

details). As an example consider

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

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 $S_1 = (S \setminus \{H\}) \cup \{\mathbf{TP0}\}$ and $S_2 = (S \setminus \{H\}) \cup \{\mathbf{TP1}\}$ are the subsequent sets of S . Now consider S_1 . Since $\sigma_{\delta_{S_1}}$ realizes $\mathbf{TP0}$ 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 $S_3 = (S_1 \setminus \{H\}) \cup \{\mathbf{TP2}\}$ and $S_4 = (S_1 \setminus \{H\}) \cup \{\mathbf{TP3}\}$ 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 $S = \{\mathbf{F}(A \rightarrow B), \mathbf{T}((A \rightarrow B) \rightarrow C), \mathbf{FC}, \mathbf{TD}\}$. 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{TD} but $\underline{K}(S)$ does not realize S . To prove the realizability of S , the realizability of $S_1 = \{\mathbf{TA}, \mathbf{FB}, \mathbf{T}((A \rightarrow B) \rightarrow C), \mathbf{TD}\}$ and one between $S_2 = \{\mathbf{TA}, \mathbf{Fp}, \mathbf{T}(B \rightarrow p), \mathbf{T}(p \rightarrow C), \mathbf{TD}\}$ and $S_3 = \{\mathbf{F}(A \rightarrow B), \mathbf{TC}, \mathbf{FC}, \mathbf{TD}\}$ 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 $\alpha \leq \rho$, $\alpha \Vdash D$ and $\alpha \Vdash A$ such that $\underline{K}(S_1) \triangleright S_1$. If we glue $\underline{K}(S_1)$ 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 $\underline{K} \triangleright S$, we do not need to apply $\mathbf{T} \rightarrow \rightarrow$ to S in order to obtain $S_2 = \{\mathbf{TA}, \mathbf{Fp}, \mathbf{T}(B \rightarrow p), \mathbf{T}(B \rightarrow C), \mathbf{TD}\}$ (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 $\{H_1, \dots, H_u\} \subseteq S$ the \mathcal{C}_3 and \mathcal{C}_4 -swffs of S . If there exists a H_j such that $\underline{K} \not\triangleright H_j$, then $\text{Rule}(H_j)$ 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 $\underline{K}_i \triangleright \{H_1, \dots, H_u\}$ 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 $S_1 = \{\mathbf{TA}, \mathbf{FB}\}$. 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 $S_2 = \{\mathbf{TC}, \mathbf{FD}\}$. 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

$$S = \{ \mathbf{T}(((P0 \rightarrow (P1 \vee P2)) \rightarrow (P1 \vee P2))), \\ \mathbf{T}(((P2 \rightarrow (P1 \vee P0)) \rightarrow (P1 \vee P0))), \\ \mathbf{T}(((P1 \rightarrow (P2 \vee P0)) \rightarrow (P2 \vee P0))), \mathbf{F}((P1 \vee (P2 \vee P0))) \},$$

where only a few steps are needed to obtain S starting from $\{\mathbf{FH}\}$, 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 $\underline{K}(S_2) \triangleright S_2$ 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, $S_3 = (S \setminus \{H\})_c \cup \mathcal{R}_1(H)$ 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 $\beta \Vdash P2$. $\underline{K}(S_3) \triangleright S_3$. From S_5 we define the Kripke model $\underline{K}(S_5) = \langle \{\gamma\}, \leq, \gamma, \Vdash \rangle$ where $\gamma \leq \gamma$ and $\gamma \Vdash P1$. $\underline{K}(S_5) \triangleright S_5$. 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 , $\underline{K} \triangleright H$ does not imply $\underline{K} \triangleright \tau(H)$. 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 $\mathbf{T}(A \rightarrow B)$. 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 $\underline{p} = \{p_1, \dots, p_n\}$ 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 $\text{TAB}((S \setminus \{H\}) \cup \mathcal{R}_1(H))$ returns a proof π , then TAB returns the proof $\frac{S}{\pi} \text{Rule}(H)$, 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 $\delta_S \not\triangleright H$, then TAB returns (the proof) S . Otherwise, let H be a *cle*-swff such that $\sigma_{\delta_S} \not\triangleright H$, 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 $\frac{S}{\pi_1 \mid \tau^{-1}(\pi_1)} \text{Rule}(H)$. 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 $\frac{S}{\pi_1 \mid \pi_2} \text{Rule}(H)$, otherwise (π_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 $\sigma_{\delta_S} \triangleright S$, 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 $\{H_1, \dots, H_n\}$ be all the \mathcal{C}_3 -swffs in S . For $i = 1, \dots, n$, the following instructions are iterated: if there is no swff H_j ($j \in \{1, \dots, i-1\}$) 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 $\frac{S}{\pi} \text{Rule}(H_i)$;

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

(4.2.1) If there is neither swff H_j ($j \in \{1, \dots, i-1\}$) 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 $j \in \{1, \dots, i-1\}$, 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 $\frac{S}{\pi_1 \mid \pi_{2,i}} \text{Rule}(H_i)$.

(4.2.2) If Point (**4.2.1**) does not hold, then there exists a permutation τ and a swff H_j ($j \in \{1, \dots, i-1\}$) 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 $u \in \{1, \dots, i-1\}$, 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 $\frac{S}{\pi_1 \mid \tau^{-1}(\pi_{2,j})} \text{Rule}(H_i)$.

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 $\frac{S}{\pi} \text{Rule}(H)$, 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 $\frac{S}{\pi_1 \mid \tau^{-1}(\pi_1)} \text{Rule}(H)$.

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 $\frac{S}{\pi_1 \mid \pi_2} \text{Rule}(H)$, otherwise (π_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 “ $\sigma_{\delta_S} \triangleright S$ ”) 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 $\sigma_{\delta_S} \not\triangleright S$ we deduce $\delta_S \not\triangleright S$. Since $\delta_S \not\triangleright S$, 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 , $\text{Rule}(H)$ 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 $\text{Rule}(H_j)$ 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 $\rho \triangleright S$. To prove the proposition we need to introduce the function \deg defined as follows:

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

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 $\rho \triangleright S$.*

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 $\rho \triangleright S$.

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 $\rho \triangleright (S \setminus \{H\}) \cup \mathcal{R}_1(H)$, with $H \in \mathcal{C}_1$. We prove $\rho \triangleright H$ by proceeding according to the cases of H . If H is of the kind $\mathbf{T}(A \wedge B)$, then by induction hypothesis $\rho \triangleright \{\mathbf{T}A, \mathbf{T}B\}$, thus $\rho \Vdash A$ and $\rho \Vdash B$, and therefore $\rho \Vdash A \wedge B$. This implies $\rho \triangleright \mathbf{T}(A \wedge B)$. The other cases for $H \in \mathcal{C}_1$ are similar.

Case 2: the instruction labeled r2 has been performed. Thus $\sigma_{\delta_S} \triangleright S$ holds. We use σ_{δ_S} to define a Kripke model \underline{K} with a single world ρ such that $\rho \Vdash p$ iff $\sigma(p) = \text{true}$. Since ρ behaves as a classical model, $\rho \triangleright S$ holds.

Case 3: the instruction labeled r3 has been performed. By induction hypothesis there

exists a model \underline{K} such that $\rho \triangleright (S \setminus \{H_i\}) \cup \mathcal{R}_2(H_i)$, where $H_i \in \mathcal{C}_4$. Let us suppose that H is of the kind $\mathbf{T}((A \rightarrow B) \rightarrow C)$, thus $\rho \triangleright \mathbf{TC}$ and this implies $\rho \triangleright H_i$. The proof goes similarly if H_i is of the kind $\mathbf{T}(\neg A \rightarrow B)$.

*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) $\text{TAB}((S \setminus \{H\})_c \cup \mathcal{R}_1(H)) = \text{NULL}$, thus by induction hypothesis there exists a Kripke model $\underline{K}_H = \langle P_H, \leq_H, \rho_H, \Vdash_H \rangle$ such that $\rho_H \triangleright (S \setminus \{H\})_c \cup \mathcal{R}_1(H)$; (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 $\text{TAB}((S \setminus \{H'\})_c \cup \mathcal{R}_1(H')) = \text{NULL}$ 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 $\rho_{H'} \triangleright (S \setminus \{H'\})_c \cup \mathcal{R}_1(H')$. 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 $\underline{K}_H \triangleright (S \setminus \{H\})_c \cup \mathcal{R}_1(H)$; (ii) for every $H \in S \cap \mathcal{C}_4$, we have two cases: (iia) $\text{TAB}((S \setminus \{H\})_c \cup \mathcal{R}_1(H)) = \text{NULL}$, thus by induction hypothesis there exists a Kripke model $\underline{K}_H = \langle P_H, \leq_H, \rho_H, \Vdash_H \rangle$ such that $\rho_H \triangleright (S \setminus \{H\})_c \cup \mathcal{R}_1(H)$. (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 $\text{TAB}((S \setminus \{H'\})_c \cup \mathcal{R}_1(H')) = \text{NULL}$ 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 $\rho_{H'} \triangleright (S \setminus \{H'\})_c \cup \mathcal{R}_1(H')$. 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 $\underline{K}_H \triangleright (S \setminus \{H\})_c \cup \mathcal{R}_1(H)$. Let $\underline{K} = \langle P, \leq, \rho, \Vdash \rangle$ be a Kripke model defined as follows:

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

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

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

By construction of \underline{K} , $\rho \triangleright S$

*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 $\rho_H \triangleright (S \setminus \{H\})_c \cup \mathcal{R}_2(H)$, where $H \in \mathcal{C}_6$. Let $\underline{K} = \langle P, \leq, \rho, \Vdash \rangle$ be a Kripke defined as follows: $P = P_H \cup \{\rho\}$, $\leq = \leq_H \cup \{(\rho, \alpha) \mid \alpha \in P\}$, $\Vdash = \Vdash_H \cup \{(\rho, p) \mid \mathbf{T}p \in S\}$. By the construction of \underline{K} , $\rho \triangleright S$, in particular, by induction hypothesis $\rho_H \Vdash \neg B$ and therefore $\rho_H \Vdash \neg(A \wedge B)$. This implies $\rho \triangleright \mathbf{F}_c(A \wedge B)$.

*Case 6: the instruction **r6** has been carried out.* In this case S contains atomic swffs and swffs of the kind $\mathbf{T}(p \rightarrow A)$ 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 $\rho \triangleright S$. As a matter of fact, if $\mathbf{T}(p \rightarrow A) \in S$, since $\mathbf{T}p \notin S$, $\rho \not\Vdash p$ therefore $\rho \Vdash p \rightarrow A$. \square

By Lemma 4.1 we immediately get the completeness of TAB .

Theorem 4.2 (Completeness) *If $A \in \text{Int}$, then $\text{TAB}(\{\mathbf{F}A\})$ 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. Zakharyashev. *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.