fCube: An Efficient Prover for Intuitionistic Propositional Logic

Mauro Ferrari[1], Camillo Fiorentini[2], and Guido Fiorino[3]

[1] DICOM, Univ. degli Studi dell'Insubria, Via Mazzini 5, 21100, Varese, Italy
[2] DSI, Univ. degli Studi di Milano, Via Comelico, 39, 20135 Milano, Italy
[3] DIMEQUANT, Univ. degli Studi di Milano-Bicocca
P.zza dell'Ateneo Nuovo 1, 20126 Milano, Italy

Abstract. We present fCube, a theorem prover for Intuitionistic propositional logic based on a tableau calculus. The main novelty of fCube is that it implements several optimization techniques that allow to prune the search space acting on different aspects of proof-search. We tested the efficiency of our techniques by comparing fCube with other theorem provers. We found that our prover outperforms the other provers on several interesting families of formulas.

1 Introduction

fCube[1] is a theorem prover for Intuitionistic propositional logic based on a tableau calculus. The main topic of this paper is the description of the strategy and the main optimizations on which fCube relies. Here, speaking of optimization we mean a family of techniques that allow us to reduce the search space acting on different aspects of proof search. In particular fCube implements *simplification* techniques, which reduce the size of the formulas treated by the prover so to avoid inessential branching and backtracking.

Unlike what happened for classical logic, where optimization techniques of the above kind have been investigated from the very beginning (see, e.g., [3,6]), in the case of tableau calculi for Intuitionistic and non classical logics very little work has been done in this direction. As far as we know, the only works that address these issues in the context of tableau calculi are [5,7] that essentially refer to classical and modal logics, and [4] which addresses the case of Intuitionistic tableau calculi. The optimization rules implemented in fCube are those presented in [4]. We remark that fCube is a prototype Prolog implementation of the above techniques and we did very little work to "optimize the implementation"; e.g., fCube is based on a very rough implementation of the relevant data structures. In spite of this, as discussed in Section 5, fCube outperforms other provers on several interesting families of formulas. The above considerations suggest that the study of optimization techniques is a promising line of research to improve the performances of Intuitionistic theorem provers.

[1] Available at `http://web-nuovo.dimequant.unimib.it/~guidofiorino/fcube.jsp`

C. Fermüller and A. Voronkov (Eds.): LPAR-17, LNCS 6397, pp. 294–301, 2010.

2 Preliminaries and Tableau Calculus

We consider the language $\mathcal{L}$ based on a denumerable set of propositional variables $\mathcal{PV}$, the logical connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$ and the logical constants $\top$ and $\bot$.

We recall the main definitions about Kripke semantics (see, e.g., [2] for more details). A *Kripke model* for $\mathcal{L}$ is a structure $\underline{K} = \langle P, \leq, \rho, V\rangle$, where $\langle P, \leq, \rho\rangle$ is a poset with minimum ρ and V is a monotone function (the *valuation function*) mapping every $\alpha \in P$ to a subset of $\mathcal{PV}$. The forcing relation $\Vdash \subseteq P \times \mathcal{L}$ is defined as follows:

- $\alpha \Vdash \top$, $\alpha \nVdash \bot$ and, for $p \in \mathcal{PV}$, $\alpha \Vdash p$ iff $p \in V(\alpha)$;
- $\alpha \Vdash A \wedge B$ iff $\alpha \Vdash A$ and $\alpha \Vdash B$;
- $\alpha \Vdash A \vee B$ iff $\alpha \Vdash A$ or $\alpha \Vdash B$;
- $\alpha \Vdash A \rightarrow B$ iff, for every $\beta \in P$ such that $\alpha \leq \beta$, $\beta \Vdash A$ implies $\beta \Vdash B$;
- $\alpha \Vdash \neg A$ iff, for every $\alpha \leq \beta$, $\beta \nVdash A$ (i.e., $\beta \Vdash A$ does not hold).

Monotonicity property holds for arbitrary formulas, i.e., $\alpha \Vdash A$ and $\alpha \leq \beta$ imply $\beta \Vdash A$. A formula A *is valid in* $\underline{K}$ iff $\rho \Vdash A$. Intuitionistic propositional logic **Int** coincides with the set of formulas valid in all Kripke models [2].

The calculus implemented in fCube treats *signed formulas* of the kind $\mathbf{T}A$ or $\mathbf{F}A$, where $A \in \mathcal{L}$ The semantics of formulas extends to signed formulas as follows. Given a Kripke model $\underline{K} = \langle P, \leq, \rho, V\rangle$, $\alpha \in P$ and a signed formula H, α *realizes* H *in* $\underline{K}$ ($\underline{K}, \alpha \rhd H$) iff:

- $H \equiv \mathbf{T}A$ and $\alpha \Vdash A$;
- $H \equiv \mathbf{F}A$ and $\alpha \nVdash A$.

We say that $\underline{K}$ realizes H ($\underline{K} \rhd H$) iff $\underline{K}, \rho \rhd H$; H is *realizable* iff $\underline{K} \rhd H$ for some Kripke model $\underline{K}$. The above definitions extend in the obvious way to sets Δ of signed formulas; for instance, $\underline{K}, \alpha \rhd \Delta$ means that $\underline{K}, \alpha \rhd H$, for every $H \in \Delta$. By definition, $A \in$ **Int** iff $\mathbf{F}A$ is not realizable. We remark that, by the monotonicity property, $\mathbf{T}$-signed formulas are upward persistent ($\underline{K}, \alpha \rhd \mathbf{T}A$ and $\alpha \leq \beta$ imply $\underline{K}, \beta \rhd \mathbf{T}A$), while $\mathbf{F}$-signed formulas are downward persistent ($\underline{K}, \alpha \rhd \mathbf{F}A$ and $\beta \leq \alpha$ imply $\underline{K}, \beta \rhd \mathbf{F}A$).

fCube is based on the tableau calculus **Tab** of Fig.1[2]. In the formulation of the rules we use the notation Δ, H as a shorthand for $\Delta \cup \{H\}$. In the premise of a rule, writing Δ, H we assume that $H \notin \Delta$. Every rule applies to a set of signed formulas, but only acts on the signed formula H explicitly indicated in the premise; we call H the *major premise* of the rule, whereas we call all the other signed formulas *minor premises* of the rule. A rule r is *invertible* iff r is sound and, for every set Δ in the consequent, the realizability of Δ implies the realizability of the premise. A set Δ is *contradictory* iff either $\mathbf{T}\bot \in \Delta$ or $\mathbf{F}\top \in \Delta$ or, for some $A \in \mathcal{L}$, $\{\mathbf{F}A, \mathbf{T}A\} \subseteq \Delta$. A *proof-table* τ for Δ is defined as usual; when all the leaves of τ are contradictory, we say that τ is *closed* and Δ is provable.

As proved in [1], **Tab** is a complete for **Int**, that is $A \in$ **Int** iff $\mathbf{F}A$ is provable in **Tab**. The decision procedure described in Section 4 is inspired by [1].

[2] **Tab** essentially corresponds to the calculus of [1], the only difference is the absence of the sign $\mathbf{F_c}$. Here $\mathbf{F_c}A$ is replaced by the equivalent signed formula $\mathbf{T}\neg A$.

$$\frac{\Delta, \mathbf{T}(A \wedge B)}{\Delta, \mathbf{T}A, \mathbf{T}B}\,\mathbf{T}\wedge \qquad \frac{\Delta, \mathbf{F}(A \wedge B)}{\Delta, \mathbf{F}A \mid \Delta, \mathbf{F}B}\,\mathbf{F}\wedge \qquad \frac{\Delta, \mathbf{T}\neg(A \wedge B)}{\Delta_{\mathbf{T}}, \mathbf{T}\neg A \mid \Delta_{\mathbf{T}}, \mathbf{T}\neg B}\,\mathbf{T}\neg\wedge$$

$$\frac{\Delta, \mathbf{T}(A \vee B)}{\Delta, \mathbf{T}A \mid \Delta, \mathbf{T}B}\,\mathbf{T}\vee \qquad \frac{\Delta, \mathbf{F}(A \vee B)}{\Delta, \mathbf{F}A, \mathbf{F}B}\,\mathbf{F}\vee \qquad \frac{\Delta, \mathbf{T}\neg(A \vee B)}{\Delta, \mathbf{T}\neg A, \mathbf{T}\neg B}\,\mathbf{T}\neg\vee$$

$$\frac{\Delta, \mathbf{T}A, \mathbf{T}(A \to B)}{\Delta, \mathbf{T}A, \mathbf{T}B}\,MP \qquad \frac{\Delta_{\mathbf{T}}, \mathbf{T}(A \to B)}{\Delta_{\mathbf{T}}, \mathbf{T}\neg A \mid \Delta_{\mathbf{T}}, \mathbf{T}B}\,\mathbf{T}\to\text{-special}$$

$$\frac{\Delta, \mathbf{F}(A \to B)}{\Delta_{\mathbf{T}}, \mathbf{T}A, \mathbf{F}B}\,\mathbf{F}\to \qquad \frac{\Delta, \mathbf{T}\neg(A \to B)}{\Delta_{\mathbf{T}}, \mathbf{T}A, \mathbf{T}\neg B}\,\mathbf{T}\neg\to \qquad \frac{\Delta, \mathbf{F}\neg A}{\Delta_{\mathbf{T}}, \mathbf{T}A}\,\mathbf{F}\neg \qquad \frac{\Delta, \mathbf{T}\neg\neg A}{\Delta_{\mathbf{T}}, \mathbf{T}A}\,\mathbf{T}\neg\neg$$

$$\frac{\Delta, \mathbf{T}((A \wedge B) \to C)}{\Delta, \mathbf{T}(A \to (B \to C))}\,\mathbf{T}\to\wedge \qquad \frac{\Delta, \mathbf{T}(\neg A \to B)}{\Delta_{\mathbf{T}}, \mathbf{T}A \mid \Delta, \mathbf{T}B}\,\mathbf{T}\to\neg$$

$$\frac{\Delta, \mathbf{T}((A \vee B) \to C)}{\Delta, \mathbf{T}(A \to p), \mathbf{T}(B \to p), \mathbf{T}(p \to C)}\,\mathbf{T}\to\vee \quad \text{with } p \text{ a new atom}$$

$$\frac{\Delta, \mathbf{T}((A \to B) \to C)}{\Delta_{\mathbf{T}}, \mathbf{T}A, \mathbf{F}p, \mathbf{T}(p \to C), \mathbf{T}(B \to p) \mid \Delta, \mathbf{T}C}\,\mathbf{T}\to\to \quad \text{with } p \text{ a new atom}$$

$$\text{where } \Delta_{\mathbf{T}} = \{\mathbf{T}A \mid \mathbf{T}A \in \Delta\}$$

Fig. 1. The **Tab** calculus

3 Simplification Rules

In this section we describe the *simplification rules* implemented in fCube. The aim of these rules is to reduce the size of the formulas to be analyzed as much as possible before applying a rule of Fig. 1.

The first kind of simplification implemented in fCube exploits the well-known *boolean simplification rules* [4,7]. These rules simplify formulas containing the constants $\top$ and $\bot$ using Intuitionistic equivalences; e.g., $(A \vee \top) \wedge B$ simplifies to B, by the equivalences $A \vee \top \equiv \top$ and $B \wedge \top \equiv B$.

The other simplification rules used by fCube have been introduced in [4] and are described in Fig. 2. Given a signed formula H, $H[B/A]$ denotes the signed formula obtained by replacing every occurrence of A with B in H. Now, let Z, A and B be formulas; $Z\{B/A\}$ denotes the *partial substitution* of A with B in Z defined as follows: if $Z = A$ then $Z\{B/A\} = B$; if $Z = (X \odot Y)$ and $\odot \in \{\wedge, \vee\}$ then $Z\{B/A\} = X\{B/A\} \odot Y\{B/A\}$; if $Z = X \to Y$, $Z = \neg X$ or $Z \in \mathcal{PV}$ and $Z \neq A$, then $Z\{B/A\} = Z$. Note that partial substitutions do not act on subformulas under the scope of $\to$ or $\neg$. Given a signed formula $H = \mathcal{S}Z$, $H\{B/A\} = \mathcal{S}(Z\{B/A\})$. For a set of signed formulas Δ, $\Delta[B/A]$ (resp. $\Delta\{B/A\}$) is the set of signed formulas $H[B/A]$ (resp. $H\{B/A\}$) such that

$$\frac{\Delta, \mathbf{T}A}{\Delta[\top/A], \mathbf{T}A}\text{Replace-}\mathbf{T} \quad \frac{\Delta, \mathbf{T}\neg A}{\Delta[\bot/A], \mathbf{T}\neg A}\text{Replace-}\mathbf{T}\neg \quad \frac{\Delta, \mathbf{F}A}{\Delta\{\bot/A\}, \mathbf{F}A}\text{Replace-}\mathbf{F}$$

$$\frac{\Delta}{\Delta[\top/p]}\ \mathbf{T}\text{-perm if } p \preceq^{+} \Delta \quad \frac{\Delta}{\Delta[\bot/p]}\ \mathbf{T}\neg\text{-perm if } p \preceq^{-} \Delta \quad \frac{\Delta}{\Delta\{\bot/p\}}\ \mathbf{F}\text{-perm if } p \preceq^{-}_{w} \Delta$$

- $p \preceq^{-} \mathbf{F}p$ and $p \preceq^{+} \mathbf{T}p$
- $p \preceq^{l} \mathcal{S}\top$ and $p \preceq^{l} \mathcal{S}\bot$
- $p \preceq^{l} \mathcal{S}q$, where $q \in \mathcal{PV}$ and $q \neq p$
- $p \preceq^{l} \mathcal{S}(A \odot B)$ iff $p \preceq^{l} \mathcal{S}A$ and $p \preceq^{l} \mathcal{S}B$, where $\odot \in \{\wedge, \vee\}$
- $p \preceq^{l} \mathbf{F}(A \to B)$ iff $p \preceq^{l} \mathbf{T}A$ and $p \preceq^{l} \mathbf{F}B$
- $p \preceq^{l} \mathbf{T}(A \to B)$ iff $p \preceq^{l} \mathbf{F}A$ and $p \preceq^{l} \mathbf{T}B$
- $p \preceq^{l} \mathbf{F}\neg A$ iff $p \preceq^{l} \mathbf{T}A$
- $p \preceq^{l} \mathbf{T}\neg A$ iff $p \preceq^{l} \mathbf{F}A$.
- $p \preceq^{-}_{w} \mathcal{S}\top$ and $p \preceq^{-}_{w} \mathcal{S}\bot$
- $p \preceq^{-}_{w} \mathbf{F}A$ and $p \preceq^{-}_{w} \mathbf{T}\neg A$ for every A
- $p \preceq^{-}_{w} \mathbf{T}q$, where $q \in \mathcal{PV}$ and $q \neq p$
- $p \preceq^{-}_{w} \mathbf{T}(A \odot B)$ iff $p \preceq^{-}_{w} \mathbf{T}A$ and $p \preceq^{-}_{w} \mathbf{T}B$, where $\odot \in \{\wedge, \vee\}$
- $p \preceq^{-}_{w} \mathbf{T}(A \to B)$ iff $p \preceq^{-}_{w} \mathbf{T}B$.

where $\mathcal{S} \in \{\mathbf{T}, \mathbf{F}\}$ and $l \in \{+, -\}$

Given a set of signed formulas Δ and $\preceq \in \{\preceq^{+}, \preceq^{-}, \preceq^{-}_{w}\}$, $p \preceq \Delta$ iff, for every $H \in \Delta$, $p \preceq H$.

Fig. 2. Simplification rules and polarities

$H \in \Delta$. The rules in the second line of Fig. 2 enable the substitution of $p \in \mathcal{PV}$ occurring with constant *polarity* in a set Δ (see the definition of $p \preceq^{+} H$, $p \preceq^{-} H$ and $p \preceq^{-}_{w} H$) with $\top$ or $\bot$. In [4] it is proved that:

Theorem 1. *The rules of Fig.2 are invertible.* □

Thus, simplification rules do not require backtracking.

4 FCUBE Strategy

Here we describe the main function F of FCUBE which implements the proof-search strategy (see Fig. 3). Let Δ be a set of signed formulas; $\text{F}(\Delta)$ returns either a proof for Δ or a countermodel $\underline{K}$ for Δ, namely a Kripke model $\underline{K}$ such that $\underline{K} \rhd \Delta$. We introduce some notations. Given $H \in \Delta$, $\mathcal{R}_H(\Delta)$ is the instance of the rule of **Tab** having H as major premise and $\Delta \setminus \{H\}$ as minor premises. By $\Delta_{\mathbf{F}}$ we denote the set of **F**-signed formulas of Δ. A *local formula* is a signed formula $\mathbf{F}L$ such that:

$$L ::= p \mid L \vee L \mid L \wedge A \mid A \wedge L \quad \text{where } p \in \mathcal{PV} \text{ and } A \text{ is any formula}$$

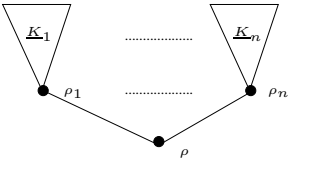
$\mathcal{LF}$ is the set of local formulas. An important property of $\mathcal{LF}$ is stated by the following theorem:

Theorem 2. *Let $\underline{K} = \langle P, \leq, \rho, V\rangle$ be a Kripke model, $\alpha \in P$ and $\mathbf{F}L \in \mathcal{LF}$. If $\alpha \nVdash p$ for every p occurring in L, then $\alpha \nVdash L$.* □

Function $\textsc{f}(\Delta)$

1. Apply to Δ the rules of Fig. 2 and boolean simplification rules as long as possible.
2. If Δ is a contradictory set, then return the proof $\pi = \Delta$.
3. If there exists $H \in \Delta$ such that $H \notin \mathcal{LF}$ and one of the rules $\mathbf{T}\wedge$, $\mathbf{T}\neg\neg$, MP, $\mathbf{T}\to\wedge$, $\mathbf{T}\to\vee$ and $\mathbf{F}\vee$ applies to H, let $\mathcal{R}_H(\Delta) = \dfrac{\Delta}{\Delta'}r$ and $\pi' = \textsc{f}(\Delta')$.
   If π' is a proof, then return the proof $\dfrac{\Delta}{\pi'}r$, else return the model π'.
4. If there exists $H \in \Delta$ such that $H \notin \mathcal{LF}$ and one of the rules $\mathbf{T}\vee$, $\mathbf{F}\wedge$, $\mathbf{T}\to -special$ applies to H, let $\mathcal{R}_H(\Delta) = \dfrac{\Delta}{\Delta' \mid \Delta''}r$, $\pi' = \textsc{f}(\Delta')$ and $\pi'' = \textsc{f}(\Delta'')$.
   If there is a model $\tau \in \{\pi', \pi''\}$ then return τ, else return the proof $\dfrac{\Delta}{\pi' | \pi''}r$.
5. Let $\Gamma_1 = \{\, H \in \Delta \mid H = \mathbf{F}(A \to B)$ or $H = \mathbf{F}\neg A \,\}$.
   Let $\Gamma_2 = \{\, K \in \Delta \mid K = \mathbf{T}((A \to B) \to C)$ or $K = \mathbf{T}(\neg A \to B) \,\}$.
   If $\Gamma_1 \cup \Gamma_2 \neq \emptyset$ then
   Let $\mathcal{M} = \emptyset$ ($\mathcal{M}$ is a set of Kripke models)
   For each $H \in \Gamma_1$ do
   Let $\mathcal{R}_H(\Delta) = \dfrac{\Delta}{\Delta'}r$ and $\pi' = \textsc{f}(\Delta')$.
   If π' is a proof then return the proof $\dfrac{\Delta}{\pi'}r$
   else if $\textsc{Real}(\pi', \Delta_{\mathbf{F}})$ then return π' else $\mathcal{M} = \mathcal{M} \cup \{\pi'\}$.
   For each $K \in \Gamma_2$ do
   Let $\mathcal{R}_K(\Delta) = \dfrac{\Delta}{\Delta' \mid \Delta''}r$, $\pi' = \textsc{f}(\Delta')$ and $\pi'' = \textsc{f}(\Delta'')$.
   If π'' is a model then return π''
   else if both π' and π'' are proofs, then return the proof $\dfrac{\Delta}{\pi' | \pi''}r$
   else if $\textsc{Real}(\pi', \Delta_{\mathbf{F}})$ then return π', else $\mathcal{M} = \mathcal{M} \cup \{\pi'\}$.
   Return the model $\textsc{Cm}(\Delta, \mathcal{M})$.
6. If there exists $H \in \Delta$ such that one of the rules $\mathbf{T}\neg\to$, $\mathbf{T}\neg\neg$ applies to H, let $\mathcal{R}_H(\Delta) = \dfrac{\Delta}{\Delta'}r$ and $\pi' = \textsc{f}(\Delta')$.
   If π' is a proof then return the proof $\dfrac{\Delta}{\pi'}r$, else return $\textsc{Cm}(\Delta, \{\pi\})$.
7. If $H = \mathbf{T}\neg(A \wedge B) \in \Delta$, let $\mathcal{R}_H(\Delta) = \dfrac{\Delta}{\Delta' \mid \Delta''}\mathbf{T}\neg\wedge$, $\pi' = \textsc{f}(\Delta')$ and $\pi'' = \textsc{f}(\Delta'')$.
   If there is a model $\tau \in \{\pi', \pi''\}$ then return $\textsc{Cm}(\Delta, \{\tau\})$, else return $\dfrac{\Delta}{\pi' | \pi''}\mathbf{T}\neg\wedge$.
8. Return $\textsc{Cm}(\Delta, \emptyset)$.

Function $\textsc{Cm}(\Delta, \mathcal{M})$
Let $\mathcal{M} = \{\underline{K}_1, \ldots, \underline{K}_n\}$, with $\underline{K}_i = \langle P_i, \leq_i, \rho_i, V_i\rangle$.
Let $\rho \notin \bigcup_{1\leq i\leq n} P_i$.
Return $\underline{K} = \langle P, \leq, \rho, V\rangle$ where:
$P = \{\rho\} \cup \bigcup_{1\leq i\leq n} P_i$
$\leq = \{\, (\rho, \alpha) \mid \alpha \in P \,\} \cup \bigcup_{1\leq i\leq n} \leq_i$
$V = \{\, (\rho, p) \mid \mathbf{T}p \in \Delta \,\} \cup \bigcup_{1\leq i\leq n} V_i$



The model $\textsc{Cm}(\Delta, \mathcal{M})$

Fig. 3. The functions $\textsc{f}(\Delta)$ and $\textsc{Cm}(\Delta, \mathcal{M})$

We also exploit the functions Cm and Real defined as follows:

- The function Cm (see Fig. 3) takes as input a set of signed formulas Δ and a (possibly empty) set of Kripke models $\mathcal{M}$ and builds a countermodel $\underline{K}$ for Δ using a standard technique to glue the models in $\mathcal{M}$ (see, e.g,. [2]).
- The function Real takes as input a Kripke model $\underline{K} = \langle P, \leq, \rho, V\rangle$ and a set of signed formulas Δ and returns true *only if* ρ realizes Δ, i.e. if $\underline{K}, \rho \rhd \Delta$. In our strategy Real is applied when the realizability of Δ can be decided only considering the valuation $V(\rho)$ (that is, without considering elements $\alpha > \rho$). In this case the test requires time linear in the size of Δ.

A high-level definition of f is given in Fig. 3. In the computation of $\mathrm{f}(\Delta)$, we firstly try to reduce Δ by applying the simplification rules described in Section 3. Note that this step can reduce the search space; for instance in Step 4, if the set $\Delta \cup \{\mathbf{T}(A \vee B)\}$ simplifies to $\Delta \cup \{\mathbf{T}B\}$, we avoid the call $\mathrm{f}(\Delta \cup \{\mathbf{T}A\})$. Applying **Tab** rules, f gives precedence to invertible rules and, among them, single-conclusion rules are applied first. We point out that $\mathcal{LF}$-formulas are treated as atomic formulas since they are never decomposed by the **Tab** rules (they can be treated only by simplification rules), and this avoids useless computation. Finally, we remark that if one of the tests $\mathrm{Real}(\pi', \Delta_{\mathbf{F}})$ in Step 5 successes, the iteration terminates and f returns the model π'.

We briefly account on the correctness of f. It is easy to check, by induction on Δ, that whenever $\mathrm{f}(\Delta)$ returns a proof π, π is actually a proof of Δ. Let us assume that $\mathrm{f}(\Delta)$ returns a Kripke model $\underline{K} = \langle P, \leq, \rho, V\rangle$; we have to prove that (*) $\underline{K}, \rho \rhd \Delta$. If $\underline{K}$ is returned in Step 3 or 4, (*) immediately follows. Let us consider Step 5. Suppose that $\underline{K} = \pi'$ is returned inside one of the for-each loops. By induction hypothesis $\underline{K}, \rho \rhd \Delta'$, which implies $\underline{K}, \rho \rhd \Delta_{\mathbf{T}}$. Moreover, being $\mathrm{Real}(\underline{K}', \Delta_{\mathbf{F}})$ true, we also have $\underline{K}, \rho \rhd \Delta_{\mathbf{F}}$, hence (*) holds. Suppose now that $\underline{K} = \mathrm{Cm}(\Delta, \mathcal{M})$. Then, (*) follows by construction of $\underline{K}$ and the induction hypothesis. We only show that $\underline{K}, \rho \rhd \mathbf{F}A$, for every $\mathbf{F}A \in \Delta$. Let $\mathbf{F}A = \mathbf{F}(B \to C)$. Since $\mathbf{F}A \in \Gamma_1$, there is $\underline{K}' = \langle P', \leq', \rho', V'\rangle \in \mathcal{M}$ such that $\underline{K}', \rho' \rhd \mathbf{T}B$ and $\underline{K}', \rho' \rhd \mathbf{F}C$. Since $\rho < \rho'$ in $\underline{K}$, we have $\underline{K}, \rho \rhd \mathbf{F}(B \to C)$. If $A \in \mathcal{PV}$, then $\mathbf{T}A \notin \Delta$ (Step 2 guarantees that Δ is not contradictory), hence $\underline{K}, \rho \rhd \mathbf{F}A$. It only remains to consider the case $\mathbf{F}A \in \mathcal{LF}$. Note that in Step 1 all the $p \in \mathcal{PV}$ such that $\mathbf{T}p \in \Delta$ have been replaced by $\top$, hence for every p occurring in A, we have $\underline{K}, \rho \rhd \mathbf{F}p$. By Theorem 2 we get $\underline{K}, \rho \rhd \mathbf{F}A$. This concludes the proof of (*). The discussion about steps 6–8 is similar. Note that in Step 8 the set Δ only contains formulas of the kind $\mathbf{T}\top$, $\mathbf{F}\bot$, $\mathbf{T}p$, $\mathbf{F}p$, with $p \in \mathcal{PV}$, $\mathbf{T}(p \to A)$, with $\mathbf{T}p \notin \Delta$, and $\mathcal{LF}$-formulas; in this case the returned model $\mathrm{Cm}(\Delta, \emptyset)$ is a classical model for Δ.

The termination of f follows from the fact that at each recursive call the size of Δ strictly decreases.

5 Evaluation and Conclusions

We have performed some experiments to compare fCube to Imogen [8], which is the fastest among the provers tested on the formulas of the ILTP Library [9], and

Formula	Imogen	fCube	Basic	+BackT	+Branch
SYJ201+1.018	11.32	14.46	timeout	timeout	16.16
SYJ201+1.019	16.28	17.84	timeout	timeout	20.72
SYJ201+1.020	17.00	22.34	timeout	timeout	26.00
SYJ202+1.006	timeout	6.18	timeout	timeout	7.08
SYJ202+1.007	timeout	52.98	timeout	timeout	61.18
SYJ202+1.008	timeout	529.36	timeout	timeout	570.88
SYJ206+1.018	2.26	0.00	0.00	0.00	0.00
SYJ206+1.019	2.12	0.00	0.00	0.00	0.01
SYJ206+1.020	2.14	0.00	0.01	0.01	0.01
SYJ207+1.018	77.02	4.72	timeout	5.53	timeout
SYJ207+1.019	104.38	6.08	timeout	7.15	timeout
SYJ207+1.020	143.32	7.44	timeout	8.94	timeout
SYJ208+1.015	timeout	174.22	220.71	209.98	187.78
SYJ208+1.016	timeout	286.56	351.73	349.72	312.99
SYJ208+1.017	timeout	472.07	570.48	569.66	541.81
SYJ209+1.018	0.20	0.08	timeout	0.07	timeout
SYJ209+1.019	0.024	0.09	timeout	0.09	timeout
SYJ209+1.020	0.028	0.09	timeout	0.10	timeout
Nishimura.011	8.2	0.02	0.02	0.02	0.02
Nishimura.012	132	0.04	0.03	0.04	0.04
Nishimura.013	timeout	0.07	0.06	0.07	0.07

Fig. 4. Timings on ILTP library

to check how our optimizations affect the performances of fCube itself. In the experiments we considered formulas of the ILTP Library and some axiom-formulas characterizing intermediate logics. In Fig. 4, the second and third column describe the timings of Imogen and fCube, respectively. Times are expressed in seconds and the timeout is 600s[3]. We notice that fCube outperforms Imogen on the families SYJ202, SYJ206, SYJ207, SYJ208 and Nishimura. Imogen is slightly faster than fCube on the families SYJ201 and SYJ209. As regards the performances of fCube on the SYJ201 family, we emphasize that fCube strategy relies on PITP strategy [1] and PITP decides the formula SYJ201.20 in 0.01s. In this case the timings of fCube essentially depend on its rough data structures that do not handle efficiently multiple occurrences of the same formula. This highly affects the performances on formulas like SYJ201.20 $((\wedge_{i=0}^{40}(p_i \equiv p_{(i+1) \bmod 41} \rightarrow \wedge_{j=0}^{40} p_j)) \rightarrow \wedge_{j=0}^{40} p_j)$ where $\wedge_{j=0}^{40} p_j$ occurs 42-times. Indeed rewriting SYJ201.20 as $((q \equiv \wedge_{j=0}^{40} p_j \ \wedge \ \wedge_{i=0}^{40}(p_i \equiv p_{(i+1) \bmod 41} \rightarrow q)) \rightarrow q)$ fCube solves it in 2.23s while Imogen requires 33s. We also remark that the proof table for the latter formula contains 250 nodes, whereas the proof table for the original formula contains 246 nodes. This is a further clue that the lack of advanced data structures penalizes the strategy. According to these considerations we expect that the above techniques can highly improve the performances of provers using advanced data structures.

[3] Experiments performed on a Intel(R) Xeon(TM) CPU 3.00GHz, Linux OS.

The last three columns of Fig. 4 analyze how the various optimizations affect fCube performances. Here we denote with Basic the version of fCube in which the only optimizations applied are those performed in Step 1; hence in steps 3 and 4 $\mathcal{LF}$-formulas are decomposed according to tableau rules and in Step 5 the test Real is omitted. +Branch is Basic with the special treatment of $\mathcal{LF}$ formulas and +BackT is Basic with the Real test. Note that the decomposition of $\mathcal{LF}$-formulas according to tableau rules increase the branch degree of a proof and the lack of the Real-test increases the backtrack degree of proof-search. The timings show that Basic cannot decide the families SYJ201, SYJ202, SYJ207, SYJ209. +Branch decides the families SYJ201, SYJ202. +BackT decides the families SYJ207, SYJ209. The simplification rules of Fig. 2 also have a deep impact on the proof strategy as we showed in [4].

To conclude, fCube is a Prolog theorem prover for Intuitionistic propositional logic implementing some optimization techniques. In this paper we have briefly discussed the optimization techniques and we have shown how such optimizations can highly improve the performances of a tableau-based prover for Intuitionistic propositional logic. As a future work we aim to study further optimization techniques, their application to modal logics and the extension to the first-order case.

References

1. Avellone, A., Fiorino, G., Moscato, U.: Optimization techniques for propositional intuitionistic logic and their implementation. Theoretical Computer Science 409(1), 41–58 (2008)
2. Chagrov, A., Zakharyaschev, M.: Modal Logic. Oxford University Press, Oxford (1997)
3. Davis, M., Logemann, G., Loveland, D.: A machine program for theorem-proving. Communications of the ACM 5, 394–397 (1962)
4. Ferrari, M., Fiorentini, C., Fiorino, G.: Towards the use of simplification rules in intuitionistic tableaux. In: Gavanelli, M., Riguzzi, F. (eds.) CILC 2009: 24-esimo Convegno Italiano di Logica Computazionale (2009)
5. Hustadt, U., Schmidt, R.A.: Simplification and backjumping in modal tableau. In: de Swart, H. (ed.) TABLEAUX 1998. LNCS (LNAI), vol. 1397, pp. 187–201. Springer, Heidelberg (1998)
6. Davis, M., Putnam, H.: A computing procedure for quantification theory. Journal of the ACM 7, 201–215 (1960)
7. Massacci, F.: Simplification: A general constraint propagation technique for propositional and modal tableaux. In: de Swart, H. (ed.) TABLEAUX 1998. LNCS (LNAI), vol. 1397, pp. 217–231. Springer, Heidelberg (1998)
8. McLaughlin, S., Pfenning, F.: Imogen: Focusing the polarized inverse method for intuitionistic propositional logic. In: Cervesato, I., Veith, H., Voronkov, A. (eds.) LPAR 2008. LNCS (LNAI), vol. 5330, pp. 174–181. Springer, Heidelberg (2008)
9. Raths, T., Otten, J., Kreitz, C.: The ILTP problem library for intuitionistic logic. Journal of Automated Reasoning 31, 261–271 (2007)