Tableaux for Temporal Logics of Knowledge: Synchronous Systems of Perfect Recall or No Learning

Clare Dixon, Cláudia Nalon, Michael Fisher
Department of Computer Science
The University of Liverpool,
Liverpool L69 7ZF, UK
{clare,claudia,michael}@csc.liv.ac.uk

Abstract

The paper describes tableaux based proof methods for temporal logics of knowledge allowing interaction axioms between the modal and temporal components. Such logics can be used to specify systems that involve the knowledge of processes or agents and which change over time, for example agent based systems or knowledge games. The interaction axioms allow the description of how knowledge evolves over time and makes reasoning in such logics theoretically more complex. Completeness arguments for the tableaux are discussed.

1 Introduction

We describe a tableau-based proof method for temporal logics of knowledge allowing interaction axioms between the modal and temporal parts. Temporal logics of knowledge have been used for the specification and verification of distributed and multi-agent systems [8, 10, 18], analysing security protocols [24, 9], knowledge games such as the muddy children problem [3] etc.

Tableau based proof methods involve the systematic construction of a structure, for a formula φ, from which a model for φ can be obtained. To show a formula φ valid the tableau algorithm is applied to $\neg\varphi$. If the resultant structure is empty then $\neg\varphi$ is unsatisfiable and φ valid. The logic we consider is linear time temporal logic with finite past and infinite future combined with the modal logic S5, allowing the interactions synchrony and perfect recall and synchrony and no learning [6, 13, 14, 15]. Such interactions allow us to consider how knowledge evolves over time. For example, for certain classes of systems, interaction occurs between knowledge and time so that, for example, statements such as *"if an agent knows that in the next moment p holds then, in the next moment, the agent knows p holds"* are valid. Systems with this property have been termed as *synchronous* with *no forgetting*, or *unbounded memory* or *perfect recall* [6, 13, 14, 15]. For other systems we may want the converse of this. Systems with this property are known as *synchronous* with *no learning* [6, 13, 14, 15]. For many of the interactions described in the literature [14, 15] this increases the complexity of the logic (sometimes dramatically) even for the single agent case. Thus, proof methods for such systems are even rarer than those for the basic logic of time and knowledge (i.e. the fusion of these logics).

Unusually for tableau systems our method uses a satisfiability preserving translation into a normal form, SNF_K, for temporal logics of knowledge. This normal form removes all temporal operators apart from $\bigcirc$ (in the next moment in time) and $\Diamond$ (sometime in the future) and the resulting clauses hold in all states. We extend this normal form slightly and due to the structure of the clauses obtained we can easily add new clauses resulting from the explicit application of the axiom for synchrony and perfect recall or synchrony and no learning. This ensures the constraints of the particular interaction axiom are added to the clauses in the normal form before the construction of the tableau itself. The tableau construction is similar to that for temporal logics of knowledge (see for example [3]) but due to the translation to normal form, which removes many of the temporal operators and requires formulae to be in a particular form, we require the definition of fewer alpha and beta rules. Further, complex deletion or expansion rules are not required to satisfy the grid-like synchrony and perfect recall or no learning requirement as we have explicitly applied this information earlier in the method. As part of the completeness proof we show we can construct models from non-empty tableau constructions.

The structure of the paper is as follows. In Section 2 we give the syntax and semantics of *KL* a temporal logic of knowledge and define the synchrony and perfect recall and synchrony and no learning restrictions. In Section 3 we define the normal form SNF_K and its extensions ESNF_{SPR} and ESNF_{SNL}. In Section 4 we show how to translate any temporal logic of knowledge formula into SNF_K, ESNF_{SPR} and ESNF_{SNL}. In Section 5 we present tableau algorithms for the single agent case of systems of synchrony and perfect recall and synchrony and no learning. Examples are given in Section 6 and proofs of correctness of our algorithms are given in Section 7. Finally we present conclusions, discuss related work and mention future work in Section 8.

2 Syntax and Semantics

We first give the syntax and semantics of a logic *KL*, a *temporal logic of knowledge* where the modal relation K is restricted to be an equivalence relation [15]. The temporal component is interpreted over a discrete linear model of time with finite past and infinite future; an obvious choice for such a flow of time is $(\mathbb{N}, <)$, i.e., natural numbers ordered by the usual 'less than' relation. This logic has been studied in detail [15] and is the most commonly used temporal logic of knowledge.

2.1 Syntax

Formulae are constructed from a set $\mathcal{P} = \{p, q, r, \ldots\}$ of *primitive propositions*. The language *KL* contains the standard propositional connectives $\neg$ (not), $\vee$ (or), $\wedge$ (and) and $\Rightarrow$ (implies). For knowledge we introduce a unary modal connective K, where a formula $K\phi$ is read as "the agent knows ϕ". For the temporal dimension we take the usual set of future-time temporal connectives $\bigcirc$ (*next*), $\diamondsuit$ (*sometime* or *eventually*), $\square$ (*always*), $\mathcal{U}$ (*until*) and $\mathcal{W}$ (*unless* or *weak until*).

The set of well-formed formulae of *KL*, WFF_K is defined as follows:

- $\mathbf{false}$, $\mathbf{true}$ and any element of $\mathcal{P}$ is in WFF_K;
- if A and B are in WFF_K then so are $\neg A$, $A \vee B$, $A \wedge B$, $A \Rightarrow B$, KA, $\diamondsuit A$, $\square A$, $A\,\mathcal{U}\,B$, $A\,\mathcal{W}\,B$, $\bigcirc A$.

A *literal* is either p, or $\neg p$, where $p \in \mathcal{P}$. A *modal literal* is either Kl or $\neg Kl$, where l is a literal.

2.2 Semantics

First, we assume that the world may be in any of a set, S, of *states*. A *timeline* t, is an infinitely long, linear, discrete sequence of states, indexed by the natural numbers. Let *TLines* be the set of all timelines. A *point* q, is a pair $q = (t, u)$, where $t \in$ *TLines* is a timeline and $u \in \mathbb{N}$ is a temporal index into t. Let *Points* be the set of all points. A *valuation* π, is a function $\pi : Points \times \mathcal{P} \rightarrow \{T, F\}$. A *model* M, is a structure $M = \langle TL, R, \pi \rangle$, where:

- $TL \subseteq$ *TLines* is a set of timelines, with a distinguished timeline t_0;
- R is the agent accessibility relation over $Points$, i.e., $R \subseteq Points \times Points$ where R is an equivalence relation;
- π is a valuation.

As usual, we define the semantics of the language via the satisfaction relation '$\models$'. For *KL*, this relation holds between pairs of the form $\langle M, q \rangle$ (where M is a model and q is a point in $TL \times \mathbb{N}$), and formulae in WFF_K. The rules defining the satisfaction relation are given below (we omit many propositional connectives as they are standard).

$\langle M, (t,u) \rangle \models \mathbf{true}$

$\langle M, (t,u) \rangle \not\models \mathbf{false}$

$\langle M, (t,u) \rangle \models p$ iff $\pi((t,u), p) = T$ (where $p \in \mathcal{P}$)

$\langle M, (t,u) \rangle \models \neg A$ iff $\langle M, (t,u) \rangle \not\models A$

$\langle M, (t,u) \rangle \models A \vee B$ iff $\langle M, (t,u) \rangle \models A$ or $\langle M, (t,u) \rangle \models B$

$\langle M, (t,u) \rangle \models \bigcirc A$ iff $\langle M, (t, u+1) \rangle \models A$

$\langle M, (t,u) \rangle \models \square A$ iff $\forall u' \in \mathbb{N}$, if $(u \leq u')$ then $\langle M, (t, u') \rangle \models A$

$\langle M, (t,u) \rangle \models \diamondsuit A$ iff $\exists u' \in \mathbb{N}$ such that $(u \leq u')$ and $\langle M, (t, u') \rangle \models A$

$\langle M, (t,u) \rangle \models A\,\mathcal{U}\,B$ iff $\exists u' \in \mathbb{N}$ such that $(u' \geq u)$ and $\langle M, (t, u') \rangle \models B$, and $\forall u'' \in \mathbb{N}$, if $(u \leq u'' < u')$ then $\langle M, (t, u'') \rangle \models A$

$\langle M, (t,u) \rangle \models A\,\mathcal{W}\,B$ iff $\langle M, (t,u) \rangle \models A\,\mathcal{U}\,B$ or $\langle M, (t,u) \rangle \models \square A$

$\langle M, (t,u) \rangle \models KA$ iff $\forall t' \in TL, \forall u' \in \mathbb{N}$, if $((t,u),(t',u')) \in R$ then $\langle M, (t', u') \rangle \models A$

For any formula A, if there is some model M and timeline t such that $\langle M, (t, 0) \rangle \models A$, then A is said to be satisfiable. If for any formula A, for all models M there exists a timeline t such that $\langle M, (t, 0) \rangle \models A$ then A is said to be valid. Note that validity and satisfiability are evaluated at the beginning of time. Informally, this corresponds to the point at which computations (for which we are interested in proving properties) would start.

Definition 1 *Systems with synchrony and perfect recall are those such that for all points* $(r, n+1)$, *if* $((r, n+1), (s, m+1)) \in R$ *then* $((r, n), (s, m)) \in R$ *and* $m = n$.

Definition 2 *Systems with synchrony and no learning are those such that for all points* (r,n)*, if* $((r,n),(s,m)) \in R$ *then* $((r,n+1),(s,m+1)) \in R$ *and* $m = n$*.*

We denote the logics that capture the class of (single agent) systems with synchrony and perfect recall as KL^{SPR} and with synchrony and no learning as KL^{SNL}. We sometimes abbreviate the expression synchrony and perfect recall by SPR and synchrony and no learning by SNL.

It has been shown that the usual axiomatisations of propositional reasoning, knowledge (i.e. S5 modal logic), linear-time temporal logic plus the axiom

$$\vdash K\bigcirc\varphi \Rightarrow \bigcirc K\varphi$$

give a sound and complete axiomatisation for systems of synchrony and perfect recall [14]. Similarly the usual axiomatisations of propositional reasoning, knowledge, linear-time temporal logic plus the axiom

$$\vdash \bigcirc K\varphi \Rightarrow K\bigcirc\varphi$$

give a sound and complete axiomatisation for systems of synchrony and no learning [14]. In the single agent case the complexity of the validity problem for synchrony and perfect recall is double exponential time, and for synchrony and no learning is EXPSPACE. For more than one agent each problem is nonelementary [15].

3 A Normal Form for Temporal Logics of Knowledge

Due to the complexity of these logics we first translate formulae into a normal form, SNF_K, which has the property that for any formula φ is satisfiable if and only if its translation into SNF_K is satisfiable. Further, we give extensions to SNF_K required for the tableau constructions for systems of synchrony and no learning and synchrony and perfect recall.

3.1 Normal Form

Formulae in *KL* can be transformed into a normal form SNF_K (Separated Normal Form for temporal logics of knowledge). SNF was described in [7] for linear-time temporal logics and was extended for temporal logics of knowledge, SNF_K, in [3]. The form of SNF_K we give here is more restricted than in [3] as there initial, step and modal clauses may have disjunctions on the right hand side and step, sometime and modal clauses may have conjunctions on the left hand side. For the purposes of the normal form we introduce a symbol $\mathbf{start}$ such that

$$\langle M, (t_0, 0)\rangle \models \mathbf{start}\,.$$

This is not necessary but allows the normal form to be implications. The translation to SNF_K removes many of the temporal operators that do not appear in the normal form by rewriting them using their fixpoint definitions. Also the translation uses the renaming technique [20] where complex subformulae are replaced by new propositions and the truth value of these propositions is linked to the formulae they replaced in all states. In linear time temporal logic this is achieved by ensuring that such formulae are in the scope of a $\square$ operator, i.e. they hold at all reachable states. For temporal logics of knowledge we introduce the $\square^*$ operator, which allows nesting of K, $\square$ and *always in the past* operators to achieve a similar result. First we define the *always in the past* operator, $\square^-$, as

$$\begin{array}{l}\langle M, (t,u)\rangle \models \square^- A \text{ iff } \forall u' \in \mathbb{N},\ \text{ if } (0 \le u' \le u) \\ \qquad \text{then } \langle M, (t,u')\rangle \models A\end{array}$$

and then let

$$\square^{\pm} A \Leftrightarrow\ \square A \wedge \square^- A.$$

Finally, the $\square^*$ operator is defined as the maximal fixpoint of

$$\square^*\phi \ \Leftrightarrow\ \square^{\pm}(\phi \wedge K\square^*\phi).$$

3.2 Definition of the Normal Form

Formulae in SNF_K are of the general form $\square^* \bigwedge_j T_j$ where each T_j, known as a *clause*, must be in one of the varieties described below.

$$\begin{array}{rcll}
\mathbf{start} & \Rightarrow & l & (\textit{initial}\text{ clause}) \\
k & \Rightarrow & \bigcirc l & (\textit{step}\text{ clause}) \\
k & \Rightarrow & \Diamond l & (\textit{sometime}\text{ clause}) \\
k & \Rightarrow & m & (\textit{modal}\text{ clause}) \\
\mathbf{true} & \Rightarrow & \bigvee_{b=1}^{r} l_b & (\textit{literal}\text{ clause})
\end{array}$$

where k, l and l_b are literals and m are modal literals. The outer '$\square^*$' operator and conjunction that surrounds clauses is usually omitted and we consider just the set of clauses T_j.

To apply the tableau algorithm, additional forms of step clauses are allowed. A SPR step clause is of the form

$$K\bigvee_{a=1}^{g} k_a \Rightarrow \bigcirc K\left(\bigvee_{b=1}^{r} l_b\right)$$

and a SNL step clause is of the form

$$\neg K \neg \bigwedge_{a=1}^{g} k_a \Rightarrow \bigcirc \neg K \neg \left(\bigwedge_{b=1}^{r} l_b\right)$$

A formula is in Extended SNF_K for Synchrony and Perfect Recall, ESNF_{SPR}, if it is in SNF_K or it is a SPR step clause. A formula is in Extended SNF_K for Synchrony and No Learning, ESNF_{SNL}, if it is in SNF_K or it is a SNL step clause.

Informally, we add additional step clauses, SPR and SNL step clauses (given above) to which the synchrony and perfect recall (respectively synchrony and no learning) axiom has been applied. Due to the translation into normal form, SNF_K clauses hold at all states, denoted by the external $\Box^*$ operator. Due to the properties of the $\Box^*$ operator (see [4]) for any clause C, KC must also hold in each state. Thus for synchrony and perfect recall, for each step clause (and additional clauses we must construct from the step clauses), we distribute the K operator applied to clauses over the implication. On the right hand side of the resultant formula the sequence of K and $\bigcirc$ operators exactly matches the left hand side of the synchrony and perfect recall algorithm, so it can be applied. The argument for synchrony and no learning is similar except we use the contrapositive of the step clauses and axiom. Rather than just applying this process of distributing the K operator over implication and applying the relevant axiom to step clauses only we must apply it to particular combinations of step clauses. This process is described in more detail below.

First we take non-empty subsets of the step clauses and disjoin (respectively conjoin) the left and right hand sides to form disjunctive (respectively conjunctive) step clauses (see Section 4.2). Thus for systems of synchrony and perfect recall, for a step or disjunctive step clause or the form $\varphi \Rightarrow \bigcirc\psi$, we distribute an external K due to the $\Box^*$ operator over the implication, i.e. from $K(\varphi \Rightarrow \bigcirc\psi)$ to $K\varphi \Rightarrow K\bigcirc\psi$ and apply the SPR axiom ($K\bigcirc\chi \Rightarrow \bigcirc K\chi$) to the right hand side obtaining $K\varphi \Rightarrow \bigcirc K\psi$. For synchrony and no learning the axiom is $\bigcirc K\chi \Rightarrow K\bigcirc\chi$ and we take the contrapositive $\neg K\bigcirc\chi \Rightarrow \neg\bigcirc K\chi$ which is equivalent to $\neg K\bigcirc\chi \Rightarrow \bigcirc\neg K\chi$. For any step or conjunctive step clause $\varphi \Rightarrow \bigcirc\psi$, take the contrapositive $\neg\bigcirc\psi \Rightarrow \neg\varphi$. Distributing an external K, $K(\neg\bigcirc\psi \Rightarrow \neg\varphi)$, over the implication we obtain $K\neg\bigcirc\psi \Rightarrow K\neg\varphi$. Again taking the contrapositive we obtain $\neg K\neg\varphi \Rightarrow \neg K\neg\bigcirc\psi$, and we apply the contrapositive of the axiom to the right hand side to obtain $\neg K\neg\varphi \Rightarrow \bigcirc\neg K\neg\psi$.

Further, as any sometime formula, $\Diamond l$, is equivalent to $l \vee \bigcirc\Diamond l$ we add additional step and literal clauses such that when the additional SPR or SNL step clauses are added, this accounts for the application of the relevant axiom to this equivalence. So for any sometime clause $k \Rightarrow \Diamond l$ we rename $\Diamond l$ by a new proposition $e_{\Diamond l}$, giving $k \Rightarrow e_{\Diamond l}$ and add $e_{\Diamond l} \Rightarrow \Diamond l$. To capture the *unwinding* (i.e. the equivalence above) of $\Diamond l$ we add the clausal form of the following $e_{\Diamond l} \Rightarrow l \vee \bigcirc e_{\Diamond l}$ to the clause set.

4 Translation into the Normal Form

The translation to SNF_K is carried out by renaming complex subformulae with new propositional variables and linking the truth of the subformula to that of the proposition at all moments. Temporal operators are removed using their fixpoint definitions. Classical and temporal equivalences (see for example [5]) are also used to get formulae into the correct format. See [3, 7] for more details.

Example As a small example we show how $K\bigcirc p \wedge \neg\bigcirc Kp$ is translated into SNF_K. First we rename the formula by new proposition x and obtain the clauses

$$\begin{array}{rrcl} 1. & \mathbf{start} & \Rightarrow & x \\ 2. & x & \Rightarrow & K\bigcirc p \wedge \neg\bigcirc Kp \end{array}$$

The conjunction in clause 2 is split giving

$$\begin{array}{rrcl} 3. & x & \Rightarrow & K\bigcirc p \\ 4. & x & \Rightarrow & \neg\bigcirc Kp \end{array}$$

The subformulae $\bigcirc p$ from clause 2 is renamed by a new proposition y. The right hand side of clause 4 is rewritten as the equivalent formula $\bigcirc\neg Kp$ and $\neg Kp$ from the rewritten version of clause 4 is renamed by new proposition z.

$$\begin{array}{rrcl} 5. & x & \Rightarrow & Ky \\ 6. & y & \Rightarrow & \bigcirc p \\ 7. & x & \Rightarrow & \bigcirc z \\ 8. & z & \Rightarrow & \neg Kp \end{array}$$

Clauses 1,5–8 are in the normal form SNF_K.

4.1 Replacement of Sometime Clauses

Let T be a set of SNF_K clauses and $R \subseteq T$ be the set of sometime clauses in T. For each $k \Rightarrow \Diamond l \in R$ replace $k \Rightarrow \Diamond l$ by $e_{\Diamond l} \Rightarrow \Diamond l$ and add

$$\begin{array}{rcl} \mathbf{true} & \Rightarrow & \neg k \vee e_{\Diamond l} \\ t_{\Diamond l} & \Rightarrow & \bigcirc e_{\Diamond l} \\ \mathbf{true} & \Rightarrow & t_{\Diamond l} \vee \neg e_{\Diamond l} \vee l. \end{array}$$

Let the new set of clauses be $Rep(T)$.

4.2 Addition of Clauses to form ESNF_{SPR} and ESNF_{SNL}

Let $Rep(T)$ be a set of SNF_K clauses where the sometime clauses have been replaced and new step and literal clauses added as described in Section 4.1 and $S \subseteq Rep(T)$ be the set of step clauses in $Rep(T)$. The set of disjunctive clauses of $Rep(T)$, $Dis(Rep(T))$ (respectively conjunctive clauses of $Rep(T)$, $Con(Rep(T))$) is the set of clauses

$$\bigvee_h k_h \Rightarrow \bigcirc \bigvee_j l_j \qquad \left(\bigwedge_h k_h \Rightarrow \bigcirc \bigwedge_j l_j\right)$$

such that for each subset X of S, $X \neq \emptyset$,

- each k_h is on the left hand side of a step clause in X;
- each l_j is on the right hand side of a step clause in X;
- there is no k on the left hand side of a step clause in X without a corresponding k_h;
- there is no l on the right hand side of a step clause in X without a corresponding l_j.

Simplify $Dis(Rep(T))$ or $Con(Rep(T))$ and remove any clauses whose right hand side is equivalent to true. The set of SPR step clauses for $Dis(Rep(T))$, $SPR(Dis(Rep(T)))$, and the set of SNL step clauses for $Con(Rep(T))$, $SNL(Con(Rep(T)))$, are given in Figure 1.

5 Tableau Algorithms

Here we present tableau based decision procedures for the logics KL^{SPR} and KL^{SNL}. That is, given a formula φ of KL^{SPR} or KL^{SNL} the algorithms will decide whether φ is valid or not for the respective logic. Tableau is a refutation method, i.e. to show φ is valid we negate φ and try to construct a structure from which a model can be constructed. If the structure constructed is empty it means that the negated formula is unsatisfiable and therefore φ is valid. If the structure is non-empty it can be used to construct a model for the negation of the formula, i.e. $\neg\varphi$ is satisfiable and φ is not valid.

Definition 3 *A set of formulae, Δ, is proper, denoted $proper(\Delta)$, if and only if* $\mathbf{false} \notin \Delta$ *and if* $\varphi \in \Delta$ *then* $\neg\varphi \notin \Delta$.

Definition 4 *Let T be a set of formulae. The set $next(T)$ is defined as $next(T) = \{\varphi \mid \bigcirc\varphi \in T\}$.*

Definition 5 *Let T be a set of formulae. The set $possible(\neg K\varphi, T)$ is defined as $possible(\neg K\varphi, T) = \{\neg\varphi\} \cup \{\psi \mid K\psi \in T\} \cup \{K\psi \mid K\psi \in T\} \cup \{\neg K\psi \mid \neg K\psi \in T\}$.*

Definition 6 *Let T be a set of formulae. The set $know(T)$ is defined as $know(T) = \{\psi \mid K\psi \in T\} \cup \{K\psi \mid K\psi \in T\} \cup \{\neg K\psi \mid \neg K\psi \in T\}$*

Alpha Rules

α	α_1	α_2
$\phi \wedge \psi$	ϕ	ψ
$\neg(\phi \vee \psi)$	$\neg\phi$	$\neg\psi$

Beta Rules

β	β_1	β_2
$\phi \vee \psi$	ϕ	ψ
$\phi \Rightarrow \psi$	$\neg\phi$	ψ
$\neg(\phi \wedge \psi)$	$\neg\phi$	$\neg\psi$
$\Diamond\phi$	ϕ	$\neg\phi \wedge \bigcirc\Diamond\phi$

Delta Rules

δ	δ_1
$K\phi$	ϕ
$\neg\neg\phi$	ϕ
$\neg\mathbf{true}$	$\mathbf{false}$
$\neg\mathbf{false}$	$\mathbf{true}$

Figure 2. Tableau Rules

Next we provide alpha and beta rules with which to apply to formulae in ESNF$_{SPR}$ or ESNF$_{SNL}$. These are given in Figure 2. This set is smaller than what is required for temporal logics or combined temporal and modal logics because due to the translation to SNF$_K$ most temporal operators (except $\bigcirc$ and $\Diamond$) have been removed and clauses are in a restricted form. The last group of rules, we have called delta rules, have just one consequence.

Then we construct the *Propositional Tableaux* for a set of *KL* formulae and ESNF$_{SPR}$ clauses (respectively for a set of *KL* formulae and ESNF$_{SNL}$ clauses). This involves satisfying the set of ESNF$_{SPR}$ or ESNF$_{SNL}$ clauses by applying beta rules and then applying alpha, beta or delta rules to the resultant formulae. For any modal formula $K\varphi$ (or $\neg K\varphi$) occurring on the right hand side of a modal clause, or left hand side of a SPR step clause or SNL step clause we add either $K\varphi$ or its negation (or equivalent, where $\neg\neg K\varphi$ is associated with $K\varphi$). Finally any sets that contain **false** or a formula and its negation are deleted.

5.1 Construction of Propositional Tableau

Propositional tableaux are constructed from a set, T, of ENSF$_{SPR}$ or ENSF$_{SPR}$ clauses. Remove any initial clauses from T. Let Δ_0 be a set of formulae and $\mathcal{F} = \{\Delta_0\}$. The construction of the set of propositional tableau for $\mathcal{F}$ and T is as follows.

1. *Satisfying literal clauses.* For each literal clause $\mathbf{true} \Rightarrow \bigvee_{b=1}^{r} l_b$ in T, for each $\Delta \in \mathcal{F}$ let $\mathcal{F} = \mathcal{F} \setminus \{\Delta\} \cup \{\Delta \cup \{\bigvee_{b=1}^{r} l_b\}\}$

$$SPR(Dis(Rep(T))) = \left\{ K\left(\bigvee_h k_h\right) \Rightarrow \bigcirc K\left(\bigvee_j l_j\right) \mid \bigvee_h k_h \Rightarrow \bigcirc \bigvee_j l_j \in Dis(Rep(T))\right\}$$

$$SNL(Con(Rep(T))) = \left\{ \neg K\neg\left(\bigwedge_h k_h\right) \Rightarrow \bigcirc\neg K\neg\left(\bigwedge_j l_j\right) \mid \bigwedge_h k_h \Rightarrow \bigcirc \bigwedge_j l_j \in Con(Rep(T))\right\}$$

Figure 1. SPR and SNL Step Clauses

2. *Adding modal formulae and satisfying step, sometime or modal clauses.*

- for each step clause in T of the form $k \Rightarrow \bigcirc l$ for each $\Delta \in \mathcal{F}$, let $\mathcal{F} = \mathcal{F} \setminus \{\Delta\} \cup \{\Delta \cup \{\neg k\}\} \cup \{\Delta \cup \{k, \bigcirc l\}\}$.
- for each sometime clause in T of the form $e_{\Diamond l} \Rightarrow \Diamond l$ for each $\Delta \in \mathcal{F}$, let $\mathcal{F} = \mathcal{F} \setminus \{\Delta\} \cup \{\Delta \cup \{\neg e_{\Diamond l}\}\} \cup \{\Delta \cup \{e_{\Diamond l}, \Diamond l\}\}$.
- For each modal clause in T of the form $k \Rightarrow Kl$ for each $\Delta \in \mathcal{F}$, let $\mathcal{F} = \mathcal{F} \setminus \{\Delta\} \cup \{\Delta \cup \{Kl\}\} \cup \{\Delta \cup \{\neg k, \neg Kl\}\}$.
- For each modal clause in T of the form $k \Rightarrow \neg Kl$, for each $\Delta \in \mathcal{F}$, let $\mathcal{F} = \mathcal{F} \setminus \{\Delta\} \cup \{\Delta \cup \{\neg Kl\}\} \cup \{\Delta \cup \{\neg k, Kl\}\}$.
- For each SPR step clause in T of the form $K\varphi \Rightarrow \bigcirc K\psi$ for each $\Delta \in \mathcal{F}$ let $\mathcal{F} = \mathcal{F} \setminus \{\Delta\} \cup \{\Delta \cup \{K\varphi, \bigcirc K\psi\}\} \cup \{\Delta \cup \{\neg K\varphi\}\}$.
- For each SNL step clause in T of the form $\neg K\neg\varphi \Rightarrow \bigcirc\neg K\neg\psi$ for each $\Delta \in \mathcal{F}$ let $\mathcal{F} = \mathcal{F} \setminus \{\Delta\} \cup \{\Delta \cup \{K\neg\varphi\}\} \cup \{\Delta \cup \{\neg K\neg\varphi, \bigcirc\neg K\neg\psi\}\}$.

3. *Applying alpha and beta rules.*

(a) If $\varphi \in \Delta$ is an α formula let $\mathcal{F} = \mathcal{F} \setminus \{\Delta\} \cup \{\Delta \cup \{\alpha_1, \alpha_2\}\}$

(b) If $\varphi \in \Delta$ is an β formula let $\mathcal{F} = \mathcal{F} \setminus \{\Delta\} \cup \{\Delta \cup \{\beta_1\}\} \cup \{\Delta \cup \{\beta_2\}\}$

(c) If $\varphi \in \Delta$ is an δ formula let $\mathcal{F} = \mathcal{F} \setminus \{\Delta\} \cup \{\Delta \cup \{\delta_1\}\}$

4. *Deleting sets that are not proper.* Delete any $\Delta \in \mathcal{F}$ such that $proper(\Delta)$ is false.

Note that step 2 is the equivalent of both applying beta formulae to modal or SPR (SNL) step clauses and ensuring $K\varphi$ or $\neg K\varphi$ is added to each propositional tableau constructed for clauses with $K\varphi$ on the left or right hand side. Explicitly adding $K\varphi$ or $\neg K\varphi$ to the propositional tableau for modal or SPR/SNL clauses containing $K\varphi$ or $\neg K\varphi$ prevents us from having to return to previously constructed states to add modal formulae. For example, consider a modal clause $k \Rightarrow Kl$. This clause might be satisfied in the state we are currently constructing, state s, by adding $\neg k$ to the label of the state. Take some state s' in the same equivalence class as s where k is forced to hold (for example to satisfy the formula $\neg K\neg k$ in s). To satisfy the above modal clause Kl must also be added to the label of s'. As s and s' are in the same equivalence class then s should also contain Kl and there is no guarantee of this without returning to s to explicitly add Kl.

For step and sometime clauses we only explicitly add formulae with future time commitments ($\bigcirc l$, $\Diamond l$) when we have to, i.e. when the left hand side of the clause is satisfied.

Definition 7 *A structure H is a tuple $H = (S, E, G, \eta, L)$ where: S is a set of states; E is a set of equivalence classes; $G \subseteq S \times E$ is a set of state equivalence class pairs; $\eta \subseteq G \times G$ is a binary relation on G; and $L : S \to \mathcal{P}(KL)$ takes each state and labels it with a set of temporal logic of knowledge formulae.*

Definition 8 *If $H = (S, E, G, \eta, L)$ is a structure, $(s, e) \in G$ where $s \in S$ and $e \in E$, and η^* is the reflexive transitive closure of η then $\Diamond l$ is resolvable in H from (s, e) (notation $resolvable(\Diamond l, (s, e), H)$ if there exists $(s', e') \in G$ such that $((s, e), (s', e')) \in \eta^*$ and $l \in L(s')$.*

Next we show how to construct a tableau for a formula φ. If we want to show φ valid we negate, apply the algorithm to $\neg\varphi$ and if the algorithm returns *unsuccessful* then $\neg\varphi$ is unsatisfiable and φ is valid.

5.2 Tableau for Synchrony and Perfect Recall or No Learning

Given the formulae φ_0 of KL^{SPR} or KL^{SNL} to be shown satisfiable, perform the following steps.

1. Translate into normal form $\tau_0(\varphi_0)$. Let T_0 be the set of formulae in normal form.

2. For each sometime clause in T_0 replace the sometime clause and add new step and literal clauses as described in Section 4.1 giving $Rep(T_0)$.

3. For synchrony and perfect recall construct $Dis(Rep(T_0))$ and $SPR(Dis(Rep(T_0)))$. Let $T = Rep(T_0) \cup SPR(Dis(Rep(T_0)))$. For synchrony and no learning construct $Con(Rep(T_0))$ and $SNL(Con(Rep(T_0)))$. Let $T = Rep(T_0) \cup SNL(Con(Rep(T_0)))$.

4. Construct Δ where $l \in \Delta$ iff $\mathbf{start} \Rightarrow l \in T$ and construct the set of propositional tableau $\mathcal{F}$ for Δ and T. Call these the *initial states* of the tableau. For each $\Delta_i \in \mathcal{F}$ repeat steps (5)–(7) below, until none apply and then apply step (8).

5. Let $S = E = G = \eta = L = \emptyset$. For some member $\Delta' \in \mathcal{F}$ add a new state s to S, labelled by $L(s) = \Delta'$, and add e to E and (s, e) to G where e is a new equivalence class. Call e the initial equivalence class.

6. *Satisfying $\neg K\varphi$ formulae.*

   For any state (s, e) labelled by formulae $L(s)$, where $L(s)$ is proper and a propositional tableau, for each formula of the form $\neg K\psi \in L(s)$ create the set of formulae $P_{\neg K\psi} = possible(\neg K\psi, L(s))$. For each $P_{\neg K\psi}$ construct $\mathcal{F}$, the set of propositional tableaux for $P_{\neg K\psi}$ and T. For each member $\Delta' \in \mathcal{F}$ if $\exists s'' \in S$ such that $\Delta' = L(s'')$ then add (s'', e) to G, otherwise add a new state s' to S, labelled by $L(s') = \Delta'$, and add (s', e) to G.

7. *Creating η successors.*

   For any state (s, e) labelled by formulae $L(s)$, where $L(s)$ is proper and a propositional tableau, if $\bigcirc\psi \in L(s)$ create the set of formulae $\Delta = next(L(s))$. Construct $\mathcal{F}$ the set of propositional tableaux for Δ and T. For each member $\Delta' \in \mathcal{F}$ if $\exists s'' \in S$ such that $\Delta' = L(s'')$ and $(s'', e'') \in G$ then add $((s, e), (s'', e''))$ to η, otherwise add a new state s' to the set of states S, and new equivalence class e' to E where $L(s') = \Delta'$ and add (s', e') to G and $((s, e), (s', e'))$ to η.

8. *Contraction.*

   Delete any state-equivalence class pair (s, e) where

   (a) there exists $\Diamond\psi \in L(s)$ such that $\neg resolvable(\Diamond\psi, (s, e), H)$; or

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

   (c) there exists $\psi \in L(s)$ such that ψ is of the form $\neg K\chi$ and $\nexists(s', e) \in G$ such that $\neg\chi \in L(s')$.

   until no further deletions are possible. Delete any $s \in S$ and $(s, \Delta) \in L$ such that there is no $(s, e) \in G$.

The tableau algorithm is termed *successful* if and only if at least one of the initial states remains undeleted. If the tableau applied to φ_0 is successful then φ_0 is satisfiable, otherwise if the tableau is unsuccessful then φ_0 is unsatisfiable.

6 Examples

Example 1. First we show that the axiom for synchrony and perfect recall $K\bigcirc p \Rightarrow \bigcirc Kp$ is valid in KL^{SPR}. We negate and show the negation is unsatisfiable i.e. the tableau algorithm applied to $K\bigcirc p \wedge \neg\bigcirc Kp$ is unsuccessful. First we translate to the normal form obtaining

$$\begin{array}{rcl} \mathbf{start} & \Rightarrow & x \\ x & \Rightarrow & Ky \\ y & \Rightarrow & \bigcirc p \\ x & \Rightarrow & \bigcirc z \\ z & \Rightarrow & \neg Kp \end{array}$$

where x, y and z are new propositions (step 1 of the synchrony and perfect recall tableau algorithm). Let us call this set of clauses T. Next we would replace any eventualities (step 2 of the synchrony and perfect recall tableau algorithm) but there are none. Using step 3 of the algorithm we construct $Dis(T)$ and $SPR(Dis(T))$ which are respectively

$$\{y \Rightarrow \bigcirc p, x \Rightarrow \bigcirc z, x \vee y \Rightarrow \bigcirc(p \vee z)\}$$

and

$$\{Ky \Rightarrow \bigcirc Kp, Kx \Rightarrow \bigcirc Kz, K(x \vee y) \Rightarrow \bigcirc K(p \vee z)\},$$

adding the latter to T. Next (step 4) we construct propositional tableau from $\mathcal{F} = \{x\}$. Let

$$B = \{x, Ky, y, \bigcirc p, \bigcirc z, \bigcirc Kp\}.$$

The set of propositional tableaux (having deleted any non proper sets) for $\{x\}$ and T are:

$$\begin{array}{l} \{B \cup \{Kp, \neg z, Kx, \bigcirc Kz, K(x \vee y), \bigcirc K(p \vee z)\}, \\ \;B \cup \{Kp, \neg z, Kx, \bigcirc Kz, \neg K(x \vee y)\}, \\ \;B \cup \{Kp, \neg z, \neg Kx, K(x \vee y), \bigcirc K(p \vee z)\}, \\ \;B \cup \{Kp, \neg z, \neg Kx, \neg K(x \vee y)\}, \\ \;B \cup \{\neg Kp, Kx, \bigcirc Kz, K(x \vee y), \bigcirc K(p \vee z)\}, \\ \;B \cup \{\neg Kp, Kx, \bigcirc Kz, \neg K(x \vee y)\}, \\ \;B \cup \{\neg Kp, \neg Kx, K(x \vee y), \bigcirc K(p \vee z)\}, \\ \;B \cup \{\neg Kp, \neg Kx, \neg K(x \vee y)\} \qquad\qquad \} \end{array}$$

Pick one of these, Δ and let $S = \{s\}$, $L(s) = \Delta$, $E = \{e\}$ and $G = (s, e)$. When we construct η successors to any states labelled by these sets of formulae they will contain $\{p, z, Kp\}$, from applying $next$ to B, as well as other formulae. Constructing propositional tableau for any set of formulae containing this set will include $\neg Kp$, from applying the modal clause expansion rules (step 2 of the construction of propositional tableau algorithm) to the clause $z \Rightarrow \neg Kp$. This set contains both Kp and $\neg Kp$ which is improper. Hence, any η successors to states labelled by the above sets of formulae will be deleted and the initial states will all be deleted. So the tableau algorithm is unsuccessful as the tableau constructed from each initial state is empty. So $K\bigcirc p \wedge \neg\bigcirc Kp$ is unsatisfiable and $K\bigcirc p \Rightarrow \bigcirc Kp$ is valid.

Example 2. The axiom for systems of synchrony and no learning is $\bigcirc Kp \Rightarrow K\bigcirc p$. We negate to obtain $\bigcirc Kp \wedge \neg K\bigcirc p$ and show the output from the *KL*^{SNL} tableau algorithm is unsuccessful. First we translate into normal form obtaining

$$\begin{array}{rcl} \mathbf{start} & \Rightarrow & x \\ x & \Rightarrow & \bigcirc y \\ y & \Rightarrow & Kp \\ x & \Rightarrow & \neg K\neg z \\ z & \Rightarrow & \bigcirc\neg p \end{array}$$

where x, y and z are new propositions. Let this set be T. Next we construct $Con(T)$ and then $SNL(Con(T))$ which is

$$\begin{array}{lrcll} \{ & \neg K\neg x & \Rightarrow & \bigcirc\neg K\neg y & \\ & \neg K\neg z & \Rightarrow & \bigcirc\neg Kp & \\ & \neg K\neg(x \wedge z) & \Rightarrow & \bigcirc\neg K\neg(y \wedge \neg p) & \} \end{array}$$

and add the latter to T. Following step 4 of the tableau algorithm for *KL*^{SNL} we construct the set of propositional tableau for $\{x\}$ and T. Each of these contains

$$\{x, \bigcirc y, \neg K\neg z, \bigcirc\neg Kp, \neg K\neg x, \bigcirc\neg K\neg y\}$$

in addition to other formulae to satisfy the remaining clauses in T. For any of these propositional tableau when we construct the η successor (step 7 of the tableau algorithm) they will contain $\{y, \neg Kp, \neg K\neg y\}$. To satisfy the clause $y \Rightarrow Kp$, the set will also contain Kp. As both $\neg Kp$ and Kp are both in this state every η successor of an initial state is improper, therefore each initial state is deleted and the tableau is unsuccessful. Thus $\bigcirc Kp \wedge \neg K\bigcirc p$ is unsatisfiable and $\bigcirc Kp \Rightarrow K\bigcirc p$ valid.

7 Correctness

We now show that the tableau algorithms presented above are correct. Due to lack of space we merely outline the proof arguments. The full proofs are in [4].

As the tableau algorithms work on formulae in normal form we first show that translation to SNF$_K$ and then, that the addition of clauses to form ESNF$_{SPR}$ or ESNF$_{SNL}$ preserves satisfiability.

Theorem 9 *Let ϕ be a well-formed formula in KL^{SPR} (respectively KL^{SNL}) and $\tau(\phi) = \Box^* \bigwedge_i T_i$ where T_i is the set of clauses translated into ESNF$_{SPR}$ (respectively ESNF$_{snl}$). ϕ is satisfiable in KL^{SPR} (respectively KL^{SNL}) if and only if $\tau(\phi)$ is satisfiable in KL^{SPR} (respectively KL^{SNL}).*

This result can be established in a similar manner to that for the translation into the normal form from linear-time temporal logic [7] of that for the fusion of temporal logic and knowledge [3], with some additional arguments that the newly added clauses preserve satisfiability.

Theorem 10 *The tableau algorithm applied to ϕ_0, a formula of KL^{SPR} (respectively KL^{SNL}), returns* successful*, if and only if ϕ_0 is KL^{SPR} (respectively KL^{SNL}) satisfiable.*

Proof [Sketch]
By Theorem 9 we show that the translation to normal form preserves satisfiability. Next, given a successful tableau we show how to use the tableau to construct a model. For synchrony and perfect recall this involves showing for any two states (s', e'), (s'', e') in the same equivalence class where one (say (s', e')) has a predecessor, (s, e), we can add η relations to the tableau providing a predecessor to (s'', e') in the same equivalence class as (s, e) which preserves the required properties of the tableau. From this extended tableau construction we can construct timelines in which satisfy all eventualities, and reconstruct the modal relation such that formulae of the form $\neg K\varphi$ are satisfied and we obtain the required model structure for synchrony and perfect recall . For synchrony and no learning we construct several timelines (from an equivalence class) together and show each satisfies all eventualities. Finally we must show that for states at times greater than 0 with no predecessors we can construct predecessors that satisfy the set of ESNF$_{SNL}$ clauses. This argument depends on the fact that the set of ESNF$_{SNL}$ clauses constructed from a *KL*^{SNL} formula is a particular form with newly introduced propositions on the left hand side of clauses (or in the case of literal clauses each having a negated newly introduced propositional variable on the right hand side).

To show that unsuccessful tableau applied to ϕ_0 means ϕ_0 is unsatisfiable we again use Theorem 9 to show the translation to normal form preserves satisfiability. We show each step in the tableau algorithm preserves satisfiability and that an improper set of formulae is unsatisfiable. The construction of an unsuccessful tableau means there was a chain of deletions from an improper node, back to the initial node meaning the original formula was unsatisfiable.

Termination must occur because the application of the tableau rules to the set of normal form formulae result in a finite set of formulae. Eventually when constructing η successors we must generate a state labelled by formulae we have constructed before. $\Box$

8 Conclusions, Related and Future Work

We have presented a sound, complete and terminating tableau for the single agent case of synchrony and prefect recall and synchrony and no learning. Such interactions are necessary if we want to allow knowledge to evolve over time.

Allowing interaction between knowledge and time makes the complexity of the logic high. Thus, as well as a detailed study of complexity of the method, we need to devise strategies for efficient construction of the tableau to avoid the worst case complexities where possible. For example taking subsets of step clauses to construct SPR or SNL clauses is an exponential procedure. To avoid having to do this exponential construction in every case, we could investigate the following:

- apply the SPR or SNL axiom to single step clauses, construct the tableau and see whether we obtain an empty tableau, if so we are done; otherwise
- construct the full set of SPR or SNL step clauses and update the tableau already constructed.

For the examples in Section 6 we would have derived an empty tableau from just applying the relevant axiom to the individual step clauses. Similarly, while constructing the propositional tableau it may be possible to add what we need and fill in the details later.

Future work involves both the investigation of such strategies, and the use of case studies to see which interactions occur commonly in real world problems. An interaction corresponding to the synchrony and perfect recall axiom has been highlighted as desirable during work with the developers of the multi-agent specification language KARO [26]. We intend to extend the ideas developed in this paper on the single agent version of the logics, to the multi-agent version of this logic, and to other axioms suggested in [12], unique initial state, perfect recall and no learning (without synchrony) [25]. However this will need some thought as for the extension to non-synchronous versions of these systems it is not obvious how to add clauses to the normal form to ensure models are of the correct form and the multi-agent versions of the synchronous logics will require the addition of more complex clauses.

Complete axioms systems and complexity of the satisfiability for a combination of propositional linear and branching time temporal logics with multi-modal S5 allowing a variety of interactions are considered [6, 12, 14, 15]. Resolution based proof methods for the single agent cases of synchrony and perfect recall are given in [2], and for synchrony and no learning [19].

A tableau based proof method for the fusion of PTL plus either S5 or KD45 is given in [28]. This is essentially the combination of tableau methods for propositional linear time temporal logics [27] and that for the modal logics S5 and KD45 [11]. It does not require the translation to any particular normal form. The work on proof methods for BDI-logics given in [21, 22] give a tableau based proof method for the fusion of either linear or branching-time (CTL or CTL*) with the modal logics KD45 for belief, and KD for desire and intention. We presented tableau algorithms (without the use of normal forms or explicit application of the interaction axioms) for temporal logics of knowledge with synchrony and perfect recall or synchrony and no learning in [1] but these were later found to be unsound, prompting this work.

The papers [17, 23] give tableaux for linear time temporal logics combined with description logics. The restrictions on domains such as expanding, or constant are similar to the imposition of interactions we use here. In [16] a method for constructing tableau for first-order temporal logics is given. Here, the use of *surrogates* is similar to the translation to the normal form used in this paper.

Acknowledgements This work was partially supported by EPSRC research grant GR/M44859/02 and a CAPES studentship.

References

[1] C. Dixon and M. Fisher. Tableaux for Synchronous Systems of Knowledge and Time with Interactions. In G. Grahne, editor, *Proceedings of the Sixth Scandinavian Conference on Artificial Intelligence (SCAI)*, August 1997.

[2] C. Dixon and M. Fisher. Clausal Resolution for Logics of Time and Knowledge with Synchrony and Perfect Recall. In H. Wansing and F. Wolter, editors, *Proceedings of ICTL-00 the Third International Conference on Temporal Logic*, Leipzig, Germany, October 2000.

[3] C. Dixon, M. Fisher, and M. Wooldridge. Resolution for Temporal Logics of Knowledge. *Journal of Logic and Computation*, 8(3):345–372, 1998.

[4] C. Dixon, C. Nalon, and M. Fisher. Tableaux for Logics of Time and Knowledge with Interactions Relating to Synchrony. Submitted, 2003.

[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] M. Fisher, C. Dixon, and M. Peim. Clausal Temporal Resolution. *ACM Transactions on Computational Logic*, 2(1):12–56, January 2001.

[8] M. Fisher and M. Wooldridge. On the Formal Specification and Verification of Multi-Agent Systems. *International Journal of Cooperative Information Systems*, 6(1), January 1997.

[9] J. Glasgow, G. MacEwen, and P.Panangaden. A Logic to Reason About Security. *ACM Transactions on Computer Systems*, 10(3):226–264, Aug 1992.

[10] J. Y. Halpern. Using reasoning about knowledge to analyze distributed systems. *Annual Review of Computer Science*, 2, 1987.

[11] J. Y. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54(3):319–379, April 1992.

[12] J. Y. Halpern, R. van der Meyden, and M. Vardi. Complete axiomatizations for reasoning about knowledge and time. Submitted for publication, 1999.

[13] J. Y. Halpern and M. Y. Vardi. The Complexity of Reasoning about Knowledge and Time: Extended Abstract. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, pages 304–315, Berkeley, California, May 1986.

[14] J. Y. Halpern and M. Y. Vardi. The Complexity of Reasoning about Knowledge and Time: Synchronous Systems. Technical Report RJ 6097, IBM Almaden Research Center, San Jose, California, April 1988.

[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] R. Kontchakov, C. Lutz, F. Wolter, and M. Zakharyaschev. Temporalizing tableaux. To appear in Studia Logica, 2003.

[17] C. Lutz, H. Sturm, F. Wolter, and M. Zakharyaschev. Tableaux for temporal description logic with constant domains. In *Automated Reasoning (LNAI volume 2083)*, pages 121–136. Springer Verlag, 2001.

[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, 1995.

[19] C. Nalon, C. Dixon, and M. Fisher. Resolution for synchrony and no learning: Preliminary report. In *Proceedings of the 8th International Workshop on Logic Language, Information and Computation - WoLLIC'2001*, 2001.

[20] D. A. Plaisted and S. A. Greenbaum. A Structure-Preserving Clause Form Translation. *Journal of Symbolic Computation*, 2(3):293–304, September 1986.

[21] A. S. Rao. Decision Procedures for Propositional Linear-Time Belief-Desire-Intention Logics. In M. Wooldridge, K. Fischer, P. Gmytrasiewicz, N. R. Jennings, J. P. Müller, and M. Tambe, editors, *IJCAI-95 Workshop on Agent Theories, Architectures, and Languages*, pages 102–118, Montréal, Canada, August 1995.

[22] A. S. Rao and M. P. Georgeff. Decision procedures for BDI logics. *Journal of Logic and Computation*, 8(3):293–342, 1998.

[23] H. Sturm and F. Wolter. A tableau calculus for temporal description logic: the expanding domain case. *Journal of Logic and Computation*, 2002.

[24] P. Syverson. Adding Time to a Logic of Authentication. In *Proceedings of the First ACM Conference on Computer and Communications Security*, pages 97–101. ACM Press, 1993.

[25] 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.

[26] B. van Linder, W. van der Hoek, and J.J. Ch. Meyer. Formalising abilities and opportunities of agents. Technical Report UU-CS-1998-08, Department of Computer Science,Utrecht University, Centrumgebouw Noord, Padualaan 14, De Uithof, 3584 CH Utrecht, 1998.

[27] P. Wolper. The Tableau Method for Temporal Logic: An Overview. *Logique et Analyse*, 110–111:119–136, June-Sept 1985.

[28] M. Wooldridge, C. Dixon, and M. Fisher. A Tableau-Based Proof Method for Temporal Logics of Knowledge and Belief. *Journal of Applied Non-Classical Logics*, 8(3):225–258, 1998.