ELSEVIER

Theoretical Computer Science 288 (2002) 45–83

Theoretical Computer Science

www.elsevier.com/locate/tcs

Streams and strings in formal proofs

A. Carbone *

Mathématiques/Informatique, Université de Paris XII, 61 Avenue du Général de Gaulle, 94010 Créteil, France

Abstract

Streams are *acyclic* directed subgraphs of the logical flow graph of a proof representing bundles of paths with the same origin and the same end. The notion of stream is used to describe the evolution of proofs during cut-elimination in purely algebraic terms. The algebraic and combinatorial properties of flow graphs emerging from our analysis serve to elucidate logical phenomena. However, the full logical significance of the combinatorics, e.g. the absence of certain patterns within flow graphs, remains unclear.

Keywords: Directed graphs; Logical flow graphs; Cut elimination; Cycles in proofs; Structure of proofs; Proof complexity; Duplication

1. Introduction

The *analytical method* which divides proofs into blocks, analyses them separately and puts them together again, proved its failure: by "cutting up" it destroys what it seeks to understand, that is the dynamics within proofs [8]. This important point has been understood and emphazised by J.-Y. Girard who, in 1987, introduced *proof nets* to study proofs as *global* entities and to study the way that formulas interact in a proof through logical connectives. In 1991, another notion of graph associated to proofs has been introduced by S. Buss [1] for different purposes (namely, as a tool to show the undecidability of k-provability) and has been employed in [2, 3, 5] to study dynamics in proofs. This graph, called *logical flow graph*, traces the flow of occurrences of formulas in a proof.

It would be of much interest to characterize combinatorial patterns that might appear in proofs within different logical systems. As a first step in this direction, in Theorem 5

* Tel.: +33-01-45171647; fax: +33-01-45171649.
E-mail address: carbone@ihes.fr (A. Carbone).

0304-3975/02/$ - see front matter
PII: S0304-3975(01)00145-1

(Section 4) we exhibit a specific cyclic pattern that is necessarily missing from logical flow graphs. This type of cycles is probably prohibited by the consistency of the logic, but the full significance of this remains unclear. Also, the combinatorics and the complexity of the evolution of logical flow graphs of proofs under cut-elimination are particularly complicated and intruiging. An overview can be found in [8] and a combinatorial analysis is developed in [4]. These difficulties constitute the main reason for looking at simpler but well-defined *subgraphs* of logical flow graphs and try to study their properties and behavior in proofs.

We shall concentrate on *streams* (defined in Section 3). A stream represents a bundle of paths traversing occurrences of the same atomic formula in a proof and having the same origin and the same target. A proof is usually constituted by several streams. They interact with each other because of logical rules and share common paths because of contractions. There are cases where a bundle of paths needs to be exponentially large in size like in the propositional cut-free proofs of the pigeonhole principle for instance (this is a consequence of [14] and a formal argument is found in [6]), and the study of streams becomes relevant for the study of complexity of the sequent calculus proofs (see Theorem 35 and Remark 38).

Our interest lies on the *topological properties* of streams. We shall be concerned only with a *rough* description of logical paths in a stream. This description will be based on axioms, cuts and contraction rules occurring in the proof. Rules introducing logical connectives will not play any role. This simplified treatment of logical paths allows for a description of proofs as *strings* (Section 6), and for a natural *algebraic* manipulation of proofs (Sections 7 and 8). When logical flow graphs contain cycles, this description leads to precise relations between proofs and finitely presented groups [7]. Here, we will only look at subgraphs of the logical flow graph which are *acyclic* and we shall develop a theory which relates algebraic strings to streams. We prove that any stream can be described by an algebraic string, and that for any string there is always a proof with a logical flow graph which is a stream described by the string (Section 6). Usually, several strings can describe the same stream, but there are two kinds of strings, one representing the most *compact* description of the stream, and the other representing the most *explicit* description, that are unique and hence useful for our analysis (see Sections 6 and 6.2).

In Section 8 we show that the transformation of streams during the procedure of cut-elimination (Section 2.2) can be simulated by a finite set of *rewriting rules* (Theorem 32). The notion of stream, though simple, is shown to be very powerful at the computational level: Example 33 illustrates how, at times, a purely algebraic manipulation of streams can *completely* describe the proof transformation. Theorem 34 pinpoints how *weak* formulas in a proof influence the complexity of streams during cut-elimination. Theorem 35 says that a growth in complexity is either already "explicit" in a proof (i.e. a proof contains a stream with large *arithmetical value*; this notion is defined in Section 7) or it is due to purely global effects induced by local rules of transformation.

To conclude, let us mention that our algebraic analysis of proofs seems adequate to approach the problem of the introduction of cuts in proofs, an important topic in proof theory and automated deduction. It seems plausible that a theory of the flow of information in a proof might lead to develop methods for the *introduction* of cuts in proofs.

We wish to thank an anonymous referee for his/her insightful comments.

2. Basic notions and notation

In this section we briefly recall known concepts. The reader might like to consult [13, 19], and also [9], for a more extensive introduction.

2.1. Formal proofs

Formal proofs are described in the *sequent calculus LK*. The system *LK* is constituted by *axioms* which are sequents of the form

$$A, \Gamma \to \Delta, A$$

where A is any formula and Γ, Δ are any multisets of formulas, by *logical* rules for the introduction of logical connectives

$$\neg: \textit{left} \quad \frac{\Gamma \to \Delta, A}{\neg A, \Gamma \to \Delta} \qquad \neg: \textit{right} \quad \frac{A, \Gamma \to \Delta}{\Gamma \to \Delta, \neg A}$$

$$\wedge: \textit{right} \quad \frac{\Gamma_1 \to \Delta_1, A \quad \Gamma_2 \to \Delta_2, B}{\Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2, A \wedge B} \qquad \wedge: \textit{left} \quad \frac{A, B, \Gamma \to \Delta}{A \wedge B, \Gamma \to \Delta}$$

$$\vee: \textit{left} \quad \frac{A, \Gamma_1 \to \Delta_1 \quad B, \Gamma_2 \to \Delta_2}{A \vee B, \Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2} \qquad \vee: \textit{right} \quad \frac{\Gamma \to \Delta, A, B}{\Gamma \to \Delta, A \vee B}$$

$$\supset: \textit{left} \quad \frac{\Gamma_1 \to \Delta_1, A \quad B, \Gamma_2 \to \Delta_2}{A \supset B, \Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2} \qquad \supset: \textit{right} \quad \frac{A, \Gamma \to \Delta, B}{\Gamma \to \Delta, A \supset B}$$

$$\exists: \textit{right} \quad \frac{\Gamma \to \Delta, A(t)}{\Gamma \to \Delta, \exists x A(x)} \qquad \exists: \textit{left} \quad \frac{A(a), \Gamma \to \Delta}{\exists x A(x), \Gamma \to \Delta}$$

$$\forall: \textit{right} \quad \frac{\Gamma \to \Delta, A(a)}{\Gamma \to \Delta, \forall x A(x)} \qquad \forall: \textit{left} \quad \frac{A(t), \Gamma \to \Delta}{\forall x A(x), \Gamma \to \Delta}$$

where, in the $\exists: \textit{left}$ and $\forall: \textit{right}$ inferences the free variable b is called the *eigenvariable* and must not appear in the lower sequent. The variable x must be freely substitutable into A for all four quantifier inferences.

Finally, there are a few rules in *LK* that do not explicitly involve connectives. They are called *structural rules*:

$$\textit{Cut} \qquad \frac{\Gamma_1 \to \Delta_1, A \quad A, \Gamma_2 \to \Delta_2}{\Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2}$$

$$\textit{Contraction} \qquad \frac{\Gamma \to \Delta, A, A}{\Gamma \to \Delta, A} \qquad\qquad \frac{A, A, \Gamma \to \Delta}{A, \Gamma \to \Delta}$$

In our notation, a rule is always denoted by a bar. The sequent(s) above the bar is called *antecedent* of the rule and the sequent below the bar is called *consequent*. The pair of formulas A in the upper sequent of the contraction rule are called *contraction formulas*. In logical and structural rules, the formulas A (B) in the upper sequent(s) are referred to as *auxiliary* formulas and the one obtained by the application of a rule is called *main* formula. Notice that the cut-rule has no main formula. In axioms and rules, the formulas in Γ, Δ are called *side* formulas. In an axiom $A, \Gamma \to \Delta, A$, the two formulas A are called *distinguished* formulas. A formula A in Π which has been introduced as a side formula in some axiom is called *weak* formula.

We shall extend *LK* with the rule

$$F\textit{-rule} \qquad \frac{\Gamma_1 \to \Delta_1, F(s) \quad \Gamma_2 \to \Delta_2, F(t)}{\Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2, F(s \cdot t)}$$

where F is a unary predicate and $\cdot$ is a binary function. The F-rule is added to *LK* because it allows to speak more directly about *computations*. It was considered already in [4, 5, 9, 10].

In the following we will frequently use the notion of *occurrence* of a formula in a proof as compared to the formula itself which may occur many times. A formula B occurs *positively* (*negatively*) in a *formula* A if it appears under the scope of an even (odd) number, possibly 0, of negations. A formula B occurs *positively* (*negatively*) in a *sequent* if it occurs positively (negatively) in some formula of the consequent of the sequent, or it occurs negatively (positively) in some formula of the antecedent of the sequent. While counting the number of nested negations for a formula, one should take into account negations "hidden" in front of the antecedent of implications. Notice that $A \supset B$ is equivalent to $\neg A \vee B$ and that A appears negatively in $A \supset B$.

A formal proof is a *binary tree* of sequents, where each occurrence of a sequent in a proof can be used at most once as premise of a rule. The root of the tree is labelled by the theorem, its leaves are labelled by axioms and its internal nodes are labelled by sequents derived from one or two sequents (which label the antecedents of the node in the tree) through the rules of *LK* and the F-rule.

The *height of a rule* R in a proof Π is defined over the proof-tree describing Π. It is the distance between the node of the proof-tree labelled by the consequent of R and the root of the proof-tree.

At times we shall consider proofs Π which are *reduced* in the sense of [4], i.e. there are no superfluous redundancies in the proof which have been built with the help of weak occurrences. More formally, no binary rule or contraction rule is applied to a weak formula, no unary logical rule is applied to two weak formulas and no occurrence in cut-formulas is weak. In [4] it is shown that given any proof, we can always find a *reduced* proof of the same end-sequent which has a number of lines and symbols bounded by the ones of the original proof.

2.2. Cut-elimination

In 1934 Gentzen [11] proved the following result:

Any proof in LK can be effectively transformed into a proof which never uses the cut-rule. This works for both propositional and predicate logic.

The statement holds for the extension of LK with the F-rule as well.

This is a fundamental result in proof theory, and in [8] the reader can find a presentation of its motivations and consequences. The computational aspects of the theorem have been largely investigated but we are still far from an understanding of the dynamical process which can occur within proofs [2, 3, 5]. After the elimination of cuts, the resulting proof may have to be much larger than the proof with cuts. For propositional proofs, this expansion might be *exponential* and for proofs with quantifiers, it can be *multi-exponential*, i.e. an exponential tower of 2's. For the bounds, the reader can consult [16–18, 20].

The *contraction rule* plays a key role in this expansion. While it may seem harmless enough, it can be very powerful in connection with the cut rule. Imagine that you have some piece of information which is represented by a formula A which you can prove, but that in your reasoning you actually need to use this piece of information twice. By using a contraction rule (on the left hand side of the sequent arrow) and a cut you can code your argument in such a way that you only need to verify A once. On the other hand, a cut-free proof represents 'direct' reasoning, where lemmas are not allowed, and in practice this forces one to duplicate the proof of A (see [8] for more details).

Thus the cut rule provides a mechanism by which the contraction rule can have the effect of a "duplication" rule.

Let us look more closely at how the procedure introduced by Gentzen works. There are distinguished cases and different recipes that we need to follow depending on the structure of the cut formula and whether it came from a contraction. The idea is to push the cuts up towards the axioms and eliminate them afterwards.

Let us begin by considering the case of a cut applied over a formula which comes directly from an axiom, either as a distinguished occurrence or as a weak occurrence. Consider first the situation where the cut formula comes from a distinguished occurrence

in an axiom, as in the following:

$$\dfrac{\Gamma_1, A \to A, \Delta_1 \quad \overset{\Pi_*}{A, \Gamma_2 \to \Delta_2}}{\Gamma_1, A, \Gamma_2 \to \Delta_1, \Delta_2} \tag{1}$$

In this case we can remove the axiom from the proof and simply add the weak occurrences in Γ_1 and Δ_1 to the subproof Π_* without trouble, thereby obtaining a new proof of the sequent $\Gamma_1, A, \Gamma_2 \to \Delta_1, \Delta_2$ in which the last cut has been eliminated.

Suppose instead that we have a cut over a formula which comes from a weak occurrence in an axiom, as in the following situation:

$$\dfrac{\Gamma_1, A \to A, \Delta_1, C \quad \overset{\Pi_0}{C, \Gamma_2 \to \Delta_2}}{\Gamma_1, A, \Gamma_2 \to A, \Delta_1, \Delta_2} \tag{2}$$

To eliminate the cut one can simply eliminate the subproof Π_0, take out the (weak) occurrence of C in the axiom, and add Γ_2 and Δ_2 to the axiom as weak occurrences. In other words, the sequent

$$\Gamma_1, \Gamma_2, A \to A, \Delta_1, \Delta_2 \tag{3}$$

is itself an axiom already. By doing this one *removes* a possibly large part of the proof.

If the two cut-formulas have been introduced by logical rules the procedure will substitute the cut with two new ones of smaller logical complexity. The idea is illustrated in the following proof:

$$\begin{array}{c} \dfrac{\dfrac{\overset{\Pi_1'}{\Gamma_1 \to \Delta_1, A, B}}{\Gamma_1 \to \Delta_1, A \vee B} \quad \dfrac{\overset{\Pi_2'}{A, \Gamma_2 \to \Delta_2} \quad \overset{\Pi_3'}{B, \Gamma_3 \to \Delta_3}}{A \vee B, \Gamma_{2,3} \to \Delta_{2,3}}}{\Gamma_{1,2,3} \to \Delta_{1,2,3}} \\ \vdots\lambda \end{array} \tag{4}$$

where the cut-elimination procedure will reduce the complexity of $A \vee B$ as follows:

$$\begin{array}{c} \dfrac{\dfrac{\overset{\Pi_1'}{\Gamma_1 \to \Delta_1, A, B} \quad \overset{\Pi_2'}{A, \Gamma_2 \to \Delta_2}}{\Gamma_{1,2} \to \Delta_{1,2}, B} \quad \overset{\Pi_3'}{B, \Gamma_3 \to \Delta_3}}{\Gamma_{1,2,3} \to \Delta_{1,2,3}} \\ \vdots\lambda \end{array} \tag{5}$$

The other connectives are treated similarly. In case the cut formula is quantified the procedure proceeds in a similar manner. The following proof

$$\dfrac{\dfrac{\overset{\Pi_1}{\Gamma_1 \to \Delta_1, A(t)}}{\Gamma_1 \to \Delta_1, \exists x.A(x)} \quad \dfrac{\overset{\Pi_2}{A(a), \Gamma_2 \to \Delta_2}}{\exists x.A(x), \Gamma_2 \to \Delta_2}}{\Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2} \tag{6}$$

will be transformed into

$$\dfrac{\begin{array}{cc}\Pi_1 & \Pi_2' \\ \Gamma_1 \to \Delta_1, A(t) & A(t), \Gamma_2 \to \Delta_2\end{array}}{\Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2} \tag{7}$$

where the proof Π_2' is obtained by substituting all occurrences of the eigenvariable a in Π_2 with the term t.

We now consider the case of contractions. The following diagram shows the basic problem:

$$\dfrac{\begin{array}{cc}\Pi_1 & \dfrac{\begin{array}{c}\Pi_2 \\ A^1, A^2, \Gamma_2 \to \Delta_2\end{array}}{A, \Gamma_2 \to \Delta_2} \\ \Gamma_1 \to \Delta_1, A & \end{array}}{\Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2} \tag{8}$$

That is, A^1 and A^2 denote two occurrences of the same formula A, and they are contracted into a single occurrence before the cut is applied. (The contraction could just as well be on the left, and this would be treated in the same way.) To push the cut above the contraction one duplicates the subproof Π_1 as indicated below:

$$\dfrac{\begin{array}{cc}\Pi_1 & \dfrac{\begin{array}{cc}\Pi_1 & \Pi_2 \\ \Gamma_1 \to \Delta_1, A & A^1, A^2, \Gamma_2 \to \Delta_2\end{array}}{A^2, \Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2} \\ \Gamma_1 \to \Delta_1, A & \end{array}}{\begin{array}{c}\Gamma_1, \Gamma_1, \Gamma_2 \to \Delta_1, \Delta_1, \Delta_2 \\ \vdots \ \textit{contractions} \\ \Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2\end{array}} \tag{9}$$

The steps of transformation we described are essentially all what one needs. To make them to work though, one needs to change the order of the rules in a proof. In the cases considered above, the cut rule was applied to formulas which were main formulas, i.e. their principal connective was introduced by the immediate preceding rule. This configuration is in general not there and one needs to change the order of the rules in the proof by pushing cuts upwards until a pair of cut formulas which are main formulas has been reached; this can always be done as the following diagrams illustrate:

$$\dfrac{\begin{array}{cc}\Pi_1 & \dfrac{\begin{array}{cc}\Pi_2 & \Pi_3 \\ A, C, \Gamma_2 \to \Delta_2 & B, \Gamma_3 \to \Delta_3\end{array}}{C, A \vee B, \Gamma_2, \Gamma_3 \to \Delta_2, \Delta_3} \\ \Gamma_1 \to \Delta_1, C & \end{array}}{A \vee B, \Gamma_1, \Gamma_2, \Gamma_3 \to \Delta_1, \Delta_2, \Delta_3} \tag{10}$$

which will be transformed into

$$\dfrac{\begin{array}{cc}\Pi_1 & \dfrac{\begin{array}{cc}\Pi_2 & \\ C, A, \Gamma_2 \to \Delta_2 & \end{array}}{A, \Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2} \\ \Gamma_1 \to \Delta_1, C & \end{array} \quad \begin{array}{c}\Pi_3 \\ B, \Gamma_3 \to \Delta_3\end{array}}{A \vee B, \Gamma_1, \Gamma_2, \Gamma_3 \to \Delta_1, \Delta_2, \Delta_3} \tag{11}$$

One might be interested to permute cuts and in this case the procedure transforms the diagram

$$\dfrac{\dfrac{\begin{matrix}\Pi_3\\ \Gamma_3 \to \Delta_3, D^1\end{matrix} \qquad \dfrac{\begin{matrix}\Pi_1\\ D^2, \Gamma_1 \to \Delta_1, C^1\end{matrix} \quad \begin{matrix}\Pi_2\\ C^2, \Gamma_2 \to \Delta_2\end{matrix}}{D^2, \Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2}}{\Gamma_1, \Gamma_2, \Gamma_3 \to \Delta_1, \Delta_2, \Delta_3} \tag{12}$$

into the following one:

$$\dfrac{\dfrac{\begin{matrix}\Pi_3\\ \Gamma_3 \to \Delta_3, D^1\end{matrix} \quad \dfrac{\begin{matrix}\Pi_1\\ D^2, \Gamma_1 \to \Delta_1, C^1\end{matrix}}{\Gamma_1, \Gamma_3 \to \Delta_1, \Delta_3, C^1} \quad \begin{matrix}\Pi_2\\ C^2, \Gamma_2 \to \Delta_2\end{matrix}}{\Gamma_1, \Gamma_2, \Gamma_3 \to \Delta_1, \Delta_2, \Delta_3} \tag{13}$$

or vice versa.

It is important to point out a fundamental feature of cut-elimination: there is no canonical way to do it. In the passage from (4) to (5) we could have chosen to cut first B and then A instead of the other way around. In the passage from (8) to (9), if both appearances of the cut formula A in (8) were obtained from contractions, then we would have a choice as to which subproof duplicate first (either Π_1 or Π_2). In principle, we can have procedures of cut-elimination which go on forever. Of course, the point of the theorem is that one can always find a way to eliminate cuts in a finite number of steps. One can even make deterministic procedures by imposing conditions on the manner in which the transformations are carried out. See [12].

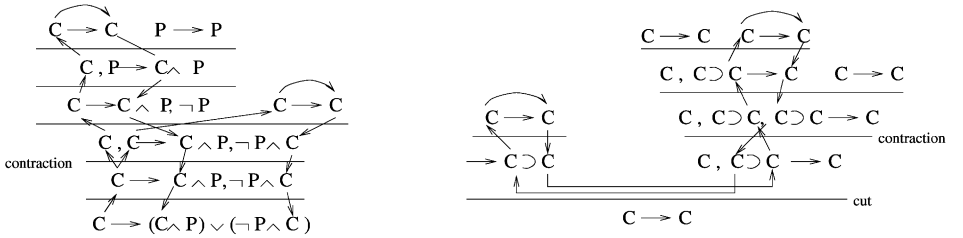
In essence, the principle of the procedure is to push the cuts up higher in the proof while being careful about the notion of "progress". In fact, we typically increase the number of cuts at each stage of the process as well as the number of contractions. In the case of contractions we have made progress in the sense that we reduced the number of contractions above the cut-formula, even though we may increase the total number of contractions by adding them below the cut. In the case of conjunctions we reduced the complexity of the cut-formula. It is not hard to exhaust the possibilities, but a complete proof requires a tedious verification of cases and we shall not provide it. See [13, 19].

Besides the method of cut-elimination proposed by Gentzen, everal others methods have been proposed in the literature but we shall not discuss them here. The ideas and results presented in this paper will be formulated for Gentzen's procedure. Extensions and adaptations to different logical frameworks are conceivable.

2.3. Logical flow graphs

As described in [2], one can associate to a given proof a logical flow graph by tracing the flow of *atomic* occurrences in it. (The notion of logical flow graph was

first introduced by Buss in [1] and a similar notion is due to Girard and appeared in [12]. Here we restrict Buss' notion to *atomic* formulas.) We give the formal definition later and we start by illustrating the idea with an example. Consider the two formal proofs below formalized in the language of propositional logic and the sequences of edges that one can trace through them



Each step of deduction manipulates formulas following a logically justified rule, and precise links between the formulas involved in the logical step are traced (the arrows indicated in the figures above represent *some* of these links). Formulas in a proof correspond to nodes in the graph and logical links induced by rules and axioms correspond to edges. As a side effect different occurrences of a formula in a proof might be logically linked even if their position in the proof is apparently very far apart. Between any two logically linked occurrences there is a path. The graph that we obtain is in general disconnected and each connected component corresponds to a different atomic formula in the proof.

The structure of the proof on the left is interesting because it shows that paths in a proof can get together through contraction of formulas, and the structure on the right shows that cyclic paths might be formed.

Let us now detail the formal definition of *logical flow graph*.

For each axiom, we trace an edge, called *axiom-edge*, from any negative atomic occurrence in the distinguished formula of the antecedent to the corresponding positive atomic occurrence in the distinguished formula of the consequent. We also trace an edge from any negative atomic occurrence of the distinguished formula in the consequent of the axiom to the corresponding positive atomic occurrence in the distinguished formula of its antecedent.

For each rule, we trace an edge between any positive occurrence of a formula B in the upper sequent(s) of the rule and the corresponding occurrence of B in the lower sequent of the rule. Similarly, we trace an edge between any negative occurrence of a formula B in the lower sequent of the rule and the corresponding occurrence of B in the upper sequent(s) of the rule. (For quantifiers rules, the formula that corresponds to $A(a)$ or $A(t)$ is $A(x)$.)

If the rule is a contraction, then there are two edges going to (coming from) the negative (positive) occurrences of B in the contraction formulas and coming from (going to) the relative occurrence of B in the main formula.

In the case of a cut rule applied to sequents $\Gamma_1 \to \Delta_1, A$ and $A, \Gamma_2 \to \Delta_2$, edges, called *cut-edges*, are traced between the two cut-formulas A as follows: for any subformula B of A occurring positively (negatively) in $\Gamma_1 \to \Delta_1, A$, there is an edge going from (coming to) it to (from) the correspondent occurrence of B in A of $A, \Gamma_2 \to \Delta_2$.

This concludes the definition of the edges of the graph. We say that the *logical flow graph* of a proof Π is the directed graph which we can read off the proof, whose nodes are labelled by the occurrences of formulas in Π, and whose edges are the links induced by the rules of Π, as defined above.

The orientation on the edges of a logical flow graph is induced by natural considerations on the *validity* of the rules of inference which we shall not discuss here (see [2]). In the following we will not really exploit the direction of the paths. We will use directions only to establish that a path starts and ends somewhere. We might speak of a path going up or down, and of an edge being horizontal, in case the edge appears in an axiom or between cut-formulas.

In the sequel, we call *bridge* any maximal oriented path that starts from a negative occurrence, ends in a positive occurrence and does not traverse cut-edges. The maximality condition implies that both the starting and ending occurrences of a bridge should lie either in a cut-formula or in the end-sequent of the proof. Two bridges are *nested* when they share the input node and the output node. (Nested bridges are formed by means of contractions.)

To conclude we shall introduce some terminology that concerns oriented graphs with nodes of degree at most 3. (We count here, both incoming and outcoming edges.) Logical flow graphs are graphs of this kind.

A node of a graph is a *branching point* if it has exactly three edges attached to it. We say that a node is a *focusing* branching point if there are two edges oriented towards it. A node is a *defocusing* branching point if the two edges are oriented away from it. A node is an *input vertex* if there are no edges in the graph which are oriented towards it. A node is an *output vertex* if there are no edges in the graph which are oriented away from it. Input and output nodes are called *extremal*.

By a *focal pair* we mean an ordered pair (u, w) of vertices in the graph for which there is a pair of distinct paths from u to w. We also require that these paths arrive at w along different edges flowing into w.

Two graphs have the same *topological structure* if they can both be reduced to the same graph by collapsing each edge between pairs of points of degree at most 2 to a vertex. Such a graph will be called *topologically minimal graph*.

The notions of bridge, focal pairs, topological structure, focusing and defocusing point, input and output vertices have been introduced in [4, 10] where the reader can find properties and intuition. Here, let us just say that in a proof, branching points of the logical flow graph correspond to formulas obtained by contraction or by F-rule. Also, extremal points correspond either to weak occurrences or to occurrences of formulas in the end-sequent.

3. Streams

A *stream* is an acyclic directed graph with *one* input vertex v and *one* output vertex w. The pair $[v,w]$ is called *base* of the stream. All other vertices in the stream are *not* extremal.

If G is a directed graph, then a *stream in* G is a subgraph of G which is a stream. A *full* stream in G based on $[v,w]$ is a stream in G such that all directed paths lying in G between v and w belong to the stream.

A *substream* of a stream is a subgraph of the stream that is a stream based on a pair $[v,w]$, where v,w are nodes of the stream. A substream might have the same base of the stream, but this is not required.

A *stream of a proof* Π is a stream in the logical flow graph G of Π such that the input and output vertices are extremes of G. A stream of a proof Π is based on a pair of formulas $[A,B]$, where each formula A and B lies either in the end-sequent of Π or it is a weak occurrence in an axiom of Π.

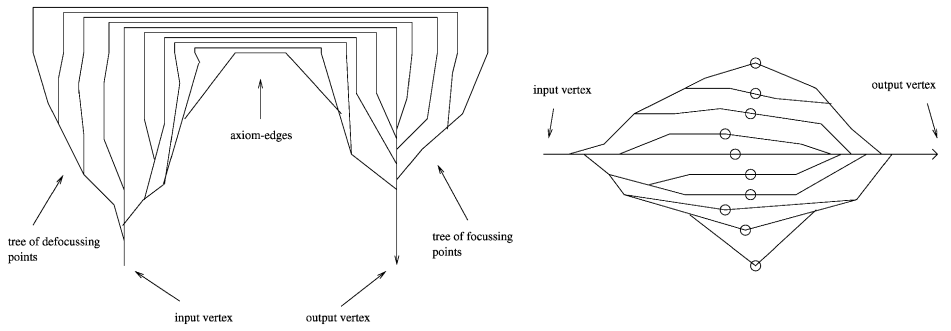
The logical flow graph of the proof on the right hand side of the picture in Section 2.3 is a stream based on the two occurrences C in the end-sequent. The bridge along the formula P in the proof on the left, is also a stream. There are streams of proofs whose base does not lie in the end-sequent. Two examples are given below

$$\dfrac{\dfrac{\dfrac{\dfrac{A\to A}{\to \neg A, A}\quad B\to B,A}{B\to B, A\wedge\neg A, A}\quad B\to B,\neg A}{B,B\to B,B,A\wedge\neg A, A\wedge\neg A}\qquad \dfrac{\dfrac{A\to A}{A,\neg A\to}}{A\wedge\neg A\to}}{B\to B,A\wedge\neg A}}{B\to B}$$

$$\dfrac{\dfrac{\dfrac{B\to B\quad C\to C}{A\to A,B\quad B,C\to B\wedge C}}{A,C\to B\wedge C,A}}{A\wedge C\to B\wedge C,A}}{A\wedge C\to (B\wedge C)\vee A}$$

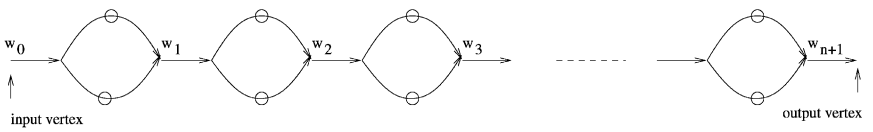
where the proof on the left displays a cycle that passes through occurrences of A, and where no A appears in the end-sequent (this example is taken from [6]). In the proof on the right, the path passing through the occurrences of B is a stream of the proof. It originates in a weak occurrence and it ends-up in the end-sequent.

The simplest forms of streams in a *cut-free* proof are either directed paths connecting a weak occurrence in an axiom to some formula in the end-sequent, or they are bridges. But usually, streams in cut-free proofs look roughly as in the picture below



where one sees (in the figure on the left-hand side) a tree of defocusing points (on the left) and a tree of focusing points (on the right) combined together through axiom edges. Nodes and edges are not explicitly denoted in the figure. Instead, oriented *paths* from the input vertex to the output vertex of the stream are drawn. These paths correspond to bridges, the bifurcation points correspond to the presence of contractions or F-rules in a proof, the horizontal lines correspond to axiom-edges through which each path has to pass. The paths are oriented from left to right, and all bifurcation points on the left hand side of the axiom-edges correspond to contractions on negative occurrences and the bifurcation points on the right correspond to contractions on positive occurrences or applications of F-rules. On the right-hand side of the figure, the same graph is illustrated in a more convenient form for our future discussion. Axiom-edges have been substituted by circles. We shall be free from now on to illustrate streams by stretching them in this way.

If the proof contains *cuts*, a stream might or might not contain cut-edges. If it *does not* contain cut-edges then its shape is one of those discussed for cut-free proofs. If it *does* contain cut-edges then its shape might be quite complicated. For instance, it might contain arbitrarily long *chains* of *focal pairs* $\{(w_i, w_{i+1})\}_{i=0}^{n}$ as illustrated by the following figure:

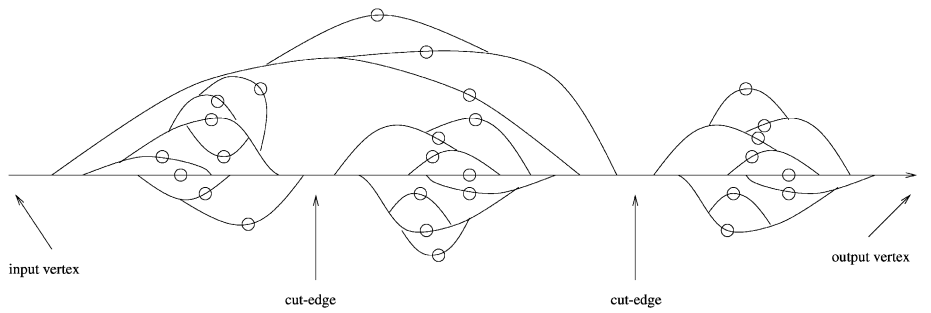


where (w_i, w_{i+1}) is a focal pair, for $i = 0, \ldots, n$, each edge passing through a circle represents a bridge, and each horizontal edge (linking a focusing point to a defocusing one) corresponds to a cut-edge in the proof. (See [4] for lower and upper bounds on proof complexity based on the presence of focal pairs in proofs.)

In general, the paths of a stream lying in a logical flow graph of a *proof with cuts* have to satisfy the following two properties:

1. if a defocusing point is followed by a focusing point, then there is an axiom-edge lying between them,
2. if a focusing point is followed by a defocusing point, then there is a cut-edge lying between them.

An example of stream for a proof with cuts is illustrated below



where one can recognize three substreams associated to cut-free proofs. The shape of these substreams is similar to the one illustrated in the first picture of the section where a tree of defocusing points is followed by a tree of focusing ones.

The following property of streams illustrates their regularity.

Proposition 1. *The number of focusing points in a stream is the same as the number of its defocusing points.*

Proof. The claim follows from a simple fact. Let P be an acyclic directed graph which is connected, contains one input vertex, n focusing points and m defocusing ones. Then, P has $m - n + 1$ output nodes. This is easily proved by induction on the values of n, m by noticing that a defocusing node induces the number of distinct output nodes to increase by 1, and a focusing node induces the number of distinct output nodes to decrease by 1. From this we conclude that if P is a stream, then $m = n$, since P has exactly one input vertex and one output vertex. □

Remark 2. If a proof contains cuts, then its logical flow graph might contain *cyclic* paths [2]. Given a logical flow graph G of a proof and a pair of formulas A, B lying in the end-sequent of the proof, there might be no *full* stream in G based on $[A, B]$, even if there is some path linking A to B in G. This is because G might contain some cyclic directed path from A to B, and streams, by definition, are acyclic graphs.

Notice that there are provable formulas whose proofs have "small" size only when cycles appear in their logical flow graph. The use of quantifiers in these proofs allows for the codification of term substitution through cyclic paths and for a *multi-exponential* compression of the size of the proof [5]. In the context of propositional logic, the situation is different. In fact, any propositional proof with a cyclic logical flow graph can be transformed into a proof with acyclic logical flow graph by a *polynomial* expansion of the size [6]. The reader interested in cycles in proofs can refer to [3, 4, 7] for an analysis of their combinatorics and complexity.

4. Clusters and interaction of streams in proofs

A logical flow graph of a proof is a union of connected *components*. There might be many of them, and each component corresponds to a distinguished atomic formula in the proof. For instance, the proof on the left-hand side in Section 2.3 displays two connected components; one corresponds to the atomic formula C and the other to P. The proof on the right-hand side displays only one connected component.

Theorem 3. *Each connected component C of a logical flow graph G is a directed graph which has input nodes as well as output nodes. In particular, for any cycle in C there is a path in G that starts at an input node of G, ends at an output node of G and passes through C.*

The last part of the theorem emphasizes that there are no "isolated" parts in a logical flow graph of a proof. Namely, for any node x of the graph, there is a path that starts from some input node of G and ends in x, *and* there is a path that starts in x and ends in some output node of G.

To show the statement, let us add some terminology. We say that a *loop* is a cycle which does not pass over the same edge twice. We say that a loop *meets* another loop (or equivalently that two loops are *nested*) if they have one vertex in common. A *cluster* of nested loops is a maximal connected subgraph of a cyclic graph which has no input node, no output node, and where there is a directed path from any point to any other point. A loop in a logical flow graph has a *way in* if one of its vertices is a focusing point. A loop has a *way out* if one of its vertices is a defocusing point. Also, a cluster has a *way in* if one of its vertices is a focusing point with an incoming edge that does not belong to the cluster. We call this focusing point a *way-in-node*. A cluster has a *way out* if one of its vertices is a defocusing point with an outgoing edge that does not belong to the cluster. We call such a point *way-out-node*.

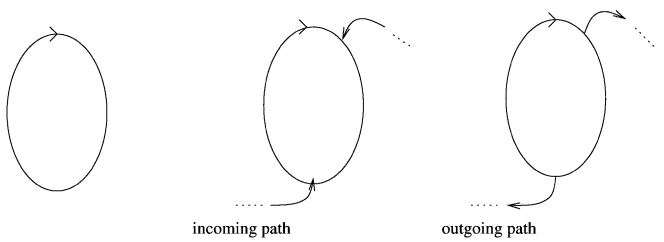
The following result holds

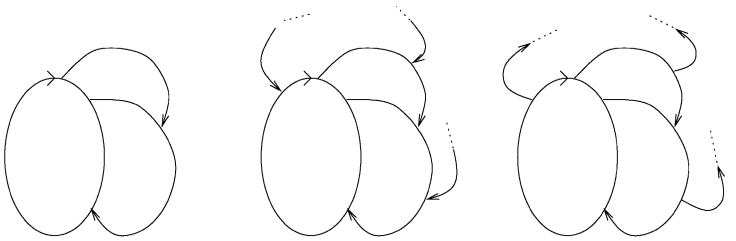
Theorem 4 (Carbone [4, Theorem 54]). *Any loop in a logical flow graph has a way in and a way out.*

To show Theorem 3 we shall prove a generalization of Theorem 4.

Theorem 5. *Any cluster in a logical flow graph has a way in and a way out.*

Proof. Let Π be a proof. We want to prove that connected components in the logical flow graph of Π cannot be loops, nor nested loops, nor they contain a cluster with incoming edges and no outgoing ones, nor they contain a cluster with outgoing edges and no incoming ones. These possibilities are illustrated in the following picture:





where the figures in the first row represent single loops, which are either the connected component itself (on the left), or a *part* of a larger component that is connected to the loop only by incoming edges or only by outgoing ones (these two cases correspond to the last two figures on the right). The figures in the second row illustrate analogous situations for clusters. The figure on the left represents a cluster with no input and output nodes, and the last two figures illustrate clusters as part of a larger component. The clusters are connected to the rest of the component only by incoming edges or only by outgoing ones.

Let us suppose that the logical flow graph of Π contains one of the connected components illustrated above. Call it K.

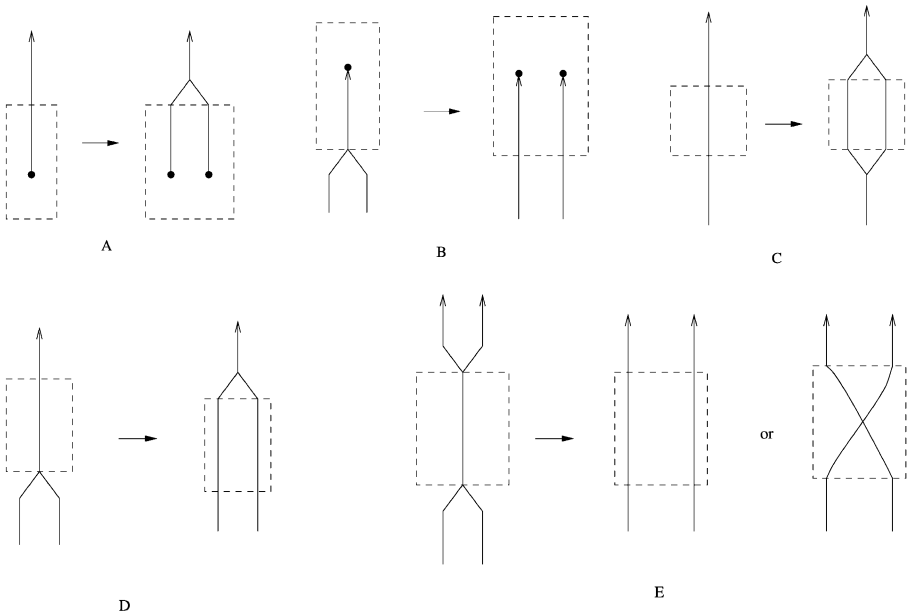
If the cluster in K is formed by only one loop we apply Theorem 4 and we obtain a contradiction. If the cluster has several loops as in the shapes illustrated in the second row of the picture above, the situation is more subtle. In fact, each loop has incoming edges and outgoing ones, but these edges might belong to some other loop and we cannot apply Theorem 4 to derive the claim.

Let us suppose that the cluster in K has no way out. The case where the cluster has no way in is treated similarly, and the case where there are no ways in nor ways out is a subcase of the one we treat.

We apply the procedure of cut-elimination to Π but only on cut-formulas that are *not* weak occurrences in some axiom. The result of such a process will be a proof Π' where cuts (if any) are *all* applied to weak formulas, and whose logical flow graph is acyclic. The procedure will transform a proof Π_i into a proof Π_{i+1} where $i = 1,\ldots,n$, Π_1 is Π and Π_n is Π'. The important property of such a transformation is that the topology of the logical flow graphs of the Π_i's changes only when contractions are resolved (going from Π_i to Π_{i+1}), that is when subproofs are duplicated. (This is discussed in [4].) This means that any connected component K_i of the logical flow graph of Π_i might be transformed into a connected component K_{i+1} in Π_{i+1}, where the topology of K_{i+1} might be different from the topology of K_i. In particular, the topological structure of the cluster K might evolve during the process of cut-elimination.

We want to prove that for all j, where $1 \leqslant j \leqslant n$, there is a cluster in Π_j with no ways out. This is in contradiction with the fact that the logical flow graph of Π' $(= \Pi_n)$ is acyclic.

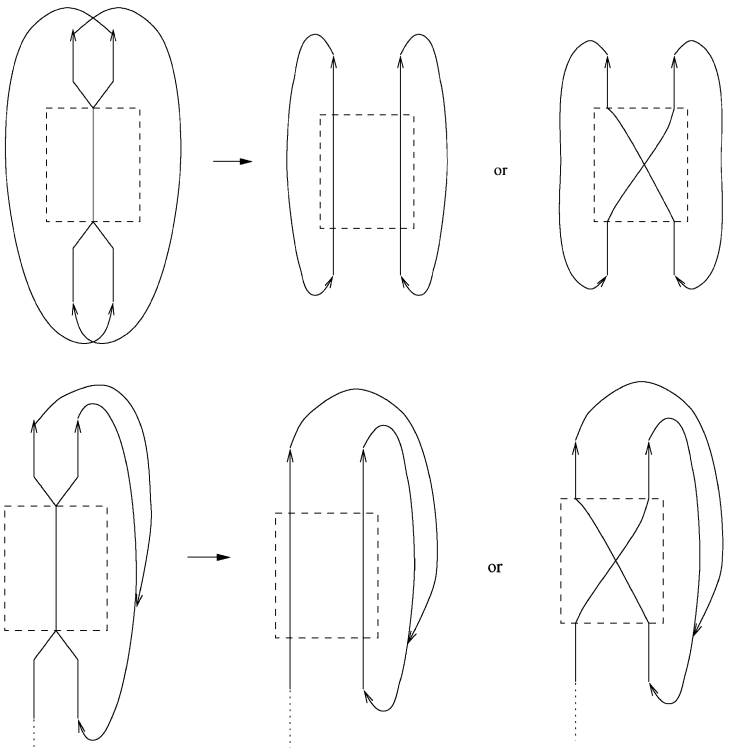
Our analysis will proceed with an explicit checking of the way that paths (in the logical flow graph of Π_i) split and duplicate during the transformation from Π_i to Π_{i+1}. As explained in [4], the duplication of a subproof of Π_i corresponds to the duplication of a subgraph of the logical flow graph of Π_i. In particular, a path passing through Π_i can be duplicated in one of the following five ways:



where black dots represent extremal nodes in the component, and where edges are not explicitly indicated in the picture. Each one of the five situations (i.e. A–E) represents the path before (on the left of the arrow) and after (on the right) duplication. The dotted boxes that are placed around portions of paths illustrate the part to be duplicated (on the left) and the part that has been duplicated (on the right).

Case D moves the position of focusing points in the graph, and case C might create new nested loops (as well as new nested bridges). This means that the number of loops in a cluster might augment. Also, new input and output nodes of the component can be created as in A and B. This implies that the number of extremes in a connected component, which are linked by a directed path to the cluster, might augment. Also, the number of ways in and ways out to a cluster might augment by duplication. An important point, is that this number can *augment* if it is *non-zero*. If there is no way in the cluster, then no way in can be created by duplication, and the same holds for ways out. This follows from the fact that duplication does not allow neither the identification of two distinct nodes of the logical flow graph nor the identification of a sequence of edges. Since no input and output nodes of a connected component can be identified by duplication, two distinct paths of the logical flow graph cannot be glued one after the other.

The duplication induced by E allows cycles to split and the number of cycles in the cluster to *decrease* eventually. The basic cases are illustrated in the figure below



where, in the upper row, we suppose that two loops of the cluster share a common part and we see that the duplication induces either the formation of two distinct loops or the constitution of a single loop. In the lower row we consider the case of two nested loops with a way in. After duplication the result is a single loop where the path going in is extended. Also, the absence of ways out is preserved in both the cases illustrated in the picture.

We can conclude that duplication can transform a cluster with no way out into a cluster with no way out. (To be precise, this follows by combining the above with Proposition 18 in [4] saying that, if the cluster is a single loop, then duplication cannot disrupt the cycle.) In particular, the process of cut-elimination that we are considering, transforms a logical flow graph containing a cluster with no way out into a logical flow graph containing a cluster with no way out. This implies that the cluster contained in K can be reduced by duplication to a cluster with no way out and with a possibly smaller number of cycles, but it cannot be eliminated.

Since our procedure of cut-elimination transforms Π into a proof Π' with an *acyclic* logical flow graph, then we are in contradiction with the hypothesis that there is a cluster in the logical flow graph of Π with no way out. □

Proof (*Theorem* 3). Each connected component K of the logical flow graph of a proof Π is a *directed* graph because it is, by definition, a subgraph of the logical flow graph.

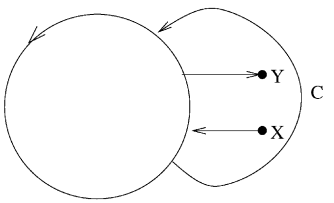
When no cycles appear in K, then the statement is obvious.

If K is cyclic, then let C be a cycle in K and let H be the cluster in K that contains C. By Theorem 5, there is a way in and a way out in H. Also, by definition of cluster, either there is an acyclic path that starts from an input node of K and goes to the way-in-node of H (without passing through other clusters of K), or there is an acyclic path that starts from a way-out-node of some cluster in K (disjoint by H) and goes to the way-in-node of H. If the first case holds, then we call p_1 such a path. Otherwise, since K is finite, there is an input node x of K and a sequence of disjoint clusters in K such that there is an acyclic path starting at x, passing through the clusters in the sequence and ending in the way-in-node of H. (By definition of cluster, the path must pass exactly once through the clusters in the sequence.) Call p_1 this path.

With a similar argument, we can show that there is an acyclic path p_2 from a way-out-node of H to some output node of K.

The paths p_1, p_2 allow to conclude that there is a path in K that starts at an input node of K, ends in an output node of K and passes through C. To find such a path it is enough to extend p_1 with a path in H that starts at the way-in-node of H, passes through C, ends in the way-out-node of H and proceeds as p_2. □

The path that passes through the cycle C in the proof of Theorem 5 might be cyclic. Take for instance the following figure:



where C is the most external cycle in the picture. It is clear that there is no acyclic path going from X to Y and passing through the loop C.

As mentioned in Remark 2 there might be no full stream associated to a pair of formulas in the logical flow graph of a proof. The following corollary ensures that given a connected component of the graph and a cluster in it, there is always a stream whose extremes are extremes for the graph, that shares edges with the cluster.

Corollary 6. *Let K be a connected component in the logical flow graph of a proof. There is a stream, lying in K, whose extremes are input and output nodes of K. In particular, given a cluster H in K, there is a stream passing through H whose extremes are input and output nodes of K.*

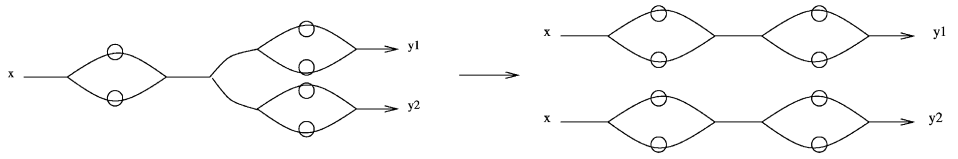
Proof. If K is acyclic, then the statement is obvious. If K is cyclic, let H be a cluster in K, let x be a way-in-node in H, and let y be a way-out-node in H. Such nodes

exist by Theorem 3. Consider an acyclic path in K that goes from an input node X of K to x (such a path exist as noticed in the proof of Theorem 3), it passes through H by taking the *shortest* way between x and y in H, and continues without forming any cycle from y to an output node Y of K (again, a path from y to Y exists as noticed in the proof of Theorem 3). This is an acyclic path from X to Y in K because the path in H was chosen to be the shortest.

To derive the statement, it is enough to consider, as a stream in K, any stream that contains the path that we just constructed. Notice that the path itself is a stream. □

Coming back to our last example, notice that there is an acyclic path from X to Y that shares edges with the *cluster*.

To conclude, let us observe that each connected component of the logical flow graph of a proof is usually constituted by several *streams* as illustrated by the graph on the left



which contains two streams, the first has base $[x, y_1]$ and the second has base $[x, y_2]$ (as illustrated on the right).

Different connected components and streams occurring in a specific component define two different types of *interaction* in a proof:

1. Distinct streams can share subgraphs (as in the figure above) that can influence one another. This interaction is analyzed in [4] through a combinatorial study of cut-elimination.
2. Different connected components belong to the same logical flow graph because the proof ties them together by means of logical connectives. The interaction between distinguished connected components has been studied through the notion of *proof-net* by Girard et al. and many others. Girard's seminal paper [12] introduces the reader to the area. We shall skip here the numerous references.

In this paper we shall not address any question concerning *interaction* with the exception of Section 8.

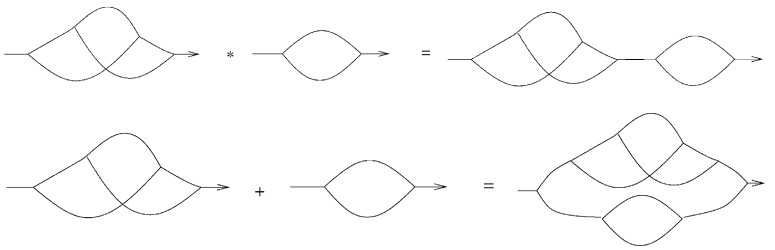
5. Strings and stream theory

In this section we introduce a language to represent a stream as a string formalized in a language of three symbols $b, *, +$, where b is a constant, $*$ is the binary operation of *concatenation*, and $+$ is the binary operation of *bifurcation*. We will also need two extra symbols (,) to be used as separators. The language will be called $\mathscr{S}$.

Definition 7. A *string* is a word in the language $\mathscr{S}$ satisfying one of the following conditions:

(i) b is a string,
(ii) if w_1, w_2 are strings then $w_1 * w_2$ is a string,
(iii) if w_1, w_2 are strings then $w_1 + w_2$ is a string.

Example 8. The string b corresponds to a stream which looks like a sequence of edges. The diagrams below illustrate the behavior of the operations of concatenation $*$ and of bifurcation $+$ on streams



where to "concatenate two streams" means to align them one after the other, and to "bifurcate two streams" means to align them one parallel to the other.

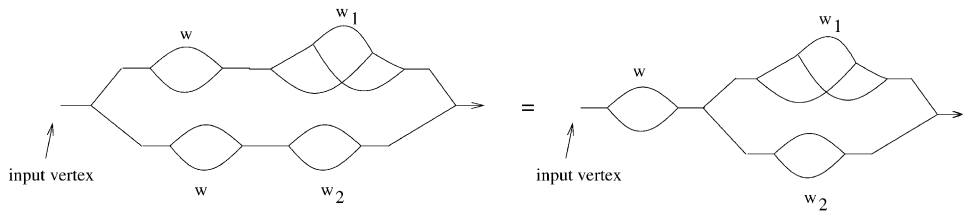
The topology of streams changes with the application of both operations since new branching points are generated. Through concatenation we create sequences of focal pairs and through bifurcation we create new focal pairs.

We define an equational first order theory describing streams.

Definition 9. *Stream theory* contains the following axioms:

A1 $b * w = w$ (left identity),
A1′ $w * b = w$ (right identity),
A2 $(w * w_1) + (w * w_2) = w * (w_1 + w_2)$ (left distributivity),
A2′ $(w_1 * w) + (w_2 * w) = (w_1 + w_2) * w$ (right distributivity),
A3 $w_1 + w_2 = w_2 + w_1$ (commutativity of $+$),
A4 $w_1 + (w_2 + w_3) = (w_1 + w_2) + w_3$ (associativity of $+$).

Axioms A1, A1′ tell us that no topological change is achieved by concatenating a sequence of edges to a stream. Axioms A2, A2′ can be illustrated in terms of streams as follows:



By the point of view of the input vertex, the structure of the two graphs is identical. Namely, the number of paths going from the input vertex to the output vertex remains unchanged for both graphs. Axioms A3 and A4 guarantee that the topological structure of a stream is preserved by commutativity and associativity of $+$.

Definition 10. Stream theory is called *associative* when it is extended with the axiom
A5 $w_1 * (w_2 * w_3) = (w_1 * w_2) * w_3$ (associativity of $*$)
and it is called *commutative* when it is extended with the axiom
A6 $w_1 * w_2 = w_2 * w_1$ (commutativity of $*$).

One can think of A1–A6 as universally quantified axioms over variables w, w_1, w_2, w_3. Stream theory is constituted by the theory of abelian additive semi-groups and by a multiplicative part. The distributivity law holds. Notice also that the operation of concatenation is not commutative in general and that in *associative* Stream theory we do not distinguish substrings of the form $(w_1 * w_2) * w_3$ and $w_1 * (w_2 * w_3)$.

Since the operation of *bifurcation* is associative, in the following we will drop parenthesis when not necessary. For instance, the word $(w_1 + w_2) + w_3$ will be written $w_1 + w_2 + w_3$. We will also use the shorthand notation w^n instead of

$$\underbrace{w + w + \cdots + w}_{n \text{ times}}$$

and we will call n the *multiplicity* of w.

Proposition 11. *The following properties are provable in Stream theory*
1. $w * w_1 + \cdots + w * w_n = w * (w_1 + \cdots + w_n)$
2. $w_1 * w + \cdots + w_n * w = (w_1 + \cdots + w_n) * w$
3. $w_1 + \cdots + w_n = w_{\pi(1)} + \cdots + w_{\pi(n)}$ *for any permutation* π.

Proof. To check the three properties it is a routine. Note that they correspond to axioms A2, A2′, A3. They are derived from their corresponding axiom with the help of A4. □

In any stream structure there are infinitely many non-equivalent strings. Namely, for any two positive integers n, m, we have $b^n \neq b^m$. Therefore the cardinality of a stream structure is at least countable.

Proposition 12 (Normalization). *Let w be a string. There is a unique integer $k \geqslant 1$ such that $w = b^k$ is provable in Stream theory.*

Proof. Let the height $h(w)$ of a string w be defined as follows: $h(b) = 1$, $h(w_1 * w_2) = h(w_1 + w_2) = \max\{h(w_1), h(w_2)\} + 1$.

By induction on the height of the string w we show that there is a k such that w is equivalent to b^k.

If w is b then $k = 1$.

If w is of the form $w_1 * w_2$ then by induction hypothesis, w_1 is equivalent to b^m and w_2 is equivalent to b^l. Therefore $w_1 * w_2$ is equivalent to $b^m * b^l$. On the other hand, by Property 2 in Proposition 11 and axiom A1 one derives

$$\begin{aligned} b^m * b^l &= \underbrace{b * b^l + \cdots + b * b^l}_{m \text{ times}} \\ &= \underbrace{b^l + \cdots + b^l}_{m \text{ times}} \\ &= b^{m \cdot l}. \end{aligned}$$

If w is of the form $w_1 + w_2$ then by induction hypothesis w_1, w_2 are equivalent to b^m, b^l respectively. Therefore $w_1 + w_2$ is equivalent to $b^m + b^l$ and hence to b^{m+l} by axiom A6. □

Proposition 13 (Cancellation modulo torsion). *The following properties are provable:*
1. $b * w_1 = b * w_2 \Rightarrow w_1 = w_2$ (*left cancellation*),
2. $w_1 * b = w_2 * b \Rightarrow w_1 = w_2$ (*right cancellation*),
3. $w * w_1 = w * w_2 \Rightarrow w_1^n = w_2^n$ *where* $b^n = w$,
4. $w_1 * w = w_2 * w \Rightarrow w_1^n = w_2^n$ *where* $b^n = w$.

Proof. Properties 1, 2 are derived using axioms A1, A1′. To derive property 3 we apply Proposition 12 to the string w and observe that there is a positive integer n such that w is equivalent to b^n. Hence $w * w_1 = b^n * w_1$ and $w * w_2 = b^n * w_2$. By right distributivity (i.e. axiom A2), $b^n * w_1 = (b * w_1 + \cdots + b * w_1)$ (where the right-hand side contains n strings of the form $b * w_1$) and analogously $b^n * w_2 = (b * w_2 + \cdots + b * w_2)$. Hence $w_1 + \cdots + w_1 = w_2 + \cdots + w_2$ by axiom A1, i.e. $w_1^n = w_2^n$. Similarly, one shows property 4. In this case axioms A1′ and A2′ should be used instead. □

6. Strings and streams of proofs

Before proceeding we would like to pose for a moment and give some justification for our analysis. In Section 2.3, formalized proofs are mapped into the space of graphs with nodes of degree at most 3. The image of such a map contains graphs (maybe cyclic) which are rather special. They satisfy the properties described in Theorems 3 and 5, for instance. Also, the topology of the graphs in the image induces an equivalence relation in the space of proofs. Namely, if we call l the map from proofs to logical flow graphs of proofs, and i the map from graphs to topologically minimal graphs, then the map il defines an equivalence relation on the space of proofs as follows:

$$\Pi_1 \equiv \Pi_2 \text{ iff } il(\Pi_1) \sim il(\Pi_2),$$

where Π_1, Π_2 are proofs and $\sim$ denotes the isomorphism between graphs. The quotient space induced by the equivalence relation $\equiv$ seems not very useful to investigate the

combinatorics and complexity of cut-elimination. This is the reason to look at special subgraphs of the logical flow graphs, the streams, and try to analyze whether the complexity of cut-elimination on streams is related to their *topological* structure. The basic advantage in using the notion of stream resides on the possibility of associating to it a one-dimensional representation, that is a *string*, which can be manipulated in purely algebraic terms. The process of cut-elimination, seen as a process of manipulation of streams, can be described in purely algebraic terms also.

We shall start this section by describing how, given a stream of a proof, one defines a string associated to it. It turns out that several strings might be associated to the same stream (they will all be equivalent in the sense of Stream theory), and that both *compact* strings and *explicit* strings (to be defined later) uniquely identify a stream.

To associate a string to a stream in a proof, we think of the logical flow graph of the proof as being embedded in the plane, we read bridges and directed paths connecting weak formulas to occurrences lying either in the end-sequent or in a cut-formula, as the constant b, we read a cut-edge as performing the operation of concatenation $*$, and we express the nesting of bridges through the operation of bifurcation $+$.

Definition 14. A *decomposition* of an acyclic directed graph P is a set of streams $\{P_1,\ldots,P_n\}$ lying on P such that

(i) each directed path in P belongs to exactly one of the streams P_j,
(ii) the input node of P_j is an input node of P,
(iii) the output node of P_j is an output node of P.

If P is a graph lying in the logical flow graph of a proof Π, then the notation $P\restriction_{\Pi'}$ represents the restriction of P to the logical flow graph of a subproof Π' of Π.

Definition 15. Let Π be a proof whose last rule is applied to the subproofs Π_1,Π_2. (If the rule is unary, then consider Π_1 only.) A stream P is called an extension of $P_1,\ldots,P_n$ if P is a stream of Π and $\{P_1,\ldots,P_n\}$ is a decomposition of $P\restriction_{\Pi_1}, P\restriction_{\Pi_2}$.

In Definition 15, if the last rule of Π is a *cut*, then the number of streams n can be arbitrarily large. In fact, a stream P might pass through the cut-formulas, back and forth, several times.

Definition 16. Let P be a stream of a proof Π. A string *associated* to a stream P is built by induction on the height of the subproofs Π' of Π in such a way that the following conditions are satisfied:

(i) if Π' does not contain cuts, let $\{P_1,\ldots,P_n\}$ be a decomposition of $P\restriction_{\Pi'}$. Each string associated to P_i is b^m, where m is the number of distinguished directed paths lying in P_i,
(ii) if the last rule of Π' is not a *cut* and it is applied to Π_1,Π_2 (if the rule is unary, then consider Π_1 only), let $\{P_1,\ldots,P_n\}$ be the decomposition of $P\restriction_{\Pi_1}$ and $P\restriction_{\Pi_2}$.

Then the decomposition of $P \restriction_{\Pi'}$ is $\{P'_1,\dots,P'_l\}$ where

(a) P'_i is an extension of P_j and the string associated to P'_i is w_j, where w_j is the string associated to P_j, or

(b) P'_i is a stream obtained by the union of streams $P'_{j_1},\dots,P'_{j_k}$ which are extensions of $P_{j_1},\dots,P_{j_k} \in \{P_1,\dots,P_n\}$ and are based on the same pair $[v_i,w_i]$. The string associated to P'_i is $w_{j_1}+\cdots+w_{j_k}$, where w_{j_r} is the string associated to P_{j_r}, for $r=1,\dots,k$,

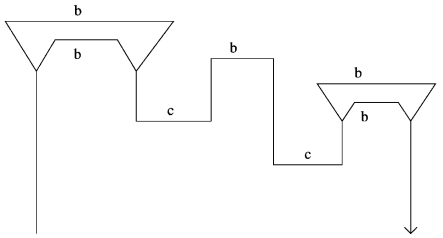
(iii) if the last rule of Π' is a *cut* applied to subproofs Π_1,Π_2 and $\{P_1,\dots,P_n\}$ is a decomposition of $P \restriction_{\Pi_1}$ and $P \restriction_{\Pi_2}$, then the decomposition of $P \restriction_{\Pi'}$ is $\{P'_1,\dots,P'_l\}$, where the P'_i's are all possible extensions of the streams in $\{P_1,\dots,P_n\}$ such that

(a) P'_i is an extension of $P_{j_1},\dots,P_{j_k}$ and the string associated to P'_i is $w_{j_1}*\cdots*w_{j_k}$ (possibly $k=1$), where w_{j_r} is the string associated to P_{j_r}, for $r=1,\dots,k$, or

(b) P'_i is a stream obtained by the union of streams $P'_{j_1},\dots,P'_{j_k}$ where each P'_{j_r} is an extension of $P_{j_r,1},\dots,P_{j_r,l} \in \{P_1,\dots,P_n\}$ and all the P'_{j_r} are based on the same pair $[v_j,w_j]$. The string associated to P'_i is $w'_{j_1}+\cdots+w'_{j_k}$ where w'_{j_r} is $w_{j_r,1}*\cdots*w_{j_r,l}$, and $w_{j_r,s}$ is the string associated to $P_{j_r,s}$ for $s=1,\dots,l$.

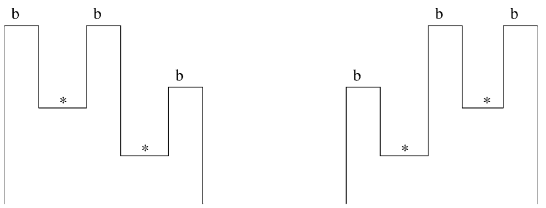
Let us now give a couple of examples to illustrate how streams of proofs can be read as strings.

Example 17. Consider the following stream of a proof



where a path starts on the left-hand side of the picture, goes up until it reaches a branching point. Two distinguished paths depart from this branching point, they pass through two axioms in the proof and they rejoin into another branching point to pass through a cut-edge, form a new bridge, pass through a second cut-edge, go up into a new subproof, split once more, join once more and end-up into the right-hand side of the picture. The structure of this proof can be described with different strings. For instance, the strings $(b^2*b)*b^2$, $(b*b)*b^2+(b*b)*b^2$ and $(b*b)*b+(b*b)*b+(b*b)*b+(b*b)*b$ are descriptions of the above structure. One can easily check that the three strings are equivalent in Stream theory. In fact, the first and the second string are equivalent by A2′ and the second is equivalent to the third by A2.

Example 18. Consider two streams of proofs having the following form:



where the height of a cut in the proof is reflected by the position of horizontal cut-edges in the graph. We read the stream on the left as $(b*b)*b$ and the stream on the right as $b*(b*b)$. The parenthesis denote the *height* of a cut in a proof. In this way the string $(b*b)*b$ represents a cut between a subproof containing a stream $b*b$ and a second subproof containing a bridge. The representation of $b*(b*b)$ is symmetric.

Definition 19. A string associated to a stream in a proof Π is *compact* if it is determined as described in Definition 16, where we require that the streams P_i lying in decompositions $\{P_1,\ldots,P_n\}$ relative to subproofs Π' of Π have distinct bases $[v_i,w_i]$, for $i=1,\ldots,n$, i.e. $v_i\neq v_j$ or $w_i\neq w_j$, for all $i\neq j$.

Proposition 20. *Given a stream of a proof, there is a unique compact string associated to it (up to commutativity of $+$).*

Proof. The statement follows directly from the fact that, for all subproofs Π' of Π, there is only one stream P' (up to commutativity of $+$) which is defined as $P\restriction_{\Pi'}$ on a pair $[v',w']$. □

In Example 17, the string $(b^2*b)*b^2$ is compact. Compact strings are a *succinct* way to represents streams. All other representations have larger complexity, that is a larger number of symbols. Consider, for instance, a string of the form $w_1+w_2+\cdots+w_n$ describing a stream P of a proof based on $[v,w]$. If each w_i describes a *simple* path in P based on $[v,w]$, then w_i is of the form $b*b*b\ldots b*b$. In this case we say that the string $w_1+w_2+\cdots+w_n$ describes the stream P *explicitly*: all paths are described one by one. This description is the most expensive in terms of the number of symbols and we refer to it as *explicit* representation, or *explicit string*.

Proposition 21. *Let Π be a cut-free proof. All the streams of Π are described by strings of the form b^n, where n is bounded by the number of axiom-edges of Π.*

Proof. Either a stream of Π is a directed path connecting a weak occurrence in an axiom with an occurrence of a formula in the end-sequent of Π, or it is constituted by several nested bridges. In the first case, the string is b and the claim follows since a proof has always at least one axiom. In the second case, the claim follows from Definition 16 (assertion 1) and the following observation. Each path belonging

to a stream passes through an axiom-edge by definition. Suppose that more than one path belonging to the same stream passes through the same axiom. These paths will pass through the pair of *distinguished* formulas of the axiom and, in particular, through distinct atomic occurrences. Therefore there should be a moment along the proof, where the occurrences need to identify (since a stream has *one* input vertex and *one* output vertex.) But the identification is impossible because of the subformula property which holds for cut-free proofs. □

Remark 22. The cut-free proof of $F(2) \to F(2^{2^n})$ which can be easily constructed from axioms of the form $F(t) \to F(t)$, the F-rule, and contractions on the left, is an example of proof where the number n in Proposition 21 corresponds *exactly* to the number of axioms in the proof. The logical flow graph of this proof is a stream based on $[F(2), F(2^{2^n})]$. Since distinguished formulas in axioms are atomic and all of them are linked to the end-sequent, then the number of axioms in the proof must coincide with the number of paths of the stream.

Remark 23. In proofs we can only compose and bifurcate bridges having the same orientation. This justifies the fact that stream structures are not defined to have a group on their additive part but simply an additive semi-group.

6.1. From strings to streams

We have seen that there are maps from the space of streams to the space of strings that associate to a stream, in a unique way, a string. An example is given by the map that associates *compact* strings, and another example is the map that associates *explicit* strings. There are also well-defined maps from strings to streams of proofs.

Theorem 24. *For each string there is a proof whose logical flow graph is the stream described by the string.*

Proof. Let w be a string. We shall build a proof Π_w whose end-sequent is $F(x) \to F(t(x))$, for some term $t(x)$, and whose logical flow graph is a stream associated to w. The construction is done by induction on the complexity of the substrings.

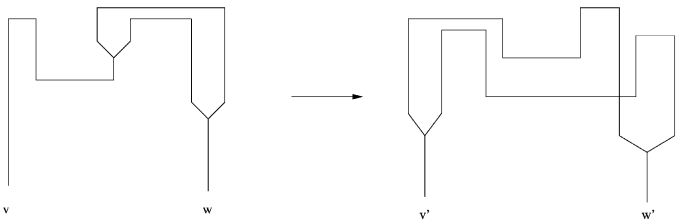
If w is b then Π_w is an axiom of the form $F(x) \to F(x)$.

If w is $w_1 * w_2$ then by induction we know Π_{w_1} and Π_{w_2}. The end-sequents of Π_{w_1} and Π_{w_2} are $F(x) \to F(f(x))$ and $F(x) \to F(g(x))$, for some term $f(x)$ and $g(x)$. By substituting the occurrences of the variable x in Π_{w_2} with the term $f(x)$ we obtain a proof Π'_{w_2} with end-sequent $F(f(x)) \to F(g(f(x)))$, the same logical structure as Π_{w_2} and the same logical graph. (This is straightforward to check.) Then, we combine with a cut on the formula $F(f(x))$ the proofs Π_{w_1} and Π'_{w_2} and obtain a proof of the sequent $F(x) \to F(g(f(x)))$ whose associated string is $w_1 * w_2$ (by Definition 16).

If w is $w_1 + w_2$ then by induction we know Π_{w_1} and Π_{w_2}. Their end-sequents are of the form $F(x) \to F(f(x))$ and $F(x) \to F(g(x))$, for some term $f(x)$ and $g(x)$. We apply the F-rule to Π_{w_1} and Π_{w_2} to obtain a proof of $F(x), F(x) \to F(f(x) * g(x))$.

By applying a contraction to the occurrences $F(x)$ on the left, we obtain the sequent $F(x) \to F(f(x) \cdot g(x))$ and a proof with associated string $w_1 + w_2$ (by Definition 16). □

Remark 25. The construction proposed in the proof of Theorem 24 associates a proof to a given string, where the logical flow graph of the proof is *a* stream described by the string. In fact, the derivation constructed in the proof of the theorem is by no means the only one that could be given. To illustrate this point, we give an example that we shall use later in our discussion. Take the following transformation of streams due to the procedure of cut-elimination (the existence of such transformations is proved in [4])



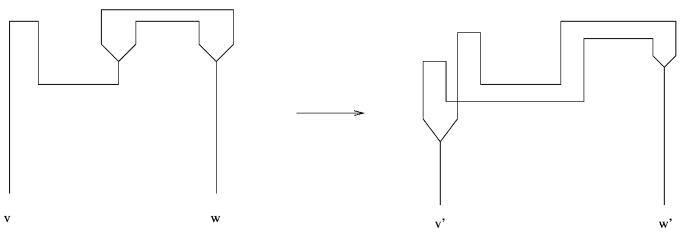
where one can see that the topology of the stream is preserved by the transformation. The figure corresponds to the resolution of a contraction during Gentzen's procedure, namely it corresponds to the transformation from (8) to (9) described in Section 2.2. The stream on the left is associated to a proof where a contraction lies just above the cut (i.e. the branching point lying above the horizontal edge) as in (8). The transformation from (8) to (9) moves this contraction before the pair of new cuts as in (9). In the picture on the right, we see the stream in (9), where the branching point lies below the two horizontal edges introduced by the transformation.

The second contraction, which lies just above the output node w, is applied much after the cut rule and it is not involved in the transformation. The streams, before and after cut-elimination, are described by the *same* string $b * b + b * b$, by Definition 16. Notice also that this is the only possible string describing the streams above. (The fact that the height of the second contraction is smaller than the height of the cut rule, plays a crucial role here.)

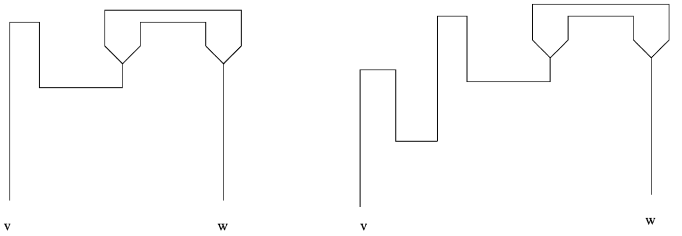
Remark 26. The proof Π constructed in Theorem 24 is formalized in the extension of the propositional sequent calculus with F-rules. Notice that there are no weak occurrences in Π and that cuts are only on atomic formulas. In particular, the proof Π is reduced. One might be unhappy with the presence of F-rules in Π and might like to look for proofs in pure propositional logic. To find proofs containing a required stream is not difficult once we allow an arbitrary use of weak occurrences in Π. To find a propositional proof Π which is also reduced is, on the other hand, a very difficult task and it is not at all clear whether there is a *uniform* algorithm that given a stream, returns a reduced propositional proof which contains it.

6.2. Strings and topology of streams

Usually, proofs having streams with the same topological structure, might have different *compact* and *explicit* strings associated to them. Take, for instance, the following transformation of streams due to the procedure of cut-elimination [4]:



where the proof on the left is described by the compact string $b*b^2$ and by the explicit string $(b*b)^2$. For the proof on the right, the string $(b*b)^2$ is both compact and explicit. Also, consider the following pair of streams:



where the proof on the left is described as above, and the one on the right is described by the compact string $b*(b*b^2)$ and by the explicit string $(b*b*b)^2$.

We say that a stream P in Π is *minimal* if for any subproof Π' of Π whose end-sequent is combined with some cut-rule, the graph $P\restriction_{\Pi'}$ does not contain neither simple bridges nor directed paths to or from weak occurrences, as connected components.

Proposition 27. *Let G_1, G_2 be two minimal streams. If G_1 and G_2 have the same topological structure then they are described by the same explicit strings.*

Proof. Let G_1 be a stream based on $[v,w]$ and G_2 be a stream based on $[x,y]$. By definition an explicit string for a stream is a *bifurcation* of strings which are concatenations of b's and describe simple paths in the stream. Since G_1, G_2 have the same topological structure, the number of paths between v,w and x,y must be the same, say n. In particular, the two streams are minimal by hypothesis and therefore they have the same number of cut-edges lying along each path. This is enough to conclude that if $w_1+\cdots+w_n$ is an *explicit* string of G_1 then $w_1+\cdots+w_n$ must also be a string of G_2. Moreover, this string is *unique* up to commutativity of $+$. □

7. Arithmetical value of strings and complexity

If a proof contains cuts, then the *compact* description of its streams might be *much* shorter than the *explicit* ones. Let us illustrate this point with a concrete example where the presence of a chain of focal pairs in a stream is described by a compact string of size n, and by an explicit string of size 2^n.

Example 28. We suppose that the binary function $\cdot$ is the arithmetical multiplication, and we look at a proof of $F(2) \to F(2^{2^n})$. (This example is taken from [4].) There is no use of quantifiers and the formalization takes place on the propositional part of predicate logic. Our basic building block is given by

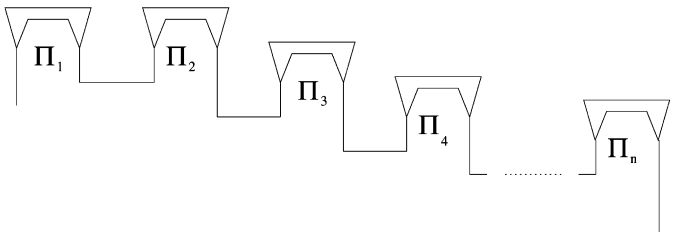
$$F(2^{2^{j-1}}) \to F(2^{2^j}), \tag{14}$$

which can be proved for each j in only a few steps. (One starts with two copies of the axiom $F(2^{2^j}) \to F(2^{2^j})$ and combines them with the F-rule to get

$$F(2^{2^j}), F(2^{2^j}) \to F(2^{2^{j+1}}).$$

Then one applies a contraction to the two occurrences of $F(2^{2^j})$ on the left and derives the sequent.) We can then combine a sequence of these proofs together using cuts to get a proof of $F(2) \to F(2^{2^n})$ in $O(n)$ steps.

The logical flow graph for the proof of $F(2) \to F(2^{2^n})$ looks roughly as follows:



where the notation Π_j, $1 \leqslant j \leqslant n$ refers to the proofs of $F(2) \to F(2^{2^n})$. The logical flow graph of each Π_j contains two branches, one for the contraction of two occurrences of $F(2^{2^{j-1}})$ on the left, and another for the use of the F-rule on the right. Along the graph we notice a *chain* of n pairs of branches which gives rise to an exponential number of paths starting at $F(2)$ and ending in $F(2^{2^n})$. There are no cycles in the proof and the logical flow graph of this proof is a stream. The compact string associated to it is $b^2 * b^2 * \cdots * b^2$, where each of the n factors b^2 corresponds to a focal pair in the graph. The explicit string is b^{2^n}.

Can we detect a chain of focal pairs lying in a stream by reading its associated string? To answer, let us introduce some more notation. A string can always be seen as a *concatenation* of strings either of the form b or $w_1 + w_2 + \cdots + w_n$, where $w_1, w_2, \ldots, w_n$ are strings and $n > 1$. For instance, take the string w of the form $(b * (b^2 + b * b^3)) * b^3$.

We say that b, $b^2 + b * b^3$ and b^3 are concatenated to each other and we call them *factors* of w. A factor of the form $w_1 + w_2 + \cdots + w_n$ is called *non-trivial*. The number of non-trivial factors of w is the *index* of w.

Proposition 29. *Let P be a stream based on $[v, w]$ and lying in the logical flow graph of a proof Π. Let w be the string representing P. If w contains a substring of index n then there is a chain of n focal pairs in the logical flow graph of Π.*

Proof. Let P be a stream and w be the string associated to it. By Definition 16, any substring w' of w describes a stream lying in some subproof Π' of Π. If w' has index n then w' is of the form $w_1 * \cdots * w_m$, where $w_{i_1}, \ldots, w_{i_n}$ are *non-trivial* factors, for $i_j \in \{1, 2, \ldots, m\}$ and $j = 1, \ldots, n$. By Definition 16, the substrings $w_1, \ldots, w_m$ correspond to streams lying in subproofs Π_i linked through cuts, and based on pairs $[A_i, B_i]$, where the atomic formula A_i is B_{i+1} (in Π' the occurrences A_i, B_{i+1} are linked by a cut-edge). In particular, the substrings w_{i_j} are of the form

$$w_{i_j,1} + \cdots + w_{i_j,m_{i_j}} \quad \text{for } m_{i_j} > 1.$$

By Definition 16, $w_{i_j,1}, \ldots, w_{i_j,m_{i_j}}$ are strings associated to streams based on the *same* pair $[A, B]$. Therefore, there is at least a focal pair lying in the subproof Π_{i_j} (because $m_{i_j} > 1$). This means that in Π' we have a chain of focal pairs which is defined by the cut-edges connecting the subproofs $\Pi_1, \ldots, \Pi_m$ and the focal pairs in the Π_{i_j}'s. □

As illustrated in Example 28, a chain of n focal pairs lying in a stream gives rise to at least 2^n distinct paths. In Proposition 30, we show that the number of paths in a stream can be computed precisely by means of an arithmetical interpretation of strings. We say that the *arithmetical value* $t(w)$ associated to a string w is defined as follows: $t(b)$ is 1, $t(w_1 * w_2)$ is $t(w_1) * t(w_2)$ and $t(w_1 + w_2)$ is $t(w_1) + t(w_2)$.

Proposition 30. *Let w be a string associated to a stream P. Then the number of directed paths from the input vertex to the output vertex of P is $t(w)$.*

Proof. This follows in a straightforward way from the interpretation between streams and strings described in Example 8. □

Proposition 31. *Let w be a string and w' be a substring of w. Then any substitution of w' with a string w'' where $t(w') = t(w'')$, gives a string w^* such that $t(w^*) = t(w)$.*

Proof. The arithmetical term $t(w)$ (once considered in its syntactical form) contains the arithmetical subterm $t(w')$. If we substitute the occurrence of $t(w')$ with $t(w'')$ we shall obtain the arithmetical term $t(w^*)$ whose value is $t(w)$ since $t(w') = t(w'')$. □

8. Strings and cut-elimination

How does a stream of a proof evolve through the procedure of cut-elimination? A more general version of this question was addressed in [2, 4] where the combinatorial operation of "duplication" on directed graphs was introduced and used to analyze the combinatorics of the transformations induced by cut-elimination. Here we would like to show that the evolution of streams can be analyzed through simple *algebraic* manipulations. We give a number of rewriting rules and show that these rules describe the transformation.

The set of rewriting rules that we want to consider contains the *computational rules*:

R1 $b * w \to w$,

R1′ $w * b \to w$,

R2 $w * (w_1 + \cdots + w_n) \to w * w_1 + \cdots + w * w_n$,

R2′ $(w_1 + \cdots + w_n) * w \to w_1 * w + \cdots + w_n * w$,

R3 $w_1 + \cdots + w_n \to w_{\pi(1)} + \cdots + w_{\pi(n)}$ for any permutation π,

R5 $w_1 * (w_2 * w_3) \to (w_1 * w_2) * w_3$,

R5′ $(w_1 * w_2) * w_3 \to w_1 * (w_2 * w_3)$,

which follow from the axioms A1–A3 and A5. Axiom A4 does not have a counterpart here because from now on we shall consider only compact strings associated to proofs. This justifies the absence of a rule R4. It also contains the *local structural rule*

R6 $w \to w + w$

which represents the possibility to duplicate the same substrings, and the *global structural rules*

R7$_1$ $(w_1 + w_2) * w_3 * (w_4 + w_5) \to (w_1 * w_3 * w_4) + (w_2 * w_3 * w_5)$,

R7$_2$ $w_1 * w_3 * (w_4 + w_5) \to w_1 * w_3 * w_4$,

R7$_3$ $w_1 * w_3 * (w_4 + w_5) \to w_1 * w_3 * w_5$

which cancel some of the substrings. It is clear that local and global structural rules allow a string to *grow* and *shrink*. Theorem 32 shows how the process of cut-elimination induces streams to shrink and grow. Notice that if w is a string transformed by R6 into w' then $t(w) < t(w')$. If w is transformed in w' by R7$_i$, for $i = 1, \ldots, 3$, then $t(w') < t(w)$. On the other hand, if any of the rules R1–R5 are used then $t(w) = t(w')$.

Before stating Theorem 32, we need to introduce some more definitions. A *reduction* is a sequence of applications of rewriting rules that transforms a string w into a string w'. An application of a rewriting rule $s \to t$ to w replaces an occurrence of the substring s in w with the substring t. A reduction of a string is called *final* if it leads to a string of the form b^n, for some n.

We say that a *path* in the logical flow graph of a proof is *disrupted* by a step of cut-elimination when given two nodes of the path, after the transformation there is no more path between them. A *stream* is disrupted when one of its paths is disrupted. This notion was introduced in [2] where the reader can find concrete examples.

Theorem 32. *Let Π be a proof and let w be the compact string associated to some stream of Π. For any process of elimination of cuts which gives a cut-free proof with n axioms, either there is a reduction of w to a string b^m (where $m \leqslant n$) through the rules* R1–R7, *or the stream is disrupted by some step of elimination of cuts either on weak occurrences or on contractions.*

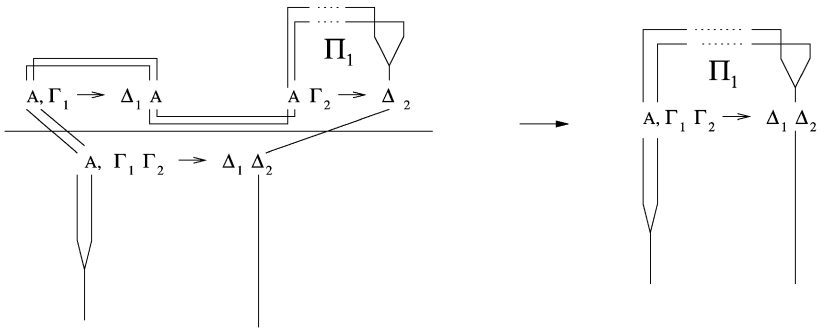
Proof. The proof consists of checking that at each stage of the procedure of cut-elimination, the deformation of streams in the proof is regulated by the set of rules R1–R7. Namely, if w is a compact string associated to a stream P in Π, and if Π' is the proof obtained by transforming Π through a step of elimination of cuts, then there is a compact string associated to a stream P' in Π' which is obtained from w after rewriting one or several of its substrings with the rules R1–R7. The stream P' is the deformation of the stream P in Π induced by the manipulation of the proof Π. Since paths of the logical flow graph of Π might split after manipulation, it might happen that the stream P is disrupted and that P' does not exist. This is a well-defined possibility and we shall discuss it later in the proof. In case P is transformed into P', notice that several paths of P might be involved in the transformation of the same cut, and this implies that several substrings of w will be simultaneously affected.

In the analysis of the steps of cut-elimination we shall follow the presentation of Section 2.2. We shall divide this analysis into two distinct cases; first we treat the case of a stream P that passes through the cut-formulas A and second, we look at the case of P that passes through the side formulas of the antecedents of the cut-rule.

Suppose to be in the first case. If A is atomic, then notice that only one path of the stream P passes through A. If A is *not* atomic, then a directed *path* belonging to the stream might pass through the same cut-formula several times and different portions of the same path might behave differently. Moreover, several *distinct* paths of the stream might pass through A. Their behavior will be captured by a simultaneous applications of rules R1–R7 to substrings of w which describe different *portions* of the stream involved in the transformation. These aspects of the transformation will be clearer while proceeding with our step-by-step analysis.

Let us start by considering the elimination of a cut when one of the cut-formulas is a distinguished occurrence in an *axiom* as in (1). There might be several paths of the stream P that pass through the distinguished occurrences and each of these paths will be denoted b (in the string w) because of compactness. This is shown by an easy chasing of Definition 16. If the axiom appears on the left (as it is the case in display (1)), then we use R1 to replace substrings $b * w'$ in w by w'. If the axiom appears on the right, then R1′ allows to replace substrings of the form $w' * b$ in w by w'. The string that we obtain is compact. Let us illustrate this case with a concrete example. Consider the stream P illustrated by the figure on

the left



The stream P is constituted by two paths that depart from a common input vertex, they pass through the cut-formula A and go towards Π_1, they move along Π_1 and finally they recombine into a common output vertex. The compact string associated to P is $(b * w_1) + (b * w_2)$, where w_1, w_2 correspond to substreams in Π_1. After manipulation, P is transformed into the stream P' illustrated on the right, which has compact string $w_1 + w_2$. To pass from $(b * w_1) + (b * w_2)$ to $w_1 + w_2$, one applies R1 twice to the substrings $b * w_1$ and $b * w_2$.

If both cut-formulas are main formulas of two *logical rules*, we have to distinguish the case where both rules have one single antecedent (that is, the case of quantifier rules and negation rule) from the case where one of the logical rules has two antecedents. In the first case, no rule, among R1–R7, need to be applied. The second case is more complicated and we suppose, without loss of generality, that the cut-formulas are of the form $A \vee B$ as in (4) to (5). Two situations might arise.

First, suppose that there is a substream lying in the stream that passes through *both A and B* and that it is described by a substring $(w_A * w_{AB}) * w_B$, where w_A represents a substream passing through A in Π'_2, w_{AB} describes a substream passing through both A and B in Π'_1, and w_B describes a substream passing through B in Π'_3. After cut-elimination, the proof turns either into (5) or into a proof that looks like (5) where the cut on B is performed before the cut on A. If the cut on A precedes the cut on B, then no rewriting rule is applied and the substream in (5) is described by the string $(w_A * w_{AB}) * w_B$; if the cut on B precedes the cut on A, then R5 is applied and the string that describes the substream becomes $w_A * (w_{AB} * w_B)$.

Second, suppose that there are paths of the stream that pass through *exactly one* of the disjuncts. In this case the paths will be simply stretched and no change in their associated strings will take place.

These are the only two possible situations that might occur and the treatment of the transformations on other logical connectives is similar. Of course, a path might pass through A and B several times, or w_{AB} might describe a stream passing through A and B on a cut-formula on the right (in case the logical connective is $\wedge$ for instance), or the initial substream might be described by $w_A * (w_{AB} * w_B)$. It is easy to imagine all

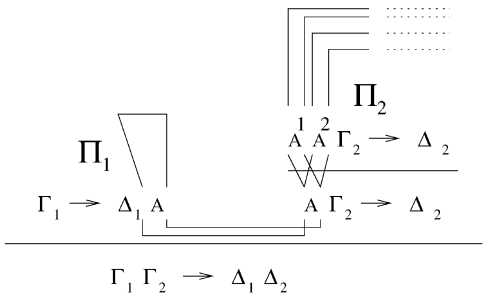
combinations. The main point is that the treatment specified above adapts easily to all other variants. In particular, rule R5′ might be used instead of R5.

If a cut is applied to a formula A obtained from a *contraction* on two occurrences A^1, A^2 as in (8), then the procedure of cut-elimination yields a duplication of the subproof Π_1 as in (9) and this creates quite intriguing situations.

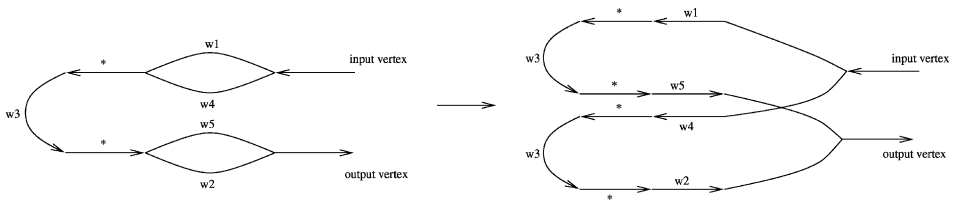
We start to handle the simplest case. Suppose that there is a substream $\bar{P}$ of P lying in Π_1 with exactly one of its extremes that occurs in the cut-formula A. Then, $\bar{P}$ either passes through the side formulas in Γ_1, Δ_1 or ends-up into some weak formula in Π_1. Let w_3 be the string describing it. The stream P connects $\bar{P}$ with the rest of the stream lying in Π_2 through a cut-edge which splits into two paths, one passing through A^1 and the other passing through A^2. Let w_1, w_2 be substrings describing the portion of P lying in P_2 that originates in A^1, A^2 respectively. The topology we have just described is represented by a substring of w of the form $w_3 * (w_1 + w_2)$. After duplication of the subproof the substring will be transformed into the substring $(w_3 * w_1) + (w_3 * w_2)$ and this is done by applying rules R2, R2′.

This transformation is illustrated, in a concrete way, in the picture of Remark 25, where a stream P is drawn on the left. The substrings w_1, w_2, w_3 represent bridges in P and the topological structure of P (described abstractly by $w_3 * (w_1 + w_2)$) is represented, in this example, by the string $b * (b + b)$. After the transformation, the stream P' (on the right hand side of the picture in Remark 25) is represented by $(b * b) + (b * b)$. Algebraically, this string is obtained by applying R2 to $b * (b + b)$.

Suppose now that both the extremes of the substream $\bar{P}$ in Π_1 lie in the cut-formula A. This case is the most intriguing. As before, let w_3 be the substring describing $\bar{P}$. After passing through two cut-edges, the stream P will go up to the contraction formulas A^1, A^2, where four paths will depart as illustrated in the figure below:



It might be that not all of the four paths belong to P and because of this we shall handle different cases. We suppose first that *all* four paths belong to P: two of them come from its input vertex, and are represented by the strings w_1, w_4, and the other two go towards its output vertex, and are represented by the strings w_2, w_5. We illustrate the transformation of this portion of P as follows:



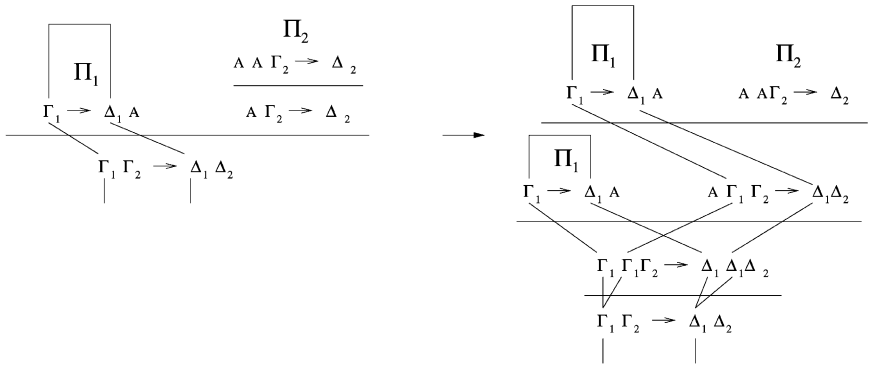
where we think of the streams as being stretched. Algebraically, the transformation is described by rule $R7_1$, where the paths $w_1 * w_3 * w_2$ and $w_4 * w_3 * w_5$ are lost. If w_1, w_4, w_5 belong to the stream, the transformation is described by $R7_2$ or $R7_3$. Any other combination of three paths is handled similarly. If only w_1, w_5 or w_4, w_2 belong to the stream, then the substring is unaltered. If the paths w_1, w_2 or w_4, w_5 belong to the stream, then the stream will be disrupted and the second part of the statement holds. This concludes the treatment of the contraction rule. (To be precise, since the operation of bifurcation $+$ is commutative, we might need to use R3 to rearrange the order of the substrings of the form $w_1 + \cdots + w_k$ for handling properly the contraction case.)

If two cuts are *permuted* as in (12) to (13), we suppose first that the stream passes through *both* pairs of cut-formulas C^1, C^2 and D^1, D^2. By Definition 16, the substream P_2 of P that lies in Π_2 and passes through C^2 might be described in one of the following two ways: either by a compact string of the form $w_1 * \cdots * w_n$, or by a compact string of the form $(w_{1,1} * \cdots * w_{1,n_1}) + \cdots + (w_{k,1} * \cdots * w_{k,n_k})$. Let w' be a substring associated to a substream P_1 lying in Π_1, passing through C^1, D^1, D^2 and connected by a cut-edge to P_2. In the first case, w will contain a substring of the form $w' * (w_1 * \cdots * w_n)$; then, we apply R2 and obtain $(w' * w_1) * (w_2 * \cdots * w_n)$. In the second case, w will contain a substring of the form $w' * ((w_{1,1} * \cdots * w_{1,n_1}) + \cdots + (w_{k,1} * \cdots * w_{k,n_k}))$; then, we apply R5 and obtain $(w' * w_{1,1} * \cdots * w_{1,n_1}) + \cdots + (w' * w_{k,1} * \cdots * w_{k,n_k})$.

In the symmetric case, when two cuts are permuted as in (13) to (12), then rule $R2'$ has to be applied instead of R2. If two cuts are permuted and the stream passes through *exactly one* of the cut-formulas, then no rule is applied.

If a cut is *pushed upwards* (through logical rules or a contraction) as in (10) to (11), then it might happen that a contraction is pushed below the cut. This might imply that a branching point of the stream (maybe several of them) will be pushed below a cut-edge and therefore that the compact description of the stream might change. The new compact description is obtained with the application of rules R2 and $R2'$.

In all cases treated above, the stream P passes through the cut-formulas simplified by the step of the procedure. The paths of P that pass through the *side formulas* of the antecedents of a cut, are stretched and no modification of the substrings associated to them is needed. The only exception is the contraction case (from (8) to (9)), where a substream passing through the side formulas of the subproof Π_1, might be duplicated with the duplication of Π_1. In this case, rule R6 is used. This is illustrated in the following picture (discussed in [4])



To conclude we consider the case of a cut-formula which is *weak* in an axiom, as in (2). The procedure of cut-elimination will induce a disruption of the structure of the proof due to the removal of the subproof Π_0. In case the stream passes through Π_0, then the stream will be disrupted and the second part of the statement holds.

Let us notice that all along the proof one needs to verify that the string associated to the proof Π' is compact. This is a straightforward verification and we leave it to the reader.

To conclude, if the stream has not been disrupted then from Proposition 21 it follows that w has been reduced to a string b^m where m is smaller than the number of axioms in Π. □

Example 33. Consider the proof of the sequent $F(2) \to F(2^{2^n})$ given in Example 28 and described by the compact string $b^2 * b^2 * b^2 \cdots b^2 * b^2$, where the terms b^2 are exactly n. We can calculate the exponential expansion of this proof after cut-elimination through a purely *algebraic* manipulation of strings as we shall show for $n = 4$, namely for the string $b^2 * b^2 * b^2 * b^2$. For an arbitrary n the approach is similar. By applying R4 to the substrings $b^2 * b^2$ we get $(b^2 * b + b^2 * b) * (b^2 * b + b^2 * b)$; we apply R3 to all substrings $b^2 * b$ to get $((b * b + b * b) + (b * b + b * b)) * ((b * b + b * b) + (b * b + b * b))$ and by R1 we obtain $((b + b) + (b + b)) * ((b + b) + (b + b))$, or in short $b^4 * b^4$. By applying again R4, R3 and R1 we get

$$
\begin{aligned}
(b^4 * b^2 + b^4 * b^2) &\to ((b^4 * b + b^4 * b) + (b^4 * b + b^4 * b)) \\
&\to (((b^2 * b + b^2 * b) + (b^2 * b + b^2 * b)) \\
&\quad + ((b^2 * b + b^2 * b) + (b^2 * b + b^2 * b))) \\
&\to ((((b * b + b * b) + (b * b + b * b)) \\
&\quad + ((b * b + b * b) + (b * b + b * b))) \\
&\quad + (((b * b + b * b) + (b * b + b * b))
\end{aligned}
$$

$$+((b*b+b*b)+(b*b+b*b))))$$
$$\to b^{16}.$$

Note that b^{16} is the minimum expected value for a cut-free proof computing $F(2^{2^4})$ from $F(2)$. In fact the minimal tree of computation of $F(2^{2^4})$ has 2^4-1 branching points corresponding to 2^4-1 multiplications, and it uses 2^4 $(=16)$ times $F(2)$.

Theorem 34. *Let Π be a proof of $F(x_1),\dots,F(x_r)\to F(f(x_1,\dots,x_r))$, with atomic cuts and no weak formulas. Let w_i be the full compact string based on $[F(x_i), F(f(x_1,\dots,x_r))]$, for $i=1,\dots,r$. Then, all procedures of cut-elimination transforming Π into Π' and w_i into w_i', for $i=1,\dots,r$, are simulated by* R1–R5 *and w_i' is b^{n_i} where $n_i=t(w_i)$.*

Proof. The proof Π has a very simple structure. Here are some properties:

1. The proof Π contains no logical rules and all formulas appearing in Π are atomic. This is because cuts are defined on atomic formulas and formulas in the end-sequent are atomic.
2. For any sequent in Π, exactly one formula lies on the right-hand side of the sequent. This follows from 1.
3. There are no contractions on the right in Π. This is because there are no weak formulas and no logical rule can be applied on negative formulas in Π.

Properties 1–3 imply some properties of the flow graph of Π:

(a) No path passes twice through the same cut-formula, since cut-formulas are atomic.
(b) No path passes twice through the side formulas of a sequent used in a cut-rule. This is because at any stage of the procedure, the sequents have the form $F(s_1),\dots,F(s_k)\to F(g(s_1,\dots,s_k))$, and contractions are on the left only. Therefore, if the cut-formula in the sequent is a positive occurrence of the form $F(g(s_1,\dots,s_k))$, then all side formulas are negative occurrences and no path can start and end in them; if the cut-formula is a negative occurrence $F(s_j)$, for some $j=1,\dots,k$, then there might be several paths passing through the side formulas (in fact, all of them should pass through the formula on the right of the sequent) but the proof where $F(s_j)$ occurs cannot be duplicated because contractions can be applied only to negative formulas.

Properties (a) and (b) ensure that the rewriting rules R6 and R7 are not used by the simulation. In particular R1–R5 are rewriting rules of the form $p\to q$ where $t(p)=t(q)$. This implies that the final string w_i' are of the form b^{n_i} where $n_i=t(w_i)$. □

Theorem 35. *Let Π be a proof of the sequent S such that the number of symbols in S is n. If any cut-free proof Π' of S has at least $2^{\Omega(n)}$ lines then*

1. *either there is a stream w in Π such that $t(w)$ is $2^{\Omega(n)}$,*
2. *or any process of cut-elimination from Π to Π' is simulated by* R6.

Proof. If Π' has at least $2^{\Omega(n)}$ lines then there is a stream b^k in it where k is $2^{\Omega(n)}$. This means that b^k has been obtained from a string w in Π with or without the help of R6. If R6 has not been used then the arithmetical value $t(w)$ is at least k since rules R1–R5 and R7 cannot augment it. □

Remark 36. Rule R6 has a *global* effect. In fact, it does *not* concern the cut-formulas involved in the step of elimination of cuts, but the structure of the proof itself. It corresponds to the existence of a path in the proof which passes twice through the side formulas of a subproof that is duplicated by the procedure of cut-elimination.

Even if a proof might be such that *no path passes twice through the side formulas of a sequent applied to a cut-rule*, during cut-elimination this property might be lost. It is easy to check that permutation of cuts, contraction and resolution of cut-formulas which are main formulas of logical rules, might produce a proof which falsifies this property. Once the property is violated, rule R6 might play a role in the transformation.

Problem 37. To decide whether w_1 and w_2 can be reduced to the same string b^k, for some k, by using rules R1–R5, can be done in polynomial time. In fact w_1 and w_2 can be polynomially reduced to some b^{k_1}, b^{k_2} for some fixed k_1, k_2, and it is sufficient to check whether the values k_1 and k_2 are the same or not. If we allow the rules R1–R7, does the question become NP-complete?

Remark 38. All theorems in this paper are proved for predicate logic. In contrast, the *exponential* values of Theorem 35 (see also Example 33) suggest that the analysis of cut-elimination based on streams concerns *propositional* logic more than predicate logic. The multi-exponential bounds that can be reached in the context of predicate logic are a consequence of the *interaction* of streams. This line of investigation is wide open and needs new ideas as well as new combinatorial tools for the study of interaction. The reader might like to consult [5, 7] for an analysis of proofs in the presence of cycles and multi-exponential bounds.

References

[1] S. Buss, The undecidability of k-provability, Ann. Pure Appl. Logic 53 (1991) 72–102.
[2] A. Carbone, Interpolants, cut elimination and flow graphs, Ann. Pure Appl. Logic 83 (1997) 249–299.
[3] A. Carbone, Turning cycles into spirals, Ann. Pure Appl. Logic 96 (1999) 57–73.
[4] A. Carbone, Duplication of directed graphs and exponential blow up of proofs, Ann. Pure Appl. Logic 100 (1999) 1–76.
[5] A. Carbone, Cycling in proofs and feasibility, Trans. Am. Math. Soc. 352 (2000) 2049–2075.
[6] A. Carbone, The cost of a cycle is a square, J. Symbolic Logic, 2001, to appear.
[7] A. Carbone, Asymptotic cyclic expansion and bridge groups of formal proofs, J. Algebra, 2001, to appear.
[8] A. Carbone, S. Semmes, Making proofs without modus ponens: an introduction to the combinatorics and complexity of cut elimination, Bull. Amer. Math. Soc. 34 (1997) 131–159.
[9] A. Carbone, S. Semmes, Looking from the inside and the outside, Synthèse 125 (2000) 385–416.

[10] A. Carbone, S. Semmes, A Graphic Apology for Symmetry and Implicitness – Combinatorial Complexity of Proofs, Languages and Geometric Constructions. Mathematical Monographs, Oxford University Press, Oxford, 2000.
[11] G. Gentzen, Untersuchungen über das logische Schließen I–II, Math. Z. 39 (1934) 176–210, 405–431.
[12] J.Y. Girard, Linear logic, Theoret. Comput. Sci. 50 (1987) 1–102.
[13] J.Y. Girard, in: Proof Theory and Logical Complexity, Studies in Proof Theory, Monographs, Vol. 1, Bibliopolis, Napoli, Italy, 1987.
[14] A. Haken, The intractability of resolution, Theoret. Comput. Sci. 39 (1985) 297–308.
[15] V.P. Orevkov, Lower bounds for increasing complexity of derivations after cut elimination, J. Sov. Math. 20 (4) (1982).
[16] R. Statman, Structural complexity of proofs, Ph.D. Thesis, Stanford University, 1974.
[17] R. Statman, Bounds for proof-search and speed-up in predicate calculus, Ann. Math. Logic 15 (1978) 225–287.
[18] R. Statman, Lower bounds on Herbrand's Theorem, Proc. Am. Math. Soc. 75 (1979) 104–107.
[19] G. Takeuti, in: Proof Theory, 2nd Edition, Studies in Logic, Vol. 81, North-Holland, Amsterdam, 1987.
[20] G.S. Tseitin, Complexity of a derivation in the propositional calculus, Zap. Nauchn. Sem. Leningrad Otd. Mat. Inst. Akad. Nauk SSSR 8 (1968) 234–259.