# Using Tables to Construct Non-Redundant Proofs

Vivek Nigam

INRIA & LIX/École Polytechnique, Palaiseau, France
nigam at lix.inria.fr

**Abstract.** Proofs containing more than one subproof for a common subgoal are less preferred in frameworks such as Proof Carrying Code, where proofs are stored and communicated, than proofs that don't contain such redundancies. In this paper, we show how (cut-free) proofs can be transformed into non-redundant cut-proofs. Two main questions arise when trying to construct these non-redundant proofs: First, which cut-formulas should be used; Second, where to perform cut rules. Some advances in proof theory, namely, our better understanding of focused proofs, allows us to propose the following answers: We use only atomic subgoals of the original proof; and we place cut rules only at the end of the *asynchronous phases*. The backbone of a non-redundant proof is a tree, called *tree of multicut derivations* (**tmcd**), where a node is a derivation containing only multicut rules, and an edge represents the provability dependency between a subgoal introduced by a node's multicut rule and another (tree of) multicut derivation. We show how to obtain a **tmcd** from an existing proof.

## 1 Introduction

Frameworks such as Proof Carrying Code [10, 2], where mobile codes are sent with proofs that assure that these codes satisfy certain properties, provide "real world" concerns, not only for *provability*, but also for the *shape* and *format* of proofs. In these frameworks, since proofs need to be stored and communicated, proofs have to attend certain engineering aspects; for instance, the size of proofs is relevant; more precisely, smaller proofs are preferred.

Consider as a motivational example, the proofs that the 12th Fibonacci number is 144, denoted by (fib 12 144), obtained from the following logic specification $\{fib\ 1\ 1, fib\ 2\ 1, \forall XYZ.[fib\ X\ Y \wedge fib\ (X+1)\ Z \supset fib\ (X+2)\ (Y+Z)]\}$. Two types of proofs can be distinguished: one where a *forward chaining* behavior is adopted, and another where a *backward chaining* behavior is adopted. In the frameworks previously mentioned, the former linear size non-redundant proof is preferred to the later exponential size redundant proof.

We propose a procedure to construct a non-redundant sequent calculus proof from a redundant sequent calculus proof. This is done by collecting (or *tabling*) from an existing proof a set of atomic lemmas, called *table*. Theses lemmas are then used as cut formulas to construct a non-redundant proof containing

cuts. The use of cuts in non-redundant proofs is not surprising; it is well known that, when compared with cut proofs, cut-free proofs are potentially bigger (also known as the *cut-elimination blow-up*).

There are two main problems to be addressed: (1) which lemmas should be used; and (2) when to use these lemmas, that is, while constructing the non-redundant proof, when should a cut be used. Our answers for these question lies in the structure of *focused proofs*.

By distinguishing rules that are invertible, called asynchronous rules, from rules that are not invertible, called synchronous rules, focused proofs are organized in two alternating phases: *asynchronous phases*, where asynchronous rules are eagerly applied; and *synchronous phases*, where a formula is picked, or *focused* on, and synchronous rules are applied hereditarely to its subformulas (for more about focused proofs, we invite the reader to [1]). Some recent advances on our understanding about focused proofs in classical and intuitionistic logics [6] and about the effect of *atomic polarities*[1] in the shape of proofs [3, 8], provides us with the machinery necessary to propose the following answers to the previous questions:

1) We collect (or table), in the existing proof, all the atomic subgoals. This restriction allows us to construct non-redundant proofs with a polarized cut-rule [8], where an atomic cut-formula has negative polarity in one branch of the proof tree and positive polarity on the other branch of the proof tree. As we investigate elsewhere [8], by using this polarized cut-rule, it is possible to mix a forward chaining behavior with a backward chaining behavior, what enforces some subgoals not to be re-proven;

2) The idea is to use the lemmas as close as possible to the root of the tree, so that if a lemma is to be proved again later in the tree, then it would already be available in the set of hypothesis of the sequent and allow to immediately complete the proof with an initial rule. However, it may happen that a lemma can't be proved right from the bottom of the tree and should be introduced into the proof only when there is an increment in the set of hypothesis of a sequent (by for example, a right implication rule). We show that focusing provides the discipline necessary to identify the places in a proof tree where new lemmas should be introduced, namely, at the end of the *asynchronous* phases.

This paper is structured as follows: we introduce in Section 2 some key concepts related to *focusing* and introduce the intuitionistic system $LJF^t$ and the use of tables to specify *multicut derivations* (*mcd*). In Section 3, we specify how to extract *tree of multicut derivations* (**tmcd**), that is the backbone to construct non-redundant proofs, from different types of proofs, namely Horn Theory proofs, Uniform proofs, and $LJF^t$ proofs. In Section 4, we show and discuss some experimental results, and finally in Section 5, we finish with some concluding remarks.

[1] In a focused system, atoms are assigned either positive or negative polarity. This assignment is necessary to organize focused proofs.

## 2 Preliminaries

### 2.1 $LJF^t$

We say a formula has *positive* polarity if it is the formula *true*, $\bot$, a positive atom, or its main connective is either $\wedge$, $\vee$, or $\exists$. Otherwise, a formula has *negative* polarity.

Liang and Miller proposed the focused intuitionistic system *LJF* [6], that, differently from other focused intuitionistic systems [4, 5], allows, as in Andreoli's original focused system in Linear Logic [1], atoms to be assigned arbitrary polarities.

The Figure 1 depicts the inference rules in *LJF* , where four different types of sequents can be identified: **(1)** The sequent $[\Gamma] - _A \rightarrow$ is a *right-focusing* sequent (the focus is $A$); **(2)** The sequent $[\Gamma] \xrightarrow{A} [R]$: is a *left-focusing* sequent (with focus on $A$); **(3)** The sequent $[\Gamma], \Theta \longrightarrow \mathcal{R}$ is an *unfocused sequent*. Here, $\Gamma$ contains negative formulas and positive atoms, and $\mathcal{R}$ is either in brackets, written as $[R]$, or without brackets; **(4)** The sequent $[\Gamma] \longrightarrow [R]$ is an instance of the previous sequent where $\Theta$ is empty.

Asynchronous phase use the third type of sequent above (the unfocused sequents): in that case, $\Theta$ contains positive or negative formulas. If $\Theta$ contains positive formulas, then an introduction rule (either $\wedge_l, \exists_l$ or $false_l$) is used to decompose it; if it is negative, then the formula is moved to the $\Gamma$ context (by using the $[]_l$ rule). The end of the asynchronous phase is represented by the fourth type of sequent. Such a sequent is then established by using one of the decide rules, $D_r$ or $D_l$. The application of one of these decide rules then selects a formula for focusing and switches proof search to the *synchronous phase* or *focused phase*. This focused phase then proceeds by applying sequences of inference rules on focused formulas: in general, backtracking may be necessary in this phase of search. Moreover, the rules $\wedge_l, \exists_l, false_l, \supset_r, \forall_r, []_l, []_r, \wedge_r^-$ are *asynchronous rules*, and the remaining rules are *synchronous rules*.

As pointed out elsewhere [6, 3, 8], the atomic polarities play an important role in the shape of the proofs, without affecting in no way provability. For instance, if all atoms have positive polarity, only proofs with a forward chaining behavior are possible, and on the other hand, if all atoms have negative polarity, only proofs with a backward chaining behavior are possible, for example *uniform proofs* [7].

$LJF^t$ capitalizes on the observation that atomic polarities can be arbitrarily assigned, and extends *LJF* in two ways. **(1)** Extends the *LJF* sequents with a polarity context, $\mathcal{P}$, which specifies all the positive atoms in a sequent. An atom in a sequent, $\mathcal{P}; [\Gamma] \longrightarrow [R]$, is positive if and only if $A \in \mathcal{P}$; **(2)** extends *LJF* with the following polarized multicut rule:

$$\frac{\mathcal{P}; [\Gamma] \longrightarrow [A_1] \quad \cdots \quad \mathcal{P}; [\Gamma] \longrightarrow [A_n] \quad \mathcal{P} \cup \Delta; [\Gamma \cup \Delta] \longrightarrow [R]}{\mathcal{P}; [\Gamma] \longrightarrow [R]}\ mc.$$

Where $\Delta = \{A_1, \ldots, A_n\}$ is a set of atoms. The multicut rule is the only rule that can change the polarity context in a proof. Since, from this point on, we only use $LJF^t$, we name its rules with the same names used for the *LJF* rules.

$$\frac{[N,\Gamma] \xrightarrow{N} [R]}{[N,\Gamma] \longrightarrow [R]}\ D_l \quad \frac{[\Gamma]-_P\!\rightarrow}{[\Gamma] \longrightarrow [P]}\ D_r \quad \frac{[\Gamma], P \longrightarrow [R]}{[\Gamma] \xrightarrow{P} [R]}\ R_l \quad \frac{[\Gamma] \longrightarrow N}{[\Gamma]-_N\!\rightarrow}\ R_r$$

$$\frac{}{[\Gamma] \xrightarrow{A_n} [A_n]}\ I_l \quad \frac{}{[\Gamma, A_p]-_{A_p}\!\rightarrow}\ I_r \quad \frac{[\Gamma, N_a], \Theta \longrightarrow \mathcal{R}}{[\Gamma], \Theta, N_a \longrightarrow \mathcal{R}}\ []_l \quad \frac{[\Gamma], \Theta \longrightarrow [P_a]}{[\Gamma], \Theta \longrightarrow P_a}\ []_r$$

$$\frac{}{[\Gamma], \Theta, \bot \longrightarrow \mathcal{R}}\ \mathit{false}_l \quad \frac{[\Gamma], \Theta \longrightarrow \mathcal{R}}{[\Gamma], \Theta, \mathit{true} \longrightarrow \mathcal{R}}\ \mathit{true}_l \quad \frac{}{[\Gamma]-_{\mathit{true}}\!\rightarrow}\ \mathit{true}_r$$

$$\frac{[\Gamma], \Theta, A, B \longrightarrow \mathcal{R}}{[\Gamma], \Theta, A \wedge B \longrightarrow \mathcal{R}}\ \wedge_l \quad \frac{[\Gamma]-_A\!\rightarrow \quad [\Gamma]-_B\!\rightarrow}{[\Gamma]-_{A\wedge B}\!\rightarrow}\ \wedge_r \quad \frac{[\Gamma]-_A\!\rightarrow \quad [\Gamma] \xrightarrow{B} [R]}{[\Gamma] \xrightarrow{A \supset B} [R]}\ \supset_l$$

$$\frac{[\Gamma] \xrightarrow{A_i} [R]}{[\Gamma] \xrightarrow{A_1 \wedge^- A_2} [R]}\ \wedge_l^- \quad \frac{[\Gamma], \Theta \longrightarrow A \quad [\Gamma], \Theta \longrightarrow B}{[\Gamma], \Theta \longrightarrow A \wedge^- B}\ \wedge_r^- \quad \frac{[\Gamma], \Theta, A \longrightarrow B}{[\Gamma], \Theta \longrightarrow A \supset B}\ \supset_r$$

$$\frac{[\Gamma], \Theta, A \longrightarrow \mathcal{R}}{[\Gamma], \Theta, \exists y A \longrightarrow \mathcal{R}}\ \exists_l \quad \frac{[\Gamma]-_{A[t/x]}\!\rightarrow}{[\Gamma]-_{\exists x A}\!\rightarrow}\ \exists_r \quad \frac{[\Gamma] \xrightarrow{A[t/x]} [R]}{[\Gamma] \xrightarrow{\forall x A} [R]}\ \forall_l \quad \frac{[\Gamma], \Theta \longrightarrow A}{[\Gamma], \Theta \longrightarrow \forall y A}\ \forall_r$$

**Fig. 1.** *LJF* : Here, $\Gamma$ is a set of formulas, $\Delta$ is a list of formulas, $A_n$ denotes a negative atom, $A_p$ a positive atom, and $P$ a positive formula, $N$ a negative formula, $N_a$ a negative formula or an atom, and $P_a$ a positive formula or an atom. All other formulas are arbitrary and $y$ is not free in $\Gamma, \Theta$ or $R$.

We often omit sequent's polarity context, when it is easy to infer it from the context.

**Remarks:** (1) The results in this paper could be easily applied to (focused) classic logics, such as *LKF* [6]. (2) Here we only consider focused proofs; however, it seems possible to apply the results obtained here to a more general setting where non-focused proofs are considered, by using methods such as in [9] to convert non-focused proofs to focused proofs, but this is left out of the scope of this paper.

In the next subsection, we use this polarized multicut rule to construct multicut derivations that are the basic element used to construct non-redundant proofs.

### 2.2 Tables as Multicut Derivations

We consider a table as a partially ordered finite set of atoms.

**Definition 1.** *A* table *is a tuple* $\mathcal{T} = \langle \mathcal{A}, \prec \rangle$*, where* $\mathcal{A}$ *is some finite set of atoms, and* $\prec$ *is a partial order relation over the elements of* $\mathcal{A}$*.*

In a table, each atom represents, intuitively, a provable sub-goal necessary in the proof of a sequent (say $\Gamma \longrightarrow G$), and the order relation the provability dependency between the atoms, that is, if $A \prec B$ then $A$ is a subgoal used to prove the goal $B$.

The next definition specifies a derivation composed only of multicut rules, represented by a table.

**Definition 2.** *Let $\mathcal{T} = \langle \mathcal{A}, \prec \rangle$ be a table. The* multicut derivation *for $\mathcal{T}$ and the sequent $\mathcal{S} = \Gamma \longrightarrow G$, written as $mcd(\mathcal{T}, \mathcal{S})$, is defined inductively as follows: if $\mathcal{A}$ is empty, then $mcd(\mathcal{T}, \mathcal{S})$ is the derivation containing just the sequent $\Gamma \longrightarrow G$. Otherwise, if $\{A_1, \ldots, A_n\}$ is the collection of $\prec$-minimal elements in $\mathcal{A}$ and if $\Pi$ is the multicut derivation for the smaller table $\langle \mathcal{A} \setminus \{A_1, \ldots, A_n\}, \prec \rangle$ and the sequent $\Gamma, A_1, \ldots, A_n \longrightarrow G$, then $mcd(\mathcal{T}, \mathcal{S})$ is the derivation*

$$\frac{\Gamma \longrightarrow A_1 \quad \cdots \quad \Gamma \longrightarrow A_n \qquad \overset{\Pi}{\Gamma, A_1, \ldots, A_n \longrightarrow G}}{\Gamma \longrightarrow G}\ mc$$

*Multicut derivations are always* open *derivations (that is, they contain leaves that are not proved). A* proof of a multicut derivation *is any (closed) proof that extends this open derivation.*

Elsewhere [8], we investigated the use of tables for obtaining non-redundant proofs in the Horn fragment. We used the following observation:

**Proposition 1.** *[8] Let $\Gamma$ be a set of Horn clauses, $A \in \mathcal{P} \cap \Gamma$, and $\Xi$ be an arbitrary $LJF^t$ proof tree for $\mathcal{P}; [\Gamma] -_G \rightarrow$. Then every occurrence of a sequent with right-hand side the atom $A$ is the conclusion of an $I_r$ rule.*

If in a proof of $\Xi$, there are several non-trivial proofs for the subgoal $A$, that is, proofs that contain more than an inference rule, one could table $A$ and construct the corresponding multicut derivation for this table and construct a non-redundant proof. For example, when comparing the two derivation below, the left derivation could have several non-trivial subproofs for $A$, while the right derivation must have only one non trivial proof for $A$: the proof of the cut's left branch.

$$\frac{\Gamma \longrightarrow A \qquad \Gamma \longrightarrow G}{\Gamma \longrightarrow A \wedge G} \Longrightarrow \frac{\mathcal{P}; [\Gamma] \longrightarrow [A] \qquad \mathcal{P} \cup \{A\}; [\Gamma, A] \longrightarrow [A \wedge G]}{\mathcal{P}; [\Gamma] \longrightarrow [A \wedge G]}\ mc.$$

In the next sections, we use *mcd*s to construct non-redundant proofs. To represent a *mcd*, when we specify the algorithms to extract non-redundant proofs, we use the following data type mcd ::= sequent * atom list[2].

## 3 Tree of Multicut Derivations - tmcd

The backbone of a non-redundant proof is a tree, called *tree of multicut derivations* (**tmcd**), where a node is a derivation containing only multicut rules, and an edge represents the provability dependency between a subgoal introduced by a node's multicut rule and another (tree of) multicut derivation. The architecture of a **tmcd** is depicted in Figure 2. The idea is that each multicut derivation is only used when the context is augmented with new atoms or new positive formulas. Since the context can only be augmented by the asynchronous rules

[2] We use a list of atoms representing the topological sort of a table's partial ordering.

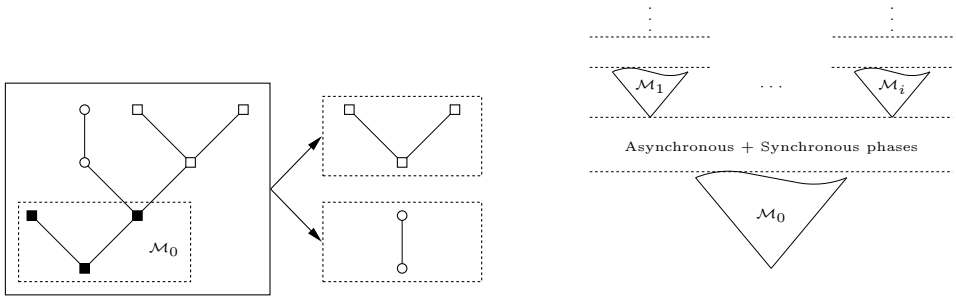


**Fig. 2.** The left figure illustrates of how the function buildTmcd extracts a **tmcd** from a proof tree. The right figure depicts the general architecture of a completed **tmcd** proof. The triangles represent *mcd*s and the dashed lines represents end of asynchronous phases.

$\forall_r, \exists_l$, and $\supset_r$[3] and asynchronous rules are invertible, focusing provides a natural way to identify where to place a multicut derivation, namely, whenever an asynchronous phase ends and the context is augmented with a new positive formula or a new atom. While focused cut-free proofs are structured in two alternating phases, namely, asynchronous phases and synchronous phases, focused cut proofs are structured with one more phase, called *cut phase*, appearing always between an asynchronous phase and the following synchronous phase.

In the following subsections we specify algorithms to extract **tmcd**s from proofs in Horn Theory, from Uniform Proofs [7], and from *LJF* proofs. We represent *tmcd*s by the following data type: tmcd ::= atom * mcd * tmcd list.

### 3.1 Horn Theory

A characteristic of uniform Horn Theory proofs, such as the proofs generated by Prolog, is that the context of a sequent never changes. Therefore, all proved atoms in such a proof are provable from the initial context. This property enables us to construct a **tmcd** with only one node, obtained from the function buildTable shown in Figure 3. This function uses the function atomPOT that performs a postorder traversal (i.e., process a nodes premises before processing the node), and uses the function eliRed that retains only the first occurrence of any repeated atomic formula in a list of formulas.

The correctness of this algorithm can be shown by a simple induction and can be found elsewhere [8]. It is easy to show that buildTable extracts a *mcd* from a Horn Theory proof in time $\mathcal{O}(n)$, where $n$ is the size of the inputed proof.

### 3.2 Uniform Proofs

We now specify how to extract a tree of multicut derivations from a finite (goal-directed) proof tree, that is, *LJF* proofs where all atoms are assigned negative

[3] The first two rules augment the context with a new eigenvariable, and the last rule augments the context with some formula.

```
buildTable : tree → mcd
    input: Ξ
        (rootOf(Ξ), eliRed(atomPOT Ξ))
where:
atomPOT : tree → atom list
    input: Node(seq, b_1, ..., b_n)
        if: seq = [Γ] ⟶ [A]
            then: atomPOT b_1 :: ··· ::
                  atomPOT b_n :: A
        else: atomPOT b_1 :: ··· ::
                  atomPOT b_n

buildTmcd : tree → tmcd
    input: Ξ
        eliRed(tmcdAux Empty Ξ)
where:
tmcdAux : (formula * tree) → tmcd
    input: (A, Ξ)
        (A, buildTable(getSCT (contextOf(Ξ)) Ξ),
         map tmcdAux (getChT (contextOf(Ξ)) A (Ξ :: [])))
```

```
getSCT : context → tree → tree
    input: Γ,Node(seq,branches)
        if: Γ = contextOf(seq)
            then: Node(seq, map (getSCT Γ) branches)
        else: Nil.

getChT : context → formula → tree list → (formula * tree) list
    input: Γ,F,Node(seq,branches)::list
        if: seq = [Γ] ⟶ [A]
            then: (getChT Γ A branches) :: (getChT Γ F list)
        else if: seq = [Γ'] ⟶ [_] and Γ ≠ Γ'
            then: (F, Node(seq,branches)) :: (getChT Γ F list)
        else: (getChT Γ F branches) :: (getChT Γ F list)
```

**Fig. 3.** Functions used to extract a **tmcd** from a goal directed proof. Here the functions: contextOf returns the bracket context of a sequent; map applies a function to all the elements of a list.

polarity. In this more general class of proofs, it can happen that, after an asynchronous phase, the context is augmented with a new positive formula or with a new atom. Hence, differently from the Horn Theory case, not all atomic subgoals are provable from the initial context, and therefore, a **tmcd** with more than one node will be the backbone of the non-redundant proof of an uniform proof.

The function buildTmcd, shown in Figure 3, extracts a **tmcd** from an uniform proof. We use the illustration in Figure 2 to explain how this function works. The three different kinds of nodes (filled squares, ellipses, and blank squares) represent sequents with different bracket contexts. First, the subtree with the filled squares is extracted, by using the function getSCT, shown in Figure 3, and, from this subtree, the multicut derivation $\mathcal{M}_0$ is constructed by using the function buildTable. Second, by using the function getChT, also shown in Figure 3, the remaining children trees are extracted together with an atom, appearing in $\mathcal{M}_0$, representing the provability dependency of two nodes of the extracted **tmcd**. Third, buildTmcd is recursively applied to each child tree. Fourth, as done with in Horn Theory case, we eliminate redundancies with the function eliRed as follows: let $\mathcal{M}_a$ be a multicut derivation containing the atom $A$, and let $\mathcal{M}_d$ be a descendent multicut derivation of $\mathcal{M}_a$. If $\mathcal{M}_d$ has the base sequent $[\Gamma] \longrightarrow [A]$ then we obtain a new **tmcd**, by removing the **tmcd**'s subtree with root $\mathcal{M}_d$; or if $A$ is in $\mathcal{M}_d$, then we obtain a new **tmcd** by removing $A$ from $\mathcal{M}_d$ and its possible descendent subtrees from the tree of multicut derivations. There is a final step that is not shown here concerning with the sequent's polarity context, $\mathcal{P}$ and context $\Gamma$, which need to be augmented with the previous application of multicut rules. This can be done in a straightforward way, by traversing the obtained **tmcd**.

We prove the following proposition by induction on the height of the inputed tree $\Xi$.

**Proposition 2.** *Let $\Xi$ be a uniform LJF proof and let $\tau$ =*buildTmcd($\Xi$) *be the tree of multicut derivations obtained from $\Xi$. Then $\tau$ can be completed to a proof by adding derivations containing only one $D_l$ rule.*

### 3.3 *LJF* proof

In the previous subsection, we considered only uniform *LJF* proofs, that is, proofs where all atoms have negative polarity. We now extend the results obtained before to a more richer class of *LJF* proofs where there can also be atoms with positive polarity. When considering this more general class of proofs, there are two main differences with respect to uniform *LJF* proofs: **(1) Initial Right Rule** - Initial right rules can end the proof; **(2) Reaction Left with Atomic Formula** - By allowing atoms with positive polarity, proofs can perform forward chaining steps. For instance, consider the following derivation in which the atom $A$ has positive polarity:

$$\dfrac{[\Gamma]{-_G\!\!\rightarrow}\quad \dfrac{[\Gamma, A] \longrightarrow [G']}{[\Gamma] \xrightarrow{A} [G']}\, R_r, []_r}{[\Gamma] \xrightarrow{G \supset A} [G']}\, \supset_l$$

This type of derivation is not possible to occur in a uniform proof, since the only rule that introduces a focused negative atom in the left is the initial left rule.

To accommodate these differences, we change the functions atomPOT, getSCT, and getChT. For the first difference, namely that initial right rules can finish the proof, it suffices to table the positive atom used to finish the proof, and therefore, in the extracted **tmcd**, this atom will have its polarity changed to positive[4]. For the second difference, namely that atomic reaction left rules can happen, we table the forward chained atom performing, in the extracted **tmcd**, the following transformation:

$$\dfrac{\dfrac{[\Gamma]{-_G\!\!\rightarrow}\quad \dfrac{\dfrac{\Xi}{[\Gamma, A] \longrightarrow [G']}\, R_r, []_l}{[\Gamma] \xrightarrow{A} [G']}\, \supset_l}{[\Gamma] \xrightarrow{G \supset A} [G']}\, D_l}{[\Gamma] \longrightarrow [G']} \quad \Longrightarrow \quad \dfrac{\dfrac{\mathcal{P}; [\Gamma]{-_G\!\!\rightarrow}\quad \dfrac{}{\mathcal{P}; [\Gamma] \xrightarrow{A} [A]}\, I_l}{\mathcal{P}; [\Gamma] \xrightarrow{G \supset A} [A]}\, \supset_l}{\mathcal{P}; [\Gamma] \longrightarrow [A]}\, D_l \qquad \mathcal{P}'; [\Gamma, A] \longrightarrow [G']}{\mathcal{P}; [\Gamma] \longrightarrow [G']}\, mc$$

However, a new question arises: where in the **tmcd** should we perform a cut with $A$ as cut formula. The answer is to insert this cut before inserting the cuts with the atomic subgoals appearing in $\Xi$ because to prove these subgoals it might be necessary to use $A$.

Accordingly, we add new cases, shown in Figure 4, to the functions atomPOT, getSCT, and getChT. To atomPOT, the first case added inserts a positive atom focused on the left to the beginning of the list of atomic subgoals; the second case added inserts to the list of atomic subgoals any positive atom that is used to finish a proof with an initial right rule. The function getSCT, that is used to

[4] Remember that at the base of any **tmcd**, all atoms are assigned negative polarity.

extract subtrees used to construct **tmcd**'s nodes, is modified so that extracted subtrees include atomic reaction left rules. Since the subtree extracted by getSCT is modified, getChT is modified to extract correctly the children subtrees.

function: atomPOT = ...
  **else if:** seq = $[\Gamma] \xrightarrow{A} [G]$ and $A \notin \Gamma$
  **then:** $A$ :: atomPOT $b_1$ :: $\cdots$ ::
    atomPOT $b_n$
  **else if:** seq = $[\Gamma]-_A\rightarrow$
    **then:** $A$ ...
function: getSCT = ...
  **else if:** seq = $[\Gamma] \xrightarrow{A} [G]$
    **then:** Node(seq, map (getSCT ($\Gamma \cup \{A\}$)) branches) ...
function: getChT = ...
  **else if:** seq = $[\Gamma] \xrightarrow{A} [G]$
    **then:** (getChT ($\Gamma \cup \{A\}$) F branches) :: (getChT ($\Gamma \cup \{A\}$) F list) ...

**Fig. 4.** New cases added to the functions shown in Figure 2 to handle all $LJF^t$ proofs.

We prove the following proposition by induction on the height of the inputed tree $\Xi$.

**Proposition 3.** *Let $\Xi$ be an LJF proof and let $\tau$ =*buildTmcd($\Xi$) *be the tree of multicut derivations obtained from $\Xi$. Then $\tau$ can be completed to a proof by adding derivations containing only one $D_l$ rule.*

The complexity of buildTmcd lies in the comparison of two context (e.g. contexof(seq) = $\Gamma$). Considering an upperbound, $s$, on the length of formulas, we can show that buildTmcd extracts a **tmcd** from a $LJF^t$ proof in time $\mathcal{O}(ns\,(log\,n))$, where $n$ is the size of the inputed proof. This is done by assigning an ordering to clauses, represented by strings; for instance by using the ASCII numbering. To determine that two sequents have the same context, it suffices to sort the contexts of the sequents and then, check one by one the equivalence of the clauses in the contexts.

## 4 Experiments

As the experimental results in the following table shows, by completing the **tmcd** extracted from the original tree ($\Xi$), we obtain a considerably smaller proof. Also, the execution time of buildTmcd is much lesser than the time needed to find the original proof.

| | Size - number of nodes | | Time - ms | | |
|---|---|---|---|---|---|
| | $\Xi$ | proof from **tmcd** | buildTmcd($\Xi$) | find($\Xi$) | complete **tmcd** |
| $5^{th}$ fib | 15 | 15 | 2.8 | 45.0 | 50 |
| $8^{th}$ fib | 67 | 27 | 3.1 | 85.0 | 120 |
| $11^{th}$ fib | 287 | 39 | 4.8 | 330.0 | 170 |

In the case for the $8^{th}$ Fibonacci, the time needed to complete a **tmcd** is much higher than of finding the original proof. However, we observed in our experiments that it takes a constant time of 20 ms to complete one of the open branches of a **tmcd** and there are only a linear number of them, that is, one open branch for each subgoal. Therefore, we expect that while the time to search for proofs exponentially grows with the Fibonacci number, the time of completing the extracted **tmcd** should increase linearly.

## 5 Conclusions

This paper presents an approach, from a proof theoretic point of view, for reducing size of proofs through redundancy elimination. By a careful study of focused proofs, we propose a new structure, called tree of multicut derivations, to be the backbone for the construction of non-redundant proofs. The theoretical and experimental results in this papers suggests that the procedure proposed can be used to obtain smaller proofs.

*Acknowledgments* I thank Dale Miller, Miki Hermann, David Baelde, and anonymous reviewers for their helpful comments and discussions. This work has been supported in part by INRIA through the "Equipes Associées" Slimmer and by the Information Society Technologies programme of the European Commission, Future and Emerging Technologies under the IST-2005-015905 MOBIUS project.

## References

1. Jean-Marc Andreoli. Logic programming with focusing proofs in linear logic. *J. of Logic and Computation*, 2(3):297–347, 1992.
2. Andrew W. Appel. Foundational proof-carrying code. In Proceeding of LICS'01, 2001.
3. Kaustuv Chaudhuri, Frank Pfenning, and Greg Price. A logical characterization of forward and backward chaining in the inverse method. In Proceedings of IJCAR'06, 2006.
4. Pierre-Louis Curien and Hugo Herbelin. The duality of computation. In Proceedings of ICFP '00, 2000.
5. Radha Jagadeesan, Gopalan Nadathur, and Vijay Saraswat. Testing concurrent systems: An interpretation of intuitionistic logic. In Proceedings of FSTTCS, 2005.
6. Chuck Liang and Dale Miller. Focusing and polarization in intuitionistic logic. In Proceedings of CSL'07, 2007.
7. Dale Miller, Gopalan Nadathur, Frank Pfenning, and Andre Scedrov. Uniform proofs as a foundation for logic programming. *Annals of Pure and Applied Logic*, 51:125–157, 1991.
8. Dale Miller and Vivek Nigam. Incorporating tables into proofs. In Proceedings of CSL'07, 2007.
9. Dale Miller and Alexis Saurin. From proofs to focused proofs: a modular proof of focalization in linear logic. In Proceedings of CSL'07, 2007.
10. George C. Necula. Proof-carrying code. In Proceedings of POPL'97, 1997.

## 6 Appendix with Proofs

**Proposition 4.** *Let $\Xi$ be a uniform LJF proof and let $\tau$ =*buildTmcd($\Xi$) *be the tree of multicut derivations obtained from $\Xi$. Then $\tau$ can be completed to a proof by adding derivations containing only one $D_l$ rule.*

**Proof** (*Sketch*) We proceed by induction on the depth of the tree of multicut derivations $\tau$.

**Base Case:** Suppose that the depth of $\tau$ is 0, that is there is only one multicut derivation, $\mathcal{M}$, in $\tau$. We prove the base case, by another induction on the height of $\mathcal{M}$. The base case is trivial, since it would mean that the multicut $\mathcal{M}$ contains only one cut, and this cut would have to use an atom that is already present in the context of the proof. And therefore, it would require only one decide rule to complete the derivation to a proof.

Now the inductive step: Consider that the sequent $\mathcal{P}; [\Gamma \cup \mathcal{P}] \longrightarrow [A_i]$ is a sequent branch in the multicut derivation $\mathcal{M}$. We can find a proof for this sequent with one decision left rule, by proceeding as follows: We check in $\Xi$ the formula $F$ that was focused on to prove $A_i$. After this decision is made, it suffices to perform decide right rules, since this would decompose the possible subgoal originated by performing the previous decide left rule (for instance if $F = G \supset A$). Because of how the tables are constructed, when this subgoal is completely decomposed it must be the case that it encounters a previously proved atom, that is, a positive atom, and therefore, another decision right finishes the proof.

**Inductive Step:** We now have to show that there is a derivation, between a branch sequent of a multicut derivation and its direct descendent multicut derivation, that contains only one decision left rule. This is similar to the base case; it suffices to decide in the same formula as in $\Xi$; and, after performing a synchronous phase and later a possible asynchronous phase, there are two possible outcomes: 1) The goal is not a positive atom and hence there must be a descendent multicut for this goal 2) It is a positive atom, a right decision rule is enough to finish the proof for this branch of the multicut derivation. In both cases, there is only one decision left rule. □

**Proposition 5.** *Let $\Xi$ be an LJF proof and let $\tau$ =*buildTmcd($\Xi$) *be the tree of multicut derivations obtained from $\Xi$. Then $\tau$ can be completed to a proof by adding derivations containing only one $D_l$ rule.*

**Proof** (*Sketch*) This proposition is proved in a similar way as Proposition 2. The only extra consideration is with respect to the forward chaining steps, which adds an extra step in the inductive proof. When there is a forward chaining step, over an atom $A$, in $\Xi$, it means that there is a branch in a multicut derivation of the form $\mathcal{P}; [\Gamma] \longrightarrow [A]$. It suffices to check which formula in $\Gamma$ was used to make a forward chaining step and decide on the same formula. This branch would need therefore of one decide left rule to reach its descendent multicut derivation(s), or to finish the proof, similarly as depicted in the transformation above. □

A more precise proof of the time complexity of buildTmcd:

**Proof** Let $\Xi$ be the inputed proof tree, $n$ be the number of symbols in $\Xi$, and $s$ be an upperbound in the size of a clause in a sequent. We breakdown the algortihm in two parts. 1) The procedure to determine the subproof, $\Xi_\Gamma$, of $\Xi$, containing sequents with context $\Gamma$; 2) building a *mcd* from the $\Xi_\Gamma$.

1) Let $k$ be the number of sequents in $\Xi$ and $p$ the maximum number of clauses in a sequent. We consider the clauses in a sequent as a set of strings. Since we can assign an ordering in the strings, for instance the ASCII number, we can assume that a clause represents a number. To determine that two sequents have the same context, it suffices to sort the contexts of the sequents, and check if all clauses, $p$, are the same, that is, check if the string that they represent are the same. It is necessary to perform this operation for all the sequents in $\Xi$ $(k)$. Therefore the complexity of this procedure is $\mathcal{O}(ksp\,log(p))$, but since $\mathcal{O}(n) \sim \mathcal{O}(kp)$, the algortihm's asymptotic complexity reduces to $\mathcal{O}(ns\,log(n))$.

2) We have that the complexity of the postorder traversal algorithm is $\mathcal{O}(n)$ that is smaller than $\mathcal{O}(ns\,log(n))$. Hence, the complexity of the algorithm is still $\mathcal{O}(ns\,log(n))$. □