Automating natural deduction for linear-time temporal logic.

Alexander Bolotov[1]
Oleg Grigoriev[2]
Vasilyi Shangin[2]

[1] Harrow School of Computer Science, University of Westminster
[2] Department of Logic, Faculty of Philosophy, Moscow State University

Automated Natural Deduction for Propositional Linear-time Temporal Logic[*]

Alexander Bolotov
Harrow School of Computer Science
University of Westminster
Watford Road, Harrow HA1 3TP, UK.
A.Bolotov@wmin.ac.uk

Oleg Grigoriev, Vasilyi Shangin
Department of Logic, Faculty of Philosophy
Moscow State University, Moscow, 119899, Russia.
{shangin,grig}@philos.msu.ru

Abstract

We present a proof searching technique for the natural deduction calculus for the propositional linear-time temporal logic and prove its correctness. This opens the prospect to apply our technique as an automated reasoning tool in a number of emerging computer science applications and in a deliberative decision making framework across various AI applications.

1 Introduction

In this paper we present a proof searching technique for the natural deduction proof system for the propositional linear-time temporal logic PLTL [9] and establish its correctness. The particular approach to build a natural deduction calculus we are interested in is a modification of Quine's representation of subordinate proof [15] developed for classical propositional and first-order logic. Recall that natural deduction calculi (abbreviated in this paper by 'ND') of this type were originally developed by Jaskowski [11], improved by Fitch [8] and simplified by Quine [15].

The ND technique initially defined for classical propositional logic was extended to first-order logic [3, 4] and subsequently to the non-classical framework of propositional intuitionistic logic [13]. In [2] it was further extended to capture propositional linear-time temporal logic PLTL and in [5] the ND system was proposed for the computation tree logic CTL.

The computer science community has recently become more interested in ND systems [1, 14] mostly due to its potential to represent the goal-directed nature of the proof. This makes the ND method applicable in many AI areas, most notably, in agent engineering [18]. Among other interesting and even surprising applications of ND systems is for example their use in the verification of security protocols [6]. Obviously, the extension of ND to the temporal framework, widely used in agent engineering and verification, opens broader prospectives for research in ND constructions. However, from the practical point of view, its success depends on the automation of the proof searching procedure. The latter is the subject of the current paper.

We are extending the proof searching technique which was initially developed for the classical case [3, 4], and also extended to intuitionistic logic [13]. We are not aware of any other proof search algorithm for temporal ND systems. For example, the only other ND constructions for linear-time logic [10] and branching-time logic [16] which we are aware of have not been followed by any presentation of the relevant proof searching techniques.

Note that while working on the mechanisation of the ND system for PLTL, known as $\mathrm{PLTL_{ND}}$, and its correctness we also found a simpler formulation of the underlying ND system.

The paper is organized as follows. In §2 we describe $\mathrm{PLTL_{ND}}$ reviewing the PLTL syntax and semantics in §2.1 and formulating the natural deduction calculus in §2.2. Subsequently, in §3, we introduce the main proof-searching procedures (§3.1), the proof-searching algorithm (§3.2), give an example of the algorithmic construction of the proof (§3.3) and provide the correctness argument (§3.4). Finally, in §4, we provide concluding remarks and identify future work.

2 Natural Deduction System $\mathrm{PLTL_{ND}}$

In this section we review the logic PLTL and the calculus $\mathrm{PLTL_{ND}}$.

2.1 Syntax and Semantics of PLTL

In the syntax of PLTL we identify a set, $Prop$, of atomic propositions: $p, q, r, \ldots, p_1, q_1, r_1, \ldots, p_n, q_n, r_n, \ldots$; classical operators: $\neg, \wedge, \Rightarrow, \vee$, and temporal operators: $\Box$ ('always in the future'), $\Diamond$ ('at sometime in the future'), $\bigcirc$ ('at the next moment in time'), and $\mathcal{U}$ ('until').

The set of *well-formed formulae* of PLTL, wff_{PLTL} is defined as follows.

[*] This research was partially supported by Russian Foundation for Humanities, grant No 06-03-00020a.

14th International Symposium on Temporal Representation and Reasoning (TIME'07
0-7695-2836-8/07 $20.00 © 2007 IEEE

Definition 1 (PLTL syntax) *1. All atomic propositions (members of Prop) are in wff$_{PLTL}$.*

2. *If A and B are in* wff$_{PLTL}$*, then so are $A \wedge B$, $\neg A$, $A \vee B$, and $A \Rightarrow B$.*

3. *If A and B are in* wff$_{PLTL}$*, then so are $\Box A$, $\Diamond A$, $\bigcirc A$, and $A \mathcal{U} B$.*

For the semantics of PLTL we utilise the notation of [7]. A model for PLTL formulae, is a discrete, linear sequence of states $\sigma = s_0, s_1, s_2, \ldots$ which is isomorphic to the natural numbers, $\mathcal{N}$, and where each state, $s_i,\ 0 \leq i$, consists of the propositions that are true in it at the i-th moment of time. If a well-formed formula A is satisfied in the model σ at the moment i then we abbreviate it by $\langle\sigma, i\rangle \models A$. Below, in Figure 1, we define the relation $\models$, where indices $i, j, k \in \mathcal{N}$.

$\langle\sigma, i\rangle$	$\models p$	iff	$p \in s_i$, for $p \in Prop$
$\langle\sigma, i\rangle$	$\models \neg A$	iff	$\langle\sigma, i\rangle \not\models A$
$\langle\sigma, i\rangle$	$\models A \wedge B$	iff	$\langle\sigma, i\rangle \models A$ and $\langle\sigma, i\rangle \models B$
$\langle\sigma, i\rangle$	$\models A \vee B$	iff	$\langle\sigma, i\rangle \models A$ or $\langle\sigma, i\rangle \models B$
$\langle\sigma, i\rangle$	$\models A \Rightarrow B$	iff	$\langle\sigma, i\rangle \not\models A$ or $\langle\sigma, i\rangle \models B$
$\langle\sigma, i\rangle$	$\models \Box A$	iff	for each j if $i \leq j$ then $\langle\sigma, j\rangle \models A$
$\langle\sigma, i\rangle$	$\models \Diamond A$	iff	there exists j such that $i \leq j$ and $\langle\sigma, j\rangle \models A$
$\langle\sigma, i\rangle$	$\models \bigcirc A$	iff	$\langle\sigma, i+1\rangle \models A$
$\langle\sigma, i\rangle$	$\models A \mathcal{U} B$	iff	there exists j such that $i \leq j$ and $\langle\sigma, j\rangle \models B$ and for each k, if $i \leq k < j$ then $\langle\sigma, k\rangle \models A$

Figure 1. Semantics for PLTL

Definition 2 (PLTL Satisfiability) *A well-formed formula, A, is satisfiable if, and only if, there exists a model σ such that $\langle\sigma, 0\rangle \models A$.*

Definition 3 (PLTL Validity) *A well-formed formula, A, is valid if, and only if, A is satisfied in every possible model, i.e.for each σ, $\langle\sigma, 0\rangle \models A$.*

2.2 The Calculus PLTL$_{\text{ND}}$

Here we present the formulation of PLTL$_{\text{ND}}$ with a slightly different set of rules in comparison with its original formulation in [2]. Namely, now we have two new rules, application of negation to $\mathcal{U}$ and $\Diamond$ operators, but fewer introduction rules for $\mathcal{U}$ (see details below).

The core idea of a natural deduction proof technique for a logic L is to establish rules of the following two classes: *elimination* rules which decompose formulae and *introduction* rules aimed at constructing formulae, introducing new logical constants. Given a task to prove some formula A of L, we aim at synthesising A. Every proof commences with an assumption and, in general, we are allowed to introduce assumptions at any step of the proof. In the type of natural deduction that we are interested in, assumptions have conditional interpretation. Namely, given that a formula A is preceded in a proof by assumptions $C_1, C_2, \ldots C_n$ we interpret this situation as follows: if $C_1, C_2, \ldots C_n$ are satisfiable in L then A is satisfiable in L. Thus, if A is a theorem (a valid formula in L) and we want to obtain its proof then we must interpret A 'unconditionally', i.e. it should not depend on any assumptions. In our system, the corresponding process is called *discarding* of assumptions, which accompanies the application of several introduction rules. As we will see below, in a proof of a theorem in our system the set of non-discarded assumptions should be empty.

Another feature of our construction of PLTL$_{\text{ND}}$ is the use of the labeling technique. In the language of PLTL$_{\text{ND}}$ we use labeled PLTL formulae and a specific type of expressions that use labels themselves, called *relational judgements*. Thus, additionally to elimination and introduction rules, we also establish rules to manipulate with relational judgements.

Extended PLTL Syntax and Semantics.

We extend the PLTL language by introducing labels. Labels are terms, elements of the set, $Lab = \{x, y, z, x_1, x_2, x_3, \ldots\}$, where $x, y, z \ldots$ are variables. When constructing a PLTL$_{\text{ND}}$ proof, we associate formulae appearing in the proof with a model σ described in §2.1 such that labels in the proof are interpreted over the states of σ. Since σ is isomorphic to natural numbers, we can introduce the operations on labels: $\simeq$, which stands for the equality between labels, $\preceq$ and $\prec$, which are syntactic analogues of the $\leq$ and $<$ relation in σ. Thus, $\preceq$ satisfies the following properties:

(2.1) For any $i \in Lab: \ i \preceq i$ (reflexivity),

(2.2) For any $i, j, k \in Lab$ if $i \preceq j$ and $j \preceq k$ then $i \preceq k$ (transitivity).

(2.3) For any $i, j, k \in Lab$ if $i \preceq j$ and $i \preceq k$ then $j \prec k$ or $k \prec j$ or $j \simeq k$ (linearity).

(2.4) For any $i \in Lab$, there exists $j \in Lab$ such that $i \preceq j$ (seriality).

Now, we define a relation $Next \subset Lab^2$: $Next(x, y) \Leftrightarrow x \prec y$ and there is no $z \in Lab$ such that $x \prec z$ and $z \prec y$.

$Next$ is the 'predecessor-successor' relation which satisfies the seriality property: for any $i \in Lab$, there exists $j \in Lab$ such that $Next(i, j)$.

Let $'$ abbreviate the operation which being applied to $i \in Lab$ gives us $i' \in Lab$ such that $Next(i, i')$.

14th International Symposium on Temporal Representation and Reasoning (TIME'07
0-7695-2836-8/07 $20.00 © 2007 IEEE
IEEE COMPUTER SOCIETY

As we have already mentioned above, now we are able to introduce the expressions representing the properties of relations '$\preceq$', '$\prec$', '$\simeq$' and 'Next', and the operation $'$ which, following [17], we call *relational judgements*.

Definition 4 (PLTL$_{\mathbf{ND}}$ Syntax)

- *If A is a* PLTL *formula and $i \in Lab$ then $i : A$ is a* PLTL$_{\rm ND}$ *formula.*
- *Any relational judgement of the type $Next(i, j)$, $i \preceq j$, $i \prec j$ and $i \simeq j$ is a* PLTL$_{\rm ND}$ *formula.*

Some useful and rather straightforward properties relating operations on labels are given below.

(2.5) For any $i, j \in Lab$ if $Next(i, j)$ then $i \preceq j$.

(2.6) For any $i, j \in Lab$ if $i \prec j$ then $i \preceq j$.

PLTL$_{\mathbf{ND}}$ Semantics. For the interpretation of PLTL$_{\rm ND}$ formulae we adapt the semantical constructions defined in §2.1 for the logic PLTL. In the rest of the paper we will use capital letters $A, B, C, D, \ldots$ as metasymbols for PLTL formulae, and calligraphic letters $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D} \ldots$ to abbreviate formulae of PLTL$_{\rm ND}$, i.e. either labelled formulae or relational judgements. The intuitive meaning of $i : A$ is that A is satisfied at the world i.

Let Γ be a set of PLTL$_{\rm ND}$ formulae, let $D_\Gamma = \{x | x : A \in \Gamma\}$, let σ be a model as defined in §2.1 and let f be a function which maps elements of D_Γ into $\mathcal{N}$ (recall that a PLTL model σ is isomorphic to natural numbers).

Definition 5 (Realisation of PLTL$_{\mathbf{ND}}$ formulae in a model) *Model σ* realises *a set, Γ, if there is a mapping, f, which satisfies the following conditions.*

(1) For any $x \in D_\Gamma$, and for any A, if $x : A \in \Gamma$ then $\langle \sigma, f(x) \rangle \models A$,

(2) For any x, y, if $x \preceq y \in \Gamma$, and $f(x) = i$, and $f(y) = j$ then $i \leq j$,

(3) For any x, y, if $Next(x, y) \in \Gamma$, and $f(x) = i$, and $f(y) = j$ then $j = i + 1$.

The set Γ in this case is called realisable.

Definition 6 (PLTL$_{\mathbf{ND}}$ Validity) *A well-formed* PLTL$_{\mathbf{ND}}$ *formula, $\mathcal{A} = i : B$, is valid (abbreviated as $\models_{ND} \mathcal{A}$) if, and only if, the set $\{\mathcal{A}\}$ is realisable in every possible model, for any function f.*

Rules of Natural Deduction System.

In Figure 2 we define these sets of elimination and introduction rules, where prefixes '*el*' and '*in*' abbreviate an elimination and an introduction rule, respectively.

Elimination Rules :

$$\wedge\, el_1 \quad \frac{i : A \wedge B}{i : A} \qquad\qquad \wedge\, el_2 \quad \frac{i : A \wedge B}{i : B}$$

$$\Rightarrow el \quad \frac{i : A \Rightarrow B, \quad i : A}{i : B} \qquad\qquad \neg\, el \quad \frac{i : \neg\neg A}{i : A}$$

$$\vee\, el \quad \frac{i : A \vee B, \quad i : \neg A}{i : B}$$

Introduction Rules :

$$\vee\, in_1 \quad \frac{i : A}{i : A \vee B} \qquad\qquad \vee\, in_2 \quad \frac{i : B}{i : A \vee B}$$

$$\wedge\, in \quad \frac{i : A, \quad i : B}{i : A \wedge B} \qquad\qquad \Rightarrow\, in \quad \frac{[i : C], \quad i : B}{i : C \Rightarrow B}$$

$$\neg\, in \quad \frac{[j : C], \quad i : B, \quad i : \neg B}{j : \neg C}$$

Figure 2. PLTL$_{\mathbf{ND}}$-rules for Booleans

- In the formulation of the rules '$\Rightarrow in$' and '$\neg\ in$' formulae $[i : C]$ and $[j : C]$ respectively must be the most recent non discarded [4] assumption occurring in the proof. When we apply one of these rules on step n and discard an assumption on step m, we also discard all formulae from m to $n - 1$. We will write $[m$ - $(n-1)]$ to indicate this situation.

Now, we add an additional rule which is deeply involved into our searching procedure.

$$\neg\vee \quad \frac{i : \neg(A \vee B)}{i : \neg A \wedge \neg B}$$

This rule simply represents one of De Morgan laws and is derivable from the set of classical rules mentioned above. Hence, it is a technical addition, connected with the searching procedure.

We keep the notions of *flagged* and *relatively flagged* label with the meaning similar to the notions of flagged and relatively flagged variable in first order logic [4]. By saying that the label, j, is flagged, abbreviated as $\mapsto j$, we mean that it is bound to a state and, hence, cannot be rebound to some other state. By saying that a variable i is relatively flagged (bound) by j, abbreviated as $j \mapsto i$ we mean that a bounded variable, j, restricts the set of runs for i that is linked to it in the relational judgment, for example $i \preceq j$.

Now in Figure 3 we introduce the following rules to manipulate with relational judgements which correspond to the properties (2.1)-(2.6).

reflexivity $\quad \dfrac{}{i \preceq i}$ $\qquad$ $\bigcirc$ *seriality* $\quad \dfrac{}{Next(i,i')}$

$\prec / \preceq \quad \dfrac{i \prec j}{i \preceq j}$ $\qquad$ $\bigcirc / \preceq \quad \dfrac{Next(i,i')}{i \preceq i'}$

transitivity $\quad \dfrac{i \preceq j,\ j \preceq k}{i \preceq k}$

$\preceq$ *linearity* $\quad \dfrac{i \preceq j,\ i \preceq k}{(j \preceq k) \vee (j \simeq k) \vee (k \preceq j)}$

Figure 3. PLTL$_{\rm ND}$-rules for relational judgements

The linearity rule needs some additional comments. Strictly speaking, in the PLTL$_{\rm ND}$ language, to avoid unnecessary complications, we do not allow either Boolean combination of relational judgements or their negations. Obviously, the conclusion of the $\preceq$ linearity rule violates this constraint. However, it expresses an obvious property of the linear time model structure and to make our presentation more transparent we explicitly formulate a corresponding rule. Our justification here is very simple: the only way in which the conclusion of this rule is involved into the construction of the proof is reasoning by cases - see more details in the discussion of the relevant searching rule (5.2) in §3.1.

Next, in Figures 4 and Figures 5 we define elimination and introduction rules for the temporal logic operators and the induction rule.

$\star$ When applying $\bigcirc_{el}$ the conclusion $i' : A$ becomes marked by M_1. This affects other rules:

- the condition $\forall C(j : C \notin M1)$ in the rules $\Diamond_{el}$, $\mathcal{U}_{el_1}$ means that the label j should not occur in the proof in any formula, $j{:}C$, that is marked by $M1$,

- the condition $j{:}A \notin M1$ in the rule $\Box_{in}$ means that $j{:}A$ is not marked by $M1$.

$\star\star$ In $\mathcal{U}_{el2}$ the expression $i^{[AB]}$ is used with the following meaning: a variable i in the proof can be marked with $[AB]$ if it has been introduced in the proof as a result of the application of the rule $\mathcal{U}_{el1}$ to $i{:}A\mathcal{U}B$.

$\star\star\star$ In $\Box_{in}$ and the induction rules formula $i \preceq j$ must be the most recent assumption and a variable j is new in a derivation; applying the rule on the step n of the proof, we discard $i \preceq j$ and all subsequent formulae until the step n.

Finally, we add two more rules which are also deeply involved into our searching procedure.

$\Box_{el} \quad \dfrac{i{:}\Box A, \quad i \preceq j}{j{:}A}$

$\Diamond_{el} \quad \dfrac{i{:}\Diamond A}{i \preceq j, \quad j{:}A}$ $\quad \Big|\ \begin{array}{l}\forall C(j{:}C \notin M1)\\ \mapsto j,\ j \mapsto i\end{array}$

$\bigcirc el^{\star} \quad \dfrac{i{:}\bigcirc A}{i'{:}A}$ $\quad \big|\ i'{:}A \in M1$

$\mathcal{U}_{el1} \quad \dfrac{i{:}A\mathcal{U}B, \quad i{:}\neg B}{i{:}A, \quad j{:}B, \quad i \prec j}$ $\quad \Big|\ \begin{array}{l}\forall C(j{:}C \notin M1)\\ \mapsto j,\ j \mapsto i\end{array}$

$\mathcal{U}^{\star\star}_{el2} \quad \dfrac{i^{[AB]} \preceq j^{[AB]}, i^{[AB]} \preceq k, k \prec j^{[AB]}}{k{:}A}$

Figure 4. Elimination rules for temporal operators

$\Box in^{\star\star\star} \quad \dfrac{j{:}A, \quad [i \preceq j]}{i{:}\Box A}$ $\quad \Big|\ \begin{array}{l}j{:}A \notin M1\\ \mapsto j,\ j \mapsto i\end{array}$

$\Diamond_{in} \quad \dfrac{i{:}A}{i{:}\Diamond A}$ $\qquad$ $\bigcirc_{in} \quad \dfrac{i'{:}A, \quad Next(i,i')}{i{:}\bigcirc A}$

$\mathcal{U}_{in} \quad \dfrac{i{:}B}{i{:}A\mathcal{U}B}$

Induction***

$\dfrac{i{:}A \quad [i \preceq j] \quad j{:}A \Rightarrow \bigcirc A}{i{:}\Box A}$ $\quad \Big|\ \begin{array}{l}j{:}A \notin M1\\ \mapsto j,\ j \mapsto i\end{array}$

Figure 5. Introduction rules for temporal operators

$\neg\mathcal{U} \qquad \dfrac{i : \neg(A\mathcal{U}B)}{i : \ \Box\neg B \vee \neg B\mathcal{U}(\neg A \wedge \neg B)}$

$\neg\Diamond \qquad \dfrac{i : \neg\Diamond A}{i : \ \Box\neg A}$

While the first rule, $\neg\mathcal{U}$ is not derivable from the set of rules for temporal operators given above, the second rule, $\neg\Diamond$ can be easily derived from them. However, the addition of these rules as part of the main rules of the system significantly simplifies our searching procedure. Note also that together with the use of fewer introduction rules for $\mathcal{U}$, it also leads us to a new ND formulation of PLTL.

Definition 7 (PLTL$_{\rm ND}$ proof) *An ND proof of a* PLTL *formula B is a finite sequence of* PLTL$_{\rm ND}$ *formulae*

$\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_n$ which satisfies the following conditions:

- *every $\mathcal{A}_i$ $(1 \leq i \leq n)$ is either an assumption, in which case it should have been discarded, or the conclusion of one of the ND rules, applied to some foregoing formulae,*
- *the last formula, $\mathcal{A}_n$, is $x\!:\!B$, for some label x,*
- *no variable - world label is flagged twice or relatively binds itself.*

When B has a PLTL$_{\mathbf{ND}}$ proof we will abbreviate it as $\vdash_{ND} B$ indicating that B is a theorem.

Theorem 1 *[*PLTL$_{\mathbf{ND}}$ *Soundness]*

PROOF: The new rule, $\neg\mathcal{U}$, introduced into the system, is based on the similar equivalence, $\neg A \mathcal{U} B \equiv \Box\neg B \vee \neg B \mathcal{U} (\neg A \wedge \neg B)$. Similarly, the second rule, $\neg\Diamond$, is based on the following famous equivalence relating the $\Box$ and $\Diamond$ operators $\neg\Diamond A \equiv \Box\neg A$. It is an easy exercise to show that both of these new rules preserve satisfiability. This, together with the soundness theorem of the original formulation of the ND system in [2], gives us the soundness of the new formulation. (END)

Theorem 2 *[*PLTL$_{\mathbf{ND}}$ *Completeness]*

PROOF: We can also show that with the addition of the new rules, $\neg\Diamond$ and $\neg\mathcal{U}$, we are able to prove all the theorems of the logic PLTL. This completeness proof would be very similar to that contained in [2] being different only in establishing the fact that all the axioms of PLTL are derivable in a new system with these new rules. (END)

3 ND-proof Searching Technique

The proof searching strategy is *goal-directed*. The core idea behind it is the creation of the two sequences of formulae: *list_proof* and *list_goals*. The first sequence represents formulae which form a proof. In the second sequence we keep track of the list of goals. Here, each goal is either a formula or two arbitrary contradictory formulae. We will abbreviate this designated type of goal by $\bot$. An *algo-proof* is a pair (*list_proof*, *list_goals*) whose construction is determined by the searching procedure described below. At each step of constructing an algo-proof, a specific goal is chosen, which we aim to reach at the *current stage*. Thus, the appropriate name for such a goal would be a *current_goal*. The first goal of *list_goals* is extracted from the given task, we will refer to this goal as to the *initial goal*.

Definition 8 (Reachability of a current goal) *A current goal, G_n, $0 \leq n$, occurring in* list_goals= *$\langle G_1, G_2, \dots, G_n\rangle$, is reached if the following conditions are satisfied:*

- *If $G_n \neq \bot$ then G_n is reached if there is a formula $\mathcal{A}$ in* list_proof *such that $\mathcal{A}$ is not discarded and $\mathcal{A} = G_n$,* else
- *G_n is $\bot$ and it is reached if*
  - *there are two non-discarded contradictory formulae $i\!:\!A$ and $i\!:\!\neg A$ (for some i) in* list_proof.
  - *$\bot$ is derived in each of three tasks following Procedure (2.2.6), reasoning by cases (see §3.1).*

In general, when we construct a proof, we check whether the *current_goal* has been reached. If it has been reached then we apply the appropriate introduction rule, and this is *the only reason* for the application of introduction rules. As we will see later, such an application of an introduction rule is absolutely *determined* by the structure of the previous goal and by the formulation of introduction rules. Alternatively, (if the *current_goal* has not been reached), we continue searching for a possible update of *list_proof* and *list_goals*. Note that the construction of these sequences is determined either by the structure of the *current_goal*, or by the structure of complex formulae occurring within *list_proof*. Additionally, we introduce a mechanism of marking formulae within *list_proof* and *list_goals* to prevent an infinite application of searching rules.

Now we describe a set of *procedures* which guide the construction of an algo-proof.

3.1 Proof-Searching Procedures

Procedure 1. Here we update a sequence *list_proof* by searching (in a breadth-first manner) for an applicable elimination rule, $\neg\Diamond$, $\neg\mathcal{U}$ or $\neg\vee$ rules. If we find a formula, or two formulae, which can serve as premises of one of these rules, the rule is enforced and the sequence *list_proof* is updated by the conclusion of the rule. We apply these rules in the following order: rules to eliminate a Boolean operation, $\neg\Diamond$, $\neg\mathcal{U}$, $\neg\vee$, $\mathcal{U}_{el}$, $\Diamond_{el}$, and, finally, $\Box_{el}$. Note that $\Box_{el}$ applies to some formula $i : \Box A$ any time when the new label j appears in the *list_proof* such that $i \leq j$.

Procedure 2. Here a new goal is synthesized in a backward chaining style following one of the subprocedures described below. They apply when Procedure 1 terminates, i.e. when no elimination rule can be applied, and the current goal, G_n, is not reached. The type of G_n determines *how* the sequences *list_proof* and *list_goals* must be updated.

Procedure 2.1. This procedure is invoked when G_n is not $\bot$. Here we update sequences *list_proof* and *list_goals* analysing the structure of G_n. Let *list_proof* = Γ and *list_goals* = $G_1, \ldots, G_n$, where G_n is the current goal. Given that G_n is not reachable, then looking at its structure, we derive a new goal G_{n+1} and set the latter as the current goal. Below we identify various cases of applying Procedure 2.1, where $G_n = i\!:\!A | i\!:\!\neg A | i\!:\!A \wedge B | i\!:\!A \vee B | i\!:\!A \Rightarrow B | i : \Box A | i : \Diamond A | i : \bigcirc A | i : A \mathcal{U} B$, here i is some label and A, B are any PLTL formulae. Let $G_1, \ldots, G_{n-1} = \Delta$. The '$\longrightarrow$' in the rules below indicates that some given task $\Gamma \vdash \Delta, \ldots$ on its left hand side generates a new task on its right hand side.

(2.1.1)*
$$\Gamma \vdash \Delta, i\!:\!A \quad \longrightarrow \Gamma, i\!:\!\neg A \vdash \Delta, i\!:\!A, \bot$$

(2.1.2)
$$\Gamma \vdash \Delta, i\!:\!\neg A \quad \longrightarrow \Gamma, i\!:\!A \vdash \Delta, i\!:\!\neg A, \bot$$

(2.1.3)
$$\Gamma \vdash \Delta, i\!:\!A \wedge B \longrightarrow \Gamma \vdash \Delta, i\!:\!A \wedge B, i\!:\!B, i\!:\!A$$

(2.1.4.1)**
$$\Gamma \vdash \Delta, i\!:\!A \vee B \longrightarrow \Gamma \vdash \Delta, i\!:\!A \vee B, i\!:\!A$$

(2.1.4.2)**
$$\Gamma \vdash \Delta, i\!:\!A \vee B \longrightarrow \Gamma \vdash \Delta, i\!:\!A \vee B, i\!:\!B$$

(2.1.5)
$$\Gamma \vdash \Delta, i\!:\!A \Rightarrow B \longrightarrow \Gamma, i\!:\!A \vdash \Delta, i\!:\!A \Rightarrow B, i\!:\!B$$

(2.1.6)***
$$\Gamma \vdash \Delta, i\!:\!\Box A \quad \longrightarrow \Gamma, i \preceq j \vdash \Delta, j\!:\!A$$

(2.1.7)****
$$\Gamma \vdash \Delta, i\!:\!\Diamond A \quad \longrightarrow \Gamma \vdash \Delta, i\!:\!A$$

(2.1.8)
$$\Gamma \vdash \Delta, i\!:\!\bigcirc A \quad \longrightarrow \Gamma, Next(i, i') \vdash \Delta, i'\!:\!A$$

(2.1.9)****
$$\Gamma \vdash \Delta, i\!:\!A \mathcal{U} B \longrightarrow \Gamma \vdash \Delta, i\!:\!B$$

$\star$ Procedure (2.1.1) applies when A is either a propositional variable or has a form $B \vee C$, $\Diamond B$ or $B \mathcal{U} C$ and Procedures (2.1.4), (2.1.7) and (2.1.9) have been already applied; additionally, A itself should not have been previously set up as a goal appearing due to Procedure 2.2 (see below).

$\star\star$ Searching rule (2.1.4.2) applies when rule (2.1.4.1) fails, i.e. when applying Procedure (2.1.4.1), we have not managed to reach A (the left disjunct of the goal $A \vee B$) in which case the subroutine invoked into this attempt is deleted and Procedure (2.1.4.2) is fired. In both cases we require to terminate the subroutine if it fails to derive a goal A or B straightforwardly using the elimination rules.

$\star\star\star$ In the Procedure (2.1.6) j is a new variable and it is absolutely flagged and i is relatively flagged.

$\star\star\star\star$ In Procedures (2.1.7) and (2.1.9) if we cannot derive goals $i : A$ and $i : B$ straightforwardly using the elimination rules, then we delete these goals.

As we mentioned above, if applying Procedure (2.1.4) we could not reach goals $i : A,\ i : B$ then we delete these goals, leaving the current goal, $i\!:\!A \vee B$. Similarly, when applying Procedures (2.1.7) and (2.1.9), if we could not reach $i\!:\!A$ or $i\!:\!B$, respectively, we delete these goal and are left with the current goals $i\!:\!\Diamond A$ in case of (2.1.7) and $A \mathcal{U} B$ in case of (2.1.9). Since in each of these cases current goals are not reached, applying Procedure (2.1.1), we would put $\neg(A \vee B)$, $\neg\Diamond A$ or $\neg(A \mathcal{U} B)$ as a new assumption and $\bot$ as a new goal with the subsequent application of $\neg\vee$, $\neg\Diamond$ or $\neg\mathcal{U}$ rule as part of Procedure 1.

Procedure 2.2. This procedure is invoked when G_n is $\bot$. It searches for those formulae in *list_proof* which can serve as sources for new goals. If such a formula is found then its structure will determine the new goal to be generated. Below by Γ, Ψ we understand a list of formulae in *list_proof* with the designated formula Ψ which is considered as a source for new goals. The idea behind this procedure is to search for "missing" premises to apply a relevant elimination rule to Ψ.

(2.2.1)*
$$\Gamma, i\!:\!\neg A \vdash \Delta, \bot \quad \longrightarrow \Gamma, i\!:\!\neg A \vdash \Delta, \bot, i\!:\!A$$

(2.2.2)**
$$\Gamma, i\!:\!A \vee B \vdash \Delta, \bot \longrightarrow \Gamma, i\!:\!A \vee B \vdash \Delta, \bot, i\!:\!\neg A$$

(2.2.3)**
$$\Gamma, i\!:\!A \Rightarrow B \vdash \Delta, \bot \longrightarrow \Gamma, i\!:\!A \Rightarrow B \vdash \Delta, \bot, i\!:\!A$$

(2.2.4)**
$$\Gamma, i\!:\!A \mathcal{U} B \vdash \Delta, \bot \longrightarrow \Gamma, i\!:\!A \mathcal{U} B \vdash \Delta, \bot, i\!:\!\neg B$$

(2.2.5)***
$$\begin{aligned}\Gamma, i\!:\!A \vdash \Delta, \bot \longrightarrow \Gamma, i\!:\!A, i \preceq j \vdash \Delta,\ \bot,\ i : \Box A,\\ j\!:\!A \Rightarrow \bigcirc A\end{aligned}$$

(2.2.6)****
$$\begin{aligned}\Gamma, Lin(i,j) \vdash \Delta, \bot \longrightarrow & \Gamma, Lin(i,j),\ i \preceq j \vdash \Delta,\ \bot\\ & \Gamma, Lin(i,j),\ i \simeq j \vdash \Delta,\ \bot\\ & \Gamma, Lin(i,j),\ j \preceq i \vdash \Delta,\ \bot\end{aligned}$$

$\star$ Applying the Procedure (2.2.1) we have $\neg A$ in the proof and are aiming to derive, A itself. If we are successful then this would give us a contradiction.

$\star\star$ When we apply Procedures (2.2.2-2.2.4), our target is to derive formulae that being in the proof would enable us to apply a relevant elimination rule, $\vee_{el}, \Rightarrow_{el}$ or $\mathcal{U}_{el_1}$.

$\star\star\star$ Procedure (2.2.5) applies after (2.2.1)-(2.2.4) and only when there is at least one formula with the outer $\Box$

in *list_proof*. Applying this procedure, we aim at finding the conditions which would enable us to apply the induction rule. Thus, the side conditions here require that A in $i : A$ does not have $\Box$ as its main operator and $\mapsto j,\ j \mapsto i$.

$\star\star\star\star$ $Lin(i,j)$ abbreviates the linearity constraint $(i \preceq j) \vee (i \simeq j) \vee (j \preceq i)$. This searching rule represents reasoning by cases. If a linearity constraint is introduced into the proof and the current goal is $\bot$ then the rule requires to derive $\bot$ making each of the disjuncts of the linearity constraints in turn as a new assumption. See also comments to Procedure (5.2) below.

Procedure 3. This procedure checks reachability of the current goal in the sequence *list_goals*. If, according to Definition 8, the current goal G_n is reached then the sequence *list_goals* is updated by deleting G_n and setting G_{n-1} as the current goal.

Procedure 4. Procedure 4 indicates that one of the introduction ND-rules, i.e. a rule which introduces a logical connective or a temporal operator, must be applied. We will see below that any application of the introduction rule is completely determined by the current goal of the sequence of goals. This property of our proof searching technique protects us from inferring by introduction rules an infinite number of formulae in *list_proof*.

Procedure 5. This procedure regulates our business with relational judgements.

(5.1) Relational judgements, $i \preceq i$, are introduced into *list_proof* immediately after the introduction of any new label i.

(5.2) Any time when *list_proof* contains two statements $i \preceq j$ and $i \preceq k$, we derive the linearity constraint $(j \preceq k) \vee (j \simeq k) \vee (k \preceq j)$.

(5.3) Any time when *list_proof* contains two statements $i \preceq j$ and $j \preceq k$, we derive the transitivity constraint $i \preceq k$.

(5.4) Any time when *list_proof* contains two statements $Next(i, i')$ and $i \prec j$, we apply the $\bigcirc\ seriality$ and $\prec / \preceq$ rules deriving $i \preceq i'$ and $i \preceq j$. Note that the $\bigcirc$ seriality constraints are introduced into *list_proof* by Procedure (2.1.8).

Procedure (5.2) needs additional comments. As we mentioned, in the PLTL$_{\mathrm{ND}}$ language we do not allow either Boolean combination of relational judgements or their negations. However, to express an obvious property of the linear time model structure and to make our presentation more transparent, we explicitly formulated a corresponding, linearity, rule in §2. Procedure (5.2) introduces the corresponding linearity constraint $(j \preceq k) \vee (j \simeq k) \vee (k \preceq j)$. However, the only way, in which this constraint is involved into the proof is the subsequent application of reasoning by cases, Procedure (2.2.6) where, informally speaking, we split the current task to derive $\bot$ making each of the components of the linearity constraint as a new assumption. If we successful in doing this then, by reasoning by cases, $\bot$ is derivable from the linearity constraint itself. Hence, since the linearity is the property of any linear model, corresponding to our interpretation of the formulae in *list_proof*, we will mark $\bot$ as reached. An obvious reasoning rule used here would be to derive $j : A$ from $i : A$ and $i = j$ for any $i, j \in Lab$.

Now we are ready to describe a searching algorithm, specifying the application of the procedures above.

3.2 Proof-Searching Algorithm $\mathbf{PLTL}^{\mathbf{alg}}_{\mathbf{ND}}$

Let us explain, schematically, the performance of the proof-searching algorithm by describing its major components. These components correspond to the searching procedures presented in §3.1.

Given a task $\vdash G$, we commence the algorithm by setting the initial goal, $G_0 = G$. Then for any goal G_i $(0 \leq i)$, apply Procedure 3, to check if G_i is reached. If G_i is not reached we apply Procedure 1 and Procedure 5, obtaining all possible conclusions of the elimination rules to obtain G_i. If we fail, then Procedure 2 is invoked, and, dependent on the structure of the goal G_i the sequence *list_proof* is updated by adding new assumptions and the sequence *list_goals* by adding new goals. If the *current_goal* is reached, then we determine which introduction rules are to be applied. Otherwise, which could only be in the case, when *current_goal* is set as $\bot$ and we do not have contradictory formulae in *list_proof*, we update *list_goals* looking for possible sources of new goals in *list_proof*. We continue searching until either we reach the initial goal, G_0, in which case we terminate having found the desired proof, or until *list_proof* and *list_goals* cannot be updated any further. In the latter case we terminate, and no proof has been found and a (finite) counterexample can be extracted.

Before formulating the main stages of the proof-searching algorithm we have to describe our *marking* technique which introduces and eliminates special marks for formulae in *list_proof* and *list_goals*. Most of these marks are devoted to prevent looping either in application of elimination rules or in searching. Thus, we mark:

- formulae that were used as premisses of the rules invoked in Procedures 1 and 5;
- goals $A \vee B$, $\Diamond A$ and $A\,\mathcal{U}\,B$ in Procedures (2.1.4), (2.1.7) and (2.1.9) respectively to allow deletion of the subsequent goals, see comments $\star\star$ and $\star\star\star\star$ to these rules;

- those formulae in *list_proof* which were considered as sources of new goals in Procedure 2.2 and these new goals themselves to prevent looping in Procedure (2.1.1) - see comments $\star$ to this procedure; note that if a formula A has generated a goal B in this way, but later B has been reached, hence discarded, from the proof, we get rid of the mark for A allowing to consider this formula again as a source of new goals;
- the label of the conclusion of the $\bigcirc_{el}$ rule to preserve satisfiability in rules $\Diamond_{el}$, $\Box_{in}$ and the induction rule.

Formulation of the algorithm.

(1) Given a task $\vdash G$, we consider G as the *initial goal* of the proof and write G into *list_goals*. If the set of given assumptions in Γ is not empty then these assumptions are written in a *list_proof*. Set *current_goal* = G, *go to* (2).

(2) Apply Procedure 3 (analysis of the reachability of the current goal, G_n).

(2a) If G_n is reached then *go to* (3) else

(2b) if elimination rules are applicable *go to* (4) else
(if no more elimination rules are applicable) *go to* (5) else
(if no more rules from Procedure 5 applied to relational judgements are applicable) *go to* 6.

(3) Based on the structure of the goal reached

(3a) If G_n (reached) is the initial goal then *go to* (7a) else

(3b) (G_n is reached and it is not the initial goal). Apply Procedure 4 (which invokes introduction rules), *go to* 2.

(4) Apply Procedure 1 , *go to* (2).

(5) Apply Procedure 5, *go to* (2).

(6) Apply Procedure 2.

(6a) If $G_n \neq \bot$ then apply Procedure 2.1 (analysis of the structure of G_n), *go to* (2) else

(6b) Apply Procedure 2.2 (searching for the sources of new goals in *list_proof*), *go to* (2) else

(6c) (if all formulae in *list_proof* are marked, i.e. have been considered as sources for new goals), *go to* (7b).

(7) Termination

(7a) The desired ND proof has been found. EXIT,

(7b) No ND proof has been found. EXIT.

3.3 Example Proof

Now we give an example of the PLTL$_{\mathbf{ND}}$ algo-proof establishing that the following formula is a theorem.

$$\Box(p \Rightarrow \bigcirc p) \Rightarrow (p \Rightarrow \Box p) \tag{1}$$

We will provide sufficient comments explaining how the main parts of the searching procedure, *list_proof* and *list_goals* are constructed. We will also split the construction of the PLTL$_{\mathrm{ND}}^{\mathrm{alg}}$ into stages to ease the understanding of the techniques applied.

The proof starts with setting a formula $x : \Box(p \Rightarrow \bigcirc p) \Rightarrow (p \Rightarrow \Box p)$ as the main goal, G_0, in the *list_goals*. Consecutive applications of Procedure (2.1.5) to this formula result in adding new assumptions $x : \Box(p \Rightarrow \bigcirc p)$ and $x : p$ in the *list_proof*. On the second step the current goal is $x : \Box p$, so we apply Procedure (2.1.6) to obtain the current goal $y : p$ where y must be flagged and x must be relatively flagged. At the same time a new assumption $x \preceq y$ is added to the *list_proof* at step 3. Here we apply Procedure 1 inferring $y : p \Rightarrow \bigcirc p$ at step 4. However, the current goal, $y : p$, is still non reachable, hence, by Procedure (2.1.1), the current goal is $\bot$ with adding at step 5 a new assumption $y : \neg p$. Thus, we have

$list\_proof$	$analysis$	$list\_goals$
		G_0
1. $x : \Box(p \Rightarrow \bigcirc p)$	$assum.$	$G_0,\ G_1 = (x : p \Rightarrow \Box p)$
2. $x : p$	$assum.$	$G_0,\ G_1, x : \Box p$
		$G_0,\ G_1, x : \Box p, y : p$
3. $x \preceq y$	$assum.$	$\mapsto y,\ y \mapsto x$
4. $y : p \Rightarrow \bigcirc p$	$\Box_{el}, 1, 3$	
5. $y : \neg p$	$assum.$	$G_0,\ G_1, x : \Box p, y : p, \bot$

The current goal, $\bot$, is non-reachable. Here the algorithm applies Procedure (2.2) searching for non-discarded formulae in the *list_proof*. Note, that $x : \Box(p \Rightarrow \bigcirc p)$ has been marked since the $\Box_{el}$ rule was applied to it. The first formula to apply Procedure (2.2) is $y : p \Rightarrow \bigcirc p$ (step 4) giving us the new goal, $y : p$. It is not reachable, hence by Procedure (2.1.1), the new assumption is $y : \neg p$ (step 6) and the new goal is $\bot$. At this moment all complex formulae are marked, but an $\Box$-formula is in the *list_proof* and the algorithm is looking for a prospective application of the induction rule by Procedure (2.2.5). Thus, we update the *list_goals* with $x : \Box p$, $z : p \Rightarrow \bigcirc p$, where z is flagged and x is relatively flagged in *list_goals*. Additionally, a new assumption, $x \preceq z$, is added at step 7 allowing us to infer $z : p \Rightarrow \bigcirc p$ at step 8. Therefore, $z : p \Rightarrow \bigcirc p$ is reached and is discharged from the *list_goals*. The current goal now is $x : \Box p$ which is reachable via the induction rule (step 9) requiring to discard formulae 7-8 from *list_proof* and flag variables.

$list\_proof$	$analysis$	$list\_goals$
		$G_0,\ G_1, x\colon \Box p, y\colon p, \bot, y\colon p,$
$6.\ y\colon \neg p$	$assum.$	$G_0,\ G_1, x\colon \Box p, y\colon p, \bot, y\colon p,$
		$\bot$
		$G_0,\ G_1, x\colon \Box p, y\colon p, \bot, y\colon p,$
		$\bot, x\colon \Box p, z\colon p \Rightarrow \bigcirc p$
$7.\ x \preceq z$	$assum.$	$\mapsto z,\ z \mapsto x$
$8.\ z\colon p \Rightarrow \bigcirc p$	$\Box_{el}\ 1, 7$	$G_0,\ G_1, x\colon \Box p, y\colon p, \bot, y\colon p,$
		$\bot, x\colon \Box p$
$9.\ x\colon \Box p$	$induction$	
	$2, 7, 8,$	
	$[7 - 8]$	
	$\mapsto z$	
	$z \mapsto x$	$G_0,\ G_1, x\colon \Box p, y\colon p, \bot, y\colon p,$
		$\bot$

The current goal is now $\bot$ which is easily reached: apply the $\Box_{el}$ rule to formulae 3 and 9 deriving 10 which is contradiction with 6. Hence we reached the goal $\bot$, and apply the $\neg_{in}$ rule to 6 and 10 deriving step 11 and discarding formulae 6-10 from *list_proof*. The current goal, $y : p$, is reached at step 12 by eliminating double negation from 11. The current goal now is $\bot$, and we apply the $\neg_{in}$ rule to steps 5 and 12, deriving 13 and discarding formulae 5-12 from *list_proof*. The current goal $y : p$ is reachable by $\neg_{el}$ from 13. Now $x : \Box p$ is reachable by applying the $\Box_{in}$ rule to 3 and 14 deriving 15 and discarding formulae 3-14.

$list\_proof$	$analysis$	$list\_goals$
$10.\ y\colon p$	$\Box_{el}, 3, 9$	$G_0,\ G_1, x\colon \Box p, y\colon p, \bot, y\colon p, \bot$
$11.\ y\colon \neg\neg p$	$\neg_{in}, 6, 10,$	
	$[6 - 10]$	$G_0,\ G_1, x\colon \Box p, y\colon p, \bot, y\colon p$
$12.\ y\colon p$	$\neg_{el}, 11$	$G_0,\ G_1, x\colon \Box p, y\colon p, \bot$
$13.\ y\colon \neg\neg p$	$\neg_{in}, 5, 12,$	
	$[5 - 12]$	$G_0,\ G_1, x\colon \Box p, y\colon p$
$14.\ y\colon p$	$\neg_{el}, 13$	$G_0,\ G_1, x\colon \Box p$
$15.\ x\colon \Box p$	$\Box_{in}, 3, 14,$	
	$[3 - 14],$	
	$\mapsto y,$	
	$y \mapsto x$	$G_0,\ G_1 = (x\colon p \Rightarrow\ \Box p)$

The last steps are obvious applications of the $\Rightarrow_{in}$ rule at steps 16 and 17. Note that at step 16 we discard formulae 2-14 while on step 17 we discard in *list_proof* all the remaining formulae, 1-16.

$list\_proof$	$analysis$	$list\_goals$
$16.\ x\colon p \Rightarrow\ \Box p$	$\Rightarrow_{in}, 15,$	
	$[2 - 15]$	G_0
$17.\ x\colon \Box(p \Rightarrow \bigcirc p) \Rightarrow$		
$(p \Rightarrow\ \Box p)$	$\Rightarrow_{in}, 16,$	
	$[1 - 16]$	$G_0\ reached$

The set of non-discarded assumptions is empty, the initial goal is reached, hence we have a desired proof of formula (1). Note that in the presentation of this proof, due to the space limit, we omitted a few steps, such as $\Rightarrow_{el}$ applied to 10 and 4 and a few subsequent steps as they do not contribute to the proof. Observe also that all of our introduction rules were guided by alive goals in the *list_goals*. For example, given that we have reached the goal, $y : p$, at step 14, the structure of the previous goal, $x : \Box p$, determines the subsequent application of the $\Box_{in}$ rule.

An interested reader may wish to compare this algorithmic proof of formula (1) with the manual proof of the same formula in [2]: this would illustrate nicely the necessary complications and even obvious redundancy (at this stage of developing a searching method) introduced by a proof searching routine comparing it with a 'hand-made' proof.

3.4 Correctness

There are three necessary conditions that a proof search procedure for a decidable system should have: termination, soundness and completeness. Being decidable, PLTL encourages researchers at presenting algorithms that effectively tell us if any given input formula is a theorem building up a desired proof or there is an assignment falsifying it, providing a counter-model. Below we will sketch proofs of all these properties of $\mathrm{PLTL}^{\mathrm{alg}}_{\mathrm{ND}}$.

Theorem 3 $\mathrm{PLTL}^{\mathbf{alg}}_{\mathbf{ND}}$ *terminates for any input* PLTL *formula.*

PROOF: For the termination we need to establish that both main sequences, *list_proof* and *list_goals*, that constitute $\mathrm{PLTL}^{\mathrm{alg}}_{\mathrm{ND}}$, are finite, and also that there are no loops in the searching procedure. Two observations are important here. Firstly, the marking technique guarantees the finite number of application of rules in Procedures 1 and 5, and the finite number of formulae that are introduced into *list_proof* and *list_goals* by Procedure 2. Note that our special procedures to deal with the most difficult for the natural deduction cases related to disjunctive goals, namely, with the goals of the type $i : A \vee B$, $i : \Diamond A$ and $i : A \mathcal{U} B$ reflected in the Procedures (2.1.4), (2.1.7) and (2.1.9), prevent us of being involved into loops.

Secondly, any application of an introduction rule is completely determined by the algorithm. Namely, if the current goal is reached and it is a contradiction, $\bot$, then we apply the $\neg_{in}$ rule to the contradictory formulae introducing into the proof the negation of the most recent non-discarded assumption. Any other type of the current goal which is reached required us to consider the previous goal and the corresponding introduction rule is fired. Thus, for example, if the current goal (reached) is $i\ :\ A$ or $i\ :\ B$ and the previous goal is $i\colon\ A \vee B$, see Procedure (2.1.4), then we apply the $\vee_{in}$ rule to either $i\ :\ A$ or $i\ :\ B$ reaching the previous goal, $i : \forall \vee B$ by simply adding the missing component of disjunction. Similarly, if the current goal (reached) is $j : A$ and the previous goal is $i : \Box A$, see Procedure (2.1.6), then we would have introduced into the proof an assumption $i \preceq j$, and thus, the $\Box_{in}$ rule is applied. (END)

Theorem 4 $\mathrm{PLTL}_{\mathbf{ND}}^{\mathbf{alg}}$ *is sound.*

PROOF: Soundness of $\mathrm{PLTL}_{\mathrm{ND}}^{\mathrm{alg}}$ follows immediately from the fact that *list_proof* obtained following the steps of the algorithm is a proof in the calculus $\mathrm{PLTL}_{\mathrm{ND}}$. Hence, if there is an algo-proof of A then a *list_proof* of this algo-proof is a proof of A in the system $\mathrm{PLTL}_{\mathrm{ND}}$. By Theorem 1, $\mathrm{PLTL}_{\mathrm{ND}}$ is sound. Therefore, $\mathrm{PLTL}_{\mathrm{ND}}^{\mathrm{alg}}$ is sound, too. (END)

Lemma 1 *From an exhausted non successful algo-proof for a* $\mathrm{PLTL}_{\mathbf{ND}}$ *formula A we can extract a model which falsifies A.*

PROOF: We adopt a technique developed in many sources, see for example [12]. In an exhausted non successful algo-proof for A the algorithm terminates without finding a proof, having applied all its procedures and with the final goal $\bot$ which is not reached. We can show that in this case *list_proof* contains a set of indexed literals, $i : l$, (l is a propositional variable or its negations) sufficient to construct a model, say σ, which falsifies A: if l is a propositional variable than there is a mapping, f, such that $\langle \sigma, f(i) \rangle \models l$ otherwise, $\langle \sigma, f(i) \rangle \models \neg l$. Procedure 2 plays the main role here: we construct σ from a collection of saturated sets, atoms, along with an accesibility relation. In our case this relation is encoded by a set of relational judgements, while the assignment to the literals as shown above provides us with a set of atoms. (END)

Theorem 5 $\mathrm{PLTL}_{\mathbf{ND}}^{\mathbf{alg}}$ *is complete.*

PROOF: We must show that for every $\mathrm{PLTL}_{\mathrm{ND}}$ valid formula, A, $\mathrm{PLTL}_{\mathrm{ND}}^{\mathrm{alg}}$ finds a $\mathrm{PLTL}_{\mathrm{ND}}$ proof. This is a simple consequence (by contraposition) of Lemma 1. (END)

Termination, soundness and completeness results imply the fundamental property of our algorithm reflected in the following theorem.

Theorem 6 *For any input formula A, the* $\mathrm{PLTL}_{\mathrm{ND}}$ *terminates either building up a* $\mathrm{PLTL}_{\mathrm{ND}}$*-proof for A or providing a counter-model.*

Constructing a counter-model. Let us now illustrate how the algorithm works with a non-provable formula, $\Diamond q \Rightarrow p\,\mathcal{U}\,q$. We commence this proof by setting up the main goal $G_0 = \Diamond q \Rightarrow p\,\mathcal{U}\,q$. Applying Procedure (2.1.5), we update *list_proof* by $x : \Diamond q$ at step 1 and *list_goals* by the new goal $G_1 = x : p\,\mathcal{U}\,q$. Next, we apply $\Diamond_{el}$ rule to formula at step 1 obtaining steps 2 and 3 and setting up required restrictions on the labels x and y. The current goal, G_1 is not reachable, therefore, by Procedure (2.1.1) we continue by refutation updating *list_proof* by the new assumption $G_1 = x : \neg(p\,\mathcal{U}\,q)$ and the new goal $\bot$. At this stage we apply Procedure 1, namely, the $\neg\,\mathcal{U}$ rule to derive step 5 from step 4.

list_proof	*analysis*	*list_goals*
		G_0
1. $x : \Diamond q$	$assump.$	$G_0,\ G_1$
2. $x \preceq y$	$\Diamond_{el}, 1, \mapsto y$	
	$y \mapsto x$	
3. $y : q$	$\Diamond_{el}, 1$	
4. $x : \neg(p\,\mathcal{U}\,q)$	$assump.$	$G_0, G_1, \bot$
5. $x : \Box\neg q \vee (\neg q\,\mathcal{U}\,(\neg p \wedge \neg q))$	$\neg\,\mathcal{U}\,, 4$	

The current goal, $\bot$, is not reachable, hence, by Procedure (2.2.2) we set up the new goal, $x : \neg\Box\neg q$. Again, this is not reachable, therefore, by Procedure (2.1.2) we update *list_proof* by $x : \Box\neg q$ and *list_goals* by the new goal, $\bot$. Next, we apply Procedure 1, namely, $\Box_{el}$ rule to steps 2 and 6, deriving step 7. This gives us the desired contradiction - steps 3 and 7. Hence, we apply the $\neg_{in}$ rule to these formulae obtaining step 8 and discarding formulae 6-7 from *list_proof*.

list_proof	*analysis*	*list_goals*
		$G_0, G_1, \bot, x : \neg\Box\neg q$
6. $x : \Box\neg q$	$assump.$	$G_0, G_1, \bot, x : \neg\Box\neg q, \bot$
7. $y : \neg q$	$\Box_{el}, 2, 6$	
8. $x : \neg\Box\neg q$	$\neg_{in}, 3, 7$	
	$[6 - 7]$	$G_0, G_1, \bot$

Applying Procedure 1, namely, the $\vee_{el}$ rule to 5 and 8, we derive step 9.

list_proof	*analysis*	*list_goals*
9. $x : \neg q\,\mathcal{U}\,(\neg p \wedge \neg q)$	$\vee_{el}, 5, 8$	$G_0, G_1, \bot, x : \Box\neg q$

The current goal is $x : \Box\neg q$, Therefore, by Procedure (2.1.6), we update *list_proof* by the new assumption, $x \preceq z$ and set up the new goal, $z : \neg q$. Note that at this stage, z must be a new flagged variable, and x becomes relatively flagged, which is reflected in our comments in the algo-proof, $\mapsto z, z \mapsto x$. Since we have a new relational judgement added into *list_proof*, we apply Procedure 5 to derive the linearity constraint. Namely, by Procedure (5.2) from 2 and 10 we derive step 11, where $Lin(y, z) = y \preceq z \vee y \simeq z \vee z \preceq y$. The current goal, $z : \neg q$, is not reachable, hence, we apply Procedure (2.1.2) and update *list_proof* by formula 10 setting up the new goal, $\bot$.

14th International Symposium on Temporal Representation and Reasoning (TIME'07)
0-7695-2836-8/07 \$20.00 © 2007 IEEE

$list\_proof$	$analysis$	$list\_goals$
10. $x \preceq z$	$assum.$	$G_0, G_1, \bot, x : \Box\neg q, z : \neg q$ $\mapsto z, z \mapsto x$
11. $Lin(y, z)$	$linearity,$ $2, 10$	
12. $z : q$	$assum.$	$G_0, G_1, \bot, x : \Box\neg q, z : \neg q, \bot$

The current goal, $\bot$ is not reachable, therefore, we apply Procedure 2 looking for the sources for new goals. The first formula which should be considered as the source for new goals, is formula 9, $x : \neg q\,\mathcal{U}\,(\neg p \wedge \neg q)$. Thus, by Procedure (2.2.4), we update *list_goals* by $x : \neg(\neg p \wedge \neg q)$. Now *list_goals* is $G_0, G_1, \bot, x : \Box\neg q, z : \neg q, \bot, x : \neg(\neg p \wedge \neg q)$. The current goal is not reachable, therefore, by Procedure 2.1.2, we update *list_proof* by formula 13 and *list_goals* by the new goal, $\bot$. Procedure 1 ($\wedge_{el}$) will give us steps 14 and 15.

$list\_proof$	$analysis$	$list\_goals$
13. $x : \neg p \wedge \neg q$	$assum.$	$G_0, G_1, \bot, x : \Box\neg q, z : \neg q, \bot,$ $\neg(\neg p \wedge \neg q), \bot$
14. $x : \neg p$	$\wedge_{el}, 13$	
15. $x : \neg q$	$\wedge_{el}, 13$	

Our current goal is $\bot$, no more elimination rules are applicable hence, we apply Procedure 2 looking for the sources for new goals. Procedure 2 finds the linearity constraint at step 11 and apply Procedure (2.2.6) to update *list_proof* with the new assumption, $y \preceq z$ at step 16.

$list\_proof$	$analysis$	$list\_goals$
16. $y \preceq z$	$assumption$	

Note that at step 16 we take the first disjunct of the linearity constraint as a new assumption but already here failed to reach $\bot$, hence, the algorithm terminates having not found the desired proof. This means that the input formula is not valid and we can extract the following counter-model considering the set of literals and relational judgements in *list_proof*. Indeed, consider a model, σ, and a mapping, f, such that $\langle \sigma, f(y) \rangle \models q$, which along with $f(x) \leq f(y)$ gives $\langle \sigma, f(x) \rangle \models \Diamond q$. At the same time $\langle \sigma, f(x) \rangle \not\models p$ and $\langle \sigma, f(x) \rangle \not\models q$, hence $\langle \sigma, f(x) \rangle \not\models p\,\mathcal{U}\,q$. Therefore, $\Diamond q \Rightarrow p\,\mathcal{U}\,q$ is not realisable in the obtained model σ and hence is not valid.

4 Discussion

We have presented $\mathrm{PLTL}_{\mathrm{ND}}$, a proof searching algorithm for the natural deduction formulation of the logic PLTL and established its correctness. To the best of our knowledge, the only other ND construction for linear-time logic [10] does not have a proof searching technique behind it. $\mathrm{PLTL}_{\mathrm{ND}}^{\mathrm{alg}}$ not only enables a full mechanisation of the underlying ND calculus but also allows us to use it as a decision procedure. As we have shown, for any input PLTL formula, $\mathrm{PLTL}_{\mathrm{ND}}^{\mathrm{alg}}$ terminates either finding a proof indicating that the input formula is a theorem, hence valid; otherwise, in case of its termination without a proof, a counter-model can be extracted.

Our approach extends the proof searching technique for classical propositional logic which has been implemented and is available on-line [3]. Most of searching procedures involved into $\mathrm{PLTL}_{\mathrm{ND}}^{\mathrm{alg}}$, as well as the algorithm itself, are structurally similar to those used in the classical logic setting. We believe that this fact reflects the uniform nature of our approach to natural deduction constructions for various logics.

Following the extension of natural deduction to branching-time logic CTL [5], one of the topics for future research would be a corresponding extension of $\mathrm{PLTL}_{\mathrm{ND}}^{\mathrm{alg}}$ to automate the natural deduction representation of this useful logic. Another important part of our future work will be study of complexity of the method and the refinement of the searching technique with the subsequent implementation. We hope that the structural similarity of proof searching algorithms for classical and temporal settings will play here a significant role. Note also, that during the implementation we are planning to embed one of the existing constraint solvers to deal with the algebra of relational judgements.

References

[1] D. Basin, S. Matthews, and L. Viganò. Natural deduction for non-classical logics. *Studia Logica*, 60(1):119–160, 1998.

[2] A. Bolotov, A. Basukoski, O. Grigoriev, and V. Shangin. Natural deduction calculus for linear-time temporal logic. In *Joint European Conference on Artificial Intelligence (JELIA-2006)*, pages 56–68, 2006.

[3] A. Bolotov, V. Bocharov, A. Gorchakov, V. Makarov, and V. Shangin. *Let Computer Prove It.* Logic and Computer. Nauka, Moscow, 2004. (In Russian), Implementation of the proof search technique for classical propositional logic available on-line at `http://prover.philos.msu.ru`.

[4] A. Bolotov, V. Bocharov, A. Gorchakov, and V. Shangin. Automated first order natural deduction. In *Proceedings of IICAI*, pages 1292–1311, 2005.

[5] A. Bolotov, O. Grigoriev, and V. Shangin. Natural deduction calculus for computation tree logic. In *IEEE John Vincent Atanasoff Symposium on Modern Computing*, pages 175–183, 2006.

[6] E. Clarke, S. Jha, and W. R. Marrero. Using state space exploration and a natural deduction style message derivation engine to verify security protocols. In *Proceedings of the IFIP TC2/WG2.2,2.3 International Conference on Programming Concepts and Methods*, pages 87–96, June 1998.

[7] M. Fisher, C. Dixon, and M. Peim. Clausal temporal resolution. *ACM Transactions on Computational Logic (TOCL)*, 1(2):12–56, 2001.

[8] F. Fitch. *Symbolic Logic.* NY: Roland Press, 1952.

[9] D. Gabbay, A. Phueli, S. Shelah, and J. Stavi. On the temporal analysis of fairness. In *Proceedings of 7th ACM Symposium on Principles of Programming Languages*, pages 163–173, Las Vegas, Nevada, 1980.

[10] A. Indrzejczak. A labelled natural deduction system for linear temporal logic. *Studia Logica*, 75(3):345–376, 2004.

[11] S. Jaskowski. On the rules of suppositions in formal logic. In *Polish Logic 1920-1939*, pages 232–258. Oxford Univ. Press, 1967.

[12] O. Lichtenstein and A. Pnueli. Propositional temporal logics: Decidability and completeness. *Logic Journal of the IGPL*, 8(1), 2000.

[13] V. Makarov. Automatic theorem-proving in intuitionistic propositional logic. In *Modern Logic: Theory, History and Applications. Proceedings of the 5th Russian Conference*, StPetersburg, 1998. (In Russian).

[14] F. Pfenning. Logical frameworks. In J. A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, chapter XXI, pages 1063–1147. Elsevier, 2001.

[15] W. Quine. On natural deduction. *Journal of Symbolic Logic*, 15:93–102, 1950.

[16] C. Renteria and E. Haeusler. Natural deduction for CTL. *Bulletin of the Section of Logic, Polish Acad. of Sci.*, 31(4):231–240, 2002.

[17] A. Simpson. *The Proof Theory and Semantics of Intuitionistic Modal Logic.* PhD thesis, College of Science and Engineering, School of Informatics, University of Edinburgh, 1994.

[18] M. Wooldridge. *Reasoning about Rational Agents.* MIT Press, 2000.

14th International Symposium on Temporal Representation and Reasoning (TIME'07
0-7695-2836-8/07 $20.00 © 2007 **IEEE**