Tableaux-based decision method for single-agent linear time synchronous temporal epistemic logics with interacting time and knowledge (Technical report*)

Mai Ajspur[1] and Valentin Goranko[2]

[1] Roskilde University, ajspur@ruc.dk,
[2] Technical University of Denmark and University of Johannesburg, vfgo@imm.dtu.dk

Abstract. Temporal epistemic logics are known, from results of Halpern and Vardi, to have a wide range of complexities of the satisfiability problem: from PSPACE, through non-elementary, to highly undecidable. These complexities depend on the choice of some key parameters specifying, inter alia, possible interactions between time and knowledge, such as synchrony and agents' abilities for learning and recall. In this work we develop practically implementable tableau-based decision procedures for deciding satisfiability in single-agent synchronous temporal-epistemic logics with interactions between time and knowledge. We discuss some complications that occur, even in the single-agent case, when interactions between time and knowledge are assumed and show how the method of incremental tableaux can be adapted to work in EXPSPACE, respectively 2EXPTIME, for these logics, thereby also matching the upper bounds obtained for them by Halpern and Vardi.

1 Introduction

Knowledge and time are among the most important aspects of agency. Various *temporal-epistemic logics*, proposed as logical frameworks for reasoning about these aspects of single- and multi-agent systems were actively studied in a number of publications during the 1980's, eventually summarized and uniformly presented in a comprehensive study by Halpern and Vardi [4]. In [4], the authors considered several essential characteristics of temporal-epistemic logics: *one vs. several agents*, *synchrony vs. asynchrony*, *(no) learning*, *(no) forgetting (aka, perfect recall or no recall)*, *linear vs. branching time*, and existence (or not) of a *unique initial state*. Based on these, they identified and analyzed a total of 96 temporal-epistemic logics and obtained lower bounds for the complexity of a satisfiability problem in them. In [5] matching upper bounds were claimed for all, and established for most of these logics. It turned out that most of the logics that involve more than one agents, *whose knowledge interacts with time* (e.g., who do not learn or do not forget) – are undecidable (with common knowledge), or decidable but with non-elementary time lower bound (without common knowledge). Even in the single-agent case the interaction between knowledge and time proved to be quite costly, pushing the complexities of deciding satisfiability up to EXPSPACE and 2EXPTIME. These complexity lower bounds were established in [4] and the matching upper bounds are claimed and proved for all synchronous cases in [5]. For the single-agent synchronous cases, these results follow from the non-elementary upper bounds for the multi-agent cases. However, we are not aware of *both optimal and practically implementable* decision methods developed for these logics so far (but, see further discussion on related work). By "practically implementable decision method" we mean one that would only hit the worst case complexity in 'really bad'

* This is the full version, including a technical appendix, of [1]

cases – usually seldom occurring in practice – but would perform reasonably well in most of the practically occurring input instances, whereas a non-practically implementable method is one that essentially always – or, always when the answer is e.g., 'no' – would perform with the theoretically worst case complexity. For instance, the method of semantic tableau for testing tautologies in classical propositional logic is practically implementable, whereas the method using explicitly constructed truth-tables is not.

In this paper we develop such theoretically optimal and practically implementable (modulo the established complexities, of course) tableau-based procedures for deciding satisfiability in single-agent synchronous temporal-epistemic logics with interactions between time and knowledge, by building on the incremental tableau construction, described in [3] for both synchronous and asynchronous multi-agent temporal-epistemic logics with common and distributed knowledge, but with no interactions between time and knowledge (other than synchrony). The method developed there works in EXPTIME, which is the optimal complexity for the logics considered there. It was not clear whether and how that method could be adapted to produce optimal decision procedures for the cases of interacting time and knowledge, where complications arise even in the single-agent case. Here we discuss and illustrate these complications and then extend and adapt the incremental tableaux-based decision method to the single-agent case over linear time synchronous systems, for all cases of interaction between knowledge and time involving combinations of assumptions of 'no learning', 'no forgetting' and 'unique initial state', that are not easily covered by the tableau method from [3]. The basic procedure developed here works in 2EXPTIME and we describe in Section 7 how it is optimized to work in EXPSPACE, thereby also matching the EXPSPACE upper bounds obtained for these logics in [5]. For better readability we put some technical details and proofs in a technical appendix and omit the more routine ones.

In order to delineate the contribution of this paper, we should put it in the context of related works. On the one hand, as discussed above, Halpern and Vardi have established in [5] theoretically optimal upper bounds (for the multi+agent cases), by means of essentially combinatorial estimates of the size of 'small' tree-like models satisfying models, but that proof is far from a practically implementable method as it requires enumerating and checking all models within the prescribed size. On the other hand, a non-optimal, yet apparently implementable tableau method for the cases of single-agent synchronous temporal-epistemic logics with no learning or with no forgetting is developed by Dixon et al in [2], where many of the concepts used here (states, pre-bubbles and bubbles, etc.) have close analogies. That method works by first transforming the input formula into a certain clausal normal form that employs a number of new atoms, used for renaming of subformulae, and then applying a tableau-like method to the resulting set. It uses double exponential space in the number of logical connectives (except negations) in the formula and does not cover the cases with unique initial state. A resolution based approach to the logics has been developed in [7], while [6] develops tableaux for first-order temporal logics, covering (under constant domain assumption) some single-agent temporal epistemic logics, too.

The tableau method developed here originates from the incremental tableaux, first developed by Pratt for PDL [8] and later by Wolper for LTL [9], and is an adaptation of the tableau for the linear time multi-agent temporal epistemic logic with no time-knowledge interaction in [3], to which we refer the reader for further references. Besides combining optimality and implementability, we believe that our tableau method is also somewhat more intuitive and more flexible and amenable to further extensions, incl. covering multi-agent logics and asynchrony.

2 Preliminaries

For lack of space, we only provide here the very basic preliminaries on the logics under consideration and on the incremental tableau method. For further details the reader is referred to [4], [5], [3].

2.1 The Single-agent Linear Time Temporal Epistemic Logic $\mathsf{TEL}^1(\mathsf{LT})$

Syntax and semantics The language of $\mathsf{TEL}^1(\mathsf{LT})$ contains a set $\mathtt{AP}$ of atomic propositions, the Booleans $\neg$ ("not") and $\wedge$ ("and"), the temporal operators $\mathcal{X}$ ("next") and $\mathcal{U}$ ("until") of the logic LTL, as well as the epistemic operator $\mathbf{K}$. The formulas of $\mathsf{TEL}^1(\mathsf{LT})$ are defined as follows:

$$\varphi := p \mid \neg\varphi \mid (\varphi_1 \wedge \varphi_2) \mid \mathcal{X}\varphi \mid (\varphi_1 \mathcal{U} \varphi_2) \mid \mathbf{K}\varphi$$

where p ranges over $\mathtt{AP}$. All other standard Boolean and temporal connectives can be defined as usual. Formulas of the type $\mathbf{K}\varphi$ or their negations will be called *knowledge formulas* and formulas of the type $\mathcal{X}\varphi$, $\varphi_1 \mathcal{U} \varphi_2$ or their negations will be called *temporal formulas*. We omit parentheses when this does not result in ambiguity.

Definition 1 (Temporal-epistemic frames and models). *A* (single-agent) temporal-epistemic frame *(TEF) is a tuple $\mathfrak{S} = (S, R, \mathcal{R})$, where S is a non-empty set of* states*; $R \subseteq S^{\mathbb{N}}$ is a non-empty set of* runs*; and $\mathcal{R} \subseteq (R \times \mathbb{N})^2$ is an equivalence relation, representing the epistemic uncertainty of the agent. A* temporal-epistemic model *(TEM) is a tuple $\mathcal{M} = (\mathfrak{F}, L)$, where $\mathfrak{F}$ is a TEF and $L : R \times \mathbb{N} \to \mathcal{P}(\mathtt{AP})$ is a* labeling function.

We denote the set $R \times \mathbb{N}$ by $P(\mathfrak{S})$. An element $(r, n) \in P(\mathfrak{S})$ is called a *point*. We note that the points and not the states of a model are the elements of interest in a model, since both the epistemic relation and the labelling function are defined with respect to points and not states.

Truth and satisfiability Truth of formulas at a point of a TEM is defined recursively as usual, by combining the semantics for LTL and that of the standard epistemic logic:

$\mathcal{M}, (r, n) \Vdash \mathcal{X}\varphi$ iff $\mathcal{M}, (r, n+1) \Vdash \varphi$;

$\mathcal{M}, (r, n) \Vdash \varphi \mathcal{U} \psi$ iff $\mathcal{M}, (r, i) \Vdash \psi$ for some $i \geq n$ such that $\mathcal{M}, (r, j) \Vdash \varphi$ for every $n \leq j < i$;

$\mathcal{M}, (r, n) \Vdash \mathbf{K}\varphi$ iff $\mathcal{M}, (r', n') \Vdash \varphi$ for every $((r, n), (r', n')) \in \mathcal{R}$;

A formula φ is *satisfiable* (resp., *valid*) if $\mathcal{M}, (r, n) \Vdash \varphi$ for some (resp., every) TEM M and a point (r, n) in it. Satisfiability and validity in a class of models is defined likewise.

Remark 1. Our definition of a temporal epistemic frame (and model) can obviously be extended to the multiagent case by including a relation $\mathcal{R}_a$ for all agents and extending the definition of truth and satisfiability in a model similarly. Note that in both the multi- and single agent case, our notion of temporal-epistemic model is somewhat more general than the semantical structure in [4] and [5], called an 'interpreted system'. In an interpreted system the (global) system states are tuples of local states of all agents, and runs are defined as functions from $\mathbb{N}$ to the set of global states. The epistemic relations in [4] and [5] are defined *between states, not between points*, although one could infer otherwise from the notation used there: $(r, n) \sim_a (r', n')$, but that means by definition $r(n) \sim_a r'(n')$, which again is defined to

mean that the current local state of $r(n)$ and $r'(n')$ w.r.t. agent a is the same. Thus, every interpreted system as defined in [4] and [5] can obviously be redefined as a TEM in our sense, by lifting the epistemic relation from states to points. Our semantics, where runs are defined as more abstract entities, is mentioned in [4] and [5] too, and it is stated in these papers that the semantics are equivalent, but without providing arguments. This is true since every TEM $\mathcal{M}$ in our sense can be *transformed* into an equi-satisfiable interpreted system $\widehat{\mathcal{M}}$, where the local states for each agent are the respective equivalence classes of the points in $\mathcal{M}$. Then, for each run r in $\mathcal{M}$ a corresponding run r' in $\widehat{\mathcal{M}}$ is defined canonically, and the labeling from $\mathcal{M}$ is transferred canonically to $\widehat{\mathcal{M}}$. If there are no r and s in $\mathcal{M}$ for which $((r,n),(s,m)) \in \mathcal{R}_a$ for all agents, then there is a bijection between the runs in $\mathcal{M}$ and $\widehat{\mathcal{M}}$, and it is easy to show that satisfiability of formulas in $\mathcal{M}$ and $\widehat{\mathcal{M}}$ coincide. If there are such runs r and s one can do a 'trick', by adding a new starting point to each new run r' which is unique for r; these new starting points can then be labelled with $\emptyset$. Thus, the notions of satisfiability (at some point of a run in some model) in both semantics coincide.

2.2 Some important properties of temporal-epistemic models

Definition 2 (Properties of TEF and TEM). *A TEF $\mathfrak{S} = (S, R, \mathcal{R})$ has the property of:*

- *Indistinguishable_Initial_States (iis), if for all runs $r, r' \in R$, $((r,0),(r',0)) \in \mathcal{R}$.*
- *No_Learning (nol), if whenever $((r,n),(r',n')) \in \mathcal{R}$, for every $k \geq n$ there exists a $k' \geq n'$ such that $((r,k),(r',k')) \in \mathcal{R}$.*
- *No_Forgetting (nof), if whenever $((r,n),(r',n')) \in \mathcal{R}$, for every $0 \leq k \leq n$ there exists a $0 \leq k' \leq n'$ such that $((r,k),(r',k')) \in \mathcal{R}$.*
- *Synchrony (sync), if $((r,n),(r',n')) \in \mathcal{R}$ implies $n = n'$.*

A TEM $\mathcal{M} = (\mathfrak{F}, L)$ has the property of $x \in \{\mathsf{nol}, \mathsf{nof}, \mathsf{sync}, \mathsf{iis}\}$ if $\mathfrak{F}$ does so.

The meaning of iis is clear; sync means that the agent can perceive time, i.e., has a clock; nol means that the agent does not learn over time in the sense that if it cannot distinguish two runs at any given time instance, it will not be able to do so later on. Likewise, nof means that if at a given time instance the agent can tell two different runs apart, the agent must have been able to do so at any previous time instance. We notice that if a TEF or TEM has the properties nol and iis, then it follows that it has the property nof too.

We denote the classes of all TEMs satisfying property $x \in \{\mathsf{nol}, \mathsf{nof}, \mathsf{sync}, \mathsf{iis}\}$ by TEM_x, and we further denote $\mathrm{TEM}_X = \cap_{x \in X} \mathrm{TEM}_x$ for $X \subseteq \{\mathsf{nol}, \mathsf{nof}, \mathsf{sync}, \mathsf{iis}\}$.

Extensions of $\mathsf{TEL}^1(\mathsf{LT})$ We denote the extension of the logic $\mathsf{TEL}^1(\mathsf{LT})$ with semantics restricted to the class TEM_X, by $\mathsf{TEL}^1(\mathsf{LT})_X$ for all $X \subseteq \{\mathsf{nol}, \mathsf{nof}, \mathsf{sync}, \mathsf{iis}\}$. Thus, validity/ satisfiability in $\mathsf{TEL}^1(\mathsf{LT})_X$ means validity/satisfiability in a model from TEM_X. In this paper we focus on the logics for synchronous models $\mathsf{TEL}^1(\mathsf{LT})_X$ (i.e. $\mathsf{sync} \in X$) and either $\mathsf{nol} \in X$ or $\mathsf{nof} \in X$. For convenience, we also denote $\mathsf{TEL}^1(\mathsf{LT}) = \mathsf{TEL}^1(\mathsf{LT})_\emptyset$ in cases where we want to emphasize that no interaction conditions are assumed.

Remark 2. The properties of sync, iis, nol and nof are all preserved when transforming a (single- or multi-agent) interpreted system into a model as in Remark 1. On the other hand, consider a (single- or multi-agent) model $\mathcal{M}$. If there are no two runs r and s in $\mathcal{M}$ such that $((r,n),(s,n)) \in \mathcal{R}_a$ for all agents a, then the transformation described in Remark 1 without the additional 'trick' preserves the properties of sync, uis, nol and nof. If there are such r

and s, one can in most cases of combinations of the interaction properties perform suitable modifications that enable us to still find an equi-satisfiable interpreted system with the same interaction properties.

However, this is not the case for the single- or multiagent temporal epistemic logic with interaction properties X, where $X \in \{ \{\mathsf{sync}, \mathsf{nol}, \mathsf{nof}\}, \{\mathsf{sync}, \mathsf{nol}, \mathsf{nof}, \mathsf{iis}\}, \{\mathsf{sync}, \mathsf{nol}, \mathsf{iis}\}, \{\mathsf{sync}, \mathsf{nof}, \mathsf{iis}\} \}$. In these cases the two semantics differ, since e.g. the formula $\theta = p \wedge \neg \mathbf{K}_a p$ is satisfiable in our semantics, while it is unsatisfiable in the semantics presented in [4] and [5]. This is due to the fact that the interaction properties implies that if $\mathcal{M}, (r, n) \Vdash \theta$ for a point (r, n) in the interpreted system $\mathcal{M}$, then if $((s, n), (r, n)) \in \mathcal{R}_a$ then $((s, n'), (r, n')) \in \mathcal{R}_a$ for all n', and hence $r = s$ since R according to the definition is a *set* of runs. Thus we must have that $\mathcal{M}, (r, n) \Vdash p, \neg p$ which is a contradiction. Note, however, that the formula *is* satisfiable if e.g. the property sync is dropped.

3 Temporal Epistemic Hintikka Structures

Even though we are ultimately interested in testing formulas of $\mathsf{TEL}^1(\mathsf{LT})$ for satisfiability in a TEM, the tableau procedure we will present here tests for satisfiability in a more general kind of semantic structures, namely *temporal epistemic Hintikka structures* (TEHS). The important aspect of a Hintikka structure for a formula θ is that it contains just as much semantic information about the satisfying model of θ as it is necessary, and no more. More precisely, while a TEM provides the truth value of every formula of the language at every state, a Hintikka structure only determines the truth of formulas that are directly involved in the evaluation of the input formula θ.

Some terminology: we distinguish *conjunctive formulas*, also called $\curlywedge$*-formulas* and *disjunctive formulas*, also called $\curlyvee$-formulas, each with a respective set of components, as in the tables below:

$\curlywedge$-formula	Set of $\curlywedge$-components
$\neg\neg\varphi$	$\{\varphi\}$
$\varphi \wedge \psi$	$\{\varphi, \psi\}$
$\mathbf{K}\varphi$	$\{\mathbf{K}\varphi, \varphi\}$

$\curlyvee$-formula	Set of $\curlyvee$-components
$\neg(\varphi \wedge \psi)$	$\{\neg\varphi, \neg\psi\}$
$(\varphi \mathcal{U} \psi)$	$\{\psi, \varphi \wedge \mathcal{X}(\varphi \mathcal{U} \psi)\}$
$\neg(\varphi \mathcal{U} \psi)$	$\{\neg\psi \wedge \neg\varphi, \neg\psi \wedge \neg\mathcal{X}(\varphi \mathcal{U} \psi)\}$

It can be easily shown, that any $\curlywedge$-formula is equivalent to the conjunction of its $\curlywedge$-components, and that any $\curlyvee$-formula is equivalent to the disjunction of its $\curlyvee$-components.

Definition 3 (Fully expanded sets). *A set of formulae Δ of $\mathsf{TEL}^1(\mathsf{LT})$ is* fully expanded *if:*

1. *Δ is not* patently inconsistent*, i.e. if $\varphi \in \Delta$ then $\neg\varphi \notin \Delta$.*
2. *If $\alpha \in \Delta$ is a $\curlywedge$-formula, then all of its $\curlywedge$-components are in Δ,*
3. *If $\beta \in \Delta$ is a $\curlyvee$-formula, then at least one of its $\curlyvee$-components are in Δ,*

Definition 4 (Temporal Epistemic Hintikka Structure). *A* temporal-epistemic Hintikka structure *(TEHS) is a tuple $(\mathfrak{H}, H)$, where $\mathfrak{H} = (S, R, \mathcal{R})$ is a TEF, and H is a labeling of points in $P(\mathfrak{H})$ with sets of formulae, satisfying the following conditions, for all $(r, n) \in P(\mathfrak{H})$:*

1. *$H(r, n)$ is* fully expanded*.*
2. *If $\neg\mathbf{K}\varphi \in H(r, n)$ then $\neg\varphi \in H(r', n')$ for some $(r', n') \in P(\mathfrak{H})$ such that $((r, n), (r', n')) \in \mathcal{R}$.*

3. *If* $((r,n),(r',n')) \in \mathcal{R}$, *then* $\mathbf{K}\varphi \in H(r,n)$ *iff* $\mathbf{K}\varphi \in H(r',n')$.
4. *If* $\mathcal{X}\varphi \in H(r,n)$, *then* $\varphi \in H(r,n+1)$ *and if* $\neg\mathcal{X}\varphi \in H(r,n)$, *then* $\neg\varphi \in H(r,n+1)$.
5. *If* $\varphi\,\mathcal{U}\,\psi \in H(r,n)$, *then there exists* $i \geq n$ *such that* $\psi \in H(r,i)$ *and* $\varphi \in H(r,j)$ *holds for every* $n \leq j < i$.

$(\mathfrak{H}, H)$ *has the property* $x \in \{\mathsf{nol}, \mathsf{nof}, \mathsf{sync}, \mathsf{iis}\}$ *if* $\mathfrak{H}$ *has the property* x.

It was proved in [3] that any temporal-epistemic formula of the multi-agent linear time temporal epistemic logic with synchrony but without interaction of time and knowledge, is satisfiable in a TEM iff it is satisfiable in a TEHS. Adding any combination of the interaction conditions nol, nof or iis does not affect the truth of this claim in the 1-agent case, so we can from now on restrict attention to satisfiability in TEHS.

4 Tableaux for synchronous $\mathsf{TEL}^1(\mathsf{LT})$ with interaction conditions

4.1 Overview of the tableau procedure for $\mathsf{TEL}^1(\mathsf{LT})_\emptyset$

The tableaux method for testing the satisfiability of an input formula θ of $\mathsf{TEL}^1(\mathsf{LT})_\emptyset$ is used as a starting point for the procedure for $\mathsf{TEL}^1(\mathsf{LT})_X$ where $X \neq \emptyset$. To aid the presentation of the procedure for $\mathsf{TEL}^1(\mathsf{LT})_X$, we first outline the essentials of the tableau procedure for $\mathsf{TEL}^1(\mathsf{LT})_\emptyset$ developed for multi-agent case in [3]; the reader is referred to the latter for more detail.

The tableaux procedure for $\mathsf{TEL}^1(\mathsf{LT})_\emptyset$ consists of three major phases: *pretableau construction*, *prestate elimination*, and *state elimination*. It constructs a directed graph $\mathcal{T}^\theta$ (called a *tableau*) with nodes labelled by finite sets of formulas, and directed edges between nodes, representing temporal, epistemic, or label-expansion relations.

The pretableau construction phase produces the so-called the *pretableau* $\mathcal{P}^\theta$ for the input formula θ, where the nodes are of two kinds: *states* and *prestates*. States are fully expanded sets, meant to represent states of a TEHS, while prestates are finite sets of formulas and play a temporary role in the construction of $\mathcal{T}^\theta$. The pretableau phase consists of alternative constructions of epistemic and temporal successor prestates of a given state, and expanding a given prestate Γ into fully expanded sets, denoted by $\mathbf{states}(\Gamma)$, which label new or existing states.

The prestate elimination phase creates a smaller graph $\mathcal{T}^\theta_0$ out of $\mathcal{P}^\theta$, called the *initial tableau for* θ, by eliminating all the prestates from $\mathcal{P}^\theta$ and accordingly redirecting its edges.

Finally, the state elimination phase removes, in successive steps, all the states, if any, that cannot be satisfied in a TEHS, because they lack necessary successors (epistemic or temporal) or because they contain unrealized eventualities. When no more states can be removed, the elimination procedure produces a (possibly empty) subgraph $\mathcal{T}^\theta$ of $\mathcal{T}^\theta_0$, called the *final tableau for* θ. If some state Δ of $\mathcal{T}^\theta$ contains θ, the procedure declares θ satisfiable and a TEHS satisfying θ can be extracted from it; otherwise it declares θ unsatisfiable.

4.2 Complications arising with interacting temporal and epistemic operators

In the tableau construction for the basic logic $\mathsf{TEL}^1(\mathsf{LT})_\emptyset$, when identifying the set of formulas that must be put in the label of a temporal successor-prestate Γ for a state Δ, the procedure only has to take into account formulas that come from Δ. When the logic assumes time-knowledge interaction, e.g. nol, this is no longer the case because there will also be formulas

coming from other states that are epistemically related to the immediate predecessor state Δ, that will be relevant for defining the successor (pre)state Γ. For instance, if two states are epistemically related, then they need respective successors that are epistemically related, and therefore it is necessary that these successors contain the same knowledge formulas. Likewise for the logic assuming nof: if a state (that is not in the 'first' temporal layer) is epistemically related to another state, then both states need to have predecessor-states which are also epistemically related. Therefore, the procedure has to create enough states at any temporal layer so that the states needed in the next temporal layer have respective predecessor states.

4.3 Bubbles and bubble-paths

Here we call any set of formulas Δ of $\mathsf{TEL}^1(\mathsf{LT})$ a *prestate*. A fully expanded prestate will be called a *state*. For any set of formulas Γ, we denote by $\mathbf{states}(\Gamma)$ the set of full expansions of Γ that are produced by the tableau- procedure for $\mathsf{TEL}^1(\mathsf{LT})_\emptyset$.

We let $K(\Delta) := \{\, \mathbf{K}\varphi \mid \mathbf{K}\varphi \in \Delta \,\}$, $\mathrm{Epi}(\Delta) := K(\Delta) \cup \{\, \neg\mathbf{K}\varphi \mid \neg\mathbf{K}\varphi \in \Delta \,\}$ and $\mathrm{Next}(\Delta) := \{\, \varphi \mid \mathcal{X}\varphi \in \Delta \,\} \cup \{\, \mathtt{neg}\varphi \mid \neg\mathcal{X}\varphi \in \Delta \,\}$, where $\mathtt{neg}\varphi = \varphi$ if the main-connective of φ is $\neg$ and $\mathtt{neg}\varphi = \neg\varphi$ otherwise. Note that in the tableaux for $\mathsf{TEL}^1(\mathsf{LT})_\emptyset$, the set $\{\neg\varphi\} \cup \mathrm{Epi}(\Delta) \setminus \{\neg\mathbf{K}\varphi\}$ is the epistemic successor prestate for the state Δ created for the diamond formula $\neg\mathbf{K}\varphi \in \Delta$, while $\mathrm{Next}(\Delta)$ is the temporal prestate created for Δ.

In order to deal with the complications discussed above, the tableau procedure presented here will act *not on single states* but on special kinds of sets of states representing possible epistemic clusters, which we will call *bubbles*, formally defined below. Any finite set of states will be called a *pre-bubble*.

Definition 5 (Bubbles). *A* bubble B *is a pre-bubble such that:*

- *for all $\Delta \in B$ and all $\neg\mathbf{K}\varphi \in \Delta$ there exists a $\Delta' \in B$ such that $\neg\varphi \in \Delta'$.*
- *B is* knowledge-consistent*, i.e. $K(\Delta) = K(\Delta')$ for all $\Delta, \Delta' \in B$.*

Definition 6 (Successor- and predecessor-sets). *A set of states S is a* successor-set *for a bubble B if for all $\Delta \in B$ there is a $\Delta' \in S$ s.t. $Next(\Delta) \subseteq \Delta'$. In that case, we write $B \rightarrow_{\forall\exists} S$. Respectively, S is a* predecessor-set *for B if for all $\Delta \in B$ there is a $\Delta' \in S$ s.t. $Next(\Delta') \subseteq \Delta$. In that case we write $S \rightarrow_{-1\forall\exists} B$.*

Definition 7 (Bubble-paths). *A sequence of bubbles $\mathcal{B} = (B_i)_{0\leq i\leq m}$ is a called a bubble-path. It is a* successor-bubble-path*, if $B_i \rightarrow_{\forall\exists} B_{i+1}$ for all $0 \leq i < m$. It is a* predecessor-bubble-path *if $B_i \rightarrow_{-1\forall\exists} B_{i+1}$ for all $0 \leq i < m$.*

A sequence of states $\pi = (\Delta_i)_{0\leq i\leq m}$ is a temporal path *if $Next(\Delta_i) \subseteq \Delta_{i+1}$ for all $0 \leq i < m$. The temporal path $\pi = (\Delta_i)_{0\leq i\leq m}$* follows *the bubble-path $\mathcal{B} = (B_i)_{0\leq i\leq m}$ if $\Delta_i \in B_i$ for all $0 \leq i \leq m$.*

The tableau-procedure will construct bubble-paths. For logics that satisfy nol, these bubble-paths will be successor-bubble-paths, and for logics that satisfy nof, they will be predecessor-bubble-path. If the logic satisfies both nol and nof, the bubble-paths will be both successor-and predecessor-bubble-paths at the same time.

Definition 8 (Realization of eventualities). *Let Δ be a state in a bubble B. Let $\varphi\mathcal{U}\psi$ be an eventuality in Δ. Then $\varphi\mathcal{U}\psi \in \Delta$ is* realized on a bubble-path $\mathcal{B}$ in a tableau $\mathcal{T}$ by a

temporal path π *if* B *equals the first bubble in* $\mathcal{B}$, Δ *equals the first state in* π, π *follows* $\mathcal{B}$, *and there is a subpath* π' *of* π *starting in* Δ, *where* ψ *belongs to the last state of* π', *while* φ *belongs to all previous states in* π'.

We need the next technical notion in order to define satisfaction of a bubble-path in a TEM $\mathcal{M}$.

Definition 9 (State-point-assignment). *Let* $\mathcal{B} = (B_k)_{0 \leq k \leq m}$ *be a bubble-path. Let* $\mathcal{M}$ *be a TEM with a point* (r, n). *Then a* state-point-assignment *for* $\mathcal{M}$, (r, n) *and* $\mathcal{B}$ *is a set*

$$A \subseteq \bigcup_{0 \leq k \leq m} \Big(B_k \times \{ (r', n') \mid ((r', n'), (r, n+k)) \in \mathcal{R} \} \Big).$$

We imagine the states in the bubbles in the specified bubble-path being assigned to points of the model, so that the states in the first bubble are assigned to points epistemically related to the specified point (r, n), and states in the second bubble are assigned to points in $\mathcal{M}$ that are epistemically realted to $(r, n+1)$, and so on.

Definition 10 (Satisfiability of a bubble-path). *Let* $\mathcal{M}$ *be a* TEM_X *and* $(\bar{r}, n)$ *a point in* $P(\mathcal{M})$. *Let* $\mathcal{B} = (B_i)_{0 \leq i \leq m}$ *be a bubble-path. Then we say that* $\mathcal{M}$ satisfies $\mathcal{B}$ at $(\bar{r}, n)$ by A, *and write* $\mathcal{M}, (\bar{r}, n) \Vdash^A \mathcal{B}$, *if* A *is a states-point assignment for* $\mathcal{M}$, $(\bar{r}, n)$ *and* $\mathcal{B}$, *such that*

- $(\Delta, (r, n+k)) \in A$ *implies that* $\mathcal{M}, (r, n+k) \Vdash \Delta$.
- *if* $\mathsf{nol} \in X$ *then for all* $0 \leq k < m$ *and all* $\Delta \in B_k$ *there is a run* r, *such that* $(\Delta, (r, n+k)) \in A$ *and* $(\Delta', (r, n+k+1)) \in A$ *for some* $\Delta' \in B_{k+1}$ *with* $\mathit{Next}(\Delta) \subseteq \Delta'$.
- *if* $\mathsf{nof} \in X$ *then for all* $0 < k \leq m$ *and all* $\Delta \in B_k$ *there is a run* r *such that* $(\Delta, (r, n+k)) \in A$ *and* $(\Delta', (r, n+k-1)) \in A$ *for some* $\Delta' \in B_{k-1}$ *with* $\mathit{Next}(\Delta') \subseteq \Delta'$.

4.4 Construction of the pretableau

For the logic $\mathsf{TEL}^1(\mathsf{LT})_X$ where $\mathsf{sync} \in X$ and either $\mathsf{nol} \in X$ or $\mathsf{nof} \in X$, the procedure is split into three parts: construction of the pretableau, where pre-bubbles and bubbles are added to the tableau; construction of the initial tableau, where the pre-bubbles are removed; and construction of the final tableau, where bubbles are eliminated. The construction of the pretableau for θ works as follows:

1. For all $\Delta \in \mathbf{states}(\{\theta\})$, make $\{\Delta\}$ a pre-bubble in $\mathcal{T}$.
2. Expand each not yet expanded pre-bubble A into bubbles by applying the procedure ExpandPrebubble(A, X), outlined further. For every returned bubble B produce an arrow $A \dashrightarrow B$.
3. Produce temporal successor-prebubbles for each bubble B for which this has not been done so far, by applying the procedure TempSuccessorPrebubbles(B, X) outlined further. Add any such pre-bubble A to $\mathcal{T}$ if it is not already there and produce an arrow $B \mathrel{—\!\!\!\!\!\!\rightarrow\!\!\!\!\!\!—} A$.
4. Repeat step 2 and 3 in cycles until no new bubbles or pre-bubbles are created.

When producing successor-pre-bubbles and expanding pre-bubbles, we use the procedures from the basic algorithm for $\mathsf{TEL}^1(\mathsf{LT})_\emptyset$ for expanding prestates into states and producing temporal successor-prestates and epistemic alternatives for the states in the bubbles. These operations are performed 'on the side', and are not part of the bubble-based tableau construction itself. However, for efficiency we keep the expanded states on the side, so that we e.g. do not have to recompute full expansions of a state. The procedures ExpandPrebubble and

TemporalSuccessorPrebubbles depend on X and are explained below. They use 3 other procedures, which do not depend on X, described first.

MakeKnowledgeConsistent takes a set of states S as input, and returns a set L of all knowledge-consistent 'alternatives' of S. That is, if S' is in the returned set L, then S' is knowledge-consistent and there is a surjective function $f : S \to S'$ such that if $\Delta \in S$ then $\Delta \subseteq f(\Delta)$, i.e. some formulas might be added to every state in S. If S is already knowledge consistent, $L = \{S\}$ is returned. Otherwise, for every state Δ in S we collect in K_Δ the $\mathbf{K}$-formulas in the states in S which are not in Δ, and the $\neg\mathbf{K}$-formulas in the states of S are collected in nK. Then the method returns $L = \{\, \{\Delta_0 \cup \Sigma_0, \ldots, \Delta_n \cup \Sigma_n\} \mid \Sigma_i \in \mathbf{states}(K_{\Delta_i}), \Delta_i \cup \Sigma_i$ and $\Sigma_i \cup nK$ are not patently inconsistent for all $i\,\}$, where $\Delta_0, \ldots, \Delta_n$ are the states in S. However, if there is a $\Delta \in S$ such that $\Delta \cup \Sigma_j$ or $\Sigma_j \cup nK$ is patently inconsistent for all $\Sigma_j \in K_\Delta$, then $\emptyset$ is returned.

The procedure LocalBubble(A, Δ) (where $\Delta \in A$ and A is a prebubble) returns a set L consisting of epistemic alternatives for all formulas $\neg\mathbf{K}\varphi$ in Δ (i.e. diamond formulas), for which there is no $\Delta' \in A$ that contains $\neg\varphi$, i.e. it returns $L = \{\, \{\Delta_0, \ldots, \Delta_n\} \mid \Delta_i \in \mathbf{states}(\Gamma_i)$ for all $i\,\}$, where $\neg\mathbf{K}\varphi_0, \ldots, \neg\mathbf{K}\varphi_n$ are the 'unfulfilled' epistemic diamond formulas in Δ and $\Gamma_i = \{\neg\varphi_i\} \cup \mathrm{Epi}(\Delta) \setminus \{\neg\mathbf{K}\varphi_i\}$ for all i. Though, if any of the sets $\mathbf{states}(\Gamma_i) = \emptyset$, then $\emptyset$ is returned.

ExpandToBubble takes as input a knowledge-consistent prebubble S, and returns a set L consisting of bubbles, each containing S. That set is constructed by first adding S to L, and then repeatedly replacing a set S' in L (which is not marked 'closed') with $S' \cup L_1, \ldots, S' \cup L_n$, where $\{L_1, \ldots, L_n\}$ is the set returned by LocalBubble(S', Δ) for an unmarked $\Delta \in S'$ (after which $\Delta \in S' \cup L_i$ is marked). If LocalBubble(S', Δ) returns $\emptyset$, then S' is marked 'closed'. When each set in L is either marked 'closed' or all states in the set are marked, the 'closed' sets are removed from L, and L is returned.

TemporalSuccessorPrebubbles(B, X) returns a set L of temporal successor pre-bubbles for a bubble B. When $\mathsf{nol} \in X$, all states in B needs to have successor-states in the same bubble, so the method returns $L = \{\, \{\Delta'_0, \ldots, \Delta'_n\} \mid \Delta'_i \in \mathbf{states}(\mathrm{Next}(\Delta_i))$ for all $i\,\}$, where $\Delta_0, \ldots, \Delta_n$ are the states in B. Though, if $\mathbf{states}(\mathrm{Next}(\Delta_i)) = \emptyset$ for any $\Delta_i \in B$, the method returns $\emptyset$. When $\mathsf{nol} \notin X$, the successors of the states in B need not be in the same bubble but there should be a successor-bubble for every state in B. Thus, the returned set is $L = \{\, \{\Delta'\} \mid \Delta' \in \mathbf{states}(\mathrm{Next}(\Delta))$ for a $\Delta \in B\,\}$. If $\mathbf{states}(\mathrm{Next}(\Delta)) = \emptyset$ for any $\Delta \in B$, then $L = \emptyset$ is returned.

When $\mathsf{nof} \in X$ or $\{\mathsf{nol}, \mathsf{iis}\} \subseteq X$, the expansion of a prestate (that is not in the first temporal layer) is done with respect to the immediate predecessor bubble, for which the pre-bubble was created. Thus, in these cases we annotate any created successor-pre-bubble with the bubble that created it, and two pre-bubbles are not considered the same, unless they have the same annotation. There are thus $2^{2 \cdot \#sts_\theta}$ possible pre-bubbles, where $\#sts_\theta$ are the number of possible states belonging to a bubble in the tableau for a formula θ. The number of prebubbles is, however, still double-exponential in the length of the input-formula.

Before describing the next procedure, we note that the expanding procedure for prestates into states in the tableau method for $\mathsf{TEL}^1(\mathsf{LT})_\emptyset$ uses analytic cuts to ensure that if $\Delta \xrightarrow{\neg\mathbf{K}\varphi} \Delta'$ and $\mathbf{K}\psi \in \Delta'$, then $\mathbf{K}\psi \in \Delta$. That is, for any $\alpha \in \Delta$ where $\alpha = \mathbf{K}\psi$ or $\alpha = \neg\mathbf{K}\psi$, if

$\mathbf{K}\varphi \in \mathsf{Sub}(\alpha)$ and there are no $\mathcal{X}$s on the parse tree between α and $\mathbf{K}\varphi$, then $\mathbf{K}\varphi \in \Delta$ or $\neg\mathbf{K}\varphi \in \Delta$.

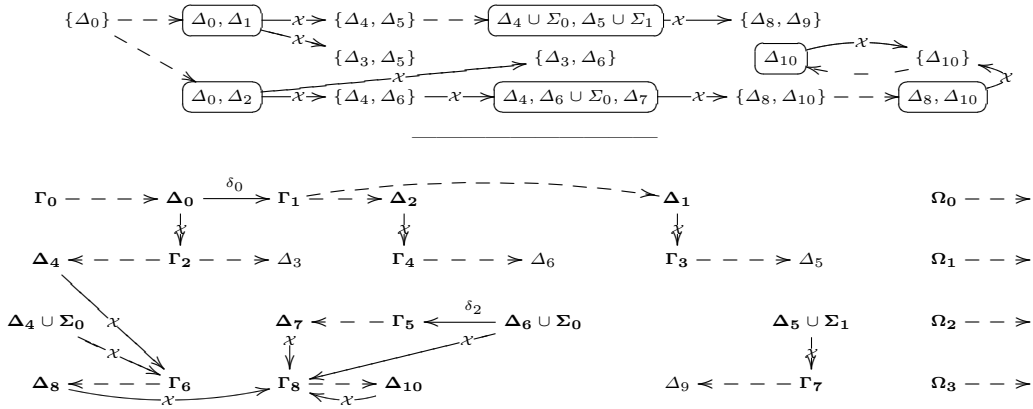
ExpandPrebubble(A, X) works as follows: When $\mathsf{nof} \notin X$ and $\{\mathsf{iis}, \mathsf{nol}\} \not\subseteq X$ (in which case nof is implied) the method first considers all knowledge-consistent versions of A (as returned by MakeKnowledgeConsistent), and then expand these to bubbles (by calls to ExpandToBubble). However, when $\mathsf{nof} \in X$ or $\{\mathsf{iis}, \mathsf{nol}\} \subseteq X$ (in which case nof is implied) things are more complicated. First of all, every state in a bubble B constructed as described above needs to have a predecessor in the bubble B' that created A (i.e. the annotation of A); of course, in the first temporal layer (when the annotation of $A = \emptyset$) this is not required. Secondly, we might later on encounter a state in a bubble that needs predecessors in the bubble B in question, so we have to ensure there are 'enough' states in B. Any state that can possibly be added to B needs to contain, as a minimum, the $\mathbf{K}$-formulas of any other state in B, and thus the states in $\mathbf{states}(K(B))$ contain the 'minimal' formulas for a state belonging to B. Adding a $\Sigma \in \mathbf{states}(K(B))$ with $K(\Sigma) = K(B)$ to B will still yield a bubble, because if Σ contains a diamond-formula $\neg\mathbf{K}\varphi$, then there will be a state in B containing $\neg\mathbf{K}\varphi$ (because of the cuts). In temporal layers different from the first, there might have to be added more formulas to these states, but in any case, these states are denoted as the 'minimal' states of B, B_{min}.

The method thus works as follows. To keep the method working within double-exponential time, it builds a 'mini-tableau' on the side: pre-bubbles A', are expanded into knowledge-consistent pre-bubbles (to which A' is linked by a $\xrightarrow{\text{KC}}$-arrow). The knowledge-consistent pre-bubbles S are then expanded into bubbles (to which S is linked by $\xrightarrow{\text{Bubble}}$). For any of these bubbles $\tilde{B}$, it might be the case that not every state Δ in $\tilde{B}$ has a potential predecessor in B', the bubble creating A, i.e. there is no state $\Delta' \in B'$ where $\text{Next}(\Delta') \subseteq \Delta$. So for all states $\Delta \in \tilde{B}$ that do not have a potential predecessor in B' we take any of the 'minimal' states $\Sigma \in B'_{min}$, and try to make this the predecessor of Δ; this step is of course omitted when expanding prebubbles in the first temporal layer. This is done by modifying a copy of $\tilde{B}$, where each Δ without a predecessor has been replaced with $\Delta \cup \Omega$ for a $\Omega \in \mathbf{states}(\text{Next}(\Sigma))$ where $\Delta \cup \Omega$ is not patently inconsistent. $\tilde{B}$ is then linked to each of these 'copies' with an arrow $\xrightarrow{\text{pred}}$. These pre-bubbles are not necessarily knowledge-consistent, so the outlined steps are repeated. We always reuse pre-bubbles, knowledge-consistent pre-bubbles and bubbles whenever possible. At some point, no new pre-bubbles, knowledge-consistent pre-bubbles or bubbles are produced. The bubbles in the mini-tableau, where all states have predecessors, i.e. $\tilde{B}$s for which $\tilde{B} \underset{\text{pred.}}{\overset{\text{KC}}{\rightleftarrows}} \tilde{B} \xrightarrow{\text{Bubble}} \tilde{B}$, now needs to have the 'minimal' states added to them. For each such bubble $\tilde{B}$ and each $Y \in \mathcal{P}(\{\, \Delta \in \mathbf{states}(K(\tilde{B})) \mid K(\Delta) = K(\tilde{B}) \,\})$ we therefore add $\tilde{B} \cup Y$ to the mini-tableau as a knowledge-consistent pre-bubble, if it is not already there, and we let the states in Y be 'minimal'; if $\tilde{B} \cup Y$ is present with another set of 'minimal' states, we just add the states in Y as 'minimal'. Then we expand the mini-tableau again, until no new pre-bubbles are added. Whenever we add formulas to a 'minimal' state (in making it knowledge-consistent or adding predecessors), we let the modified 'minimal' state be 'minimal' in the resulting pre-bubble. At saturation, we return the bubbles $\tilde{B}$ for which $\tilde{B} \underset{\text{pred.}}{\overset{\text{KC}}{\rightleftarrows}} \tilde{B} \xrightarrow{\text{Bubble}} \tilde{B}$.

We note that the procedures are so constructed that if $B \text{—}\mathcal{X}\!\!\succ A \dashrightarrow \tilde{B}$, then $\tilde{B}$ is a successor of B if $\mathsf{nol} \in X$, and B is a predecessor of $\tilde{B}$ if $\mathsf{nof} \in X$.

The concept of bubbles and our use of them in the tableau procedure is similar to the assignment of states to equialence classes, that the procedure in [2] makes use of. However, our procedure does not require the input formula to be transformed initially, and instead it uses the bubbles as the main entities in the procedure and construct the temporal relation between bubbles (and thereby points) directly such that the required interaction properties will hold for the model that will be extracted from it. The direct use of bubbles as the main enitites of the tableau further allows for an easy adaption of the method to the asynchronous case (i.e. where sync is not required).

Example 1. Figure 1 contains the pretableau for $\theta = \neg\mathcal{X}\Big(\neg(\neg\mathbf{K}p \wedge \mathbf{K}\neg r) \wedge \neg(\mathcal{X}q \wedge \mathbf{K}u)\Big) \wedge \neg\mathbf{K}\Big(\neg\mathcal{X}(\mathcal{X}\mathbf{K}\neg q \wedge \mathbf{K}p) \wedge \neg\mathcal{X}(r \wedge \neg\mathbf{K}\neg v)\Big)$ in $\mathsf{TEL}^1(\mathsf{LT})_{\mathsf{sync,nol}}$. To help readability, the bubbles are framed with rounded boxes while the pre-bubbles are not. The pretableau is the part containing the bubbles, while the part consisting of states and prestates just are intermediate results; here, the expanded (pre)states are marked in bold.



$\delta_0 = \neg\mathbf{K}(\neg\mathcal{X}(\mathcal{X}\mathbf{K}\neg q \wedge \mathbf{K}p)$
$\Gamma_0 = \{\theta\}$
$\Gamma_1 = \{\neg(\neg\mathcal{X}(\mathcal{X}\mathbf{K}\neg q \wedge \mathbf{K}p) \wedge \neg\mathcal{X}(r \wedge \neg\mathbf{K}\neg v))\}$
$\Gamma_2 = \{\neg(\neg(\neg\mathbf{K}p \wedge \mathbf{K}\neg r) \wedge \neg(\mathcal{X}q \wedge \mathbf{K}u))\}$
$\Gamma_3 = \{\mathcal{X}\mathbf{K}\neg q \wedge \mathbf{K}p)\}$
$\Gamma_4 = \{r \wedge \neg\mathbf{K}\neg v\}$
$\Gamma_5 = \{v, \mathbf{K}u\}$
$\Gamma_6 = \{q\}$
$\Gamma_7 = \{\mathbf{K}\neg q\}$
$\Gamma_8 = \{\top\}$

$\Delta_0 = \{\theta, \neg\mathcal{X}(\neg(\neg\mathbf{K}p \wedge \mathbf{K}\neg r) \wedge \neg(\mathcal{X}q \wedge \mathbf{K}u)), \neg\mathbf{K}(\neg\mathcal{X}(\mathcal{X}\mathbf{K}\neg q \wedge \mathbf{K}p) \wedge \neg\mathcal{X}(r \wedge \neg\mathbf{K}\neg v)), \top\}$
$\Delta_1 = \Gamma_1 \cup \{\neg\neg\mathcal{X}(\mathcal{X}\mathbf{K}\neg q \wedge \mathbf{K}p), \mathcal{X}(\mathcal{X}\mathbf{K}\neg q \wedge \mathbf{K}p), \top\}$
$\Delta_2 = \Gamma_1 \cup \{\neg\neg(\neg\mathbf{K}p \wedge \mathbf{K}\neg r), \neg\mathbf{K}p \wedge \mathbf{K}\neg r, \neg\mathbf{K}p, \mathbf{K}\neg r, \neg r, \mathcal{X}\top, \top\}$
$\Delta_3 = \Gamma_2 \cup \{\neg\neg(\neg\mathbf{K}p \wedge \mathbf{K}\neg r), \neg\mathbf{K}p \wedge \mathbf{K}\neg r, \neg\mathbf{K}p, \mathbf{K}\neg r, \neg r, \mathcal{X}\top, \top\}$
$\Delta_4 = \Gamma_2 \cup \{\neg\neg(\mathcal{X}q \wedge \mathbf{K}u), \mathcal{X}q \wedge \mathbf{K}u, \mathcal{X}q, \mathbf{K}u, u, \top\}$
$\Delta_5 = \Gamma_3 \cup \{\mathcal{X}\mathbf{K}\neg q, \mathbf{K}p, p, \top\}$
$\Delta_6 = \Gamma_4 \cup \{r, \neg\mathbf{K}\neg v, \mathcal{X}\top, \top\}$
$\Delta_7 = \{v, \mathbf{K}u, u, \mathcal{X}\top, \top\}$
$\Delta_8 = \{q, \mathcal{X}\top, \top\}$

$\Omega_0 = K(\Delta_4, \Delta_5) \setminus K(\Delta_4) = K(\Delta_3, \Delta_5) \setminus K(\Delta_3) = K(\Delta_4, \Delta_6) \setminus K(\Delta_6) = \{\mathbf{K}p\}$
$\Sigma_0 = \{\mathbf{K}p, p, \mathcal{X}\top, \top\}$
$\Omega_1 = K(\Delta_4, \Delta_5) \setminus K(\Delta_5) = \{\mathbf{K}u\}$
$\Sigma_1 = \{\mathbf{K}u, u, \mathcal{X}\top, \top\}$
$\Omega_2 = K(\Delta_3, \Delta_5) \setminus K(\Delta_5) = \{\mathbf{K}\neg r\}$
$\Sigma_2 = \{\mathbf{K}\neg r, \neg r\}$
$\Omega_3 = K(\Delta_8, \Delta_9) \setminus K(\Delta_8) = \{\mathbf{K}\neg q\}$
$\Sigma_3 = \{\mathbf{K}\neg q, \neg q, \mathcal{X}\top, \top\}$
$\Delta_9 = \{\mathbf{K}\neg q, \neg q, \top\}$
$\Delta_{10} = \{\top, \mathcal{X}\top\}$

Fig. 1. The pretableau for θ in Example 1

Example 2. Figure 2 contains the pretableau for $\theta = \mathbf{K}\mathcal{X}p \wedge \neg\mathcal{X}\mathbf{K}p$ in $\mathsf{TEL}^1(\mathsf{LT})_{\mathsf{sync,nof}}$. A pretableau's annotation is written in superscript next to it, and minimal states denoted with

a ^{m}. When e.g., trying to expand $\{\Delta_2\}^{B_0}$, the produced 'mini-tableau' is

$$\{\Delta_2\} \xrightarrow{\text{KC}} \{\Delta_2\} \xrightarrow{\text{Bubble}} \{\Delta_2, \Delta_4\}.$$

$\{\Delta_2, \Delta_4\}$ cannot be expanded to a set of states where all elements have predecessors, because the only 'minimal' state in B_0 is Δ_1, and the only state Δ' for which $\Delta_1 — x \succ * \dashrightarrow \Delta'$ is Δ_3, and $\Delta_3 \cup \Delta_4$ is patently inconsistent.

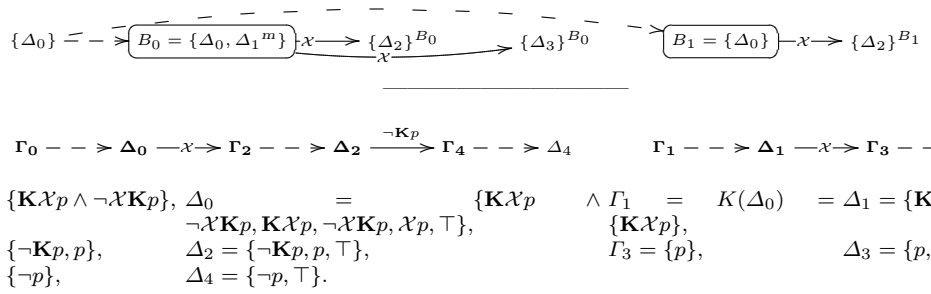


Fig. 2. The pretableau for $\theta = \mathbf{K}\mathcal{X}p \wedge \neg\mathcal{X}\mathbf{K}p$ in Example 2

4.5 Construction of the initial and final tableau

After having constructed the pretableau, the initial tableau is then produced from the pretableau by taking each pre-bubble A in the pretableau, redirecting the arrows to and from A and then deleting A. I.e., for every bubbles B and B', where $B — x \succ A$ and $A \dashrightarrow B'$, we let $B — x \succ B'$ and delete A. To ease the checking for realization of eventualities, we then add arrows between the individual states in two successive bubbles: if $\Delta \in B$ and $\Delta' \in B'$, and $\text{Next}(\Delta) \subseteq \Delta'$, then we add an arrow $\Delta \cdots\succ \Delta'$ between Δ in B and Δ' in B', though, technically, the individual states are not entities in the tableaux.

Finally, the phase of building the final tableau from the initial tableau works by repeatedly making calls to two procedures, Elim-NoTempSuc(B) and Elim-UnrealEven(B), for all bubbles B in the tableau, until no bubble gets deleted in a loop. We define these procedures as follows:

Elim-NoTempSuc(bubble B): If there is no bubble B' in the current tableau such that $B — x \succ B'$, then delete B and all arrows associated with it.

Elim-UnrealEven(bubble B): If the condition (E), defined below, is not satisfied in the current tableau, then delete B and all arrows associated with it.

(E) For any $\Delta \in B$ and any eventuality $\xi \in \Delta$ there exists a bubble-path $\mathcal{B} = (B_i)_{0 \le i \le m}$ with $B_0 = B$ and a temporal path π such that ξ is realized on $\mathcal{B}$ by π.

Definition 11. *The tableau $\mathcal{T}$ for a formula θ is open if there is a bubble $B \in \mathcal{T}$ and a $\Delta \in B$ such that $\theta \in \Delta$.*

Example 3. Figure 3 depicts the initial tableau for the formula θ from Example 1. In the final tableau, the two leftmost bubbles are deleted. The tableau is open (since $\theta \in \Delta_0$ and

$\Delta_0 \in \{\Delta_0, \Delta_2\}$).

The initial tableau for θ in Example 2 simply consists of two bubbles:

In the elimination-procedure, the two bubbles get deleted in the first loop, and the initial tableau is empty. Thus, the tableau closes and θ is declared unsatisfiable.

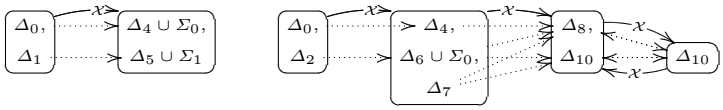


Fig. 3. The initial for θ from Example 1.

5 Soundness

Theorem 1. *The tableaux procedure for each* $\mathsf{TEL}^1(\mathsf{LT})_X$ *is sound.*

Soundness of the tableaux method means that if the input formula is satisfiable, then the procedure will indeed produce an open tableau. The argument in a nutshell is that if the input formula θ is satisfiable, then there is also a satisfiable state Δ in $\mathbf{states}(\{\theta\})$, and the pre-bubble $\{\Delta\}$ will be expanded to a number of bubbles. At least one of them 'survives' in the final tableau. See the appendix for a proof-sketch.

6 Completeness

Theorem 2. *The tableaux procedure for each* $\mathsf{TEL}^1(\mathsf{LT})_X$ *is complete.*

Completeness of the procedure means that an open tableau can be turned into a model, or equivalently into a temporal-epistemic Hintikka structure. In somewhat simplified terms, this is done by making runs corresponding to the 'realizing' temporal path of the states in the bubble B, that ensures that the tableau is open, and then doing the same for each state in the bubbles that these paths pass by. See the proof sketch in the appendix.

7 Complexity

Recall that $\#sts_\theta$ denotes the number of possible states in the bubbles in the tableaux for a formula θ, which is exponential in the length of the input formula, while $\#Bs := 2^{\#sts_\theta}$ is the possible number of bubbles in the tableau.

Theorem 3. *The tableaux procedure for each* $\mathsf{TEL}^1(\mathsf{LT})_X$ *runs in 2EXPTIME.*

The proof of this theorem relies on the fact that all presented methods run in time polynomial in the number of bubbles in the tableau, i.e. the procedure runs in double-exponential time. See the appendix for the proof.

Theorem 4. *For $\mathsf{TEL}^1(\mathsf{LT})_X$ where $\mathsf{nol} \in X$, the tableaux-procedure can be modified to work in EXPSPACE.*

Proof. If $\mathsf{nol} \in X$, the bubble-path constructed in Section 5 can be shortened to a suitable size. In the construction we several times find a bubble-path $\mathcal{B}$ starting in a bubble B, so that a state $\Delta \in B$ has an associated path π^Δ such that a given eventuality $\xi \in \Delta$ is realized on $\mathcal{B}$ by π^Δ. $\mathcal{B}$ and π can now be shortened so that π does not pass through the same state in the same bubble, i.e. the length of π and $\mathcal{B}$ will be at most $\#Bs \cdot \#sts_\theta$; if $B_i, B_j \in \mathcal{B}$ with $B_i = B_j$ and $\pi_i = \pi_j$, then we just remove the bubbles in between B_i and B_{j-1} (including both). The new bubble-path emerging from this operation will be satisfied by possibly another model and state-point-assignment.

In this way, we can obtain a bound on the indexes ρ and m of the ultimately periodic bubble-path $B \text{—}\mathcal{X}\text{➔} \cdots \text{—}\mathcal{X}\text{➔} B_\rho \overset{\overset{\mathcal{X}}{\frown}}{\text{⇐}\mathcal{X}\text{➔} \cdots \text{—}\mathcal{X}\text{➔}} B_{m-1}$. These bounds can now be used to turn the procedure into a procedure that runs in NEXPSPACE=EXPSPACE. This procedure is similar to the one for LTL with an 'exponential step' added to it. If θ is a satisfiable formula, then the following procedure finds the ultimately periodic bubble-path with indexes ρ and m, and if not, "false" is returned. The procedure works as follows:

1. Guess ρ, m and the starting bubble which is kept in the variable $CurB$; check that $CurB$ *is* a bubble. Return "false" if this is not the case. Set $k = 0$.
2. Guess the successor-bubble of $CurB$, which is stored in the variable $NextB$; check that $NextB$ *is* a bubble and that it *is* a successor-set for $CurB$, and return "false" otherwise. If $\mathsf{nof} \in X$, also check that $CurB$ is a predecessor-set for $NextB$, and return "false" otherwise. Assign $NextB$ to $CurB$, and add one to k.
3. Repeat step 2 until $k = \rho$ ($CurB$ now corresponds to B_ρ). Let the variable $RepB$ store the bubble $CurB$. For all $\Delta \in CurB$ and all eventualities $\varphi \mathcal{U} \psi \in \Delta$, add $\varphi \mathcal{U} \psi$ to the set $Real_\Delta$ (create $Real_\Delta$ if it does not already exist).
4. Guess the successor-bubble of $CurB$, and store it in $NextB$. Check that $NextB$ *is* a bubble, that it *is* a successor-set for $CurB$, and if $\mathsf{nof} \in X$, check that $CurB$ is a predecessor-set for $NextB$. Return "false" otherwise. For all $\Delta' \in NextB$ create a new $Real_{\Delta'}$. For all states $\Delta \in CurB$, guess the successor $\Delta' \in NextB$ of Δ. For all eventualities $\varphi \mathcal{U} \psi$ in $Real_\Delta$, add $\varphi \mathcal{U} \psi$ to $Real_{\Delta'}$ if $\psi \notin \Delta'$. Remove $Real_\Delta$ from memory. Assign $NextB$ to $CurB$, and add one to k.
5. Repeat step 4 until $k = m - 1$ ($CurB$ now corresponds to B_{m-1}). Check that $RepB$ is a successor-set of $CurB$, and if $\mathsf{nof} \in X$, check that $CurB$ is a predecessor-set for $RepB$. For all $\Delta \in CurB$, guess the successor $\Delta' \in RepB$ of Δ. Check that all eventualities $\varphi \mathcal{U} \psi \in Real_\Delta$ are realized in Δ', i.e. $\psi \in \Delta'$. If not, return "false".
6. If the algorithm has not terminated so far, return "true".

At any point in time the procedure only keeps 3 bubbles in memory (which takes $3 \cdot \#sts_\theta$ space) and the number of variables $Real_\Delta$ is at most $2 \cdot \#sts_\theta$, and the length of each $Real_\Delta$ is at most $|\theta|$, where $|\theta|$ is the length of the input-formula θ. Thus, the procedure runs in space $\mathcal{O}(poly(\#sts_\theta))$.

8 Concluding remarks

We have substantially extended and adapted the incremental tableau procedure sketched in [3] to work for all cases of single-agent synchronous temporal-epistemic logics with interac-

tions between time and knowledge considered in [4] and [5] and have thus developed a uniform, optimal and practically implementable method for deciding satisfiability in these logics. The method is amenable to easy adaptations to systems where the synchrony-assumption is dropped, and to 1-agent branching-time temporal epistemic logics. It can further be extended to the multiagent case, however, it will work in non-elementary time (due to the complexity of these logics). Tableaux for these and other related cases are part of the future agenda of this project.

References

1. M. Ajspur and V. Goranko. Tableaux-based decision method for single-agent linear time synchronous temporal epistemic logics with interacting time and knowledge. In *Proceedings of the 5th Indian Conference on Logic and its Applications ICLA'2013*, LNCS. Springer-Verlag, 2013, to appear.
2. C. Dixon, C. Nalon, and M. Fisher. Tableaux for logics of time and knowledge with interactions relating to synchrony. *Journal of Applied Non-Classical Logics*, 14(4):397–445, 2004.
3. Valentin Goranko and Dmitry Shkatov. Tableau-based decision procedure for full coalitional multiagent temporal-epistemic logic of linear time. In Decker, Sichman, Sierra, and Castelfranchi, editors, *Proc. of AAMAS'2009*, 2009.
4. Joseph Y. Halpern and Moshe Y. Vardi. The complexity of reasoning about knowledge and time I: Lower bounds. *Journal of Computer and System Sciences*, 38(1):195–237, 1989.
5. Joseph Y. Halpern and Moshe Y. Vardi. Reasoning about knowledge and time: Synchronous systems, 1989. IBM Research Report,.
6. R. Kontchakov, C. Lutz, F. Wolter, and M. Zakharyaschev. Temporalizing tableaux. *Studia Logica*, 76(1):91–134, 2004.
7. C. Nalon, C. Dixon, and M. Fisher. Resolution for synchrony and no learning. In *in: Advances in Modal Logic 5, R. Schmidt et al (eds)*, pages 231–248, 2004.
8. Vaughan R. Pratt. A practical decision method for propositional dynamic logic. In *Proc. of the 10th Annual ACM Symposium on the Theory of Computing*, pages 326–337, San Diego, California, May 1979.
9. Pierre Wolper. The tableau method for temporal logic: an overview. *Logique et Analyse*, 28(110–111):119–136, 1985.

Appendix

Proof sketch for Theorem 1: For $\mathsf{TEL}^1(\mathsf{LT})_X$ with $\mathsf{nol} \in X$, this is the case since there is an ultimately periodic bubble-path

$$B \text{—}\mathcal{X}\text{➛} \cdots \text{—}\mathcal{X}\text{➛} B_\rho \overset{\mathcal{X}}{\text{—}\mathcal{X}\text{➛} \cdots \text{—}\mathcal{X}\text{➛}} B_{m-1}$$

in the initial tableau that starts in a bubble $B \ni \Delta$, where none of the bubbles on the bubble-path will be deleted by any of the two elimination rules, so that the ultimately periodic path will also exist in the final tableau. If $\mathsf{nol} \notin X$, we can identify a tree-structure, where each node n in the tree corresponds to a bubble $B(n)$ in the initial tableau, and the root corresponds to a bubble $B \ni \Delta$. In this tree, if a node n is the parent of the node n', then $B(n)\text{—}\mathcal{X}\text{➛} B(n')$. None of the bubbles corresponding to nodes in the tree will be deleted by any of the two elimination rules, and thus B will also exist in the final tableau. Below we outline how this bubble-path or 'bubble-tree' can be constructed.

First of all, if $\mathcal{M}, (r, n) \Vdash \theta$, where θ is the input formula of the procedure, then there exists a state $\Delta \ni \theta$ such that $\mathcal{M}, (r, n) \Vdash \Delta$, and then there exists a bubble $B \ni \Delta$, such that $\mathcal{M}, (r, n) \Vdash^A B$. This bubble is in the initial tableau, and will 'survive' to the end of the procedure.

The argument builds on the fact that a bubble-path $\mathcal{B} = (B_i)_{0 \leq i \leq m}$ in the initial tableau for which $\mathcal{M}, (r, n) \Vdash^A \mathcal{B}$ can always be 'expanded' with another bubble B_{m+1}, i.e. there exists a bubble-path $\mathcal{B}' = (B'_i)_{0 \leq i \leq m+1}$ in the initial tableau, where $B_i \subseteq B'_i$ for all $0 \leq i \leq m$, and $\mathcal{M}, (r, n) \Vdash^{A'} \mathcal{B}'$. In case $X = \{\mathsf{sync}, \mathsf{nol}\}$ we still have $B'_i = B_i$. The reason why we cannot simply add B_{m+1} to $\mathcal{B}$ when $X \neq \{\mathsf{sync}, \mathsf{nol}\}$ is that even though all states in a satisfiable bubble *do* have a predecessor state, it is not certain that these predecessors are in the given predecessor-bubble. But, if they are not, we have previously made sure to produce a bubble similar to the predecessor-bubble, just with the needed predecessors added.

This fact can then be used to prove that if $\mathcal{B} = (B_i)_{0 \leq i \leq m}$ is a bubble-path in the initial tableau where $\mathcal{M}, (r, n) \Vdash^A \mathcal{B}$ for a model $\mathcal{M}$ and a point (r, n), and there is a state $\Delta \in B_m$ containing an eventuality ξ, then $\mathcal{B}$ can be extended (in the way described above) with a bubble-path $\mathcal{B}'$, such that there is a temporal path π where $\xi \in \Delta$ is realized on $\mathcal{B}'$ by π. The resulting bubble-path will likewise be modelled by $\mathcal{M}$ at (r, n).

We first consider the case where $\mathsf{nol} \in X$. Let Δ be a satisfiable state in $\mathbf{states}(\{\theta\}))$. By induction we can then find a bubble-path $\mathcal{B} = (B_i)_{0 \leq i \leq m}$ and a path π, such that all eventualities in Δ are realized on $\mathcal{B}$ by π. This works since any eventuality in a state is either realized on a temporal path starting in that state, or is in the end state of the path. Since $\mathsf{nol} \in X$, each B_{i+1} is a successor-set for B_i, so there exists a $\pi^{\Delta'}$ following $\mathcal{B}$ and starting in Δ' for all $\Delta' \in B_0$. As before, every eventuality in Δ' that is not realized on $\mathcal{B}$ by $\pi^{\Delta'}$, belongs to the end state of $\pi^{\Delta'}$. By induction, we can expand $\mathcal{B}$, so we end with a bubble-path $\mathcal{B}^0$ starting in $B^0_0 \supseteq B_0$, where for all states $\Delta' \in B^0_0$ there is a $\pi^{\Delta'}$, such that all eventualities in Δ' are realized on $\mathcal{B}^0$ by $\pi^{\Delta'}$. For the last bubble in $\mathcal{B}^0$, we repeat the process above, so that we get a bubble-path $\mathcal{B}^1$ with starting bubble $B^1_0 \supseteq B_0$. For all bubbles $B^1_i \in \mathcal{B}^1$ where i is at most the length of $\mathcal{B}^0$, all states $\Delta' \in B^1_i$ has an associated $\pi^{\Delta'}$ following $\mathcal{B}^1$ from B^1_i and onwards that realizes all eventualities of Δ'.

We can repeat the outlined steps for the last bubble in $\mathcal{B}^1$, and so on. At some point, the last bubble of the constructed bubble-path $\mathcal{B}^j$ will be a bubble $B^j_i \in \mathcal{B}^i$ where i is at most the length of $\mathcal{B}^{j-1}$. So, for all states Δ' in B^j_i there is a temporal path $\pi^{\Delta'}$ such that all eventualities of Δ' are realized on $\mathcal{B}^j$ from B^j_i and onwards by $\pi^{\Delta'}$. The procedure stops now. By construction, no bubble in $\mathcal{B}^j$ can be eliminated by any of the elimination rules, so they are all in the final tableau.

When $\mathsf{nol} \notin X$, all states in a bubble do not necessarily have successors in the same successor-bubble so we cannot build one single bubble-path; instead we build a bubble-tree. Consider again a state $\Delta \ni \theta$. Then there is bubble $B \ni \Delta$ for which $\mathcal{M}, (r, n) \Vdash^A B$, and there is no other bubble $B' \supset B$ in the first level of the tableau with $\mathcal{M}, (r, n) \Vdash^{A'} B'$. For all states Δ' in B we can find a satisfiable bubble-path $\mathcal{B}^{\Delta'} = (B^{\Delta'}_i)_{0 \leq i \leq m_{\Delta'}}$ with $B^{\Delta'}_0 = B$ (since B is maximal) and a $\pi^{\Delta'}$ such that all eventualities in Δ' are realized on $\mathcal{B}^{\Delta'}$ by $\pi^{\Delta'}$. The satisfiable bubbles on $\mathcal{B}^{\Delta'}$ can be chosen to be maximal, i.e. for all $i > 0$ there is no $B'_i \supset B^{\Delta'}_i$ such that $\mathcal{M}, (r, n+i) \Vdash B'_i$ and $B^{\Delta'}_{i-1} —\mathcal{X}\!\!\rightarrow B'_i$. This is done inductively by possibly replacing $B^{\Delta'}_i$ with a bigger, satisfiable B' for which $B^{\Delta'}_{i-1} —\mathcal{X}\!\!\rightarrow B'$. In the tree under construction, we let B be the root, and for each $\mathcal{B}^{\Delta'}$ we let $B^{\Delta'}_i$ be a child of $B^{\Delta'}_{i-1}$ for $1 \leq i \leq m_{\Delta'}$.

We now do the same for any bubble B' in the second level of the tree being build, i.e. we find a maximal $\mathcal{B}^{\Delta'}$ for each $\Delta' \in B'$ starting in B' such that all eventualities of Δ' are realized on $\mathcal{B}^{\Delta'}$ by a π' following $\mathcal{B}^{\Delta'}$. When this is done, the bubbles on the bubble-paths are added in the obvious way to the tree.

We then continue for the next level and do the same, and so on, eventually building up the needed tree of bubbles. When at some level we consider a bubble B', then if it is already in a previous level of the tree, we do not 'expand' it, but make a reference to the bubble in that previous level. Since there is only a finite number of bubbles, and we do not 'expand' a bubble which has already been 'expanded', the process will stop at some point, and then we get a finite tree where all bubbles have a successor in the initial tableau, and further, for all states Δ' in a bubble B' there exists a bubble-path $\mathcal{B}^{\Delta'}$ starting in B' and a temporal path $\pi^{\Delta'}$ such that all eventualities of Δ' is realized on $\mathcal{B}^{\Delta'}$ by $\pi^{\Delta'}$. Therefore, none of the bubbles in this tree will be deleted in the first loop of the elimination procedure, and if they are not deleted in the ith loop, they will not be deleted in the $i+1$'th loop. Thus, they are all present in the final tableau.

Proof sketch of Theorem 2: To prove the completeness of the procedure, we need to show that if the procedure returns an open tableau, then there is a model for the input formula. As discussed earlier, it is sufficient to show how to build a Hintikka-structure with the right properties. Again we start by considering the cases where $\mathsf{nol} \in X$. We construct a bubble-path $\mathcal{B} = (B_i)_{0 \leq i \leq m}$ as described in the proof sketch of Theorem 1. The states of the TEHS are now the states in the bubbles on $\mathcal{B}$. In case $X = \{\mathsf{sync}, \mathsf{nol}\}$, we further add a 'bogus' state $\{\top\}$. When $X \neq \{\mathsf{sync}, \mathsf{nol}\}$, every state in a bubble in $\mathcal{B}$ has a path of predecessors, which will be used as a 'history' for the state. Otherwise we give each state a 'history' of bogus states.

The runs are then build inductively for $i \in \mathbb{N}$, by building a run for each $\Delta \in B_i$; for $i > m$, B_i is defined to be the i'th bubble on the ultimately periodic bubble-path. For $\Delta \in B_i$ we let the run follows its 'history' until it reaches Δ at time instant i. Then it follows a path to a state $\Delta' \in B_\rho$ (which is possible, since each B_{j+1} is a successor-set for B_j), after which it continues along the 'realizing path' for $\Delta' \in B_\rho$. This 'realizing path' is a path that realizes all eventualitites in Δ' (as described in the proof sketch of Theorem 1). It then follows the bubble-path until it reaches $B_m = B_\rho$ again, and so on.

The equivalence classes of the TEHS with regard to $\mathcal{R}$ correspond to the bubbles on $\mathcal{B}$, and if $X = \{\mathsf{sync}, \mathsf{nol}\}$, there will additionally be an equivalence class for each point corresponding to a 'bogus' state. In this way, we can ensure that the TEHS will have No_Learning, and then $X \neq \{\mathsf{sync}, \mathsf{nol}\}$, it will also have No_Forgetting. When $\mathsf{iis} \in X$, there will not be any 'bogus' states, and the starting point of all constructed runs will be in the same bubble, and hence the constructed TEHS will have Indistinguishable_Initial_States.

When $\mathsf{nol} \notin X$, we construct a bubble-tree as described in the proof sketch of Theorem 1. From this tree we build a TEHS as follows: The points of the TEHS are the states in the bubbles of the tree. The runs are defined inductively by making a run for each state Δ in a bubble in the i'th level of the tree; if B' is a bubble at level i that refers back to B'' at level $j < i$ (i.e. $B' = B''$), then B'' is also considered to be at level i. The run for $\Delta \in B_i$ must start in the root-bubble but since parent-bubbles are predecessor-sets for the bubbles corresponding to their children, we can find a path starting in the root-bubble and ending in Δ. We then let the run follow the 'realizing path' for Δ, i.e. a path that realizes all eventualities of Δ. This path ends in a state Δ' in another bubble B' in the tree, and we let the run follow the 'realizing path' for $\Delta' \in B'$, and so on.

The equivalence classes of the points of the TEHS then correspond to the bubbles in the tree. This Hintikka Structure will have No_Forgetting since, again, a parent-bubble is a predecessor-set for each of the bubbles corresponding to its children. By construction, this TEHS will have Indistinguishable_Initial_States.

Proof (Theorem 3). Firstly, we argue that all methods involved in making the pretableu and initial tableau runs in $\mathcal{O}(poly(\#Bs))$. Finding $\mathbf{states}(\Gamma)$ for a set of formulas Γ and checking whether the union of two fully expanded sets are patently inconsistent takes time $\mathcal{O}(poly(\#sts_\theta))$. Constructing the set $\{s_0, \ldots, s_n\}$ for elements $(s_0, \ldots, s_n)$ in $\times_{i=0}^{n} S_i$, where $n \leq \#sts_\theta$ and $|S_i| \leq \#sts_\theta$, takes time $\mathcal{O}(poly(\#sts_\theta))$, and adding a set $\{s_0, \ldots, s_n\}$ to a set of set of states takes time polynomial in $\#Bs$. Therefore, MakeKnowledgeConsistent, LocalBubble and TemporalSuccessorPrebubbles run in total time polynomial in $\#Bs$. In ExpandToBubble, while the set L to be returned is under construction, $|L|$ is always less than $\#Bs$. At each loop, a call is made to LocalBubble, and at most $\#sts_\theta \cdot |\theta|$ elements are added to L (while one element is deleted). Thus a loop runs in $\mathcal{O}(poly(\#Bs))$ time. At each loop, at least one more state in one of the set of states in L gets marked, so the procedure runs in $\mathcal{O}(poly(\#Bs))$ time.

For $X = \{\mathsf{sync}, \mathsf{nol}\}$, ExpandPrebubble makes calls to functions running in $\mathcal{O}(poly(\#Bs))$ time, and it constructs a set with at most $\#Bs$ elements, so the method runs in $\mathcal{O}(poly(\#Bs))$ time. When $X \neq \{\mathsf{sync}, \mathsf{nol}\}$ this is also the case when the method is run on prebubbles in the first temporal layer. For other layers, each step in the described 'mini-tableau' takes time $\mathcal{O}(poly(\#Bs))$, also the step ensuring predecessors. Since prebubbles and bubbles are reused in the mini-tableau, each step is only done once for each (pre)bubble, so the method runs in total in time $\mathcal{O}(poly(\#Bs))$.

Then, the initial tableau is constructed in time $\mathcal{O}(\#Bs^2 \cdot \#sts_\theta^2)$, i.e. $\mathcal{O}(poly(\#Bs))$. Producing the final tableau, likewise, takes time polynomial in $\#Bs$, since the repeat-loop (of repeatedly applying the two elimination rules) is executed at most $\#Bs$ times (at least one bubble has to be deleted in each loop), and the conditions in the two elimination-procedures can be checked in $\mathcal{O}(poly(\#Bs))$ time for any given B: checking that there is a B' such that $B \xrightarrow{\mathcal{X}} B'$ can be done in $\mathcal{O}(poly(\#Bs))$ time. Likewise, for each eventuality $\varphi\mathcal{U}\psi$ in a state $\Delta \in B$, we can check whether $\varphi\mathcal{U}\psi$ can be realized by first marking the states in different bubbles, that contain ψ; if B_1 and B_2 both contain $\Delta \ni \psi$, $\Delta \in B_1$ and $\Delta \in B_2$ are treated as different states and marked separately. Then we go backwards via the temporal relations, i.e. if $\Delta \in B$ has been marked and $\Delta' \in B'$, where $B' \xrightarrow{\mathcal{X}} B$ and $\Delta' \dashrightarrow \Delta$, then we mark $\Delta' \in B'$, and so on. We only do this for marked states for which we have not done it before, so the overall procedure of marking states in bubbles where a given eventuality can be realized takes time $\mathcal{O}(\#Bs \cdot \#sts_\theta)$. If $\Delta \in B$ gets marked by this procedure, then the condition (E) in Elim-UnrealEven for $\varphi\mathcal{U}\psi \in \Delta$ and B is satisfied and otherwise it is not satisfied. For a given B, Elim-UnrealEven(B) thus runs in $\mathcal{O}(poly(\#Bs))$.