = \{S, R, L\}$ be any total extension of $\mathcal{M}$ and let $s_0, s_1, s_2, \ldots$ be a path in $\mathcal{M}'$ such that its corresponding computation $\sigma$ satisfies $\eta$.*
   *Then any final tableau $\mathcal{T}^{\mathcal{M},s_0,\eta}$ contains a path $\pi = (s_0, \Delta_0, A_0), (s_1, \Delta_1, A_1), (s_2, \Delta_2, A_2), \ldots$ such that for each $n$ we have $A_n \preceq L'$.*
2. Completeness with respect to the existence of a solution for EE.
   *If a solution to* EE *exists than $\mathcal{T}^{\mathcal{M},s_0,\eta}$ is open.*

**Proof.**

1. To prove this item we build $\pi$ inductively, so that each step of the construction satisfies the following invariants:
   **a.** For any $i$, $A_i \preceq L'$.
   **b.** For any tableau state $(s_i, \Delta_i, A_i)$ in $\pi$, we have $\sigma, i \models \Delta_i$.
   Here is the construction. The initial node is the state $(s_0, \Delta_0, A_{\Delta_0})$ defined as follows. The state description $\Delta_0$ is obtained by applying Exp to the original prestate $(s_0, \Gamma_0, A_{\Gamma_0})$ where $\Gamma_0 := \{\eta\} \cup \mathsf{Litt}(s_0, ecl(\eta))$ and the initial annotation function $A_{\Gamma_0}$ coincides with $L$. By the assumption that $\sigma$ satisfies $\eta$ and by definition of $\mathsf{Litt}(s_0, ecl(\eta))$, we have that $\sigma, 0 \models \Gamma_0$. Then we choose $\Delta_0$ to be a full expansion of $\Gamma_0$ such that $\sigma, 0 \models \Delta_0$. The annotation function $A_{\Delta_0}$ is determined by $\Delta_0$, and by construction obviously $A_{\Delta_0} \preceq L'$.

   Now, suppose the sub-path $\pi_j$ of $\pi = (s_0, \Delta_0, A_0), \cdots, (s_j, \Delta_j, A_j)$ has been already defined. If $\Delta_j$ contains a still pending eventuality $e = \varphi \,\mathsf{U}\, \psi$ by the hypothesis on $\sigma$ we have $\sigma, j \models e$. Therefore there is finite initial segment of $\sigma$, say $s_0, \cdots s_j, s_{j+1}, \cdots s_k$ , for some

$k > j$, such that $\sigma, k \models \psi$ and for any $r$ in $[j, ..., k-1]$ we have $\sigma, r \models \varphi$. Then the tableau sub-path $\pi_k = (s_0, \Delta_0, A_0), \cdots, (s_j, \Delta_j, A_j), (s_{j+1}, \Delta_{j+1}, A_{j+1}) \cdots, (s_k, \Delta_k, A_k)$ can be obtained by choosing in the suitable way (via applications of Next) the corresponding prestates $(s_{j+1}, \Gamma_{j+1}, A_{j+1}), ..., (s_k, \Gamma_k, A_k)$ in the pretableau, and choosing the suitable full expansions of their $\Gamma$-components. It is worthwhile observing that each given $\Gamma_{j+i}$ component determines its corresponding annotation function $A_i$ (while Next does not produce updates of the annotation function). If $\Delta_j$ contains no pending eventuality then any tableau sub-path having $\pi_j$ has initial sub-path will do.

This completes the inductive definition of $\pi$. Let us observe that the invariants 1a and 1b hold for $\pi$ and a third property holds: any eventuality $e$ in a description component $\Delta_i$ is realized on a finite sub-path of $\pi$. These three properties of $\pi$ imply that no state in $\pi$ is removed by any application of the rules StateElim1 or StateElim2. In fact, a routine induction on the number of rounds in the elimination phase suffices to prove this claim. Hence the defined $\pi$ survives to the elimination procedure and, as required, for each $n$ we have $A_n \preceq L'$.

2. This claim of the theorem follows from the previous one. ◀

▶ **Corollary 32** (Soundness and completeness of the tableau). *Let $\mathscr{M} = (S, R, L)$ be a partial model, $s_0 \in S$ a state in it and $\eta$ an* LTL *formula. Then the tableau $\mathscr{T}^{\mathscr{M}, s_0, \eta}$ is open if and only if the answer to the problem* EE *is YES.*

## 6 Harvest: solving all problems

So far we have only considered the problems EE and $\mathsf{EE}^{\mathsf{S}}$ 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 leave out the routine details and 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 algorithm D-$\mathsf{EE}^{\mathsf{S}}$ (Section 4.2.2) on $(\mathscr{M}, s_0, \neg\eta)$ generates *all* total extensions where $\neg\eta$ is existentially true at $s_0$ (Thm. 31, claim 1).
      - If there is at least one (amongst the finitely many) total extension $\mathscr{M}'$ of $\mathscr{M}$ that is not returned by D-$\mathsf{EE}^{\mathsf{S}}$, this means that $\neg\eta$ is not existentially true in $\mathscr{M}'$ at $s_0$. Hence, $\eta$ 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 the case of transition extensions is representative enough for that more general case.) First, the tableau based method presented in Section 4.1 is modified to solve EE and $\mathsf{EE}^{\mathsf{S}}$ 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 $\eta$ 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 $(s, \Delta, A_\Delta)$, 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 $\mathsf{EE}^{\mathsf{S}}$ 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 $\mathsf{EE}^{\mathsf{S}}$ is solved just like in the case of label extensions, with an algorithm generating all (finitely many) total transition extensions satisfying $\eta$. 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 $\eta$ 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 $\eta$.

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 $\eta$, 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 $\eta$ iff it satisfies $\eta \wedge l_{\mathscr{M}}(s_0, \eta)$. On the other hand, any linear LTL model $\sigma$ 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 $\sigma$ 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 EE. Thus, EE 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. [11, 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 EE is sought, in the 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, respectively universal truth of $\eta$. 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 $\eta$ solving the problem EA 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 of LTL satisfiability, cf. e.g., [11, Chapter 6]) initially with labels assigning ? to all atomic propositions, and then solving the EA 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 EA 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. [11, 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 $\mathsf{EE}^{\mathsf{S}}$ 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 [20], 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 [20], 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 [13] two distinct, though related, problems are studied, both named *Generalized Model checking* (LTL GMC). The first problem had already been introduced in [9] and amounts to deciding, given a transition system $\mathscr{M}$ where the state labels are 3-valued, and an LTL formula $\eta$, whether there exists a 2-valued partial transition system $\mathscr{M}'$ that is "more complete" than $\mathscr{M}$ and that universally satisfies $\eta$, *i.e.* all the paths issued from its initial state satisfy $\eta$. 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 $\eta$, whereas the above described LTL GMC problem asks for the existence of *some* model $\mathscr{M}'$ satisfying $\eta$, 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 [13]) is based on a simpler pre-order on partial transition systems (previously suggested in [9]). It is less directly related to our work, so we will not discuss it.

Other, conceptually or technically related earlier works include:

- [18], taking an algebraic approach to the completion of partial first-order models, again representing partial information about the actual world;
- [2] (see also, [3], [10]) 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.
- [15], 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* [21], [8] and Gabbay's *reactive Kripke semantics* [12]. 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].

## 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 [19]. 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 R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, 2002.

2 Henrik Reif Andersen. Partial model checking (extended abstract). In *Proc. of LICS 1995*, pages 398–407. IEEE Computer Society, 1995.

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

4 Carlos Areces, Raul Fervari, and Guillaume Hoffmann. Relation-changing modal operators. *Logic Journal of the IGPL*, 23(4):601–627, 2015.

5 Carlos Areces, Raul Fervari, Guillaume Hoffmann, and Mauricio Martel. Undecidability of relation-changing modal logics. In *Proc. of DALI 2017*, volume 10669 of *LNCS*, pages 1–16. Springer, 2017.

6 Carlos Areces, Raul Fervari, Guillaume Hoffmann, and Mauricio Martel. Satisfiability for relation-changing logics. *J. Log. Comput.*, 28(7):1443–1470, 2018.

7 Guillaume Aucher, Philippe Balbiani, Luis Fariñas del Cerro, and Andreas Herzig. Global and local graph modifiers. *Electr. Notes Theor. Comput. Sci.*, 231:293–307, 2009.

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

9 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. `doi:10.1007/3-540-44618-4_14`.

10 Gabriele Costa, David A. Basin, Chiara Bodei, Pierpaolo Degano, and Letterio Galletta. From natural projection to partial model checking and back. In *Proc. of TACAS 2018*, volume 10805 of *LNCS*, pages 344–361. Springer, 2018.

11 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`.

12 Dov M. Gabbay. *Reactive Kripke Semantics*. Cognitive Technologies. Springer, 2013.

13 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. `doi:10.1007/978-3-540-93900-9_11`.

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

15 Orna Kupferman and Moshe Y. Vardi. Synthesis with incomplete informatio. In Howard Barringer, Michael Fisher, Dov Gabbay, and Graham Gough, editors, *2nd Intern. Conf. on Advances in Temporal Logic*, pages 109–127, Dordrecht, 2000. Springer Netherlands.

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

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

18 Bozena Staruch and Bogdan Staruch. First order theories for partial models. *Studia Logica*, 80(1):105–120, 2005.

19 Colin Thomas, Maximilien Cosme, Cédric Gaucherel, and Franck Pommereau. Model-checking ecological state-transition graphs. *PLoS Comput. Biol.*, 18(6), 2022. `doi:10.1371/journal.pcbi.1009657`.

20 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.

21 Johan van Benthem. An essay on sabotage and obstruction. In *Mechanizing Mathematical Reasoning, Essays in Honor of Jörg H. Siekmann on the Occasion of His 60th Birthday*, volume 2605 of *LNCS*, pages 268–276. Springer, 2005.

22 Pierre Wolper. The tableau method for temporal logic: An overview. *Logique et Analyse*, 28:119–136, 1985.