Tractable Temporal Reasoning *

Clare Dixon, Michael Fisher and Boris Konev
Department of Computer Science, University of Liverpool,
Liverpool L69 3BX, UK
{clare,michael,konev}@csc.liv.ac.uk

Abstract

Temporal reasoning is widely used within both Computer Science and A.I. However, the underlying complexity of temporal proof in discrete temporal logics has led to the use of simplified formalisms and techniques, such as temporal interval algebras or model checking. In this paper we show that tractable sub-classes of propositional linear temporal logic can be developed, based on the use of XOR fragments of the logic. We not only show that such fragments can be decided, tractably, via clausal temporal resolution, but also show the benefits of combining multiple XOR fragments. For such combinations we establish completeness and complexity (of the resolution method), and also describe how such a temporal language might be used in application areas, for example the verification of multi-agent systems. This new approach to temporal reasoning provides a framework in which tractable temporal logics can be engineered by intelligently combining appropriate XOR fragments.

1 Introduction

Temporal logics have been used to describe a wide variety of systems, from both Computer Science and Artificial Intelligence. The basic idea of proof, within propositional, discrete temporal logics, is also both intuitive and appealing. However the complexity of satisfiability for such logics is high. For example, the complexity of satisfiability for propositional linear time temporal logic (PTL) is PSPACE-complete [Sistla and Clarke, 1985]. Consequently, model checking [Clarke *et al.*, 1999] has received much attention as it also allows users to check that a temporal property holds for some underlying model of the system.

Often temporal problems involve an underlying structure, such as an automaton, where a key property is that the automaton can be in exactly one state at each moment. Such problems frequently involve several process or agents, each with underlying automaton-like structures, and we are interested in properties relating to how the agents progress under particular models of concurrency such as synchrony, asynchrony etc., or particular coordination or cooperation actions.

In this paper we consider a new fragment of PTL that incorporates the use of XOR operators, denoted $(q_1 \oplus q_2 \oplus \ldots \oplus q_n)$ meaning that exactly one q_i holds for $1 \leq i \leq n$. Since the complexity of unsatisfiability for XOR clauses in classical propositional logic is low [Schaefer, 1978], there is the potential to carry much of this over to the temporal case.

Thus, in this paper we provide several results. First, we introduce the PTL fragment to be considered, called TLX, and show a complete clausal resolution system for this. The fragment allows us to split the underlying set of propositions into distinct subsets such that each subset (except one) represents a set of propositions where exactly one proposition can hold (termed *XOR sets*); the remaining set has no such constraints. Then we show that deciding unsatisfiability of specifications in such a logic is, indeed, tractable.

2 XOR Temporal Logic

The logic we consider is called "TLX", and its syntax and semantics essentially follow that of PTL [Gabbay *et al.*, 1980], with models (isomorphic to the Natural Numbers, $\mathbb{N}$) of the form: $\sigma = t_0, t_1, t_2, t_3, \ldots$ where each state, t_i, is a set of proposition symbols, representing those propositions which are satisfied in the i^{th} moment in time. The notation $(\sigma, i) \models A$ denotes the truth (or otherwise) of formula A in the model σ at state index $i \in \mathbb{N}$. This leads to semantic rules:

$(\sigma, i) \models \bigcirc A$ iff $(\sigma, i+1) \models A$
$(\sigma, i) \models \Diamond A$ iff $\exists k \in \mathbb{N}.\ (k \geqslant i)$ and $(\sigma, k) \models A$
$(\sigma, i) \models \Box A$ iff $\forall j \in \mathbb{N}.$ if $(j \geqslant i)$ then $(\sigma, j) \models A$

For any formula A, model σ, and state index $i \in \mathbb{N}$, then either $(\sigma, i) \models A$ holds or $(\sigma, i) \models A$ does not hold, denoted by $(\sigma, i) \not\models A$. If there is some σ such that $(\sigma, 0) \models A$, then A is said to be *satisfiable*. If $(\sigma, 0) \models A$ for all models, σ, then A is said to be *valid* and is written $\models A$.

The main novelty in TLX is that it is parameterised by XOR-sets $\mathcal{P}_1$, $\mathcal{P}_2$,..., and the formulae of TLX($\mathcal{P}_1, \mathcal{P}_2, \ldots$) are constructed under the restrictions that *exactly* one proposition from every set $\mathcal{P}_i$ is true in every state. For example, if we consider just one set of propositions $\mathcal{P}$, we have

$$\Box(p_1 \oplus p_2 \oplus \ldots \oplus p_n) \text{ for all } p_i \in \mathcal{P}.$$

Furthermore, we assume that there exists a set of propositions in addition to those defined by the parameters, and that

*The work of the first and last authors was partially supported by EPRSC grant number GR/S63182/01 "Dynamic Ontologies: a Framework for Service Descriptions".

these propositions are unconstrained as normal. Thus, TLX() is essentially a standard propositional, linear temporal logic, while TLX($\mathcal{P}$,$\mathcal{Q}$,$\mathcal{R}$) is a temporal logic containing at *least* the propositions $\mathcal{P} \cup \mathcal{Q} \cup \mathcal{R}$, where $\mathcal{P} = \{p_1, p_2, \ldots, p_l\}$, $\mathcal{Q} = \{q_1, q_2, \ldots, q_m\}$, and $\mathcal{R} = \{r_1, r_2, \ldots, r_n\}$ where $\mathcal{P}$, $\mathcal{Q}$ and $\mathcal{R}$ are disjoint, but also satisfying

$$\Box[(p_1 \oplus p_2 \oplus \ldots \oplus p_l) \wedge (q_1 \oplus q_2 \oplus \ldots \oplus q_m) \wedge (r_1 \oplus r_2 \oplus \ldots \oplus r_n)]$$

2.1 Normal Form

Assume we have n sets of XOR propositions $P_1 = \{p_{11}, \ldots p_{1N_1}\}$, ..., $P_n = \{p_{n1}, \ldots p_{nN_n}\}$ and a set of additional propositions $A = \{a_1, \ldots a_{N_a}\}$. In the following:

- $\hat{P}^-_{ij}$ denotes a conjunction of negated XOR propositions from the set P_i;
- $\check{P}^+_{ij}$ denotes a disjunction of (positive) XOR propositions from the set P_i;
- $\hat{A}_i$ denotes a conjunction of non-XOR literals;
- $\check{A}_i$ denotes a disjunction of non-XOR literals.

A normal form for TLX is of the form $\Box \bigwedge_i C_i$ where each C_i is an *initial*, *step* or *sometime* clause (respectively) as follows:

$$\begin{array}{rl} \textbf{start} \Rightarrow & \check{P}^+_{1i} \vee \ldots \vee \check{P}^+_{ni} \vee \check{A}_i \\ \hat{P}^-_{1j} \wedge \ldots \hat{P}^-_{nj} \wedge \hat{A}_j \Rightarrow & \bigcirc(\check{P}^+_{1j} \vee \ldots \vee \check{P}^+_{nj} \vee \check{A}_j) \\ \textbf{true} \Rightarrow & \Diamond(\check{P}^+_{1k} \vee \ldots \vee \check{P}^+_{nk} \vee \check{A}_k). \end{array}$$

Note that due to the semantics of the XOR clauses, if $i \neq k$

$$p_{ji} \wedge p_{jk} \equiv \textbf{false} \qquad \neg p_{ji} \vee \neg p_{jk} \equiv \textbf{true}$$

and $$\bigwedge_{i=1}^{N_j} \neg p_{ji} \equiv \textbf{false} \qquad \bigvee_{i=1}^{N_j} p_{ji} \equiv \textbf{true}.$$

Also $$p_{ji} \equiv \bigwedge_{p_{jk} \in P_j, k \neq i} \neg p_{jk} \qquad \neg p_{ji} \equiv \bigvee_{p_{jk} \in P_j, k \neq i} p_{jk}$$

allow us to maintain positive XOR propositions on the right hand sides of clauses and negated XOR propositions on the left hand side of clauses.

2.2 Resolution Rules

We decide the validity of formulae in TLX using a form of clausal temporal resolution [Fisher *et al.*, 2001]. The resolution rules are split into three types: *initial resolution*, *step resolution* and *temporal resolution*. These are presented in Fig. 1. Initial resolution resolves constraints holding in the initial moment in time. Step resolution involves resolving two step clauses or deriving additional constraints when a contradiction in the next moment is derived. Temporal resolution resolves a sometime clause with a constraint that ensures that the right hand side of this clause cannot occur.

In the conclusion of these resolution rules $com(\check{P}^+_{ij}, \check{P}^+_{ik})$ denotes the disjunction of the propositions in *both* $\check{P}^+_{ij}$ and $\check{P}^+_{ik}$ or **false** if there are no propositions common to both. For example, $com(p_1 \vee p_2, p_2 \vee p_3) = p_2$.

Observe that IRES$_A$ and SRES$_A$ apply classical resolution to the right hand side of the parent clauses whereas IRES$_{P_k}$ and SRES$_{P_k}$ involve XOR resolution. Note we can only apply IRES$_A$ and SRES$_A$ between clauses with complementary (non-XOR) literals on the right hand side. We can also apply the IRES$_{P_k}$ and SRES$_{P_k}$ rules to these clauses but the disjunct $\check{A}_1 \vee \check{A}_2$ on the right hand side of the conclusion will be equivalent to **true**.

3 Soundness and Completeness

Similarly to [Fisher *et al.*, 2001; Degtyarev *et al.*, 2006], one can show that whenever the parent clauses are satisfiable then so is the resolvent. Since all the rules of *initial*, and *step* resolution follow the same pattern, we first prove the classical propositional counterpart of the completeness theorem, and then use it to prove the completeness of temporal resolution. Consider the following *classical* set of resolution rules consisting of the rule RES_A:

$$\frac{(\check{P}^+_{11} \vee \ldots \check{P}^+_{n1} \vee \check{A}_1 \vee a); \quad (\check{P}^+_{12} \vee \ldots \check{P}^+_{n2} \vee \check{A}_2 \vee \neg a)}{(\check{P}^+_{11} \vee \ldots \check{P}^+_{n1} \vee \check{P}^+_{12} \vee \ldots \check{P}^+_{n2} \vee \check{A}_1 \vee \check{A}_2)}$$

and, for every $k \in \{1, \ldots, n\}$, the rule RES_{P_k}:

$$\frac{\begin{array}{c}(\check{P}^+_{11} \vee \ldots \check{P}^+_{k1} \vee \ldots \check{P}^+_{n1} \vee A_1) \\ (\check{P}^+_{12} \vee \ldots \check{P}^+_{k2} \vee \ldots \check{P}^+_{n2} \vee A_2)\end{array}}{(\check{P}^+_{11} \vee \check{P}^+_{12} \vee \ldots com(\check{P}^+_{k1}, \check{P}^+_{k2}) \vee \ldots \vee \check{P}^+_{n1} \vee \check{P}^+_{n2} \vee \check{A}_1 \vee \check{A}_2)}$$

Lemma 1 *If a set of classical propositional clauses is unsatisfiable than its unsatisfiability can be established by the rules RES_A and RES_{P_k} in $O(N_1 \times N_2 \times \cdots \times N_n \times 2^{N_a})$ time.*

Proof: First we show that if an unsatisfiable set of clauses $\mathcal{C}$ does not contain non-XOR literals, then its unsatisfiability can be established by rules RES_{P_k}. Note that any such set of clauses $\mathcal{C}$ is unsatisfiable if, and only if, for every l, $0 < l \leq n$, and every set of propositions $p_1, p_2, \ldots, p_l$, where $p_i \in P_i$, the set $\mathcal{C}_{p_1,\ldots,p_l}$ of clauses from $\mathcal{C}$, which contain *none of* p_1,..., p_l, is nonempty. Indeed, otherwise every clause from $\mathcal{C}$ contains at least one of the propositions p_1,... p_l, so making $p_1, \ldots, p_l$ true satisfies $\mathcal{C}$.

Assume all clauses from $\mathcal{C}$ consist of propositions from P_1, ..., P_k only (originally, $k = n$) and show that with the rule RES_{P_k} one can obtain an unsatisfiable set of clauses $\mathcal{C}'$ in which all clauses consist of propositions from P_1,..., P_{k-1} only.

Take arbitrary propositions $p_1 \in P_1$, $p_2 \in P_2$, ... $p_{k-1} \in P_{k-1}$ and take arbitrary clauses $C_1 \in \mathcal{C}_{p1,p2,\ldots,p_{k-1},p_{k,1}}$, $C_2 \in \mathcal{C}_{p1,p2,\ldots,p_{k-1},p_{k,2}}$,..., $C_{N_k} \in \mathcal{C}_{p1,p2,\ldots,p_{k-1},p_{k,N_k}}$. Applying rule RES_{P_k} to C_1,..., C_{N_k} one can obtain a clause C' consisting of propositions from P_1,..., P_{k-1} only such that C' does not contain any of $p_1, \ldots, p_{k-1}$. The set $\mathcal{C}'$ is formed from such clauses C' for all possible combinations of $p_1 \in P_1$, $p_2 \in P_2$, ... $p_{k-1} \in P_{k-1}$. Clearly, for every l, $0 < l \leq n$, and every set of propositions $p_1, p_2, \ldots, p_l$, where $p_i \in P_i$, the set $\mathcal{C}'_{p_1,\ldots,p_l}$ is nonempty, hence, $\mathcal{C}'$ is unsatisfiable. Applying this reasoning at most n times, one can obtain an empty clause.

Consider now a set of clauses $\mathcal{C}$, which may contain non-XOR literals. For arbitrary $p_1 \in P_1$,... $p_n \in P_n$ consider $\mathcal{C}_{p_1,\ldots,p_n}$. Similarly to the previous case, every such

Initial Resolution:

$$\text{IRES}_A\quad \frac{\begin{array}{rcl}\textbf{start} & \Rightarrow & (\overset{\vee}{P}{}^{+}_{11} \vee \ldots \overset{\vee}{P}{}^{+}_{n1} \vee \overset{\vee}{A}_1 \vee a)\\ \textbf{start} & \Rightarrow & (\overset{\vee}{P}{}^{+}_{12} \vee \ldots \overset{\vee}{P}{}^{+}_{n2} \vee \overset{\vee}{A}_2 \vee \neg a)\end{array}}{\textbf{start} \Rightarrow (\overset{\vee}{P}{}^{+}_{11} \vee \ldots \overset{\vee}{P}{}^{+}_{n1} \vee \overset{\vee}{P}{}^{+}_{12} \vee \ldots \overset{\vee}{P}{}^{+}_{n2} \vee \overset{\vee}{A}_1 \vee \overset{\vee}{A}_2)}$$

For every $k \in \{1, \ldots, n\}$ we have the rule.

$$\text{IRES}_{P_k}\quad \frac{\begin{array}{rcl}\textbf{start} & \Rightarrow & (\overset{\vee}{P}{}^{+}_{11} \vee \ldots \vee \overset{\vee}{P}{}^{+}_{k1} \vee \ldots \overset{\vee}{P}{}^{+}_{n1} \vee \overset{\vee}{A}_1)\\ \textbf{start} & \Rightarrow & (\overset{\vee}{P}{}^{+}_{12} \vee \ldots \vee \overset{\vee}{P}{}^{+}_{k2} \vee \ldots \overset{\vee}{P}{}^{+}_{n2} \vee \overset{\vee}{A}_2)\end{array}}{\textbf{start} \Rightarrow (\overset{\vee}{P}{}^{+}_{11} \vee \overset{\vee}{P}{}^{+}_{12} \vee \ldots \vee com(\overset{\vee}{P}{}^{+}_{k1}, \overset{\vee}{P}{}^{+}_{k2}) \vee \ldots \vee \overset{\vee}{P}{}^{+}_{n1} \vee \overset{\vee}{P}{}^{+}_{n2} \vee \overset{\vee}{A}_1 \vee \overset{\vee}{A}_2)}$$

Step Resolution:

$$\text{SRES}_A\quad \frac{\begin{array}{rcl}\hat{A}_1 \wedge \hat{P}^{-}_{11} \wedge \ldots \hat{P}^{-}_{n1} & \Rightarrow & \bigcirc(\overset{\vee}{P}{}^{+}_{11} \vee \ldots \overset{\vee}{P}{}^{+}_{n1} \vee \overset{\vee}{A}_1 \vee a)\\ \hat{A}_2 \wedge \hat{P}^{-}_{12} \wedge \ldots \hat{P}^{-}_{n2} & \Rightarrow & \bigcirc(\overset{\vee}{P}{}^{+}_{12} \vee \ldots \overset{\vee}{P}{}^{+}_{n2} \vee \overset{\vee}{A}_2 \vee \neg a)\end{array}}{\hat{A}_1 \wedge \hat{A}_2 \wedge \hat{P}^{-}_{11} \wedge \ldots \hat{P}^{-}_{n1} \wedge \hat{P}^{-}_{12} \wedge \ldots \hat{P}^{-}_{n2} \Rightarrow \bigcirc(\overset{\vee}{P}{}^{+}_{11} \vee \ldots \overset{\vee}{P}{}^{+}_{n1} \vee \overset{\vee}{P}{}^{+}_{12} \vee \ldots \overset{\vee}{P}{}^{+}_{n2} \vee \overset{\vee}{A}_1 \vee \overset{\vee}{A}_2)}$$

For every $k \in \{1, \ldots, n\}$ we have the rule

$$\text{SRES}_{P_k}\quad \frac{\begin{array}{rcl}\hat{A}_1 \wedge \hat{P}^{-}_{11} \wedge \ldots \hat{P}^{-}_{n1} & \Rightarrow & \bigcirc(\overset{\vee}{P}{}^{+}_{11} \vee \ldots \vee \overset{\vee}{P}{}^{+}_{k1} \vee \ldots \vee \overset{\vee}{P}{}^{+}_{n1} \vee A_1)\\ \hat{A}_2 \wedge \hat{P}^{-}_{12} \wedge \ldots \hat{P}^{-}_{n2} & \Rightarrow & \bigcirc(\overset{\vee}{P}{}^{+}_{12} \vee \ldots \vee \overset{\vee}{P}{}^{+}_{k2} \vee \ldots \overset{\vee}{P}{}^{+}_{n2} \vee A_2)\end{array}}{\hat{A}_1 \wedge \hat{A}_2 \wedge \hat{P}^{-}_{11} \wedge \ldots \hat{P}^{-}_{n1} \wedge \hat{P}^{-}_{12} \wedge \ldots \hat{P}^{-}_{n2} \Rightarrow \bigcirc(\overset{\vee}{P}{}^{+}_{11} \vee \overset{\vee}{P}{}^{+}_{12} \vee \ldots \vee com(\overset{\vee}{P}{}^{+}_{k1}, \overset{\vee}{P}{}^{+}_{k2}) \vee \ldots \vee \overset{\vee}{P}{}^{+}_{n1} \vee \overset{\vee}{P}{}^{+}_{n2} \vee A_1 \vee A_2)}$$

$$\text{CONV}\quad \frac{\hat{A}_1 \wedge \hat{P}^{-}_{11} \wedge \ldots \hat{P}^{-}_{n1} \Rightarrow \bigcirc\textbf{false}}{\textbf{start} \Rightarrow (\neg \hat{A}^{-}_1 \vee \neg \hat{P}^{-}_{11} \vee \ldots \neg \hat{P}^{-}_{n1});\qquad \textbf{true} \Rightarrow \bigcirc(\neg \hat{A}^{-}_1 \vee \neg \hat{P}^{-}_{11} \vee \ldots \neg \hat{P}^{-}_{n1})}$$

Temporal Resolution:

$$\text{TRES}\quad \frac{\begin{array}{rcl}L & \Rightarrow & \Box(\neg \overset{\vee}{P}{}^{+}_{11} \wedge \ldots \wedge \neg \overset{\vee}{P}{}^{+}_{n1} \wedge \neg A_1)\\ \textbf{true} & \Rightarrow & \Diamond(\overset{\vee}{P}{}^{+}_{11} \vee \ldots \vee \overset{\vee}{P}{}^{+}_{n1} \vee A_1)\end{array}}{\textbf{start} \Rightarrow \neg L \qquad \textbf{true} \Rightarrow \bigcirc \neg L}$$

Figure 1: Resolution Rules for the XOR Fragment

$\mathcal{C}_{p_1,\ldots,p_n}$ should be nonempty. Consider the set $\widetilde{\mathcal{C}}_{p_1,\ldots,p_n}$ of clauses obtained by deleting all XOR-propositions from clauses of $\mathcal{C}_{p_1,\ldots,p_n}$. Every $\widetilde{\mathcal{C}}_{p_1,\ldots,p_n}$ must be unsatisfiable (otherwise, extending the satisfying assignment for $\widetilde{\mathcal{C}}_{p_1,\ldots,p_n}$ with $p_1, \ldots, p_n$ we satisfy all the clauses in $\mathcal{C}$). Then classical binary resolution will be able to prove unsatisfiability of $\widetilde{\mathcal{C}}_{p_1,\ldots,p_n}$. Applying RES_A "in the same way", one can obtain a clause C', which does not contain neither non-XOR literals, nor $p_1, \ldots, p_n$. The set $\mathcal{C}'$, formed from such clauses C' for all possible combinations of $p_1 \in P_1$, $p_2 \in P_2$, $\ldots p_{k-1} \in P_{k-1}$, is an unsatisfiable set of clauses not containing non-XOR literals.

Finally, one can see that it is possible to implement the described procedure in $O(N_1 \times N_2 \times \cdots \times N_n \times 2^{N_a})$ time. □

Next we sketch the proof of completeness of temporal resolution, which is obtained combining the ideas of [Fisher *et al.*, 2001; Degtyarev *et al.*, 2002] and Lemma 1.

Definition 1 (Behaviour Graph) *We split the set of temporal clauses into three groups. Let $\mathcal{I}$ denote the* initial *clauses; $\mathcal{T}$ be the set of all* step *clauses; and $\mathcal{E}$ be the* sometime *clauses.*

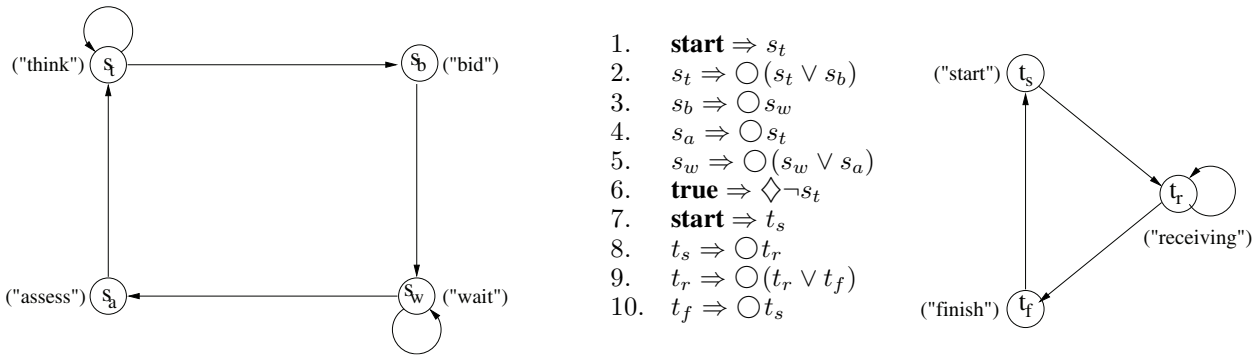
Given a set of clauses over a set of propositional symbols $\mathcal{P}$, we construct a finite directed graph G as follows. The nodes of G are interpretations of the set of propositions, that satisfy the XOR constraints over the XOR subsets. Notice that, because of the XOR-constraints, exactly one proposition from each set of XOR propositions P_i and any subset of propositions in A are true in I. This means that there at at most $N_1 \times N_2 \times \cdots \times N_n \times 2^{N_a}$ nodes in the behaviour graph.

For each node, I, we construct an edge in G to a node I' if, and only if, the following condition is satisfied:

- *For every step clause $(P \Rightarrow \bigcirc Q) \in \mathcal{T}$, if $I \models P$ then $I' \models Q$.*

A node, I, is designated an initial node of G if $I \models \mathcal{I}$. The behaviour graph *G of the set of clauses is the maximal subgraph of G given by the set of all nodes reachable from initial nodes.*

If G is empty then the set $\mathcal{I}$ is unsatisfiable. In this case there must exist a derivation by IRES_A and IRES_{P_k} as described in Lemma 1 (and in $O(N_1 \times N_2 \times \cdots \times N_n \times 2^{N_a})$ time).



1. **start** $\Rightarrow s_t$
2. $s_t \Rightarrow \bigcirc(s_t \vee s_b)$
3. $s_b \Rightarrow \bigcirc s_w$
4. $s_a \Rightarrow \bigcirc s_t$
5. $s_w \Rightarrow \bigcirc(s_w \vee s_a)$
6. **true** $\Rightarrow \Diamond\neg s_t$
7. **start** $\Rightarrow t_s$
8. $t_s \Rightarrow \bigcirc t_r$
9. $t_r \Rightarrow \bigcirc(t_r \vee t_f)$
10. $t_f \Rightarrow \bigcirc t_s$

Figure 2: Automata for agents S and T, together with corresponding clauses in normal form.

Now suppose G is not empty. Let I be a node of G which has no successors. Let $\{(P_i \Rightarrow \bigcirc Q_i)\}$ be the set of all step clauses such that $I \models P_i$, then $\wedge Q_i$ is unsatisfiable. Using Lemma 1, one can show that step resolution proves $\wedge P_i \Rightarrow \bigcirc$**false**. After the set of clauses is extended by the conclusion of the CONV rule, $\vee\neg P_i$, the node I is deleted from the graph.

In the case when all nodes of G have a successor, a contradiction can be derived with the help of the temporal resolution rule TRES. Note that we impose no restriction on this rule (it coincides with the temporal resolution rule for the general calculi presented in [Fisher *et al.*, 2001; Degtyarev *et al.*, 2002]) and the proof of completeness is no different from what is already published [Fisher *et al.*, 2001; Degtyarev *et al.*, 2002].

4 Complexity

Again, we consider initial and step resolution first.

Lemma 2 *Using the rules of initial and step resolution, it is possible to reduce a set of temporal clauses to one whose behaviour graph does not have nodes without successors in* $O\left(\left(N_1 \times N_2 \times \cdots \times N_n \times 2^{N_a}\right)^3\right)$ *time.*

Proof: Consider the following resolution strategy. For every set of propositions $p_1 \in P_1, \ldots, p_n \in P_n$ and $a \in A$, consider the set of all step-clauses

$$\hat{A}_1 \wedge \hat{P}^-_{11} \wedge \ldots \hat{P}^-_{n1} \Rightarrow \bigcirc(\check{P}^+_{11} \vee \ldots \check{P}^+_{n1} \vee \check{A}_1)$$

such that $\hat{A}_1, \hat{P}^-_{11}, \ldots, \ldots \hat{P}^-_{n1}$ do not contain any of $a, p_1, \ldots, p_n$ (there are at most $N_1 \times N_2 \times \cdots \times N_n \times 2^{N_a}$ such sets of clauses), and try establishing the unsatisfiability of the conjunction of the right-hand sides together with the universal clauses by step resolution (as Lemma 1 shows, this can be done in $O(N_1 \times N_2 \times \cdots \times N_n \times 2^{N_a})$ time. Then, all nodes without successors will be deleted from the behaviour graph (but some new such nodes may emerge). After $N_1 \times N_2 \times \cdots \times N_n \times 2^{N_a}$ repetitions, we obtain a graph in which every node has a successor. □

Lemma 3 *Given a set of temporal clauses, it is possible to find* L *such that* $L \Rightarrow \Box \bigwedge_k \neg q_k$*, as required for the* TRES *rule, in time polynomial in* $N_1 \times N_2 \times \cdots \times N_n \times 2^{N_a}$.

Proof: To find such L, it suffices to find a strongly connected component in the behaviour graph of the set of clauses, such that for every node I of this component, $I \models \bigwedge_k \neg q_k$. The simplest brute-force algorithm would analyse all pairs of nodes (and there are $(N_1 \times N_2 \times \cdots \times N_n \times 2^{N_a})^2$ such pairs), and this can be done more efficiently with step resolution as in [Degtyarev *et al.*, 2006]. □

Theorem 4 *If a set of temporal clauses is unsatisfiable, temporal resolution will deduce a contradiction in time polynomial in* $N_1 \times N_2 \times \cdots \times N_n \times 2^{N_a}$.

5 Example

Having described the underlying approach, we will now consider an example that makes use of some of these aspects. In particular, we will have multiple XOR fragments, together with standard propositions (unconstrained by XOR clauses). The example we will use is a simplification and abstraction of agent negotiation protocols; see, for example [Ballarini *et al.*, 2006]. Here, several (in our case, two) agents exchange information in order to come to some agreement. Each agent essentially has a simple control cycle, which can be represented as a finite state machine. In fact, we have simplified these still further, and sample basic control cycles are given in Fig. 2 (for both agents S and T).

Thus, we aim to use these automata as models of the agents, then formalise these within our logic. Importantly, we will add additional clauses (and propositions) characterising agreements or concurrency and, finally, we will show how our resolution method can be used to carry out verification.

We begin by characterising each agent separately as a set of clauses within our logic. To achieve this, we use a set of propositions for each agent. Thus, the automaton describing agent S is characterised through propositions of the form s_a, s_b, etc., while the automaton describing agent T is characterised using propositions such as t_r, t_s, etc. Both these sets are XOR sets. Thus, exactly one of s_a, s_b, ..., and exactly one of t_r, t_s, ..., must be true at any moment in time.

Now, the set of clauses characterising the two automata are given in Fig. 2. Regarding automaton S's description, note that clause 6 ensures that the automaton is infinitely often in

1.	**start**	$\Rightarrow$	s_t	
2.	$\neg s_b \wedge \neg s_w \wedge \neg s_a$	$\Rightarrow$	$\bigcirc(s_t \vee s_b)$	
3.	$\neg s_t \wedge \neg s_w \wedge \neg s_a$	$\Rightarrow$	$\bigcirc s_w$	
4.	$\neg s_t \wedge \neg s_b \wedge \neg s_w$	$\Rightarrow$	$\bigcirc s_t$	
5.	$\neg s_t \wedge \neg s_b \wedge \neg s_a$	$\Rightarrow$	$\bigcirc(s_w \vee s_a)$	
6.	**true**	$\Rightarrow$	$\Diamond(s_b \vee s_w \vee s_a)$	
7.	**start**	$\Rightarrow$	t_s	
8.	$\neg t_r \wedge \neg t_f$	$\Rightarrow$	$\bigcirc t_r$	
9.	$\neg t_s \wedge \neg t_f$	$\Rightarrow$	$\bigcirc(t_r \vee t_f)$	
10.	$\neg t_s \wedge \neg t_r$	$\Rightarrow$	$\bigcirc t_s$	
11.	**true**	$\Rightarrow$	$\Diamond agree$	
12.	$(agree \wedge \neg s_t \wedge \neg s_b \wedge \neg s_a \wedge \neg t_s \wedge \neg t_f)$	$\Rightarrow$	$\bigcirc s_a$	
13.	$(agree \wedge \neg s_t \wedge \neg s_b \wedge \neg s_a \wedge \neg t_s \wedge \neg t_f)$	$\Rightarrow$	$\bigcirc t_f$	
14.	$(\neg agree \wedge \neg s_t \wedge \neg s_b \wedge \neg s_a)$	$\Rightarrow$	$\bigcirc s_w$	
15.	$(\neg agree \wedge \neg t_s \wedge \neg t_f)$	$\Rightarrow$	$\bigcirc t_r$	
16.	$(agree \wedge \neg s_t \wedge \neg s_b \wedge \neg s_a \wedge \neg t_r)$	$\Rightarrow$	$\bigcirc s_w$	
17.	$(agree \wedge \neg s_w \wedge \neg t_s \wedge \neg t_f)$	$\Rightarrow$	$\bigcirc t_r$	
18.	**true**	$\Rightarrow$	$\bigcirc(s_b \vee s_w \vee s_a \vee t_r \vee t_f)$	
19.	$(\neg s_t \wedge \neg s_b \wedge \neg s_w \wedge \neg t_s \wedge \neg t_r)$	$\Rightarrow$	$\bigcirc$**false**	$[18, 10, 4\ \mathsf{SRES}_{P_k}]$
20.	**true**	$\Rightarrow$	$\bigcirc(s_t \vee s_b \vee s_w \vee t_s \vee t_r)$	$[19\ \mathsf{CONV}]$
21.	$(agree \wedge \neg s_t \wedge \neg s_b \wedge \neg s_a \wedge \neg t_s \wedge \neg t_f)$	$\Rightarrow$	$\bigcirc$**false**	$[20, 12, 13\ \mathsf{SRES}_{P_k}]$
22.	**true**	$\Rightarrow$	$\bigcirc(\neg agree \vee s_t \vee s_b \vee s_a \vee t_s \vee t_f)$	$[21\ \mathsf{CONV}]$
23.	$(\neg agree \wedge \neg s_t \wedge \neg s_b \wedge \neg s_a \wedge \neg t_s \wedge \neg t_f)$	$\Rightarrow$	$\bigcirc \neg agree$	$[22, 14, 15\ \mathsf{SRES}_{P_k}]$
24.	**true**	$\Rightarrow$	$\bigcirc(agree \vee s_t \vee s_b \vee s_a \vee t_s \vee t_f)$	$[23, 15, 14, 11\ \mathsf{TRES}]$
25.	**true**	$\Rightarrow$	$\bigcirc(s_t \vee s_b \vee s_a \vee t_s \vee t_f)$	$[24, 22\ \mathsf{SRES}_A]$
26.	$(\neg s_t \wedge \neg s_w \wedge \neg s_a)$	$\Rightarrow$	$\bigcirc(t_s \vee t_f)$	$[25, 3\ \mathsf{SRES}_{P_k}]$
27.	$(\neg agree \wedge \neg s_t \wedge \neg s_w \wedge \neg s_a \wedge \neg t_s \wedge \neg t_f)$	$\Rightarrow$	$\bigcirc$**false**	$[26, 15\ \mathsf{SRES}_{P_k}]$
28.	**true**	$\Rightarrow$	$\bigcirc(agree \vee s_t \vee s_w \vee s_a \vee t_s \vee t_f)$	$[27\ \mathsf{CONV}]$
29.	$(agree \wedge \neg s_t \wedge \neg s_w \wedge \neg s_a \wedge \neg t_s \wedge \neg t_f)$	$\Rightarrow$	$\bigcirc$**false**	$[26, 17\ \mathsf{SRES}_{P_k}]$
30.	**true**	$\Rightarrow$	$\bigcirc(\neg agree \vee s_t \vee s_w \vee s_a \vee t_s \vee t_f)$	$[29\ \mathsf{CONV}]$
31.	**true**	$\Rightarrow$	$\bigcirc(s_t \vee s_w \vee s_a \vee t_s \vee t_f)$	$[28, 30\ \mathsf{SRES}_A]$
32.	$\neg s_b \wedge \neg s_w \wedge \neg s_a$	$\Rightarrow$	$\bigcirc(s_t \vee t_s \vee t_f)$	$[31, 2\ \mathsf{SRES}_{P_k}]$
33.	$(\neg agree \wedge \neg s_b \wedge \neg s_w \wedge \neg s_a \wedge \neg t_s \wedge \neg t_f)$	$\Rightarrow$	$\bigcirc s_t$	$[32, 15\ \mathsf{SRES}_{P_k}]$
34.	$(agree \wedge \neg s_b \wedge \neg s_w \wedge \neg s_a \wedge \neg t_s \wedge \neg t_f)$	$\Rightarrow$	$\bigcirc s_t$	$[32, 17\ \mathsf{SRES}_{P_k}]$
35.	**true**	$\Rightarrow$	$\bigcirc(s_b \vee s_w \vee s_a \vee t_s \vee t_f)$	$[33, 15, 34, 17, 6\ \mathsf{TRES}]$
36.	$(\neg agree \wedge \neg s_t \wedge \neg s_b \wedge \neg s_w \wedge \neg t_s \wedge \neg t_f)$	$\Rightarrow$	$\bigcirc$**false**	$[35, 15, 4\ \mathsf{SRES}_{P_k}]$
37.	**true**	$\Rightarrow$	$\bigcirc(agree \vee s_t \vee s_b \vee s_w \vee t_s \vee t_f)$	$[36\ \mathsf{CONV}]$
38.	$(agree \wedge \neg s_t \wedge \neg s_b \wedge \neg s_w \wedge \neg t_s \wedge \neg t_f)$	$\Rightarrow$	$\bigcirc$**false**	$[35, 17, 4\ \mathsf{SRES}_{P_k}]$
39.	**true**	$\Rightarrow$	$\bigcirc(\neg agree \vee s_t \vee s_b \vee s_w \vee t_s \vee t_f)$	$[38\ \mathsf{CONV}]$
40.	**true**	$\Rightarrow$	$\bigcirc(s_t \vee s_b \vee s_w \vee t_s \vee t_f)$	$[37, 39\ \mathsf{SRES}_A]$
41.	**true**	$\Rightarrow$	$\bigcirc(t_s \vee t_f)$	$[40, 35, 31, 25\ \mathsf{SRES}_{P_k}]$
42.	$\neg t_r \wedge \neg t_f$	$\Rightarrow$	$\bigcirc$**false**	$[41, 8\ \mathsf{SRES}_{P_k}]$
43.	**start**	$\Rightarrow$	$t_r \vee t_f$	$[42\ \mathsf{CONV}]$
44.	**start**	$\Rightarrow$	**false**	$[43, 7\ \mathsf{IRES}_{P_k}]$

Figure 3: Resolution Proof for Automata Agents Example.

a state other than s_t, ensuring that the automaton can not remain in state s_t forever.

We can also characterise how the computations within each automaton relate. To begin with, we assume a simple, synchronous, concurrent model where both automata make a transition at the same time (see Section 5 for variations on this). Next we add a key aspect in negotiation protocols, namely a description of what happens when an *agreement* is reached. In our example, this is characterised as a synchronised communication act. Logically, we use the proposition $agree$ to denote this, and add the following clauses.

11. **true** $\Rightarrow \Diamond agree$
12. $(agree \wedge s_w \wedge t_r) \Rightarrow \bigcirc s_a$
13. $(agree \wedge s_w \wedge t_r) \Rightarrow \bigcirc t_f$
14. $(\neg agree \wedge s_w) \Rightarrow \bigcirc s_w$
15. $(\neg agree \wedge t_r) \Rightarrow \bigcirc t_r$
16. $(s_w \wedge agree \wedge \neg t_r) \Rightarrow \bigcirc s_w$
17. $(\neg s_w \wedge agree \wedge t_r) \Rightarrow \bigcirc t_r$

Here, we say that agreements *will* occur infinitely often in the future (clause 11). Clauses 12 and 13 capture the exact synchronisation. If an agreement occurs while automaton S is in state s_w and automaton T is in t_r, then the automata make transitions forward to states s_a and t_f respectively. Finally, clauses 14–17 ensure that, if no synchronised agreement is possible, then the automata remain in their relevant states.

The clauses above represent the specification of a simple system. As an example of how resolution can be used, we also wish to verify that the system is *simultaneously* in states s_t and t_s eventually. To verify this, we add the negation of this property, as characterised by clause 18:

18. **true** $\Rightarrow \bigcirc(\neg s_t \vee \neg t_s)$

Thus, if we can derive a contradiction from clauses 1–18 then we know the negated property is valid for this specification. We first rewrite clauses 1–18 in the correct format for the normal form. The refutation is given in Figure 3.

The example above essentially captures activity within a synchronous, truly concurrent, system. If we wish to move to

more complex models of computation, we can do so, essentially by introducing the notion of a *turn*. Thus, when it is automaton S's turn to move, $turn_s$ is true; when it is automaton T's turn to move, $turn_t$ is true. Then, each clause describing an automaton transition, for example, 3. $s_b \Rightarrow \bigcirc s_w$ is replaced by two clauses

$$\begin{array}{lrcl} 3a. & (s_b \wedge turn_s) & \Rightarrow & \bigcirc s_w \\ 3b. & (s_b \wedge \neg turn_s) & \Rightarrow & \bigcirc s_b \,. \end{array}$$

In the example above, $turn_s$ and $turn_t$ are effectively both true together (and forever). However, we can modify the synchronisation clauses and model a different form of concurrency. For example, if we were to introduce *interleaving* concurrency, we might use the following clauses[1]:

$$\textbf{start} \Rightarrow turn_s \quad turn_s \Rightarrow \bigcirc turn_t \quad turn_t \Rightarrow \bigcirc turn_s$$

If we go further still, and introduce an asynchronous model of concurrency, then we might get

$$\textbf{true} \Rightarrow \Diamond turn_s \qquad \textbf{true} \Rightarrow \Diamond turn_t$$

In both the above cases if we want to ensure that exactly one of $turn_s$ and $turn_t$ hold at each moment we implicitly have $\Box(turn_s \oplus turn_t)$ and so we are effectively using TLX($\mathcal{S}$,$\mathcal{T}$,$\{turn_s, turn_t\}$).

6 Concluding Remarks and Related Work

In this paper we have developed a tractable sub-class of temporal logic, based on the central use of XOR operators. This logic can be decided, tractably, via clausal temporal resolution. Importantly, multiple XOR fragments can be combined. This new approach to temporal reasoning provides a framework in which tractable temporal logics can be engineered by intelligently combining appropriate XOR fragments. Further, this has the potential to provide a deductive approach, with a similar complexity to model checking, thus obtaining a practical verification method. In addition, this approach has the potential to be extended to first-order temporal logics which can deal with infinite state systems.

The complexity result means that TLX is more amenable to efficient implementation than other similar temporal logics. Moreover, since no two propositions from the same XOR set can occur in the right- (or left-) hand side of any temporal clause, one can efficiently represent disjunctions of (positive) propositions (and conjunctions of negated propositions) as bit vectors and the rules of temporal resolution as bit-wise operations on such bit vectors. Thus, temporal reasoning in TLX can be efficient not only in theory, but also in practice.

Demri and Schnoebelen [2002] consider sub-fragments of PTL, particularly those restricting the number of propositions, the temporal operators allowed, and the depth of temporal nesting in formulae. Demri and Schnoebelen show that, since the formulae tackled in practical model checking often fall within such fragments, then this provides a natural explanation for the viability of model checking in PTL.

Recent results relating to a clausal resolution calculus for propositional temporal logics can be found in [Fisher *et al.*, 2001; Hustadt and Konev, 2003; Hustadt *et al.*, 2004]. Since deciding unsatisfiability of PTL is also PSPACE-complete, then deductive verification of PTL formulae would seem to be an impractical way to proceed. However, just as Demri and Schnoebelen showed how PTL model checking can be seen as being tractable when we consider fragments of PTL, so we have been examining fragments of PTL that allow clausal resolution to be tractable. The fine grained complexity analysis shows that the calculus is polynomial in the number of XOR propositions (and exponential in the non-XOR propositions) making it efficient for problems with large numbers of XOR propositions and just a few non-XOR propositions.

Related to the fragment presented in this paper is a more restricted case in [Dixon *et al.*, 2006] which can be used to represent Büchi Automata. In that paper, a particular fragment allowing two XOR sets of propositions but where the allowable clauses were further restricted is considered and a polynomial resolution calculus given. One can show that every resolvent within that calculus can be derived by applying resolution rules from the resolution calculus proposed in this paper restricted to two XOR sets.

[1]Note that a different model of concurrency might also require modification in the *agreement* clauses.

References

[Ballarini *et al.*, 2006] P. Ballarini, M. Fisher, and M. Wooldridge. Automated Game Analysis via Probabilistic Model Checking: a case study. *Electronic Notes in Theoretical Computer Science*, 149(2):125–137, 2006.

[Clarke *et al.*, 1999] E.M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, December 1999.

[Degtyarev *et al.*, 2002] A. Degtyarev, M. Fisher, and B. Konev. A Simplified Clausal Resolution Procedure for Propositional Linear-Time Temporal Logic. In *Proc. TABLEAUX-02*, *LNCS* vol. 2381, pages 85–99. Springer-Verlag, 2002.

[Degtyarev *et al.*, 2006] A. Degtyarev, M. Fisher, and B. Konev. Monodic Temporal Resolution. *ACM Transactions on Computational Logic*, 7(1), January 2006.

[Demri and Schnoebelen, 2002] S. Demri and P. Schnoebelen. The Complexity of Propositional Linear Temporal Logic in Simple Cases. *Information and Computation*, 174(1):84–103, 2002.

[Dixon *et al.*, 2006] C. Dixon, M. Fisher, and B. Konev. Is There a Future for Deductive Temporal Verification? In *Proc. TIME-06*. IEEE Computer Society Press, 2006.

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

[Gabbay *et al.*, 1980] D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. The Temporal Analysis of Fairness. In *Proc. POPL-80*, pages 163–173, January 1980.

[Hustadt and Konev, 2003] U. Hustadt and B. Konev. TRP++ 2.0: A Temporal Resolution Prover. In *Proc. CADE-19*, *LNAI* vol. 2741, pages 274–278. Springer, 2003.

[Hustadt *et al.*, 2004] U. Hustadt, B. Konev, A. Riazanov, and A. Voronkov. TeMP: A Temporal Monodic Prover. In *Proc. IJCAR-04*, *LNAI* vol. 3097, pages 326–330. Springer, 2004.

[Schaefer, 1978] T. J. Schaefer. The Complexity of Satisfiability Problems. In *Proc. STOC-78*, pages 216–226, 1978.

[Sistla and Clarke, 1985] A. P. Sistla and E. M. Clarke. Complexity of Propositional Linear Temporal Logics. *Journal of the ACM*, 32(3):733–749, July 1985.