How to reduce backtracking in propositional Intuitionistic logic

Guido Fiorino

Dipartimento di Informatica Sistemistica e Comunicazione, Università degli Studi di Milano-Bicocca, P.zza dell’Ateneo Nuovo 1, 20126 Milano, Italy

Abstract. We discuss a technique to reduce search space in propositional Intuitionistic logic. Our technique is a syntactical criterion based on the sign property and can be used by any kind of calculus.

1 Introduction

In this paper we continue our investigations to speed-up proof search in propositional Intuitionistic logic (Int). In [3] we have introduced a calculus that advances the well-known calculus of [8], in [5] we have identified conditions allowing one to replace propositional variables with logical constants and an implementation of these achievements is the prover fCube [4]. In the cited papers the focus is on the rules or on the logical apparatus as a whole and the strategy using the calculus is not considered. In this note we introduce a criterion applicable to any strategy to reduce the non-determinism in Int proof search. To sake of concreteness, we present our results applied to a specific tableau calculus. However, it will be clear that our results can be applied to any tableau or sequent calculus.

Every step of a deduction has multiple choices and if the proof search fails, then every step is a point where to backtrack because the order of application of the rules is relevant, that is, not permutable. However, there is a special kind of rules that does not require to backtrack, because if a rule of this kind is used and a proof is not found, then no proof exists. This kind of rules is called invertible: the lack of a proof for the conclusion of an invertible rule implies the lack of a proof for the premise. The order of application of invertible rules is irrelevant: different permutations of their application always brings to the same result. On the other hand, the non-invertible rules require to backtrack: the lack of a proof for the conclusion of a non-invertible rule does not imply the lack of a proof for the premise. The application of non-invertible rules is not permutable, because different permutations of their applications produce different results. Thus, if in a step of a deduction a non-invertible rule $R$ is applied and subsequently to this step no proof is found, then a different application of $R$ or the application of a different rule could bring to find a proof. Note that such application is useless if we already know that it does not bring to find a proof.

Thus, it is useful to find strategies to reduce as much as possible the application of non-invertible rules, because this shrinks the search space for a proof. The aim of this paper is to present a criterion to get this goal. This work can be
seen as an extension of [2] where the syntax of sequent calculus LJQ gives rise to a strategy allowing to avoid some backtracking steps. The criterion we present is strictly related to our previous investigations in [5]. For this reason, after the preliminaries, we first describe a logical rule that generalizes the permanence rules of [5]. The application of our rule is subjected to a syntactical condition which subsumes the syntactical condition for the permanence rules of [5]. We use this syntactical condition to design a decision procedure that under some syntactical conditions avoids the application of non-invertible rules. The main work is to prove that the decision procedure is complete, a result that we show by exploiting the model theoretic characterization of \textbf{Int} by means of Kripke models.

\section{Preliminaries}

We consider the propositional language $\mathcal{L}$ based on a denumerable set of propositional variables $\mathcal{P}$, the logical connectives $\land, \lor, \rightarrow$ and the logical constants $\top, \bot$. We refer to [1,6,7] for details about \textbf{Int} and tableau systems. For sake of concreteness we consider the terminating calculus in Figure 1 whose rules handle \textit{signed formulas}, namely formulas of $\mathcal{L}$ prefixed with one of the well-known signs $\top$ or $\bot$. The satisfiability of a signed formula $H$ in a world $\alpha$ of a Kripke model $K$ is defined as follows: $\alpha$ \textit{realizes $H$ in $K$} ($K, \alpha \models H$) \textit{iff}: (i) $H \equiv \top \alpha$ and $\alpha \models A$; (ii) $H \equiv F \alpha$ and $\alpha \models A$. A model $K$ realizes $H$ ($K \models H$) \textit{iff} $K, \alpha \models H$ for some $\alpha \in \mathcal{P}$; a formula $H$ is \textit{realizable} \textit{iff} $K \models H$ for some Kripke model $K$.

When possible, in the following sections we will omit notation and we write: $\alpha \models H$ and $\alpha \models \Delta$ in place of $K, \alpha \models H$ and $K, \alpha \models \Delta$ when it is clear from the context to which model the realizability relation refers to.

The tableau calculus of Figure 1 is adapted from the calculus presented in [9].

Our aim is to consider a simple tableau calculus to decide \textbf{Int}, whose deductions are always finite. Thus, differently from [9], calculus in Figure 1 does not handle negation and only employs the signs $\top$ and $\bot$. Calculus of Figure 1 has also similarities with Fitting’s tableau calculus [6]. In the rules of the calculus we distinguish the \textit{premise}, the set of formulas above the line and the \textit{consequence}, the set(s) of formulas below the line that we call \textit{conclusion(s)}, separated by a vertical line when the consequence of the rule contains two conclusions. In the premise, the \textit{main formula of the premise} is the formula whose connectives are in evidence, the other formulas are the \textit{minor formulas of the premise}. The formulas
in evidence in a conclusion are the main formulas of the conclusion. We say that a rule applies to a set of formulas $\Gamma$, if the premise of the rule can be instantiated with $\Gamma$. We implicitly always consider duplication-free instantiations, that is after the instantiation of the premise of a rule, the main formula premise does not occur in $\Delta$ (that is, it does not occur as minor formula premise). Given a non-atomic signed formula $H$, we denote with $\text{Rule}(H)$ the name of the rule whose main formula premise can be instantiated with $H$. A set $\Gamma$ of signed formulas is inconsistent if $\{\text{T}A, \text{FA}\} \in \Gamma$ or $\text{T} \perp \in \Gamma$ and $\Gamma$ is consistent in the other cases. A tableau for a formula $A$ is a tree obtained from the root $\{\text{FA}\}$ by subsequently instantiating the premise of a rule with consistent leaves. If all the leaves of a tableau are inconsistent, then we say that the tableau is closed, the tableau is a proof for $A$ and $A$ is provable.

The rules in Figure 1 are sound: the realizability of the premise implies that there exists a conclusion which is realizable. It is easy to prove that calculus in Figure 1 is also complete for $\text{Int}$: for every formula $A$, $A$ is provable iff $\text{FA}$ is not realizable. To decide the provability of $A$ it is sufficient to search for a closed tableau. An obvious algorithm tries to build a proof for $A$ by applying the rules in all possible ways. This simple method can be improved by noticing that some rules are invertible: the realizability of one of the sets in the conclusion implies the realizability of the premise. Thus backtracking is not required when an invertible rule is applied. This is a well-known strategy, also employed in [6,9]. We remark that such a strategy based on the calculi in Figure 1 and in [9] is terminating. In the following sections we aim to show that further restrictions to the search space of proofs are possible. Thus it should be possible to reduce the backtrack and preserve the completeness.

Our first result is a generalization of the permanence rules introduced in [5] to reduce the search space. When a propositional variable $p$ fulfills a syntactic constraint, the permanence rules allow to deduce a set where all the occurrences of $p$ are replaced with a logical constant. The syntactic constraint is defined in terms of positive and negative occurrence of a propositional variable $p$ in a signed formula $H$ by the relations $p^{\geq^+}H$ ($p$ positively occurs in $H$) and $p^{\geq^-}H$ ($p$ negatively occurs in $H$). Hereafter we use $\mathcal{S}$ to denote either $\text{T}$ or $\text{F}$. The definition of $p^{\leq^l}H$, with $l \in \{+, -, \}$, is by induction on the structure of $H$: 

(i) $p^{\leq^-} \text{F}p$ and $p^{\leq^+} \text{T}p$; 
(ii) $p^{\leq^+} \mathcal{S}^+T$ and $p^{\leq^-} \mathcal{S}^\perp$; 
(iii) $p^{\leq^+} \mathcal{S}^q$, where $q$ is any propositional variable such that $q \neq p$; 
(iv) $p^{\leq^l} \mathcal{S}(A \odot B)$ iff $p^{\leq^l} \mathcal{S}A$ and $p^{\leq^l} \mathcal{S}B$, where $\odot \in \{\land, \lor\};$ 
(v) $p^{\leq^l} \mathcal{F}(A \rightarrow B)$ iff $p^{\leq^l} \mathcal{T}A$ and $p^{\leq^l} \mathcal{F}B$; 
(v) $p^{\leq^l} \mathcal{T}(A \rightarrow B)$ iff $p^{\leq^l} \mathcal{F}A$ and $p^{\leq^l} \mathcal{T}B$. Given a set $\Delta$ of signed formulas, $p^{\leq^l} \Delta$ iff for every $H \in \Delta$, $p^{\leq^l} H$. In paper [5] it is proved that permanence rules given in the following are invertible:

$$\frac{\Delta}{\Delta[\top/p]} \text{ T-perm, provided } p^{\leq^+} \Delta$$

$$\frac{\Delta}{\Delta[\bot/p]} \text{ T} \rightarrow \bot\text{-perm, provided } p^{\leq^-} \Delta,$$

where $\Delta[A/B]$ is the result of replacing in $\Delta$ every occurrence of $B$ with $A$. Intuitively, these rules state that, given a set $\Delta$, if the syntactical constraint $p^{\leq^+} \Delta$ (resp. $p^{\leq^-} \Delta$) is fulfilled, then it is correct to replace every occurrence of $p$ in $\Delta$ with the logical constant $\top$ (resp. $\bot$).
3 An extension of the permanence rules

Now we aim to study a weaker condition for the correct application of rules \( T\text{-perm} \) and \( T \rightarrow \bot\text{-perm} \). By exploiting such a weaker condition, in next section we introduce a decision procedure allowing to reduce the search space in proof search for \( \text{Int} \). The procedure we describe does not need to employ the permanence rules as further rules of the logical apparatus. This means that the results we show are not tied neither to the calculus given in Figure 1 nor to the proof system: a decision procedure based on new tableau calculus or a sequent calculus can use our results to reduce the search space for a proof.

As first step, let us go back over the side conditions on the applicability of the permanence rules. To this aim we introduce a new notion of replacement. Let \( H \) be a signed formula and let \( p \) a propositional variable. We define replacement in a formula \( H \) of all positive occurrences of a propositional variable \( p \) with \( \top \), the formula denoted with \( H[\top/p] \), obtained from \( H \) as follows:

- if \( H = Tp \), then \( H[\top/p] = \top \top \);
- if \( H = Tq \), with \( q \in \mathcal{PV} \setminus \{p\} \), then \( H[\top/p] = H \);
- if \( H = Fq \), then \( H[\top/p] = H, q \in \mathcal{PV} \);
- if \( H = S(A_1 \odot A_2) \), then \( H[\top/p] = S(A'_1 \odot A'_2) \), with \( S \mathcal{A}_i = S \mathcal{A}_i[\top/p] \), for \( i = 1, 2 \), \( S \in \{T, F\} \) and \( \odot \in \{\land, \lor\} \);
- if \( H = F(A \rightarrow B) \), then \( H[\top/p] = F(A' \rightarrow B') \), where \( T \mathcal{A} = T \mathcal{A}[\top/p] \) and \( F \mathcal{B} = F \mathcal{B}[\top/p] \);
- if \( H = T(A \rightarrow B) \), then \( H[\top/p] = T(A' \rightarrow B') \), where \( F \mathcal{A} = F \mathcal{A}[\top/p] \) and \( T \mathcal{B} = T \mathcal{B}[\top/p] \).

The following are examples of replacement of \( q \) and \( p \) respectively:

\[
F(p \rightarrow (q \lor q \rightarrow \bot))[\top/q] = F(p \rightarrow (q \lor \top \rightarrow q));
\]

\[
F((p \lor p \rightarrow \bot) \rightarrow q)[\top/p] = F((\top \lor p \rightarrow \bot) \rightarrow q).
\]

Analogously, we define replacement in a formula \( H \) of all negative occurrences of a propositional variable \( p \) with \( \bot \), the formula denoted with \( H[\bot/p] \) obtained from \( H \) as follows:

- if \( H = Tq \), then \( H[\bot/p] = H \), if \( q \in \mathcal{PV} \);
- if \( H = Fp \), then \( H[\bot/p] = F \bot \);
- if \( H = Fq \), with \( q \in \mathcal{PV} \setminus \{p\} \), then \( H[\bot/p] = H \);
- if \( H = S(A_1 \odot A_2) \), then \( H[\bot/p] = S(A'_1 \odot A'_2) \), with \( S \mathcal{A}_i = S \mathcal{A}_i[\bot/p] \), for \( i = 1, 2 \), \( S \in \{T, F\} \) and \( \odot \in \{\land, \lor\} \);
- if \( H = F(A \rightarrow B) \), then \( H[\bot/p] = F(A' \rightarrow B') \), where \( T \mathcal{A} = T \mathcal{A}[\bot/p] \) and \( F \mathcal{B} = F \mathcal{B}[\bot/p] \);
- if \( H = T(A \rightarrow B) \), then \( H[\bot/p] = T(A' \rightarrow B') \), where \( F \mathcal{A} = F \mathcal{A}[\bot/p] \) and \( T \mathcal{B} = T \mathcal{B}[\bot/p] \).

The result of \( H[\top/p] \) (resp. \( H[\bot/p] \)) is a formula having the same sign of \( H \) and containing zero or more occurrences of the logical constant \( \top \) (resp. \( \bot \)). Hereafter we use \( [\cdot]/\cdot \)-replacement to refer to both kind of replacements defined above. We extend \( [\cdot]/\cdot \)-replacement to sets of signed formulas in the obvious way. If
Lemma 1. Let \( p \leq^+ \Delta \) (resp. \( p \leq^\Delta \)) then \( \Delta \models \top/p \) (resp. \( \Delta \models \bot/p \)) coincides with \( \Delta \models \top \) (resp. \( \Delta \models \bot \)). Given a signed formula \( H \) and a set of signed formulas \( \Delta \), we write \( H \models \Delta \) (resp. \( \models \Delta \)) to refer both formula \( H \models \top \) and \( H \models \bot \) (resp. set \( \Delta \models \top \) and \( \Delta \models \bot \)). Let \( A \) be a formula, \( \text{eval}(A) \) denotes the formula obtained by applying to \( A \) the usual boolean reductions based on the meaning of the logical constants. Given a signed formula \( S \), where \( S \in \{ \text{T}, \text{F} \} \), the evaluation of a signed formula \( S \), denoted as \( \text{eval}(S) \), is the signed formula \( \text{Seval}(A) \). The following is the evaluation of the formulas obtained from the previous example: \( \text{eval}(\text{T}(p \rightarrow (q \lor q \rightarrow \bot))) = \text{T}(p \rightarrow q) \); \( \text{eval}(\text{F}(p \lor (q \lor q \rightarrow \bot))) = \text{F}(p \lor q) \).

The invertibility of the boolean simplification rules implies that \( H \models \Delta \) is realizable iff \( \text{eval}(H \models \Delta) \) is realizable, with \( \Delta \in \{ \top, \bot \} \). The following Propositions 1 and 2 aim to give the whole picture of the relationship between the realizability of \( H \) and \( \text{eval}(H \models \Delta) \). We start with a technical lemma:

**Theorem 1.** The rules \(+\text{-indep}\) and \(-\text{-indep}\) are invertible.
\[ H[\top/p_1, \ldots, \top/p_n, \bot/q_1, \ldots, \bot/q_m] = H[\top/p, \bot/q], \text{ where } p \cap q = \emptyset, \]
as follows:

- if \( H = T v \), then \( H[\top/p, \bot/q] = \begin{cases} T \top \text{ if } v \in p \\ H \text{ otherwise} \end{cases} \)
- if \( H = F v \), then \( H[\top/p, \bot/q] = \begin{cases} F \bot \text{ if } v \in q \\ H \text{ otherwise} \end{cases} \)
- if \( H = S(A_1 \circ A_2) \), then \( H[\top/p, \bot/q] = S(A'_1 \circ A'_2) \),
  with \( S A'_i = S A_i[\top/p, \bot/q] \), for \( i = 1, 2 \), \( S \in \{ T, F \} \) and \( \circ \in \{ \wedge, \vee \} \);
- if \( H = F(A \rightarrow B) \), then \( H[\top/p, \bot/q] = F(A' \rightarrow B') \), where \( T A' = TA[\top/p, \bot/q] \) and \( F B' = FB[\top/p, \bot/q] \);
- if \( H = T(A \rightarrow B) \), then \( H[\top/p, \bot/q] = T(A' \rightarrow B') \), where \( F A' = FA[\top/p, \bot/q] \) and \( T B' = TB[\top/p, \bot/q] \).

In a similar way we can define the meaning of \( H[\top/p, \bot/q] \). The extension of these notions to sets of formulas is obvious. The rule

\[
\frac{\Delta}{\Delta[\top/p, \bot/q]} \text{ \(-\text{indep}, \) provided } (p \cup q) \cap \mathcal{P} \mathcal{V}(\text{eval}(\Delta[\top/p, \bot/q])) = \emptyset \text{ and } p \cap q = \emptyset
\]
generalizes +-indep and --indep and is invertible. When \( (p \cup q) \cap \mathcal{P} \mathcal{V}(\text{eval}(\Delta[\top/p, \bot/q])) = \emptyset \) holds we say that the set of variables \( p \cup q \) is independent in \( \Delta \). We remark that the intuitively obvious request \( p \cap q = \emptyset \) is necessary to prove the invertibility of \( \pm \)-indep, the analogous of Proposition 2. As a matter of fact, if \( v \in p \cap q \) we should build a Kripke model behaving as \( K^+ \) and \( K^- \), which is obviously impossible.

Rule \( \pm \)-indep is a consequence of the definition of []-replacement and it represents a generalization of the permanence rules: under the syntactical conditions stated for rule \( \pm \)-indep, it is possible to replace in a set \( \Delta \) the occurrences of a propositional variable \( p \) with a logical constant and such a replacement is correct also if neither \( p^+ \Delta \) nor \( p^- \Delta \) holds. Since variables are replaced by constants, rule \( \pm \)-indep is a mechanism to reduce the search space. Rule \( \pm \)-indep can be inserted in a procedure for deciding \( \mathbf{Int} \) and since permanence rules are subsumed by \( \pm \)-indep, they can be removed from the deductive system without any loss. To apply \( \pm \)-indep one has to find an independent set of propositional variables fulfilling the side condition for \( \pm \)-indep. This could be a computationally expensive task. We are not interested to use the rule in this way. To bound the backtracking in \( \mathbf{Int} \) proof search, the decision procedure we provide checks if a given set of propositional variables is independent in a set. When the condition is fulfilled, then rule \( \pm \)-indep is applicable, but the important point is that in this case some backtracking steps are avoidable. Summarizing, the independence of a set of propositional variables in a set of formulas is the key point we use to avoid the backtracking.

4 Reducing backtracking in \( \mathbf{Int} \)

Now we want to exploit []-replacement and Propositions 1 and 2 to avoid useless applications of non-invertible rules. Since we are going to avoid some rule appli-
cations, the question is to prove the decision procedure we provide is complete. To this aim we need to prove the following theorem:

**Theorem 2.** Let \( \Delta \) be a set of \( T \)-signed formulas and let \( K = \{ p, \leq, \rho, \models \} \) be a model such that \( \rho \models \Delta \) and \( \rho \models p \) iff \( Tp \in \Delta \). Let

\[
\Gamma = \Delta \cup \{ Tp_1, \ldots, Tp_n, Fq_1, \ldots, Fq_m, T(h_1 \rightarrow H_1), \ldots, T(h_l \rightarrow H_l) \}
\]

be a consistent set with \( p \cap h = \emptyset \), where \( p = \{ p_1, \ldots, p_n \} \) and \( h = \{ h_1, \ldots, h_l \} \).

If \( (p \cup h) \cap PV(\text{eval}(\Delta \cap T/p, \bot/h)) = \emptyset \) holds, then: 1. \( \Gamma \) is realizable by the Kripke model \( M = \langle P, \leq, \rho, \models \rangle \) such that \( \models h = (\models (\bot p) \setminus (P \times h)) \cap (P \times h) \); 2. if \( K, \rho \models F(A \rightarrow B) \) and \( (p \cup h) \cap PV(\text{eval}(F(A \rightarrow B)(\top/p, \bot/h))) = \emptyset \), then \( M, \rho \models F(A \rightarrow B) \).

Roughly speaking, the idea to avoid backtracking as applied in following Function Bb, can be explained as follows: let \( S \) and \( S' \) be two sets of formulas and \( M = \langle P, \leq, \rho, \models \rangle \) a Kripke model such that \( M, \rho \models S \). We are investigating under which syntactical conditions of \( S \) and \( S' \) we can change the forcing in \( M \) to get a model \( M' = \langle P, \leq, \rho, \models' \rangle \) such that \( M', \rho \models S' \). More precisely, we are asking under which syntactical conditions the realizability of \( S \) implies the realizability of \( S' \). This is of interest because, if an attempt to find a proof starting from \( S \) fails, then another attempt starting from another set has possibly to be done. The failed attempts witnesses that some sets are realizable by a Kripke model. Thus we wonder if the failed attempt implies that some other attempts we are going to try, for example starting from \( S' \), will fail. If we found a general criterion, computationally not expensive, to check if this case, then we can reduce the search space and speed-up the procedure: such a criterion is the side condition stated for rule \( \pm \)-indep. Function Bb works by locking sets of \( T \)-signed formulas. Locked formulas are not at disposal of the backtracking steps in the sense that Function Bb does not use locked formulas as main formulas. The idea behind the locking is to mark a set a formulas having a model. This avoids the useless backtracking generated by taking the marked formulas as main formula premise of the non-invertible rule \( T \rightarrow \rightarrow \). To decide a formula \( A \), the call \( \text{Bb}(\{ FA \}_., \_) \) is performed, where \( \_ \) emphasizes that the second parameter is irrelevant. Rules are applied according to a priority which is standard: \( T \rightarrow \rightarrow \) and \( F \lor \lor \) are rules of the lowest priority because if a proof is not found, they require to backtrack. Function Bb returns a proof or a structure that in completeness theorem is proved to be a Kripke model \( K = \langle P, \leq, \rho, \models \rangle \) such that \( \rho \models \Delta \). In the following, we call \( \alpha \)-rules \( T \wedge, T \rightarrow \wedge, T \rightarrow \lor, F \rightarrow \) and \( \beta \)-rules \( T \lor, F \wedge \).

**Function Bb(\( \Delta, M \))**

1. If \( \Delta \) is an inconsistent set, then return the proof \( \Delta \).
2. If the premise of MP can be instantiated with \( \Delta \), where \( TA \) and \( TA \rightarrow B \) can be locked or unlocked, then let \( \Delta_1 \) be the conclusion and let \( \pi = \text{Bb}(\Delta_1, M) \), where \( TB \) is unlocked in \( \Delta_1 \). If \( \pi \) is a proof, then return the proof \( \frac{\Delta}{\pi} \) MP, otherwise return \( \pi \).
3. If the premise of an \( \alpha \)-rule can be instantiated with \( \Delta \), then let \( H \) be the main formula premise, \( \Delta_1 \) the conclusion and \( \pi = \text{Bb}(\Delta_1, M) \). If \( \pi \) is a proof, then return \( \frac{\Delta}{\pi} \text{Rule}(H) \), otherwise return \( \pi \).
4. If the premise of a \( \beta \)-rule can be instantiated with \( \Delta \), then let \( H \) be the main premise, \( \Delta_1 \) and \( \Delta_2 \) the conclusions and, for \( i = 1, 2 \) let \( \pi_i = \text{Bb}(\Delta_i, M) \). If \( \pi_i \) is a structure,
with $i \in \{1, 2\}$, then return $\pi_i$, otherwise return the proof $\frac{\Delta}{\pi_1, \pi_2}$. 

5. (i) (only backtracking on $FV$-formula is required). If $\Delta_T$ is not empty and all the formulas in $\Delta_T$ are locked and $F(A_1 \lor A_2) \in \Delta$, then let $\Delta_i = \Delta_T \cup \{FA_i\}$ and $\pi_i = \text{Bn}(\Delta_i, M_i)$, for $i = 1, 2$. If $\pi_i$ is a proof, with $i \in \{1, 2\}$, then return the proof $\frac{\Delta}{\pi_i}$; otherwise, let $M_i = (P_{M_i, \leq M_i, \rho_{M_i}, \|M_i\})$ and $\pi_i = (P_i, \leq, \rho_i, \|\pi_i\}$, for $i = 1, 2$. Return the structure $K = (P_i, \leq, \rho, \|\pi_i\}$ defined as follows:

$P = P_{M_i} \cup P_1 \cup P_2$; $\leq = \leq_{M_i} \cup \leq_1 \cup \leq_2 \cup \{(\rho, \alpha) | \alpha \in P_1 \cup P_2\}$;

$\rho = \rho_{M_i}$; $\|\pi_i\| = \|P_{M_i} \cup \|\pi_i\| \cup \|\pi_i\|_2$.

(ii) (Full standard backtracking is required) If $\Delta_T$ contains an unlocked formula of the kind $T((A \rightarrow B) \rightarrow C)$ or, all formulas in $\Delta_T$ are unlocked and $F(A \lor B) \in \Delta$, then return $\text{BACKT}(\Delta, M)$.

(iii) If $\Delta_T$ contains both locked and unlocked formulas (note that the unlocked formulas are atomic or of the kind $T(p \rightarrow B)$), then let $\Phi = \{F(A \rightarrow B) | T((A \rightarrow B) \rightarrow C) \in \Delta\}$, let $\Delta_U = \{TA \in \Delta | TA \text{ is unlocked }\}$, $P = \{p | Tp \in \Delta_U\}$, $q = \{q | T(q \rightarrow H) \in \Delta_U\}$ and let $K = (P, \leq, \rho, \|\pi_i\}$ be the structure defined as follows:

$P = P_{M_i} \cup P_1 \cup P_2$; $\leq = \leq_{M_i} \cup \leq_1 \cup \leq_2 \cup \{(\rho, \alpha) | \alpha \in P_2\}$; $\|\pi_i\| = \|P_{M_i} \cup \|\pi_i\|_1 \cup \|\pi_i\|_2$.

(A) If $F(A_1 \lor A_2) \notin \Delta$ and $(p \lor q) \notin \mathcal{P}V(\text{eval}(\Delta | p \lor q))$, then return $K$.

(B) If $F(A_1 \lor A_2) \notin \Delta$ and $(p \lor q) \notin \mathcal{P}V(\text{eval}(\Delta | p \lor q))$, for $i = 1, 2$, let $\pi_i = \text{Bn}(\Delta_T \cup \{FA_i\}, K)$, where in the recursive call all formulas in $\Delta_T$ are locked.

If there exists $i \in \{1, 2\}$ such that $\pi_i$ is a proof, then return the proof $\frac{\Delta}{\pi_i}$, else let $\pi_i = (P_i, \leq, \rho_i, \|\pi_i\}$; return the structure $\langle P', \leq', \rho', \|\pi_i'\rangle$ defined as follows:

$P' = P \cup P_1 \cup P_2$; $\leq' = \leq_1 \cup \leq_2 \cup \{(\rho, \alpha) | \alpha \in P_1\}$; $\|\pi_i'\| = \|\pi_i\|_1 \cup \|\pi_i\|_2$.

(C) Unlock the formulas in $\Delta$ and return $\text{Bn}(\Delta, \pi_i)$.

6. (if we are here $F(A \lor B) \notin \Delta$. If all the formulas in $\Delta_T$ are locked, then (i) return $M_i$ else (ii) return $\{\rho\}, \{\rho, \rho\}, \{\rho, \rho \mid Tp \in \Delta\})$.

End Function $\text{Bn}$.

Function $	ext{BACKT}(\Delta, M)$

Let $(T((A_i \rightarrow B_i) \rightarrow C_i))_{1 \leq i \leq n} = \{T((A \rightarrow B) \rightarrow C) \in \Delta\}$;

for $i = 1, \ldots, n$

- let $\phi_i = \text{Bn}(\Delta \setminus \{T((A_i \rightarrow B_i) \rightarrow C_i)\}) \cup \{TC_i, M_i\}$, where in the recursive call $TC_i$ is not locked and the locking of the other formulas is left unchanged;

- if $\phi_i$ is a structure, then return $\phi_i$.

Unlock all the formulas in $\Delta$;

for $i = 1, \ldots, n$

- let $\pi_i = \text{Bn}(\Delta_T \setminus \{T((A_i \rightarrow B_i) \rightarrow C_i)\}) \cup \{TA_i, FB_i, T(B_i \rightarrow C_i)\}$;

- if $\pi_i$ is a proof, then return the proof $\frac{\Delta}{\pi_i}$.

Given the structures $\pi_i = (P_i, \leq, \rho_i, \|\pi_i\}$, with $i = 1, \ldots, n$, define $K = \langle P, \leq, \rho, \|\pi_i\}$ as follows:

$P = \rho \cup P_{C_{\pi_i}}$; $\leq = \leq_{\pi_i} \cup \{(\rho, \alpha) | \alpha \in P\}$; $\|\pi_i\| = \|\rho\|_1 \cup \{(\rho, \rho) \mid Tp \in \Delta\}$.

If $F(A_1 \lor A_2) \notin \Delta$, then return $K$.

For $i = 1, 2$

- let $\pi_{n+1} = \text{Bn}(\Delta_T \cup \{FA_i\}, K)$, where in the recursive call all the formulas in $\Delta_T$ are locked; if $\pi_{n+1}$ is a proof, then return the proof $\frac{\Delta}{\pi_{n+1}}$. 

8
Return the structure $\langle P, \leq, \rho, \| \rangle$ defined as follows:

$$P = \rho \cup \rho \cup \rho \cup \rho$$

$$\leq = \cup_{i=1}^{n+2} P_i; \quad \leq = \cup_{i=1}^{n+2} \leq, \cup \{\langle \rho, \alpha \rangle \mid \alpha \in P\}; \quad \| = \cup_{i=1}^{n+2} \|, \cup \{\langle \rho, p \rangle \mid Tp \in \Delta\}.$$

End Function BACKT.

Function Bb locks the $T$-formulas after the application of rule $F \lor_i$. When this stage is reached, the realizability of $\Delta_T$ has been proved by backtracking steps involving the non-invertible conclusion of the rule $T \rightarrow \rightarrow$ and $M$ is the model realizing $\Delta_T$. The formulas in $\Delta_T$ will be unlocked when a new $T \rightarrow \rightarrow$-formula is introduced. If a set $\Gamma$ only containing locked formulas, signed atomic formulas and $T(p \rightarrow H)$-formulas, with $Tp \not\in \Gamma$, is gotten, then, by Theorem 2, we can decide if the realizability of $\Delta_T$ implies the realizability of $\Gamma$. This check is performed in Step 5(iii)(A). By construction $\Delta_T \subseteq \Gamma$ and $\Delta_T$ coincides with the formulas of $\Gamma$ that are locked. Function Bb contains a call to Function BACKT.

Function BACKT implements the backtracking mechanism necessary to handle formulas of the kind $T((A \rightarrow B) \rightarrow C)$ and $F(A \lor B)$. This is a standard phase, necessary to guarantee the completeness of the decision procedure. If after all possible instantiations of the premise of $T \rightarrow \rightarrow$ with $\Delta$ no proof is found, then the structure $K$ is a Kripke model whose root realizes $\Delta_T \cup \{F(A \rightarrow B)|T((A \rightarrow B) \rightarrow C) \in \Delta\}$. At this point, Bb instantiates, if possible, the premise of $F \lor_1$ and $F \lor_2$ with $\Delta$. In the subsequent applications the formulas in $\Delta_T$ are locked, thus $T \rightarrow \rightarrow$-formulas are not used as main formulas. This is also the strategy employed by sequent calculus LJQ [2]. The strategy of Bb diverges from LJQ by the fact that after an application of $F \rightarrow$, LJQ unlocks the formulas, whereas Bb keeps them locked until the only applicable rules are $T \rightarrow \rightarrow$ or $F \lor$. This corresponds to reach Step 5(ii) or 5(iii). If a new $T \rightarrow \rightarrow$-formula appears, then Step 5(ii) is performed and BACKT is called. In BACKT, when the invertible conclusion of rule $T \rightarrow \rightarrow$ is handled, the locking of the minor premises is left unchanged. When the non-invertible conclusion of $T \rightarrow \rightarrow$ is used the locked formulas of $\Delta$ become unlocked, thus Bb behaves as LJQ. If Step 5(iii) is reached, then it means that subsequently to the locking of the formulas, new $T$-formulas have been added, but they are atomic or of the kind $T(p \rightarrow A)$ only.

In Steps 5(iii)(A) and 5(iii)(B) function Bb attempts to avoid backtracking steps. Bb performs a purely syntactic check involving $\|\$-replacement. In practice, Bb checks if the side condition on the applicability of rule $\pm$-indep is fulfilled. It has to be noted that the check in Step 5(iii)(A) is different from the check in Step 5(iii)(B). To prove the completeness for the case of Step 5(iii)(B) we needed to know that the model $K$ obtained from $M$ fulfills the property that for every $T((A \rightarrow B) \rightarrow C) \in \Delta$, the root of $K$ realizes $F(A \rightarrow B)$. Since by hypothesis the given model $M$ fulfills such a property, the check in Step 5(iii)(B) aims to prove that for every $T((A \rightarrow B) \rightarrow C) \in \Delta$, the realizability $F(A \rightarrow B)$ is independent of the forcing of the propositional variables $\{p_1, \ldots, p_m, q_1, \ldots, q_n\}$.

Finally we notice that the check in Step 5(iii)(A) and Step 5(iii)(B) is only based on the locking and the formulas in the set $\Delta$ at hand.

Let $\Delta$ be a set of formulas. In the next theorem we show that (1) if $\Delta$ does not contain locked formulas and $Bb(\Delta, \_)$ returns a structure $K$, then $K$ is a Kripke model such $K \triangleright \Delta$ and Bb does not require any further information;
(2) if $\Delta$ contains some locked formula, $M$ is a Kripke model that realizes $\Delta_T$ and $\text{Bb}(\Delta, M)$ returns a structure $K$, then $K \models \Delta$ and $K$ is possibly built on the Kripke model $M$.

**Theorem 3 (Completeness).** Let $\Delta$ be a set of formulas. 1. If $\Delta_T$ does not contain locked formulas and $\text{Bb}(\Delta, M)$ returns a structure $K$, then $K \models \Delta$ and $K$ is defined independently of $M$; 2. if $\Delta_T$ contains a subset $\Gamma$ of locked formulas and $M = \langle P_M, \leq_M, \rho_M, \vdash_M \rangle$ is a Kripke model such that $M \models \Gamma$, for every $T((A \rightarrow B) \rightarrow C) \in \Gamma$, there exists $\alpha \in P_M$ such that $\rho_M \neq \alpha$ and $\alpha \models T A, FB$ and $\rho_M \not\models p$ if $Tp \in \Gamma$, then $\text{Bb}(\Delta, M)$ returns a structure $K$ such that $K \models \Delta$.

5 Conclusions and Future Works

We have presented a new optimization rule called $\pm$-indep that generalizes the permanence rules given in [5]. By using a general result on the correctness for $\pm$-indep, we have introduced a criterion to bound the non-determinism in Int proof search. Our results do not depend on a particular tableau calculus or proof system and do not modify the logical apparatus a decision procedure is based on. As an example, our results can also be used by decision procedures based on proof systems obeying the subformula property.

We are working on more advanced decision procedures based on the results presented here, where the backtracking is bounded in more cases, also using information external to the set at hand and returned by the recursive calls. Moreover, apart the extension to other propositional intermediate logics, we are interested to investigate the application of these ideas to logics that require backtracking to be decided, such as some modal and temporal logics.

References