A Tableau Decision Procedure for Propositional Intuitionistic Logic

Alessandro Avellone, Guido Fiorino, Ugo Moscato
Dipartimento di Metodi Quantitativi per l'Economia,
Università Milano-Bicocca,Piazza dell'Ateneo Nuovo, 1, 20126 Milano, Italy
{alessandro.avellone,guido.fiorino,ugo.moscato}@unimib.it

November 3, 2006

Abstract

The contribution of this paper consists in some techniques to bound the proof search space in propositional intuitionistic logic. These techniques are justified by Kripke semantics and they are the backbone of a tableau based theorem prover (PITP) implemented in C++ language. PITP and some known theorem provers are compared by the formulas of ILTP v1.1.1 benchmark library. It turns out that PITP is, at the moment, the propositional prover that solves most formulas of the library.

1 Introduction

The development of effective theorem provers for intuitionistic and constructive logics is of interest both for the investigations and applications of such logics to formal software/hardware verification and program synthesis (see i.e. [ML84, Con86, ES99, Men00, FFO02, BC04, AFF+06]).

In this paper we present a strategy and an implementation based on a tableau calculus for propositional intuitionistic logic. Our decision procedure implements the tableau calculus of [AFM04] (this calculus is an enhancement of the calculus given in [Fit69] and it is related to the tableau and sequent calculi of [MMO97, Hud93, Dyc92]).

We introduce some new techniques utilized by our decision procedure to narrow the search space and the width of the proofs. The PSPACE-completeness of intuitionistic validity ([Sta79]) suggests that backtracking and branching cannot be eliminated. In order to improve the time efficiency of the implementations and make them usable, strategies have to be developed to bound backtracking and branching as much as possible.

The optimizations we present are explained by the Kripke semantics for intuitionistic logic. Such semantical techniques are related to the fact that tableau calculi are strictly joined to the semantics of the logic at hand. Loosely speaking, a tableau proof for a formula is the attempt to build a model satisfying the formula. The construction of such a model proceeds by increasing, step by step, the information necessary to define such a model (thus, step by step the accuracy of the model increases). If the proof ends in a contradiction, then there is no model for the formula. Otherwise, a model satisfying

the formula is immediately derived from the proof. With this machinery at hand, first we provide a sufficient condition allowing us to stop a tableau proof without loosing the completeness. Then we describe a technique to bound branching on the formulas which only contain conjunctions and disjunctions. Finally we present a technique to deduce the satisfiability of a set of formulas S, when the satisfiability of a set S' and a permutation τ such that $S = \tau(S')$ are known. Such a technique allows us to bound backtracking. Our technique to bound backtracking is different from the semantical technique provided in [Wei98].

Besides the strategy and its completeness, in the final part of the paper we present some experimental results on the implementation PITP. PITP is written in C++ and it is tested on the propositional part of ILTP v1.1.1 benchmark library ([ROK06]). Of 274 propositional benchmarks contained in ILTP v1.1.1, PITP decides 215 formulas including 13 previously unsolved problems within the time limit of ten minutes. To give the reader more elements to evaluate PITP strategies, comparisons with different versions of PITP are provided.

2 Notation and Preliminaries

We consider the propositional language $\mathcal{L}$ based on a denumerable set of propositional variables or atoms $\mathcal{PV}$ and the logical connectives $\neg, \wedge, \vee, \rightarrow$. (Propositional) Kripke models are the main tool to semantically characterize (propositional) intuitionistic logic **Int** (see [CZ97] and [Fit69] for the details). A Kripke model for $\mathcal{L}$ is a structure $\underline{K} = \langle P, \leq, \rho, \Vdash \rangle$, where $\langle P, \leq, \rho \rangle$ is a poset with minimum ρ and $\Vdash$ is the forcing relation, a binary relation between elements α of P and p of $\mathcal{PV}$ such that $\alpha \Vdash p$ and $\alpha \leq \beta$ imply that $\beta \Vdash p$. The forcing relation is extended in a standard way to arbitrary formulas of $\mathcal{L}$ as follows:

1. $\alpha \Vdash A \wedge B$ iff $\alpha \Vdash A$ and $\alpha \Vdash B$;
2. $\alpha \Vdash A \vee B$ iff $\alpha \Vdash A$ or $\alpha \Vdash B$;
3. $\alpha \Vdash A \rightarrow B$ iff, for every $\beta \in P$ such that $\alpha \leq \beta$, $\beta \Vdash A$ implies $\beta \Vdash B$;
4. $\alpha \Vdash \neg A$ iff for every $\beta \in P$ such that $\alpha \leq \beta$, $\beta \Vdash A$ does not hold.

We write $\alpha \nVdash A$ when $\alpha \Vdash A$ does not hold.

It is easy to prove that for every formula A, if $\alpha \Vdash A$ and $\alpha \leq \beta$, then $\beta \Vdash A$. A formula A *is valid in a Kripke model* $\underline{K} = \langle P, \leq, \rho, \Vdash \rangle$ if and only if $\rho \Vdash A$. It is well known that **Int** coincides with the set of formulas valid in all Kripke models.

If we consider Kripke models $\underline{K} = \langle P, \leq, \rho, \Vdash \rangle$ such that $|P| = 1$ we get classical models for (propositional) classical logic **Cl**. Classical models are usually seen as functions σ from $\mathcal{PV}$ to $\{true, false\}$. Given a formula A and a model σ, we use $\sigma \models A$ with the usual meaning of satisfiability. Finally, given a set S, the set $\mathcal{PV}(S)$ denotes the elements of $\mathcal{PV}$ occurring in S.

In the following presentation, we give a brief overview of the tableau calculus **Tab** of [AFM04] which is implemented by our decision procedure. The rules of the calculus are given in Table 1. The calculus works on signed well formed formulas (swff for short),

where a swff is a (propositional) formula prefixed with a sign $\mathbf{T}$, $\mathbf{F}$ or $\mathbf{F_c}$. Given a Kripke model $\underline{K} = \langle P, \leq, \rho, \Vdash \rangle$, a world $\alpha \in P$, a formula A and a set of swffs S, the meaning of the signs is as follows:

- $\alpha \rhd \mathbf{T}A$ (α realizes $\mathbf{T}A$) iff $\alpha \Vdash A$;
- $\alpha \rhd \mathbf{F}A$ iff $\alpha \nVdash A$;
- $\alpha \rhd \mathbf{F_c}A$ iff $\alpha \Vdash \neg A$;
- $\alpha \rhd S$ iff α realizes every swff in S.
- $\underline{K} \rhd S$ iff $\rho \rhd S$.

A proof table (or proof tree) for S is a tree, rooted with S and obtained by the subsequent application of the rules of the calculus. As an example, let $\Gamma = \{\mathbf{T}(A \wedge B), \mathbf{T}(B \wedge C), \mathbf{F}(A \vee B)\}$. With " rule $\mathbf{T}\wedge$ applies to Γ taking $H \equiv \mathbf{T}(A \wedge B)$ as main swff" we mean that $\mathbf{T}\wedge$ applies to Γ as $\frac{\Gamma}{\Gamma \setminus \{\mathbf{T}(A \wedge B)\}, \mathbf{T}A, \mathbf{T}B}\mathbf{T}\wedge$. If no confusion arises we say that *a rule applies to* Γ or equivalently *a rule applies to* H. Finally with $Rule(H)$ we mean the rule related to H (in our example $Rule(\mathbf{T}(A \wedge B))$ is $\mathbf{T}\wedge$).

For every proof table for S, the depth and the number of symbols occurring in the nodes is linearly bounded in the number of symbols occurring in S (see [AFM04] for further details). This is the key feature to implement a depth-first decision procedure whose space complexity is $O(n \lg n)$ (as a matter of fact, it is well known that to generate all the proof tables in the search space and to visit them with a depth-first strategy, it is sufficient to have a stack containing, for every node of the visited branch, the index of the main swff and a bit to store if the leftmost branch has been visited [Hud93]).

We emphasize that the sign $\mathbf{F_c}$ is introduced to give a specialized treatment of the negated formulas. In this sense the rules for the formulas signed with $\mathbf{F_c}$ could be rewritten replacing in the $\mathbf{F_c}$-rules every occurrence of the sign $\mathbf{F_c}$ with $\mathbf{T}\neg$.

Given a set of swffs S, the signed atoms of S are the elements of the set $\delta_S = \{H | H \in S$ and H is a signed atom$\}$. We say that S contains a complementary pair iff $\{\mathbf{T}A, \mathbf{F}A\} \subseteq S$ or $\{\mathbf{T}A, \mathbf{F_c}A\} \subseteq S$. Given δ_S, we denote with σ_{δ_S} the (classical) model defined as follows: if $\mathbf{T}p \in \delta_S$, then $\sigma_{\delta_S}(p) = true$, $\sigma_{\delta_S}(p) = false$ otherwise.

Given a classical model σ, (I) $\sigma \rhd \mathbf{T}H$ iff $\sigma \models H$; (II) $\sigma \rhd \mathbf{F}H$ and $\sigma \rhd \mathbf{F_c}H$ iff $\sigma \not\models H$. Given a set of swffs S, a swff H (respectively a set of swffs S') and the set of signed atoms δ_S defined as above, we say that δ_S *realizes* H (respectively δ_S *realizes* S'), and we write $\delta_S \rhd H$ (respectively $\delta_S \rhd S'$), iff there exists a classical model σ fulfilling the following conditions:

(i) $\sigma \rhd H$ (respectively $\sigma \rhd H'$ for every $H' \in S'$).

(ii) if $\mathbf{T}p \in \delta_S$, then $\sigma(p) = true$;

(iii) if $\mathbf{F}p \in \delta_S$ or $\mathbf{F_c}p \in \delta_S$, then $\sigma(p) = false$

In other words, the relation $\rhd$ between δ_S and a signed formula H holds if there exists a classical model σ both realizing δ_S and H. If $\{\mathbf{T}p, \mathbf{F_c}p, \mathbf{F}p\} \cap S = \emptyset$ then there is no condition on the truth value that σ gives to p.

$$\frac{S, \mathbf{T}(A \wedge B)}{S, \mathbf{T}A, \mathbf{T}B}\mathbf{T}\wedge \quad \frac{S, \mathbf{F}(A \wedge B)}{S, \mathbf{F}A | S, \mathbf{F}B}\mathbf{F}\wedge \quad \frac{S, \mathbf{F_c}(A \wedge B)}{S_c, \mathbf{F_c}A | S_c, \mathbf{F_c}B}\mathbf{F_c}\wedge$$

$$\frac{S, \mathbf{T}(A \vee B)}{S, \mathbf{T}A | S, \mathbf{T}B}\mathbf{T}\vee \quad \frac{S, \mathbf{F}(A \vee B)}{S, \mathbf{F}A, \mathbf{F}B}\mathbf{F}\vee \quad \frac{S, \mathbf{F_c}(A \vee B)}{S, \mathbf{F_c}A, \mathbf{F_c}B}\mathbf{F_c}\vee$$

$$\frac{S, \mathbf{T}A, \mathbf{T}(A \to B)}{S, \mathbf{T}A, \mathbf{T}B}\mathbf{T}\to \textit{Atom}, \text{ with } A \text{ an atom}$$

$$\frac{S, \mathbf{F}(A \to B)}{S_c, \mathbf{T}A, \mathbf{F}B}\mathbf{F}\to \quad \frac{S, \mathbf{F_c}(A \to B)}{S_c, \mathbf{T}A, \mathbf{F_c}B}\mathbf{F_c}\to$$

$$\frac{S, \mathbf{T}(\neg A)}{S, \mathbf{F_c}A}\mathbf{T}\neg \qquad \frac{S, \mathbf{F}(\neg A)}{S_c, \mathbf{T}A}\mathbf{F}\neg \qquad \frac{S, \mathbf{F_c}(\neg A)}{S_c, \mathbf{T}A}\mathbf{F_c}\neg$$

$$\frac{S, \mathbf{T}((A \wedge B) \to C)}{S, \mathbf{T}(A \to (B \to C))}\mathbf{T}\to\wedge \quad \frac{S, \mathbf{T}(\neg A \to B)}{S_c, \mathbf{T}A | S, \mathbf{T}B}\mathbf{T}\to\neg$$

$$\frac{S, \mathbf{T}((A \vee B) \to C)}{S, \mathbf{T}(A \to p), \mathbf{T}(B \to p), \mathbf{T}(p \to C)}\mathbf{T}\to\vee$$

$$\frac{S, \mathbf{T}((A \to B) \to C)}{S_c, \mathbf{T}A, \mathbf{F}p, \mathbf{T}(p \to C), \mathbf{T}(B \to p) | S, \mathbf{T}C}\mathbf{T}\to\to$$

where $S_c = \{\mathbf{T}A | \mathbf{T}A \in S\} \cup \{\mathbf{F_c}A | \mathbf{F_c}A \in S\}$ and p is a new atom

Table 1: the **Tab** calculus

We say that a wff or a swff is classically evaluable (*cle*-wff and *cle*-swff for short) iff conjunctions and disjunctions are the only connectives occurring in it. Finally, a set S of swffs is contradictory if at least one of the following conditions holds:

1. S contains a complementary pair;
2. S contains a *cle*-swff H such that $\delta_S \not\vdash H$;
3. $\delta_S \not\vdash S$ and for every propositional variable p occurring in S, $\mathbf{T}p \in S$ or $\mathbf{F_c}p \in S$.

Proposition 2.1 *If a set of swffs S is contradictory, then for every Kripke model $\underline{K} = \langle P, \leq, \rho, \Vdash \rangle$, $\rho \not\Vdash S$.*

Proof: If S is contradictory because the first condition holds, then for some formula A, $\{\mathbf{T}A, \mathbf{F}A\} \subseteq S$ or $\{\mathbf{T}A, \mathbf{F_c}A\} \subseteq S$ holds. By the meaning of the signs and the definition of the forcing relation in Kripke models, the claim immediately follows. If S is contradictory because the second condition holds, then $\delta_S \not\vdash H$. Thus, there is no classical model realizing both the signed atoms of S (that is δ_S) and H. Since H

is a *cle*-swff, its classical and intuitionistic realizability coincide, thus no Kripke model realizes S. If S is contradictory because the third condition holds, then let us suppose there exists a Kripke model $\underline{K} = \langle P,\leq,\rho,\Vdash\rangle$ such that $\rho \rhd S$. Then for every $p \in \mathcal{PV}(S)$, $\rho \Vdash p$ or $\rho \Vdash \neg p$ and this means that every world in $\underline{K}$ forces the same propositional variables occurring in S, that is for every $\alpha, \beta \in P$ and for every $p \in \mathcal{PV}(S)$, $\alpha \Vdash p$ iff $\beta \Vdash p$. Let $\phi \in P$ be a maximal state in the poset $\langle P, \leq, \rho\rangle$. Since ϕ behaves as a classical model, by hypothesis, $\phi \not\rhd S$. Then, since every world of $\underline{K}$ forces the same propositional variables of S we deduce that $\rho \not\rhd S$. □

A closed proof table is a proof table whose leaves are all contradictory sets. A closed proof table is a proof of the calculus: a formula A is provable if there exists a closed proof table for $\{\mathbf{F}A\}$. For every rule of the calculus it is easy to prove that if there exists a Kripke model $\underline{K} = \langle P,\leq,\rho,\Vdash\rangle$ and $\alpha \in P$ such that α realizes the premise of the rule, then there exists (a possibly different) Kripke model $\underline{K}' = \langle P', \leq', \rho', \Vdash'\rangle$ and $\beta \in P'$ such that β realizes the conclusion. This is the main step to prove the soundness of the calculus:

Theorem 2.2 (Soundness) *Let A be a wff. If there exists a closed proof table starting from $\{\mathbf{F}A\}$, then A is valid.*

3 The optimizations and the proof search algorithm

Below we describe a recursive procedure Tab(S) which given a set S of swffs, returns either a closed proof table for S or `NULL` (if there exists a Kripke model realizing S).

To describe Tab we use the following notation. Let S be a set of swffs, let $H \in S$ and let S_1 or $S_1 \mid S_2$ be the nodes of the proof tree obtained by applying to S the rule $Rule(H)$ corresponding to H. If Tab_1 and Tab_2 are closed proof tables for S_1 and S_2 respectively, then $\dfrac{S}{Tab_1}\,{\scriptstyle Rule(H)}$ or $\dfrac{S}{Tab_1 \mid Tab_2}\,{\scriptstyle Rule(H)}$ denote the closed proof table for S defined in the obvious way. Moreover, $\mathcal{R}_i(H)$ $(i = 1, 2)$ denotes the set containing the swffs of S_i which replaces H. For instance:
$\mathcal{R}_1(\mathbf{T}(A \wedge B)) = \{\, \mathbf{T}A, \mathbf{T}B \,\}$,
$\mathcal{R}_1(\mathbf{T}(A \vee B)) = \{\mathbf{T}A\}$, $\mathcal{R}_2(\mathbf{T}(A \vee B)) = \{\mathbf{T}B\}$,
$\mathcal{R}_1(\mathbf{T}((A \to B) \to C)) = \{\, \mathbf{T}A, \mathbf{F}p, \mathbf{T}(B \to p), \mathbf{T}(p \to C) \,\}$,
$\mathcal{R}_2(\mathbf{T}((A \to B) \to C)) = \{\mathbf{T}C\}$.

As stated in the introduction, Tab uses substitutions. For our purposes, the only substitutions we are interested in are permutations from $\mathcal{PV}$ to $\mathcal{PV}$. Given a substitution τ and a swff H (respectively, a set of swffs S and tableau T) $\tau(H)$ (respectively, $\tau(S)$ and $\tau(T)$) means the swff (respectively, the set of swffs and the tableau) obtained by applying the substitution in the obvious way.

The procedure Tab divides the formulas in six groups according to their behavior with respect to branching and backtracking:
$\mathcal{C}_1 = \{\mathbf{T}(A \wedge B), \mathbf{F}(A \vee B), \mathbf{F_c}(A \vee B), \mathbf{T}(\neg A), \mathbf{T}(p \to A)$ with p an atom, $\mathbf{T}((A \wedge B) \to C)$, $\mathbf{T}((A \vee B) \to C)\}$;
$\mathcal{C}_2 = \{\mathbf{T}(A \vee B), \mathbf{F}(A \wedge B)\}$;
$\mathcal{C}_3 = \{\mathbf{F}(\neg A), \mathbf{F}(A \to B)\}$;

$\mathcal{C}_4 = \{\mathbf{T}((A \to B) \to C), \mathbf{T}(\neg A \to B)\}$;
$\mathcal{C}_5 = \{\mathbf{F_c}(A \to B), \mathbf{F_c}(\neg A)\}$;
$\mathcal{C}_6 = \{\mathbf{F_c}(A \wedge B)\}$.
We call $\mathcal{C}_i$-swffs ($i = 1, \ldots, 6$), respectively $\mathcal{C}_i$-rules, the swffs of the group $\mathcal{C}_i$, respectively the rules related to $\mathcal{C}_i$-swffs.

The intuition behind these groups can be explained as follows. It is well known that in classical logic the order in which the rules are applied does not affect the completeness of the decision procedure: if a proof exists it is found independently of the order in which the rules are applied. Thus the search space consists of a single proof tree whose branches have to be visited by the decision procedure. The rules having two conclusions give rise to branches. Rules of this kind are $\mathbf{T}\vee$ and $\mathbf{F}\wedge$. In intuitionistic logic the order in which the rules are applied is relevant and affect the completeness. Given a set Γ, there are many ways to go on with the proof (that is many swffs can be chosen as main swff). Since the order is relevant, if the choice of a swff as main swff does not give a closed proof table, we have to backtrack and try with another swff as main swff. This means that in intuitionistic logic there is a space of proof tables to be visited by backtracking. Rules requiring backtracking are, i.e., $\mathbf{F} \to$, $\mathbf{T} \to\to$. In order to bound time consumption, TAB applies the rules not requiring branching and backtracking first, then the rules not requiring backtracking, finally the rules requiring backtracking. In the Completeness Lemma (Lemma 4.1, page 11) we prove that we do not lose completeness if $\mathcal{C}_5$ and $\mathcal{C}_6$-rules are applied only when no other rule applies. Thus the application of $\mathcal{C}_5$ and $\mathcal{C}_6$-rules is invertible and no backtracking is required. On the other hand, to get completeness, backtracking is unavoidable when $\mathcal{C}_3$ and $\mathcal{C}_4$-rules are applied.

Now we come to the optimizations. First we discuss two checks that allow us to bound the depth of the proofs. Let S be a set such that $\sigma_{\delta_S} \rhd S$. Thus the Kripke model coinciding with the classical model σ_{δ_S} realizes S and we do not need to go on with the proof. The second check is related to Point 3 in the definition of contradictory set. If $\delta_S \not\rhd S$ and every propositional variable occurring in S occurs in S as swff signed $\mathbf{T}$ or $\mathbf{F_c}$, then there is no Kripke model realizing S and we do not need to proceed with the proof. Although these checks could be performed after every rule application, our strategy performs it when neither a $\mathcal{C}_1$ nor a $\mathcal{C}_2$-rule applies to S (in Section 5 this optimization is referred as **opt1**).

In order to bound branching, TAB treats in a particular way the *cle*-swffs in S, that is the swffs in which only $\wedge$ and $\vee$ occur (in other words swffs whose intuitionistic truth coincides with classical truth). When TAB treats $\mathcal{C}_2$-swffs (Point 3 of the algorithm given below), first of all TAB checks if in S there exists a *cle*-swff H such that $\delta_S \not\rhd H$. If S fulfills this condition, then S is not realizable, as we pointed out in Proposition 2.1. Otherwise, if in S a *c*le-swff H occurs, such that σ_{δ_S} does not satisfy H, the splitting rule $Rule(H)$ is applied (we recall that σ_{δ_S} is the classical model defined from S by taking as true the atoms p such that $\mathbf{T}p \in S$). Thus the *c*le-swffs of S, satisfying the underlying model, are never applied. Despite this, from the realizability of one among $(S \setminus \{H\}) \cup \mathcal{R}_1(H)$ and $(S \setminus \{H\}) \cup \mathcal{R}_2(H)$ we have enough information to prove the realizability of S. In other words, we consider only the *cle*-swffs of $\mathcal{C}_2$ that are not realized by the model underlying S (see Point 3 and Completeness Lemma for the

details). As an example consider

$$\begin{aligned} S \ = \{ \ & \mathbf{F}(P0 \wedge P2), \mathbf{F}(P0 \wedge P4), \mathbf{F}(P2 \wedge P4), \mathbf{F}(P1 \wedge P3), \mathbf{F}(P1 \wedge P5), \\ & \mathbf{F}(P3 \wedge P5), \mathbf{T}(P0 \vee P1), \mathbf{T}(P2 \vee P3), \mathbf{T}(P4 \vee P5) \ \}. \end{aligned}$$

Since σ_{δ_S} realizes the $\mathbf{F}$-swffs in S but σ_{δ_S} does not realize any of the $\mathbf{T}$-swffs of S, then TAB chooses one of them, let us suppose $H = \mathbf{T}(P0 \vee P1)$. The rule $\mathbf{T}\vee$ is applied to S and $S_1 = (S \setminus \{H\}) \cup \{\mathbf{T}P0\}$ and $S_2 = (S \setminus \{H\}) \cup \{\mathbf{T}P1\}$ are the subsequent sets of S. Now consider S_1. Since $\sigma_{\delta_{S_1}}$ realizes $\mathbf{T}P0$ and all the $\mathbf{F}$-swffs in S_1, but $\sigma_{\delta_{S_1}}$ realizes neither $\mathbf{T}(P2 \vee P3)$ nor $\mathbf{T}(P4 \vee P5)$, TAB chooses one of them, let us suppose $H = \mathbf{T}(P2 \vee P3)$. The rule $\mathbf{T}\vee$ is applied to S_1 and $S_3 = (S_1 \setminus \{H\}) \cup \{\mathbf{T}P2\}$ and $S_4 = (S_1 \setminus \{H\}) \cup \{\mathbf{T}P3\}$ are the subsequent sets of S_1. Since δ_{S_3} does not realize $\mathbf{F}(P0 \wedge P2)$ we deduce that S_3 is contradictory. Similarly for the other sets. We emphasize that at every step, the $\mathcal{C}_2$-rules applicable to S_i are only the ones where the related swffs are not realized by $\sigma_{\delta_{S_i}}$. Without this strategy a huge closed proof table could arise (in Section 5 this optimization is referred as **opt2**).

To bound the search space, [Wei98] describes a decision procedure in which backtracking is bounded by a semantical technique inspired by the completeness theorem. The completeness theorem proves the satisfiability (realizability in our context) of a set S under the hypothesis that S does not have any closed proof table. As an example, let $S = \{\mathbf{F}(A \rightarrow B), \mathbf{T}((A \rightarrow B) \rightarrow C), \mathbf{F}C, \mathbf{T}D\}$. From S we can define the Kripke model $\underline{K}(S) = \langle P,\leq,\rho,\Vdash\rangle$ such that $P = \{\rho\}$ and $\rho \Vdash D$. Note that $\underline{K}(S)$ realizes $\mathbf{T}D$ but $\underline{K}(S)$ does not realize S. To prove the realizability of S, the realizability of $S_1 = \{\mathbf{T}A, \mathbf{F}B, \mathbf{T}((A \rightarrow B) \rightarrow C), \mathbf{T}D\}$ and one between $S_2 = \{\mathbf{T}A, \mathbf{F}p, \mathbf{T}(B \rightarrow p), \mathbf{T}(p \rightarrow C), \mathbf{T}D\}$ and $S_3 = \{\mathbf{F}(A \rightarrow B), \mathbf{T}C, \mathbf{F}C, \mathbf{T}D\}$ have to be proved. Since S_3 is not realizable, the realizability of S_1 and S_2 must be proved. From S_1 we define the Kripke model $\underline{K}(S_1) = \langle\{\alpha\},\leq,\alpha,\Vdash\rangle$, where $\alpha \leq \alpha$, $\alpha \Vdash D$ and $\alpha \Vdash A$ such that $\underline{K}(S_1) \rhd S_1$. If we glue $\underline{K}(S1)$ above $\underline{K}(S)$ we get a new Kripke model $\underline{K} = \langle\{\rho, \alpha\},\leq,\rho,\Vdash\rangle$ where $\rho \leq \rho$, $\rho \leq \alpha$, $\alpha \leq \alpha$, $\rho \Vdash D$,$\alpha \Vdash D$ and $\alpha \Vdash A$. Since $\underline{K} \rhd S$, we do not need to apply $\mathbf{T} \rightarrow\rightarrow$ to S in order to obtain $S_2 = \{\mathbf{T}A, \mathbf{F}p, \mathbf{T}(B \rightarrow p), \mathbf{T}(B \rightarrow C), \mathbf{T}D\}$ (from S_2 a Kripke model $\underline{K}(S_2)$ is definable; $\underline{K}(S_2)$ glued above $\underline{K}(S)$ gives rise to a Kripke model realizing S). In this case the work on S_2 is spared. In the general case, see [Wei98], the information collected from non closed proof tables built from a set S is used to build a Kripke model $\underline{K}$. As a matter of fact, let S be a set such that no $\mathcal{C}_1$ or $\mathcal{C}_2$-rule is applicable. Let $\{H_1, \ldots, H_u\} \subseteq S$ the $\mathcal{C}_3$ and $\mathcal{C}_4$-swffs of S. If there exists a H_j such that $\underline{K} \not\rhd H_j$, then $Rule(H_j)$ have to be applied. If a closed proof table is found, then S is not realizable, otherwise the Kripke model $\underline{K}$ can be extended in a new one, $\underline{K}_j$ satisfying H_j. The procedure of [Wei98] continues until a closed proof table or a Kripke model $\underline{K}_i$, $1 \leq i \leq u$, such that $\underline{K}_i \rhd \{H_1, \ldots, H_u\}$ is found. The procedure prunes the search space, since in S not all the swffs requiring backtracking are considered, but only the swffs which, when checked, are not realized from the Kripke model at hand.

Now, consider $S = \{\mathbf{F}(A \rightarrow B), \mathbf{F}(C \rightarrow D)\}$. From S we can define the Kripke model $\underline{K}(S) = \langle P,\leq,\rho,\Vdash\rangle$ such that $P = \{\rho\}$ and $\Vdash$ is the empty set. $\underline{K}(S)$ does not realize S. By applying $\mathbf{F} \rightarrow$ to S with $\mathbf{F}(A \rightarrow B)$ as the main formula we get $S_1 = \{\mathbf{T}A, \mathbf{F}B\}$. The underlying model is $\underline{K}(S_1) = \langle\{\alpha\},\leq,\alpha,\Vdash\rangle$ with $\alpha \Vdash A$. $\underline{K}(S_1)$ glued above $\underline{K}(S)$ gives rise to a model that does not realize $\mathbf{F}(C \rightarrow D)$. Thus we must backtrack. We

apply $\mathbf{F}\rightarrow$ to S with $\mathbf{F}(C\rightarrow D)$ as the main formula. We get $S_2=\{\mathbf{T}C,\mathbf{F}D\}$. The underlying model is $\underline{K}(S_2)=\langle\{\beta\},\leq,\beta,\Vdash\rangle$ such that $\beta\leq\beta$ and $\beta\Vdash C$ realizes S_2. By gluing $\underline{K}(S_1)$ and $\underline{K}(S_2)$ above $\underline{K}(S)$ the resulting model $\underline{K}=\langle\{\rho,\alpha,\beta\},\leq,\rho,\Vdash\rangle$ such that

$$\rho\leq\alpha,\rho\leq\beta,\rho\leq\rho,\alpha\leq\alpha,\beta\leq\beta,\alpha\Vdash A \text{ and } \beta\Vdash C$$

realizes S. But by a permutation $\tau:\mathcal{PV}\rightarrow\mathcal{PV}$ such that $\tau(C)=A$ and $\tau(D)=B$, $\tau(S_2)=S_1$ and we can build $\underline{K}(S_2)=\langle P,\leq,\Vdash',\rho,\rangle$ from $\underline{K}(S_1)$ as follows: $\underline{K}(S_2)$ has the same poset as $\underline{K}(S_1)$ and $\Vdash'$ is: for every $\alpha\in P$ and for every $p\in\mathcal{PV}$, $\alpha\Vdash' p$ iff $\alpha\Vdash\tau(p)$. In other words, $\underline{K}(S_1)$ can be translated into $\underline{K}(S_2)$ via τ and we can avoid backtracking on S. As another example consider

$$\begin{array}{rl} S \;=\{ & \mathbf{T}(((P0\rightarrow(P1\vee P2))\rightarrow(P1\vee P2))),\\ & \mathbf{T}(((P2\rightarrow(P1\vee P0))\rightarrow(P1\vee P0))),\\ & \mathbf{T}(((P1\rightarrow(P2\vee P0))\rightarrow(P2\vee P0))),\mathbf{F}((P1\vee(P2\vee P0)))\;\},\end{array}$$

where only a few steps are needed to obtain S starting from $\{\mathbf{F}H\}$, where H is the axiom schema $\bigwedge_{i=0}^{2}\left((P_i\rightarrow\bigvee_{j\neq i}P_j)\rightarrow\bigvee_{j\neq i}P_j\right)\rightarrow\bigvee_{i=0}^{2}P_i$ characterizing the logic of binary trees (a logic in the family of k-ary trees logics, [CZ97], also known as Gabbay-de Jongh logics). From S we can define the model $\underline{K}(S)=\langle P,\leq,\rho,\Vdash\rangle$ such that $P=\{\rho\}$ and $\Vdash$ is the empty set. $\underline{K}(S)$ does not realize S. By applying $\mathbf{T}\rightarrow\!\!\rightarrow$ to S with $H=\mathbf{T}(((P0\rightarrow(P1\vee P2))\rightarrow(P1\vee P2)))$ we get $S_1=(S\setminus\{H\})\cup\mathcal{R}_2(H)$ and $S_2=(S\setminus\{H\})_c\cup\mathcal{R}_1(H)$. Since S_1 is not realizable, to prove the realizability of S we have to prove the realizability of S_2. S_2 defines the Kripke model $\underline{K}(S_2)=\langle\{\alpha\},\leq,\alpha,\Vdash\rangle$, where $\alpha\leq\alpha$ and $\alpha\Vdash P0$. Thus $\underline{K}(S_2)\rhd S_2$ holds. Now, if we glue $\underline{K}(S_2)$ above $\underline{K}(S)$ we get a new model $\underline{K}'(S)=\langle\{\rho,\alpha\},\leq,\rho,\Vdash\rangle$, where $\rho\leq\rho,\rho\leq\alpha,\alpha\leq\alpha$ and $\alpha\Vdash P0$. $\underline{K}'(S)$ does not realize S. Thus we must backtrack twice:

(i) by applying $\mathbf{T}\rightarrow\!\!\rightarrow$ to S with $H=\mathbf{T}(((P2\rightarrow(P1\vee P0))\rightarrow(P1\vee P0)))$ we get, $S_3=(S\setminus\{H\})_c\cup\mathcal{R}_1$ and $S_4=(S\setminus\{H\})\cup\mathcal{R}_2(H)$;

(ii) by applying $\mathbf{T}\rightarrow\!\!\rightarrow$ to S with $H=\mathbf{T}(((P1\rightarrow(P2\vee P0))\rightarrow(P2\vee P0)))$ we get $S_5=(S\setminus\{H\})\cup\mathcal{R}_2(H)$ and $S_6=(S\setminus\{H\})_c\cup\mathcal{R}_1(H)$.

In a few steps we find that S_4 and S_6 are not realizable. From S_3 we define the Kripke model $\underline{K}(S_3)=\langle\{\beta\},\leq,\beta,\Vdash\rangle$ where $\beta\leq\beta$ and $\beta\Vdash P2$. $\underline{K}(S_3)\rhd S_3$. From S_5 we define the Kripke model $\underline{K}(S_5)=\langle\{\gamma\},\leq,\gamma,\Vdash\rangle$ where $\gamma\leq\gamma$ and $\gamma\Vdash P1$. $\underline{K}(S_5)\rhd S_5$. Thus by gluing $\underline{K}(S_2),\underline{K}(S_3)$ and $\underline{K}(S_5)$ above $\underline{K}(S)$ we get a model $\underline{K}$ realizing S. Since we can define the permutations τ_1 and τ_2 such that $\tau_1(S_3)=S_2$ and $\tau_2(S_5)=S_2$ we can avoid backtracking. Thus no proof of realizability of S_3 or S_5 is needed and the Kripke models realizing S_3 and S_5 can be obtained by applying the permutations on the forcing relation of the Kripke model for S_2.

Thus to avoid backtracking Tab builds a permutation τ between sets of swffs. Let H be $\mathcal{C}_3$-swff. Before applying $Rule(H)$ we check if there exists a permutation τ from $\mathcal{PV}(S)$ to $\mathcal{PV}(S)$ such that $\tau((S\setminus\{H\})_c\cup\mathcal{R}_1(H))=(S\setminus\{H'\})_c\cup\mathcal{R}_1(H')$. We already know that the set $(S\setminus\{H'\})_c\cup\mathcal{R}_1(H')$ obtained treating H' is realizable by a Kripke model $\underline{K}'$. Since we have a permutation τ, then the set $(S\setminus\{H\})_c\cup\mathcal{R}_1(H)$ is realized by the Kripke model $\underline{K}$ having the same poset as $\underline{K}'$ and such that for every world α and

every propositional variable p, $\alpha \Vdash_{\underline{K}'} p$ iff $\alpha \Vdash_{\underline{K}} \tau(p)$. This means that the permutation τ allows us to go from $\underline{K}'$ to $\underline{K}$ (and the permutation τ^{-1} allows us to go from $\underline{K}$ to $\underline{K}'$). In particular, τ and τ^{-1} translate the forcing relation between the models. Analogously if H is a $\mathcal{C}_4$-swff. We emphasize that given a Kripke model $\underline{K}$, a permutation τ and a swff H, $\underline{K} \rhd H$ does not imply $\underline{K} \rhd \tau(H)$. Thus we have taken a different route with respect to [Wei98], where the realizability of $\tau(H)$ is checked on $\underline{K}$ that realizes H and the two methods work in different situations. We emphasize that both methods imply a certain computational cost. The method of [Wei98] implies checking the realizability on a Kripke model, which is time consuming for swffs of the kind $\mathbf{T}(A \to B)$. Our method can be time consuming if we perform a search of a permutation among the $Pv(S)!$ possible permutations. However, as we describe in Section 5, the procedure searches in a small subset of all possible permutations (in Section 5, this optimization is referred as **opt3**).

Finally, we could also define a permutation to prove that a set is not realizable. As a matter of fact, if S is not realizable and there exists a permutation τ such that $\tau(S) = S'$, then S' is not realizable. Thus, given a set S and a $\mathcal{C}_2$ or $\mathcal{C}_6$-swff $H \in S$, if $(S \setminus \{H\}) \cup \mathcal{R}_1(H)$ is closed and there exists a permutation τ such that $\tau((S \setminus \{H\}) \cup \mathcal{R}_1(H)) = (S \setminus \{H\}) \cup \mathcal{R}_2(H)$ then $(S \setminus \{H\}) \cup \mathcal{R}_2(H)$ is not realizable and the tableau proof for $(S \setminus \{H\}) \cup \mathcal{R}_1(H)$ can be translated via τ in a tableau proof for $(S \setminus \{H\}) \cup \mathcal{R}_2(H)$ (see Points 3 and 6 of TAB). As a trivial application, consider a valid wff $H(\underline{p})$, where $\underline{p} = \{p_1, \ldots, p_n\}$ are all the propositional variables occurring in H. To prove that $\{\mathbf{F}(H(\underline{p}) \wedge H(\underline{q}))\}$ is closed, it is sufficient to prove, by an application of $\mathbf{F}\wedge$, that $\{\mathbf{F}H(\underline{p})\}$ is closed and there exists a permutation such that $\{\mathbf{F}H(\underline{p})\} = \tau(\{\mathbf{F}H(\underline{q})\})$.

To save work space, we describe TAB in natural language. The algorithm is divided in seven main points. We recall that the input of **Tab** is a set S of swffs. If S is realizable, then **Tab** returns `NULL`, otherwise **Tab** returns a closed proof table for S. In the following description, given a set V of swffs, TAB (V) is the recursive call to TAB with actual parameter V. Some instructions 'return `NULL`' are labeled with **r1**,..., **r6**. In the Completeness Lemma we refer to such instructions by means of these labels.

FUNCTION TAB (S)

1. If S contains a complementary pair, then TAB returns the proof S;

2. If a $\mathcal{C}_1$-rule applies to S, then let H be a $\mathcal{C}_1$-swff. If TAB$((S \setminus \{H\}) \cup \mathcal{R}_1(H))$ returns a proof π, then TAB returns the proof $\dfrac{S}{\pi}\,Rule(H)$, otherwise TAB returns `NULL` (**r1**);

3. If a $\mathcal{C}_2$-rule applies to S, then if there exists a $\mathcal{C}_2$-swff H such that H is a *cle*-swff and $\delta_S \not\rhd H$, then TAB returns (the proof) S. Otherwise, let H be a *cle*-swff such that $\sigma_{\delta_S} \not\rhd H$, if there is any, otherwise let H be a $\mathcal{C}_2$-swff. Let $\pi_1 = \text{TAB}(S \setminus \{H\} \cup \mathcal{R}_1(H))$. If π_1 is `NULL`, then TAB returns `NULL`. If there exists a permutation τ such that $(S \setminus \{H\}) \cup \mathcal{R}_1(H) = \tau(S \setminus \{H\}) \cup \mathcal{R}_2(H))$, then TAB returns the proof $\dfrac{S}{\pi_1 \mid \tau^{-1}(\pi_1)}\,Rule(H)$. If such a permutation does not exist, then let $\pi_2 = \text{TAB}(S \setminus \{H\} \cup \mathcal{R}_2(H))$. If π_2 is a proof, then TAB returns the proof $\dfrac{S}{\pi_1 \mid \pi_2}\,Rule(H)$, otherwise ($\pi_2$ is `NULL`) TAB returns `NULL`;

4. If a $\mathcal{C}_3$ or $\mathcal{C}_4$-rule applies to S, then TAB proceeds as follows: if $\sigma_{\delta_S} \rhd S$, then TAB

returns NULL (**r2**). If for every $p \in \mathcal{PV}(S)$, $\mathbf{T}p \in S$ or $\mathbf{F_c}p \in S$, then TAB returns S. If the previous points do not apply, then TAB carries out the following Points **4.1** and **4.2**:

4.1 Let $\{H_1, \ldots H_n\}$ be all the $\mathcal{C}_3$-swffs in S. For $i = 1, \ldots, n$, the following instructions are iterated: if there is no swff H_j ($j \in \{1, \ldots, i-1\}$) and a permutation τ such that $(S \setminus \{H_j\})_c \cup \mathcal{R}_1(H_j) = \tau((S \setminus \{H_i\})_c \cup \mathcal{R}_1(H_i))$, then let $\pi = \text{TAB}((S \setminus \{H_i\})_c \cup \mathcal{R}_1(H_i))$. If π is a proof, then TAB returns the proof $\dfrac{S}{\pi}\, Rule(H_i)$;

4.2 Let $\{H_1, \ldots H_n\}$ be all the $\mathcal{C}_4$-swffs in S. For $i = 1, \ldots, n$, the following Points (**4.2.1**) and (**4.2.2**) are iterated.

(**4.2.1**) If there is neither swff H_j ($j \in \{1, \ldots, i-1\}$) nor a permutation τ such that $(S \setminus \{H_j\}) \cup \mathcal{R}_2(H_j) = \tau((S \setminus \{H_i\}) \cup \mathcal{R}_2(H_i))$, then let $\pi_{2,i} = \text{TAB}((S \setminus \{H_i\}) \cup \mathcal{R}_2(H_i))$. If $\pi_{2,i}$ is NULL, then TAB returns NULL (**r3**). If there is neither swff H_j, with $j \in \{1, \ldots, i-1\}$, nor a permutation τ such that $(S \setminus \{H_j\})_c \cup \mathcal{R}_1(H_j) = \tau((S \setminus \{H_i\})_c \cup \mathcal{R}_1(H_i))$, then if $\text{TAB}((S \setminus \{H_i\})_c \cup \mathcal{R}_1(H_i))$ returns a proof π_1, TAB returns the proof $\dfrac{S}{\pi_1 \mid \pi_{2,i}}\, Rule(H_i)$.

(**4.2.2**) If Point (**4.2.1**) does not hold, then there exists a permutation τ and a swff H_j ($j \in \{1, \ldots, i-1\}$) such that $(S \setminus \{H_j\}) \cup \mathcal{R}_2(H_j) = \tau((S \setminus \{H_i\}) \cup \mathcal{R}_2(H_i))$. If there is no swff H_u, with $u \in \{1, \ldots, i-1\}$, and a permutation τ such that $(S \setminus \{H_u\})_c \cup \mathcal{R}_1(H_u) = \tau((S \setminus \{H_i\})_c \cup \mathcal{R}_1(H_i))$, then if $\text{TAB}((S \setminus \{H_i\})_c \cup \mathcal{R}_1(H_i))$ returns a proof π_1, TAB returns the proof $\dfrac{S}{\pi_1 \mid \tau^{-1}(\pi_{2,j})}\, Rule(H_i)$.

If in Points (**4.1**) and (**4.2**) TAB does not find any closed proof table, then TAB returns NULL (**r4**).

5. If a $\mathcal{C}_5$-rule applies to S, then let H be a $\mathcal{C}_5$-swff. If $\text{TAB}((S \setminus \{H\})_c \cup \mathcal{R}_1(H))$ returns a proof π, then TAB returns the proof $\dfrac{S}{\pi}\, Rule(H)$, otherwise TAB returns NULL;

6. If a $\mathcal{C}_6$-rule applies to S, then let H be a $\mathcal{C}_6$-swff. Let $\pi_1 = \text{TAB}((S \setminus \{H\})_c \cup \mathcal{R}_1(H))$. If π_1 is NULL, then TAB returns NULL. If there exists a permutation τ such that $(S \setminus \{H\})_c \cup \mathcal{R}_1(H) = \tau((S \setminus \{H\})_c \cup \mathcal{R}_2(H))$, then TAB returns the proof $\dfrac{S}{\pi_1 \mid \tau^{-1}(\pi_1)}\, Rule(H)$. If such a permutation does not exist, then let $\pi_2 = \text{TAB}((S \setminus \{H\})_c \cup \mathcal{R}_2(H))$. If π_2 is a proof, then TAB returns $\dfrac{S}{\pi_1 \mid \pi_2}\, Rule(H)$, otherwise ($\pi_2$ is NULL) TAB returns NULL (**r5**);

7. If none of the previous points apply, then TAB returns NULL (**r6**).

END FUNCTION TAB.

Point 4 deserves some comments. If the classical model σ_{δ_S} realizes S (condition "$\sigma_{\delta_S} \rhd S$") then such a model is a Kripke model realizing S. If for every propositional variable $p \in Pv(S)$, $\mathbf{T}p \in S$ or $\mathbf{F_c}p \in S$ holds, then the subsequent sets of S do not contain more information than S and from $\sigma_{\delta_S} \not\rhd S$ we deduce $\delta_S \not\rhd S$. Since $\delta_S \not\rhd S$, then S is not realizable (see Proposition 2.1). The iteration in Points **4.1** and **4.2** can be summarized as follows: a proof for S is searched: for every $\mathcal{C}_3$ or $\mathcal{C}_4$-swff H in S, $Rule(H)$ is applied to S. If no proof is found, then S is realizable. Now consider $\mathcal{C}_3$-swffs. If for a previous iteration j the set obtained by applying $Rule(H_j)$ to S is realizable and can be translated via a permutation τ in $(S \setminus \{H_i\})_c \cup \mathcal{R}(H_i)$, then TAB does not

apply $Rule(H_i)$ to S. The permutation τ and the realizability of $(S \setminus \{H_j\})_c \cup \mathcal{R}(H_j)$ imply the realizability of $(S \setminus \{H_i\})_c \cup \mathcal{R}(H_i)$ (see Case 4 in Completeness Lemma). TAB applies the same idea in Point **4.2** to $\mathcal{C}_4$-swffs of S. This point is more complex because $\mathcal{C}_4$-rules have two conclusions.

4 Completeness

In order to prove the completeness of TAB, we prove that given a set of swffs S, if the call TAB(S) returns `NULL`, then we have enough information to build a countermodel $\underline{K} = \langle P, \leq, \rho, \Vdash \rangle$ such that $\rho \rhd S$. To prove the proposition we need to introduce the function deg defined as follows:

- if p is an atom, then $\deg(p) = 0$;
- $\deg(A \wedge B) = \deg(A) + \deg(B) + 2$;
- $\deg(A \vee B) = \deg(A) + \deg(B) + 3$;
- $\deg(A \to B) = \deg(A) + \deg(B)+$ (number of implications occurring in A) + 1;
- $\deg(\neg A) = \deg(A) + 1$;
- $\deg(S) = \sum_{H \in S} \deg(H)$.

It is easy to show that if S' is obtained from a set of swffs S by an application of a rule of **Tab**, then $\deg(S') < \deg(S)$.

Lemma 4.1 (Completeness) *Let S be a set of swffs and suppose that* TAB(S) *returns the* `NULL` *value. Then, there is a Kripke model $\underline{K} = \langle P, \leq, \rho, \Vdash \rangle$ such that $\rho \rhd S$.*

Proof: The proof goes by induction on the complexity of S, measured with respect to the function $\deg(S)$.
Basis: if $\deg(S) = 0$, then S contains atomic swffs only. TAB(S) carries out the instruction labeled (***r6***). Moreover, S does not contain sets of the kind $\{\mathbf{T}p, \mathbf{F}p\}$ and $\{\mathbf{T}p, \mathbf{F_c}p\}$. Let $\underline{K} = \langle P, \leq, \rho, \Vdash \rangle$ be the Kripke model such that $P = \{\rho\}$ and $\rho \Vdash p$ iff $\mathbf{T}p \in S$. It is easy to show that $\rho \rhd S$.

Step: Let us assume by induction hypothesis that the proposition holds for all sets S' such that $\deg(S') < \deg(S)$. We prove that the proposition holds for S by inspecting all the possible cases where the procedure returns the `NULL` value.
Case 1: the instruction labeled **r1** *has been performed.* By induction hypothesis there exists a Kripke model $\underline{K} = \langle P, \leq, \rho, \Vdash \rangle$ such that $\rho \rhd (S \setminus \{H\}) \cup \mathcal{R}_1(H)$, with $H \in \mathcal{C}_1$. We prove $\rho \rhd H$ by proceeding according to the cases of H. If H is of the kind $\mathbf{T}(A \wedge B)$, then by induction hypothesis $\rho \rhd \{\mathbf{T}A, \mathbf{T}B\}$, thus $\rho \Vdash A$ and $\rho \Vdash B$, and therefore $\rho \Vdash A \wedge B$. This implies $\rho \rhd \mathbf{T}(A \wedge B)$. The other cases for $H \in \mathcal{C}_1$ are similar.
Case 2: the instruction labeled **r2** *has been performed.* Thus $\sigma_{\delta_S} \rhd S$ holds. We use σ_{δ_S} to define a Kripke model $\underline{K}$ with a single world ρ such that $\rho \Vdash p$ iff $\sigma(p) = true$. Since ρ behaves as a classical model, $\rho \rhd S$ holds.
Case 3: the instruction labeled **r3** *has been performed.* By induction hypothesis there

exists a model $\underline{K}$ such that $\rho \rhd (S \setminus \{H_i\}) \cup \mathcal{R}_2(H_i)$, where $H_i \in \mathcal{C}_4$. Let us suppose that H is of the kind $\mathbf{T}((A \to B) \to C)$, thus $\rho \rhd \mathbf{T}C$ and this implies $\rho \rhd H_i$. The proof goes similarly if H_i is of the kind $\mathbf{T}(\neg A \to B)$.
Case 4: the instruction labeled **r4** *has been performed.* This implies that: (i) for every $H \in S \cap \mathcal{C}_3$, we have two cases: (ia) $\textsc{Tab}((S \setminus \{H\})_c \cup \mathcal{R}_1(H)) = \texttt{NULL}$, thus by induction hypothesis there exists a Kripke model $\underline{K}_H = \langle P_H, \leq_H, \rho_H, \Vdash_H \rangle$ such that $\rho_H \rhd (S \setminus \{H\})_c \cup \mathcal{R}_1(H)$; (ib) there exists a permutation τ from $\mathcal{PV}(S)$ to $\mathcal{PV}(S)$ and a swff $H' \in S \cap \mathcal{C}_3$ such that $\textsc{Tab}((S \setminus \{H'\})_c \cup \mathcal{R}_1(H')) = \texttt{NULL}$ and $(S \setminus \{H'\})_c \cup \mathcal{R}_1(H') = \tau((S \setminus \{H\})_c \cup \mathcal{R}_1(H))$. Thus by Point (a) applied to H', there exists a Kripke model $\underline{K}_{H'} = \langle P_{H'}, \leq_{H'}, \rho_{H'}, \Vdash_{H'} \rangle$ such that $\rho_{H'} \rhd (S \setminus \{H'\})_c \cup \mathcal{R}_1(H')$. By using τ we can translate $\underline{K}_{H'}$ into a model $\underline{K}_H = \langle P_H, \leq_H, \rho_H, \Vdash_H \rangle$, where $P_H = P_{H'}$, $\leq_H = \leq_{H'}$, $\rho_H = \rho_{H'}$ and for every world $\alpha \in P_H$, if $p \in \mathcal{PV}(S)$, then $\alpha \Vdash_H \tau(p)$ iff $\alpha \Vdash_{H'} p$. By definition of $\underline{K}_H$, it follows $\underline{K}_H \rhd (S \setminus \{H\})_c \cup \mathcal{R}_1(H))$; (ii) for every $H \in S \cap \mathcal{C}_4$, we have two cases: (iia)$\textsc{Tab}((S \setminus \{H\})_c \cup \mathcal{R}_1(H)) = \texttt{NULL}$, thus by induction hypothesis there exists a Kripke model $\underline{K}_H = \langle P_H, \leq_H, \rho_H, \Vdash_H \rangle$ such that $\rho_H \rhd (S \setminus \{H\})_c \cup \mathcal{R}_1(H)$. (iib) there exist a permutation τ from $\mathcal{PV}(S)$ to $\mathcal{PV}(S)$ and a swff $H' \in S \cap \mathcal{C}_4$ such that $\textsc{Tab}((S \setminus \{H'\})_c \cup \mathcal{R}_1(H')) = \texttt{NULL}$ and $(S \setminus \{H'\})_c \cup \mathcal{R}_1(H') = \tau((S \setminus \{H\})_c \cup \mathcal{R}_1(H))$. Thus by Point (a) applied to H', there exists a Kripke model $\underline{K}_{H'} = \langle P_{H'}, \leq_{H'}, \rho_{H'}, \Vdash_{H'} \rangle$ such that $\rho_{H'} \rhd (S \setminus \{H'\})_c \cup \mathcal{R}_1(H')$. By using τ we can translate $\underline{K}_{H'}$ into a model $\underline{K}_H = \langle P_H, \leq_H, \rho_H, \Vdash_H \rangle$, where $P_H = P_{H'}$, $\leq_H = \leq_{H'}$, $\rho_H = \rho_{H'}$ and for every world $\alpha \in P_H$, if $p \in \mathcal{PV}(S)$, then $\alpha \Vdash_H \tau(p)$ iff $\alpha \Vdash_{H'} p$. By definition of $\underline{K}_H$, it follows $\underline{K}_H \rhd (S \setminus \{H\})_c \cup \mathcal{R}_1(H))$. Let $\underline{K} = \langle P, \leq, \rho, \Vdash \rangle$ be a Kripke model defined as follows:

$$P = \bigcup_{H \in S \cap (\mathcal{C}_3 \bigcup \mathcal{C}_4)} P_H \cup \{\rho\};$$

$$\leq = \bigcup_{H \in S \cap (\mathcal{C}_3 \bigcup \mathcal{C}_4)} \leq_H \cup \{(\rho, \alpha) | \alpha \in P\};$$

$$\Vdash \;\; = \;\; \bigcup_{H \in S \cap (\mathcal{C}_3 \bigcup \mathcal{C}_4)} \Vdash_H \cup \{(\rho, p) | \mathbf{T}p \in S\}.$$

By construction of $\underline{K}$, $\rho \rhd S$
Case 5: the instruction labeled **r5** *has been performed.* We point out that $S \cap \mathcal{C}_1 = S \cap \mathcal{C}_2 = S \cap \mathcal{C}_3 = S \cap \mathcal{C}_4 = S \cap \mathcal{C}_5 = \emptyset$. By induction hypothesis there exists a model $\underline{K}_H = \langle P_H, \leq_H, \rho_H, \Vdash_H \rangle$ such that $\rho_H \rhd (S \setminus \{H\})_c \cup \mathcal{R}_2(H)$, where $H \in \mathcal{C}_6$. Let $\underline{K} = \langle P, \leq, \rho, \Vdash \rangle$ be a Kripke defined as follows: $P = P_H \cup \{\rho\}$, $\leq = \leq_H \cup \{(\rho, \alpha) | \alpha \in P\}$, $\Vdash = \Vdash_H \cup \{(\rho, p) | \mathbf{T}p \in S\}$. By the construction of $\underline{K}$, $\rho \rhd S$, in particular, by induction hypothesis $\rho_H \Vdash \neg B$ and therefore $\rho_H \Vdash \neg(A \wedge B)$. This implies $\rho \rhd \mathbf{F_c}(A \wedge B)$.
Case 6: the instruction **r6** *has been carried out.* In this case S contains atomic swffs and swffs of the kind $\mathbf{T}(p \to A)$ and with $\mathbf{T}p \notin S$. Let $\underline{K} = \langle P, \leq, \rho, \Vdash \rangle$ be the Kripke model such that $P = \{\rho\}$ and $\rho \Vdash p$ iff $\mathbf{T}p \in S$. It is easy to show that $\rho \rhd S$. As a matter of fact, if $\mathbf{T}(p \to A) \in S$, since $\mathbf{T}p \notin S$, $\rho \nVdash p$ therefore $\rho \Vdash p \to A$. □

By Lemma 4.1 we immediately get the completeness of Tab.

Theorem 4.2 (Completeness) *If* $A \in \mathbf{Int}$*, then* $\textsc{Tab}(\{\mathbf{F}A\})$ *returns a closed proof table starting from* $\mathbf{F}A$.

5 Implementation and Results

We have implemented Tab as an iterative procedure in C++ language. At present there are some features that are missing. First, there is no any kind of lexical normalization. This feature, together with backjumping ([Bak95]) and BCP ([Fre95]), only to give a partial list of the possible optimization techniques, is typical in theorem provers and will be one of the changes in the new version of the implementation. Moreover, when PITP applies $\mathcal{C}_3$ and $\mathcal{C}_4$-rules, the search for a permutation proceeds as follows: let S be a set of swffs and let H and H' be $\mathcal{C}_3$-swffs in S. PITP does not perform a full search among the $Pv(S)!$ possible permutations. PITP tries to build a permutation τ such that $H = \tau(H')$ and $\tau = \tau^{-1}$. If such a τ fulfills $(S \setminus \{H\})_c \cup \mathcal{R}_1(H) = \tau((S \setminus \{H'\})_c \cup \mathcal{R}_1(H'))$, then τ is used. Otherwise PITP considers that no permutation exists and solves $(S \setminus \{H'\})_c \cup \mathcal{R}_1(H')$. Analogously for $\mathcal{C}_4$-swffs. Since our search is narrow, many permutations are disregarded. This problem is made worse by the fact that conjunctions and disjunctions are not implemented as lists of formulas. Thus at present this optimization is applied only to two families of formulas of ILTP v1.1.1 library. Finally PITP does not implement the search for a permutation in Points 3 and 6 of Tab. Despite the fact that some optimizations are

	ft Prolog	ft C	LJT	STRIP	PITP
solved	188	199	175	202	215
(%)	68.6	72.6	63.9	73.7	78.5
proved	104	106	108	119	128
refuted	84	93	67	83	87
solved after:					
0-1s	173	185	166	178	190
1-10s	5	6	4	11	10
10-100s	6	7	2	11	9
100-600s	4	1	3	2	6
(>600s)	86	75	47	43	58
errors	0	0	52	29	1

Table 2: ft Prolog, ft C, LJT, STRIP and PITP on ILTP v1.1.1 formulas

	SYJ202+1 provable	SYJ205+1 provable	SYJ206+1 provable	SYJ207+1 refutable	SYJ208+1 refutable	SYJ209+1 refutable	SYJ211+1 refutable	SYJ212+1 refutable
ft Prolog	07 (516.55)	08 (60.26)	10 (144.5)	07 (358.05)	08 (65.41)	10 (543.09)	04 (66.62)	20 (0.01)
ft C	07 (76.3)	09 (85.84)	11 (481.98)	07 (51.13)	17 (81.41)	10 (96.99)	04 (17.25)	20 (0.01)
LJT	02 (0.09)	20 (0.01)	05 (0.01)	03 (2.64)	08 (0.18)	10 (461.27)	08 (546.46)	07 (204.98)
STRIP	06 (11.28)	14 (267.39)	20 (37.64)	04 (9.3)	06 (0.24)	10 (132.55)	09 (97.63)	20 (36.79)
PITP	09 (595.79)	20 (0.01)	20 (4.07)	04 (11.11)	08 (83.66)	10 (280.47)	20 (526.16)	11 (528.08)

Table 3: ILTP v1.1.1 formulas solved by classes

missing, results in Table 2 (this table is taken from `http://www.iltp.de/download/-ILTP-v1.1.1-prop-comparison.txt`) shows that PITP outperforms the known theorem provers on ILTP v1.1.1 library. Within 10 minutes PITP decides 215 out of 274 formulas of ILTP v1.1.1 The library divides the formulas in several families. Every family contains formulas sharing the same pattern of increasing complexity. In Table 3

	SYJ201+1	SYJ202+1	SYJ207+1	SYJ208+1	SYJ209+1	SYJ211+1	SYJ212+1
PITP none	20 (1.29)	03 (0.01)	04 (43.77)	04 (2.50)	10 (596.55)	20 (526.94)	11 (527.72)
PITP -opt1	20 (0.03)	08 (44.59)	04 (44.76)	08 (93.60)	10 (325.93)	20 (558.11)	11 (548.01)
PITP -opt2	20 (1.67)	03 (0.01)	04 (12.18)	04 (2.37)	10 (311.37)	19 (293.34)	10 (88.92)
PITP -opt3	20 (0.03)	08 (44.21)	04 (11.36)	08 (94.30)	10 (591.68)	19 (291.18)	10 (92.05)
PITP ALL	20 (0.03)	08 (45.30)	04 (12.74)	08 (90.11)	10 (297.83)	19 (313.11)	10 (93.18)

Table 4: Comparison between PITP optimizations

(this table is taken from `http://www.iltp.de/`) for every family (some families of ILTP v1.1.1 are missing because they are decided within 1s by all provers) we report the index of the largest formula which every prover is able to decide within 600s CPU time and in parenthesis the CPU time necessary to solve such a formula. PITP solves all the formulas in three families and it is the best prover in three families (SYJ202+1, SYJ206+1, SYJ211+1), ft-C solves all the formulas of SYJ212+1 and it is the best prover in four families (SYJ207+1, SYJ208+1, SYJ210+1, SYJ212+1), finally STRIP solves all the formulas in two families but in no class is it the best prover. Finally we run PITP on our Xeon 3.2GHz machine to evaluate the effect of using the optimizations described above. It is well known to people working in ATP that an optimization can be effective for one class of formulas and be negative for other classes. In Table 4 we compare different optimizations and give the results of their use on some classes of ILTP v1.1.1 formulas. First, PITP without optimizations outperforms the other versions of PITP on the families SYJ211+1 and SYJ212+1. To give an idea of the overhead of the optimizations on formulas where such optimizations do not apply, PITP without optimizations solves the 19th formula of SYJ211+1 in 247.19 seconds and the 10th formula of SYJ212+1 in 76.55 seconds. Among the optimizations the most important seems **opt2**. When **opt2** is not active the performances decrease. Thus even if this optimization can be used only on particular classes of formulas, it dramatically influences the performances (in our opinion this gives an idea of the great importance of bounding branching in propositional intuitionistic logic). With regard to the other optimizations, there are some advantages in some classes and disadvantages in others. In table 5 we provide the results of the comparison between PITP and STRIP on twelve thousand random formulas with three hundred connectives (since the performance of ft-C was worse than STRIP and PITP, table 5 lacks of ft-C). Given the time limit of five minutes, STRIP does not decide 780 formulas, PITP does not decide 16 formulas.

References

[AFF+06] A. Avellone, M. Ferrari, C. Fiorentini, G. Fiorino, and U. Moscato. Esbc: an application for computing stabilization bounds. *ENTCS*, 153(1):23–33, 2006.

[AFM04] A. Avellone, G. Fiorino, and U. Moscato. A new $O(n \lg n)$-space decision procedure for propositional intuitionistic logic. In Andrei Voronkov Matthias Baaz, Johann Makowsky, editor, *LPAR 2002: Short Contributions, CSL 2003: Extended Posters*, volume VIII of *Kurt Gödel Society, Collegium Logicum*, 2004.

[Bak95] A. B. Baker. *Intelligent Backtracking on Constraint Satisfaction Problems: Experimental and Theoretical Results*. PhD thesis, University of Oregon, 1995.

[BC04] Yves Bertot and P. (Pierre) Castéran. *Interactive theorem proving and program development: Coq'Art: the calculus of inductive constructions*. Texts in theoretical computer science. Springer-Verlag, pub-SV:adr, 2004.

[Con86] R. Constable. *Implementing Mathematics with the Nuprl Proof Development System*. Prentice–Hall, Englewood Cliffs, New Jersey, 1986.

[CZ97] A. Chagrov and M. Zakharyaschev. *Modal Logic*. Oxford University Press, 1997.

[Dyc92] R. Dyckhoff. Contraction-free sequent calculi for intuitionistic logic. *Journal of Symbolic Logic*, 57(3):795–807, 1992.

[ES99] Uwe Egly and Stephan Schmitt. On intuitionistic proof transformations, their complexity, and application to constructive program synthesis. *Fundam. Inform.*, 39(1-2):59–83, 1999.

[FFO02] M. Ferrari, C. Fiorentini, and M. Ornaghi. Extracting exact time bounds from logical proofs. In A. Pettorossi, editor, *Logic Based Program Synthesis and Transformation, 11th International Workshop, LOPSTR 2001, Selected Papers*, volume 2372 of *Lecture Notes in Computer Science*, pages 245–265. Springer-Verlag, 2002.

[Fit69] M.C. Fitting. *Intuitionistic Logic, Model Theory and Forcing*. North-Holland, 1969.

[Fre95] Jon William Freeman. *Improvements to propositional satisfiability search algorithms*. PhD thesis, University of Pennsylvania, 1995.

[Hud93] J. Hudelmaier. An $O(n \log n)$-SPACE decision procedure for intuitionistic propositional logic. *Journal of Logic and Computation*, 3(1):63–75, 1993.

[Men00] M. Mendler. Timing analysis of combinational circuits in intuitionistic propositional logic. *Formal Methods in System Design*, 17(1):5–37, 2000.

[ML84] P. Martin-Löf. *Intuitionistic Type Theory*. Studies in Proof Theory. Bibliopolis, Napoli, 1984.

[MMO97] P. Miglioli, U. Moscato, and M. Ornaghi. Avoiding duplications in tableau systems for intuitionistic logic and Kuroda logic. *Logic Journal of the IGPL*, 5(1):145–167, 1997.

[ROK06] Thomas Raths, Jens Otten, and Christoph Kreitz. The ILTP problem library for intuitionistic logic. release v1.1. *To appear in Journal of Automated Reasoning*, 2006.

	provable			
	0-1s	1-10s	10-200s	200-300s
STRIP	200	19	15	0
PITP	192	36	11	2

	unprovable			
	0-1s	1-10s	10-200s	200-300s
STRIP	10139	404	394	49
PITP	11663	49	27	4

	$> 300s$
STRIP	780
PITP	16

Table 5: PITP and STRIP on random generated formulas

[Sta79] R. Statman. Intuitionistic logic is polynomial-space complete. *Theoretical Computer Science*, 9(1):67–72, 1979.

[Wei98] Klaus Weich. Decision procedures for intuitionistic propositional logic by program extraction. In *TABLEAUX*, pages 292–306, 1998.