RESOLUTION FOR SYNCHRONY AND NO LEARNING

C. NALON, C. DIXON, AND M. FISHER

ABSTRACT. We present a clausal resolution-based method for temporal logics of knowledge with synchrony and no learning. This and related logics admit axioms which include operators from both the temporal and epistemic logics, which enable the description of how knowledge evolves over time. Instead of proposing new resolution rules, further information is added to the set of clauses in order to deal with this particular interaction. We give outline of the proofs of soundness, termination, and completeness.

1. INTRODUCTION

Logics have been used in Computer Science for many years as a natural way for describing properties of complex systems. More recently, there has been an increasing interest in combined modal logics, as different logical languages are more suitable to specify different properties within a system. Typical examples are the specification and verification of distributed systems [5] and the specification of agent-based systems [11, 12]. Given such logical characterisation of a system, it is then desirable to have the appropriate tools in order to verify whether a particular property holds for this system. By verifying that a property holds, we mean to *prove* that the property is a logical consequence of the specification.

There is a wide range of logics that could be chosen in order to model and characterise such complex systems. Moreover, there is a variety of ways of combining the chosen logics. In the following, we concentrate on a particular combination that has been proved useful in modelling distributed and agent-based systems, namely, we are looking at *Propositional Temporal Logics of Knowledge* ($\mathsf{KL}_{(n)}$, for short). In such logics, the dynamic component is described by a propositional linear temporal logic and the informational component is described by a propositional logic of knowledge. When this combination is given by the union of the axiomatic systems of both logics, where the underlying logics are then said to be *independent*, proof methods can be *relatively* easily obtained by taking the union of proof methods for the logics considered alone and making sure that enough information is passed to each component.

Proof methods for combined logics cannot be obtained in a straightforward way, however, when the logics *interact*, that is, when, besides the characterisation of both logics (given by their axiomatic systems), further axioms, including operators of both logics, are needed in order to model a specific situation. Contexts where we are particularly interested in how the knowledge of an agent evolves over time is a typical example where *interaction axioms* are required. Interactions often increase the complexity of the validity problem for the logical language and proof methods for such logics are, to our knowledge, rare.

In this paper, we introduce a proof method for a particular interaction between time and knowledge: *synchrony and no learning*. This property was firstly discussed in the context of blindfold games [9]. Recently, a similar characterisation has proved useful in the description of non-decreasing domains [8]. In such systems, once two situations are indistinguishable to an agent, the agent will never acquire any knowledge that would allow her to distinguish between such situations. Although the complexity of the interacting logic is high (non-elementary, for the multi-agent case), the axiom that expresses this property has a simple form, which allowed us to investigate in detail the requirements for its proof method.

Key words and phrases. Proof theory, interacting modal logics, temporal and epistemic reasoning.

The structure of the paper is as follows. In Section 2, we review the syntax, semantics and the resolution method for the combined logic of time and knowledge. In Section 3, we present a resolution method for synchronous systems with no learning. The resolution method introduces additional information into the set of clauses, instead of introducing new (possibly complicated) resolution inference rules. The multi-agent case is non-trivial, therefore we discuss the issues related to the granularity of the information that needs to be provided in order to obtain completeness for these systems. We discuss the correctness results of the method for the multi-agent case in Section 4. In Section 5, we discuss our results and other directions for future research.

2. Temporal Logics of Knowledge

The syntax of $\mathsf{KL}_{(n)}$ comprises a set of modal operators and a set of temporal operators. Formulae are constructed from a denumerable set, $\mathcal{P} = \{p, q, p', q', \ldots\}$, of propositional symbols; nullary connectives, **true** and **false**; propositional connectives, $\neg$, $\wedge$, $\vee$, $\Rightarrow$, and $\Leftrightarrow$; temporal connectives, $\Diamond$, $\Box$, $\bigcirc$, $\mathcal{U}$, and $\mathcal{W}$; and a set of unary modal operators K_i, for all $i \in \mathcal{A}$, where $\mathcal{A} = \{1, \ldots, n\}$ is the set of agents.

Definition 1. *The set of well-formed formulae* WFF *is defined recursively as follows:*

- *the nullary connectives and propositional symbols are in* WFF*;*
- *if* ϕ *and* φ *are in* WFF*, then so are* $\neg\varphi$, $(\varphi \wedge \phi)$, $(\varphi \vee \phi)$, $(\varphi \Rightarrow \phi)$, $(\varphi \Leftrightarrow \phi)$, $\Diamond\varphi$, $\Box\varphi$, $\bigcirc\varphi$, $(\varphi\,\mathcal{U}\,\phi)$, $(\varphi\,\mathcal{W}\,\phi)$ *and* $K_i\,\varphi$, $\forall i \in \mathcal{A}$.

A *literal* is either a proposition or its negation; a *modal literal* is either $\mathrm{K}_i\,l$ or $\neg\mathrm{K}_i\,l$, where l is a literal and $i \in \mathcal{A}$; and an *eventuality* is in the form $\Diamond l$, where l is a literal.

The semantics of $\mathsf{KL}_{(n)}$ interprets formulae over a set of temporal lines, each of which corresponds to a discrete, linear model of time with finite past and infinite future, together with the agents' accessibility relations $\mathcal{K}_i$. We define a *timeline* t as 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 $t \in TLines$ and $u \in \mathbb{N}$ is a temporal index to t. Let $Points$ be the set of all points. A *model* is a structure $M = \langle TL, \mathcal{K}_1, \ldots \mathcal{K}_n, \pi\rangle$ where $TL \subseteq TLines$ is a set of timelines with a distinguished timeline t_0; $\mathcal{K}_i$, for all $i \in \mathcal{A}$, is the accessibility relation over points, i.e., $\mathcal{K}_i \subseteq Points \times Points$, where each $\mathcal{K}_i$ is an equivalence relation; and π is a function $\pi : Points \times \mathcal{P} \to \{true, false\}$.

Definition 2. *Truth of a formula is given as follows:*

- $\langle M, (t, u)\rangle \models$ **true**
- $\langle M, (t, u)\rangle \models p$ *iff* $\pi(t, u)(p) = true$*, where* $p \in \mathcal{P}$
- $\langle M, (t, u)\rangle \models \neg\varphi$ *iff* $\langle M, (t, u)\rangle \not\models \varphi$
- $\langle M, (t, u)\rangle \models (\varphi \wedge \phi)$ *iff* $\langle M, (t, u)\rangle \models \varphi$ *and* $\langle M, (t, u)\rangle \models \phi$
- $\langle M, (t, u)\rangle \models \bigcirc\varphi$ *iff* $\langle M, (t, u + 1)\rangle \models \varphi$
- $\langle M, (t, u)\rangle \models \varphi\mathcal{U}\phi$ *iff* $\exists k \in \mathbb{N}$, $k \geq u$, $\langle M, (t, k)\rangle \models \phi$ *and* $\forall j \in \mathbb{N}$, $u \leq j < k$, $\langle M, (t, j)\rangle \models \varphi$
- $\langle M, (t, u)\rangle \models K_i\,\varphi$ *iff* $\forall t', u'$*, such that* $((t, u), (t', u')) \in \mathcal{K}_i$, $\langle M, (t', u')\rangle \models \varphi$.

The semantics of the other connectives are given by **false** $\equiv \neg$**true**, $(\varphi \vee \psi) \equiv \neg(\neg\varphi \wedge \neg\psi)$, $(\varphi \Rightarrow \psi) \equiv (\neg\varphi \vee \psi)$, $(\varphi \Leftrightarrow \psi) \equiv ((\varphi \Rightarrow \psi) \wedge (\psi \Rightarrow \varphi))$, $\Diamond\varphi \equiv (\mathbf{true}\,\mathcal{U}\varphi)$, $\Box\varphi \equiv \neg\Diamond\neg\varphi$, and $(\varphi\,\mathcal{W}\,\psi) \equiv (\Box\neg\varphi \vee \varphi\mathcal{U}\psi)$. We write $(t, u) \sim_i (t', u')$, if $((t, u), (t', u')) \in \mathcal{K}_i$. A formula φ is said to be *satisfiable* if there is a model M such that $\langle M, (t_0, 0)\rangle \models \varphi$; a formula φ is *valid* if $\langle M, (t_0, 0)\rangle \models \varphi$, for every model M.

The resolution-based proof method for $\mathsf{KL}_{(n)}$ in [3] combines the inference rules for temporal and multi-modal knowledge logics when considered alone. A formula is first translated into a normal form, called *Separated Normal Form for Logics of Knowledge* (SNF_K). A nullary connective, **start**, which intuitively represents the beginning of time, is introduced. Formally, we have $\langle M, (t, u)\rangle \models$ **start** if, and only if, $t = t_0$ and $u = 0$, where M is a model and (t, u) is a point. Formulae are represented by a conjunction of clauses, which are true in all states, that is, they have the general form $\Box^* \bigwedge_i A_i$, where the universal operator is defined as $\Box^*\varphi \Leftrightarrow \Box^{\pm}(\varphi \wedge C\Box^*\varphi)$ (with $\Box^{\pm}\varphi \Leftrightarrow \Box\varphi \wedge \Box^{-}\varphi$ and $\langle M,(t,u)\rangle \models \Box^{-}\varphi$ if, and only if, $\forall k, k \in \mathbb{N}$, if $0 \leq k \leq u$, then $\langle M,(t,k)\rangle \models \varphi$), C is the common knowledge operator (i.e. $C\varphi \Leftrightarrow E(\varphi \wedge C\varphi)$, where $E\varphi \Leftrightarrow \bigwedge_{i\in\mathcal{A}} \mathrm{K}_i\,\varphi$), and A_i is a clause which is in one of the following forms, where l, l_i, k_i are literals, m_{i_j} are literals or modal literals in the form $\mathrm{K}_i\,l$ or $\neg\mathrm{K}_i\,l$:

$$\begin{array}{llcl}
\textbf{Initial clause:} & \textbf{start} & \Rightarrow & \bigvee_{b=1}^{r} l_b \\[2ex]
\textbf{Sometime clause:} & \bigwedge_{a=1}^{g} k_a & \Rightarrow & \Diamond\, l \\[2ex]
\textbf{Step clause:} & \bigwedge_{a=1}^{g} k_a & \Rightarrow & \bigcirc \bigvee_{b=1}^{r} l_b \\[2ex]
\mathbf{K}_i\ \textbf{-clause:} & \textbf{true} & \Rightarrow & \bigvee_{b=1}^{r} m_{i_b} \\[2ex]
\textbf{Literal clause:} & \textbf{true} & \Rightarrow & \bigvee_{b=1}^{r} l_b
\end{array}$$

Transformation into the SNF_K, whose satisfiability preserving transformation rules are given in [4] and [3], depends on three main operations: the renaming of complex subformulae by introducing new variables whose truth value is linked to the formula they replace in all states; the removal of temporal operators; and classical style rewrite operations. Once a formula has been transformed into SNF_K, the resolution method can be applied. The method consists of two main procedures: the first performs initial, modal and step resolution; the second performs temporal resolution. Each procedure is performed until a contradiction (either **true** $\Rightarrow$ **false** or **start** $\Rightarrow$ **false**) is generated or no new clauses can be generated. In the following l, l_i are literals; m_i are literals or modal literals; D, D' are disjunctions of literals; M, M' are disjunction of literals or modal literals; and C, C' are conjunctions of literals.

Initial Resolution is applied to clauses that hold at the beginning of time:

$$\begin{array}{lrcl}
\text{[IRES1]} & \textbf{true} & \Rightarrow & (D \vee l) \\
 & \textbf{start} & \Rightarrow & (D' \vee \neg l) \\ \hline
 & \textbf{start} & \Rightarrow & (D \vee D')
\end{array}
\qquad
\begin{array}{lrcl}
\text{[IRES2]} & \textbf{start} & \Rightarrow & (D \vee l) \\
 & \textbf{start} & \Rightarrow & (D' \vee \neg l) \\ \hline
 & \textbf{start} & \Rightarrow & (D \vee D')
\end{array}$$

Modal Resolution is applied between clauses referring to the same agent (i.e. two K_i -clauses; a literal and a K_i -clause; or two literal clauses):

$$\begin{array}{lrcl}
\text{[MRES1]} & \textbf{true} & \Rightarrow & (M \vee m_i) \\
 & \textbf{true} & \Rightarrow & (M' \vee \neg m_i) \\ \hline
 & \textbf{true} & \Rightarrow & (M \vee M')
\end{array}
\qquad
\begin{array}{lrcl}
\text{[MRES2]} & \textbf{true} & \Rightarrow & (M \vee \mathrm{K}_i\,l) \\
 & \textbf{true} & \Rightarrow & (M' \vee \mathrm{K}_i\,\neg l) \\ \hline
 & \textbf{true} & \Rightarrow & (M \vee M')
\end{array}$$

$$\begin{array}{lrcl}
\text{[MRES3]} & \textbf{true} & \Rightarrow & (M \vee \mathrm{K}_i\,l) \\
 & \textbf{true} & \Rightarrow & (M' \vee \neg l) \\ \hline
 & \textbf{true} & \Rightarrow & (M \vee M')
\end{array}
\qquad
\begin{array}{lrcl}
\text{[MRES5]} & \textbf{true} & \Rightarrow & (D \vee \mathrm{K}_i\,l_1 \vee \mathrm{K}_i\,l_2 \vee \ldots) \\ \hline
 & \textbf{true} & \Rightarrow & (D \vee l_1 \vee l_2 \vee \ldots)
\end{array}$$

$$\begin{array}{lrcl}
\text{[MRES4]} & \textbf{true} & \Rightarrow & (M \vee \neg\mathrm{K}_i\,l) \\
 & \textbf{true} & \Rightarrow & (M' \vee l) \\ \hline
 & \textbf{true} & \Rightarrow & (M \vee mod_i(M'))
\end{array}
\qquad
\text{where}\quad
\begin{array}{rcl}
mod_i(A \vee B) & = & mod_i(A) \vee mod_i(B) \\
mod_i(\mathrm{K}_i\,l) & = & \mathrm{K}_i\,l \\
mod_i(\neg\mathrm{K}_i\,l) & = & \neg\mathrm{K}_i\,l \\
mod_i(l) & = & \neg\mathrm{K}_i\neg\,l
\end{array}$$

The first inference rule corresponds to classical resolution. MRES2 is justified by the axiom D, i.e. $\vdash\ \mathrm{K}_i\,\varphi \Rightarrow \neg\mathrm{K}_i\neg\,\varphi$, for any formula φ. The rules MRES3 and MRES5 by the axiom T, i.e. $\vdash\ \mathrm{K}_i\,\varphi \Rightarrow \varphi$, for any formula φ. The rule MRES4 is justified by the external common knowledge operator surrounding each clause. Recall that each clause in SNF_K is of the form

$\square^* \bigwedge_i A_i$. By modal reasoning, this implies $\mathrm{K}_i \bigwedge_i A_i$, for each agent $i \in \mathcal{A}$. A literal or K_i-clause as **true** $\Rightarrow (D' \vee l)$ is, then, an abbreviated form of $\mathrm{K}_i\,(\mathbf{true} \Rightarrow (D' \vee l))$. By propositional reasoning, this formula can be rewritten as $(\mathrm{K}_i\,(\neg D' \Rightarrow l))$. From the axiom K, the knowledge operator can be distributed over the implication, resulting in $(\mathrm{K}_i\,\neg D' \Rightarrow \mathrm{K}_i\,l)$. Rewriting this formula back into the normal form results in **true** $\Rightarrow \neg\mathrm{K}_i\neg\, D' \vee \mathrm{K}_i\,l$. The resolution inference rule is, then, applied between the clauses containing the complementary modal literals $\neg\mathrm{K}_i\,l$ (from the first premise) and $\mathrm{K}_i\,l$ (from the transformation of the second premise). The function mod_i generates the clausal form of $\neg\mathrm{K}_i\neg\, D'$. This function makes use of the axioms K (for distributing the knowledge operator over D'), 4, and 5 (for modal simplification, that is, replacing formulae as $\neg\mathrm{K}_i\neg\,\mathrm{K}_i\,l$ and $\neg\mathrm{K}_i\,\mathrm{K}_i\,l$, where l is a literal, by $\mathrm{K}_i\,l$ and $\neg\mathrm{K}_i\,l$, respectively).

Step Resolution is applied to clauses that hold at the same moment in time:

$$\text{[SRES1]}\quad \begin{array}{rcl} C & \Rightarrow & \bigcirc(D \vee l) \\ C' & \Rightarrow & \bigcirc(D' \vee \neg l) \\ \hline (C \wedge C') & \Rightarrow & \bigcirc(D \vee D') \end{array} \qquad \text{[SRES2]}\quad \begin{array}{rcl} \mathbf{true} & \Rightarrow & (D \vee l) \\ C & \Rightarrow & \bigcirc(D' \vee \neg l) \\ \hline C & \Rightarrow & \bigcirc(D \vee D') \end{array}$$

together with the following simplification rule:

$$\text{[SIMP1]}\quad \begin{array}{rcl} C & \Rightarrow & \bigcirc\mathbf{false} \\ \hline \mathbf{true} & \Rightarrow & \neg C \end{array}$$

Temporal Resolution is applied between an eventuality $\Diamond l$ and a *set of clauses* which forces l always to be false. Schematically, the temporal resolution rule is (where A is a conjunction of formulae, C is a conjunction of literals, and l is a literal):

$$\text{[TRES]}\quad \begin{array}{rcl} A & \Rightarrow & \bigcirc\square\neg l \\ C & \Rightarrow & \Diamond l \\ \hline C & \Rightarrow & (\neg A)\,\mathcal{W}\,l \end{array}$$

The intuition behind this rule is that, once C is satisfied, then there must be no other clauses forcing l to always be false. Thus, A can only be satisfied after the right-hand side of the sometime clause, l, is satisfied. In detail, the temporal resolution rule is (where A_j is a conjunction of literals, B_j is a disjunction of literals, and C and l are as above):

$$\text{[TRES]}\quad \begin{array}{c} A_0 \Rightarrow \bigcirc B_0 \\ \vdots \\ A_n \Rightarrow \bigcirc B_n \\ C \Rightarrow \Diamond l \\ \hline C \Rightarrow \left(\bigwedge_{i=0}^{n} (\neg A_i) \right) \mathcal{W}\,l \end{array} \qquad \text{where} \quad \begin{array}{rcl} \forall i, 0 \le i \le n, \vdash B_i & \Rightarrow & \neg l \\ \forall i, 0 \le i \le n, \vdash B_i & \Rightarrow & \bigvee_{j=0}^{n} A_j \end{array}$$

The set of clauses that satisfy these side conditions are together known as *a loop in* $\neg l$. Algorithms for finding such a loop can be found in [1]. We note that each $A_j \Rightarrow \bigcirc B_j$ are step clauses in *merged* SNF_K, that is, they correspond to a conjunction of step clauses in SNF_K. A translation of the resolvent into the normal form is given below (where t is a new proposition):

$$\begin{array}{rcl} \mathbf{true} & \Rightarrow & (\neg C \vee \neg A_i \vee l) \\ t & \Rightarrow & \bigcirc(\neg A_i \vee l) \\ \mathbf{true} & \Rightarrow & (\neg C \vee t \vee l) \\ t & \Rightarrow & \bigcirc(t \vee l) \end{array}$$

Clauses are kept in their simplest form by performing classical style simplification. Classical subsumption is also applied and valid formulae, as **false** $\Rightarrow \bigcirc A$ and $A \Rightarrow \bigcirc\mathbf{true}$, can be removed during simplification as they cannot contribute to the generation of a contradiction.

3. SYNCHRONOUS SYSTEMS WITH NO LEARNING

We now describe a clausal resolution system for temporal logics of knowledge in synchronous systems with no learning ($\mathsf{KL}_{(n)}^{sync,nl}$). A system is said to be synchronous if the agent has access to a common external clock. Intuitively, if the system is synchronous, the agent *knows* the time. Moreover, this time is common to all agents. The agent is said to have the property of *no learning*, if her knowledge does not increase over time. Formally, in the class of models for synchronous system with no learning, if two points, (s, m) and (t, n), are in the accessibility relation of agent i, i.e. $(s, m) \sim_i (t, n)$, then, because of synchrony, they share the same time index ($m = n$) and, because of no learning, their successors are also indistinguishable to agent i, i.e. $(s, m + 1) \sim_i (t, n + 1)$. Note that the number of states that an agent considers possible stays the same or increases over time and that once the agent considers two timelines indistinguishable, those timelines will always be indistinguishable to her. Figure 1 shows an example of a synchronous system with no learning, where horizontal lines represent timelines, time increases from left to right, the dots represent points, and the dashed lines enclose points in the same equivalence class.

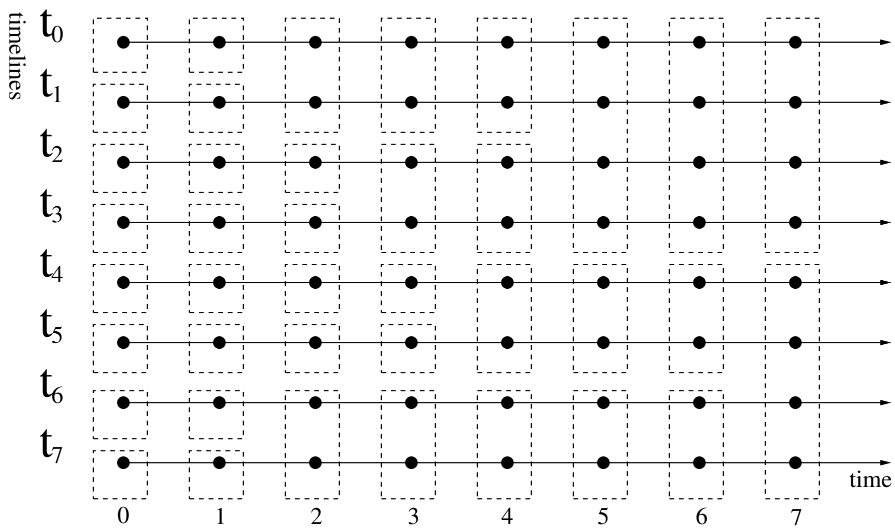

FIGURE 1. Example of a Synchronous System with No Learning

The syntax and semantics for $\mathsf{KL}_{(n)}^{sync,nl}$ are the same as for $\mathsf{KL}_{(n)}$. A complete axiomatisation for $\mathsf{KL}_{(n)}^{sync,nl}$ comprises the set of axioms of PTL, the set of axioms of $\mathsf{S5}_{(n)}$, together with the axioms $\vdash \bigcirc \mathrm{K}_i \varphi \Rightarrow \mathrm{K}_i \bigcirc \varphi$ (SNL), for all agents $i \in \mathcal{A}$ [7]. The validity problem for such class of systems is EXPSPACE for $n = 1$ and non-elementary space for $n \geq 2$ [7].

3.1. **Proof Method.** The general approach for dealing with synchrony and no learning is as follows. Given a set of clauses in SNF_K, we first add some new clauses which ensure that the constraints expressed by the SNL axiom are made explicit before applying the resolution rules presented in the previous section. As making such constraints explicit is the essential part of the proof method, we explain in more detail its motivation.

In resolution-based proof methods, generally speaking, one has to identify complementary formulae (or sets of formulae) in order to apply the inference rules. This procedure can be relatively easy for basic logics. For instance, for propositional logics, there is only one resolution inference rule, which is applied to clauses containing complementary literals, l and $\neg l$. However, for more complex logics, trying to identify complementary formulae can be non-trivial, costly, and often achieved by the introduction of new inference rules. This is the case, for instance, in the modal epistemic case, where several modal (resolution) inference rules are introduced in order to resolve a literal l with its possible complements, namely $\neg l$ and $\mathrm{K}_i \neg l$. For the temporal case, besides the (step resolution) rules to resolve complementary literals which are true at the same moment in time, the temporal resolution rule is introduced, which requires the identification of a set of clauses that imply $\bigcirc \Box \neg l$ to be resolved with an eventuality $\Diamond l$. For the combined logics of knowledge and time, the proof method might seem, at first, rather simple, as it consists of the same inference rules for the logics considered alone. Nevertheless, its apparent simplicity comes from the separation of the different dimensions (via the normal form) and from making sure that all relevant information is made available to these different dimensions (through the propositional language, which is shared by all logics, via simplification rules). Thus, there is no need for new inference rules: separation provides an elegant way to deal with the combined logic.

Although elegance and simplicity are desirable features for any proof method, this cannot be achieved in a straightforward way when dealing with interactions. In this case, by definition, different dimensions are not separated. We have chosen to adopt the same set of inference rules of $\mathsf{KL}_{(n)}$, as the proof method for the interacting logic must still comprise all the inference rules for the underlying languages, so that we are still able to provide refutations for formulae in those languages. Having chosen that, some extra mechanism should be added to the proof method in order to deal with the interactions. For synchrony and no learning, instead of adding rather complex inference rules or trying to identify two complementary sets of clauses, we have chosen to add further information to the set of clauses.

We remark that we use the contrapositive form of the SNL axiom: $\vdash \neg \mathrm{K}_i \neg \bigcirc \varphi \Rightarrow \bigcirc \neg \mathrm{K}_i \neg \varphi$. A set of clauses satisfying its antecedent is written into the normal form as (at least) two clauses: a clause (or a set of clauses which imply) **true** $\Rightarrow \psi \vee \neg \mathrm{K}_i \neg l$ and a step clause (or a set of step clauses which imply) $l \Rightarrow \bigcirc \varphi$, where ψ is a disjunction of literals or modal literals, and φ and l are literals. That is, those clauses together imply $\neg \psi \Rightarrow \neg \mathrm{K}_i \neg \bigcirc \varphi$. Because of the SNL axiom, those clauses also imply $\neg \psi \Rightarrow \bigcirc \neg \mathrm{K}_i \neg \varphi$. This is the extra information that we make available by introducing the new clauses. Instead of looking for such a set of clauses, we introduce new clauses for every step clause. Recall that a step clause is in the general form

$$X \Rightarrow \bigcirc Y$$

where X is a conjunction and Y is a disjunction of literals. The information we wish to make explicit is

$$\neg \mathrm{K}_i \neg X \Rightarrow \bigcirc \neg \mathrm{K}_i \neg Y.$$

The general approach to generating the new clauses consists of taking the contrapositive form of a step clause and distributing the knowledge operator K_i through this clause[1]. Then, we take the contrapositive form of the resulting clause, exchange the knowledge and temporal operators, and rename the modal literals to keep the normal form. That is, if X is a conjunction, then we replace X by a new propositional symbol new_X, called $\wedge$-proposition; then, the modal literals in the temporal clause are renamed by new propositional symbols, $nkn_i(X)$ and $nkn_i(Y)$, called *SNL*$_i$ propositions. The resulting clause, with the new propositional symbols representing the modal literals, is called a *SNL*$_i$ clause. The *SNL*$_i$ clauses and the clauses defining the new propositional symbols are those added to the set of clauses.

3.2. **Generating New Clauses.** Here we give formal definitions for the new literals and clauses informally discussed in the previous section. First, we make the distinction between the already existing literals and the new ones to be added. *Basic literals* are any literals in the original formula and any new literals introduced during translation into SNF_K. We use the term *literal* alone, if there is no need to distinguish which type of literal we are referring.

We firstly rename conjunctions of basic literals, adding the corresponding definitions to the set of clauses. Subsequently, we can re-use these definitions for the renaming of conjunctions

[1]Note that we cannot distribute the knowledge operator $\neg \mathrm{K}_i \neg$ over an implication, as $\neg \mathrm{K}_i \neg (\varphi \Rightarrow \psi) \Rightarrow (\neg \mathrm{K}_i \neg \varphi \Rightarrow \neg \mathrm{K}_i \neg \psi)$ (for any formulae φ and ψ) is not a valid formula in $\mathsf{S5}_{(n)}$

involving any kind of literals. To aid this process, we define a function, *NEW*, which takes a conjunction of literals, φ, as its argument and returns the new name for this conjunction, new_φ. Those new propositions are called $\wedge$-propositions.

Definition 3. *Let $c_1 \wedge \ldots \wedge c_n$, $d_1 \wedge \ldots \wedge d_m$, $d'_1 \wedge \ldots \wedge d'_{m'}$, $n, m, m' \geq 2$, be conjunctions of basic and/or SNL_i literals in the language of $\mathsf{KL}^{sync,nl}_{(n)}$. Assume there is an order over the set of literals, such that $l_i < \neg l_i < l_j < \neg l_j$, if $i < j$, for all positive literals l_i and l_j. Let SIMP(φ) be the result of applying simplification rules to φ and of ordering the conjuncts. We define the function NEW as follows:*

- *NEW*(**false**) $=$ **false**
- *NEW*(**true**) $=$ **true**
- *NEW*$(l) = l$*, for any literal* l
- *NEW*$(c_1 \wedge \ldots \wedge c_n) = new_{SIMP(c_1 \wedge \ldots \wedge c_n)}$
- *NEW*$(c_1 \wedge \ldots \wedge c_n \wedge new_{d_1 \wedge \ldots \wedge d_m} \wedge \ldots \wedge new_{d'_1 \wedge \ldots \wedge d'_{m'}}) =$ *NEW*(*SIMP*$(c_1 \wedge \ldots \wedge c_n \wedge d_1 \wedge \ldots \wedge d_m \wedge d'_1 \wedge \ldots \wedge d'_{m'}))$

The new proposition is labelled by the simplified, ordered form of the conjunction it is renaming. Simplification is given by usual classical rules, i.e. by deleting repeated literals and/or **true** from conjunctions, and by reducing contradictions to **false** and tautologies to **true**. Although the definition is applied to all literals, we do not need to rename either a constant, a literal or conjuncts which are *SNL*$_i$-literals (as we shall see later). We also do not rename conjuncts which are $\wedge$-propositions as they are only names for conjunctions. For instance, *NEW*$(a \wedge new_{b \wedge c})$ is $new_{a \wedge b \wedge c}$, as $new_{b \wedge c}$ is the name for $b \wedge c$.

Definition 4. *For each $\wedge$-proposition, $new_{c_1 \wedge \ldots \wedge c_n}$, $n \geq 2$, we add* **true** $\Rightarrow new_{c_1 \wedge \ldots \wedge c_n} \vee \neg c_1 \vee \ldots \vee \neg c_n$ *to the set of clauses. These new clauses are called $\wedge$-clauses.*

The added $\wedge$-clauses correspond to the normal form of one direction of the double implication $new_{c_1 \wedge \ldots \wedge c_n} \Leftrightarrow c_1 \wedge \ldots \wedge c_n$, which defines the $\wedge$-propositions. As we rename conjunctions on the left-hand side of step clauses (i.e. formulae of negative polarity), we only need the equivalent (in SNF_K) to $c_1 \wedge \ldots \wedge c_n \Rightarrow new_{c_1 \wedge \ldots \wedge c_n}$.

Once the $\wedge$-propositions have been generated, we define the new names for modal literals, which are the result of distributing the knowledge operator through step clauses. Renaming is used here in order to retain the normal form. We define a set of renaming functions, *REN*$_i$, one for each agent $i \in \mathcal{A}$, each of which takes as its argument a conjunction or a disjunction of literals, say φ, returning the new name for $\neg \mathrm{K}_i \neg \varphi$, that is, $nkn_i(\varphi)$. Because conjunctions are firstly renamed and the knowledge operator can be distributed over disjunctions, these functions will only be applied to literals. The new names, $nkn_i(l)$, where l is a literal, are called *SNL*$_i$-propositions. A *SNL*$_i$-literal is a *SNL*$_i$-proposition or its negation.

Definition 5. *Let $\bigvee_j l_j$ be a disjunction of literals and $\bigwedge l_b$, $\bigwedge l_{s_i}$, $\bigwedge l_{s_j}$, and $\bigwedge l_{\bigwedge_k}$ be conjunctions of basic, SNL$_i$, SNL$_j$ ($j \neq i$), and $\wedge$-literals respectively.*

- *REN*$_i(l) = nkn_i(l)$*, if l is either a basic, SNL$_j$ ($j \neq i$) or $\wedge$-literal;*
- *REN*$_i(l) = l$*, if l is a SNL$_i$-literal;*
- *REN*$_i(\bigvee_j l_j) = \bigvee_j$ *REN*$_i(l_j)$*, for any literal l_j;*
- *REN*$_i(\bigwedge l_{s_i} \wedge \bigwedge l_{s_j} \wedge \bigwedge l_b \wedge \bigwedge l_{\bigwedge_k}) = \bigwedge l_{s_i} \wedge$ *REN*$_i$(*NEW*$(\bigwedge l_{s_j} \wedge \bigwedge l_b \wedge \bigwedge l_{\bigwedge_k})$)*, where $j \neq i$.*

The last case says that we rename the conjunctions which involve *SNL*$_j$ literals, for $j \neq i$ by the corresponding $\wedge$-proposition before renaming the modal literal, but we do not need to rename the *SNL*$_i$ literals in the conjunction (because $\vdash \neg \mathrm{K}_i \neg (\neg \mathrm{K}_i \neg \varphi \wedge \neg \mathrm{K}_i \neg \psi) \Leftrightarrow (\neg \mathrm{K}_i \neg \varphi \wedge \neg \mathrm{K}_i \neg \psi)$, for any formulae φ and ψ). For instance, *REN*$_i(a \wedge new_{b \wedge c} \wedge nkn_i(d) \wedge nkn_j(e)) = nkn_i(d) \wedge nkn_i(new_{a \wedge b \wedge c \wedge nkn_j(e)})$. We also remark that clauses are kept in their simplest form (e.g. *REN*$_i(a \wedge nkn_i(a)) = nkn_i(a)$).

Definition 6. *Let l be a basic, a SNL$_j$ ($j \neq i$) or a $\wedge$-literal l. We add SNL$_i^{\Rightarrow}(l)$: **true** $\Rightarrow \neg nkn_i(l) \vee \neg K_i \neg\, l$ and SNL$_i^{\Leftarrow}(l)$: **true** $\Rightarrow nkn_i(l) \vee K_i \neg\, l$ to the set of clauses. The new clauses are called SNL$_i$ definition clauses.*

These clauses correspond to the definitions of the *SNL$_i$* literals, i.e. the equivalence $nkn_i(l) \Leftrightarrow \neg \mathrm{K}_i \neg\, l$ for each literal l. We need both sides of the double implication, because *SNL$_i$* literals can occur with both negative and positive polarities in the set of clauses.

Definition 7. *Given a step clause $X \Rightarrow \bigcirc Y$, the corresponding SNL$_i$-clause is*

$$\mathit{REN}_i(X) \Rightarrow \bigcirc(\mathit{REN}_i(Y))$$

where X is a conjunction and Y is a disjunction of literals, and REN$_i$ is the function defined above.

Note that the *SNL$_i$*-clauses are defined for both the initial set of step clauses and those step clauses generated while performing resolution.

Note also that these definitions alone could lead to the generation of an infinite number of new literals. If we consider only one agent, it is clear that this process terminates, because once the $\wedge$-propositions have been generated, due to simplification, we can determine all the *SNL$_1$* literals that need to be generated. However, when we consider multiple agents, it is not clear where we could stop generating new literals. Suppose, for instance, that $\mathcal{A} = \{1, 2\}$ and the set of basic literals is $\{a, b\}$. In this case, we generate (among others) the $\wedge$-proposition $new_{a \wedge b}$. Then, we generate (among others) the *SNL$_1$* and *SNL$_2$* literals, $nkn_1(new_{a\wedge b})$ and $nkn_2(new_{a\wedge b})$. We might need, now, to consider these new propositions as part of possible conjunctions and generate the respective *SNL$_i$* literals, as simplification might not apply in this case. For instance, we might need to generate (among others) $new_{a \wedge nkn_1(new_{a\wedge b})}$ and the *SNL$_2$* literal $nkn_2(new_{a\wedge nkn_1(new_{a\wedge b})})$.

We can prove that the number of literals that need to be generated depends on the structure of the original formula that we are trying to refute. We define the nesting depth of a *SNL$_i$* literal, $|nkn_i(l)|_{snl}$, as being the number of times that different *REN$_i$* renaming functions have been applied to any literal: $|l|_{snl} = 0$, if l is a basic literal; $|new_{l_1 \wedge \ldots \wedge l_n}|_{snl} = max(|l_1|_{snl}, \ldots, |l_n|_{snl})$, if $new_{l_1\wedge\ldots\wedge l_n}$ is a $\wedge$-literal; and $|nkn_i(l)|_{snl} = 1 + |l|_{snl}$, otherwise. Note that we have to consider the *SNL$_j$*-literals, $j \neq i$, that are renamed by the $\wedge$-propositions (e.g. $|new_{a\wedge nkn_1(b)}|_{snl} = max(|a|_{snl}, |nkn_1(b)|_{snl}) = max(0, 1 + |b|_{snl}) = max(0, 1) = 1$). The maximum nesting depth of *SNL$_i$*-literals needed in the resolution method is at most the same as the number of alternations of distinct knowledge operators in the original formula, that is, the *alternating modal depth* of the formula. Given such an upper bound for the number of literals that need to generated, that is, by allowing only a finite number of literals, termination of the method is guaranteed.

We call SNF_{snl} the set of clauses resulting from the transformation of a formula into the SNF_K, the *SNL$_i$*, the $\wedge$, and the *SNL$_i$* definition clauses. The resolution method applied to a set of SNF_{snl} clauses is essentially the same as that described in Section 2, except that we extend the function mod_i so that $mod_i(l) = l$, if l is a *SNL$_i$*-literal.

Below, we illustrate the use of the method. The first example shows that $\bigcirc \mathrm{K}_1 \mathrm{K}_2\, \varphi \Rightarrow \mathrm{K}_1 \mathrm{K}_2 \bigcirc \varphi$ is valid in $\mathsf{KL}_{(n)}^{sync,nl}$. We start by negating the formula and then transforming it into its normal form.

$$\begin{array}{rrcl}
1. & \textbf{start} & \Rightarrow & x \\
2. & x & \Rightarrow & \bigcirc y \\
3. & \textbf{true} & \Rightarrow & \neg y \vee \mathrm{K}_1\, z \\
4. & \textbf{true} & \Rightarrow & \neg z \vee \mathrm{K}_2\, \varphi
\end{array}
\qquad
\begin{array}{rrcl}
5. & \textbf{true} & \Rightarrow & \neg x \vee \neg \mathrm{K}_1 \neg\, w \\
6. & \textbf{true} & \Rightarrow & \neg w \vee \neg \mathrm{K}_2 \neg\, t \\
7. & t & \Rightarrow & \bigcirc \neg \varphi
\end{array}$$

where clauses (1-4) correspond to $\bigcirc \mathrm{K}_1 \mathrm{K}_2\, \varphi$ and the other clauses correspond to $\neg \mathrm{K}_1 \mathrm{K}_2 \bigcirc \varphi$. We add the *SNL$_i$*-clauses

$$\begin{array}{rrcll}
8. & nkn_1(x) & \Rightarrow & \bigcirc nkn_1(y) & [2, \mathit{SNL}_1] \\
9. & nkn_2(nkn_1(x)) & \Rightarrow & \bigcirc nkn_2(nkn_1(y)) & [8, \mathit{SNL}_2]
\end{array}$$

and SNL_i definition clauses that will be used in the refutation:

$$\begin{array}{lrcl}
\mathit{SNL}_1^{\Leftarrow}(x). & \textbf{true} & \Rightarrow & nkn_1(x) \lor \mathrm{K}_1 \neg\, x \\
\mathit{SNL}_2^{\Leftarrow}(nkn_1(x)). & \textbf{true} & \Rightarrow & nkn_2(nkn_1(x)) \lor \mathrm{K}_2 \neg\, nkn_1(x) \\
\mathit{SNL}_1^{\Rightarrow}(y). & \textbf{true} & \Rightarrow & \neg nkn_1(y) \lor \neg \mathrm{K}_1 \neg\, y \\
\mathit{SNL}_2^{\Rightarrow}(nkn_1(y)). & \textbf{true} & \Rightarrow & \neg nkn_2(nkn_1(y)) \lor \neg \mathrm{K}_2 \neg\, nkn_1(y)
\end{array}$$

The refutation proceeds as follows.

$$\begin{array}{lrcll}
10. & \textbf{true} & \Rightarrow & \neg nkn_1(y) \lor \mathrm{K}_1\, z & [3, \mathit{SNL}_1^{\Rightarrow}(y), \mathrm{MRES4}] \\
11. & \textbf{true} & \Rightarrow & \neg nkn_1(y) \lor z & [10, \mathrm{MRES5}] \\
12. & \textbf{true} & \Rightarrow & \neg nkn_2(nkn_1(y)) \lor \neg \mathrm{K}_2 \neg\, z & [11, \mathit{SNL}_2^{\Rightarrow}(nkn_1(y)), \mathrm{MRES4}] \\
13. & \textbf{true} & \Rightarrow & \neg nkn_2(nkn_1(y)) \lor \mathrm{K}_2\, \varphi & [12, 4, \mathrm{MRES4}] \\
14. & \textbf{true} & \Rightarrow & \neg nkn_2(nkn_1(y)) \lor \varphi & [13, \mathrm{MRES5}] \\
15. & nkn_2(nkn_1(x)) & \Rightarrow & \bigcirc \varphi & [14, 9, \mathrm{SRES1}] \\
16. & t \land nkn_2(nkn_1(x)) & \Rightarrow & \bigcirc \textbf{false} & [15, 7, \mathrm{SRES2}] \\
17. & \textbf{true} & \Rightarrow & \neg t \lor \neg nkn_2(nkn_1(x)) & [16, \mathrm{SIMP1}] \\
18. & \textbf{true} & \Rightarrow & \neg w \lor \neg nkn_2(nkn_1(x)) & [17, 6, \mathrm{MRES4}] \\
19. & \textbf{true} & \Rightarrow & \neg w \lor \mathrm{K}_2\, \neg nkn_1(x) & [18, \mathit{SNL}_2^{\Leftarrow}(nkn_1(x)), \mathrm{MRES1}] \\
20. & \textbf{true} & \Rightarrow & \neg w \lor \neg nkn_1(x) & [19, \mathrm{MRES5}] \\
21. & \textbf{true} & \Rightarrow & \neg x \lor \neg nkn_1(x) & [20, 5, \mathrm{MRES4}] \\
22. & \textbf{true} & \Rightarrow & \neg x \lor \mathrm{K}_1 \neg\, x & [21, \mathit{SNL}_1^{\Leftarrow}(x), \mathrm{MRES1}] \\
23. & \textbf{true} & \Rightarrow & \neg x & [22, \mathrm{MRES5,\ SIMP}] \\
34. & \textbf{start} & \Rightarrow & \textbf{false} & [23, 1, \mathrm{IRES1}]
\end{array}$$

The following example is the proof that $\Box \mathrm{K}_1\, \mathrm{K}_2\, \varphi \Rightarrow \mathrm{K}_1\, \mathrm{K}_2\, \Box \varphi$ is valid in $\mathsf{KL}_{(n)}^{sync,nl}$. We start by transforming the negation of this formula into its normal form.

$$\begin{array}{lrcl@{\qquad}lrcl}
1. & \textbf{start} & \Rightarrow & x & 6. & \textbf{true} & \Rightarrow & \neg y \lor \mathrm{K}_1\, w \\
2. & \textbf{true} & \Rightarrow & \neg x \lor y & 7. & \textbf{true} & \Rightarrow & \neg w \lor \mathrm{K}_2\, \varphi \\
3. & \textbf{true} & \Rightarrow & \neg x \lor z & 8. & \textbf{true} & \Rightarrow & \neg x \lor \neg \mathrm{K}_1 \neg\, r \\
4. & z & \Rightarrow & \bigcirc y & 9. & \textbf{true} & \Rightarrow & \neg r \lor \neg \mathrm{K}_2 \neg\, s \\
5. & z & \Rightarrow & \bigcirc z & 10. & s & \Rightarrow & \Diamond \neg \varphi
\end{array}$$

Then, we add the new SNL_i clauses:

$$\begin{array}{lrcll}
11. & nkn_1(z) & \Rightarrow & \bigcirc nkn_1(y) & [4, \mathit{SNL}_1] \\
12. & nkn_1(z) & \Rightarrow & \bigcirc nkn_1(z) & [5, \mathit{SNL}_1] \\
13. & nkn_2(nkn_1(z)) & \Rightarrow & \bigcirc nkn_2(nkn_1(y)) & [11, \mathit{SNL}_2] \\
14. & nkn_2(nkn_1(z)) & \Rightarrow & \bigcirc nkn_2(nkn_1(z)) & [12, \mathit{SNL}_2]
\end{array}$$

and also the SNL_i definition clauses that will be needed in the refutation:

$$\begin{array}{lrcl}
\mathit{SNL}_1^{\Rightarrow}(y): & \textbf{true} & \Rightarrow & \neg nkn_1(y) \lor \neg \mathrm{K}_1 \neg\, y \\
\mathit{SNL}_1^{\Leftarrow}(z): & \textbf{true} & \Rightarrow & nkn_1(z) \lor \mathrm{K}_1 \neg\, z \\
\mathit{SNL}_2^{\Rightarrow}(nkn_1(y)): & \textbf{true} & \Rightarrow & \neg nkn_2(nkn_1(y)) \lor \neg \mathrm{K}_2 \neg\, nkn_1(y) \\
\mathit{SNL}_2^{\Leftarrow}(nkn_1(z)): & \textbf{true} & \Rightarrow & nkn_2(nkn_1(z)) \lor \mathrm{K}_2 \neg\, nkn_1(z)
\end{array}$$

The refutation now proceeds as follows

$$\begin{array}{lrcll}
15. & \textbf{true} & \Rightarrow & \neg nkn_1(y) \lor \mathrm{K}_1\, w & [6, \mathit{SNL}_1^{\Rightarrow}(y), \mathrm{MRES4}] \\
16. & \textbf{true} & \Rightarrow & \neg nkn_1(y) \lor w & [15, \mathrm{MRES5}] \\
17. & \textbf{true} & \Rightarrow & \neg nkn_2(nkn_1(y)) \lor \neg \mathrm{K}_2 \neg\, w & [16, \mathit{SNL}_2^{\Rightarrow}(nkn_1(y)), \mathrm{MRES4}] \\
18. & \textbf{true} & \Rightarrow & \neg nkn_2(nkn_1(y)) \lor \mathrm{K}_2\, \varphi & [17, 7, \mathrm{MRES4}]
\end{array}$$

19.	$\mathbf{true}$	$\Rightarrow$	$\neg nkn_2(nkn_1(y)) \lor \varphi$	$[18, \text{MRES5}]$
20.	$nkn_2(nkn_1(z))$	$\Rightarrow$	$\bigcirc \varphi$	$[13, 19, \text{SRES2}]$
21.	s	$\Rightarrow$	$\neg nkn_2(nkn_1(z))\, \mathcal{W}\, \neg \varphi$	$[20, 14, 10, \text{TRES}]$
22.	$\mathbf{true}$	$\Rightarrow$	$\neg s \lor \neg nkn_2(nkn_1(z)) \lor \neg \varphi$	$[21, \mathsf{SNF}]$
23.	$\mathbf{true}$	$\Rightarrow$	$\neg r \lor \neg nkn_2(nkn_1(z)) \lor \neg \mathrm{K}_2\, \varphi$	$[22, 9, \text{MRES4}]$
24.	$\mathbf{true}$	$\Rightarrow$	$\neg r \lor \neg nkn_2(nkn_1(z)) \lor \neg w$	$[23, 7, \text{MRES1}]$
25.	$\mathbf{true}$	$\Rightarrow$	$\neg r \lor \mathrm{K}_2 \neg nkn_1(z) \lor \neg w$	$[24, \mathit{SNL}_2^{\Leftarrow}(nkn_1(z)), \text{MRES1}]$
26.	$\mathbf{true}$	$\Rightarrow$	$\neg r \lor \neg nkn_1(z) \lor \neg w$	$[25, \text{MRES5}]$
27.	$\mathbf{true}$	$\Rightarrow$	$\neg x \lor \neg nkn_1(z) \lor \neg \mathrm{K}_1\, w$	$[26, 8, \text{MRES4}]$
28.	$\mathbf{true}$	$\Rightarrow$	$\neg x \lor \neg nkn_1(z) \lor \neg y$	$[27, 6, \text{MRES1}]$
29.	$\mathbf{true}$	$\Rightarrow$	$\neg x \lor \neg nkn_1(z)$	$[28, 2, \text{MRES1}]$
30.	$\mathbf{true}$	$\Rightarrow$	$\neg x \lor \mathrm{K}_1 \neg z$	$[29, \mathit{SNL}_1^{\Leftarrow}(z), \text{MRES1}]$
31.	$\mathbf{true}$	$\Rightarrow$	$\neg x$	$[30, 3, \text{MRES3}]$
32.	$\mathbf{start}$	$\Rightarrow$	$\mathbf{false}$	$[31, 1, \text{IRES1}]$

4. Correctness

In this section, we give the results for soundness, termination, and completeness of the resolution method for $\mathsf{KL}_{(n)}^{sync,nl}$. Full proofs can be found in [10].

Soundness. The soundness proof consists of showing that, given a formula φ in $\mathsf{KL}_{(n)}^{sync,nl}$, its transformation into SNF_{snl} is satisfiability preserving and that the application of the inference rules to the set of clauses in the normal form is also satisfiability preserving.

Theorem 1. *Let φ be a well-formed formula in $\mathsf{KL}_{(n)}^{sync,nl}$ and $\tau(\varphi) = \Box^* \bigwedge_i T_i$ be the transformation of φ into SNF_{snl}, where every T_i is a clause. φ is satisfiable in $\mathsf{KL}_{(n)}^{sync,nl}$ if, and only if, $\tau(\varphi)$ is satisfiable in $\mathsf{KL}_{(n)}^{sync,nl}$.*

This result is obtained from correctness of transformation into SNF_K (given in [3]) and from Lemmas 1.1, 1.2, and 1.3, which show that the addition of new literals and definition clauses are satisfiability preserving.

Lemma 1.1. *Let T be a set of clauses in SNF_{snl}. Let T' be the set of clauses augmented with the definitions of $\land$-propositions, that is, $T' = T \cup \{\mathbf{true} \Rightarrow new_{l_1 \land \ldots \land l_n} \lor \neg l_1 \lor \ldots \lor \neg l_n \mid$ for all literals l_i occurring in $T\}$. T is satisfiable in $\mathsf{KL}_{(n)}^{sync,nl}$ if, and only if, T' is satisfiable $\mathsf{KL}_{(n)}^{sync,nl}$.*

Proof of Lemma 1.1 (outline) We show that given a model M for T, we can construct a model M' for T'. For every state (t, u) in M, let the corresponding state (t', u') in M' be exactly as (t, u), except that $\pi(t', u')(new_{l_1 \land \ldots \land l_n}) = \mathit{true}$, if $\langle M, (t, u)\rangle \models l_1 \land \ldots \land l_n$ for all possible conjunctions of literals. Temporal and equivalence relations are kept as in the original model. Obviously, M' satisfies all clauses in T; also, it follows from the construction of M', that such a model satisfies all definition clauses for the $\land$-literals. For the "only if part", a model M for T can be obtained from M by ignoring the values of the $\land$-literals. $\Box$

Lemma 1.2. *Let T be a set of clauses in SNF_{snl}. Let T' be the set of clauses augmented with the definitions of SNL_i-literals, that is, $T' = T \cup \{\mathit{SNL}_i^{\Rightarrow}(l) : \mathbf{true} \Rightarrow \neg nkn_i(l) \lor \neg K_i \neg l, \mathit{SNL}_i^{\Leftarrow}(l) : \mathbf{true} \Rightarrow nkn_i(l) \lor K_i \neg l \mid$ for all literals l occurring in $T\}$. T is satisfiable $\mathsf{KL}_{(n)}^{sync,nl}$ if, and only if, T' is satisfiable $\mathsf{KL}_{(n)}^{sync,nl}$.*

Proof of Lemma 1.2 (outline) This proof can be obtained in a similar way that the proof for Lemma 1.1, but where for every state (t, u) in M, the corresponding state (t', u') in M' is exactly as (t, u), except that $\pi(t', u')(nkn_i(l)) = \mathit{true}$ if, and only if, $\langle M, (t, u)\rangle \models \neg \mathrm{K}_i \neg l$ for all agents $i \in \mathcal{A}$ and literals l. $\Box$

Lemma 1.3. *Let T be a set of clauses in SNF_{snl}. Let T' be the set of clauses augmented with the SNL_i-clauses, i.e. $T' = T \cup \{REN_i(X) \Rightarrow \bigcirc REN_i(Y) \mid \text{for all } \Box^*(X \Rightarrow \bigcirc Y) \in T\}$, where REN_i is the renaming function, X is a conjunction and Y is a disjunction of literals . T is satisfiable $\mathsf{KL}_{(n)}^{sync,nl}$ if, and only if, T' is satisfiable $\mathsf{KL}_{(n)}^{sync,nl}$.*

Proof of Lemma 1.3 (outline) If X is a conjunction, by Lemma 1.1, there is a $\wedge$-proposition, new_X, such that, if $\langle M,(t,u)\rangle \models X$, then $\langle M,(t,u)\rangle \models new_X$. Thus, without loss of generality, we assume that X is a proposition. This proof can be, then, obtained from the semantics of the universal operator, $\Box^*$, which surrounds all clauses, propositional reasoning and applications of the axioms K and *SNL*. The "only if" part is obvious: if there is a model that satisfies all clauses in T', $T \subset T'$, the model satisfies all clauses in T. $\Box$

The set of inference rules is the same as that described in [3], except that we add to the definition of the function mod_i that $mod_i(l) = l$, if l is a SNL_i-literal. We remark that redefining this function is not essential, but it saves some steps in the refutation.

Theorem 2. *Let T be a set of clauses in the SNF_{snl}. Let T' be the set of clauses augmented with any resolvents obtained by applying the inference rules IRES1-2, MRES1-5 (where $mod_i(l) = l$, if l is a SNL_i-literal), SRES1-2, SIMP1, and TRES to T. T is satisfiable in $\mathsf{KL}_{(n)}^{sync,nl}$ if, and only if, T' is satisfiable in $\mathsf{KL}_{(n)}^{sync,nl}$.*

Proof of Theorem 2 This result follows from soundness of the resolution method for $\mathsf{KL}_{(n)}$ [3], that is, that all inference rules, including MRES4 without modification, are sound, and from the observation that the same resolvent from the modified MRES4 can also be obtained from successive applications of the original MRES4 and MRES1 to the original resolvent and the SNL_i definition clauses. $\Box$

Termination. The algorithm for the method presented here is based on that for $\mathsf{KL}_{(n)}$. The difference is that SNL_i and $\wedge$-literals, together with their corresponding definition clauses, are introduced before starting the application of the resolution method. Also, SNL_i-clauses, corresponding to existing or newly generated step clauses, are introduced when performing the algorithm. It has been shown in [3] that the method for $\mathsf{KL}_{(n)}$ terminates, i.e. given a finite number of clauses only a finite number of clauses (modulo order and simplification) can be generated, so at some point either **false** is generated or no new clauses are generated. In order to transfer the termination results from $\mathsf{KL}_{(n)}$ to $\mathsf{KL}_{(n)}^{sync,nl}$, we have to show that all propositional symbols that might be needed for the refutation can be defined before the resolution rules are applied.

Firstly, it has been shown in [6] that the new propositional symbols required for translating the resolvent obtained by an application of the temporal resolution rule can be added at the beginning of the proof. No other inference rule requires the introduction of new symbols. As there is a finite number of symbols, due to simplification, only a finite number of clauses is generated. Secondly, simplification is applied when generating the $\wedge$-literals and SNL_i-literals, so the number of definition clauses for these literals is also finite. Thirdly, the number of step clauses is (at any point) finite, and so it is the number of SNL_i clauses. Finally, given the alternating modal depth of the original formula, the maximum nesting depth of SNL_i-literals can be determined, and so the number of new literals that might be needed in the refutation is finite. Given that only a finite number of new symbols and definition clauses is introduced, there is only a finite number of clauses that can be defined. Thus, as the resolution method applied to the set of clauses in the SNF_{snl} is the same as the method for $\mathsf{KL}_{(n)}$, termination follows from the results in [3].

Completeness. This proof is based on that given in [6], where a graph is built from a set of clauses. The construction of the graph is given in more detail in [10]. The proof consists in showing that an empty graph corresponds to an unsatisfiable set of clauses and that, in this case, there is a refutation by the resolution method presented here.

Let T be a set of clauses into SNF_{snl}. We construct a finite direct *graph* $G = \langle N, E\rangle$ *for* T, where N is a set of nodes and E is a set of labelled edges, as follows. A node $\eta = (V, Y)$ is a pair, where V is a maximal consistent set of basic literals, modal literals, *SNL*$_i$ literals, and $\wedge$-literals; and Y is a subset of basic literals occurring on the right hand side of a sometime clause. Intuitively, V corresponds to states and Y is a set of witnesses for eventualities, that is, it corresponds to eventualities that have not been satisfied by the predecessors of V. There are $n+1$ types of edges: one for the temporal dimension plus one for each agent in $\mathcal{A} = \{1, \dots, n\}$. For every consistent set V, we construct nodes $\eta = (V, Y)$, where Y is any of the possible subsets of literals occurring on the right-hand side of sometime clauses in the set of clauses. We delete any nodes that do not immediately satisfy the literal and modal clauses, including the definition clauses, in T.

Given a non-empty set of nodes, we construct the set of labelled edges by firstly building the equivalence classes for each agent $i \in \mathcal{A}$. There is an i-edge between two nodes $\eta = (V, Y)$ and $\eta' = (V', Y')$, if, and only if, V and V' contain the same set of modal literals for that agent. We say that a node η' is i-reachable from η, if there is an i-edge between η and η'. We say that a node η' is $\{i_0, \dots, i_m\}$-reachable from η, if there is a sequence of nodes $\eta_0, \dots, \eta_{m+1}$, such that $\eta_0 = \eta$, $\eta_{m+1} = \eta'$, and there is a i_j-edge between every two nodes η_j and η_{j+1}, for $0 \leq j \leq m$. We define $[\eta]_i$ as the set of nodes that are i-reachable from η. Obviously, every $[\eta]_i$ defines an equivalence relation over the set of nodes.

Then, we construct the temporal edges. We start with a full (temporal) graph, i.e. there is a t-edge linking every two nodes in the graph (because **true** $\Rightarrow \bigcirc$**true**). We say that a node η' is t-reachable from η, if there is a sequence of nodes $\eta_0, \dots, \eta_m$, such that $\eta_0 = \eta$, $\eta_m = \eta'$, and there is a t-edge between every two nodes η_j and η_{j+1}, for $0 \leq j < m$. We say that η is a predecessor of η', if η' is t-reachable from η.

For every step clause $(\bigwedge l \Rightarrow \bigcirc \bigvee l') \in T$, we delete a t-edge between $\eta = (V, Y)$ and $\eta' = (V', Y')$, if $\eta \models \bigwedge l$ and $\eta' \not\models \bigvee l'$. For each sometime clause, $\varphi \Rightarrow \Diamond l$, in the set of clauses, we also delete a node $\eta = (V, Y)$, if $V \models \varphi$ and $l \notin Y$; and a t-edge from $\eta = (V, Y)$ to $\eta' = (V', Y')$, if (a) $l \in Y$, $l \notin Y'$, and $V \not\models l$; or (b) $l \in Y'$, $V \models l$, and $V' \not\models \varphi$. This ensures that (a) eventualities not satisfied by a predecessor are not "forgotten" and (b) there will be no edge from a node that satisfies an eventuality to another node that is "waiting" for the same eventuality to be satisfied, unless this successor satisfies the left-hand side of a sometime clause that also says that this eventuality should hold at some moment in the future. In other words, we are only waiting for eventualities to occur that originally came from satisfying the left-hand side of a sometime clause.

A node $\eta = (V, Y)$ is an *initial node* if, and only if, V satisfies all initial clauses in T and, for each sometime clause $\varphi \Rightarrow \Diamond l$, such that $V \models \varphi$, if, and only if, $l \in Y$. We say that a node η' is *reachable* from a node η, if, and only if, there is a sequence of nodes $\eta_0, \dots, \eta_m$, such that, $\eta_0 = \eta$, $\eta_m = \eta'$, and either η_{j+1} is t-reachable or i-reachable from η_j, for $0 \leq j < m$ and $i \in \mathcal{A}$.

We further delete any nodes that are not predecessors of any node which is reachable from an initial node. This reduces the graph to (possibly disjoint) connected components which include at least one initial node. The resulting graph is called a *behaviour graph for T*. Given a behaviour graph for T, we recursively delete any nodes (V, Y) (and edges to it) that have no temporal successors (no infinite temporal line can be constructed from the node, so it is not part of any model); satisfy the left-hand side of a sometime clause, but there is no node satisfying the eventuality that is t-reachable from this node; and/or satisfy a formula as $\neg \mathrm{K}_i \neg p$ but there is no modal i-edge to a node which satisfies p. The resulting graph is called *reduced behaviour graph*. Deletions of either nodes or edges are repeated non-deterministically until the graph is empty or no other deletion can be done. This procedure corresponds, respectively, to application of temporal simplification (SIMP1), temporal resolution (TRES), and modal resolution to the set of clauses. If the graph is empty, then the set of clauses is not satisfiable in $\mathsf{KL}_{(n)}^{sync,nl}$.

If the reduced behaviour graph is not empty, we show that given two nodes η and μ in the same equivalence class, i.e. $[\eta]_i = [\mu]_i$, for agent $i \in \mathcal{A}$, such that η' is a successor of η, then μ has also a successor in $[\eta']_i$.

Theorem 3. *Let G be a non-empty reduced behaviour graph for a set of clauses T in* SNF_{snl}. *Let η, μ, η', μ', be nodes in G, such that $[\eta]_i = [\mu]_i$ and there is a t-edge from η to η'. Then there is a node μ', $[\mu']_i = [\eta']_i$, such that there is a t-edge from μ to μ'.*

Proof of Theorem 3 We show, by contradiction, that if the graph is not empty and we have a temporal edge from a node η to a node η', then all nodes in $[\eta]_i$ have successors in $[\eta']_i$.

Suppose that there is a temporal edge from η to η' and $\mu \in [\eta]_i$ has no successors in $[\eta']_i$. Thus, we can identify a set of step clauses that together imply $\psi \Rightarrow \bigcirc\chi$, such that μ satisfies ψ and, for all $\mu' \in [\eta']_i$, μ' satisfies $\neg\chi$. If we could not identify this set of clauses, the temporal edges would not have been removed.

If all $\mu' \in [\eta']_i$ satisfies $\neg\chi$, then, by the semantics of the knowledge operator, $\mu' \models \mathrm{K}_i\neg\chi$. The addition of the *SNL$_i$*-clause corresponding to $\psi \Rightarrow \bigcirc\chi$, i.e. $REN_i(\psi) \Rightarrow \bigcirc REN_i(\chi)$, ensures that every node must satisfy $\neg\mathrm{K}_i\neg\psi \Rightarrow \bigcirc\neg\mathrm{K}_i\neg\chi$. We have that μ satisfies ψ and, by the semantics of the knowledge operator, it also satisfies $\neg\mathrm{K}_i\neg\psi$ (or the corresponding $\wedge$-proposition, new_ψ, and the formula $\neg\mathrm{K}_i\neg new_\psi$, in the case where ψ is a conjunction). In fact, every node in $[\eta]_i$ satisfies $\neg\mathrm{K}_i\neg\psi$. If all nodes in $[\eta']_i$ satisfy $\mathrm{K}_i\neg\chi$, because of the *SNL$_i$* clause, then there must be no temporal edge from η to η'. This contradicts with our initial assumptions, so there must be a temporal edge between μ and μ' or none of the nodes in $[\eta]_i$ have successors in $[\eta']_i$.□

Theorem 3 shows that every two nodes in the same equivalence class, say $[\eta]_i$, have successors in the same equivalence class, say $[\eta']_i$. However, there might be nodes in $[\eta']_i$ that have no predecessors in $[\eta]_i$. In order to be able to construct a model, we also have to show that every node has a predecessor.

Theorem 4. *Let G be a non-empty reduced behaviour graph for a set of clauses T in* SNF_{snl}. *Let η be a node in G. Then, there is a node η' such that η' is a predecessor of η.*

Proof of Theorem 4 (outline) If η has no predecessor, we introduce a node $\eta' = (V', E')$ (and a temporal edge from η' to η) such that $V' \models \neg\mathbf{start}$ and $V' \models \neg p$ for all propositional symbols occurring in the set of SNF_{snl} clauses. Note that such a node satisfies trivially all clauses in the set of SNF_{snl} clauses, either by falsifying the left-hand side of a (initial, step, or sometime clause) or by satisfying the right-hand side of a modal or literal clause (as, from translation, the right-hand side of such a clause has at least one negative literal occurring as a disjunct).□

Hence, if the reduced behaviour graph is not empty, we can inductively construct a model in which, if two nodes, (s, m) and (t, m), are in the same equivalence class, i.e. $(s, m) \sim_i (t, m)$, then their successors are also in the same equivalence class, i.e. $(s, m+1) \sim_i (t, m+1)$, which suffices to prove that the model is in the class for synchronous systems with no learning [7]. In order to build the first timeline for the agent i, we choose an initial node η_0 from the graph, which corresponds to the point $(t_0, 0)$. Let the nodes of $[\eta_0]_i$ be the initial points of other timelines for agent i. Note that only the first point at the first timeline needs to be an initial node in the graph. Then, we choose η_1 from the immediate successors (i.e. t-reachable by one t-edge) of η_0. We build the next point in each timeline, by choosing the immediate successors of each node in $[\eta_0]_i$ from the nodes in $[\eta_1]_i$. From Theorem 3, this is always possible. From Theorem 4, for points at time greater than zero, we can construct the predecessors of a node back to initial point of every timeline.

We note that the introduction of the *SNL$_i$*-clauses not only delete edges which are not part of a model for $\mathsf{KL}_{(n)}^{sync,nl}$, but also contribute to the temporal resolution procedure. Temporal resolution between a sometime clause, say $l \Rightarrow \Diamond p$, and a set of clauses that together imply $\bigcirc\Box\neg p$, corresponds to removing from the graph subcomponents which satisfy l, but never satisfy the eventuality p. As we are considering the combined logics of knowledge and time, the sometime clauses may be preceded by a chain of knowledge operators. For instance, the set of clauses (taken from the second example on Page 9):

$$\begin{array}{rrcl} 8. & \mathbf{true} & \Rightarrow & \neg x \vee \neg\mathrm{K}_1\neg r \\ 9. & \mathbf{true} & \Rightarrow & \neg r \vee \neg\mathrm{K}_2\neg s \\ 10. & s & \Rightarrow & \Diamond\neg\varphi \end{array}$$

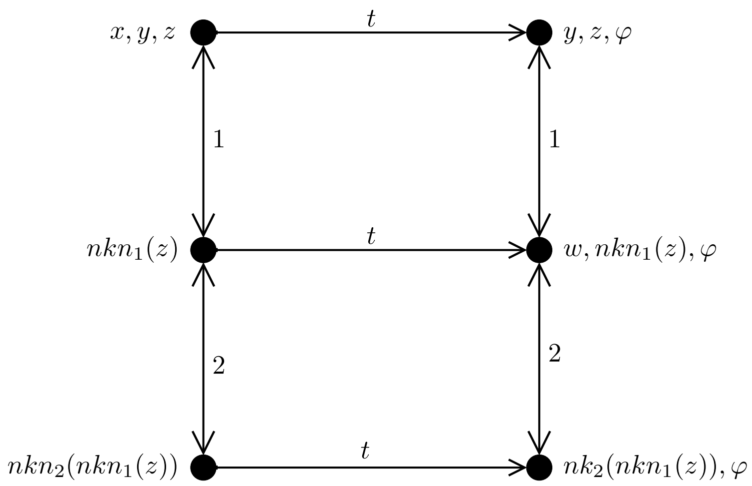

FIGURE 2. Some temporal loops related to the second example.

shows that the left-hand side of the sometime clause is preceded by the knowledge operators $\neg \mathrm{K}_1 \neg$ and $\neg \mathrm{K}_2 \neg$. In systems with synchrony and no learning, $\neg \mathrm{K}_1\, \mathrm{K}_2 \neg \Diamond \neg \varphi$ implies $\Diamond \neg \mathrm{K}_1\, \mathrm{K}_2 \neg\, \varphi$, which should be resolved with a set of clauses that together imply $\bigcirc \square \mathrm{K}_1\, \mathrm{K}_2\, \varphi$. Note that because of the axiom T, the same loop implies $\bigcirc \square \varphi$. Although this loop could be resolved with the sometime clause, the resolvents generated by temporal resolution would not contribute to removing the chain of knowledge operators preceding the eventuality. The sometime clause occurs in a timeline that is $\{1, 2\}$-reachable from the initial timeline. Therefore, we are interested in a loop that is also $\{1, 2\}$-reachable from the initial timeline. In Figure 2, we show this loop, where the states are represented by circles; each state is labelled by the propositional symbols which hold at that state; and the edges are labelled either by t (indicating the temporal successor) or by the agent index (indicating that those states are in the same equivalence class). We note that we could have introduced *SNL*$_i$-literals in sometime clauses in the same way that those literals where introduced in step clauses, but this is not necessary for completeness. The next theorem ensures that the appropriate loop is found, if a chain of knowledge operators precede the sometime clause.

Theorem 5. *Let C be the set of SNF_{snl} clauses that together imply $\neg K_{i_1}\, K_{i_2}\, \ldots K_{i_m} \neg \Diamond \neg \varphi$ where φ is a basic literal and every two consecutive knowledge operators differ in their index. If there is a set of clauses that imply $\bigcirc \square K_{i_1}\, K_{i_2}\, \ldots K_{i_m}\, \varphi$, then there is a set of nested SNL$_i$ clauses C' that imply $\bigcirc \square \varphi$.*

Specifically, if χ is the conjunction of literals on the left-hand side of the step clauses which satisfy the loop conditions, then we can prove that there is a loop given by $nkn_{i_{m\ldots1}}(\chi) \Rightarrow \bigcirc\varphi$ (where $nkn_{i_{m\ldots1}}(l)$ is an abbreviation for successively applying the renaming functions $REN_1, \ldots, REN_m$ to a literal l). The temporal resolvents, the set of clauses that represent the chain of knowledge operators preceding the sometime clause, and the (nested) *SNL*$_i$ definitions of χ can be successively resolved by applying the modal inference rules. Intuitively, the nesting of *SNL*$_i$ literals transforms the interaction between knowledge and time into two separated problems: a temporal and a modal problem. Firstly, by introducing the *SNL*$_i$ propositions, the interaction is represented as a (propositional) temporal problem. Once the temporal problem is solved, modal resolution takes place, "translating" back those *SNL*$_i$ propositions into their corresponding modal literals and performing resolution in the modal portion of the language.

In the proof, we assume that subsequent knowledge operators in the chain preceding a temporal formula had different indices. This corresponds to the alternating modal depth of the knowledge operators in the original formula and gives an upper bound for the maximum nesting of *SNL*$_i$ literals. Note that this assumption can be made without loss of generality, as chains of modal clauses with same index can be successively resolved together, resulting in one modal clause of that index.

5. CONCLUSIONS

We have shown how to extend the method presented in [3] to deal with synchronous systems with no learning. The addition of new information to the set of clauses, together with the use of renaming, provides an intuitive way of dealing with this particular interaction. Although the introduction of new clauses is costly, we avoid the introduction of new inference rules and, potentially, expensive searches over the set of clauses. As there is no change in the set of inference rules, implementation is relatively easily obtained by adapting and re-using existing theorem-provers. Correctness for the multi-agent case has been established. Results for the multi-agent case of the method for synchrony and perfect recall given in [2] can be established in a similar way and it is ongoing work.

Synchronous systems with either perfect recall or no learning have complete axiomatisations given by axioms with simple structures. We have been investigating how the ideas behind our proof method could be applied to logics that include axioms of the (simple) form $\mathrm{K}_i \bigcirc^p \varphi \Rightarrow \bigcirc^q \mathrm{K}_i\, \varphi$ or $\bigcirc^p \mathrm{K}_i\, \varphi \Rightarrow \mathrm{K}_i \bigcirc^q \varphi$ where p and q are integers ($p, q > 0$), and the iterated next operator is defined as $\bigcirc^0 \varphi = \varphi$, and $\bigcirc^p \varphi = \bigcirc^{p-1} \bigcirc \varphi$, for $p > 0$, where φ is a formula. Although the generation of new clauses would not be straightforward, we have also been investigating other interactions expressed by finite axiomatisations which include more complex axioms, as, for instance, the no learning axiom, $\vdash (\mathrm{K}_i\, \varphi_1\, \mathcal{U}\, \mathrm{K}_i\, \varphi_2) \Rightarrow \mathrm{K}_i\, (\mathrm{K}_i\, \varphi_1\, \mathcal{U}\, \mathrm{K}_i\, \varphi_2)$, when synchrony is not required.

REFERENCES

[1] C. Dixon. Temporal Resolution using a Breadth-First Search Algorithm. *Annals of Mathematics and Artificial Intelligence*, 22:87–115, 1998.
[2] C. Dixon and M. Fisher. Clausal Resolution for Logics of Time and Knowledge with Synchrony and Perfect Recall. In *Proceedings of ICTL 2000*, Leipzig, Germany, 2000.
[3] C. Dixon and M. Fisher. Resolution-Based Proof for Multi-Modal Temporal Logics of Knowledge. In S. Goodwin and A. Trudel, editors, *Proceedings of the Seventh International Workshop on Temporal Representation and reasoning (TIME'00)*, pages 69–78, Cape Breton, Nova Scotia, Canada, July 2000. IEEE Computer Society Press.
[4] C. Dixon, M. Fisher, and M. Wooldridge. Resolution for Temporal Logics of Knowledge. *Journal of Logic and Computation*, 8(3):345–372, 1998.
[5] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge*. MIT Press, 1995.
[6] M. Fisher, C. Dixon, and M. Peim. Clausal Temporal Resolution. *ACM Transactions on Computational Logic*, 2(1), January 2001.
[7] J. Y. Halpern, R. van der Meyden, and M. Y. Vardi. Complete Axiomatizations for Reasoning About Knowledge and Time. *SIAM Journal on Computing*, 33(3):674–703, 2004.
[8] B. Heinemann. Linear Tense Logics of Increasing sets. *Journal of Logic and Computation*, 12(4):583–606, 2002.
[9] R. E. Ladner and J. H. Reif. The logic of distributed protocols (preliminary report). In Joseph Y. Halpern, editor, *Theoretical Aspects of Reasoning about Knowledge: Proceedings of the First Conference*, pages 207–222, Los Altos, California, 1986. Morgan Kaufmann Publishers, Inc.
[10] C. Nalon. *Resolution for Synchrony and No Learning*. PhD thesis, University of Liverpool, March 2004.
[11] A. S. Rao and M. P. Georgeff. Modeling Rational Agents within a BDI-Architecture. In R. Fikes and E. Sandewall, editors, *Proceedings of Knowledge Representation and Reasoning (KR&R-91)*, pages 473–484. Morgan-Kaufmann, April 1991.
[12] B. van Linder, W. van der Hoek, and J. J. Ch. Meyer. Formalising Motivational Attitudes of Agents: On Preferences, Goals and Commitmentes. In M. Wooldridge, J. P.Püller, and M. Tambe, editors, *Intelligent Agents II*, volume 1037 of *Lecture Notes in Artificial Intelligence*, pages 17–32. Springer-Verlag, 1996.

ACKNOWLEDGMENTS

This work was partially supported by CAPES (grant BEX 1158-99-6) and the EPSRC research grants GR/R45376 and GR/S63182.

DEPARTMENT OF COMPUTER SCIENCE, UNIVERSITY OF LIVERPOOL, LIVERPOOL, L69 7ZF, UK.
E-mail address: {C.Nalon, C.Dixon, M.Fisher}@csc.liv.ac.uk