A Sequent Calculus Proof Search Procedure and Counter-model Generation based on Natural Deduction Bounds

Jefferson de Barros Santos∗ 1,3, Bruno Lopes Vieira† 2 and Edward Hermann Haeusler‡ 3

1 Escola Brasileira de Administração Pública e de Empresas , Fundação Getúlio Vargas , Rio de Janeiro - RJ, Brazil
2 Instituto de Computação , Universidade Federal Fluminense , Niterói - RJ, Brazil
3 Departamento de Informática , Pontifícia Universidade Católica do Rio de Janeiro , Rio de Janeiro - RJ, Brazil

December 22, 2021

Abstract

In a previously published ENTCS paper (Santos et al. (2016)), we introduced a sequent calculus called \( \text{LMT} \rightarrow \) for Minimal Implicational Propositional Logic (\( \text{M} \rightarrow \)). This calculus provides a proof search procedure for \( \text{M} \rightarrow \) that works in a bottom-up approach. We proved there that \( \text{LMT} \rightarrow \) is sound and complete. We also suggested a strategy to guarantee termination of the proof search procedure. In this current paper, we refined this strategy and presented a new strategy for \( \text{LMT} \rightarrow \) termination. Considering this new strategy, we also provide a (new) completeness proof for the system, which improves the previous version. Besides that, we present explicit upper bounds on the proof search procedure, derived from this new strategy. We also provide a full soundness proof of the system.

Keywords: Logic; Propositional Minimal Implicational Logic; Sequent Calculus; Proof Search; Counter-model Generation

∗jefferson.santos@fgv.br; Corresponding author
†bruno@ic.uff.br
‡hermann@inf.puc-rio.br
1 Introduction

Propositional Minimal Implicational Logic ($M \rightarrow$) is the fragment of the Propositional Minimal Logic ($\text{Min}$) containing only the logical connective $\rightarrow$.

The $\text{TAUT}$ problem for $M \rightarrow$ is the general problem of deciding if a formula $\alpha \in M \rightarrow$ is always true. $\text{TAUT}$ is a PSPACE-Complete problem as stated by Statman (1974), who also shows that this logic polynomially simulates Propositional Intuitionistic Logic. Statman’s simulation can also be used to simulate Propositional Classical Logic polynomially.

Furthermore, Haeusler (2015) shows that $M \rightarrow$ can polynomially simulate not only Propositional Classical and Intuitionistic Logic but also the full Propositional Minimal Logic and any other decidable propositional logic with a Natural Deduction system where the Subformula Principle holds (see Prawitz (2006)).

Moreover, $M \rightarrow$ has a strong relation with open questions about the Computational Complexity Hierarchy, as we can see from the statements below.

- If $\text{CoNP} \neq \text{NP}$ then $\text{NP} \neq P$.
- If $\text{PSPACE} = \text{NP}$ then $\text{CoNP} = \text{NP}$.
- $\text{CoNP} = \text{NP}$, iff $\exists DS$, a deductive system, such that $\forall \alpha \in TAUT_{\text{Cla}}$, there is $\Pi$, a proof of $\alpha$ in $DS$, $\text{size}(\Pi_{DS}) \leq Poly(|\alpha|)$ and the fact that $\Pi$ is a proof of $\alpha$ is verifiable in polynomial time on the size of $\alpha$.
- $\text{PSPACE} = \text{NP}$ iff $\exists DS$, a deductive system, such that $\forall \alpha \in TAUT_{M \rightarrow}$, there is $\Pi$, a proof of $\alpha$ in $DS$, $\text{size}(\Pi_{DS}) \leq Poly(|\alpha|)$ and the fact that $\Pi$ is a proof of $\alpha$ is verifiable in polynomial time on the size of $\alpha$.

Those characteristics show us that $M \rightarrow$ is as hard to implement as the most popular propositional logics. This fact, together with its straightforward language (only one logical connective), makes $M \rightarrow$ a research object that can provide us with many insights about the complexity class relationships mentioned above and about the complexity of many other logics. Moreover, the problem of conducting a proof search in a deductive system for $M \rightarrow$ has the complexity of $\text{TAUT}$ as a lower bound. Thus, the size of propositional proofs may be huge, and automated theorem provers (ATP) should take care of super-polynomially sized proofs. Therefore, the study of deductive systems for $M \rightarrow$ can directly influence in techniques to improve the way provers manage such proofs.

In Santos et al. (2016), we presented a sound and complete sequent calculus for $M \rightarrow$. We named it $\text{LMT} \rightarrow$. This calculus establishes a bottom-up approach for proof search in $M \rightarrow$ using a unified procedure either for provability and counter-model generation. $\text{LMT} \rightarrow$ was designed to avoid the usage of loop checkers and
mechanisms for backtracking in its implementation. Counter-model generation (using Kripke semantics) is achieved as a consequence of the way the tree (produced by a failed proof search) is constructed during a proof search process.

In this current work, we present an upper bound to the proof search procedure of LMT → via translation functions from very known deductive systems for M→, Prawitz’s Natural Deduction (Prawitz (2006)) and Gentzen’s sequent calculus (Gentzen (1935)). These translation functions together with a strategy to apply the rules of LMT → provide termination for the proof search procedure. Besides that, we show here all counter-model generation cases, including those missed in Santos et al. (2016).

We start in the next section with a brief discussion on the syntax and semantics of M→ used through this text. Section 3 discusses some related work and state of the art in the field of deductive systems for M→. In Section 4 we present a study about the size of proofs in M→ to establish a bound for proof search that can be used as a limit in the termination procedure of LMT →. Section 5 presents LMT → itself and its main features: termination, soundness, and completeness (with the counter-model generation as a corollary). Section 6 concludes the paper discussing some open problems and future work.

2 Minimal Implicational Logic

We can formally define the language L for M→ as follows.

Definition 1 The alphabet of L consists of:

• An enumerable set of propositional symbols, called atoms.

• The binary connective (or logical operator) for implication (→).

• parentheses: “(” and “)”.

Definition 2 We can define the general notion of a formula in L inductively:

• Every propositional symbol is a formula in L. We call them atomic formulas.

• If A and B are formulas in L then (A → B) are also.

As usual, parentheses are used for disambiguation. We use the following conventions through the text:

• Upper case letters to represent atomic formulas: A, B, C, ...

• Lower Greek letters to represent generic formulas: α, β, ...
• Upper case Greek letters are used to represent sets of formulas. For example, \( \Delta, \Gamma, \ldots \).

• If parentheses are omitted, implications are interpreted right nested.

The semantics of \( \mathbf{M} \to \) is the intuitionistic semantics restricted to \( \to \) only. Thus, given a propositional language \( \mathcal{L} \), a \( \mathbf{M} \to \) model is a structure \( \langle U, \preceq, \mathcal{V} \rangle \), where \( U \) is a non-empty set (worlds), \( \preceq \) is a partial order relation on \( U \) and \( \mathcal{V} \) is a function from \( U \) into the power set of \( \mathcal{L} \), such that if \( i, j \in U \) and \( i \preceq j \) then \( \mathcal{V}(i) \subseteq \mathcal{V}(j) \). Given a model, the satisfaction relationship \( \models \) between worlds in models and formulas is defined as in Intuitionistic Logic, namely:

1. \( \langle U, \preceq, \mathcal{V} \rangle \models p, p \in \mathcal{L}, \text{iff, } p \in \mathcal{V}(i) \)
2. \( \langle U, \preceq, \mathcal{V} \rangle \models \alpha_1 \to \alpha_2, \text{iff, for every } j \in U, \text{such that } i \preceq j, \text{if } \langle U, \preceq, \mathcal{V} \rangle \models_j \alpha_1 \text{ then } \langle U, \preceq, \mathcal{V} \rangle \models_j \alpha_2 \).

As usual a formula \( \alpha \) is valid in a model \( \mathcal{M} \), namely \( \mathcal{M} \models \alpha \), if and only if, it is satisfiable in every world \( i \) of the model, namely \( \forall i \in U, \mathcal{M} \models_i \alpha \). A formula is a \( \mathbf{M} \to \) tautology, if and only if, it is valid in every model.

3 Related Work

It is known that Prawitz’s Natural Deduction System for Propositional Minimal Logic with only the \( \to \)-rules (\( \to \)-Elim and \( \to \)-Intro) is sound and complete for the \( \mathbf{M} \to \) regarding Kripke semantics. As a consequence of this, Gentzen’s \( \mathbf{LJ} \) system (Gentzen (1935)) containing only right and left \( \to \)-rules is also sound and complete.

In Gentzen (1935), Gentzen proved the decidability of the Propositional Intuitionistic Logic (\( \textbf{Int} \)), which also includes the cases for \( \textbf{Min} \) and \( \mathbf{M} \to \).

However, Gentzen’s approach was not conceived to be a bottom-up proof search procedure. Figure 1 shows structural and logic rules of an adapted Gentzen’s sequent calculus for \( \mathbf{M} \to \), called \( \mathbf{LJ} \to \). We restrict the right side of a sequent to one and only one formula (we are in \( \mathbf{M} \to \); thus, sequents with an empty right side does not make sense). This restriction implies that structural rules can only be considered for main formulas on the left side of a sequent. \( \mathbf{LJ} \to \) incorporates contraction in the \( \to \)-left, with the repetition of the main formula of the conclusion on the premises. We use those adaptations to explain the difficulties in using sequent calculus systems for proof search in \( \textbf{Int}, \textbf{Min}, \) and \( \mathbf{M} \to \). In Dyckhoff (2016), Dyckhoff describes in detail the evolution of those adaptations over Gentzen’s original \( \mathbf{LJ} \) system. Those adaptations are attempting to improve bottom-up proof search mechanisms for \( \textbf{Int} \).
A central aspect when considering mechanisms for proof search in $M^\rightarrow$ (and also for $\text{Int}$) is the application of the $\rightarrow$-left rule. The $LK$ system proposed by Gentzen (1935), the sequent calculus for Classical Logic, with some adaptations (e.g., Seldin (1998)) can ensure that each rule reduces the degree (the number of atomic symbols occurrences and connectives in a formula) of the main formula of the sequent (the formula to which the rule is applied), when applied in a bottom-up manner during the proof search. This fact implies the termination of the system. However, the case for $\text{Int}$ is more complicated. First, we have the “context-splitting” (using an expression from Dyckhoff (2016)) nature of $\rightarrow$-left, i.e., the formula on the right side of the conclusion sequent is lost in the left premise of the rule application. Second, as we can reuse a hypothesis in different parts of a proof, the main formula of the conclusion must be available to be used again by the generated premises. Thus, the $\rightarrow$-left rule has the repetition of the main formula in the premises, a scenario that allows the occurrence of loops in automatic procedures.

Since Gentzen, many others have explored solutions to deal with the challenges mentioned above, proposing new calculi (sets of rules), strategies and proof search procedures to allow more automated treatment to the problem.

Unfortunately, the majority of these results are focused on $\text{Int}$, with very few works dedicated explicitly to $\text{Min}$ or $M^\rightarrow$. Thus, we needed to concentrate our literature review in the $\text{Int}$ case, adjusting the found results to the $M^\rightarrow$ context by ourselves.

A crucial source of information was the work of Dyckhoff (2016), appropriately entitled “Intuitionistic decision procedures since Gentzen” that summarizes in chronological order the main results of this field. In the next paragraphs, we highlight the most important of those results presented in Dyckhoff (2016).
A common way to control the proof search procedure in $M \rightarrow$ (and in Int) is by the definition of routines for loop verification as proposed in Underwood (1990). Dyckhoff present loop checkers as very expensive procedures for automatic reasoning (Dyckhoff (2016)), although they are effective to guarantee termination of proof search procedures. The work in Heuerding et al. (1996) and Howe (1997) are examples of techniques that can be used to minimize the performance problems that can arise with the usage of such procedures.

To avoid the use of loop checkers, Dyckhoff (1992) proposed a terminating contraction-free sequent calculus for Int, named LJT, using a technique based on the work of Vorob’ev (1970) in the 50s. Pinto and Dyckhoff (1995) extended this work showing a method to generate counter-examples in this system. They proposed two calculi, one for proof search and another for counter-model generation, forming a way to decide about the validity or not of formulas in Int. A characteristic of their systems is that the subformula property does not hold on them. In Ferrari et al. (2013), a similar approach is presented using systems where the subformula property holds. They also proposed a single decision procedure for Int, which guarantees minimal depth counter-model.

Focused sequent calculi appeared initially in the Andreoli’s work on linear logic Andreoli (1992). The author identified a subset of proofs from Gentzen-style sequent calculus, which is complete and tractable. Liang and Miller (2007) proposed the focused sequent calculi LJF where they used a mapping of Int into linear logic and adapted the Andreoli’s system to work with the image. Dyckhoff and Lengrand (2006) presented the focused system LJQ that work direct in Int. Focusing is used in their system as a way to implement restrictions in the $\rightarrow$-left rule as proposed by Vorob’ev (1970) and Hudelmaier (1993). The work of Dyckhoff and Lengrand (2006) follows from the calculus with the same name presented in Herbelin (1995).

Dyckhoff (2016) also identify a list of features particularly of interest when evaluating mechanisms for proof search in Int that we will follow when comparing our solution to the other existent ones. They are: termination (proof search procedure stops both for theorems and non-theorem formulas), bicompleteness (extractability of models from failed proof searches), avoidance of backtracking (backtracking being a very immediate approach to deal with the context split in $\rightarrow$-left, but it is also a complex procedure to implement), simplicity (allows easier reasoning about systems).

4 The Size of Proofs in $M \rightarrow$

Hirokawa has presented an upper bound for the size of normal form Natural Deduction proofs of implicational formulas in Int (that correspond to $M \rightarrow$ formulas).
Hirokawa (1991) showed that for a formula $\alpha \in \mathbf{M} \rightarrow$, this limit is $|\alpha| \cdot 2^{|\alpha|+1}$. As the Hirokawa result concerns normal proofs in Natural Deduction we present now a translation of this system to a cut-free sequent calculus, following the $\mathbf{LJ} \rightarrow$ rules presented in Section 3, thus we can establish the limit for proof search in $\mathbf{LJ} \rightarrow$ too.

Figure 2 presents a recursively defined function\(^1\) to translate Natural Deduction normal proofs of $\mathbf{M} \rightarrow$ formulas into $\mathbf{LJ} \rightarrow$ proofs (in a version of the system without the cut rule).

In this definition, $c$ is a function that returns the conclusion (last sequent) of a $\mathbf{LJ} \rightarrow$ demonstration as showed in (1). Also, $\rightarrow -lc$ is a function that receives two $\mathbf{LJ} \rightarrow$ sequents and a formula to construct the conclusion of a $\rightarrow -left$ rule application, as defined in (2).

\[
c \left( \frac{\Pi}{\Gamma \Rightarrow \gamma} \right) = \Gamma \Rightarrow \gamma \tag{1}
\]

\[
\rightarrow -lc(\Gamma \Rightarrow \alpha; \beta, \Gamma \Rightarrow \gamma; \alpha \rightarrow \beta) = \Gamma, \alpha \rightarrow \beta \Rightarrow \gamma \tag{2}
\]

\(^1\)We use a semicolon to separate arguments of functions (in function definitions and function calls) instead of the most common approach to using commas. This change in convention aims to avoid confusion with the commas used to separate formulas and sets of formulas in sequent notation.
**Proposition 1** Let $\Gamma \prod \alpha$, a normal Natural Deduction derivation and $\Gamma$, the set of undischarged formulas in $\Gamma$ then,

$$F\left( \prod_{\alpha} \Gamma ; \emptyset \right) = \prod'_{\Gamma \Rightarrow \alpha}$$

is a proof in $\text{LJ}^\rightarrow$.

**Proof 1** By induction in the size of $\prod$.

As an example of the translation produced by the function of Figure 2, we show below each step of the translation of a Natural Deduction proof (3) into an $\text{LJ}^\rightarrow$ proof (4). To shorten the size of the proofs we collapsed repeated occurrences of formulas when passed as the hypothesis argument of the recursive function call.

\[
\begin{align*}
[B]^2 & \quad [A]^1 \quad [A \rightarrow (B \rightarrow C)]^3 \\
& \quad B \rightarrow C \quad \rightarrow E \\
& \quad A \rightarrow C \quad \rightarrow I^1 \\
& \quad B \rightarrow (A \rightarrow C) \quad \rightarrow I^2 \\
& \quad (A \rightarrow (B \rightarrow C)) \rightarrow (B \rightarrow (A \rightarrow C)) \quad \rightarrow I^3
\end{align*}
\]

\[
\begin{align*}
F \left( \prod_1 \left( \frac{[A \rightarrow (B \rightarrow C)]^3}{B \rightarrow (A \rightarrow C)} \right) ; \emptyset \right) \\
& \quad \Rightarrow (A \rightarrow (B \rightarrow C)) \rightarrow (B \rightarrow (A \rightarrow C)) \quad \rightarrow I^3
\end{align*}
\]

\[
\begin{align*}
F \left( \prod_2 \left( \frac{[B]^2, A \rightarrow (B \rightarrow C)}{A \rightarrow C} \right) ; \{A \rightarrow (B \rightarrow C)\} \right) \\
& \quad \Rightarrow (A \rightarrow (B \rightarrow C)) \rightarrow (B \rightarrow (A \rightarrow C)) \quad \rightarrow r
\end{align*}
\]
\[
F \left( \frac{[A]^1, B, A \rightarrow (B \rightarrow C)}{\Pi_3 \frac{C}{A \rightarrow C} \rightarrow 1^1} \right) \rightarrow r \\
A \rightarrow (B \rightarrow C) \Rightarrow B \rightarrow (A \rightarrow C) \rightarrow r \\
\Rightarrow (A \rightarrow (B \rightarrow C)) \rightarrow (B \rightarrow (A \rightarrow C)) \rightarrow r
\]

\[\nabla\]

\[
F \left( \begin{array}{c}
B \\
A \\
C
\end{array} \frac{A \rightarrow (B \rightarrow C)}{B \rightarrow C} \rightarrow E; \{A, B, A \rightarrow (B \rightarrow C)\} \right) \rightarrow r \\
B, A \rightarrow (B \rightarrow C) \Rightarrow A \rightarrow C \rightarrow r \\
A \rightarrow (B \rightarrow C) \Rightarrow B \rightarrow (A \rightarrow C) \rightarrow r \\
\Rightarrow (A \rightarrow (B \rightarrow C)) \rightarrow (B \rightarrow (A \rightarrow C)) \rightarrow r
\]

In the following steps consider that \(\Gamma = \{A, B, A \rightarrow (B \rightarrow C)\}\).

\[\nabla\]

\[
F(A; \Gamma) \rightarrow \neg lc \left( c(F(A; \Gamma)) : c \left( F \left( \begin{array}{c}
B \\
C
\end{array} \frac{B \rightarrow C}{B \rightarrow C} \rightarrow E; \Gamma \right) \right) ; A \rightarrow (B \rightarrow C) \right) \rightarrow l
\]

\[
\frac{B, A \rightarrow (B \rightarrow C) \Rightarrow A \rightarrow C}{A \rightarrow (B \rightarrow C) \Rightarrow B \rightarrow (A \rightarrow C)} \rightarrow r \\
\Rightarrow (A \rightarrow (B \rightarrow C)) \rightarrow (B \rightarrow (A \rightarrow C)) \rightarrow r
\]

\[\nabla\]

\[
\Gamma \Rightarrow A \\
\frac{F(B; \{\Gamma, B \rightarrow C\})}{\neg lc(c(F(B; \{\Gamma, B \rightarrow C\}) ; c(F(B; \{\Gamma, B \rightarrow C\}) ; B \rightarrow C)) \rightarrow l}
\]

\[
\frac{F(C; \{\Gamma, B \rightarrow C\})}{\neg lc(c(F(C; \{\Gamma, B \rightarrow C\}) ; c(F(C; \{\Gamma, B \rightarrow C\}) ; B \rightarrow C)) \rightarrow l}
\]

\[
\frac{\Gamma \Rightarrow C}{B, A \rightarrow (B \rightarrow C) \Rightarrow A \rightarrow C \rightarrow r \\
A \rightarrow (B \rightarrow C) \Rightarrow B \rightarrow (A \rightarrow C) \rightarrow r \\
\Rightarrow (A \rightarrow (B \rightarrow C)) \rightarrow (B \rightarrow (A \rightarrow C)) \rightarrow r
\]

\[\nabla\]
The size of proofs in $\text{LJ}^\rightarrow$ considering only implicational tautologies is the same of that in Natural Deduction, i.e. for an implicational formula $\alpha$, a proof in $\text{LJ}^\rightarrow$ has maximum height of $|\alpha| \cdot 2^{|\alpha|+1}$.

Proof 2 This proof follows directly from the translation function as each step in the Natural Deduction proof is translated into precisely one step in the $\text{LJ}^\rightarrow$ resultant proof.

5 The Sequent Calculus $\text{LMT}^\rightarrow$

In this section, we present a sound and complete sequent calculus for $\text{M}^\rightarrow$. We call this system $\text{LMT}^\rightarrow$. We can prove for each rule that if all premises are valid, then the conclusion is also valid, and if at least one premise is invalid, then the conclusion also is. This proof is constructive, i.e., for any sequent, we have an effective way to produce either a proof or a counter-model of it.

We start defining the concept of sequent used in the proposed calculus. A sequent in our system has the following general form:

$$\{\Delta'\}, \Upsilon_1^{p_1}, \Upsilon_2^{p_2}, ..., \Upsilon_n^{p_n}, \Delta \Rightarrow [p_1, p_2, ..., p_n], \varphi$$

where $\varphi$ is a formula in $\mathcal{L}$ and $\Delta$, $\Upsilon_1^{p_1}, \Upsilon_2^{p_2}, ..., \Upsilon_n^{p_n}$ are bags\(^2\) of formulas. Each $\Upsilon_i^{p_i}$ represents formulas associated with an atomic formula $p_i$.

A sequent has two focus areas, one in the left side (curly bracket)\(^3\) and another on the right (square bracket). Curly brackets are used to control the application of the $\rightarrow$-left rule and square brackets are used to keep control of formulas that are related to a particular counter-model definition. $\Delta'$ is a set of formulas and

\(^2\)A bag (or a multiset) is a generalization of the concept of a set that, unlike a set, takes repetitions into account: a bag $\{A, A, B\}$ is not the same as the bag $\{A, B\}$.

\(^3\)Note that the symbols $\{}$ and $\{\}$ here do not represent a set. Instead, these symbols work as an annotation in the sequent to determine the left side focused area. Therefore, $\Delta'$ instead is a set of formulas in the focused area.
\( p_1, p_2, \ldots, p_n \) is a sequence that does not allow repetition. We call context of the sequent a pair \((\alpha, q)\), where \( \alpha \in \Delta' \) and \( \varphi = q \), where \( q \) is an atomic formula on the right side of the sequent.

The axioms and rules of \( \text{LMT} \rightarrow \) are presented in Figure 3. In each rule, \( \Delta' \subseteq \Delta \).

Rules are inspired by their backward application. In a \( \rightarrow \)-left rule application, the atomic formula, \( q \), on the right side of the conclusion goes to the \( || \)-area in the left premise. \( \Delta \) formulas in the conclusion are copied to the left premise and marked with a label relating each of them with \( q \). The left premise also has a copy of \( \Delta \) formulas without the \( q \)-label. This mechanism keeps track of proving attempts. The form of the restart rule is better understood in the completeness proof on Section 5.5. A forward reading of rules can be achieved by considering the notion of validity, as described in Section 5.4.

### 5.1 A Proof Search Strategy

The following is a general strategy to be applied with the rules of \( \text{LMT} \rightarrow \) to generate proofs from an input sequent (a sequent that is a candidate to be the conclusion of a proof), which is based on a bottom-up application of the rules. From the proposed strategy, we can then state a proposition about the termination of the proving process.

A goal sequent is a new sequent in the form of (5). It is a premise of one of the system’s rules, generated by the application of this rule on an open branch during the proving process. If the goal sequent is an axiom, the branch where it is will stop. Otherwise, apply the first applicable rule in the following order:

1. Apply \( \rightarrow \)-right rule if it is possible, i.e., if the formula on the right side of the sequent, outside the \( || \)-area, is not atomic. The premise generated by this application is the new goal of this branch.

2. Choose one formula on the left side of the sequent, not labeled yet, i.e., a formula \( \alpha \in \Delta \) that is not occurring in \( \Delta' \), then apply the focus rule. The premise generated by this application is the new goal of this branch.

3. If all formulas on the left side have already been focused, choose the first formula \( \alpha \in \Delta' \) such that the context \((\alpha, q)\) was not yet tried since the last application of a restart rule. We say that a context \((\alpha, q)\) is already tried when a formula \( \alpha \) on the left was expanded (by the application of \( \rightarrow \)-left rule), with \( q \) as the formula outside the \( || \)-area on the right side of the sequent. The premises generated by this application are new goals of the respective new branches.
4. Choose the leftmost formula inside the $[]$-area that was not chosen before in this branch and apply the restart rule. The premise generated by this application is the new goal of the branch.

**Observation 1** From the proof strategy we can make the following observations about a tree generated during a proving process:

(i) A top sequent is the highest sequent of a branch in the tree.

(ii) In a top sequent of a branch on the form of sequent (5), if $\varphi \in \Delta$ then the top sequent is an axiom and the branch is called a closed branch. Otherwise, we say that the branch is open and $\varphi$ is an atomic formula.

(iii) In every sequent of the tree, $\Delta' \subseteq \Delta$.

(iv) For $i = 1, \ldots, n$, $\Upsilon_{i-1} \subseteq \Upsilon_i$.

We call this strategy $S$-$\text{LMT} \rightarrow$. Figure 4 shows a generic schema of a completely expanded branch in a tree, not necessarily a proof, generated by the application of $S$-$\text{LMT} \rightarrow$.

### 5.2 An Upper Bound for the Proof Search in $\text{LMT} \rightarrow$

Using the same approach applied in Section 4, we now propose a translation from $\text{LJ} \rightarrow$ proofs into the system $\text{LMT} \rightarrow$. The translation function needs to adapt a sequent in $\text{LJ} \rightarrow$ form to a sequent in $\text{LMT} \rightarrow$ form. Figure 5 presents the definition of the translation function$^4$.

We use some abbreviations to shorten the function definition of Figure 5. We present them below.

\[ D_1' = F'(D_1; \Delta \cup \{\alpha \rightarrow \beta\}; \Upsilon \cup \Gamma q \cup \{(\alpha \rightarrow \beta)^q\}; \Sigma \cup \{q\}; \Pi') \]

\[ D_2' = F'(D_2; \Gamma, \alpha \rightarrow \beta \Rightarrow \alpha; \Delta \cup \{\alpha \rightarrow \beta\}; \Upsilon \cup \Gamma q \cup \{(\alpha \rightarrow \beta)^q\}; \Sigma \cup \{q\}; \Pi') \]

\[ \Pi' = PROOFUNTIL(FOCUS(\{\Delta\}, \Upsilon, \Gamma, \alpha \rightarrow \beta \Rightarrow [\Sigma], q)) \]

\[ \Gamma q = \] means that all formulas of the set $\Gamma$ are labeled with a reference to the atomic formula $q$

$^4$As in Section 4, we use semicolon to separate function arguments here
\((\alpha \rightarrow \beta)^q\) means the same for the individual formula \(\alpha \rightarrow \beta\).

The complicated case occurs when the function \(F'\) is applied to a proof fragment in which the last \(\text{LJ} \rightarrow\) rule applied is an \(\rightarrow\)-left. In this case, \(F'\) needs to inspect the proof fragment constructed until that point to identify whether the context \((\alpha \rightarrow \beta, q)\) was already used or not. This inspection has to be done since \(\text{LMT} \rightarrow\) does not allow two or more applications of the same context between two applications of the restart rule. To deal with this, we use some auxiliary functions described below.

\(\text{FOCUS}\) is a function that receives a fragment of proof in \(\text{LMT} \rightarrow\) form and builds one application of the focus rule on the top of the proof fragment received in the case that the main formula of the rule is not already focused. The main formula is also an argument of the function. In the function definition (6), we have the constraint that \(\alpha \in \Gamma\).
The function \( \text{PROOFUNTIL} \) also receives a fragment of a proof in \( \text{LMT}^\rightarrow \) form where \((\alpha \rightarrow \beta, q)\) is one of the available contexts, applies the restart rule with an atomic formula \( p \) such that \( p \in \Sigma \) in the top of this fragment of proof and, then, conducts a sequence of \( \text{LMT}^\rightarrow \) rule applications following the \( \mathcal{S}\text{-LMT}^\rightarrow \) until the point that the context \((\alpha \rightarrow \beta, q)\) is available again. This mechanism has to be done in the case that the context \((\alpha \rightarrow \beta, q)\) is already applied in an \( \rightarrow \)-left application, some point after the last restart rule application in the proof fragment received as the argument \( \Pi \). Otherwise, the proof fragment is returned unaltered. Function \( \text{PROOFUNTIL} \) is described in the function definition (7).
As an example of this translation, we use here the formula 
\(((\((A \rightarrow B) \rightarrow A\)) \rightarrow A) \rightarrow B\). We know from Dowek and Jiang (2006) that this formula needs two repetitions of the hypothesis 
\(((\((A \rightarrow B) \rightarrow A\)) \rightarrow A) \rightarrow B\) to be proved in \(M\). To shorten the proof tree we use the following abbreviation: 
\(((A \rightarrow B) \rightarrow A) \rightarrow A = \epsilon\). Thus, its normal proof in Natural Deduction can be represented as shown in Proof (8).

Using the translation presented in Figure 2, we achieve the following \(LJ\) cut-free proof. To shorten the proof, we numbered each subformula of the initial formula that we want to prove and use these numbers to refer to those subformulas all over the proof. Let us call this \(LJ\) version of the proof \(D\) (Proof (9)).
The translation to \( \text{LMT} \rightarrow \) starts by applying the function \( F' \) to the full proof \( D \).

\[
F'
\begin{pmatrix}
\Gamma &= \emptyset; \\
\Delta &= \emptyset; \\
\Sigma &= \emptyset; \\
\Pi &= \text{nil}
\end{pmatrix}
D, \ 
\begin{pmatrix}
\Gamma &= \emptyset; \\
\Delta &= \emptyset; \\
\Sigma &= \emptyset; \\
\Pi &= \text{nil}
\end{pmatrix}
\]

This first call of \( F' \) produces the end sequent of the proof in \( \text{LMT} \rightarrow \) and calls the function \( F' \) recursively to the rest of the original proof in \( \text{LJ} \rightarrow \). This \( \text{LJ} \rightarrow \) fragment has now, as its last rule application, an \( \rightarrow \)-left.

\[
\Pi = \{\} \Rightarrow [], (((A \rightarrow_0 B) \rightarrow_1 A) \rightarrow_2 A) \rightarrow_3 B) \rightarrow_4 B
\]

Then, as the main formula (((A \rightarrow_0 B) \rightarrow_1 A) \rightarrow_2 A) \rightarrow_3 B (in the proof represented only by \( \rightarrow_3 \)) is not focused yet, the call of function \( F' \) first constructs an application of the focus rule on the top of the \( \Pi \) fragment received as a function argument. Also, the context \( (\rightarrow_3, B) \) was not expanded yet. Thus, the recursive step can proceed directly without the need of a restart (this is controlled by the \( \text{PROOFUNTIL} \) function as shown in \( F' \) definition in Figure 5).

16
The call of $F'$ on the right premise constructs an axiom. Thus this branch in the $\text{LMT} \rightarrow$ proof translation being built is closed. The next recursive call "pastes" on the top of the right branch of the new version of $\Pi$ as follows.

\[
\begin{array}{c}
\frac{\{ \rightarrow_3, \rightarrow_3 \Rightarrow [B], \rightarrow_2 \{ \rightarrow_3 \}, \rightarrow_3, B \Rightarrow [] , B \}}{\Pi} \quad \rightarrow 1
\end{array}
\]

This process goes until a point where the context $(\rightarrow_3, B)$ is again found, and the translation needs to deal with a restart in the $\text{LMT} \rightarrow$ translated proof. This situation happens when the recursion of $F'$ reaches the point below.

\[
\begin{array}{c}
\frac{\{ \rightarrow_3 \}, \rightarrow_3 \Rightarrow [B], \rightarrow_2 \{ \rightarrow_3 \}, \rightarrow_3, B \Rightarrow [] , B \}}{\Pi} \quad \rightarrow 1
\end{array}
\]
The \( \Pi \) fragment constructed in this step of the recursion is presented below.

\[
\begin{array}{c}
\{ \to_3, \to_1 \} \Rightarrow [B, A], \to_0 \quad \{ \to_3, \to_1 \} \Rightarrow [B, A] \quad \to \quad \{ \to_3, \to_1 \} \Rightarrow [B, A] \\
\{ \to_3, \to_1 \} \Rightarrow \{ \to_3, \to_1 \} \Rightarrow [B, A] \quad \to \quad \text{focus} \\
\{ \to_3, \to_1 \} \Rightarrow \{ \to_3, \to_1 \} \Rightarrow [B, A] \quad \to \quad \text{r} \\
\{ \to_3, \to_1 \} \Rightarrow [B, A] \quad \to \quad \text{r} \\
\{ \to_3, \to_1 \} \Rightarrow [B, A] \quad \to \quad \text{r} \\
\{ \to_3, \to_1 \} \Rightarrow [B, A] \quad \to \quad \text{r} \\
\{ \to_3, \to_1 \} \Rightarrow [B, A] \quad \to \quad \text{r} \\
\end{array}
\]

The \( \Pi' \) fragment in \( \Pi \) is built as the result of a proof search from the top sequent of the leftmost branch of \( \Pi \) until the point that there is a repetition of the context \( \to_3, B \) and the \( \to \)-left rule can be applied again without offending the \textbf{LMT}\( \to \) strategy. This result is produced by the \textit{PROOFUNTIL} call when applying \textbf{F}' to an \textbf{LJ}\( \to \) fragment that end with a \( \to \)-left rule application.

\[
\begin{array}{c}
\Pi'_1 \quad \Pi'_2 \\
\{ \to_3, \to_1 \} \Rightarrow [B, A], A \quad \to \quad \text{focus} \\
\{ \to_3, \to_1 \} \Rightarrow [B, A], A \quad \to \quad \text{restart} \\
\end{array}
\]

where \( \Pi'_1 \) is:

\[
\begin{array}{c}
\{ \to_3, \to_1 \} \Rightarrow [B, A], B \\
\{ \to_3, \to_1 \} \Rightarrow [B, A], B \\
\{ \to_3, \to_1 \} \Rightarrow [B, A], B \\
\{ \to_3, \to_1 \} \Rightarrow [B, A], B \\
\{ \to_3, \to_1 \} \Rightarrow [B, A], B \\
\{ \to_3, \to_1 \} \Rightarrow [B, A], B \\
\{ \to_3, \to_1 \} \Rightarrow [B, A], B \\
\end{array}
\]

and \( \Pi'_2 \) is:

\[
\begin{array}{c}
\{ \to_3, \to_1 \} \Rightarrow [B, A], B \\
\{ \to_3, \to_1 \} \Rightarrow [B, A], B \\
\{ \to_3, \to_1 \} \Rightarrow [B, A], B \\
\{ \to_3, \to_1 \} \Rightarrow [B, A], B \\
\{ \to_3, \to_1 \} \Rightarrow [B, A], B \\
\{ \to_3, \to_1 \} \Rightarrow [B, A], B \\
\{ \to_3, \to_1 \} \Rightarrow [B, A], B \\
\end{array}
\]

18
The top sequent of the fragment $\Pi'_1$ is the point where we can apply the $\to$-left rule to the context $(\to_3, B)$ again. Thus the next recursion call becomes:

$$
\frac{\Pi'_3 \quad \Pi'_4}{\Pi} \to 1
$$

where

$$\Pi'_3 = F' \left( \begin{array}{c}
\to_3, \to_1, A \Rightarrow A \\
\to_3, \to_1, A \Rightarrow \to_2
\end{array} \right) ; \quad 
\begin{array}{c}
\Gamma = \{ \to_3, \to_1, A \} ; \\
\Delta = \{ \to_3, \to_1 \} ; \\
\Upsilon = \{ \to_3^A, \to_1^A, \to_3^B, \to_1^B, A^B \} ; \\
\Sigma = \{ B, A \} ;
\end{array}$$

and

$$\Pi'_4 = F' \left( \begin{array}{c}
\to_3, \to_1, A, B \Rightarrow B \\
\end{array} \right) ; \quad 
\begin{array}{c}
\Gamma = \{ \to_3, \to_1, A \} ; \\
\Delta = \{ \to_3, \to_1 \} ; \\
\Upsilon = \{ \to_3^A, \to_1^A, \to_3^B, \to_1^B, A^B \} ; \\
\Sigma = \{ B, A \} ;
\end{array}$$

Finally, after finish the translation process we obtained the translated proof of (10).
Axioms:

\[ F(\alpha; \Gamma) = \Gamma, \alpha \Rightarrow \alpha \]

Case of → Introduction:

\[
F \left( \frac{\prod_{\beta} \alpha \rightarrow \beta}{\alpha \rightarrow \beta} : \Gamma \right) = \frac{\prod_{\beta} \alpha \rightarrow \beta : \{\alpha\} \cup \Gamma}{\Gamma \Rightarrow \alpha \rightarrow \beta} \rightarrow r
\]

Case of → Elimination:

\[
F \left( \frac{\prod_{\beta} \alpha \rightarrow \beta}{\beta \rightarrow \Pi_2 : \Gamma} \right) = \frac{F(\prod_{\alpha} \alpha \rightarrow \beta : \{\alpha \rightarrow \beta\} \cup \Gamma)}{F(\prod_{\beta} \alpha \rightarrow \beta : \{\alpha \rightarrow \beta\} \cup \Gamma)} \rightarrow 1
\]

\[ \rightarrow -lc \left( c \left( F \left( \frac{\prod_{\alpha} \alpha \rightarrow \beta : \{\alpha \rightarrow \beta\} \cup \Gamma} \right) \right), c \left( F \left( \frac{\prod_{\beta} \alpha \rightarrow \beta : \{\alpha \rightarrow \beta\} \cup \Gamma} \right) \right), \alpha \rightarrow \beta \right) \]

Figure 2: A recursively defined function to translate Natural Deduction proofs into \( \text{LJ} \rightarrow \)
Axiom:
\[
\{ \Delta', q \}, \Upsilon_1^{p_1}, \Upsilon_2^{p_2}, \ldots, \Upsilon_n^{p_n}, \Delta \Rightarrow [p_1, p_2, \ldots, p_n], q
\]

Focus:
\[
\{ \Delta', \alpha \}, \Upsilon_1^{p_1}, \Upsilon_2^{p_2}, \ldots, \Upsilon_n^{p_n}, \Delta, \alpha \Rightarrow [p_1, p_2, \ldots, p_n], \beta
\]
\[
\{ \Delta \}, \Upsilon_1^{p_1}, \Upsilon_2^{p_2}, \ldots, \Upsilon_n^{p_n}, \Delta, \alpha \Rightarrow [p_1, p_2, \ldots, p_n], \beta
\]

Restart:
\[
\{ \}, \Upsilon_1, \Upsilon_2, \ldots, \Upsilon_i, \Upsilon_{i+1}^{p_{i+1}}, \ldots, \Upsilon_n^{p_n}, \Delta^q \Rightarrow [p_1, p_2, \ldots, p_{i+1}, \ldots, p_n, q], p_i
\]
\[
\{ \Delta' \}, \Upsilon_1^{p_1}, \Upsilon_2^{p_2}, \ldots, \Upsilon_i^{p_i}, \Upsilon_{i+1}^{p_{i+1}}, \ldots, \Upsilon_n^{p_n}, \Delta \Rightarrow [p_1, p_2, \ldots, p_i, p_{i+1}, \ldots, p_n], q
\]

→Right
\[
\{ \Delta' \}, \Upsilon_1^{p_1}, \Upsilon_2^{p_2}, \ldots, \Upsilon_n^{p_n}, \Delta \Rightarrow [p_1, p_2, \ldots, p_n], \beta
\]
\[
\{ \Delta' \}, \Upsilon_1^{p_1}, \Upsilon_2^{p_2}, \ldots, \Upsilon_n^{p_n}, \Delta \Rightarrow [p_1, p_2, \ldots, p_n], \alpha \rightarrow \beta
\]

→Left

Considering \( \Upsilon = \bigcup_{i=1}^{n} \Upsilon_i^{p_i} \) and \( \bar{p} = p_1, p_2, \ldots, p_n \), we have:
\[
\{ \alpha \rightarrow \beta, \Delta' \}, \Upsilon, \Delta^q, \Delta \Rightarrow [\bar{p}, q], \alpha
\]
\[
\{ \alpha \rightarrow \beta, \Delta' \}, \Upsilon, \Delta, \beta \Rightarrow [\bar{p}], q
\]
\[
\{ \alpha \rightarrow \beta, \Delta' \}, \Upsilon, \Delta \Rightarrow [\bar{p}], q
\]
\( \{ \Delta', \Gamma_1^{p_1}, \Gamma_2^{p_2}, \Gamma_{i-1}^{p_{i-1}}, \ldots, \Gamma_i^{p_i}, \Delta \Rightarrow [p_1, p_2, \ldots, p_{i-1}, p_i], p_k \} \) (where \( k = 1, 2, \ldots, i \))

\[ \text{a sequence of focus, } \rightarrow \text{-left and } \rightarrow \text{-right} \]

\[ \{ \}, \Gamma_1^{p_1}, \Gamma_2^{p_2}, \ldots, \Gamma_{i-1}^{p_{i-1}}, \Delta \Rightarrow [p_1, p_2, \ldots, p_{i-1}, p_i] \]

\[ \text{a sequence of focus, } \rightarrow \text{-left, } \rightarrow \text{-right and restart (for each atomic formula in the } \lbrack \rbrack \text{-area)} \]

\[ \{ \varphi \rightarrow \psi \}, \Gamma_2^{p_2}, \ldots, \Gamma_{i-1}^{p_{i-1}}, \Delta^{p_i}, \Gamma_i^{p_i}, \Gamma_1^{p_1}, \varphi_1, \ldots, \varphi_n \Rightarrow [p_2, \ldots, p_{i-1}, p_i, p_1], p_2 \]

\[ \text{a sequence } \rightarrow \text{-right} \]

\[ \{ \varphi \rightarrow \psi \}, \Gamma_2^{p_2}, \ldots, \Gamma_{i-1}^{p_{i-1}}, \Delta^{p_i}, \Gamma_i^{p_i}, \Gamma_1^{p_1}, \varphi \Rightarrow \{ \}, p_2, \ldots, p_{i-1}, p_i, p_1 \]

\[ \text{a sequence of focus, } \rightarrow \text{-left and } \rightarrow \text{-right} \]

\[ \{ \varphi \rightarrow \psi \}, \Gamma_1^{p_1}, \Gamma_i^{p_i}, \Gamma_1^{p_1}, \varphi \Rightarrow \{ \}, p_1 \]

\[ \{ \varphi \rightarrow \psi \}, \Gamma_1^{p_1}, \Gamma_i^{p_i}, \Gamma_1^{p_1}, \varphi \Rightarrow \{ \}, p_1 \rightarrow \text{-left} \]

\[ \{ \varphi \rightarrow \psi \}, \Gamma_1^{p_1}, \Gamma_i^{p_i}, \Gamma_1^{p_1}, \varphi \Rightarrow \{ \}, \gamma_1, \ldots, \gamma_m \Rightarrow \{ \}, p_1 \rightarrow \text{-right} \]

Figure 4: Generic schema of a tree generated following S-LMT
Axioms:
\[ F'(\Gamma, \alpha \Rightarrow \alpha; \Delta; \Upsilon; \Sigma; \Pi) = \{\Delta\}, \Upsilon, \Gamma, \alpha \Rightarrow [\Sigma], \alpha \]
\[ \Pi \]

Last rule is \(\rightarrow\)-right:
\[
F'(\frac{D}{\Gamma \Rightarrow \alpha \rightarrow \beta} : \Delta; \Upsilon; \Sigma; \Pi) \]
\[
F'(\frac{D}{\Gamma, \alpha \Rightarrow \beta : \Delta; \Upsilon; \Sigma; \{\Delta\}, \Upsilon, \Gamma \Rightarrow [\Sigma], \alpha \Rightarrow \beta \Pi} : \Delta; \Upsilon, \Gamma \Rightarrow [\Sigma], \alpha \Rightarrow \beta \Pi) \]
\[ \rightarrow r \]

Last rule is \(\rightarrow\)-left:
\[
F'(\frac{D_1}{\Gamma, \alpha \Rightarrow \beta : \Delta; \Upsilon; \Sigma; \Pi} = \frac{D_2}{\Gamma, \alpha \Rightarrow \beta \Rightarrow q} : \Delta; \Upsilon; \Sigma; \Pi) \]
\[
F'(\frac{D_1'}{\Gamma, \alpha \Rightarrow \beta \Rightarrow q} = \frac{D_2'}{\{\Delta\}, \Upsilon, \Gamma, \alpha \Rightarrow [\Sigma], q \Pi} : \Delta; \Upsilon, \Gamma, \alpha \Rightarrow [\Sigma], q \Pi) \]
\[ \rightarrow l \]

Figure 5: A recursive function to translate \(LJ\rightarrow\) into \(LMT\rightarrow\)
5.3 Termination

To control the end of the proof search procedure of $\text{LMT} \rightarrow$ our approach is to define an upper bound limit to the size of its proof search tree. Then, we need to show that the $\text{LMT} \rightarrow$ strategy here proposed allows exploring all the possible ways to expand the proof tree until it reaches this size.

From Theorem 1, we know that the upper bound for cut-free proofs based on $\text{LJ} \rightarrow$ is $|\alpha| \cdot 2^{|\alpha|+1}$, where $\alpha$ is the initial formula that we want to prove. We use the translation presented in Figure 5 on the previous Section to find a similar limit for $\text{LMT} \rightarrow$ proofs. We have to analyze three cases to establish an upper bound for $\text{LMT} \rightarrow$.

The cases are described below and are summarized in Table 1.

(i) Axioms of $\text{LJ} \rightarrow$ maps one to one with axioms of $\text{LMT} \rightarrow$;
(ii) $\rightarrow$-right applications of $\text{LJ} \rightarrow$ maps one to one with $\rightarrow$-right applications of $\text{LMT} \rightarrow$;
(iii) $\rightarrow$-left applications of $\text{LJ} \rightarrow$ maps to $\text{LMT} \rightarrow$ in three different possible sub-cases, according to the context $(\alpha \rightarrow \beta, q)$ in which the rule is being applied in $\text{LJ} \rightarrow$. We have to consider the fragment of $\text{LMT} \rightarrow$ already translated to decide the appropriate case.

- If the context is not yet focused neither expanded

  Then, one application of $\rightarrow$-left in $\text{LJ} \rightarrow$ maps to two applications of rules in $\text{LMT} \rightarrow$: first, a focus application, then an $\rightarrow$-left application.

- If the context is already focused but not yet expanded

  Then, one application of $\rightarrow$-left in $\text{LJ} \rightarrow$ maps to one application of $\rightarrow$-left in $\text{LMT} \rightarrow$.

- If the context $(\alpha \rightarrow \beta, q)$ is already focused and expanded

  Then, the one application of $\rightarrow$-left in $\text{LJ} \rightarrow$ maps to the height of the $\text{LMT} \rightarrow$ proof fragment produced by the execution of the $\text{PROOFUNTIL}$ function. Let this height be called $h$.  

25
| LJ$\rightarrow$ | LMT$\rightarrow$ | Map |
|----------------|----------------|-----|
| axiom          | axiom          | 1:1 |
| $\rightarrow$-right | $\rightarrow$-right | 1:1 |
| focused        | expanded       |     |
| No             | No             | 1:2 | 1 focus and 1 $\rightarrow$-left |
| Yes            | No             | 1:1 | 1 $\rightarrow$-left |
| Yes            | Yes            | 1:h | 1 to the size of $PROOFUNTIL$ |

Table 1: Mapping the size of $LJ\rightarrow$ proofs into $LMT\rightarrow$

**Lemma 1** The height $h$ that defines the size of the proof fragment returned by the function $PROOFUNTIL$ has a maximum limit of $2^{2\log_2|\alpha|}$, where $\alpha$ is the main formula of the initial sequent of the proof in $LMT\rightarrow$.

**Proof 3** Consider a proof $\prod_{LJ\rightarrow}$ of an initial sequent in $LJ\rightarrow$ with the form $\Rightarrow \alpha$. The process of translating $\prod_{LJ\rightarrow}$ to $LMT\rightarrow$ produces a proof $\prod_{LMT\rightarrow}$ with the initial sequent in the form $\{} \Rightarrow [] \alpha$. Consider that $\alpha$ has the form $\alpha_1 \rightarrow \alpha_2$. In some point of the translation to $LMT\rightarrow$, we reach a point where a context $(\psi \rightarrow \varphi, q)$ is already focused and expanded in the already translated part of the proof $\prod_{LMT\rightarrow}$. At this point, the function $PROOFUNTIL$ generates a fragment of the proof $\prod_{LMT\rightarrow}$, call it $\Sigma$ of size $h$. The height $h$ is bound by the number of applications of $\rightarrow$-left rules in $\Sigma$, which can be determined by the multiplication of the degree of the formula $\alpha_1$ (that bounds the number of possible implicational formulas in the left side of a sequent in $LMT\rightarrow$) by the maximum number of atomic formulas ($n$) inside the $[]$-area in the highest branch of $\Sigma$ (each $p_i$ inside the $[]$-area allows one application of the restart rule). Thus we can formalize this in the following manner:

\[ h = n \times |\alpha_1| \]
\[ h = |\alpha| \times |\alpha| \]
\[ h = |\alpha|^2 \]

Since
\[ |\alpha|^2 = 2^{2\log_2|\alpha|} \]

Then, we have that
\[ h = 2^{2\log_2|\alpha|} \]

**Theorem 2** Let $\alpha$ be a $M\rightarrow$ tautology. The size of any proof of $\alpha$ generated by $S-LMT\rightarrow$ has an upper bound of $|\alpha| \cdot 2^{\alpha+1+2\log_2|\alpha|}$.
Proof 4 Considering the size of proofs for a formula $\alpha$ using $\text{LJ} \rightarrow$, the mapping in Table 1 and the Lemma 1, the proof follows directly.

Theorem 3 $\text{S-LMT} \rightarrow$ always terminates.

Proof 5 To guarantee termination, we use the upper bound presented in Theorem 2 to limit the height of opened branches during the $\text{LMT} \rightarrow$ proof search process. The strategy presented in Section 5.1 forces an ordered application of rules that produces all possible combinations of formulas to be expanded in the right and left sides of generated sequents. In other words, when the upper bound is reached, the proof search had, for sure, applied all possible expansion. The procedure can only continue by applying an already done expansion.

- $\rightarrow$-right rule is applied until we obtain an atomic formula on the right side.
- focus rule is applied until every non-labeled formula becomes focused. The same formula can not be focused twice unless a restart rule is applied.
- $\rightarrow$-left rule can not be applied more than once to the same context unless a restart rule is applied.
- between two applications of the restart rule in a branch there is only one possible application of a $\rightarrow$-left rule for a context $(\alpha, q)$. $\alpha$ and $q$ are always subformulas of the initial formula.
- restart rule is applied for each atomic formula that appears on the right side of sequents in a branch in the order of its appearance in the |]-area, which means that proof search will apply the restart rule for each $p_i$, $i = 1, \ldots, n$ until the branch reaches the defined limit.

The proof of completeness of $\text{LMT} \rightarrow$ is closely related to this strategy and with the way the proof tree is labeled during the proving process. Section 5.4 presents the soundness proof of $\text{LMT} \rightarrow$ and Section 5.5, the completeness proof.

5.4 Soundness

In this section, we prove the soundness of $\text{LMT} \rightarrow$. A few basic facts and definitions used in the proof follow.

Definition 3 A sequent $\{\Delta\}, \gamma_{p_1}^1, \gamma_{p_2}^2, \ldots, \gamma_{p_n}^n, \Delta \Rightarrow \{p_1, p_2, \ldots, p_n\}, \varphi$ is valid, if and only if, $\Delta \dashv \vdash \varphi$ or $\exists i(\bigcup_{k=1}^{n} \gamma_{p_k}) \vdash p_i$, for $i = 1, \ldots, n$. 27
Definition 4 We say that a rule is sound, if and only if, in the case of the premises of the sequent are valid sequents, then its conclusion also is.

A calculus is sound, if and only if, each of its rules is sound. We prove the soundness of \( \text{LMT} \rightarrow \) by showing that this is the case for each one of its rules.

Proposition 2 Considering validity of a sequent as defined in Definition 3, \( \text{LMT} \rightarrow \) is sound.

Proof 6 We show that supposing that premises of a rule are valid then, the validity of the conclusion follows. In the sequel, we analyze each rule of \( \text{LMT} \rightarrow \).

\( \rightarrow \)-left We need to analyze both premises together. Thus we have the combinations described below.

- Supposing the left premise is valid because \( \alpha \rightarrow \beta, \Delta', \Delta \models \alpha \) and the right premise is valid because \( \alpha \rightarrow \beta, \Delta', \Delta, \beta \models q \). We also know that \( \alpha \rightarrow \beta \in \Delta \) and \( \Delta' \subseteq \Delta \). In this case, the conclusion holds:

\[
\begin{array}{c}
\alpha \rightarrow \beta \\
\prod_{\alpha} \Delta' \Delta \\
\hline
\beta \\
\end{array}
\]

- Supposing the left premise is valid because \( \exists i (\bigcup_{k=1}^{i} \Upsilon_{p_k}) \models p_i \), for \( i = 1, \ldots, n \), the conclusion holds as it is the same. Supposing the left premise is true because \( \Delta^q \models q \), the conclusion also holds, as \( \Delta^q = \Delta \).

- Supposing the right premise is valid because \( \exists i (\bigcup_{k=1}^{i} \Upsilon_{p_k}) \models p_i \), for \( i = 1, \ldots, n \), then conclusion also holds.

restart Here, we have three cases to evaluate.

- Supposing the premise is valid because \( \Upsilon_{p_1}, \Upsilon_{p_2}, \ldots, \Upsilon_{p_i} \models p_i \), then \( \exists i (\bigcup_{k=1}^{i} \Upsilon_{p_k}) \models p_i \), for \( i = 1, \ldots, n \). The conclusion is also valid.

- Supposing the premise is valid because \( \exists j (\bigcup_{k=i+1}^{j} \Upsilon_{p_k}) \models p_j \), for \( j = i+1, \ldots, n \), then conclusion also holds.
• Supposing the premise is valid because $\Delta^q \models q$, then $\Delta \models q$ and, as $\Delta' \subseteq \Delta$, $\Delta', \Delta \models q$.

$\rightarrow$-right

• Supposing the premise is valid because $\Upsilon_{p_1}^1, \Upsilon_{p_2}^2, \ldots, \Upsilon_{p_i}^i \models p_i$, then $\exists i(\bigcup_{k=1}^i \Upsilon_{p_k}^i) \models p_i$, for $i = 1, \ldots, n$. This is also valid in the conclusion.

• Supposing the premise is valid because $\Delta', \Delta, \alpha \models \beta$, then every Kripke model that satisfies $\Delta'$, $\Delta$ and $\alpha$ also satisfies $\beta$. We know that $\Delta' \subseteq \Delta$. Those models also satisfies $\alpha \rightarrow \beta$ and, then, conclusion also holds.

focus

• Supposing the premise is valid because $\Upsilon_{p_1}^1, \Upsilon_{p_2}^2, \ldots, \Upsilon_{p_i}^i \models p_i$, then $\exists i(\bigcup_{k=1}^i \Upsilon_{p_k}^i) \models p_i$, for $i = 1, \ldots, n$. This is also valid in the conclusion.

• Supposing the premise is valid because $\Delta', \alpha, \Delta \models \beta$, then the conclusion also holds as $\Delta', \Delta, \alpha \models \beta$.

From Proposition 2, we conclude that $\text{LMT}^\rightarrow$ only prove tautologies.

5.5 Completeness

By Observation 1.ii we know that a top sequent of an open branch in an attempt proof tree has the general form below, where $q$ is an atomic formula:

$$
\{\Delta'\}, \Upsilon_{p_1}^1, \Upsilon_{p_2}^2, \ldots, \Upsilon_{p_i}^i, \Delta \Rightarrow [p_1, p_2, \ldots, p_n], q
$$

From Definition 3 and considering that $\Delta' \in \Delta$ in any sequent of an attempt proof tree following our proposed strategy, we can define an invalid sequent as follows:

**Definition 5** A sequent is invalid if and only if $\Delta \not\models q$ and $\forall i(\bigcup_{k=1}^i \Upsilon_{p_k}^i) \not\models p_i$, for $i = 1, \ldots, n$.

Our proof of completeness starts with a definition of atomic formulas on the left and right sides of a top sequent.

**Definition 6** We can construct a Kripke counter-model $M$ that satisfies atomic formulas on the right side of a top sequent, and that falsifies the atomic formula on the left. This construction can be done in the following way:
1. The model $\mathcal{M}$ has an initial world $w_0$.

2. By the proof strategy, we can conclude that, in any sequent of the proof tree, $\Upsilon^p_1 \subseteq \Upsilon^p_2 \subseteq \cdots \subseteq \Upsilon^p_n \subseteq \Delta$. We create a world in the model $\mathcal{M}$ corresponding for each one of these bags of formulas and, using the inclusion relation between them, we define a respective accessibility relation in the model $\mathcal{M}$ between such worlds. That is, we create worlds $w_{\Upsilon^p_1}, w_{\Upsilon^p_2}, \ldots, w_{\Upsilon^p_n}, w_\Delta$ related in the following form: $w_{\Upsilon^p_1} \preceq w_{\Upsilon^p_2} \preceq \cdots \preceq w_{\Upsilon^p_n} \preceq w_\Delta$. As $w_0$ is the first world of $\mathcal{M}$, it precedes $w_{\Upsilon^p_1}$, that is, $w_0 \preceq w_{\Upsilon^p_1}$ is also included in the accessibility relation. If $\Upsilon^p_i = \Upsilon^p_{i+1}$, for $i = 1, \ldots, n$, then the associated worlds that correspond to those sets have to be collapsed in a single world $w_{\Upsilon^p_i} \preceq w_{\Upsilon^p_{i+1}}$. In this case, the previous relation $w_{\Upsilon^p_i} \preceq w_{\Upsilon^p_{i+1}}$ is removed from the relation of the model $\mathcal{M}$ and the pairs $w_{\Upsilon^p_{i-1}} \preceq w_{\Upsilon^p_i}$ and $w_{\Upsilon^p_{i+1}} \preceq w_{\Upsilon^p_{i+2}}$ become respectively $w_{\Upsilon^p_{i-1}} \preceq w_{\Upsilon^p_i} \preceq w_{\Upsilon^p_{i+1}}$ and $w_{\Upsilon^p_{i+1}} \preceq w_{\Upsilon^p_{i+2}}$.

3. By the Definition 5 of an invalid sequent, $\Delta \not\models q$. The world $w_\Delta$ will be used to guarantee this. We set $q$ false in $w_\Delta$, i.e., $\mathcal{M} \not\models w_\Delta q$. We also set every atomic formula that is in $\Delta$ as true, i.e., $\forall p, p \in \Delta, \mathcal{M} \models w_\Delta p$.

4. By the Definition 5 of an invalid sequent, we also need that $\forall i \left( \bigcup_{k=1}^{i} \Upsilon^p_k \right) \not\models p_i$, for $i = 1, \ldots, n$. Thus, for each $i, i = 1, \ldots, n$ we set $\mathcal{M} \not\models w_{\Upsilon^p_i} p_i$ and $\forall p, p \in \Upsilon^p_i$, being $p$ an atomic formula, $\mathcal{M} \models w_{\Upsilon^p_i} p$. In the case of collapsed worlds, we keep the satisfaction relation of the previous individual worlds in the collapsed one.

5. In $w_0$ set every atomic formula inside the $\lbrack\lbrack$-area (all of them are atomic) as false. That is, $\mathcal{M} \not\models w_0 p_i$, for $i = 1, \ldots, n$. We also set the atomic formula outside the $\lbrack\lbrack$-area false in this world: $\mathcal{M} \not\models w_0 q$. Those definitions make $w_0$ consistent with the $\preceq$ relation of $\mathcal{M}$.

The Figure 6 shows the general shape of counter-models following the steps enumerated above. This procedure to construct counter-model allows us to state the following lemma:

**Lemma 2** Let $S$ be a top sequent of an open branch in an attempt proof tree generated by the strategy presented in Section 5.1. Then we can construct a Kripke model $\mathcal{M}$ with a world $u$ where $\mathcal{M} \not\models u S$, using the proposed counter-model generation procedure.
Proof 7 We can prove this by induction on the degree of formulas in $\Delta$. From Definition 6, items 3 and 4 we know the value of each atomic formula in the worlds $w_\Delta$ and in each world $w_{\Upsilon^p_i}$. The inductive hypothesis is that every formula in $\Delta$ is true in $w_\Delta$. Thus, as $\Upsilon^p_1 \subseteq \Upsilon^p_2 \subseteq \cdots \subseteq \Upsilon^p_n \subseteq \Delta$, every formula in $\Upsilon^p_i$ is true in $w_{\Upsilon^p_i}$, for $i = 1, \ldots, n$.

Thus, we have two cases to consider:

1. The top sequent is in the rightmost branch of the proof tree ($\perp$-area is empty).

Let $\alpha \rightarrow \beta$ be a formula in $M^\rightarrow$ that is in $\Delta$. We show that $M \vDash w_\Delta \alpha \rightarrow \beta$. In this case, by the proof strategy, $\beta \equiv (\beta_1 \rightarrow (\beta_2 \rightarrow \cdots \rightarrow (\beta_m \rightarrow p)))$, where $p$ is an atomic formula. By Definition 6.3 $\vDash w_\Delta p$. As $w_\Delta$ has no accessible world from it (except for itself), $\vDash w_\Delta \beta$. By the proof strategy, $\beta_m \rightarrow p, \beta_{m-1} \rightarrow \beta_m \rightarrow p, \ldots, \beta_2 \rightarrow \cdots \rightarrow \beta_m \rightarrow \beta_m \rightarrow p, \beta_1 \rightarrow \beta_2 \rightarrow \cdots \rightarrow \beta_{m-1} \rightarrow \beta_m \rightarrow p$ also are in $\Delta$. The degree of each of these formulas is less than the degree of $\alpha \rightarrow \beta$ and, by the induction hypothesis, all of them are true in $w_\Delta$. Thus $\vDash w_\Delta \beta$ and $\vDash w_\Delta \alpha \rightarrow \beta$.

As the $\perp$-area is empty, the sets $\Upsilon^p_i$ are also empty. The counter-model only has two words, $w_0$ and $w_\Delta$, following the properties described in Definition 6.

2. The top sequent is in any other branch that is not the rightmost one ($\perp$-area is not empty).
Let $\alpha \rightarrow \beta$ be a formula in $M^\rightarrow$ that is in $\Delta$. We show that $M \models_{w_\Delta} \alpha \rightarrow \beta$. In this case, by the proof strategy, $\alpha \equiv (\alpha_1 \rightarrow (\alpha_2 \rightarrow \cdots \rightarrow (\alpha_m \rightarrow q)))$, where $q$ is the atomic formula in the right side of the sequent, out of the $[|]-$area. By Definition 6.3 $\not \models_{w_\Delta} q$. By the proof strategy, $\alpha_1, \alpha_2, \ldots, \alpha_m$ also are in $\Delta$. The degree of each of these formulas is less than the degree of $\alpha \rightarrow \beta$ and, by the induction hypothesis, all of them are true in $w_\Delta$. This ensures $\not \models_{w_\Delta} \alpha$ and $\models_{w_\Delta} \alpha \rightarrow \beta$.

Considering now a formula $\alpha \rightarrow \beta$ from $M^\rightarrow$ that is in $\Upsilon^p_i$. By Definition 6.2, $\alpha \rightarrow \beta$ also belongs to $\Delta$. From the last paragraph, we show that, for any formula $\alpha \rightarrow \beta \in \Delta$, $\not \models_{w_\Delta} \alpha$. As $\not \models_{w_\Delta} \alpha$, by the accessibility relation of the Kripke model, $\not \models_{w_{\Upsilon^p_i}} \alpha$, for each $i = 1, \ldots, n$. Thus, the value of $\alpha \rightarrow \beta$ is defined in any of these worlds by the value of $\alpha \rightarrow \beta$ in $w_\Delta$, that we showed to be true. Thus, $\models_{w_{\Upsilon^p_i}} \alpha \rightarrow \beta$.

As stated in Definition 6.2, $\Upsilon^p_1 \subseteq \Upsilon^p_2 \subseteq \cdots \subseteq \Upsilon^p_n \subseteq \Delta$ and following the accessibility relation rule of the $M^\rightarrow$ semantic (relations are reflexive and transitive) we conclude that:

$$
M \models_{w_0} \Upsilon^p_1 \not \models_{w_0} p_1 \\
\models_{w_0} \Upsilon^p_2 \not \models_{w_0} p_2 \\
\vdots \\
\models_{w_0} \Upsilon^p_n \not \models_{w_0} p_n \\
\models_{w_0} \Delta \not \models_{w_0} q
$$

Proving Lemma 2.

**Definition 7** A rule is said invertible or double-sound iff the validity of its conclusion implies the validity of its premises.

In other words, by Definition 7, we know that a counter-model for a top sequent of a proof tree that can not be expanded anymore can be used to construct a counter-model to every sequent in the same branch of the tree until the conclusion (root sequent). In the case of the $\rightarrow$-right rule in our system, not just if the premise of the rule has a counter-model, then so does the conclusion, but the same counter-model will do. Weich (1998) called rules with this property preserving counter model. Dyckhoff (personal communication, 2015) proposed to call this
kind of rules of strongly invertible rules. In the case of the \( \rightarrow \)-left rule, this is the same when one of the premises is valid, but, considering the case that both premises are not valid, we need to mix the counter-models of both sides to construct the counter-model for the conclusion of the rule. This way to produce counter-models is what we call a weakly invertible rule.

**Lemma 3** The rules of \( \text{LMT} \rightarrow \) are invertible.

**Proof 8** We show that the rules of \( \text{LMT} \rightarrow \) are invertible when considering a proof tree labeled in the schema presented in Section 5.1. We prove that for the structural rules (focus and restart) and \( \rightarrow \)-right, from the existence of a Kripke model that makes the premise of the rule invalid follows that the conclusion is also invalid. For the \( \rightarrow \)-left rule, from the Kripke models of the premises, we can construct a Kripke model that also makes the conclusion of the rule invalid.

\[ \text{\( \rightarrow \)-right} \]

If the premise is invalid, then there is a Kripke model \( \mathcal{M} \) where \( \Delta', \Delta, \alpha \not\models \beta \) and \( \forall i (\bigcup_{k=1}^{i} \Upsilon_k^p_i) \not\models p_i \), for \( i = 1, \ldots, n \) in a given world \( u \) of \( \mathcal{M} \). Thus, in the conclusion we have:

- By the definition of semantics of Section ??, there have to be a world \( v \), \( u \preceq v \), in the model \( \mathcal{M} \) where \( \Delta', \Delta, \alpha \) are satisfied and where \( \beta \) is not. Thus, in \( u \), \( \alpha \rightarrow \beta \) can not hold.
- By the model \( \mathcal{M} \), for each \( i \), exists a world \( v_i \), \( u \preceq v_i \), where \( \models_{v_i} \Upsilon_i^p_i \) and \( \not\models_{v_i} p_i \).
- Thus, the conclusion is also invalid.

\[ \text{\( \rightarrow \)-left} \]

Considering that one of the premises of \( \rightarrow \)-left is not valid, the conclusion also is. We have to evaluate three cases:

1. **The right premise is invalid but the left premise is valid.** Then there is a Kripke model \( \mathcal{M} \) where \( \alpha \rightarrow \beta, \Delta', \Delta, \beta \not\models q \) and \( \forall i (\bigcup_{k=1}^{i} \Upsilon_k^p_i) \not\models p_i \), for \( i = 1, \ldots, n \) from a given world \( u \). Thus, in the conclusion we have:

   - By the model \( \mathcal{M} \), there have to be a world \( v \), \( u \preceq v \), in the model where \( \alpha \rightarrow \beta, \Delta', \Delta, \beta \) are satisfied and where \( q \) is not.
   - By the model \( \mathcal{M} \), for each \( i \), exists a world \( v_i \), \( u \preceq v_i \), where \( \models_{v_i} \Upsilon_i^p_i \) and \( \not\models_{v_i} p_i \).
   - Thus, the conclusion is invalid too.
2. **The left premise is invalid but the right premise is valid.** Then there is a Kripke model $\mathcal{M}$ where $\alpha \rightarrow \beta, \Delta', \Delta \not\models \alpha$ and $\forall i(\bigcup_{k=1}^{i} \Upsilon_k^p) \not\models p_i$, for $i = 1, \ldots, n$, and $\Delta^q \not\models q$ from a given world $u$. Thus, in the conclusion we have:

- By the model $\mathcal{M}$, there have to be a world $v$, $u \preceq v$, in the model where $\alpha \rightarrow \beta, \Delta', \Delta$ are satisfied and where $\alpha$ is not.
- By the model $\mathcal{M}$, for each $i$, exists a world $v_i$, $u \preceq v_i$, where $\models_{v_i} \Upsilon_i^p$ and $\not\models_{v_i} p_i$.
- We also know by $\mathcal{M}$ that there is a world $v_{\Delta^q}$, $u \preceq v_{\Delta^q}$, where $\models_{v_{\Delta^q}} \Delta^q$ and $\not\models_{v_{\Delta^q}} q$. We also have that $\Delta^q = \Delta$ and that $\alpha \rightarrow \beta \in \Delta$. Therefore, $\models_{v_{\Delta^q}} \Delta'$ and $\models_{v_{\Delta^q}} \alpha \rightarrow \beta$.
- Thus, the conclusion can not be valid.

3. **Both left and right premises are invalid.** Then there are two models $\mathcal{M}_1$ and $\mathcal{M}_2$, from the right and left premises respectively. In $\mathcal{M}_1$ there is a world $u_1$ that makes the right sequent invalid as described in item 1. In $\mathcal{M}_2$ there is a world $u_2$ that makes the sequent of the left premise invalid as described in item 2. Considering the way Kripke models are constructed based on Lemma 2, we know that $u_1$ and $u_2$ are root worlds of their respective counter-models. Thus, converting the two models into one, $\mathcal{M}_3$, by mixing $u_1$ and $u_2$ in the root of $\mathcal{M}_3$, called $u_3$, we have that in $u_3$:

- $\alpha \rightarrow \beta, \Delta', \Delta$ are satisfied and $\alpha$ is not.
- for $i = 1, \ldots, n$, we have that $\models_{u_3} \Upsilon_i^p$ and $\not\models_{u_3} p_i$.
- $\not\models_{u_3} q$
- Thus, the conclusion is also invalid.

**focus** If we have a model that invalidates the premise, this model also invalidates the conclusion as the sequents in the premise and in the conclusion are the same despite the repetition of the focused formula $\alpha$.

**restart** If the restart premise is invalid, then there is a Kripke model $\mathcal{M}$ and a world $u$ from which $\Upsilon_1, \Upsilon_2, \ldots, \Upsilon_i \not\models p_i$ and $\forall j(\bigcup_{k=1}^{j} \Upsilon_k^p) \models p_j$, for $j = i + 1, \ldots, n$, and $\Delta^q \not\models q$. Thus, in the conclusion we have:

- By the model $\mathcal{M}$, there have to be a world $v$, $u \preceq v$, in the model where $\Upsilon_1, \Upsilon_2, \ldots, \Upsilon_i$ are satisfied and where $p_i$ is not. Each $\Upsilon_k$ has the same formulas as $\Upsilon_k^p$ and, by the restart condition, we know that $\not\models p_k$, for $k = 1, \ldots, i$.  

34
• By the model $\mathcal{M}$, for each $j$, exists a world $v_j$, $u \preceq v_j$, where $\models_{v_j} \top^p_j$ and $\not\models_{v_j} p_j$.
• We also know by $\mathcal{M}$ that there is a world $v_q$, $u \preceq v_q$, where $\models_{v_q} \Delta^q$ and $\not\models_{v_q} q$.
  We also have that $\Delta' \subseteq \Delta$. Therefore, $\models_{v_q} \Delta'$.
• Thus, the conclusion is invalid.

Now we can state a proposition about completeness of $\text{LMT}^{\rightarrow}$:

**Proposition 3** $\text{LMT}^{\rightarrow}$ is complete regarding the proof strategy presented in Section 5.1

**Proof 9** It follows direct from Proposition 3 ($\text{LMT}^{\rightarrow}$ terminates) and Lemma 2 (we can construct a counter-model for a top sequent in a terminated open branch of $\text{LMT}^{\rightarrow}$) and Lemma 3 (the rules of $\text{LMT}^{\rightarrow}$ are invertible).

### 5.6 Examples

**Example 8** As an example, consider the Peirce formula, $((A \rightarrow B) \rightarrow A) \rightarrow A$, which is very known to be a Classic tautology, but not a tautology in Intuitionistic neither in Minimal Logic (which includes $\text{M}^{\rightarrow}$). Proof (11) below shows the open branch of an attempt proof tree for this formula. Following our termination criteria, this branch should be higher than the fragment below, but, to help improve understanding, we stop that branch in the point from which proof search just produces repetition (similar to the use of a loop checker).
\[
\begin{aligned}
\vdots \quad & \rightarrow_1, A, (\rightarrow_1)^A, \{ \rightarrow_1, A \} \Rightarrow 28 [A, B], B \\
\rightarrow_1, A, (\rightarrow_1)^A, \{ \rightarrow_1, A \} \Rightarrow 26 [A, B], B & \quad \text{focus} \rightarrow_1 \\
\rightarrow_1, A, (\rightarrow_1)^A, \{ \rightarrow_1 \} \Rightarrow 25 [A, B], B & \quad \text{focus} \rightarrow_1 \\
\rightarrow_1, A, (\rightarrow_1)^A, \{ \} \Rightarrow 22 [A, B], B & \quad \text{restart} \rightarrow_1 \\
\rightarrow_1, (\rightarrow_1)^A, \{ \} \Rightarrow 19 [B, A], B & \quad \text{restart} \rightarrow_1 \\
\rightarrow_1, (\rightarrow_1)^A, \{ \} \Rightarrow 16 [B, A], B & \quad \text{restart} \rightarrow_1 \\
\rightarrow_1, (\rightarrow_1)^A, \{ \} \Rightarrow 13 [B, A], \rightarrow_0 & \quad \text{restart} \rightarrow_1 \\
\rightarrow_1, (\rightarrow_1)^A, \{ \} \Rightarrow 9 [A], B & \quad \text{restart} \rightarrow_1 \\
\rightarrow_1, (\rightarrow_1)^A, \{ \} \Rightarrow 6 [A], B & \quad \text{restart} \rightarrow_1 \\
\rightarrow_1, (\rightarrow_1)^A, \{ \} \Rightarrow 3 [A], \rightarrow_0 & \quad \text{restart} \rightarrow_1 \\
\rightarrow_1, (\rightarrow_1)^A, \{ \} \Rightarrow 0 (\{ A \Rightarrow 0 \} \Rightarrow 2 A), \{ \} & \quad \text{restart} \rightarrow_2 \\
\end{aligned}
\]
From Lemma 3, we can extend this counter-model to falsify the initial sequent $(\Rightarrow_0)$, showing that the Pierce rule does not hold on $M^\rightarrow$.

Example 9 As another example, we can consider the Dummett formula: $(A \rightarrow B) \lor (B \rightarrow A)$. It is known that a Kripke counter-model that falsifies this formula needs at least two branches in $\text{Int}$ and $\text{Min}$, so it is also in $M^\rightarrow$. This example allows us to understand how to mix the right and left premises counter-models to falsifies a conclusion sequent of a $\rightarrow$-left rule.

As we want to use $\text{LMT}^\rightarrow$, we need to convert the Dummett formula from the form above to its implicational version. We use here the general translation presented in Haeusler (2015). Thus, the translated version is a formula $\alpha$ as follows:

$$\alpha \equiv (((A \rightarrow B) \rightarrow A) \rightarrow (((B \rightarrow A) \rightarrow A) \rightarrow (C \rightarrow A))) \rightarrow (((A \rightarrow B) \rightarrow B) \rightarrow (((B \rightarrow A) \rightarrow B) \rightarrow (C \rightarrow B)))$$

To shorten the presentation, consider the following abbreviations:

$$\alpha_1 = ((A \rightarrow B) \rightarrow A) \rightarrow (((B \rightarrow A) \rightarrow A) \rightarrow (C \rightarrow A))$$
$$\alpha_2 = ((A \rightarrow B) \rightarrow B) \rightarrow (((B \rightarrow A) \rightarrow B) \rightarrow (C \rightarrow B))$$

The tree (13), presents a shortened version of a completely expanded attempt proof tree in $\text{LMT}^\rightarrow$ for the Dummett formula in implicational form. We concentrate in show the application of the operational rules ($\rightarrow$-left and $\rightarrow$-right). We
removed from the tree the focus area on the left of each sequent and the applications of structural rules. We also exclude from the tree the labeled formulas, just showing the necessary formulas for the specific point in the tree. The list of sequents above the boxed sequents in the tree represents the top sequents of each branch after the total expansion of $\alpha_2$. 
From the open branches of the tree (13) we extract the following three models. Models $M_1$ repeats in some branches. Thus, we only represent it once here. We indicated in tree (13), in the top sequents of each branch, the corresponding model generated on it, following the Definition 6.

\[
\begin{array}{c}
M_1 \\
w_0 \\
w_1 \not\models C \quad w_2 \not\models C \quad w_3 \not\models C \quad w_4 \not\models C \\
| \quad | \quad | \\
w_{11} \not\models B \quad w_{21} \models A \\
| \quad | \\
w_{12} \models A \quad w_{22} \models B
\end{array}
\]

\[
\begin{array}{c}
M_2 \\
w_0' \\
w_1' \not\models C \quad w_2' \not\models C \\
| \quad | \\
w_{11}' \not\models B \quad w_{21}' \models B \\
| \quad | \not\models C \\
w_{12}' \models B \not\models A
\end{array}
\]

\[
\begin{array}{c}
M_3 \\
w_0'' \\
w_1'' \not\models C \quad w_2'' \not\models C \quad w_3'' \not\models C \quad w_4'' \not\models C \\
| \quad | \quad | \\
w_{11}'' \not\models B \quad w_{21}'' \not\models B \quad w_{31}'' \models A \quad w_{41}'' \models A \\
| \quad | \quad | \\
w_{12}'' \models A \not\models B \not\models C
\end{array}
\]

40
Therefore, at the point in the tree where the rule $\to$-left is applied to the context $(\alpha_1, C)$ (the single labeled rule application in the tree (13)) we have the join of these models. Counter-model $M_4$ below represent this unification (to legibility we remove repeated branches in $M_4$). Thus, $M_4 \not\models \alpha$:

\[
\begin{array}{cccc}
M_4 & w_0^* \\
\mid & \mid & \mid & \mid \\
w_1^* & w_2^* & w_3^* & w_4^* & w_5^* \\
\not\models C & \not\models C & \not\models C & \not\models C & \not\models C \\
\mid & \mid & \mid & \mid & \mid \\
w_{11}^* & w_{21}^* & w_{31}^* & w_{51}^* \\
\not\models B & \not\models B & \models A & \models A \\
\mid & \mid & \mid & \mid \\
w_{12}^* & w_{22}^* \\
\not\models B & \not\models A \\
\mid & \mid \\
\not\models B & \not\models B \\
\models A & \models B \\
\models A & \models B \\
\models A & \models B \\
\end{array}
\]

6 Conclusion

We presented here the definition of a sequent calculus for proof search in the context of the Propositional Minimal Implicational Logic ($\text{M}^\rightarrow$). Our calculus, called $\text{LMT}^\rightarrow$, aims to perform the proof search for $\text{M}^\rightarrow$ formulas in a bottom-up, forward-always approach. Termination of the proof search is achieved without using loop checkers during the process. $\text{LMT}^\rightarrow$ is a deterministic process, which means that the system does not need an explicit backtracking mechanism to be complete. In this sense, $\text{LMT}^\rightarrow$ is a bicomplete process, generating Kripke counter-models from search trees produced by unsucess proving processes.

In the definition of the calculus, we also presented some translations between deductive systems for $\text{M}^\rightarrow$: ND to $\text{LJ}^\rightarrow$ and $\text{LJ}^\rightarrow$ to $\text{LMT}^\rightarrow$. We also established a relation between $\text{LMT}^\rightarrow$ and Fitting’s Tableaux Systems for $\text{M}^\rightarrow$ regards the counter-model generation in those systems.

We keep the development of a theorem prover for $\text{LMT}^\rightarrow$ in (https://github.com/jeffsantos/GraphProver).

As future work, we can enumerate some features to be developed or extended in the system as well as some new research topics that can be initialized.

- **Precise upper bound for termination** The upper bound used here for achieving termination in $\text{LMT}^\rightarrow$ is a very high bound. Many non-theorems can be identified in a small number of steps. We can still explore options to shorten the size of the proof search tree. Even with theorems, our labeling
mechanism, in conjunction with the usage of the restart rule, produces many repetitions in the proof tree.

- **Compression and sharing** Following the techniques proposed by Gordeev and Hausler (2016) we can explore new ways to shorten the size of proofs generated by LMT→.

- **Minimal counter-models** The size of the generated counter-model in LMT→ still takes into account every possible combination of subformulas, yielding Kripke models with quite a lot of worlds. There is still work to be done in order to produce smaller models. Stoughton (1996) presents an implementation of the systems in Dyckhoff (1992) and in Pinto and Dyckhoff (1995) with the property of “minimally sized, normal natural deduction proofs of the sequent, or it finds a "small" tree-based Kripke counter-model of the sequent” using the words of the author. These references can be a good start point to improve LMT→ counter-model generation.

References

Andreoli, J.-M. (1992). Logic programming with focusing proofs in linear logic. *Journal of Logic and Computation*, 2(3):297–347.

Dowek, G. and Jiang, Y. (2006). Eigenvariables, bracketing and the decidability of positive minimal predicate logic. *Theoretical Computer Science*, 360(1):193–208.

Dyckhoff, R. (1992). Contraction-free sequent calculi for intuitionistic logic. *The Journal of Symbolic Logic*, 57(03):795–807.

Dyckhoff, R. (2016). Intuitionistic decision procedures since gentzen. In *Advances in Proof Theory*, pages 245–267. Springer.

Dyckhoff, R. and Lengrand, S. (2006). LJJQ: a strongly focused calculus for intuitionistic logic. In *Logical Approaches to Computational Barriers*, pages 173–185. Springer.

Ferrari, M., Fiorentini, C., and Fiorino, G. (2013). Contraction-free linear depth sequent calculi for intuitionistic propositional logic with the subformula property and minimal depth counter-models. *Journal of automated reasoning*, 51(2):129–149.

Gentzen, G. (1935). Untersuchungen über das logische schließen. i. *Mathematische zeitschrift*, 39(1):176–210.
Gordeev, L. and Haeusler, E. H. (2016). NP vs PSPACE. *arXiv preprint arXiv:1609.09562.*

Haeusler, E. H. (2015). Propositional logics complexity and the sub-formula property. *Electronic Proceedings in Theoretical Computer Science,* 179:1–16. Proceedings of DCM2014, Vienna, 2014.

Herbelin, H. (1995). A $\lambda$-calculus structure isomorphic to Gentzen-style sequent calculus structure. In *Computer Science Logic,* pages 61–75. Springer.

Heuerding, A., Seyfried, M., and Zimmermann, H. (1996). Efficient loop-check for backward proof search in some non-classical propositional logics. In *Theorem Proving with Analytic Tableaux and Related Methods,* pages 210–225. Springer.

Hirokawa, S. (1991). Number of proofs for implicational formulas. *Introduction to Mathematical Analysis (in Japanese),* 772:72–74.

Howe, J. M. (1997). Two loop detection mechanisms: a comparison. In *Automated Reasoning with Analytic Tableaux and Related Methods,* pages 188–200. Springer.

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

Liang, C. and Miller, D. (2007). Focusing and polarization in intuitionistic logic. In *Computer Science Logic,* pages 451–465. Springer.

Pinto, L. and Dyckhoff, R. (1995). Loop-free construction of counter-models for intuitionistic propositional logic. In *Symposia Gaussiana, Conf A,* pages 225–232. Walter de Gruyter & Co (Berlin).

Prawitz, D. (2006). *Natural deduction: A proof-theoretical study.* Courier Dover Publications.

Santos, J. d. B., Vieira, B. L., and Haeusler, E. H. (2016). A unified procedure for provability and counter-model generation in minimal implicational logic. *Electronic Notes in Theoretical Computer Science,* 324:165–179.

Seldin, J. P. (1998). Manipulating proofs.

Statman, R. (1974). *Structural Complexity of Proofs.* PhD thesis, Stanford University.

Stoughton, A. (1996). Porgi: a proof-or-refutation generator for intuitionistic propositional logic. In *CADE Workshop on Proof-search in Type-theoretic Languages,* pages 109–116.
Underwood, J. (1990). A constructive completeness proof for intuitionistic propositional calculus. Technical report, Cornell University.

Vorob’ev, N. N. (1970). A new algorithm for derivability in the constructive propositional calculus. *American Mathematical Society Translations*, 94(2):37–71.

Weich, K. (1998). Decision procedures for intuitionistic propositional logic by program extraction. In *Automated Reasoning with Analytic Tableaux and Related Methods*, pages 292–306. Springer.