A Tableau-Based Proof Method for Temporal Logics of Knowledge and Belief*

Michael Wooldridge
Department of Electronic Engineering
Queen Mary and Westfield College
University of London
London E1 4NS, United Kingdom
M.J.Wooldridge@qmw.ac.uk

Clare Dixon and Michael Fisher
Department of Computing
Manchester Metropolitan University
Chester Street
Manchester M1 5GD, United Kingdom
{C.Dixon,M.Fisher}@doc.mmu.ac.uk

ABSTRACT. *In this paper we define two logics, KL_n and BL_n, and present tableau-based decision procedures for both. KL_n is a* temporal logic of knowledge. *Thus, in addition to the usual connectives of linear discrete temporal logic, it contains a set of unary modal connectives for representing the* knowledge *possessed by* agents. *The logic BL_n is somewhat similar; it is a temporal logic that contains connectives for representing the* beliefs *of agents. In addition to a complete formal definition of the two logics and their decision procedures, the paper includes a brief review of their applications in AI and mainstream computer science, correctness proofs for the decision procedures, a number of worked examples illustrating the decision procedures, and some pointers to further work.*
KEYWORDS: Temporal logics of knowledge and belief, theorem proving, tableau.

1 Introduction

This paper presents two logics, called KL_n and BL_n respectively, and gives tableau-based decision procedures for both. The logic KL_n is a *temporal logic of knowledge*. That is, in addition to the usual connectives of linear discrete temporal logic [5], KL_n contains an indexed set of unary modal connectives that allow us to represent the information possessed by a group of agents.

*This work was supported by EPSRC Research Grant GR/K57282.

These connectives satisfy analogues of the axioms of the modal system S5 [3], which is widely recognised as a logic of *idealised knowledge* [6]. It is for this reason that we call KL_n a temporal logic of knowledge.

Syntactically, the logic BL_n is identical to KL_n. It is also a temporal logic that contains connectives for representing the information possessed by a group of agents. However, the *agent modalities* in BL_n satisfy analogues of the modal axioms KD45; this system is widely accepted as a logic of *idealised belief* [14]. For this reason, we say that BL_n is a *temporal logic of belief.*

Multi-modal logics like BL_n and KL_n have a number of applications, in both mainstream computer science and artificial intelligence (AI); these applications are briefly reviewed in the following subsection. Some notational conventions are then introduced. In section 2, complete formal definitions of the syntax and semantics of the two logics are presented. Tableau-based decision procedures for both logics are presented in section 3. These decision procedures represent generalisations of both the tableau method for the underlying temporal logic [12, 25], and for the modal systems associated with agent modalities (S5 in the case of KL_n, and KD45 in the case of BL_n) [9, 14]. A number of worked examples, illustrating the decision procedures, are presented in section 4, and some possible improvements to the algorithm are discussed in section 5. Areas for future work are identified in section 6.

Finally, a note about our intentions in this work. For the moment, our primary concern is *not* computational efficiency. Instead, the algorithms we develop emphasise conceptual simplicity. We leave efficiency considerations for future work.

1.1 Applications and Related Work

Temporal logics of knowledge are becoming increasingly important in both mainstream computer science and AI. For example, in distributed systems theory, they are used for the specification and verification of *knowledge-based protocols* [13]. Briefly, the idea is that when designing a distributed system, one often makes use of statements such as 'if agent (process) a_1 knows that agent a_2 has received message m_1, then a_1 should *eventually* send message m_2'. Temporal logics of knowledge are used to formalise this kind of reasoning; knowledge is given a precise interpretation, in terms of the states of a process. In AI, temporal logics of belief are used as knowledge representation formalisms (cf. [21]), and may be used in the specification and verification of distributed intelligent systems [26].

Temporal logics of knowledge and belief have previously been studied by many researchers, (e.g., [6, 15, 16, 18, 24]). In [15] the complexity of the validity decision problem for ninety-six variants of the logic we call KL_n is examined, and it is shown that for the simplest variant (corresponding exactly to our logic KL_n), the problem is PSPACE-complete. It is also shown that for some simple variants of this logic, the validity decision problem is undecidable — even in the propositional case. Although tableau methods for linear-time temporal logics have been developed in [12, 25] and tableaux for modal logics have been studied in, for example, [9, 10, 14] decision procedures for the logic KL_n and its variant BL_n have not, to the best of our knowledge, been studied. A tableau-based decision procedure for a simple temporal belief logic is presented in [27]. However, this result is achieved by greatly simplifying the semantics of belief, and in particular, by separating out the semantics of belief from the temporal dimension. Also, the decision procedure in [27] does not deal with axioms 4 and 5, which are ultimately important for any belief logic, and it is not obvious how the decision procedure could be extended in a deterministic way to deal with these axioms. Tableau-based methods have been used for the belief, desire, intention (BDI) logics of Rao and Georgeff in [20]. Finally, we note that resolution based proof methods for temporal logics of knowledge and belief are presented in [8].

1.2 Notation

If L is a logical language, then we write $Form(L)$ for the set of well-formed formulae of L. We use the lowercase Greek letters φ, ψ, and χ as meta-variables ranging over formulae of the logical languages we consider, the uppercase Greek letter Δ as a meta-variable ranging over sets of formulae, and the calligraphic letter $\mathcal{F}$ to denote sets of sets of formulae. If R is a binary relation over some set, then we write R^* for the reflexive transitive closure of R, $dom(R)$ for the domain of R, and $ran(R)$ for the range of R. If S is a set, then we write $\wp(S)$ for the powerset of S.

2 Temporal Logics of Knowledge and Belief

In this section, we formally present the syntax and semantics of two logics: BL_n is a *temporal logic of belief*, and KL_n is a *temporal logic of knowledge*. These logics actually share a common syntax, which we shall call the language $\mathcal{L}$.

2.1 Preliminaries

First, note that $\mathcal{L}$ is not a quantified language. We shall thus build formulae from a set $\Phi = \{p, q, r, \ldots\}$ of *primitive propositions*. In fact, the language $\mathcal{L}$ generalises classical propositional logic, and thus it contains the standard propositional connectives $\neg$ (not) and $\vee$ (or); the remaining connectives ($\wedge$ (and), $\Rightarrow$ (implies), and $\Leftrightarrow$ (if, and only if)) are assumed to be introduced as abbreviations in the usual way. With respect to temporal connectives, we take as primitive just two: $\bigcirc$ (for 'next'), and $\mathcal{U}$ (for 'until'). We interpret these connectives over a *flow of time* that is linear, discrete, bounded in the past, and infinite in the future. An obvious choice for such a flow of time is $(\mathbf{N}, <)$, that is, the natural numbers ordered by the usual 'less than' relation.

With respect to belief/knowledge connectives, we assume a set $Ag = \{1, \ldots, n\}$ of *agents*. We then build an indexed set of unary modal connectives $\{[i] \mid i \in Ag\}$, where a formula $[i]\varphi$ is to be read (in BL_n) as 'agent i believes that φ', or (in KL_n) as 'agent i knows that φ'. In both cases, $\varphi \in Form(\mathcal{L})$.

2.2 Syntax

In this section, we formally present the syntax of $\mathcal{L}$.

Definition 1 *The alphabet of language $\mathcal{L}$ contains the following symbols:*

1. *A set $\Phi = \{p, q, r, \ldots\}$ of* primitive propositions*;*
2. *The nullary connectives* **false** *and* **true***;*
3. *The binary propositional connective $\vee$ (or), and unary propositional connective $\neg$ (not);*
4. *A set $Ag = \{1, \ldots, n\}$ of agents;*
5. *The open and close square bracket symbols ']' and '[', and open and close parentheses ')' and '(';*
6. *The unary temporal connective $\bigcirc$ (next), and binary temporal connective $\mathcal{U}$ (until).*

Definition 2 *The set $Form(\mathcal{L})$ of (well-formed) formulae of $\mathcal{L}$ is defined by the following rules:*

1. *(Primitive propositions are formulae): if $p \in \Phi$ then $p \in Form(\mathcal{L})$;*
2. *(Nullary connectives are formulae):* **false** $\in Form(\mathcal{L})$, **true** $\in Form(\mathcal{L})$*;*
3. *(Unary connectives): if $\varphi \in Form(\mathcal{L})$ then $\neg\varphi \in Form(\mathcal{L})$, $\bigcirc\varphi \in Form(\mathcal{L})$, and $(\varphi) \in Form(\mathcal{L})$;*
4. *(Binary connectives): if $\varphi, \psi \in Form(\mathcal{L})$, then $\varphi \vee \psi \in Form(\mathcal{L})$ and $\varphi \mathcal{U} \psi \in Form(\mathcal{L})$;*
5. *(Agent modalities): if $\varphi \in Form(\mathcal{L})$ and $i \in Ag$ then $[i]\varphi \in Form(\mathcal{L})$.*

2.3 Semantics

Unfortunately, the semantics of $\mathcal{L}$ are rather more complex than its syntax. First, we introduce the notion of a *state*. It is assumed that the world may be in any of a set S of *states*. We generally use s (with annotations, e.g., s_0, s', $\ldots$) to denote a state. The internal structure of states is not an issue in this work, and will not be considered further; see [7, pp335–339] for a discussion. As we interpret $\mathcal{L}$ over linear temporal structures, it is natural to introduce the idea of a *timeline*, representing the history of a system.

Definition 3 *A* timeline, l, *is an infinitely long, linear, discrete sequence of states, indexed by the natural numbers.*

For convenience, we define a timeline l to be a (total) function $l : \mathbf{N} \to S$. Let $TLines$ be the set of all timelines. We shall sometimes find it convenient to view a timeline as an infinite sequence, rather than as a function — the two notions are interchangeable. Note that timelines correspond to the *runs* of Halpern and Vardi [15]. The idea is that we ultimately want to use temporal logics of knowledge to be able to reason about programs. In this view, runs correspond to the histories traced out by programs as they execute. It is for this reason that we explicitly introduce timelines into our model structures. An alternative would be to replace timelines by states together with a binary next-time relation over them, though this would slightly complicate the semantics of our temporal connectives. In any event, the two representations are interchangeable.

If we were dealing with a linear temporal logic, with no agent modalities, then timelines would be our key underlying semantic structures. However, our language includes such modalities, with semantics given in terms of possible worlds. We want to use these semantics to capture the idea that an agent not only does not know what the state of the world currently is, but in addition does not know which *timeline* it is in. The obvious way to define an agent's accessibility relation R_i is thus over timelines, i.e., $R_i \subseteq TLines \times TLines$. However, this would not allow us to capture an agent's *temporal* uncertainty — intuitively, uncertainty about 'what the time is'. For this reason, we introduce *points*.

Definition 4 *A* point, p, *is a pair* $p = (l, u)$, *where* $l \in TLines$ *is a timeline and* $u \in \mathbf{N}$ *is a temporal index into* l.

Any point (l, u) will uniquely identify a state $l(u)$; however, the same state may occur in *many different timelines*. Let the set of all points (over S) be $Points$. We then let an agent's knowledge or belief accessibility relation R_i hold over $Points$, i.e., $R_i \subseteq Points \times Points$, for all $i \in Ag$. This captures the idea of an agent being uncertain both about which timeline it is in, and how far along that timeline it is. A *valuation* for $\mathcal{L}$ is a function that takes a point and a proposition, and says whether that proposition is true (T) or false (F) at that point.

Definition 5 *A* valuation, π, *is a function* $\pi : Points \times \Phi \to \{T, F\}$.

We can now define models for $\mathcal{L}$.

Definition 6 *A* model, M, *for* $\mathcal{L}$ *is a structure* $M = (TL, R_1, \ldots, R_n, \pi)$, *where:*

- $TL \subseteq TLines$ *is a set of timelines;*
- R_i, *for all* $i \in Ag$, *is an agent accessibility relation over* $Points$, *i.e.,* $R_i \subseteq Points \times Points$; *and*

- $\pi : Points \times \Phi \to \{T, F\}$ *is a valuation.*

As usual, we define the semantics of the language via the satisfaction relation '$\models$'. For $\mathcal{L}$, this relation holds between pairs of the form $(M, (l, u))$ (where M is a model and $(l, u) \in Points$), and $\mathcal{L}$ formulae. The rules defining the satisfaction relation are given in Figure 1. Satisfiability and validity in $\mathcal{L}$ are

$(M, (l, u)) \not\models \mathbf{false}$		
$(M, (l, u)) \models \mathbf{true}$		
$(M, (l, u)) \models p$	iff	$\pi((l, u), p) = T$ (where $p \in \Phi$)
$(M, (l, u)) \models \neg\varphi$	iff	$(M, (l, u)) \not\models \varphi$
$(M, (l, u)) \models \varphi \vee \psi$	iff	$(M, (l, u)) \models \varphi$ or $(M, (l, u)) \models \psi$
$(M, (l, u)) \models [i]\varphi$	iff	$\forall l' \in TL, \forall v \in \mathbf{N}$, if $((l, u), (l', v)) \in R_i$, then $(M, (l', v)) \models \varphi$
$(M, (l, u)) \models \bigcirc\varphi$	iff	$(M, (l, u + 1)) \models \varphi$
$(M, (l, u)) \models \varphi\,\mathcal{U}\,\psi$	iff	$\exists v \in \mathbf{N}$ such that $(v \geq u)$ and $(M, (l, v)) \models \psi$, and $\forall w \in \mathbf{N}$, if $(u \leq w < v)$ then $(M, (l, w)) \models \varphi$

Figure 1: Semantics of $\mathcal{L}$

defined in the usual way.

Definition 7 *An $\mathcal{L}$ formula φ is* satisfiable *iff there is some $(M, (l, u))$ such that $(M, (l, u)) \models \varphi$, and* unsatisfiable *otherwise. An $\mathcal{L}$ formula φ is* valid *in a model M iff $(M, (l, u)) \models \varphi$ for every point (l, u) in M. If C is a class of models, then φ is C* valid *iff φ is valid in every model in C (notation: $\models_C \varphi$). The notion of satisfiability with respect to a class of models is defined in a similar way. Finally, φ is valid* simpliciter *iff it is valid in the class of all models (notation: $\models \varphi$).*

We state without proof that the tautologies of propositional logic are valid, as are those formulae that are valid in the underlying propositional temporal logic, and the underlying normal modal logic of agent modalities; see, e.g., [5].

Derived Temporal Connectives

Other standard temporal connectives are introduced as abbreviations, in terms of $\mathcal{U}$:

$$\begin{aligned}
\Diamond\varphi &\stackrel{\text{def}}{=} \mathbf{true}\,\mathcal{U}\,\varphi \\
\Box\varphi &\stackrel{\text{def}}{=} \neg\Diamond\neg\varphi \\
\varphi\,\mathcal{W}\,\psi &\stackrel{\text{def}}{=} (\varphi\,\mathcal{U}\,\psi) \vee \Box\varphi
\end{aligned}$$

We now informally consider the meaning of the temporal connectives. First, consider the two basic connectives: $\bigcirc$ and $\mathcal{U}$. The $\bigcirc$ connective means 'at the next time'. Thus $\bigcirc\varphi$ will be satisfied at some time if φ is satisfied at the *next* time. The $\mathcal{U}$ connective means 'until'. Thus $\varphi\,\mathcal{U}\,\psi$ will be satisfied at some time if ψ is satisfied at that time or some time in the future, and φ is satisfied at all times until the time that ψ is satisfied. Of the derived connectives, $\Diamond$ means 'either now, or at some time in the future'. Thus $\Diamond\varphi$ will be satisfied at some time if either φ is satisfied at that time, or some later time. The $\Box$ connective means 'now, and at all future times'. Thus $\Box\varphi$ will be satisfied at some time if φ is satisfied at that time and at all later times. The binary $\mathcal{W}$ connective means 'unless'. Thus $\varphi\,\mathcal{W}\,\psi$ will be satisfied at some time if either φ is satisfied until such time as ψ is satisfied, or else φ is always satisfied. Note that $\mathcal{W}$ is similar to, but weaker than, the $\mathcal{U}$ connective; for this reason it is sometimes called 'weak until'.

Models for Knowledge and Belief

We shall now define two classes of $\mathcal{L}$ models: KL_n models are models of *knowledge*, and BL_n models are models of *belief*.

Definition 8 *An $\mathcal{L}$ model $M = (TL, R_1, \ldots, R_n, \pi)$ is a KL_n model iff R_i is an equivalence relation, for all $i \in Ag$.*

It should be clear that as agent accessibility relations in KL_n models are equivalence relations, the axioms of the normal modal system S5 are valid in the class of KL_n models.

Theorem 1

$$\models_{KL_n} [i](\varphi \Rightarrow \psi) \Rightarrow ([i]\varphi \Rightarrow [i]\psi) \quad (1)$$
$$\models_{KL_n} [i]\varphi \Rightarrow \neg[i]\neg\varphi \quad (2)$$
$$\models_{KL_n} [i]\varphi \Rightarrow \varphi \quad (3)$$
$$\models_{KL_n} [i]\varphi \Rightarrow [i][i]\varphi \quad (4)$$
$$\models_{KL_n} \neg[i]\varphi \Rightarrow [i]\neg[i]\varphi \quad (5)$$

Proof: Standard; see, e.g., [3, p98]. ∎

These axioms are called K, D, T, 4, and 5, respectively. The system S5 is widely recognised as the logic of idealised *knowledge*, and for this reason we say KL_n is a *temporal logic of knowledge*. (Our logic KL_n in fact corresponds exactly to Halpern and Vardi's logic $KL_{(m)}$ [15].) We now define *belief models*.

Definition 9 *An $\mathcal{L}$ model $M = (TL, R_1, \ldots, R_n, \pi)$ is a BL_n model iff for all $i \in Ag$, R_i is Euclidean, serial, and transitive.*

It is well-known that the axioms K, D, 4, and 5 from normal modal logic are valid in models whose accessibility relations satisfy properties (1)–(3) of Definition 9; however, axiom T is not. Axiom T is generally taken to be the axiom that distinguishes knowledge from belief: it says that if an agent knows φ, then φ is true. As this axiom is not BL_n valid, we say that BL_n is a *temporal logic of belief*.

3 Tableau-Based Proof Methods for KL_n and BL_n

In this section, we present tableau-based decision procedures for the logics KL_n and BL_n. More precisely, we present two algorithms that are guaranteed to determine whether or not an $\mathcal{L}$ formula is KL_n satisfiable or BL_n satisfiable, respectively. These procedures then form refutation-based decision procedures for KL_n and BL_n validity: to show that a formula φ is KL_n (respectively, BL_n) valid, i.e., that $\models_{KL_n} \varphi$ (respectively, $\models_{BL_n} \varphi$), show that $\neg\varphi$ is KL_n (respectively, BL_n) unsatisfiable. In what follows, we assume that the reader is familiar with the fundamentals of tableau-based theorem proving [23].

The two procedures are actually very similar. The basic idea in both cases is to systematically search for a model of the input formula. As both KL_n and BL_n are, in a sense, two-dimensional modal logics, this search must be carried out in two dimensions — to generate the temporal dimension, and to generate the agent accessibility relations. Expansion along the temporal dimension involves unwinding a *next time* relation, η. Although the model of time we use in KL_n and BL_n is linear, η will not be, as nodes in η correspond to states, and states in KL_n and BL_n can have many possible predecessors and successors. We interleave unwinding of the next-time relation η with unwinding of the agent accessibility relations R_i. Once we have fully unwound a structure corresponding to these two modal dimensions, we can begin to systematically remove "inconsistent" states. Once this is done, a simple check is all that is required to see whether our original input formula is satisfied.

We begin by defining certain types of formula.

Definition 10 *If $\varphi \in Form(\mathcal{L})$, then:*

1. *If φ is of the form $[i]\psi$ or $\neg[i]\psi$ then φ is an* agent atom*;*
2. *If φ is an atomic proposition or the negation of an atomic proposition, then φ is a* literal*;*
3. *If φ is an agent atom or a literal, then φ is an* atom*;*
4. *If φ is of the form $\bigcirc\psi$, then φ is a* next-time formula*.*

If $\Delta \subseteq Form(\mathcal{L})$, then define $next(\Delta)$ by:

$$next(\Delta) \stackrel{\text{def}}{=} \{\varphi \mid \bigcirc\varphi \in \Delta\}.$$

In the following, $\neg\neg\varphi$ is always identified with φ so that the sets of formulae we deal with are finite.

To make the presentation easier we use the following table of equivalences for temporal formulae to push negations through until they precede atoms. In particular, when adding the formula $\neg\psi$ to a set of formulae, or performing membership tests such as $\neg\psi \in \Delta$ we will actually add an equivalent formula with the negation pushed as far as possible, or perform the membership test using a formula equivalent to $\neg\psi$ with the negation pushed through until it precedes an atom. The relevant rules for classical connectives are assumed, see for example [12].

formula	formula with negation pushed
$\neg \Box \varphi$	$\Diamond \neg\varphi$
$\neg \Diamond \varphi$	$\Box \neg\varphi$
$\neg \bigcirc \varphi$	$\bigcirc \neg\varphi$
$\neg(\varphi \mathcal{W} \psi)$	$\neg\psi \mathcal{U} (\neg\varphi \wedge \neg\psi)$
$\neg(\varphi \mathcal{U} \psi)$	$\neg\psi \mathcal{W} (\neg\varphi \wedge \neg\psi)$

Figure 2: Negation Equivalences

As with all tableau-based decision procedures, our procedure relies upon *alpha* and *beta* equivalences; these equivalences are defined in Figure 3. (We omit the rules for classical connectives, as these are standard [23].) The only

α	α_1	α_2
$\Box\varphi$	φ	$\bigcirc \Box\varphi$

β	β_1	β_2
$\Diamond\varphi$	φ	$\neg\varphi \wedge \bigcirc\Diamond\varphi$
$\varphi \mathcal{U} \psi$	ψ	$\neg\psi \wedge \varphi \wedge \bigcirc(\varphi \mathcal{U} \psi)$
$\varphi \mathcal{W} \psi$	ψ	$\neg\psi \wedge \varphi \wedge \bigcirc(\varphi \mathcal{W} \psi)$

Figure 3: Alpha and Beta Equivalences

difference between the KL_n and BL_n algorithms is that the following additional alpha rule is required for the KL_n procedure; this rule is *not* used by the BL_n procedure.

α	α_1	α_2
$[i]\varphi$	φ	$[i]\varphi$

Intuitively, this rule corresponds to the reflexivity condition placed on KL_n models: if agent i knows φ, then φ must be true in the current state. Note that we must be careful when using this rule not to apply it more than once to the same formula, or else our algorithm will not terminate.

Next, we give the definitions related to the construction of *propositional subformula complete tableaux* (PC-tableaux), which involve the application of alpha and beta formula and adding particular subformulae or their negations in the case of agent atoms. The latter is required for KL formulae to ensure that we can construct models from the structure resulting from applying the tableau algorithm.

Definition 11 *A* propositional tableau *is a set of formulae where no further alpha and beta rules may be applied.*

Definition 12 *If* $\varphi \in Form(\mathcal{L})$ *then* $sub(\varphi)$ *is the set of all subformulae of* φ*:*

$$sub(\varphi) = \begin{cases} \{\varphi\} & \textit{if } \varphi \in \Phi \textit{ or } \varphi = \mathbf{true} \textit{ or } \varphi = \mathbf{false} \\ \{\neg\psi\} \cup sub(\psi) & \textit{if } \varphi = \neg\psi \\ \{\psi \vee \chi\} \cup sub(\psi) \cup sub(\chi) & \textit{if } \varphi = \psi \vee \chi \\ \{[i]\psi\} \cup sub(\psi) & \textit{if } \varphi = [i]\psi \\ \{\bigcirc\psi\} \cup sub(\psi) & \textit{if } \varphi = \bigcirc\psi \\ \{\psi\,\mathcal{U}\,\chi\} \cup sub(\psi) \cup sub(\chi) & \textit{if } \varphi = \psi\,\mathcal{U}\,\chi \end{cases}$$

Definition 13 *A set of formulae* Δ *is said to be* subformula complete *iff for every* $[i]\varphi \in \Delta$ *for all* $[i]\psi \in sub(\varphi)$ *either* $[i]\psi \in \Delta$ *or* $\neg[i]\psi \in \Delta$.

Definition 14 *A* propositional subformula complete tableau *or* PC-tableau *is a set of formulae* Δ *that is both a propositional tableau and subformula complete.*

The internal consistency of states during tableau generation is established by checking whether they are *proper*.

Definition 15 *If* $\Delta \subseteq Form(\mathcal{L})$ *then* Δ *is* proper, *(notation* $proper(\Delta)$*) iff:*

1. $\mathbf{false} \notin \Delta$;
2. *If* $\varphi \in \Delta$, *then* $\neg\varphi \notin \Delta$.

Improper sets have the following obvious property.

Lemma 1 *If* $\Delta \subseteq Form(\mathcal{L})$ *and* $\neg proper(\Delta)$ *then* Δ *is unsatisfiable.*

We can now describe how to construct the set of *PC-tableaux* for a set of formulae Δ. These sets are so-called as they are essentially the structures generated by tableau methods in classical logics. However we have extra alpha and beta rules for dealing with the additional temporal operators, and require the sets to be subformula complete. Note, we only apply a rule if it modifies the structure. Thus, for example, we don't keep applying the alpha rule for $\square$ to the same $\square$ formula.

Constructing the set of PC-tableaux. Given a set of formulae Δ, the set of PC-tableaux can be constructed by applying the following rules (1) and (2) to $\mathcal{F} = \{\Delta\}$ until they cannot be applied any further.

1. *Forming a propositional tableau.*

   For any $\Delta' \in \mathcal{F}$ such that $proper(\Delta')$, take any formula $\varphi \in \Delta'$ on which alpha or beta rules have not been applied. If φ is an alpha formula with components α_1 and α_2 replace Δ' by $\Delta' \cup \{\alpha_1, \alpha_2\}$. If φ is a beta formula with beta components β_1 and β_2 then let

   $$\mathcal{F} = \mathcal{F} - \Delta' \cup \{\Delta' \cup \{\beta_1\}\} \cup \{\Delta' \cup \{\beta_2\}\}.$$

2. *Forming a subformula complete tableau.*

   For any $\Delta' \in \mathcal{F}$ such that $proper(\Delta')$, if $[i]\varphi \in \Delta'$ and $[i]\psi \in sub(\varphi)$ such that neither $[i]\psi \in \Delta'$ nor $\neg[i]\psi \in \Delta'$ then let

   $$\mathcal{F} = \mathcal{F} - \Delta' \cup \{\Delta' \cup \{[i]\psi\}\} \cup \{\Delta' \cup \{\neg[i]\psi\}\}.$$

3. *Deleting improper sets.*

   Delete any $\Delta' \in \mathcal{F}$ such that $\neg proper(\Delta')$.

Step 2 corresponds to the use of the *analytic cut* rule [22, 9, 11]. Thus formulae are added to the sets being constructed which are subformulae of existing formulae.

Lemma 2 *If the set of PC-tableaux for $\{\varphi\}$ is empty then φ is unsatisfiable.*

Proof: The algorithm constructing the set of PC-tableaux for φ generates (almost) a classical tableau for φ (recall we have extra alpha and beta rules for the temporal connectives and an extra alpha rule for KL_n). We therefore simply note that the alpha and beta rules are sound and that a set will only be deleted if it is labelled with an improper set. Such sets, (by Lemma 1) are unsatisfiable. Construction of the algorithm terminates either when the empty set is obtained or when no other rule can be applied — the procedure will therefore terminate. ∎

We now move on to the model-like structures that will be generated by the tableau algorithm. Note that $State = \{s, s', \ldots\}$ is the set of all states.

Definition 16 *A* structure, *H, is a tuple $H = (S, \eta, R_1, \ldots, R_n, L)$, where:*

- *$S \subseteq State$ is a set of states;*
- *$\eta \subseteq S \times S$ is a binary* next-time *relation on S;*
- *$R_i \subseteq S \times S$ represents an accessibility relation over S for agent $i \in Ag$;*

- $L : S \to \wp(Form(\mathcal{L}))$ *labels each state with a set of* $\mathcal{L}$ *formulae.*

Definition 17 *If* $\varphi \in Form(\mathcal{L})$ *is of the form* $\chi \mathcal{U} \psi$ *or* $\diamondsuit\psi$ *then* φ *is said to have* eventuality ψ. *If* $(S, \eta, R_1, \ldots, R_n, L)$ *is a structure,* $s \in S$ *is a state,* η^* *is the reflexive transitive closure of* η*, and* $\varphi \in Form(\mathcal{L})$*, then* φ *is said to be* resolvable *in* $(S, \eta, R_1, \ldots, R_n, L)$ *from* s*, (notation* $resolvable(\varphi, s, (S, \eta, R_1, \ldots, R_n, L)))$*, iff if* φ *has eventuality* ψ*, then* $\exists s' \in S$ *such that* $(s, s') \in \eta^*$ *and* $\psi \in L(s')$.

The tableau algorithm first expands the structure and then contracts it. We try to construct a structure from which a model may possibly be extracted, and then delete states in this structure that are labelled with formulae such as $\diamondsuit p$ or $\neg[i]p$ which are not satisfied in the structure. Expansion uses the formulae in the labels of each state to build η and R_i successors. The alpha and beta rules that have been previously applied split formulae into those relating to the current moment, those relating to the next moment in time, and agent atoms. Temporal successors (η-successors) to a state s are constructed by taking the set of formulae in the label of s whose main connective is $\bigcirc$, removing the outermost $\bigcirc$ operator, labelling a new state s' by this set if one does not already exist and adding (s, s') to η. This corresponds to part 3 of Wolper's graph construction algorithm [25, page 124].

R_i successors are slightly more complicated. Recall that for KL_n the axioms for the normal modal logic S5 are valid, and for BL_n the axioms for the normal modal logic KD45 are valid. If the label for a state s contains the formula $\neg[i]\psi$ then for this formula to be satisfied we must build an R_i successor s' containing $\neg\psi$. If s also contains $[i]\chi$ formulae then the label of s' must also contain χ and $[i]\chi$, as all R_i successors of s must contain χ, to satisfy the semantics of the modal logic, and $[i]\chi$ to satisfy axiom 4. Further, for all formulae $\neg[i]\chi$ in s, in order to satisfy axiom 5 the label of s' must also contain $\neg[i]\chi$. These conditions correspond with step 2(c$''$) in Halpern and Moses [14, page 368]. Note that if s contains no formulae of the form $\neg[i]\chi$ but does contain $[i]\chi$ formulae we must build an R_i successor containing χ and $[i]\chi$, as above, to satisfy the seriality conditions imposed by axiom D. For KL_n tableau the T axiom is incorporated by an extra alpha rule (see above).

Having built a structure, states must be deleted that can never be part of a model. Considering the temporal dimension, states with unresolvable eventualities are deleted. Also, states containing next-time formulae without η successors are deleted (corresponding to Wolper's elimination rules E3 and E2 respectively [25, page 124]). For the modal dimension we must ensure that any state labelled by a formula of the form $\neg[i]\chi$ has an R_i successor containing $\neg\chi$, and also that every modal state containing $[i]\chi$ has at least one R_i successor and all its R_i successors contain χ. This corresponds to the opposite of marking nodes satisfiable in Halpern and Moses [14, page 362] part (d)(iii).

3.1 The Tableau Algorithm

Given the $\mathcal{L}$ formulae φ_0 to be shown unsatisfiable, perform the following steps.

1. *Initialisation.*

   First, set
   $$S = \eta = R_1 = \cdots = R_n = L = \emptyset.$$
   Construct $\mathcal{F}$, the set of PC-tableaux for $\{\varphi_0\}$. For each $\Delta_i \in \mathcal{F}$ create a new state s_i and let $L(s_i) = \Delta_i$ and $S = S \cup \{s_i\}$. For each $\Delta_i \in \mathcal{F}$ repeat steps (2)–(3) below, until none apply and then apply step 4.

2. *Creating R_i successors.*

   For any state s labelled by formulae $L(s)$, where $L(s)$ is proper and a PC-tableau, for each formula of the form $\neg[i]\psi \in L(s)$ create a set of formula
   $$\Delta = \{\neg\psi\} \cup \{\chi \mid [i]\chi \in L(s)\} \cup \{[i]\chi \mid [i]\chi \in L(s)\} \cup \{\neg[i]\chi \mid \neg[i]\chi \in L(s)\}.$$
   If s contains no such formulae but there exists $[i]\psi \in L(s)$ then construct the set of formulae
   $$\Delta = \{\chi \mid [i]\chi \in L(s)\} \cup \{[i]\chi \mid [i]\chi \in L(s)\}.$$
   For each Δ above construct $\mathcal{F}$, the set of PC-tableaux for Δ, and for each member $\Delta' \in \mathcal{F}$ if $\exists s'' \in S$ such that $\Delta' = L(s'')$ then add (s, s'') to R_i, otherwise add a new state s' to S, labelled by $L(s') = \Delta'$, and add (s, s') to R_i. (Note that the second part of this rule is required because of the D axiom, which is valid in both KL_n and BL_n, and which requires states to have R_i successors.)

3. *Creating η successors.*

   For any state s labelled by formulae $L(s)$, where $L(s)$ is proper and a PC-tableau, if $\bigcirc\psi \in L(s)$ create the set of formulae $\Delta = next(L(s))$ For each Δ construct $\mathcal{F}$ the set of PC-tableaux for Δ, and for each member $\Delta' \in \mathcal{F}$ if $\exists s'' \in S$ such that $\Delta' = L(s'')$ then add (s, s'') to η, otherwise add a state s' to the set of states, labelled by $L(s') = \Delta'$ and add (s, s') to η.

4. *Contraction.*

   Continue deleting any state s where

   (a) $\exists\psi \in L(s)$ such that $\neg resolvable(\psi, s, (S, \eta, R_1, \ldots, R_n, L))$; or

   (b) $\exists\psi \in L(s)$ such that ψ is of the form $\bigcirc\chi$ and $\nexists s' \in S$ such that $(s, s') \in \eta$; or

(c) $\exists \psi \in L(s)$ such that ψ is of the form $\neg[i]\chi$ and $\nexists s' \in S$ such that $(s, s') \in R_i$ and $\neg\chi \in L(s')$; or

(d) $\exists \psi \in L(s)$ such that ψ is of the form $[i]\chi$ and $\nexists s' \in S$ such that $(s, s') \in R_i$ and $\chi \in L(s')$

until no further deletions are possible.

Note we cannot interleave expansion steps with deletion steps. For example to determine whether an eventuality is resolvable, the structure must be fully expanded, otherwise states may be wrongly deleted.

If $\varphi_0 \in Form(\mathcal{L})$, then we say the tableau algorithm is *successful* iff the structure returned contains a state s such that $\varphi_0 \in L(s)$. We claim that a formula φ_0 is BL_n satisfiable iff the tableau algorithm performed on φ_0 is successful without using the extra KL_n alpha rule, and KL_n satisfiable iff the tableau algorithm performed on φ_0 is successful using the extra KL_n alpha rule. We shall now prove this claim.

3.2 Correctness

We are now obliged to show that the algorithms we have presented for determining KL_n and BL_n satisfiability are totally correct. (In fact, we prove total correctness only for the KL_n algorithm — the BL_n case is very similar.) Total correctness requires that

- the algorithm claims a formula is satisfiable if, and only if, that formula is indeed satisfiable; and
- the algorithm is guaranteed to terminate, for any acceptable input.

We begin by defining what it means for a structure to be a KL_n *tableau* for some formula.

Definition 18 *If $\varphi_0 \in Form(\mathcal{L})$, and $H = (S, \eta, R_1, \ldots, R_n, L)$ is a structure, then H is said to be a KL_n* tableau *for φ_0 iff:*

1. *$\exists s \in S$ such that $\varphi_0 \in L(s)$;*

and, $\forall s \in S$, we have:

2. *$proper(L(s))$;*
3. *If $\psi \in L(s)$ and ψ is and alpha formula with alpha components α_1 and α_2, then $\alpha_1 \in L(s)$ and $\alpha_2 \in L(s)$;*
4. *If $\psi \in L(s)$ and ψ is a beta formula with beta components β_1 and β_2, then either $\beta_1 \in L(s)$ or $\beta_2 \in L(s)$;*
5. *If $[i]\psi \in L(s)$ or $\neg[i]\psi \in L(s)$ and $[i]\varphi \in sub(\psi)$, then either $[i]\varphi \in L(s)$ or $\neg[i]\varphi \in L(s)$;*

6. *If* $[i]\psi \in L(s)$ *then* $\forall s' \in S$ *if* $(s, s') \in R_i$ *then* $\psi \in L(s')$*;*

7. *If* $\neg[i]\psi \in L(s)$ *then* $\exists s' \in S$ *such that* $(s, s') \in R_i$ *and* $\neg\psi \in L(s')$*;*

8. *If* $[i]\psi \in L(s)$ *then* $\psi \in L(s)$*;*

9. $\forall s' \in S$*, if* $(s, s') \in R_i$ *then* $[i]\psi \in L(s)$ *iff* $[i]\psi \in L(s')$*;*

10. *If* $\bigcirc\psi \in L(s)$*, then* $\exists s' \in S$ *such that* $(s, s') \in \eta$*;*

11. *If* $\bigcirc\psi \in L(s)$*, then* $\forall s' \in S$*, if* $(s, s') \in \eta$ *then* $\psi \in L(s')$*;*

12. *If* $\psi \in L(s)$*, then* $resolvable(\psi, s, (S, \eta, R_1, \ldots, R_n, L))$*.*

Note that for BL_n tableau, we remove clauses (8) and (9), and replace them with the following [14, p359]:

- (BL_n1): if $(s, s'), (s, s'') \in R_i$ and $[i]\psi \in L(s')$ then $\{[i]\psi, \psi\} \subseteq L(s'')$;
- (BL_n2): if $[i]\psi \in L(s)$, then $\exists s'$ such that $(s, s') \in R_i$;
- (BL_n3): if $[i]\psi \in L(s)$ and $(s, s') \in R_i$ then $[i]\psi \in L(s')$.

Condition (BL_n1) corresponds to the Euclidean property; condition (BL_n2) correponds to seriality, and condition (BL_n3) corresponds to transitivity.

The first (and longest) part of the proof for partial correctness involves showing that if formula has a KL_n tableau, then that formula is KL_n satisfiable.

Theorem 2 *If* $\varphi_0 \in Form(\mathcal{L})$ *and* $H = (S, \eta, R_1, \ldots, R_n, L)$ *is a* KL_n *tableau for* φ_0*, then* φ_0 *is* KL_n *satisfiable.*

Proof: Our proof is constructive. We show that if $H = (S, \eta, R_1, \ldots, R_n, L)$ is a KL_n tableau for a formula $\varphi_0 \in Form(\mathcal{L})$, then we can extract a model $M = (TL, R'_1, \ldots, R'_n, \pi)$ from H in which φ_0 is satisfied.

First, we show how to construct the set TL of timelines. The basic idea is to unwind paths through η, the next-time relation on S. However, we cannot do this in an arbitrary way, as loops in η could lead to paths (and hence timelines) containing unresolved eventualities. We must therefore unwind only *fulfilling* paths: those in which all eventualities are resolved. These paths will be our timelines. We now formally define fulfilling paths, and prove a lemma that shows how fulfilling paths may be constructed.

Definition 19 *A path,* ρ*, through a next-time relation* η *is a connected linear relation* $\rho \subseteq \eta$*.*

Definition 20 *Let* $(S, \eta, R_1, \ldots, R_n, L)$ *be a structure, and* $\rho \subseteq \eta$ *be a path through* η*. Then* ρ *is* fulfilling *iff* $\forall s \in (dom(\rho) \cup ran(\rho))$*, and* $\forall \psi \in L(s)$*, we have* $resolvable(\psi, s, (S, \rho, R_1, \ldots, R_n, L))$*.*

Lemma 3 *Suppose $(S, \eta, R_1, \ldots, R_n, L)$ is a (non-empty) structure returned by the tableau algorithm, and $s \in S$ is a state. Then there is a fulfilling path through η rooted at s.*

Proof: Starting from s, we unwind a path through η, systematically fulfilling eventualities as we go. Start by creating a list of unfulfilled eventualities, E_{OLD}, initialised to those that occur, unsatisfied, in $L(s)$ and an empty list E_{NEW} to contain new eventualities encountered along the way whilst satisfying those in E_{OLD}. Then take the first eventuality ψ from E_{OLD}, and find a state $s' \in S$ such that $(s, s') \in \eta^*$ and $\psi \in L(s')$. Note that there must be such a state, or else s would have been deleted by condition 4(a) of the tableau algorithm. Then remove all eventualities encountered on the path from s to s' through η from the list of unfulfilled eventualities in either E_{OLD} or E_{NEW}, and add any that occur on the way to E_{NEW}. Then repeat the whole process, by taking the next unfulfilled eventuality $\chi \in E_{OLD}$, picking another state s'' such that $(s', s'') \in \eta^*$ and $\chi \in L(s'')$, and so on. Repeat this process until the list of unfulfilled eventualities in E_{OLD} is empty. If E_{NEW} is empty then this is a fulfilling path. If E_{NEW} is not empty call the state where this occurs s_0, set $E_{OLD} = E_{NEW}$, $E_{NEW} = \emptyset$ and begin fulfilling each eventuality in E_{OLD} as before. As the structure is finite then eventually we reach a node, s_i, (where we have just fulfilled all the eventualities in E_{OLD}) we've reached before, i.e. $s_i = s_j$ for some $i > j$. So by infinitely unwinding through the path constructed between nodes s_j and s_i we fulfill all the eventualities we encounter. The path that we trace out from s by this process gives us a fulfilling path rooted at s. ∎

The set of paths that we obtain from H by this method are almost the set of timelines that we require; however, they may be finite, and timelines are infinite. To turn such a finite fulfilling path into a timeline, we simply continue unwinding through η. If ever we come across another eventuality, then we simply generate and append another fulfilling path, using the procedure described above. If ever we reach the end of the η relation, we append an infinite sequence of states containing just **true**. In this way, we will generate an infinite timeline in which all eventualities are fulfilled, and in which, by construction, all next-time constraints are realised. We let TL be the set of sequences we obtain in this way.

Next, we must recover each agent's accessibility relation R'_i. To do this, we must map the state-based relations of tableau structures into the point-based ones required by models. First, if s is a state and TL is a set of timelines, then let $points(s, TL)$ denote the set of points in TL whose state is s:

$$points(s, TL) \stackrel{\text{def}}{=} \{(l, u) \mid l \in TL \text{ and } l(u) = s\}$$

(Recall that we can view a timeline l either as an infinite sequence or else as a function $l : \mathbf{N} \to S$.) The actual transformation from state-based accessibility relations R_i to point-based accessibility relations R'_i is achieved as follows:

$$R_i' \stackrel{\text{def}}{=} \{((l,u),(l',v)) \mid (s,s') \in R_i, (l,u) \in points(s,TL), \\ \text{and } (l',v) \in points(s',TL)\}.$$

Equivalently,

$$((l,u),(l',v)) \in R_i' \text{ iff } l(u) = s, l'(v) = s', \text{ and } (s,s') \in R_i.$$

The relation R_i' will be empty iff R_i is empty. Note that R_i' will not be an equivalence relation; we thus take the reflexive symmetric transitive closure of R_i' to get the accessibility relations as required.

As an aside, note that the BL_n case is slightly different. To ensure that R_i is serial, we proceed as follows. First, if $L(s)$ contains any formulae of the form $[i]\psi$ or $\neg[i]\psi$, then we know s must have an R_i successor, or else s would have been deleted during the contraction stage. If $L(s)$ contains no formulae of this type, then we can ensure that R_i' is serial by creating a state s', setting $L(s')$ to $\{\mathbf{true}\}$, and adding (s,s') and (s',s') to to R_i. We then take the euclidean transitive closure and do the conversion to point-based models to obtain R_i'. Note that this procedure preserves properties (BL_n1)–(BL_n3). As a corollary note that if (s,s') is an arc added during the generation of the Euclidean closure, and $[i]\psi \in L(s)$, then $\{[i]\psi, \psi\} \subseteq L(s')$.

Henceforth, we write R_i' for the point-based relations that we obtain from the state-based relations R_i in this way.

Recovering the valuation function π is straightforward. For every point (l,u) in TL, and for every $p \in \Phi$, define π by:

$$\pi((l,u),p) \stackrel{\text{def}}{=} \begin{cases} T & \text{if } p \in L(l(u)) \\ F & \text{if } \neg p \in L(l(u)) \end{cases}$$

This completes our recovery of the model $M = (TL, R_1', \ldots, R_n', \pi)$. We must now show that our original input formula φ_0 is satisfied in M. In fact, we prove something slightly more general than this: we show that if $\psi \in L(s)$, (respectively, $\neg\psi \in L(s)$), and (l,u) is a point in $M = (TL, R_1', \ldots, R_n', \pi)$ such that $l(u) = s$, then $(M,(l,u)) \models \psi$ (respectively, $(M,(l,u)) \models \neg\psi$).

The proof is by induction on the structure of ψ.

1. The base case.

   This is where $\psi = p$ is a primitive proposition or $\psi = \neg p$ is the negation of a primitive proposition. In the first case, we know by construction of π that $\pi((l,u),p) = T$, and hence (by semantic rules) $(M,(l,u)) \models p$. In the second case, we know, again by construction of π, that $\pi((l,u),p) = F$, and so $(M,(l,u)) \not\models p$, and hence (by semantic rules), $(M,(l,u)) \models \neg p$.

2. Inductive assumption.

   Assume that if $\psi \in L(s)$ (respectively, $\neg\psi \in (s)$), and (l,u) is a point in M such that $l(u) = s$, then $(M,(l,u)) \models \psi$ (respectively, $(M,(l,u)) \models \neg\psi$).

3. Inductive step.

   We should consider all the forms that ψ may take; however, we shall only do proofs for the main forms, and leave the remainder (including all derived temporal connectives) as an exercise for the reader.

   (a) ψ is of the form $\chi \wedge \chi'$.
   By clause (3) of the definition of KL_n tableaux, it must be that $\chi \in L(s)$ and $\chi' \in L(s)$. By the inductive assumption, therefore, $(M,(l,u)) \models \chi$ and $(M,(l,u)) \models \chi'$ and hence (by semantic rules) $(M,(l,u)) \models \chi \wedge \chi'$.

   (b) ψ is of the form $\chi \vee \chi'$.
   By clause (4) of the definition of KL_n tableaux, it must be that either $\chi \in L(s)$ or $\chi' \in L(s)$. By the inductive assumption, therefore, either $(M,(l,u)) \models \chi$ or $(M,(l,u)) \models \chi'$ and hence (by semantic rules) $(M,(l,u)) \models \chi \vee \chi'$.

   (c) ψ is of the form $\neg[i]\chi$.
   By the semantic rule for $[i]$, for $(M,(l,u)) \models \neg[i]\chi$, there must be some (l',v) such that $((l,u),(l',v)) \in R'_i$ and $(M,(l',v)) \models \neg\chi$. Now, by clause (7) of the definition of KL_n tableaux, there must be some state $s' \in S$ such that $(s,s') \in R_i$ and $\neg\chi \in L(s)$. Let (l',v) be any member of $points(s',TL)$. Clearly, $((l,u),(l',v)) \in R'_i$. Moreover, by the induction hypothesis, $(M,(l',v)) \models \neg\chi$. Hence $(M,(l,u)) \models \neg[i]\chi$.

   (d) ψ is of the form $[i]\chi$.
   We need to show that if $(M,(l,u)) \models [i]\chi$, then $(M,(l',v)) \models \chi$ for all (l',v) such that $((l,u),(l',v)) \in R'_i$.
   The proof for KL_n is as follows. There must have been some sequence $(s_0,\ldots,s_k)$ of states, such that $s_0 = s$ and $(l',v) \in points(s_k,TL)$, such that $\forall m \in \{0,\ldots,k-1\}$, either $(s_m,s_{m+1}) \in R_i$ or else $(s_{m+1},s_m) \in R_i$. From clause (9) of the definition of KL_n tableaux, it is easy to see that $[i]\chi \in L(s_m)$, for all $m \in \{0,\ldots,k\}$. From clause (8) of the definition of KL_n tableaux, it must be that $\chi \in L(s_m)$, for all $m \in \{0,\ldots,k\}$. By the induction hypothesis, therefore, $(M,(l',v)) \models \chi$, and we are done.
   The proof for BL_n is as follows. Let s be the state from which the point (l,u) was created, and s' be the state from which (l',v) was created. There must exist a sequence $(s_0,\ldots,s_k)$ such that $s_0 = s$, $s_k = s'$, and $\forall m \in \{0,\ldots,k-1\}$, either $(s_m,s_{m+1}) \in R_i$ or else (s_m,s_{m+1}) was created by the Euclidean closure procedure. We want to show that $\psi \in L(s_k)$. An easy induction on m achieves this. For the inductive step, assume $[i]\psi \in L(s_l)$, and consider (s_l,s_{l+1}). If $(s_l,s_{l+1}) \in R_i$, then $\psi \in L(s_{l+1})$ by construction. Otherwise, (s_l,s_{l+1}) was added when generating the Euclidean closure of R_i.

But in this case, we know from (BL_n1) that $\{[i]\psi, \psi\} \subseteq L(s_{l+1})$. Hence $\psi \in L(s_k)$, and so $(M,(l',v)) \models \psi$, and we are done.

(e) ψ is of the form $\bigcirc\chi$.

We must show that $(M,(l,u+1)) \models \chi$. This follows easily from clauses (10) and (11) of the definition of KL_n tableaux, together with the induction hypothesis.

(f) ψ is of the form $\chi \mathcal{U} \chi'$.

To show that $(M,(l,u)) \models \chi \mathcal{U} \chi'$, we must show that $\exists v \geq u$ such that $(M,(l,v)) \models \chi'$ and $\forall w \in \{u,\ldots,v\}$, we have $(M,(l,w)) \models \chi$. The first part follows easily from the fact that all eventualities must be resolved (clause (12) of the definition of KL_n tableaux, together with the induction hypothesis). For the second, note that the beta rule for $\mathcal{U}$ formulae requires that if $\chi \mathcal{U} \chi' \in L(s)$, then either $\chi' \in L(s)$ or $\{\neg\chi'\} \cup \{\chi\} \cup \{\bigcirc(\chi \mathcal{U} \chi')\} \subseteq L(s)$. In the first case, by the induction hypothesis, we have $(M,(l,u)) \models \chi'$ (and hence $(M,(l,u)) \models \chi \mathcal{U} \chi'$). In the second, we have $(M,(l,u)) \models \chi$ and $(M,(l,u+1)) \models \chi \mathcal{U} \chi'$. After a finite number of steps of this reasoning, are guaranteed to reach a situation where $(M,(l,v)) \models \chi'$ with $(M,(l,w)) \models \chi$ for all $w \in \{u,\ldots,w\}$, and hence $(M,(l,u)) \models \chi \mathcal{U} \chi'$.

This concludes the proof of Theorem 2. ∎

Next, we claim that the structures returned by successful invocations of the tableau algorithm are in fact KL_n tableau structures for the input.

Lemma 4 *If $\varphi_0 \in Form(\mathcal{L})$, the output from the tableau algorithm on φ_0 is $T = (S,\eta,R_1,\ldots,R_n,L)$, and $\exists s \in S$ such that $\varphi_0 \in L(s)$, then $(S,\eta,R_1,\ldots,R_n,L)$ is a KL_n tableau for φ_0.*

Proof: Assume that the tableau algorithm constructs the structure $T = (S,\eta,R_1,\ldots,R_n,L)$ where $\exists s \in S$ such that $\varphi_0 \in L(s)$. We check that each part of Definition 18 holds for T.

1. $\exists s \in S$ such that $\varphi_0 \in L(s)$ — holds by assumption.

2. For all states $s \in S$ we have $proper(L(s))$ as improper states are deleted at step (3) of the construction of the set of PC-tableaux.

3. For all states $s \in S$ if $\psi \in L(s)$ is an alpha formula with alpha components α_1 and α_2 then the construction of the set of PC-tableaux for $L(s)$, step (1), ensures that $\alpha_1, \alpha_2 \in L(s)$.

4. For all states $s \in S$ if $\psi \in L(s)$ is an beta formula with beta components β_1 and β_2 then during the construction of the set of PC-tableaux for $L(s)$ step (1), ensures that either $\beta_1 \in L(s)$ or $\beta_2 \in L(s)$.

5. For all states $s \in S$ if $[i]\psi \in L(s)$ or $\neg[i]\psi \in L(s)$ then step (2) of the construction of the set of PC-tableaux for $L(s)$ ensures that for $[i]\varphi \in sub(\psi)$ either $[i]\varphi \in L(s)$ or $\neg[i]\varphi \in L(s)$.

6. For all states $s \in S$ if $[i]\psi \in L(s)$ then step (2) of the tableau algorithm ensures that for all s' such that $(s, s') \in R_i$, we have $\psi \in s$.

7. For all states $s \in S$ if $\neg[i]\psi \in L(s)$ then step (2) of the tableau algorithm ensures that we construct a R_i successor s' to s such that $\neg\psi \in L(s')$. If s' happens to be deleted during the contraction phase of the tableau algorithm then step (4d) ensures that s is also deleted.

8. For all states $s \in S$ if $[i]\psi \in L(s)$ then the additional alpha rule for KL_n applied during the construction of the set of PC-tableau means that $\psi \in s$.

9. For all states $s \in S$ if $[i]\psi \in L(s)$ then applying step (2) of the tableau algorithm ensures that for all states s' such that $s' \in R_i$, $[i]\psi \in L(s')$. If $[i]\psi \in L(s')$ either $[i]\psi$ was added to s' as $[i]\psi \in L(s)$ and we are done or $[i]\psi$ must be a subformula of φ such that either $[i]\varphi \in L(s)$ or $\neg[i]\varphi \in L(s)$. From step (2) of the construction of PC-tableau either $[i]\psi \in L(s)$ or $\neg[i]\psi \in L(s)$. The latter is not possible as by step (2) of the tableau algorithm $\neg[i]\psi \in L(s')$ and as $[i]\psi \in L(s')$ by assumption state s' is not proper. For the former we have if $[i]\psi \in L(s')$ then $[i]\psi \in L(s)$ and are done.

10. For all states $s \in S$ if $\bigcirc\psi \in L(s)$ then by applying step (3) of the tableau algorithm we construct an η successor state to s that contains ψ.

11. For all states $s \in S$ if $\bigcirc\psi \in L(s)$ then by applying step (3) of the tableau algorithm all η successors we construct to s contain ψ.If for some reason all such states are deleted then s is also deleted during step $(4b)$.

12. If there exists a state $s \in S$ such that $\psi \in L(s)$ and $\neg resolvable(\psi, s, (S, \eta, R_1, \ldots, R_n, L))$ then step $(4a)$ of the tableau algorithm would delete s so that for all states $s \in S$ such that $\psi \in L(s)$, $resolvable(\psi, s, (S, \eta, R_1, \ldots, R_n, L))$.

For BL_n, the conditions (BL_n1)-(BL_n3) are as follows. To see that (BL_n1) holds, observe that if $(s, s'), (s, s'') \in R_i$, then both s' and s'' must have been created from s by step (2) of the tableau algorithm. In particular, it must have been the case that $[i]\psi \in L(s)$. The state that would have been created would contain both ψ and $[i]\psi$. Hence both $\{[i]\psi, \psi\} \subseteq L(s')$ and $\{[i]\psi, \psi\} \subseteq L(s'')$. To see that (BL_n2) holds, observe that if $[i]\psi \in L(s)$, then there must be some $(s, s') \in R_i$ or else s would have been deleted during contraction. To see that (BL_n3) holds, observe that if $(s, s') \in R_i$, then s' must have been created by an application of the step (2) (creating R_i successors) of the tableau algorithm. But in this case, by construction, $[i]\varphi \in L(s')$. ∎

Lemma 4, when taken together with Theorem 2, has the following important corollary, for which the proof is immediate.

Theorem 3 *If $\varphi_0 \in Form(\mathcal{L})$, and tableau algorithm on φ_0 returns a structure $(S, \eta, R_1, \ldots, R_n, L)$ such that $\varphi_0 \in L(s)$, then φ_0 is KL_n satisfiable.*

Next, we must show that if an invocation of the tableau algorithm is unsuccessful, then the input formula to the invocation is KL_n unsatisfiable.

Theorem 4 *If $\varphi_0 \in Form(\mathcal{L})$, the tableau algorithm φ_0 returns a structure $(S, \eta, R_1, \ldots, R_n, L)$, and $\not\exists s \in S$ such that $\varphi_0 \in L(s)$, then φ_0 is KL_n unsatisfiable.*

Proof: There are two possibilities:

1. No such state was ever created.

   But the algorithm *will* attempt to create such a state, and will only fail if the set of PC-tableaux is empty meaning, by Lemma 2, φ_0 is unsatisfiable.

2. Such a state was created, but was subsequently deleted by step (4), the contraction part of the tableau algorithm.

   In this case, let the state that was created be s. There must have been some sequence of states $(s_0, \ldots, s_k)$ such that s_0 was first deleted, leading to the removal of s_1, and so on, until finally $s_k = s$ was deleted. An easy induction on this sequence shows that in this case φ_0 must have been unsatisfiable also.

∎

Theorems 3 and 4 lead immediately to partial correctness.

Theorem 5 *If $\varphi_0 \in Form(\mathcal{L})$, then φ_0 is KL_n satisfiable if, and only if, the tableau algorithm applied to φ_0 returns a structure $(S, \eta, R_1, \ldots, R_n, L)$ in which $\exists s \in S$ such that $\varphi_0 \in L(s)$.*

All that remains for total correctness is to show termination.

Theorem 6 *If $\varphi_0 \in Form(\mathcal{L})$, then the tableau algorithm applied to φ_0 terminates.*

Proof: First note that the PC-tableau algorithm generates sets of formulae by applying alpha and beta rules and filling out these sets with subformulae or their negations where necessary. After a certain number of applications of these rules, we will generate a set of sets of formulae that is either empty, or else contains only atoms and next-time formulae. (We must be careful when applying the alpha, beta and subformula rules, not to apply the rule to a formula more than once, or else we will not terminate.)

Now consider the three stages of the tableau algorithm:

- The first stage builds the set of PC-tableau. As described above this step will terminate.
- It is not immediately obvious that steps (2) and (3) of the tableau algorithm, the creation of R_i and η successors, terminates, as the states that we introduce do not appear to be 'smaller' than those from which they were spawned. However the input formula φ is finite and the set of formulae that may appear in a state is also finite. Thus we do not duplicate states labelled by the same formula, and hence the algorithm terminates. (This is know as the *analytic superformula* property in [11].)
- The final part of the algorithm involves the removal of states from the structure. Eventually, we will either remove all states or reach a point at which no more states will be removed, and hence this step terminates.

∎

Theorems 5 and 6 together imply complete correctness.

4 Worked Examples

In this section, we demonstrate the KL_n and BL_n algorithms via a number of worked examples.

Example 1

For the first example, we shall use the decision procedure to show that the formula

$$([1]\,\Box p) \wedge \Box\neg p \tag{6}$$

is BL_n satisfiable, but KL_n unsatisfiable. Starting with the BL_n construction, we construct the set of PC-tableaux for (6). We apply the alpha rule for conjunction to $([1]\,\Box p) \wedge \Box\neg p$ obtaining the formulae $[1]\,\Box p$ and $\Box\neg p$. Then the alpha rule for $\Box$ is applied to $\Box\neg p$ giving $\neg p$ and $\bigcirc\Box\neg p$. No other alpha, beta, or subformula rules can be applied and we obtain the state, s_1, labelled as follows.

$$L(s_1) = \{([1]\,\Box p) \wedge \Box\neg p, [1]\,\Box p, \Box\neg p, \neg p, \bigcirc\Box\neg p\}$$

Next, belief and temporal successors to s_1 are constructed. First considering belief successors, as $L(s_1)$ contains $[1]\,\Box p$, a state s_2 is constructed containing the PC-tableau expansion of $\{[1]\,\Box p, \Box p\}$. The alpha rule for $\Box$ is applied to $\Box p$ giving p and $\bigcirc\Box p$ and as no other rules can be applied the state s_2 is

$$L(s_2) = \{[1]\,\Box p, \Box p, p, \bigcirc\Box p\}.$$

As $L(s_1)$ contains $\bigcirc\Box\neg p$ temporal expansion of s_1 then begins, resulting in a state s_3 being created containing the PC-tableau of $\{\Box\neg p\}$. Applying the

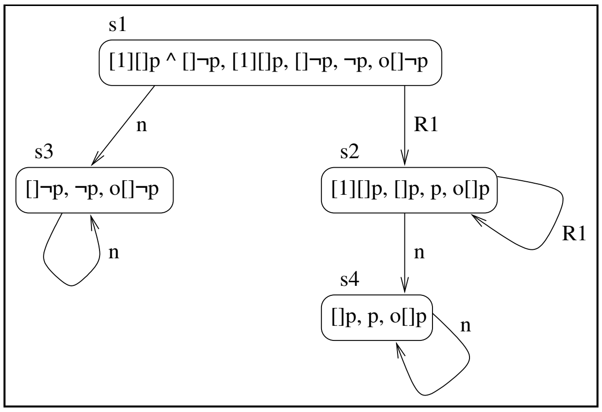

Figure 4: The Structure Returned for Example 1

alpha rule for $\Box$ to $\Box\neg p$ we obtain $\neg p$ and $\bigcirc\Box\neg p$ giving the set s_3 labelled as follows:

$$L(s_3) = \{\Box\neg p, \neg p, \bigcirc\Box\neg p\}.$$

The arc (s_1, s_2) is added to the R_1 relation and (s_1, s_3) is then added to η, the next-time relation.

Next, considering $L(s_2)$ it again contains the agent atom $[1]\Box p$. We try construct the R_1 successor to state s_2. This is formed by constructing the PC-tableau for $\{[1]\Box p, \Box p\}$. Applying the alpha rule for $\Box$ to $\Box p$ we obtain p and $\bigcirc\Box p$. No other rules can be applied so the PC-tableau for the R_1 successor to s_2 is $\{[1]\Box p, \Box p, p, \bigcirc\Box p\}$, i.e. s_2 itself. We add (s_2, s_2) to R_1. For the temporal successors of s_2 we construct the PC-tableau expansion of $next(L(s_2)) = \{\Box p\}$. Applying the alpha rule to $\Box p$ we obtain p and $\bigcirc\Box p$. No other rules can be applied so we have a new set s_4 where

$$L(s_4) = \{\Box p, p, \bigcirc\Box p\}.$$

An arc (s_2, s_4) is added to η. There are no belief successors for s_3 as $L(s_3)$ doesn't contain any formulae of the form $[i]\psi$ or $\neg[i]\psi$. As $next(L(s_3))$ is the same as $next(L(s_1))$ $(= \{\Box\neg p\})$ we add (s_3, s_3) to η.

Looking at $L(s_4)$ as it contains no agent atoms, so no states or arcs are created. The PC-tableau expansion of $next(L(s_4)) = \{\Box p\}$ turns out to be the same as $L(s_4)$, and so an arc (s_4, s_4) is added to η. No further states are available for processing, and so structure expansion ends, and contraction begins. However, no states are deleted and the following structure is returned (see Figure 4).

$$
\begin{array}{rcl}
L(s_1) & = & \{([1]\,\Box p) \wedge \Box\neg p, [1]\,\Box p,\ \Box\neg p, \neg p, \bigcirc\,\Box\neg p\} \\
L(s_2) & = & \{[1]\,\Box p,\ \Box p, p, \bigcirc\,\Box p\} \\
L(s_3) & = & \{\,\Box\neg p, \neg p, \bigcirc\,\Box\neg p\} \\
L(s_4) & = & \{\,\Box p, p, \bigcirc\,\Box p\} \\
R_1 & = & \{(s_1, s_2), (s_2, s_2)\} \\
\eta & = & \{(s_1, s_3), (s_2, s_4), (s_3, s_3), (s_4, s_4)\}
\end{array}
$$

As there is a state (s_1) labelled with (6), we know that (6) must be satisfiable. For the KL_n construction, the PC-tableau expansion of (6) is the set:

$$\{([1]\,\Box p) \wedge \Box\neg p, [1]\,\Box p,\ \Box\neg p,\ \Box p, p, \bigcirc\,\Box p, \neg p, \bigcirc\,\Box\neg p\}.$$

The formulae $[1]\,\Box p$ and $\Box\neg p$ have been generated by applying the alpha rule for conjunction to $([1]\,\Box p) \wedge \Box\neg p$. The formula $\Box p$ is generated when applying the KL_n alpha rule for knowledge modalities to $[1]\,\Box p$. The formulae p and $\bigcirc\,\Box p$ are generated when the alpha rule for $\Box$ is applied to $\Box p$. The formulae $\neg p$ and $\bigcirc\,\Box\neg p$ are generated when the alpha rule for $\Box$ is applied to the formula $\Box\neg p$. This set is obviously improper (as it contains p and $\neg p$), and so construction proceeds no further. The structure returned is empty, and thus (6) is KL_n unsatisfiable.

Example 2

For this example, we use the decision procedure to prove that the schema

$$[i]\,\Box\varphi \Rightarrow \Box[i]\varphi \tag{7}$$

is neither BL_n valid nor BL_n unsatisfiable. To do this, we take an instance of the schema, *viz.*

$$[1]\,\Box p \Rightarrow \Box[1]p \tag{8}$$

and show by refutation that it is not valid, and then show directly that it is satisfiable. We begin by showing that (8) is not valid. Negating (8) gives

$$[1]\,\Box p \wedge \neg\,\Box[1]p. \tag{9}$$

We now generate sets of PC-tableaux for (9). Applying the alpha rule for conjunction to $[1]\,\Box p \wedge \neg\,\Box[1]p$ and pushing the negation through the $\Box$ operator in the second formula we obtain $[1]\,\Box p$ and $\Diamond\neg[1]p$. Next applying the beta rule to $\Diamond\neg[1]p$ we generate two sets: one containing $\neg[1]p$, the other containing $[1]p \wedge \bigcirc\Diamond\neg[1]p$. To the latter we can apply the alpha rule for conjunction to obtain $[1]p$ and $\bigcirc\Diamond\neg[1]p$. As no more alpha or beta rules can be applied to either set we obtain the following two states.

$L(s_1) = \{[1]\,\Box p \wedge \neg\,\Box[1]p, [1]\,\Box p, \Diamond\neg[1]p, \neg[1]p\}$
$L(s_2) = \{[1]\,\Box p \wedge \neg\,\Box[1]p, [1]\,\Box p, \Diamond\neg[1]p, [1]p \wedge \bigcirc\Diamond\neg[1]p, [1]p, \bigcirc\Diamond\neg[1]p\}$

We apply the tableau algorithm to each of these states. We begin with state s_1. It contains the agent atoms $\{[1]\,\Box p, \neg[1]p\}$, and so to construct the R_1 successors of s_1 we construct the PC-tableau for $\{[1]\,\Box p, \neg[1]p,\ \Box p, \neg p\}$. Applying the alpha rule for $\Box$ to $\Box p$ we obtain p and $\bigcirc\,\Box p$ and thus obtain the following set of formulae.

$$\{[1]\,\Box p, \neg[1]p,\ \Box p, \neg p, p, \bigcirc\,\Box p\}$$

As this set of formulae is not proper it is deleted. Next, considering the temporal expansion of s_1 as there are no next-time formulae no η-successors are created. Thus state s_1 is the result of the expansion phase. During contraction we see that $\neg[1]p$ is a member of $L(s_1)$ but there is no R_1 successor of s_1 containing $\neg p$. Thus s_1 is deleted and an empty structure is returned.

Next we apply the tableau algorithm to s_2. First we attempt to construct belief successors for s_2. As $L(s_2)$ contains the agent modalities $\{[1]\,\Box p, [1]p\}$ we construct the PC-tableau for $\{[1]\,\Box p, [1]p,\ \Box p, p\}$. Applying the alpha rule for $\Box$ to $\Box p$ we obtain the state s_3 labelled as follows

$$L(s_3) = \{[1]\,\Box p, [1]p,\ \Box p, p, \bigcirc\,\Box p\}$$

and add (s_2, s_3) to R_1. During temporal expansion, a PC-tableau is generated for the set $next(L(s_2)) = \{\Diamond\neg[1]p\}$. There are two proper PC-tableaux, and thus two states are created, s_4 and s_5:

$$\begin{aligned} L(s_4) &= \{\Diamond\neg[1]p, \neg[1]p\} \\ L(s_5) &= \{\Diamond\neg[1]p, [1]p \wedge \bigcirc\Diamond\neg[1]p, [1]p, \bigcirc\Diamond\neg[1]p\} \end{aligned}$$

The arcs (s_2, s_4) and (s_2, s_5) are added to η. Continuing processing with s_3 we have s_3 as an R_1 successor of itself and add (s_3, s_3) to R_1. The η successor of s_3 is PC tableau formed from $next(L(s_3)) = \{\,\Box p\}$ that is the state s_6

$$L(s_6) = \{\,\Box p, p, \bigcirc\,\Box p\}$$

so we add (s_3, s_6) to η. Next we process s_4. It contains a negated agent modality, and so a PC-tableau is generated for the set $\{\neg[1]p, \neg p\}$ which results in a single PC-tableau. A state s_7 is created, labelled:

$$L(s_7) = \{\neg[1]p, \neg p\}.$$

The arc (s_4, s_7) is added to R_1. Temporal expansion begins but $next(L(s_4)) = \emptyset$ so s_4 has no η successors. State s_5 is then processed; it contains the agent atom $[1]p$ so the PC-tableau is constructed for $\{[1]p, p\}$, and a new state s_8 is created

$$L(s_8) = \{[1]p, p\}$$

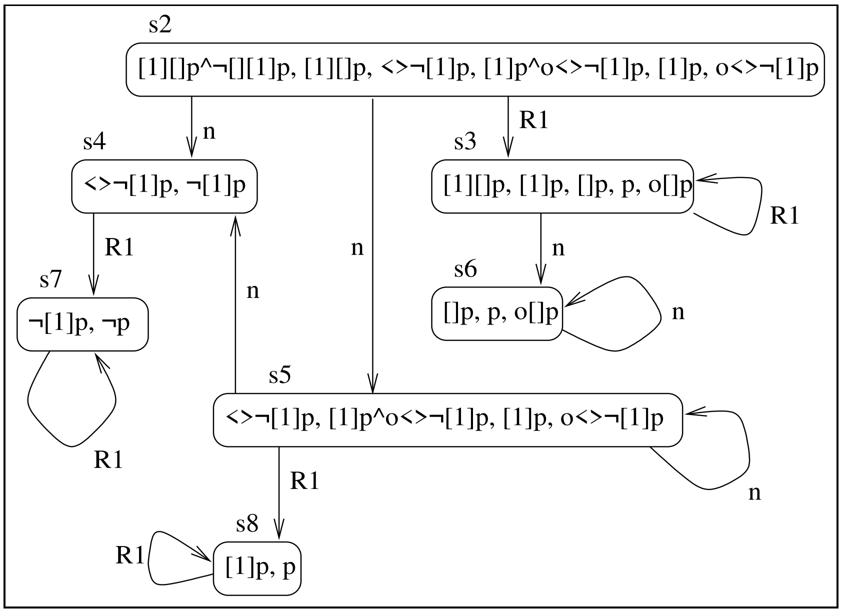

Figure 5: The Structure Returned for Example 2

and (s_5, s_8) is added to R_1. Temporal expansion begins, but $next(L(s_5))$ turns out to be the same as $next(L(s_2))$, and so arcs (s_5, s_4) and (s_5, s_5) are added to η.

State s_6 is then processed. It contains no agent modalities so no edges are added to R_1. As for temporal successors $next(L(s_6)) = next(L(s_3)) = \{\Box p\}$ so the η successor to s_6 is the same as for s_3, i.e. itself and (s_6, s_6) is added to η. States s_7 and s_8 each are belief successors of themselves and have no temporal successors so (s_7, s_7) and (s_8, s_8) are added to R_1 There are no further states to be processed, and so the structure expansion stage finishes, and contraction begins: no states can be deleted so the structure returned is as follows (see Figure 5).

$$
\begin{array}{rcl}
L(s_2) & = & \{[1]\Box p \wedge \neg \Box [1]p, [1]\Box p, \Diamond\neg[1]p, [1]p \wedge \bigcirc\Diamond\neg[1]p, [1]p, \bigcirc\Diamond\neg[1]p\} \\
L(s_3) & = & \{[1]\Box p, [1]p, \Box p, p, \bigcirc\Box p\} \\
L(s_4) & = & \{\Diamond\neg[1]p, \neg[1]p\} \\
L(s_5) & = & \{\Diamond\neg[1]p, [1]p \wedge \bigcirc\Diamond\neg[1]p, [1]p, \bigcirc\Diamond\neg[1]p\} \\
L(s_6) & = & \{\Box p, p, \bigcirc\Box p\} \\
L(s_7) & = & \{\neg[1]p, \neg p\} \\
L(s_8) & = & \{[1]p, p\}
\end{array}
$$

$$
\begin{array}{rcl}
R_1 & = & \{(s_2, s_3), (s_3, s_3), (s_4, s_7), (s_5, s_8), (s_7, s_7), (s_8, s_8)\} \\
\eta & = & \{(s_2, s_5), (s_2, s_4), (s_3, s_6), (s_5, s_4), (s_5, s_5), (s_6, s_6)\}
\end{array}
$$

As there is a state (s_2) such that (9) $\in L(s_2)$, it must be that (9) is BL_n satisfiable, implying that (8) is not BL_n valid, and thus the schema (7) is not BL_n valid.

To show that (7) is BL_n satisfiable, we prove that (8) is BL_n satisfiable. In the interests of brevity, we shall not give details of the construction. Instead, we shall simply note that the expansion stage of one of the PC-tableaux of (8) terminates after generating the following structure.

$$\begin{array}{rcl}
L(s_1) & = & \{[1]\,\Box p \Rightarrow \Box[1]p, \neg[1]\,\Box p\} \\
L(s_2) & = & \{\neg[1]\,\Box p, \Diamond\neg p, \neg p\} \\
L(s_3) & = & \{\neg[1]\,\Box p, \Diamond\neg p, p \wedge \bigcirc\Diamond\neg p, p, \bigcirc\Diamond\neg p\} \\
L(s_4) & = & \{\Diamond\neg p, \neg p\} \\
L(s_5) & = & \{\Diamond\neg p, p \wedge \bigcirc\Diamond\neg p, p, \bigcirc\Diamond\neg p\} \\
R_1 & = & \{(s_1, s_2), (s_1, s_3), (s_2, s_2), (s_2, s_3), (s_3, s_2), (s_3, s_3)\} \\
\eta & = & \{(s_3, s_4), (s_3, s_5), (s_5, s_4), (s_5, s_5)\}
\end{array}$$

No states in the structure are removed during contraction, and as there is at least one state (s_1) containing (8) (recall that we have not expanded all PC-tableaux generated from (8)), it must be that (8) is BL_n satisfiable, and hence (7) is BL_n satisfiable.

Example 3

In the spirit of Example 2, we shall use the decision procedure to show that schema (10) is neither BL_n valid nor BL_n unsatisfiable.

$$\Box[i]\varphi \Rightarrow [i]\,\Box\varphi \tag{10}$$

We proceed as before, by showing in turn that an instance of the schema, *viz.*

$$\Box[1]p \Rightarrow [1]\,\Box p \tag{11}$$

is neither BL_n valid nor BL_n unsatisfiable. To show that (11) is not BL_n valid, we negate it, and show that the resultant formula, (12), is BL_n satisfiable.

$$\Box[1]p \wedge \neg[1]\,\Box p \tag{12}$$

We start by generating the set of PC-tableaux for (12). There is one state, leading to the creation of state s_1:

$$L(s_1) \quad = \quad \{\,\Box[1]p \wedge \neg[1]\,\Box p,\ \Box[1]p, \neg[1]\,\Box p, [1]p, \bigcirc\,\Box[1]p\}$$

We begin processing with state s_1. As this set contains the agent modalities $\{\neg[1]\,\Box p, [1]p\}$, we construct a PC-tableau for the following set:

$$\{\neg[1]\,\Box p, [1]p, \Diamond\neg p, p\}.$$

This leads to one set, and the creation of state s_2 labelled:

$$L(s_2) = \{\neg[1]\,\Box p, [1]p, \Diamond\neg p, p \wedge \bigcirc\Diamond\neg p, p, \bigcirc\Diamond\neg p\}.$$

The arc (s_1, s_2) is added to R_1. During temporal expansion, we have

$$next(L(s_1)) = \{\,\Box[1]p\}.$$

This leads to a single PC-tableau, and a new state s_3:

$$L(s_3) = \{\,\Box[1]p, [1]p, \bigcirc\,\Box[1]p\}.$$

The arc (s_1, s_3) is added to η. State s_2 is then processed; as it contains the agent modalities $\neg[1]\,\Box p$ and $[1]p$ we construct the PC-tableau for the following set:

$$\{\neg[1]\,\Box p, [1]p, \Diamond\neg p, p\}.$$

However, we end up with the PC-tableau being the same as that for $L(s_2)$, so we do not create any new states, but simply add the arc (s_2, s_2) to R_1. For the temporal expansion of s_2, we have

$$next(L(s_2)) = \{\Diamond\neg p\}$$

and so we construct a PC-tableau for this set. This gives two sets, leading to the creation of states s_4 and s_5:

$$\begin{aligned} L(s_4) &= \{\Diamond\neg p, \neg p\} \\ L(s_5) &= \{\Diamond\neg p, p \wedge \bigcirc\Diamond\neg p, p, \bigcirc\Diamond\neg p\}. \end{aligned}$$

Arcs (s_2, s_4) and (s_2, s_5) are added to η. State s_3 is then processed; as it contains the agent modality $[1]p$ we create a new state

$$L(s_6) = \{[1]p, p\}$$

and add (s_3, s_6) to R_1. During the temporal expansion stage, the only PC-tableau generated for the set $next(L(s_3))$ is found to be labelled with $L(s_3)$, and so an arc (s_3, s_3) is added to η. During the processing of state s_4, no arcs are made or states created. State s_5 is then processed; it contains no agent atoms, so temporal expansion begins. During this stage, it is found that $next(L(s_5)) = next(L(s_2))$, and so arcs (s_5, s_4) and (s_5, s_5) are added to η. State s_6 is then processed. As it contains $[1]p$ it has itself as an R_1 successor and we add (s_6, s_6) to R_1. As $next(L(s_6)) = \emptyset$ there are no η successors to s_6. Structure contraction then begins, but no states are removed. The final state of the structure built from s_1 is as follows (see Figure 6).

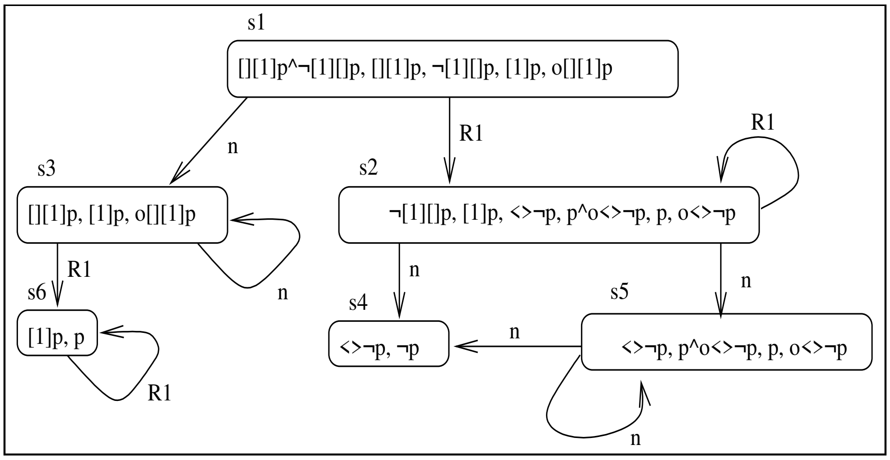

Figure 6: The Structure Returned for Example 3

$$
\begin{array}{rcl}
L(s_1) & = & \{\square[1]p \wedge \neg[1]\square p,\ \square[1]p, \neg[1]\square p, [1]p, \bigcirc\square[1]p\} \\
L(s_2) & = & \{\neg[1]\square p, [1]p, \Diamond\neg p, p \wedge \bigcirc\Diamond\neg p, p, \bigcirc\Diamond\neg p\} \\
L(s_3) & = & \{\square[1]p, [1]p, \bigcirc\square[1]p\} \\
L(s_4) & = & \{\Diamond\neg p, \neg p\} \\
L(s_5) & = & \{\Diamond\neg p, p \wedge \bigcirc\Diamond\neg p, p, \bigcirc\Diamond\neg p\} \\
L(s_6) & = & \{[1]p, p\} \\
R_1 & = & \{(s_1, s_2), (s_2, s_2), (s_3, s_6), (s_6, s_6)\} \\
\eta & = & \{(s_1, s_3), (s_2, s_4), (s_2, s_5), (s_3, s_3), (s_5, s_4), (s_5, s_5)\}
\end{array}
$$

As (12) $\in L(s_1)$, it must be that (12) is satisfiable, so (11) is not valid, and thus (10) is not valid.

To show that (10) is satisfiable, we show that (11) is satisfiable. The set of PC-tableaux generated from (11) has several states one of which is shown below, as state s_1.

$$L(s_1) = \{\square[1]p \Rightarrow [1]\square p, [1]\square p\}$$

State s_1 is processed; as it contains agent atoms we construct a PC-tableau for the set $\{[1]\square p,\ \square p\}$, which results in a state s_2 being created, labelled as follows:

$$L(s_2) = \{[1]\square p,\ \square p, p, \bigcirc\square p\}$$

The arc (s_1, s_2) is added to R_1. The temporal expansion of s_1 begins. As $next(L(s_1)) = \emptyset$ there are no temporal successors to s_1. State s_2 is then processed. It contains the same agent atoms as state s_1 so has itself as an R_1 successor and (s_2, s_2) is added to R_1. As $next(L(s_2)) = \{\Box p\}$ we construct the PC-tableau for this set creating state

$$L(s_3) = \{\Box p, p, \bigcirc\Box p\}$$

and add (s_2, s_3) to η.

State s_3 is then processed. It contains no agent atoms so no new states are added or arcs to R_1. As $next(L(s_3)) = next(L(s_2))$, the η successor for s_3 is the same as that for s_2 so we add (s_3, s_3) to η. This concludes the expansion stage of the algorithm, leaving the following structure.

$$\begin{array}{rcl} L(s_1) & = & \{\Box[1]p \Rightarrow [1]\Box p, [1]\Box p, \} \\ L(s_2) & = & \{[1]\Box p, \Box p, p, \bigcirc\Box p\} \\ L(s_3) & = & \{\Box p, p, \bigcirc\Box p\} \\ R_1 & = & \{(s_1, s_2), (s_2, s_2)\} \\ \eta & = & \{(s_2, s_3), (s_3, s_3)\} \end{array}$$

No states are deleted during contraction, and as $L(s_1)$ contains (11), it must be that (11) is satisfiable.

5 Refinements

The algorithm we have presented above is naive, in the sense that it is computationally greedy. However, there are several obvious ways in which the algorithm could be improved in this respect. Here, we identify some of the more obvious of these.

5.1 Deleting non-minimal sets

The number of sets of PC-tableau can be reduced by adding the following step at the end of that algorithm. Doing some 'pre-processing' of the sets generated at this stage can subsequently reduce the overall amount of computational effort required by the algorithm.

4. *Deleting non-minimal sets.* Delete any $\Delta' \in \mathcal{F}$ such that there exists $\Delta'' \in \mathcal{F}$ and $\Delta'' \subset \Delta'$.

The justification for this step is outlined below. Let the tableau algorithm including this step be called the reduced tableau algorithm.

Lemma 5 *The tableau algorithm for the formula φ_0 returns a non-empty tableau structure T if and only if the reduced tableau algorithm for the formula φ_0 returns a non-empty tableau structure T'.*

Proof: Let φ_0 be a satisfiable formula, then the tableau algorithm constructs a tableau structure T. Let s_i for $i = 1, \ldots, m$ for $m \geq 2$ be the set of PC-tableaux constructed for φ_0 and $s_2 \subset s_1$. As the tableau algorithm returns a non-empty structure then some s_i must be in this structure. If s_1 is not in the structure then its deletion makes no difference as the algorithm still returns a non-empty structure. However if s_1 is in the structure returned then this means that the conjunction of the set of formulae contained in s_1 is satisfiable so the conjunction of any subset of formulae in s_1 is satisfiable, in particular the subset that makes up s_2. So a tableau constructed from s_2 must also be in the structure and the deletion of s_1 makes no difference.

Conversely, let φ_0 be an unsatisfiable formula. Then the tableau algorithm returns an empty structure. That is, all the sets s_i above must have been deleted, in particular s_2 has been deleted so that the conjunction of formulae in s_2 must be unsatisfiable. Thus as s_2 is a subset of s_1 the conjunction of subformulae in s_1 must also be unsatisfiable so its deletion makes no difference. ∎

5.2 Constructing R_i successors for the KL_n tableau

In the tableau algorithm when constructing R_i successors we state that *if s contains no such formulae but $[i]\psi \in L(s)$ then construct the set of formulae* $\Delta = \{\chi \mid [i]\chi \in L(s)\} \cup \{[i]\chi \mid [i]\chi \in L(s)\}$. We only really need to do this for BL_n tableau because we need to ensure seriality. With KL_n tableau the reflexivity requirement on models of KL_n formula is captured by the additional alpha formula. When constructing a model from the tableau we take the reflexive transitive closure of R_i thus ensuring that every state containing formulae of the form $[i]\chi$ has itself as a successor. Note that if this refinement is adopted for KL_n tableau we must also delete step (4d) of the tableau algorithm.

5.3 The creation of PC-tableaux

The algorithm for the creation of sets of PC-tableaux for a set of formulae allows the application of steps (1) and (2) in any order. However if we first apply any alpha rules, then any beta rules and finally fill out with subformulae or their negations where necessary fewer sets will be created before deletions.

The number of possible states constructed in the algorithm is at worst of the order of the number of propositionally consistent subsets of $sub(\varphi)$. The large number of states constructed in this type of verification has been considered for example in [17] and has led to work considering on-the-fly verification [1, 19]. The number of states created may be able to be reduced in many cases by the application of this type of on-the-fly verification.

6 Closing Remarks

In this paper, we have defined KL_n, a temporal logic of knowledge, and BL_n, a temporal logic of belief, and have presented and proved correct basic tableau-based decision procedures for both of these logics. An implementation of these decision procedures, written in Eiffel, is described in [4]. There are many ways in which we hope to extend this work in the future: (i) undertake a systematic analysis of *interaction axioms*, which characterise interactions between time and knowledge/belief (cf. [2]), and extend the decision procedure to deal with these axioms; (ii) use the implementation developed to further investigate the efficiency of the method; (iii) extend the decision procedures to the first-order case; (iv) develop different proof techniques for KL_n and BL_n (such as translation methods); and finally, (v) compare the different methods.

Acknowledgements

The authors would like to thank Ian Elliot and two anonymous referees for their comments on the paper.

References

[1] G. Bhat, R. Cleavland, and O. Grumberg. Efficient On-the-Fly Model Checking for CTL*. In *Proceedings of the 10th International Conference on Logic in Computer Science (LICS'95)*, pages 388–397. IEEE Computer Society Press, 1995.

[2] L. Catach. Normal multimodal logics. In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88)*, pages 491–495, St. Paul, MN, 1988.

[3] B. Chellas. *Modal Logic: An Introduction.* Cambridge University Press: Cambridge, England, 1980.

[4] I. Elliot. An Implementation of a Tableau-Based Decision Procedure for Temporal Logics of Knowledge and Belief. Master's thesis, Department of Computing, Manchester Metropolitan University, October 1996.

[5] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 996–1072. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1990.

[6] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge.* MIT Press, 1995.

[7] R. Fagin, J. Y. Halpern, and M. Y. Vardi. What can machines know? On the properties of knowledge in distributed systems. *Journal of the ACM*, 39(2):328–376, 1992.

[8] M. Fisher, M. Wooldridge, and C. Dixon. A resolution-based proof method for temporal logics of knowledge and belief. In D. M. Gabbay and H. J. Ohlbach, editors, *Proceedings of the International Conference on Formal and Applied Practical Reasoning (FAPR-96)*, Springer-Verlag, June 1996.

[9] M. Fitting. *Proof Methods for Modal and Intuitionistic Logics.* D. Reidel Publishing Company: Dordrecht, The Netherlands, 1983. (Synthese library volume 169).

[10] R. Goré. *Cut-Free Sequent Systems for Propositional Normal Modal Logics.* PhD thesis, Computer Laboratory, University of Cambridge, U.K., June 1992. (Technical report no. 257).

[11] R. Goré. Tableaux Methods for Modal and Temporal Logics. Technical Report TR-ARP-15-95, Australian National University, May 1995. To appear as chapter in the Handbook of Tableau Methods.

[12] G. D. Gough. Decision procedures for temporal logic. Master's thesis, Department of Computer Science, Manchester University, Oxford Rd., Manchester M13 9PL, UK, October 1984.

[13] J. Y. Halpern. Using reasoning about knowledge to analyze distributed systems. *Annual Review of Computer Science*, 2:37–68, 1987.

[14] J. Y. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54:319–379, 1992.

[15] J. Y. Halpern and M. Y. Vardi. The complexity of reasoning about knowledge and time. I. Lower bounds. *Journal of Computer and System Sciences*, 38:195–237, 1989.

[16] S. Kraus and D. Lehmann. Knowledge, belief and time. *Theoretical Computer Science*, 58:155–174, 1988.

[17] Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems: Safety.* Springer-Verlag, New York, 1995.

[18] J.-J. C. Meyer and W. van der Hoek. *Epistemic Logic for Computer Science and Artificial Intelligence*, volume 41 of *Cambridge Tracts in Theoretical Computer Science.* Cambridge University Press: Cambridge, England, 1995.

[19] M.Y. Vardi R. Gerth, D. Peled and P. Wolper. Simple On-the-fly Automatic Verification of Linear Temporal Logic. In *Proceedings of the International Symposium on Protocol Specification, Testing and Verification*, 1995.

[20] A. S. Rao and M. P. Georgeff. Formal Models and Decision Procedures for Multi-Agent Systems. Technical Report 61, Australian Artificial Intelligence Institute, June 1995.

[21] Y. Shoham. *Reasoning About Change: Time and Causation from the Standpoint of Artificial Intelligence.* The MIT Press: Cambridge, MA, 1988.

[22] R. M. Smullyan. Analytic Cut. *Journal of Symbolic Logic*, 33(4), 1968.

[23] R. M. Smullyan. *First-Order Logic.* Springer-Verlag: Heidelberg, Germany, 1968.

[24] R. van der Meyden. Axioms for Knowledge and Time in Distributed Systems with Perfect Recall. In *Proceedings of the Ninth IEEE Symposium on Logic in Computer Science*, pages 448–457, 1994.

[25] P. Wolper. The tableau method for temporal logic: An overview. *Logique et Analyse*, 110–111, 1985.

[26] M. Wooldridge. *The Logical Modelling of Computational Multi-Agent Systems.* PhD thesis, Department of Computation, UMIST, Manchester, UK, October 1992. (Also available as Technical Report MMU–DOC–94–01, Department of Computing, Manchester Metropolitan University, Chester St., Manchester, UK).

[27] M. Wooldridge and M. Fisher. A decision procedure for a temporal belief logic. In D. M. Gabbay and H. J. Ohlbach, editors, *Temporal Logic — Proceedings of the First International Conference (LNAI Volume 827)*, pages 317–331. Springer-Verlag: Heidelberg, Germany, July 1994.