

Characterization by Regulated Grammars of Calculations in the Gentzen System G

Rubén Monjaraz Hernández-Imbert¹ & José de Jesús Lavalle Martínez² *

Abstract— The aim of the present work is to present a characterization of the Gentzen System G by means of a regulated grammar, according to the Formal Languages Theory. Given a proposition, which is supposed to be a tautology, a grammar will be produced and the generation of the respective language will represent a calculation in G . This language can be interpreted as either, a proof or a counter-example of the given proposition. Mechanization of related tasks is made by means of the functional language ML. *Keywords:* propositional logic, Gentzen, regulated grammars, characterization, ML

1 Introduction

The Gentzen's system G [1] formally deals with logical propositions, establishing directly whether an arbitrary proposition, say \mathcal{A} , is a *tautology* (i.e. a proposition that is true for every possible truth assignment). This decision procedure is based on an attempt to falsify \mathcal{A} . A failure of this attempt indicates certainly that \mathcal{A} is a tautology. The method relies mainly on the concepts of *sequents* and *deduction trees*. System G can be characterized by programs written in languages such as ML, Prolog or Maude, by matrix-like structures [2] or by a Turing machine. The Wang's algorithm itself was originally written in punched cards ! [3].

This work proposes another kind of characterization for it, via regulated grammars, as defined in Formal Languages Theory. It can be seen that G inference rules and axiom schema have a *rewriting nature* : they all imply the substitution of an expression by another one in some determined context. The approach proposed here aims, on the basis of *constructivism*, to transform the nature of this procedure, in other words, a device that emulates G in a *context-free* way is introduced. From this point of view, the authors think, a regulated grammar (one that combines enough expressive power with the simplicity of context-free productions) is a suitable mechanism to perform *calculations* accordingly to G .

It must be noted that the ultimate goal of this work is

to simplify, at a certain extent, that facet of system G which involves rewriting operations, and thus we do not pursue an improvement in time.

Central ideas. Firstly, suppose that for a given arbitrary proposition \mathcal{A} , a grammar \mathcal{G} (of a certain kind) is constructed in such a way that it contains information regarding system G inference rules and axiom schema (how to apply them) as well as information about \mathcal{A} internal structure, such as its connectives, subpropositions and letters (to whom apply G inference rules and axiom schema). Secondly, since a grammar is inherently associated to a derivation tree, the language generated by \mathcal{G} will have a meaningful correspondence with the deduction tree of \mathcal{A} , generated by system G , in the sense that it will contain *words* representing the leaves of \mathcal{A} 's tree. As we will see, this will follow from the fact that a calculation in G will be associated to a number of derivations in \mathcal{G} . Thirdly, an examination of the language generated by \mathcal{G} will be enough to determine whether \mathcal{A}

- i. is a tautology: every word so produced represents an impossible truth assignment (a letter is assigned values *true* and *false* at once), equivalently, in G every leaf of the related deduction tree is labeled by a truth assignment of this kind, the attempt to falsify \mathcal{A} has just failed, or
- ii. is not a tautology: there is at least one word that represents a truth assignment which falsifies \mathcal{A} , equivalently, in G the deduction tree contains at least one leaf labeled in this way.

Finally, we mechanize the related tasks (proposition analysis, construction of the grammar, generation of the related language, etc.) with the help of a series of modules developed in ML.

2 Background

Definition 1 ([1],[4]). We denote by A, B, C, p, q the **propositional letters** (or letters), which will be combined with the **logical connectives** $\wedge, \vee, \supset, \neg$, to form **propositional sentences** (or propositions), as follows

1. A letter is a proposition,

*Submitted on February 19, 2007. Benemérita Universidad Autónoma de Puebla - Facultad de Ciencias Físico Matemáticas¹, Facultad de Ciencias de la Computación². Puebla, Pue. México 72570

2. If \mathcal{A} and \mathcal{B} are propositions, then $(\mathcal{A} \wedge \mathcal{B})$, $(\mathcal{A} \vee \mathcal{B})$, $(\mathcal{A} \supset \mathcal{B})$ and $(\neg \mathcal{A})$ are also propositions.
3. An expression is said to be a proposition only if it is obtained by conditions 1 and 2 above.

Definition 2 ([1],[2]). A **sequent** is a pair (Γ, Δ) of finite (possibly empty) sets $\Gamma = \{\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_n\}$ and $\Delta = \{\mathcal{B}_1, \mathcal{B}_2, \dots, \mathcal{B}_m\}$ of propositions. A sequent will be denoted as $\Gamma \rightsquigarrow \Delta$, where the members of Γ are considered to be true and those of Δ to be false. We will write Γ, \mathcal{A} and \mathcal{B}, Δ for the sets $\Gamma \cup \{\mathcal{A}\}$ and $\Delta \cup \{\mathcal{B}\}$, respectively. To keep notation as simple as possible, we will omit braces and write $\Gamma = \mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_n$ and $\Delta = \mathcal{B}_1, \mathcal{B}_2, \dots, \mathcal{B}_m$.

Definition 3 ([1],[2]). The Gentzen system G is summarized in Table 1 below, where the well known axiom schema has been replaced by a particular case (i.e., without loss of generality, the axiom schema will be used only at the letter level)

Table 1: System G

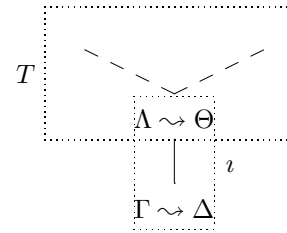
Axioms	$\Gamma, \mathcal{A} \rightsquigarrow \mathcal{A}, \Delta$
Inference Rules	
$\frac{\Gamma, \mathcal{A}, \mathcal{B} \rightsquigarrow \Delta}{\Gamma, \mathcal{A} \wedge \mathcal{B} \rightsquigarrow \Delta} (\wedge : l)$	$\frac{\Gamma \rightsquigarrow \mathcal{A}, \Delta \quad \Gamma \rightsquigarrow \mathcal{B}, \Delta}{\Gamma \rightsquigarrow \mathcal{A} \wedge \mathcal{B}, \Delta} (\wedge : r)$
$\frac{\Gamma, \mathcal{A} \rightsquigarrow \Delta \quad \Gamma, \mathcal{B} \rightsquigarrow \Delta}{\Gamma, \mathcal{A} \vee \mathcal{B} \rightsquigarrow \Delta} (\vee : l)$	$\frac{\Gamma \rightsquigarrow \mathcal{A}, \mathcal{B}, \Delta}{\Gamma \rightsquigarrow \mathcal{A} \vee \mathcal{B}, \Delta} (\vee : r)$
$\frac{\Gamma \rightsquigarrow \mathcal{A}, \Delta \quad \Gamma, \mathcal{B} \rightsquigarrow \Delta}{\Gamma, \mathcal{A} \supset \mathcal{B} \rightsquigarrow \Delta} (\supset : l)$	$\frac{\Gamma, \mathcal{A} \rightsquigarrow \mathcal{B}, \Delta}{\Gamma \rightsquigarrow \mathcal{A} \supset \mathcal{B}, \Delta} (\supset : r)$
$\frac{\Gamma \rightsquigarrow \mathcal{A}, \Delta}{\Gamma, \neg \mathcal{A} \rightsquigarrow \Delta} (\neg : l)$	$\frac{\Gamma, \mathcal{A} \rightsquigarrow \Delta}{\Gamma \rightsquigarrow \neg \mathcal{A}, \Delta} (\neg : r)$

Every inference rule is composed by one or two upper sequents called *premises* and a lower sequent called the *conclusion*. The proposition that is affected by the application of any rule is said to be the *principal formula* while the propositions introduced in the premises are said to be the *side formulae* (these are the major subpropositions of the principal formula, here they are denoted by \mathcal{A} and \mathcal{B}); the rest of the propositions (Γ and Δ), which are simply copied into the premises, are known as *extra formulae*. We will call such an application of a rule an *instance* (of the rule). Instances of these rules are called *reductions* if they are read from conclusion to premises, or *inferences* otherwise. Occasionally, we will denote reductions as $\Gamma \rightsquigarrow \Delta \xrightarrow{\iota} \Lambda \rightsquigarrow \Theta$, where ι is an inference rule and $\Lambda \rightsquigarrow \Theta$ is the resulting premise (or one of multiple premises). Note that this is just a *convention*, we

are not dealing with sequents in a linear way, as a matter of fact, branching will be preserved by our device. When we apply the axiom schema to a letter, obtaining at the same time the extra formulae, we have an instance of the axiom schema.

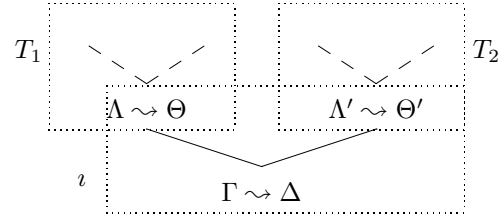
Definition 4 ([1],[2]). A **deduction tree** is defined in the following way

1. Every one-node tree labeled with an arbitrary sequent $\Gamma \rightsquigarrow \Delta$ (with the sole exception of $\Gamma = \Delta = \emptyset$) is a deduction tree.
2. For any deduction tree T with root labeled with a sequent $\Lambda \rightsquigarrow \Theta$ and for any instance of a G inference rule ι with single premise $\Lambda \rightsquigarrow \Theta$ and conclusion $\Gamma \rightsquigarrow \Delta$, the tree



is also a deduction tree.

3. For any two deduction trees T_1 and T_2 whose roots are respectively labeled with sequents $\Lambda \rightsquigarrow \Theta$ and $\Lambda' \rightsquigarrow \Theta'$ and for any instance of a G inference rule ι with two premises $\Lambda \rightsquigarrow \Theta$ and $\Lambda' \rightsquigarrow \Theta'$ and conclusion $\Gamma \rightsquigarrow \Delta$, the tree



is also a deduction tree.

A **proof tree** is nothing more than a deduction tree defined inductively over all one-node trees that are labeled with an instance of the axiom schema (instead of arbitrary ones) and conditions 2 and 3 as above. A **counterexample tree** is a deduction tree such that at least one of its leaves is labeled with a sequent $\Gamma \rightsquigarrow \Delta$ where Γ and Δ consist exclusively of letters and $\Gamma \cap \Delta = \emptyset$.

For a given arbitrary proposition \mathcal{A} we will compute the respective deduction tree starting with the node labeled with the sequent $\rightsquigarrow \mathcal{A}$ upwards.

We now focus on the so-called programmed grammar¹.

¹Please refer to [5] or [7] for some notions such as symbol, word or empty-word (λ).

Definition 5 ([5]). A **programmed grammar**² is a system $\mathcal{G} = (V_N, V_T, S, P)$ where V_N and V_T are the alphabets of nonterminals and terminals symbols, respectively, S is the start symbol and P is a (finite, non-empty) set of *programmed rules* or *productions* of the form

$$(r : \alpha \rightarrow \beta, \sigma(r), \varphi(r))$$

where $r : \alpha \rightarrow \beta$ is a production, r is a label assigned in a one-to-one way, α, β are words such that $\alpha \in (V_T \cup V_N)^* V_N (V_T \cup V_N)^*$, $\beta \in (V_T \cup V_N)^*$, and the two last components, denominated respectively as the **success** and **failure fields**, are mappings of the form $\sigma, \varphi : Lab(P) \rightarrow 2^{Lab(P)}$, where $Lab(P)$ is the set of all labels of productions of P . When at least one failure field in the rules is non-empty, then we say that \mathcal{G} has **appearance checking** features, denoted as *ac*.

Definition 6 ([5],[6]). Given a programmed grammar $\mathcal{G} = (V_N, V_T, S, P)$ and pairs $(x, r), (y, r') \in (V_T \cup V_N)^* \times Lab(P)$, where r is the label of some production $(r : \alpha \rightarrow \beta, \sigma(r), \varphi(r)) \in P$, we say that (x, r) **directly derives** (y, r') , denoted as $(x, r) \Rightarrow (y, r')$, if and only if one of the following conditions hold:

- $x = w_1 \alpha w_2, y = w_1 \beta w_2, w_1, w_2 \in (V_T \cup V_N)^*$ and $r' \in \sigma(r)$, that is, x is effectively transformed into y by the application of the (core) production $\alpha \rightarrow \beta$ and some label r' is chosen from the success field to indicate the next production to be used,
- or
- $x = y, \alpha$ does not occur in x and $r' \in \varphi(r)$, which means that in the absence of α , production $\alpha \rightarrow \beta$ is “applied” in the appearance checking mode, leaving x unmodified but allowing further derivations, as indicated by labels from the failure field.

We denote by \Rightarrow^* the reflexive and transitive closure of the relation \Rightarrow . The language generated by \mathcal{G} is defined as

$$L(\mathcal{G}) = \{x : x \in V_T^*, (S, r) \Rightarrow^* (x, r'), r, r' \in Lab(P)\}.$$

Definition 7 ([5]). Given a programmed grammar $\mathcal{G} = (V_N, V_T, S, P)$ we say that it is *context-free*, denoted CF, if and only if all the productions in P are of the form $A \rightarrow w$ with $A \in V_N, w \in (V_N \cup V_T)^*$. Furthermore we say that \mathcal{G} is λ -free, denoted CF- λ , when $w \neq \lambda$ in the above condition.

3 The Problem

Firstly, we establish an association between deduction trees and languages in the following way.

²Although we could have used the simpler, perhaps more elegant, definition given in [6] this one was more suitable to be implemented in a programming language.

Definition 8. We consider a sequent $\Gamma \rightsquigarrow \Delta$ where $\Gamma = \mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_n$ and $\Delta = \mathcal{B}_1, \mathcal{B}_2, \dots, \mathcal{B}_m$, as usual. We define the property $\ell(\mathcal{A})$ as “proposition \mathcal{A} is a letter” and, abusing notation, the functions

$$\chi(\mathcal{A}) = \begin{cases} A', & \ell(\mathcal{A}) \text{ and } \mathcal{A} = A, \\ \lambda, & \neg \ell(\mathcal{A}) \end{cases}$$

where A' stands for a symbol and

$$\chi(\Gamma \rightsquigarrow \Delta) = \begin{cases} \chi(\mathcal{A}_i) \triangleright \chi(\mathcal{B}_j), & \ell(\mathcal{A}_i), \ell(\mathcal{B}_j), \\ & \mathcal{A}_i = \mathcal{B}_j, \\ & 1 \leq i \leq n, \\ & 1 \leq j \leq m, \\ \chi(\mathcal{A}_1) \cdots \chi(\mathcal{A}_n) \triangleright \chi(\mathcal{B}_1) \cdots \chi(\mathcal{B}_m), & \Gamma \cap \Delta = \emptyset \end{cases}$$

Symbol \triangleright is a marker. Here, the first case corresponds to sequents labeled with an axiom instance while the second case corresponds to arbitrary sequents.

Example 1. $\chi(\neg r, q \rightsquigarrow q) = q \triangleright q, \quad \chi(p, q \rightsquigarrow r) = p q \triangleright r.$

Definition 9. Given any arbitrary proposition \mathcal{A} , let T be a deduction tree for \mathcal{A} . We define the language

$$L_T = \{\chi(\Gamma \rightsquigarrow \Delta) : \Gamma \rightsquigarrow \Delta \text{ is the label of a leaf of } T\}$$

Secondly, we structure logical propositions as follows.

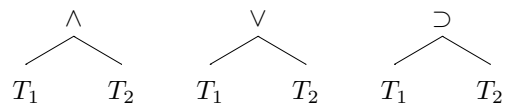
Definition 10. A **propositional tree** (with root at the top) is defined inductively as follows

1. Every one-node tree labeled with any letter A is a propositional tree.
2. For any propositional tree T , the tree



is also a propositional tree.

3. For any two propositional trees T_1 and T_2 , the following trees



are also propositional trees.

Furthermore, for any arbitrary proposition \mathcal{A} we denote by $\lambda(\mathcal{A})$ the root of its propositional tree.

Thus a propositional tree is a binary tree such that the internal nodes are labeled with connectives and the leafs are labeled with letters. Furthermore, we will adopt the following convention: each occurrence of the connectives will be distinguished from the others by means of indexes or by their position (cf. [1]) in the tree.

Note 1. In the rest of the examples to come, unless stated otherwise, we will consider a particular proposition $\ulcorner \mathcal{A} \urcorner = (p \supset q) \supset (\neg q \supset \neg p)$.

Example 2. Consider proposition $\ulcorner \mathcal{A} \urcorner$ and its respective propositional tree

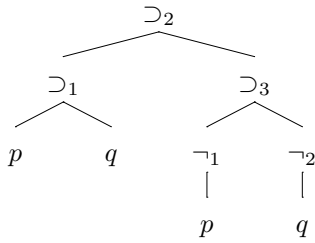


Figure 1: Propositional tree for $\ulcorner \mathcal{A} \urcorner$ with $\lambda(\ulcorner \mathcal{A} \urcorner) = \supset_2$.

The idea behind definition 10 is that the propositional tree provides us with *pointers* to every subproposition of any arbitrary proposition. Clearly the election of any G inference rule to be applied depends only on the main connective of a proposition, so we can “exclude” at this stage the so-called side formulae. When an inference rule has been chosen, we can introduce in the premises only the main connectives of the side formulae, which again will determine the election of the next inference rules to be applied, and so on. For this *succinct* approach to work, we need to remember where the nodes of the propositional tree point to.

Example 3. Consider the partial deduction trees for $\ulcorner \mathcal{A} \urcorner$ presented in figure 2.

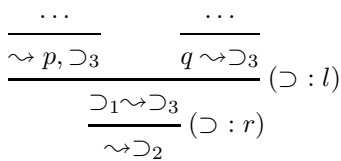
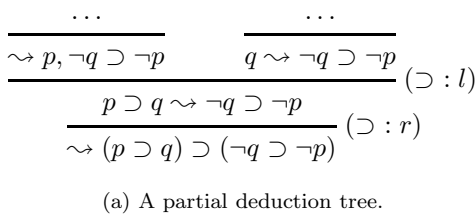


Figure 2: Deduction trees for $\ulcorner \mathcal{A} \urcorner$.

In figure 2(b), we start with the sequent $\sim \supset_2$, which only contains the major connective \supset_2 , and we omit the side formulae $(p \supset q)$ and $(\neg q \supset \neg p)$, in contrast to the tree of figure 2(a). When \supset_2 is to be replaced, only the connectives \supset_1, \supset_3 are introduced in the respective premise, keeping aside the side formulae $p, q, \neg q$ and $\neg p$. Observe that this procedure is regulated by the propositional tree: we decompose it in a *node-by-node* fashion, involving just the major connectives of the (sub)propositions.

Note 2. In the rest of the article we will use the terms *proposition* and *propositional tree* indistinctly, since the later is just a structured representation of the former.

Definition 11 (Propositions as Symbols). Let \mathcal{A} be an arbitrary proposition with $X = \lambda(\mathcal{A})$. We define the symbols

$$[X \cdot Y], \text{ with } Y \in \{L, R\},$$

where L and R stand for “left” or “right”, respectively, and indicate the side of the related proposition within a sequent – we must consider both cases.

All the propositions (their roots precisely) in a sequent can be coded into a word by means of symbols as formulated in definition 11. So, considering the tree in figure 2(b), we could associate to those propositions the words

$$\begin{aligned} w_1 &= [\supset_2 \cdot R] & w_2 &= [\supset_1 \cdot L][\supset_3 \cdot R] \\ w_3 &= [p \cdot R][\supset_3 \cdot R] & w_4 &= [q \cdot L][\supset_3 \cdot R] \end{aligned}$$

Note 3. Consider now the words

$$w'_3 = [\supset_3 \cdot R][p \cdot R] \quad w'_4 = [\supset_3 \cdot R][q \cdot L]$$

which are clearly different from w_3 and w_4 , respectively. For our purposes, however, w'_3 could be also a valid word, since $\{p, \supset_3\} = \{\supset_3, p\}$ and we are considering sequents as pairs of sets. In the case of w'_4 , the position of its symbols does not disrupt the semantic of the sequent $q \sim \supset_3$ ($[\supset_3 \cdot R]$ is actually on the left-hand side fo the word), since this information is coded into the symbols themselves. Thus, we will say that $w_3 \equiv w'_3$ and $w_4 \equiv w'_4$.

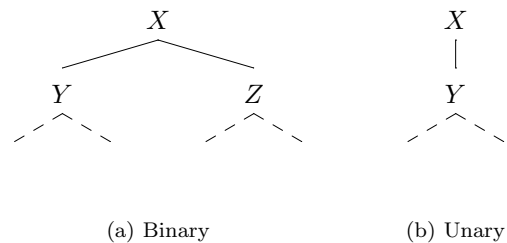


Figure 3: Some structures of logical propositions in the sense of definition 10.

Definition 12 (Rewriting of Propositions). Keeping in mind definition 10, lets consider figure 3.

1. Given a proposition \mathcal{A} as shown in fig. 3(a), we construct the context-free productions:

$$\begin{array}{ll} [X \cdot S_1] \rightarrow [Y \cdot S_2][Z \cdot S_3] & \text{where} \\ [X \cdot S_1] \rightarrow [Y \cdot S_2] & S_i \in \{L, R\}, \\ [X \cdot S_1] \rightarrow [Z \cdot S_3] & 1 \leq i \leq 3. \end{array}$$

Here, Y and Z denote the roots of the subpropositions (side formulae) that are to be introduced into the premises after an application of a G inference rule. This rule will be determined by connective X and the position of \mathcal{A} in the sequent, denoted by S_1 , while the positions S_2, S_3 of the side formulae will be determined accordingly to the rule in question.

2. Given a proposition \mathcal{A} as in fig. 3(b), we construct the context-free productions:

$$[X \cdot S_1] \rightarrow [Y \cdot S_2], \quad S_i \in \{L, R\}, 1 \leq i \leq 2.$$

Productions in definition 12 will be further combined with other required components, i.e. labels, success and failure fields (see section 4), in order to build programmed productions.

Example 4. With respect to proposition $\lceil \mathcal{A} \rceil$, we build some preliminary³ programmed productions which have as foundation the productions introduced in definition 12.

$$\begin{array}{l} (r_1 : [\supset_2 \cdot R] \rightarrow [\supset_1 \cdot L][\supset_3 \cdot R], \{r_2, r_3\}, \emptyset) \\ (r_2 : [\supset_1 \cdot L] \rightarrow [p \cdot R], \{r\}, \emptyset) \\ (r_3 : [\supset_1 \cdot L] \rightarrow [q \cdot L], \{r'\}, \emptyset) \end{array}$$

Figure 4(b) shows the derivations obtained with these productions. In order to appreciate the flavor of these derivations, and for the sake of comfort, we reproduce also the partial deduction tree given in figure 2(b).

$$\frac{\frac{\dots}{\rightsquigarrow p, \supset_3} \quad \frac{\dots}{q \rightsquigarrow \supset_3}}{\supset_1 \rightsquigarrow \supset_3} (\supset : l) \\ \frac{\supset_1 \rightsquigarrow \supset_3}{\rightsquigarrow \supset_2} (\supset : r)$$

(a) Yet again, a partial deduction tree for $\lceil \mathcal{A} \rceil$.

$$\begin{array}{ccc} ([p \cdot R][\supset_3 \cdot R], r) & & ([q \cdot L][\supset_3 \cdot R], r') \\ \uparrow & & \uparrow \\ ([\supset_1 \cdot L][\supset_3 \cdot R], r_2) & & ([\supset_1 \cdot L][\supset_3 \cdot R], r_3) \\ \swarrow & & \searrow \\ & ([\supset_2 \cdot R], r_1) & \end{array}$$

(b) Derivations for $\lceil \mathcal{A} \rceil$.

Figure 4: Derivations vs Reductions.

³The programmed productions of this example differ a little from those defined in section 4, but they illustrate the intended derivation mechanism.

We observe that the success field of production r_1 has two labels, namely r_2 and r_3 , which allow further derivations to be made in such a way that branching, as it is induced by G inference rules, is preserved.

Thirdly, we propose the next (recursive) procedure that receives a sequent as input and carries out reductions on it accordingly to system G . The output consists of a set of words.

Algorithm 1. Take as input the sequent $\Gamma \rightsquigarrow \Delta$ and proceed as follows.

1. **Do $\ell(\mathcal{A})$, $\ell(\mathcal{B})$ and $(\mathcal{A} = \mathcal{B})$ hold for some $\mathcal{A} \in \Gamma$, $\mathcal{B} \in \Delta$?** If true, this is an instance of the axiom schema, since we have a letter appearing at both sides of the sequent in question, return the word $\chi(\Gamma \rightsquigarrow \Delta)$ (def. 8), otherwise go to 2.
2. **Let Ω be the set $\{\mathcal{A} : \neg \ell(\mathcal{A}), \mathcal{A} \in \Gamma \cup \Delta\}$ of all propositions in $\Gamma \rightsquigarrow \Delta$ that are not letters. Does condition $\Omega \neq \emptyset$ hold ?** If this is true, apply the algorithm to each sequent $\Lambda \rightsquigarrow \Theta$, where $\Gamma \rightsquigarrow \Delta \xrightarrow{i} \Lambda \rightsquigarrow \Theta$ and i is a G inference rule pointed out by each proposition $\mathcal{A} \in \Omega$, otherwise go to 3.
3. **Conditions $\ell(\mathcal{A})$, for every $\mathcal{A} \in \Gamma \rightsquigarrow \Delta$, and $\Gamma \cap \Delta = \emptyset$ must hold at this stage.** This means that sequent $\Gamma \rightsquigarrow \Delta$ is composed exclusively of letters that represent a falsifying truth assignment. Return $\chi(\Gamma \rightsquigarrow \Delta)$.

4 The Characterization

Definition 13. For a given arbitrary proposition \mathcal{A} , structured as in def. 10, we define its set of letters $\theta_0(\mathcal{A}) = \{A_1, A_2, \dots, A_{n_0}\}$, as well as the sets of occurrences of each kind of connective

$$\begin{array}{ll} \theta_1(\mathcal{A}) = \{\wedge_1, \wedge_2, \dots, \wedge_{n_1}\}, & \theta_2(\mathcal{A}) = \{\vee_1, \vee_2, \dots, \vee_{n_2}\}, \\ \theta_3(\mathcal{A}) = \{\supset_1, \supset_2, \dots, \supset_{n_3}\}, & \theta_4(\mathcal{A}) = \{\neg_1, \neg_2, \dots, \neg_{n_4}\}, \end{array}$$

where $1 \leq n_0$, $0 \leq n_i$, for $1 \leq i \leq 4$.

Definition 14. For a given arbitrary proposition \mathcal{A} the following alphabet is defined

$$\mathcal{U}(\mathcal{A}) = \{[X \cdot Y] : X \in \bigcup_{i=0}^4 \theta_i(\mathcal{A}), Y \in \{L, R\}\}.$$

Example 5.

$$\begin{array}{ll} \theta_0(\lceil \mathcal{A} \rceil) = \{p, q\} & \theta_1(\lceil \mathcal{A} \rceil) = \theta_2(\lceil \mathcal{A} \rceil) = \emptyset \\ \theta_3(\lceil \mathcal{A} \rceil) = \{\supset_1, \supset_2, \supset_3\} & \theta_4(\lceil \mathcal{A} \rceil) = \{\neg_1, \neg_2\} \end{array}$$

$$\begin{array}{l} \mathcal{U}(\lceil \mathcal{A} \rceil) = \{[p \cdot Y], [q \cdot Y], [\supset_1 \cdot Y], [\supset_2 \cdot Y] \\ [\supset_3 \cdot Y], [\neg_1 \cdot Y], [\neg_2 \cdot Y] : Y \in \{L, R\}\} \end{array}$$

Definition 15. For $X \in \{\wedge, \vee, \supset, \neg\}$ the symbols $[X]$ will serve to indicate the appearance in an arbitrary sequent (regardless of the position) of a proposition whose major connective is X . Similarly, the symbols $[A_L]$, $[A_R]$ will indicate, respectively, the appearance of a letter at left or right of an arbitrary sequent. We denote by \mathcal{V} the set of all these symbols. Notice that the number of propositions and letters in an arbitrary sequent will determine the number of such symbols. As an example, we will produce a word with symbols $[A_R]$, $[\supset]$ for the sequent $\rightsquigarrow p, \supset_3$.

Definition 16. Given an arbitrary proposition \mathcal{A} , let the function $\rho : \left(\bigcup_{i=0}^4 \theta_i(\mathcal{A})\right) \times \{L, R\} \rightarrow \mathcal{V}$ be defined as

$$\rho(X, Y) = \begin{cases} [A_L], & X \in \theta_0(\mathcal{A}) \text{ and } Y = L, \\ [A_R], & X \in \theta_0(\mathcal{A}) \text{ and } Y = R, \\ [\wedge], & X \in \theta_1(\mathcal{A}), \\ [\vee], & X \in \theta_2(\mathcal{A}), \\ [\supset], & X \in \theta_3(\mathcal{A}), \\ [\neg], & X \in \theta_4(\mathcal{A}). \end{cases}$$

Definition 17 (Characterizing Grammar). For a given arbitrary proposition \mathcal{A} , the programmed grammar $\mathcal{G} = (V_N, V_T, S, P)$, of type CF, is constructed as follows. Let V_N and V_T be the alphabets defined as

$$V_N = \{S, \$, [\omega], [\omega_*]\} \cup \mathcal{U}(\mathcal{A}) \cup \mathcal{V}$$

$$V_T = \{A : A \in \theta_0(\mathcal{A})\} \cup \{\triangleright\}$$

We define P as the set which consists of all programmed productions formulated in definitions 18 – 25.

Definition 18 (Control Productions). Let \mathcal{A} be the proposition consider in def. 17. We define

$$(r_0 : S \rightarrow [\omega]\rho(\wedge(\mathcal{A}), R) \$[\wedge(\mathcal{A}) \cdot R], \{r_1\}, \emptyset) \quad (1)$$

This production starts the derivation process introducing a word where:

- symbol $[\omega]$ indicates that the sequent remains unmodified (the contrary will be indicated by means of the symbol $[\omega_*]$),
- $\$$ is just a marker which, together with the previous symbols, belongs to the *control space* of the word (i.e. a prefix where all the decision steps in algorithm 1 are performed),
- $[\wedge(\mathcal{A}) \cdot R]$ represents actually the root of \mathcal{A} , and it belongs to the *work space* of the word (i.e. a suffix in which all the calculations in algorithm 1 take place).

$$\begin{aligned} (r_1 : [A_L] &\rightarrow [A_L], \{r_2\}, \{r_3\}) \\ (r_2 : [A_R] &\rightarrow [A_R], \{A_{1_1}\}, \{r_3\}) \end{aligned} \quad (2)$$

Productions r_1 and r_2 test the presence of letters at both sides of the sequent (*step 1 of the algorithm*). With a positive answer, rule r_2 fires the productions needed to apply the axiom (starting with the one labeled by A_{1_1} , where $A_1 \in \theta_0(\mathcal{A})$), otherwise it fires the productions that will check for connectives.

$$\begin{aligned} (r_3 : [\wedge] &\rightarrow [\wedge], \{r_4\}, \{r_5\}) \\ (r_4 : [\omega] &\rightarrow [\omega_*], \{r_5\} \cup L_\wedge, \{r_5\} \cup L_\wedge) \\ (r_5 : [\vee] &\rightarrow [\vee], \{r_6\}, \{r_7\}) \\ (r_6 : [\omega] &\rightarrow [\omega_*], \{r_7\} \cup L_\vee, \{r_7\} \cup L_\vee) \\ (r_7 : [\supset] &\rightarrow [\supset], \{r_8\}, \{r_9\}) \\ (r_8 : [\omega] &\rightarrow [\omega_*], \{r_9\} \cup L_\supset, \{r_9\} \cup L_\supset) \\ (r_9 : [\neg] &\rightarrow [\neg], \{r_{10}\}, \{r_\omega\}) \\ (r_{10} : [\omega] &\rightarrow [\omega_*], L_\neg, L_\neg) \end{aligned} \quad (3)$$

Productions r_3, r_5, r_7, r_9 test the presence of connectives. On success, a jump to the next production, r_4, r_6, r_8 or r_{10} , will take place, to indicate that a change in the sequent will be produced, and the corresponding productions (here we denote by $L_X, X \in \{\wedge, \vee, \supset, \neg\}$, the sets which contain labels for every connective of the proposition in question) will be fired to apply the related inference rules of G . Observe that all this productions are *chained*, i.e. all of them are to be used, even in the case of failure. In this way the presence of all types of connectives is verified (*step 2*).

$$\begin{aligned} (r_\omega : [\omega] &\rightarrow [\omega], \{A'_{1_1}\}, \emptyset) \\ (r_{\omega_*} : [\omega_*] &\rightarrow [\omega], \{r_1\}, \emptyset) \\ (r_f : \$ &\rightarrow \triangleright, \{r_f\}, \emptyset) \end{aligned} \quad (4)$$

Production r_ω determines if the sequent has suffered a change since the last iteration, if this is not the case, this means that no G inference rule nor the axiom has been applied and the word obtained so far represents a falsifying truth assignment (*step 3*), so it fires production A'_{1_1} . Production r_{ω_*} resets the state of the sequent and launches again the whole process. Production r_f is applied just at the end of the derivation.

We claim that productions in equations 1 to 4 accurately perform the decision steps described in algorithm 1. (written in bold font.)

Example 6. Control productions for proposition $\lceil \mathcal{A} \rceil$

$$\begin{aligned} (r_0 : S &\rightarrow [\omega][\supset] \$[\supset_2 \cdot R], \{r_1\}, \emptyset) \\ (r_1 : [A_L] &\rightarrow [A_L], \{r_2\}, \{r_3\}) \\ (r_2 : [A_R] &\rightarrow [A_R], \{q_1\}, \{r_3\}) \\ (r_3 : [\supset] &\rightarrow [\supset], \{r_4\}, \{r_5\}) \\ (r_4 : [\omega] &\rightarrow [\omega_*], \sigma(r_4), \varphi(r_4)) \\ (r_5 : [\neg] &\rightarrow [\neg], \{r_6\}, \{r_\omega\}) \\ (r_6 : [\omega] &\rightarrow [\omega_*], \sigma(r_6), \varphi(r_6)) \\ (r_\omega : [\omega] &\rightarrow [\omega], \{q'_1\}, \emptyset) \\ (r_{\omega_*} : [\omega_*] &\rightarrow [\omega], \{r_1\}, \emptyset) \\ (r_f : \$ &\rightarrow \triangleright, \{r_f\}, \emptyset) \end{aligned}$$

where $\sigma(r_4) = \varphi(r_4) = \{r_5\} \cup \{\supset_{i_1}, \supset_{i_3}, \supset_{i_5} : i = 1, 2, 3\}$
and $\sigma(r_6) = \varphi(r_6) = \{\neg_{i_1}, \neg_{i_3} : i = 1, 2\}$

Definition 19 (Productions for the axiom schema). Consider again the proposition \mathcal{A} stated in def. 17. The productions below, where A_i , $1 \leq i \leq n_0$, is a letter from $\theta_0(\mathcal{A})$, emulate the application of the axiom schema,

$$\begin{aligned} (A_{i_1} : [A_i \cdot L] \rightarrow [A_i \cdot L], \{A_{i_2}\}, \delta(A_i)) \\ (A_{i_2} : [A_i \cdot R] \rightarrow A_i, \{A_{i_3}\}, \delta(A_i)) \\ (A_{i_3} : \$ \rightarrow A_i \$, \{e_1\}, \emptyset) \end{aligned}$$

$$\text{where } \delta(A_i) = \begin{cases} \{A_{(i+1)_1}\}, & 1 \leq i < n_0, \\ \{r_3\}, & i = n_0. \end{cases}$$

Productions A_{i_1} and A_{i_2} determine the presence of A_i at left and right, respectively, in the sequent. Success in both operations implies the replacement of the right occurrence of A_i by its corresponding terminal symbol and a copy of this terminal will be introduced at the left of symbol $\$$ (by A_{i_3}). After that, there is a jump to production e_1 in order to complete the ‘‘application’’ of χ (cf. def. 25). If A_i is absent in at least one side of the sequent then letter A_{i+1} will be tested, and so on, until no more letters remain to try. In this case the derivation flow jumps to production r_3 , i.e. the test of connectives.

Example 7. For $\sqsupset\mathcal{A}$ we have

$$\begin{aligned} (q_1 : [q \cdot L] \rightarrow [q \cdot L], \{q_2\}, \{p_1\}) \\ (q_2 : [q \cdot R] \rightarrow q, \{q_3\}, \{p_1\}) \\ (q_3 : \$ \rightarrow q \$, \{e_1\}, \emptyset) \\ (p_1 : [p \cdot L] \rightarrow [p \cdot L], \{p_2\}, \{r_3\}) \\ (p_2 : [p \cdot R] \rightarrow p, \{p_3\}, \{r_3\}) \\ (p_3 : \$ \rightarrow p \$, \{e_1\}, \emptyset) \end{aligned}$$

The next groups of productions characterize G inference rules, their core rules follows from definitions 3 and 12. They rewrite the symbols associated with propositions (of the form $[X \cdot S]$) and those associated with their types ($[\wedge]$, $[\vee]$, \dots). Notice that they must be recursively applied to every subproposition of the proposition we want to treat.

Definition 20 (Productions for \wedge). Let \mathcal{A} be a proposition of form described in fig 3(a). Let $X \in \theta_1(\mathcal{A})$. We define the productions

$$\begin{aligned} (\wedge : l) \\ \hline (X_1 : [X \cdot L] \rightarrow [Y \cdot L][Z \cdot L], \{X_2\}, \emptyset) \\ (X_2 : [\wedge] \rightarrow \rho(Y, L)\rho(Z, L), \{r_{\omega_*}\}, \emptyset) \\ \\ (\wedge : r) \\ \hline (X_3 : [X \cdot R] \rightarrow [Y \cdot R], \{X_4\}, \emptyset) \\ (X_4 : [\wedge] \rightarrow \rho(Y, R), \{r_{\omega_*}\}, \emptyset) \\ (X_5 : [X \cdot R] \rightarrow [Z \cdot R], \{X_6\}, \emptyset) \\ (X_6 : [\wedge] \rightarrow \rho(Z, R), \{r_{\omega_*}\}, \emptyset) \end{aligned}$$

Definition 21 (Productions for \vee). Consider a proposition \mathcal{A} as in fig 3(a). Given $X \in \theta_2(\mathcal{A})$, we define

$$\begin{aligned} (\vee : l) \\ \hline (X_1 : [X \cdot L] \rightarrow [Y \cdot L], \{X_2\}, \emptyset) \\ (X_2 : [\vee] \rightarrow \rho(Y, L), \{r_{\omega_*}\}, \emptyset) \\ (X_3 : [X \cdot L] \rightarrow [Z \cdot L], \{X_4\}, \emptyset) \\ (X_4 : [\vee] \rightarrow \rho(Z, L), \{r_{\omega_*}\}, \emptyset) \\ \\ (\vee : r) \\ \hline (X_5 : [X \cdot R] \rightarrow [Y \cdot R][Z \cdot R], \{X_6\}, \emptyset) \\ (X_6 : [\vee] \rightarrow \rho(Y, R)\rho(Z, R), \{r_{\omega_*}\}, \emptyset) \end{aligned}$$

Definition 22 (Productions for \supset). For a given proposition \mathcal{A} as in fig 3(a) let $X \in \theta_3(\mathcal{A})$. We build the productions

$$\begin{aligned} (\supset : l) \\ \hline (X_1 : [X \cdot L] \rightarrow [Y \cdot R], \{X_2\}, \emptyset) \\ (X_2 : [\supset] \rightarrow \rho(Y, R), \{r_{\omega_*}\}, \emptyset) \\ (X_3 : [X \cdot L] \rightarrow [Z \cdot L], \{X_4\}, \emptyset) \\ (X_4 : [\supset] \rightarrow \rho(Z, L), \{r_{\omega_*}\}, \emptyset) \\ \\ (\supset : r) \\ \hline (X_5 : [X \cdot R] \rightarrow [Y \cdot L][Z \cdot R], \{X_6\}, \emptyset) \\ (X_6 : [\supset] \rightarrow \rho(Y, L)\rho(Z, R), \{r_{\omega_*}\}, \emptyset) \end{aligned}$$

Definition 23 (Productions for \neg). Let \mathcal{A} be a proposition as depicted in fig 3(b). Given $X \in \theta_4(\mathcal{A})$, we have

$$\begin{aligned} (\neg : l) \\ \hline (X_1 : [X \cdot L] \rightarrow [Y \cdot R], \{X_2\}, \emptyset) \\ (X_2 : [\neg] \rightarrow \rho(Y, R), \{r_{\omega_*}\}, \emptyset) \\ \\ (\neg : r) \\ \hline (X_3 : [X \cdot R] \rightarrow [Y \cdot L], \{X_4\}, \emptyset) \\ (X_4 : [\neg] \rightarrow \rho(Y, L), \{r_{\omega_*}\}, \emptyset) \end{aligned}$$

Example 8. Productions which correspond to inference rules for $\sqsupset\mathcal{A}$

$$\begin{aligned} (\supset_{21} : [\supset_2 \cdot L] \rightarrow [\supset_1 \cdot R], \{\supset_{22}\}, \emptyset) \\ (\supset_{22} : [\supset] \rightarrow [\supset], \{r_{\omega_*}\}, \emptyset) \\ (\supset_{23} : [\supset_2 \cdot L] \rightarrow [\supset_3 \cdot L], \{\supset_{24}\}, \emptyset) \\ (\supset_{24} : [\supset] \rightarrow [\supset], \{r_{\omega_*}\}, \emptyset) \\ (\supset_{25} : [\supset_2 \cdot R] \rightarrow [\supset_1 \cdot L][\supset_3 \cdot R], \{\supset_{26}\}, \emptyset) \\ (\supset_{26} : [\supset] \rightarrow [\supset][\supset], \{r_{\omega_*}\}, \emptyset) \end{aligned}$$

$$\begin{aligned}
 (\supset_{11} : [\supset_1 \cdot L] &\rightarrow [p \cdot R], \{\supset_{12}\}, \emptyset) \\
 (\supset_{12} : [\supset] &\rightarrow [A_R], \{r_{\omega_*}\}, \emptyset) \\
 (\supset_{13} : [\supset_1 \cdot L] &\rightarrow [q \cdot L], \{\supset_{14}\}, \emptyset) \\
 (\supset_{14} : [\supset] &\rightarrow [A_L], \{r_{\omega_*}\}, \emptyset) \\
 (\supset_{15} : [\supset_1 \cdot R] &\rightarrow [p \cdot L][q \cdot R], \{\supset_{16}\}, \emptyset) \\
 (\supset_{16} : [\supset] &\rightarrow [A_L][A_R], \{r_{\omega_*}\}, \emptyset)
 \end{aligned}$$

$$\begin{aligned}
 (\supset_{31} : [\supset_3 \cdot L] &\rightarrow [\neg_1 \cdot R], \{\supset_{32}\}, \emptyset) \\
 (\supset_{32} : [\supset] &\rightarrow [\neg], \{r_{\omega_*}\}, \emptyset) \\
 (\supset_{33} : [\supset_3 \cdot L] &\rightarrow [\neg_2 \cdot L], \{\supset_{34}\}, \emptyset) \\
 (\supset_{34} : [\supset] &\rightarrow [\neg], \{r_{\omega_*}\}, \emptyset) \\
 (\supset_{35} : [\supset_3 \cdot R] &\rightarrow [\neg_1 \cdot L][\neg_2 \cdot R], \{\supset_{36}\}, \emptyset) \\
 (\supset_{36} : [\supset] &\rightarrow [\neg][\neg], \{r_{\omega_*}\}, \emptyset)
 \end{aligned}$$

$$\begin{aligned}
 (\neg_{11} : [\neg_1 \cdot L] &\rightarrow [q \cdot R], \{\neg_{12}\}, \emptyset) \\
 (\neg_{12} : [\neg] &\rightarrow [A_R], \{r_{\omega_*}\}, \emptyset) \\
 (\neg_{13} : [\neg_1 \cdot R] &\rightarrow [q \cdot L], \{\neg_{14}\}, \emptyset) \\
 (\neg_{14} : [\neg] &\rightarrow [A_L], \{r_{\omega_*}\}, \emptyset)
 \end{aligned}$$

$$\begin{aligned}
 (\neg_{21} : [\neg_2 \cdot L] &\rightarrow [p \cdot R], \{\neg_{22}\}, \emptyset) \\
 (\neg_{22} : [\neg] &\rightarrow [A_R], \{r_{\omega_*}\}, \emptyset) \\
 (\neg_{23} : [\neg_2 \cdot R] &\rightarrow [p \cdot L], \{\neg_{24}\}, \emptyset) \\
 (\neg_{24} : [\neg] &\rightarrow [A_L], \{r_{\omega_*}\}, \emptyset)
 \end{aligned}$$

Next productions will be applied when the sequent has exclusively letters as members. For each letter of the original proposition we introduce at most one terminal symbol at left or right of symbol \$, accordingly to the side of the sequent the letter appears in, if at all. At the end, a jump to production e_1 (cf. def. 25) is carried out in such a way that a word is generated accordingly to function χ .

Definition 24 (Productions for falsifying truth assignments). Let \mathcal{A} be an arbitrary proposition. For $A_i \in \theta_0(\mathcal{A})$ we have

$$\begin{aligned}
 (A'_{i_1} : [A_i \cdot L] &\rightarrow [A_i \cdot L], \{A'_{i_2}\}, \{A'_{i_3}\}) \\
 (A'_{i_2} : \$ &\rightarrow A_i \$, \delta(A'_i), \emptyset) \\
 (A'_{i_3} : [A_i \cdot R] &\rightarrow A_i, \delta(A'_i), \delta(A'_i)) \\
 \text{where } \delta(A'_i) &= \begin{cases} \{A'_{(i+1)_1}\}, & 1 \leq i < n_0, \\ \{e_1\}, & i = n_0. \end{cases}
 \end{aligned}$$

Example 9. From last definition we obtain the productions related to $\lceil \mathcal{A} \rceil$

$$\begin{aligned}
 (q'_1 : [q \cdot L] &\rightarrow [q \cdot L], \{q'_2\}, \{q'_3\}) \\
 (q'_2 : \$ &\rightarrow q \$, \{p'_1\}, \emptyset) \\
 (q'_3 : [q \cdot R] &\rightarrow q, \{p'_1\}, \{p'_1\}) \\
 (p'_1 : [p \cdot L] &\rightarrow [p \cdot L], \{p'_2\}, \{p'_3\}) \\
 (p'_2 : \$ &\rightarrow p \$, \{e_1\}, \emptyset) \\
 (p'_3 : [p \cdot R] &\rightarrow p, \{e_1\}, \{e_1\})
 \end{aligned}$$

Finally, the following productions – in collaboration with those of definitions 19 and 24 – will be used to “apply” and χ (def. 8) to the current string.

Definition 25 (Erasing Productions). For a given arbitrary proposition \mathcal{A} , we define the set of symbols

$\mathcal{E} = \{[\omega]\} \cup \mathcal{U}(\mathcal{A}) \cup \mathcal{V}$ with $|\mathcal{E}| = m$. We define for $E_i \in \mathcal{E}$, $1 \leq i \leq m$, the production

$$\begin{aligned}
 (e_i : E_i &\rightarrow \lambda, \{e_i\}, \varphi(e_i)) \\
 \text{where } \varphi(e_i) &= \begin{cases} \{e_{i+1}\}, & 1 \leq i < m, \\ \{r_f\}, & i = m. \end{cases}
 \end{aligned}$$

Example 10. For $\lceil \mathcal{A} \rceil$, we have

$$\begin{aligned}
 (e_1 : [q \cdot L] &\rightarrow \lambda, \{e_1\}, \{e_2\}) \\
 (e_2 : [q \cdot R] &\rightarrow \lambda, \{e_2\}, \{e_3\}) \\
 (e_3 : [p \cdot L] &\rightarrow \lambda, \{e_3\}, \{e_4\}) \\
 (e_4 : [p \cdot R] &\rightarrow \lambda, \{e_4\}, \{e_5\}) \\
 (e_5 : [\supset_2 \cdot L] &\rightarrow \lambda, \{e_5\}, \{e_6\}) \\
 (e_6 : [\supset_2 \cdot R] &\rightarrow \lambda, \{e_6\}, \{e_7\}) \\
 (e_7 : [\supset_1 \cdot L] &\rightarrow \lambda, \{e_7\}, \{e_8\}) \\
 (e_8 : [\supset_1 \cdot R] &\rightarrow \lambda, \{e_8\}, \{e_9\}) \\
 (e_9 : [\supset_3 \cdot L] &\rightarrow \lambda, \{e_9\}, \{e_{10}\}) \\
 (e_{10} : [\supset_3 \cdot R] &\rightarrow \lambda, \{e_{10}\}, \{e_{11}\}) \\
 (e_{11} : [\neg_1 \cdot L] &\rightarrow \lambda, \{e_{11}\}, \{e_{12}\}) \\
 (e_{12} : [\neg_1 \cdot R] &\rightarrow \lambda, \{e_{12}\}, \{e_{13}\}) \\
 (e_{13} : [\neg_2 \cdot L] &\rightarrow \lambda, \{e_{13}\}, \{e_{14}\}) \\
 (e_{14} : [\neg_2 \cdot R] &\rightarrow \lambda, \{e_{14}\}, \{e_{15}\}) \\
 (e_{15} : [A_L] &\rightarrow \lambda, \{e_{15}\}, \{e_{16}\}) \\
 (e_{16} : [A_R] &\rightarrow \lambda, \{e_{16}\}, \{e_{17}\}) \\
 (e_{17} : [\supset] &\rightarrow \lambda, \{e_{17}\}, \{e_{18}\}) \\
 (e_{18} : [\neg] &\rightarrow \lambda, \{e_{18}\}, \{e_{19}\}) \\
 (e_{19} : [\omega] &\rightarrow \lambda, \{e_{19}\}, \{r_f\})
 \end{aligned}$$

Having defined our grammar, we present the next result.

Theorem 1. Given an arbitrary proposition \mathcal{A} let T be a deduction tree for \mathcal{A} . Then a programmed grammar \mathcal{G} can be constructed, in the sense of definition 17, such that $L(\mathcal{G}) = L_T$.

Proof. The behavior of productions in definitions 18, 19, 24 and 25 has already been explained. We examine the more interesting case of the inference rules.

Consider a successful application of productions r_3, r_4 (cf. def. 18, eq. 3) and the sequent $\Gamma \rightsquigarrow \Delta$ where

$$\Gamma = \mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_n, \quad \Delta = \mathcal{B}_1, \mathcal{B}_2, \dots, \mathcal{B}_m.$$

Then the word $[\omega_*]\alpha_1\beta_1 \$ \alpha_2\beta_2$ can be associated to that sequent such that $\alpha_1, \beta_1 \in \mathcal{V}^*$, $\alpha_2, \beta_2 \in \mathcal{U}(\mathcal{A})^*$. Without loss of generality (cf. note 3) we can assume that

$$\begin{aligned}
 \alpha_1 &= \rho(\lambda(\mathcal{A}_1), L) \cdots \rho(\lambda(\mathcal{A}_n), L) \\
 \beta_1 &= \rho(\lambda(\mathcal{B}_1), R) \cdots \rho(\lambda(\mathcal{B}_m), R)
 \end{aligned}$$

and

$$\begin{aligned}
 \alpha_2 &= [\lambda(\mathcal{A}_1) \cdot L] \cdots [\lambda(\mathcal{A}_n) \cdot L] \\
 \beta_2 &= [\lambda(\mathcal{B}_1) \cdot R] \cdots [\lambda(\mathcal{B}_m) \cdot R]
 \end{aligned}$$

Then it must be proved, for single-premise rules, that

$$([\omega_\star]\alpha_1\beta_1 \$ \alpha_2\beta_2, r) \stackrel{*}{\Rightarrow} ([\omega_\star]\alpha'_1\beta'_1 \$ \alpha'_2\beta'_2, r')$$

$$\text{iff} \quad \frac{\Lambda \rightsquigarrow \Theta}{\Gamma \rightsquigarrow \Delta} \iota$$

or either, for two-premises rules,

$$([\omega_\star]\alpha_1\beta_1 \$ \alpha_2\beta_2, r) \stackrel{*}{\Rightarrow} ([\omega_\star]\alpha'_1\beta'_1 \$ \alpha'_2\beta'_2, r'')$$

$$([\omega_\star]\alpha_1\beta_1 \$ \alpha_2\beta_2, r') \stackrel{*}{\Rightarrow} ([\omega_\star]\alpha''_1\beta''_1 \$ \alpha''_2\beta''_2, r'')$$

$$\text{iff} \quad \frac{\Lambda \rightsquigarrow \Theta \quad \Lambda' \rightsquigarrow \Theta'}{\Gamma \rightsquigarrow \Delta} \iota$$

where ι is some inference rule in G , $[\omega_\star]\alpha'_1\beta'_1 \$ \alpha'_2\beta'_2$, $[\omega_\star]\alpha''_1\beta''_1 \$ \alpha''_2\beta''_2$ are words associated to $\Lambda \rightsquigarrow \Theta$ and $\Lambda' \rightsquigarrow \Theta'$, respectively, and r, r', r'' are some adequate labels. We proceed as follows.

Case 1 (for $(\wedge : l)$).

Take a proposition $\mathcal{A}_i = \mathcal{A}'_i \wedge_i \mathcal{A}''_i$. By definition 16 we have $\rho(\lambda(\mathcal{A}_i), L) = \rho([\wedge_i \cdot L]) = [\wedge]$ and we know that $[\lambda(\mathcal{A}_i) \cdot L] = [\wedge_i \cdot L]$. Furthermore, by definition 20, we have the productions

$$(\wedge_{i_1} : [\wedge_i \cdot L] \rightarrow [\lambda(\mathcal{A}'_i) \cdot L][\lambda(\mathcal{A}''_i) \cdot L], \{\wedge_{i_2}\}, \emptyset)$$

$$(\wedge_{i_2} : [\wedge] \rightarrow \rho(\lambda(\mathcal{A}'_i), L)\rho(\lambda(\mathcal{A}''_i), L), \{r_{\omega_\star}\}, \emptyset)$$

Let also

$$w_1 = \rho(\lambda(\mathcal{A}_1), L) \cdots \rho(\lambda(\mathcal{A}_{i-1}), L)$$

$$w_2 = \rho(\lambda(\mathcal{A}_{i+1}), L) \cdots \rho(\lambda(\mathcal{A}_n), L)$$

$$w_3 = [\lambda(\mathcal{A}_1) \cdot L] \cdots [\lambda(\mathcal{A}_{i-1}) \cdot L]$$

$$w_4 = [\lambda(\mathcal{A}_{i+1}) \cdot L] \cdots [\lambda(\mathcal{A}_n) \cdot L]$$

Since $\wedge_{i_1} \in \sigma(r_4)$, then the derivation

$$\begin{aligned} &([\omega_\star]\alpha_1\beta_1 \$ \alpha_2\beta_2, \wedge_{i_1}) \Rightarrow \\ &([\omega_\star]\alpha_1\beta_1 \$ \alpha'_2\beta'_2, \wedge_{i_2}) \Rightarrow \\ &([\omega_\star]\alpha'_1\beta'_1 \$ \alpha'_2\beta'_2, r_{\omega_\star}) \end{aligned}$$

where

$$\alpha_1 = w_1\rho(\lambda(\mathcal{A}_i), L)w_2$$

$$\alpha'_1 = w_1\rho(\lambda(\mathcal{A}'_i), L)\rho(\lambda(\mathcal{A}''_i), L)w_2$$

$$\alpha_2 = w_3[\lambda(\mathcal{A}_i) \cdot L]w_4$$

$$\alpha'_2 = w_3[\lambda(\mathcal{A}'_i) \cdot L][\lambda(\mathcal{A}''_i) \cdot L]w_4$$

and $\beta_1 = \beta'_1$, $\beta_2 = \beta'_2$, $r = \wedge_{i_1}$, $r' = r_{\omega_\star}$, takes place if and only if

$$\frac{\Gamma, \mathcal{A}'_i, \mathcal{A}''_i \rightsquigarrow \Delta}{\Gamma, \mathcal{A}_i = (\mathcal{A}'_i \wedge_i \mathcal{A}''_i) \rightsquigarrow \Delta} (\wedge : l)$$

where $\Gamma = \mathcal{A}_1, \dots, \mathcal{A}_{i-1}, \mathcal{A}_{i+1}, \dots, \mathcal{A}_n$.

Case 2 (for $(\wedge : r)$).

Take some proposition $\mathcal{B}_j = \mathcal{B}'_j \wedge_j \mathcal{B}''_j$. Once more, from def. 16 and 20, we have $\rho(\lambda(\mathcal{B}_j), R) = \rho([\wedge_j \cdot R]) = [\wedge]$, $[\lambda(\mathcal{B}_j) \cdot R] = [\wedge_j \cdot R]$, and the productions

$$(\wedge_{j_3} : [\wedge_j \cdot R] \rightarrow [\lambda(\mathcal{B}'_j) \cdot R], \{\wedge_{j_4}\}, \emptyset)$$

$$(\wedge_{j_4} : [\wedge] \rightarrow \rho(\lambda(\mathcal{B}'_j), R), \{r_{\omega_\star}\}, \emptyset)$$

$$(\wedge_{j_5} : [\wedge_j \cdot R] \rightarrow [\lambda(\mathcal{B}''_j) \cdot R], \{\wedge_{j_6}\}, \emptyset)$$

$$(\wedge_{j_6} : [\wedge] \rightarrow \rho(\lambda(\mathcal{B}''_j), R), \{r_{\omega_\star}\}, \emptyset)$$

Let

$$w_1 = \rho(\lambda(\mathcal{B}_1), R) \cdots \rho(\lambda(\mathcal{B}_{j-1}), R)$$

$$w_2 = \rho(\lambda(\mathcal{B}_{j+1}), R) \cdots \rho(\lambda(\mathcal{B}_m), R)$$

$$w_3 = [\lambda(\mathcal{B}_1) \cdot R] \cdots [\lambda(\mathcal{B}_{j-1}) \cdot R]$$

$$w_4 = [\lambda(\mathcal{B}_{j+1}) \cdot R] \cdots [\lambda(\mathcal{B}_m) \cdot R]$$

We have that $\wedge_{j_3}, \wedge_{j_5} \in \sigma(r_4)$, so the derivations

$$\begin{aligned} &([\omega_\star]\alpha_1\beta_1 \$ \alpha_2\beta_2, \wedge_{j_3}) \Rightarrow \\ &([\omega_\star]\alpha_1\beta_1 \$ \alpha'_2\beta'_2, \wedge_{j_4}) \Rightarrow \\ &([\omega_\star]\alpha'_1\beta'_1 \$ \alpha'_2\beta'_2, r_{\omega_\star}) \end{aligned}$$

and

$$\begin{aligned} &([\omega_\star]\alpha_1\beta_1 \$ \alpha_2\beta_2, \wedge_{j_5}) \Rightarrow \\ &([\omega_\star]\alpha_1\beta_1 \$ \alpha''_2\beta''_2, \wedge_{j_6}) \Rightarrow \\ &([\omega_\star]\alpha''_1\beta''_1 \$ \alpha''_2\beta''_2, r_{\omega_\star}) \end{aligned}$$

where

$$\beta_1 = w_1\rho(\lambda(\mathcal{B}_j), R)w_2$$

$$\beta'_1 = w_1\rho(\lambda(\mathcal{B}'_j), R)w_2$$

$$\beta''_1 = w_1\rho(\lambda(\mathcal{B}''_j), R)w_2$$

$$\beta_2 = w_3[\lambda(\mathcal{B}_j) \cdot R]w_4$$

$$\beta'_2 = w_3[\lambda(\mathcal{B}'_j) \cdot R]w_4$$

$$\beta''_2 = w_3[\lambda(\mathcal{B}''_j) \cdot R]w_4$$

and $\alpha_1 = \alpha'_1 = \alpha''_1$, $\alpha_2 = \alpha'_2 = \alpha''_2$, $r = \wedge_{j_3}$, $r' = \wedge_{j_5}$, $r'' = r_{\omega_\star}$, occur if and only if

$$\frac{\Gamma \rightsquigarrow \mathcal{B}'_j, \Delta \quad \Gamma \rightsquigarrow \mathcal{B}''_j, \Delta}{\Gamma \rightsquigarrow \mathcal{B}_j = (\mathcal{B}'_j \wedge_j \mathcal{B}''_j), \Delta} (\wedge : r)$$

where $\Delta = \mathcal{B}_1, \dots, \mathcal{B}_{j-1}, \mathcal{B}_{j+1}, \dots, \mathcal{B}_m$. The cases for the rest of the inference rules are analogous. \square

We introduce also the following result but we delay the discussion about it until section 6.

Theorem 2. *Given an arbitrary proposition \mathcal{A} , let T be a deduction tree of \mathcal{A} . Then, with minor modifications to def. 17, a programmed grammar \mathcal{G} of type CF- λ can be constructed such that $L(\mathcal{G})$ is the set of words representing the labels of the leaves of T .*

5 An Illustration

Example 11 (continues ex. 2). Consider the grammar $\langle \mathcal{G} \rangle = (V_N, V_T, S, P)$ associated with $\langle \mathcal{A} \rangle$, whose components were presented or defined separately in examples 5 to 10 and definition 17. The language of $\langle \mathcal{G} \rangle$, calculated mechanically in ML, is

$$\{p \triangleright p, q \triangleright q\}$$

Now, we show how the word $p \triangleright p$ was produced

(S , r_0)
 ([w] ^ [=] ^ \$ ^ [=] _2.R] , r_1)
 ([w] ^ [=] ^ \$ ^ [=] _2.R] , r_3)
 ([w] ^ [=] ^ \$ ^ [=] _2.R] , r_4)
 ([w+] ^ [=] ^ \$ ^ [=] _2.R] , =>_2_5)
 ([w+] ^ [=] ^ \$ ^ [=] _1.L] ^ [=] _3.R] , =>_2_6)
 ([w+] ^ [=] ^ [=] ^ \$ ^ [=] _1.L] ^ [=] _3.R] , r_w+)

([w] ^ [=] ^ [=] ^ \$ ^ [=] _1.L] ^ [=] _3.R] , r_1)
 ([w] ^ [=] ^ [=] ^ \$ ^ [=] _1.L] ^ [=] _3.R] , r_3)
 ([w] ^ [=] ^ [=] ^ \$ ^ [=] _1.L] ^ [=] _3.R] , r_4)
 ([w+] ^ [=] ^ [=] ^ \$ ^ [=] _1.L] ^ [=] _3.R] , =>_1_1)
 ([w+] ^ [=] ^ [=] ^ \$ ^ [p.R] ^ [=] _3.R] , =>_1_2)
 ([w+] ^ [A_R] ^ [=] ^ \$ ^ [p.R] ^ [=] _3.R] , r_w+)

([w] ^ [A_R] ^ [=] ^ \$ ^ [p.R] ^ [=] _3.R] , r_1)
 ([w] ^ [A_R] ^ [=] ^ \$ ^ [p.R] ^ [=] _3.R] , r_3)
 ([w] ^ [A_R] ^ [=] ^ \$ ^ [p.R] ^ [=] _3.R] , r_4)
 ([w+] ^ [A_R] ^ [=] ^ \$ ^ [p.R] ^ [=] _3.R] , =>_3_5)
 ([w+] ^ [A_R] ^ [=] ^ \$ ^ [p.R] ^ [-_1.L] ^ [-_2.R] , =>_3_6)
 ([w+] ^ [A_R] ^ [-] ^ [-] ^ \$ ^ [p.R] ^ [-_1.L] ^ [-_2.R] , r_w+)

([w] ^ [A_R] ^ [-] ^ [-] ^ \$ ^ [p.R] ^ [-_1.L] ^ [-_2.R] , r_1)
 ([w] ^ [A_R] ^ [-] ^ [-] ^ \$ ^ [p.R] ^ [-_1.L] ^ [-_2.R] , r_3)
 ([w] ^ [A_R] ^ [-] ^ [-] ^ \$ ^ [p.R] ^ [-_1.L] ^ [-_2.R] , r_5)
 ([w] ^ [A_R] ^ [-] ^ [-] ^ \$ ^ [p.R] ^ [-_1.L] ^ [-_2.R] , r_6)
 ([w+] ^ [A_R] ^ [-] ^ [-] ^ \$ ^ [p.R] ^ [-_1.L] ^ [-_2.R] , -_2_3)
 ([w+] ^ [A_R] ^ [-] ^ [-] ^ \$ ^ [p.R] ^ [-_1.L] ^ [p.L] , -_2_4)
 ([w+] ^ [A_R] ^ [A_L] ^ [-] ^ \$ ^ [p.R] ^ [-_1.L] ^ [p.L] , r_w+)

([w] ^ [A_R] ^ [A_L] ^ [-] ^ \$ ^ [p.R] ^ [-_1.L] ^ [p.L] , r_1)
 ([w] ^ [A_R] ^ [A_L] ^ [-] ^ \$ ^ [p.R] ^ [-_1.L] ^ [p.L] , r_2)
 ([w] ^ [A_R] ^ [A_L] ^ [-] ^ \$ ^ [p.R] ^ [-_1.L] ^ [p.L] , q_1)
 ([w] ^ [A_R] ^ [A_L] ^ [-] ^ \$ ^ [p.R] ^ [-_1.L] ^ [p.L] , p_1)
 ([w] ^ [A_R] ^ [A_L] ^ [-] ^ \$ ^ [p.R] ^ [-_1.L] ^ [p.L] , p_2)
 ([w] ^ [A_R] ^ [A_L] ^ [-] ^ \$ ^ p ^ [-_1.L] ^ [p.L] , p_3)
 ([w] ^ [A_R] ^ [A_L] ^ [-] ^ p ^ \$ ^ p ^ [-_1.L] ^ [p.L] , e_1)
 ([w] ^ [A_R] ^ [A_L] ^ [-] ^ p ^ \$ ^ p ^ [-_1.L] ^ [p.L] , e_2)
 ([w] ^ [A_R] ^ [A_L] ^ [-] ^ p ^ \$ ^ p ^ [-_1.L] ^ [p.L] , e_3)
 ([w] ^ [A_R] ^ [A_L] ^ [-] ^ p ^ \$ ^ p ^ [-_1.L] , e_3)
 ...
 ([w] ^ [A_R] ^ [A_L] ^ [-] ^ p ^ \$ ^ p ^ [-_1.L] , e_11)
 ([w] ^ [A_R] ^ [A_L] ^ [-] ^ p ^ \$ ^ p , e_11)
 ...
 ([w] ^ [A_R] ^ [A_L] ^ [-] ^ p ^ \$ ^ p , e_15)
 ([w] ^ [A_R] ^ [-] ^ p ^ \$ ^ p , e_15)
 ([w] ^ [A_R] ^ [-] ^ p ^ \$ ^ p , e_16)
 ([w] ^ [-] ^ p ^ \$ ^ p , e_16)
 ([w] ^ [-] ^ p ^ \$ ^ p , e_17)
 ([w] ^ [-] ^ p ^ \$ ^ p , e_18)
 ([w] ^ p ^ \$ ^ p , e_18)

([w] ^ p ^ \$ ^ p , e_19)
 (p ^ \$ ^ p , e_19)
 (p ^ \$ ^ p , r_f)
 (p ^ -> ^ p , r_f)

In the above words, we observe an interesting pattern

Word	Represented Sequent	Inference Rule to be used
[=>_2.R]	$\rightsquigarrow \supset_2$	$(\supset : r)$
[=>_1.L] ^ [=] _3.R]	$\supset_1 \rightsquigarrow \supset_3$	$(\supset : l)$
[p.R] ^ [=] _3.R]	$\rightsquigarrow p, \supset_3$	$(\supset : r)$
[p.R] ^ [-_1.L] ^ [-_2.R]	$\neg_1 \rightsquigarrow p, \neg_2$	$(\neg : r)$
[p.R] ^ [-_1.L] ^ [p.L]	$\neg_1, p \rightsquigarrow p$	Axiom

Turn this upside down and you will get a branch of the deduction tree of $\langle \mathcal{A} \rangle$. The other branch, which lead to the obtention of $q \triangleright q$, starts after $(\supset : l)$ (cf. production r_3 in example 6)

6 Further Discussion

In [8] the authors present an interesting overview of the mathematical, logical and philosophical standpoint regarding the concept of (automated) proof. We do not pretend here to follow that debate; instead we would like to highlight some characteristics, as pointed out there, a proof should exhibit.

A proof is formalizable. That is the quintessence of this work.

A proof is examinable. Section 5 was devoted to show how examinable our proofs are (of course, some familiarity with our construction is required). On the one hand, since production labels are inherently contained in the chains of derivation, it is absolutely possible to determine which production has been applied. On the other hand, a symbol susceptible to be rewritten indicates clearly which inference rule of G is to be used – for instance $[=>_2.R]$ implies the use of $(\supset : r)$ as noticed above. This can be further improved by using a set of production labels more meaningful than the one proposed.

A proof is convincing. The tautology problem in Propositional Logic is certainly decidable and computable: we can have a Turing machine that solves it. And programmed grammars are used basically as Turing machines, so there is no room for surprise here – a simulated proof in G should be taken as granted. However, as stated earlier, G has a rewriting nature which is essentially context-driven. Although “atypical” our theorem prover has the ability to solve this problem (we must say with conclusive evidence since it does not bring only a *yes-no* answer) on the basis of context-freedom⁴, that is,

⁴Remember that, since the very beginning, we were dealing with logical propositions in a context-free way (cf. definition 12).

it involves a simplification (in terms of conciseness) of the rewriting operations involved in the task.

It is nice to come up with such a fair and understandable word like $p \triangleright p$. Readability was therefore the motivating force behind definitions 8 (function χ) and 25 (erasing productions), which in turn lead us to the formulation of a grammar of type CF. But, just in case you feel uncomfortable with the use of λ -productions, we can follow an alternative direction. Suppose we were to produce words of the form $w_1 p w_2 \triangleright w_3 p w_4$, were $w_i \in (V'_T)^*$ and V'_T contains additional terminal symbols not members of the original V_T , for instance

$$\{w\}^{\wedge}\{A_R\}^{\wedge}\{A_L\}^{\wedge}\{-\}^{\wedge}p^{\wedge}\rightarrow^{\wedge}\{-_1.L\}^{\wedge}p$$

As a consequence, “human” legibility will be diminished, but the meaning will remain untouched along with our capability to examine proofs. The twist is indeed feasible and it represents no harm to our previous results. All we need is to transform in an adequate manner⁵ the so-called erasing productions. What is the overall effect of this minor modification? No λ -productions at all! This argument has in fact motivated theorem 2. (A far more elegant approach could be based instead on *pure programmed grammars* [5]: we work with just one single alphabet and stop the derivation process after the introduction of the letters.)

Again in [8] the authors describe various categories for theorem provers, which are determined by some criterion (technique, for example). It is worth noting that in this classification there is no such a place for provers for Propositional Logic. On their own words

Today the most essential theoretical and practical work on this logic is focused on the SAT problem (and SAT solvers)

We believe that in the realm of Propositional Logic other problems deserve to be further explored.

References

- [1] Gallier, J. H., *Logic for Computer Science*, John Wiley and Sons, 1987.
- [2] Wallen, L., *Automated Proof Search in Non-Classical Logics (Efficient Matrix Proof Methods for Modal and Intuitionistic Logics)*, The MIT Press, 1990.
- [3] Wang, H., “Toward Mechanical Mathematics,” *IBM Journal of Research and Development*, V4, N1, pp. 2–22, 1960
- [4] Mendelson, E., *Introduction to Mathematical Logic*, 4th edition, Chapman and Hall, 2001.
- [5] Păun, G., Dassow, J., *Regulated rewriting in Formal Language Theory*, Springer-Verlag, Berlin, 1989.
- [6] Dassow, J., *Grammars with Regulated Rewriting*, Otto-von-Guericke-Universität, Magdeburg http://fuzzy.cs.uni-magdeburg.de/theo/dassow_eng.html 2004.
- [7] Hopcroft, J., Ullman, J., *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley.
- [8] Caferra, R., Leitsch, A., Peltier, N., *Automated Model Building*, Applied Logic Series 31, Kluwer, 2004.

⁵Replace λ with the appropriate terminal symbol in V'_T . Notice that these productions have no influence over the characterization of G inference rules.