Labelled Superposition for PLTL

Martin Suda and
Christoph Weidenbach

MPI–I–2012–RG1-001 January 2012

Authors' Addresses

Martin Suda
Max-Planck-Institut für Informatik
Campus E1 4
66123 Saarbrücken
Germany

Christoph Weidenbach
Max-Planck-Institut für Informatik
Campus E1 4
66123 Saarbrücken
Germany

Publication Notes

This report is an extended version of an article published elsewhere.

Abstract

This paper introduces a new decision procedure for PLTL based on labelled superposition. Its main idea is to treat temporal formulas as infinite sets of purely propositional clauses over an extended signature. These infinite sets are then represented by finite sets of labelled propositional clauses. The new representation enables the replacement of the complex temporal resolution rule, suggested by existing resolution calculi for PLTL, by a fine grained repetition check of finitely saturated labelled clause sets followed by a simple inference. The completeness argument is based on the standard model building idea from superposition. It inherently justifies ordering restrictions, redundancy elimination and effective partial model building. The latter can be directly used to effectively generate counterexamples of non-valid PLTL conjectures out of saturated labelled clause sets in a straightforward way.

Keywords

Propositional Linear Temporal Logic, Superposition, Resolution

Contents

1 Introduction

Propositional linear temporal logic [25] is an extension of classical propositional logic for reasoning about time. It introduces temporal operators such as $\Diamond P$ meaning P holds *eventually* in the future, $\Box P$ meaning P holds *always* in the future, and $\bigcirc P$ meaning P holds at the *next* time point. Time is considered to be a linear discrete sequence of time points represented by propositional valuations, called *worlds*. Such a potentially infinite sequence forms a PLTL interpretation. A decision procedure for PLTL takes a PLTL formula P and checks whether it is valid, i.e., that all PLTL interpretations are actually models for P. For example, the PLTL formula $\Box P \rightarrow \bigcirc P$ is valid (a theorem) whereas the PLTL formula $\bigcirc P \rightarrow \Box P$ is not, but is satisfiable, i.e., there is a PLTL model for it.

Attempts to use clausal resolution to attack the decision problem for PLTL appeared first in [4, 31]. The most recent resolution-based approach is the one of [13]. It relies on a satisfiability preserving clausal translation of PLTL formulas, where, in particular, all nestings of temporal operators are reduced to formulas (and, eventually, clauses) of the form P, $\Box(P \rightarrow \bigcirc Q)$, and $\Box(P \rightarrow \Diamond Q)$, where P and Q do not contain temporal operators. Classical propositional resolution is extended to cope with "local" temporal reasoning within neighbouring worlds, while an additional inference rule called *temporal resolution* is introduced to deal with *eventuality* ($\Box\Diamond$) clauses. The temporal resolution rule is quite complex. It requires a search for certain combinations of clauses that together form a *loop*, i.e. imply that certain sets of worlds must be discarded from consideration, because an eventuality clause would be unsatisfied forever within them. This is verified via an additional proof task. Finally, the conclusion of the rule needs to be transformed back into the clause form.

Our labelled superposition calculus builds on a refinement of the above clause normal form [5]. It introduces a notion of labelled clauses in the spirit of [22] and replaces the temporal resolution rule by saturation and a new *Leap* rule. Although in PLTL equality is not present, the principles of

superposition are fundamental for our calculus. Our completeness result is based on a model generation approach with an inherent redundancy concept based on a total well-founded ordering on the propositional atoms.

The main contributions of our work are: 1) we replace the temporal resolution rule by a much more streamlined saturation of certain labelled clauses followed by a simple Leap inference, 2) our inference rules are guided by an ordering restriction that is known to reduce the search space considerably, 3) the completeness proof justifies an abstract redundancy notion that enables strong reductions, 4) if a contradiction cannot be derived, a temporal model can be extracted from a saturated clause set.

This report is organized as follows. We fix our notation and formalize the problem to be solved in Chapter 2. Then in Chapter 3 we show how to use labelled clauses as a tool to "lift" the standard propositional calculus to reason about PLTL-satisfiability. Our calculus is introduced in Chapter 4 and used as a basis for an effective decision procedure in Chapter 5. We deal with abstract redundancy, its relation to the completeness proof, and model building in Chapter 6. Discussion of previous work and an experimental comparison to existing resolution approaches appear in Chapter 7 and Chapter 8, respectively. Finally, Chapter 9 concludes.

2 Preliminaries

2.1 PLTL, SNF, and specifications

Propositional Linear Temporal Logic, PLTL, is an extension of classical propositional logic by temporal operators. Its language is built over signature $\Sigma = \{p, q, \ldots\}$ of propositional variables, using the standard propositional connectives together with unary operators $\bigcirc$ ('at the next moment'), $\Box$ ('always in the future'), $\Diamond$ ('eventually in the future'), and U ('until') (see e.g. [11]). Thus the set of PLTL formulas is defined as follows: any propositional variable $p \in \Sigma$ is an *atomic* PLTL formula or simply an *atom*, if P and Q are PLTL formulas then so are $\neg P$, $P \vee Q$, $\bigcirc P$, $\Box P$, $\Diamond P$, $P \mathsf{U} Q$. As usual, a (standard) *literal* is an atomic formula or its negation, and a (standard) *clause* is a multiset of literals understood as their disjunction. In the following, the symbol $\mathbb{N}$ stands for the naturals, and $\mathbb{N}^+$ denotes the set $\mathbb{N} \setminus \{0\}$.

By a propositional *valuation*, or simply a *world*, we mean a mapping $V : \Sigma \to \{0, 1\}$. We write $V \models P$ if a propositional formula P is satisfied by V. The semantics of PLTL is based on discrete linear model of time, where the structure of possible time points is isomorphic to $\mathbb{N}$. A *PLTL-interpretation* is a sequence $\mathfrak{V} = (V_i)_{i \in \mathbb{N}}$ of propositional valuations. The truth relation $\mathfrak{V}_i \models P$ is defined recursively as follows:

$\mathfrak{V}_i \models p$	iff $V_i \models p$
$\mathfrak{V}_i \models \neg P$	iff not $\mathfrak{V}_i \models P$
$\mathfrak{V}_i \models P \vee Q$	iff $\mathfrak{V}_i \models P$ or $\mathfrak{V}_i \models Q$
$\mathfrak{V}_i \models \bigcirc P$	iff $\mathfrak{V}_{i+1} \models P$
$\mathfrak{V}_i \models \Box P$	iff for every $j \geq i$, $\mathfrak{V}_j \models P$
$\mathfrak{V}_i \models \Diamond P$	iff for some $j \geq i$, $\mathfrak{V}_j \models P$
$\mathfrak{V}_i \models P \mathsf{U} Q$	iff there is $j \geq i$ such that $\mathfrak{V}_j \models Q$ and $\mathfrak{V}_k \models P$ for every k, $i \leq k < j$

A PLTL-interpretation $\mathfrak{V}$ is a *model* of a formula P if $\mathfrak{V}_0 \models P$. We say that

a formula P is *satisfiable* if there exists a model of P. Formula P is said to be *valid* if every PLTL-interpretation is a model of P.

Refutational theorem proving is based on the observation that a formula is valid (i.e. a theorem) if and only if its negation is unsatisfiable. The negated formula is typically first transformed into an equisatisfiable normal form to allow efficient automation of the following process, which seeks to derive a contradiction and thus to *refute* the negated formula.

The approach presented in this paper is based on a normal form called Separated Normal Form (SNF) described in [13] and further simplified in [5]. The translation to SNF is based on a renaming and unwinding technique which substitutes non-atomic subformulas by new propositional symbols and their definitions, and replaces the temporal operators by their fixpoint definitions; see [13]. The starting point of our work is our own variant of SNF introduced in Definition 1. It no longer explicitly mentions any temporal operator, which brings it nearer to the classical setup, but requires additional conventions.

In order to be able to talk about several neighbouring worlds at once we introduce copies (i.e. pairwise disjoint, bijectively equivalent sets) of the basic signature Σ. We use priming to denote the shift from one signature to the next (thus Σ' is the set of symbols $\{p', q', \ldots\}$), and shorten repeated primes by parenthesised integers (e.g. p''' is the same thing as $p^{(3)}$). This notational convention can be extended from symbols and signatures to formulas, and also to valuations in a natural way. For example, if V is a valuation over $\Sigma^{(i)}$ we write V' for the valuation over $\Sigma^{(i+1)}$ such that $V'(p^{(i+1)}) = V(p^{(i)})$ for every $p \in \Sigma$. We also need to consider formulas over two consecutive joined signatures, e.g. over $\Sigma \cup \Sigma'$. Such formulas can be evaluated over the respective joined valuations. When both V_1 and V_2 are valuations over Σ, we write $[V_1, V_2]$ as a shorthand for the mapping $V_1 \cup (V_2)' : (\Sigma \cup \Sigma') \to \{0, 1\}$.

Definition 1. *A* PLTL-specification S *is a quadruple* (Σ, I, T, G) *such that*

- Σ *is a finite propositional signature,*
- I *is a set of* initial *clauses* C_i *(over the signature* Σ*),*
- T *is a set of* step *clauses* $C_t \vee D'_t$ *(over the joint signature* $\Sigma \cup \Sigma'$*),*
- G *is a set of* goal *clauses* C_g *(over the signature* Σ*).*

The initial and step clauses match their counterparts from [13] in the obvious way.[1] Our goal clauses are a generalization of a single unconditional

[1]Note that unlike the previous work where temporal clauses are introduced, we only work with propositional clauses in specifications and leave their temporal meaning to be determined by context.

sometimes clause that can be obtained using the transformations described in [5]. The whole specification represents the PLTL formula:

$$\left(\bigwedge C_i\right) \wedge \Box\left(\bigwedge (C_t \vee \bigcirc D_t)\right) \wedge \Box\Diamond\left(\bigwedge C_g\right).$$

Example 1. *We will be using the valid PLTL formula* $\Box((a \to b) \to \bigcirc b) \to \Diamond\Box(a \vee b)$ *as a running example that will guide us through the whole theorem proving process presented in this paper. By negating the formula and performing standard transformations we obtain* $\Box(a \vee \bigcirc b) \wedge \Box(\neg b \vee \bigcirc b) \wedge \Box\Diamond(\neg a \wedge \neg b)$*, which gives us the following PLTL-specification* $S = (\{a, b\}, \emptyset, \{a \vee b', \neg b \vee b'\}, \{\neg a, \neg b\})$.

It is a known fact that when considering satisfiability of PLTL formulas attention can be restricted to *ultimately periodic* [28] interpretations. These start with a finite sequence of worlds and then repeat another finite sequence of worlds forever. This observation, which is also one of the key ingredients of our approach, motivates the following definition.

Definition 2. *Let* $K \in \mathbb{N}$*, and* $L \in \mathbb{N}^+$ *be given. A PLTL-interpretation* $(V_i)_{i\in\mathbb{N}}$ *is a* (K, L)-model *of* $S = (\Sigma, I, T, G)$ *if*

1. *for every* $C \in I$*,* $V_0 \models C$*,*
2. *for every* $i \in \mathbb{N}$ *and every* $C \in T$*,* $[V_i, V_{i+1}] \models C$*,*
3. *for every* $i \in \mathbb{N}$ *and every* $C \in G$*,* $V_{(K+i\cdot L)} \models C$*.*

A PLTL-specification is satisfiable *if it has a* (K, L)*-model for some* K *and* L*.*

Note that the eventuality represented by the goal clauses of S is satisfied infinitely often as the standard PLTL semantics dictates. Moreover, we keep track of the worlds where this is bound to happen by requiring they form an arithmetic progression with K as the initial term and L the common difference. This additional requirement doesn't change the notion of satisfiability thanks to the observation mentioned above. We will call the pair (K, L) the *rank* of a model.

We close this section by providing a more detailed argument on why restricting our attention to (K, L)-models doesn't change the notion of satisfiabilit/validity of PLTL-specifications.

Definition 3. *A PLTL-interpretation* $(V_i)_{i\in\mathbb{N}}$ *is an* unconstrained model *of* $S = (\Sigma, I, T, G)$ *if*

1. *for every* $C \in I$, $V_0 \models C$,

2. *for every* $i \in \mathbb{N}$ *and every* $C \in T$, $[V_i, V_{i+1}] \models C$,

3. *for every* $i \in \mathbb{N}$ *there is* $j \geq i$ *such that for every* $C \in G$, $V_j \models C$.

A PLTL-specification S *is* standard satisfiable *if it has an unconstrained model.*

Lemma 1. *Any standard satisfiable PLTL-specification* $S = (\Sigma, I, T, G)$ *has a* (K, L)*-model for some* $K \in \mathbb{N}$ *and* $L \in \mathbb{N}^+$.

Proof. Let $(V_i)_{i\in\mathbb{N}}$ be an unconstrained model of S. There are only finitely many different valuations over Σ, so only finitely many that satisfy all the clauses $C \in G$. At least one of them thus has to appear infinitely often in $(V_i)_{i\in\mathbb{N}}$. Let $K \in \mathbb{N}$ be an index such that V_K is such a valuation, and let L be the smallest number in $\mathbb{N}^+$ such that $V_K = V_{K+L}$. We now define a new PLTL-interpretation $(W_i)_{i\in\mathbb{N}}$ by setting

- $W_i = V_i$ for every $i \leq K$,
- $W_i = V_{K+(i-K) \bmod L}$ for any $i > K$.

It is easy to see that $(W_i)_{i\in\mathbb{N}}$ is a (K, L)-model of S. □

Note that the model $(W_i)_{i\in\mathbb{N}}$ constructed in the proof of the previous lemma is an *ultimately periodic* model of [28], i.e. from a certain time point the respective valuations repeat periodically. The notion of (K, L)-models is slightly more general as it only requires that all the goal clauses are satisfied in periodically recurring worlds.

2.2 Ordered resolution and model construction

Now we briefly review the ordered resolution calculus for classical propositional logic [2], that plays an important role "in the background", and we refer to it in our completeness result. Although we do not reason about equality in this paper, we denote the calculus *PSup* (as a shorthand for Propositional Superposition) to stress the presence of an inherent redundancy concept based on an ordering on the propositional atoms.

The calculus is parameterized by a strict well-founded ordering[2] on the signature $<$ which is further extended to literals by setting $A < \neg A$ and $(\neg)A < (\neg B)$ if and only if $A < B$. It consists of the following two inference rules:

Ordered Resolution **Ordered Factoring**

$$\mathcal{I}\frac{C \vee A \quad D \vee \neg A}{C \vee D} \qquad \mathcal{I}\frac{C \vee A \vee A}{C \vee A}$$

where the atom A is maximal in C, and $\neg A$ is maximal in D, meaning that there is no greater literal w.r.t $<$ in the respective clauses. The clauses above the line are referred to as *premises*, and the clause below as the inference's *conclusion*. As usual, the inferences are used to derive new clauses from a given clause set with the ultimate goal of deriving the empty clause $\bot$, which means the given clauses set is contradictory.

Ordered resolution comes also with a strong notion of redundancy, i.e. a way to recognize that a particular clause in the context of the other clauses is not needed for deriving the empty clause and can be removed without compromising completeness. The most important instances of redundancy are the following reduction rules.

Subsumption **Tautology Deletion**

$$\mathcal{R}\frac{C \quad D}{C} \qquad \mathcal{R}\frac{E}{}$$

where C is a sub-multiset of D, and E contains two complementary literals A and $\neg A$. Note that the reduction's conclusion is not simply added to the generated clause set but replaces the reduction's premises. This, e.g., amounts to simple deletion in the case of our second reduction. By *saturation* we mean the process of updating a clause set by inferences and reductions, and we call the resulting clause set, after a fixpoint has been reached, *saturated*. Further details can be found in [2].

We conclude this section by having a closer look on the abstract redundancy notion and model building operator which is used in the completeness proof of *PSup*. They both rely on a multiset extension $<^c$ of the ordering $<$ on literals described earlier to compare clauses (i.e. multisets of literals), which is also necessarily well-founded.

[2]Typically, there is also a *selection function* of negative literals, that may be used to further influence the search space. In order to keep things simple, we don't include selection function to our presentation.

Definition 4. *A standard clause C is* redundant *with respect to a set of standard clauses N, if there are clauses $C_1, \ldots, C_n \in N$ such that for every $i = 1, \ldots n$, $C_i <^c C$, and $C_1, \ldots, C_n \models C$.*

Definition 5. *A set of clauses N is* saturated up to redundancy*, if for every PSup inference from N such that its premises are not redundant w.r.t. N, the conclusion is either redundant w.r.t. N or contained in N.*

Now the model is constructed by considering which literals have to be satisfied in a given clause, starting from the smallest clause w.r.t. the clause ordering. Note that we follow standard convention here and consider *propositional interpretation* to be a set of atoms. Our notion of valuation introduced earlier is, in fact, a characteristic function of this interpretation.

Definition 6 (Model Construction)**.** *Let $<^c$ be a multiset extension of literal ordering $<$ and let N be a set of clauses. For a clause $C \in N$ we define by a well-founded recursion over $<^c$ a propositional interpretation $\mathcal{I}^{<^c}(C)$ and a set ϵ_C as follows. We set $\mathcal{I}^{<^c}(C) = \bigcup_{D <^c C} \epsilon_D$, and if the clause C*

- *is of the form $C_0 \vee A$, where the atom A is the maximal literal in C, and*
- *is false in $\mathcal{I}^{<^c}(C)$,*

then we set $\epsilon_C = \{A\}$; otherwise $\epsilon_C = \emptyset$. Finally, we define $\mathcal{I}^{<^c}(N) = \bigcup_{C \in N} \epsilon_C$.

A clause C is said to be *productive* and said to *produce* the atom A if and only if $\epsilon_C = \{A\}$.

Theorem 1 ([2])**.** *Let N be a set of clauses that is saturated up to redundancy w.r.t. PSup, and does not contain the empty clause, then N is satisfiable. In fact, $\mathcal{I}^{<^c}(N) \models N$.*

3 Labelled clauses

Recall that we defined a PLTL-interpretation as an infinite sequence of propositional valuations over the finite signature Σ. Alternatively though, it can be viewed as a *single* propositional valuation over the *infinite* signature $\Sigma^* = \bigcup_{i \in \mathbb{N}} \Sigma^{(i)}$. We simply index the signature symbols by the time moments to obtain this isomorphic representation. If we now examine Definition 2 of (K, L)-models from this perspective, we can reveal a simple (though at first sight not very useful) reduction of satisfiability in a (K, L)-model to propositional satisfiability of a potentially infinite set of clauses over Σ^*. For a specification $S = (\Sigma, I, T, G)$ this clause set will consist of copies of the clauses from I, T, and G that are "shifted in time" to proper positions, such that the whole set is (propositionally) satisfiable if and only if S has a (K, L)-model. Formally, the set is the union of $\{C^{(0)} \mid C \in I\}$, $\{C^{(i)} \mid C \in T, i \in \mathbb{N}\}$, and $\{C^{(K+i \cdot L)} \mid C \in G, i \in \mathbb{N}\}$. See Fig. 3.1 for the intuition behind this idea.

In order to make use of the above described reduction we need to show how to solve for infinitely many values of K and L the propositional satisfiability

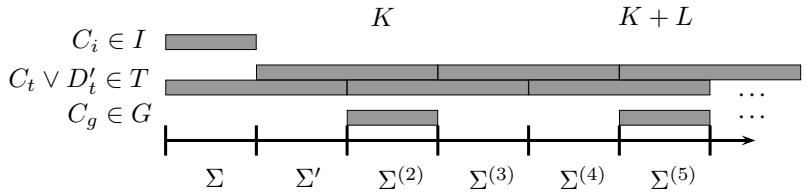

Figure 3.1: Schematic presentation of the potentially infinite set of clauses that is satisfiable iff a PLTL-specification $S = (\Sigma, I, T, G)$ has a model of rank $(2, 3)$. The axis represents the the infinite signature Σ^*, while the grey bars stand for the individual copies of the initial, step, and goal clauses, respectively.

problem consisting of infinitely many clauses. We do this by assigning *labels* to the clauses of S such that a labelled clause represents up to infinitely many standard clauses over Σ^*. Then an inference performed between labelled clauses corresponds to infinitely many inferences on the level of Σ^*. This is not dissimilar to the idea of "lifting" from first-order theorem proving where clauses with variables represent up to infinitely many ground instances. Here, however, we deal with the additional dimension of performing infinitely many proof tasks "on the ground level" in parallel, one for each rank (K, L).

Formally, a *label* is a pair (b, k) where b is either $*$ or 0, and k is either $*$ or an element of $\mathbb{N}$. A *labelled clause* is a pair $(b, k) \,||\, C$ consisting of a label and a (standard) clause over $\Sigma \cup \Sigma'$. Given a PLTL-specification $S = (\Sigma, I, T, G)$, the *initial labelled clause set* N_S for S is defined to contain

- labelled clauses of the form $(0, *) \,||\, C$ for every $C \in I$,
- labelled clauses of the form $(*, *) \,||\, C$ for every $C \in T$, and
- labelled clauses of the form $(*, 0) \,||\, C$ for every $C \in G$.

We can think of the first label component b as relating the clause to the beginning of time, while the second component relates the clause to the indices of the form $K + i \cdot L$, where the goal should be satisfied. In both cases, $*$ stands for a "don't care" value, thus, e.g., the label $(*, *)$ marks clauses that occupy every possible index. It turns out that during inferences we also need to talk about clauses that reside k steps before indices of the goal. That is why the second label component may assume any value from $\mathbb{N}$. The semantics of labels is given via a map to world indices. Formal definition of labels' semantics is given next.

Let (K, L) be a rank. We define a set $R_{(K,L)}(b, k)$ of indices *represented* by the label (b, k) as the set of all $t \in \mathbb{N}$ such that

$$[b \neq * \to t = 0] \wedge [k \neq * \to \exists s \in \mathbb{N}. t + k = K + s \cdot L] \ .$$

Observe that while $R_{(K,L)}(0, k) \subseteq \{0\}$, the sets $R_{(K,L)}(*, k)$ are always infinite, and for $k \in \mathbb{N}$ constitute a range of an arithmetic progression with difference L. Now a standard clause of the form $C^{(t)}$ is said to be *represented by the labelled clause* $(b, k) \,||\, C$ *in* (K, L) if $t \in R_{(K,L)}(b, k)$. We denote the set of all standard clauses represented in (K, L) by the labelled clauses N by the symbol $N_{(K,L)}$. In mathematical notation we obtain defining equation

$$N_{(K,L)} = \{C^{(t)} \,|\, \text{clause } (b, k) \,||\, C \in N \text{ and } t \in R_{(K,L)}(b, k)\} \ .$$

Example 2. *Our example specification* $S = (\{a, b\}, \emptyset, \{a \vee b', \neg b \vee b'\}, \{\neg a, \neg b\})$ *contains among others the single literal goal clause* $\neg a$*. In the initial labelled clause set* N_S *this goal clause becomes* $(*, 0) \,||\, \neg a$*. If we now, for example, fix the same rank* $(2, 3)$ *as in Fig. 3.1, our labelled clause will in that rank represent all the standard clauses* $(\neg a)^{(t)}$ *with* $t \in R_{(2,3)}(*, 0) = \{2, 5, 8, \dots\}$.

We summarize the main message of this chapter in the next lemma. Its proof follows from the definitions and ideas already given.

Lemma 2. *Let a rank* (K, L) *and a PLTL-specification* S *be given and let* N_S *be the initial labelled clause set for* S*. Then the set* $(N_S)_{(K,L)}$ *is satisfiable if and only if* S *has a* (K, L)*-model.*

Proof. Let us fix a rank (K, L) and a PLTL-specification $S = (\Sigma, I, T, G)$. First, we follow the definitions to show that

$$(N_S)_{(K,L)} = \{C^{(0)} \,|\, C \in I\} \cup \{C^{(i)} \,|\, C \in T, i \in \mathbb{N}\} \cup \{C^{(K+i \cdot L)} \,|\, C \in G, i \in \mathbb{N}\}.$$

Clauses $C \in I$ are turned into labelled clauses of the form $(0, *) \,||\, C \in N_S$. Their label's semantics dictates that they in (K, L) represent standard clauses $C^{(0)} = C$. Clauses $C \in T$ are turned into labelled clauses of the form $(*, *) \,||\, C \in N_S$. Their label's semantics dictates that they in (K, L) represent standard clauses $C^{(i)}$ for any $i \in \mathbb{N}$. Clauses $C \in G$ are turned into labelled clauses of the form $(*, 0) \,||\, C \in N_S$. Their label's semantics dictates that they in (K, L) represent standard clauses $C^{(K+i \cdot L)}$ for any $i \in \mathbb{N}$.

Satisfiability of $(N_S)_{(K,L)}$ can now be transferred to the existence of a (K, L)-model and back, by applying the following bijection between valuations V^* over the extended signature $\Sigma^* = \bigcup_{i \in \mathbb{N}} \Sigma^{(i)}$ on the one hand, and PLTL-interpretations $(V_i)_{i \in \mathbb{N}}$ on the other. The bijection is defined by the equation

$$V_i(p) = V^*(p^{(i)})$$

for every $p \in \Sigma$. □

4 Labelled superposition calculus *LPSup*

In this chapter we present our calculus for labelled clauses *LPSup*. We continue building on the idea that labelled clauses represent standard clauses from the "ground level" of deciding the existence of (K, L)-models, and show how to "lift" the operation of a sound and complete calculus on these ground proof tasks and abbreviate it into a single saturation process on the level of labelled clauses. In particular, we lift the ordered resolution calculus of [2], which we call *PSup* for Propositional Superposition, and transfer to *LPSup* its valuable properties, including the ordering restrictions of inferences.[1] For that purpose, we parameterize *LPSup* by a total ordering $<$ on the symbols of the signature Σ, which we implicitly extend to indexed signatures by first comparing the indices and only then the actual symbols. This means that $p^{(i)} < q^{(j)}$ if and only if $i < j$, or $i = j$ and $p < q$.[2] We then use the standard extension of this ordering to compare literals in clauses.

Before we proceed to the actual presentation of the calculus, we need to define how labels are updated by inferences. Two labelled clauses should only interact with each other when they actually represent standard clauses that interact on the ground level. Moreover, the resulting labelled clause should represent exactly all the possible results of the interactions on the ground level. We define the *merge* of two labels (b_1, k_1) and (b_2, k_2) as the label (b, k) such that

- if $b_1 = b_2 = *$ then $b = *$, otherwise $b = 0$,

[1] Selection of negative literals can also be carried over in a straightforward way, but we leave it out from the current presentation to keep things simple.

[2] In the case of labelled clauses this amounts to saying that the symbols of Σ' are considered larger than those of Σ. Our definition, however, also makes sense over the infinite signature Σ^* and it is this particular ordering that restricts the inferences on the level of standard clauses.

- if $k_1 = *$ then $k = k_2$; if $k_2 = *$ then $k = k_1$; if $k_1 = k_2 \neq *$ then $k = k_1 = k_2$.

In the case when $k_1, k_2 \in \mathbb{N}$ and $k_1 \neq k_2$, the merge operation is *undefined*. The idea that the merged label represents the intersection of the sets of indices represented by the arguments is captured by the following lemma.

Lemma 3. *Let (b, k) be the merge of the labels (b_1, k_1) and (b_2, k_2), and (K, L) any rank. Then*

$$R_{(K,L)}(b, k) = R_{(K,L)}(b_1, k_1) \cap R_{(K,L)}(b_2, k_2) \ .$$

Proof. The proof is straightforward from the definitions. We check by case analysis that $[b_1 \neq * \rightarrow t = 0]$ in conjunction with $[b_2 \neq * \rightarrow t = 0]$ is equivalent to $[b \neq * \rightarrow t = 0]$, and also that $[k_1 \neq * \rightarrow \exists s \in \mathbb{N}.t + k = K + s \cdot L]$ in conjunction with $[k_2 \neq * \rightarrow \exists s \in \mathbb{N}.t + k = K + s \cdot L]$ is equivalent to $[k \neq * \rightarrow \exists s \in \mathbb{N}.t + k = K + s \cdot L]$, under the condition that $k_1 = * \vee k_2 = * \vee k_1 = k_2$. □

The calculus *LPSup* consists of the inference rules Ordered Resolution, Ordered Factoring, Temporal Shift, and Leap. They operate on a clause set N, an initial labelled clause set of a given PLTL-specification. While Ordered Resolution and Ordered Factoring constitute the labelled analogue of inferences of *PSup*, Temporal Shift and Leap are "structural" in nature, as they only modify the syntactic format, but the underlying represented set of standard clauses remains the same.

(i) *Ordered Resolution*

$$\mathcal{I}\frac{(b_1, k_1) \,||\, C \vee L \quad (b_2, k_2) \,||\, D \vee \bar{L}}{(b, k) \,||\, C \vee D}$$

where literal L is maximal in C, its complement $\bar{L}$ is maximal in D, and the merge of labels (b_1, k_1) and (b_2, k_2) is defined and equal to (b, k),

(ii) *Ordered Factoring*

$$\mathcal{I}\frac{(b, k) \,||\, C \vee A \vee A}{(b, k) \,||\, C \vee A}$$

where A is an atom maximal in C,

(iii) *Temporal Shift*

$$\mathcal{I}\frac{(*, k) \,||\, C}{(*, k') \,||\, (C)'}$$

where C is a non-empty clause over Σ only, and $k = k' = *$ or $k \in \mathbb{N}$ and $k' = k + 1$,

(iv) *Leap*

$$\mathcal{I}\frac{\{(b, u + i \cdot v) \,||\, C\}_{i\in\mathbb{N}} \text{ derivable from } N}{(b, u - v) \,||\, C}$$

where $u \geq v > 0$ are integers and C is an arbitrary standard clause.[3]

Further explanation is needed for the inference rule Leap. In its present form it requires an infinite number of premises, one for each $i \in \mathbb{N}$, and thus cannot, strictly speaking, become applicable in any finite derivation (or possibly only "in the limit"). Here it is only a mathematical abstraction. In the next chapter, where we discuss termination of *LPSup*, we show how to effectively generate and finitely represent infinite sets of labelled clauses from which it will follow that Leap is, in fact, effective.

Going back to the other inferences note that the merge operation on labels ensures that the conclusion of Ordered Resolution represents exactly all the conclusions of the standard ordered resolution inferences between the standard clauses represented by the premises. Ordered Factoring carries over from *PSup* in a similar fashion.[4] The Temporal Shift operates only on clauses over the signature Σ. We will from now on call such clauses *simple*. Notice that the restriction to simple clauses is essential as it keeps the symbols of the conclusion to stay within $\Sigma \cup \Sigma'$.

Example 3. *The initial labelled clause set N_S of our running example contains among others also clauses $(*, *) \,||\, a \vee b'$ and $(*, 0) \,||\, \neg b$. We can apply Temporal Shift to the second to obtain $(*, 1) \,||\, \neg b'$. Now b' is the only literal over Σ' in the first clauses and therefore maximal. So the first clause and the newly derived one can participate in Ordered Resolution inference with conclusion $(*, 1) \,||\, a$.*

Although the rules Temporal Shift and Leap derive new labelled clauses, the represented sets of standard clauses remain the same in any rank (K, L). This is easy to see for Temporal Shift, but a little bit more involved for Leap, where it relies on the periodicity of (K, L)-models. The overall soundness of *LPSup* is established by relating it to the same property of the standard calculus *PSup*. By formalizing the above given ideas we obtain the following.

Lemma 4. *Let (K, L) be a rank. Any standard clause represented in (K, L) by the conclusion of the Ordered Resolution inference or the Ordered Factoring inference of LPSup can be derived by the corresponding PSup inference*

[3] In fact, we can restrict the Leap inference to the case where C is a simple clause, $b = *$, and even $s = t$. This, however, makes the completeness proof more involved.

[4] Here we present the rule in a form as close as possible to the one in [2]. In practical implementation, however, it is reasonable to remove duplicate literals as soon as they occur without regard to ordering restrictions.

from some standard clauses represented in (K, L) by the premises of the inference.

Proof. This is a routine verification. First we check that the necessary standard assumptions are available, i.e. represented in (K, L) by the premises of the *LPSup*-inference. This follows from Lemma 3. In the second step, we verify that the ordering restrictions for the *PSup*-inference are satisfied. This follows from how we extended the ordering on Σ to the whole Σ^*, namely from the fact that we have $p < q$ if and only if $p^{(k)} < q^{(k)}$ for any $p, q \in \Sigma$ and $k \in \mathbb{N}$. □

Lemma 5. *Let (K, L) be a rank. Any standard clause represented in (K, L) by the conclusion of the Temporal Shift inference is represented in (K, L) by its premise.*

Proof. Let $(C')^{(t)}$ be a standard clause represented in (K, L) by the conclusion $(*, k') \,||\, (C)'$ of Temporal Shift inference. This means that $t \in R_{(K,L)}(*, k')$. We either have $k = k' = *$, or $k \in \mathbb{N}$ and $k' = k + 1$. In any case $t + 1 \in R_{(K,L)}(*, k)$, and thus $C^{(t+1)} = (C')^{(t)}$ is represented in (K, L) by the premise $(*, k) \,||\, C$ of the inference. □

Lemma 6. *Let (K, L) be a rank. Any standard clause represented in (K, L) by the conclusion of the Leap inference is represented in (K, L) by one of its premises.*

Proof. Let $C^{(t)}$ be a standard clause represented in (K, L) by the conclusion $(b, u - v) \,||\, C$ of Leap inference, i.e. we have $t \in R_{(K,L)}(b, u - v)$. We need to show that $t \in \bigcup_{i \in \mathbb{N}} R_{(K,L)}(b, u + i \cdot v)$ and thus $C^{(t)}$ is represented in (K, L) by one of the inference's premise $(b, u + i \cdot v) \,||\, C$. This, in particular, boils down to showing that when

$$t + (u - v) = K + s_1 \cdot L$$

for some $s_1 \in \mathbb{N}$, we can find $i, s_2 \in \mathbb{N}$ such that

$$t + (u + i \cdot v) = K + s_2 \cdot L \ .$$

A routine inspection confirms that setting $i = L - 1$, and $s_2 = s_1 + v$ does the trick. □

Theorem 2 (Soundness of *LPSup*)**.** *Let N_S be the initial labelled clause set for a PLTL-specification S, and $(b, k) \,||\, C$ a labelled clause derivable from N_S by LPSup. Then for any rank (K, L) and any $t \in R_{(K,L)}(b, k)$ the standard clause $C^{(t)}$ is derivable from $(N_S)_{(K,L)}$ by PSup.*

If an empty labelled clause $(b, k) \,||\, \bot$ is derivable from N_S by LPSup, such that $R_{(K,L)}(b, k) \neq \emptyset$, then S doesn't have a (K, L)-model.

Proof. By induction on the length of the derivation, using Lemma 4, 5, and 6. The corollary then follows from Lemma 2 and the soundness of *PSup*. □

Notice that in *LPSup* the fact that an empty labelled clause $(b, k) \,||\, \bot$ is derived does not necessarily mean that the whole clause set is unsatisfiable. It only rules out those (K, L)-models for which $R_{(K,L)}(b, k)$ is non-empty. This motivates the following definition.

Definition 7. *An empty labelled clause* $(b, k) \,||\, \bot$ *is called* conditional *if* $b = 0$ *and* $k \in \mathbb{N}$*, and* unconditional *otherwise. We say that a set of labelled clauses* N *is* contradictory *if it contains an unconditional empty clause, or* $(0, k) \,||\, \bot$ *is in* N *for every* $k \in \mathbb{N}$*.*

In Chapter 6 we demonstrate that a (K, L)-model can be found for any non-contradictory set of labelled clauses that is saturated by *LPSup*.

To complete the picture of *LPSup* we move on to mention reduction rules. As we discuss in detail in Chapter 6, these are justified by the abstract redundancy notion [2] which our calculus inherits from *PSup*. Thus the following are only examples and other reductions can be developed and used as long as they satisfy the criteria of abstract redundancy.

Tautology Deletion allows us to remove from the search any labelled clause the standard part of which contains both a literal and its complement. Another useful reduction is *Subsumption*[5]

$$\mathcal{R}\frac{(b_1, k_1) \,||\, C \quad (b_2, k_2) \,||\, D}{(b_1, k_1) \,||\, C}$$

where C is a sub-multiset of D and the merge of labels (b_1, k_1) and (b_2, k_2) is defined and equal to (b_2, k_2).

[5]We use the letter $\mathcal{I}$ and $\mathcal{R}$ to distinguish between *inference rules*, whose premises are kept after the conclusion has been added to the given set of clauses, and *reduction rules*, whose premises are replaced by the conclusion.

5 Decision procedure

In this chapter we explain how to turn the calculus *LPSup* into an effective decision procedure for PLTL. First, we have a look at termination.

Example 4. *We have already derived the labelled clause* $(*, 1) \,||\, \neg b'$ *from our set* N_S *of initial clauses for* S *by Temporal Shift. Ordered Resolution between this clause and the clause* $(*, *) \,||\, \neg b \vee b'$ *yields* $(*, 1) \,||\, \neg b$ *to which Temporal Shift is again applicable, giving us* $(*, 2) \,||\, \neg b'$*. We see that the clause we started with differs from the last one only in the label where the* k*-component got increased by one. The whole sequence of inferences can now be repeated, allowing us to eventually derive labelled clauses* $(*, k) \,||\, \neg b$ *and* $(*, k) \,||\, \neg b'$ *for any* $k \in \mathbb{N}^+$*.*

The example demonstrates how the Temporal Shift inference may cause non-termination when the k-component of the generated labelled clauses increases one by one. It also suggests, however, that from a certain point the derived clauses don't add any new information and the inferences essentially repeat in cycles. Detecting these repetitions and finitely representing the resulting infinite clause sets is the key idea for obtaining a termination result for our calculus.

Given a set of labelled clauses N, it is convenient to think of N as being separated into *layers*, sets of clauses with the same value of their labels' second component k. This way we obtain the $*$-layer of clauses with the label of the form $(b, *)$ for $b \in \{*, 0\}$, and similarly layers indexed by $k \in \mathbb{N}$. The following list of observations forms the basis of our strategy for saturating clause sets by *LPSup*.

(1) In an initial labelled clause set only the $*$-layer and 0-layer are non-empty.

(2) If all premises of Ordered Resolution, Factoring or Temporal Shift inference belong to the $*$-layer, so does the conclusion of the respective inference.

(3) If a premise of Ordered Resolution or Factoring inference belongs to the k-layer for $k \in \mathbb{N}$, so does the inference's conclusion.

(4) If a premise of Temporal Shift belongs to the k-layer for $k \in \mathbb{N}$, the inference's conclusion belongs to the layer with index $(k+1)$.

(5) The number of clauses in each layer is bounded by a constant depending only on the size of the signature.

We are ready to describe what we call *layer-by-layer* saturation of an initial labelled clause set. During this process we don't yet consider the Leap inference, which will be incorporated later. It follows from our observations that the $*$-layer can always be finitely saturated. We then perform all the remaining Ordered Resolution and Factoring inferences (together with possible reductions) to saturate the 0-layer, again in a finite number of steps. After that we exhaustively apply the Temporal Shift rule to populate the 1-layer and again saturate this layer by Ordered Resolution and Factoring. This process can be repeated in the described fashion to saturate layers of increasing indices. It is important that the new clauses of the higher layers can never influence (by participating on inferences or reductions) clauses in the lower, already saturated, layers. Eventually, thanks to point (5) above, we will encounter a layer we have seen before and then we stop. More precisely, in a finite number of steps we are bound to obtain a set of labelled clauses N such that there are integers $o \in \mathbb{N}$ and $p \in \mathbb{N}^+$ and

- the o-layer of N is equal to the $(o+p)$-layer of N (up to reindexing[1]),
- the clause set is saturated by *LPSup* (without Leap), except, possibly, for Temporal Shift inferences with premise in layer $(o+p)$,
- the layers with index larger than $(o+p)$ are empty.

Now we need a final observation to finish the argument. The applicability of Ordered Resolution, Factoring and Temporal Shift (as well as that of the reductions of *LPSup*) is "invariant under the move from one layer to another". In other words, exactly the same (up to reindexing) inferences (and reductions) that have been performed to obtain, e.g., the saturated layer of index $(o+1)$, can now be repeated to obtain the saturated layer of index $(o+p+1)$. We can therefore stop the saturation process here and define:

[1]Meaning the first mentioned set would be identical to the second if we changed the second label component of all its clauses from o to $(o+p)$.

Definition 8. *Let N be a clause set obtained by layer-by-layer saturation as described above. We call the numbers o and p the* offset *and* period *of N, respectively. The* infinite extension *of such N is the only set of labelled clauses N^* for which $N \subseteq N^*$ and such that for every $i \in \mathbb{N}$ the $(o+i)$-layer of N^* is equal to the $(o + i \bmod p)$-layer of N (up to reindexing).*

The infinite extension of N is completely saturated by *LPSup* (without Leap).

Example 5. *In our running example, the $*$-layer and 0-layer are already saturated. The next layers we obtain are*

$$\{(*,1) \,||\, \neg a', (*,1) \,||\, \neg b', (*,1) \,||\, a, (*,1) \,||\, \neg b\} \ , \tag{5.1}$$

$$\{(*,2) \,||\, a', (*,2) \,||\, \neg b', (*,2) \,||\, a, (*,2) \,||\, \neg b\} \tag{5.2}$$

As the 3-layer is then equal to the previous (up to reindexing), layer-by-layer saturation terminates with offset 2 and period 1.

In layer-by-layer saturation we always give priority to Ordered Resolution and Factoring inferences, and only when these are no longer applicable in the current clause set, we perform all the pending Temporal Shift inferences, and possibly repeat. Similarly, the *overall saturation procedure* which we present next combines layer-by-layer saturation phases with an exhaustive application of the Leap inference:

1. Set N_1 to the initial labelled clause set N_S of a given PLTL-specification S.
2. Set N_2 to the layer-by-layer saturation on N_1.
3. If the clause set N_2^* is contradictory, stop and report UNSAT.
4. Set N_3 to be the set N_2 enriched by all the possible conclusions of Leap inference with premises in N_2^*, possibly reduced.
5. If $N_3 = N_2$ stop and report SAT, else go back to step 2 resetting $N_1 := N_3$.

Note that if we go to line 2 for the second time, N_1 is no longer an initial labelled clause set. Although we didn't discuss it previously, it is straightforward to perform layer-by-layer saturation of any finitely represented clause set.

On lines 3 and 4 we refer to the infinite extension N_2^*. It actually means that we operate with the layer-by-layer saturation N_2 together with offset o

and period p. Now N_2^* is bound to be contradictory if and only if N_2 contains an unconditional empty clause or $(0, k) \,||\, \bot$ is in N_2 for every $0 \leq k < o + p$. Similarly, a labelled clause $(b, j) \,||\, C$ with $j < o^2$ can be derived by Leap inference with premises in N_2^* if and only if there is a clause $(b, i) \,||\, C$ in N_2 such that $o \leq i < o + p$ and p divides $i - j$.

Finally note that while the values of offset and period associated with N_2 may change from one repetition to another, their sum is each time bounded by the same constant depending only on the size of the signature, namely the number of different possible layers (up to reindexing). Moreover, thanks to the fact that we only work with a fixed finite signature, there is also a bound on the number of non-trivial additions to the individual layers on line 4. These together ensure that the procedure always terminates.

Example 6. *In our example, the infinite extension of the layer-by-layer saturation contains the premises* $\{(*, 1 + i) \,||\, a\}_{i \in \mathbb{N}}$ *of a Leap inference with conclusion* $(*, 0) \,||\, a$*. This clause together with the already present* $(*, 0) \,||\, \neg a$ *gives us the empty clause* $(*, 0) \,||\, \bot$ *by Ordered Resolution, which eventually terminates the overall procedure, because the empty clause is unconditional and therefore the overall set becomes contradictory.*

[2]Leap conclusion with $j \geq o$ is always redundant.

6 Redundancy, completeness and model building

The calculus *LPSup* comes with an abstract notion of redundancy in the spirit of [2]. Also here one can recognize the idea of "lifting", which relates the standard level of *PSup* to the level of labelled clauses. Recall that a standard clause C is called redundant with respect to a set of standard clauses N if there are clauses $C_1, \ldots, C_n \in N$ such that for every $i = 1, \ldots n$, $C_i < C$, and $C_1, \ldots, C_n \models C$. On the level of labelled clause we define:

Definition 9. *A labelled clause* $(b, k) \,||\, C$ *is* redundant *with respect to a set of labelled clauses* N*, if for any rank* (K, L) *every standard clause represented by* $(b, k) \,||\, C$ *in* (K, L) *is redundant w.r.t.* $N_{(K,L)}$*.*

A set of labelled clauses N *is* saturated up to redundancy *with respect to LPSup, if for every inference from* N *such that its premises are not redundant w.r.t.* N*, the conclusion is either redundant w.r.t.* N *or contained in* N*.*

Note that the reductions of *LPSup* described in Chapter 4 are instances of redundancy elimination. This is easy to see for Tautology deletion, and follows from the semantics of the merge operation on labels for the Subsumption reduction:

Lemma 7. *Let* $(b_1, k_1) \,||\, C$ *and* $(b_2, k_2) \,||\, D$ *be the premises the Subsumption reduction as described in Sect. 4, i.e.,* C *is a sub-multiset of* D *and the merge of labels* (b_1, k_1) *and* (b_2, k_2) *is defined and equal to* (b_2, k_2)*. Then* $(b_2, k_2) \,||\, D$ *is redundant w.r.t.* $\{(b_1, k_1) \,||\, C\}$*.*

Proof. Let us fix a rank (K, L). Let $D^{(t)}$ be a standard clause represented in (K, L) by $(b_2, k_2) \,||\, D$, i.e., with $t \in R_{(K,L)}(b_2, k_2)$. Because the merge of labels (b_1, k_1) and (b_2, k_2) is defined and equal to (b_2, k_2) we obtain from Lemma 3 that $R_{(K,L)}(b_2, k_2) = R_{(K,L)}(b_1, k_1) \cap R_{(K,L)}(b_2, k_2)$, in other words $R_{(K,L)}(b_2, k_2) \subseteq R_{(K,L)}(b_1, k_1)$ and therefore $t \in R_{(K,L)}(b_1, k_1)$. Because C is

a (strict[1]) sub-multiset of D, we obtain that $C^{(t)} <^c D^{(t)}$ and that $C^{(t)} \models D^{(t)}$. □

It is important to note that these are just examples and further reductions can be developed and used. As long as they fit into the framework prescribed by Definition 9, they are guaranteed to preserve completeness and the underlying proof need not be changed.

Our main theorem relates completeness of *LPSup* to the same property of the underlying calculus *PSup* via the notion of redundancy.

Theorem 3 (Completeness of *LPSup*)**.** *Let N be a labelled clause set saturated in a layer-by-layer fashion with offset o and period p and let N^*, the infinite extension of N, be a non-contradictory set of labelled clause saturated up to redundancy w.r.t. LPSup. We set K to be the smallest number from $\mathbb{N}$ such that $(0, K) \,||\, \bot$ is not in N^* (note that N^* is non-contradictory), and further set L to the smallest positive multiple of p that is not smaller than o. Then the set $N^*_{(K,L)}$ does not contain the (standard) empty clause and is saturated up to redundancy w.r.t. PSup.*

Proof. We first show that the set $N^*_{(K,L)}$ doesn't contain the empty clause. Because N^* is a non-contradictory set, the only empty labelled clauses it potentially contains are of the form $(0, k) \,||\, \bot$. Moreover, we know that N^* does not contain the empty clause $(0, K) \,||\, \bot$. Assume there is a different empty clause of the form $(0, k) \,||\, \bot$ such that $R_{(K,L)}(0, k)$ is non-empty. This, according to the semantics of labels, implies the equation $0 + k = K + s \cdot L$ has a solution for some $s \in \mathbb{N}$. As k cannot be equal to K, it is necessarily greater than o, and therefore $(0, k) \,||\, \bot$ belongs to the "periodic part" of N^*. Moreover, L has to divide $k - K$. This implies (by the choice of L) there had to be a Leap inference with the conclusion $(0, K) \,||\, \bot$. But that is impossible as N^* is saturated.

We now move to showing that $N^*_{(K,L)}$ is saturated up to redundancy. Let us take an Ordered Resolution inference of *PSup* (Ordered Factoring is similar, only simpler) with premises $C \vee A$ and $D \vee \neg A$ in $N^*_{(K,L)}$ that are non-redundant w.r.t. $N^*_{(K,L)}$. There has to be labelled clauses $(b_1, k_1) \,||\, C_1 \vee A_1$ and $(b_2, k_2) \,||\, D_2 \vee \neg A_2$ in N^* and an integer $t \in R_{(K,L)}(b_1, k_1) \cap R_{(K,L)}(b_2, k_2)$ such that

- $C \vee A$ equals $(C_1 \vee A_1)^{(t)}$, and

[1]Our notion of redundancy of labelled clauses in the presented form does not justify redundancy of "non-strict" subsumption reductions (subsumptions where the standard parts of the two labelled clauses are the same). A technically more involved definition could be used for that purpose. We don't present it here for the sake of simplicity.

- $D \vee \neg A$ equals $(D_2 \vee \neg A_2)^{(t)}$.[2]

We look for an Ordered Resolution inference of *LPSup* to "represent" the above inference on the level of labelled clauses. It is easy to check that we don't need to worry about ordering restrictions, because we assume both calculi are constrained by the same ordering on the extended signature (as defined in Sect. 4). If either k_1 or k_2 equals $*$, or $k_1 = k_2 \in \mathbb{N}$, then the merge operation on the labels is defined, and so the inference on the labelled clauses can be performed.

In the case when $k_1, k_2 \in \mathbb{N}$ and $k_1 \neq k_2$, we necessarily have that L divides $k_1 - k_2$, because otherwise $R_{(K,L)}(b_1, k_1) \cap R_{(K,L)}(b_2, k_2)$ would be empty. We call a labelled clause $(b, k) \,||\, C$ in N^* *periodic* if $k \in \mathbb{N}$ is greater or equal to the offset of N. If both the potential premises $(b_1, k_1) \,||\, C_1 \vee A_1$ and $(b_2, k_2) \,||\, D_2 \vee \neg A_2$ are periodic, then we can find a different representative for, say, $C \vee A$ in the form of $(b_1, k_2) \,||\, C_1 \vee A_1$ (note that L is a multiple of p), such that the merge of the two labels (b_1, k_2) and (b_2, k_2) is now defined and the inference can be performed. If one of the premises, without loss of generality $(b_1, k_1) \,||\, C_1 \vee A_1$, is periodic, and the other, $(b_2, k_2) \,||\, D_2 \vee \neg A_2$, is not, then, again using the fact that L divides $k_1 - k_2$, there must have been a Leap inference with a conclusion $(b_1, k_2) \,||\, C_1 \vee A_1$. Thus, also in this case, the resolution inference can be performed with the Leap's conclusion replacing the periodic premise. Finally, note that it cannot be the case that both $(b_1, k_1) \,||\, C_1 \vee A_1$ and $(b_2, k_2) \,||\, D_2 \vee \neg A_2$ are non-periodic, because in that case we would have $\max(k_1, k_2) < o \leq L$ (by definition of L) and it would not be possible for L to divide $k_1 - k_2$.

Now from Lemma 3 we know that the conclusion $C \vee D$ is represented in (K, L) by the conclusion $(b, k) \,||\, C_1 \vee D_2$ of the resolution inference of the above labelled clauses. If the conclusion is present in N^* we are done. Otherwise it has to be redundant in N^* in which case $C \vee D$ is redundant in $N^*_{(K,L)}$. This concludes the proof. □

Recall the overall saturation procedure of the previous chapter. Its input is a PLTL-specification which is immediately transformed into the initial labelled clause set. If the procedure reports UNSAT, we know the input is unsatisfiable, because we derived (using a sound calculus) a contradictory set of labelled clauses, which rules out any (K, L)-model. If, on the other hand, the procedure reports SAT, we may apply Theorem 3 together with

[2] In the case when one of the premises in $N^*_{(K,L)}$ is over two consecutive signatures $\Sigma^{(t)} \cup \Sigma^{(t+1)}$ and the other premise over "the upper" of these two, i.e., over $\Sigma^{(t+1)}$, we rely here on N^* being saturated (up to redundancy), in particular, by the Temporal Shift inference.

completeness of *PSup* to conclude that the set $N^*_{(K,L)}$ is satisfiable, and, therefore, the specification we started with has a (K, L)-model. Thus the overall saturation procedure decides satisfiability of PLTL-specifications.

We close this chapter by commenting on the possibility of using our method to provide counterexamples to non-valid PLTL formulas. Due to space restrictions, we cannot describe the method in full detail, but to those familiar with the model construction for classical logic based on *PSup* [2], it should be clear that with Theorem 3 proven, we are practically done.

Given a non-contradictory set of labelled clauses N^* that is saturated up to redundancy w.r.t. *LPSup*, we pick (K, L) as described in Theorem 3 and generate the standard clauses of $N^*_{(K,L)}$ one by one with increasing $<$. We apply classical model construction to these clauses to gradually build a (partial) valuation over $\Sigma^* = \bigcup_{i\in\mathbb{N}} \Sigma^{(i)}$, which, as we know, corresponds in the obvious way to a (K, L)-model $(V_i)_{i\in\mathbb{N}}$. We can stop the generation as soon as a particular (already completed) valuation repeats (i.e. $V_i = V_{i-j}$ for some $j \in \mathbb{N}^+$) and the goal has already been reached (i.e. $i > K$). An ultimately periodic model is then output as a result.

7 Related work

7.1 Overview

We now compare our calculus to Clausal Temporal Resolution [13]. Older resolution based approaches to PLTL are [4, 31], but they don't seem to be used or developed any further nowadays. Besides resolution there are approaches to PLTL satisfiability based on tableaux deduction [34, 27], and on automata theory [26]. These seem to be less related and we don't discuss them here further.

It can be shown that operationally there is a close connection between *LPSup* and the Clausal Temporal Resolution (CTR) of [13] (see the next section for the details). From this perspective, our formalism of labelled clauses can be seen as a new way to derive completeness of CTR that justifies the use of ordering restrictions and redundancy elimination in a transparent way. This has not been achieved yet in full by previous work: [18] contains a proof theoretic argument, but only for the use of ordering restrictions, [21] sketches the idea how to justify tautology removal and subsumption, but not the general redundancy notion in the style of [2] that we provide.

Moreover, there is also a correspondence between our layer-by-layer saturation followed by the application of the Leap inference and the BFS-Loop search of CTR as described in [14, 23]. Apart from being interesting in its own right, this view sheds new light on explaining BFS-Loop search, as it gives meaning to the intermediate clauses generated in the process, and we thus don't need to take the detour through the DNF representation of [6, 7]. Even here, the idea of labels clearly separates logical content of the clauses from the meta-logical one (c.f. the ad hoc marker literal of [14]).

Despite these similarities between *LPSup* and CTR, the calculi are by no means identical. As discussed before, a temporal model can be extracted in a straightforward way from a satisfiable set of labelled clauses saturated by *LPSup*. This doesn't hold for CTR, where a more complex approach that simulates the model construction of [2] only locally needs to be applied [24].

In particular, because saturation by CTR doesn't give the model building procedure any guidance as to where to look for the goal, in each considered world all the possible orderings on the signature (in the worse case) need to be tried out in a fair way to make sure a goal world is eventually reached. As each change of the ordering calls for a subsequent resaturation of the clause set in question (so that the local model construction still works), it obviously diminishes the positive effect orderings in general have on reducing the search space.

Finally note that since we eventually rely on propositional superposition, we can also take into account the explicit use of partial models to further guide the search for a proof or saturation. The idea is to build a partial model based on the ordering on propositional literals. Then it can be shown that resolution can be restricted to premises where one is false and the other true in the partial model [1]. This superposition approach on propositional clauses is closely related to the state of the art CDCL calculus (see, e.g. [35]) for propositional logic. The missing bit is to "lift" this setting to our labelled clauses. This will be one direction for future research.

7.2 Details

The CTR calculus is defined over *LTL-clauses* of the Separated Normal Form (SNF), a modification of which we also adopt (see Sect. 2). CTR [13] distinguishes three kinds of LTL-clauses, *initial*, *step*, and *sometime* clauses. While the correspondence of the first two to our labelled counterparts is obvious (see Table 7.1), the relation of sometime clauses to our goal clauses needs further explanation. In CTR the input SNF formula may contain several sometime clauses of the form

$$\Box\left(\bigwedge_{b\in B} k_b \Rightarrow \Diamond l\right) \quad . \tag{7.1}$$

LPSup uses the ideas of [5] to obtain a problem with only one eventuality with no side condition (as if $B = \emptyset$ in (7.1)), but, on the other hand, relaxes the requirement that the eventuality be represented by a single literal. Instead, the eventuality is described by the whole set of the goal clauses of the problem, understood conjunctively:

$$\Box\Diamond\left(\bigwedge_{(*,0)\,||\,C} C\right) \quad . \tag{7.2}$$

Table 7.1: Clause alignment between *LPSup* and CTR (k_a, l_c, and l_d denote literals; $\overline{k_a}$ stands for the complement of literal k_a)

Name	CTR	*LPSup*
Initial	$\mathbf{start} \Rightarrow \bigvee_{c \in C} l_c$	$(0, *) \,\|\|\, \bigvee_{c \in C} l_c$
Step	$\bigwedge_{a \in A} k_a \Rightarrow \bigcirc \bigvee_{d \in D} l_d$	$(*, *) \,\|\|\, \bigvee_{a \in A} \overline{k_a} \vee \bigvee_{d \in D} l'_d$

Note that the techniques of [5] allow us to obtain a formulation of a problem that contains only single unconditional eventuality in a form of a single literal. That is the "intersection" format directly accessible to both *LPSup* and CTR.

Having established the correspondence between the clauses we move on to the inference rules. It is not difficult to see that the two *step resolution* rules of CTR as described in [13], are simulated by our resolution inference on initial and step (labelled) clauses. Similarly, the "upper" half of the following clause conversion rule

$$\mathcal{R} \frac{\phi \Rightarrow \bigcirc \mathbf{false}}{\begin{array}{c}\mathbf{true} \Rightarrow \bigcirc \neg\phi \\ \mathbf{start} \Rightarrow \neg\phi\end{array}} \tag{7.3}$$

can be matched by our Temporal Shift inference. The other half of the rule, which turns the step clause with unsatisfiable succedent into an initial clause is not needed in *LPSup*, where the assumption is kept instead and allowed to interact with initial clauses directly.

What remains to be compared is the role played by the goal clauses and the Leap rule of *LPSup* on the one hand, and the temporal resolution inference of CTR on the other. As presented in the original paper [13], the temporal resolution inference combines several step clauses into groups (so called merged-SNF clauses) and resolves those against one sometime clause. A nontrivial side condition, which needs to be verified, amounts to proving that the step clauses involved form a so called *loop*, meaning that they together conditionally imply that the eventuality will become false forever. As a last step, the inference's conclusion which is not a temporal clause in general must be translated into SNF after it is computed.

Several methods have been proposed on how to actually implement temporal resolution [6]. Here we focus on breadth first search for the loop as described in [14]. The idea is to perform the loop search by iteratively applying step resolution inferences to certain clauses and to organize the individual iterations by enriching the participating clauses with a special marking literal (see also [23]). The marking literal, which is numbered by the iteration index, separates the clauses from each iteration, and allows for their reuse in subsequent loop searches (for the same eventuality literal). Now, it can be

seen, that in our case a similar role is played by the label of the form $(*, k)$ of clauses associated to the goal, where k should be identified with the iteration index of the marker. The difference lies, however, in the way clauses with new index k get created. In *LPSup* they arise as conclusions of Temporal Shift inference

$$\mathcal{I}\frac{(*, k) \,||\, C}{(*, k+1) \,||\, C'} \tag{7.4}$$

where the premise is a simple clause, i.e. only over the basic signature Σ. In CTR the corresponding inference (with the same side condition) in our notation becomes

$$\mathcal{I}\frac{(*, k) \,||\, C}{(*, k+1) \,||\, C' \vee L'} \tag{7.5}$$

where L is the respective eventuality literal. The addition of the extra literal has interesting semantic consequence which we discuss shortly. Now we focus on one final difference of the two approaches. When we in *LPSup* detect repetition in the layers of generated goal clauses, we invoke the Leap inference, and potentially derive additional *goal clauses*, or more precisely labelled clauses with the second label component $k \neq *$. In CTR, a successful repetition check[1] concludes the current loop search and the new clauses collected by an equivalent of Leap get the status of *simple step clauses* (universal clauses in the terminology of [23]). This last distinction is also related to the fact that in CTR the initial clauses do not participate on inferences with (the equivalent of our) goal clauses, so there is no equivalent of clauses with label $(0, k)$.

For a semantic comparison, we need to recall the notion of *behaviour graph* as described in [5], adapted to our setting. Given a set of labelled clauses N, the nodes of the graph are all the propositional valuations over Σ, i.e. worlds, and there is an edge between two worlds V_1 and V_2, if $[V_1, V_2] \models C$ for every labelled clause $(*, *) \,||\, C \in N$. Moreover, the worlds satisfying all the initial clauses are marked as initial, and the worlds satisfying all the goal clauses are marked as goal worlds. An (unconstrained) model for a set of labelled clauses can be obtained as an infinite path through the behaviour graph that starts form an initial world and passes through a goal world infinitely often.

If we now look at layer-by-layer saturation on one side and on the loop search on the other from the semantic perspective we notice the following:[2] In *LPSup* the set of labelled clauses of the form $(*, k) \,||\, C$ for one particular

[1] It can be shown that under some reasonable conditions imposed on the saturation procedure, which in particular regulate how reductions may be used, the period detected in CTR is always equal to 1.

[2] Proofs of the following here informally stated observations can be derived from the completeness of ordered resolution [2].

k can be thought as representing the subset of nodes of the behaviour graph that can reach a goal node in *exactly* k steps. In CTR, on the other hand, the equivalent of such layer represents the nodes that can reach a goal node in *at most* k steps. This difference is the promised consequence of the eventuality literal being added in (7.5), which, intuitively, in effect "reinserts" the goal worlds into each newly generated layer. As the sequence of worlds represented by these layers in CTR grows monotonically with each iteration, there is a better theoretical bound on the number of iterations before a repetition occurs than the one that can be established for *LPSup*[3]. On the other hand, as we demonstrate by our experiment, inserting the eventuality literal into the clauses has negative effect on the performance of CTR in practice, because it means that typically much more inferences need to be performed before the computation proceeds from one layer to the next.

[3]It is size of the set of all possible worlds, i.e. $2^{|\Sigma|}$, that bounds the number of different layers in CTR, whereas for *LPSup* our current best bound derives from the number of possible subsets of all the worlds, which is exponentially more.

8 Experiments

We implemented a simple prototype of both *LPSup* and CTR (with BFS loop-search in the style of [14]), in order to compare the two calculi on non-trivial examples. In this chapter we report on our experiment. The prototype, written in SWI-Prolog, is available along with the test examples at [29].

The core of the implementation of our prototype, shared by both calculi, is a saturation loop driven by Ordered Resolution inferences of labelled clauses, employing forward and backward subsumption and tautology deletion. The list of passive clauses is ordered in such a way that saturation of initial and step clauses is performed before the start of the layer-by-layer saturation of the goal clauses. The individual layers are being checked for repetition which, once occurring, may trigger a Leap inference (which corresponds to Temporal Resolution inference in the case of CTR), or allow us to conclude satisfiability, if the Leap addition is trivial.

The input format for the procedure accommodates the restrictions imposed by both calculi. This, in particular, means that an input specification may only contain one unconditional eventuality (because of *LPSup*) given in the form of a single goal literal (because of CTR). Such normal form can be achieved by known techniques [5].

There are three places in the implementation where the calculi differ from each other.

1. CTR doesn't have $(0, k)$-clauses. Modified version of the merge operation on labels blocks for CTR inferences between initial clauses and the clauses derived from the goal.

2. For CTR we modify the Temporal Shift (Clause Conversion) inference of $(*, k)$-clauses, such that the conclusion is additionally extended by the (unique) goal literal.

3. Finally, the conclusion of the Leap inference takes the form of goal[1] clauses in the case of *LPSup* and the form of simple step clauses for CTR. This is just a matter of assigning different labels.

For the experiment we took two formula families $\mathcal{C}_n^1$ and $\mathcal{C}_n^2$ from [19]. Both of them consist of a certain pattern of temporal clauses together with a set of essentially standard clauses that encode a random k-SAT problem. Because the tested calculi treat initial and step clauses identically, we decided to drop the random part and only compare how effectively they deal with the temporal aspects of the formulas. Thus we obtained the following two families of temporal formulas:

$$\mathcal{C}_n^1 \equiv \Box(\neg p_1 \vee \Diamond p_2) \wedge \Box(\neg p_2 \vee \Diamond p_3) \wedge \cdots \wedge \Box(\neg p_n \vee \Diamond p_1),$$

$$\begin{aligned}\mathcal{C}_n^2 \equiv\ & r_1 \wedge (\neg r_1 \vee q_1) \wedge (\neg r_1 \vee \neg q_n) \wedge \\ & \Box(\neg r_n \vee \bigcirc r_1) \wedge \Box(\neg r_{n-1} \vee \bigcirc r_n) \wedge \cdots \wedge \Box(\neg r_1 \vee \bigcirc r_2) \wedge \\ & \Box(\neg r_n \vee \bigcirc \neg q_n) \wedge \Box(\neg r_{n-1} \vee \bigcirc \neg q_n) \wedge \cdots \wedge \Box(\neg r_1 \vee \bigcirc \neg q_n) \wedge \\ & \Box(\neg q_1 \vee \Diamond s_2) \wedge \Box(\neg s_2 \vee q_2 \vee \bigcirc q_n \vee \cdots \vee \bigcirc q_3) \wedge \\ & \vdots \\ & \Box(\neg q_{n-1} \vee \Diamond s_n) \wedge \Box(\neg s_n \vee q_n)\end{aligned}$$

Note that while $\mathcal{C}_n^1$ are trivially satisfiable, $\mathcal{C}_n^2$ is unsatisfiable.

In addition to $\mathcal{C}_n^1$ and $\mathcal{C}_n^2$, we also tested the calculi on formulas from two families specifically constructed to highlight the respective weaknesses of *LPSup* and CTR. They are both based on the idea of putting together several independent "cycles", and are both parameterized by lists of integers, these cycles' lengths. The cycles, however, play different conceptual roles in each family, and the resulting problems are in fact very different.

We define the implicit cycles problem $\mathcal{I}_{(l_1+\cdots+l_k)}$ when the sum of the cycles' lengths is equal to a power of two, i.e. $l_1 + \cdots + l_k = 2^n$. We then build it over a signature Σ of size n. The clauses of $\mathcal{I}_{(l_1+\cdots+l_k)}$ are selected in such way, that the behaviour graph of the problem, which is necessarily a graph over 2^n vertices, consists exactly of k independent (oriented) cycles of lengths $l_1, \ldots, l_k$, respectively. Moreover, on each cycle there is exactly one world which is a goal world. Finally, the set of initial clauses is left empty and thus every world of the graph is an initial one. The number of clauses

[1] It can be shown that we can restrict the Leap inferences to the case where the conclusion clause C is simple with label $(*, 0)$. That is what we do in our implementation.

needed to achieve[2] this is polynomially bounded in 2^n, the number of vertices of the graph.

While the implicit cycles are semantic in nature, the explicit cycles problem $\mathcal{E}_{(l_1+\cdots+l_k)}$ is purely syntactic. For the i-th of the k cycles, there are variables $p^i_1, \ldots, p^i_{l_i}$ and $(*,*)$-clauses with standard parts $\neg p^i_1 \vee (p^i_2)', \neg p^i_2 \vee (p^i_3)', \ldots, \neg p^i_{l_i} \vee (p^i_1)'$. Moreover, for each such cycle there is also a $(*,*)$-clause $\neg(p^i_1)' \vee \neg(g)'$, and finally, there is one goal clause of the form $(*, 0) \,||\, g$. Thus the size of the signature for $\mathcal{E}_{(l_1+\cdots+l_k)}$, is $1 + l_1 + \cdots + l_k$, and the problems consists of $1 + k + l_1 + \cdots + l_k$ labelled clauses. Note that both the implicit and explicit cycles problems are satisfiable.

In the presented experiments with the prototype, we chose a variable ordering for restricting resolution inferences which gave good results for both calculi on the tested examples. In particular, we optimized the order of groups of variables, where one group was formed by all variables coming directly from the input formula, and several others contained different kinds of auxiliary variables introduced during the translation of multiple eventualities into single unconditional one [5].

[2]Our encoding works as follows. Recall that the vertices of the behaviour graph are in fact valuations over Σ. For any pair of valuations (V_1, V_2) which is *not* intended to be an edge of the graph, the problem contains the *unique* clause C (labelled $(*,*)$) over $\Sigma \cup \Sigma'$ such that $[V_1, V_2]$ is the only valuation over the clause's signature that makes it false. The encoding of the goal worlds works similarly.

Table 8.1: Results of comparing our implementations of *LPSup* and CTR with TRP++

		LPSup-Prolog			CTR-Prolog			TRP++	
Problem	Size	Cl-gen	Lits-gen	Cl-subs	Cl-gen	Lits-gen	Cl-subs	Cl-gen	Cl-subs
$\mathcal{C}^1_{10}$	56	53	202	100	174	576	110	363	300
$\mathcal{C}^1_{15}$	81	78	377	145	334	1161	240	688	595
$\mathcal{C}^1_{20}$	106	103	602	190	544	1946	420	1113	990
$\mathcal{C}^2_3$	22	442	1376	324	984	3972	909	1146	968
$\mathcal{C}^2_4$	30	1937	7649	1612	5298	26086	5047	3560	3053
$\mathcal{C}^2_5$	38	6287	28576	5635	18724	102704	18134	7925	6922
$\mathcal{I}_{(3+5)}$	62	406	1563	368	203	1022	194	86	160
$\mathcal{I}_{(3+5+8)}$	253	8010	42024	7356	1087	7613	1145	390	745
$\mathcal{E}_{(2+3)}$	8	23	25	4	131	424	78	177	77
$\mathcal{E}_{(2+3+4)}$	13	52	55	6	1061	4490	595	1597	627

Table 8 summarizes the results of our experiments. For each problem and for both calculi we report the number of clauses in the input, the number of derived[3] clauses and literals, and the number of subsumed clauses. For comparison, we also include in the last two columns clause data obtained by running the temporal prover TRP++ [17][4], which also implements the CTR calculus, to provide evidence that our experimental results are not biased. We used the default mode (no extra options) for running TRP++ and collected the information on the number of generated and subsumed clauses from the output the prover by default provides. We decided not to report on running times as our aim here is to compare the calculi rather than the implementations. The number of generated clauses (literals) should provide a good measure on the amount of data to be processed by any prover, which is, moreover, independent on the choice programming language or the use of particular data structures.

As we can see, *LPSup* needs to generate consistently less clauses to draw its conclusion for both $\mathcal{C}_n^1$ and $\mathcal{C}_n^2$. The implicit cycles examples $\mathcal{I}$, which are the only ones where *LPSup* needs to generate more clauses than CTR, are constructed in such a way that the number of non-repeating layers for *LPSup* is equal to the least common multiple of the cycle sizes (i.e. 15, and 120 for our two instances, respectively), while the number of non-repeating layers for CTR depends on the size of the largest cycle only (the detected offsets for CTR are 6 and 9, respectively).

Although in explicit cycles examples $\mathcal{E}$, the period detected by *LPSup* again grows as the least common multiple of the cycle sizes, the clauses "belonging" to the individual cycles don't interfere with each other, as the reader can easily see from manual inspection (or see Sect. 8.1). On the other hand, CTR suffers here from the same bound on the number of layers to be processed (here manifested as a high value of offset). Moreover, the intermediate layers are formed by clauses mixing literals from different cycles, and the whole layer-by-layer saturation eventually converges only due to factoring and subsumption.

LPSup seems to come considerably better off out of our experiments. Although the inferior performance of CTR on $\mathcal{C}_n^1$ and $\mathcal{C}_n^2$ could possibly be stemming from the translation to single eventuality formulation of the problems (while efficient heuristics may perhaps be devised to treat the individual eventualities separately), the other two families $\mathcal{I}$ and $\mathcal{E}$ contain single eventuality from the outset. Only further tests on examples from practice may

[3]This covers all the resolvents, plus the clauses derived by non-trivial Leap inference. (Leap conclusions subsumed by other clauses are not generated at all.)

[4]We used version 2.1 available at `http://www.csc.liv.ac.uk/~konev/software/trp++/`.

reveal which of the two phenomena exemplified by families $\mathcal{I}$ and $\mathcal{E}$, respectively, have higher impact on practical utility.

8.1 Example run of *LPSup* and CTR on $\mathcal{E}_{(2+3)}$

Here we compare the behaviour of our calculus *LPSup* to that of CTR on the example $\mathcal{E}_{(2+3)}$ as described in the previous section. To make the example easier to read we use the symbols a, b, and c, d, e, respectively, for the variables of the two cycles; the variable g plays the role of the goal (the unique eventuality literal). The whole problem thus consists of the following labelled clauses:

$$
\begin{array}{lll}
(*,*) \,||\, \neg a \vee b', & (*,*) \,||\, \neg c \vee d', & (*,0) \,||\, g, \\
(*,*) \,||\, \neg b \vee a', & (*,*) \,||\, \neg d \vee e', & \\
 & (*,*) \,||\, \neg e \vee c', & \\
(*,*) \,||\, \neg a' \vee \neg g', & (*,*) \,||\, \neg c' \vee \neg g'. &
\end{array}
$$

If we assume that g is the largest symbol in the ordering used, no resolution inference among the step clauses is possible. Table 8.2 shows clauses derived by the respective calculi during layer-by-layer saturation process. Not all the clauses are displayed. We only show the simple $(*, k)$-clauses, i.e. exactly those that will become premises of the Temporal Shift inference, the conclusion of which will "enter" the next layer. In the case of CTR this inference will not only shift the literals' symbols in time, but also enrich the clause by the eventuality literal. What is not shown here are the intermediate $(*, k)$-clauses over the mixed signature $\Sigma \cup \Sigma'$.

For each layer, there are two columns displayed in the case of CTR. The first column shows the simple $(*, k)$-clauses in their "raw" form, before factoring is applied, the literals are reordered (to make the process easier to follow here), and before some of them are removed due to subsumption. The resulting reduced clause set is displayed in the other column.

In layer 7 repetition occurs for both calculi. We see that in the case of *LPSup*, layer 7 is equal to layer 1, for CTR it is equal to layer 6. It should be obvious that in the case of CTR, there is much more work to be done, before the problem can be announced satisfiable. We believe that the phenomenon exemplified here by $\mathcal{E}_{(2+3)}$ is the reason why *LPSup* seems to more efficient that CTR in practice.

Table 8.2: Simple $(*, k)$-clauses generated by *LPSup* and CTR on $\mathcal{E}_{(2+3)}$.

Layer index k	*LPSup*	CTR generated	CTR reduced
0	g	g	g
1	$\neg b$ $\neg e$	$\neg b$ $\neg e$	$\neg b$ $\neg e$
2	$\neg a$ $\neg d$	$\neg a \vee \neg b$ $\neg a \vee \neg e$ $\neg d \vee \neg b$ $\neg d \vee \neg e$	$\neg a \vee \neg b$ $\neg a \vee \neg e$ $\neg b \vee \neg d$ $\neg d \vee \neg e$
3	$\neg b$ $\neg c$	$\neg b \vee \neg a \vee \neg b$ $\neg b \vee \neg a \vee \neg e$ $\neg b \vee \neg d \vee \neg b$ $\neg b \vee \neg d \vee \neg e$ $\neg a \vee \neg c \vee \neg b$ $\neg a \vee \neg c \vee \neg e$ $\neg c \vee \neg d \vee \neg b$ $\neg c \vee \neg d \vee \neg e$	$\neg a \vee \neg b$ $\neg b \vee \neg d$ $\neg a \vee \neg c \vee \neg e$ $\neg c \vee \neg d \vee \neg e$
4	$\neg a$ $\neg e$	$\neg b \vee \neg a \vee \neg b$ $\neg b \vee \neg a \vee \neg e$ $\neg a \vee \neg c \vee \neg b$ $\neg a \vee \neg c \vee \neg e$ $\neg b \vee \neg e \vee \neg d \vee b$ $\neg b \vee \neg e \vee \neg d \vee e$ $\neg e \vee \neg c \vee \neg d \vee b$ $\neg e \vee \neg c \vee \neg d \vee e$	$\neg a \vee \neg b$ $\neg a \vee \neg c \vee \neg e$ $\neg b \vee \neg d \vee \neg e$ $\neg c \vee \neg d \vee \neg e$
5	$\neg b$ $\neg d$	$\neg b \vee \neg a \vee \neg b$ $\neg b \vee \neg a \vee \neg e$ $\neg b \vee \neg e \vee \neg d \vee \neg b$ $\neg b \vee \neg e \vee \neg d \vee \neg e$ $\neg a \vee \neg c \vee \neg d \vee \neg b$ $\neg a \vee \neg c \vee \neg d \vee \neg e$ $\neg e \vee \neg c \vee \neg d \vee \neg b$ $\neg e \vee \neg c \vee \neg d \vee \neg e$	$\neg a \vee \neg b$ $\neg b \vee \neg d \vee \neg e$ $\neg c \vee \neg d \vee \neg e$
6	$\neg a$ $\neg c$	$\neg b \vee \neg a \vee \neg b$ $\neg b \vee \neg a \vee \neg e$ $\neg a \vee \neg c \vee \neg d \vee \neg b$ $\neg a \vee \neg c \vee \neg d \vee \neg e$ $\neg e \vee \neg c \vee \neg d \vee \neg b$ $\neg e \vee \neg c \vee \neg d \vee \neg e$	$\neg a \vee \neg b$ $\neg c \vee \neg d \vee \neg e$
7	$\neg b$ $\neg e$	$\neg b \vee \neg a \vee \neg b$ $\neg b \vee \neg a \vee \neg e$ $\neg e \vee \neg c \vee \neg d \vee \neg b$ $\neg e \vee \neg c \vee \neg d \vee \neg e$	$\neg a \vee \neg b$ $\neg c \vee \neg d \vee \neg e$

9 Conclusion

We applied the ideas of labelled superposition to develop a new decision procedure for propositional linear temporal logic. On the presentation level, it replaces the complex temporal resolution rule from the previously proposed calculus by a simple check for repetition in the derived clause set and a subsequent inference. Its unique treatment of goal clauses enables straightforward partial model building of satisfiable clause sets which could potentially be used to further restrict inferences. Moreover, the experimental comparison to previous work suggests that the new calculus typically explores smaller search spaces to derive its conclusion. Development of an optimized implementation, to be tested on a set of representative benchmarks, will be part of our future work.

Bibliography

[1] L. Bachmair and H. Ganzinger. On restrictions of ordered paramodulation with simplification. In *CADE-10*, volume 449 of *LNCS*, pages 427–441. Springer, 1990.

[2] L. Bachmair and H. Ganzinger. Resolution theorem proving. In J. A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, pages 19–99. Elsevier and MIT Press, 2001.

[3] A. Biere, A. Cimatti, E. M. Clarke, M. Fujita, and Y. Zhu. Symbolic model checking using SAT procedures instead of BDDs. In *Proceedings of the 36th annual ACM/IEEE Design Automation Conference*, pages 317–320, New York, NY, USA, 1999. ACM.

[4] A. Cavalli and L. del Cerro. A decision method for linear temporal logic. In *CADE-7*, volume 170 of *LNCS*, pages 113–127. Springer, 1984.

[5] A. Degtyarev, M. Fisher, and B. Konev. A simplified clausal resolution procedure for propositional linear-time temporal logic. In *TABLEAUX '02*, volume 2381 of *LNCS*, pages 85–99. Springer, 2002.

[6] C. Dixon. Search strategies for resolution in temporal logics. In *CADE-13*, volume 1104 of *LNCS*, pages 673–687. Springer, 1996.

[7] C. Dixon. Temporal resolution using a breadth-first search. *Annals of Mathematics and Artificial Intelligence*, 22:87–115, January 1998.

[8] C. Dixon. Temporal resolution using a breadth-first search algorithm. *Annals of Mathematics and Artificial Intelligence*, 22:87–115, 1998.

[9] C. Dixon. Using Otter for temporal resolution. In *In Advances in Temporal Logic*, pages 149–166. Kluwer, 2000.

[10] C. Dixon, M. Fisher, and H. Barringer. A graph-based approach to resolution in temporal logic. In D. Gabbay and H. Ohlbach, editors, *Temporal Logic*, volume 827 of *LNCS*, pages 415–429. Springer Berlin / Heidelberg, 1994.

[11] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of theoretical computer science (vol. B)*, pages 995–1072. MIT Press, Cambridge, MA, USA, 1990.

[12] M. Fisher. A resolution method for temporal logic. In *Proceedings of the 12th international joint conference on Artificial intelligence - Volume 1*, pages 99–104. Morgan Kaufmann Publishers Inc., 1991.

[13] M. Fisher, C. Dixon, and M. Peim. Clausal temporal resolution. *ACM Trans. Comput. Logic*, 2:12–56, January 2001.

[14] M. C. F. Gago, M. Fisher, and C. Dixon. Algorithms for guiding clausal temporal resolution. In *KI '02*, volume 2479 of *LNCS*, pages 235–252. Springer, 2002.

[15] V. Goranko, A. Kyrilov, and D. Shkatov. Tableau tool for testing satisfiability in LTL: Implementation and experimental analysis. *Electron. Notes Theor. Comput. Sci.*, 262:113–125, May 2010.

[16] U. Hustadt and B. Konev. TRP++: A temporal resolution prover. In *Collegium Logicum*, pages 65–79. Kurt Gödel Society, 2002.

[17] U. Hustadt and B. Konev. Trp++ 2.0: A temporal resolution prover. In *CADE-19*, volume 2741 of *LNCS*, pages 274–278. Springer, 2003.

[18] U. Hustadt, B. Konev, and R. Schmidt. Deciding monodic fragments by temporal resolution. In *CADE-20*, volume 3632 of *LNCS*, pages 738–738. Springer, 2005.

[19] U. Hustadt and R. Schmidt. Scientific benchmarking with temporal logic decision procedures. In *KR '02*, pages 533–546. Morgan Kaufmann, 2002.

[20] H. Kautz and B. Selman. Planning as satisfiability. In N. B., editor, *Proceedings of the 10th European conference on Artificial intelligence*, pages 359–363, New York, NY, USA, 1992. John Wiley & Sons, Inc.

[21] B. Konev, A. Degtyarev, C. Dixon, M. Fisher, and U. Hustadt. Mechanising first-order temporal resolution. *Inf. Comput.*, 199:55–86, May 2005.

[22] T. Lev-Ami, C. Weidenbach, T. Reps, and M. Sagiv. Labelled clauses. In *CADE-21*, pages 311–327. Springer, 2007.

[23] M. Ludwig and U. Hustadt. Fair derivations in monodic temporal reasoning. In *CADE-22*, volume 5663 of *LNCS*, pages 261–276. Springer, 2009.

[24] M. Ludwig and U. Hustadt. Resolution-based model construction for PLTL. In *TIME '09*, pages 73–80. IEEE Computer Society, 2009.

[25] A. Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science*, pages 46–57. IEEE, 1977.

[26] K. Y. Rozier and M. Y. Vardi. LTL satisfiability checking. In *14th International SPIN Workshop*, volume 4595 of *LNCS*, pages 149–167. Springer, 2007.

[27] S. Schwendimann. A new one-pass tableau calculus for PLTL. In *TABLEAUX '98*, volume 1397 of *LNCS*, pages 277–291. Springer, 1998.

[28] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *J. ACM*, 32:733–749, July 1985.

[29] M. Suda and C. Weidenbach. Prototype implementation of LPSup. Website, 2011. `http://www.mpi-inf.mpg.de/~suda/supLTL.html`.

[30] M. Suda and C. Weidenbach. Labelled Superposition for PLTL. Research Report MPI-I-2012-RG1-001, Max-Planck-Institut für Informatik, Saarbrücken, 2012.

[31] G. Venkatesh. A decision method for temporal logic based on resolution. In *FSTTCS*, pages 272–289. Springer, 1985.

[32] C. Weidenbach. Combining superposition, sorts and splitting. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume 2, chapter 27, pages 1965–2012. Elsevier, 2001.

[33] P. Wolper. Temporal logic can be more expressive. In *Foundations of Computer Science, 1981. SFCS '81. 22nd Annual Symposium on*, pages 340 –348, 1981.

[34] P. Wolper. The tableau method for temporal logic: An overview. *Logique et Analyse*, 28:119–136, 1985.

[35] L. Zhang, C. F. Madigan, M. W. Moskewicz, and S. Malik. Efficient conflict driven learning in boolean satisfiability solver. In *ICCAD*, pages 279–285, 2001.

Below you find a list of the most recent research reports of the Max-Planck-Institut für Informatik. Most of them are accessible via WWW using the URL `http://www.mpi-inf.mpg.de/reports`. Paper copies (which are not necessarily free of charge) can be ordered either by regular mail or by e-mail at the address below.

Max-Planck-Institut für Informatik
– Library and Publications –
Campus E 1 4

D-66123 Saarbrücken

E-mail: `library@mpi-inf.mpg.de`

MPI-I-2009-RG1-002	P. Wischnewski, C. Weidenbach	Contextual rewriting
MPI-I-2009-5-006	S. Bedathur, K. Berberich, J. Dittrich, N. Mamoulis, G. Weikum	Scalable phrase mining for ad-hoc text analytics
MPI-I-2009-5-004	N. Preda, F.M. Suchanek, G. Kasneci, T. Neumann, G. Weikum	Coupling knowledge bases and web services for active knowledge
MPI-I-2009-5-003	T. Neumann, G. Weikum	The RDF-3X engine for scalable management of RDF data
MPI-I-2008-RG1-001	A. Fietzke, C. Weidenbach	Labelled splitting
MPI-I-2008-5-004	F. Suchanek, M. Sozio, G. Weikum	SOFI: a self-organizing framework for information extraction
MPI-I-2008-5-003	F.M. Suchanek, G. de Melo, A. Pease	Integrating Yago into the suggested upper merged ontology
MPI-I-2008-5-002	T. Neumann, G. Moerkotte	Single phase construction of optimal DAG-structured QEPs
MPI-I-2008-5-001	F. Suchanek, G. Kasneci, M. Ramanath, M. Sozio, G. Weikum	STAR: Steiner tree approximation in relationship-graphs
MPI-I-2008-4-003	T. Schultz, H. Theisel, H. Seidel	Crease surfaces: from theory to extraction and application to diffusion tensor MRI
MPI-I-2008-4-002	W. Saleem, D. Wang, A. Belyaev, H. Seidel	Estimating complexity of 3D shapes using view similarity
MPI-I-2008-1-001	D. Ajwani, I. Malinger, U. Meyer, S. Toledo	Characterizing the performance of Flash memory storage devices and its impact on algorithm design
MPI-I-2007-RG1-002	T. Hillenbrand, C. Weidenbach	Superposition for finite domains
MPI-I-2007-5-003	F.M. Suchanek, G. Kasneci, G. Weikum	Yago : a large ontology from Wikipedia and WordNet
MPI-I-2007-5-002	K. Berberich, S. Bedathur, T. Neumann, G. Weikum	A time machine for text search
MPI-I-2007-5-001	G. Kasneci, F.M. Suchanek, G. Ifrim, M. Ramanath, G. Weikum	NAGA: searching and ranking knowledge
MPI-I-2007-4-008	J. Gall, T. Brox, B. Rosenhahn, H. Seidel	Global stochastic optimization for robust and accurate human motion capture
MPI-I-2007-4-007	R. Herzog, V. Havran, K. Myszkowski, H. Seidel	Global illumination using photon ray splatting
MPI-I-2007-4-006	C. Dyken, G. Ziegler, C. Theobalt, H. Seidel	GPU marching cubes on shader model 3.0 and 4.0
MPI-I-2007-4-005	T. Schultz, J. Weickert, H. Seidel	A higher-order structure tensor
MPI-I-2007-4-004	C. Stoll, E. de Aguiar, C. Theobalt, H. Seidel	A volumetric approach to interactive shape editing
MPI-I-2007-4-003	R. Bargmann, V. Blanz, H. Seidel	A nonlinear viseme model for triphone-based speech synthesis
MPI-I-2007-4-002	T. Langer, H. Seidel	Construction of smooth maps with mean value coordinates
MPI-I-2007-4-001	J. Gall, B. Rosenhahn, H. Seidel	Clustered stochastic optimization for object recognition and pose estimation

MPI-I-2007-2-001	A. Podelski, S. Wagner	A method and a tool for automatic veriication of region stability for hybrid systems
MPI-I-2007-1-003	A. Gidenstam, M. Papatriantafilou	LFthreads: a lock-free thread library
MPI-I-2007-1-002	E. Althaus, S. Canzar	A Lagrangian relaxation approach for the multiple sequence alignment problem
MPI-I-2007-1-001	E. Berberich, L. Kettner	Linear-time reordering in a sweep-line algorithm for algebraic curves intersecting in a common point
MPI-I-2006-5-006	G. Kasnec, F.M. Suchanek, G. Weikum	Yago - a core of semantic knowledge
MPI-I-2006-5-005	R. Angelova, S. Siersdorfer	A neighborhood-based approach for clustering of linked document collections
MPI-I-2006-5-004	F. Suchanek, G. Ifrim, G. Weikum	Combining linguistic and statistical analysis to extract relations from web documents
MPI-I-2006-5-003	V. Scholz, M. Magnor	Garment texture editing in monocular video sequences based on color-coded printing patterns
MPI-I-2006-5-002	H. Bast, D. Majumdar, R. Schenkel, M. Theobald, G. Weikum	IO-Top-k: index-access optimized top-k query processing
MPI-I-2006-5-001	M. Bender, S. Michel, G. Weikum, P. Triantafilou	Overlap-aware global df estimation in distributed information retrieval systems
MPI-I-2006-4-010	A. Belyaev, T. Langer, H. Seidel	Mean value coordinates for arbitrary spherical polygons and polyhedra in $\mathbb{R}^3$
MPI-I-2006-4-009	J. Gall, J. Potthoff, B. Rosenhahn, C. Schnoerr, H. Seidel	Interacting and annealing particle filters: mathematics and a recipe for applications
MPI-I-2006-4-008	I. Albrecht, M. Kipp, M. Neff, H. Seidel	Gesture modeling and animation by imitation
MPI-I-2006-4-007	O. Schall, A. Belyaev, H. Seidel	Feature-preserving non-local denoising of static and time-varying range data
MPI-I-2006-4-006	C. Theobalt, N. Ahmed, H. Lensch, M. Magnor, H. Seidel	Enhanced dynamic reflectometry for relightable free-viewpoint video
MPI-I-2006-4-005	A. Belyaev, H. Seidel, S. Yoshizawa	Skeleton-driven laplacian mesh deformations
MPI-I-2006-4-004	V. Havran, R. Herzog, H. Seidel	On fast construction of spatial hierarchies for ray tracing
MPI-I-2006-4-003	E. de Aguiar, R. Zayer, C. Theobalt, M. Magnor, H. Seidel	A framework for natural animation of digitized models
MPI-I-2006-4-002	G. Ziegler, A. Tevs, C. Theobalt, H. Seidel	GPU point list generation through histogram pyramids
MPI-I-2006-4-001	A. Efremov, R. Mantiuk, K. Myszkowski, H. Seidel	Design and evaluation of backward compatible high dynamic range video compression
MPI-I-2006-2-001	T. Wies, V. Kuncak, K. Zee, A. Podelski, M. Rinard	On verifying complex properties using symbolic shape analysis
MPI-I-2006-1-007	H. Bast, I. Weber, C.W. Mortensen	Output-sensitive autocompletion search
MPI-I-2006-1-006	M. Kerber	Division-free computation of subresultants using bezout matrices
MPI-I-2006-1-005	A. Eigenwillig, L. Kettner, N. Wolpert	Snap rounding of Bézier curves
MPI-I-2006-1-004	S. Funke, S. Laue, R. Naujoks, L. Zvi	Power assignment problems in wireless communication
MPI-I-2005-5-002	S. Siersdorfer, G. Weikum	Automated retraining methods for document classification and their parameter tuning
MPI-I-2005-4-006	C. Fuchs, M. Goesele, T. Chen, H. Seidel	An emperical model for heterogeneous translucent objects
MPI-I-2005-4-005	G. Krawczyk, M. Goesele, H. Seidel	Photometric calibration of high dynamic range cameras
MPI-I-2005-4-004	C. Theobalt, N. Ahmed, E. De Aguiar, G. Ziegler, H. Lensch, M.A. Magnor, H. Seidel	Joint motion and reflectance capture for creating relightable 3D videos
MPI-I-2005-4-003	T. Langer, A.G. Belyaev, H. Seidel	Analysis and design of discrete normals and curvatures
MPI-I-2005-4-002	O. Schall, A. Belyaev, H. Seidel	Sparse meshing of uncertain and noisy surface scattered data