Implicates and reduction techniques for temporal logics*

I.P. de Guzmán, M. Ojeda-Aciego, A. Valverde

Dept. Matemática Aplicada, Universidad de Málaga,
P.O. Box 4114, 29080. Málaga, SPAIN,
Email: {guzman,aciego,a_valverde}@ctima.uma.es

Abstract. Reduction strategies are introduced for the future fragment of a temporal propositional logic on linear discrete time, named FNext. These reductions are based in the information collected from the syntactic structure of the formula, which allow the development of efficient strategies to decrease the size of temporal propositional formulas, viz. new criteria to detect the validity or unsatisfiability of subformulas, and a strong generalisation of the pure literal rule. These results, used as a preprocessing step, allow to improve the performance of any automated theorem prover.

1 Introduction

The temporal dimension of information, the change of information over time and knowledge about how it changes has to be considered by many AI systems. There is obvious interest in designing computationally efficient temporal formalisms, specially when intelligent tasks are considered, such as planning relational actions in a changing environment, building common sense reasoning into a moving robot, in supervision of industrial processes, ...

Temporal logics are widely accepted and frequently used for specifying concurrent and reactive agents (which can be either physical devices or software processes), and in the verification of temporal properties of programs. To verify a program, one specifies the desired properties of the program by a formula in temporal logic. The program is correct if all its computations satisfy the formula. However, in its generality, an algorithmic solution to the verification problem is hopeless. For *propositional* temporal logic, checking the satisfiability of a formula can be done algorithmically, and theoretical work on the complexity of program verification is being done [3]. The complexity of satisfiability and determination of truth in a particular finite structure are considered for different propositional linear temporal logics in [7].

Linear-time temporal logics have proven [5] to be a successful formalism for the specification and verification of concurrent systems; but have a much wider range of applications, for instance, in [2] a generalisation of the temporal

* Partially supported by Spanish CICYT project TIC97-0579-C02-02 and EC action COST-15: *Many-valued logics for computer science applications.*

propositional logic of linear time is presented, which is useful for stating and proving properties of the generic execution sequence of a parallel program. On the other hand, relatively complete deductive systems for proving *branching* time temporal properties of reactive systems [4] have been recently developed.

In recent years, several fully automatic methods for verifying temporal specifications have been introduced, in [6] a tableaux calculus is treated at length; a first introduction to the tableaux method for temporal logic can be seen in [8]. However, the scope of these methods is still very limited. Theorem proving procedures for temporal logics have been traditionally based on syntactic manipulations of the formula A to be proved but, in general, do not incorporate the substitution of subformulas in A like in a rewrite system in which the rewrite relation preserves satisfiability. One source of interest of these strategies is that can be easily included into any prover, specifically into those which are non-clausal.

In this work we focus on the development of a set of reduction strategies which, through the efficient determination and manipulation of lists of unitary implicant and implicates, *investigates exhaustively* the possibility of decreasing the size of the formula being analysed. The interest of such a set of reduction techniques is that the performance of a given prover for linear-time temporal logic can be improved because the size of a formula can be decreased, at a polynomial cost, as much as possible *before* branching.

Lists of unitary models, so-called Δ-lists, are associated to each node in the syntactic tree of the formula and used to study whether the structure of the syntactic tree has or has not direct information about the validity of the formula. This way, either the method ends giving this information or, otherwise, it decreases the size of the problem before applying the next transformation. So, it is possible to decrease the number of branchings or, even, to avoid them all.

The ideas in this paper generalise the results in [1], in a self-contained way, by explicitly extending the reduction strategy to linear-time temporal logic and, what is more important, by complementing the information in the Δ-lists by means of the so-called $\widehat{\Delta}$-sets. The former allow derivation of an equivalent and smaller formula; the latter also allow derivation of a smaller formula, not equivalent to the previous one, but equisatisfiable.

The paper is organised as follows:

- Firstly, preliminary concepts, notation and basic definitions are introduced: specifically, it is worth to note the definition of literal and the way some of them will be denoted.
- Secondly, Δ-lists, our basic tool, are introduced; its definition integrates some reductions into the calculation of the Δ-lists. The required theorems to show how to use the information collected in those lists are stated.
- Later, the $\widehat{\Delta}$-sets are defined and results that use the information in these sets are stated. One of these is a generalisation of the pure literal rule.

2 Preliminary Concepts and Definitions

In this paper, our object language is the future fragment of the Temporal Propositional Logic FNext with linear and discrete flow of time, and connectives $\neg$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction), F (sometime in the future), G (always in the future), and $\oplus$ (tomorrow); $\mathcal{V}$ denotes the set of propositional variables p, q, r, ... (possibly subscripted) which is assumed to be completely ordered with the lexicographical order, e.g. $p_n \leq q_m$ for all n, m, and $p_n \leq p_m$ if and only if $n \leq m$. Given $p \in \mathcal{V}$, the formulas p and $\neg p$ are the classical literals on p.

Definition 1. *Given a classical propositional literal ℓ, the* temporal literals[1] *on ℓ, denoted* $\mathtt{Lit}(\ell)$*, are those wff of the form $\oplus^n \ell$, $F\oplus^n \ell$, $G\oplus^n \ell$, $FG\ell$, $GF\ell$ for all $n \in \mathbb{N}$.*

The notion of temporal negation normal formula*, denoted tnnf, is recursively defined as follows:*

1. *Any literal is a tnnf.*
2. *If A and B are tnnf, then $A \vee B$ and $A \wedge B$ are tnnf, which are called disjunctive and conjunctive tnnf, respectively.*
3. *If A is a disjunctive tnnf, then GA is a tnnf.*
4. *If A is a conjunctive tnnf, then FA is a tnnf.*
5. *A formula is a tnnf if and only if it can be constructed by the previous rules.*

For formulas in tnnf, we will write $\overline{p}$ for the classical negated literal $\neg p$.

As usual, a clause is a disjunction of literals and a cube is a conjunction of literals. In addition, a G-clause is a formula GB where B is a classical clause, and a F-cube is a formula FB in which B is a classical cube.

We denote $\vartheta\ell$ to mean a temporal literal on ℓ, where ϑ is said to be its *temporal prefix*; if $\vartheta\ell$ is a temporal literal, then $|\vartheta|$ denotes the number of temporal connectives in ϑ, and $\overline{\vartheta\ell}$ denotes its opposite literal, where $\overline{F} = G$, $\overline{G} = F$, $\overline{FG} = GF$, $\overline{GF} = FG$ and $\overline{\oplus} = \oplus$

The transformation of any wff into tnnf is linear by recursively applying the transformations induced by the double negation, the de Morgan laws and the equivalences in Fig. 1.

In addition, by using the associative laws we will consider expressions like $A_1 \vee \cdots \vee A_n$ or $A_1 \wedge \cdots \wedge A_n$ as formulas.

We will use the standard notion of tree and address of a node in a tree. Given a tnnf A, the *syntactic tree* of A, denoted by T_A, is defined as usual. An address η in T_A will mean, when no confusion arises, the subformula of A corresponding to the node of address η in T_A; the address of the root node will be denoted ε.

If T_C is a subtree of T_A, then the *temporal order of T_C in T_A*, denoted $ord_A(C)$, is the number of temporal ancestors of T_C in T_A.

[1] As we will be concerned only on temporal literals, in the rest of the paper we will drop the adjective *temporal*. In addition, we will use the notation $\oplus^n$ to denote the n-folded application of the connective $\oplus$.

$$
\begin{array}{llll}
\neg\oplus A \equiv \oplus\neg A & \oplus FA \equiv F\oplus A & \oplus GA \equiv G\oplus A \\
FFA \equiv F\oplus A & GGA \equiv G\oplus A & FGFA \equiv GFA \\
GFGA \equiv FGA & FG\oplus A \equiv FGA & GF\oplus A \equiv GFA \\
\oplus\bigvee A_i \equiv \bigvee\oplus A_i & \oplus\bigwedge A_i \equiv \bigwedge\oplus A_i & \neg FA \equiv G\neg A \\
\neg GA \equiv F\neg A & F(\bigvee_{i\in J} A_i) \equiv \bigvee_{i\in J} FA_i & G(\bigwedge_{i\in J} A_i) \equiv \bigwedge_{i\in J} GA_i
\end{array}
$$

Fig. 1.

We will also use lists with its standard notation, `nil`, for the empty list. Elements in a list will be written in juxtaposition.

If α and β are lists of literals and $\vartheta\ell$ is a literal, $\vartheta\ell \in \alpha$ denotes that $\vartheta\ell$ is an element of α; and $\alpha \subseteq \beta$ means that all elements of α are elements of β. If $\alpha = \vartheta_1\ell_1\vartheta_2\ell_2\ldots\vartheta_n\ell_n$, then $\overline{\alpha} = \overline{\vartheta_1\ell_1}\,\overline{\vartheta_2\ell_2}\ldots\overline{\vartheta_n\ell_n}$.

Definition 2. *A* temporal structure *is a tuple $S = (\mathbb{N}, <, h)$, where $\mathbb{N}$ is the set of natural numbers, $<$ is the standard strict ordering on $\mathbb{N}$, and h is a* temporal interpretation*, which is a function $h : \mathcal{L} \longrightarrow 2^{\mathbb{N}}$, where $\mathcal{L}$ is the language of the logic, satisfying:*

1. $h(\neg A) = \mathbb{N} \setminus h(A)$; $\quad h(A \vee B) = h(A) \cup h(B)$
2. $h(A \to B) = (\mathbb{N} \setminus h(A)) \cup h(B)$; $\quad h(A \wedge B) = h(A) \cap h(B)$
3. $t \in h(FA)$ *iff* t' *exists with* $t < t'$ *and* $t' \in h(A)$
4. $t \in h(GA)$ *iff for all* t' *with* $t < t'$ *we have* $t' \in h(A)$
5. $t \in h(\oplus A)$ *iff we have* $t + 1 \in h(A)$

A formula A is said to be *satisfiable* if there exists a temporal structure $S = (\mathbb{N}, <, h)$ such that $h(A) \neq \varnothing$; if $t \in h(A)$, then h is said to be a *model of* A in t; if $h(A) = \mathbb{N}$, then A is said to be *true in the temporal structure* S; if A is true in every temporal structure, then A is said to be *valid*, and we denote it $\models A$.

Formulas A and B are said to be *equisatisfiable* if A is satisfiable iff B is satisfiable; $\equiv$ denotes the semantic equality, i.e. $A \equiv B$ if and only if for every temporal structure $S = (\mathbb{N}, <, h)$ we have that $h(A) = h(B)$; finally, the symbols $\top$ and $\bot$ mean truth and falsity, i.e. $h(\top) = \mathbb{N}$ and $h(\bot) = \varnothing$ for every temporal structure $S = (\mathbb{N}, <, h)$.

If Γ_1 and Γ_2 are sets of subformulas in A and X and Y are subformulas, then the expression $A[\Gamma_1/X, \Gamma_2/Y]$ denotes the formula obtained after substituting in A every occurrence of elements in Γ_1 by X and every occurrence of elements in Γ_2 by Y.

If η is an address in T_A and X, then the expression $A[\eta/X]$ is the formula obtained after substituting in A the subtree rooted in η by X.

3 Adding Information to the Tree: Δ-lists

The idea underlying the reduction strategy we are going to introduce is the use of information given by partial assignments. We associate to each tnnf A two

lists of literals denoted $\Delta_0(A)$ and $\Delta_1(A)$ (the associated Δ-lists of A)[2] and two sets of lists, denoted $\widehat{\Delta_0}(A)$ and $\widehat{\Delta_1}(A)$, whose elements are obtained out of the associated Δ-lists of the subformulas of A.

The Δ-lists and the $\widehat{\Delta}$-sets are the key tools of our method to reduce the size of the formula being analysed. These reductions allow to study its satisfiability with as few branching as possible.

In a nutshell, $\Delta_0(A)$ and $\Delta_1(A)$ are, respectively, lists of temporal implicates and temporal implicants of A. The purpose of these lists is two-fold:

1. To transform the formula A into an equivalent and smaller-sized one (see Sect. 3.3).
2. To be used in the definition the $\widehat{\Delta_b}$ sets (see Sect. 4), which will be used to transform the formula A into an equisatisfiable and smaller-sized one. Furthermore, information to build a countermodel (if it exists) is provided.

The sense in which we mean temporal implicant/implicate is the following:

Definition 3.

- *A literal $\vartheta\ell$ is a* temporal implicant *of A if $\models \vartheta\ell \rightarrow A$.*
- *A literal $\vartheta\ell$ is a* temporal implicate *of A if $\models A \rightarrow \vartheta\ell$.*

3.1 The Lattices of Literals

Definition 4. *For each classical propositional literal ℓ we define an ordering in $\mathtt{Lit}(\ell) \cup \{\bot, \top\}$ as follows:*

1. *$\vartheta\ell \leq \varrho\ell$ if and only if $\models \vartheta\ell \rightarrow \varrho\ell$*
2. *$\vartheta\ell \leq \top$ for all (possibly empty) ϑ.*
3. *$\vartheta\ell \geq \bot$ for all (possibly empty) ϑ.*

Each set $\mathtt{Lit}(\ell) \cup \{\bot, \top\}$ provided with this ordering is a lattice, depicted in Figure 2. For each literal $\vartheta\ell$ we will consider its upward and downward closures, denoted $\vartheta\ell\uparrow$ and $\vartheta\ell\downarrow$.

3.2 Definition of the Δ-lists

Definition 5. *Given a tnnf A, we define $\Delta_0(A)$ and $\Delta_1(A)$ to be the lists of literals recursively defined below*

$$
\begin{aligned}
&\Delta_0(\vartheta\ell) = \Delta_1(\vartheta\ell) = \vartheta\ell \\
&\Delta_0\left(\textstyle\bigwedge_{i=1}^{n} A_i\right) = \mathtt{Union}_{\wedge}(\Delta_0(A_1), \ldots, \Delta_0(A_n)) \\
&\Delta_0\left(\textstyle\bigvee_{i=1}^{n} A_i\right) = \mathtt{Intersection}(\Delta_0(A_1), \ldots, \Delta_0(A_n)) \\
&\Delta_1\left(\textstyle\bigwedge_{i=1}^{n} A_i\right) = \mathtt{Intersection}(\Delta_1(A_1), \ldots, \Delta_1(A_n)) \\
&\Delta_1\left(\textstyle\bigvee_{i=1}^{n} A_i\right) = \mathtt{Union}_{\vee}(\Delta_1(A_1), \ldots, \Delta_1(A_n)) \\
&\Delta_b\left(FA\right) = \mathtt{Add}_{\mathtt{F}}(\Delta_b(A)) \\
&\Delta_b\left(GA\right) = \mathtt{Add}_{\mathtt{G}}(\Delta_b(A))
\end{aligned}
$$

[2] It can be shown that either A is equivalent to a literal, or at most one of these lists is non-empty.

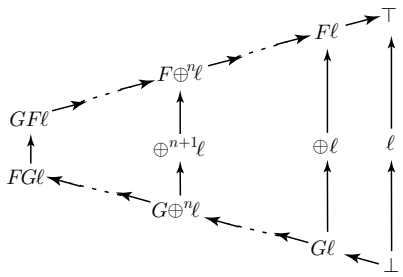


Fig. 2. Lattice $\mathtt{Lit}(\ell) \cup \{\bot, \top\}$

The description of the operators involved in the definition above is the following:

1. The operators `Add` add a temporal connective to each element of a list of literals and simplify the results to a tnnf according to the rules in Fig. 1.
2. The two versions of `Union` arise because of the intended interpretation of these sets:
   (a) Elements in Δ_0 are considered as conjunctively connected, so we use $\mathtt{Union}_\wedge$. This way, we obtain *minimal* implicates.
   (b) Elements in Δ_1 are considered as disjunctively connected, so we use $\mathtt{Union}_\vee$. This way, we obtain *maximal* implicants.

Remark 1. By conjunctively connected, we mean that two literals in Δ_0 are substituted by its conjunction if it is either a literal or $\top$ or $\bot$, i.e. $\vartheta\ell \wedge \vartheta\ell\uparrow = \vartheta\ell$, $\vartheta\ell \wedge \overline{\vartheta\ell}\downarrow = \bot$ and the pair of literals $G\oplus^{n+1}\ell$ and $\oplus^{n+1}\ell$ is simplified to $G\oplus^n\ell$, for all n.

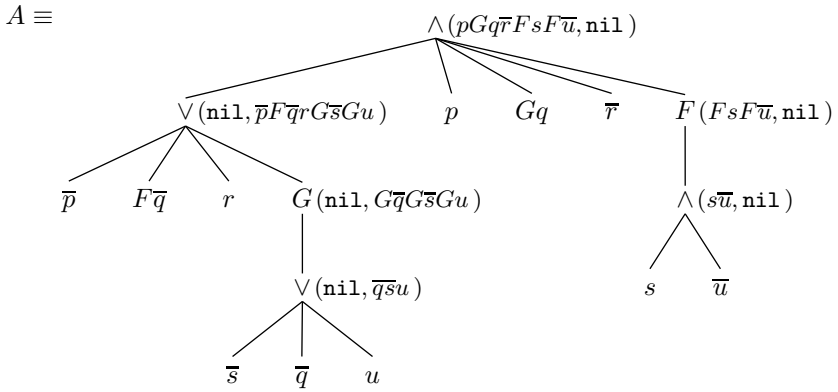
Similarly, the disjunctive connection in Δ_1 means the application of the following rules $\vartheta\ell \vee \vartheta\ell\downarrow = \vartheta\ell$, $\vartheta\ell \vee \overline{\vartheta\ell}\uparrow = \top$, and the pair of literals $F\oplus^{n+1}\ell$ and $\oplus^{n+1}\ell$ is simplified to $F\oplus^n\ell$ in Δ_1, for all n.

It is easy to see that, for all ℓ, we have that $\Delta_b(A) \cap \mathtt{Lit}(\ell)$ contains at most one literal in the set $\{F\oplus^k\ell, G\oplus^k\ell, FG\ell, GF\ell\}$ and, possibly, several of the type $\oplus^k\ell$.

Definition 6. *If a A is a tnnf, then to* Δ-label *A means to label each node η in A with the ordered pair $(\Delta_0(\eta), \Delta_1(\eta))$.*

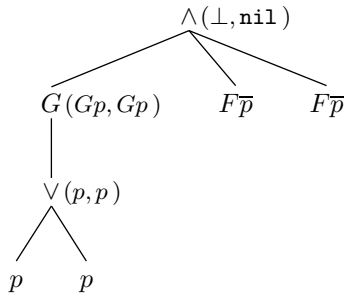
Example 1. Consider the formula $A = (\neg p \vee \neg Gq \vee r \vee G(\neg s \vee \neg q \vee u)) \wedge \neg(\neg p \vee \neg Gq \vee r \vee G(\neg s \vee u))$; the Δ-labelled tree of A is[3]

[3] For the sake of clarity, the Δ-labels of the leaves are not written.



Note that in node 1, literals $F\overline{q}$ and $G\overline{q}$ are collapsed into $F\overline{q}$, because of the disjunctive connection in Δ_1.

Example 2. Let us study the validity of $A = G(\neg p \to p) \to (\neg Gp \to Gp)$. The Δ-labelled tree equivalent to $\neg A$ is



In this case, $\Delta_0(\varepsilon) = \bot$, because of the simplification of Gp and $F\overline{p}$ due to the conjunctive nature of the Δ_0-sets. We will see later that $\Delta_0(\varepsilon) = \bot$ implies that the input formula, that is $\neg A$, is unsatisfiable, therefore A is valid.

3.3 Information in the Δ-lists

As indicated above, the purpose of defining Δ_0 and Δ_1 is to collect implicants and implicates of A, as shown in the following theorem.

Theorem 1. *Let A be a tnnf,*

1. *If $\vartheta\ell \in \Delta_0(A)$, then $\models A \to \vartheta\ell$.*
2. *If $\vartheta\ell \in \Delta_1(A)$, then $\models \vartheta\ell \to A$.*

The theorem above will be used in the following equivalent form:

1. If $\vartheta\ell \in \Delta_0(A)$, then $A \equiv A \wedge \vartheta\ell$.
2. If $\vartheta\ell \in \Delta_1(A)$, then $A \equiv A \vee \vartheta\ell$.

As a literal is satisfiable, by Theorem 1 item 2, we have the following result:

Corollary 1. *If $\Delta_1(A) \neq \mathtt{nil}$, then A is satisfiable.*

3.4 Strong Meaning-Preserving Reductions

A lot of information can be extracted from the Δ-lists as corollaries of Theorem 1. The first result is a structural one, for it says that either one of the Δ-lists is empty, or both are equal and singletons.

Corollary 2. *If A is not a literal and $\Delta_1(A) \neq \texttt{nil} \neq \Delta_0(A)$, then there exists $\vartheta\ell$ such that $\Delta_1(A) = \Delta_0(A) = \vartheta\ell$. Such tnnf A is said to be $\vartheta\ell$-simple.*

The corollary below states conditions on the Δ-lists which allow to determine the validity or unsatisfiability of the formula we are studying.

Corollary 3. *Let A be a tnnf, then*

1. (a) *If $\Delta_0(A) = \bot$, then $A \equiv \bot$.*
   (b) *If $A = \bigwedge_{i=1}^{n} A_i$ in which a conjunct A_{i_0} is a clause such that $\overline{\Delta_1(A_{i_0})} \subseteq \Delta_0(A)\uparrow$, then $A \equiv \bot$.*
   (c) *If $A = \bigwedge_{i=1}^{n} A_i$ in which a conjunct A_{i_0} is a G-clause GB such that $\overline{\texttt{Add}_{\oplus}(\Delta_1(B))} \subseteq \Delta_0(A)\uparrow$, then $A \equiv \bot$.*
2. (a) *If $\Delta_1(A) = \top$, then $A \equiv \top$.*
   (b) *If $A = \bigvee_{i=1}^{n} A_i$ in which a disjunct A_{i_0} is a cube such that $\overline{\Delta_0(A_{i_0})} \subseteq \Delta_1(A)\downarrow$, then $A \equiv \top$.*
   (c) *If $A = \bigvee_{i=1}^{n} A_i$ in which a disjunct A_{i_0} is an F-cube FB such that $\overline{\texttt{Add}_{\oplus}(\Delta_0(B))} \subseteq \Delta_1(A)\downarrow$, then $A \equiv \top$.*

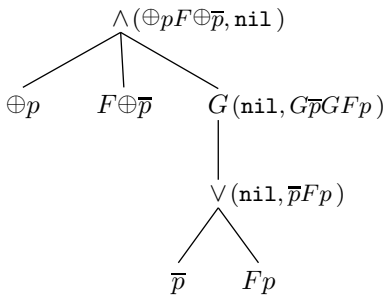
The following definition gives a name to those formulas which have been simplified by using the information in the Δ-lists.

Definition 7. *Let A be an tnnf then it is said that A is:*

1. finalizable *if either $A = \top$, or $A = \bot$ or $\Delta_1(A) \neq \texttt{nil}$.*
2. *A tnnf verifying either (a) or (b) or (c) of item 1 in Corollary 3 is said to be Δ_0-conclusive.*
3. *A tnnf verifying either (a) or (b) or (c) of item 2 in Corollary 3 is said to be Δ_1-conclusive.*
4. *A tnnf A is said to be* Δ-restricted *if it has no subtree which is either Δ_0-conclusive, or Δ_1-conclusive, or $\vartheta\ell$-simple.*
5. *To* Δ-restrict *a tnnf A means to substitute each Δ_1-conclusive formula by $\top$, each Δ_0-conclusive formula by $\bot$, and each $\vartheta\ell$-simple formula by $\vartheta\ell$; and then eliminate the constants $\top$ and $\bot$ by applying the 0-1 laws.*
   Note that Δ-restricting is a meaning-preserving transformation.

Example 3. Given the transitivity axiom $A = FFp \rightarrow Fp$; the tnnf equivalent to $\neg A$ is $F{\oplus}p \wedge G\overline{p}$; since $\Delta_0(F{\oplus}p \wedge G\overline{p}) = \bot$, we have that $\neg A$ is Δ_0-conclusive, therefore $\neg A$ is unsatisfiable and A is valid.

Example 4. Given the formula $A = {\oplus}p \wedge {\oplus}F\overline{p} \wedge G(p \rightarrow Fp)$, its Δ-labelled tree is



This tree is Δ_0-conclusive, since $\overline{\mathtt{Add}_{\oplus}(\Delta_1(31))} = \oplus pF\oplus\overline{p} \subseteq \Delta_0(\varepsilon)\uparrow$. In fact, what we have in this example is $\overline{\mathtt{Add}_{\oplus}(\Delta_1(31))} = \Delta_0(\varepsilon)$

3.5 Weak Meaning-Preserving Reductions

The aim of this section is to give more general conditions allowing to use the information in the Δ-lists which has not been able to be used by the strong reductions. Specifically, a strong reduction uses the information in the Δ-lists in a *strong sense*, that is, to substitute a whole subformula by either $\top$, or $\bot$, or a literal. As in the propositional case, sometimes this is not possible and we can only use the information in a *weak sense*, that is, to decrease the size of the formula by eliminating literals depending on the elements of the Δ-lists.

The following notation is used in the statement of some results hereafter:

- If $\mathcal{S}$ is a set of literals in a tnnf A, then $\mathcal{S}^0$ denotes the set of all the occurrences of literals $\vartheta\ell \in \mathcal{S}$ of temporal order 0 in A
- $\mathtt{Lit}(\ell, n) = \{\eta \mid \eta = \vartheta\ell \text{ and } |\vartheta| + ord_A(\eta) \geq n+1\}$
- $\mathtt{Lit}(\overline{\ell}, n) = \{\eta \mid \eta = \vartheta\overline{\ell} \text{ and } |\vartheta| + ord_A(\eta) \geq n+1\}$

Theorem 2. *Let A be a tnnf and $\vartheta\ell$ a literal in A:*

1. *If $\vartheta\ell \in \Delta_0(A)$, then $A \equiv \vartheta\ell \wedge A[(\vartheta\ell\uparrow)^0/\top, (\overline{\vartheta\ell}\downarrow)^0/\bot]$*
2. *If $\vartheta\ell \in \Delta_1(A)$, then $A \equiv \vartheta\ell \vee A[(\vartheta\ell\downarrow)^0/\bot, (\overline{\vartheta\ell}\uparrow)^0/\top]$*

This theorem cannot be improved for an arbitrary literal $\vartheta\ell$; although, for some particular cases, it is possible to get more literals reduced, as shown by the following theorem, which generalises the result in Theorem 2, by dropping the restriction of order 0 for all the literals in the upward/downward closures.

Theorem 3. *Let A be a tnnf,*

1. *If $\vartheta\ell \in \Delta_0(A)$ with $\vartheta\ell \in \{FG\ell, GF\ell\} \cup \{G\oplus^n\ell \mid n \in \mathbb{N}\}$, then*

$$A \equiv \vartheta\ell \wedge A[\vartheta\ell\uparrow/\top, \overline{\vartheta\ell}\downarrow/\bot]$$

2. *If $\vartheta\ell \in \Delta_1(A)$ with $\vartheta\ell \in \{FG\ell, GF\ell\} \cup \{F\oplus^n\ell \mid n \in \mathbb{N}\}$, then*

$$A \equiv \vartheta\ell \vee A[\vartheta\ell\downarrow/\bot, \overline{\vartheta\ell}\uparrow/\top]$$

Finally, the theorem below states a number of additional reductions that can be applied when $\vartheta\ell$ equals either $G \oplus^n \ell$ or $F \oplus^n \ell$.

Theorem 4. *Let A be a tnnf and $\vartheta\ell$ a literal in A:*

1. *If $G \oplus^n \ell \in \Delta_0(A)$, then $A \equiv G \oplus^n \ell \wedge A[\mathtt{Lit}(\ell, n)/\top, \mathtt{Lit}(\overline{\ell}, n)/\bot]$*
2. *If $F \oplus^n \ell \in \Delta_1(A)$, then $A \equiv F \oplus^n \ell \vee A[\mathtt{Lit}(\ell, n)/\bot, \mathtt{Lit}(\overline{\ell}, n)/\top]$*

4 Adding Information to the Tree: $\widehat{\Delta}$-sets

In the previous sections, the information in the Δ-lists has been used *locally*, that is, the information in $\Delta_b(\eta)$ has been used to reduce η. The purpose of defining a new structure, the $\widehat{\Delta}$-sets, is to allow the globalisation of the information, in that the information in $\Delta_b(\eta)$ can be *refined* by the information in its ancestors.

Given a Δ-restricted tnnf A, we define the sets $\widehat{\Delta_0}(A)$ and $\widehat{\Delta_1}(A)$, whose elements are pairs (α, η) where α is a *reduced* Δ-list (to be defined below) associated to a subformula B of A, and η is the address of B in A. These sets allow to transform the formula A into an equisatisfiable and smaller sized one, as seen in Section 4.1.

The following result uses those cases in Theorems 2, 3 and 4 which allow to delete a whole subformula. The rest of possibilities only allow to delete literals; these literals will be called *reducible*.

Theorem 5. *Let A be a tnnf, B a subformula of A, and η the address in the tree of A of a subformula of B:*

1. (a) *If $\vartheta\ell$ is any literal satisfying $\vartheta\ell \in \Delta_0(\eta)\uparrow \cap (\Delta_1(B) \cup \overline{\Delta_0(B)})$ and $ord_B(\eta) = 0$, then $A \equiv A[\eta/\bot]$.*
   (b) *If $\vartheta\ell \in \{FG\ell, GF\ell\} \cup \{G \oplus^n \ell \mid n \in \mathbb{N}\}$ and satisfies and $\vartheta\ell \in \Delta_0(\eta)\uparrow \cap (\Delta_1(B) \cup \overline{\Delta_0(B)})$, then $A \equiv A[\eta/\bot]$.*
   (c) *If $\vartheta\ell \in \Delta_0(\eta)\uparrow$, and $F \oplus^n \ell \in \Delta_1(B) \cup \overline{\Delta_0(B)}$, and $|\vartheta| + ord_B(\eta) \geq n+1$, then $A \equiv A[\eta/\bot]$.*
2. (a) *If $\vartheta\ell$ is any literal satisfying $\vartheta\ell \in \Delta_1(\eta)\downarrow \cap (\Delta_0(B) \cup \overline{\Delta_1(B)})$ and $ord_B(\eta) = 0$, then $A \equiv A[\eta/\top]$.*
   (b) *If $\vartheta\ell \in \{FG\ell, GF\ell\} \cup \{G \oplus^n \ell \mid n \in \mathbb{N}\}$ and satisfies and $\vartheta\ell \in \Delta_1(\eta)\downarrow \cap (\Delta_0(B) \cup \overline{\Delta_1(B)})$, then $A \equiv A[\eta/\top]$.*
   (c) *If $\vartheta\ell \in \Delta_1(\eta)\downarrow$, and $G \oplus^n \ell \in \Delta_0(B) \cup \overline{\Delta_1(B)}$, and $|\vartheta| + ord_B(\eta) \geq n+1$, then $A \equiv A[\eta/\top]$*

This theorem can be seen as a generalisation of Corollary 3, in which a subformula B can be substituted by a constant even when that subformula *is not equivalent to* that constant.

The subformula at address η in A is said to be 0-*conclusive in A (resp.* 1-*conclusive in A)* if it verifies some of the conditions in item 1 (resp. item 2) above.

Definition 8. *Given a tnnf A and an address η, the* reduced Δ*-lists for A, $\Delta_b^A(\eta)$ for $b \in \{0,1\}$, are defined below,*

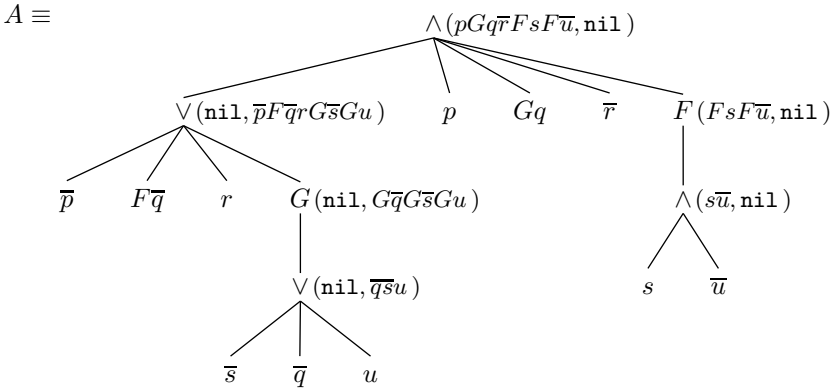
1. *If η is 0-conclusive in A, then $\Delta_0^A(\eta) = \bot$.*
2. *If η is 1-conclusive in A, then $\Delta_1^A(\eta) = \top$.*
3. *Otherwise, $\Delta_b^A(\eta)$ is the list $\Delta_b(\eta)$ in which the reducible literals have been deleted.*

We define the sets $\widehat{\Delta_b}(A)$ as follows

$$\widehat{\Delta_b}(A) = \{(\Delta_b^A(\eta), \eta) \mid \ \eta \text{ is a non-leaf address in } T_A \text{ with } \Delta_b(\eta) \neq \texttt{nil}\}$$

If A is a tnnf, to label *A means Δ-label A and to associate to the root of A the ordered pair $\left(\widehat{\Delta_0}(A), \widehat{\Delta_1}(A)\right)$.*

Example 5. From Example 1 we had the following tree



Note that literals $\overline{p}, F\overline{q}$ and r in Δ_1 of node 1 are reducible in A because of the occurrence of its duals in Δ_0 of the root. Similarly $G\overline{q}$ is also reducible in node 14, and $\overline{q}$ is reducible in 141. Therefore, the calculation of the $\widehat{\Delta}$-sets leads to

$$\widehat{\Delta}_0(A) = \{(pGq\overline{r}FsF\overline{u}, \varepsilon), (FsF\overline{u}, 5), (s\overline{u}, 51)\}$$
$$\widehat{\Delta}_1(A) = \{(G\overline{s}Gu, 1), (G\overline{s}Gu, 14), (\overline{s}u, 141)\}$$

4.1 Satisfiability-Preserving Results

In this section we study the information which can be extracted from the $\widehat{\Delta}$-sets.

Definition 9. *Let A be a tnnf then it is said that A is* restricted *if it is Δ-restricted and satisfies the following:*

- *There are not elements $(\bot, \eta)$ in $\widehat{\Delta_0}(A)$.*
- *There are not elements $(\top, \eta)$ in $\widehat{\Delta_1}(A)$.*

Remark 2. A restricted and equivalent tnnf can be obtained by using the 0-1 laws in conjunction with the elimination of conclusive subformulas in A, according to Theorem 5.

The following results will allow, by using the information in the $\widehat{\Delta}$-sets, to substitute a tnnf A by an equisatisfiable and smaller sized A'.

Complete Reduction

This section is named after Theorem 6, because after its application on a literal $G\oplus^n\ell$, gives an equisatisfiable formula whose only literals in ℓ are of the form $\oplus^n\ell$.

Definition 10. *A tnnf A is said to be* $G\oplus^n\ell$-completely reducible *if $G\oplus^n\ell \in \alpha$ for $(\alpha, \varepsilon) \in \widehat{\Delta_0}(A)$.*

Theorem 6. *If A be a $G\oplus^n\ell$-completely reducible tnnf, then A is satisfiable if and only if*

$$B[G\oplus^k\ell/\oplus^{k+1}\ell \wedge \ldots \wedge \oplus^n\ell, F\oplus^k\overline{\ell}/\oplus^{k+1}\overline{\ell} \vee \ldots \vee \oplus^n\overline{\ell}]$$

where $B = A[\mathtt{Lit}(\ell, n) \cup G\oplus^n\ell\uparrow/\top, \mathtt{Lit}(\overline{\ell}, n) \cup F\oplus^n\overline{\ell}\downarrow/\bot]$.

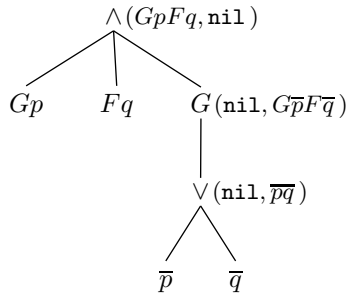
Furthermore, if h is a model of B in t, then the interpretation h' such that $h'(q) = h(q)$ if $q \neq p$ and $h'(p) = h(p) \cup [t + n + 2, \infty)$ is a model of A in t.

Example 6. Given the density axiom $A = Fp \rightarrow FFp$; the formula $\neg A$ is equivalent to the tnnf $Fp \wedge G\oplus\overline{p}$.

We have that $\Delta_0(Fp \wedge G\oplus\overline{p}) = FpG\oplus\overline{p}$. Note that, as the conjunction of Fp and $G\oplus\overline{p}$ is not a literal, no simplification can be applied. In addition, its $\widehat{\Delta_0}$-set is $\{(FpG\oplus\overline{p}, \varepsilon)\}$, thus $\neg A$ is completely reducible.

Now applying Theorem 6, we get that $\neg A$ is satisfiable if and only if $\oplus p$ is satisfiable. Therefore $\neg A$ is satisfiable, a model being $h(\overline{p}) = [2, \infty)$, $h(p) = \{1\}$.

Example 7. Given the formula $A = (Gp \wedge Fq) \rightarrow F(p \wedge q)$, we have $\neg A \equiv Gp \wedge Fq \wedge G(\overline{p} \vee \overline{q})$; its Δ-restricted form is



and its $\widehat{\Delta}$-sets are:

$$\widehat{\Delta_0}(A) = \{(GpFq, \varepsilon)\} \qquad \widehat{\Delta_1}(A) = \{(G\overline{p}F\overline{q}, 3), (\overline{pq}, 31)\}$$

This formula is completely reducible, by an application of Theorem 6, the leaf in node 1 is deleted, and node 3 is substituted by $G\overline{q}$.

The resulting formula is $Fq \wedge G\overline{q}$, which is 0-conclusive and, therefore, unsatisfiable.

The Pure Literal Rule

The result introduced here is an extension of the well known *pure literal rule* for Classical Propositional Logic. Existing results in the bibliography allow a straightforward extension of the concept of pure literal. Our definition makes use of the $\widehat{\Delta}$-sets, which allow to focus only on those literals which are essential parts of the formula; this is because reducible literals *are not included* in the $\widehat{\Delta}$-sets.

Definition 11. *Let A be a tnnf.*

1. *A classical literal ℓ is said to be $\widehat{\Delta}$-pure in A if a literal $\vartheta\ell$ occurs in $\widehat{\Delta_0}(A) \cup \widehat{\Delta_1}(A)$ and no literal on $\vartheta'\overline{\ell}$ occurs in $\widehat{\Delta_0}(A) \cup \widehat{\Delta_1}(A)$.*
2. *A classical literal ℓ is said to be $\widehat{\Delta}$-k-pure in A if $\oplus^k\ell$ occurs in an $(\alpha, \eta) \in \widehat{\Delta_0}(A) \cup \widehat{\Delta_1}(A)$ with $ord_A(\eta) = 0$, $\oplus^k\overline{\ell}$ does not occur in any $(\alpha, \eta) \in \widehat{\Delta_0}(A) \cup \widehat{\Delta_1}(A)$ with $ord_A(\eta) = 0$, and for any other literal $\vartheta\ell$ or $\vartheta'\ell$, occurring in some element $(\alpha, \eta) \in \widehat{\Delta_0}(A) \cup \widehat{\Delta_1}(A)$, we have $|\vartheta| + ord_A(\eta) > k$.*

Theorem 7. *Let A be a tnnf, ℓ a $\widehat{\Delta}$-pure literal in A, and B the formula obtained from A by the following substitutions*

1. *If $(\alpha, \eta) \in \widehat{\Delta_0}(A)$ with $\vartheta\ell \in \alpha$, then η is substituted by*

$$\begin{cases} \eta[\mathtt{Lit}(\ell, n) \cup G\oplus^n\ell\uparrow/\top, \mathtt{Lit}(\overline{\ell}, n) \cup F\oplus^n\overline{\ell}\downarrow/\bot] & \text{if } \vartheta\ell = G\oplus^n\ell \\ \eta[\vartheta\ell\uparrow/\top, \overline{\vartheta\ell}\downarrow/\bot] & \text{if } \vartheta\ell \in \{GF\ell, FG\ell\} \\ \eta[(\vartheta\ell\uparrow)^0/\top, (\overline{\vartheta\ell}\downarrow)^0/\bot] & \text{otherwise} \end{cases}$$

2. *If $(\alpha, \eta) \in \widehat{\Delta_1}(A)$ with $\vartheta\ell \in \alpha$, then η is substituted by $\top$.*

Then, A is satisfiable if and only if B is satisfiable. Furthermore, if h is a model of B in t, then the interpretation h' such that $h'(\ell') = h(\ell')$ if $\ell' \neq \ell$ and $h'(\ell) = [t, \infty)$ is a model of A in t.

Theorem 8. *Let A be a tnnf, ℓ a $\widehat{\Delta}$-k-pure literal in A, and B the formula obtained from A by the following substitutions*

1. *If $(\alpha, \eta) \in \widehat{\Delta_0}(A)$ with $\oplus^k\ell \in \alpha$ and $ord_A(\eta) = 0$, then η is substituted by $\eta[(\oplus^k\ell\uparrow)^0/\top, (\oplus^k\overline{\ell}\downarrow)^0/\bot]$*
2. *If $(\alpha, \eta) \in \widehat{\Delta_1}(A)$ with $\oplus^k\ell \in \alpha$, then η is substituted by $\top$*

Then, A is satisfiable if and only if B is satisfiable. Furthermore, if h is a model of B in t, then the interpretation h' such that $h'(\ell') = h(\ell')$ if $\ell' \neq \ell$ and $h'(\ell) = h(\ell) \cup \{t + k\}$ is a model of A in t.

Example 8. Following with the formula in Example 5, we had

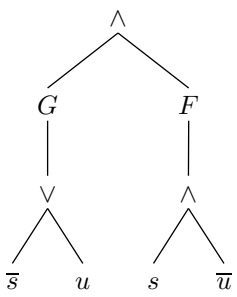
$$\widehat{\Delta_0}(A) = \{(pGq\overline{r}, \varepsilon), (FsF\overline{u}, 5), (s\overline{u}, 51)\}$$
$$\widehat{\Delta_1}(A) = \{(G\overline{s}Gu, 1)(G\overline{s}Gu, 14), (\overline{s}u, 141)\}$$

therefore

1. It is completely reducible: $Gq \in \alpha$ with $(\alpha, \varepsilon) \in \widehat{\Delta}_0(A)$.
2. literals p and $\overline{r}$ are 0-pure.

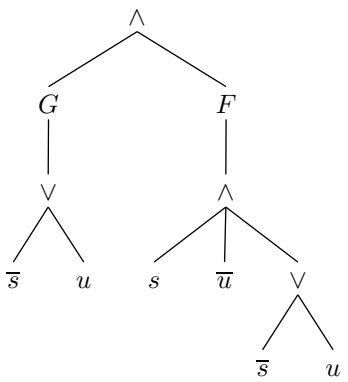
When applying the corresponding substitutions we get

$B =$



This formula cannot be reduced any longer. By applying a branching rule[4] we obtain

$B =$



It is easy to check that node 21 is Δ_0-conclusive, by substituting this node by $\bot$ we get $\bot$ as a final result. Therefore the formula is unsatisfiable.

5 Conclusions and Future Work

We have introduced techniques for defining and manipulating lists of unitary implicants/implicates which can improve the performance of a given prover for temporal propositional logics by decreasing the size of the formulas to be branched. These strategies are interesting because can be used in *any* existing theorem prover, specially in non-clausal ones.

As future work, the information in the Δ-lists can be increased by refining the process of generation of temporal implicants/implicates. In addition, current work on G-clauses and F-cubes appears to be a new source of reduction results.

[4] Every prover for linear-time temporal logic has such rules, in the example we use just one of those in the literature.

References

1. G. Aguilera, I. P. de Guzmán, and M. Ojeda. Increasing the efficiency of automated theorem proving. *Journal of Applied Non-Classical Logics*, 5(1):9–29, 1995.
2. R. Ben-Eliyahu and M. Magidor. A temporal logic for proving properties of topologically general executions. *Information and Computation* 124(2):127–144, 1996.
3. C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *Journal of the ACM* 42(4):857–907, 1995.
4. L. Fix and O. Grumberg. Verification of temporal properties. *Journal of Logic and Computation* 6(3):343–362, 1996.
5. Z. Manna and A. Pnueli. *The temporal logic of reactive and concurrent systems: specifications.* Springer-Verlag, 1992.
6. Z. Manna and A. Pnueli. *Temporal verification of reactive systems: Safety.* Springer-Verlag, 1995.
7. A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733-749, 1985.
8. P. Wolper. The tableaux method for temporal logic: an overview. *Logique et Analyse* 28 année, 110-111:119–136, 1985.