Bouncing threads for circular and non-wellfounded proofs – Towards compositionality with circular proofs (Extended version)

David Baelde, Amina Doumane, Denis Kuperberg, Alexis Saurin

Bouncing threads for circular and non-wellfounded proofs

Towards compositionality with circular proofs

(Extended version) *

David Baelde
ENS Rennes
david.baelde@irisa.fr

Amina Doumane
CNRS
amina.doumane@ens-lyon.fr

Denis Kuperberg
CNRS
denis.kuperberg@ens-lyon.fr

Alexis Saurin
CNRS, Université Paris-Cité & INRIA
alexis.saurin@irif.fr

Abstract

Given that (co)inductive types are naturally modelled as fixed points, it is unsurprising that fixed-point logics are of interest in the study of programming languages, via the Curry-Howard (or proofs-as-programs) correspondence. This motivates investigations of the structural proof-theory of fixed-point logics and of their cut-elimination procedures.

Among the various approaches to proofs in fixed-point logics, circular – or cyclic – proofs, are of interest in this regard but suffer from a number of limitations, most notably from a quite restricted use of cuts. Indeed, the validity condition which ensures soundness of non-wellfounded derivations and productivity of their cut-elimination prevents some computationally-relevant patterns of cuts. As a result, traditional circular proofs cannot serve as a basis for a theory of (co)recursive programming by lack of compositionality: there are not enough circular proofs and they compose badly.

The present paper addresses some of these limitations by developing the circular and non-wellfounded proof-theory of multiplicative additive linear logic with fixed points (μMALL) beyond the scope of the seminal works of Santocanale and Fortier and of Baelde et al. We define bouncing-validity: a new, generalized, validity criterion for $\mu\mathsf{MALL}^\infty$, which takes axioms and cuts into account. We show soundness and cut elimination theorems for bouncing-valid non-wellfounded proofs: as a result, even though bouncing-validity proves the same sequents (or judgments) as before, we have many more valid proofs at our disposal. We illustrate the computational relevance of bouncing-validity on a number of examples. Finally, we study the decidability of the criterion in the circular case: we prove it is undecidable in general but identify a hierarchy of decidable sub-criteria.

*This research has been partially supported by ANR project *RECIPROG*, project reference ANR-21-CE48-019-01.

Long version of the LICS 2022 paper, 07/22,
2022.

Keywords Circular Proofs, Linear Logic, Cut Elimination, Decidability, Curry-Howard, Fixed points.

1 Introduction

Fixed points in computer science Fixed-point theory has proved to be a valuable tool in computer science, in particular for reasoning formally about software systems. Its explicit uses may be traced back to the first formal semantics of programming languages in the late 1960s [33] and it is now pervasive in programming language semantics, concurrency, automata theory and software verification techniques. As part of the increasing use of fixed points in computer science, logics featuring fixed points, generally referred to as *μ-calculi*, were developed and studied [27, 32]. Decades later, fixed points are used in various specification languages, e.g. to specify temporal or spatial properties of program executions. Fixed points are also present in most programming languages as recursive types. More interestingly, the Curry-Howard correspondence, which allows to view proofs as programs and formulas as types, has been extended in various ways to encompass fixed-point types, e.g. in System F extended with least and greatest fixed point types [28, 29], in Coq's calculus of (co)inductive constructions [19], and in functional reactive programming types [12].

Fixed-point logics and proofs. Proof systems for fixed point logics can naturally be obtained by taking (co)induction rules that closely reflect fixed point theorems, e.g. Knaster-Tarski's characterization of least fixed points as least pre-fixed points. This is the basis for Kozen's famous axiomatization [27] of the μ-calculus and for other proof-systems [5]. When building proofs for such logics, one must identify (co)invariants which may be significantly more complex than the property to establish – a phenomenon encountered by students learning to prove properties by induction on natural numbers.

An alternative to these explicit (co)induction rules is to consider proof systems featuring non-wellfounded derivation trees: this makes it possible to reason on fixed points

by unfolding them, possibly infinitely often. However, the soundness of such systems requires not only that each inference step follows the legitimate inference rules of the logic, but also that a global *validity* condition is met, which ensures that some progress is made along each infinite branch. Consider for instance the following (regular) infinite derivation, which repeatedly applies the fixed point unfolding rule (σ) where $\sigma \in \{\mu, \nu\}$: $\dfrac{\vdash \sigma X.X}{\vdash \sigma X.X}\,(\sigma)$. The system would be unsound if this derivation were declared valid for both $\sigma = \nu$ and $\sigma = \mu$ as the empty sequent could be derived thanks to the cut rule (least and greatest fixed points are logically dual of each other). However, the derivation when $\sigma = \nu$ should be accepted since $\nu X.X \equiv \top$ (a tautology).

In the case of arithmetic, this style of reasoning is akin to proofs by infinite descent rather than by induction, and has been formally studied in infinitary sequent calculus by Brotherston and Simpson [11]. For propositional μ-calculus, several such infinitary deduction systems have been proposed [13, 25, 27, 37]. Because they are easier to work with than the finitary proof systems (or axiomatizations) based on Kozen-Park (co)induction schemes, they are often found in completeness arguments for such finitary systems [17, 26, 27, 39–41]. They are also better suited for automated reasoning [13, 37]. Among non-wellfounded proofs, regular derivation trees play a particular role. Such proofs, that we call *circular*, can be represented by a finite tree with back-edges: they can be easily finitely represented and algorithmically processed. As such, they have found many applications[1].

Fixed-points in types. On the other end of the Curry-Howard correspondence, one finds programming languages equipped with (co)recursion constructs whose typing naturally reflects the Kozen-Park (co)induction rules [12, 29]. Writing programs in these systems may be difficult, as it involves coming up with complex (co)invariants. These difficulties are only partially lifted through the use of guarded (co)recursion [19] or sized types [1] in Coq or Agda respectively. Furthermore, (co)recursion involves a suspended computation which makes it difficult to analyze the behavior of a program.

In particular, retrictions such as Coq's guard condition results in a serious lack of compositionality properties as well known and analyzed by various authors [2, 3, 9, 19]. As an illustration, we show in Fig. 1 some example of Coq coinductive terms `drop`, `incdrop` and `filter1everyk`. While all are productive coinductive terms, only the first two are valid Coq terms, the third one does not pass the guard condition:
1) `drop` filters one elements every two of its input stream;
2) `incdrop` takes an input stream of nats and filters one element every two, incrementing it;
3) `filter1everyk`'s behaviour depends on a natural number k: it takes an input streams of booleans and filters one element out of k and returning, depending on the parity of k, either the chosen elements or their negation.

As an alternative, one could naturally consider infinitary (or circular) programs, equipped with a global validity condition ensuring that they behave well – in particular that they are terminating, or productive for inhabitants of coinductive types. There is surprisingly little work following this approach. We note the work of Hyvernat [24] whose use of size-change termination can be seen as a form of validity checking: it would be interesting to develop this work from a Curry-Howard perspective to compare it with infinitary proof systems. and foundations are missing.

This can be understood from the fact that the aforementioned infinitary proof systems for fixed point logics are all cut-free; hence, the role of the validity condition in (syntactic) cut-elimination remains unclear from these works. This shortcoming has been addressed first by Santocanale and Fortier: in [18] they consider an infinitary sequent calculus for purely additive logic, featuring cuts and an extended notion of validity, and they show that cuts can be eliminated from valid proofs (in that setting, cut-elimination is not terminating but productive, and converges to a valid cut-free derivation). A key insight of this work is that the validity condition ensures both soundness of the infinitary proof system and productivity of cut-elimination. The result has been generalized later to the multiplicative and additive linear logic with fixed points, μMALL, at the cost of a more complex argument, by Baelde et al. [7]. Through these syntactic cut-elimination results, infinitary proofs for μMALL are given a computational content, which is an important step towards a Curry-Howard correspondence for that logic.

Lack of compositionality of circular proofs. Unfortunately, existing notions of validity impose a quite limited use of cuts in non-wellfounded proofs rejecting many proofs that could be accepted as valid. In particular, this prevents writing circular proofs in a compositional manner, as exemplified in the following (supported by Figure 2):

Example 1.1. Consider formulas $N = \mu X.1 \oplus X$ and $S = \nu Y.N \otimes Y$ encoding natural numbers and streams of natural numbers in μMALL. Indeed, in a typed language/proof assistant such as Coq, these definitions would correspond to the following constructs:

```
    Inductive nat := O: nat | S : nat -> nat.
CoInductive Stream := Cons : (nat * Stream) -> Stream.
```

[1] For instance using circular proofs [35] as a system for representing morphisms in μ-bicomplete categories [34, 36], to study the relationship between induction and infinite descent in first-order arithmetic [11], to generate invariants for program verification in separation logic [10], or as an intermediate between ludics' designs and proofs in linear logic with fixed points [6].

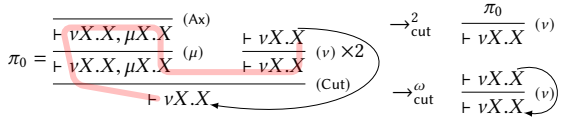
Figure 2 presents two circular derivation trees, in the two-sided μMALL$^\infty$ sequent calculus[2].

These examples correspond (at a somehow informal level) to the Coq coinductive terms `drop` and `incdrop` of Fig. 1. The computational interpretation of the left-hand derivation is that of a function from streams of nats to streams of nats which drops its elements in odd position, keeping half of its elements only[3]. The rightmost proof has a slightly different computational interpretation: it drops one element every two but also increments the other element that is returned in the output stream. This is achieved by using a cut in the proof, depicted in the box, which corresponds to `hdinc` and does the increment. Although both proofs are productive when they are cut with streams of nats, only the leftmost proof is valid for [7] (up to axiom expansion): the rightmost proof has no valid thread inhabiting the infinite branch because of the cut with the boxed sub-proof occurring in a cycle.

Towards bouncing threads. This paper improves the compositionality of circular proofs, contributing to a line of research aiming at providing and analyzing the computational content of circular proofs. From the Curry-Howard perspective, considering more relaxed validity criteria is an interesting and important challenge as *more circular proofs means more flexibility to write valid programs on coinductive types.*

Indeed, while the previous related cut-elimination results [7, 18] were significant steps, they suffered from strong restrictions on the use of cuts along non-wellfounded branches (or cycles in proofs) as described above. We introduce here a new validity condition for μMALL$^\infty$, the infinitary proof system for MALL with fixed points. Taking inspiration from Geometry of Interaction [21], this criterion generalizes the existing one by enriching the structure of threads: bouncing threads can leave the branch they validate and "bounce" (*i.e.* change direction, moving upward but also downward along proof branches) on axioms and cut rules.

The circular derivation π_0 shown below illustrates the intuitive idea behind validation by bouncing threads: it is not valid according to straight threads since its only infinite branch contains no infinite thread at all, but it will be bouncing-valid (or "b-valid"). Remark that one can trace the coinductive progress by following $\nu X.X$ upwards and $\mu X.X$ downwards while changing directions and moving from a formula to its dual when reaching axioms and cuts. This is formalized by the bouncing thread represented in red. Indeed, after reducing twice the cut in each repetition of the cycle, it yields a cut-free proof which is validated by a (straight) thread, which can be viewed as the "straightened" version of the above mentioned *bouncing thread*:



We investigate the proof-theoretic properties of this new bouncing validity criterion for μMALL$^\infty$ pre-proofs. We prove that it guarantees soundness (Theorem 5.4) and productivity of the cut-elimination process (Theorem 5.1). The criterion is compatible with simple compositions using cuts, as shown in Example 1.1. Even when considering straight threads only, our work already extends the results of [7, 18], as we provide a treatment of both axioms and multiplicatives. Dealing with the axioms actually introduces substantial difficulties in the soundness and cut-elimination proofs.

Decidability properties of the bouncing criterion. In Theorem 6.2, we show that this new bouncing validity condition is undecidable, already in the purely multiplicative case. The proof is intricate, and works by reducing the halting problem of two-counter machines into our bouncing criterion. The strong constraints on the proof system give rise to complex gadgets to propagate relevant information about the current configuration of the machine. Moreover, the linearity of the proof system forces a "reversibility" constraint on this encoding, which can be dealt with by using the same kind of technique as Bennett [8] to prove Turing-completeness of reversible Turing machines: a history of the computation is produced to guarantee reversibility, then this history is erased by rewinding the computation.

Although the criterion is undecidable, we show (Theorems 6.5 and 6.6) that it can be decomposed into a hierarchy of decidable criteria, via a parameter called "height": for each fixed height $k \in \mathbb{N}$, the "$b(k)$-criterion", where height is

[2]We follow this convention for the example in order to exhibit more clearly the computational interpretation, even though the rest of the paper will be developed in the one-sided sequent calculus which is more concise.
[3]Notice that N is erasable and duplicable *on the left* in μMALL$^\infty$, hence the use of the derivable (WNat$_l$) rule, which allows to drop one nat every two.

```
CoInductive Stream := Cons : (nat * Stream) → Stream.
CoFixpoint drop (s : Stream) : Stream := match s with
  | Cons (a, Cons (b, s')) ⇒ Cons (b, (drop s')) end.
Definition hdinc (s: Stream) : Stream := match s with
  | Cons (a, s') ⇒ Cons (S a, s') end.
CoFixpoint incdrop (s : Stream) : Stream := match s with
  | Cons (a, Cons (b, s')) ⇒
hdinc (Cons (b, incdrop s')) end.
                    (***********)
CoInductive BStream := BCons : (bool * BStream) → BStream.
Definition neghd (s: BStream) : BStream := match s with
  | BCons (a, s') ⇒ BCons (negb a, s') end.
CoFixpoint filter1everyk (m : nat) (s : BStream) : BStream :=
  match (m, s) with
  | (0, BCons (a, s')) ⇒ BCons (a, filter1everyk k s')
  | (S m', BCons (a, s')) ⇒ neghd (filter1everyk m' s')
end.
```

Figure 1. Examples of coinductive definitions in Coq.

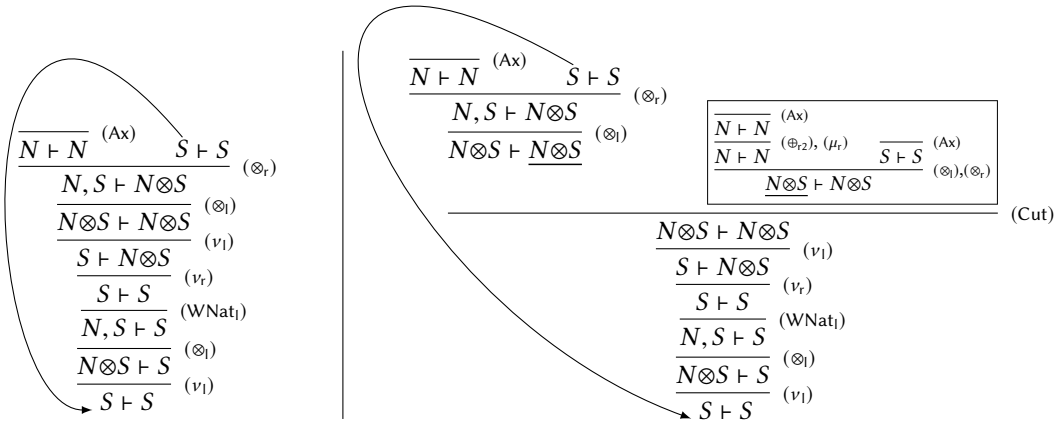


Figure 2. Examples of a valid and an invalid circular pre-proof.

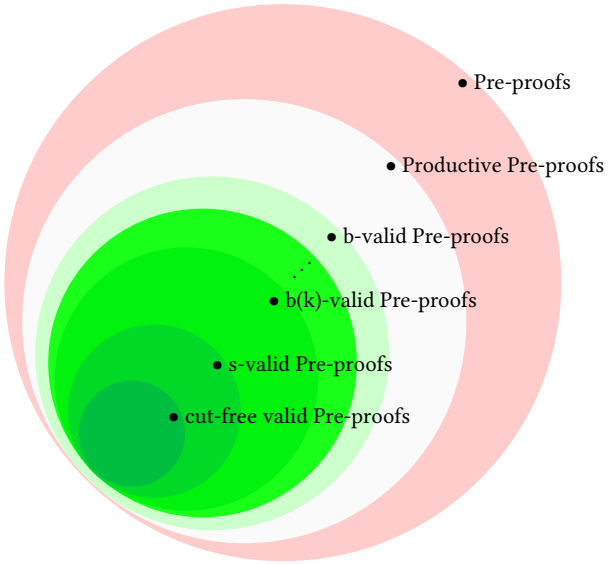


Figure 3. Hierarchy of validity conditions

bounded by k, is decidable. Moreover, any b-valid circular proof is $b(k)$-valid for some $k \in \mathbb{N}$. This allows us to show that the general b-validity criterion is Σ^0_1-complete, i.e. recursively enumerable. The hierarchy of different criteria is represented in Figure 3.

Organization of the contributions. In Section 2 we recall the basic definitions for the non-wellfounded proof system $\mu\mathsf{MALL}^\infty$. We then define, in Section 3 the cut-elimination procedure. In Section 4, we introduce our new bouncing validity condition, and show in Section 5 that it guarantees soundness of the system and productivity of the cut-elimination procedure. We finally study in Section 6 the decidability of our criterion in the multiplicative case. Omitted proofs and developments can be found in appendices.

2 The pre-proofs of $\mu\mathsf{MALL}^\infty$

In this section we introduce the multiplicative additive linear logic extended with least and greatest fixed point operators, and a system of infinitary (pre-)proofs for that logic.

Definition 2.1. Given infinite sets of ***atoms*** $\mathcal{A} = \{a, b, \dots\}$ and of ***fixed-point variables*** $\mathcal{V} = \{X, Y, \dots\}$, $\mu\mathsf{MALL}^\infty$-***pre-formulas*** are built over the following syntax:

$$\begin{array}{lcll} \varphi, \psi & ::= & a \mid a^\perp \quad a \in \mathcal{A}, & \text{(atoms)} \\ & \mid & \mu X.\varphi \mid \nu X.\varphi \mid X \quad X \in \mathcal{V} & \text{(fixed points)} \\ & \mid & \bot \mid \mathbf{1} \mid \varphi \parr \psi \mid \varphi \otimes \psi & \text{(multiplicatives)} \\ & \mid & \mathbf{0} \mid \top \mid \varphi \oplus \psi \mid \varphi \& \psi. & \text{(additives)} \end{array}$$

The connectives μ and ν bind the variable X in φ. $\mu\mathsf{MALL}^\infty$-***formulas*** are those pre-formulas without free fixed point variables. The formulas of $\mu\mathsf{MLL}^\infty$, the multiplicative fragment, are those $\mu\mathsf{MALL}^\infty$ formulas which do not contain $\&, \oplus, \top$ or $\mathbf{0}$ (*ie.* omitting the last line in the grammar).

Definition 2.2 (Negation)**.** $(\_)^\perp$ is the involution on formulas satisfying: $(a^\perp)^\perp = a$; $X^\perp = X$; $(\nu X.\varphi)^\perp = \mu X.\varphi^\perp$; $\bot^\perp = \mathbf{1}$; $(\varphi \parr \psi)^\perp = \varphi^\perp \otimes \psi^\perp$; $(\varphi \oplus \psi)^\perp = \varphi^\perp \& \psi^\perp$.

Setting $X^\perp = X$ would be incorrect when considering formulas with free variables, but it yields the proper dualization for closed formulas, e.g. $(\mu X.X)^\perp = \nu X.X$. Since negation is not a connective, formulas enjoy the positivity condition by construction: all fixed-point expressions are monotonic.

There are several presentations of sequents in the literature. Considering that we are aiming at the proofs-as-program correspondence we could take sequents as lists of formulas or as sets of occurrences of formulas[4]. Here, we make the choice of sequents as named formulas that we call *formula occurrences*, following [7].

We recall next their formal definition. A formula occurrence is the pair of a formula and an address. In a derivation, all the conclusion (and cut) formula occurrences will have pairwise distinct and incomparable addresses: this invariant

[4]Details on the formulations of sequents can be found in Appendix A.1.

will be preserved by each inference. When a rule is applied to a formula occurrence, the addresses of its sub-occurrences will be extended by $\{\mathtt{l}, \mathtt{r}, \mathtt{i}\}$ (standing for left, right and inside respectively) in order to record their provenance. This is of great importance for our developments: the validity criterion traces the evolution of formulas, which is completely explicit in their addresses.

Definition 2.3. Let $\mathcal{A}_{fresh}$ be an infinite set of ***atomic addresses***, $\mathcal{A}^{\perp}_{fresh} = \{\alpha^{\perp}_{\mathrm{at}} \mid \alpha_{\mathrm{at}} \in \mathcal{A}_{fresh}\}$, and $\Sigma = \{\mathtt{l}, \mathtt{r}, \mathtt{i}\}$. An ***address*** is a word of the form $\alpha_{\mathrm{at}}.w$, where $\alpha_{\mathrm{at}} \in \mathcal{A}_{fresh} \cup \mathcal{A}^{\perp}_{fresh}$ and $w \in \Sigma^*$. Let us call *Addr* the set of addresses. We say that α' is a ***sub-address*** of α when α is a prefix of α', written $\alpha \sqsubseteq \alpha'$. We say that α and β are ***disjoint*** when α and β are incomparable wrt. $\sqsubseteq$.

Definition 2.4. A ***formula occurrence***, or simply ***occurrence***, is given by a formula φ and an address α, and written φ_α. Occurrences will be denoted by F, G, H. Occurrences are ***disjoint*** when their addresses are. The occurrences φ_α and ψ_β are ***structurally equivalent***, written $\varphi_\alpha \equiv \psi_\beta$, if $\varphi = \psi$.

A ***sequent*** is a set of disjoint occurrences.

Example 2.5. $\dfrac{\vdash \varphi_{\alpha\mathtt{l}}, \varphi_{\alpha\mathtt{r}}, \psi_\beta}{\vdash (\varphi \parr \varphi)_\alpha, \psi_\beta}\ (\parr)$ is an example of an occurrence-based inference (α and β are assumed to be disjoint). Note that the relation of sub-address is the inverse of the prefix relation, which is coherent with the sub-formula relation. For instance, in the example above, φ is a sub-formula of $\varphi \parr \varphi$, but its address is $\alpha\mathtt{r}$ while the address of $\varphi \parr \varphi$ is α.

We now define the rules of linear logic with fixed points in the framework of sequents as sets of occurrences. As seen above, inferences are address-sensitive: they look at the structure of the formula underlying an occurrence, decompose it following a standard μMALL rule, then assign addresses to its subformulas in the obvious way. One can make the address implicit in the inference by defining the syntax of $\mu\mathsf{MALL}^\infty$ to operate directly on occurrences:

Definition 2.6. Logical connectives are lifted to operations on occurrences as:

- For any $\star \in \{\parr, \otimes, \oplus, \&\}$, if $F = \varphi_{\alpha\mathtt{l}}$ and $G = \psi_{\alpha\mathtt{r}}$ then $F \star G = (\varphi \star \psi)_\alpha$.
- For any $\sigma \in \{\mu, \nu\}$, if $F = \varphi_{\alpha\mathtt{i}}$ then $\sigma X.F = (\sigma X.\varphi)_\alpha$.

Definition 2.7. We define a duality over *Addr* by setting $(\alpha.w)^\perp = \alpha^\perp.w$ and $(\alpha^\perp.w)^\perp = \alpha.w$ for all $\alpha \in \mathcal{A}_{fresh}$ and $w \in \Sigma^*$. We then define $(\varphi_\alpha)^\perp = (\varphi^\perp)_{\alpha^\perp}$, and write $F \perp G$ when $F^\perp = G$. We define substitution over occurrences as follows: $(\varphi_\alpha)[\psi_\beta/X] = (\varphi[\psi/X])_\alpha$.

We are ready to introduce our infinitary sequent calculus.

Definition 2.8. A $\mu\mathsf{MALL}^\infty$ ***pre-proof*** is a possibly infinite tree, coinductively generated by the rules of Fig. 4. Given a sequent s in a pre-proof π, we denote by $\mathrm{premiss}(s)$ the set of sequents which are premisses of the rule of conclusion s in π. Rules other than (Ax) and (Cut) are called ***logical rules***. For every instance of one such rule we call ***principal occurrence*** the occurrence in its conclusion sequent that is decomposed to obtain the premisses.

The infinite derivations of $\mu\mathsf{MALL}^\infty$ may be quite complex trees, possibly not even computable. In practical uses one would turn to sub-systems, typically the fragment of circular pre-proofs [11, 17, 18, 35]. In a nutshell, a circular derivation is an infinite derivation which has only finitely many distinct sub-trees up to renaming of addresses [16].

Notation 1 (Two-sided notation)**.** *While it is proof-theoretically convenient to work with one-sided sequents as in the previous definition, it is more illuminating for some examples, especially when aiming at illustrating the computational interpretation of some proofs, to allow to use the usual two-sided sequent calculi. In the following (and in the examples of the introduction), two-sided sequents may be used:*

$$F_1, \ldots, F_n \vdash \Gamma \text{ should be read as } \vdash \Gamma, F_1^\perp, \ldots, F_n^\perp.$$

Regarding the labelling of inference rules, we allow ourselves two conventions: either the inference rules are written with the labels introduced in Fig. 4 or, as in the introductory example, we use their two-sided names, for instance $(\otimes_{\mathsf{l}})$ *and* $(\otimes_{\mathsf{r}})$*, in which case this is a notation for the corresponding rule in the one-sided sequent calculus, respectively* $(\parr)$ *and* $(\otimes)$ *here.*

Example 2.9. In Fig.5, π_{succ} & π_{dup} illustrate pre-proofs.

Pre-proofs are obviously unsound: any sequent can be derived. Hence, a validity condition shall be required for a pre-proof to be a proof. First we define cut-reduction.

3 The cut elimination process

In this section we introduce the cut-elimination rules for $\mu\mathsf{MALL}^\infty$ pre-proofs. In general, the cut-elimination procedure is not productive. However, we will show in Section 5 that when we restrict to valid pre-proofs (that will be defined in Section 4), the process is productive and outputs a valid pre-proof.

3.1 The multicut rule

In finitary proof theory, cut elimination may proceed by reducing topmost cuts. In the infinitary setting however, by non-wellfoundedness, there is no such thing, in general, as a topmost cut inference. In [7, 18], this issue is dealt with by reducing ***bottom-most cuts***, and when encountering during the reduction a cut which is immediately above another one, instead of permuting two consecutive cuts, merging them into a new rule called ***multicut*** and noted (mcut). A multicut can be seen as a meta-rule to represent a finite tree of cuts.

We will also use this multicut approach[5], but we now have to deal with axiom/cut reductions. This leads us to enrich

[5]Note that there are various approaches to cut-elimination in infinitary settings, for non-wellfounded derivations or for logics including an Ω-rule, in particular Mints continuous cut-elimination [31].

$$\frac{\vdash F, G, \Gamma}{\vdash F \parr G, \Gamma}\ (\parr) \qquad \frac{\vdash \Gamma}{\vdash \bot, \Gamma}\ (\bot) \qquad \frac{\vdash F, \Gamma \quad \vdash G, \Gamma}{\vdash F \& G, \Gamma}\ (\&) \qquad \frac{}{\vdash \top, \Gamma}\ (\top) \qquad \left|\ \frac{\vdash G[\nu X.G/X], \Gamma}{\vdash \nu X.G, \Gamma}\ (\nu) \right. \qquad \left|\ \frac{F \equiv G}{\vdash F, G^\perp}\ (\text{Ax}) \right.$$

$$\frac{\vdash F, \Gamma \quad \vdash G, \Delta}{\vdash F \otimes G, \Gamma, \Delta}\ (\otimes) \qquad \frac{}{\vdash \mathbf{1}}\ (\mathbf{1}) \qquad \frac{\vdash F_i, \Gamma}{\vdash F_1 \oplus F_2, \Gamma}\ (\oplus_i),\ i \in \{1,2\} \qquad \text{(no rule for } \mathbf{0}) \qquad \left|\ \frac{\vdash F[\mu X.F/X], \Gamma}{\vdash \mu X.F, \Gamma}\ (\mu) \right. \qquad \left|\ \frac{\vdash \Gamma, F \quad \vdash F^\perp, \Delta}{\vdash \Gamma, \Delta}\ (\text{Cut}) \right.$$

Figure 4. Rules of the proof system μMALL^∞

$$\pi_{\text{succ}} \quad = \quad \dfrac{\dfrac{}{N \vdash N''}\ (\text{Ax})}{\dfrac{N \vdash \mathbf{1} \oplus N''}{N \vdash N'}\ (\mu)}\ (\oplus_2) \qquad\qquad \pi_{\text{dup}} \quad = \quad \dfrac{\dfrac{\dfrac{}{\vdash N_1}\ (\mu),(\oplus_1),(\mathbf{1}) \qquad \dfrac{}{\vdash N_2}\ (\mu),(\oplus_1),(\mathbf{1})}{\mathbf{1} \vdash N_1 \otimes N_2}\ (\bot),(\otimes) \qquad \dfrac{N' \vdash N_1' \otimes N_2' \qquad \dfrac{\pi_{\text{succ}} \qquad \pi_{\text{succ}}}{N_1' \otimes N_2' \vdash N_1 \otimes N_2}\ (\parr),(\otimes)}{N' \vdash N_1 \otimes N_2}\ (\text{Cut})}{N \vdash N_1 \otimes N_2}\ (\nu),(\&)$$

Figure 5. Examples of pre-proofs π_{succ} and π_{dup}.

the structure of multicuts, by allowing those to perform a renaming. A multicut is a rule written as:

$$\frac{\vdash \Gamma_1 \quad \dots \quad \vdash \Gamma_n}{\vdash \Gamma}\ \text{mcut}(\iota, \perp\!\!\!\perp)$$

and comes with a function ι which shows how the occurrences of the conclusion are distributed over the premisses (modulo renaming), and a relation $\perp\!\!\!\perp$ specifying which occurrences are cut-connected. Below is an example of a multicut rule: the function ι is represented by the red lines, the relation $\perp\!\!\!\perp$ is represented by the blue ones.

$$\frac{\vdash F', G \qquad \vdash G^\perp, H \qquad \vdash H^\perp, K}{\vdash F, K}\ \text{mcut}(\iota, \perp\!\!\!\perp)$$

Precise definitions and more explanations are given in appendix A.2. Later, if clear from the context, we omit to specify ι and $\perp\!\!\!\perp$ in the rule name.

From now, we add the multicut rule to our proof system.

Definition 3.1. We call $\mu\text{MALL}^\infty_{\text{m}}$ the infinitary proof system obtained from μMALL^∞ by adding the multicut rule.

3.2 Reduction rules and strategy

The reduction rules are the same as in [7, 18], adapting them in a straightforward way to account for the extra labellings $\iota, \perp\!\!\!\perp$ in multicut rules. We give examples of such reductions in this section.

There are two kinds of cut reductions: *external* ones that push the multicut deeper in the pre-proof (Example in fig. 6.a), and *internal* ones, that keep the multicut at the same level, and are not productive (Example in fig. 6.b). The rules in the first category are said to be *productive*, since they contribute to the output of the process. Intuitively, the cut-elimination process succeeds if infinitely many productive rules occur on each branch of the proof. An exhaustive description of the $\mu\text{MALL}^\infty_{\text{m}}$ cut-reduction rules is given in appendix A.3.

We now describe a procedure to eliminate cuts from μMALL^∞ proofs, using as an intermediary framework the system with multicuts. We start by embedding μMALL^∞ in $\mu\text{MALL}^\infty_{\text{m}}$ by adding a unary multicut at the root of the pre-proof, with the identity as ι and $\perp\!\!\!\perp = \emptyset$. We then apply internal and external reduction rules to this multicut. We will require reduction sequences to be ***fair***, in the sense that every redex is eventually fired.

The next section introduces the validity condition that will guarantee productivity of this cut elimination process.

4 Bouncing threads and pre-proof validity

We now formally introduce our bouncing threads and the corresponding notion of validity for pre-proofs. Given an alphabet A, we denote by A^ω the set of infinite words over A, and define A^∞ to be $A^* \cup A^\omega$. We will make use of the letter λ to denote ordinals in $\omega + 1$, i.e. either ω or a finite ordinal in $\mathbb{N}$. For such an ordinal, recall that $1 + \lambda = \lambda$ iff $\lambda = \omega$. Finally, we will make use of a special concatenation: given $u = (u_i)_{i \le n < \omega}$ and $v = (v_i)_{i \in \lambda}$ such that $u_n = v_0$, we define $u \odot v$ as the standard concatenation of u and v without its first element, i.e. $u \cdot (v_i)_{i \in \lambda \setminus \{0\}}$. For example $aba \odot aab = abaab$.

4.1 Threads

We start with a naive notion of pre-thread, defined as a sequence of pointed sequents (i.e. sequents with a marked formula) with a direction: a pre-thread follows occurrences in consecutive sequents, travelling up- or downwards.

Definition 4.1. A **pre-thread** is a sequence $(F_i, s_i, d_i)_{i \in \lambda}$ of tuples of a formula, a sequent and a direction, such that for all $i \in \lambda$, $F_i \in s_i$, $d_i \in \{\uparrow, \downarrow\}$ and if $i + 1 \in \lambda$ then one of the following clauses holds:

- $d_i = d_{i+1} = \uparrow$, $s_{i+1} \in \text{premiss}(s_i)$, and $F_{i+1} \sqsubseteq F_i$;
- $d_i = d_{i+1} = \downarrow$, $s_i \in \text{premiss}(s_{i+1})$, and $F_i \sqsubseteq F_{i+1}$;
- $d_i = \downarrow$, $d_{i+1} = \uparrow$, s_i and s_{i+1} are the two premisses of the same cut rule, and $F_i = F_{i+1}^\perp$;
- $d_i = \uparrow$, $d_{i+1} = \downarrow$ and $s_i = s_{i+1} = \{F_i, F_{i+1}\}$ is the conclusion of an axiom rule (so that $F_i \equiv F_{i+1}^\perp$).

(a) Example of an external reduction rule:

$$\dfrac{\mathcal{C} \qquad \dfrac{\vdash \Delta, F'[\mu X.F'/X]}{\vdash \Delta, \mu X.F'}\,(\mu)}{\vdash \Sigma, \mu X.F}\,(\text{mcut}) \quad\longrightarrow\quad \dfrac{\dfrac{\mathcal{C} \qquad \vdash \Delta, F'[\mu X.F'/X]}{\vdash \Sigma, F[\mu X.F/X]}\,(\text{mcut})}{\vdash \Sigma, \mu X.F}\,(\mu)$$

(b) Examples of internal reduction rules:

$$\dfrac{\mathcal{C} \qquad \dfrac{\vdash \Delta, F \qquad \vdash \Gamma, F^\perp}{\vdash \Delta, \Gamma}\,(\text{Cut})}{\vdash \Sigma}\,(\text{mcut}) \quad\longrightarrow\quad \dfrac{\mathcal{C} \qquad \vdash \Delta, F \qquad \vdash F^\perp, \Gamma}{\vdash \Sigma}\,(\text{mcut})$$

$$\dfrac{\mathcal{C} \qquad \dfrac{\vdash \Delta, F[\mu X.F/X]}{\vdash \Delta, \mu X.F}\,(\mu) \qquad \dfrac{\vdash F'^\perp[\nu X.F'^\perp/X], \Gamma}{\vdash \nu X.F'^\perp, \Gamma}\,(\nu)}{\vdash \Sigma}\,(\text{mcut}) \quad\longrightarrow\quad \dfrac{\mathcal{C} \qquad \vdash \Delta, F[\mu X.F/X] \qquad \vdash F'^\perp[\nu X.F'^\perp/X], \Gamma}{\vdash \Sigma}\,(\text{mcut})$$

Figure 6. Examples of external and internal reduction rules.

If $\lambda = n + 1$ is finite we call F_0 and F_n the **endpoints** of the pre-thread.

Example 4.2. Consider the formulas $\varphi = \nu X.X$, $F = \varphi_\alpha$, $F' = \varphi_\beta$, $F'' = \varphi_{\beta.\mathtt{i}}$ where α and β are disjoint addresses. Let G, G' be two disjoint occurrences such that $G \equiv G'$. Below, the red and blue lines are two pre-threads[6]:

$$\dfrac{\dfrac{}{\vdash F, F'^\perp}\,(\text{Ax}) \qquad \dfrac{}{\vdash G, G'^\perp}\,(\text{Ax})}{\vdash F \parr G, F'^\perp \otimes G'^\perp}\,(\parr, \otimes) \qquad \dfrac{\dfrac{\vdots}{\vdash F'', G'}}{\vdash F', G'}\,(\nu)}{\vdash F \parr G, F' \parr G'}\,(\parr) \qquad \vdash F \parr G\,(\text{Cut})$$

We shall define threads as pre-threads satisfying a particular condition that will make them compatible with cut reduction, in the sense that they will have residuals after cut-elimination steps. In Example 4.2, the red thread has no residual if one performs a cut elimination step on $F' \parr G'$, because it comes from the right-hand subformula of $F'^\perp \otimes G'^\perp$ and goes to the left-hand subformula of $F' \parr G'$. In contrast, the blue thread can meaningfully be simplified to persist over cut elimination steps: its residual is well-defined. Geometry of Interaction [21] provides a formalization of these notions, assigning weights to pre-threads and determining which weights correspond to meaningful computations. We follow this inspiration, adapting it to our framework.

Definition 4.3. Let $t = (F_i, s_i, d_i)_{i \in 1+\lambda}$ be a pre-thread. The **weight** of t is a word $(w_i)_{i \in \lambda} \in \{\mathtt{l}, \mathtt{r}, \mathtt{i}, \bar{\mathtt{l}}, \bar{\mathtt{r}}, \bar{\mathtt{i}}, \mathsf{W}, \mathsf{A}, \mathsf{C}\}^\infty$, written $\mathrm{w}(t)$ and defined as follows. For every $i \in \lambda$ one of the following clauses holds:

- $w_i = x$ if $F_i = \varphi_\alpha$ and $F_{i+1} = \psi_{\alpha x}$ for $x \in \{\mathtt{l}, \mathtt{r}, \mathtt{i}\}$;
- $w_i = \bar{x}$ if $F_i = \varphi_{\alpha x}$ and $F_{i+1} = \psi_\alpha$ for $x \in \{\mathtt{l}, \mathtt{r}, \mathtt{i}\}$;
- $w_i = \mathsf{A}$ if $d_i = \uparrow$ and $d_{i+1} = \downarrow$ (corresponding to bouncing on an axiom rule);
- $w_i = \mathsf{C}$ if $d_i = \downarrow$ and $d_{i+1} = \uparrow$ (corresponding to bouncing on a cut rule);
- $w_i = \mathsf{W}$ if $F_i = F_{i+1}$.

Example 4.4. The blue pre-thread of Example 4.2 has a weight of the form $\mathsf{W}\mathtt{l}\mathsf{W}\mathsf{A}\bar{\mathtt{l}}\mathsf{W}\mathsf{C}\mathtt{l}\mathtt{i}\ldots$.

The weight should be seen as a bracketed expression, where each symbol $\bar{x}$, $x \in \{\mathtt{l}, \mathtt{r}, \mathtt{i}\}$, is an opening bracket with matching closing bracket x. When defining threads from pre-threads, we will be particularly interested in the following classes of well-bracketed words:

Definition 4.5. Let $\mathcal{B}$ and $\mathcal{H}$ be the set of words defined inductively as follows:

$$\mathcal{B} := \mathsf{C} \mid \mathcal{B}\mathsf{W}^*\mathsf{A}\mathsf{W}^*\mathcal{B} \mid \bar{x}\mathsf{W}^*\mathcal{B}\mathsf{W}^*x \qquad \mathcal{H} := \epsilon \mid \mathsf{A}\mathsf{W}^*\mathcal{B}$$

A (finite) pre-thread is called a b-**path** if $\mathrm{w}(t) \in \mathcal{B}$. It is called an h-**path** if $\mathrm{w}(t) \in \mathcal{H}$. It is called an ϵ-**path.** if $\mathrm{w}(t) \in \mathsf{W}^*\mathcal{H}$.

The b-paths start downwards and end upwards: they consist of a series of U-shapes centered around cuts, glued together by axioms. The endpoints of b-paths are negations of each other (up to renaming). The h-paths start and end going upwards, and their endpoints are structurally equivalent (up to renaming). Intuitively, h-paths will be simplified during cut elimination, and eventually disappear completely.

Definition 4.6. A pre-thread t is a thread when it can be written $\odot_{i \in 1+\lambda}(H_i \odot V_i)$ where for all $i \in 1 + \lambda$:

- $\mathrm{w}(H_i) \in \mathcal{H}$ and it is non-empty if $i \neq 0$.
- $\mathrm{w}(V_i) \in \{\mathtt{l}, \mathtt{r}, \mathtt{i}, \mathsf{W}\}^\infty$ and it is non-empty if $i \neq \lambda$;

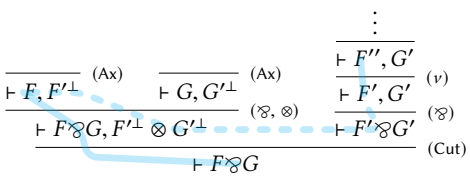
Notice that such a decomposition is unique, except possibly for some neutral W^* factors, for which an arbitrary choice is

[6] ... which respectively correspond to the following sequences:
$t_r = (F \parr G; \vdash F \parr G; \uparrow) \cdot (F \parr G; \vdash F \parr G, F'^\perp \otimes G'^\perp; \uparrow) \cdot (F; \vdash F, F'^\perp; \uparrow) \cdot (F'^\perp; \vdash F, F'^\perp; \downarrow) \cdot (F'^\perp \otimes G'^\perp; \vdash F \parr G, F'^\perp \otimes G'^\perp; \downarrow) \cdot (F' \parr G'; \vdash F' \parr G'; \uparrow) \cdot (F'; \vdash F', G'; \uparrow) \cdot (F''; \vdash F'', G'; \uparrow)$ and
$t_b = (F \parr G; \vdash F \parr G; \uparrow) \cdot (F \parr G; \vdash F \parr G, F'^\perp \otimes G'^\perp; \uparrow) \cdot (G; \vdash G, G'^\perp; \uparrow) \cdot (G'^\perp; \vdash G, G'^\perp; \downarrow) \cdot (F'^\perp \otimes G'^\perp; \vdash F \parr G, F'^\perp \otimes G'^\perp; \downarrow) \cdot (F' \parr G'; \vdash F' \parr G'; \uparrow) \cdot (F'; \vdash F', G'; \uparrow) \cdot (F''; \vdash F'', G'; \uparrow)$.

made[7]. We call $(V_i)_{i\in 1+\lambda}$ the ***visible part*** of t, and we denote it by $\mathrm{vp}(t)$, and $(H_i)_{i\in 1+\lambda}$ its ***hidden part*** and we denote it by $\mathrm{hp}(t)$. A thread is ***stationary*** when its visible part is a finite sequence (of finite words), or when there exists $k \in 1+\lambda$ such that $\mathrm{w}(V_i) \in \{\mathrm{W}\}^\infty$ for all $k \le i \in 1+\lambda$.

For instance if a pre-thread $t = (F_i, s_i, \uparrow)_{i\in\lambda_t}$ of length λ_t goes only upwards ($\mathrm{w}(t) \in \{\mathtt{l}, \mathtt{r}, \mathtt{i}, \mathrm{W}\}^{\lambda_t}$), then the above decomposition is given by $\lambda = 0$, $H_0 = (F_0, s_0, \uparrow)$ and $V_0 = t$.

Example 4.7. Let us consider the blue pre-thread of Example 4.2. We can decompose it into a visible part (plain line) and a hidden part (dashed line) as shown below:



The blue pre-thread is then indeed a thread. On the contrary, the red pre-thread from example 4.2 admits no such decomposition.

If we consider the sequence of formulas followed by a non-stationary thread on its visible part, ignoring its hidden parts (which have equivalent formulas on their endpoints), and skipping the steps in the visible parts corresponding to W weights, we obtain an infinite sequence of formulas as in [7] where each formula is an immediate subformula or an unfolding of the previous formula. It is then well known [16] that the formulas appearing infinitely often in that sequence admit a minimum w.r.t. the subformula ordering. We call this formula the ***minimal formula of the thread***.

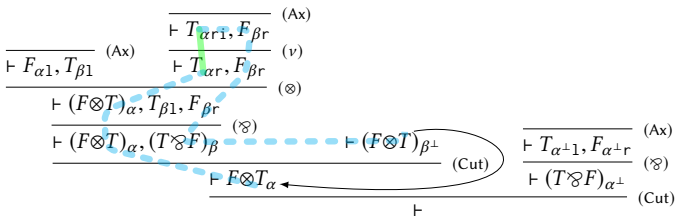
Definition 4.8. A non-stationary thread is ***valid*** if its minimal formula is a ν-formula.

Consider for example the formula $F = \mu X.\nu Y.X$. The minimal formula obtained by unfolding F infinitely often is F itself, a μ-formula, so the corresponding thread is invalid.

4.2 Pre-proof validity: the multiplicative case

The previous notion of valid thread suggests a first extension of the notion of valid proof based on straight threads [7]: one might say that a branch β is valid when there is a valid bouncing thread which meets β infinitely often, and declare a pre-proof valid when all its branches are. However, this notion of *weak validity* turns out to allow unsound proofs, as shown next.

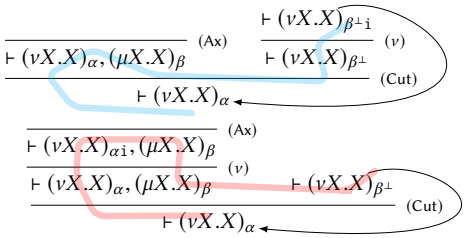
Example 4.9. Let $T := \nu X.X$ and $F := \mu X.X$. The following is a weakly valid proof of the empty sequent. The hidden part of the decomposition (and the prefix of the thread up to the first axiom) is dashed, the visible part (except the prefix of the thread up to the first axiom) is shown in green. The minimal formula along the thread is T.



A proper notion of validity must therefore be more constraining. We shall consider the following one, which requires that the visible part of the valid thread t is contained in the infinite branch β.

Definition 4.10. Let π be a $\mu\mathsf{MLL}^\infty$ pre-proof. An infinite branch β of π is said to be ***valid*** if there is a valid thread t starting from one of its sequents, whose visible part is contained in this branch. A $\mu\mathsf{MLL}^\infty$ ***proof*** is a $\mu\mathsf{MLL}^\infty$ pre-proof in which every infinite branch is valid.

Example 4.11 (valid and invalid pre-proofs)**.**



The topmost pre-proof is valid: its infinite branch is supported by the valid blue thread, whose visible part belongs to the infinite branch. The bottommost pre-proof is not valid, because the red thread, though valid, has a visible part that is not contained in the infinite branch.

4.3 Pre-proof validity: accommodating the additives

The previous definition of validity is too weak to ensure cut-elimination for $\mu\mathsf{MALL}^\infty$, which is not a strictly linear sequent calculus (as $\mu\mathsf{MLL}^\infty$ is) since commutation/external reductions for the $(\&)$ connective induce the duplication of a sub-proof. As a result, the extension of the validity condition in Section 4.2 fails to ensure productivity and validity of cut-elimination as shown in figure 7.(i). The result of cut-elimination on the proofs in the sequence $(\pi'_k)_{k\ge 0}$ can be split into the following cases:
(i) from π'_0, cut-elimination is productive and produces a valid cut-free proof;
(ii) from π'_1, cut-elimination produces an *invalid* pre-proof (see Figure 7.(ii)): any infinite branch following only finitely many times the left back-edge is invalid;
(iii) from π'_k, for $k \ge 2$ it is not even productive. Indeed, in these examples, each π'_k contains exactly one infinite branch which is supported by a thread on T bouncing on the left-most axiom and this thread is valid.

[7] It can happen that a W^* factor can be put either in the visible part or the hidden part. Since such choices will play no role in the following, we can make now an arbitrary choice for these factors, and consider the decomposition unique.

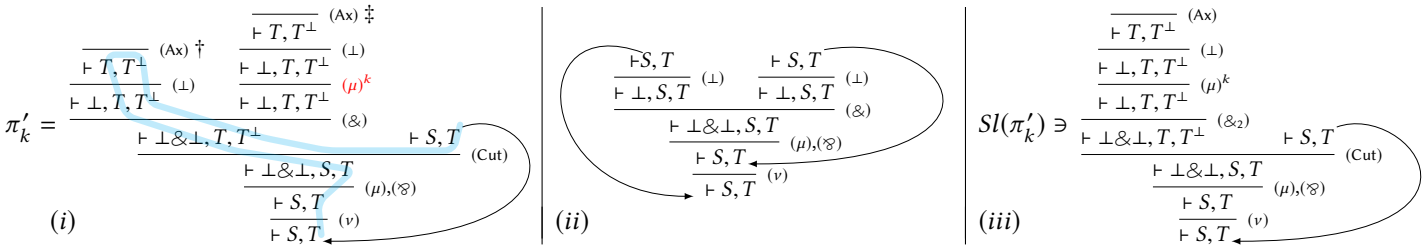


Figure 7. (i) Pre-proof family $(\pi'_k)_{k\in\mathbb{N}}$ with $S = \mu Y.((\bot\&\bot)\parr Y), T = \nu X.X$. Note that we omit the occurrences and that k is a parameter fixing how many times the μ rule (in red) should be applied to the sequent $\vdash \bot, T, T^\bot$. (ii) Result of applying (infinitary) cut-elimination to π'_1. (iii) Example of a slice of π'_k.

To understand the problem, consider the first step of cut-reduction (from π'_k, for any k): it is a (Cut)/(&) commutation step, which copies the right-premiss of the cut (*ie.* the non-wellfounded part of the proof): after this step, the pre-proof contains two infinite branches, but only one thread to validate them. While the leftmost copy can be validated by the original thread, the rightmost copy does not contain a residual of the original thread. Of course, one might consider a thread originated in the cut inference, but that will not suffice to ensure validity, nor productivity, as π'_2 exemplifies: its rightmost branch produces the bottom rule.

Sliced proof system. This issue is solved by refining the criterion using slices [20, 22, 23, 38] and requiring that there exists a supporting thread not only for every infinite branch of the proof, but also for every infinite branch *of every persistent slice* of the pre-proof. In linear logic, an additive slice is a subtree of a sequent proof obtained by removing, for any of its (&) inference, the subtree rooted in one of its premisses (see Appendix A.5 for details and precise definitions).

Definition 4.12. μSMALL$^\infty$ is obtained by extending μMALL$^\infty$ with the following three inference rules:

$$\frac{\vdash A, \Gamma}{\vdash A\&B, \Gamma}\ (\&_1) \qquad \frac{\vdash B, \Gamma}{\vdash A\&B, \Gamma}\ (\&_2) \qquad \frac{}{\vdash \Gamma}\ (\Omega)$$

Definition 4.13 (Additive slice)**.** ***Partially sliced*** pre-proofs are the non-wellfounded μSMALL$^\infty$ pre-proofs. A ***slice*** is a (&)-free, (Ω)-free, μSMALL$^\infty$-pre-proof.

To a μMALL$^\infty$ sequent (pre-)proof, one can associate a set of slices by keeping, for each (&) inference, only one of its premisses and replacing the (&) with the corresponding inference in $(\&_1), (\&_2)$. More precisely:

Definition 4.14 (Slicing of a pre-proof)**.** The set of ***slices of*** π, $Sl(\pi)$, is defined corecursively by

$$Sl\left(\dfrac{\dfrac{\pi_1}{\vdash A_1, \Gamma} \quad \dfrac{\pi_2}{\vdash A_2, \Gamma}}{\vdash A_1\&A_2, \Gamma}\ (\&)\right) = \left\{ \dfrac{\dfrac{\pi'_i}{\vdash A_i, \Gamma}}{\vdash A_1\&A_2, \Gamma}\ (\&_i),\ \begin{array}{c}\pi'_i \in Sl(\pi_i),\\ i \in \{1,2\}\end{array} \right\}$$

(The other inferences are treated homomorphically.)

Example 4.15. Fig. 7.(iii) gives an example of a slice.

Cut-reductions for slices. Cut-reduction rules for (partial) slices of μSMALL$^\infty$ extend those for μMALL$^\infty$ with specific rules for sliced additives and (Ω). A problematic situation is when ($\&_1$) interacts with ($\oplus_2$): cut-elimination cannot rely on sub-proofs as usual, and in this case (Ω) is used to solve this mismatch of inferences.

Definition 4.16 (Cut reductions for slices)**.** The sliced additive principal case is reduced as follows, if $\{A_1^\bot\&A_2^\bot, A'_1 \oplus A'_2\} \in \perp\!\!\!\perp$, with $r = (\text{princ}, \{A_1^\bot\&A_2^\bot, A'_1 \oplus A'_2\})$.

$$\dfrac{C \quad \dfrac{\dfrac{\pi_i}{\vdash A_i^\bot, \Gamma}}{\vdash A_1^\bot\&A_2^\bot, \Gamma}\ (\&_i) \qquad \dfrac{\dfrac{\pi'_j}{\vdash A'_j, \Gamma}}{\vdash A'_1 \oplus A'_2, \Delta}\ (\oplus_j)}{\vdash \Sigma}\ \text{mcut}(\iota, \perp\!\!\!\perp)$$

$$\xrightarrow[r]{} \begin{cases} \dfrac{}{\vdash \Sigma}\ (\Omega) & \text{if } i \neq j \\[2ex] \dfrac{C \quad \dfrac{\pi_i}{\vdash A_i^\bot, \Gamma} \qquad \dfrac{\pi'_i}{\vdash A'_i, \Delta}}{\vdash \Sigma}\ \text{mcut}(\iota, \perp\!\!\!\perp') & \text{if } i = j \\ \text{where } \perp\!\!\!\perp' = \perp\!\!\!\perp \cup \{\{A_i^\bot, A'_i\}\} \end{cases}$$

$$\text{or} \quad \dfrac{C \quad \dfrac{}{\vdash \Gamma}\ (\Omega)}{\vdash \Sigma}\ \text{mcut}(\iota, \perp\!\!\!\perp) \xrightarrow[r]{} \dfrac{}{\vdash \Sigma}\ (\Omega) \quad \text{with } r = (\text{princ}, \Omega).$$

Notions of b-paths and h-paths can be naturally extended to additive slices.

Persistent slices. Persistent slices are introduced precisely as those in which no case of the above mismatch ever occurs:

Definition 4.17 (Persistent slice)**.** Given a slice π, a ($\&_i$) rule of principal formula $A_1\&A_2$ occurring in π is said to be ***well-sliced*** if no b-path starting down from the $A_1\&A_2$ occurrence of this sequent ends in a formula $A_1^\bot \oplus A_2^\bot$ that is the principal formula for a ($\oplus_j$) inference with $i \neq j$. A slice is ***persistent*** if all its ($\&_i$) occurrences are well-sliced.

Example 4.18. The slice in Fig. 7.(iii) is trivially persistent as it contains no $\oplus$ inference. The sliced (&) rule depicted in definition 4.16 is well-sliced if, and only if, $i = j$.

The following two properties of persistent slices are the key for the cut-elimination property (see proof in A.5):

Proposition 4.19. *All reducts of a persistent slice are (Ω)-free, and therefore are slices.*

Proposition 4.20 (Pull-back property)**.** *If $\pi \to^* \pi'$ (resp. $\pi \to^\omega \pi'$) and $S' \in Sl(\pi')$, then there is a $S \in Sl(\pi)$ such that $S \to^* S'$ (resp. $S \to^\omega S'$).*

Additive validity. Def 4.1 and 4.5 of (pre-)threads directly adapt to additive slices – as they are not specific to $\mu\mathsf{MLL}^\infty$– and allow us to consider the following definition:

Definition 4.21. A persistent slice is ***valid*** if it is valid in the sense of Definition 4.8[8]. A $\mu\mathsf{MALL}^\infty$ pre-proof π is ***valid*** if all its persistent slices are valid.

Example 4.22. Pre-proof π_{dup} of Figure 5 is valid, the only infinite branch is validated by a straight. Among the pre-proofs of $(\pi'_k)_{k\in\mathbb{N}}$ given in Figure 7, only π'_0 is valid.

The circular pre-proof of Figure 8 is valid. It corresponds to the last program considered in the introduction.

5 Cut elimination theorem for $\mu\mathsf{MALL}^\infty$

In this section, we shall establish our central result:

Theorem 5.1. *Fair reduction sequences on $\mu\mathsf{MALL}^\infty_{\mathsf{m}}$ proofs produce cut-free $\mu\mathsf{MALL}^\infty$ proofs.*

For expository reasons, we focus on the multiplicative case here. The treatment of additives, while bringing new cases, is similar and can be found in Appendix A.5.

The proof follows the same lines as the proof of [7] for straight threads. We only sketch it here and emphasize the new phenomena due to the presence of axioms and bouncing threads. The full proof can be found in Appendix A.4.

The proof of Theorem 5.1 is in two parts. We first prove that we cannot have an infinite fair reduction sequence made only of (unproductive) internal reductions. Hence cut elimination is productive, i.e., reductions of $\mu\mathsf{MLL}^\infty_{\mathsf{m}}$ proofs converge to cut-free $\mu\mathsf{MLL}^\infty$ *pre*-proofs. We then establish that the obtained pre-proof is a valid proof. In this section, we will only show productivity, validity of the resulting proof is shown in a similar way (See Appendix A.4).

To show productivity, we proceed by contradiction, assuming that there exists a fair infinite sequence of internal reductions from a given proof π of conclusion Γ. We will also assume w.l.o.g. that π has only one multicut at the root. Note that since we perform only internal reduction rules, and since the latter do not duplicate multicuts, there is only one multicut progressing in the proof during this sequence of reductions. In the following, we refer to it as "the" multicut.

5.1 Trace of a reduction sequence

Let us first introduce an important tool to analyse internal reduction sequences, called their ***trace***. Along an internal reduction sequence (π_i) from a proof π, some sequents π become a premise of the multicut rule of some π_i. The trace of an internal reduction sequence is defined as the collection of those sequents together with the conclusion sequent and the corresponding inference rules of the starting proof. By analyzing the reduction rules, it is easy to see that the trace of a proof is its proof tree from which some branches have been pruned and replaced by open leaves:

Proposition 5.2. *Given a $\mu\mathsf{MLL}^\infty$ proof π and a fair reduction sequent ρ, the trace of π after ρ is a subtree (possibly with open leaves) of the original proof π.*

An example of a trace is shown below: sequents not in the trace are grayed.

$$\dfrac{\dfrac{\dfrac{\vdots}{\vdash \mu X.X_{\beta^\perp \mathtt{i}}}}{\vdash \mu X.X_{\beta}}\,(\mu) \qquad \dfrac{\dfrac{\dfrac{\vdots}{\vdash \mu X.X_{\beta^\perp \mathtt{i}}, \mu X.X_\gamma}\,(\nu)}{\vdash \mu X.X_{\beta^\perp \mathtt{i}}, \mu X.X_\gamma}\,(\nu) \qquad \dfrac{\dfrac{\vdots}{\vdash \nu X.X_{\gamma^\perp \mathtt{i}}, \perp_\alpha}\,(\nu)}{\vdash \nu X.X_{\gamma^\perp}, \perp_\alpha}\,(\nu)}{\vdash \nu X.X_{\beta^\perp \mathtt{i}}, \perp_\alpha}\,(\mathrm{Cut})}{\vdash \nu X.X_{\beta^\perp}, \perp_\alpha}\,(\nu)}{\vdash \perp_\alpha}\,(\mathrm{mcut})$$

We shall get a contradiction using the trace in three steps:

1. We will define an extension of the proof system $\mu\mathsf{MLL}^\infty$, and show that it is sound *wrt.* to a boolean semantics.
2. Then we will show that the trace can be seen as a proof of a false sequent in this extended proof system.
3. This contradicts soundness and concludes the proof.

5.2 The trace is almost a $\mu\mathsf{MLL}^\infty$ proof

As said above, we will need to see the trace as a genuine proof. Being a subtree of the original proof π, the trace is almost a $\mu\mathsf{MLL}^\infty$ proof; it may not be a proof for two reasons:

1) The trace may have unjustified sequents: this happens when a sequent S enters the multicut during the reduction sequence but never gets reduced. It will then be part of the trace but the subtree of π rooted in S will not. This is the case of the sequent $\vdash \nu X.X_{\gamma^\perp \mathtt{i}}, \perp_\alpha$ in the example above.

2) The infinite branches of the trace may not be valid: they are of course also infinite branches of the proof π, and thus are supported by valid bouncing threads *of* π. However, since the threads can bounce, they might leave the branch and might not be entirely included in the trace.

We will show later how to handle the first problem of unjustified sequents. As for the second problem, we show that this actually never happens:

Proposition 5.3. *Let T be the trace of a reduction sequence starting from a proof π, and let β be an infinite branch of T. If t is a bouncing thread of π validating β, then t is also a bouncing thread of T.*

This is one of the difficulties specific to the bouncing threads. This result is trivial with straight threads [7], since threads belong to the branch they support.

[8] That is, every infinite branch of the slice is visited by a valid thread having its visible part contained in the branch.

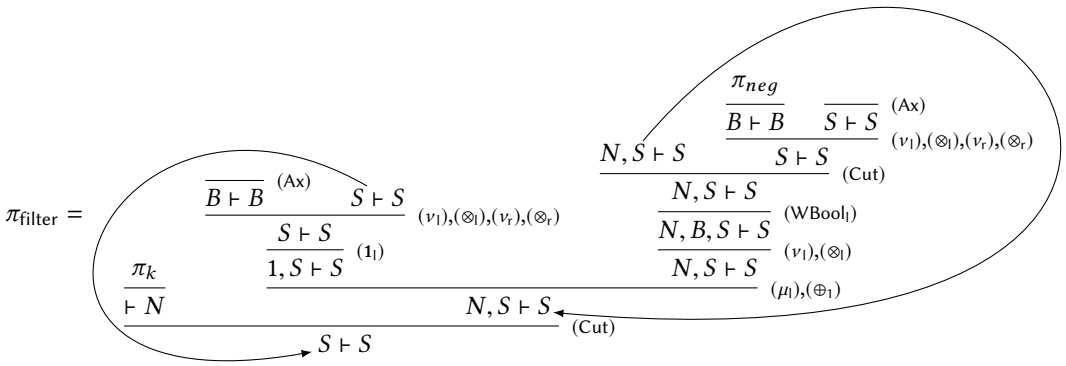


Figure 8. Example of an additive circular proof

5.3 Truncated proof system

To see the trace as a proof, we need to overcome the problem of unjustified sequents. For that, we will embed the trace in a so-called ***truncated proof system***, extending $\mu\mathsf{MLL}^\infty$.

This proof system is parameterized by a partial function $\tau : \mathit{Addr} \to \{\top, \mathbf{0}\}$ (from addresses to the formulas $\top, \mathbf{0}$) called a ***truncation***. To get a sound proof system, we impose a coherence condition on truncations: they should assign dual values to dual addresses. The rules of the truncated proof system are the same as those of $\mu\mathsf{MLL}^\infty$, with an extra rule which allows to replace an occurrence by its image in τ. Pre-proofs and the validity condition are defined in the same way as for $\mu\mathsf{MLL}^\infty$. The advantage of the truncated proof system is that it allows to close sequents easily: if the address of an occurrence of the sequent is mapped to $\top$ by τ, we can justify the sequent by a $\top$ rule.

The boolean semantics can be extended in the presence of truncations in a natural way: the occurrences whose addresses are in the domain of the truncation obtain as a boolean value their image by τ. Boolean values are propagated through the connectives as usual. We show that the truncated proof system is sound for this semantics.

Note that $\mu\mathsf{MLL}^\infty$ can be seen as a truncated proof system, where the truncation has empty domain. The truncated boolean semantics then coincides with the classical boolean semantics. Hence $\mu\mathsf{MLL}^\infty$ is sound for the boolean semantics, a result which can be extended to $\mu\mathsf{MALL}^\infty$:

Theorem 5.4. *$\mu\mathsf{MALL}^\infty$ is sound for the boolean semantics.*

5.4 Trace as a truncated proof

Let us see how to transform the trace into a proof in a truncated proof system. For this, we need to find a truncation τ that can allow us to close every unjustified sequent. In other words, we need to find a strategy for selecting an occurrence in each unjustified sequent, to which we will assign $\top$ by the truncation τ. This strategy should be coherent in the sense that it should not assign $\top$ to two dual occurrences.

In [7], such a strategy is given in a simpler setting: select the formula occurrence in the unjustified sequent that is principal in the proof π. Axioms complicate the situation: if F is the occurrence that has been selected in an unjustified sequent, then its dual might appear in an axiom rule $\vdash F^\perp, G$. By coherence of the truncation, the address of $F^\perp$ must be assigned $\mathbf{0}$ and the axiom rule cannot be soundly applied anymore. To justify $\vdash F^\perp, G$, we need to assign $\top$ to the address of G. Since the same can happen on the G side, we need to show that it remains possible to define τ in a coherent way.

To get our desired contradiction, we need in addition for τ to assign $\mathbf{0}$ to the conclusion, thereby obtaining a proof of a false sequent. This needs to be done while still respecting the aforementioned constraints induced by axioms.

Summary. To sum up, we have found a truncation τ i) which assigns $\mathbf{0}$ to the conclusion formula and ii) for which the trace can be seen as a proof in the corresponding truncated proof system. Since the proof system is sound, we get a contradiction. This concludes the proof of productivity.

6 Decidability properties of $\mu\mathsf{MLL}^\omega$

6.1 An operational approach to threads

In this section, we explain how threads can be recognized by a specific deterministic ω-pushdown automaton reading only the weight of a pre-thread. This allows us to define the *height* of a thread and the notion of *constraint stack*.

Let $\mathcal{A}_{thread}$ be the deterministic ω-pushdown automaton described in Fig. 9, on alphabet $\Sigma = \{\mathtt{l}, \mathtt{r}, \mathtt{i}, \bar{\mathtt{l}}, \bar{\mathtt{r}}, \bar{\mathtt{i}}, \mathtt{A}, \mathtt{C}, \mathtt{W}\}$ and stack alphabet $\Gamma = \{\mathtt{l}, \mathtt{r}, \mathtt{i}, \perp\}$ where $\perp$ is the empty stack symbol. The transitions are labelled "$(a, \gamma) \mid \tau$", where $a \in \Sigma$ is the input letter, $\gamma \in \Gamma$ is the topmost stack symbol, and τ is the action performed on the stack (no action if τ is not specified). If no stack symbol is specified, the stack is left unchanged. Symbol x stands for an element in $\{\mathtt{l}, \mathtt{r}, \mathtt{i}\}$. No acceptance condition is specified: any run is accepting. Only the absence of an available transition can cause the automaton to reject its input, *e.g.* reading $\mathtt{l}$ with topmost stack symbol $\mathtt{r}$ in state $\uparrow$. The transition marked with a double arrow corresponds to the visible part of the thread.

The stack of $\mathcal{A}_{thread}$ is referred to as the *constraint stack*.

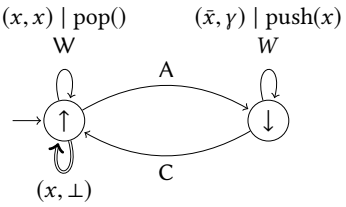


Figure 9. The deterministic ω-pushdown automaton $\mathcal{A}_{thread}$

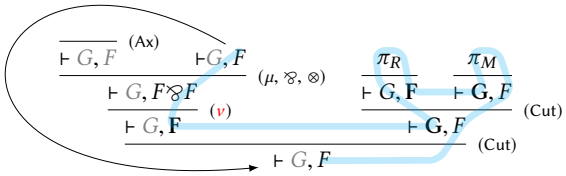


Figure 10. A sketch of the main pre-proof P

Lemma 6.1. *Let t be a pre-thread. Then t is a thread if and only if $\mathsf{w}(t)$ is accepted by $\mathcal{A}_{thread}$.*

Proof. Constraints on the stack of $\mathcal{A}_{thread}$ match the grammar of Def. 4.5. □

6.2 Undecidability of bouncing validity

In this section we present the proof of the following result:

Theorem 6.2. *The bouncing validity condition is already undecidable for $\mu\mathsf{MLL}^{\omega}$.*

This motivates the following section introducing decidable subcriteria constituting a hierarchy of criteria.

To show undecidability, we reduce from the halting problem for Minsky Machines, i.e. two-counter machines (2CM) able to perform increment, decrement, and zero test on the counters. The halting problem for 2CM is known to be Σ^0_1-complete [30].

The proof is only sketched here, and some technicalities have been abstracted away for clarity purposes. See Appendix A.6.1 for exact definitions and encodings.

We encode the halting problem of a 2CM M using a bouncing thread. The thread of interest will always follow a formula $F = \nu X.(X \parr X)$ when going upwards, and its dual $G = \mu X.(X \otimes X)$ when going downwards. The idea is to use the constraint stack to encode the value of counters, and the position in the graph to encode the control state of the machine. The general shape of the main pre-proof P performing the desired reduction is represented in Fig. 10. Boldface formulas are those introduced in cuts, and grayed formulas are the ones that are not part of the thread of interest. We ignore addresses in this sketch, unless relevant to our encoding.

We build P so that the only branch which is not clearly validated is the one going infinitely many times through the loop. A thread validating this branch (in blue in Fig. 10) must go through the two cuts, and bounce on axioms in π_M and π_R. The trajectory of this thread in π_M will simulate the run of M. It will be allowed to exit π_M if and only if M terminates.

We now give an example of one of the simplest gadgets used to perform this simulation: the increment gadget on the first counter. Consider a state p of the machine M, whose action is to increment the first counter and go to state q. Assume counter values (n, m) are encoded by a constraint stack $\mathbf{l}^n\mathbf{r}\mathbf{l}^m\mathbf{r}$, where $\mathbf{l}$ (resp. $\mathbf{r}$) stands for a left (resp. right) constraint on the unfolding of F, i.e. a relative address `il` (resp. `ir`). This means that to increment the first counter, we need to add a left constraint at the top of the stack. This can be performed by the following gadget, where nodes labeled (p) and (q) encode the current control state:

$$\dfrac{(q) \vdash G, \mathbf{F}, A \qquad \dfrac{\dfrac{\dfrac{}{\vdash G_{\mathbf{l}}, F}\,(\mathrm{Ax}) \qquad \dfrac{}{\vdash G_{\mathbf{r}}, A}\,(\infty)}{\vdash G_{\mathbf{l}} \otimes G_{\mathbf{r}}, F, A}\,(\otimes)}{\vdash \mathbf{G}, F, A}\,(\mu)}{(p) \vdash G, F, A}\,(\mathrm{Acut})$$

Here A is an auxiliary formula $\nu X.(X \parr X) \otimes X$, that can be duplicated as required and used to build axiom-less valid proofs, denoted by an (∞) meta-rule. The rule (Acut) denotes a cut combined with a duplication of A. A thread entering node (p) upwards with constraint stack $\mathbf{l}^n\mathbf{r}\mathbf{l}^m\mathbf{r}$ will enter node (q) with constraint stack $\mathbf{l}^{n+1}\mathbf{r}\mathbf{l}^m\mathbf{r}$.

In order to fully simulate the run of M, we also need to design gadgets simulating increment on the second counter, as well as decrement and zero test on both counters. The main difficulty lies in the tests performed by the machine: we want the thread to follow a conditional branching, depending on the value of the constraint stack. This can be done, but because of the linearity of the proof system, we cannot avoid leaving some extra constraints encoding the results of the tests. These "garbage constraints" will be collected by the thread on its path downwards in π_M, after the simulation of the machine is completed. Since we want to finish with empty constraint, we need to erase these garbage constraints. To do this, we add a second gadget π_R performing the computation in a dual way: garbage constraints are fed to the thread, which rewinds the computation while erasing these unwanted constraints. All gadgets in π_R are dual versions of those in π_M. This technique is reminiscent of the one used by Bennett [8] to prove Turing-completeness of reversible Turing machines, where a history of the computation is produced to guarantee reversibility, then this history is erased by rewinding the computation.

We can finally exit this detour with no constraint, and perform a visible ν-unfolding on the main branch (in red in Fig. 10), before looping back to the root of the proof.

The global pre-proof P will be a valid proof according to the criterion if and only if the machine M halts.

Notice that among the simplifications we made here for clarity of exposition, the auxiliary formula A needed in some gadgets has been removed from the main pre-proof P.

6.3 A hierarchy of decidable validity conditions

In order to recover a decidable criterion, we will consider restrictions on the constraint stack of valid threads.

Definition 6.3. If t is a thread, we define its *height* $h(t) \in \omega + 1$ to be the supremum of the size of the stack of $\mathcal{A}_{thread}$ along its run on $\mathsf{w}(t)$.

Definition 6.4. Let $k \in \mathbb{N}$. An infinite branch is *k-valid* if there is a thread of height at most k validating it. A proof P is a *k-proof* if every infinite branch of P is k-valid.

The following two theorems show that the height parameter k induces a hierarchy of decidable criteria – see Figure 3 on page 4 – whose union matches the full validity criterion.

Theorem 6.5. *If P is a valid circular proof of $\mu\mathsf{MLL}^\omega$, there exists $k \in \mathbb{N}$ such that P is a k-proof.*

Theorem 6.6. *Given a circular pre-proof P of $\mu\mathsf{MLL}^\omega$ and an integer k, it is decidable whether P is a k-proof.*

We now give a brief proof sketch to give an intuition on how to prove Theorems 6.5 and 6.6. See Appendix for details.

Proof. (Sketch) We will use the fact that once a starting point for a thread has been chosen, the thread evolves deterministically along the proof tree until a visible event occurs. We define the notion of *minimal shortcut* which is a part of a thread with no visible weight, bouncing on an axiom, and ending in the first point where the constraint stack is empty. It corresponds to an ϵ-path.

By bounding the maximal height of the stack by k, we can detect loops or declare stack overflow, and we are able to compute the unique minimal shortcut (if it exists) for each starting point in the finite proof graph. Now, checking validity of the proof can be done using an algorithm for straight threads [16], allowing them to take these shortcuts.

Theorem 6.5 is obtained by taking the maximal height reached by all minimal shortcuts of the proof graph. □

Combining Theorems 6.5 and 6.6, we obtain that validity of a circular pre-proof of $\mu\mathsf{MLL}^\omega$ is in Σ^0_1, i.e. recursively enumerable. Together with the reduction from Sec. 6.2, we obtain the following corollary:

Corollary 6.7. *The problem of deciding whether a circular pre-proof of $\mu\mathsf{MLL}^\omega$ is a proof is Σ^0_1-complete.*

7 Conclusion

We have studied non-wellfounded and circular proofs of $\mu\mathsf{MALL}^\infty$ and defined an extended validity criterion for the pre-proofs of $\mu\mathsf{MALL}^\infty$ compared to previous works [7, 18]. We have shown that our criterion enjoys cut elimination and soundness, but reaches the barrier of undecidability: in the purely multiplicative fragment already, a parameter has to be bounded by an explicit value to make the criterion decidable. mull For future work, we plan to investigate whether this decidability result still holds when adding the additives.

We also want to extend these results to more relaxed criteria: we conjecture that requiring the visible parts to meet the validated branch infinitely often is sufficient to ensure productivity and soundness, generalizing Theorems 5.1 and 5.4 to a more relaxed validity condition. A less sequential variant of circular proofs has been developed by De *et al.* [14, 15]: the canonicity and absence of commutation rules of proof nets may have good properties with respect to cut-elimination and we expect bouncing validity to be fruitful in that setting.

Finally, the present work is a first step in improving the compositionality of circular proofs. We demonstrated the flexibility of bouncing threads on a number of examples: current work is pursued to develop a proof-term syntax, in the style of system L, for circular $\mu\mathsf{MALL}$. In addition to strengthening our cut-elimination result as mentioned above, we plan to investigate how one can import results from sized types [1] or copattern [4] approach which also have good properties with respect to compositionality and may be used in an infinitary scenario [2, 3].

References

[1] Andreas Abel. 2007. Mixed inductive/coinductive types and strong normalization. In *Asian Symposium on Programming Languages and Systems*. Springer, 286–301.

[2] Andreas Abel. 2016. Compositional Coinduction with Sized Types. (2016). Abstract for the invited talk at the 13th IFIP WG 1.3 International Workshop on Coalgebraic Methods in Computer Science (CMCS 2016), Eindhoven, the Netherlands, 2-3 April 2016.

[3] Andreas Abel and Brigitte Pientka. 2016. Well-founded recursion with copatterns and sized types. *Journal of Functional Programming* 26 (2016), 61. https://doi.org/10.1017/S0956796816000022 ICFP 2013 special issue.

[4] Andreas Abel, Brigitte Pientka, David Thibodeau, and Anton Setzer. 2013. Copatterns: programming infinite structures by observations. In *The 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '13, Rome, Italy - January 23 - 25, 2013*, Roberto Giacobazzi and Radhia Cousot (Eds.). ACM, 27–38. https://doi.org/10.1145/2429069.2429075

[5] David Baelde. 2012. Least and greatest fixed points in linear logic. *ACM Transactions on Computational Logic (TOCL)* 13, 1 (2012), 2.

[6] David Baelde, Amina Doumane, and Alexis Saurin. 2015. Least and Greatest Fixed Points in Ludics. In *24th EACSL Annual Conference on Computer Science Logic, CSL 2015, September 7-10, 2015, Berlin, Germany (LIPIcs)*, Stephan Kreutzer (Ed.), Vol. 41. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 549–566. https://doi.org/10.4230/LIPIcs.CSL.2015.549

[7] David Baelde, Amina Doumane, and Alexis Saurin. 2016. Infinitary Proof Theory: the Multiplicative Additive Case. In *25th EACSL Annual Conference on Computer Science Logic, CSL 2016, August 29 - September 1, 2016, Marseille, France (LIPIcs)*, Vol. 62. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 42:1–42:17. http://www.dagstuhl.de/dagpub/978-3-95977-022-4

[8] C. H. Bennett. 1973. Logical Reversibility of Computation. *IBM J. Res. Dev.* 17, 6 (Nov. 1973), 525–532.

[9] Yves Bertot and Pierre Castéran. 2004. *Interactive Theorem Proving and Program Development - Coq'Art: The Calculus of Inductive Constructions*. Springer. https://doi.org/10.1007/978-3-662-07964-5

[10] James Brotherston and Nikos Gorogiannis. 2014. Cyclic Abduction of Inductively Defined Safety and Termination Preconditions. In *Static Analysis - 21st International Symposium, SAS 2014, Munich, Germany, September 11-13, 2014. Proceedings (Lecture Notes in Computer Science)*, Markus Müller-Olm and Helmut Seidl (Eds.), Vol. 8723. Springer, 68–84. https://doi.org/10.1007/978-3-319-10936-7_5

[11] James Brotherston and Alex Simpson. 2011. Sequent calculi for induction and infinite descent. *Journal of Logic and Computation* 21, 6 (Dec. 2011), 1177–1216.

[12] Andrew Cave, Francisco Ferreira, Prakash Panangaden, and Brigitte Pientka. 2014. Fair Reactive Programming. In *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '14)*. ACM, New York, NY, USA, 361–372. https://doi.org/10.1145/2535838.2535881

[13] Christian Dax, Martin Hofmann, and Martin Lange. 2006. A Proof System for the Linear Time μ-Calculus. In *FSTTCS 2006: Foundations of Software Technology and Theoretical Computer Science, 26th International Conference, Kolkata, India, December 13-15, 2006, Proceedings*. 273–284. https://doi.org/10.1007/11944836_26

[14] Abhishek De, Luc Pellissier, and Alexis Saurin. 2021. Canonical proof-objects for coinductive programming: infinets with infinitely many cuts. In *PPDP*. ACM, 7:1–7:15.

[15] Abhishek De and Alexis Saurin. 2019. Infinets: the parallel syntax for non-wellfounded proof-theory. In *Automated Reasoning with Analytic Tableaux and Related Methods – TABLEAUX 2019 (Lecture Notes in Computer Science)*, Serenella Cerrito and Andrei Popescu (Eds.), Vol. 11714. Springer, 297–316. https://doi.org/10.1007/3-540-44904-3_18

[16] Amina Doumane. 2017. *On the infinitary proof theory of logics with fixed points. (Théorie de la démonstration infinitaire pour les logiques à points fixes)*. Ph.D. Dissertation. Paris Diderot University, France. https://tel.archives-ouvertes.fr/tel-01676953

[17] Amina Doumane, David Baelde, Lucca Hirschi, and Alexis Saurin. 2016. Towards Completeness via Proof Search in the Linear Time μ-calculus: The case of Büchi inclusions. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, Martin Grohe, Eric Koskinen, and Natarajan Shankar (Eds.). ACM, 377–386. https://doi.org/10.1145/2933575.2933598

[18] Jérôme Fortier and Luigi Santocanale. 2013. Cuts for circular proofs: semantics and cut-elimination. In *Computer Science Logic 2013 (CSL 2013), CSL 2013, September 2-5, 2013, Torino, Italy (LIPIcs)*, Simona Ronchi Della Rocca (Ed.), Vol. 23. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 248–262.

[19] Eduardo Giménez. 1998. Structural Recursive Definitions in Type Theory. In *Proceedings 25th Int. Coll. on Automata, Languages and Programming, ICALP'98, Aalborg, Denmark, 13–17 July 1998*, K. G. Larsen, S. Skyum, and G. Winskel (Eds.). LNCS, Vol. 1443. Springer-Verlag, Berlin, 397–408.

[20] Jean-Yves Girard. 1987. Linear Logic. *Theor. Comput. Sci.* 50 (1987), 1–102. https://doi.org/10.1016/0304-3975(87)90045-4

[21] Jean-Yves Girard. 1989. Towards a Geometry of Interaction. In *Categories in Computer Scienc e and Logic (Contemporary Mathematics)*. AMS, 69–108.

[22] Jean-Yves Girard. 2001. Locus Solum. *MSCS* 11 (2001), 301–506.

[23] Dominic J. D. Hughes and Rob J. van Glabbeek. 2005. Proof nets for unit-free multiplicative-additive linear logic. *ACM Trans. Comput. Log.* 6, 4 (2005), 784–842. https://doi.org/10.1145/1094622.1094629

[24] Pierre Hyvernat. 2014. The Size-Change Termination Principle for Constructor Based Languages. *Logical Methods in Computer Science* 10, 1 (2014). https://doi.org/10.2168/LMCS-10(1:11)2014

[25] David Janin and Igor Walukiewicz. 1995. Automata for the Modal mu-Calculus and related Results. In *Mathematical Foundations of Computer Science 1995, 20th International Symposium, MFCS'95, Prague, Czech Republic, August 28 - September 1, 1995, Proceedings (Lecture Notes in Computer Science)*, Jirí Wiedermann and Petr Hájek (Eds.), Vol. 969. Springer, 552–562. https://doi.org/10.1007/3-540-60246-1_160

[26] Roope Kaivola. 1995. Axiomatising Linear Time Mu-calculus. In *CONCUR '95: Concurrency Theory, 6th International Conference, Philadelphia, PA, USA, August 21-24, 1995, Proceedings*. 423–437.

[27] Dexter Kozen. 1983. Results on the Propositional mu-Calculus. *Theoretical Computer Science* 27 (1983), 333–354.

[28] Ralph Matthes. 1999. Monotone Fixed-Point Types and Strong Normalization. Vol. 1584. Berlin, 298–312.

[29] N. P. Mendler. 1991. Inductive Types and Type Constraints in the Second Order Lambda Calculus. *Annals of Pure and Applied Logic* 51, 1 (1991), 159–172.

[30] Marvin L. Minsky. 1961. Recursive Unsolvability of Post's Problem of "Tag" and other Topics in Theory of Turing Machines. *Annals of Mathematics* 74, 3 (1961), 437–455. http://www.jstor.org/stable/1970290

[31] Grigori E Mints. 1978. Finite investigations of transfinite derivations. *Journal of Soviet Mathematics* 10, 4 (1978), 548–596.

[32] Vaughan R Pratt. 1981. A decidable mu-calculus: Preliminary report. In *Foundations of Computer Science, 1981. SFCS'81. 22nd Annual Symposium on*. IEEE, 421–427.

[33] Davide Sangiorgi. 2009. On the origins of bisimulation and coinduction. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 31, 4 (2009), 15.

[34] Luigi Santocanale. 2002. μ-Bicomplete Categories and Parity Games. *ITA* 36, 2 (2002), 195–227. https://doi.org/10.1051/ita:2002010

[35] Luigi Santocanale. 2002. A Calculus of Circular Proofs and Its Categorical Semantics. In *Foundations of Software Science and Computation Structures (Lecture Notes in Computer Science)*, Mogens Nielsen and Uffe Engberg (Eds.), Vol. 2303. Springer, 357–371.

[36] Luigi Santocanale. 2002. Free μ-lattices. *Journal of Pure and Applied Algebra* 168, 2–3 (March 2002), 227–264. https://doi.org/10.1016/S0022-4049(01)00098-6

[37] Robert S. Streett and E. Allen Emerson. 1989. An Automata Theoretic Decision Procedure for the Propositional Mu-Calculus. *Information and Computation* 81, 3 (1989), 249–264. https://doi.org/10.1016/0890-5401(89)90031-X

[38] Kazushige Terui. 2011. Computational ludics. *Theoretical Computer Science* 412, 20 (2011), 2048–2071. https://doi.org/10.1016/j.tcs.2010.12.026

[39] Igor Walukiewicz. 1993. On Completeness of the mu-calculus. In *Proceedings of the Eighth Annual Symposium on Logic in Computer Science (LICS '93), Montreal, Canada, June 19-23, 1993*. IEEE Computer Society, 136–146. https://doi.org/10.1109/LICS.1993.287593

[40] Igor Walukiewicz. 1995. Completeness of Kozen's Axiomatisation of the Propositional mu-Calculus. 14–24.

[41] Igor Walukiewicz. 2000. Completeness of Kozen's Axiomatisation of the Propositional mu-Calculus. *Information and Computation* 157, 1-2 (2000), 142–182.

A Appendices

A.1 On sequents as sets of occurrences

There are several presentations of sequents in the literature: a sequent can be defined as a set of formulas, a multiset of formulas, a list of formulas or a set of named formulas. The first two presentations (sets and multisets of formulas) are not suitable in a Curry-Howard perspective as they identify proofs having completely different computational behaviours: we need not only a "ressource-aware" notion of sequents but also an "occurrence-aware", essentially using lists of formulas or occurrences of formulas.

The last two presentations are the most used in the proofs-as-programs framework. Considering sequents as lists of formulas requires a constant use of the exchange rule, which is very heavy. In this paper, we made the choice to work with sequents as sets of named formulas, also called *formula occurrences*.

A formula occurrence is a formula together with an address. In a derivation, all the conclusion (and cut) formula occurrences will have pairwise distinct addresses.

We consider a set of atomic addresses which is used to provide an infinite set of incomparable addresses (this could be achieved via words at the cost of an encoding). This intuition is that atomic addresses and their duals will be assigned to the conclusions and cut formulas, ensuring those occurrences are disjoint, and all the addresses appearing in the proofs will be sub-addresses of these addresses: disjointness is preserved by each inference rule.

When a rule is applied to a formula occurrence, the addresses of its sub-occurrences will be extended by $\{\mathtt{l}, \mathtt{r}, \mathtt{i}\}$ (standing for left, right and inside respectively) in order to record their provenance. This is of great importance for our developments: our validity criterion traces the evolution of formulas, and this evolution is completely explicit in their addresses.

Example A.1. We show in the following an example of an application of the $\parr$ rule in the framework of sequents as sets, as multisets and as sets of formula occurrences respectively:

$$\frac{\vdash \varphi}{\vdash \varphi \parr \varphi}\ (\parr) \qquad \frac{\vdash \varphi, \varphi}{\vdash \varphi \parr \varphi}\ (\parr) \qquad \frac{\vdash \varphi_{\alpha 1}, \varphi_{\alpha r}}{\vdash (\varphi \parr \varphi)_\alpha}\ (\parr)$$

In the first case, the two subformulas of $\varphi \parr \varphi$ collapse into one formula, in the second framework we keep track of the multiplicity but we cannot distinguish between the formula coming from the right and the one coming from the left. In the framework of formula occurrences, we can do this thanks to the tags $\mathtt{l}$ and $\mathtt{r}$ in their addresses.

A.2 The multicut rule

A new phenomenon occurs in the presence of axioms. Consider for instance the following pre-proof where $F = \varphi_\alpha$ and $G = \varphi_\beta$:

$$\dfrac{\dfrac{}{\vdash F, G^\perp}\ (\mathrm{Ax}) \qquad \dfrac{\pi}{\vdash G, \Gamma} \quad \dots}{\vdash F, \Gamma'}\ (\mathrm{mcut})$$

In the finitary cut-elimination procedure, we would reduce this multicut to the derivation labelled (mcut_1) below. Doing so, we have to perform a substitution on addresses (denoted by $[\alpha/\beta]$) to relocate the subderivation π on the required occurrence. Another option, described by the derivation (mcut_2) below, is to avoid the renaming by keeping a link explicitely in the multicut rule.

$$\dfrac{\dfrac{\pi[\beta/\alpha]}{\vdash F, \Gamma} \quad \dots}{\vdash F, \Gamma'}\ (\mathrm{mcut}_1) \qquad \text{or} \qquad \dfrac{\dfrac{\pi}{\vdash G, \Gamma} \quad \dots}{\vdash F, \Gamma'}\ (\mathrm{mcut}_2)$$

We choose the last option to avoid the global renaming, which would complicate our technical development.

A multicut rule will now be written as:

$$\frac{\vdash \Gamma_1 \quad \dots \quad \vdash \Gamma_n}{\vdash \Gamma}\ \mathrm{mcut}(\iota, \between)$$

and comes with a function ι which shows how the occurrences of the conclusion are distributed over the premisses (modulo renaming), and a relation $\between$ specifiying which occurrences are cut-connected. A precise definition of the multicut rule is given below.

Definition A.2. Given sequents $s, s_1, \dots, s_n$ where $n > 0$ and such that s_i, s_j are disjoint for all $i \neq j$, a ***multicut*** of conclusion s and premisses $(s_i)_{i\in[1;n]}$ is given by an injection $\iota : s \mapsto \cup_{i\in[1;n]} s_i$ and a symmetric relation $\between \subseteq (\cup_{i\in[1;n]} s_i)^2$ such that:

- For all $F \in s$, $\iota(F) \equiv F$.
- For all $F, G \in \cup_{i\in[1;n]} s_i$, $F \between G$ implies $F \equiv G^\perp$.
- $\mathrm{dom}(\between) = (\cup_{i\in[1;n]} s_i) \setminus \mathrm{im}(\iota)$.
- Given two sequents s_i and s_j, we say that they are $\between$-connected on the formula occurrences F, G when $F \in s_i$ and $G \in s_j$ such that $F \between G$. We say that they are $\between$-connected, and we write $s_i \between s_j$, when they are $\between$-connected on some F, G. The relation $\between$ on sequents must satisfy two conditions:
  - two sequents must be $\between$-connected on at most one pair of occurrences F, G;
  - the graph of the relation $\between$ must be connected and acyclic.

We write this multicut rule as:

$$\frac{s_1 \quad \dots \quad s_n}{s}\ \mathrm{mcut}(\iota, \between)$$

A.3 Cut elimination rules

We detail the rules of cut elimination introduced in section 3.

Definition A.3. ***External reductions*** are defined in fig. 11. In the first external rule, the sets C_Δ and C_Γ are the subsets of

C which are respectively connected to Δ and Γ respectively. More precisely,

$$C_\Delta = \{s \mid \exists s', s \mathrel{⫫}^* s' \text{ and } s' \text{ is } ⫫\text{-connected to } \vdash \Delta, \Gamma, F \otimes G \text{ on an occurrence of } \Delta\},$$

where $⫫^*$ is the transitive closure of the relation $⫫$ on sequents. C_Γ is defined similarly.

Remark 1. *Note that the* $(\otimes)/(\text{mcut})$ *external reduction yields multiple multicuts, though always on disjoint sub-trees. Thus,* $\mu\mathsf{MLL}^\infty_\mathsf{m}$ *is stable by external reductions.*

In external reductions, we pushed a multicut away from the root, above a logical rule. If we start with a $\mu\mathsf{MLL}^\infty_\mathsf{m}$ pre-proof and apply a reduction sequence where external rules are applied infinitely often to each multicut, we will produce at the limit a cut-free proof. This is the reason why we say that external reductions are ***productive***. This is not the case for the internal reduction rules given next.

Definition A.4. ***Internal reductions*** are the ***principal*** reductions given in fig. 12 together with the following two reductions:

- the merge (mcut)/(Cut) reduction

$$\dfrac{\dfrac{C \quad \dfrac{\vdash \Delta, F \qquad \vdash \Gamma, F^\perp}{\vdash \Delta, \Gamma}\,(\text{Cut})}{\vdash \Sigma}\,\text{mcut}(\iota, ⫫)}{} \xrightarrow[r]{} \dfrac{C \quad \vdash \Delta, F \qquad \vdash \Gamma, F^\perp}{\vdash \Sigma}\,\text{mcut}(\iota, ⫫')$$

  where $⫫'$ extends $⫫$ with $F \mathrel{⫫}' F^\perp$ and $r = (\text{merge}, \{F, F^\perp\})$.
- the axiom reduction (mcut)/(Ax)

$$\dfrac{C \quad \vdash F, F'^\perp \qquad \vdash F'', \Gamma}{\vdash \Sigma}\,\text{mcut}(\iota, ⫫) \xrightarrow[r]{} \dfrac{C \quad \vdash F'', \Gamma}{\vdash \Sigma}\,\text{mcut}(\iota', ⫫')$$

  where $r = (\text{CutAx}, \{F, F'^\perp\})$, $F'^\perp \mathrel{⫫} F''$ and $\iota', ⫫'$ are defined as follows:
  - for all $G \in \Sigma$, if $\iota(G) = F$ then $\iota'(G) = F''$, otherwise $\iota'(G) = \iota(G)$;
  - $⫫' = ⫫ \cup \{\{F'', G\} | \{F, G\} \in ⫫\}$.

In internal reductions, the multicut remains at the root of the redex. Thus, if a sequence of multicut reductions eventually involved only internal reductions, it would not be productive.

The use of labels in reductions allows us to define in full details our notion of reduction sequence and fairness.

Definition A.5. A ***reduction sequence*** is a finite or infinite sequence $\sigma = (\pi_i, r_i)_{i \in 1+\lambda}$ with $\lambda \in \omega + 1$, where the π_i are $\mu\mathsf{MLL}^\infty_\mathsf{m}$ pre-proofs, the r_i are labels identifying multicut reduction rules and, for all $i \in \lambda$, $\pi_i \xrightarrow[r_i]{} \pi_{i+1}$. The sequence is ***fair*** if for all $i \in \lambda$ and r such that $\pi_i \xrightarrow[r]{} \pi'$ there is some $j \in \lambda$ such that $j \geq i$ and $\pi_j \xrightarrow[r]{} \pi_{j+1}$.

A.4 Cut elimination for $\mu\mathsf{MLL}^\infty$

A.4.1 Trace of a reduction sequence

If $\mathcal{R}$ is a reduction sequence starting from π, we start by defining the ***trace*** of $\mathcal{R}$ to be the subtree of π whose sequents occur in the reduction sequence as premisses of some multicut. Note that each node of the trace corresponds to a well-formed inference: indeed, if sequents S and S' are premisses of a same inference (which must thus be a tensor or cut) and S enters a multicut at some point in the reduction sequence, then S' must also enter a multicut – though not necessarily the same one. However, the trace may have unjustified sequents: this happens when a sequents S enters the multicut during the reduction sequence but never leaves it; it will then be part of the trace but the subtree of π rooted in S will not.

The unjustified sequents of the trace are called its ***border sequents***. Note that a border sequent cannot be the conclusion of an axiom rule nor a cut rule in the initial derivation π. If this were the case, by fairness, it would have been absorbed by an (Ax)/(mcut) or a (Cut)/(mcut) reduction respectively. This allows to define the ***distinguished occurrence*** of a border sequent as the principal occurrence of the logical rule applied to the border sequent in π.

There is another reason why the trace of a reduction sequence might not be a proof: its infinite branches may not be valid. The infinite branches of the trace are also infinite branches of the proof π, thus they are supported by valid threads of π, but these threads might not be included in the trace. We show that this actually never happens.

A.4.2 The bouncing threads of the trace belong to the trace

This section is dedicated to proving the following theorem.

Proposition A.6. *Let* T *be the trace of a reduction sequence starting from a proof* π*, and let* β *be an infinite branch of* T*. If* t *is a thread of* π *validating* β*, then* t *is also a thread of* T*.*

We now introduce a a useful technical tool called the ***residual of a pre-thread***.

Definition A.7 (Residual of a pre-thread)**.** Let $\mathcal{R}$ be a finite reduction path starting from π to π' and let t be a pre-thread of π. Let S' be the set of sequents of π'. We call the ***residual of*** t after the reduction $\mathcal{R}$ the pre-thread $t \cap \{(F, s, d) | s \in S', d \in \{\uparrow, \downarrow\}\}$.

By definition, the length of the residual of t is smaller than the length of t.

Proposition A.8. *Let* $\mathcal{R}$ *be a finite reduction path starting from* π *to* π'*, let* T *be its trace. Let* t *be a* b*-thread of* π *and* t' *its residual after* $\mathcal{R}$*. Then* t' *is a* b*-thread. Furthermore, if* t *is a B-path of* π*, then* t' *is a B-path of* T*, and* t' *has the same endpoints as* t*.*

$$\dfrac{C \quad \dfrac{\vdash \Delta, F' \qquad \vdash \Gamma, G'}{\vdash \Delta, \Gamma, F' \otimes G'}\,(\otimes)}{\vdash \Sigma_\Delta, \Sigma_\Gamma, F \otimes G}\,\mathrm{mcut}(\iota,\, \perp\!\!\!\perp) \quad \xrightarrow[r]{} \quad \dfrac{\dfrac{C_\Delta \qquad \vdash \Delta, F'}{\vdash \Sigma_\Delta, F}\,\mathrm{mcut}(\iota',\, \perp\!\!\!\perp) \qquad \dfrac{C_\Gamma \qquad \vdash \Gamma, G'}{\vdash \Sigma_\Gamma, G}\,\mathrm{mcut}(\iota'',\, \perp\!\!\!\perp)}{\vdash \Sigma_\Delta, \Sigma_\Gamma, F \otimes G}\,(\otimes)$$

$$\dfrac{C \quad \dfrac{\vdash \Delta, F', G'}{\vdash \Delta, F' \parr G'}\,(\parr)}{\vdash \Sigma, F \parr G}\,\mathrm{mcut}(\iota,\, \perp\!\!\!\perp) \quad \xrightarrow[r]{} \quad \dfrac{C \qquad \vdash \Delta, F', G'}{\dfrac{\vdash \Sigma, F, G}{\vdash \Sigma, F \parr G}\,(\parr)}\,\mathrm{mcut}(\iota',\, \perp\!\!\!\perp)$$

$$\dfrac{C \quad \dfrac{\vdash \Delta, F'[\sigma X.F'/X]}{\vdash \Delta, \sigma X.F'}\,(\sigma)}{\vdash \Sigma, \sigma X.F}\,\mathrm{mcut}(\iota,\, \perp\!\!\!\perp) \quad \xrightarrow[r]{} \quad \dfrac{C \qquad \vdash \Delta, F'[\sigma X.F'/X]}{\dfrac{\vdash \Sigma, F[\sigma X.F/X]}{\vdash \Sigma, \sigma X.F}\,(\sigma)}\,\mathrm{mcut}(\iota',\, \perp\!\!\!\perp)$$

$$\dfrac{C \quad \dfrac{\vdash \Delta}{\vdash \Delta, \bot_\beta}\,(\bot)}{\vdash \Sigma, \bot_\alpha}\,\mathrm{mcut}(\iota,\, \perp\!\!\!\perp) \quad \xrightarrow[r]{} \quad \dfrac{C \qquad \vdash \Delta}{\dfrac{\vdash \Sigma}{\vdash \Sigma, \bot_\alpha}\,(\bot)}\,\mathrm{mcut}(\iota',\, \perp\!\!\!\perp) \qquad \dfrac{\dfrac{}{\vdash \mathbf{1}_\beta}\,(\mathbf{1})}{\vdash \mathbf{1}_\alpha}\,\mathrm{mcut}(\iota,\, \perp\!\!\!\perp) \xrightarrow[r]{} \dfrac{}{\vdash \mathbf{1}_\alpha}\,(\mathbf{1})$$

In the first reduction ((⊗)/(mcut)) we require that $\iota(F \otimes G) = F' \otimes G'$ and take ι' and ι'' that coincide with ι on Σ_Δ and Σ_Γ respectively, and such that $\iota'(F) = F'$ and $\iota''(G) = G'$. In the other reductions ι and ι' are similarly constrained.

Figure 11. External reduction rules, where $r = (\mathrm{ext}, F)$ and F is the principal occurrence.

$$\dfrac{C \quad \dfrac{\vdash \Delta, F \qquad \vdash \Gamma, G}{\vdash \Delta, \Gamma, F \otimes G}\,(\otimes) \qquad \dfrac{\vdash \Theta, F'^{\perp}, G'^{\perp}}{\vdash \Theta, F'^{\perp} \parr G'^{\perp}}\,(\parr)}{\vdash \Sigma}\,\mathrm{mcut}(\iota,\, \perp\!\!\!\perp) \quad \xrightarrow[r]{} \quad \dfrac{C \qquad \vdash \Delta, F \qquad \vdash \Gamma, G \qquad \vdash \Theta, F'^{\perp}, G'^{\perp}}{\vdash \Sigma}\,\mathrm{mcut}(\iota,\, \perp\!\!\!\perp')$$

where $F \otimes G \perp\!\!\!\perp G'^{\perp} \parr G'^{\perp}$ and $\perp\!\!\!\perp'$ coincides with $\perp\!\!\!\perp$ except for $F \perp\!\!\!\perp' F'^{\perp}$ and $G \perp\!\!\!\perp' G'^{\perp}$

$$\dfrac{C \quad \dfrac{\vdash \Delta, F'[\mu X.F'/X]}{\vdash \Delta, \mu X.F'}\,(\mu) \qquad \dfrac{\vdash \Gamma, F^{\perp}[\nu X.F^{\perp}/X]}{\vdash \Gamma, \nu X.F^{\perp}}\,(\nu)}{\vdash \Sigma}\,\mathrm{mcut}(\iota,\, \perp\!\!\!\perp) \quad \xrightarrow[r]{} \quad \dfrac{C \qquad \vdash \Delta, F'[\mu X.F'/X] \qquad \vdash \Gamma, F^{\perp}[\nu X.F^{\perp}/X]}{\vdash \Sigma}\,\mathrm{mcut}(\iota,\, \perp\!\!\!\perp')$$

where $\mu X.F' \perp\!\!\!\perp \nu X.F^{\perp}$ and $\perp\!\!\!\perp'$ coincides with $\perp\!\!\!\perp$ except for $F'[\mu X.F'/X] \perp\!\!\!\perp' F^{\perp}[\nu X.F^{\perp}/X]$

$$\dfrac{C \quad \dfrac{\vdash \Gamma}{\vdash \Gamma, \bot_\alpha}\,(\bot) \qquad \dfrac{}{\vdash \mathbf{1}_\beta}\,(\mathbf{1})}{\Sigma}\,\mathrm{mcut}(\iota,\, \perp\!\!\!\perp) \quad \xrightarrow[r]{} \quad \dfrac{C \qquad \vdash \Gamma}{\Sigma}\,\mathrm{mcut}(\iota,\, \perp\!\!\!\perp) \quad \text{where } \bot_\alpha \perp\!\!\!\perp \mathbf{1}_\beta$$

Figure 12. Principal reductions, where $r = (\mathrm{princ}, \{F, F'^{\perp}\})$ with $\{F, F'^{\perp}\}$ the principal occurrences that have been reduced.

Proof. It suffices to consider a single reduction step. Most of the claims follow from a simple inspection of the reduction rules. For the last one (i.e. t and t' have the same endpoints) we have to additionally rule out the possibility that, if s and s' are the endpoints of t, s gets reduced at some point of the reduction while s' does not: this could only happen if s (or s') was part of an (Ax)/(mcut) reduction, but that would mean that our B-path can be extended into an h-path (if the axiom

is at the beginning of the path) or the reverse of an h-path (if the axiom is at the end). □

Definition A.9. A pre-thread t is a B-path of π if:

- it is a maximal b-path of π, i.e. there is no b-path of π which contains t as an infix;
- it cannot be extended as an h-path or as the reverse of an h-path.

Intuitively, the second condition means that the path cannot be extended by an axiom on either side, possibly after silent steps corresponding to W weights.

Lemma A.10. *Let $\mathcal{R} = \{\pi_i, r_i\}_{i\in\omega}$ be a reduction sequence and let $T = Tr(\mathcal{R})$ be its trace. If t is a B-path of T, then there is an index i such that the endpoints of t are mcut-connected in the multicut of π_i.*

Proof. As reductions are performed, the thread t is simplified into *residuals*. As long as these residuals remain non-empty, they are still b-paths in their respective derivations, and they keep the same endpoints because the only way to reduce one endpoint without the other is through an (Ax)/(mcut) reduction. Moreover, the length of residuals only decreases. In fact, since t is in the trace, it strictly decreases infinitely often. Thus, at some point, the two endpoints of t are directly mcut-connected. □

Proof of proposition 5.3. Let t be a thread validating the branch β in π. The visible part of t belongs to β, hence its belongs to the trace T. One needs to prove that the hidden part belongs also to the trace.

Suppose by contradiction that some $H = (F_i, s_i, d_i)_{1\le i\le n} \in \mathsf{hp}(t)$ leaves the trace. Thus there is an index $j \le n$ such that s_j is a border sequent of the trace. Take the maximal such j. If $d_j = \uparrow$ then, since the hidden part H ends in a sequent of β, there must be a position k with $j < k < n$ such that the path exits the subtree rooted in s_j to re-enter β: we would then have $s_k = s_j$, contradicting the maximality of j. Hence, $d_j = \downarrow$.

Since H is a b-path, and $d_j = \downarrow$, then there is $k > j$ such that $H[j,k]$ is a b-path: keeping with the intuition that b-paths are well-bracketed words, the opening bracket at position j must have a corresponding closing bracket at position k such that the word in between is well-bracketed, thus a b-path. By maximality of j, $H[j,k]$ is in the trace.

Let k be maximal with this property. Note that $H[j,k]$ is a maximal b-path in the trace, since s_j is a border sequent and k is chosen to be maximal. Let us show that $H[j,k]$ cannot be extended to an ϵ-path of the trace. Suppose by contradiction that this is the case. Since s_j is a border sequent, $H[j,k]$ can be extended only on the right. Thus there is $l > k$ such that $H[j,l]$ is an ϵ-path. Since $H[j,l]$ starts with a downward direction, and since ϵ-paths have the same direction in their endpoints, we have that $d_l = \downarrow$. Thus by the same reasoning as before, there is $m > l$ such that $H[l,m]$ is a b-path. Thus $H[j,m]$ is a b-path, which contradicts the maximality of k.

We can now apply lemma A.10 to $H[j,k]$: at some point of the reduction, the sequents s_j and s_k are mcut-connected through the occurrences F_j and F_k. Note that s_{k+1} belongs to T (by maximality of j) and that F_{k+1} is a strict sub-occurrence of F_k (otherwise, this would contradict the maximality of k). Since F_{k+1} is in the trace, this means that F_j has been reduced which is not possible since s_j is in the border of the trace. □

In order to view a trace as a $\mu\mathsf{MLL}^\infty_{\mathsf{m}}$ proof, we shall devise a way to justify its border sequents. Intuitively, we will identify each distinguished occurrence with the true constant $\top$. To achieve this formally, we introduce a *truncated* proof system in the next section. Before that, let us mention a key technical result, which builds on the intuition that b-paths are simplified during cut elimination.

A.4.3 Truncated proof system

The truncated proof system builds on a truncation that forces a semantics on particular occurrences.

Definition A.11. A ***truncation*** τ is a partial function from Σ^* to $\{\top, \mathbf{0}\}$ such that:

- For any $\alpha \in \Sigma^*$, if $\alpha \in \mathtt{Dom}(\tau)$, then $\alpha^\perp \in \mathtt{Dom}(\tau)$ and $\tau(\alpha) = \tau(\alpha^\perp)^\perp$.
- If $\alpha \in \mathtt{Dom}(\tau)$ then for any $\beta \in \Sigma^+$, $\alpha.\beta \notin \mathtt{Dom}(\tau)$.

Definition A.12. Given a truncation τ, the infinitary ***proof system*** $\mu\mathsf{MLL}^\infty_\tau$ is obtained by taking all the rules of $\mu\mathsf{MLL}^\infty$ together with the following rule for $\top$:

$$\frac{}{\vdash \Gamma, \top_\alpha}\ (\top)$$

with the following proviso. The rules of $\mu\mathsf{MLL}^\infty$ only apply when the address of their principal occurrence is not in the domain of τ, otherwise the following rule has to be applied:

$$\frac{\vdash \tau(\alpha)_{\alpha i}, \Delta}{\vdash \varphi_\alpha, \Delta}\ (\tau) \qquad \text{if } \alpha \in \mathtt{Dom}(\tau)$$

The notions of thread and validity are the same as in $\mu\mathsf{MLL}^\infty$.

As in [7] we define a classical truth semantics for our truncated proof system. Truncated occurrences (i.e. whose address is in $\mathtt{Dom}(\tau)$) are assigned their value under τ. The semantics of a unit $\top$ or 0 is itself. Then this semantics is propagated to more complex formulas inductively, interpreting $\parr$ as disjunction, $\otimes$ as conjunction, and μ, ν as least and greatest fixed points respectively. The semantics of an occurrence F under a truncation τ is noted $[\![F]\!]$. We establish, in the same way as in [7] that $\mu\mathsf{MLL}^\infty_\tau$ is sound wrt. this semantics:

Proposition A.13. *If $\vdash \Gamma$ is provable in $\mu\mathsf{MLL}^\infty_\tau$, then $[\![F]\!] = \top$ for some $F \in \Gamma$.*

Since $\mu\mathsf{MLL}^\infty$ is a sub-system of $\mu\mathsf{MLL}^\infty_\tau$, we obtain as a corollary that $\mu\mathsf{MLL}^\infty$ is sound wrt. the boolean semantics.

A.4.4 From traces to truncated proofs

Definition A.14. Let π be a pre-proof. We define the relations $\approx_\pi$ and $\perp\!\!\!\perp_\pi$ as follows:

- $F \approx_\pi G$ if there is an h-path in π from F to G, or from G to F.
- $F \perp\!\!\!\perp_\pi G$ if there is a b-path in π between F and G.

The relations $\perp\!\!\!\perp_\pi$ and $\approx_\pi$ are symmetric — note that the reverse of a b-path from F to G is a b-path from G to F. The relation $\approx_\pi$ is reflexive.

Proposition A.15. *Let $\mathcal{R}$ be a reduction. The trace of $\mathcal{R}$ cannot contain an occurrence F and two distinguished occurrences G and H such that $F \perp\!\!\!\perp_T G$ and $F \approx_T H$.*

Proof. We proceed by contradiction. Let t_2 be the b-path from G to F, starting with a $\downarrow$ direction and ending with $\uparrow$. Let t_1 be the path from F to H. It must be an h-path, starting and ending with $\uparrow$. Indeed, the reverse of an h-path would reach H with a $\downarrow$ which is absurd since H is a distinguished occurrence of a border sequent.

Let t be the b-path obtained by gluing the path from G to F with the path from F to H. This path is a B-path of T, since its endpoints are distinguished formulas, so they are in the border of the trace. By applying lemma A.10, there is a point in the reduction where the occurrences G and H are directly mcut-connected.

Since G and H are distinguished, they are principal occurrences of the rules applied to their border sequents in π. Thus, considering the point of the reduction where they are directly mcut-connected, there is an internal redex on G and H. By fairness it is reduced, which contradicts the fact that they are distinguished occurrences of border sequents. □

By proposition A.15 we can define the truncation and truncated proof associated to a trace.

Definition A.16. Let $\mathcal{R}$ be a reduction sequence and T be its trace. The truncation τ associated with $\mathcal{R}$ is defined by setting:

- $\tau(F) = \mathbf{0}$ if there is a distinguished occurrence G such that $F \perp\!\!\!\perp_T G$,
- $\tau(F) = \top$ if there is a distinguished occurrence G such that $G \approx_T F$.

Definition A.17. Let T be the trace of a infinite internal reduction sequence starting from π, and let τ be the truncation associated to this reduction. The ***truncated proof*** π_τ is obtained from T by replacing every border sequent $\vdash \varphi_\alpha, \Gamma$, whose distinguished occurrence is φ_α by the following derivation:

$$\dfrac{\dfrac{}{\vdash \top_{\alpha.i}, \Gamma}\ (\top)}{\vdash \varphi_\alpha, \Gamma}\ (\tau)$$

It is now easy to establish productivity of cut elimination.

Proposition A.18. *Any fair reduction sequence produces a $\mu\mathsf{MLL}^\infty$ pre-proof.*

Proof. By contradiction, consider a fair infinite sequence of internal multicut reductions starting from π. Let π_τ be the truncated proof of its trace. Since no external reduction occurs, it means that an occurrence F in the conclusion of π_τ can only be principal in an (Ax)/(mcut) reduction of the considered sequence. If ι is the injection associated to the multicut after that reduction, the same observation holds for $\iota(F)$, and so on. In short, any occurrence $F' \approx_{\pi_\tau} F$ will never be principal in a logical rule. Hence we can replace all these occurrences by occurrences of $\bot$. We thus obtain a proof of the sequent $\vdash \bot, \dots, \bot$ which contradicts the soundness of $\mu\mathsf{MLL}^\infty_\tau$. □

A.4.5 Proof of cut elimination

We have shown in proposition A.18 that multicut reduction is productive. To establish cut-elimination (theorem 5.1), it only remains to prove that the resulting (cut-free) pre-proof is actually a valid proof.

Proof of theorem 5.1. Let π be a $\mu\mathsf{MLL}^\infty_{\mathsf{m}}$ proof of conclusion $\vdash A$, and π' the cut-free pre-proof obtained by proposition A.18, i.e., the limit of the multicut reduction process. Any branch of π' corresponds to a multicut reduction path. For the sake of contradiction, assume that π' is invalid. It must thus have an invalid infinite branch $\beta = (s_i)_{i\in\omega}$, corresponding to an infinite reduction path $\mathcal{R}$. Let τ and θ be the associated truncation and truncated proof in $\mu\mathsf{MLL}^\infty_\tau$.

We set $\mathsf{Froz}(\beta)$ to be the set of occurrences of β which are never principal. For convenience we will use the weakening rule:

$$\dfrac{\vdash \Gamma}{\vdash \Gamma, \Delta}\ (\mathsf{W})$$

Weakening is indeed admissible as long as the derivation on which it is applied contains an infinite branch, since one can let the weakened occurrences "travel" into this infinite branch. Without loss of generality, we now assume that all occurrences of $\mathsf{Froz}(\beta)$ have been weakened away in β.

We define the truncation τ' to be the truncation obtained by extending τ as follows. For every occurrence $F_1 \parr F_2$ which is principal in β, we set, for $i \in \{1,2\}$, $\tau'(F_i) = \mathbf{0}$ if $F_i \in \mathsf{Froz}(\beta)$.

Let $\beta_{\geq i}$ be the suffix of β starting from the i^{th} element. If $\vdash \Gamma$ is the conclusion of $\beta_{\geq i}$, then for every $\Delta \subseteq \Gamma$, we define coinductively the proof $\beta_{\geq i}{}^{\perp}(\Delta)$ of conclusion $\vdash \Delta^\perp$ as follows. We proceed by case analysis on the rule applied to the conclusion of β. This rule can be either a logical rule or a weakening. If the rule is a weakening, then it is simulated by a weakening rule. If it is a logical rule, then let F be its principal occurrence. We set $\Delta' = \Delta \setminus \{F\}$. We have either:

- $F = \sigma X.G$ where $\sigma \in \{\mu, \nu\}$. We set $\overline{\sigma}$ to be the dual of σ, and:

$$\beta^{\perp}_{\geq i}(\Delta) = \dfrac{\dfrac{\beta^{\perp}_{\geq i+1}(\Delta', G[F/X])}{\vdash \Delta'^{\perp}, G^{\perp}[F^{\perp}/X]}}{\vdash \Delta'^{\perp}, F^{\perp}}\ {\scriptstyle (\overline{\sigma})}$$

- $F = G \otimes H$. Suppose wlog. that $G \in s_{i+1}$ (i.e., H left the branch β). We set $\Delta'' = \Delta' \cap s_{i+1}$ and:

$$\beta^{\perp}_{\geq i}(\Delta) = \dfrac{\dfrac{\dfrac{\beta^{\perp}_{\geq i+1}(\Delta'', G)}{\vdash \Delta''^{\perp}, G^{\perp}}\ {\scriptstyle (\mathsf{W})}}{\vdash \Delta'^{\perp}, G^{\perp}, H^{\perp}}\ {\scriptstyle (\parr)}}{\vdash \Delta'^{\perp}, F^{\perp}}$$

- $F = G \parr H$ and $G \in \mathtt{Dom}(\tau')$ and $H \notin \mathtt{Dom}(\tau')$, or symmetrically. We set:

$$\beta^{\perp}_{\geq i}(\Delta) = \dfrac{\dfrac{}{\vdash G^{\perp}}\ {\scriptstyle (\tau'),\ (\top)} \qquad \dfrac{\beta^{\perp}_{\geq i+1}(\Delta', H)}{\vdash \Delta'^{\perp}, H^{\perp}}}{\vdash \Delta'^{\perp}, F^{\perp}}\ {\scriptstyle (\otimes)}$$

- $F = G \parr H$, $G \in \mathtt{Dom}(\tau')$ and $H \in \mathtt{Dom}(\tau')$. In this case we set:

$$\beta^{\perp}_{\geq i}(\Delta) = \dfrac{\dfrac{}{\vdash G^{\perp}}\ {\scriptstyle (\tau'),\ (\top)} \qquad \dfrac{}{\vdash \Delta'^{\perp}, H^{\perp}}\ {\scriptstyle (\tau'),\ (\top)}}{\vdash \Delta'^{\perp}, F^{\perp}}\ {\scriptstyle (\otimes)}$$

- $F = G \parr H$, $G \notin \mathtt{Dom}(\tau')$ and $H \notin \mathtt{Dom}(\tau')$. In this case we set:

$$\beta^{\perp}_{\geq i}(\Delta) = \dfrac{\dfrac{\beta^{\perp}_{\geq i+1}(G)}{\vdash G^{\perp}} \qquad \dfrac{\beta^{\perp}_{\geq i+1}(\Delta', H)}{\vdash \Delta'^{\perp}, H^{\perp}}}{\vdash \Delta'^{\perp}, F^{\perp}}\ {\scriptstyle (\otimes)}$$

Observe now that $\beta^{\perp} := \beta^{\perp}_{\geq 0}$ is a proof in the truncated proof system $\mu\mathsf{MLL}^{\infty}_{\tau'}$ of conclusion $A^{\perp}$. Indeed, its threads (which are necessarily straight threads since no cuts and axioms are involved) are the duals of the threads of the branch β, which are by hypothesis not valid.

Since $\mu\mathsf{MLL}^{\infty}_{\tau'}$ is sound, we have that $[\![A^{\perp}]\!] = \top$. But the truncated proof θ is a $\mu\mathsf{MLL}^{\infty}_{\tau}$ proof of conclusion A, and again by soundness we have that $[\![A]\!] = \top$: contradiction. □

A.5 Extending $\mu\mathsf{MLL}^{\infty}$ cut-elimination to the additives

In this appendix, we give details on the proof of cut-elimination theorem for full $\mu\mathsf{MALL}^{\infty}$. First, we provide precise definition for the sliced proof system and the associated partial cut-reduction relation and introduce persistent slices. We can then formulate precisely the additive validity criterion and prove the cut-elimination theorem. The proof schema of additive cut-elimination follows the same pattern as in the multiplicative case but we check that the multiplicative result can indeed be lifted. Most of the definitions can be very straightforwardly lifted to the additive but for the soundness result some work has to be done: we show that we do not need a full soundness result but that a soundness result wrt. a specific class of derivations, called τ-adapted proofs, which then allows us to prove productivity of cut-elimination and preservation of validity by fair-reduction sequences.

A.5.1 Sliced proof system and its cut-reduction

To solve the previous issue, we will make use of slices, originally introduced by Girard in his seminal paper and later used in the analysis of interaction and cut-elimination of linear logic in the setting of Ludics [22, 38] or in the design of additive proof-nets [23].

Definition A.19 (Additive slice)**.** A ***sliced pre-proof*** is a pre-proof built on a variant of $\mu\mathsf{MALL}^{\infty}$, $\mu\mathsf{SMALL}^{\infty}$, where the inference rule $(\&)$ has been replaced by the following two rules:

$$\dfrac{\vdash A, \Gamma}{\vdash A \& B, \Gamma}\ {\scriptstyle (\&_1)} \qquad \dfrac{\vdash B, \Gamma}{\vdash A \& B, \Gamma}\ {\scriptstyle (\&_2)}$$

Definition A.20 (Slicing of a pre-proof)**.** To a $\mu\mathsf{MALL}^{\infty}$ sequent (pre-)proof, one can associate a set of slices by keeping, for each $(\&)$ inference, only one of its premisses and replacing the $\&$ by the corresponding inference in $(\&_1), (\&_2)$. More precisely, a ***slice of*** π is any $\mu\mathsf{SMALL}^{\infty}$ derivation obtained from π by applying corecursively one of the following two reductions (the other inferences are treated homomorphically):

$$\dfrac{\dfrac{\pi_1}{\vdash A_1, \Gamma} \quad \dfrac{\pi_2}{\vdash A_2, \Gamma}}{\vdash A_1 \& A_2, \Gamma}\ {\scriptstyle (\&)} \longrightarrow \dfrac{\dfrac{\pi_1}{\vdash A_1, \Gamma}}{\vdash A_1 \& A_2, \Gamma}\ {\scriptstyle (\&_1)}, \dfrac{\dfrac{\pi_2}{\vdash A_2, \Gamma}}{\vdash A_1 \& A_2, \Gamma}\ {\scriptstyle (\&_2)}$$

that is:

$$Sl\left(\dfrac{\dfrac{\pi_1}{\vdash A_1, \Gamma} \quad \dfrac{\pi_2}{\vdash A_2, \Gamma}}{\vdash A_1 \& A_2, \Gamma}\ {\scriptstyle (\&)}\right) = \left\{ \dfrac{\dfrac{\pi'_i}{\vdash A_i, \Gamma}}{\vdash A_1 \& A_2, \Gamma}\ {\scriptstyle (\&_i)},\ \begin{array}{c} \pi'_i \in Sl(\pi_i), \\ i \in \{1,2\} \end{array} \right\}$$

A.5.2 Cut-reductions for sliced proofs

Cut-reduction rules for slices of $\mu\mathsf{SMALL}^{\infty}$ are identical to those for $\mu\mathsf{MALL}^{\infty}$ except for the sliced additives. In this case, one may have a problematic situation when a $(\&_1)$ shall interact with a $(\oplus_2)$: cut-elimination cannot be performed. Among the several ways to cope with this problem, we choose here to introduce a special inference, (Ω), a generalized axiom rule allowing to derive any sequent, which denotes the fact that a bad interaction occurred.

$$\dfrac{}{\vdash \Gamma}\ {\scriptstyle (\Omega)}$$

This does not impact the technical development since this serves essentially the purpose of defining those slices which avoid the mismatch.

Considering the (Ω) inference, cut-reductions for slices are specified as follows[9]:

[9] This reduction is straightforwardly extended to multicuts and some care shall be taken in treating (Ω), in particular any cut involving (Ω) is reduced to (Ω) itself as standard in ludics.

Definition A.21 (Cut reductions for slices)**.** The sliced additive principal case is reduced as follows, if $\{A_1^\perp \& A_2^\perp, A'_1 \oplus A'_2\} \in \perp\!\!\!\perp$, with $r = (\text{princ}, \{A_1^\perp \& A_2^\perp, A'_1 \oplus A'_2\})$.

$$\dfrac{C \quad \dfrac{\dfrac{\pi_i}{\vdash A_i^\perp, \Gamma}}{\vdash A_1^\perp \& A_2^\perp, \Gamma}\,(\&_i) \qquad \dfrac{\dfrac{\pi'_j}{\vdash A'_j, \Gamma}}{\vdash A'_1 \oplus A'_2, \Delta}\,(\oplus_j)}{\vdash \Sigma}\,\text{mcut}(\iota, \perp\!\!\!\perp)$$

$$\xrightarrow[r]{} \begin{cases} \dfrac{}{\vdash \Sigma}\,(\Omega) & \text{if } i \neq j \\[2ex] \dfrac{C \quad \dfrac{\pi_i}{\vdash A_i^\perp, \Gamma} \quad \dfrac{\pi'_i}{\vdash A'_i, \Delta}}{\vdash \Sigma}\,\text{mcut}(\iota, \perp\!\!\!\perp') & \text{if } i = j \\ \text{where } \perp\!\!\!\perp' = \perp\!\!\!\perp \cup \{\{A_i^\perp, A'_i\}\} \end{cases}$$

Notions of b-paths and ϵ-paths can be naturally extended to additive slices.

A.5.3 Persistent slices

To state the validity criterion for the additives, one needs to describe ***persistent*** slices that will never produce a (Ω):

Definition A.22 (Persistent slice)**.** Given a slice π, a $(\&_i)$ rule of principal formula $A_1 \& A_2$ occurring in π is said to be ***well-sliced*** if no b-path starting down from the $A_1 \& A_2$ occurrence of this sequent ends in a $A_1^\perp \oplus A_2^\perp$ which is the principal formula of a $(\oplus_j)$ inference with $i \neq j$. A slice is ***persistent*** if all its $(\&_i)$ occurrences are well-sliced.

Lemma A.23. *In a persistent slice, $(\&_i)$ rules are characterized by the following property: (i) either there exists a b-path starting in A_i, (ii) or there is a maximal pre-thread t starting from A_i such that $\mathsf{w}(t)$ is prefix of a word in $\mathcal{B}$ ends in the conclusion sequent or in the conclusion of a $\top$ rule (iii) or no such maximal pre-thread t (starting from A_i such that $\mathsf{w}(t)$ is prefix of a word in $\mathcal{B}$) exists and they mutually extend into an infinite pre-thead.*

Proof. By case distinction, the fourth case being disabled by the condition of well-sliced $(\&_i)$ rules. □

In establishing the cut-elimination result, an intermediate proof system will be useful, that of partially sliced $\mu\mathsf{MALL}^\infty$ pre-proofs, in which both $(\&)$, $(\&_1)$ and $(\&_2)$ occur:

The reader will notice that the additive $\&$-inferences occurring in the trace of a reduction path will always be sliced inferences ($(\&_1)$ or $(\&_2)$).

Proposition A.24. *All reducts of a persistent slice are (Ω)-free.*

Proof. The property relies on the simple observation that (i) any cut-reduction step from a persistent slice results in a persistent slice and (ii) if there is a reduction step from slice S which creates a (Ω) rule, then S is non-persistent. □

Proposition A.25 (Pull-back property)**.** *If $\pi \to^* \pi'$ (resp. $\pi \to^\omega \pi'$) and S' is a slice of π', then there is a slice S of π such that $S \to^* S'$ (resp. $S \to^\omega S'$).*

Proof. In the case of the finitary reduction, this is a well-known property of slices.

For the infinite fair reductions, it results from the fact that fair reductions are strongly convergent in the sense of infinitary rewriting. As a consequence, for any position in the resulting slice, one can find a finite prefix of the reduction sequence which produces it and we can trace it back to a set of slices of the original proof which can produce this prefix. The intersection of all those sets define a non-empty set of slices of which any element can be considered. The obtained slice is of course persistent since it reduces to S'. □

A.5.4 Additive bouncing validity criterion

Definitions 4.1 and 4.5 of (pre-)threads directly adapt to the additives as they are not specific to the multiplicative fragment.

Definition A.26 (Validity)**.** A slicing is ***valid*** if it is persistent and if it is valid in the multiplicative sense[10]. A $\mu\mathsf{MALL}^\infty$ pre-proof π is ***valid*** if all its persistent slicings are valid.

A.5.5 Additive cut-elimination theorem

We now state the cut-elimination theorem and give a schema of the proofs.

Theorem A.27. *Fair infinite cut-reduction on $\mu\mathsf{MALL}^\infty$ proofs is productive and produces valid proofs.*

Schema of the proof For cut-elimination, the proof goes by contradiction: assuming that we have a non productive fair cut-elimination, we may assume that it consists only of internal reduction steps and the trace of this cut-elimination is actually a slice with open premisses. It is actually contained in a persistent slice of π. As a consequence, there is an infinite branch of π which is entirely visited by the trace and this branch is visited by a thread thanks to additive validity. By adapting the truncated proof system to the additives and proving a restricted soundness result, we transform the valid persistent slice in a truncated derivation of the empty sequent by pruning the conclusion formulas which are never principal in the trace, from which results the contradiction.

For proving validity, it goes also by contradiction: assume the produced cut-free proof of $\vdash F\ \pi'$ contains a persistent slice S' containing an invalid branch β'. By the pull-back property, we find a persistent slice S of π reducing to S' which is valid by assumption. From the invalid branch β' one can build a cut-free proof $\beta'^\perp$ of $\vdash F^\perp$ together with a truncation τ' ensuring that F is interpreted a 0 while validity of S and adaptation wrt. τ' ensures that F is interpreted as $\top$, a contradiction.

[10] That is, every infinite branch of the slicing is visited by a valid thread having its visible part contained in the branch.

We now establish productivity of $\mu\mathsf{MALL}^\infty$ cut-elimination:

Theorem A.28. *Fair reduction sequences of $\mu\mathsf{MALL}^\infty$ are productive.*

To do so, first notice that the following notions of appendix A.4 can be straightforwardly adapted to the additive case (or to additive slices):

- Definition A.7 and Lemma A.10 adapt without change to *persistent slices* as it is not specific to the multiplicative case (but persistency is needed).
- Proposition 5.3 applies to the trace of a persistent slice π.
- Missing and unjustified sequents can be extended to reduction paths of $\mu\mathsf{MALL}^\infty$ pre-proofs (non-sliced) for the $(\&)$ connective as done already in [7].
- While truncations need no adaptation, the truncated semantics shall be adapted to the additives by adding the following clauses:

$$[\![(\varphi \& \psi)_\alpha]\!]^{\mathcal{E}} = [\![\varphi_{\alpha 1}]\!]^{\mathcal{E}} \wedge [\![\psi_{\alpha r}]\!]^{\mathcal{E}},$$

$$[\![(\varphi \oplus \psi)_\alpha]\!]^{\mathcal{E}} = [\![\varphi_{\alpha 1}]\!]^{\mathcal{E}} \vee [\![\psi_{\alpha r}]\!]^{\mathcal{E}}$$

- The truncated proof system is extended in the most natural way (as in[7]).
- Truncation induced by a reduction path is lifted to the additive case and it is well-defined since the additive inference would simply add a tricky case of *missing* sequent for a $\&$ premiss erased when reducing a $(\mathsf{Cut})/(\&)$ cut but there cannot be an ϵ-path in this case. Therefore the only case to treat is that of distinguished occurrences of unjustified sequents of type 1 which works as for the multiplicative.

As for adapting soundness (Prop. A.13), we actually do not need the full soundness result but only soundness wrt. a class of derivations that we introduce now:

Definition A.29. Given a truncation τ, a τ-adapted $\mu\mathsf{MALL}^\infty_\tau$ derivation π is a $\mu\mathsf{MALL}^\infty_\tau$ pre-proof such that (i) for all $(\varphi \& \psi)_\alpha$ occurring in π, either $\alpha \in \mathit{Dom}(\tau)$ or $\{\alpha \mathtt{l}, \alpha \mathtt{r}\} \cap \tau^{-1}(\top) \neq \emptyset$. (ii) given a $(\&)$ occurring in π of conclusion sequent s and principal formula $(\varphi_1 \& \varphi_2)_\alpha$, if $\alpha \mathtt{l} \notin \mathit{Dom}(\tau)$ (resp. $\alpha \mathtt{r} \notin \mathit{Dom}(\tau)$) there is no b-path starting down from the $(\varphi_1 \& \varphi_2)_\alpha$ occurrence of s ending in a $(\varphi_1^\perp \oplus \varphi_2^\perp)_\beta$ which principal formula of a $(\oplus_\mathsf{j})$ inference with $i \neq j$.

We will use soundness for τ-adapted proofs:

Proposition A.30. *Given a truncation τ and a valid τ-adapted $\mu\mathsf{MALL}^\infty_\tau$ derivation π of conclusion $\vdash \Gamma$, there exists a formula $F \in \Gamma$ such that $[\![F]\!] = \top$.*

Proof. The soundness proof for $\mu\mathsf{MLL}^\infty_\tau$ of proposition A.13 can be extended to this setting: first notice that the relation $\leq$ on pointed sequents that is used to transfer the marking through ϵ-path can be extended and we will use it in the following consistently with the underlying slice τ-adaptation suggests.

Construction of (s_i) shall now treat the additive inferences: (i) for the $(\oplus)$ rule, nothing is to be changed since the rule is unary: s_{i+1} is the premiss of s_i; (ii) for the $(\&)$ inference on s_i of principal occurrence $G = H \& K$, we reason as in [7]: since $[f_i(G)] = \mathbf{0}$ and $f_i(G)$ is of the form $H_m \& K_m$ where H_m and K_m are marking of H and K respectively, then either $[H_m] = \mathbf{0}$ or $[K_m] = \mathbf{0}$. Moreover, by τ-adaptation, we know that the address of one of H, K is in $\tau^{-1}(\top)$ and therefore we necessarily choose the other disjunct suppose w.l.o.g. that $[H_m] = \mathbf{0}$. We set s_{i+1} to be the premiss of s_i that contains H.

The definition of the sequence of markings (f_i) is trivially extended (more precisely the clauses for the $\&$ and $\oplus$ are those used in [7].

The multiplicative soundness argument can be carried over in this setting since by τ-adaptation, the branch we have built is part of the persistent valid slice induced by τ (that is the purpose for the τ-adaptation requirement) and we therefore have a thread for which the decreasing sequence of ordinal can be applied concluding soundness for those derivations. □

Note that even this restricted soundness for the truncated $\mu\mathsf{MALL}$ proof-system ensure soundness for the (untruncated) $\mu\mathsf{MALL}^\infty$ b-valid proofs. Indeed, assuming some valid proof π derives a sequent $\mathcal{S}$ which can be valued to $\perp$. One can define a truncation τ for which π is τ-adapted and such that for any $\&$ formula which is principal in π and interpreted as false, at least one of its premisses interpreted as false is not in the domain of the truncation, which is possible thanks to inhabitation of any persistent slice by a bouncing thread. From which the previous result can be applied providing the desired contradiction. Hence:

Theorem A.31. *$\mu\mathsf{MALL}^\infty$ is sound for the boolean semantics.*

We can finally establish productivity of cut-elimination:

Proof. Let π be a $\mu\mathsf{MALL}^\infty$ valid proof. By contradiction, assume that π has a fair infinite sequence of internal reductions. *Wlog.* we can assume that all reductions steps from π are internal.

For each $(\&)$ inference of π, the trace of this cut-elimination contains at most one premiss *ie.* it is contained in a slice: one can actually find a persistent slice S which contains the trace.

The previous remark ensures that for each $\&$ formula principal in the trace, the truncation τ of the reduction path contains one of its subformulas in its domain and that the truncated proof π_τ associated with the trace is τ-adapted.

Since the reduction contains only internal reductions, the conclusion formulas of π_τ are never principal in the π_τ and therefore we can erase them resulting in π'_τ which is a τ-adapted $\mu\mathsf{MALL}^\infty_\tau$ valid derivation of the empty sequent which cannot be by soundness (prop A.30). □

Theorem A.32. *Given π a μMALL$^\infty$ proof, any fair mcut-reduction from π produces a μMALL$^\infty$ proof.*

Proof. Let π be a $\mu\mathsf{MALL}^\infty_{\mathsf{m}}$ proof of conclusion $\vdash F$ and π' the cut-free pre-proof resulting from the previous property.

By contradiction, assume π' is non-valid. That means there exists a slice S' of π' (π' is cut-free so there is no persistency assumption applying here) and an infinite branch β' of S' such that β' is supported by no valid thread. By the pull-back property, S' has been built by reducing a persistent slice S of π and therefore β' corresponds to a reduction path from S. Since we are working in a persistent additive slice, the multiplicative validity can be extended in order to extract an infinite branch β invalid in S. The construction of the proof $\beta'^{\perp}$ used to obtain the contradiction is lifted to the additive case as follows:

- first one shall define truncation τ' not only by considering the occurrences of $F_1 \parr F_2$ which are principal in β', but also those of $F_1 \oplus F_2$ and extend the truncation to $\tau'(F_i) = 0$ if $F_1 \oplus F_2$ is the principal occurrence of a $(\oplus_{\mathsf{j}})$ rule with $i \neq j$.
- then the construction of the dual of branch β, $\beta^\perp$, is extended with the following clauses to the definition of page 20:
  - $F = G \& H$. Suppose *wlog.* that $G \in s_{i+1}$ (i.e., H left the branch β).
$$\beta^\perp_{\geq i}(\Delta) = \dfrac{\dfrac{\beta^\perp_{\geq i+1}(\Delta', G)}{\vdash \Delta'^\perp, G^\perp}}{\vdash \Delta'^\perp, F^\perp}\ (\oplus_1)$$
  - $F = G \oplus H$ and $G \in \mathtt{Dom}(\tau')$ and $H \notin \mathtt{Dom}(\tau')$. We set:
$$\beta^\perp_{\geq i}(\Delta) = \dfrac{\vdash \Delta'^\perp, G^\perp\ (\tau'),\ (\top) \qquad \dfrac{\beta^\perp_{\geq i+1}(\Delta', H)}{\vdash \Delta'^\perp, H^\perp}}{\vdash \Delta'^\perp, F^\perp}\ (\&)$$

$\beta'^{\perp}$ is cut-free and valid (therefore valid in the sense of [7]) since its threads are dual of the threads of β (which are invalid).

$\beta'^{\perp}$ is τ'-adapted (as $\beta^\perp$ is cut-free, this amounts to checking that the appropriate premiss of each $(\&)$ has its formula in the domain of τ' which is by design) and by proposition A.30, we conclude the desired contradiction since one the one hand we have $[\![F]\!] = \top$ and on the other hand we have $[\![F^\perp]\!] = \top$. □

A.6 (Un)decidability properties

In this appendix, we prove the undecidability of the general bouncing criterion and introduce a hierarchy of decidable sub-criteria.

Proof of decidability for the bounded height criterion

We start by detailing some structure of proofs, via the notion of *shortcut*.

Definition A.33. A *shortcut* is a finite pre-thread $t = uv$ where $\mathsf{w}(u) \in W^*A$ and v is a b-path. A shortcut t is *minimal* if no strict prefix of t is a shortcut.

Notice that if t is a shortcut, then t is an ϵ-path.

We will note (F, s) a pointed sequent: s is a sequent of the proof, and $F \in s$. We want to be able to follow threads where shortcuts have been removed. These threads behave like straight threads, except on cuts where they are allowed to jump from the starting point of a shortcut to its end. We will now formalize a description of such "jumping" threads.

Let $\Sigma_{jump} = \{\mathtt{W}, \mathtt{i}, \mathtt{l}, \mathtt{r}, c_l, c_r\}$, where c_l (resp. c_r) stands for left (resp. right) cut occurrences. A word $\tau \in \Sigma^*_{jump}$ will be called a ***relative address***. If P is a pre-proof and (F, s) is a pointed sequent of P, then a relative address τ points to another pointed sequent $\tau@(F, s)$ in P. We define this by induction on τ:

- If $\tau = \epsilon$ then $\tau@(F, s) = (F, s)$.
- If $\tau = \mathtt{W}\tau'$ then $\tau@(F, s) = \tau'@(F, s')$, where s' is the premiss of s containing F.
- If $\tau = \mathtt{i}\tau'$, $F = \sigma X.G$ (for some $\sigma \in \{\mu, \nu\}$) is principal in s with premiss s', then $\tau@F = \tau'@(G[F/X], s')$.
- If $\tau = \mathtt{l}\tau'$ (resp. $\tau = \mathtt{r}\tau'$), $F = G \star H$ (for some $\star \in \{\parr, \otimes\}$) is principal in s, then $\tau@(F, s) = \tau'@(F', s')$ where $F' = G$ (resp. $F' = H$) and s' is the premiss of s containing F'.
- If $\tau = c_l\tau'$ (resp. $\tau = c_r\tau'$), and the rule applied to s in P is a cut, then $\tau@(F, s) = \tau'@(F', s')$, where F' is the occurrence introduced by the cut on the left (resp. right) premiss s' of this cut.
- Otherwise, $\tau@(F, s)$ is undefined.

Lemma A.34. *Let P be a circular pre-proof. If t is a minimal shortcut from (F, s) to (F', s'), then $F \equiv F'$. Moreover, there is a relative address τ such that $(F', s') = \tau@(F, s)$. This τ is called the* effect *of t and noted effect(t). For each pointed sequent (F, s), there is at most one minimal shortcut starting in (F, s)*

Proof. Since the weight of minimal shortcut starts with W^*A, no choice is possible before an axiom is encountered. When going downwards, constraints will be pushed on the constraint stack, and the shortcut is again uniquely defined. When going upwards (i.e. after having seen a cut, and before the next axiom), two cases can occur. Either the constraint stack is not empty, and therefore it uniquely determines the path followed by the shortcut, or it is empty, which marks the end of the minimal shortcut t. The relative address τ is given by the position of the end of t relatively to the beginning of t in the proof tree. The fact that $F \equiv F'$ follows from the fact that any shortcut is an ϵ-path. □

If P is a pre-proof and (F, s) is a pointed sequent in P, we note $\mathit{short}(F, s)$ the minimal shortcut starting in (F, s) if it exists. If not, we fix $\mathit{short}(F, s) := \epsilon$. We also fix $\mathit{effect}(\epsilon) = \epsilon$. We will abbreviate $\mathit{effect}(\mathit{short}(F, s))$ by $\mathit{effect}(F, s)$ to lighten notations.

Remark 2. *If P is a circular pre-proof, and $(F, s), (F', s')$ are two pointed sequents of P corresponding to the same occurrence in the finite graph of P, then $\mathit{effect}(F, s) = \mathit{effect}(F', s')$.*

This remark allows us to compute only finitely many effects: one for each pointed sequent in the finite proof graph.

Definition A.35. An *s-thread* is a sequence $(F_i, s_i, \uparrow)_{i\in\omega}$ that obeys the same rules as a thread going only upwards, with some relaxation in the constraints between $(F_i, s_i, \uparrow)$ and $(F_{i+1}, s_{i+1}, \uparrow)$ defining a pre-thread. Indeed we add a new clause allowing the s-thread to take minimal shortcuts: (F_{i+1}, s_{i+1}) can be reached from (F_i, s_i) following the relative address $\mathit{effect}(F_i, s_i)$, i.e. $(F_{i+1}, s_{i+1}) = \mathit{effect}(F_i, s_i)@(F_i, s_i)$.

The weight of an s-thread is defined by generalizing the definition of weight of a thread, matching this new clause with $w_i = \mathrm{W}$. The notion of visible part and validity of an s-thread is then induced by this definition.

Notice that the visible part of an s-thread is obtained by simply removing steps introduced by this new clause, corresponding to shortcuts.

Lemma A.36. *An infinite branch is validated by a thread if and only if it is validated by an s-thread.*

Proof. The s-thread is obtained from the thread by compressing minimal shortcuts and replacing them with the new clause. Conversely, the thread can be obtained from the s-thread by replacing the new clause with minimal shortcuts. This transformation preserves the visible part. □

We now give the proof of Theorem 6.5, stating that any valid proof of $\mu\mathsf{MLL}^\omega$ is a k-proof for some $k \in \mathbb{N}$.

Proof. Let P be a valid proof of $\mu\mathsf{MLL}^\omega$. Each pointed sequent in P can be annotated with its effect, and with the height of its minimal shortcut (or with $(\epsilon, 0)$ if this minimal shortcut does not exist).

By Lemma A.36, all infinite branches of P are validated by s-threads, following effects annotating P. Let k be the maximal height annotating a pointed sequent in P. We obtain that P is a k-proof. □

The rest of the section is devoted to proving Theorem 6.6, stating that given a pre-proof P and an integer k, it is decidable whether P is a k-proof.

Proof. Let us note

$$\mathit{effect}_k(F, s) = \begin{cases} \mathit{effect}(F, s) & \text{if } \mathit{short}(F, s) \text{ has height} \le k \\ \epsilon & \text{otherwise} \end{cases}$$

For each (F, s) in the graph, $\mathit{effect}_k(F, s)$ can be computed in a finite time. Indeed, it suffices to follow the only possible thread starting in (F, s) in the graph of P, until we find a minimal shortcut or we detect a failure. Reasons for failure are:

- The weight does not begin with $\mathrm{W}^*\mathrm{A}$, i.e. an unfolding happen before the first axiom,
- the constraint stack gets higher than k,
- we detect a loop: the same pointed sequent is visited twice with identical stack content.

This corresponds to turning the automaton $\mathcal{A}_{thread}$ into a DFA, by bounding the size of the stack to k, and accept words of the form $\mathrm{W}^*\mathrm{A}\Sigma^*C$ by empty stack. This allows us to annotate each pointed sequent (F, s) with $\mathit{effect}_k(F, s)$.

We can now verify that the pre-proof is a k-proof, using a nondeterministic parity automaton $\mathcal{A}_s$ reading branches of P and guessing the existence of an s-thread. This automaton is identical to the one in [16] for straight threads, except that it can follow effects. While following the relative address given by an effect, the thread is considered hidden, and the action preformed on it does not influence the accepting condition of $\mathcal{A}_s$. Since the length of effects is globally bounded (there are finitely many of them), the relative adresses to follow can be stored in the state space of the automaton. The pre-proof P is a k-proof if and only if $\mathcal{A}_s$ accepts all branches. □

A.6.1 Details on the undecidability proof

We show that the validity condition is already undecidable for the proof system $\mu\mathsf{MLL}^\omega$.

We reduce from the halting problem for two-counter machines (2CM), known to be Σ^0_1-complete [30].

Here is a brief outline of the proof.

We start by recalling the definition of 2CM in the next section. These are finite-state deterministic machines manipulating two counters, able to perform Zero test, increment and decrement on each counter.

We then show how to encode the halting problem of a 2CM M using bouncing threads. The idea is to use the constraint stack to encode the value of counters, and position in the graph to encode the control state. Gadgets allow to increment or decrement each counter. The main difficulty lies in the tests performed by the machine: we want to design a conditional branching on the thread, depending on the value of the constraint stack. This can be done, but because of the linearity of the proof system, we cannot avoid leaving some extra constraints encoding the results of the tests, that will be collected by the thread later. Since we want to finish with empty constraint, we need to erase this extra information. To do this, we add a second gadget performing the computation in a dual way: results of tests are fed to the thread, that rewinds the computation while erasing these extra constraints. We can finally exit the detour with (almost) no constraints, and perform a visible ν-unfolding on the main branch, before looping back to the root of the proof.

The global pre-proof will be a valid proof according to the criterion if and only if the machine M halts.

Two Counter Machines

A 2CM M is a tuple (Q, q_0, q_f, δ) where Q is a finite set of states, q_0 is the initial state, q_f is the final state, and δ is the transition function. The machine has access to two counters storing nonnegative integer values. The counters are initialized to 0.

The possible actions of the machine are the following, where $\tau \in \{1, 2\}$ identifies one of the counters:

- $\mathrm{Inc}_\tau(q)$: increment counter τ, and jump to state q
- $\mathrm{Dec}_\tau(q)$: decrement counter τ, and jump to state q
- $\mathrm{Test}_\tau(q_Z, q_P)$: if the current value of counter τ is 0, jump to q_Z, else jump to q_P.

Let $\mathrm{Act} = \{Inc_\tau(q) \mid \tau \in \{1,2\}, q \in Q\} \cup \{Dec_\tau(q) \mid \tau \in \{1,2\}, q \in Q\} \cup \{\mathrm{Test}_\tau(q_Z, q_P) \mid \tau \in \{1,2\}, q_Z, q_P \in Q\}$ be the set of possible actions.

The transition function of M is a function $\delta : Q \setminus \{q_f\} \to \mathrm{Act}$, specifying which action is executed when each state is reached. No action is mapped to q_f, since the run stops when q_f is reached.

A *configuration* of the machine M is a triple $(p, k[1], k[2]) \in Q \times \mathbb{N}^2$ specifying the current state and the values for the two counters.

A *run* of the machine M is a sequence of configurations $(p_i, k[1]_i, k[2]_i)_{0 \le i \le n}$ such that $p_0 = q_0$, $k[1]_0 = k[2]_0 = 0$, $p_n = q_f$, and consistent with δ, i.e. for all $i \in [0, n-1]$:, we have

- if $\delta(p_i) = \mathrm{Inc}_\tau(q)$ then $p_{i+1} = q$ and $k[\tau]_{i+1} = k[\tau]_i + 1$.
- if $\delta(p_i) = \mathrm{Dec}_\tau(q)$ then $p_{i+1} = q$ and $k[\tau]_{i+1} = k[\tau]_i - 1$.
- if $\delta(p_i) = \mathrm{Test}_\tau(q_Z, q_P)$, then $k[\tau]_{i+1} = k[\tau]_i$, and
  - if $k[\tau]_i = 0$ then $p_{i+1} = q_Z$.
  - if $k[\tau]_i > 0$ then $p_{i+1} = q_P$.

In all cases the other counter is left unchanged, i.e. $k[3-\tau]_{i+1} = k[3-\tau]_i$

Without loss of generality, we can also assume that the run ends with both counter values equal to 0, i.e. $k[1]_n = k[2]_n = 0$.

The next theorem states that the halting problem is undecidable for Two Counter Machines.

Theorem A.37. *[30] Given a Two Counter Machine $M = (Q, q_0, q_f, \delta)$, it is undecidable to determine whether M has a run, by a reduction from Turing Machines halting problem.*

From machines to proofs

We will now encode the halting problem for 2CM into the problem of deciding whether a pre-proof is a proof.

We fix a machine $M = (Q, q_0, q_f, \delta)$.

We will build a pre-proof P such that the leftmost infinite branch can be validated by a bouncing thread if and only if there exists a run of the machine M. All the other branches of P will be validated by non-bouncing threads.

We will use throughout the proof the formulas F, G, where $F = \nu X.(X \parr X)$, $G = F^\perp = \mu X.(X \otimes X)$, and auxiliary formulas $A = \nu X.(X \parr X) \otimes X$ and $B = \mu X.(X \parr A \parr A)$. Their addresses will sometimes be omitted, keep in mind that a letter can represent different occurrences in a proof tree.

The thread will always follow a formula F when going upwards, and G when going downwards. The formula A will be used to ensure that all infinite branches except the leftmost one are validated by non-bouncing threads.

The conclusion of the proof P is the sequent G, F, B.

The idea of the construction is to use a bouncing thread to encode a run of M, by storing the current configuration on M in the stack of constraints that the thread must satisfy.

The general shape of the pre-proof P is given in Figure 13. By convention, formulas introduced in cuts will always be F on the left and G on the right. This means that a thread going upwards following a formula F will always turn right on cuts, bounce on axioms in the right part, and finally come back to visit the left part of the cut.

Auxiliary formulas are grayed to emphasize the trajectory of the thread of interest.

The thread on the branch with infinitely many ($\star$) must use formula F, as it is the only one performing a ν-unfolding. This means it has to go through the two cuts and bounce on axioms in π_M and π_R. All other infinite branches are validated by non-bouncing threads.

As described in the outline, the goal of π_M is to use the bouncing thread stemming from F to simulate a run of M in a deterministic way, accumulating "garbage constraints" for every test. The role of π_R is to erase these garbage constraints, by mirroring the behaviour of π_M.

After π_R, the formula F is unfolded twice before looping back to the root. The first unfolding is used to match a left-over constraint, that cannot be erased in π_R for technical reasons detailed later. The second unfolding contributes to the visible part of the thread.

Encoding of counters in the constraint stack

We describe here how the pre-proof π_M will be able to simulate a run of M via a thread following formula F.

When F is unfolded, a thread following F can either go to the left disjunct or to the right, corresponding to weights `il` or `ir`. We will use the alias **l** for `il` and **r** for `ir`, since unfoldings will always alternate with left/right choices. We will refer to the word $u \in \{\mathbf{l}, \mathbf{r}\}^*$ storing the current constraints as the *constraint stack*. For instance if the constraint stack u is of the form $\mathbf{l}v$, then when the thread goes up and encounters an unfolding of F, it has to follow the left disjunct, and update the constraint stack from u to v. Constraints are updated according to the stack of the pushdown automaton $\mathcal{A}_{thread}$ described in Section 6.1.

We are now ready to detail how configurations of the machine will be encoded in the constraint stack. A counter

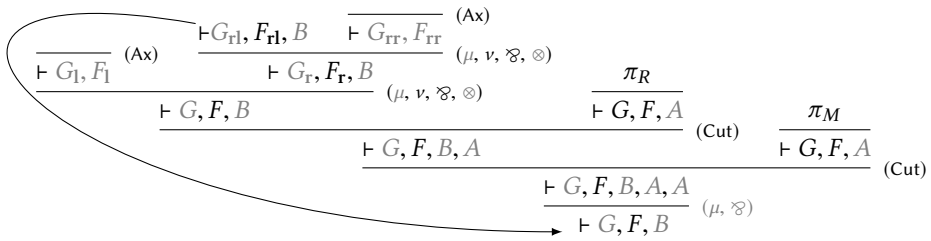


Figure 13. The main pre-proof P

of value n will be encoded by the sequence of constraints $(\mathbf{rl}^n\mathbf{r})$. Therefore, when the thread is simulating the run of M, its constraint stack is of the form $(\mathbf{rl}^n\mathbf{r})(\mathbf{rl}^m\mathbf{r})$ to denote that $k[1] = n$ and $k[2] = m$.

The current state is not encoded in the constraint stack, but in the current position of the thread in the pre-proof: for instance if the thread is in the node labeled (q_0) in the proof graph, then the current state of the corresponding run is q_0.

"Garbage constraints" are constraints that will not be part of the constraint stack during the simulation of the run of M, but will be pushed on the stack by the thread when the simulation has succeeded, i.e. after the node labeled (q_f) has been reached by the thread. In order to exit the cut, the thread must go down from (q_f) to (q_0), and that is where garbage constraints may be pushed. These extra constraints will encode the results of all tests performed during the computation. The result of a test is $\mathbf{r}$ if the tested counter was Zero, and $\mathbf{l}$ otherwise, so in general it is a letter $X \in \{\mathbf{r}, \mathbf{l}\}$. Garbage constraints will be of the form $\mathbf{r}X$ for a test on the first counter, and of the form $\mathbf{rl}^k\mathbf{rr}X$ for a test on second counter, where X is the result of the test and k is the value of the first counter.

Auxiliary metarules for π_M

In order to make the construction readable and modular, we start by describing metarules that will be used as building blocks throughout the section.

Rules relative to A

We will use the ∞ notation to denote an infinite tree obtained by unfolding a proof of A. This yields a valid cut-free proof tree that contains no axiom.

We explicit this from sequent Γ, A, where Γ can be any sequent.

$$\dfrac{}{\vdash \Gamma, A}\ (\infty) \equiv \dfrac{(\Delta) \qquad \dfrac{\dfrac{(\spadesuit) \qquad (\spadesuit)}{\vdash A, (A \parr A) \otimes A}\ (\otimes, \parr)}{(\spadesuit) \vdash A, A}\ (\nu)}{\dfrac{\vdash \Gamma, (A \parr A) \otimes A}{(\Delta) \vdash \Gamma, A}\ (\nu)}\ (\otimes, \parr)$$

We will often want to duplicate A using its $\parr$ connective. Let us define the following metarule, valid for any sequent Γ, A:

$$\dfrac{\vdash \Gamma, A, A}{\vdash \Gamma, A}\ (\parr_A) \equiv \dfrac{\dfrac{\vdash \Gamma, A, A}{\vdash \Gamma, A \parr A}\ (\parr) \qquad \dfrac{}{\vdash A}\ (\infty)}{\dfrac{\vdash \Gamma, (A \parr A) \otimes A}{\vdash \Gamma, A}\ (\nu)}\ (\otimes)$$

We will define an alias for a cut rule allowing to duplicate the A formula on both sides, noted (Acut). The effect of rule (Acut) is simply a cut for formulas F, G, so it can be considered as such for threads of interests following F upwards and G downwards.

$$\dfrac{\vdash G, F, A \qquad \vdash G, F, A}{\vdash G, F, A}\ (\text{Acut}) \equiv \dfrac{\vdash G, F, A \qquad \vdash G, F, A}{\dfrac{\vdash G, F, A, A}{\vdash G, F, A}\ (\parr_A)}\ (\text{Cut})$$

Copying left and right constraints

We now describe a helpful metarule: the expanding rule (exp), represented in Fig. 14. It allows to unfold F and G once, pairing the left (resp. right) unfolding of F with the left (resp. right) unfolding of G. This ensures that if the thread goes left (resp. right) upwards on F, it will also go left (resp. right) downwards on G. This can therefore be understood as a copying operator: the bit of the constraint stack will be the same before and after bouncing on the current cut. As before, we will gray formulas that will always be avoided by the thread of interest.

In most constructions, the left (resp. right) conjunct of F will be paired with the left (resp. right) conjunct of G when both are expanded, so we will often omit the labels. The left expansions will be represented on the left branch of the proof.

We give metarule (**l**) (resp. (**r**)) forcing the reading and copying of a left (resp. right) constraint on the stack. This is enforced by preventing the thread from bouncing on an axiom if the forbidden bit is read, thanks to the axiomless infinite proofs described by the (∞) metarules.

$$\frac{\vdash G_{\mathbf{l}}, F_{\mathbf{l}}, A \qquad \vdash G_{\mathbf{r}}, F_{\mathbf{r}}, A}{\vdash G, F, A}\ (\text{exp}) \equiv \quad \dfrac{\dfrac{\dfrac{\vdash G_{\mathbf{l}}, F_{\mathbf{l}}, A \qquad \vdash G_{\mathbf{r}}, F_{\mathbf{r}}, A}{\vdash G_{\mathbf{l}} \otimes G_{\mathbf{r}}, F_{\mathbf{l}}, F_{\mathbf{r}}, A, A}\ (\otimes)}{\vdash G, F, A, A}\ (\mu, \nu, \parr)}{\vdash G, F, A}\ (\parr_{\mathrm{A}})$$

Figure 14. the metarule (exp)

$$\frac{\vdash G, F, A}{\vdash G, F, A}\ (\mathbf{l}) \equiv \quad \frac{\vdash G_{\mathbf{l}}, F_{\mathbf{l}}, A \qquad \dfrac{}{\vdash G_{\mathbf{r}}, F_{\mathbf{r}}, A}\ (\infty)}{\vdash G, F, A}\ (\text{exp})$$

$$\frac{\vdash G, F, A}{\vdash G, F, A}\ (\mathbf{r}) \equiv \quad \frac{\dfrac{}{\vdash G_{\mathbf{l}}, F_{\mathbf{l}}, A}\ (\infty) \qquad \vdash G_{\mathbf{r}}, F_{\mathbf{r}}, A}{\vdash G, F, A}\ (\text{exp})$$

Constraint introduction

We might also want to push a right constraint without popping one. For instance this is needed at the beginning when the stack is empty. This can be done via the following metarules $\pi_{\mathbf{r}_{\mathrm{i}}}$ and $(\mathbf{r}_{\mathrm{i}})$, where i stands for "Introduction". We define both variants because we might want to use one or the other depending on the context.

Let $\pi_{\mathbf{r}_{\mathrm{i}}}$ be the following pre-proof:

$$\frac{\pi_{\mathbf{r}_{\mathrm{i}}}}{\vdash G, F, A} \equiv \quad \dfrac{\dfrac{\dfrac{}{\vdash G_{\mathbf{l}}, A}\ (\infty) \qquad \dfrac{}{\vdash G_{\mathbf{r}}, F}\ (\text{Ax})}{\vdash G_{\mathbf{l}} \otimes G_{\mathbf{r}}, F, A}\ (\otimes)}{\vdash G, F, A}\ (\mu)$$

This pre-proof allows a thread to go upwards following F without any event, bounce on an axiom, and go downwards following G while pushing a right constraint.

We now combine $\pi_{\mathbf{r}_{\mathrm{i}}}$ with a cut in order to go back to a thread going upwards, having a constraint stack starting with an extra $\mathbf{r}$.

$$\frac{\vdash G, F, A}{\vdash G, F, A}\ (\mathbf{r}_{\mathrm{i}}) \equiv \quad \frac{\vdash G, F, A \qquad \dfrac{\pi_{\mathbf{r}_{\mathrm{i}}}}{\vdash G, F, A}}{\vdash G, F, A}\ (\text{Acut})$$

This means that the effect of the rule $(\mathbf{r}_{\mathrm{i}})$ on the constraint stack can be summarized by $\epsilon \mapsto \mathbf{r}$, adding an extra $\mathbf{r}$ at the top of the constraint stack.

The above rule can be similarly defined to introduce a $\mathbf{l}$ constraint instead, by simply switching the G, F axiom to the left premiss instead of the right in $\pi_{\mathbf{r}_{\mathrm{i}}}$. This dual version will be noted by $\pi_{\mathbf{l}_{\mathrm{i}}}$ and metarule $(\mathbf{l}_{\mathrm{i}})$.

The initialization metarule

We describe here the first metarule encountered in π_M. Its role is to initialize the constraint stack to encode two counters with value 0, and go to the (q_0) node to start the simulation of the run of M.

It must therefore allow the thread to enter with empty constraint stack and exit the cut with a constraint stack $(\mathbf{rr})(\mathbf{rr})$. This is straightforward now that we have the $(\mathbf{r}_{\mathrm{i}})$ rule:

$$\frac{(q_0) \vdash G, F, A}{\vdash G, F, A}\ (\text{init}) \equiv \quad \dfrac{\dfrac{\dfrac{\dfrac{(q_0) \vdash G, F, A}{\vdash G, F, A}\ (\mathbf{r}_{\mathrm{i}})}{\vdash G, F, A}\ (\mathbf{r}_{\mathrm{i}})}{\vdash G, F, A}\ (\mathbf{r}_{\mathrm{i}})}{\vdash G, F, A}\ (\mathbf{r}_{\mathrm{i}})$$

In the following, we describe how to build the pre-proof π_M, by connecting nodes of the form (p) to their successors in the computation. So for each $p \in Q$, a pre-proof of "local root" (p) will be built with hypotheses of the form (q) with $q \in Q$. Once such a pre-proof has been built for each node (p), the hypotheses (q) are connected to their corresponding "local root" node trough back loops.

Encoding the action Inc

We now assume that we are at a node of the proof graph labeled by (p), where p is a state of the machine M, and that the current constraint stack encodes the counter value as described earlier, i.e. $(\mathbf{rl}^{k[1]}\mathbf{r})(\mathbf{rl}^{k[2]}\mathbf{r})$.

Assume $\delta(p) = \mathrm{Inc}_1(q)$ with $q \in Q$. We will build a metarule (Inc_1) updating the configuration by acting on the constraint stack, and ending up in node (q):

$$\frac{(q) \vdash G, F, A}{(p) \vdash G, F, A}\ (\mathrm{Inc}_1) \equiv \quad \frac{(q) \vdash G, F, A \qquad \dfrac{\dfrac{\pi_{\mathbf{l}_{\mathrm{i}}}}{\vdash G_{\mathbf{r}}, F_{\mathbf{r}}, A}}{\vdash G, F, A}\ (\mathbf{r})}{(p) \vdash G, F, A}\ (\text{Acut})$$

Notice that in order to bounce on the axiom, the thread must see $\mathbf{r}$ while going up, and $\mathbf{rl}$ on the way down. The rest of the current stack (of the form $\mathbf{l}^*\mathbf{r}(\mathbf{rl}^*\mathbf{r})$) is left unchanged. This rule turns a stack of the form $\mathbf{r}u$ into $\mathbf{rl}u$, thereby incrementing the first counter. We will abbreviate this action $\mathbf{r} \mapsto \mathbf{rl}$.

We might also need to increment the second counter, which is deeper in the stack. For this, let us devise another auxiliary metarule (counter), allowing us to skip the part of the stack encoding the first counter.

$$\dfrac{\vdash G, F, A}{\vdash G, F, A}\ (\text{counter}) \equiv \dfrac{\dfrac{(\dagger) \vdash G_{\mathbf{rl}}, F_{\mathbf{rl}}, A \qquad \vdash G_{\mathbf{rr}}, F_{\mathbf{rr}}, A}{(\dagger) \vdash G_{\mathbf{r}}, F_{\mathbf{r}}, A}\ (\text{exp})}{\vdash G, F, A}\ (\mathbf{r})$$

This metarule processes constraints of the form $(\mathbf{rl}^*\mathbf{r})$ on the way up, and these same constraints will be copied back when the thread returns downwards from the formula G of the hypothesis. This means that this gadget allows the right premiss sequent to access the encoding of the second counter, while leaving the first one untouched.

We can now give the pre-proof allowing to increment the second counter, by simply adding the (counter) metarule at the appropriate place:

$$\dfrac{(q) \vdash G, F, A}{(p) \vdash G, F, A}\ (\text{Inc}_2) \equiv \dfrac{(q) \vdash G, F, A \qquad \dfrac{\dfrac{\dfrac{\pi_{\mathbf{l}_i}}{\vdash G_{\mathbf{r}}, F_{\mathbf{r}}, A}\ (\mathbf{r})}{\vdash G, F, A}\ (\text{counter})}{\vdash G, F, A}}{(p) \vdash G, F, A}\ (\text{Acut})$$

The effect of this metarule is $(\mathbf{rl}^*\mathbf{r})\mathbf{r} \mapsto (\mathbf{rl}^*\mathbf{r})\mathbf{rl}$

This achieves the treatment of states performing an increment. For all nodes (p) where p performs an increment of counter τ before going to q, we link node (p) of the proof with node (q) through the metarule (Inc_τ).

Encoding the action Dec

Assume $\delta(p) = \text{Dec}_1(q)$ with $q \in Q$.

This means we want to build a metarule with action $\mathbf{rl} \mapsto \mathbf{r}$ on the stack, in order to decrease the value of the first counter by 1.

This is done by the following metarule (Dec_1):

$$\dfrac{(q) \vdash G, F, A \qquad \dfrac{\dfrac{\dfrac{\dfrac{\overline{G_{\mathbf{l}}, F_{\mathbf{l}}, F_{\mathbf{rr}}, A}\ (\infty) \qquad \overline{G_{\mathbf{r}}, F_{\mathbf{rl}}}\ (\text{Ax})}{\vdash G_{\mathbf{l}} \otimes G_{\mathbf{r}}, F_{\mathbf{l}}, F_{\mathbf{rl}}, F_{\mathbf{rr}}, A}\ (\otimes)}{\vdash G_{\mathbf{l}} \otimes G_{\mathbf{r}}, F_{\mathbf{l}}, F_{\mathbf{r}}, A}\ (\nu, \parr)}{\vdash G, F, A}\ (\mu, \nu, \parr)}{(p) \vdash G, F, A}\ (\text{Acut})$$

Notice that if the thread does not start with **rl**, it gets lost in an (∞) proof, corresponding to a failure of the run.

If $\delta(p) = \text{Dec}_2(q)$, the construction is similar, using again the metarule (counter) to leave the first counter untouched and access the second one. This is done by the following metarule (Dec_2).

$$\dfrac{(q) \vdash G, F, A \qquad \dfrac{\dfrac{\dfrac{\dfrac{\overline{G_{\mathbf{l}}, F_{\mathbf{l}}, F_{\mathbf{rr}}, A}\ (\infty) \qquad \overline{G_{\mathbf{r}}, F_{\mathbf{rl}}}\ (\text{Ax})}{\vdash G_{\mathbf{l}} \otimes G_{\mathbf{r}}, F_{\mathbf{l}}, F_{\mathbf{rl}}, F_{\mathbf{rr}}, A}\ (\otimes)}{\vdash G_{\mathbf{l}} \otimes G_{\mathbf{r}}, F_{\mathbf{l}}, F_{\mathbf{r}}, A}\ (\nu, \parr)}{\vdash G, F, A}\ (\mu, \nu, \parr)}{\vdash G, F, A}\ (\text{counter})}{(p) \vdash G, F, A}\ (\text{Acut})$$

Encoding the action Test

It remains to describe how to modify the constraint stack for actions of type Test.

Test on the first counter

Let us assume first that $\delta(p) = \text{Test}_1(q_Z, q_P)$.

Notice that a zero test can be performed by simply testing whether the stack starts with **rr** or with **rl**, i.e. by identifying the second letter of the stack. Therefore, these first two letters can be used to branch to the result of the test. However, since they must be still be part of the encoding, we need to reintroduce them in the stack after having read them. We define the metarule (Test_1) accordingly:

$$\dfrac{\dfrac{\dfrac{(q_P) \vdash G, F, A}{\vdash G, F, A}\ (\mathbf{l}_i)}{\vdash G_{\mathbf{rl}}, F_{\mathbf{rl}}, A}\ (\mathbf{r}_i) \qquad \dfrac{\dfrac{(q_Z) \vdash G, F, A}{\vdash G, F, A}\ (\mathbf{r}_i)}{\vdash G_{\mathbf{rr}}, F_{\mathbf{rr}}, A}\ (\mathbf{r}_i)}{\vdash G_{\mathbf{r}}, F_{\mathbf{r}}, A}\ (\text{exp})}{(p) \vdash G, F, A}\ (\mathbf{r})$$

This metarule allows the thread to go to (q_Z) if the counter was zero, or to (q_P) if the counter was strictly positive, leaving the stack unchanged in both cases.

Notice that this metarule also leaves some garbage constraints in the following sense: when going back down from (q_Z) (resp. (q_P)) to (p), the thread will push extra constraints **rr** (resp. **rl**) on top of the pile, due to the (exp) and (**r**) rule in (Test_1).

Test on the second counter

We now assume that $\delta(p) = \text{Test}_2(q_Z, q_P)$, and we want to encode the corresponding metarule, linking (p) to (q_Z) and (q_P) in the pre-proof π_M.

This is more tricky, because we need to access the relevant bit encoding the result of this test, and copy the value of the first counter after it to restore the stack. This corresponds to copying an unbounded amount of information, so this cannot be done directly in the same way as in the previous construction for Test_1.

We therefore design auxiliary gadgets allowing us to copy the information bit by bit. The result T of a test will be encoded by $T = \mathbf{rrr}$ for zero and $T = \mathbf{rrl}$ for not zero.

The pre-proof π_{shift}, represented Fig. 15 has effect $\mathbf{l}^{k+1}T \mapsto \mathbf{l}^k T\mathbf{l}$, with $T \in \{\mathbf{rrr}, \mathbf{rrl}\}$. The proof can be built thanks to the following table, that explicits the transformation of the relevant prefix constraint stack:

before:	**ll**	**lrrl**	**lrrr**
after:	**ll**	**rrll**	**rrrl**

Notice that all axioms in the right part of the cut in π_{shift} are paired according to the table above.

The main interesting phenomenon in π_{shift} occurs on the (•) loop. In the case where the result starts with **l**, this loop allows to enter the cut again, after the popping of one **l** constraint. This will therefore perform the wanted transformation on the constraint stack. Let us take an explicit example to see this gadget at work: consider a thread entering π_{shift} with constraint $u = \mathbf{llrrr}$. The thread will bounce on axiom $G_{\mathbf{ll}}, F_{\mathbf{ll}}$, leaving this constraint unchanged, and it will enter the (•) node with constraint **lrrr**. This time, the detour will enter π_{aux}, and will pop **lrrr** and push **rrrl** onto the stack. When exiting the cut, the thread will have a constraint starting with **r** and therefore will immediately bounce on the axiom. When going back, it will push back the first **l** on the way down to the original root. It will finally exit with constraint **lrrrl**, which is the wanted result of the mapping $\mathbf{l}^k T \mapsto \mathbf{l}^{k-1}T\mathbf{l}$.

We can now iterate π_{shift} in order to move the result T on top of the stack.

Let us start with an auxiliary metarule ($\text{copy}_\mathbf{l}$) (Fig. 16) copying the first letter of the stack if it is **l**, and do nothing if the stack starts with **r**. I.e. it has action $\begin{cases} \mathbf{l} \mapsto \mathbf{ll} \\ \mathbf{r} \mapsto \mathbf{r} \end{cases}$ on the stack. It will actually be the case that if the constraint starts with **r**, it starts with **rr**.

We now build the proof π_{move} (Fig. 17), iterating π_{shift}, allowing to copy an unbounded quantity of information past the test result T. The proof π_{move} has the following action on the stack: $\begin{cases} \mathbf{l}^k T \mapsto \mathbf{l}^k T \mathbf{l}^k \\ \mathbf{r} \mapsto \mathbf{r} \end{cases}$. The principle is to first duplicate the leading **l**, so that the extra occurrence can be used to test whether we want to perform a shifting using π_{shift}. If the stack starts with **r** (actually with **rr**), then the ($\text{copy}_\mathbf{l}$) metarule will do nothing, and the thread will just bounce on the right axiom. This allows us to iteratively call π_{shift}, until we reach the encoded result T. When exiting this gadget, the last π_{shift} leaves constraint $T\mathbf{l}^k$, and one extra **l** is collected by each (★) loop. That is why the constraint afer π_{move} is $\mathbf{l}^k T \mathbf{l}^k$.

Another auxiliary metarule (prep) (Fig. 18) will allow us to prepare the input for π_{move} from the standard counter encoding, i.e. performing action $\mathbf{rl}^k\mathbf{rr}X \mapsto \mathbf{rl}^k(\mathbf{rr}X)\mathbf{rr}X$, with $X \in \{\mathbf{r}, \mathbf{l}\}$. The parenthesized expression is the T that we will want to move to the top. Notice that this corresponds to a copying of the counter $\mathbf{rl}^k\mathbf{r}$, followed by a mapping $\mathbf{rr}X \mapsto \mathbf{r}X\mathbf{rr}X$. The dots in Fig. 18 represents (∞) proofs, not detailed for concision.

Let us now combine these gadgets to define a metarule (result), moving the test result at the wanted place, while producing extra stack content before it. This metarule has action $\mathbf{rl}^k T \mapsto \mathbf{l}^k T \mathbf{l}^k T$, with $T \in \{\mathbf{rrl}, \mathbf{rrr}\}$, and leaves a garbage constraint **r** that will be seen later on the way down.

$$\frac{\vdash G, F, A}{\vdash G, F, A}\ (\text{result}) \equiv \cfrac{\cfrac{\cfrac{\vdash G, F, A \qquad \cfrac{\pi_{\text{move}}}{\vdash G, F, A}}{\vdash G_{\mathbf{r}}, F_{\mathbf{r}}, A}\ (\text{Acut})}{\vdash G, F, A}\ (\mathbf{r})}{\vdash G, F, A}\ (\text{prep})$$

We can detail the stack modifications in (result): $(\mathbf{rl}^k)T \xrightarrow{(\text{prep})} \mathbf{rl}^k TT \xrightarrow{(\mathbf{r})} \mathbf{l}^k TT \xrightarrow{\pi_{\text{move}}} \mathbf{l}^k T\mathbf{l}^k T$.

We can finally build the metarule for Test_2, performing the wanted test and leaving garbage constraint of the form $\mathbf{rl}^k\mathbf{rr}X$ with $X \in \{\mathbf{l}, \mathbf{r}\}$.

$$\cfrac{(\clubsuit) \vdash G_{\mathbf{l}}, F_{\mathbf{l}}, A \qquad \cfrac{\cfrac{\cfrac{(q_P) \vdash G, F, A}{\vdash G_{\mathbf{rrl}}, F_{\mathbf{rrl}}, A}\ (\mathbf{r}_\text{i}) \qquad \cfrac{(q_Z) \vdash G, F, A}{\vdash G_{\mathbf{rrr}}, F_{\mathbf{rrr}}, A}\ (\mathbf{r}_\text{i})}{\vdash G_{\mathbf{rr}}, F_{\mathbf{rr}}, A}\ (\text{exp})}{\vdash G_{\mathbf{r}}, F_{\mathbf{r}}, A}\ (\mathbf{r})}{(\clubsuit) \vdash G, F, A}\ (\text{exp})$$
$$\frac{(\clubsuit) \vdash G, F, A}{(p) \vdash G, F, A}\ (\text{result})$$

The principle of this gadget is the following: after preparing the stack via the (result) metarule, the (♣) loops and the (**r**) rule first pop the garbage prefix $\mathbf{l}^k\mathbf{rr}$. The following bit $X \in \{\mathbf{l}, \mathbf{r}\}$ is the wanted test result, and allows us to enter (q_Z) or (q_P) with a remaining stack that encodes the next configuration of the machine (after adding the leading **r** to complete the valid encoding).

Final state q_f

It remains to describe what happens to a thread entering the node labelled by the final state q_f. We will simply allow it to finally bounce on an axiom, thereby starting a downwards path that will gather all the garbage constraints, exit π_M, and enter the second cut and the proof π_R.

We just need to evacuate the formula A from the sequent. Since we know that the constraint stack starts with **r**, this can be done in the following way:

$$\cfrac{\cfrac{}{\vdash G_{\mathbf{l}}, F_{\mathbf{l}}, A}\ (\infty) \qquad \cfrac{}{\vdash G_{\mathbf{r}}, F_{\mathbf{r}}}\ (\text{Ax})}{(q_f) \vdash G, F, A}\ (\mu, \nu, \parr, \otimes)$$

$$\dfrac{\pi_{\mathsf{shift}}}{\vdash G, F, A} \equiv \dfrac{\dfrac{(\bullet) \vdash G_{\mathbf{l}}, F_{\mathbf{l}}, A \qquad \dfrac{}{\vdash G_{\mathbf{r}}, F_{\mathbf{r}}}\,(\mathrm{Ax})}{\vdash G, F, A}\,(\mu, \nu, \parr, \otimes) \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\vdash G_{\mathbf{ll}}, F_{\mathbf{ll}}}\,(\mathrm{Ax}) \qquad \dfrac{}{\vdash G_{\mathbf{lr}}, F_{\mathrm{r}}, A}\,(\infty)}{\vdash G_{\mathbf{l}}, F_{\mathbf{ll}}, F_{\mathrm{r}}, A}\,(\mu, \otimes) \qquad \dfrac{\pi_{\mathsf{aux}}}{\vdash G_{\mathbf{r}}, F_{\mathbf{lr}}, A}}{\vdash G_{\mathbf{l}} \otimes G_{\mathbf{r}}, F_{\mathbf{ll}}, F_{\mathbf{lr}}, F_{\mathrm{r}}, A, A}\,(\otimes)}{\vdash G, F, A, A}\,(\mu, (\nu, \parr)^2) \qquad \dfrac{}{\vdash G, F, A}\,(\parr_{\mathrm{A}})}{(\bullet) \vdash G, F, A}\,(\mathrm{Acut})$$

$$\dfrac{\pi_{\mathsf{aux}}}{\vdash G_{\mathbf{r}}, F_{\mathbf{lr}}, A} \equiv \dfrac{\dfrac{}{\vdash G_{\mathbf{rl}}, F_{\mathbf{lrl}}, A}\,(\infty) \qquad \dfrac{\dfrac{\dfrac{}{\vdash G_{\mathbf{rrll}}, F_{\mathbf{lrrl}}}\,(\mathrm{Ax}) \qquad \dfrac{}{\vdash G_{\mathbf{rrlr}}, A}\,(\infty)}{\vdash G_{\mathbf{rrl}}, F_{\mathbf{lrrl}}, A}\,(\mu, \otimes) \qquad \dfrac{\dfrac{}{\vdash G_{\mathbf{rrrl}}, F_{\mathbf{lrrr}}}\,(\mathrm{Ax}) \qquad \dfrac{}{\vdash G_{\mathbf{rrlr}}, A}\,(\infty)}{\vdash G_{\mathbf{rrr}}, F_{\mathbf{lrrr}}, A}\,(\mu, \otimes)}{\vdash G_{\mathbf{rr}}, F_{\mathbf{lrr}}, A}\,(\mu, \nu, \parr, \otimes)}{\vdash G_{\mathbf{r}}, F_{\mathbf{lr}}, A}\,(\mathrm{exp})$$

Figure 15. The pre-proof π_{shift} with its auxiliary π_{aux}

$$\dfrac{\vdash G, F, A}{\vdash G, F, A}\,(\mathsf{copy}_{\mathbf{l}}) \equiv \dfrac{\vdash G, F, A \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\vdash G_{\mathbf{ll}}, F_{\mathbf{l}}, A}\,(\mathrm{Ax}) \qquad \dfrac{}{\vdash G_{\mathbf{lr}}, A}\,(\infty)}{\vdash G_{\mathbf{l}}, F_{\mathbf{l}}, A}\,(\mu, \otimes) \qquad \dfrac{}{\vdash G_{\mathbf{r}}, F_{\mathbf{r}}}\,(\mathrm{Ax})}{\vdash G_{\mathbf{l}} \otimes G_{\mathbf{r}}, F_{\mathbf{l}}, F_{\mathbf{r}}, A}\,(\otimes)}{\vdash G, F, A}\,(\mu, \nu, \parr)}{\vdash G, F, A}\,(\mathrm{Acut})$$

Figure 16. The metarule ($\mathsf{copy}_{\mathbf{l}}$)

$$\dfrac{\pi_{\mathsf{move}}}{\vdash G, F, A} \equiv \dfrac{\dfrac{\dfrac{(\bigstar) \vdash G, F, A \qquad \pi_{\mathsf{shift}}}{\vdash G_{\mathbf{l}}, F_{\mathbf{l}}, A}\,(\mathrm{Acut}) \qquad \dfrac{}{\vdash G_{\mathbf{r}}, F_{\mathbf{r}}}\,(\mathrm{Ax})}{\vdash G, F, A}\,(\mu, \nu, \parr, \otimes)}{(\bigstar) \vdash G, F, A}\,(\mathsf{copy}_{\mathbf{l}})$$

Figure 17. The metarule π_{move}

$$\dfrac{\vdash G, F, A}{\vdash G, F, A}\,(\mathrm{prep}) \equiv \dfrac{\vdash G, F, A \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{G_{\mathbf{l}}, F_{\mathbf{l}}, F_{\mathbf{rl}}, A}\,(\infty) \qquad \dfrac{\dfrac{}{\vdash G_{\mathbf{rlrrl}}, F_{\mathbf{rrl}}}\,(\mathrm{Ax}) \qquad \dfrac{}{\vdash G_{\mathbf{rrrrr}}, F_{\mathbf{rrr}}}\,(\mathrm{Ax})}{\vdash G_{\mathbf{r}}, F_{\mathbf{rrl}}, F_{\mathbf{rrr}}, A}\,(\mu, \otimes) \quad \cdots}{\vdash G, F_{\mathbf{l}}, F_{\mathbf{rl}}, F_{\mathbf{rrl}}, F_{\mathbf{rrr}}, A, A}\,((\nu, \parr)^3, \parr_{\mathrm{A}})}{\vdash G, F, A}\,(\mathrm{counter})}{\vdash G, F, A}\,(\mathrm{Acut})}{\vdash G, F, A}\,(\otimes, \mu, \parr_{\mathrm{A}})^*$$

Figure 18. The metarule prep

Exiting the proof π_M

This concludes the description of the proof π_M, starting with $\dfrac{(q_0) \vdash G, F, A}{\vdash G, F, A}\,(\mathrm{init})$, and built as described by linking state-labelled nodes using rules we defined for the action performed by each state according to δ.

This construction ensures the following Lemma:

Lemma A.38. *A thread entering π_M with empty constraint will be able to exit it if and only if M has a run. If this is the case, the constraint after the exit is a word $u_1 u_2 \dots u_k \mathbf{r}^4$, where k is the number of tests performed, and u_i encodes the results $X \in \{\mathbf{l}, \mathbf{r}\}$ of the i^{th} test in the following way:*

- *if the test is on the first counter, then $u_i = \mathbf{r}X$,*

- *if the test is on the second counter, then* $u_i = \mathbf{rl}^k\mathbf{rr}X$*, where* k *is the value of the first counter at the time of the test.*

Proof. It is straightforward to prove by induction of the length of the thread/run: a thread entering π_M with empty constraint can reach node (p) with constraint $(\mathbf{rl}^n\mathbf{r})(\mathbf{rl}^m\mathbf{r})$ if and only if their is a partial run of M reaching state p with counter values n, m. Moreover, the garbage constraints that will be pushed back from (p) on the way back to the root of π_M encode the results of tests as described in the statement of the Lemma. The $\mathbf{r}^4$ following garbage constraints is the constraint stack reached in (q_f) at the end of computation, encoding two counters of value 0, as we assumed M ends with this configuration. □

However, we want the thread to be back on the main branch with a bounded number of constraints. We therefore must erase all these garbage constraints. This will be the role of the pre-proof π_R.

The reverse simulation proof π_R

The goal of the proof π_R will be to erase the garbage constraints instead of creating them. To achieve this, we will aim at building a dual version of the pre-proof π_M. A convenient way to think about it is the following: we try to reproduce the proof π_M, but considering this time that the thread of interest t' originates in the sequent G. Therefore it will always follow G upwards and F downwards. We call such a thread a G-thread, and the previous version used in π_M a F-thread.

The principle is that if the G-thread t' creates garbage constraints u, i.e. has a visible weight $u \in \Sigma^*$ then its dual, the identical thread considered in reverse and originating in F, has a visible weight $dual(u)$, where the function $dual : \Sigma^* \to \Sigma^*$ is defined as follows. Let $v \mapsto \bar{v} : \Sigma^* \to \Sigma^*$ be the length-preserving morphism defined on letters by the following correspondence:

x :	l	r	i	$\bar{l}$	$\bar{r}$	$\bar{i}$	A	C	W
$\bar{x}$:	$\bar{l}$	$\bar{r}$	$\bar{i}$	l	r	i	A	C	W

Let v^R be the reverse of a word v, defined by induction: $\epsilon^R = \epsilon$ and if $(u, a) \in \Sigma^* \times \Sigma$ then $(ua)^R = a(u^R)$.

We now define $dual(u) = \bar{u}^R$.

Lemma A.39. *Let* t *be a thread from* φ_α *to* ψ_β*, and* $dual(t)$ *be the identical thread considered in the other direction, from* ψ_β *to* φ_α*. Then* $\mathsf{vp}(t^R) = dual(\mathsf{vp}(t))$

Let t_M be the F-thread of the main pre-proof P going through the π_M cut and ending with the cut rule, and t_R be the analog F-thread for the π_R cut. We want to build π_R such that $\mathsf{vp}((t_R)^R) = \mathsf{vp}(t_M)$. By Lemma A.39, this implies $\mathsf{vp}(t_R) = dual(\mathsf{vp}(t_M))$, and moreover $\mathsf{vp}(t_M) \subseteq \{\bar{l}, \bar{r}, \bar{i}\}^*$, and $\mathsf{w}(t_M)$ ends with C. Therefore $\mathsf{vp}(t_M t_R) = \epsilon$.

In the following, we will describe how gadgets of π_M are dualized to create π_R, where the run of the machine is simulated by a G-thread.

Dual auxiliary metarules

Rules relatives to A are left unchanged, so we will freely used (∞), $(\parr_A)$, (Acut).

The rules (exp), $(\mathbf{r})$, $(\mathbf{l})$ also stay identical.

The main difference will occur when we want to introduce or delete a constraint, since they will now be reversed. In particular, in the previous construction, we saw that introducing a constraint could be done at will without any assumption, but removing a constraint as done in (Dec_1) needed the presence of a known bit occuring before the constraint to be removed.

We therefore redefine constraint introduction gadgets $\pi_{\mathbf{r}'_i}$ and $(\mathbf{r}'_i)$, with the notable change that the effect on stack is now $\mathbf{r} \mapsto \mathbf{rr}$, i.e. we always assume that the stack starts with $\mathbf{r}$. We keep the convention for cuts, i.e. the newly introduced cut formulas are written on the inside (left F and right G).

$$\frac{\pi_{\mathbf{r}'_i}}{\vdash G, F, A} \equiv \cfrac{\cfrac{\cfrac{}{G_{\mathbf{l}}, F_{\mathbf{l}}, F_{\mathbf{rl}}, A}\,(\infty) \qquad \cfrac{}{G_{\mathbf{r}}, F_{\mathbf{rr}}}\,(\mathrm{Ax})}{\vdash G_{\mathbf{l}} \otimes G_{\mathbf{r}}, F_{\mathbf{l}}, F_{\mathbf{rl}}, F_{\mathbf{rr}}, A}\,(\otimes)}{\vdash G_{\mathbf{l}} \otimes G_{\mathbf{r}}, F_{\mathbf{l}}, F_{\mathbf{r}}, A}\,(\nu, \parr)}{\vdash G, F, A}\,(\mu, \nu, \parr)$$

$$\frac{\vdash G, F, A}{\vdash G, F, A}\,(\mathbf{r}'_i) \equiv \cfrac{\cfrac{\pi_{\mathbf{r}'_i}}{\vdash G, F, A} \qquad \vdash G, F, A}{\vdash G, F, A}\,(\mathrm{Acut})$$

It is easily verified that a G-thread entering with constraint $\mathbf{r}$ will exit with constraint $\mathbf{rr}$.

As before, we also define the left analog $\pi_{\mathbf{l}'_i}$ and $(\mathbf{l}'_i)$, with effect $\mathbf{r} \mapsto \mathbf{rl}$.

Initialisation of π_R

We now want to initialize the dual thread. However, since the introduction rule needs to assume an $\mathbf{r}$ constraint already on the stack, we will need to assume this for the G-thread entering π_R. This means that in the end, the F-thread exiting π_R via G (the real thread of interest) will have a leftover constraint $\mathbf{r}$, that we will need to evacuate on the main branch, as done in the main pre-proof P.

Thus, we only need to add three $\mathbf{r}$ constraints to the one already assumed:

$$\frac{(q'_0) \vdash G, F, A}{\vdash G, F, A}\,(\mathrm{init}') \equiv \cfrac{\cfrac{\cfrac{(q'_0) \vdash G, F, A}{\vdash G, F, A}\,(\mathbf{r}'_i)}{\vdash G, F, A}\,(\mathbf{r}'_i)}{\vdash G, F, A}\,(\mathbf{r}'_i)$$

We will use the (p') notation with $p \in Q$ for state-labelled nodes of π_R, to distinguish them from nodes in π_M.

Dual encoding of action Inc

Assume $\delta(p) = \mathrm{Inc}_1(q)$ with $q \in Q$. We want to define a metarule (Inc'_1) updating the configuration reached in (p') by acting on the constraint stack, and ending up in node (q'), with effect $\mathbf{r} \mapsto \mathbf{rl}$ on the stack. Therefore it suffices to use the $\mathbf{l}$ introduction defined before:

$$\dfrac{(q') \vdash G, F, A}{(p') \vdash G, F, A}\ (\mathrm{Inc}'_1) \equiv \quad \dfrac{(q') \vdash G, F, A}{(p') \vdash G, F, A}\ (\mathbf{l}'_i)$$

To manipulate the second counter instead, we can use the (counter) rule which is identical as in π_M. Thus the metarule (Inc'_2) is defined as follows:

$$\dfrac{\dfrac{\dfrac{\pi_{\mathbf{l}'_i}}{\vdash G, F, A}}{\vdash G, F, A}\ (\text{counter}) \qquad (q') \vdash G, F, A}{(p') \vdash G, F, A}\ (\text{Acut})$$

The effect of this metarule on a G-thread is $(\mathbf{rl}^*\mathbf{r})\mathbf{r} \mapsto (\mathbf{rl}^*\mathbf{r})\mathbf{rl}$

Dual encoding of action Dec

We now need to encode a metarule having the effect $\mathbf{rl} \mapsto \mathbf{r}$ on G-thread.

This is done by metarule (Dec'_1), described here:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash G_{\mathbf{rl}}, F_{\mathbf{r}}}\ (\text{Ax}) \qquad \dfrac{}{\vdash G_{\mathrm{rr}}, A}\ (\infty)}{\vdash G_{\mathbf{r}}, F_{\mathbf{r}}, A}\ (\mu, \otimes)}{\vdash G, F, A}\ (\mathbf{r}) \qquad (q') \vdash G, F, A}{(p') \vdash G, F, A}\ (\text{Acut})$$

To manipulate the second counter, again we just need to have a (counter) metarule, as shown in metarule (Dec'_2):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash G_{\mathbf{rl}}, F_{\mathbf{r}}}\ (\text{Ax}) \qquad \dfrac{}{\vdash G_{\mathrm{rr}}, A}\ (\infty)}{\vdash G_{\mathbf{r}}, F_{\mathbf{r}}, A}\ (\mu, \otimes)}{\vdash G, F, A}\ (\mathbf{r})}{\vdash G, F, A}\ (\text{counter}) \qquad (q') \vdash G, F, A}{(p') \vdash G, F, A}\ (\text{Acut})$$

Dual encoding of action Test

If $\delta(p) = \mathrm{Test}_1(q_Z, q_P)$, we can define the Test rule in a similar way as in π_M. However, extra care is needed for introduction rules, as they cannot be used as freely as before.

Thus, we first define the metarule ($\mathbf{r}$Xcop) (Fig. 19), with effect $\mathbf{r}X \mapsto \mathbf{r}X\mathbf{r}X$ on a G-thread, for $X \in \{\mathbf{l}, \mathbf{r}\}$.

We can now give the metarule (Test'_1), that uses the first $\mathbf{r}X$ output by ($\mathbf{r}$Xcop) to perform the zero test:

$$\dfrac{\dfrac{(q'_P) \vdash G_{\mathbf{rl}}, F_{\mathbf{rl}}, A \qquad (q'_Z) \vdash G_{\mathbf{rr}}, F_{\mathbf{rr}}, A}{\vdash G_{\mathbf{r}}, F_{\mathbf{r}}, A}\ (\text{exp})}{\vdash G, F, A}\ (\mathbf{r})}{(p') \vdash G, F, A}\ (\mathbf{r}\text{Xcop})$$

If the G-thread enters with constraint prefix $\mathbf{rr}$, this prefix will be copied, the first copy will be read, put in the garbage to be collected later, and the G-thread will enter (q'_Z). Similarly if the constraint started with $\mathbf{rl}$, the G-thread will go to (q'_P) with unchanged stack and garbage $\mathbf{rl}$.

To treat the case of the second counter, i.e. $\delta(p) = \mathrm{Test}_2(q_Z, q_P)$, we just need to describe dual versions of all gadgets from (Test_2) in π_M. We design them so that the effect on the G-thread is exactly the same as the one of their original version on the F-thread.

The only difficulty to keep in mind is that the introduction rules now assume that the constraint stack starts with $\mathbf{r}$. We start with π'_{shift} described Fig. 20, where the pairing between G and F formulas in axioms is recalled in the table:

before:	$\mathbf{ll}$	$\mathbf{lrrl}$	$\mathbf{lrrr}$
after:	$\mathbf{ll}$	$\mathbf{rrll}$	$\mathbf{rrrl}$

For readability, we separated π'_{aux}, described Fig. 21:

As before, the effect of π'_{shift} on the G-thread is $\mathbf{l}^{k+1}T \mapsto \mathbf{l}^k T\mathbf{l}$ with $T \in \{\mathbf{rrr}, \mathbf{rrl}\}$. We continue with the dual ($\mathrm{copy}'_{\mathbf{l}}$) of ($\mathrm{copy}_{\mathbf{l}}$), represented Fig. 22, with action $\begin{cases} \mathbf{l} \mapsto \mathbf{ll} \\ \mathbf{rr} \mapsto \mathbf{rr} \end{cases}$. Notice that we now use the fact that in the present context, a stack starting with $\mathbf{r}$ actually starts with $\mathbf{rr}$, as mentioned when defining the gadget ($\mathrm{copy}_{\mathbf{l}}$).

We turn to π'_{move}, represented Fig. 23, iterating π_{shift} with effect $\mathbf{l}^k T \mapsto \mathbf{l}^k T \mathbf{l}^k$ and $\mathbf{rr} \mapsto \mathbf{rr}$, functioning along the same principle as π_{move}:

The rule (prep′) (Fig. 24) will again allow us to prepare the input for π'_{move}, using the same pairing as before. This rule has effect $(\mathbf{rl}^k\mathbf{r})\mathbf{r}X \mapsto (\mathbf{rl}^k\mathbf{r})\mathbf{r}X\mathbf{rr}X$, with $X \in \{\mathbf{l}, \mathbf{r}\}$. We will note Γ_F for a sequent composed of several non-pertinent occurrences of formula F, for concision. We continue with the metarule (result′), having action $\mathbf{rl}^k T \mapsto \mathbf{l}^k T \mathbf{l}^k T$ on the G-thread, with $T \in \{\mathbf{rrl}, \mathbf{rrr}\}$, and leaving a garbage constraint $\mathbf{r}$ that will be seen later on the way down.

$$\dfrac{\vdash G, F, A}{\vdash G, F, A}\ (\text{result}') \equiv \quad \dfrac{\dfrac{\dfrac{\pi'_{\text{move}}}{\vdash G, F, A} \qquad \vdash G, F, A}{\vdash G_{\mathbf{r}}, F_{\mathbf{r}}, A}\ (\text{Acut})}{\vdash G, F, A}\ (\mathbf{r})}{\vdash G, F, A}\ (\text{prep}')$$

Since the dual form of the ($\mathbf{r}_i$) rule used in (Test_2) needs extra care, we will devise another auxiliary gadget rule ($\mathbf{r}X\mathbf{r}_i$) (Fig. 25), with effect $\mathbf{r}X \mapsto \mathbf{r}X\mathbf{r}$ with $X \in \{\mathbf{l}, \mathbf{r}\}$. This is to restore the stack content after the bit X encoding the test result to its original value starting with $\mathbf{r}$. We can finally build the metarule for Test_2 (Fig. 26), performing the wanted

$$\dfrac{\vdash G, F, A}{\vdash G, F, A}\ (\mathbf{r}\text{Xcop}) \equiv \dfrac{\dfrac{\vdash G_{\mathbf{l}}, \Gamma_F, A}{\vdash G, \Gamma_F, F_{\mathbf{rlrl}}, F_{\mathbf{rrrr}}, A}\ ((\nu, \parr)^4)}{\vdash G, F, A}\ \dfrac{\dfrac{}{G_{\mathbf{rl}}, F_{\mathbf{rlrl}}}\ (\text{Ax}) \quad \dfrac{}{G_{\mathbf{rr}}, F_{\mathbf{rrrr}}}\ (\text{Ax})}{G_{\mathbf{r}}, F_{\mathbf{rlrl}}, F_{\mathbf{rrrr}}}\ (\mu, \otimes)}\ (\infty)\ (\mu, \otimes)$$

Figure 19. The metarule (**r**Xcop)

$$\dfrac{\dfrac{\dfrac{\pi'_{\text{aux}}}{\vdash G_{\mathbf{l}}, F_{\mathbf{ll}}, F_{\mathbf{rrll}}, F_{\mathbf{rrrl}}, A} \quad \dfrac{}{\vdash G_{\mathbf{r}}, F_{\mathbf{lr}}, F_{\mathbf{rl}}, F_{\mathbf{rlr}}, F_{\mathbf{rrr}}, A}\ (\infty)}{\vdash G, F_{\mathbf{ll}}, F_{\mathbf{lr}}, F_{\mathbf{rl}}, F_{\mathbf{rlr}}, F_{\mathbf{rrll}}, F_{\mathbf{rrlr}}, F_{\mathbf{rrrl}}F_{\mathbf{rrrr}}, A,}\ (\mu, \otimes)}{\vdash G, F, A}\ ((\nu, \parr)^6) \quad \dfrac{(\bullet) \vdash G_{\mathbf{l}}, F_{\mathbf{l}}, A \quad \dfrac{}{\vdash G_{\mathbf{r}}, F_{\mathbf{r}}}\ (\text{Ax})}{\vdash G, F, A}\ (\mu, \nu, \parr, \otimes)}{(\bullet) \vdash G, F, A}\ (\text{Acut})$$

Figure 20. The pre-proof π'_{shift}

$$\dfrac{\pi'_{\text{aux}}}{\vdash G_{\mathbf{l}}, F_{\mathbf{ll}}, F_{\mathbf{rrll}}, F_{\mathbf{rrrl}}, A} \equiv \dfrac{\dfrac{}{\vdash G_{\mathbf{ll}}, F_{\mathbf{ll}}}\ (\text{Ax}) \quad \dfrac{\dfrac{}{\vdash G_{\mathbf{lrl}}, A}\ (\infty) \quad \dfrac{\dfrac{}{\vdash G_{\mathbf{lrrl}}, F_{\mathbf{rrll}}}\ (\text{Ax}) \quad \dfrac{}{\vdash G_{\mathbf{lrrr}}, F_{\mathbf{rrrl}}}\ (\text{Ax})}{\vdash G_{\mathbf{lrr}}, F_{\mathbf{rll}}, F_{\mathbf{rrl}}}\ (\mu, \otimes)}{\vdash G_{\mathbf{lr}}, F_{\mathbf{rll}}, F_{\mathbf{rrl}}, A}\ (\mu, \otimes)}{\vdash G_{\mathbf{l}}, F_{\mathbf{ll}}, F_{\mathbf{rrll}}, F_{\mathbf{rrrl}}, A}\ (\mu, \otimes)$$

Figure 21. The pre-proof π'_{aux}

$$\dfrac{\vdash G, F, A}{\vdash G, F, A}\ (\text{copy}'_{\mathbf{l}}) \equiv \dfrac{\dfrac{\dfrac{}{\vdash G_{\mathbf{l}}, F_{\mathbf{ll}}}\ (\text{Ax}) \quad \dfrac{\dfrac{}{\vdash G_{\mathbf{rl}}, F_{\mathbf{lr}}, F_{\mathbf{rl}}, A}\ (\infty) \quad \dfrac{}{\vdash G_{\mathbf{rr}}, F_{\mathbf{rr}}}\ (\text{Ax})}{\vdash G_{\mathbf{r}}, F_{\mathbf{lr}}, F_{\mathbf{rl}}, F_{\mathbf{rr}}, A}\ (\mu, \otimes)}{\vdash G, F_{\mathbf{ll}}, F_{\mathbf{lr}}, F_{\mathbf{rl}}, F_{\mathbf{rr}}, A}\ (\otimes)}{\vdash G, F, A}\ ((\nu, \parr)^2) \quad \vdash G, F, A}{\vdash G, F, A}\ (\text{Acut})$$

Figure 22. The metarule ($\text{copy}'_{\mathbf{l}}$)

test and leaving garbage constraint of the form $\mathbf{rl}^k\mathbf{rr}X$ with $X \in \{\mathbf{l}, \mathbf{r}\}$. After entering ($\text{Test}'_2$) with stack content u, the G-thread will reach node (q'_Z) (resp. (q'_P)) if the second counter value is zero (resp. not zero), with same stack content u.

Dual final state q'_f

As before, when reaching the node (q'_f), we just need bounce on an axiom after evacuating the formula A from the sequent. The same gadget as in π_M can be used:

$$\dfrac{\dfrac{}{\vdash G_{\mathbf{l}}, F_{\mathbf{l}}, A}\ (\infty) \quad \dfrac{}{\vdash G_{\mathbf{r}}, F_{\mathbf{r}}}\ (\text{Ax})}{(q'_f) \vdash G, F, A}\ (\mu, \nu, \parr, \otimes)$$

Correctness of the pre-proof P and conclusion

By construction, if the machine M does not halt, then the F-thread entering π_M will never exit it, so it cannot validate the branch looping through the ($\star$) nodes, and the pre-proof P is not a proof. Conversely, if M halts, then the F-thread will exit π_M with a constraint u encoding the results of the tests as described in Lemma A.38. Moreover, the G-thread t' entering π_R with a single constraint $\mathbf{r}$ will exit on the same node with the same constraints u. This means that the F-thread going through π_M and π_R will exit π_R with a single constraint $\mathbf{r}$. This constraint will be popped on the first unfolding of F, and the second (left) unfolding of F will be on the visible part of the thread. The thread then reaches node ($\star$), and loops back to the root to go through the same path infinitely many times.

Thus the F-thread starting in the root validates the branch containing infinitely many ($\star$) nodes. All other infinite branches of P are validated by non-bouncing thread following formulas A, that can also be originated in the root of P, in the formula B.

Therefore, we showed the following theorem:

$$\frac{}{\vdash G, F, A}\ \pi'_{\text{move}} \equiv \dfrac{\dfrac{\dfrac{\dfrac{\pi'_{\text{shift}}}{\vdash G, F, A} \quad \dfrac{}{(\bigstar') \vdash G, F, A}}{\vdash G_{\mathbf{l}}, F_{\mathbf{l}}, A}\ (\text{Acut}) \quad \dfrac{}{\vdash G_{\mathbf{r}}, F_{\mathbf{r}}}\ (\text{Ax})}{\vdash G, F, A}\ (\mu, \nu, \parr, \otimes)}{(\bigstar') \vdash G, F, A}\ (\text{copy}'_{\mathbf{l}})$$

Figure 23. The pre-proof π'_{move}

$$\frac{\vdash G, F, A}{\vdash G, F, A}\ (\text{prep}') \equiv \dfrac{\dfrac{\dfrac{\dfrac{}{G_{\mathbf{l}}, \Gamma_F, A}\ (\infty) \quad \dfrac{\dfrac{}{\vdash G_{\mathbf{rl}}, F_{\mathbf{rlrrl}}}\ (\text{Ax}) \quad \dfrac{}{\vdash G_{\mathbf{rr}}, F_{\mathbf{rrrrr}}}\ (\text{Ax})}{\vdash G_{\mathbf{r}}, F_{\mathbf{rlrrl}}, F_{\mathbf{rrrrr}}}\ (\mu, \otimes)}{\vdash G, \Gamma_F, F_{\mathbf{rlrrl}}, F_{\mathbf{rrrrr}}, A}\ ((\nu, \parr)^5)}{\vdash G, F, A}\ (\text{counter})}{\vdash G, F, A} \quad \vdash G, F, A}{\vdash G, F, A}\ (\text{Acut})$$

Figure 24. The metarule (prep′)

$$\frac{\vdash G, F, A}{\vdash G, F, A}\ (\text{rXr}_{\mathsf{i}}) \equiv \dfrac{\dfrac{\dfrac{}{\vdash G_{\mathbf{l}}, \Gamma_F, A}\ (\infty) \quad \dfrac{\dfrac{}{G_{\mathbf{rl}}, F_{\mathbf{rlr}}}\ (\text{Ax}) \quad \dfrac{}{G_{\mathbf{rr}}, F_{\mathbf{rrr}}}\ (\text{Ax})}{G_{\mathbf{r}}, F_{\mathbf{rlr}}, F_{\mathbf{rrr}}}\ (\mu, \otimes)}{\vdash G, \Gamma_F, F_{\mathbf{rlr}}, F_{\mathbf{rrr}}, A}\ (\mu, \otimes)}{\vdash G, F, A}\ ((\nu, \parr)^3)$$

Figure 25. The metarule (rXr$_{\mathsf{i}}$)

$$\frac{(q_P) \vdash G, F, A \qquad (q_Z) \vdash G, F, A}{(p) \vdash G, F, A}\ (\text{Test}'_2) \equiv \dfrac{\dfrac{\dfrac{\dfrac{(q_P) \vdash G_{\mathbf{rrl}}, F_{\mathbf{rrl}}, A \qquad (q_Z) \vdash G_{\mathbf{rrr}}, F_{\mathbf{rrr}}, A}{\vdash G_{\mathbf{rr}}, F_{\mathbf{rr}}, A}\ (\text{exp})}{\vdash G_{\mathbf{r}}, F_{\mathbf{r}}, A}\ (\mathbf{r})}{\vdash G_{\mathbf{r}}, F_{\mathbf{r}}, A}\ (\text{rXr}_{\mathsf{i}})}{(\clubsuit') \vdash G_{\mathbf{l}}, F_{\mathbf{l}}, A}\ (\text{exp})}{(\clubsuit') \vdash G, F, A}\ (\text{result}')}{(p) \vdash G, F, A}$$

Figure 26. The metarule (Test′$_2$)

Theorem A.40. *The pre-proof P is a proof if and only if the 2CM M has a run.*

By Theorem A.37, we obtain that deciding whether a circular pre-proof of $\mu\mathsf{MLL}^\omega$ is a proof is undecidable, and more precisely it is Σ^0_1-hard. In the next section, we show that this problem is recursively enumerable, so we obtain Cor. 6.7, stating that deciding validity of a circular pre-proof of $\mu\mathsf{MLL}^\omega$ is Σ^0_1-complete.

Contents