Available online at [www.sciencedirect.com](http://www.sciencedirect.com/)

[Electronic Notes in Theoretical Computer Science 323 (2016) 181–196](http://dx.doi.org/10.1016/j.entcs.2016.06.012)

[www.elsevier.com/locate/entcs](http://www.elsevier.com/locate/entcs)

On Strong Normalization in Proof-Graphs for Propositional Logic

Marcela Quispe-Cruz[1](#_bookmark0)

*Universidad Cat´olica San Pablo (Peru´)*

Edward Haeusler[2](#_bookmark1)

*Pontificia Universidade de Rio de Janeiro (Brazil)*

Lew Gordeev[3](#_bookmark2)

*Tuebingen University, Ghent University*

**Abstract**

Traditional proof theory of Propositional Logic deals with proofs whose size can be huge. Proof theoretical studies discovered exponential gaps between normal or cut free proofs and their respective non-normal proofs. The use of proof-graphs, instead of trees or lists, for representing proofs is getting popular among proof-theoreticians. Proof-graphs serve as a way to study complexity of propositional proofs and to provide more efficient theorem provers, concerning size of propositional proofs.

Fpl-graphs were initially developed for minimal implicational logic representing proofs through references rather than copy. Thus, formulas and sub-deductions preserved in the graph structure, can be shared deleting unnecessary sub-deductions resulting in the reduced proof. In this work, we consider full minimal propositional logic and show how to reduce (eliminating maximal formulas) these representations such that strong normalization theorem can be proved by simply counting the number of maximal formulas in the original derivation. In proof-graphs, the main reason for obtaining the strong normalization property using such simple complexity measure is a direct consequence of the fact that each formula occurs only once in the proof-graph and the case of the hidden maximum formula that usually occurs in the tree-form derivation is already represented in the fpl-graph.

*Keywords:* Proof Theory, Proof Graphs, N-Graphs, Intuitionistic Logic, Sequent Calculus, Multiple-Conclusion Systems.

# Introduction

Recently the use of graphs instead of trees to represent proofs has been shown to be more efficient[[2](#_bookmark19)][[1](#_bookmark20)], while also being helpful to better address the lack of symmetry

1 Email: [mquispec@ucsp.edu.pe](mailto:mquispec@ucsp.edu.pe)

2 Email: [hermann@inf.puc-rio.br](mailto:hermann@inf.puc-rio.br)

3 Email: [lew.gordeew@uni-tuebingen.de](mailto:lew.gordeew@uni-tuebingen.de)

<http://dx.doi.org/10.1016/j.entcs.2016.06.012>

1571-0661/© 2016 The Author(s). Published by Elsevier B.V.

This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

in classical ND logic [[4](#_bookmark22)] and the complexity of the proof normalization process. Previously we have already presented mimp-graphs as a new proof system developed for minimal implicational logic [[6](#_bookmark24)], whose deductions are structured as proof-graph. The point is that in mimp-graphs it is easy to determine maximal formulas [4](#_bookmark4) and upper bounds on the length of reduction sequences leading to normal proofs. Thus a normalization theorem is proved by counting the number of maximal formulas in the original derivation. The strong normalization property is a direct consequence of such normalization, since any reduction decreases the corresponding measure of derivation complexity. In the present paper we wish to explain this procedure more clearly and expand it onto full propositional logic.

Mimp-graphs are directed graphs whose nodes and edges are labelled. Mo- reover we distinguish two parts, one representing the inferences of a proof, and the other the formulas. For the formula-part of a mimp-graph, we use directed acyclic graphs, that we denominated formula graphs, consist of basis in the mimp- graph construction and contain only formula nodes sharing formula nodes, thus each formula node only need to occur once in the graph, an example is shown in the left-hand side of Figure [1](#_bookmark3): the propositions *P* and *Q* occur once in the graph.

For the inference-part of a mimp-graph we have the rule nodes (R-nodes) that are labelled by the names of the inference rules. The logic connectives and inference names may be indexed, in order to achieve a 1-1 correspondence between formulas (inferences) and their representations (names), an shown in the right-hand side of Figure [1](#_bookmark3): the R-node ➔*E*1 has as major premise the formula graph (*P* ➔*Q*)➔(*P* ➔*Q*) and as minor premise the formula graph *P* ➔*Q*.

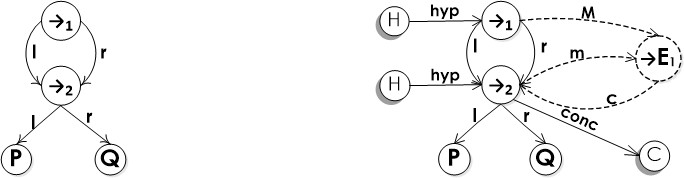


Fig. 1. Formula (*P* ➔ *Q*) ➔ (*P* ➔ *Q*) depicted as a formula graph (left-hand side) and as major premise of the R-node ➔*E*1 (right-hand side).

Any strong normalization (SN) proof has to take care of every possible detour (maximal formula) that appears in a derivation. If one considers the normaliza- tion as a dynamic process, not all detours that are eliminated in a derivation are explicitly present since the beginning of the process. For example, the permutation- conversions used by Prawitz in the (weak) normalization of intuitionistic logic were designed to take care of hidden maximal formulas (see discussion below). Hiding a detour is a feature of elimination rules similar to ∨. As far as we know, Natural De- duction systems with rules similar to ∨-elimination use permutation-conversions to prove (weak) normalization. SN should deal with these permutation-conversions as well as systems that admit it. Another instance of hidden maximal formula is when after a conversion (reduction), new maximal formulas can appear. This already

4 A *maximal formula* is a formula occurrence that is consequence of a introduction rule and the major premise of a elimimination rule.

happens in the case of the ➔-reduction. In the derivation below, after eliminating the maximal formula *A* ➔ *B*, every discharged occurrence of *A* in Π2 that is the major premise of a rule elimination will be a maximal formula. Since the number of such occurrences is unbounded, the number of maximal formulas in the original derivation is not a good upper-bound for the number of reduction applications. This is a key point when discussing SN proofs.

Π1

′

*A r*-intro *A*

*B*

[*A*]

Π2

*B A* ➔ *B*

The permutation-conversions attack another situation, as seen in the permuta- tion-conversion below. *C* ➔ *D* is a hidden maximal formula that becomes a maximal formula after the permutation-conversion (right-hand side).

[*A, C*]

Π1

*D*

[*B, C*]

Π2

*D*

[*A, C*]

Π1

Π *D* Π

[*B, C*]

Π2

*D*

Π *A* ∨ *B C*➔*D C*➔*D*

*C C*➔*D*

*C C*➔*D*

*C C*➔*D*

*D* ⊳

*A* ∨ *B D D D*

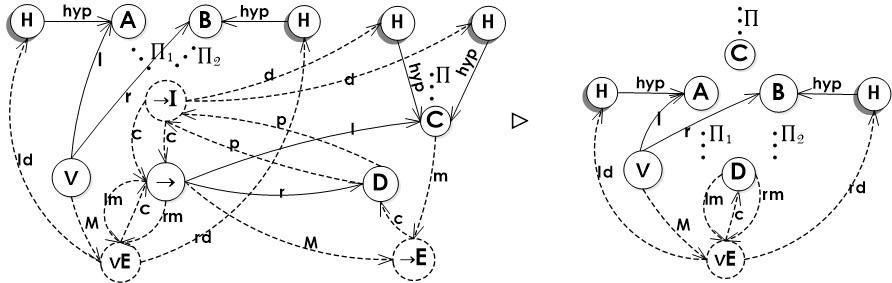


Fig. 2. Fpl-graph of the derivation above and its reduction.

In the case of our graphs, since every formula labels one and only one vertex of the graph, all the situations discussed above are explicit. That is, every possible maximal formula that can appear in the normalization proof is already there, and there is no hidden maximal formulas, as shown in Figure [2](#_bookmark5), where ➔*q* is twice conclusion of the R-node ➔*Ii*, thus it is representing already two maximal formulas. The reduction, shown on the right-hand side of figure, decreases the number of maximal formulas. Consequently, we can prove SN by induction on the number of maximal formulas.

# Proof-graphs for full propositional logic

In [[6](#_bookmark24)] we considered implication as the only logic connective. Let us now turn to a more general presentation of proof-graphs for propositional calculus that includes implication, conjunction and disjunction that we called proof-graph for full propo- sitional logic (fpl-graph). First, we define sets of labels to nodes and edges of the

graph, along with a partial ordering on its R-nodes that allows to pass through the nodes of the structure. We will also develop the normalization procedure for these proof-graphs.

We want to emphasize that the fpl-graphs put together information on formula graphs and R-nodes. To make it more transparent we can use different types of lines. In this way F-nodes and edges between them are used solid lines, whereas R- nodes and edges between them and adjacent premises and/or conclusions are used dashed lines and additionally delimiter nodes have been shaded.

**Definition 2.1** [Label types] There are five types of labels:

* *R-Labels* is the set of labels for *rule nodes*: (➔*Em*∈N*,* ➔*In*∈N, (*Io*∈N, (*Ep*∈N, v*Iq*∈N, v*Er*∈N},
* *F-Labels* is the set of labels for *formula nodes*: *{*➔*i*∈N, (*j*∈N, v*k*∈N*}* and the propo- sitional letters (*P, Q, R, ...*},
* *D-Labels* is the set of labels for *delimiter nodes*: (*Hk*∈N, *C*}.
* *EF -Labels* is the set of labels for *formula edges*: *{l* (left), *r* (right)*}*,
* *EM -Labels* is the set of labels for *rule edges*: *{ pi*∈N (premise), *rpj*∈N (right premise), *lpk*∈N (left premise), *rml*∈N (right minor premise), *lmm*∈N (left minor premise), *mn*∈N (minor premise), *Mo*∈N (major premise), *cp*∈N (conclusion), *dq*∈N (discharge), *ldr*∈N (left discharge), *rds*∈N (right discharge), *hypt*∈N (hypothesis), *conc* (final conclusion)},

The union of these five sets of label types will be called LBL. We will use the terms *αm*, *βn* and *γr* to represent the principal connective of the formula *α*, *β* and *γ* respectively.

**Definition 2.2** A proof-graph for full propositional logic (fpl-graph) *G* is a directed graph (*V, E, L*⟩ where: *V* is a set of nodes, *L* is a subset of LBL, *E* is a set of labelled *edges* (*v* ‹ *V*, *t* ‹ *EF* ∪*M* -Labels, *v*′ ‹ *V* ⟩, of source *v*, of target *v*′ and label

*t*

*t* and identified with the arrow *v*

*v*′.

——→

Fpl-graphs are recursively defined as follows:

**Basis** If *G*1 is a formula graph with root node *αm* then the graph *G*2 defined as

*conc hyp*

*G*1 with delimiter nodes *Hn* and *C* and edges *αm*——→*C* and *Hn*——→*αm* is a

fpl-graph.

* **E** If *G*1 and *G*2 are fpl-graphs, and the graph (intermediate step) obtained by

*l*

*G*1 s *G*2 contains the edge ➔*q* ——→*αm* and two nodes ➔*q* and *αm* linked to the

delimiter node *C*, then the graph *G*3 that is defined as *G*1 s *G*2 with

1. the removal of ingoing edges in the node *C* which were generated in the intermediate step (see the figure below, dotted area in *G*1 s *G*2);
2. an R-node ➔*Ei* at the top position;
3. the edges: *α*

*mnew*

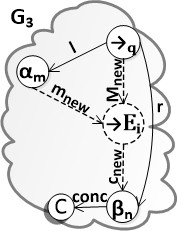
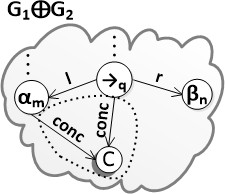
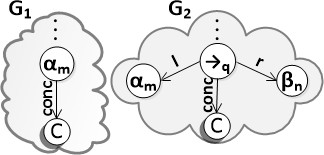
*Mnew*

*cnew*

*conc*

*m*———→➔*Ei*, ➔*q* ———→➔*Ei*, ➔*Ei*——→*βn* and *βn*——→*C*, where *new*

is a fresh (new) index ranging over all edges of kind *c*, *m* and *M* ingoing and/or outgoing of the formula-nodes *αm*, *βn* and ➔*q* ;

is a fpl-graph (see figure below).

c c

* + **I** If *G*1 is a fpl-graph and contains a node *βn* linked to the delimiter node *C* and the node *αm* linked to the delimiter node *Hk*, then the graph *G* that is defined as
    1. *G* ∶= *G*1 s *G*2, such that *G*2 is a formula graph with root node ➔*t* linked to

*l r*

F-nodes *αm* and *βn* by the edges: ➔*t*——→*αm*, ➔*t*——→*βn*;

*conc*

* + 1. with the removal of the edges: *βn*——→C;
    2. an R-node ➔*Ij* at the top position;
    3. the edges: *β*

*pnew*

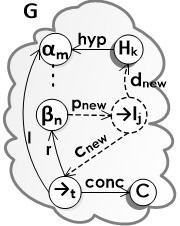
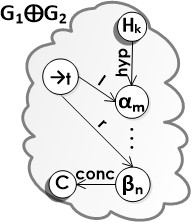
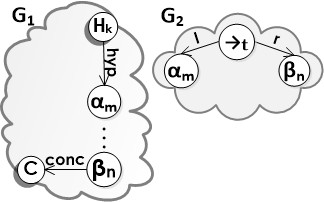
*cnew*

*conc*

*dnew*

*n*——→➔*Ij*, ➔*Ij*——→➔*t*, ➔*t*——→*C* and ➔*Ij*——→*Hk*, where *new* is

a fresh (new) index considering all edges of kind *p*, *d* and *c* ingoing and/or outgoing of the formula-nodes *αm*, *βn* and ➔*q* ;

is a fpl-graph (see figure below; the *αm*-node is *discharged* ).

c c

(**I** If *G*1 and *G*2 are propositional fpl-graphs and *G*1 contains *αm* linked to the D-node *C* and *G*2 contains *βn* linked to the D-node *C*, then the graph *G* that is defined as

1. *G* ∶= *G*1 s *G*2 s *G*3 with the removal of the ingoing edges in the node *C* which were generated in the intermediate step (see figure below, dotted area in *G*1 s *G*2 s *G*3);
2. an R-node (*Ii* at the top position;
3. the edges: *α*

*lpnew*

*I* , *β*

*rpnew*

*I* , (*I*

*cnew*

and

*conc*

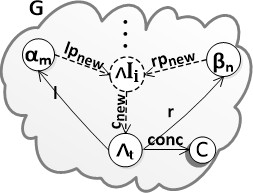
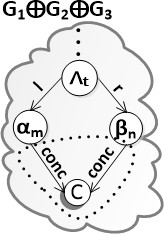
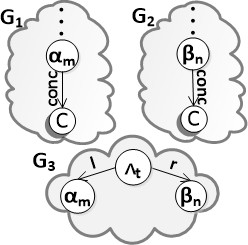
*C*,

*m*———→( *i*

*n*———→( *i*

*i*——→(*t*

(*t*——→

is a fpl-graph, see figure below.

c c

v**E** If *G*1, *G*2 and *G*3 are propositional fpl-graphs, and the graph obtained by

(*G*1 g *G*2) s *G*3 [5](#_bookmark7) (intermediate step) contains the nodes: v*t* and *σr* linked to the D-node C ( *σr* twice); and *αm* and *βn* are subformulas of v*t* and are linked to D-nodes H, then the graph *G* that is defined as (*G*1 g *G*2) s *G*3 with

1. the removal of the ingoing edges in the node *C* which were generated in the intermediate step (see figure below);
2. an R-node v*Ei* at the top position;
3. the edges: *σ*

*lmw*

*E* , *σ*

*rmw E* ,

*Mw*

*E* , v*E*

*cw σ* , v*E*

*ldw H* ,

*r*———→v *i*

*r*———→v

*i* v*t*——→v *i*

*i*——→ *r*

*i*——→ *u*

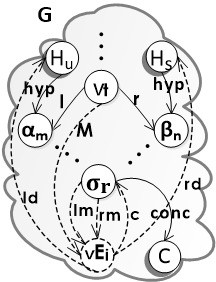
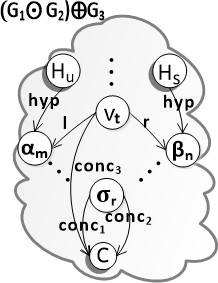
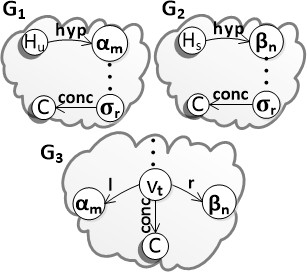
v*E rdw*

*conc*

*i*——→*Hs* and *σr*——→C, where *w* is a fresh (new) index considering all edges

of kind *p*, *d* and *c* ingoing and/or outgoing of the formula-nodes *αm*, *βn* and

* *q* ;

is a fpl-graph, see figure below.

c c

* **Iv,** (*E***l,** v*I***l,** v*I***r** Similar to other cases of construction (for an expanded version see [[5](#_bookmark23)]).

In the terminology about inference rules or R-nodes, when an R-node has more than one incoming edge, these are distinguished by calling them left, right, major or minor, or a combination of these terms and so also the F-node ‘premise’ associated with these edges. Thus, the major premise in R-node contains the connective that is eliminated; the other premise in R-node is called ‘minor’. Two premises that play a more or less equal role in the inference are called ‘left’ and ‘right’. For instance, an R-node vE has a major premise, a left minor premise and a right minor premise; an R-node (I has a left premise and a right premise.

The term *R-node sequence* is representing a deduction, and if it is a smaller part of another R-node sequence (deduction), then it is called *a subsequence* of the latter. A subsequence that derives a premise of the last R-node application in an R-node sequence is called a direct R-node subsequence. Instead of writing “the direct R-node subsequence that derives the minor premise of the last inference of an R-node sequence D”, we simply write *“the minor subsequence of D”*.

Fpl-graphs need to conform a number of restrictions. To formulate the first one, acyclicity, we need the notion of *inferential order* on R-nodes that allows to pass through the nodes of the structure preventing nodes from infinitely reoccurring in a *path*.

5 By definition *G*1 ⊙ *G*2 equalizes or collapses the R-nodes of *G*1 with the R-nodes of *G*2 that have the same set of premises and conclusion keeping the inferential order of each node, and equalizes F-nodes of *G*1 with the F-nodes of *G*2 that have the same label, and equalizes edges with the same source, target and label into one.

**Definition 2.3** Let *G* be a fpl-graph. An *inferential order* < on nodes of *G* is a partial ordering of the R-nodes of *G* such that *n* < *n*′ iff *n* and *n*′ are R-nodes and

there is an F-node *f* such that *n lbl*1 f *lbl*2 *n*′ and *lbl* is *c* and *lbl* is *m*, or *lbl*

is *c*

——→ ——→ 1 2 1

and *lbl*2 is *M* , or *lbl*1 is *c* and *lbl*2 is *p*. Node *n* is a *top position* node if *n* is maximal

w.r.t. <.

**Definition 2.4** (1) For *ni* ‹ *V* ,a *path* in fpl-graph is a sequence of vertices and edges

of the form: *n*

*lbl*1

*lbl*2

*lblk*−2

*lblk*−1

1 ——→ *n*2 ——→ *...* ———→ *nk*−1 ———→ *nk*, such that *n*1 is a hypothesis

formula node, *nk* is the conclusion formula node, *ni* alternating between a rule node and a formula node. The edges *lbli* alternate between two types of edges: the first is *lblj* ‹ (*rm, lm, m, M, rp, lp, p*} and the second *lblj* = *c*. (2) A *branch* in fpl-graph is an initial part of a *path* which stops at the conclusion F-node of the graph or at the first minor (or left) premise whose major (or right) premise is the conclusion of a rule node. (3) An *insertable branch* in fpl-graph is a branch that is bifurcated by a maximal formula: ➔*I* followed by ➔*E*.

The Lemma [2.5](#_bookmark8) below enables us to prove that a given graph *G* is a fpl-graph. Among others it says that we have to check that each node of *G* is of one of the possible types that generate the construction cases of Definition [2.2](#_bookmark6).

In order to avoid overloading of indexes, we will omit whenever possible, the indexing of edges of kind *lm*, *rm*, *lp*, *rp*, *ld* and *rd*, keeping in mind that the coherence of indexing is established by the kind of rule-node to which they are linked.

**Lemma 2.5** *G is a fpl-graph if and only if the following hold:*

1. *There exists a well-founded (hence acyclic) inferential order* < *on all rule nodes of the fpl-graph.*
2. *Every node N of G is of one of the following ten types:*

**P** *N is labelled with one of the propositional letters: {P, Q, R, ... }. N has no outgoing edges l and r.*

**F** *N has one of the following labels:* ➔*i,* (*j or* v*k, and has exactly two outgoing edges with label l and r. N has outgoing edges with labels p, m, M, lm, rm, lp, rp; and ingoing edges with label c and hyp.*

**E**→ *N has label* ➔*E*

*c*

*and has exactly one outgoing edge* ➔*E*

*β , where β*

*i i*——→ *n n*

*m*

*is a node type* **P** *or* **F***. N has exactly two ingoing edges αm*——→➔*Ei and*

*M*

* + *q* ——→➔*Ei, where αm is a node type* **P** *or* **F***. There are two outgoing edges*

*l r*

*from the node* ➔*q:* ➔*q* —→*αm and* ➔*q* ——→*βn.*

**I**→ *N has label* ➔*Ij (or* ➔*Ivj, if discharges an hypothesis vacuously), has one*

*c*

*outgoing edge* ➔*Ij*—→➔*t, and one (or zero for the case* ➔*Iv) outgoing edge*

*d p*

* + *Ij*——→*Hk. N has exactly one ingoing edge: βn*——→➔*Ij, where βn is a node*

*l*

*type* **P** *or* **F***. There are two outgoing edges from the node* ➔*t:* ➔*t*—→*αm and*

*r*

* + *t*——→*βn such that there is one (or zero for the case* ➔*Iv) ingoing edge to the*

*hyp*

*node αm: Hk*——→*αm.*

**I**∧ *N has label* (*Ii, one outgoing edge* (*I*

*c i*—→(*t*

*and exactly two ingoing edges:*

*lp rp*

*αm*——→(*Ii and βn*——→(*Ii, where αm and βn are nodes type* **P** *or* **F***. There*

*l r*

*are two outgoing edges from the node* (*t:* (*t*—→*αm and* (*t*——→*βn.*

**E**∧ *N has label* (*E , one outgoing edge* (*El c α where α (or β*

*in the*

*i i*—→ *m m n p*

*case* (*Eri is a node type* **P** *or* **F** *and has exactly one ingoing edge:* (*t*——→(*Ei.*

*l r*

*There are two outgoing edges from the node* (*t:* (*t*—→*αm and* (*t*——→*βn.*

**I**∨ *N has label* v*Il , one outgoing edge* v*Il c and has exactly one ingoing*

*i i*—→v*t*

*p*

*edge: αm*——→v*Ili where αm (or βn in the case* v*Iri) isa node type* **P** *or* **F***.*

*l r*

*There are two outgoing edges from the node* v*t:* v*t*—→*αm and* v*t*——→*βn.*

**E**∨ *N has label* v*E , three outgoing edges* v*E*

*c*

*σ ,* v*E*

*ld*

*H and*

*i i*—→ *r*

*rd*

*i*——→ *u*

*M lm*

v*Ei*——→*Hs; and it has exactly three ingoing edges:* v*t*——→v*Ei, σr*——→ v*Ei,*

*rm*

*σr*———→v*Ei where αm (or βn in the case* v*Ei) isa node type* **P** *or* **F***. There*

*l r*

*are two outgoing edges from the node* v*t:* v*t*—→*αm,* v*t*——→*βn and the hypoth-*

*hyp hyp*

*esis edges: Hu*——→*αm and Hs*——→*βn.*

**H** *N has label Hk and has exactly one outgoing edge hyp.*

**C** *N has label C and has exactly one ingoing edge conc.*

**Proof.**

c: Argue by induction on the construction of fpl-graph (Definition [2.2](#_bookmark6)). For every construction case for fpl-graphs we have to check the three properties stated in Lemma. Property (2) is immediate. For property (1), we know from the induction hypothesis that there is an inferential order < on R-nodes of the fpl-graph. In construction cases ➔*I*, ➔*E*, (*I*, (*E*l, (*E*r, v*I*l, v*I*r or v*E*, we make the new R-node that is introduced highest in the <-ordering, which yields an inferential ordering on R-nodes. In the construction case (*I*, when we have two inferential orderings, <1 on *G*1 and <2 on *G*2. Then *G*1 s *G*2 can be given an inferential ordering by taking the union of <1 and <2 and in addition putting *n* < *m* for every R-node *n, m* such that *n* ‹ *G*1*,m* ‹ *G*2. In the construction case vE, when we have three inferential orderings, <1 on *G*1, <2 on *G*2 and <3 on *G*3. Then (*G*1 g *G*2) s *G*3 can be given an inferential ordering by taking the union of <1, <2 and <3 and in addition putting *n* < *m* < *p* for every R-node *n, m, p* such that *n* ‹ *G*1*,m* ‹ *G*2*,p* ‹ *G*3.

-: Argue by induction on the number of R-nodes of *G*. Let < be the topological order that is assumed to exist. Let *n* be the R-node that is maximal w.r.t. <. Then *n* must be on the top position. When we remove node *n*, including its edges linked (if *n* is of type **I**∨) and the node type **C** is linked to the premise of the R-node, we obtain a graph *G*′ that satisfies the properties listed in Lemma. By induction hypothesis we see that *G*′ is a fpl-graph. Now we can add the node *n* again, using one of the construction cases for fpl-graphs: *Basis* if *n* is a **L** node, **F** node, ➔*E* node or ➔*I* node, **I**∧ if *n* is a (*I* node, **E**∧ if *n* is a (*E*l node or (*E*r node, **I**∨ if *n* is a v*I*l node or v*I*r node, **E**∨ if *n* is a v*E* node. □

# Normalization for fpl-graphs

* 1. *Elimination of maximal formula*

In this section, we describe the normalization process for fpl-graphs. Eliminating a maximal formula is very similar to the procedure for mimp-graphs described in [[6](#_bookmark24)], where we considered only the case of implication, now we define the maximal formulas in conjunction, disjunction and implication. The notion of *reordering* is provided as well, because when the maximal formula is removed a reordering of nodes occurs.

**Definition 3.1** A *maximal formula m* in a fpl-graph *G* is a sub-graph of *G* as follows:

* (*I* **followed by** (*E***l.** It is composed of (see Figure [3(a)](#_bookmark9)):

1. the F-nodes: *αm*, *βn* and (*q*, where (*q* has zero or more ingoing/outgoing edges,

e.g. (*q* could be premise or conclusion of others R-nodes;

1. the R-nodes: (*Ii* and (*Ell*, where (*Ii* has an inferential order lower than (*Ell* and there are zero or more maximal formulas between them [6](#_bookmark11) . If these nodes occur in different branches, a branch must be insertable in the other branch or bifurcated by an R-node vE;

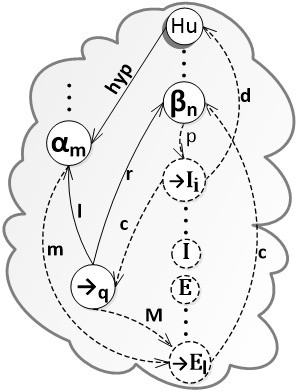
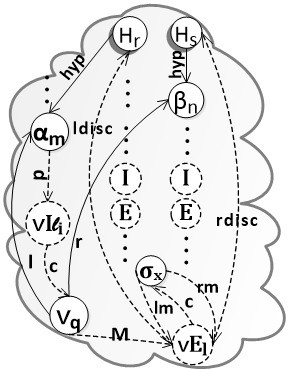
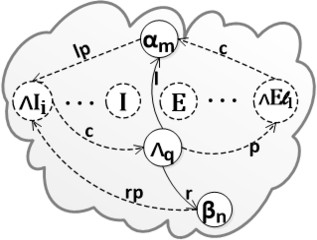
*l r lp rp c p*

1. the edges: (*q*—→*αm*, (*q*—→*βn*, *αm*—→(*Ii*, *βn*——→(*Ii*, (*Ii*—→(*q*, (*q*—→(*Ell* and

*c*

(*Ell*—→*αm*.

There is a symmetric case for (*I* followed by (*E*r.



(a) ∧*I* followed by ∧*E*l (b) ∨*I*l followed by ∨*E* (c) ➔*I* followed by ➔*E*

Fig. 3. Maximal Formulas

* + v*I***l followed by** v*E***.** It is composed of (see Figure [3(b)](#_bookmark10)):

1. the F-nodes: *αm*, *βn*, v*q* and *σx*, where v*q* has zero or more ingoing/outgoing edges;
2. the D-nodes: *Hr* and *Hs*;

6 The maximal formulas are represented in the figure by nodes labelled with *I* and *E*

1. the R-nodes in ascending inferential order: v*Ili* and v*El*, and there are zero or more maximal formulas in branches between them. If these nodes occur in different branches, a branch must be insertable in the other branch or bifurcated by an R-node v*E*;

*l r p c M lm*

1. the edges: v*q*—→*αm*, v*q*—→*βn*, *αm*—→v*Ili*, v*Ili*—→v*q*, v*q*——→v*El*, *σx*——→ v*El*,

*rm c ld rd*

*σx*——→v*El*, v*El*—→*σx*, v*El*—→*Hr* and v*El*——→*Hs*.

There is a symmetric case for v*Ir* followed by v*E*.

* + ➔*I* **followed by** ➔*E***.** It is composed of (see Figure [3(c)](#_bookmark10)):

1. the formula nodes: *αm*, *βn* and ➔*q* , where ➔*q* has zero or more ingoing/outgoing edges;
2. the D-node: *Hu*;
3. the R-nodes in ascending inferential order: ➔*Ii* and ➔*El*, and there are zero or more maximal formulas between them. If these nodes occur in different branches, a branch must be insertable in the other branch or bifurcated by an R-node v*E*;

*l r p c d M*

1. the edges: ➔*q* —→*αm*, ➔*q* —→*βn*, *βn*—→➔*Ii*, ➔*Ii*—→➔*q* , ➔*Ii*—→*Hu*, ➔*q* ——→➔*El*,

*m c*

*αm*——→➔*El* and ➔*El*—→*βn*.

**Definition 3.2** A reordering of a given fpl-graph *G* is obtaining by supplying *G*

with the following (new) inferential order on the R-nodes of *G*.

* + *o*(*tm*)= 0 for an R-node *tm* starting with hypothesis.
  + *o*(*t*)= *o*(*t*′)+ 1 if the conclusion formula of R-node *t*′ is premise, right premise or major premise of *t*.

**Proposition 3.3** *A graph obtained by a reordering according to Deﬁnition* [*3.2*](#_bookmark12) *is a fpl-graph.*

Note that the actual situation is more complicated than those sketched in Fig- ures [3(a)](#_bookmark9), [3(b)](#_bookmark10) and [3(c)](#_bookmark10). There are five sub-cases for each maximal formula due to the presence of disjunction and other maximal formulas. For brevity we only show how subcases of the elimination of v*I*l followed by v*E* are treated (for an expanded version see [[5](#_bookmark23)]).

**Definition 3.4** Given a fpl-graph *G* with a maximal formula *m*, *eliminating a maximal formula* is the following transformation of a fpl-graph:

**Elimination of** v*I***l followed by** v*E* There is a symmetric case for v*I*r followed by v*E*. The elimination of this maximal formula is the following operation on a fpl-graph:

1. If there are no *maximal formulas* in branches between the R-nodes v*Ili* and

v*El* then follow these steps:

* 1. If v*Ili* and v*El* are *not bifurcated* by one v*E* then (see cases 1 and 2 in Figure [4](#_bookmark15)).

Remove the R-nodes v*Ili* and v*El*, and their edges.

If the F-node v*q* only has outgoing edges to sub-formulas then remove it (see case 2 in Figure [4](#_bookmark15)).

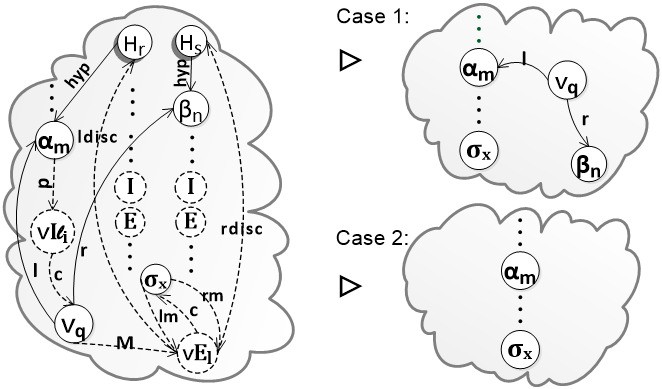


Fig. 4. Elimination of ∨*I*l followed by ∨*E*: Cases 1 and 2.

* 1. Else If v*Ili* represents two R-nodes then (see case 3 in Figure [5](#_bookmark16)): Remove R-nodes v*Ili* and v*El*, and their edges.

*lm rm c*

Eliminate edges: v*q*——→v*Ek*, v*q*———→v*Ek* and v*Ek*—→v*q*.

If the F-node v*q* only has outgoing edges to sub-formulas then remove it (see case 4 in Figure [5](#_bookmark16)).

*lm rm c*

Add the edges: *σx*——→v*Ek*, *σx*———→v*Ek* and v*Ek*—→*σx*.

Incorporate the inference orders of sequence Π*m*

*x*

of the Figure [5](#_bookmark16) in the

minor subsequence of v*Ek* (left and right).

* 1. Else (see case 5 in Figure [6](#_bookmark17))

Remove the R-node v*Ili*, and its edges.

*lm rm c*

Eliminate edges: v*q*——→v*Ek*, v*q*———→v*Ek* and v*Ek*—→v*q*.

*lm rm c*

Add the edges: *σx*——→v*Ek*, *σx*———→v*Ek* and v*Ek*—→*σx*.

Incorporate the inference order of node v*El* with its subsequences Π*m* and Π*n* as shown in Figure [6](#_bookmark17) in the right minor subsequence of v*Ek* and incor- porate the R-node sequence Π*m* in the left minor premise of v*Ek*.

*x*

*x*

*x*

* 1. Apply the operation defined in Definition [3.2](#_bookmark12) to the resulting graph. Note that Proposition [3.3](#_bookmark13) ensures that the result is a fpl-graph.

1. Otherwise eliminate the *maximal formulas* in branches between the R-nodes

v*Ili* and v*El*.

**Lemma 3.5** *If G is a fpl-graph with a maximal formula m and G*′ *is obtained from G by eliminating m, then G*′ *is also a fpl-graph. Moreover G and G’ both have the same conclusion, i.e. the F-label being the source of conc.*

**Proof.** We use Lemma [2.5](#_bookmark8). All nodes in *G*′ are of the right form: **P**, **F**, **E**→, **I**→, **E**∨, **I**∨, **E**∧, **I**∧, **H** or **C**. We verify that *G*′ has one ingoing edge with label *conc* to the D-node with label *C* and that is acyclic and connected. Finally, an inferential order on *G*′ (as defined in Definition [3.2](#_bookmark12)) between rule nodes must preserve the derivability and the conclusions. □

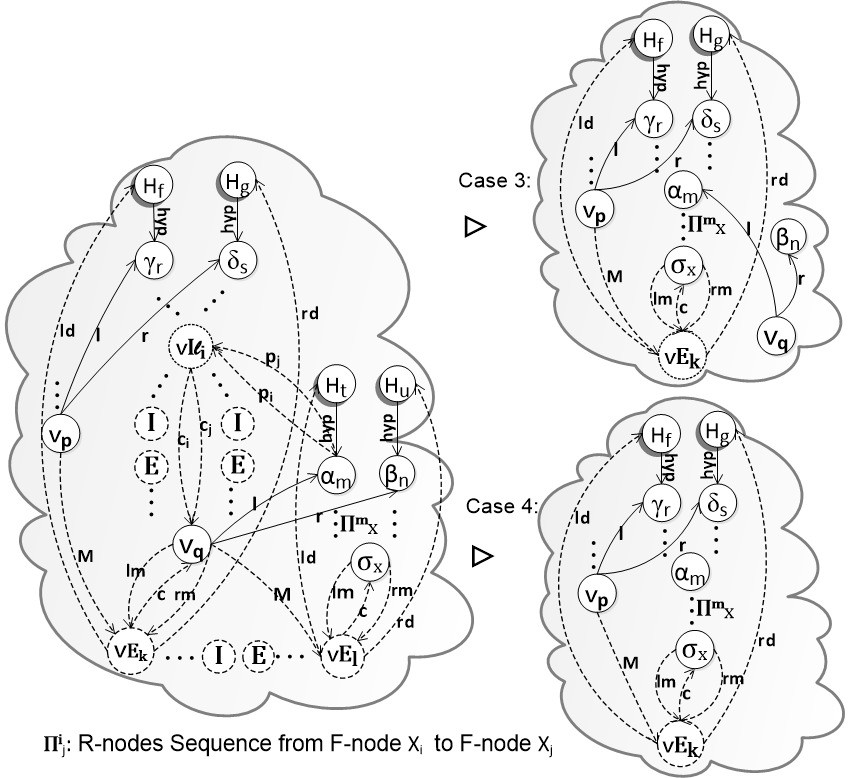


Fig. 5. Elimination of ∨*I*l followed by ∨*E*: Cases 3 and 4.

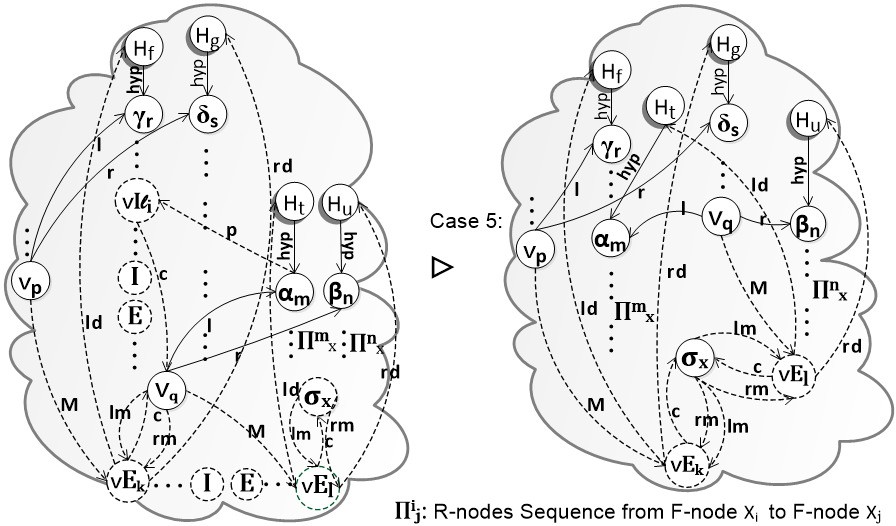


Fig. 6. Elimination of ∨*I*l followed by ∨*E*: Case 5.

* 1. *Normalization proof*

This proof is guided by the normalization measure. That is, a given fpl-graph *G* should be transformed into a non-redundant fpl-graph by applying of reduction steps and at each reduction step the measure must be decreased. The normalization

measure will be the number of maximal formulas in the fpl-graph.

**Theorem 3.6 (Normalization)** *Every fpl-graph G can be reduced to a normal fpl-graph G*′ *having the same hypotheses and conclusion as G. Moreover, for any standard tree-like natural deduction* Π*, if G* ∶= *G*Π *(the F-minimal fpl-like repre- sentation of* Π*), then the size of G*′ *does not exceed the size of G, and hence also* Π*.*

**Remark 3.7** The second assertion sharply contrasts to the well-known exponen- tial speed-up of standard normalization. Note that the latter is a consequence of the tree-like structure of standard deductions having different occurrences of equal hypotheses formulas, whereas all formulas occurring in F-minimal fpl-like represen- tations are pairwise distinct.

**Proof.** This characteristic of preservation of the premises and conclusions of the derivation is proved naturally. Through an inspection of each elimination of maxi- mal formula is observed that the reduction step (see Definition [3.4](#_bookmark14)) of the fpl-graph does not change the set of premises and conclusions (indicated by the D-nodes *H* and *C*) of the derivation that is being reduced.

In addition, the demonstration of this theorem has two primary requirements to guarantee that through the elimination of maximal formulas in the fpl-graph, one cannot generate more maximal formulas.The second requirement is to guarantee that during the normalization process, the normalization measure adopted is always reduced.

The first requirement is easily verifiable through an inspection of each case in the elimination of maximal formulas. Thus, it is observed that no case produces more maximal formulas. The second requirement is established through the normalization procedure (see Section [3.2.1](#_bookmark18)) and demonstrated through an analysis of existing cases in the elimination of maximal formulas in fpl-graphs. To support this statement, it is used the notion of normalization measure, we adopt as measure of complexity (induction parameter) the number of maximal formulas *N max*(*G*). Besides, as already mentioned, working with F-mimimal fpl-graph representations we can use as optional inductive parameter the ordinary size of fpl-graphs. □

* + 1. *Normalization Process*

We know that a specific propositional fpl-graph *G* can have one or more maximal formulas represented by *M*1*, ..., Mn*. Thus, the normalization procedure is:

* + - 1. Identify the number of maximal formulas *N max*(*G*).
      2. Choose a maximal formula represented by *Mk*.
      3. Eliminate *Mk* as defined in Definition [3.4](#_bookmark14), creating a new graph *G*.
      4. In this application one, of the following six cases may occur:
         1. The maximal formula is removed (case 1 in all eliminations of maximal formulas).
         2. The maximal formula is removed but the formula node is maintained, and,

*N max*(*G*) is decreased (case 2 in all eliminations of maximal formulas);

* + - * 1. Two maximal formula are removed (case 3 in all eliminations of maximal formulas).
        2. Two maximal formula are removed but the formula node is maintained, hence *N max*(*G*) is decreased (case 4 in all eliminations of maximal formu- las).
        3. The maximal formula is removed, the formula node is maintained and R- node sequence reordered, hence *N max*(*G*) is decreased (case 5 in all elimi- nations of maximal formulas).
        4. All maximal formulas are removed.
      1. Repeat this process until the normalization measure *N max*(*G*) is reduced to 0 and *G* becomes *a normal fpl-graph*.

Since the process of the eliminating a maximal formula on fpl-graphs always ends in the elimination of at least one maximal formula, and with the decrease in the number of vertices of the graph, we can say that this normalization theorem is directly a *strong normalization theorem*.

The following is an example illustrating the fact that exponential representation of proofs is avoided in this formalism.

**Example on Fibonacci numbers:** Consider the formulas: 1) *η* = *A*1 ➔ *A*2, and 2) *σk* = *Ak*−2 ➔ (*Ak*−1 ➔ *Ak*) for *k* > 2. Note that the size of a normal proof of *A*1 ➔ *An* from *η, σ*3*,..., σn* has size bigger than *Fibonnaci*(*n*)

|  |  |  |  |
| --- | --- | --- | --- |
| [*A*1] |  |  | [*A*1] |
| *A*1 ➔ *A*2 |  |  | *A*1 ➔ *A*2 |
| *A*1 ➔ (*A*2 ➔ *A*3) | [*A*1] *A*1 ➔ *A*2 |  | *A*1 ➔ (*A*2 ➔ *A*3) |
| Π3 | *A*2 | *A*2 ➔ (*A*3 ➔ *A*4) | Π3 |

*A*3 *A*3 ➔ *A*4 *A*3 *A*3 ➔ (*A*4 ➔ *A*5)

*A*4 *A*4 ➔ *A*5

*A*5 *A*1 ➔ *A*5

Generally, for each 5 ≤ *k* we have

[*A*1]

[*A*1] *η*

*l*(Π2)= 1

*l*(Π )= *l*(Π )+ 1

*η σ*3*,..., σk*−2 3 2

*σ*3*,..., σk*−1

Π*k*−1

Π*k*−2

*Ak*−2 *Ak*−2 ➔ (*Ak*−1 ➔ *Ak* )

*l*(Π*k*)= *l*(Π*k*−2)+ *l*(Π*k*−1)+ 2

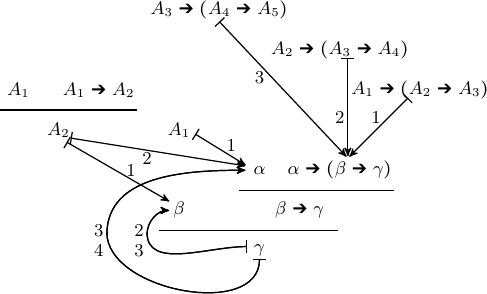
*Ak*−1 *Ak*−1 ➔ *Ak*

*Ak*

*A*1 ➔ *Ak*

*Fibonacci*(*k*)≤ *l*(Π*k*)

It is doubtful that interpolants would provide a polynomial proof for the same conclusion of this huge proof. However, using graph/dag representation it is possible to obtain a polynomial proof (cf. [[3](#_bookmark21)] and [[2](#_bookmark19)]). Now in the contexts of our present graph-representation allowing only one formula occurrence of each formula in the proof, we produce a following polynomial size graph-like proof



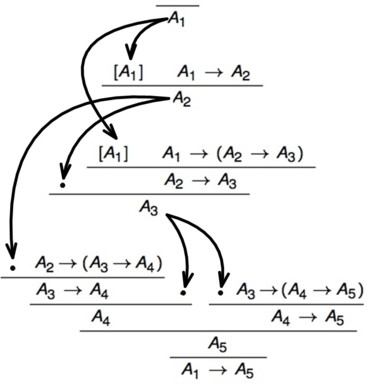
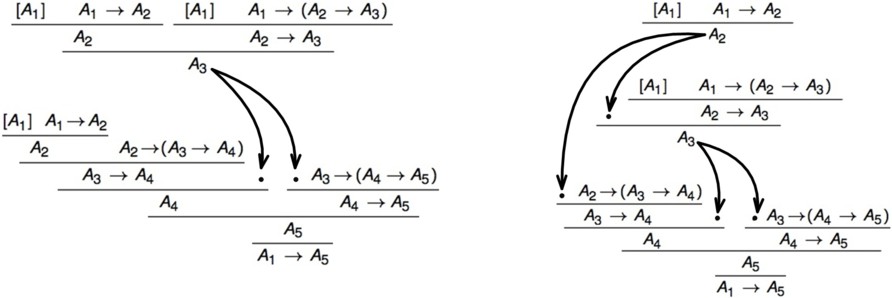
that actually can be obtained by merging distinct occurrences of identical for- mulas *A*3, *A*2, *A*1 successively as shown below.

|  |  |  |
| --- | --- | --- |
| [*A*1] |  | [*A*1] |
| *A*1 ➔ *A*2 |  | *A*1 ➔ *A*2 |
| *A*1 ➔ (*A*2 ➔ *A*3) [*A*1] *A*1 ➔ *A*2 |  | *A*1 ➔ (*A*2 ➔ *A*3) |
| Π3 *A*2 | *A*2 ➔ (*A*3 ➔ *A*4) | Π3 |

*A*3 *A*3 ➔ *A*4 *A*3 *A*3 ➔ (*A*4 ➔ *A*5)

*A*4 *A*4 ➔ *A*5

*A*5 *A*1 ➔ *A*5



This example shows that there are cases where a natural graph-like proof com- pressing allows to close exponential-size gaps between tree-like and graph-like (in

fact dag-like) proof representations [7](#_bookmark25) .

# Conclusions

The results presented for mimp-graph in [[6](#_bookmark24)] are extended for fpl-graph. Thus, fpl-graph was introduced through definitions and examples preserving the ability to represent proofs in Natural Deduction. The minimal formula representation is a key feature of the fpl-graph structure, because as we saw earlier, it is easy to determine maximal formulas and upper bounds in the length of reduction sequences to leading to normal proofs. A normalization theorem was proved by counting the number of maximal formulas in the original derivation. The strong normalization property is a direct consequence of such normalization, since any reduction decreases the corresponding measures of derivation complexity. This is a preliminary step into investigating how a theorem prover based on graphs is more efficient than usual theorem provers.

We advice the reader, that although the example on Fibonacci numbers shows an exponential gap between tree and dag-like representations, this is not the general case. There are proofs that even in dag representation have exponentially many nodes regarding the size of their conclusions.

# References

1. M. Finger. Dag sequent proofs with a substitution rule. In *We will show Them – Essays in honour of Dov Gabbay 60th birthday*, volume 1 of *Kings College Publications*, pages 671–686. Kings College, London, 2005.
2. L. Gordeev, E. H. Haeusler, and V. G. Costa. Proof compressions with circuit-structured substitutions.

*Journal of Mathematical Sciences*, 158(5):645–658, apr 2009.

1. Edward Hermann Haeusler and Vaston Gon¸calves da Costa. A discussion on compressing proofs through proof-theoretical techniques. *Christiano Braga (Ed.)*, 2012.
2. Anjolina Grisi Oliveira and Ruy J.G.B. Queiroz. Geometry of deduction via graphs of proofs. In Ruy J.G.B. Queiroz, editor, *Logic for Concurrency and Synchronisation*, volume 15 of *Trends in Logic*, pages 3–88. Springer Netherlands, 2003.
3. Marcela Quispe-Cruz. On strong normalization in proof-graphs for propositional logic. [http://www.](http://www.tecmf.inf.puc-rio.br/MarcelaCruz?action=AttachFile&do=get&target=pmimp.pdf) [tecmf.inf.puc-rio.br/MarcelaCruz?action=AttachFile&do=get&target=pmimp.pdf](http://www.tecmf.inf.puc-rio.br/MarcelaCruz?action=AttachFile&do=get&target=pmimp.pdf), 2015.
4. Marcela Quispe-Cruz, Edward Hermann Haeusler, and Lew Gordeev. Proof-graphs for minimal implicational logic. In *Proceedings 9th International Workshop on Developments in Computational Models, DCM 2013, Buenos Aires, Argentina, 26 August 2013.*, pages 16–29, 2014.

7 Some additional techniques with regard to the hypotheses discharging by ➔-intro rules are also possible, but this topic is out of the scope of our present paper