

Université de Paris

École doctorale de sciences mathématiques de Paris centre — ED 386

Institut de recherche en informatique fondamentale

Too complicated to have simple labels

Circular representations of infinite proofs for fixed-points logics: expressiveness and complexity

Par Rémi Nollet

Thèse de doctorat d'informatique fondamentale

Dirigée par Christine Tasson

Et par Alexis Saurin

Présentée et soutenue publiquement le 29 juin 2021

Devant un jury composé de :

Laurent Regnier, professeur des Universités, Université d'Aix-Marseille, rapporteur

Damien Pous, directeur de recherche, C. N. R. S. et É. N. S. de Lyon, rapporteur

Delia Kesner, professeure des Universités, Université de Paris, examinatrice

Bahareh Afshari, *associate professor*, University of Amsterdam et Göteborgs Universitet, examinatrice

Christine Tasson, professeure des Universités, Sorbonne Université, directrice

Alexis Saurin, chargé de recherche, C. N. R. S. et Université de Paris, directeur

This work is licensed under CC BY 4.0. To view a copy of this license, visit
http://creativecommons.org/licenses/by/4.0/

Titre : Représentations circulaires de preuves infinies pour les logiques à points fixes : expressivité et complexité.

Résumé : Dans le cadre des logiques adaptées aux plus petits et plus grands points fixes de formules, les preuves circulaires ont été proposées comme alternative aux principes d'induction et coinduction à invariants explicites. Ces preuves circulaires ont reçu un intérêt grandissant ces dernières années avec le développement simultané de leurs applications et de leur méta-théorie. Les preuves infinies sont maintenant bien implantées dans plusieurs sujets de la théorie de la démonstration tels que les prédicats inductifs à la Martin-Löf, la logique linéaire étendue avec des points fixes, etc.

Dans le cadre des preuves circulaires ou infinies, non bien-fondées, un critère de validité est nécessaire pour rétablir la cohérence logique. Ce critère sert à distinguer, parmi l'ensemble des prépreuves infinies ou circulaires, celles qui sont des preuves valides. Une approche maintenant standard est de considérer qu'une prépreuve est valide si chacune de ses branches infinies est justifiée par un thread qui progresse infiniment souvent.

Cette thèse étudie ces problèmes dans le cadre plus spécifique de MALL étendue avec des points fixes. On sait que, étant donnée une représentation circulaire finie d'une prépreuve non bien-fondée, il est possible de décider en espace polynomial si cette prépreuve est valide vis-à-vis du critère de threads. La première contribution de cette thèse est de démontrer que ce problème de décider le critère de threads pour μMALL^ω est en fait PSPACE-complet. Notre démonstration est fondée sur une étude approfondie des liens entre le critère de threads et le principe de size-change termination, habituellement utilisé pour garantir la terminaison de programmes.

La deuxième contribution de cette thèse est de décrire un nouveau fragment de l'ensemble des représentations circulaires valides de prépreuves, fondé sur un critère de validité plus contraignant. Ce nouveau critère s'appuie sur un étiquetage des formules et des preuves, et sa validité est purement locale. Il permet à ce fragment d'être plus simple à manipuler, tout en restant suffisamment expressif pour contenir l'image des preuves finitaires du système μMALL de David Baelde, qui fonctionnent avec des invariants explicites d'induction et de coinduction. En particulier, il est possible de décider efficacement la validité d'une prépreuve circulaire vis-à-vis de ce nouveau critère, et de calculer efficacement un étiquetage qui certifie cette validité.

De plus, et ceci constitue la troisième contribution de cette thèse, la conjecture de Brotherston-Simpson est vraie pour ce fragment : nous décrivons une méthode qui permet de transformer toute preuve circulaire étiquetée de notre fragment en une preuve finitaire, avec des invariants d'induction et de coinduction explicites. Finalement, nous explorons des manières d'étendre ces résultats de validité et de finitisation à un plus grand fragment, en relâchant les contraintes de l'étiquetage tout en conservant (1) la possibilité de certifier localement la validité des preuves et (2) dans une certaine mesure, la possibilité de finitiser les preuves circulaires ainsi étiquetées.

Mots clefs : Size-change, PSPACE-complet, complexité, calcul des séquents, preuves non bien-fondées, preuves circulaires, induction, coinduction, points fixes, recherche de preuves, logique linéaire, muMALL, finitisation.

Title: Circular representations of infinite proofs for fixed-points logics: expressiveness and complexity.

Abstract: In the context of logics with least and greatest fixed points, circular, i./,e. non wellfounded but regular proofs have been proposed as an alternative to induction and coinduction with explicit invariants. Those circular proofs have received increasing interest in recent years with the simultaneous development of their applications and metatheory: infinitary proof theory is now well-established in several proof-theoretical frameworks such as Martin-Löf's inductive predicates, linear logic with fixed points, etc.

In the setting of non wellfounded proofs, in order to recover logical consistency, it is necessary to consider a validity criterion which differentiates, among all infinite or circular derivation trees, a. k. a. preproofs, those which are logically valid proofs. A standard approach is to consider a preproof to be valid if every infinite branch is supported by an infinitely progressing thread.

This thesis focuses on circular proofs for MALL with fixed points. It is known that given a finite circular representation of a non-wellfounded preproof, one can decide in PSPACE whether this preproof is valid with respect to the thread criterion. The first contribution of this thesis is to prove that the problem of deciding thread-validity for μMALL^ω is in fact PSPACE-complete. Our proof is based on a deeper exploration of the connection between thread-validity and the size-change termination principle, which is usually used to ensure program termination.

The second contribution of this thesis is to describe a new fragment of all valid circular representations of preproofs, based on a stronger validity criterion. This new criterion is based on a labeling of formulas and proofs, whose validity is purely local. This allows this fragment to be easily handled, while being expressive enough to still contain all circular embeddings of Baelde's μMALL finite proofs with inductive and coinductive invariants. In particular, deciding validity and computing a certifying labelling can be done efficiently.

Moreover, and this is the third contribution of this thesis, the Brotherston-Simpson conjecture holds for this fragment: every labeled representation of a circular proof in the fragment is translated into a standard finitary proof. Finally, we explore how to extend these results to a bigger fragment, by relaxing the labeling discipline while retaining (1) the ability to locally certify the validity and (2) to some extent, the ability to finitize circular proofs.

Keywords: Size-change, PSPACE-complete, Complexity, sequent calculus, non-wellfounded proofs, circular proofs, induction, coinduction, fixed points, proof-search, linear logic, muMALL, finitization.

Contents

1 Introduction **9**
- 1.1 Generalities on least and greatest fixed points . . . 9
  - 1.1.1 Basic definitions . . . 9
  - 1.1.2 Proving their existence . . . 9
  - 1.1.3 Expressivity in logic . . . 11
  - 1.1.4 Expressivity in programs . . . 13
  - 1.1.5 Usage in logic . . . 14
- 1.2 Least and greatest fixed points in proof theory . . . 14
  - 1.2.1 Proof theory . . . 15
  - 1.2.2 Usage of fixed points in proof theory . . . 15
  - 1.2.3 Finite proof systems for least and greatest fixed points . . . 16
  - 1.2.4 Infinite proof systems for least and greatest fixed points . . . 17
  - 1.2.5 Finite representations and circular proof systems for least and greatest fixed points . . . 18
- 1.3 In this thesis . . . 20
  - 1.3.1 Fixed points in linear logic . . . 20
  - 1.3.2 Addressed questions, proposed answers . . . 21
  - 1.3.3 Outline of this thesis . . . 23

2 Technical background **27**
- 2.1 Proof theory for least and greatest fixed points . . . 27
  - 2.1.1 Usual proof theory and MALL . . . 27
  - 2.1.2 Finite proof trees: μMALL . . . 34
  - 2.1.3 Infinite proof trees: μMALL^∞ . . . 37
  - 2.1.4 Finite proof trees with back edges: μMALL^ω . . . 39
- 2.2 Complexity theory . . . 49
  - 2.2.1 Generalities on complexity and PSPACE . . . 49
  - 2.2.2 Some PSPACE-complete problems . . . 52
- 2.3 Parity automata . . . 52

3 PSPACE-completeness of the thread criterion **55**
- 3.1 Deciding thread validity in PSPACE . . . 56
- 3.2 PSPACE-completeness . . . 60
  - 3.2.1 Outline of the PSPACE-completeness proof . . . 60
  - 3.2.2 Definition of the reduction . . . 60

3.2.3 Main theorem . . . 62
3.2.4 Generalisation to other systems . . . 69
3.3 Comments on our approach and discussion of related works . . . 71
3.4 Conclusion . . . 72

4 A polynomial sub-criterion **81**
4.1 $\mathcal{L}$-proofs: labelling as validity . . . 83
4.2 Finite representations of circular $\mathcal{L}$-proofs. . . . 85
4.3 Two alternative characterizations of $\mu\text{MALL}^{\supset}$ . . . 89
4.4 Conclusion . . . 96

5 Finitization **99**
5.1 On Brotherston-Simpson's conjecture: finitizing circular proofs . . . 100
5.2 Relaxing the labelling of proofs . . . 103
5.3 Extending finitization . . . 108
5.4 An *ad hoc* finitization of π_∞ . . . 109
5.5 Related works on Chapters 4 and 5 . . . 118
5.6 Conclusion . . . 118

6 Conclusion **121**
6.1 General conclusion . . . 121
6.2 Future works . . . 122

1 Introduction

1.1 Generalities on least and greatest fixed points

1.1.1 Basic definitions

The general subject of this thesis is the study of least and greatest fixed points in logic.

The basic notions of least and greatest fixed points are easily described in the setting of ordered sets.

If $(E, \leqslant)$ is an order and $f\colon E \to E$ is a non-decreasing endofunction, an element $x \in E$ is a *fixed point* of f when $f(x) = x$.

More than general fixed points, what interests us are pre- and post-fixed points. An element $x \in E$ is a *pre-fixed point* of f when $f(x) \leqslant x$, and it is a *post-fixed point* of f when $x \leqslant f(x)$.

Even more precisely, what we want to talk about are least pre-fixed points and greatest post-fixed points. An element $x \in E$ is a *least pre-fixed point* of f when $f(x) \leqslant x$ and $\forall y \in E, f(y) \leqslant y \Rightarrow x \leqslant y$. And x is a greatest post-fixed point of f when $x \leqslant f(x)$ and $\forall y \in E, y \leqslant f(x) \Rightarrow y \leqslant x$. There are usual notations for that: the least pre-fixed point of a non-decreasing function f is denoted, when it exists, by $\mu x f(x)$, and similarly its greatest post-fixed point is denoted by $\nu x f(x)$.

Note that the least pre-fixed point of a non-decreasing function is always a fixed point, and therefore it is its least fixed point.

1.1.2 Proving their existence

The main theorems on the existence of least and greatest fixed points are the following:

Theorem 1. *Let $(E, \leqslant)$ be an ordered set.*

1. *If*

$$\text{every well-ordered subset of } E \text{ has a least upper bound} \tag{1.1}$$

then

$$\text{every non-decreasing endofunction on } E \text{ has a least pre-fixed point} \tag{1.2}$$

2. *Conversely, if*

$$\text{every non-decreasing endofunction on } E \text{ has a least fixed point} \tag{1.3}$$

then

$$E \text{ has a minimun and every directed subset of } E \text{ has a least upper bound} \tag{1.4}$$

in which we recall that a subset A of E is said to be directed when every finite subset of A has an upper bound in A.

Proof elements and references.

- First remark that, as already mentionned above, a least pre-fixed point is always a fixed point and, therefore, (1.2) is stronger than (1.3). Similarly, every well-ordered subset of an order is either empty or directed. Hence (1.4) is stronger than (1.1). That is why the two parts 1 and 2 of that theorem are indeed converse to each other.

- The direction 1 seems to be a folklore theorem which is difficult to trace back to a unique original author. Yet the first version of it seems to be that of Abian and Brown [1961], although it is very close to a similar theorem by Bourbaki [1949]. See [Lassez, Nguyen, and Sonenberg, 1982, Section 4] for a bibliographic study of its origins. See Davey and Priestley [2002, Exercise 8.19, p. 198] for a quick proof using ordinal induction.

- The direction 2 is more recent and has been proved by Markowsky [1976].

To the reader interested by those questions, we recommend the book by Davey and Priestley [2002, in particular ch. 8, p. 175, and the bibliographic discussion p. 285], which contains more details and references about fixed-point theorems. □

That theorem characterizes, in a sense, the preorders on which it makes sense to study the least fixed points of non-decreasing endofunctions. And it says that in these preorders,

those least fixed points are in fact least pre-fixed points. Those orders are called complete partial orders, or CPO.

Because of that theorem, we will commonly say "least fixed point" when talking about least pre-fixed points, and similarly for greatest fixed points.

1.1.3 Expressivity in logic

We will now motivate the will to add a connective to logic that has the behavior of forming a least fixed point by a first example, coming from temporal logic. We will see that least pre-fixed points and greatest post-fixed points give the possibility to define new formulas, and therefore add expressivity to the language. Such ideas are the basis of modal μ-calculus [Kozen, 1983].

Let us consider a logic like LTL [Pnueli, 1977], in which the truth of a proposition A may depend on an instant $t \in \mathbf{N}$. The boolean denotation of a formula A is a $[\![A]\!] \colon \mathbf{N} \to \{0,1\}$ and if $t \in \mathbf{N}$, we denote by $[\![A]\!]_t \in \{0,1\}$ the truth value of A at t. As an alternative, we may write $t \vDash A$ for $[\![A]\!]_t = 1$. We use one logical connective and two temporal connectives. For each of them, we give two equivalent definitions:

"or"	$[\![A \vee B]\!]_t = \max\Big([\![A]\!]_t, [\![B]\!]_t\Big)$	$t \vDash A \vee B \;\Leftrightarrow\; t \vDash A$ or $t \vDash B$
"next"	$[\![\bigcirc A]\!]_t = [\![A]\!]_{t+1}$	$t \vDash \bigcirc A \;\Leftrightarrow\; t+1 \vDash A$
"eventually"	$[\![\Diamond A]\!]_t = \max\Big([\![A]\!]_s \mid s \geqslant t\Big)$	$t \vDash \Diamond A \;\Leftrightarrow\; \exists s \geqslant t, s \vDash A$

We abuse the notation $\vDash$ by using $A \vDash B$ to denote the fact that $\forall t \in \mathbf{N}, [\![A]\!]_t \leqslant [\![B]\!]_t$, that is, equivalently: $\forall t \in \mathbf{N}$, if $t \vDash A$ then $t \vDash B$. Now we say that

Proposition 1. *For every formulas A and B, the following are true:*

$$A \vee \bigcirc\Diamond A \vDash \Diamond A \qquad (1.5) \qquad\qquad \frac{A \vee \bigcirc B \vDash B}{\Diamond A \vDash B} \qquad (1.6)$$

Proof.

1. Proof of (1.5)

Assume that $t \in \mathbf{N}$ and $t \vDash A \vee \bigcirc\Diamond A$. There are two cases:

1.1. $t \vDash A$

In that case $t \vDash \Diamond A$.

1.2. $t \vDash \bigcirc\Diamond A$

In that case $t+1 \vDash \Diamond A$, that is $\exists s \geqslant t+1, s \vDash A$, so $t \vDash \Diamond A$.

Hence, in any case, $t \vDash \Diamond A$.

2. Proof of (1.6)

Let us assume that

$$A \vee \bigcirc B \vDash B \tag{1.7}$$

and show that $\Diamond A \vDash B$.

2.1. We first prove the following lemma: $\forall u \in \mathbf{N}, \forall t \in \mathbf{N},\ t+u \vDash A \ \Rightarrow\ t \vDash B$.

Proof: By induction on u.

2.1.1. Case $u = 0$:

In that case we know that $t \vDash A$, hence $t \vDash A \vee \bigcirc B$. By assumption (1.7) we get $t \vDash B$.

2.1.2. Case $u = v+1$:

In that case we know that $t+1+v \vDash A$. By induction hypothesis on v, we get $\forall t' \in \mathbf{N}, t'+v \vDash A \ \Rightarrow\ t' \vDash B$, which we apply to $t' = t+1$ to get $t+1 \vDash B$, that is $t \vDash \bigcirc B$. Hence $t \vDash A \vee \bigcirc B$. By assumption (1.7) we deduce $t \vDash B$.

2.2. From this, we immediately deduce the following:

$$\forall s \in \mathbf{N},\ s \vDash A \ \Rightarrow\ \forall t \leqslant s,\ t \vDash B \tag{1.8}$$

2.3. We now prove that $\Diamond A \vDash B$.

Assume a $t \in \mathbf{N}$ and $t \vDash \Diamond A$, that is assume a $s \in \mathbf{N}$, $s \geqslant t$ and $s \vDash A$. It remains to show that $t \vDash B$. This is exactly given by (1.8) above.

□

This relates to our previous description of least/greatest pre-/post-fixed points in the following way. If we consider $\vDash$ as our order, we just proved that $\Diamond A$ is a least pre-fixed point of $F\colon X \mapsto A \vee \bigcirc X$, because we proved that $F(\Diamond A) \leqslant \Diamond A$ and $\forall B, F(B) \leqslant B \ \Rightarrow\ \Diamond A \leqslant B$, with $\leqslant$ being $\vDash$.

That means that if we are allowed to use least pre-fixed points in the construction of the formulas, we do not need $\Diamond$ to be given as a primitive connective of the logic; we could *define* it as $\Diamond A \coloneqq \mu X(A \vee \bigcirc X)$. Dually, you can check that the "always" operator, denoted by $\Box A$ and defined equivalently by $[\![\Box A]\!]_t = \min\Big\{[\![A]\!]_s \mid s \geqslant t\Big\}$ or $t \vDash \Box A \ \Leftrightarrow\ \forall s \geqslant t, s \vDash A$ could alternatively be defined as the greatest post-fixed point $\Box A \coloneqq \nu X(A \wedge \bigcirc A)$.

1.1.4 Expressivity in programs

Least pre-fixed points and greatest post-fixed points can also be used to model inductive and coinductive datatypes in programming languages. We will try to show that on a second example.

In this example, we will use an OCaml-like syntax, although we are not writing strictly legit OCaml. We are trying to convey a general intuition, which probably apply *mutatis mutandis* to any programming language in which one can define inductive algebraic datatypes.

Let us consider the type of integer lists, which can be described as:

```
type int list = [] | ( :: ) of int * int list
```

And let us consider the following type `t`, parameterized over a type `'a`:

```
type 'a t = Nil | Cons of int * 'a
```

Remark that the type `t` is conceptually simpler than `int list`, because it is not a recursive type. If you are familiar with algebraic datatypes, you may see that this type could be written as

```
'a t = 1 + int * 'a
```

where we use `1` to denote the singleton type:

```
type 1 = ()
```

Now we can define two functions

```
inj  : int list t -> int list
fold : ('a t -> 'a) -> int list -> 'a
```

as follows:

```
let inj = function
  | Nil          -> []
  | Cons (x, xs) -> x :: xs

let rec fold f = function
  | []      -> f Nil
  | x :: xs -> f (Cons (x, fold f xs))
```

❇

In this example, for every two types α and β, let us say that $\alpha \leqslant \beta$ when the type α `->` β is inhabited. Then if we take $F(\alpha)$ to be the type α `t`, we have $F($`int list`$) \leqslant$ `int list` because of `inj`. And each time we have a type α such that $F(\alpha) \leqslant \alpha$, we also have `int list` $\leqslant \alpha$ thanks to `fold`. So it makes sense to say that `int list` is a least pre-fixed point of `t`.

Another way of saying it is that `int list` $= \mu\alpha($`1 + int * `$\alpha)$. And the general idea is that inductive datatypes correspond to least pre-fixed points of functors. And, dually, coinductive datatypes correspond to greatest post-fixed points of functors. For instance, we could define a type of integer streams as `int stream` $= \nu\alpha($`int * `$\alpha)$.

Remark 1. Actually the previous example is not totally complete. That example characterises `int list` as the least pre-fixed point of `t`, but it happens that the singleton type `1` could fit this description as well. In fact, the description of least and greatest fixed points in terms of orders or preorders does not give enough precision. To have a complete characterisation of `int list` as a least fixed point, we would need to talk about the computational behaviour of the functions, and we would need to shift from the setting of orders and preorders to the setting of categories, which is out of the scope of this introduction.

1.1.5 Usage in logic

There has been a lot of logics allowing the use of least and greatest fixed-point constructions in their formulas. Let us simply mention here some of the most noticeable. de Bakker and de Roever [1972] define a logic with least fixed points to prove properties of recursively defined programs. Aczel [1977] study the extension of recursion theory with inductive definitions. Aho and Ullman [1979] study the relational calculus, which may be understood as first-order logic, and suggest extending it by adding a least fixed-point operator, in order to make it more expressive. Gurevich and Shelah [1986] study the expressive power of different extensions of first-order logic with fixed point induction. Dawar and Gurevich [2002] study the expressive power of fixed point logics.

1.2 Least and greatest fixed points in proof theory

Least and greatest fixed point of formulas may be seen as a way to formulate definitions by induction, or by coinduction. We will study those least and greatest fixed points from the

point of view of proof theory. We believe that, through the proof-program correspondence, the analysis we make on proofs could also be applied to programs. The advantage of proofs over programs for our study is that they provide a system which makes things visible, decomposing things in simple elements: subformula property, cut-elimination, formulas occurrences, ...

1.2.1 Proof theory

We do here a short recall of a few elements that made the success of modern proof theory and sequent calculus.

In usual sequent calculus, a proof is defined to be a finite proof tree.

The subformula property says that if you can prove a given conclusion then you can do it by using only subformulas of this conclusion. This is a fundamental property, saying, intuitively, that if I ask you a question ("Is this formula true ?"), you can answer it using only the concepts that are already present in the question, you do not need to introduce any new concept. You may still do it, because it may ease the task, but you do not need it.

The cut rule is the rule that allows you to do some deductive reasoning. It is a generalisation of two logical principles, *modus ponens*, which says that from a proof of A and a proof of $A \Rightarrow B$, you can build a proof of B, and the transitivity of implication, which says that from a proof of $A \Rightarrow B$ and a proof of $B \Rightarrow C$ you can build a proof of $A \Rightarrow C$. One of the ideas that made sequent calculus so successful is that all inferences preserve the subformula property, except for the cut rule. When following this discipline, the subformula property amounts to saying that we can eliminate cuts from any proof.

1.2.2 Usage of fixed points in proof theory

Various logical settings have been introduced to reason about inductive and coinductive statements, both at the level of the logical languages modelling (co)induction (Martin Löf's inductive predicates vs. fixed-point logics, that is μ-calculi) and at the level of the proof-theoretical framework considered (finite proofs with (co)induction *à la* Park [1969] vs. infinite proofs with fixed-point/inductive predicate unfoldings) [Brotherston, 2006, Brotherston and Simpson, 2007, 2011, Baelde and Miller, 2007, Baelde, 2009, 2012].

Moreover, such proof systems have been considered over classical logic [Brotherston, 2006, Brotherston and Simpson, 2011], intuitionistic logic [Clairambault, 2009], linear-time or branching-time temporal logic [Kozen, 1983, Kaivola, 1995b, Walukiewicz, 1993, 1995,

Dax, Hofmann, and Lange, 2006, Doumane, 2017b, Doumane, Baelde, Hirschi, and Saurin, 2016] or linear logic [Santocanale, 2002, Fortier and Santocanale, 2013, Baelde and Miller, 2007, Baelde, Doumane, and Saurin, 2016, Doumane, 2017b].

In all those proof systems, the treatment of inductive and coinductive reasoning brings some highly complex proof objects.

1.2.3 Finite proof systems for least and greatest fixed points

The first systems designed to reason about fixed points of formulas were systems with finite proofs, in which induction and coinduction principles are used in order to provide a finite proof theory for reasoning on formulas with least or greatest fixed points [Kozen, 1981, 1983, Baelde, 2012].

But finite proof systems for least and greatest fixed points have some drawbacks.

- To introduce a greatest fixed point formula or to eliminate a least fixed point formula, you have to use a rule of induction or coinduction, and you have to provide an explicit invariant, which has to be a formula of the same system. This is a restriction because the language of formulas may not be expressive enough to express all the invariants you may need to prove formulas that should intuitively be true, as was shown, for instance, by Berardi and Tatsuta [2017b, 2019] in the context of first-order classical logic with inductive definitions.

- It breaks the subformula property.

- Here, in the finitary setting for least and greatest fixed points of formulas, the failure of the subformula property can therefore be restated by saying that we cannot eliminate all cuts. Nevertheless, it makes sense to define a procedure of "cut elimination" for this setting. This procedure will not eliminate all cuts, but it will simplify them and eliminate some of them and it is still a terminating procedure, which leaves the proof in a normal form in which the only remaining cuts are of a very specific form, and are cuts that cannot be reduced anymore. The point is that in this setting, besides the fact that we cannot completely eliminate cuts, this procedure of cut elimination turns out to be quite complex.

For instance, in proof systems using (co)induction rules *à la* Park, the rules allowing to derive a coinductive property (or dually to use an inductive hypothesis) have a complex inference of the form of Figure 1.1 (when presented in the setting of fixed-point logic – here we follow the one-sided sequent tradition of MALL that we will adopt in the rest of the thesis).

$$\frac{\vdash \Gamma, S \quad \vdash S^{\perp}, F[S/X]}{\vdash \Gamma, \nu X F}\,(\nu_{\text{inv}})$$

Figure 1.1. Coinduction rule *à la* Park

It is not only difficult to figure out intuitively what is the meaning of this inference, but it is also problematic for at least two additional and more technical reasons: (i) it is hiding a cut rule that *cannot* be eliminated, which is problematic for extending the proof-program correspondence to fixed-points logics, and (ii) it breaks the subformula property, which is problematic for proof search: at each coinduction rule, one has to guess an invariant (in the same way as one has to guess an appropriate induction hypothesis in usual mathematical proofs).

1.2.4 Infinite proof systems for least and greatest fixed points

Infinite proofs have been introduced to reason on (least and greatest) fixed points of formulas.

For all these reasons, infinite (non-wellfounded) proofs, which are infinite proofs satisfying a validity criterion, have been proposed, in recent years, as an alternative to induction and coinduction with explicit invariants [Brotherston, 2006, Brotherston and Simpson, 2007, 2011]. By replacing the coinduction rule with simple fixed-point unfoldings and allowing for non-wellfounded branches, those proof systems address the problem of the subformula property for the cut-free systems. The cut-elimination dynamics for inductive-coinductive rules is also much simpler. In particular, Baelde, Doumane, and Saurin [2016], inspired notably by Dax, Hofmann, and Lange [2006], proposed a very successful system of infinite proofs, called μMALL$^{\infty}$, to reason on least and greatest fixed points of formulas. Actually, this is for the propositional setting, but such ideas already existed to reason on inductive and coinductive predicates.

Infinite, non well-founded, proofs present the advantage over explicit induction or coinduction to offer a framework in which it is possible to recover the good structural properties of sequent calculus, such as cut-elimination, subformula property and focusing, making them a more suitable tool to automated proof search. Indeed, cut-elimination and focusing have recently been extended to non well-founded proofs for μMALL by Baelde, Doumane, and Saurin [2016], Doumane [2017b].

Now the problem is that if we take this approach too naively, by simply allowing proof trees to be infinite, then two bad things happen:

1. the system becomes inconsistent, meaning that every formula becomes provable;

$$\dfrac{\dfrac{\dfrac{\vdots}{\vdash \mu X X, \Gamma}\,(\mu)}{\vdash \mu X X, \Gamma}\,(\mu) \quad \dfrac{\dfrac{\vdots}{\vdash \nu X X, \Gamma}\,(\nu)}{\vdash \nu X X, \Gamma}\,(\nu)}{\vdash \Gamma}\,(\text{cut})$$

Figure 1.2

2. the cut-elimination procedure becomes a non-terminating, non-converging one.

The price to pay is that the consistency of the logical system is broken and that a validity criterion has to be added in order to ensure consistency.

More precisely, in those proof systems when considering all possible infinite, non-wellfounded derivations, or preproofs, it is straightforward to derive any sequent Γ (see Figure 1.2).

Such preproofs are therefore unsound and one needs to impose a validity criterion to distinguish, among all preproofs, those which are logically valid proofs from the unsound ones.

A solution to that is to say that all these infinite proof trees are only preproofs, and that only some of these preproofs are valid proofs. This means that we need a criterion to distinguish valid (pre)proofs from invalid preproofs, and to reflect the inductive and coinductive nature of our fixed-point connectives.

A standard approach [Brotherston, 2006, Brotherston and Simpson, 2007, 2011, Santocanale, 2002, Baelde, Doumane, and Saurin, 2016] considers a preproof to be valid if every infinite branch is supported by an infinitely progressing thread. This is called the thread criterion.

However, doing so, the logical correctness of circular proofs becomes a non-local property, much in the spirit of proof nets correctness criteria [Girard, 1987, Danos and Regnier, 1989].

1.2.5 Finite representations and circular proof systems for least and greatest fixed points

The problem with infinite proofs, on the other hand, is mainly that they are infinite, which has two major drawbacks:

1. The first one is epistemic: we would like a proof to be a finite object, which I can communicate to you in finite time, and which you can check in finite time. The fact that a proof may be an infinite object means that I may not be able to give you a proof, but I will in fact give you the finite description of an infinite proof, formulated in the meta-theory.

2. The second one is practical: if we want our system to be used in an automated prover or in a proof assistant, we need those proofs to be finitely representable. It means that whatever system of representation you chose, you will not be able to represent all infinite proofs, because there is an uncountable number of infinite proofs and your system of finite representations will only be able to represent a countable number of them.

From that it may seem that going from finite to infinite proofs was a mistake. But infinite proofs give great insights on least and greatest fixed points of formulas, on their different proof systems and on the dynamic of cut-elimination in those systems, including the finitary ones. In particular they provide us with a new way to understand the finitary systems. Each finitary proof can be translated into an infinitary one. This means that finitary proofs may be seen as one particular way to represent some of those infinitary proofs. So now we may transform the question of designing a finitary proof system for least and greatest fixed points of formulas into the question of identifying suitable fragments of the infinitary system, which have some nice-behaved representations.

Among those non-wellfounded proofs, circular proofs, that have infinite but regular derivation trees, have attracted a lot of attention for retaining the simplicity of the inferences of non-wellfounded proof systems but being amenable to a simple finite representation making it possible to have an algorithmic treatment of those proof objects.

In this context, a very natural way of representing some infinite proofs is to use finite proof trees with back edges. This means that instead of constructing explicitely an infinite branch, we are allowed to stop at some point and point out some previous step of the construction and say: from there, we start again at that point. Such a cyclic representation has a canonical unfolding into an infinite proof tree. This system of representations allows us to represent some of the infinite preproofs. Those preproofs that can be represented by such a proof tree with back edges are called circular preproofs. Those representations are called circular representations.

Despite the need for a validity condition, circular, *i. e.* non-wellfounded but regular proofs have received increasing interest in recent years with the simultaneous development of their applications and meta-theory: infinitary proof theory is now well-established in several proof-theoretical frameworks such as Martin Löf's inductive predicates, linear logic with fixed points, *etc.*

1.3 In this thesis

Statement of the contribution On the question of how much time and space it takes to check the thread criterion on a circular representation of an infinite preproof, it was known how to do it in PSPACE, and we prove in this thesis that this problem is PSPACE-complete, which means that we cannot do substantially better.

As we would like to be able to ckeck our proofs in polynomial time, we provide a new validity criterion, which is stronger than the thread criterion while accepting at least the translations of finitary proofs, and which can be checked in quadratic time.

We also would like a proof system in which there is no global criterion to check, in which a proof is correct as soon as each inference used in it is correct. This is made possible. We provide such a proof system, obtained by adding some labelling to the circular representations accepted by my new quadratic criterion.

Finally, this new proof system proves at least as much propositions than the finitary setting and it is legit to ask how much more it does prove exactly. The answer is: No more. This result may look disappointing but what it says is that every circular representation of an infinite preproof that is valid for my quadratic criterion and every proof in my labelled system can be turned into a finitary proof with the same conclusion. And we provide an effective method to do it. In other words, we provide a method to synthetize some induction and coinduction invariants, finitize some circular proofs and partially answer the Brotherston-Simpson conjecture.

1.3.1 Fixed points in linear logic

This thesis focuses on circular proofs for MALL with fixed points. Least and greatest fixed point were already well-established in linear logic [Baelde and Miller, 2007, Baelde, 2012, Baelde, Doumane, and Saurin, 2016, Doumane, 2017b] and we build on this basis to contribute to several directions in the field of circular proofs:

1. the relationship between finite and circular proofs (at the level of provability and at the level of proofs themselves) and

2. the certification of circular proofs, that is the production of fast and/or small pieces of evidence to support validity of a circular preproof.

1.3.2 Addressed questions, proposed answers

Proof-checking Among all circular preproofs, some are valid proofs and some are not. The thread criterion gives a non-ambiguous mathematical definition of whether a circular representation represents a valid (pre)proof or an invalid preproof. The first interesting question is whether there exist a method to tell whether the circular representation of a preproof is valid or not. In other words: is the thread criterion on circular preproofs decidable? Given a finite circular representation of a non-wellfounded preproof, can one decide whether this preproof is valid with respect to the thread criterion?

The answer to that is yes: there is an algorithm which is able to tell the difference between a valid circular representation of a preproof and an invalid one. In fact several such algorithms are known.

The next question would be: how difficult is this problem? What is its computational complexity? When we started to look at this problem, the state of the art was the following: all known algorithms for this problem ran in exponential time and ran or could be made to run un polynomial space. But no subexponential algorithm was known, there was no lower bound on its complexity and the exact complexity of checking the thread criterion was still unknown. Recall that $\mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PSPACE} \subseteq \mathrm{EXP}$. The first contribution of this thesis is to prove that this problem is in fact PSPACE-complete. This implies in particular that this problem is probably not in NP and that there is probably no subexponential algorithm to solve it.

Our proof is based on a deeper exploration of the connection between thread-validity and the size-change termination principle [Lee, Jones, and Ben-Amram, 2001], which is usually used to ensure program termination.

Circular proofs have already proved useful in implementing efficient automatic provers, *e. g.* the Cyclist prover [Gorogiannis and Rowe, 2014]. The complexity avoided in the search, thanks to the fact that we need not to guess invariants, is counterbalanced by the complexity of the validity criterion at the time of proof checking. It seems however that this approach gives good performances in practice.

A new validity criterion Among all representations of valid circular proofs, a new fragment is described in Chapter 4, based on a stronger validity criterion. This new criterion is based on a labelling of formulas and proofs, whose validity is purely local.

This allows this fragment to be easily handled, while being expressive enough to still contain all circular embeddings of Baelde's μMALL finite proofs with (co)inductive invariants: in particular deciding validity and computing a certifying labelling can be done efficiently.

Finitization Comparing finite and infinite proofs is very natural. Informally, it amounts to considering the relative strength of inductive reasoning versus infinite descent: while infinite descent is a very old form of mathematical reasoning which appeared already in Euclid's *Elements* and was systematically investigated by Fermat, making precise its relationship with mathematical induction is still an open question for many proof formalisms. Their equivalence is known as the Brotherston–Simpson conjecture. While it is fairly straightforward to check that infinite descent (circular proofs) proves at least as many statements as inductive reasoning, the converse is complex and remains largely open. A few years ago, Simpson [2017], on the one hand, and Berardi and Tatsuta [2017b,a], on the other hand, made progress on this question but only in the framework of Martin Löf's inductive definitions, not in the setting of μ-calculi circular proofs in which invariant extraction is highly complex and known only for some fragments.

We will show that the Brotherston-Simpson conjecture holds for the fragment we present: every labelled representation of a circular proof in the fragment is translated into a standard finitary proof.

Propositions of extensions Finally we explore how to extend these results to a bigger fragment, by relaxing the labelling discipline while retaining (i) the ability to locally certify the validity and (ii) to some extent, the ability to finitize circular proofs.

A complex example We conclude this introduction by considering a typical example of a circular proof with a complex validating thread structure: while this infinite proof has a regular derivation tree, its branches and threads have a complex geometry. The circular proof of Figure 1.3 derives the sequent $\vdash F, G, H, I, J$ where

$$F = \mu X((X \parr G) \,\&\, (X \parr H)) \qquad G = \nu X(X \oplus \bot) \qquad H = \nu X(\bot \oplus X)$$

$$I = \mu Z((Z \parr J) \oplus \bot) \qquad J = \mu X((K \parr X) \oplus \bot) \qquad K = \nu Y\, \mu Z((Z \parr \mu X(Y \parr X) \oplus \bot) \oplus \bot)$$

This example of a circular derivation happens to be valid (it is a μMALL$^\omega$ proof) but the description of its validating threads is quite complex. Indeed, each infinite branch β is validated by exactly one thread (see next section for detailed definitions) going through either G, H or K depending on the shape of the branch *at the limit* (infinite branches of this derivations can be described as ω-words on $A = \{l, r\}$ depending on whether the left or right back edge is taken):
(i) if β *ultimately* follows always the left cycle ($A^\star \cdot l^\omega$), the unfolding of H validates β;
(ii) if β *ultimately* follows always the right cycle ($A^\star \cdot r^\omega$), the unfolding of G validates β;
(iii) if β *endlessly switches* between left and right cycles ($A^\star \cdot (r^+ \cdot l^+)^\omega$), K validates β.
The description of the thread validating this proof is thus complex. This is reflected in the difficulty to provide a local way to validate this proof and in the lack of a general

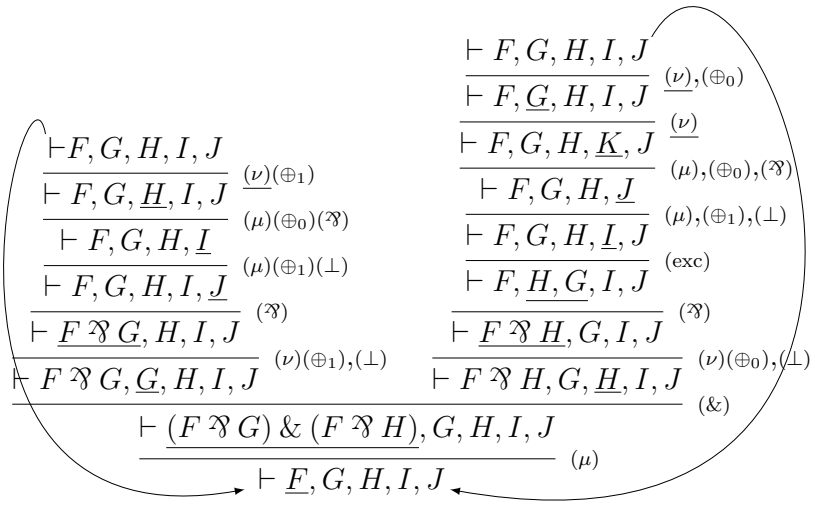


Figure 1.3. Proof π_∞

method for finitizing this into a μMALL proof: to our knowledge, the usual finitization methods (working only for fragments of μMALL^ω circular proofs) do not apply here.

1.3.3 Outline of this thesis

The subject of this thesis is the study of finite circular representations of infinite proofs and preproofs.

It consists of 6 chapters:

- Chapter 1, this chapter, is an informal introduction to the subject of the thesis.
- Chapter 2 covers the technical background needed to develop the results of the thesis.
- Chapters 3, 4 and 5 are the technical chapters, presenting the main results and contributions of this thesis.
- Chapter 6 is a conclusion chapter.

In Chapter 2, we provide the necessary technical background to our work. Section 2.1 covers infinitary and circular proof theory of multiplicative additive linear logic with least and greatest fixed points. We recall the formulas and rules of MALL (Section 2.1.1), how they can and have been extended with least and greatest fixed point to obtain μMALL (Section 2.1.2), how infinite μMALL^∞ proofs have been proposed as an alternative, in which a global validity criterion is needed (Section 2.1.3, and the standard fragment μMALL^ω of regular infinite proofs, representables by proof trees with back-edges, and

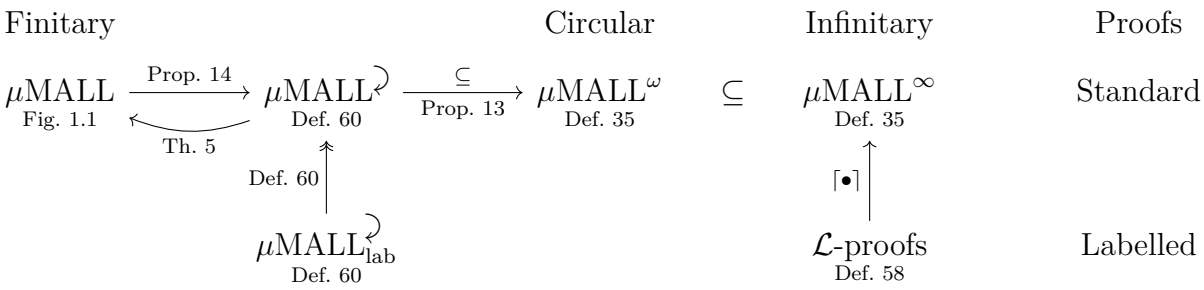


Figure 1.4. Relations between the different systems used in this thesis.

which are the true subject of this thesis (Section 2.1.4). We also recall some basic elements of the theories of algorithmic complexity (Section 2.2) and of ω-automata (Section 2.3).

Chapter 3 is concerned with the algorithmic complexity of the thread criterion. We start by recalling in Section 3.1 that the thread criterion is effectively decidable in PSPACE. The main section of the first technical chapter is Section 3.2, in which we show the PSPACE-completeness of the thread criterion for $\mu\mathrm{MALL}^\omega$, in Theorem 4. Section 3.3 is devoted to a discussion of our approach and a comparison with related works. We conclude this chapter in Section 3.4.

Chapter 4 studies an approach to circular proofs based on labellings of greatest fixed points. We first motivate in Section 4.1 such labellings as an alternative way to express the validating threads. Then, in Section 4.2 we introduce finite representations of preproofs and use such labellings in order to locally certify their validity. Finally, in Section 4.3, we turn to alternative characterizations of those circular proofs which can be labelled. The fragment of labellable proofs, while quite constrained (for instance, it does not include the example of Figure 1.3), is already enough to capture the circular proofs obtained by translation of μMALL proofs.

In Chapter 5 and Section 5.1, we address the converse: for any labelled derivation tree with back edges, we provide a corresponding μMALL proof by generating a (co)inductive invariant based on an inspection of the labelling structure. Therefore, we answer the Brotherston–Simpson conjecture in a restricted fragment. In Section 5.2, we introduce a more permissive labelling strategy that allows to label more proofs (in particular by allowing to loop not only on (ν) rules but on any rule) and that still ensures validity of the labellable derivations. For this relaxed labelling, we label the example of Figure 1.3 and show how to finitize it by adapting the method of Section 5.1. Nevertheless, there is not yet a general method applicable to the complete extended labelling fragment. We end this chapter by a discussion, in Section 5.5, of related works to both Chapters 4 and 5. Relations between the various systems considered in these two chapters are summarized in Figure 1.4.

[1] ❊

2 Technical background

2.1 Proof theory for least and greatest fixed points

2.1.1 Usual proof theory and MALL

Logic is generally built around two main concepts: formulas and proofs.

Formulas or *propositions*, model some statements that you may say, and which may, intuitively, be true or false such as "The cat is black" or "Earth is flat". But logic and proof theory are more interested in understanding the nature of the logical connectives and their dynamic than in what happens to be true in our world. That is why a formal language of formulas is generally built out of some propositional variables and some connectives. For instance a language of formulas for the classical logic LK may be defined by the following grammar:

$$A, B ::= X \mid \overline{X} \mid A \wedge B \mid A \vee B \mid \top \mid \bot$$

where X ranges over a given set of propositional variables, which we denote by $X, Y, Z, \ldots$

$\wedge$ denotes conjunction, "and", while $\vee$ denotes disjunction, "or". The constants $\top$ and $\bot$ represent their respective neutral elements, that is respectively a true statement and a false statement. The variables $X, Y, Z, \ldots$ represent some undetermined statements. That means that, for instance, $(\overline{X} \vee Y) \wedge (\overline{Y} \vee Z)$ is a formula of LK, which you may read as "not X or Y, and not Y or Z".

The main logic we will use to apply our methods in this thesis is the logic MALL, composed of the Multiplicative and Additive fragments of Linear Logic. Linear logic was designed by Girard [1987] in a successful attempt to decompose the well-known connectives of classical and intuitionistic logics into more elementary components. For a detailed introduction to linear logic, see [Girard, 1987], [Girard, Taylor, and Lafont, 1989] or [Girard, 2011].

The formulas of MALL are defined by the following grammar:

Definition 1 (MALL formulas)**.**

$$A, B ::= X \mid \overline{X} \mid A \otimes B \mid A \parr B \mid \mathbf{1} \mid \bot \mid A \oplus B \mid A \,\&\, B \mid \mathbf{0} \mid \top$$

where X ranges over a given set of propositional variables.

The set of all MALL formulas is denoted by $\mathbf{Fml}_{\mathrm{MALL}}$.

$\otimes$ and $\&$ are conjunctions of different natures and $\mathbf{1}$ and $\top$ are their respective neutral elements. $\parr$ and $\oplus$ are disjunctions of different natures and $\bot$ and $\mathbf{0}$ are their respective neutral elements.

Remark 2 (Precedence rules). The usage is that $\otimes$ and $\parr$, the "multiplicative" connectives, have a higher precedence than $\oplus$ and $\&$, the "additive" connectives. For instance $\overline{X} \parr \overline{Y} \,\&\, \overline{X} \parr \overline{Y}$ should be read as $(\overline{X} \parr \overline{Y}) \,\&\, (\overline{X} \parr \overline{Y})$.

With such a syntax of formulas come a syntax for the positions of subformulas inside formulas. To that effect we use the following alphabet

Let $A_{\mathbf{Fml}}$ be the following alphabet:

$$\mathcal{P}_{\mathrm{MALL}} = \{\otimes_0, \otimes_1, \parr_0, \parr_1, \oplus_0, \oplus_1, \&_0, \&_1\}$$

We denote by $\mathcal{P}^*_{\mathrm{MALL}}$ the free monoid of finite words on $\mathcal{P}_{\mathrm{MALL}}$. Those words are used to denote the position of a subformula in a formula.

Definition 2 (Positions of subformulas in a formula)**.** The set of positions of subformulas of a given formula A is defined by induction on A:

$$\mathsf{SubPos}(X) = \{\epsilon\} \qquad \mathsf{SubPos}(\overline{X}) = \{\epsilon\} \qquad \mathsf{SubPos}(\mathbf{1}) = \{\epsilon\} \qquad \mathsf{SubPos}(\mathbf{0}) = \{\epsilon\}$$

$$\mathsf{SubPos}(\bot) = \{\epsilon\} \qquad \mathsf{SubPos}(\top) = \{\epsilon\}$$

$$\mathsf{SubPos}(A \otimes B) = \{\epsilon\} \;\cup\; \otimes_0 \cdot \mathsf{SubPos}(A) \;\cup\; \otimes_1 \cdot \mathsf{SubPos}(B)$$

$$\mathsf{SubPos}(A \parr B) = \{\epsilon\} \;\cup\; \parr_0 \cdot \mathsf{SubPos}(A) \;\cup\; \parr_1 \cdot \mathsf{SubPos}(B)$$

$$\mathsf{SubPos}(A \oplus B) = \{\epsilon\} \;\cup\; \oplus_0 \cdot \mathsf{SubPos}(A) \;\cup\; \oplus_1 \cdot \mathsf{SubPos}(B)$$

$$\mathsf{SubPos}(A \,\&\, B) = \{\epsilon\} \;\cup\; \&_0 \cdot \mathsf{SubPos}(A) \;\cup\; \&_1 \cdot \mathsf{SubPos}(B)$$

Example 1. The set of positions of subformulas of $X \otimes (\bot \oplus Y)$ is $\{\epsilon, \otimes_0, \otimes_1\oplus_0, \otimes_1\oplus_1\}$.

Those formulas are equipped with an involutive negation, denoted by $\cdot^\bot$ and defined inductively as follows:

Definition 3.

$$X^\bot = \overline{X} \qquad \overline{X}^\bot = X \qquad (A \otimes B)^\bot = A^\bot \parr B^\bot \qquad (A \parr B)^\bot = A^\bot \otimes B^\bot \qquad \mathbf{1}^\bot = \bot$$

$$\bot^\bot = \mathbf{1} \qquad (A \oplus B)^\bot = A^\bot \,\&\, B^\bot \qquad (A \,\&\, B)^\bot = A^\bot \oplus B^\bot \qquad \mathbf{0}^\bot = \top \qquad \top^\bot = \mathbf{0}$$

Proofs are certificates used to attest the truth of a formula. In this thesis, proofs are built in sequent calculus, following the usage of modern proof theory, initiated by Gentzen [1935a,b, 1969].

In sequent calculus, a proof is a finite tree built out of some given deduction rules. We recall the basics of that setting. We will exemplify it first on the particular case of MALL, then on MALL extended with least and greatest fixed point in the system μMALL. For a more detailed introduction to sequent calculus, you may look at [Girard, Taylor, and Lafont, 1989] or [Girard, 1991].

Definition 4 (Sequent)**.** Given a set of formulas **Fml**, a *sequent* is defined to be a finite list of formulas. If Γ is a sequent of size n, we use the notation $|\Gamma|$ to denote either n, the size of the list, or the set $\{0, \dots, n-1\}$, the set of indices of this list. [1]

A sequent is denoted by the symbol $\vdash$ followed by its formulas. A sequent should be understood, intuitively, as the disjunction, in the $\parr$ sense, of its formulas.

A proof is a tree made out of some logical inferences.

Definition 5 (Inference)**.** An *inference* consists of :

- A finite number of sequents, $\Gamma_0, \dots, \Gamma_{n-1}$, which are called the *premises* of the inference
- A sequent Γ, which is called the *conclusion* of the inference

[1] This small abuse of notation is nothing more than the usual set-theoretic convention of identifying an integer (or an ordinal) with the set of integers (or ordinals) that are smaller than it: $n = \{m \mid m < n\}$

$$\frac{}{\vdash A, A^\perp}\,(\mathrm{id}) \qquad \frac{\vdash \Gamma, A \quad \vdash \Delta, A^\perp}{\vdash \Gamma, \Delta}\,(\mathrm{cut})$$

$$\frac{\vdash \Gamma, A \quad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B}\,(\otimes) \qquad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B}\,(\parr) \qquad \frac{}{\vdash \mathbf{1}}\,(\mathbf{1}) \qquad \frac{\vdash \Gamma}{\vdash \Gamma, \bot}\,(\bot)$$

$$\frac{\vdash \Gamma, A}{\vdash \Gamma, A \oplus B}\,(\oplus_0) \qquad \frac{\vdash \Gamma, B}{\vdash \Gamma, A \oplus B}\,(\oplus_1) \qquad \frac{\vdash \Gamma, A \quad \vdash \Gamma, B}{\vdash \Gamma, A \,\&\, B}\,(\&) \qquad \frac{}{\vdash \Gamma, \top}\,(\top)$$

Figure 2.1. MALL inference rules

- A partial function
$$\sigma \colon |\Gamma_0| + \cdots + |\Gamma_{n-1}| \rightharpoonup |\Gamma| \times \mathcal{P}^*$$
where $|\Gamma_0| + \cdots + |\Gamma_{n-1}|$ denotes the usual set-theoretic sum
$$\{0\} \times |\Gamma_0| \cup \cdots \cup \{n-1\} \times |\Gamma_{n-1}|.$$
This function is called the *threading function*. It can equivalently be described by n partial functions
$$\sigma_0 \colon |\Gamma_0| \rightharpoonup |\Gamma| \times \mathcal{P}^* \qquad \ldots \qquad \sigma_{n-1} \colon |\Gamma_{n-1}| \rightharpoonup |\Gamma| \times \mathcal{P}^*$$

This inference is written
$$\frac{\Gamma_0 \quad \ldots \quad \Gamma_{n-1}}{\Gamma}$$
with the threading function σ generally left implicit.

The inference rules of MALL are given on Figure 2.1.

For instance, the rule
$$\frac{\vdash \Gamma, A \quad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B}\,(\otimes)$$
should be understood, intuitively, as saying that for every lists of formulas Γ, Δ and for every formulas A, B, whenever the sequents $\vdash \Gamma, A$ and $\vdash \Delta, B$ are true, so is $\vdash \Gamma, \Delta, A \otimes B$.

Formally, it should also be understood by this rule that in a $(\otimes)$ inference, the function σ sends any position of formula in Γ or Δ in the premises to the corresponding position in the conclusion and the empty path, sends the position of A in the first premise on the position of $A \otimes B$ in the conclusion and the path $\otimes_0$ and sends the position of B in the second premise on the position of $A \otimes B$ in the conclusion and the path $\otimes_1$. Meanwhile, the order of the formulas in the sequents is not imposed by the rule. For instance, this is a correct $(\otimes)$ inference:

$$\frac{\vdash X \parr \overline{X}, X \quad \vdash X, Y, Z}{\vdash X \otimes Y, X \parr \overline{X}, X, Z}\,(\otimes)$$

with the implicit function

$$\begin{aligned}\sigma \colon \{0\} \times \{0,1\} \cup \{1\} \times \{0,1,2\} &\rightharpoonup \{0,1,2,3\} \times A^*_{\mathbf{Fml}} \\ (0,0) &\mapsto (1, \epsilon) \\ (0,1) &\mapsto (0, \otimes_0) \\ (1,0) &\mapsto (2, \epsilon) \\ (1,1) &\mapsto (0, \otimes_1) \\ (1,2) &\mapsto (3, \epsilon)\end{aligned}$$

Here this function happens to be total. The possibility to be partial is used in the (cut) rule.

Definition 6 (Proof tree)**.** A *proof* for MALL is any finite proof tree built with these inference rules.

Here is an example:

Example 2.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \overline{X}, X}\,(\mathrm{id}) \quad \dfrac{\dfrac{}{\vdash \overline{Y}, Y}\,(\mathrm{id})}{\vdash \overline{Y}, Y \oplus Z}\,(\oplus_0)}{\vdash \overline{X}, \overline{Y}, X \otimes (Y \oplus Z)}\,(\otimes)}{\vdash \overline{X} \parr \overline{Y}, X \otimes (Y \oplus Z)}\,(\parr) \quad \dfrac{\dfrac{\dfrac{}{\vdash \overline{X}, X}\,(\mathrm{id}) \quad \dfrac{\dfrac{}{\vdash \overline{Z}, Z}\,(\mathrm{id})}{\vdash \overline{Z}, Y \oplus Z}\,(\oplus_1)}{\vdash \overline{X}, \overline{Z}, X \otimes (Y \oplus Z)}\,(\otimes)}{\vdash \overline{X} \parr \overline{Z}, X \otimes (Y \oplus Z)}\,(\parr)}{\vdash \overline{X} \parr \overline{Y} \,\&\, \overline{X} \parr \overline{Z}, X \otimes (Y \oplus Z)}\,(\&)$$

One of the interests of the threading functions is that they can be composed.

Definition 7 (composition of threading functions)**.** If A, B, C are three sets and $\sigma\colon A \rightharpoonup B \times A^*_{\mathbf{Fml}}$ and $\rho\colon B \rightharpoonup C \times A^*_{\mathbf{Fml}}$ are two partial functions, we define their composite

$$\begin{aligned} \rho \circ \sigma\colon A &\rightharpoonup C \times A^*_{\mathbf{Fml}} \\ a &\mapsto (c, v \cdot u) \quad \text{where } (b,u) = \sigma(a) \text{ and } (c,v) = \rho(b) \end{aligned}$$

Remark 3. The interested reader may recognize the Kleisli category of the monad $_ \times A^*_{\mathbf{Fml}}$ on Set, whose multiplication and unit are induced by the monoidal structure of $A^*_{\mathbf{Fml}}$.

This fact allows us to extend this definition in order to relate any two sequents which are connected by a chain of successive inferences:

Definition 8 (Threading function associated to a path in a proof tree)**.** Suppose a proof tree π. Suppose a sequence $u = s_0, s_1, \ldots, s_n)$ of positions of sequents in π and a sequence $r_0, r_1, \ldots, r_{n-1}$ of inferences in π such that $\forall i \in [\![0, n-1]\!]$, s_i is the conclusion of r_i and s_{i+1} is a premise of r_i. $\forall i$, let Γ_i be the sequent at position s_i in π. Then, by composing the threading functions of $r_0, r_1, \ldots, r_{n-1}$, we get a function

$$\sigma(u)\colon |\Gamma_n| \rightharpoonup |\Gamma_0| \times \mathcal{P}^*$$

Example 3. For instance, if we denote by u the following sequence of sequents and inferences, taken from the right of Example 2:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dots \quad \dfrac{\vdash \overline{Z}, Y}{\vdash \overline{Z}, Y \oplus Z}\,(\oplus_1)}{\vdash \overline{X}, \overline{Z}, X \otimes (Y \oplus Z)}\,(\otimes)}{\vdash \overline{X} \parr \overline{Z}, X \otimes (Y \oplus Z)}\,(\parr) \quad \dots}{\vdash \overline{X} \parr \overline{Y} \,\&\, \overline{X} \parr \overline{Z}, X \otimes (Y \oplus Z)}\,(\&)$$

Then we have

$$\begin{aligned} \sigma(u)\colon |\overline{Z}, Y| &\rightharpoonup |\overline{X} \parr \overline{Y} \,\&\, \overline{X} \parr \overline{Z}, X \otimes (Y \oplus Z)| \\ 0 &\mapsto (0, \&_1 \parr_1) \\ 1 &\mapsto (1, \otimes_1 \oplus_1) \end{aligned}$$

Definition 9 (Image of a subformula in an inference)**.** Suppose a proof tree π and an inference $r = \dfrac{\Gamma_0 \quad \dots \quad \Gamma_{n-1}}{\Gamma}$ in π. Let us denote by $\sigma(r)$ the threading function of r. If Γ_i is one of its premises, if A is a formula in Γ_i, at position $f \in |\Gamma_i|$, if $p \in \mathsf{SubPos}(A)$ is

a position of subformula in A, then to this position $f \cdot p$ of subformula in Γ_i is associated, when possible, a position of subformula in Γ, denoted by $\mathfrak{t}(r)(f \cdot p)$ and defined as

$$\mathfrak{t}(r)(f \cdot p) \coloneqq g \cdot q \cdot p \qquad \text{where } (g, q) \coloneqq \sigma(r)$$

This makes $\mathfrak{t}(r)$ a partial function from positions of subformulas in the premises of r to positions of subformulas in the conclusion of r.

Similarly to what we did for threading functions in Definition 8, we can extend this definition of $\mathfrak{t}$ from rules to paths in a proof tree:

Definition 10 (Image of a subformula through a path in a proof tree)**.** Suppose a proof tree π. Suppose a sequence $u = (s_0, s_1, \dots, s_n)$ of positions of sequents in π and a sequence $r_0, r_1, \dots, r_{n-1}$ of inferences in π such that $\forall i \in [\![0, n-1]\!]$, s_i is the conclusion of r_i and s_{i+1} is a premise of r_i. $\forall i$, let Γ_i be the sequent at position s_i in π. Then, by composing the $\mathfrak{t}(r_i)$, we get a partial function from the position of subformulas in Γ_n to the positions of subformulas in Γ_0, which we denote by $\mathfrak{t}(u)$.

Proposition 2. *With the notations of the previous definition, for every formula A in Γ_n, at position $f \in |\Gamma_n|$ and every position of of subformula $p \in \mathsf{SubPos}(A)$, if $\mathfrak{t}(u)(f \cdot p)$ is defined, then the subformula at position $\mathfrak{t}(u)(f \cdot p)$ in Γ_0 is identical to the subformula at position $f \cdot p$ in Γ_n.*

Proof. Immediate by induction on the length of u and examination of each possible rule. The threading functions have been chosen precisely to witness the subformula property. □

Definition 11 (Path of subformula images)**.** Suppose a proof tree π. Suppose a sequence $u = s_0, s_1, \dots, s_n)$ of positions of sequents in π and a sequence $r_0, r_1, \dots, r_{n-1}$ of inferences in π such that $\forall i \in [\![0, n-1]\!]$, s_i is the conclusion of r_i and s_{i+1} is a premise of r_i. $\forall i$, let Γ_i be the sequent at position s_i in π and let p be a subformula in one of the formulas of Γ_n. We define a sequence of positions of subformulas $p_{i_0}, \dots, p_n$, such that $\forall i \geqslant i_0$, p_i is a subformula of a formula of Γ_i. This sequence is defined as follows:

$$p_n \coloneqq p$$
$$\forall i, p_{i-1} \coloneqq \mathfrak{t}(r_{i-1})(p_i)$$

or, equivalently:

$$\forall i, p_i \coloneqq \mathfrak{t}(s_i, \dots, s_n)(p)$$

The last defined index of this sequence, i_0, is either 0 or the first i such that $\mathfrak{t}(r_{i-1})(p_i)$ is not defined, which happens if r_{i-1} is a cut and p_i is in a cut formula. This sequence is denoted by

$$\mathfrak{T}(u)(p) \coloneqq p_{i_0}, \dots, p_n$$

2.1.2 Finite proof trees: μMALL

In this section, we introduce the logic μMALL [Baelde, 2012], its formulas and proofs, which are finite sequent calculus proofs.

μMALL formulas The grammar of formulas is obtained by extending the formulas of MALL with two fixed-point connectives, μ and ν, denoting respectively least and greatest fixed points of formulas.

Formulas of μMALL are selected among a set of preformulas. Preformulas of μMALL are obtained by taking the usual formulas of MALL and adding two monadic second order binders, μ and ν:

Definition 12 (μMALL *preformulas*)**.**

$$A, B ::= X \mid A \otimes B \mid A \parr B \mid \mathbf{1} \mid \bot \mid A \oplus B \mid A \,\&\, B \mid \mathbf{0} \mid \top \mid \mu X A \mid \nu X A$$

where X ranges over an infinite set of propositional variables.

The connectives μ and ν are binders and, as usual, preformulas are considered modulo renaming of bound variables. For instance, $\nu X(X \otimes X)$ and $\nu Y(Y \otimes Y)$ denote the same preformula.

Definition 13 (μMALL formulas)**.** A *formula* is a closed preformula. We denote by **Fml** the set of all formulas and we denote by $\leqslant$ the usual subformula ordering between formulas and preformulas.

Definition 14 (μMALL negation)**.** An involutive negation $\cdot^\perp$ is defined on every μMALL preformula, inductively specified by:

$$\begin{array}{lll} (A \otimes B)^\perp = A^\perp \parr B^\perp & \mathbf{1}^\perp = \bot & X^\perp = X \\ (A \oplus B)^\perp = A^\perp \,\&\, B^\perp & \mathbf{0}^\perp = \top & (\mu X A)^\perp = \nu X A^\perp \end{array}$$

Example 4. If A is any formula and $F = \nu X(\mu Y((A \otimes X) \parr Y))$ then $F^\perp = \mu X(\nu Y((A^\perp \parr X) \otimes Y))$.

Remark 4. It may be counterintuitive that $X^\perp = X$. Yet, in practice negation will only be applied to formulas, which are *closed* preformulas. This simple hack allows us to avoid

$$\frac{\vdash \Gamma, A[\mu X A[X]]}{\vdash \Gamma, \mu X A[X]}\,(\mu) \qquad\qquad \frac{\vdash \Gamma, B \quad \vdash B^\perp, A[B]}{\vdash \Gamma, \nu X A[X]}\,(\nu_{\mathrm{inv}})$$

Figure 2.2. μMALL rules for μ and ν



Figure 2.3. Threading functions for the (μ) and (ν_{inv}) rules of μMALL

the use of negative atoms $\overline{X}, \overline{Y}, \ldots$ The fact that we have only positive atoms garantees in turn that bound variables can only appear in covariant position, thus avoiding the need for a positivity condition when forming a fixed point formula.

In the presence of negative atoms, negation should be defined by:

$$X^\perp = \overline{X} \qquad\qquad \overline{X}^\perp = X$$
$$(\mu X A[X])^\perp = \nu \overline{X} A[X]^\perp \qquad\qquad (\nu X A[X])^\perp = \mu \overline{X} A[X]^\perp$$

which does not change its definition on closed formulas. For instance, in Example 4 we would still have:

$$\begin{aligned} F^\perp &= (\nu X(\mu Y((A \otimes X) \parr Y)))^\perp \\ &= \mu \overline{X}(\mu Y((A \otimes X) \parr Y))^\perp \\ &= \mu \overline{X}(\nu \overline{Y}((A \otimes X) \parr Y)^\perp) \\ &= \mu \overline{X}(\nu \overline{Y}((A^\perp \parr \overline{X}) \otimes \overline{Y})) \\ &= \mu X(\nu Y((A^\perp \parr X) \otimes Y)) \end{aligned}$$

μMALL **inferences and proofs** The proofs for μMALL are standard sequent calculus proofs, as described in Section 2.1.1. They are obtained by extending the inference system for MALL, in Figure 2.1 with two rules. Those rules reflect the definitions of least and greatest fixed points of non-decreasing function on ordered sets that we gave in Section 1.1.1.

Definition 15. The set of rules of the sequent calculus μMALL is the union of the rules for MALL, given in Figure 2.1 and of the two rules given in Figures 2.2 and 2.3.

Remark 5. As it was mentionned in the informal introduction, in Section 1.2.3, the (ν_{inv}) rule of μMALL sequent calculus breaks the subformula property. Indeed, the B and $B^\perp$ that appears in the premises of the rule, on Figure 2.2, do not appear in its conclusion. This means that anyone wanting to prove a ν formula by applying that rule has to find an appropriate B, which will act as an induction or coinduction invariant.

We recall two properties of μMALL, that will be used in Chapter 5. The first one is called *functoriality*:

Proposition 3. *For every variable X and every formulas $A[X], B, C$ of μMALL, the following rule is derivable in μMALL:*

$$\dfrac{\vdash B, C}{\vdash A^\perp[B], A[C]}\,[A]$$

Proof. By induction on A. See [Baelde, 2012, Section 2.4] for a full proof. □

The second one asserts the two following equivalences:

$$\nu X A[B[X]] \multimap A[\nu X B[A[X]]] \qquad \mu X A[B[X]] \multimap A[\mu X B[A[X]]]$$

Proposition 4. *For every two formulas $A[X], B[X]$ of μMALL, the following sequents are derivable in μMALL:*

$$\vdash \nu X A^\perp[B^\perp[X]], A[\mu X B[A[X]]] \qquad \vdash \mu X A^\perp[B^\perp[X]], A[\nu X B[A[X]]]$$

Proof.

$$\Pi_0(A,B) = \dfrac{\dfrac{\dfrac{\dfrac{}{\vdash B^\perp[A^\perp[\nu X B^\perp[A^\perp[X]]]], B[A[\mu X B[A[X]]]]}\,(\text{id})}{\vdash B^\perp[A^\perp[\nu X B^\perp[A^\perp[X]]]], \mu X B[A[X]]}\,(\mu)}{\vdash A^\perp[B^\perp[A^\perp[\nu X B^\perp[A^\perp[X]]]]], A[\mu X B[A[X]]]}\,[A]}{\vdash \nu X A^\perp[B^\perp[X]], A[\mu X B[A[X]]]}\,(\nu^0_{\text{inv}})$$

$$\frac{\vdash \Gamma, A[\mu X A[X]]}{\vdash \Gamma, \mu X A[X]}\,(\mu) \qquad\qquad \frac{\vdash \Gamma, A[\nu X A[X]]}{\vdash \Gamma, \nu X A[X]}\,(\nu)$$

Figure 2.4. μMALL$^\infty$ rules for μ and ν



Figure 2.5. Threading functions for the (μ) and (ν) rules of μMALL$^\infty$

$$\Pi_1(A,B) = \dfrac{\dfrac{\dfrac{\vdots\ \Pi_0(B^\perp, A^\perp)}{\vdash B^\perp[\mu X A^\perp[B^\perp[X]]], \nu X B[A[X]]}}{\vdash A^\perp[B^\perp[\mu X A^\perp[B^\perp[X]]]], A[\nu X B[A[X]]]}\,[A]}{\vdash \mu X A^\perp[B^\perp[X]], A[\nu X B[A[X]]]}\,(\mu)$$

□

2.1.3 Infinite proof trees: μMALL$^\infty$

μMALL$^\infty$ [Baelde, Doumane, and Saurin, 2016, Doumane, 2017b] is a non well-founded proof system for an extension of MALL with least and greatest fixed points operators. It was designed to fix the defects of the (ν_{inv}) rule of μMALL.

Definition 16. The formulas of μMALL$^\infty$ are the same as the formulas of μMALL, in the previous section.

Inference rules for μMALL$^\infty$. The proofs for μMALL$^\infty$ will be defined as infinite, non well-founded proof trees. They are built out of a set of inference rules, which is given in the following definition.

Definition 17 (μMALL$^\infty$ inference rules)**.** The set of inference rules for μMALL$^\infty$ is the union of the rules for MALL, given in Figure 2.1 and of the two rules given in Figures 2.4 and 2.5.

Preproofs for μMALL$^\infty$. Proofs for μMALL$^\infty$ are selected among a set of preproofs, which are potentially infinite objects, defined by allowing ordinary proof trees (Definition 6) to be infinite.

Definition 18 (μMALL$^\infty$ preproofs)**.** A preproof for μMALL$^\infty$ is any proof tree, finite or infinite, built with the inferences for μMALL$^\infty$ given in Definition 17.

Example 5. Let $F = \mu X \nu Y (X \parr Y)$ et $G = \nu Y \mu X (X \parr Y)$. The following are two μMALL$^\infty$ preproofs.

$$\pi_1 \quad = \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\pi_1}{\vdash \nu XX, \mu X(X \parr G), \mu XX}(\nu)}{\vdash \nu XX, \mu X(X \parr G), \mu XX}(\nu) \quad \dfrac{\dfrac{\pi_1}{\vdash \nu XX, \mu X(X \parr G), \mu XX}(\nu)}{\vdash \nu XX, G, \mu XX}}{\vdash \nu XX, \mu X(X \parr G), G, \mu XX}(\mathrm{cut})}{\vdash \nu XX, \mu X(X \parr G) \parr G, \mu XX}(\parr)}{\vdash \nu XX, \mu X(X \parr G), \mu XX}(\mu) \tag{2.1}$$

$$\pi_2 \quad = \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\pi_2}{\vdash \nu XX, \nu Y(F \parr Y), \mu XX}(\mu)}{\vdash \nu XX, F, \mu XX}(\nu)}{\vdash \nu XX, F, \mu XX}(\nu) \quad \dfrac{\pi_2}{\vdash \nu XX, \nu Y(F \parr Y), \mu XX}}{\vdash \nu XX, F, \nu Y(F \parr Y), \mu XX}(\mathrm{cut})}{\vdash \nu XX, F \parr \nu Y(F \parr Y), \mu XX}(\parr)}{\vdash \nu XX, \nu Y(F \parr Y), \mu XX}(\nu) \tag{2.2}$$

Remark 6 (Inconsistency of preproofs). Any μMALL sequent is the conclusion of a μMALL$^\infty$ preproof. More precisely, for any sequent Γ, the following is a μMALL$^\infty$ preproof:

$$\dfrac{\dfrac{\dfrac{\vdots}{\vdash \Gamma, \nu XX}(\nu)}{\vdash \Gamma, \nu XX}(\nu) \quad \dfrac{\dfrac{\vdots}{\vdash \mu XX}(\mu)}{\vdash \mu XX}(\mu)}{\vdash \Gamma}(\mathrm{cut})$$

Proofs and validity for μMALL^∞ The validity criterion used to distinguish proofs among preproofs will be given in Definition 35 and can be stated as: "every infinite branch must contain a valid thread". To make this formal, we will first define how a preproof induces two graphs and then define the "branches" and "threads" of a preproof as infinite paths in these graphs.

Definition 19 (thread of an infinite branch of a preproof)**.** Given an infinite branch β of a preproof π, we define the set of threads of β as follows. A thread in β is a sequence $(s_n)_{n\in\mathbf{N}}$ of positions of formulas in π such that:

- each s_n belongs to a sequent of β
- $\forall n \in \mathbf{N}$, s_n is the immediate descendant of s_{n+1} in β, according to the threading functions of β.

The following lemma is the key to the notion of a valid thread, which is defined right after it. If s is a position of formula in a proof tree, we denote by $\mathbf{fml}(s) \in \mathbf{Fml}$ the associated formula.

Definition 20. If $t = (s_n)_{n\in\mathbf{N}}$ is an infinite thread in a preproof, we define $\inf(t) = \{A \in \mathbf{Fml} \mid \forall n_0 \in \mathbf{N}, \exists n \geqslant n_0, s_n \text{ is principal and } \mathbf{fml}(s_n) = A\}$ *i. e.* the set of formulas that are infinitely often principal in t.

Definition 21 (Valid thread)**.** An infinite thread t is *valid* if $\inf(t)$ has a minimum, with respect to the subformula ordering, and this minimum is a ν-formula.

Definition 22 (proofs)**.** We say that an infinite branch b of a preproof ϖ is valid if there is a valid infinite thread t on b.

A μMALL^∞ preproof ϖ is a proof if all its infinite branches are valid.

2.1.4 Finite proof trees with back edges: μMALL^ω

The aim of μMALL^ω is to have a system of finite proofs, as in μMALL, with only the rules of fixed point unfolding instead of a rule with an explicit invariant, as in μMALL^∞.

As well as in μMALL^∞, proofs of μMALL^ω are selected among a set of preproofs. Preproofs of μMALL^ω are circular objects, defined by adding back edges to ordinary proof trees.

Definition 23 (μMALL$^\omega$ formulas)**.** The formulas of μMALL$^\omega$ are the same as the formulas of μMALL and μMALL$^\infty$ (Definition 13).

Definition 24 (Proof tree with open sequents)**.** A *proof tree with open sequents* is a proof tree in which it is not mandatory for a sequent to be the conclusion of a logical inference. Those positions of sequents that are not the conclusion of a logical inference are called *positions of open sequents*. The other positions are called *positions of closed sequents*.

Definition 25 (lower, closer to the root)**.** Whenever v, v' are two vertices of a tree, we say that v is *lower* than v', or that v is *below* v', or that v is *closer to the root* than v' when v appears on the path from v' to the root.

The following definition aims at defining some representation for the fragment of all μMALL$^\infty$ preproofs that are regular, that is those that have only a finite number of subtrees up to isomorphism.

Definition 26 ($\Pi_0(\mu\text{MALL}^\omega)$: μMALL$^\omega$ b.e.-tree preproofs)**.** A *μMALL$^\omega$ b.e.-tree preproof* consists of a finite proof tree with open sequents π (Def. 24), composed using the rules of μMALL$^\infty$ given above in Def. 17, together with a function **back**, which associates to each position s of an open sequent in π, a position $\mathbf{back}(s)$ of the same sequent in π, such that $\mathbf{back}(s)$ is strictly below s in π, *i. e.* between s and the root. We denote by $\Pi_0(\mu\text{MALL}^\omega)$ the set of all μMALL$^\omega$ b.e.-tree preproofs.

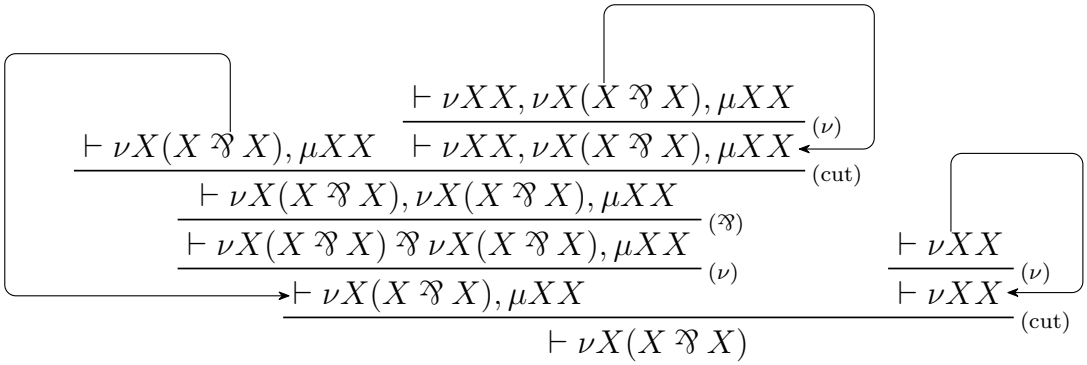
We will sometimes say simply preproof instead of b.e.-tree preproof, or circular preproof representation.

Example 6. Let π be the following proof tree, with three open sequents, and let us denote by $s_0, \dots, s_8$ the positions of its sequents, as indicated:

$$\dfrac{\dfrac{\dfrac{\dfrac{s_4 \colon \vdash \nu X(X \parr X), \mu X X \qquad \dfrac{s_6 \colon \vdash \nu X X, \nu X(X \parr X), \mu X X}{s_5 \colon \vdash \nu X X, \nu X(X \parr X), \mu X X}\,(\nu)}{s_3 \colon \vdash \nu X(X \parr X), \nu X(X \parr X), \mu X X}\,(\text{cut})}{s_2 \colon \vdash \nu X(X \parr X) \parr \nu X(X \parr X), \mu X X}\,(\parr)}{s_1 \colon \vdash \nu X(X \parr X), \mu X X}\,(\nu) \qquad \dfrac{s_8 \colon \vdash \nu X X}{s_7 \colon \vdash \nu X X}\,(\nu)}{s_0 \colon \vdash \nu X(X \parr X)}\,(\text{cut})$$

then $(\pi, \{s_4 \mapsto s_1, s_6 \mapsto s_5, s_8 \mapsto s_7\})$ is a μMALL$^\omega$ b.e.-tree preproof, that we will more

simply denote by:



The validity criterion that will be used to distinguish μMALL^ω proofs among preproofs is the same as in μMALL^∞. We simply have to adapt the formalism to be able to talk about infinite branches and infinite threads of a finite circular representation. We will first define how a preproof induces two graphs and then define the "branches" and "threads" of a preproof as infinite paths in these graphs.

Definition 27 (Graph)**.** In the following definitions, a "*graph*" always means a directed pseudograph, *i. e.* a directed graph which may have loops and in which there may be several edges between any pair of vertices.

Definition 28 (G_{branch}, branch graph of a preproof)**.** Let $(\pi, \mathbf{back})$ be a μMALL^ω b.e.-tree preproof. Its *branch graph* is the graph G_{branch} defined as follows. The vertices of G_{branch} are the positions of closed sequents in π. For each inference I with conclusion s in π and for each premise s' of I, there is an edge in G_{branch}, from s to s' if s' is a closed position of sequent in π, and from s to $\mathbf{back}(s')$ if s' is an open position of sequent in π.

Definition 29 (Infinite branch)**.** If $(\pi, \mathbf{back})$ is a b.e.-tree preproof and G_{branch} is its branch graph, we call an *infinite branch* of this preproof any infinite path in G_{branch} starting from the root of π.

To clarify the following definitions, remember that in every proof tree π, for every inference I in π, every position of formula α in a premise of I has a unique immediate descendent in the conclusion of I, except if I is a cut and α is a cut formula, in which case α has no immediate descendent.

Example 7. The infinite branches of the preproof of Example 6 are $s_0(s_7)^\omega$, $s_0(s_1s_2s_3)^\omega$ and all elements of $\{s_0(s_1s_2s_3)^k(s_5)^\omega \mid k \in \mathbf{N}\}$.

Note that, in order to be totally rigorous, we should not only give the vertices of the paths but also the edges, *i. e.* when an inference has several premises, indicate explicitly which one was chosen. These details are omitted here for concision; they will cause no ambiguity on the validity of the preproof of Example 6.

Definition 30 ($G^{\text{out}}_{\text{thread}}$, thread graph of a preproof)**.** Let $(\pi, \mathbf{back})$ be a μMALL^ω b.e.-tree preproof. Its thread graph is the graph $G^{\text{out}}_{\text{thread}}$ defined as follows. The vertices of $G^{\text{out}}_{\text{thread}}$ are the positions of formulas in the closed sequents of π. For each inference I with conclusion s in π, for each premise s' of I and for each position of formula β in s' which has an immediate descendent α in s, there is an edge in $G^{\text{out}}_{\text{thread}}$, from α to β if s' is a closed position of sequent in π, and from α to the position of the formula corresponding to β in $\mathbf{back}(s')$ if s' is an open position of sequent in π.

Definition 31 (thread)**.** A *thread* in a b.e.-tree preproof is simply a path (finite or infinite) in $G^{\text{out}}_{\text{thread}}$.

Example 8. Let us denote by $\{\alpha, \beta, \gamma, \delta, \epsilon, \zeta, \eta, \theta, \iota, \kappa, \lambda, \xi\}$ the vertices of $G^{\text{out}}_{\text{thread}}$ for the preproof shown on Example 6, as indicated here:

$$\dfrac{\dfrac{\vdash \nu X(X \parr X), \mu X X \qquad \dfrac{\vdash \nu X X, \nu X(X \parr X), \mu X X}{\vdash \nu X X_\iota, \nu X(X \parr X)_\kappa, \mu X X_\lambda}\,(\nu)}{\vdash \nu X(X \parr X)_\zeta, \nu X(X \parr X)_\eta, \mu X X_\theta}\,(\text{cut})}{\dfrac{\vdash \nu X(X \parr X) \parr \nu X(X \parr X)_\delta, \mu X X_\epsilon}{\vdash \nu X(X \parr X)_\beta, \mu X X_\gamma}\,(\nu)}\,(\parr) \qquad \dfrac{\vdash \nu X X}{\vdash \nu X X_\xi}\,(\nu)}{\vdash \nu X(X \parr X)_\alpha}\,(\text{cut})$$

The maximal threads of this preproof, that is those that cannot be extended in any direction, are $(\xi)^\omega$, $\gamma\epsilon\theta(\lambda)^\omega$, $(\iota)^\omega$, $\alpha(\beta\delta\zeta)^\omega$ and the elements of $\{\alpha(\beta\delta\zeta)^k\beta\delta\eta(\kappa)^\omega \mid k \in \mathbf{N}\}$.

Once again, in order to be totally rigorous, we should explicitly include the explicit choice of a descendent relation, in cases where a position has several immediate ancestors.

Definition 32 ($\mathbf{U} \colon G^{\text{out}}_{\text{thread}} \to G_{\text{branch}}$)**.** For any b.e.-tree preproof, there is an obvious graph morphism from $G^{\text{out}}_{\text{thread}}$ to G_{branch}, associating to every position of a formula the sequent position it belongs to. We denote this graph morphism by $\mathbf{U}$. If t is a path in $G^{\text{out}}_{\text{thread}}$ (*i. e.* an outer thread), we will also denote by $\mathbf{U}(t)$ the corresponding path in G_{branch}.

Example 9. The images, by the morphism of Definition 32, of the threads of Example 8 are, with the notations of Example 6:

$$\mathbf{U}((\xi)^\omega) = (s_7)^\omega \qquad \mathbf{U}(\gamma\epsilon\theta(\lambda)^\omega) = s_1s_2s_3(s_5)^\omega \qquad \mathbf{U}((\iota)^\omega) = (s_5)^\omega$$

$$\mathbf{U}(\alpha(\beta\delta\zeta)^\omega) = s_0(s_1s_2s_3)^\omega \qquad \forall k \in \mathbf{N}, \mathbf{U}(\alpha(\beta\delta\zeta)^k\beta\delta\eta(\kappa)^\omega) = s_0(s_1s_2s_3)^{k+1}(s_5)^\omega$$

The definition of a valid thread is then the same as in μMALL$^\infty$:

Definition 33. If $t = (s_n)_{n\in\mathbf{N}}$ is an infinite thread in a b.e.-tree preproof, we define $\inf(t) = \{A \in \mathbf{Fml} \mid \forall n_0 \in \mathbf{N}, \exists n \geqslant n_0, s_n \text{ is principal and } \mathbf{fml}(s_n) = A\}$ *i. e.* the set of formulas that are infinitely often principal in t.

Definition 34 (Valid thread)**.** An infinite thread t is *valid* if $\inf(t)$ has a minimum, with respect to the subformula ordering, and this minimum is a ν-formula.

Example 10. Among the threads of Example 8:

- $(\xi)^\omega$ is valid: its smallest infinitely principal formula is νXX, which is principal at ξ;
- $(\iota)^\omega$ is valid: its smallest infinitely principal formula is νXX, which is principal at ι;
- $\alpha(\beta\delta\zeta)^\omega$ is valid: its smallest infinitely principal formula is $\nu X(X \parr X)$, which is principal at β;
- $\gamma\epsilon\theta(\lambda)^\omega$ is not valid: it has no principal formula;
- $\forall k \in \mathbf{N}, \alpha(\beta\delta\zeta)^k\beta\delta\eta(\kappa)^\omega$ is not valid: it has no principal formula after the last position of β.

Definition 35 (proofs)**.** We say that an infinite branch b of a b.e.-tree preproof ϖ is valid if there is a valid infinite thread t of ϖ such that $\mathbf{U}(t)$ is a suffix of b.

A μMALL$^\omega$ b.e.-tree preproof ϖ is a *proof* if all its infinite branches are valid.

We denote by $\Pi(\mu\text{MALL}^\omega)$ the set of all μMALL$^\omega$ b.e.-tree proofs and we denote by $\overline{\Pi(\mu\text{MALL}^\omega)}$ its complement in $\Pi_0(\mu\text{MALL}^\omega)$, *i. e.* the set of all invalid b.e.-tree preproofs.

Example 11. The preproof of Example 6 is a proof:

- the branch $s_0(s_7)^\omega$ contains the valid thread $(\xi)^\omega$;
- the branch $s_0(s_1s_2s_3)^\omega$ contains the valid thread $(\beta\delta\zeta)^\omega$;
- $\forall k \in \mathbf{N}$, the branch $s_0(s_1s_2s_3)^k(s_5)^\omega$ contains the valid thread $(\iota)^\omega$.

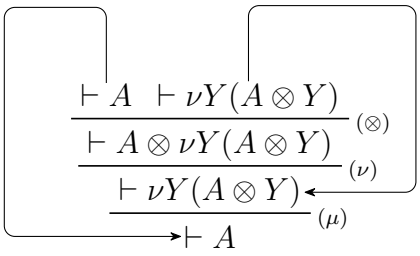
Link between circular representations and infinite preproofs

To every μMALL$^\omega$ b.e.-tree preproof ϖ is associated a canonical μMALL$^\infty$ preproof $[\![\varpi]\!]_\infty$. The finite circular preproof ϖ may then be seen as a finite representation of the infinite preproof $[\![\varpi]\!]_\infty$.

Without going into too much details, we give an explanation of how to define the transformation $[\![\cdot]\!]_\infty$. We first define a process F of expansion of a circular preproof. Let ϖ be a circular preproof. We construct $F(\varpi)$ as follows. We say that a back edge b is minimal in ϖ if the position of its target is minimal, meaning that no other back edge has a target strictly closer to the root of ϖ. For every minimal back edge b of ϖ, we substitute, in place of the source of b, the whole subtree of ϖ rooted at the target of b, including its back edges. This gives us $F(\varpi)$. Remark that the proof tree of ϖ is a strict prefix of the proof tree of $F(\varpi)$ and that the greatest prefix of ϖ without any target of back edge is, again, strictly smaller than the greatest prefix of $F(\varpi)$ without any target of back edge. Said differently: the lowest target of back edges in $F(\varpi)$ are strictly higher than in ϖ. By iterating F on ϖ we then obtain a strictly increasing sequence of circular preproofs, in which back edges targets are higher and higher and in which the prefix of the proof tree which is free of all back edge targets is increasing. $[\![\varpi]\!]_\infty$ is defined as the limit of those prefixes: $[\![\varpi]\!]_\infty = \lim_{n\to\infty} F^n(\varpi)$, in a sense that can be made formal.

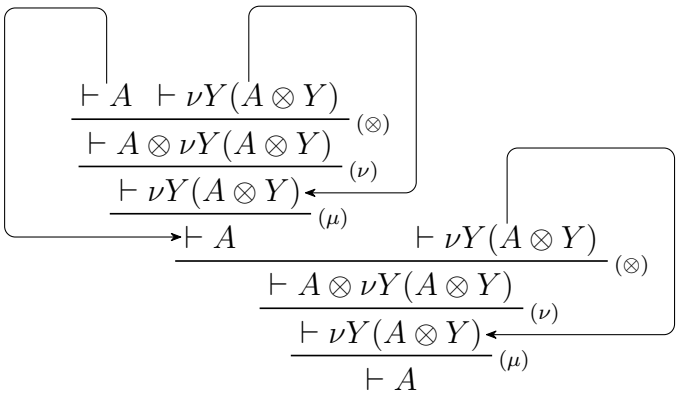
Example 12. Let us denote $A = \mu X \nu Y (X \otimes Y)$ and consider the following circular preproof:

$\varpi =$



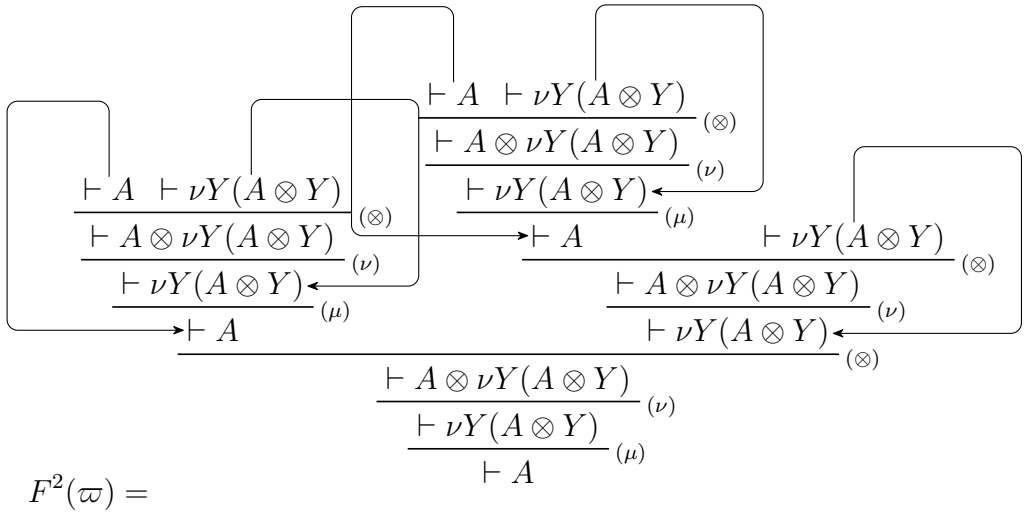
then its first step of unfolding is

$F(\varpi) =$



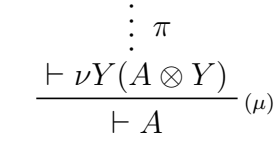
A second unfolding gives

$F^2(\varpi) =$

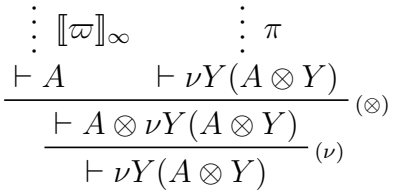


and, at the limit, we get the infinite preproof $[\![\varpi]\!]_\infty$ which is such that

$[\![\varpi]\!]_\infty =$



with $\pi =$



The principal remark that can be made about this unfolding is that $F(\varpi)$, as a μMALL^∞ preproof, has the same branches as ϖ as a μMALL^ω preproof. And each branch of $F(\varpi)$ has the same threads as the corresponding branch of ϖ. Hence, $F(\varpi)$ is valid as a

μMALL$^\infty$ preproof iff. ϖ is valid as a μMALL$^\omega$ preproof. Or: $F(\varpi)$ is a μMALL$^\infty$ proof iff. ϖ is a μMALL$^\omega$ proof. This is no coincidence and that remark is the origin of our formulation of the thread criterion for finite, circular representations.

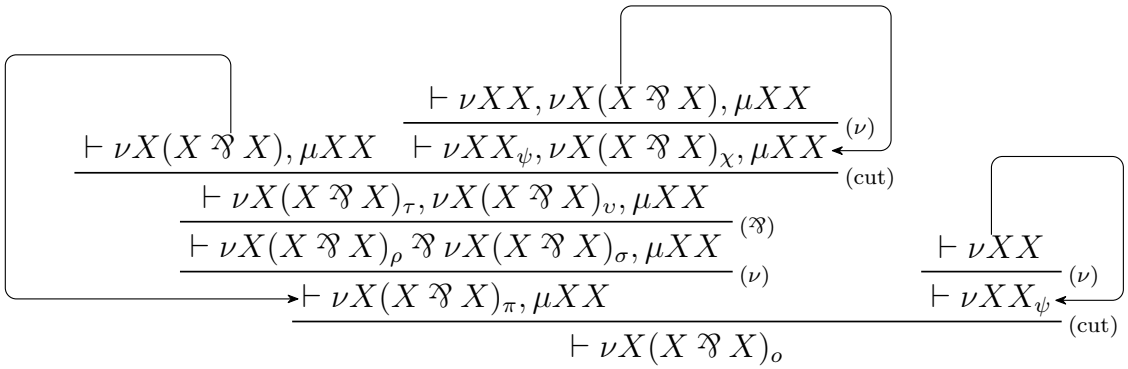
The second natural remark is that this induces an equivalence on circular preproofs: ϖ and ϖ' can be seen as representing the same infinite preproof when $F(\varpi) = F(\varpi')$. By the previous remark, this equivalence relation is compatible with thread-validity.

Inner threads

Definition 36 ($G^{\text{in}}_{\text{thread}}$, inner thread graph of a preproof)**.** Let $(\pi, \mathbf{back})$ be a μMALL$^\omega$ b.e.-tree preproof. Its inner thread graph is the graph $G^{\text{in}}_{\text{thread}}$ defined as follows. The vertices of $G^{\text{in}}_{\text{thread}}$ are the positions of ν-subformulas in the closed sequents of π. For each inference I with conclusion s in π, for each premise s' of I, for each position of formula β_0 in s' which has an immediate descendent α_0 in s and for each position of ν-subformula β in β_0, whose image in α_0 is denoted α, there is an edge in $G^{\text{in}}_{\text{thread}}$, from α to β if s' is a closed position of sequent in π, and from α to the position of the subformula corresponding to β in $\mathbf{back}(s')$ if s' is an open position of sequent in π.

Definition 37 (Inner thread)**.** An *inner thread* in a b.e.-tree preproof is simply a path (finite or infinite) in $G^{\text{in}}_{\text{thread}}$.

Example 13. Let us denote by $\{o, \pi, \rho, \sigma, \tau, \upsilon, \phi, \chi, \psi\}$ the vertices of $G^{\text{in}}_{\text{thread}}$ for the preproof shown on Example 6, as indicated here:



The maximal inner threads of this preproof are $(\psi)^\omega$, $(\psi)^\omega$, $o(\pi\rho\tau)^\omega$ and the elements of $\{o(\pi\rho\tau)^k\pi\sigma\upsilon(\chi)^\omega \mid k \in \mathbf{N}\}$.

Definition 38 ($\mathbf{out}\colon G^{\text{in}}_{\text{thread}} \to G^{\text{out}}_{\text{thread}}$)**.** For any b.e.-tree preproof, there is an obvious graph morphism from $G^{\text{in}}_{\text{thread}}$ to $G^{\text{out}}_{\text{thread}}$, associating to every position of subformula the

position of formula it belongs to. We denote this graph morphism by **out**. If t is a path in $G^{\text{in}}_{\text{thread}}$ (*i. e.* an inner thread), we will also denote by $\mathbf{out}(t)$ the corresponding path in $G^{\text{out}}_{\text{thread}}$ (*i. e.* the correponding thread).

Example 14. The images, by the morphism of Definition 38, of the inner threads of Example 13 are, with the notations of Example 8:

$$\mathbf{out}((\psi)^\omega) = (\mu)^\omega \qquad \mathbf{out}((\psi)^\omega) = (\iota)^\omega \qquad \mathbf{out}(o(\pi\rho\tau)^\omega) = \alpha(\beta\delta\zeta)^\omega$$

$$\forall k \in \mathbf{N}, \mathbf{out}(o(\pi\rho\tau)^k\pi\sigma\upsilon(\chi)^\omega) = \alpha(\beta\delta\zeta)^k\beta\delta\eta(\kappa)^\omega$$

Definition 39 (Valid inner thread)**.** An infinite inner thread $t = (s_n)_{n\in\mathbf{N}}$ is *valid* if $\forall n_0 \in \mathbf{N}, \exists n \geqslant n_0, s_n$ is principal.

Example 15. Among the inner threads of Example 13:

- $(\psi)^\omega$ is valid: ψ is principal;
- $(\psi)^\omega$ is valid: ψ is principal;
- $o(\pi\rho\tau)^\omega$ is valid: π is principal;
- $\forall k \in \mathbf{N}, o(\pi\rho\tau)^k\pi\sigma\upsilon(\chi)^\omega$ is not valid: χ is not principal and it has no principal vertex after the last π.

Proposition 5 (Equivalence between the two definitions of valid infinite threads)**.**

1. *If t is a valid infinite inner thread then $\mathbf{out}(t)$ is a valid infinite thread.*
2. *If u is a valid infinite thread then there is a unique valid infinite inner thread t such that $u = \mathbf{out}(t)$.*

Proof.

t valid $\Rightarrow$ $\mathbf{out}(t)$ valid First remark that if A is a formula (*i. e.* closed) then the images or preimages of any of its positions along any descendent relation are still positions of A. Now, if $t = (s_n)_{n\in\mathbf{N}}$ is an inner thread, let us denote by $\mathbf{fml}(t) = \mathbf{fml}(s_0)$. Then $\forall n \in \mathbf{N}, \mathbf{fml}(s_n) = \mathbf{fml}(t)$. Because t is valid, this $\mathbf{fml}(t)$ is a ν-formula. Each sequent position at which t is principal is also a position at which $\mathbf{fml}(t)$ is principal in $\mathbf{out}(t)$, $\mathbf{fml}(t)$, hence $\mathbf{fml}(t)$ is infinitely principal in $\mathbf{out}(t)$. Finally, $\mathbf{fml}(t)$ is the smallest formula of $\mathbf{out}(t)$ because it is a subformula of every formula of $\mathbf{out}(t)$. Conclusion: $\mathbf{out}(t)$ is a valid infinite thread.

Converse if $t \in G^{\text{out}}_{\text{thread}}$ is valid, let $\nu X A$ be its validating formula, let $t = t_0 t_1 t_2 \ldots$, let i_n be the sequence of all i such that t_i is a principal position of $\nu X A$. If $i_n < i < i_{n+1}$, we define u_i as the image of $t_{i_{n+1}}$ in t_i. Then $u = (u_i)_i$ is a valid inner thread such that $\mathbf{out}(u) = t$. It is easy to see that it is the only one, thanks to the following lemma.

□

Lemma 1. *The positions of $\sigma X A[X]$ in $A[\sigma X A[X]]$ are exactly the positions of X in $A[X]$.*

Proof. Otherwise, $\sigma X A[X]$ would be a subformula of $A[X]$, which is impossible. □

Example 16. The preproof of Example 6 is a proof:

- the branch $s_0(s_7)^\omega$ contains the valid inner thread $(\psi)^\omega$;
- the branch $s_0(s_1 s_2 s_3)^\omega$ contains the valid inner thread $(\pi\rho\tau)^\omega$;
- $\forall k \in \mathbf{N}$, the branch $s_0(s_1 s_2 s_3)^k(s_5)^\omega$ contains the valid inner thread $(\psi)^\omega$.

Much in the spirit of these inner threads, we can generalize Definitions 10 and 11 to the case of b.e.-tree-preproofs.

Definition 40 (Generalization of $\mathfrak{t}$ and $\mathfrak{T}$)**.** The definitions of $\mathfrak{t}$ (Definition 10) and $\mathfrak{T}$ (Definition 11) are extended in the obvious way to the setting of μMALL$^\omega$ b.e.-tree preproof: if $(\pi, \mathbf{back})$ is a μMALL$^\omega$ b.e.-tree preproof, if $u = (s_0, \ldots, s_n)$ is a path in its G_{branch} and p is a position of subformula in the sequent at position s_n, then:

- $\mathfrak{T}(u)(p)$ is the unique path $p_{i_0}, \ldots, p_n$ such that
  - $\forall i, p_i$ is a position of subformula in the sequent at position s_i,
  - $p_n = p$,
  - $\forall i, i_0 \leqslant i < n \Rightarrow$ the formula containing p_i in the sequent at position s_i is the immediate descendant of the formula containing p_{i+1} in the sequent at position s_{i+1}, and p_i is the image of p_{i+1} in that formula,
  - either $i_0 = 0$ or p_{i_0} is in a cut formula,

- $\mathfrak{t}(u)(p) = \begin{cases} p_0 & \text{if } i_0 = 0 \\ \text{undefined} & \text{if } i_0 > 0 \end{cases}$

  In this last case, p_{i_0} is in a cut formula.

Remark 7. If, in the previous definition, the subformula at position p is a ν-formula, then $\mathfrak{T}(u)(p)$ is a path in $G^{\text{in}}_{\text{thread}}$.

2.2 Complexity theory

2.2.1 Generalities on complexity and PSPACE

We give here a brief and somewhat informal recall of the essential notions of algorithmic complexity that we will need in the thesis. For an in-depth perspective on that subject, see, for instance, [Arora and Barak, 2009]. If you understand French, you may also read the excellent book by Perifel [2014].

In particular we do not go here into the choice and definition of a precise model of computation. All programs are considered deterministic. We assume that the inputs and outputs of programs contain at least $\{0,1\}^*$, the finite words on $\{0,1\}$, and that there is a pairing operation which allows to combine two entries x and y into a single one (x,y).

Definition 41 (Notations)**.** If f is a program and x an entry for this program, we denote by $\|x\|$ the size of the entry x, we denote by $\mathbf{time}(f,x)$ the execution time of f on x, which may be an integer or $+\infty$, and we denote by $\mathbf{space}(f,x)$ the memory space used by the execution of f on x, which may be an integer or $+\infty$.

Definition 42 (Problem, language)**.** A *problem* or *language* is any subset of all possible program entries.

Definition 43.

- if $t\colon \mathbf{N} \to \mathbf{N}$ is a function, a language L is in DTIME$(t(n))$ if there is a program f such that

  - $\forall x \in L, f(x) = 1$

- $\forall x \notin L, f(x) = 0$
- $\exists \alpha > 0, \forall x, \mathbf{time}(f, x) \leqslant \alpha \cdot t(\|x\|)$

- if $t \colon \mathbf{N} \to \mathbf{N}$ is a function, a language L is in $\mathrm{DSPACE}(t(n))$ if there is a program f such that
  - $\forall x \in L, f(x) = 1$
  - $\forall x \notin L, f(x) = 0$
  - $\exists \alpha > 0, \forall x, \mathbf{space}(f, x) \leqslant \alpha \cdot t(\|x\|)$

Definition 44.

- $\mathrm{P} \coloneqq \bigcup_{k \in \mathbf{N}} \mathrm{DTIME}\left(n^k\right)$
- $\mathrm{EXP} \coloneqq \bigcup_{k \in \mathbf{N}} \mathrm{DTIME}\left(2^{n^k}\right)$
- $\mathrm{PSPACE} \coloneqq \bigcup_{k \in \mathbf{N}} \mathrm{DSPACE}\left(n^k\right)$
- $\mathrm{NP} \coloneqq \{A \mid \exists p \text{ polynomial}, \exists B \in \mathrm{P}, \forall x, (x \in A \Leftrightarrow \exists y \in \{0,1\}^{p(\|x\|)}, (x, y) \in B)\}$

Proposition 6.

$$\mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PSPACE} \subseteq \mathrm{EXP}$$

The strictness of each one of these inclusions is an open problem. Experts of the domain seem to believe that all these inclusions are in fact strict. We still know for sure that not all of them can be equalities, as a consequence of Corollary 1 to Theorem 2 below.

Definition 45. A function $t \colon \mathbf{N} \to \mathbf{N}$ is *time-constructible* if there is a program f such that

- $\forall n \in \mathbf{N}, f(1^n) = 1^{t(n)}$ (where 1^n is the word of length n made only of 1)
- $\exists \alpha > 0, \forall n \in \mathbf{N}, \mathbf{time}(f, 1^n) \leqslant \alpha \cdot t(n)$

Theorem 2 (Deterministic time hierarchy)**.** *If $f, g \colon \mathbf{N} \to \mathbf{N}$ are two functions such that*

- $\forall n \in \mathbf{N}, f(n) > 0$
- g *is time-constructible*
- $g(n) \gg f(n) \cdot \log(f(n))$

then $\mathrm{DTIME}(f(n)) \subsetneq \mathrm{DTIME}(g(n))$.

Corollary 1.

$$\mathrm{P} \subsetneq \mathrm{EXP}$$

Definition 46 (Polynomial-time many-one reduction)**.** A *polynomial-time many-one reduction* from a language B to a language A is a program f which runs in polynomial time and such that $\forall x, x \in B \Leftrightarrow f(x) \in A$. If such a program exist, we say that B *reduces to* A and we denote it by $B \leqslant A$.

Definition 47. A problem A is PSPACE-*complete* if A is in PSPACE and every PSPACE problem reduces to A. Equivalently, A is PSPACE-complete iff.

$$\forall B, B \in \mathrm{PSPACE} \Leftrightarrow B \leqslant A$$

Proposition 7.

- *A problem A is in* PSPACE *iff. its complement $\overline{A}$ is in* PSPACE.
- *A problem A is* PSPACE-*complete iff. its complement $\overline{A}$ is* PSPACE-*complete.*

Proof.

- Every $\mathrm{DSPACE}(n^k)$ is closed under complement: if f recognizes A, $\overline{A}$ is recognized by $g(x) = 1 - f(x)$. Hence PSPACE is closed under complement.
- Suppose A is PSPACE-complete and B is PSPACE. We then show that B reduces to $\overline{A}$. As B is PSPACE, so is $\overline{B}$. Hence there is a polynomial-time many-one reduction f from $\overline{B}$ to A. Then we show that f is also a reduction from B to $\overline{A}$. Indeed, $\forall x, x \notin B \Leftrightarrow x \in \overline{B} \Leftrightarrow f(x) \in A$, and then $\forall x, x \in B \Leftrightarrow f(x) \notin A \Leftrightarrow f(x) \in \overline{A}$.

□

Remark 8. Although this will not be used in this thesis, we recall to the interested reader that we also have PSPACE = NPSPACE = *co*NPSPACE because of Savitch theorem.

2.2.2 Some PSPACE-complete problems

Definition 48 ($\texttt{BOOLE}_{false}$ and $\mathcal{B}_{false}$)**.** A program in $\texttt{BOOLE}_{false}$ is a sequence of instructions $b = \texttt{1:I}_1\ \texttt{2:I}_2\ \ldots\ \texttt{m:I}_m$ where an instruction can have one of the two following forms:

$$\texttt{I} ::= \texttt{X} := \neg\texttt{X} \mid \text{if } \texttt{X} \text{ then goto } \ell' \text{ else goto } \ell''$$

where X ranges over a finite set of variable names. The semantic is as expected, with all variables being equal to `false` at the beginning of the program, and the program halting when reaching instruction $m + 1$. Finally it is required of every program $b \in \texttt{BOOLE}_{false}$ that if b terminates then all variables have value *false* at the end of its execution.

We also denote the set of terminating programs as:

$$\mathcal{B}_{false} := \{b \in \texttt{BOOLE}_{false} \mid b \text{ terminates}\}$$

Remark 9. The constraint on the values of the variables at the end of the program will be useful when reducing it, in Chapter 3, to the problem of validity of a μMALL^{ω} b.e.-tree preproof. This preproof will encode the fact that the program b is terminating by connecting the final state to the initial one, hence the necessity that its initial and terminal states are the same.

Lemma 2. $\mathcal{B}_{false}$ *is* PSPACE*-hard under* LOGSPACE*-reductions:*

$$\text{PSPACE} \leqslant_{\text{L}} \mathcal{B}_{false}$$

Proof. We reduce from the problem of termination for a more expressive language, which has been defined and proved PSPACE-complete by Jones in [Jones, 1997], under the name of `BOOLE`. □

2.3 Parity automata

We recall here the basic definitions and property of parity automata. There are several definitions of parity automata, all of which are equivalent. We chose a definition that best suits our needs. For more informations on the subject of automata for infinite words, see, for instance, [Grädel, Thomas, and Wilke, 2002] or [Perrin and Pin, 2004].

We denote by ω the set of natural integers, and we also denote by ω the first infinite ordinal. We denote by $\omega + 1$ the set $\omega \cup \{\omega\}$, ordered by setting ω to be its maximum.

The ordinal ω is considered even. All of that is consistent with usual set-theoretic conventions.

Given an alphabet A, we denote by A^ω the set of infinite words on A, that is the set of sequences of elements of A, indexed by the integers.

Definition 49 (Parity automaton)**.** Given an alphabet A, a *parity automaton* on A consists of

- a set Q of *states*;
- a subset $I \subseteq Q$ of *initial states*;
- a set $T \subseteq Q \times A \times (\omega + 1) \times Q$.

Such a parity automaton is *finite* if A, Q and T are finite. In T, elements of $\omega + 1$ are called *priorities*.

Definition 50. A *run*, or *path*, in a parity automaton $\mathcal{A}$ is any infinite sequence

$$q_0 \xrightarrow[p_0]{a_0} q_1 \xrightarrow[p_1]{a_1} q_2 \xrightarrow[p_2]{a_2} q_3 \ \dots$$

such that

- $q_0 \in I$
- $\forall n \in \omega, (q_n, a_n, p_n, q_{n+1}) \in T$

If r is such a run, we denote by $\mathbf{label}(r)$ the infinite word $a_0 a_1 a_3 \cdots \in A^\omega$.

Definition 51. Such a run r is *successful* in the parity automaton if there is a $p \in \omega + 1$ such that

- p is odd (which implies $p \neq \omega$)
- there are only finitely many $n \in \omega$ such that $p_n < p$
- there are infinitely many $n \in \omega$ such that $p_n = p$

Equivalently : p is an odd priority and it is the smallest priority appering infinitely often in the run r.

Definition 52 (Language of a parity automaton)**.** The *language* of a parity automaton $\mathcal{A}$ is the set $\mathcal{L}(\mathcal{A}) \coloneqq \{\mathbf{label}(p) \mid p \text{ is a successful run in } \mathcal{A}\}$.

And the main theorem that we will use is the following.

Theorem 3. *The problem of equality of languages for finite parity automata is decidable in polynomial space.*

Proof. We do not go into the details of this complex result. It relies on the fact that every parity automaton can be translated into an equivalent Bűchi automaton with a polynomial increase of size, which is folklore, and on the the fact that the problem of equality of languages for Bűchi automata is decidable in polynomial space, which was proven by Sistla, Vardi, and Wolper [1987]. □

3 PSPACE-completeness of the thread criterion

As we explained in the introduction, the subject of this thesis is the study of finite circular representations of infinite proofs and preproofs. In this context, the first question arising is to distinguish, among circular representations of preproofs, those denoting valid preproofs, *i. e.* proofs, from those denoting invalid preproofs.

It has already been shown by Doumane [2017b] that this problem is decidable in PSPACE. In the first section of this chapter, we recall a proof of this result.

The second section of this chapter is devoted to its main result: that the problem of deciding the validity of a circular representation of a μMALL^ω preproof is in fact PSPACE-complete. This result is established by a reduction from the problem of deciding the termination of a boolean program.

Our proof takes a lot of inspiration from the proof of PSPACE-completeness of size-change termination by Lee, Jones and Ben Amram [Lee, Jones, and Ben-Amram, 2001]: in order to prove that deciding size-change termination is PSPACE-complete, they define a notion of boolean program and use the fact that the following set is complete in PSPACE:

$$\mathcal{B} = \{b \mid b \text{ is a boolean program and } b \text{ terminates.}\}$$

then they reduce $\mathcal{B}$ to the problem of size-change termination. We adapt their method by reducing $\mathcal{B}$ to the problem of thread-validity in circular μMALL^ω preproof.

It would be very interesting to get a more precise understanding of the relation between threads in circular proofs and size-change termination.

We show in Section 3.2.4 how our method adapts to other systems such as μLJ and μLK.

3.1 Deciding thread validity in PSPACE

In this section, we recall that the problem of deciding whether the circular representation of a preproof is a valid proof is in PSPACE. Several algorithms can be used for deciding this problem in PSPACE. Here we reduce this problem to the problem of deciding equality of languages for parity ω-automata, which is known to be in PSPACE. See Section 2.3 of the technical background for more details on that. More precisely, given a preproof ϖ, we define two parity automata: the language of the first one is the set of infinite branches of ϖ and the language of the second one is the set of *valid* infinite branches of ϖ.

We will immediately give two examples to show how the algorithm works.

Let $G \coloneqq \nu Y \mu X (X \parr Y)$. The following preproof (3.1) is valid. The branches that ultimately always take the left back edge are valid because of a thread going through $\nu X X$ and the branches that take infinitely often the right back edge are valid because of a thread going through $\mu X (X \parr G)$ and G.

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdash \nu X X, \mu X (X \parr G), \mu X X}{\vdash \nu X X, \mu X (X \parr G), \mu X X}\,(\nu) \quad \dfrac{\vdash \nu X X, \mu X (X \parr G), \mu X X}{\vdash \nu X X, G, \mu X X}\,(\nu)}{\vdash \nu X X, \mu X (X \parr G), G, \mu X X}\,(\text{cut})}{\vdash \nu X X, \mu X (X \parr G) \parr G, \mu X X}\,(\parr)}{\vdash \nu X X, \mu X (X \parr G), \mu X X}\,(\mu) \tag{3.1}$$

From this preproof, we build a first automaton, represented on Figure 3.1 p. 74, which recognizes the language of infinite branches of the preproof. It has one initial state and all paths are accepted. It recognizes the language $(abcd + abef)^\omega$.

And from the same preproof (3.1) we build a second automaton, represented on Figure 3.2, p. 75, which recognizes the language of valid infinite branches of the preproof (3.1). No priority indicated on an edge implicitely means ω, which is the highest non-accepting priority. There are four initial states, at the bottom of the automaton. Three edges have a priority $< \omega$

$$\nu X X \xrightarrow[1]{d} \nu X X \qquad G \xrightarrow[1]{f} \mu X (X \parr G) \qquad \mu X (X \parr G) \xrightarrow[2]{a} \mu X (X \parr G) \parr G$$

It recognizes the language

$$\begin{aligned}&(abcd + abef)^*(abcd)^\omega + ((abcd)^*abef)^\omega\\ =\ &(abcd + abef)^\omega\end{aligned}$$

Now let $F \coloneqq \mu X \nu Y (X \parr Y)$. The following preproof (3.2), unlike the preproof (3.1), is not valid. The branches taking ultimately only the left back edge are valid because of a thread going through νXX, the branches taking ultimately only the right back edge are valid because of a thread going through $\nu Y(F \parr Y)$ but the branches which takes infinitely often the left back edge and infinitely often the right back edge are not valid.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash \nu XX, \nu Y(F \parr Y), \mu XX}{\vdash \nu XX, F, \mu XX}\,(\mu)}{\vdash \nu XX, F, \mu XX}\,(\nu) \qquad \vdash \nu XX, \nu Y(F \parr Y), \mu XX}{\vdash \nu XX, F, \nu Y(F \parr Y), \mu XX}\,(\text{cut})}{\vdash \nu XX, F \parr \nu Y(F \parr Y), \mu XX}\,(\parr)}{\vdash \nu XX, \nu Y(F \parr Y), \mu XX}\,(\nu)}{\vdash \nu XX, \nu Y(F \parr Y), \mu XX} \tag{3.2}$$

The deduced automaton recognizing all infinite branches of preproof (3.2), is represented on Figure 3.3, p. 76. It has one initial state. All paths are accepted. It recognizes the language $(abcde + abf)^\omega$.

And the automaton of valid infinite branches of preproof (3.2) is represented on Figure 3.4, p. 77. It has 4 initial states. The only three transitions with priorities $< \omega$ are

$$\nu Y(F \parr Y) \xrightarrow[3]{a} F \parr \nu Y(F \parr Y) \qquad \nu XX \xrightarrow[1]{d} \nu XX \qquad F \xrightarrow[2]{e} \nu Y(F \parr Y)$$

It recognizes the language

$$\begin{aligned}&(abcde + abf)^*(abcde)^\omega + (abcde + abf)^*(abf)^\omega\\ =\ &(abcde + abf)^\omega \setminus ((abf)^*abcde(abcde)^*abf)^\omega\end{aligned}$$

We will now explain the procedure by which, in each case, those two automata were obtained.

Let ϖ be a b.e.-tree preproof. Let G_{branch} be its branch graph, as defined in Definition 28. Let $A \coloneqq E_{\text{branch}}$ be the set of its edges. This will be the input alphabet of our automata.

The first ω-automaton is $\mathcal{A}_{\text{branch}} \coloneqq (Q_{\text{branch}}, I_{\text{branch}}, T_{\text{branch}})$, where:

- The set of states is $Q_{\text{branch}} \coloneqq V_{\text{branch}}$, the set of vertices of G_{branch}, that is the set of closed positions of sequents in π.
- The set of initial states is the singleton $I_{\text{branch}} \coloneqq \{i_{\text{branch}}\}$, where i_{branch} is the position of the root of ϖ, that is the position of the conclusion sequent of ϖ.
- The set of transitions is

$$T_{\text{branch}} \coloneqq \{\ s \xrightarrow[1]{e} s' \ \mid e \text{ is an edge from } s \text{ to } s' \text{ in } G_{\text{branch}}\}$$

This amounts to saying that the accepted paths in $\mathcal{A}_{\text{branch}}$ are exactly the paths in G_{branch} which start at the root of the proof tree. With that definition, the following lemma is immediate:

Lemma 3. *The language $\mathcal{L}(\mathcal{A}_{\text{branch}})$ is the set of infinite branches of ϖ.*

For our second automaton, we need a priority assignment $\Omega\colon \mathbf{Fml} \to \omega$. It will be used to reflect the condition of validity of threads in the parity condition of our automata. We need this priority assignment to have two main properties:

1. if A is a subformula of B then $\Omega(A) \leqslant \Omega(B)$;
2. $\forall A, \quad \Omega(\mu X A)$ is even and $\Omega(\nu X A)$ is odd.

Remember that even priorities are rejecting and odd priorites are accepting (Definition 51). Thus, those two properties reflect the condition of Definition 34.

Definition 53 ($\Omega\colon \mathbf{Fml} \to \omega$)**.** A function Ω is defined by induction, which associate a priority to every *preformula*:

$$\Omega(\nu X A) \coloneqq \begin{cases} \Omega(A) & \text{if } \Omega(A) \text{ is odd} \\ \Omega(A)+1 & \text{if } \Omega(A) \text{ is even} \end{cases}$$

$$\Omega(\mu X A) \coloneqq \begin{cases} \Omega(A) & \text{if } \Omega(A) \text{ is even} \\ \Omega(A)+1 & \text{if } \Omega(A) \text{ is odd} \end{cases}$$

$$\Omega(A \odot B) \coloneqq \max\{\Omega(A), \Omega(B)\} \qquad \text{for any binary connective } \odot$$

$$\Omega(\mathbf{c}) \coloneqq 0 \qquad \text{for any propositional constant } \mathbf{c}$$

$$\Omega(X) \coloneqq 0$$

The following remarks are immediate:

Lemma 4. *For all formulas A and B:*

- *$\Omega(\nu X A)$ is odd and $\Omega(\mu X A)$ is even.*
- *If A is a subformula of B then $\Omega(A) \leqslant \Omega(B)$.*

Our second automaton is a parity ω-automaton, with priorities in $\omega + 1 = \omega \cup \{\omega\}$, ω being even.[1] This second automaton is defined as $\mathcal{A}_{\text{thread}} = (Q_{\text{thread}}, I_{\text{thread}}, T_{\text{thread}})$, where:

- the set of states is $Q_{\text{thread}} \coloneqq V_{\text{thread}} + \{\bot_s \mid s \in V_{\text{branch}}\}$, *i. e.* the vertices of $G^{\text{out}}_{\text{thread}}$ plus one extra vertex for each vertex of G_{branch}
- If we keep the notation i_{branch} to denote the position of the endsequent of ϖ, the set of initial states I_{thread} is the set of all positions of formulas in i_{branch}, plus the extra vertex $\bot_{i_{\text{branch}}}$.
- the set of transitions is

$$
\begin{aligned}
T_{\text{thread}} \coloneqq \{\bot_s \xrightarrow[\omega]{e} \bot_{s'} \mid\ & e \text{ is an edge from } s \text{ to } s' \text{ in } G_{\text{branch}}\} \\
\cup \{\ \alpha \xrightarrow[\Omega(\alpha)]{\mathbf{U}(e)} \beta \ \mid\ & e \text{ is an edge from } \alpha \text{ to } \beta \text{ in } G^{\text{out}}_{\text{thread}} \text{ and } \alpha \text{ is principal}\} \\
\cup \{\ \alpha \xrightarrow[\omega]{\mathbf{U}(e)} \beta \\
& \mid e \text{ is an edge from } \alpha \text{ to } \beta \text{ in } G^{\text{out}}_{\text{thread}} \text{ and } \alpha \text{ is not principal}\} \\
\cup \{\bot_s \xrightarrow[\omega]{e} \alpha \\
& \mid I \text{ is a (cut) inference in } \varpi, \\
& \quad s \text{ is the position of the conclusion sequent of } I, \\
& \quad s' \text{ is the position of a premise of } I, \\
& \quad e \text{ is an edge induced by } I \text{ in } G_{\text{branch}} \text{ from } s \text{ to } s' \\
& \quad \text{and } \alpha \text{ is the position of the corresponding cut-formula in } s'\}
\end{aligned}
$$

  where $q \xrightarrow[i]{e} q'$ denote a transition from state $q \in Q_{\text{thread}}$ to state $q' \in Q_{\text{thread}}$ with label $e \in A$ and priority $i \in \omega + 1$.

The acceptance condition is the one given in our definition of parity automata in Definition 49: a run is accepted if the smallest priority appearing infinitely often is *odd* (ω being even).

Once again, it should be clear from Definitions 21 and 22 that:

[1] We use the usual set-theoretic convention that an ordinal is equal to the set of ordinals strictly below it: $\alpha = \{\beta \in \text{Ord} \mid \beta < \alpha\}$

Lemma 5. *The language* $\mathcal{L}(\mathcal{A}_{\text{thread}})$ *is the set of* valid *infinite branches of* ϖ.

From these two lemmas it is immediate that

Proposition 8. *We have the inclusion* $\mathcal{L}(\mathcal{A}_{\text{thread}}) \subseteq \mathcal{L}(\mathcal{A}_{\text{branch}})$ *and the preproof* ϖ *is valid iff. this inclusion is an equality.*

Deciding this equality can be done in PSPACE (*c. f.* Theorem 3), and the constructions of these automata are obviously PSPACE, so:

Proposition 9. *The problem of deciding whether a* μMALL^ω *b.e.-tree preproof is valid is in* PSPACE.

3.2 PSPACE-completeness

3.2.1 Outline of the PSPACE-completeness proof

We now aim at proving that the problem of deciding the validity of a μMALL^ω b.e.-tree preproof is PSPACE-complete. As it is already known that this problem is in PSPACE, it remains to prove that it is PSPACE-hard.

We follow the same methodology as Lee, Jones, and Ben-Amram [2001]: in order to prove that deciding size-change termination is PSPACE-complete, they define a notion of boolean program (see Definition 48) and use the fact that the following problem is PSPACE-complete:

$$\mathcal{B} = \{b \mid b \text{ is a boolean program and } b \text{ terminates.}\}$$

then they reduce $\mathcal{B}$ to the decidability of size-change termination.

We try to adapt their method by reducing $\mathcal{B}$ to $\Pi(\mu\text{MALL}^\omega)$.

3.2.2 Definition of the reduction

We use the boolean programs that were defined in Definition 48. The following definition will be used in the proof of Proposition 10:

Definition 54 (Call graph of a program)**.** Assume a boolean program b with variables $\mathtt{X}_1, \dots, \mathtt{X}_k$ and instructions $\mathtt{1} : \mathtt{I}_1, \dots, \mathtt{m} : \mathtt{I}_m$. Define the call graph of b to be $G = (V, E)$ with

- $V = \{0, 1, \dots, m\}$
- $E = \{0 \xrightarrow{0} 1\}$
  $\cup \{\ell \xrightarrow{\ell} ((\ell + 1) \bmod (m + 1)) \mid \mathtt{I}_\ell = \text{“}\mathtt{X} \coloneqq \mathtt{not\ X}\text{”}\}$
  $\cup \{\ell \xrightarrow{\ell^+} \ell', \ell \xrightarrow{\ell^-} \ell'' \mid \mathtt{I}_\ell = \text{“}\mathtt{if\ X\ goto}\ \ell'\ \mathtt{else}\ \ell''\text{”}\}$

The following definition is the main object of this chapter. It describes a transformation that turns every boolean program b into a $\mu\mathrm{MALL}^\omega$ b.e.-tree-preproof, with the property that this preproof is thread-valid iff. the source program is non-terminating.

Definition 55 ($[\![\cdot]\!] \colon \mathtt{BOOLE}_{\mathit{false}} \to \Pi_0(\mu\mathrm{MALL}^\omega)$)**.** For every boolean program $b \in \mathtt{BOOLE}_{\mathit{false}}$, we define a preproof $[\![b]\!] \in \Pi_0(\mu\mathrm{MALL}^\omega)$. Let $\mathtt{X}_1, \dots, \mathtt{X}_k$ be the variables of b and $\mathtt{1} : \mathtt{I}_1, \dots, \mathtt{m} : \mathtt{I}_m$ its instructions. We first give names to the formulas that will appear in $[\![b]\!]$: we define a unary operation ¿, three formulas A, B, C, a family of unary operations $(¿_n)$ and two families of formulas $(D_n), (E_n)$:

$$A = ¿(\nu X ¿X) \qquad B = \nu X(\bot \oplus X) \qquad C = \mu X(B \parr X) \qquad E_n = ¿_n(\nu X ¿_n X)$$

$$¿F = \mu X(F \oplus (\bot \oplus (X \parr X))) \qquad ¿_n F = \mu X(\bot \oplus (X \parr (\underbrace{F \parr \cdots \parr F}_{n-1})))$$

$$D_n = \mu X(\underbrace{X \mathbin{\&} \cdots \mathbin{\&} X}_{n})$$

We now define $[\![b]\!]$ to be the preproof

$$\dfrac{\vdots\ [\![\mathtt{0}:]\!] \quad \vdots\ [\![\mathtt{1} : \mathtt{I}_1]\!] \quad \dots \quad \vdots\ [\![\mathtt{m} : \mathtt{I}_\mathtt{m}]\!]}{(\textsc{Root}) \vdash A^{2k}, B, C, D_2, \underline{D_m}, E_m^m} \begin{array}{l} \vdash A^{2k}, B, C, D_2, D_m, E_m^m \quad \vdash A^{2k}, B, C, D_2, D_m, E_m^m \quad \vdash A^{2k}, B, C, D_2, D_m, E_m^m \\ (\mu),\ (\mathbin{\&})^{m-1} \end{array} \tag{3.3}$$

where A^{2k} is an abbreviation for $\underbrace{A, \dots, A}_{2k}$ and E_m^m is an abbreviation for $\underbrace{E_m, \dots, E_m}_{m}$ and the subtrees $[\![\mathtt{0}:]\!], [\![\mathtt{1} : \mathtt{I}_1]\!], \dots, [\![\mathtt{m} : \mathtt{I}_\mathtt{m}]\!]$ will be defined below.

Remark 10 (Short notation for long sequents). You just saw on Eq. (3.3) that, for any formula A, we use A^n as an abbreviation for $\underbrace{A, \dots, A}_{n \text{ formulas}}$. Similarly, for any formulas A and B, we may use $(A, B)^n$ as an abbreviation for $\underbrace{A, B, \dots, A, B}_{2n \text{ formulas}}$. We do so, for instance, on Figure 3.6.

The root of the preproof $[\![b]\!]$ is constructed by translating each pair ℓ:$\mathtt{I}_\ell$ of a label and an instruction into a finite segment of branch of preproof, as defined in Eq. (3.3), with each subderivation $[\![\ell:\mathtt{I}_\ell]\!]$ defined in Fig. 3.6 and each subderivation $[\![\ell:\mathtt{goto}\ \ell']\!]$ in Fig. 3.5.

Remark 11 (Implicit threading functions). Notice that in our description of the translation in the previous definition, the threading functions were left implicit. The only exception is in the definition of $[\![\ell:\mathtt{X}_i\ :=\ \mathtt{not}\ \mathtt{X}_i]\!]$, on Figure 3.6, in which we use arrows to express the exchange of the positions of two occurrences of A.

Remark 12 (Infinite branches of $[\![b]\!] \simeq E^\omega$). The preproof $[\![b]\!]$ constructed from b by the reduction $[\![\cdot]\!]$ of Definition 55 is a finite tree with back edges in which every finite branch ends with a back edge to the root. This finite tree has exactly as many branches and consequently as many back edges to the root, as the number $\mathbf{Card}\, E$ of edges in the call-graph of b (Definition 54). This in turn entails that the set of infinite branches of the preproof $[\![b]\!]$ is in one-to-one correspondence with the set E^ω of infinite words on E. Note however that an infinite word $\overline{u} \in E^\omega$ has no reason *a priori* to be a path in G.

From now on, **we will refer directly to infinite branches of the preproof by words** $\overline{u} \in E^\omega$.

3.2.3 Main theorem

We now prove that $\Pi(\mu\mathrm{MALL}^\omega)$ is PSPACE-complete.

Remark 13 (Thread groups). We need to be more precise about the occurrences of formulas in the conclusion sequent of preproof $[\![b]\!]$:

$$\underbrace{A, \ldots, A}_{2k}, B, C, D_2, D_m, \underbrace{E_m, \ldots, E_m}_{m}$$

Let us label the occurrences of A in this sequent as follows:

$$A_1^+, A_1^-, \ldots, A_k^+, A_k^-, B, C, D_2, D_m, \underbrace{E_m, \ldots, E_m}_{m}$$

so that we can talk precisely about them. It can be seen by examining the definition of $[\![\cdot]\!]$ (def. 55) that a valid thread in the preproof cannot pass through D_2 or D_m, which contain no ν, and that the remaining formulas are divided into $k+2$ groups

$$\underbrace{A_1^+, A_1^-}, \ldots, \underbrace{A_k^+, A_k^-}, \underbrace{B, C}, \underbrace{E_m, \ldots, E_m}$$

which cannot thread-interact with each other, in the sense that, for instance, no thread can contain a B and a E_m, or a A_ℓ^ϵ and a $A_{\ell'}^{\epsilon'}$ if $\ell \neq \ell'$.

Lemma 6. *An infinite branch $\overline{u} \in E^\omega$ in the preproof contains a validating thread*

- *in the E_m group iff. no suffix of $\overline{u}$ is a valid path in G.*
- *in the B, C group iff. 0 occurs only finitely in $\overline{u}$.*

Proof. The proof goes by case on the instructions involved.

In order to prove the first part of the statement, that is that an infinite branch $\overline{u} \in E^\omega$ in the preproof contains a validating thread in the E_m group iff. no suffix of $\overline{u}$ is a valid path in G, we reason by case on the instructions involved and remark that the E_m formulas are touched only in the $[\![\ell : \texttt{goto } \ell']\!]$ parts of the preproof.

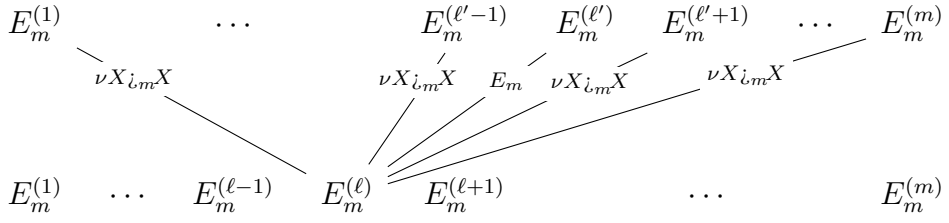
The E_m formulas are touched only in the $[\![\ell : \texttt{goto } \ell']\!]$ parts of the preproof. Observe that in $[\![\ell : \texttt{goto } \ell']\!]$ all E_m but one are erased, after which the remaining E_m recreates all of them. As a consequence, every infinite branch $\overline{u} \in E^\omega$ in the preproof has exactly one thread in the E_m group. Suppose a factor $e_1 e_2 \in E^2$ of $\overline{u}$ and suppose their respective sources and targets in G to be named $\ell_1 \xrightarrow{e_1} \ell_1'$ and $\ell_2 \xrightarrow{e_2} \ell_2'$. The crucial observation is that the piece of the E_m thread delimited by e_1 has minimal formula equal to

$$\begin{cases} E_m \text{ (which is a } \mu\text{-formula)} & \text{if } \ell_1' = \ell_2 \\ \nu X \between_m X & \text{if } \ell_1' \neq \ell_2 \end{cases}$$

the formula $\nu X \between_m X$ being a subformula of E_m able to recreate E_m. As a result:

- if $\overline{u}$ has a suffix which is a valid path in G then the principal formula of the E_m thread is E_m, hence this thread is a μ-thread, and there is no valid thread in the E_m group.
- otherwise, the principal formula of the E_m thread is $\nu X \between_m X$, and this thread is a ν-thread, infinitely progressing, hence a validating thread.

This diagram sums up the behavior of the threads of the E_m group in $[\![\ell : \texttt{goto } \ell']\!]$:



In order to prove the second part of the statement, that is that an infinite branch $\overline{u} \in E^\omega$ in the preproof contains a validating thread in the B, C group iff. 0 occurs only finitely in $\overline{u}$, we reason by case on the instructions involved.

First note that B is a ν-formula and that it is a subformula of C, which is a μ-formula. Observe also that B can recreate B, but B cannot recreate C. Therefore a $\{B, C\}$-thread, in order to be validating, must end with an infinite alternation $B \xrightarrow{(\nu)} (\bot \oplus B) \xrightarrow{(\oplus_1)} B \xrightarrow{(\nu)} \cdots$ Let us now take a closer look at what happens to B and C in the different branches of the tree with back edges p. Formulas B and C are touched only at the bottom of the $[\![\ell\colon \mathtt{I}_\ell]\!]$ parts of the preproof $[\![b]\!]$, in the following way:

- when going through branch 0, B is erased and C forks into a new copy of B and a recreated C.
- When going through any other branch, two threads are maintained: one at B and one at C, and the thread at B progresses.

Therefore: if an infinite branch $\overline{u}$ goes infinitely through 0, $\overline{u}$ contains no infinite thread at B, only an infinite thread at C, which is a μ-thread, hence $\overline{u}$ has no validating thread in the B/C part of the root sequent. If, on the contrary, $\overline{u}$ goes only finitely many times through 0, it has then a thread at B, which is a ν-thread, and which is infinitely progressing, hence a validating thread.

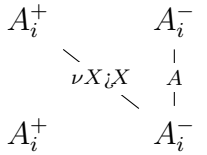
These two diagrams sum up the behavior of the threads in the $\{B, C\}$ group in $[\![\mathtt{0}\colon]\!]$ and in all other $[\![\ell\colon\ \mathtt{I}'_\ell]\!]$ respectively:



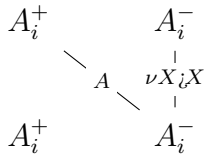
□

Remark 14. Because of lemma 6, the only infinite branches of $[\![b]\!]$ whose validity is not known in advance are the $\overline{u} \in E^\omega$ which are valid paths in G going infinitely many times through edge 0, and we know that these infinite branches may have validating threads only in one of the k groups $\{A_i^+, A_i^-\}_{1 \leqslant i \leqslant k}$. Such an infinite branch can always be factorized into $u_0 0 u_1 0 u_2 0 \cdots$ where the u_n do not contain 0. As the edge $0 \in E$ has source and target $0 \xrightarrow{} 1$, and because of the hypothesis that $\overline{u}$ is a path in G, for $n \geqslant 1$ every u_n has source and target $1 \xrightarrow{u_n}{}^+ 0$.

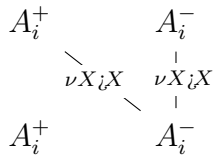
Lemma 7. *Assume* $1 \xrightarrow{u}{}^+ \ell$*, which does not contain the edge 0. If* u *is a prefix of the execution of* b *then the threads of* $\{A_i^+, A_i^-\}$ *in* $0 \xrightarrow{0u}{}^+ \ell$ *are*



if $\mathtt{X}_i = \mathit{false}$ *at the end of* u *and*

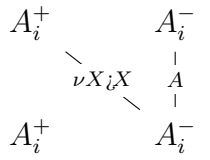


if $\mathtt{X}_i = \mathit{true}$ *at the end of* u*; and if* u *is not a prefix of the execution of* b *then there is an* $i \in [\![1, m]\!]$ *such that the threads of* $\{A_i^+, A_i^-\}$ *in* $0 \xrightarrow{0u}{}^+ \ell$ *are*

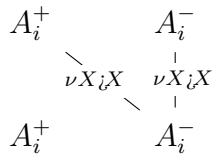


1 *Proof.* The proof goes by induction on the length of u.

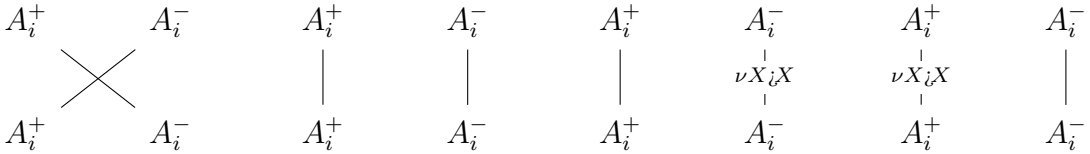
If u has length 0, $0u = 0$. It corresponds to having done 0 step of the execution of b, which implies that all variables have value *false*, and the threads of $\{A_i^+, A_i^-\}$ in p_0 are indeed



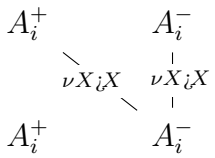
for all i. If u has length > 0, it decomposes as $u = ve$ with $e \in E \setminus \{0\}$. There are then three cases : either ve is the beginning of the execution of b, or v is but ve is not, or v is already not the beginning of the execution of b. In the third case, by induction hypothesis, there is an i such that the threads of $\{A_i^+, A_i^-\}$ in $0v$ are



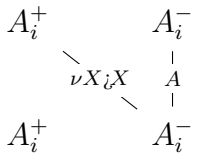
and, because $e \neq 0$, the threads of $\{A_i^+, A_i^-\}$ in e are of one of the four following forms:



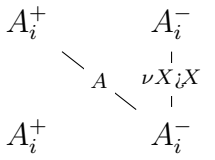
hence, by composition, the threads of $\{A_i^+, A_i^-\}$ in $0ve = 0u$ are still



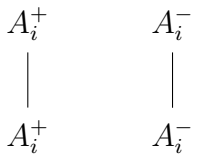
Otherwise v is a beginning of the execution of b. Then, by induction hypothesis, the threads of $\{A_i^+, A_i^-\}$ in $0 \xrightarrow{0v}{}^+ \ell$ are



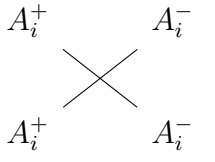
if $\mathtt{X}_i = \mathit{false}$ at the end of v and



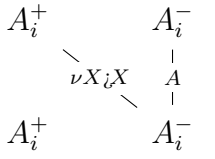
if $\mathtt{X}_i = \mathit{true}$ at the end of v. The next instruction executed by b is either $\mathtt{I}_\ell\mathtt{:X}_i := \mathtt{not\ X}_i$ or $\mathtt{I}_\ell\mathtt{:if\ X}_i\ \mathtt{goto}\ \ell'\ \mathtt{else}\ \ell''$. If the next instruction is $\mathtt{I}_\ell\mathtt{:X}_i := \mathtt{not\ X}_i$ then e has to be $\ell \xrightarrow{\ell} ((\ell + 1) \bmod (m + 1))$, so $0u = 0ve$ is still a beginning of the execution of b, and then: for every $j \neq i$, the threads of $\{A_j^+, A_j^-\}$ in e are



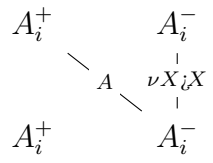
so the threads of $\{A_j^+, A_j^-\}$ in $0u$ have the same form as those in $0v$, while the value of X_j has not changed, hence the invariant is still respected for j. As for the threads in $\{A_i^+, A_i^-\}$, in e they are



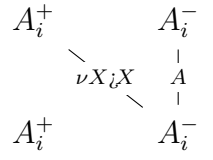
so if the threads of $\{A_i^+, A_i^-\}$ in $0v$ are



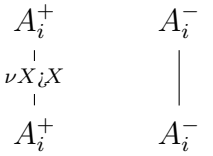
those in $0u$ are



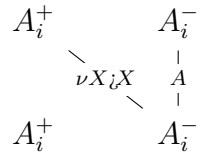
and *vice versa*. As the value of $\mathtt{X}_i$ is changed by this instruction, the invariant is still respected also for i. The last case we have to treat is when the next instruction executed by b after v is $\mathtt{I}_\ell\mathtt{:if\ X}_i\ \mathtt{goto}\ \ell'\ \mathtt{else}\ \ell''$. That means that v ends in vertex ℓ, from which there are two edges: $\ell \xrightarrow{\ell^+} \ell'$ and $\ell \xrightarrow{\ell^-} \ell''$. There are four cases, depending on the value of the variable $\mathtt{X}_i$ after v, and the choice of $e \in \{\ell^+, \ell^-\}$. If $\mathtt{X}_i = \mathit{false}$ and $e = l^-$ then $0u = 0ve$ is still a prefix of the execution of b. The threads of $\{A_i^+, A_i^-\}$ in $0v$ where, by induction hypothesis:



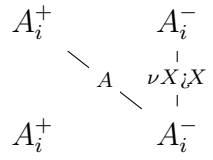
and those in $e = l^-$ are



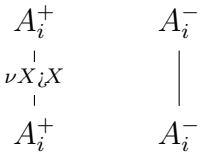
hence, by composition, the threads of $\{A_i^+, A_i^-\}$ in $0u$ are still



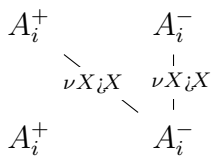
and the invariant is still respected. The situation is symmetric when $\mathtt{X}_i = \mathit{true}$ and $e = l^+$. If $\mathtt{X}_i = \mathit{true}$ and $e = l^-$ then $0u = 0ve$ is not a prefix of the execution of b. The threads of $\{A_i^+, A_i^-\}$ in $0v$ where, by induction hypothesis:



and those in $e = l^-$ are



hence, by composition, the threads of $\{A_i^+, A_i^-\}$ in $0u$ are



(recall that $\nu X¿X$ is a subformula of A). The situation is symmetric when $\mathtt{X}_i = \mathit{false}$ and $e = l^+$. So in all of these four cases the invariant is respected.

□

Proposition 10. *$[\![\cdot]\!]$ is a* LOGSPACE *reduction from $\overline{\Pi(\mu\mathrm{MALL}^\omega)}$ to $\mathcal{B}_{\mathit{false}}$.*

Proof. For the LOGSPACE character: the only data that need to be remembered while constructing the preproof are integers like k, m, ℓ, ℓ'. Because $\ell, \ell' \leqslant m$ and the entry has size $\Omega(k+m)$, this takes a space at most logarithmic in the size of the entry.

As for the fact that it is indeed a reduction: let us assume a $b \in \mathtt{BOOLE}_{\mathit{false}}$ and prove that $[\![b]\!] \notin \Pi(\mu\mathrm{MALL}^\omega) \Leftrightarrow b \in \mathcal{B}_{\mathit{false}}$. Let $G = (V, E)$ be the call-graph of b, as defined in Definition 54. Following remark 12, we denote by elements of E^ω the infinite branches of $[\![b]\!]$. There are two cases: either $b \in \mathcal{B}_{\mathit{false}}$ and we have to prove that $p \notin \Pi(\mu\mathrm{MALL}^\omega)$, or $b \notin \mathcal{B}_{\mathit{false}}$ and we have to prove that $p \in \Pi(\mu\mathrm{MALL}^\omega)$. First case: if $b \in \mathcal{B}_{\mathit{false}}$: the execution of b induces a finite path $u = 1 \to^* 0$ in G. This finite path can be completed into $v = 0 \xrightarrow{0} 1 \xrightarrow{u}{}^* 0$. Then v^ω is an invalid branch of $[\![p]\!]_\omega$. Here we use the fact that when b terminates, every variable has value *false*. Second case: if $b \notin \mathcal{B}_{\mathit{false}}$: let $\mathcal{P}_1 = \{vw_\infty \mid v \in E^*, w_\infty \in E^\omega \text{ and } w_\infty \text{ is a path in } G\}$ and $\mathcal{P}_2 = \{v_\infty \in \mathcal{P}_1 \mid 0 \text{ occurs infinitely in } v_\infty\}$. By construction, $\mathcal{P}_2 \subseteq \mathcal{P}_1 \subseteq E^\omega$. We will prove three facts: that every branch $v_\infty \in E^\omega \setminus \mathcal{P}_1$ is thread-valid, that every branch $v_\infty \in \mathcal{P}_1 \setminus \mathcal{P}_2$ is thread-valid and that every branch $v_\infty \in \mathcal{P}_2$ is thread-valid. These three facts, together with the fact that $(E^\omega \setminus \mathcal{P}_1) \cup (\mathcal{P}_1 \setminus \mathcal{P}_2) \cup \mathcal{P}_2 = E^\omega$, are enough to conclude that every branch $v_\infty \in E^\omega$ is thread-valid. The first fact, that every branch $v_\infty \in E^\omega \setminus \mathcal{P}_1$ is thread-valid, is due to the thread going through the E_m. The second fact, that every branch $v_\infty \in \mathcal{P}_1 \setminus \mathcal{P}_2$ is thread-valid, is due to the thread going through B. The third fact, that every branch $v_\infty \in \mathcal{P}_2$ is thread-valid, is due to the fact that b is non-terminating and that, because of that, one of the $2k$ threads going through the A is valid. □

Theorem 4. *The problem $\Pi(\mu\mathrm{MALL}^\omega)$ is* PSPACE*-hard:*

$$\mathrm{PSPACE} \leqslant \Pi(\mu\mathrm{MALL}^\omega)$$

Proof. We reduce from $\mathcal{B}_{\mathit{false}}$, which is PSPACE-complete by Lemma 2. More precisely, we reduce $\mathcal{B}_{\mathit{false}}$ to $\overline{\Pi(\mu\mathrm{MALL}^\omega)}$, the complement of $\Pi(\mu\mathrm{MALL}^\omega)$. This is enough because

PSPACE is closed under complements, in the same way as all deterministic classes. The reduction $[\![\cdot]\!]\colon \mathtt{BOOLE}_{false} \to \Pi_0(\mu\mathrm{MALL}^\omega)$ is defined in Definition 55. It is a LOGSPACE reduction, by Proposition 10, hence it is also a polynomial-time reduction. □

Remark 15. In fact, since our construction do not use the (cut) rule, the cut-free fragment of $\Pi(\mu\mathrm{MALL}^\omega)$ is already PSPACE-hard.

3.2.4 Generalisation to other systems

Systems of circular proofs with least and greatest fixed points can and have been built for other logics, either with different connectives (LK, LJ) or with added modalities (LK◯, LK□◊). Such systems, which can also be seen as variants of μ-calculus, are described in [Vardi, 1988, Kaivola, 1995a, Dax, Hofmann, and Lange, 2006, Doumane, Baelde, Hirschi, and Saurin, 2016, Doumane, 2017a,b]. They all share the fact that MALL can be seen as a fragment of them, and therefore it is easy to translate μMALL$^\omega$ proofs into them and transfer to them our result of PSPACE-completeness:

Remark 16. There is a μLJ defined by Clairambault [2009, 2010], which is a finitary calculus in the spirit of μMALL, with explicit invariants of induction and coinduction. The $\mu\mathrm{LJ}^\omega$ we are considering here is an infinitary one, in which proofs are distinguished from invalid preproofs by a thread criterion, in the spirit of $\mu\mathrm{MALL}^\omega$.

Corollary 2 (to Theorem 4)**.** *The decidability of the thread criterion is also* PSPACE*-complete in* $\mu\mathrm{LJ}^\omega$*,* $\mu\mathrm{LK}^\omega$*,* $\mu\mathrm{LK}\bigcirc^\omega$ *and* $\mu\mathrm{LK}\Box\Diamond^\omega$*.*

Proof. It is true for $\mu\mathrm{LJ}^\omega$ because every preproof π in $\mu\mathrm{MALL}^\omega$ can be translated to a preproof π^* in $\mu\mathrm{LJ}^\omega$, such that π^* is thread-valid iff. π is, by translating MALL connectives into LJ connectives and translating $\vdash \Gamma$ into $|\Gamma^\perp| \vdash \perp$.

It is true for the other mentionned systems because $\mu\mathrm{LJ}^\omega$ is a fragment of them, therefore every $\mu\mathrm{LJ}^\omega$ preproof is already a preproof in each of them.

Details of how the constructions adapt to LK and LJ are given below. □

Remark 17. Note that the following adaptations from MALL to LK and LJ might be simplified by replacing $F \vee F$ by F, $X \wedge X$ by X, $\perp \vee X$ by X and so on. A lot of the complexity of these formulas is there to simulate structural rules in the linear world. When we are in the classical world, these workaround may not be needed anymore. Anyway, the translation we present here is simply to show that even without thinking about that,

a straightforward translation is possible, which gives the same PSPACE-completeness result with LK and LJ and their extensions.

Generalisation to μLK, μLK$\bigcirc$ and μLK$\Box\Diamond$

$$A = ¿(\nu X ¿X) \qquad B = \nu X(\bot \vee X) \qquad C = \mu X(B \vee X) \qquad E_n = ¿_n(\nu X ¿_n X)$$

$$¿F = \mu X(F \vee (\bot \vee (X \vee X))) \qquad ¿_n F = \mu X(\bot \vee (X \vee (\underbrace{F \vee \cdots \vee F}_{n-1})))$$

$$D_n = \mu X(\underbrace{X \wedge \cdots \wedge X}_{n})$$

We define $[\![b]\!]$ to be the preproof

$$\dfrac{\begin{array}{ccccc}\vdots\,[\![\mathtt{0}:]\!] & & \vdots\,[\![\mathtt{1}:\mathtt{I}_1]\!] & \cdots & \vdots\,[\![\mathtt{m}:\mathtt{I}_\mathtt{m}]\!] \\ \vdash A^{2k}, B, C, D_2, D_m, E_m^m & \vdash A^{2k}, B, C, D_2, D_m, E_m^m & & & \vdash A^{2k}, B, C, D_2, D_m, E_m^m\end{array}}{(\textsc{Root}) \vdash A^{2k}, B, C, D_2, \underline{D_m}, E_m^m}\,{\scriptstyle (\mu),\,(\wedge)^{m-1}} \tag{3.4}$$

See the details of all branches on Figures 3.7 and 3.8.

Generalisation to μLJ

$$A = ¡(\mu X ¡X) \qquad B = \mu X(\top \wedge X) \qquad C = \nu X(B \wedge X) \qquad E_n = ¡_n(\mu X ¡_n X)$$

$$¡F = \nu X(F \wedge (\top \wedge (X \wedge X))) \qquad ¡_n F = \nu X(\top \wedge (X \wedge (\underbrace{F \wedge \cdots \wedge F}_{n-1})))$$

$$D_n = \nu X(\underbrace{X \vee \cdots \vee X}_{n})$$

We define $[\![b]\!]$ to be the preproof

$$\dfrac{\begin{array}{ccccc}\vdots\,[\![\mathtt{0}:]\!] & & \vdots\,[\![\mathtt{1}:\mathtt{I}_1]\!] & \cdots & \vdots\,[\![\mathtt{m}:\mathtt{I}_\mathtt{m}]\!] \\ A^{2k}, B, C, D_2, D_m, E_m^m \vdash & A^{2k}, B, C, D_2, D_m, E_m^m \vdash & & & A^{2k}, B, C, D_2, D_m, E_m^m \vdash\end{array}}{(\textsc{Root}) A^{2k}, B, C, D_2, \underline{D_m}, E_m^m \vdash}\,{\scriptstyle (\nu^L),\,(\vee^L)^{m-1}} \tag{3.5}$$

See the details of all branches on Figures 3.9 and 3.10.

Remark 18. The interested open question is: does the result still stand in the guarded fragments of $\mu\mathrm{LK}\bigcirc^{\omega}$ and $\mu\mathrm{LK}\Box\Diamond^{\omega}$? We conjecture that it does and that the method we illustrate here on μMALL can apply as well to the guarded cases of μ-calculi with modalities.

3.3 Comments on our approach and discussion of related works

Our proof for the PSPACE-completeness of the thread criterion is an encoding and an adaptation to our setting of the proof used by Lee, Jones and Ben Amram to prove that size-change termination is PSPACE-complete [Lee, Jones, and Ben-Amram, 2001]. We reduce, as they do, from the problem of termination of boolean programs and the thread diagrams that we have used to describe the preproof generated by the reduction are very similar to the size-change graphs generated by their reduction; this is in fact what has guided the design of this preproof: formula A mimicks the $\mathtt{X}_i, \overline{\mathtt{X}}_i$ part of their graphs and formulas B and C adapt the $\mathtt{Z}$ part of their graphs. We had to add the formulas D_2 and D_m in order to have branching rules in the preproof. One of the main novelties of our reduction, compared to the reduction of Lee, Jones and Ben Amram for size-change termination, lies in the E_m and $[\![\ell : \mathtt{goto}\ \ell']\!]$ part of the constructed preproof, which has no equivalent in the size-change graphs obtained by their reduction. This part of our construction allows us to construct a preproof which is a tree with back edges, hence proving that the thread criterion is PSPACE-complete even when preproofs are represented by trees with back edges. We could in fact drop the E_m and $[\![\ell : \mathtt{goto}\ \ell']\!]$ part of the construction by constructing $[\![b]\!]$ as a rooted graph instead of a tree with back edges. The constructions proofs are still correct — and shorter. The caveat is that it only proves the thread-criterion to be PSPACE-hard in graph-shaped preproofs and not in b.e.-tree-shaped preproofs. Furthermore, we could not have filled this gap by simply unfolding the graph into a tree with back edges, for it could lead, as shown in the following example, to an exponential blow-up in size, which would prevent the reduction to be LOGSPACE, or even P. The following boolean program:

$$
\begin{array}{l}
1{:}\mathtt{if\ X\ then\ goto}\ 2\ \mathtt{else\ goto}\ 2 \\
2{:}\mathtt{if\ X\ then\ goto}\ 3\ \mathtt{else\ goto}\ 3 \\
\qquad\qquad \vdots \\
n{:}\mathtt{if\ X\ then\ goto}\ n+1\ \mathtt{else\ goto}\ n+1
\end{array}
$$

will be translated to a graph-shaped preproof of size $\Theta(n)$ but the unfolding of this preproof into a b.e.-tree-shaped preproof will have size $\Theta(2^n)$. Therefore we had to be

clever in order to target trees with back edges by *simulating* several vertices with a single one; this is accomplished by the E_m and $[\![\ell : \texttt{goto } \ell']\!]$.

This improvement of the reduction of Lee, Jones and Ben Amram could in fact be adapted in the other direction, to show that size-change termination is already PSPACE-complete even when restricted to programs with only one function (in the terminology of [Lee, Jones, and Ben-Amram, 2001]), that is when the corresponding call graph / control flow graph has only one vertex.

If, as it is commonly believed, NP $\neq$ PSPACE, our result implies that there is no way to add a polynomial quantity of information to a preproof so that its thread-validity can be checked in polynomial time. This can be seen as a problem, both for the complexity of proof search and proof verification. It suggests trying to find restrictions of the thread criterion which will be either decidable or certifiable in polynomial time, while keeping enough expressivity to validate interesting proofs. This will be the subject of the Chapters 4 and 5 of this thesis.

We recalled in section 3.1 that thread validity is decidable in PSPACE, and we did so by reducing to the problem of language inclusion for ω-parity-automata. The original size-change article [Lee, Jones, and Ben-Amram, 2001] gives two different methods to check size-change termination, the first one is based on reducing to inclusions of ω-languages defined by finite automata while the second one is a direct, graph-based approach. It is in fact possible to use this more direct method to decide the thread criterion, and this has already been done by Dax, Hofmann, and Lange [2006], who remark furthermore that this method leads to a more efficient implementation than the automata-based one.

3.4 Conclusion

In the present chapter, we analyzed the complexity of deciding the validity of circular proofs in μMALL logic: while the problem was already known to be in PSPACE, we established here its PSPACE-completeness. In doing so, we drew inspiration from the PSPACE-completeness proof of SCT even though we defer at some crucial points on order to build our reduction and carry our proof taking into account the specific forms of circular proofs.

Our proof adapt straightforwardly to a number of other circular proof systems based on sequent calculus such as intutionnistic or classical proof systems in addition to the linear case on which we focused here.

While our result can be seen as negative one for circular proofs, it does not prevent actual implementations to be tractable and usable on many situations as examplified by

the Cyclist prover for instance. In such systems, validity checking does not seem to be the bottleneck in circular proof construction as compared with the complexity that is inherent to exploring and backtracking in the search tree [Brotherston, Gorogiannis, and Petersen, 2012, Rowe and Brotherston, 2017, Tellez and Brotherston, 2017].

Our work suggests deep connections between thread-validity and SCT that we only touched upon in the previous section. This confirms connections previously hinted by other authors [Dax, Hofmann, and Lange, 2006, Hyvernat, 2014, 2019, Lepigre and Raffalli, 2019] that we plan to investigate further in the future.

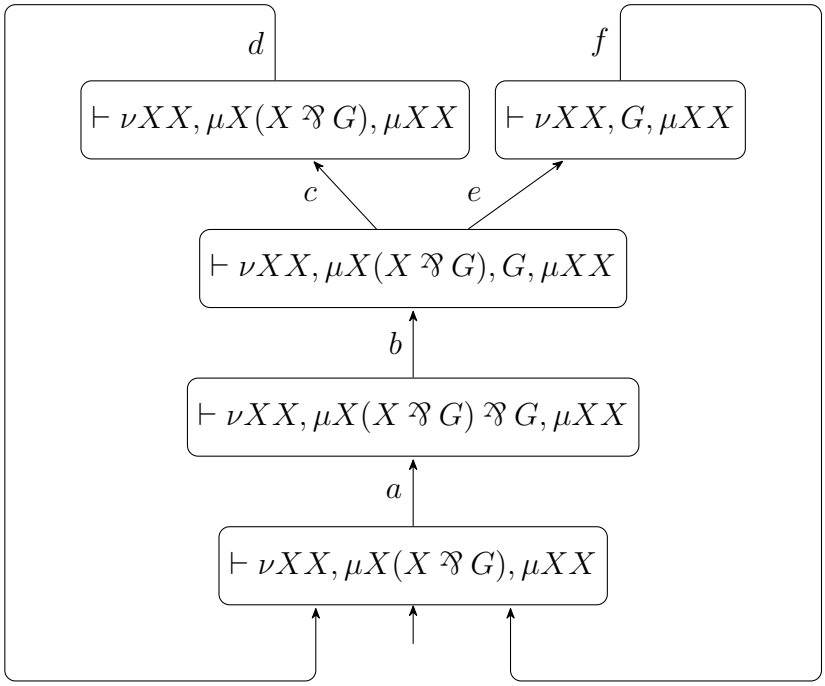


Figure 3.1. Automaton recognizing the branches of the preproof (3.1).

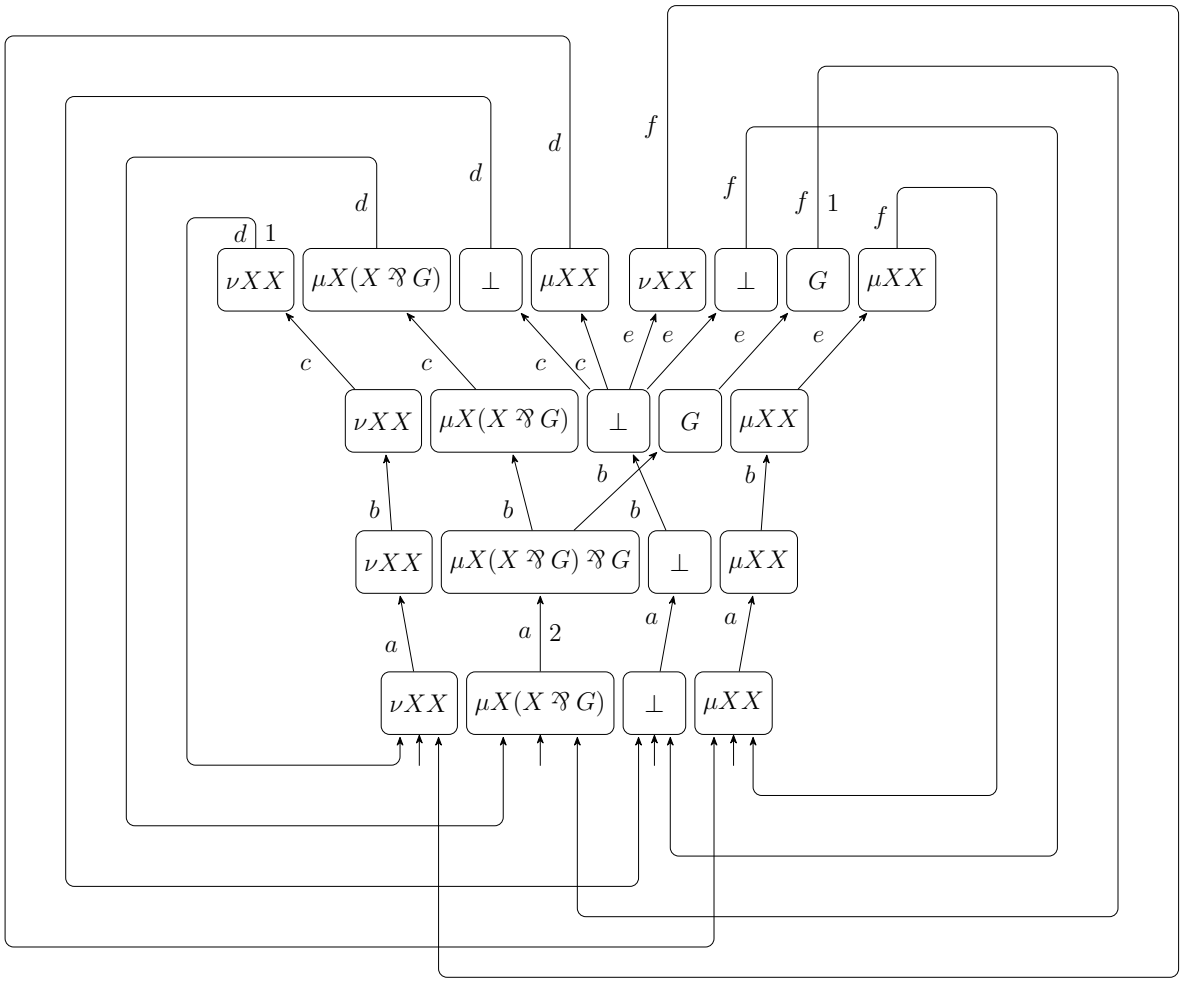


Figure 3.2. Automaton recognizing the valid branches of the preproof (3.1).

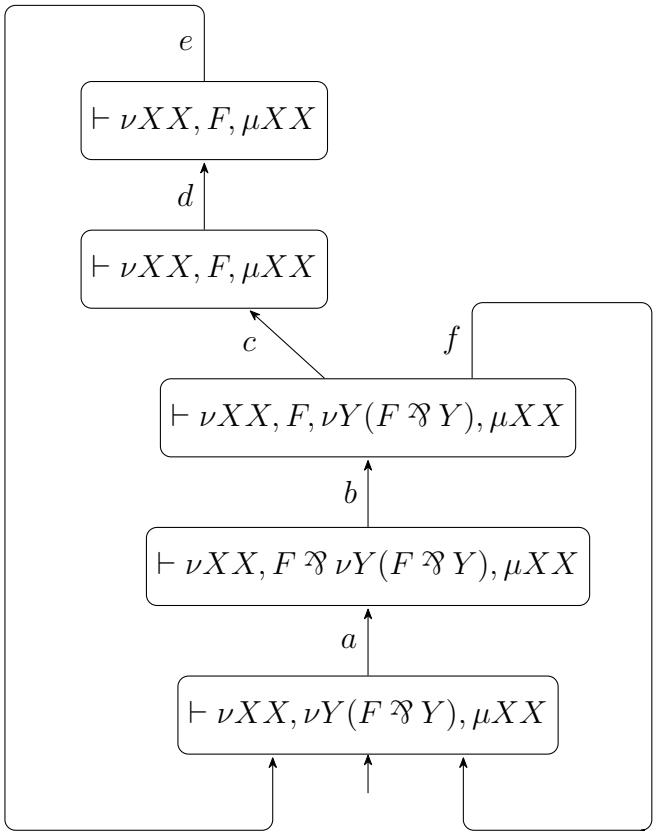


Figure 3.3. Automaton recognizing the branches of the preproof (3.2).

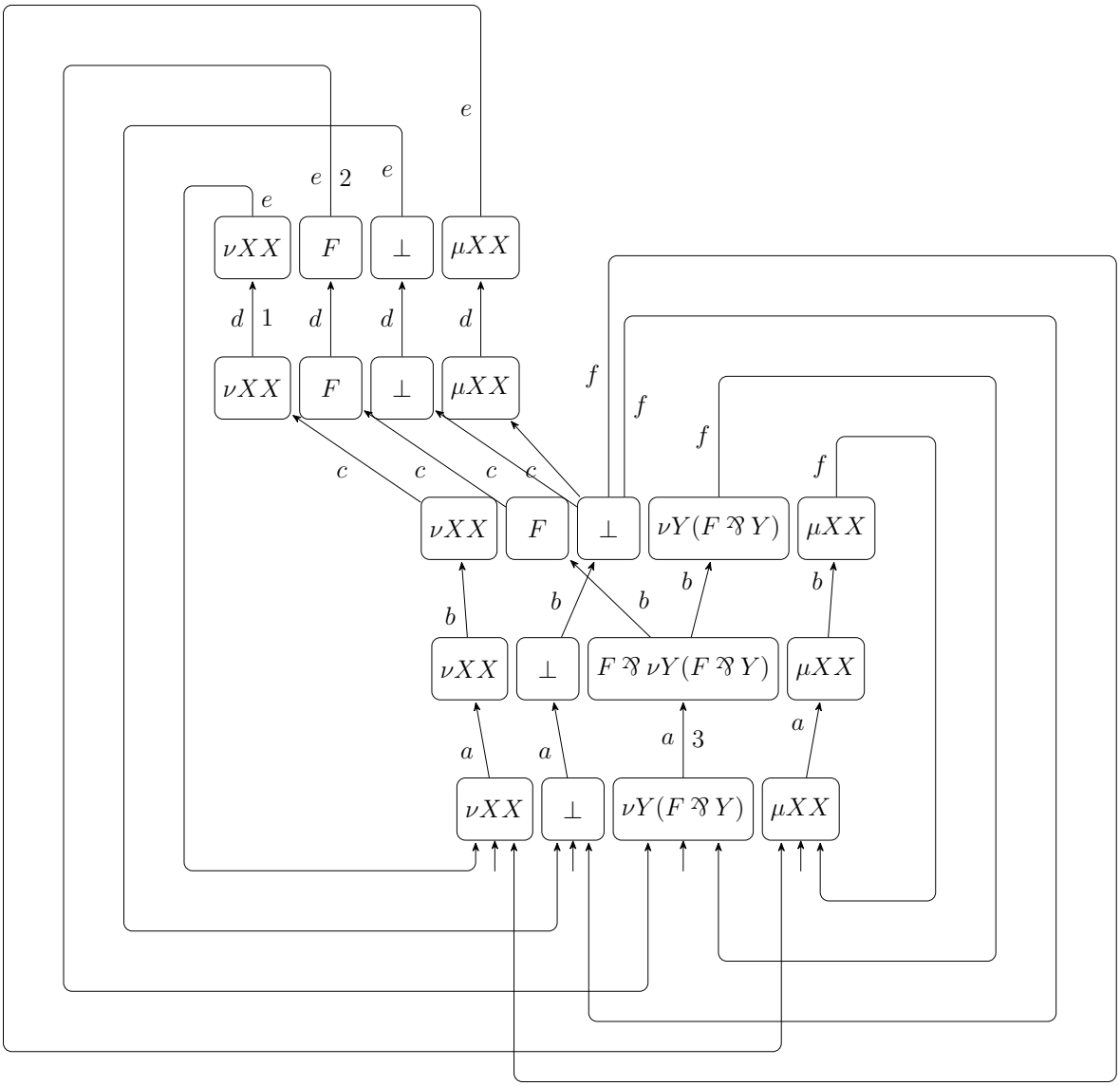


Figure 3.4. Automaton recognizing the valid branches of the preproof (3.2).

$$[\![\ell\texttt{:goto } \ell']\!] \quad :=$$

$$\dfrac{\dfrac{\text{Back edge to (Root)}}{\vdash A,\dots,A,B,C,D_2,D_m,E_m,\dots,E_m}}{\vdash A,\dots,A,B,C,D_2,D_m,\underbrace{\nu X \text{¿}_m X,\dots,\nu X \text{¿}_m X}_{\ell'-1},E_m,\underbrace{\nu X \text{¿}_m X,\dots,\nu X \text{¿}_m X}_{m-\ell'}}(\nu)^{m-1}$$

$$\dfrac{\vdash \underbrace{A,\dots,A}_{2k},B,C,D_2,D_m,\underbrace{E_m,\dots,E_m}_{\ell-1},E_m,\underbrace{E_m,\dots,E_m}_{m-\ell}}{\vdash A,\dots,A,B,C,D_2,D_m,E_m}\big((\mu),(\oplus_0),(\bot)\big)^{m-1}$$

Figure 3.5. Back edges of the preproof

$$[\![\texttt{0:}]\!] \quad := \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots\ [\![\texttt{0:goto 1}]\!]}{\vdash (A,A)^k,B,C,D_2,D_m,E_m^m}(\nu)^k}{\vdash (\nu X \text{¿} X,A)^k,B,C,D_2,D_m,E_m^m}\big((\mu),(\oplus_0)\big)^k}{\vdash (A,A)^k,B,C,D_2,D_m,E_m^m}\big((\mu),(\oplus_1),(\oplus_1),(\parr)\big)^k}{\vdash A^k,B,C,D_2,D_m,E_m^m}\big((\mu),(\oplus_1),(\oplus_0),(\bot)\big)^k}{\vdash (A,A)^k,B,C,D_2,D_m,E_m^m}(\mu),(\parr)}{\vdash A^{2k},C,D_2,D_m,E_m^m}(\nu),(\oplus_0),(\bot)}{\vdash A^{2k},B,C,D_2,D_m,E_m^m}$$

$$[\![\ell\texttt{:X}_i\texttt{ := not X}_i]\!] \quad := \quad \dfrac{\vdots\ [\![\ell\texttt{:goto } (\ell+1 \bmod m+1)]\!]}{\vdash A^{2(i-1)},A,A,A^{2(k-i)},B,C,D_2,D_m,E_m^m}(\nu),(\oplus_1)$$
$$\vdash A^{2(i-1)},A,A,A^{2(k-i)},B,C,D_2,D_m,E_m^m$$

$$[\![\ell\texttt{:if X}_i\texttt{ then goto } \ell' \texttt{ else } \ell'']\!] \quad :=$$

$$\dfrac{\dfrac{\dfrac{\vdots\ [\![\ell\texttt{:goto } \ell']\!]}{\vdash A^{2(i-1)},A,A,A^{2(k-i)},B,C,D_2,D_m,E_m^m}(\nu)}{\vdash A^{2(i-1)},A,\nu X \text{¿} X,A^{2(k-i)},B,C,D_2,D_m,E_m^m}(\mu),(\oplus_0)}{\vdash A^{2(i-1)},A,A,A^{2(k-i)},B,C,D_2,D_m,E_m^m} \quad \dfrac{\dfrac{\dfrac{\vdots\ [\![\ell\texttt{:goto } \ell'']\!]}{\vdash A^{2(i-1)},A,A,A^{2(k-i)},B,C,D_2,D_m,E_m^m}(\nu)}{\vdash A^{2(i-1)},\nu X \text{¿} X,A,A^{2(k-i)},B,C,D_2,D_m,E_m^m}(\mu),(\oplus_0)}{\vdash A^{2(i-1)},A,A,A^{2(k-i)},B,C,D_2,D_m,E_m^m}$$

$$\dfrac{\vdash A^{2k},B,C,D_2,D_m,E_m^m}{\vdash A^{2k},B,C,D_2,D_m,E_m^m}(\nu),(\oplus_1)$$

Figure 3.6. Premises p_ℓ of the preproof

$$
[\![\texttt{0:}]\!] \quad := \quad
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\vdots\ [\![\texttt{0:goto 1}]\!]
}{\vdash (A,A)^k, B, C, D_2, D_m, E_m^m}\,(\nu)^k
}{\vdash (\nu X \text{¿} X, A)^k, B, C, D_2, D_m, E_m^m}\,\big((\mu),(\vee_0)\big)^k
}{\vdash (\underline{A},A)^k, B, C, D_2, D_m, E_m^m}\,\big((\mu),(\vee_1),(\vee_1),(\vee)\big)^k
}{\vdash \underline{A}^k, B, C, D_2, D_m, E_m^m}\,\big((\mu),(\vee_1),(\vee_0),(\bot)\big)^k
}{\vdash (\underline{A},A)^k, B, C, D_2, D_m, E_m^m}\,(\mu),(\vee)
}{\vdash A^{2k}, \underline{C}, D_2, D_m, E_m^m}\,(\nu),(\vee_0),(\bot)
}{\vdash A^{2k}, \underline{B}, C, D_2, D_m, E_m^m}
$$

$$
[\![\ell\texttt{:X}_i\ \texttt{:= not X}_i]\!] \quad := \quad
\dfrac{
\vdots\ [\![\ell\texttt{:goto }(\ell+1 \bmod m+1)]\!]
}{\vdash A^{2(i-1)}, \underline{A}, \underline{A}, A^{2(k-i)}, B, C, D_2, D_m, E_m^m}\,(\nu),(\vee_1)
$$

$$
[\![\ell\texttt{:if X}_i\ \texttt{then goto }\ell'\ \texttt{else }\ell'']\!] \quad :=
$$

$$
\dfrac{
\dfrac{
\dfrac{
\vdots\ [\![\ell\texttt{:goto }\ell']\!]
}{\vdash A^{2(i-1)}, A, A, A^{2(k-i)}, B, C, D_2, D_m, E_m^m}\,(\nu)
}{\vdash A^{2(i-1)}, A, \underline{\nu X(\text{¿}X)}, A^{2(k-i)}, B, C, D_2, D_m, E_m^m}\,(\mu),(\vee_0)
}{\vdash A^{2(i-1)}, A, \underline{A}, A^{2(k-i)}, B, C, D_2, D_m, E_m^m}
\qquad
\dfrac{
\dfrac{
\vdots\ [\![\ell\texttt{:goto }\ell'']\!]
}{\vdash A^{2(i-1)}, A, A, A^{2(k-i)}, B, C, D_2, D_m, E_m^m}\,(\nu)
}{\vdash A^{2(i-1)}, \underline{\nu X(\text{¿}X)}, A, A^{2(k-i)}, B, C, D_2, D_m, E_m^m}\,(\mu),(\vee_0)
}{\vdash A^{2(i-1)}, \underline{A}, A, A^{2(k-i)}, B, C, D_2, D_m, E_m^m}\,(\mu),(\wedge)
}{\vdash A^{2k}, B, C, \underline{D_2}, D_m, E_m^m}\,(\nu),(\vee_1)
}{\vdash A^{2k}, \underline{B}, C, D_2, D_m, E_m^m}
$$

Figure 3.7. Premises p_ℓ of the preproof for $\mu\mathrm{LK}^\omega$

$$
[\![\ell\texttt{:goto }\ell']\!] \quad :=
$$

Back edge to (Root)

$$
\dfrac{
\dfrac{
\vdash A, \ldots, A, B, C, D_2, D_m, E_m, \ldots, E_m
}{\vdash A, \ldots, A, B, C, D_2, D_m, \underbrace{\nu X \text{¿}_m X, \ldots, \nu X \text{¿}_m X}_{\ell'-1}, E_m, \underbrace{\nu X \text{¿}_m X, \ldots, \nu X \text{¿}_m X}_{m-\ell'}}\,(\nu)^{m-1}
}{\vdash \underbrace{A, \ldots, A}_{2k}, B, C, D_2, D_m, \underbrace{\underline{E_m}, \ldots, \underline{E_m}}_{\ell-1}, E_m, \underbrace{\underline{E_m}, \ldots, \underline{E_m}}_{m-\ell}}\,(\mu),(\vee_1),(\vee)^{m-1}
}{\vdash A, \ldots, A, B, C, D_2, D_m, \underline{E_m}}\,\big((\mu),(\vee_0),(\bot)\big)^{m-1}
$$

Figure 3.8. Back edges of the preproof for $\mu\mathrm{LK}^\omega$

$$[\![\texttt{0:}]\!] \quad := \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots\ [\![\texttt{0:goto 1}]\!]}{(A,A)^k, B, C, D_2, D_m, E_m^m \vdash}(\mu^L)^k}{(\underline{\mu X ¡ X}, A)^k, B, C, D_2, D_m, E_m^m \vdash}\big((\nu^L),(\wedge_0^L)\big)^k}{(\underline{A}, A)^k, B, C, D_2, D_m, E_m^m \vdash}\big((\nu^L),(\wedge_1^L),(\wedge_1^L),(\wedge^L)\big)^k}{\underline{A}^k, B, C, D_2, D_m, E_m^m \vdash}\big((\nu^L),(\wedge_1^L),(\wedge_0^L),(\top^L)\big)^k}{(\underline{A}, A)^k, B, C, D_2, D_m, E_m^m \vdash}(\nu^L),(\wedge^L)}{A^{2k}, \underline{C}, D_2, D_m, E_m^m \vdash}(\mu^L),(\wedge_0^L),(\top^L)}{A^{2k}, \underline{B}, C, D_2, D_m, E_m^m \vdash}$$

$$[\![\ell\texttt{:X}_i\ \texttt{:= not X}_i]\!] \quad := \quad \dfrac{\dfrac{\vdots\ [\![\ell\texttt{:goto }(\ell+1 \bmod m+1)]\!]}{A^{2(i-1)}, \underline{A}, A, A^{2(k-i)}, B, C, D_2, D_m, E_m^m \vdash}(\mu^L),(\wedge_1^L)}{A^{2(i-1)}, \underline{A}, A, A^{2(k-i)}, \underline{B}, C, D_2, D_m, E_m^m \vdash}$$

$$[\![\ell\texttt{:if X}_i\ \texttt{then goto } \ell'\ \texttt{else } \ell'']\!] \quad :=$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots\ [\![\ell\texttt{:goto } \ell']\!]}{A^{2(i-1)}, A, A, A^{2(k-i)}, B, C, D_2, D_m, E_m^m \vdash}(\mu^L)}{A^{2(i-1)}, A, \underline{\mu X(¡X)}, A^{2(k-i)}, B, C, D_2, D_m, E_m^m \vdash}(\nu^L),(\wedge_0^L)}{A^{2(i-1)}, A, \underline{A}, A^{2(k-i)}, B, C, D_2, D_m, E_m^m \vdash} \quad \dfrac{\dfrac{\dfrac{\vdots\ [\![\ell\texttt{:goto } \ell'']\!]}{A^{2(i-1)}, A, A, A^{2(k-i)}, B, C, D_2, D_m, E_m^m \vdash}(\mu^L)}{A^{2(i-1)}, \underline{\mu X(¡X)}, A, A^{2(k-i)}, B, C, D_2, D_m, E_m^m \vdash}(\nu^L),(\wedge_0^L)}{A^{2(i-1)}, \underline{A}, A, A^{2(k-i)}, B, C, D_2, D_m, E_m^m \vdash}$$
$$(\nu^L),(\vee^L)$$
$$\dfrac{A^{2k}, B, C, \underline{D_2}, D_m, E_m^m \vdash}{A^{2k}, \underline{B}, C, D_2, D_m, E_m^m \vdash}(\mu^L),(\wedge_1^L)$$

Figure 3.9. Premises p_ℓ of the preproof for $\mu\mathrm{LJ}^\omega$

$[\![\ell\texttt{:goto } \ell']\!] \quad :=$

Back edge to (Root)

$$\dfrac{\dfrac{A, \ldots, A, B, C, D_2, D_m, E_m, \ldots, E_m \vdash}{A, \ldots, A, B, C, D_2, D_m, \underbrace{\underline{\mu X ¡_m X}, \ldots, \underline{\mu X ¡_m X}}_{\ell'-1}, E_m, \underbrace{\underline{\mu X ¡_m X}, \ldots, \underline{\mu X ¡_m X}}_{m-\ell'} \vdash}(\mu^L)^{m-1}}{\dfrac{A, \ldots, A, B, C, D_2, D_m, \underline{E_m} \vdash}{\underbrace{A, \ldots, A}_{2k}, B, C, D_2, D_m, \underbrace{\underline{E_m}, \ldots, \underline{E_m}}_{\ell-1}, E_m, \underbrace{\underline{E_m}, \ldots, \underline{E_m}}_{m-\ell} \vdash}\big((\nu^L),(\wedge_0^L),(\top^L)\big)^{m-1}}(\nu^L),(\wedge_1^L),(\wedge^L)^{m-1}$$

Figure 3.10. Back edges of the preproof for $\mu\mathrm{LJ}^\omega$

4 A polynomial sub-criterion

We have shown in the previous chapter that deciding the validity of the circular representation of a preproof, with respect to the thread criterion, is a PSPACE-complete problem.

This implies in particular that there is probably no subexponential algorithm to check the validity of a circular representation. This also implies that there is probably no way to certify the validity of a circular preproof, so that it can be checked in polynomial time, without adding to it an exponential quantity of information.

We tackle both these problems by taking a different approach. Instead of trying to handle the existing criterion, we will look for a new criterion.

The thread criterion has very good properties. In particular, Baelde, Doumane, and Saurin [2016] proved that it guarantees soundness with respect to boolean interpretation and full cut elimination. Therefore, we would like our new criterion to have the following properties:

- that any preproof valid for the new criterion is also valid for the thread criterion, so that it inherits its soundness properties,
- that this new criterion is decidable in reasonable time,
- that it is possible to add a reasonable amount of information to the circular representation of a preproof so that the validity of this certified preproof may be checked in linear time,
- that this new criterion is expressive enough.

In this chapter, we provide such a criterion, which we called the loop criterion.

There is an essential difference between the thread criterion and our loop criterion. In usual proof theory, a proof is a finite tree, made of logical inferences, and a proof is correct as soon as each one of its inferences is correct. This is not true anymore with logics such as μMALL^ω, in which the soundness of each inference only ensure that the tree is a preproof, and a proof also has to pass the thread criterion, which is a global

criterion. The PSPACE-complete nature of the thread criterion forbids that it could have a local definition. It is, and has to be, a *global* criterion, which cannot be turned into a local criterion. In contrast, our loop criterion gives rise to a proof system in which the validity of a proof is a purely local property. More precisely, with our loop criterion comes an enriched proof system, made of labelled proof trees, with two properties:

1. if the circular representation of a preproof is valid for our loop criterion, it is possible to label it so as to obtain a proof in our labelled proof system,

2. in this labelled proof system, a proof is correct if and only if each one of its individual inferences is correct.

This would not be possible with the thread criterion, or the labeling would probably need to be at least exponential in the size of the proof.

There is another important difference between the usual thread criterion and our new loop criterion. The thread criterion is really a criterion on the infinite preproof, not on its representation. This means that if you have two different circular representations of the same infinite preproof, either they are both valid or both invalid. In contrast, our criterion really depends on the finite, circular representation, and not only on the infinite preproof which is represented. This means that among the circular representations of a given infinite preproof, it may happen that some of them are valid for our loop criterion and some other are not.

In this chapter we prove that:

- If a circular representation of an infinite preproof is valid for our loop criterion then this preproof is valid for the thread criterion.

- Our loop criterion can be decided in quadratic time.

- Every circular representation which is valid for our loop criterion can be labelled so as to be turned into a proof in our labelled proof system, in which the validity of the proof only depends on the validity of each individual inference, and is therefore checkable in linear time.

- There is a canonical translation of the finitary proofs in the style of Baelde [2012] into circular representations of infinite proofs, which are valid for the thread criterion. Those representations are also valid for our more restricted loop criterion. In other words, our labelling method capture the image of all finitary μMALL proofs.

4.1 $\mathcal{L}$-proofs: labelling as validity

In this subsection, we briefly mention an alternative approach to ensure validity of μMALL$^\infty$ preproofs, aiming at motivating the tools used in the remainder of this chapter. The idea is to witness thread progress by adding labels on some formulas, much in the spirit of Stirling [2014], Afshari and Leigh [2017].

Definition 56 (Labelled formulas)**.** Let $\mathcal{L}$ be an infinite countable set of *atoms* and call *labels* any finite list of atoms. Let $\mathcal{F}^{\mathcal{L}} := \left\{\sigma^L \mid \sigma \in \{\mu, \nu\}, L \in \mathrm{list}(\mathcal{L})\right\}$. *Labelled formulas*, or $\mathcal{L}$-formulas, are defined as μMALL formulas, by replacing $\mathcal{F} = \{\mu, \nu\}$ with $\mathcal{F}^{\mathcal{L}}$ in the grammar of formulas, in Definition 12. Negation is lifted to labelled formulas, as $(\mu^L XA)^\perp = \nu^L XA^\perp$. We write σXA for $\sigma^\emptyset XA$, and standard, unlabelled formulas can thus be seen as labelled formulas where every label is empty. We define a *label-erasing* function $\lceil\bullet\rceil$ that associates to every $\mathcal{L}$-formula A the μMALL-formula $\lceil A\rceil$ obtained by erasing every label and satisfying $\lceil\sigma^L XB\rceil = \sigma X\lceil B\rceil$.

The standard μMALL$^\infty$ proof system is adapted, to handle labels, by updating the (id) and (ν) rules of Figures 2.1 and 2.4 as

$$\frac{\lceil A\rceil = \lceil B\rceil^\perp}{\vdash A, B}\,(\mathrm{id}') \qquad\qquad \frac{\vdash A[\nu^{L,a} XA[X]], \Gamma}{\vdash \nu^L XA[X], \Gamma}\,(\nu_{\mathfrak{b}}(a))$$

where, in ($\nu_{\mathfrak{b}}(a)$), a must be a fresh label name, meaning that a does not appear free in the conclusion sequent of ($\nu_{\mathfrak{b}}(a)$) (in particular, $a \notin L$).

Since we are in a one-sided framework, only labels on ν operators are relevant. Therefore, from now on, formulas have non-empty labels only on ν and we require, for the cut inference, that all labels of cut formulas are empty.

Definition 57. *$\mathcal{L}$-preproofs* are, as in Definition 18, possibly infinite derivations using $\mathcal{L}$-formulas, with the constraint that for every two inferences ($\nu_{\mathfrak{b}}(a)$) and ($\nu_{\mathfrak{b}}(b)$), if one is above the other, then $a \neq b$.

And the validity condition is expressed in terms of labels in the following definition:

Definition 58 ($\mathcal{L}$-proof)**.** An *$\mathcal{L}$-proof* is an $\mathcal{L}$-preproof such that for every infinite branch $\gamma = (s_i)_{i\in\omega}$ (seen as a sequence of sequent occurrences), there exists a sequence $(\nu^{L_i} XG_i)_{i\in\omega}$ and a strictly increasing function ϵ on natural numbers such that for every $i \in \omega$,

1. the formula $\nu^{L_i} X G_i$ is principal in $s_{\epsilon(i)}$
2. $\lceil \nu^{L_i} X G_i \rceil = \lceil \nu^{L_{i+1}} X G_{i+1} \rceil$ and
3. $\forall i \geqslant 1, L_i = (L_{i-1}, a_i)$ for some $a_i \in \mathcal{L}$.

The label-erasing function $\lceil \bullet \rceil$ is easily lifted to sequents and to $\mathcal{L}$-preproofs. We then have the following property:

Proposition 11. *If π is an $\mathcal{L}$-proof, then $\lceil \pi \rceil$ is a μMALL$^\infty$ proof.*

Proof. First, $\lceil \pi \rceil$ is a μMALL$^\infty$ preproof because, as we explained, if we remove the labels, every rule in a $\mathcal{L}$-proof is a μMALL$^\infty$ proof. Then, as we will show now, the definition of $\mathcal{L}$-proofs precisely reflects the thread criterion.

Suppose an infinite branch $\gamma = (s_i)_{i \in \omega}$; we will show that $\lceil \pi \rceil$ has a valid infinite inner thread in γ (Definition 39). There exists a sequence $(\nu^{L_i} X G_i)_{i \in \omega}$ and a strictly increasing function ϵ on natural numbers as in Definition 58. $\forall i$, let p_i be the position of the $\nu^{L_i} X G_i$ which is principal in $s_{\epsilon(i)}$.

Remark that any sequent contains only a finite number of atoms $a \in \mathcal{L}$ and therefore $\{i \in \omega \mid a_i \text{ appears in the endsequent of } \pi\}$ is a finite set. Let $i_0 \coloneqq 1 + \max\{i \in \omega \mid a_i \text{ appears in the endsequent of } \pi\}$, so that $\forall i \geqslant i_0$, the symbol a_i, contained by L_i, does not appear in the endsequent of π.

First consider any $i \geqslant i_0$, any $j \in \omega$ and any position p of a subformula equal to $\nu^{L_i} X G_i$ in s_j. The extremity $\mathfrak{t}(s_0, \dots, s_j)(p)$ of $\mathfrak{T}(s_0, \dots, s_j)(p)$ is either in a cut formula or in the endsequent of π. In both cases, it does not contain a_i, which is present in L_i. Let us denote by $g(p)$ the lowest point of $\mathfrak{T}(s_0, \dots, s_j)(p)$ which contains a_i, it has to be a subformula of the active formula of the premise of a $(\nu_{\mathfrak{b}}(a_i))$ inference. Similarly, let us denote by $f(p)$ the highest point of $\mathfrak{T}(s_0, \dots, s_j)(p)$ which does not contain a_i, it has to be the principal formula of this same inference $(\nu_{\mathfrak{b}}(a_i))$. And, of course, there is an edge $f(p) \to g(p)$ in $\mathfrak{T}(s_0, \dots, s_j)(p)$, induced by this inference.

From the last paragraph, we deduce that $\forall i \geqslant i_0$, there is a $(\nu_{\mathfrak{b}}(a_i))$ inference in γ. We also deduce that the principal formula of this inference is $\nu^{L_{i-1}} X G_{i-1}$. Remark that this is the only $(\nu_{\mathfrak{b}}(a_i))$ inference in γbecause of the constraint of Definition 57. Let $s_{\phi(i)}$ be the position of the conclusion of this inference, and let us denote by q_i the position of the $\nu^{L_{i-1}} X G_{i-1}$ which is the principal formula of this inference.

The union of the $\mathfrak{T}(s_{\phi(i-1)}, \dots, s_{\phi(i)})(q_i)$, for $i \geqslant i_0 + 1$, is an infinite path in $G^{\text{in}}_{\text{thread}}$, which contains an infinity of principal formulas. This is a valid inner thread in $\lceil \pi \rceil$. □

4.2 Finite representations of circular $\mathcal{L}$-proofs.

We now turn our attention to finite representations of (circular) $\mathcal{L}$-proofs. Immediately a difficulty occurs in comparison to non-labelled proofs: whereas an infinite non-labelled proof may happen to be regular, a valid $\mathcal{L}$-proof cannot be circular, because along every infinite branch, the sets of labels will grow endlessly. To form circular proofs with labels, some atoms must be forgotten when going bottom-up.

We introduce two more rules: $(\circlearrowleft(a))$ and (LWk). The first one allows to forget one atom, just before recreating it by means of a back edge to an already encountered ν-rule. The other one allows to forget any atom that will not be used to validate the proof. It is used to synchronise the different labels in a sequent before travelling through a back edge.

- labelled back edge: $\dfrac{}{\vdash \nu^{L,a} X A[X], \Gamma}\,(\circlearrowleft(a))$ with the constraint that it must be the source of a back edge to the conclusion of a $\dfrac{\vdash A[\nu^{L,a} X A[X]], \Gamma}{\vdash \nu^{L} X A[X], \Gamma}\,(\nu_{\mathfrak{b}}(a))$ below $(\circlearrowleft(a))$.
- labelled weakening: $\dfrac{\vdash \Gamma, B[\nu^{L} X A], \Delta}{\vdash \Gamma, B[\nu^{L,a} X A], \Delta}\,(\text{LWk})$

Definition 59 ($\mu\text{MALL}^{\circlearrowleft}_{\text{lab}}$)**.** $\mu\text{MALL}^{\circlearrowleft}_{\text{lab}}$ denotes the finite derivations of $\mathcal{L}$-sequents built from the rules in Definition 17 by replacing (ν) with $(\nu_{\mathfrak{b}}(a))$, $(\circlearrowleft(a))$, (LWk), such that

1. the endsequent of the proof has empty labels and
2. for every two $(\nu_{\mathfrak{b}}(a))$ and $(\nu_{\mathfrak{b}}(b))$ occurring in the derivation, $a \neq b$.

Since $\mu\text{MALL}^{\circlearrowleft}_{\text{lab}}$ proofs are finite, label-erasing gives rise to μMALL^{ω} b.e.-tree preproofs:

Proposition 12. *The label-erasing function $\lceil\bullet\rceil$ lifts to a translation from $\mu\text{MALL}^{\circlearrowleft}_{\text{lab}}$ to μMALL^{ω} b.e.-tree preproofs.*

Proof. Every rule of the labelled $\mu\text{MALL}^{\circlearrowleft}_{\text{lab}}$ proof is sent by $\lceil\bullet\rceil$ to a valid rule of unlabelled μMALL^{ω}, except for the (LWk) rule, which can safely be removed:

$$\frac{\vdash \Gamma, B[\nu^{L} X A], \Delta}{\vdash \Gamma, B[\nu^{L,a} X A], \Delta}\,(\text{LWk}) \qquad \text{becomes} \qquad \frac{\vdash \lceil\Gamma\rceil, \lceil B[\nu X \lceil A\rceil]\rceil, \lceil\Delta\rceil}{\vdash \lceil\Gamma\rceil, \lceil B[\nu X \lceil A\rceil]\rceil, \lceil\Delta\rceil}$$

□

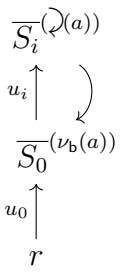
Definition 60 ($\mu\text{MALL}^{\circlearrowleft}$)**.** We denote by $\mu\text{MALL}^{\circlearrowleft}$ the set of b.e.-tree preproofs that are obtained from $\mu\text{MALL}^{\circlearrowleft}_{\text{lab}}$ by label-erasing. This is a subset of all μMALL^{ω} b.e.-tree preproofs.

Proposition 13 ($\mu\text{MALL}^{\circlearrowleft} \subseteq \mu\text{MALL}^{\omega}$)**.** *Every preproof of* μMALL^{ω} *that is the image of a proof in* $\mu\text{MALL}^{\circlearrowleft}_{\text{lab}}$ *by label-erasing satisfies thread validity.*

We immediately give a proof sketch of this proposition; we then prove the two Lemmas 8 and 9 before giving a full proof of Proposition 13.

Proof sketch of Proposition 13. Consider a preproof $\lceil\pi\rceil$ in $\mu\text{MALL}^{\circlearrowleft}$ which is the image of an $\mathcal{L}$-proof π in $\mu\text{MALL}^{\circlearrowleft}_{\text{lab}}$ and an infinite branch b in $\lceil\pi\rceil$. We want to prove that b contains a valid thread (see Definition 21). Let b_0 be the corresponding infinite $\mathcal{L}$-branch in π. We remark (Lemma 8) that there is a sequent S_0 which is the lowest target of back edge crossed infinitely often by b_0. Besides, S_0 is the conclusion of a $(\nu_{\mathfrak{b}}(a))$ rule, which unfolds some $\nu^L XA$.

We decompose b_0, with root r; S_0 conclusion of $(\nu_{\mathfrak{b}}(a))$ and $\nu^L XA$ at position p_0 in S_0; for any $i \geq 1$, S_i conclusion of a back edge $(\circlearrowleft(a))$ with $\nu^{L,a} XA$ at position p_i in S_i. Then we notice that $\mathfrak{T}(u_i)(p_0)$ (Definition 40) is a thread $(S_0, p_0) \xrightarrow{*} (S_i, p_i)$ which is progressing, as its source is the principal conclusion of the rule $(\nu_{\mathfrak{b}}(a))$. By concatenating the $\mathfrak{T}(u_i)(p_0)$ and then erasing labels, we get a valid thread of b in $\lceil\pi\rceil$.



□

Lemma 8. *Let b be an infinite branch in a b.e.-tree preproof, i.e. an infinite ascending path in its branch graph, starting from the root.*

Then there is an occurrence of sequent in this b.e.-tree preproof, i. e. a vertex s in its branch graph, which is the lowest one infinitely appearing on b.

Moreover, this occurrence of sequent (this vertex) is the target of a back edge.

Proof. This is essentially a graph-theoretic property, which does not rely on the logical content of the proof.

The crucial fact to notice is that, in general, in a tree, if S is a non empty, finite set of vertices that is connected for the relation of comparability, that is if $\forall v, v' \in S, v \leqslant v'$ or $v' \leqslant v$, then S has a minimum. This is proved by induction on the cardinal of S.

Take then for S the set of vertices occurring infinitely often on the branch b. It is a connected subgraph of G_{branch}. Because of that, it is connected for comparability in the tree: if $s_0 \to s_1$ in an edge in S then either it is induced by a logical inference, in which case s_0 is below s_1, or it is induced by a back edge, in which case s_1 is below s_0.

We can then apply the previous remark to S, and we obtain a vertex s, which is the lowest position of sequent infinitely appearing on b.

In particular, because s appears infinitely often on b, there must be another position s_0 in S and an edge $s_0 \to s$. Because s is the lowest position in S, this edge must be induced by a back edge. Hence s is a target of back edge. □

Lemma 9 (Follow-up of labels)**.** *If u is a path in the branch graph of a $\mu\text{MALL}^{\circlearrowright}_{\text{lab}}$ proof, if u does not cross the rule $(\nu_{\mathfrak{b}}(a))$, and if p is a position in the target sequent of u (its top sequent) that is labelled with a, then $\mathfrak{t}(u)(p)$ (Definition 40) is defined and is a position labelled with a in the source sequent of u (its bottom sequent).*

Proof. This is quite straightforward, by induction on the length of u, and by looking at the first (or the last) rule crossed by u. We use notably the fact that, when the induced thread $\mathfrak{T}(u)(p)$ is followed top-down, the label a cannot be erased because we do not cross $(Rec(a))$ and the thread cannot reach a cut-formula because cut-formulas do not contain labels. □

Proof of Proposition 13. Suppose π is a $\mu\text{MALL}^{\circlearrowright}_{\text{lab}}$ proof.

- Let $\lceil \pi \rceil$ be its erasure. $\lceil \pi \rceil$ is thus a μMALL^{ω} b.e.-tree preproof.
- Suppose b an infinite branch of $\lceil \pi \rceil$, that is an infinite ascending path in the b.e.-tree $\lceil \pi \rceil$, starting from the root.

- Let b_0 be the corresponding infinite branch in π.
- By Lemma 8, let S_0 be the occurrence of sequent in π which is the lowest back edge target infinitely often crossed by b_0. Being the target of some back edge(s), S_0 is the conclusion of a $(\nu_{\mathfrak{b}}(a))$ rule, which unfolds some $\nu X A[X]$.
- Let us decompose b_0 by making apparent each occurrence of S_0:

$$b_0 = r \xrightarrow[u_0]{*} S_0 \xrightarrow[u_1]{*} S_1 \underset{\text{be}}{\to} S_0 \xrightarrow[u_2]{*} S_2 \underset{\text{be}}{\to} S_0 \cdots$$

  where r is the root of π, "be" denotes a back edge, the S_i $(i \geqslant 1)$ are the sources of the back edges to S_0 and $\forall i, u_i$ does not cross S_0 except at its source.
- Let p_0 be the position in S_0 of the $\nu X A[X]$ which is its principal formula.
- Remark that, because of the existence of back edges from every S_{i+1} to S_0, all S_i are identical sequents, except for the fact that a does not appear in S_0 while it appears at the only position p_0 in S_{i+1}. In particular, p_0 is also the position of a $\nu X A[X]$ in S_{i+1}.
- For $i \geqslant 1$, $\mathfrak{T}(u_i)(p_0)$ is a ν-thread in u_i. Its target is the position p_0 in S_i, which is the position of a $\nu X A[X]$ labelled with a. Its source is a position of $\nu X A[X]$ in S_0. We show that this source position is the same position p_0 of the $\nu X A[X]$ that is principal in S_0.

  Let us decompose u_i as

$$u_i = S_0 \xrightarrow[e_0]{} S_0' \xrightarrow[u_i']{*}$$

  where S_0' is the position of sequent immediately above S_0, that is the premise of the $(\nu_{\mathfrak{b}}(a))$, and e_0 is the edge induced by the $(\nu_{\mathfrak{b}}(a))$ rule.

  By definition of u_i, the path u_i' does not cross $(\nu_{\mathfrak{b}}(a))$. By Lemma 9, the source of $\mathfrak{T}(u_i')(p_0)$ is a position labelled with a, in S_0'. Hence, this is a position p_0' of a $\nu X A[X]$ in the unfolding $A[\nu X A[X]]$ caused by $(\nu_{\mathfrak{b}}(a))$. The source of $\mathfrak{T}(u_i)(p_0)$ is then the image in S_0 of this $\nu X A[X]$ at p_0', that is the principal formula $\nu X A[X]$ at position p_0 in S_0.
- $\mathfrak{T}(u_i)(p_0)$ is then a thread $(S_0, p_0) \xrightarrow[\mathfrak{T}(u_1)(p_1)]{*} (S_1, p_0)$. It is progressing, because its source is the principal conclusion of the rule $(\nu_b(a))$.
- By glueing the $\mathfrak{T}(u_i)(p_0)$ together, we get an infinite inner thread

$$(S_0, p_0) \xrightarrow[\mathfrak{T}(u_1)(p_0)]{*} (S_1, p_0) \underset{\text{be}}{\to} (S_0, p_0) \xrightarrow[\mathfrak{T}(u_2)(p_0)]{*} (S_2, p_0) \underset{\text{be}}{\to} (S_0, p_0) \cdots$$

This thread is valid because every $\mathfrak{T}(u_i)(p_0)$ is progressing. And it is indeed a thread of

$$b_0 = r \xrightarrow[u_0]{*} S_0 \xrightarrow[u_1]{*} S_1 \xrightarrow[\text{be}]{} S_0 \xrightarrow[u_2]{*} S_2 \xrightarrow[\text{be}]{} S_0 \cdots$$

- Hence b_0 is valid, which concludes this demonstration.

□

Proposition 14. *μMALL proofs can be translated to $\mu\text{MALL}^{\circlearrowright}$.*

Proof. The target of the usual translation $\mu\text{MALL} \to \mu\text{MALL}^{\omega}$ [Doumane, 2017b] is included in $\mu\text{MALL}^{\circlearrowright}$. See key case of the translation on Figure 4.1. □

$$\dfrac{\vdash A[B], B^{\perp} \quad \vdash B, \Gamma}{\vdash \nu X A[X], \Gamma}\,(\nu^0_{\text{inv}}) \quad \equiv \quad \dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \nu^a X A[X], B^{\perp}}\,(\circlearrowright(a))}{\vdash A[\nu^a X A[X]], A[B]^{\perp}}\,[A] \qquad \vdash A[B], B^{\perp}}{\vdash A[\nu^a X A[X]], B^{\perp}}\,(\text{cut})}{\vdash \nu X A[X], B^{\perp}}\,(\nu_{\text{b}}(a)) \qquad \vdash B, \Gamma}{\vdash \nu X A[X], \Gamma}\,(\text{cut})$$

Figure 4.1. translation $\mu\text{MALL} \to \mu\text{MALL}^{\circlearrowright}_{\text{lab}}$

Remark 19. Observe that a proof in $\mu\text{MALL}^{\circlearrowright}$ is not, in general, the translation of a μMALL proof.

4.3 Two alternative characterizations of $\mu\text{MALL}^{\circlearrowright}$

In the two following sections, we give two characterizations of $\mu\text{MALL}^{\circlearrowright}$, through validating sets in Definition 64, and through a threading criterion over back edges in Definition 66. This allows us to give effective algorithms to decide whether a μMALL^{ω} b.e.-tree preproof is in $\mu\text{MALL}^{\circlearrowright}$ and, in case it is, to produce a labelling witnessing that fact.

Definition 61. Given a directed graph $G = (V, E)$ and a set $S \subseteq V$, the set of vertices accessibles from S is denoted as $S{\uparrow} \coloneqq \{v \in V \mid \exists s \in S, s \to^* v\}$. Similarly $S{\downarrow}$ is the set of vertices from which S is accessible.

Given a μMALL$^\omega$ b.e.-tree preproof, we will define a graph G_π very similar to $G^{\text{in}}_{\text{thread}}$. In $G^{\text{in}}_{\text{thread}}$, we consider only positions of subformulas in closed sequents, and the positions of subformulas in a source of back edge, which is an open sequent, are identified to those in the target of that back edge. In contrast, in G_π, we consider the positions in sources of back edges as distinct from the corresponding positions in their targets. The aim of G_π is precisely to see how we can connect those positions through paths in the graph.

Definition 62 (G_π)**.** Given a μMALL$^\omega$ b.e.-tree preproof π, we define G_π to be the following graph:

- its *vertices* are all positions of ν-formulas in all occurrences of sequents in π, plus the vertex $\bot$:

$$V_\pi \coloneqq \left\{ (v,i,p) \text{ such that } \begin{array}{r} (i)\ v \text{ position of a sequent } \Gamma \text{ in } \pi \\ (ii)\ i \text{ position of a formula } A \text{ in } \Gamma \\ (iii)\ p \text{ position of a } \nu\text{-subformula in } A \end{array} \right\} \uplus \{\bot\};$$

- for its *edges*: for every position p of a ν-subformula, if q is its image in the sequent just below, as defined by the threading structure of the preproof, then there is an edge $q \longrightarrow p$ in G_π. Moreover, for every position p of a ν-subformula of a cut formula, there is an edge $\bot \longrightarrow p$.

We illustrate the construction of the edges of the graph defined in definition 62 with the following examples in which we have indexed the apparent ν-formulas by numbers representing vertices of the graph:

$$\frac{\vdash \nu_1 XX, \nu_2 XX \quad \vdash \mathbf{1} \oplus \nu_3 XX}{\vdash \nu_4 XX \otimes (\mathbf{1} \oplus \nu_5 XX), \nu_6 XX}\,(\otimes)$$

induces edges $4 \to 1, 6 \to 2$ and $5 \to 3$,

$$\frac{\vdash \nu_1 XX, (\mathbf{1} \oplus \nu_2 XX), \nu_3 XX}{\vdash \nu_4 XX \parr (\mathbf{1} \oplus \nu_5 XX), \nu_6 XX}\,(\parr)$$

induces edges $4 \to 1, 5 \to 2$ and $6 \to 3$, and

$$\frac{\vdash \nu_4 Y \nu_5 X (\nu_6 YX \otimes X) \otimes \nu_7 X (\nu_8 YX \otimes X), \nu_9 XX}{\vdash \nu_1 X (\nu_2 YX \otimes X), \nu_3 XX}\,(\nu)$$

induces edges $2 \to 4, 2 \to 6, 2 \to 8, 1 \to 5, 1 \to 7$ and $3 \to 9$. Moreover, if the conclusion of this last rule is the target of a back edge whose source is $\vdash \nu_{10}X(\nu_{11}YX \otimes X), \nu_{12}XX$ then this back edge also induces edges $10 \to 1, 11 \to 2, 12 \to 3$.

In the case of a cut formula, the formula has no corresponding formula in the conclusion sequent and in this case it induces an ingoing edge from the extra vertex $\bot$:

$$\frac{\vdash \nu_2 XX \quad \vdash \mu XX, \nu_3 XX}{\vdash \nu_1 XX}\,(\text{cut})$$

induces edges $\bot \to 2$ and $1 \to 3$.

Remark 20. Note that a vertex of the above graph may therefore have

- many ingoing edges: in case it in a sequent which is the target of some back edges;
- no ingoing edges: essentially the extra vertex $\bot$ and the ν-formulas in the conclusion of the proof if it is not a target of back edge;
- many outgoing edges: see a ν-rule with $\nu X(X \otimes X)$ or $\nu X(X \parr X)$ as principal conclusion;
- no outgoing edges: see the νXX in the $\mathbf{1} \oplus \nu XX$ principal conclusion of a $(\oplus_1)$-rule.

In the following definition, we define, for each (ν) rule (r) of a b.e.-tree preproof π, a subgraph of G_π (Definition 62) called G_r. The aim of this subgraph is to contain all paths in G_π that uses only sequents and back edges above (r). We also define two subsets of the vertices of G_r, called S_r and T_r. As we will show in Proposition 16, π is *not* in $\mu\text{MALL}^{\circlearrowright}$ iff. there is a (ν) rule (r) such that there is a path in G_r with source in S_r and target in T_r. This gives us an effective way to check membership in $\mu\text{MALL}^{\circlearrowright}$ (Proposition 17).

Definition 63 (G_r, S_r, T_r)**.** Let π be a μMALL^{ω} b.e.-tree preproof and (r) an occurrence of a (ν)-rule. We define a subgraph of G_π, denoted by $G_r = (V_r, E_r)$, and two subsets of V_r, denoted by T_r and S_r:

- the vertices $v \in V_r$ are the extra vertex $\bot$ plus all positions that are in the conclusion of (r) and in all above sequents, that is all sequents accessibles from the conclusion of (r), in the sense of Definition 61;
- the edges $e \in E_r$ are all edges of G_π between those vertices, minus the edges of G_π that are induced by the back edges of π targetting the conclusion of (r), if there

are some.

- $T_r \subseteq V_r$ is the set of all positions of the principal formulas of the sources sequents of the back edges targetting the conclusion of (r);
- $S_r \subseteq V_r$ is the set of all positions of all subformulas of the conclusion of (r) except for the position of its principal formula, plus the extra vertex $\bot$.

Definition 64. Let (r) be an occurrence of a (ν) rule in a μMALL^ω b.e.-tree preproof π. A *validating set* for (r) is a set $L \subseteq V_r$ such that $L = L{\downarrow}$ and $T_r \subseteq L \subseteq (V_r \setminus S_r)$.

Proposition 15. *Let (r) be an occurrence of a (ν) rule of a μMALL^ω b.e.-tree preproof π. The following are equivalent:*

1. *There exists a validating set for (r)*
2. *T_r is not accessible from S_r in G_r*
3. *$T_r{\downarrow} \subseteq V_r \setminus (S_r{\uparrow})$.*

And if they are true, $T_r{\downarrow}$ is the smallest validating set of (r) and $V_r \setminus (S_r{\uparrow})$ is the biggest one.

Proof. We first prove that

$$\text{If } L \text{ is a validating set, then } T_r{\downarrow} \subseteq L \subseteq V \setminus (S_r{\uparrow}) \tag{4.1}$$

If L is a validating set, then on the one hand we have

$$T_r \subseteq L$$

hence

$$T_r{\downarrow} \subseteq L{\downarrow} = L.$$

And on the other hand

$$L \subseteq V \setminus S_r$$

hence

$$V \setminus L \supseteq S_r$$

and, because the complement of a downward-closed set is upward-closed,

$$V \setminus L = (V \setminus L)\uparrow \supseteq S_r\uparrow$$

hence

$$L \subseteq V \setminus (S_r\uparrow).$$

1 Now the implication (1) $\Rightarrow$ (3) is an immediate consequence of (4.1).

For (3) $\Rightarrow$ (2): if

$$T_r\downarrow \subseteq V \setminus (S_r\uparrow)$$

then

$$T_r \subseteq T_r\downarrow \subseteq V \setminus (S_r\uparrow)$$

hence

$$S_r\uparrow \cap T_r \subseteq \emptyset$$

2 hence T_r is not accessible from S_r.

Finally (2) $\Rightarrow$ (1): if T_r is not accessible from S_r then

$$S_r \cap T_r\downarrow \subseteq \emptyset$$

hence

$$T_r \subseteq T_r\downarrow \subseteq V \setminus S_r$$

3 hence $T_r\downarrow$ is a validating set.

4 Regarding the second part of the proposition:

- if (3) is true then
$$T_r \subseteq T_r\downarrow \subseteq V_r \setminus (S_r\uparrow) \subseteq V_r \setminus S_r$$
5 hence $T_r\downarrow$ and $V_r \setminus (S_r\uparrow)$ are validating sets
- 6 if L is any other validating set, then by (4.1), $T_r\downarrow \subseteq L \subseteq V \setminus (S_r\uparrow)$.

7 □

The following proposition gives an alternative criterion for $\mu\mathrm{MALL}^{\circlearrowright}$.

Proposition 16. *A finite representation π of a $\mu\mathrm{MALL}^{\omega}$ preproof is the erasing of a $\mu\mathrm{MALL}^{\circlearrowright}_{\mathrm{lab}}$ proof iff. every occurrence of a (ν) rule of π has a validating set.*

Proof. ($\Longleftarrow$) Let us assume that every (ν) rule of π has a validating set. There is a finite number of (ν) rules in π; we label them with distinct variables $a_1, \dots, a_n$, in such a way that if the (ν) rule labelled by a_i is below the rule labelled by a_j in the representation then $i \leqslant j$. We denote by L_i a validating set for $(\nu_{\mathfrak{b}}(a_i))$. We then do the following for each i, going from 1 to n: for each occurrence of ν-formula $\nu^V XA$ that is at a position belonging to L_i, add the variable a_i to V, that is replace this occurrence of $\nu^V XA$ with $\nu^{V,a_i} XA$. By doing this it may happen that we break the validity of some rules of the representation: because L_i, although downward closed, is in general not upward closed, so we may end with a situation in which this

$$\frac{\vdash A, C[\nu^V XD] \quad \vdash A, C[\nu^V XD]}{\vdash A \,\&\, B, C[\nu^V XD]}\,\&$$

becomes

$$\frac{\vdash A, C[\nu^{V,a} XD] \quad \vdash B, C[\nu^V XD]}{\vdash A \,\&\, B, C[\nu^{V,a} XD]}\,\&$$

which is not anymore a valid rule. We then patch this by adding as many (LWk) rules as needed on the premises:

$$\frac{\dfrac{\vdash A, C[\nu^{V,a} XD]}{\vdash A, C[\nu^{V,a} XD]}\,(\mathrm{LWk}) \quad \dfrac{\vdash B, C[\nu^V XD]}{\vdash B, C[\nu^{V,a} XD]}\,(\mathrm{LWk})}{\vdash A \,\&\, B, C[\nu^{V,a} XD]}\,\&$$

Similarly it may happen that the source of a back edge gets a bigger labelling than the target of this back edge; we patch this by adding (LWk) rules under the source sequent of the back edge. When this operation has been done for every i, from 1 to n, we obtain a validly labelled proof of $\mu\mathrm{MALL}^{\circlearrowright}_{\mathrm{lab}}$.

($\Longrightarrow$) Conversely, let π_0 be a $\mu\mathrm{MALL}^{\circlearrowright}_{\mathrm{lab}}$ representation such that $\pi = \lceil \pi_0 \rceil$. Up to renaming, we can assume that all $(\nu_{\mathfrak{b}})$ rules of π_0 are labelled with distinct variables. For every occurrence of a (ν) rule in π, consider the corresponding $(\nu_{\mathfrak{b}}(a))$ rule in π_0 and let L_a be the set of all occurrences of ν-formulas in π_0 that carry the variable a in their labelling. The constraints on the labelling of $\mu\mathrm{MALL}^{\circlearrowright}_{\mathrm{lab}}$ proof precisely say that L_a is a validating set for the considered occurrence of $(\nu_{\mathfrak{b}})$ in π. □

Proposition 17.

- *Checking validity of a* $\mu\text{MALL}^{\circlearrowright}_{\text{lab}}$ *preproof is decidable.*
- *Membership in* $\mu\text{MALL}^{\circlearrowright}$ *can be decided in a time quadratic in the size of the (circular) preproof.*

Proof.

- For the first point, it is enough to check the validity of each inference of the proof.
- For the second point, first check that each back edge targets a (ν) rule, then use the characterization of $\mu\text{MALL}^{\circlearrowright}$ by validating sets (Proposition 16): for each (ν) rule (r), build graph G_r and sets S_r and T_r and checks whether T_r is accessible from S_r in G_r. Such a test of accessibility in a graph can be done in linear time. As we have to do that for each (ν) rule, in total it may take a time quadratic in the size of the preproof, but no more than that.

□

Definition 65. Let π be a μMALL^{ω} b.e.-tree preproof and e a back edge in π, from a position of sequent s_1 to a position of sequent s_0. Let S be the set of all positions of sequents that are strictly above s_0. We say that this back edge is *strongly valid* under two conditions:

1. Its target s_0 has to be the conclusion of a (ν) rule.
2. Let us denote by p the position in s_0 of the νXA which is the principal formula of that (ν) rule. This is also a position of the same formula in s_1. The second condition is that for every path u in G_{branch}, going from s_0 to s_1, using only positions of sequents in S, $\mathfrak{t}(u)(p)$ is, again, the position p in s_0.

Definition 66. A μMALL^{ω} b.e.-tree preproof is *strongly valid* if all its back edges are strongly valid.

Proposition 18. *A* μMALL^{ω} *b.e.-tree preproof* π *is strongly valid iff. it is the erasing of a* $\mu\text{MALL}^{\circlearrowright}_{\text{lab}}$ *proof.*

Proof. We will use our first characterization by validating sets (Proposition 16) and prove that a μMALL^{ω} b.e.-tree preproof π is strongly valid iff. every ν-rule of π has a validating set.

($\Longleftarrow$) Let us assume that π has a validating set. Let us consider one occurrence $\dfrac{\vdash A[\nu X A[X]], \Gamma}{\vdash \nu X A[X], \Gamma}\,(\nu)$ of a ν-rule in π and a path u in the subgraph above this ν-rule, going down, from the source of a back edge targetting this ν-rule, to the ν-rule itself, ending by this ν-rule. u has then premise and conclusion equal to $\vdash \nu X A[X], \Gamma$.

Let us denote by L a validating set of this (ν) rule occurrence, and let us denote by t the maximal thread going down in u starting from the main $\nu X A[X]$ in its premise. This occurrence of $\nu X A[X]$ is in L, because L is a validating set. Then, because L is downward closed, all vertices of t are in L. Therefore the lowest vertex of t, which is a position in the $\vdash \nu X A[X], \Gamma$ conclusion of the considered ν-rule, or $\bot$, is also in L. But in this last sequent occurrence, the only position that is in L is the one of the main $\nu X A[X]$, which is consequently the end point of t.

($\Longrightarrow$) Conversely, let us consider an occurrence (r) of a (ν) rule in π, whose conclusion has the form $\vdash \nu X A[X], \Gamma$, and let us assume that it has no validating set. By Proposition 15, there is a path v in G_r with source $s \in S_r$ and target $t \in T_r$. Let u be the corresponding path in G_{branch}. By definition of T_r (Definition 63), t is in the source of a back edge e targetting the conclusion of (r). More precisely, the position of t in the sequent at s is the same as the position of the principal formula of (r) in the conclusion of (r). Because the definition of G_r follows the threading structure of the preproof, the source s of v is $\mathfrak{t}(u)(t)$. By definition of S_r, this position $s = \mathfrak{t}(u)(t)$ is either in a cut formula or in the conclusion of (r), at a position different from the position of its principal formula. This shows that (r) is not strongly valid. □

Corollary 3. *The fragments defined in Definitions 60 and 66 coincide.*

4.4 Conclusion

As this chapter and the next one share a lot of ideas, discussion of related works for this chapter has been placed in Section 5.5.

This chapter introduced the loop criterion for circular representations of infinite preproofs of μMALL^ω. This defines a fragment of the circular representations of μMALL^ω which we called $\mu\text{MALL}^{\circlearrowright}$.

We also defined a new proof system called $\mu\text{MALL}^{\circlearrowright}_{\text{lab}}$. The proofs of $\mu\text{MALL}^{\circlearrowright}_{\text{lab}}$ are labelings of circular representations of preproofs, valid for the loop criterion. While circular representation of preproofs make use of a global validity criterion such as the

thread criterion or the loop criterion, the validity of a proof in $\mu\text{MALL}^{\circlearrowright}_{\text{lab}}$ is a purely local property, which can be checked in linear time.

We proved in Propositions 16 and 18 that the b.e.-tree preproof obtained by erasing the labels of a $\mu\text{MALL}^{\circlearrowright}_{\text{lab}}$ proof is valid for the loop criterion and, conversely, that any circular representation that is valid for the loop criterion can be turned into a proof of $\mu\text{MALL}^{\circlearrowright}_{\text{lab}}$ by labelling some of its formulas.

We made that last property explicit by providing a procedure to check a circular representation with respect to the loop criterion and to turn it into a labelled proof of $\mu\text{MALL}^{\circlearrowright}_{\text{lab}}$ when it is valid. This algorithm runs in quadratic time.

Now that we have identified a fragment of μMALL^{ω} which proves at least all sequents provables in μMALL, it is natural to wonder whether it proves strictly more than μMALL. This question is the subject of the next and final chapter of this thesis.

5 Finitization

The first systems designed to handle least and greatest fixed points connectives were finitary systems. Infinitary and circular systems were then proposed as an alternative. In each of these circular extensions of finitary systems, there is a canonical translation from finitary proofs to circular proofs. This implies in particular that any sequent that is provable in the finitary system is also provable in the circular system. The converse is known as the Brotherston-Simpson conjecture and was first stated by Brotherston and Simpson [2007, 2011] in the context of Martin-Löf's system of inductive definitions for first-ordre classical logic. It is also called the problem of finitization: can we turn an infinitary proof into a finitary one, and how? In the context of first-order classical logic with inductive definitions, it was proved by Berardi and Tatsuta [2017b, 2019] that the conjecture is false in general, but it was shown by Simpson [2017], Berardi and Tatsuta [2017a, 2018] that it becomes true as soon as you add Peano (or even Heyting) arithmetic to the circular and finitary systems. But the question is still open in the case of μMALL and μMALL^ω, which is a quite different system, more in the family of μ-calculi. In that setting, on one hand we do not have first-order predicates and the possibility to encode Peano arithmetic, on the other hand we have access to all possible inductive and coinductive definitions of formulas, including interleaving of least and greatest fixed points.

Here the question of finitization is: if a conclusion is provable by a circular proof, valid for the thread criterion, that is a proof in μMALL^ω, is it already provable in the finitary system μMALL?

This question is, as of today, an open problem.

Now, in the previous chapter, we provided a new criterion, the loop criterion, which defines a new logic $\mu\text{MALL}^{\circlearrowleft}$, which has two properties:

- the canonical translation of any finitary proof is a circular representation which is valid for the loop criterion, so any conclusion which is provable in μMALL is also provable in $\mu\text{MALL}^{\circlearrowleft}$
- if the circular representation of a preproof is valid for the loop criterion, then it is also valid for the thread criterion, so any conclusion that is provable in $\mu\text{MALL}^{\circlearrowleft}$ is

also provable in μMALL^ω.

This allows us to split the problem of finitization into two subproblems:

- If a conclusion is provable in μMALL^ω, is it already provable in $\mu\text{MALL}^{\circlearrowright}$?
- If a conclusion is provable in $\mu\text{MALL}^{\circlearrowright}$, is it already provable in μMALL?

A positive answer to both questions would mean that finitization is possible and the Brotherston-Simpson conjecture is true for μMALL and μMALL^ω.

In this chapter, we answer positively the second question: we prove that any conclusion provable in $\mu\text{MALL}^{\circlearrowright}$ is already provable in μMALL. We do so by providing an explicit method which takes as input a circular representation of preproof, valid for the loop criterion, and output a standard, finitary μMALL proof.

5.1 On Brotherston-Simpson's conjecture: finitizing circular proofs

The aim of this section is to prove a converse of Proposition 14: *Every provable sequent of* $\mu\text{MALL}^{\circlearrowright}$ *is provable in* μMALL.

This will be proved by defining a translation from every $\mu\text{MALL}^{\circlearrowright}$ proof of a sequent $\vdash \Gamma$ containing no label variable into a μMALL proof of the same sequent.

Let us consider a $\mu\text{MALL}^{\circlearrowright}$ proof π. Up to renaming of bound variables, we can assume that all (ν_{b}) rules are labelled by distinct labels. For every two labels a and b occurring in π, we say that $a \leqslant b$ whenever $(\nu_{\mathsf{b}}(a))$ is under $(\nu_{\mathsf{b}}(b))$. This order is well-founded because finite.

Definition 67 (Context associated to a labelling atom)**.** For every rule $\dfrac{\vdash A[\nu^{V,a}X.A], \Gamma}{\vdash \nu^V X.A, \Gamma}\,(\nu_{\mathsf{b}}(a))$ we define $\Gamma_{(a)}$ to be Γ.

We now define

1. for each atom a a sequent Γ_a formed of non-labelled formulas;

2. for each formula A (with labels) occurring in the proof, a formula $[\![A]\!]$ without labels:

Definition 68 (Translation of labelled formulas to unlabelled formulas)**.** We define by mutual induction:

1. $\Gamma_a \coloneqq [\![\Gamma_{((a))}]\!]$.

2. $H_\emptyset[F] \coloneqq F$ and $H_{V,a}[F] \coloneqq \otimes\Gamma_a^\perp \oplus H_V[F]$. In other words,

$$H_V[F] \simeq \Big(\bigoplus_{a\in V} \otimes\Gamma_a^\perp\Big) \oplus F$$

3. By induction on formula A, $[\![A]\!]$ is:

   - $[\![\nu^V X.A]\!] \coloneqq \nu X H_V[[\![A]\!]]$
   - it is homomorphic on other connectives: $[\![X]\!] \coloneqq X$, $[\![\mathbf{1}]\!] \coloneqq \mathbf{1}$, $[\![\mu X A]\!] \coloneqq \mu X [\![A]\!]$, $[\![A \otimes B]\!] \coloneqq [\![A]\!] \otimes [\![B]\!]$, *etc.*

4. $[\![\cdot]\!]$ is lifted from formulas to sequences of formulas, pointwise.

This is well-founded because, since any two distinct ν_b rules wear distinct variables, the only Γ_b that are needed in the computation of Γ_a are those with $b < a$. Note that $[\![A]\!] = A$ as soon as A has no label variable. We can now state and prove the finitization theorem:

Theorem 5. *Every provable sequent of* $\mu\text{MALL}^{\circlearrowright}$ *is provable in* μMALL*.*

Proof. Let π be a $\mu\text{MALL}^{\circlearrowright}_{\text{lab}}$ proof and replace, everywhere, each formula A by $[\![A]\!]$. All rules in this (almost) new derivation are now valid instances of μMALL rules, except for (ν_b), (LWk) and ($\circlearrowright$) rules. Actually, images of these rules by sequent translation $[\![\cdot]\!]$ are derivable in μMALL as shown in fig. 5.1 (a), (b) and (c) for ($\circlearrowright$), (LWk) and (ν_b), respectively.

Replacing each instance of a (ν_b), (LWk) or ($\circlearrowright$) rule in π by its derived version, we get a fully valid proof of μMALL. If the conclusion of the original $\mu\text{MALL}^{\circlearrowright}$ proof was $\vdash \Gamma$ then what we get is a proof in μMALL of $\vdash [\![\Gamma]\!]$, *i. e.* the conclusion of the original $\mu\text{MALL}^{\circlearrowright}$ proof, if Γ contains no label variable. □

$$(a)\quad \dfrac{\dfrac{\vdash [\![A[\nu^V X.A]]\!], \Gamma}{\vdash H_V\Big[[\![A[\nu^V X.A]]\!]\Big], \Gamma}\,(\oplus_1)^{|V|}}{\vdash [\![\nu^V X.A]\!], \Gamma}\,(\nu) \qquad (b)\quad \dfrac{\dfrac{\dfrac{}{\vdash \otimes\Gamma_a^\perp, \Gamma_a}\,(\otimes),\ (\mathrm{id})}{\vdash H_{V,a}\Big[[\![A[\nu^{V,a} X.A]]\!]\Big], \Gamma_a}\,(\oplus_0)}{\vdash [\![\nu^{V,a} X.A]\!], \Gamma_a}\,(\nu)$$

$$(c)\quad \dfrac{\vdash [\![B[\nu^V X.A]]\!], \Gamma}{\vdash [\![B[\nu^{V,a} X.A]]\!], \Gamma} :=$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash H_V\Big[[\![A[\nu^V X.A]]\!]\Big], H_V\Big[[\![A[\nu^V X.A]]\!]\Big]}\,(\mathrm{id})}{\vdash H_{V,a}\Big[[\![A[\nu^V X.A]]\!]\Big], H_V\Big[[\![A[\nu^V X.A]]\!]\Big]}\,(\oplus_1)}{\vdash H_{V,a}\Big[[\![A[\nu^V X.A]]\!]\Big], [\![\nu^V X.A]\!]^\perp}\,(\mu)}{\vdash [\![\nu^{V,a} X.A]\!], [\![\nu^V X.A]\!]^\perp}\,(\nu^0_{\mathrm{inv}})}{\vdash [\![B[\nu^{V,a} X.A]]\!], [\![B[\nu^V X.A]]\!]^\perp}\,[\![B]\!] \qquad \vdash [\![B[\nu^V X.A]]\!], \Gamma}{\vdash [\![B[\nu^{V,a} X.A]]\!], \Gamma}\,(\mathrm{cut})$$

$$(d)\quad \dfrac{\dfrac{\vdash [\![A[\nu^{V,a} X.A]]\!], \Gamma_a}{\vdash [\![\nu^V X.A]\!], \Gamma_a}\,[\![\nu_b(a)]\!] :=$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdash [\![A[\nu^{V,a} X.A]]\!], \Gamma_a}{\vdash H_V\Big[[\![A[\nu^{V,a} X.A]]\!]\Big], \Gamma_a}\,(\oplus_1)^{|V|}}{\vdash H_V\Big[[\![A[\nu^{V,a} X.A]]\!]\Big], \Re\Gamma_a}\,(\Re) \qquad \dfrac{}{\vdash H_V\Big[[\![A[\nu^{V,a} X.A]]\!]\Big], H_V\Big[[\![A[\nu^{V,a} X.A]]\!]\Big]}\,(\mathrm{id})}{\vdash H_V\Big[[\![A[\nu^{V,a} X.A]]\!]\Big], H_{V,a}\Big[[\![A[\nu^{V,a} X.A]]\!]\Big]}\,(\&) \qquad \dfrac{\dfrac{\dfrac{}{\vdash \otimes\Gamma_a^\perp, \Gamma_a}\,(\otimes),\ (\mathrm{id})}{\vdash H_{V,a}\Big[[\![A[\nu^{V,a} X.A]]\!]\Big], \Gamma_a}\,(\oplus_0)}{\vdash [\![\nu^{V,a} X.A]\!], \Gamma_a}\,(\nu)}{\vdash H_V\Big[[\![A[\nu^{V,a} X.A]]\!]\Big], [\![\nu^{V,a} X.A]\!]^\perp}\,(\mu)$$

$$\vdash [\![\nu^V X.A]\!], \Gamma_a\,(\nu_{\mathrm{inv}})$$

Figure 5.1. Derivability of (a) $[\![(\nu_{\mathfrak{f}})]\!]$ rule ; (b) $[\![(\circlearrowright)]\!]$ rule ; (c) $[\![(\mathrm{LWk})]\!]$ rule and (d) $[\![(\nu_{\mathfrak{b}})]\!]$ rule .

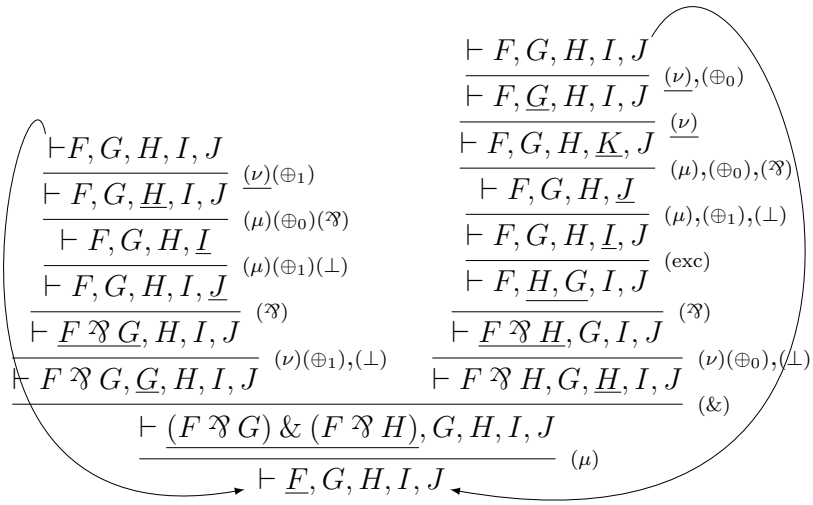


Figure 5.2. Proof π_∞.
$F = \mu X((X \parr G) \& (X \parr H))$ $G = \nu X(X \oplus \bot)$
$H = \nu X(\bot \oplus X)$ $I = \mu Z((Z \parr J) \oplus \bot)$
$J = \mu X((K \parr X) \oplus \bot)$ $K = \nu Y \mu Z((Z \parr \mu X(Y \parr X) \oplus \bot) \oplus \bot)$

5.2 Relaxing the labelling of proofs

In this section, we define an extension of the labelling defined in Section 4.2, with the following properties:

- it allows to label more μMALL$^\omega$ b.e.-tree-preproofs than the labelling of Section 4.2,
- it still ensures thread-validity,
- the validity of a labelled proof amounts to the local validity of each inference,
- to some extent, it preserves the ability to finitize circular proofs.

In order to motivate this extension, we shall consider two examples. The first one is the proof π_∞, which is on Figure 5.2, and on which we will come back later. The second one, that we will present now, has been chosen to be simpler than π_∞.

Example 17. Let D be an arbitrary formula. Lists of D can be represented as proofs of $L_0 \coloneqq \mu X(\mathbf{1} \oplus D \otimes X)$ and it is possible to encode in μMALL$^\omega$ the function taking two lists and computing the tree of all their possible interleavings, as a proof with conclusion[1] $L_0, L_0 \vdash T_0$, where $T_0 \coloneqq \mu X(L_0 \oplus ((D \otimes X) \& (D \otimes X)))$.

[1]In the following, we write $A \multimap B$ for $A^\bot \parr B$, and $\Gamma \vdash \Delta$ for $\vdash \Gamma^\bot, \Delta$.

$$(a)\quad \dfrac{\dfrac{\dfrac{\dfrac{}{D \vdash D}\,(\mathrm{id}) \quad \dfrac{(1)}{L, L \vdash T}\,(\circlearrowleft)}{D \otimes L, L \vdash D \otimes T}\,(\parr)(\otimes)}{L, L \vdash D \otimes T}\,(\nu) \quad \dfrac{\dfrac{\dfrac{}{D \vdash D}\,(\mathrm{id}) \quad \dfrac{(1)}{L, L \vdash T}\,(\circlearrowleft)}{L, D \otimes L \vdash D \otimes T}\,(\parr)(\otimes)}{L, L \vdash D \otimes T}\,(\nu)}{L, L \vdash T \quad (1)}\,(\mu),\,(\&)$$

$$(b)\quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{D \vdash D}\,(\mathrm{id}) \quad \dfrac{(1)}{L^{a+}, L \vdash T}\,(\circlearrowleft(a))}{D \otimes L^{a+}, L \vdash D \otimes T}\,(\parr)(\otimes)}{L^{a-}, L \vdash D \otimes T}\,(\nu)[a]}{L^{a-}, L^{b-} \vdash D \otimes T}\,(\mathrm{LWk}(b-)) \quad \dfrac{\dfrac{\dfrac{}{D \vdash D}\,(\mathrm{id}) \quad \dfrac{(2)}{L^{a-}, L^{b+} \vdash T}\,(\circlearrowleft(b))}{L^{a-}, D \otimes L^{b+} \vdash D \otimes T}\,(\parr)(\otimes)}{L^{a-}, L^{b-} \vdash D \otimes T}\,(\nu)[b]}{L^{a-}, L^{b-} \vdash T \quad (2)}\,(\mu)(\&)\,(Rec(b))}{L^{a-}, L \vdash T \quad (1)}\,(Rec(a))}{L, L \vdash T}$$

Figure 5.3. (a) Interleaving example; (b) Interleaving example labelled. Corresponding sources and targets of back edges are denoted by parenthesized numbers.

By replacing L_0 and T_0 with $L \coloneqq \mu X(D \otimes X)$ and $T \coloneqq \mu X((D \otimes X) \,\&\, (D \otimes X))$, we get an example equally interesting and more readable, which we present in Figure 5.3. In this example, remember that a formula on the left on the sequent means its negation on the right and note that the labels we write on L are meant to be put on its fixed-point connective. That is, a L^a on the left on the sequent means a $\nu^a X(D^\perp \parr X)$ on the right of the sequent. In this interleaving function, every recursive call leaves one of the two arguments untouched and makes the other one decrease. This guarantees that the tree of recursive calls is well-founded. Difficulties, however, arise from the fact that it is not necessarily always the same argument that will decrease.

More formally: every infinite branch in the preproof above has two interesting threads, going through the L formulas. In every branch going infinitely often to the left (resp. to the right), the thread going through the first L of the sequent (resp. the second L of the sequent) will be validating. That preproof is thus a valid $\mu\mathrm{MALL}^\omega$ proof. However, our previous labelling method cannot be applied here for two reasons:

1. in our previous setting, labelled pre-proof have the property that one can know which thread will validate a branch, just by knowing the lowest target of back edge that is visited infinitely often by the branch. This is not the case here, because the two back edges, while inducing different validating threads, have the same target;

2. in our previous setting, back edges must target (ν) rules, which is not the case here.

Both difficulties have, in fact, the same origin, namely that in our previous setting the (ν) rule has two roles: being the target of a back edge and ensuring thread progression. Both difficulties also have the same solution: dissociating these two roles. We therefore introduce, in Definition 69, a new rule (*Rec*), whose only effect is to allow its premise to be the target of a back edge, and to introduce a new label. Since (*Rec*) is disentangled from greatest fixed point unfolding, the labelling must account for the progression of a thread. That is why every atomic label is now given in one of two modes: a passive mode ($a-$) and an active one ($a+$). Only an unfolding by a (ν) can turn a $-$ into a $+$.

Let us now turn back to our introductory example: π_∞ (Figure 5.2). For that example, separating the introduction of back edges and the coinductive progress is not enough to be able to label π_∞ in a way that certify its thread validity. Indeed, since targets of back edges do not require to unfold a ν, there is *a priori* no reason to require that the sequents contain some ν-formula. While this is slightly hidden in the interleaving example of Figure 5.3, π_∞ gives a clear example of this phenomenon and suggests that the (*Rec*) inference should have the ability to add labels *deeply* in the sequent, that is not only on the topmost ν fixed-points, but also to greatest fixed points occurring under some other connectives. The same remark applies to the back edge rule since its conclusion sequents have the same structure as those of (*Rec*).

Driven by these observations, we now define a new labelling of circular preproofs and prove its correctness with respect to thread-validity.

Definition 69 (Extended labelling)**.** Labelled formulas are built on the same grammar as in Definition 56, except that labels are lists of *signed variables*, that is of pairs of a variable and a symbol in $\{+,-\}$. Derivations are built with μMALL inferences plus the following rules:

$$\frac{\vdash \nu^L X A, \Gamma}{\vdash \nu^{L,a-} X A, \Gamma}\,(\mathrm{LWk}(a-)) \qquad \frac{\vdash \nu^{L,a-,L'} X A, \Gamma}{\vdash \nu^{L,a+,L'} X A, \Gamma}\,(\mathrm{LWk}(a+)) \qquad \frac{\vdash A[\nu^{a_1+,\ldots,a_n+} X A], \Gamma}{\vdash \nu^{a_1-,\ldots,a_n-} X A, \Gamma}\,(\nu)$$

$$\frac{\vdash \Gamma[\nu^{L,a-} X A]}{\vdash \Gamma[\nu^{L} X A]}\,(\mathit{Rec}(a)) \qquad \frac{}{\vdash \Gamma[\nu^{L,a+} X A]}\,(\circlearrowleft(a))$$

and the constraints that:

- a cut-formula cannot contain a non-empty label;
- all (*Rec*) rules must wear distinct variables;
- every (*Rec*(a)) rule must have at least one occurrence of "$a-$" in its premise;

- each $\dfrac{}{\vdash \Gamma[\nu^{L,a+}XA]}\,(\circlearrowleft(a))$ rule is connected to the premise of a $\dfrac{\vdash \Gamma[\nu^{L,a-}XA]}{\vdash \Gamma[\nu^{L}XA]}\,(Rec(a))$ via a back edge. This implies in particular that this $(\circlearrowleft(a))$ must be above this $(Rec(a))$ and that the premise of this $(Rec(a))$ must be the same sequent as the conclusion of this $(\circlearrowleft(a))$ except for the change of sign of a, at every of its occurrences in the sequent.

With the extended labelling of Definition 69, we have the following.

Proposition 19 (Soundness of extended labelling)**.** *If π is an extended labelled circular representation then $\lceil\pi\rceil$ is a* valid μMALL$^\omega$ *b.e.-tree* proof.

Proof. First remark that Lemma 9, as it is stated on p. 87, still holds for this extended labelling. The proof is the same, keeping in mind to replace every mention of $(\nu_{\mathfrak{b}}(a))$ with $(Rec(a))$. As for Proposition 13, the proof of this proposition crucially relies on it.

Suppose π is a labelled circular representation. Let $\lceil\pi\rceil$ be its erasure. $\lceil\pi\rceil$ is thus a circular representation of a μMALL$^\omega$ preproof. Suppose b an infinite branch of $\lceil\pi\rceil$, that is an infinite ascending path in the b.e.-tree $\lceil\pi\rceil$, starting from the root. Let b_0 be the corresponding infinite branch in π. Le S_0 be the occurrence of the sequent in π which is the lowest back edge target infinitely often crossed by b_0. Being the target of some back edge(s), S_0 is the premise of a $(Rec(a))$ rule, for some variable a.

This implies that b_0 is of the form $b_0 = r \xrightarrow[u_0]{*} S_0 \xrightarrow[u_1]{*} S_1 \underset{\text{be}}{\to} S_0 \xrightarrow[u_2]{*} S_2 \underset{\text{be}}{\to} S_0 \cdots$ where r is the root of π and where the u_i do not cross S_0 except at their sources.

Remark that the positions labelled by a are the same in all S_i, as there are back edges from every S_{i+1} to S_0. The difference, however, is that these positions are labelled with $a-$ in S_0 and with $a+$ in every S_{i+1}. Let P_0 be the set of those positions. P_0 is finite and non empty. Now we would like, as in the proof of Proposition 13, to construct an infinite thread along b_0. However, because P_0 may contain more than one element, we cannot know by advance, for each S_i, which $p \in P_0$ will support an infinite thread. Thus, we will use Kőnig's lemma to show the existence of such a thread. Let T_0 be the tree whose vertices are the pairs (i, p) where $1 \leqslant i < \omega$ and $p \in P_0$, whose roots are the vertices of the form $(1, p)$ and where, for $i > 1$, the father of (i, p) is[2] $(i-1, \mathfrak{t}(u_i)(p))$. Here we have to prove that $\mathfrak{t}(u_i)(p)$ is defined and that it belongs to P_0 for every i and $p \in P_0$. This is ensured by Lemma 9 thanks to the labels.

Remark that every edge in T_0 induces a progressing thread. Indeed, for $i \geqslant 1$ and $p \in P_0$:

[2]Recall that $\mathfrak{t}(u)$ and $\mathfrak{T}(u)$ are defined in Definition 40, p. 48.

- $\mathfrak{T}(u_i)(p)$ is a ν-thread in u_i,
- its target is p in S_i, which is labelled with $a+$,
- and its source is p in S_0, which is labelled with $a-$.

An examination of the rules that may compose u_i shows that the only way for that to be true is that $\mathfrak{T}(u_i)(p)$ is progressing. Now T_0 is an infinite tree with a finite number of roots and an arity bounded by $\mathbf{Card}(P_0)$, hence, by Kőnig's lemma, it has an infinite branch $(1, p_1) \leftarrow (2, p_2) \leftarrow (3, p_3) \cdots$.

This infinite branch induces in turn an infinite thread

$$(S_0, p_0) \xrightarrow[\mathfrak{T}(u_1)(p_1)]{*} (S_1, p_1) \underset{\text{be}}{\to} (S_0, p_1) \xrightarrow[\mathfrak{T}(u_2)(p_2)]{*} (S_2, p_2) \underset{\text{be}}{\to} (S_0, p_2) \cdots$$

This thread is valid because every $\mathfrak{T}(u_i)(p_i)$ is progressing. And it is indeed a thread of $b_0 = r \xrightarrow[u_0]{*} S_0 \xrightarrow[u_1]{*} S_1 \underset{\text{be}}{\to} S_0 \xrightarrow[u_2]{*} S_2 \underset{\text{be}}{\to} S_0 \cdots$ Hence b_0 is valid, which concludes this demonstration. □

We now label our two examples with this new system. We will show that, while it is quite straightforward to label the interleaving example (Figure 5.3) with this new system, it requires, for π_∞ (Figure 5.2), to unfold one back edge.

π_∞ is presented, on Figure 5.5, labelled according to the extended labelling of Figure 5.5. To be able to label π_∞, two preliminary steps were necessary:

- To make K apparent as a subformula of I and J respectively by decomposing I and J as:

  $$I = I'[K] \qquad\qquad J = J'[K]$$

  where

  $$J'[Y] \coloneqq \mu X((Y \parr X) \oplus \bot) \qquad\qquad I'[Y] \coloneqq \mu Z((Z \parr J'[Y]) \oplus \bot).$$

- To do one step of unfolding on the rightmost back edge.

After that, we can take advantage of the two new facilites of the extended labelling:

1. to add three (*Rec*) rules, corresponding to the three ways for a branch of π_∞ to be valid, as summarized in the following array.

Shape of the branch	$A^\star \cdot l^\omega$	$A^\star \cdot r^\omega$	$l^\star \cdot (r^+ \cdot l^+)^\omega$
Lowest (*Rec*) visited ∞^{ly}	b	a	c
Validating ν-formula	H	G	K

2. to label the three formulas H, G and K at each corresponding (*Rec*), using for K the ability to label several occurrences at a time, and to label deeply ν-subformulas.

This indeed forms a correct labelling of π_∞ according to the extended labelling, hence ensuring its thread-validity.

5.3 Extending finitization

As we did in Section 5.1 for the labelling defined in Chapter 4, we will rely on the labelled presentation of the proofs in order to finitize them. More precisely, we will consider a fragment of the extended labelling of Definition 69, for which we will be able to finitize proofs.

Definition 70 (Core fragment of the extended labelling)**.** A μMALL$^\omega$ b.e.-tree preproof labelled with the extended labelling of Definition 69 is in the *core fragment* of this extended labelling if:

1. no ν-formula wears more than one variable
2. (*Rec*) is used only in the particular form $\dfrac{\vdash \nu^{a-} X A, \Gamma}{\vdash \nu X A, \Gamma}\,(Rec'(a))$ in which only one occurrence of $\nu X A$ is labelled, and this occurrence is a formula of the sequent and not a strict subformula.

Observe already that both the interleaving example (Figure 5.3) and π_∞ (Figure 5.2) respect the two constraints of Definition 70, except for the labelling of K in π_∞.

We show now how to finitize any labelled representation which verify those two restrictions. As the interleaving example of Figure 5.3 is labelled in this core fragment, it gives a finitization for the preproof of Figure 5.3. We will then show how to extend this method in an *ad hoc* way to finitize entirely π_∞ (Figure 5.2) from the labelling of Figure 5.5.

Definition 71. For any formula $\nu X A$ and any unlabelled context Γ, we define the following formula translation:

$$[\![\nu^{\Gamma-} X A[X]]\!]_{\mathrm{e}} := \nu X [\![A]\!]_{\mathrm{e}}[\otimes \Gamma^\perp \oplus X] \qquad [\![\nu^{\Gamma+} X A[X]]\!]_{\mathrm{e}} := \otimes \Gamma^\perp \oplus [\![\nu^{\Gamma-} X A[X]]\!]_{\mathrm{e}}$$

Remark 21. With these definitions, we have

$$[\![\nu^{\Gamma-} X\, A[X]]\!]_{\mathrm{e}} \quad \simeq \quad [\![A]\!]_{\mathrm{e}}[[\![\nu^{\Gamma+} X\, A[X]]\!]_{\mathrm{e}}]$$

through a simple ν-unfolding.

Proposition 20. *With the notations of Definition 71, the following rules are derivable:*

$$\frac{\vdash [\![\nu X\, A]\!]_{\mathrm{e}}, \Delta}{\vdash [\![\nu^{\Gamma-} X\, A]\!]_{\mathrm{e}}, \Delta}\,[\![(\mathrm{LWk}(\Gamma-))]\!]_{\mathrm{e}} \qquad \frac{\vdash [\![\nu^{\Gamma-} X\, A]\!]_{\mathrm{e}}, \Delta}{\vdash [\![\nu^{\Gamma+} X\, A]\!]_{\mathrm{e}}, \Delta}\,[\![(\mathrm{LWk}(\Gamma+))]\!]_{\mathrm{e}}$$

$$\frac{\vdash [\![\nu^{\Gamma-} X\, A]\!]_{\mathrm{e}}, \Gamma}{\vdash [\![\nu X\, A]\!]_{\mathrm{e}}, \Gamma}\,[\![(Rec'(\Gamma))]\!]_{\mathrm{e}} \qquad \frac{}{\vdash [\![\nu^{\Gamma+} X\, A]\!]_{\mathrm{e}}, \Gamma}\,[\![(\bigcirc(\Gamma))]\!]_{\mathrm{e}}$$

Proof. See derivations on Figure 5.4. □

Proposition 21. *If π is a μMALL$^\omega$ preproof with endsequent Γ, if π is labelled with the core fragment of the extended labelling of Definition 70, then π can be turned into a μMALL proof of Γ.*

Proof. As for Theorem 5, it is enough, in order to turn a labelled formula into an unlabelled one, to translate the ν connectives, leaving all other connectives untouched. The translation of Definition 68 is adapted using Definition 71 and picking, for $\Gamma_{(a)}$, the context of the rule $(Rec(a))$.

By Proposition 20, the rules of the core fragment of the extended labelling are translated to usual μMALL$^\omega$ rules involving no label.

Remark moreover that $\dfrac{\vdash [\![A[\nu^{\Gamma+} X\, A[X]]]\!]_{\mathrm{e}}, \Delta}{\vdash [\![\nu^{\Gamma-} X\, A[X]]\!]_{\mathrm{e}}, \Delta}\,(\nu)$ is the usual (ν) rule.

These allow to translate any labelled proof verifying the two constraints of Definition 70 into a μMALL finitary proof. □

5.4 An *ad hoc* finitization of π_∞

The method of the previous section works almost as well for finitizing π_∞ based on the labelling of Figure 5.5: it allows to expand everything concerning the variables a and b.

$$\frac{\vdash [\![\nu X A]\!]_{\mathrm{e}}, \Delta}{\vdash [\![\nu^{\Gamma-} X A]\!]_{\mathrm{e}}, \Delta}\,[\![(\mathrm{LWk}(\Gamma-))]\!]_{\mathrm{e}} \quad =$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \nu X[\![A]\!]_{\mathrm{e}}[X], \mu X[\![A]\!]_{\mathrm{e}}^{\perp}[X]}\,(\mathrm{id})}{\vdash \Gamma^{\perp} \oplus \nu X[\![A]\!]_{\mathrm{e}}[X], \mu X[\![A]\!]_{\mathrm{e}}^{\perp}[X]}\,(\oplus_1)}{\vdash [\![A]\!]_{\mathrm{e}}[\Gamma^{\perp} \oplus \nu X[\![A]\!]_{\mathrm{e}}[X]], [\![A]\!]_{\mathrm{e}}^{\perp}[\mu X[\![A]\!]_{\mathrm{e}}^{\perp}[X]]}\,[[\![A]\!]_{\mathrm{e}}]}{\vdash [\![A]\!]_{\mathrm{e}}[\Gamma^{\perp} \oplus \nu X[\![A]\!]_{\mathrm{e}}[X]], \mu X[\![A]\!]_{\mathrm{e}}^{\perp}[X]}\,(\mu) \qquad \vdash \nu X[\![A]\!]_{\mathrm{e}}[X], \Delta}{\vdash \nu X[\![A]\!]_{\mathrm{e}}[\Gamma^{\perp} \oplus X], \Delta}\,(\nu_{\mathrm{inv}})$$

$$\frac{}{\vdash [\![\nu^{\Gamma+} X A]\!]_{\mathrm{e}}, \Gamma}\,[\![(\circlearrowright(\Gamma))]\!]_{\mathrm{e}} \quad = \quad \frac{\dfrac{}{\vdash \otimes\Gamma^{\perp}, \Gamma}\,(\otimes)^{*}(\mathrm{id})}{\vdash \otimes\Gamma^{\perp} \oplus [\![\nu^{\Gamma-} X A]\!]_{\mathrm{e}}, \Gamma}\,(\oplus_0)$$

$$\frac{\vdash [\![\nu^{\Gamma-} X A]\!]_{\mathrm{e}}, \Delta}{\vdash [\![\nu^{\Gamma+} X A]\!]_{\mathrm{e}}, \Delta}\,[\![(\mathrm{LWk}(\Gamma+))]\!]_{\mathrm{e}} \quad = \quad \frac{\vdash [\![\nu^{\Gamma-} X A]\!]_{\mathrm{e}}, \Delta}{\vdash \otimes\Gamma^{\perp} \oplus [\![\nu^{\Gamma-} X A]\!]_{\mathrm{e}}, \Delta}\,(\oplus_1)$$

$$\frac{\vdash [\![\nu^{\Gamma-} X A]\!]_{\mathrm{e}}, \Gamma}{\vdash [\![\nu X A]\!]_{\mathrm{e}}, \Gamma}\,[\![(Rec'(\Gamma))]\!]_{\mathrm{e}} \quad =$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash [\![A]\!]_{\mathrm{e}}[[\![\nu^{\Gamma+} X A]\!]_{\mathrm{e}}], [\![A]\!]_{\mathrm{e}}[[\![\nu^{\Gamma+} X A]\!]_{\mathrm{e}}]^{\perp}}\,(\mathrm{id})}{\vdash [\![A]\!]_{\mathrm{e}}[[\![\nu^{\Gamma+} X A]\!]_{\mathrm{e}}], [\![\nu^{\Gamma-} X A]\!]_{\mathrm{e}}^{\perp}}\,(\mu) \quad \dfrac{\vdash [\![\nu^{\Gamma-} X A]\!]_{\mathrm{e}}, \Gamma}{\vdash [\![\nu^{\Gamma-} X A]\!]_{\mathrm{e}}, \Im\Gamma}\,(\Im)^{*}}{\vdash [\![\nu^{\Gamma-} X A]\!]_{\mathrm{e}}, [\![\nu^{\Gamma+} X A]\!]_{\mathrm{e}}^{\perp}}\,(\mathrm{cut}) \quad \dfrac{\dfrac{}{\vdash [\![\nu^{\Gamma-} X A]\!]_{\mathrm{e}}, [\![\nu^{\Gamma-} X A]\!]_{\mathrm{e}}^{\perp}}\,(\mathrm{id})}{}\,(\&) \quad \dfrac{\dfrac{}{\vdash \otimes\Gamma^{\perp}, \Gamma}\,(\otimes)^{*}(\mathrm{id})}{\vdash [\![\nu^{\Gamma+} X A]\!]_{\mathrm{e}}, \Gamma}\,(\oplus_0)}{\vdash [\![A]\!]_{\mathrm{e}}[[\![\nu^{\Gamma+} X A]\!]_{\mathrm{e}}], [\![\nu^{\Gamma+} X A]\!]_{\mathrm{e}}^{\perp}}$$

$$\vdash \nu X[\![A]\!]_{\mathrm{e}}, \Gamma\,(\nu_{\mathrm{inv}})$$

Figure 5.4. Derivability of $[\![(\mathrm{LWk}(\Gamma+))]\!]_{\mathrm{e}}$, $[\![(\circlearrowright(\Gamma))]\!]_{\mathrm{e}}$, $[\![(\mathrm{LWk}(\Gamma-))]\!]_{\mathrm{e}}$ and $[\![(Rec'(\Gamma))]\!]_{\mathrm{e}}$

However, as it is, it cannot be applied directly to expand the variable c, for which the second conditions of Definition 70 is not verified. We can anyway finitize π_∞, but at the cost of a somewhat *ad hoc* translation:

Remark 22. In the following formulas, the notation μ_A is used to denote the formula $\mu Z A$ where Z is a variable which does not appear in A.

$$\Gamma_c \coloneqq F, G, H \qquad [\![K^{c-}]\!]_{\mathrm{e}} \coloneqq \nu Y \mu_((\otimes \Gamma_c^\perp \oplus (I'[Y] \parr J'[Y])) \oplus \bot)$$

$$I^{c+} \coloneqq [\![I_+]\!]_{\mathrm{e}} = [\![I'[K^{c+}]]\!]_{\mathrm{e}} \coloneqq \mu_((\otimes \Gamma_c^\perp \oplus (I'[[\![K^{c-}]\!]_{\mathrm{e}}] \parr J'[[\![K^{c-}]\!]_{\mathrm{e}}])) \oplus \bot)$$

$$L^{c+} \coloneqq [\![I'[K^{c+}] \parr J'[K^{c+}]]\!]_{\mathrm{e}} \coloneqq \otimes \Gamma_c^\perp \oplus (I'[[\![K^{c-}]\!]_{\mathrm{e}}] \parr J'[[\![K^{c-}]\!]_{\mathrm{e}}])$$

These definitions allow to make finitary the derivation of Figure 5.5, by expanding every formula as explained above, and by replacing every rule dealing with labels with an appropriate derivation, while leaving untouched the structure of rules not dealing with labels. The result is presented on Figure 5.6a, in an abbreviated form, showing the similarity with the original proof. The left premise of the bottommost (&) of the finitization of π_∞ is fully expanded on Figure 5.7, the two premises of the topmost (&) are fully expanded, respectively, on Figures 5.8 and 5.9.

We now detail the analysis leading to this choice of formulas. To finitize π_∞ we try to apply the same method as for the interleaving example of Figure 5.3 p. 104, by expanding every labelled formula to a non-labelled one and expanding the rules that need it to match these transforms. This works perfectly for H and G, which appear respectively as formulas of the premises (*Rec(b)*) and (*Rec(a)*). But the situation is more delicate for K for which we have to face a double difficulty: in the premise of (*Rec(c)*), K is not a formula of the sequent but a subformula, and it appears in two different formulas.

Let us try to transform this situation into one that would fit our method. First we would like to have only one formula containing K instead of the two I and J. Unfortunately, none of them can be unlabelled without breaking the labelling. Fortunately the solution to that is easy: I, J is simply equivalent to $I \parr J$.

Now we would like $I \parr J$ to be a ν-formula that we could label. We already made use, for the interleaving example of Figure 5.3, of the isomorphism

$$A[\nu X B[A[X]]] \simeq \nu X A[B[X]] \tag{5.1}$$

to turn an almost-ν-formula into a real one. Let us apply that again.

The formula $I \parr J$ is equal to $L'[K]$ where $L'[Y] \coloneqq I'[Y] \parr J'[Y]$, that is: $I \parr J = L'[\nu Y I'[Y]]$. In order to apply an isomorphism of the form (5.1) we would like $I'[Y]$ to be of the form $M'[L'[Y]]$ for a given M'. This is unfortunately not the case as $I'[Y]$ is a subformula of $L'[Y]$. However, a careful examination of the flow of I, J and K along the loops of π_∞ makes apparent the fact that

$$I'[Y] = \mu Z((Z \parr J'[Y]) \oplus \bot) \simeq \mu_((I'[Y] \parr J'[Y]) \oplus \bot) = M'[L'[Y]]$$

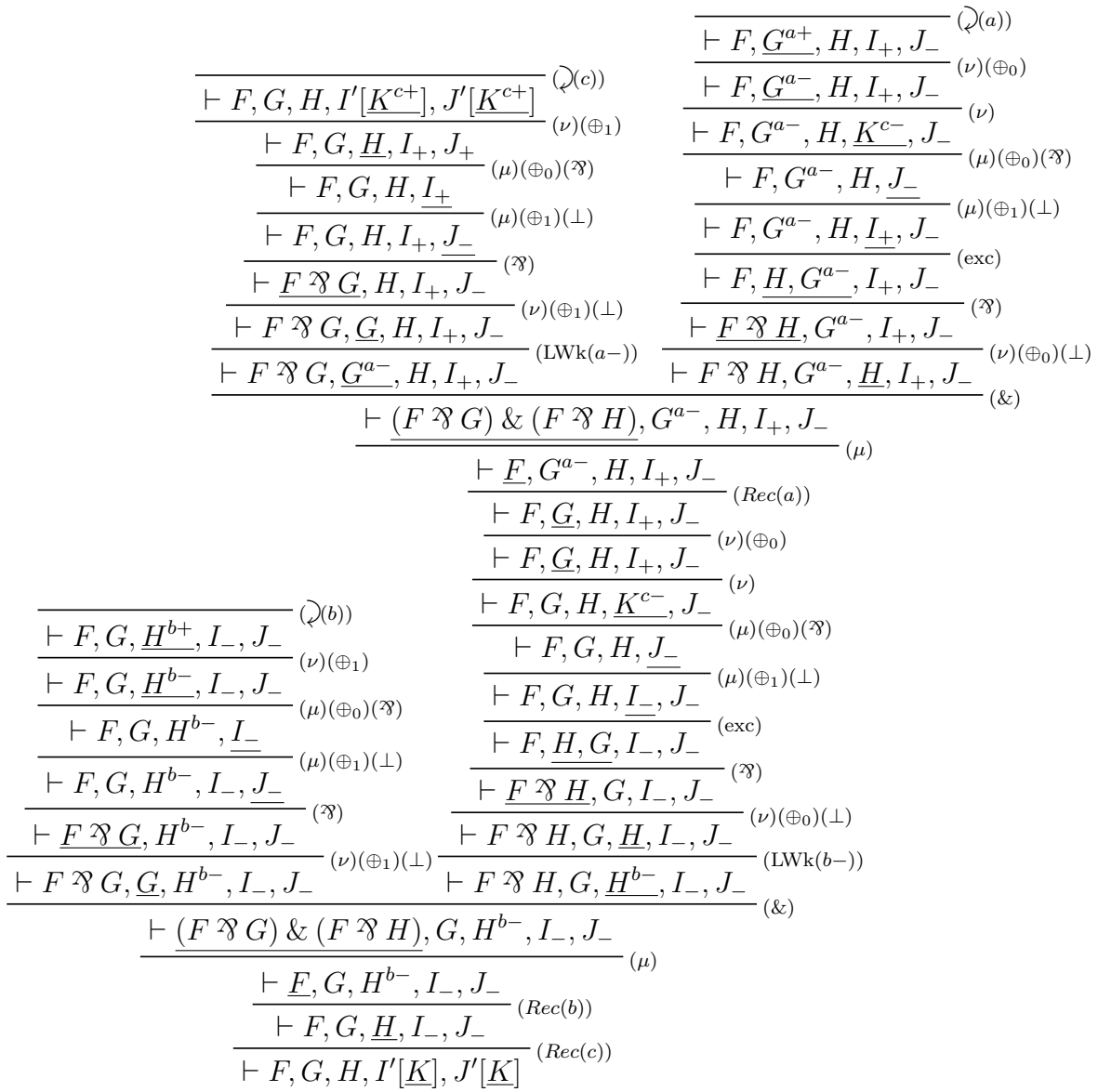


Figure 5.5. Labelling of π_∞. We use the following abbreviations: $I_- = I'[K^{c-}]$, $I_+ = I'[K^{c+}]$, $J_- = J'[K^{c-}]$ and $J_+ = J'[K^{c+}]$.

$$
\dfrac{
\dfrac{
\dfrac{

$$\dfrac{\dfrac{\dfrac{

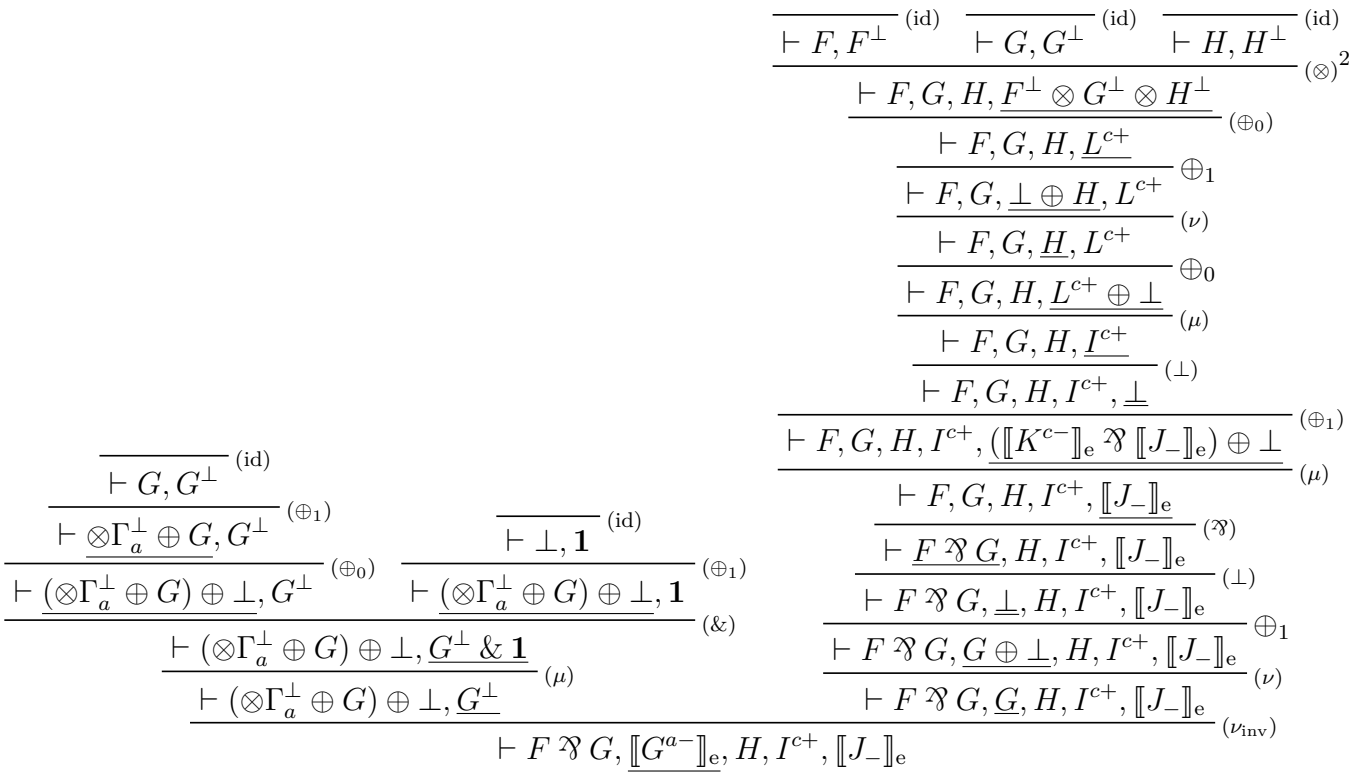


Figure 5.8. Left premise of the topmost (&) of the finitization of π_∞

$$\dfrac{\dfrac{\dfrac{

where

$$M'[Y] \coloneqq \mu_(Y \oplus \bot)$$

in which we use the notation μ_A to denote a μXA with X not appearing free in A. This degenerate μ binder could be removed to simplify the formulas involved in the finitization, but we keep it to stay as close as possible to the original structure of I, trying to preserve its head connective.

When we stick all that together we get

$$I \parr J \simeq L'[\nu Y M'[L'[Y]]] \simeq \nu Y L'[M'[Y]]$$

which is a ν-formula that we know, when labelled, how to expand into an unlabelled formula. If we stopped here our analysis, we would then define:

$$\Gamma_c \coloneqq F, G, H \qquad L^{c-} \coloneqq \nu Y L'[M'[\otimes\Gamma_c^\perp \oplus Y]] \qquad L^{c+} \coloneqq \otimes\Gamma_c^\perp \oplus L^{c-}.$$

However we will do yet a bit more work in order to get the structure of L^{c-} closer to that of $I \parr J$.

Indeed the isomorphism (5.1) can be used in the other direction:

$$\begin{aligned}\nu Y L'[M'[\otimes\Gamma_c^\perp \oplus Y]] &\simeq L'[\nu Y M'[\otimes\Gamma_c^\perp \oplus L'[Y]]] \\ &= I'[\nu Y M'[\otimes\Gamma_c^\perp \oplus L'[Y]]] \parr J'[\nu Y.M'[\otimes\Gamma_c^\perp \oplus L'[Y]]].\end{aligned}$$

This, finally, leads us to define:

$$\Gamma_c \coloneqq F, G, H \qquad \text{and} \qquad K^{c-} \coloneqq \nu Y M'[\otimes\Gamma_c^\perp \oplus L'[Y]]$$

which allows to expand $I'[K^{c-}]$ and $J'[K^{c-}]$. On the other hand, this is not sufficient to define an expansion of K^{c+}, and we still need an *ad hoc* treatment for formulas containing it:

$$\text{“}I'[K^{c+}]\text{”} \coloneqq I^{c+} \coloneqq M'[\otimes\Gamma_c^\perp \oplus L'[K^{c-}]] \qquad \text{“}I'[K^{c+}] \parr J'[K^{c+}]\text{”} \coloneqq L^{c+} \coloneqq \otimes\Gamma_c^\perp \oplus L'[K^{c-}]$$

With these expansions of labelled formulas into unlabelled formulas, we can finitize the derivation of Figure 5.5 into the very close derivation of Figure 5.6a, on which the rules dealing with labelling can be expanded into μMALL derivations.

5.5 Related works on Chapters 4 and 5

Labelling and local certification is the basis of our approach. The idea of labelling μ-formulas to gather information on fixed-points unfoldings is naturally not new, already to be found in fixed-point approximation methods (see [Dax, Hofmann, and Lange, 2006] for instance). The closest work in this direction is Stirling [2014]'s annotated proofs and the application Afshari and Leigh [2017] made of such proofs in obtaining completeness for the modal μ-calculus. Our labelling system works quite differently since only fixed-point operators are labelled while, in Stirling's annotated proofs, every formula is labelled and labels are transmitted to immediate subformulas with a label extension on greatest fixed-points.

Finitization of circular proofs has been recently a very active topic with much research effort on solving Brotherston-Simpson's conjecture. The following recent contributions were made in the setting of Martin-Löf's inductive definitions: firstly, Berardi and Tatsuta [2017b] proved that, in general, the equivalence is false by providing a counter-example inspired by the *Hydra* paradox. Secondly, Simpson [2017] on the one hand and Berardi and Tatsuta [2017a] on the other hand provided a positive answer in the restricted frameworks when the proof system contains arithmetics. While Simpson used tools from reverse mathematics and internalized circular proofs in ACA_0, a fragment of second-order arithmetic with a comprehension axiom on arithmetical statements, Tatsuta and Berardi proved an equivalent result by a direct proof translation relying on an arithmetical version of the Ramsey and Podelsky-Rybalchenko theorems.

Circular proof search triggered interest compared to proof system with explicit inductive invariants (lacking subformula property). As we already mentionned, this has actually been turned to practice by Brotherston, Gorogiannis, and Petersen [2012].

5.6 Conclusion

In this chapter, we contributed to the theory of circular proofs for μMALL in two directions: (i) expanding the results of Chapter 4 by identifying larger fragments of circular proofs for which local conditions account for the validity of circular proof objects (in contrast to the global nature of thread conditions) and (ii) designing methods for

translating circular proofs to finitary proofs (with explicit (co)induction rules). To do so, we introduced and studied several labelling systems, for circular proofs, and made the following contributions:

(i) First, we investigated how such labellings ensure validity of a labellable proof, turning a global and complex problem into a local and simpler one. Indeed, validity-checking is far from trivial in circular proof-theory for fixed-point logics: the best known bound for this problem is generally PSPACE and, in the case of μMALL and similar μ-calculi, we proved in Chapter 3 that checking thread-validity is in fact a PSPACE-complete problem. We extended the results of Chapter 4 by defining a second, more liberal labelling, which does not force back edges to target (ν) inferences. We proved that this extended labelling still ensure thread-validity, and we were able to finitize a fragment of it.

(ii) Second, we provided evidence on the usability of such labellings as a helpful guide in the generation of (co)inductive invariants which are necessary to translate a circular proof in a finitary proof system with (co)induction rules *à la* Park. We provided a full finitization method for the fairly restricted labelling system of Chapter 4, which contains at least all the translations of μMALL proofs as well as for a core fragment of the extented labelling of this chapter, which subsumes the labelling of Chapter 4. However, this fragment is too constrained to treat standard examples that we discuss in the chapter, and which contain most of the difficulties in finitizing circular proofs, namely: (i) interleaving of fixed-points and (ii) interleaving of back edges resulting in various choices of a valid thread to support a branch.

6 Conclusion

6.1 General conclusion

We started this journey by explaining that the already existing finitary systems for least and greatest fixed points of formulas, with induction and coinduction rules [Baelde, 2012], needing explicit invariants were not satisfactory. We explained that the infinitary setting designed by [Baelde, Doumane, and Saurin, 2016] was proposed as a solution to these issues. We then complained about the fact that these infinite proof trees are infinite and explained that this thesis is only concerned with the circular representations of infinite proofs. After proving that the thread criterion is PSPACE-complete on these circular representations, we argued for the search of a smaller, easier to check criterion, which would still be expressive enough for practical use. We provided such a criterion, the loop criterion, and we finally showed how any circular preproof valid for our loop criterion can be turned into a finitary proof à la Baelde with explicit induction and coinduction invariants.

It may seem that after all this work we finally reached our starting point.

But, as one may expect, we learned some valuable things through this journey.

- The result of PSPACE-completeness of the thread criterion (Theorem 4), which is the main contribution of Chapter 3, is, as far as we know, the first of its kind. Not only does it give a precise characterization of the complexity of the thread criterion, but its proof also made apparent the deep link that seems to exist between this global validity condition and program termination techniques such as size-change termination [Lee, Jones, and Ben-Amram, 2001].

- The fragments of μMALL^ω defined in Chapter 4 and in Section 5.2 combine advantages of both μMALL and μMALL^ω. One one side, when they are viewed as unlabelled systems, they are circular systems, in it is not necessary to find and write explicit invariants when introducing a ν or eliminating a μ. On the other side, when they are viewed as circular systems, they do not necessitate a global validity criterion which may be subtle to understand, but they work with basic sequent calculus constructions in which the validity of the proof, which is finite,

comes directly from the validity of each rule. Over all that, going from a labelled proof to an unlabelled one is immediate, and checking and labelling an unlabelled preproof is doable in quadratic time.

- The finitization results of Chapter 5 do not only tell that the different fragments of μMALL^ω studied in Chapter 4 and in Section 5.2 do not have more expressivity than μMALL. These are constructive results, providing an explicite method to synthetize induction and coinduction invariants and to turn a circular proof into a finitary, well-founded one.

6.2 Future works

Our work in Chapter 3 suggests deep connections between thread-validity and size-change termination (SCT) [Lee, Jones, and Ben-Amram, 2001], which was already hinted by other authors [Dax, Hofmann, and Lange, 2006, Hyvernat, 2014, 2019, Lepigre and Raffalli, 2019]. More generally, the connection between SCT and thread validity in μ-calculi is not yet well understood despite those early investigations. More than a connection, this looks like an interplay: size-change termination is originally shown decidable by using Büchi automata and size-change graphs can be used to show validity of circular proofs [Dax, Hofmann, and Lange, 2006]. There seems to be connections with our labelling system too, that we plan to investigate further in the future.

We already underlined in Section 5.5 the similarities between our system and the systems by Dax, Hofmann, and Lange [2006], Stirling [2014], Afshari and Leigh [2017]. Despite their differences, the relationships of those systems should be investigated further (in particular the role of the annotation restriction rule of Stirling [2014, Definition 4]).

In addition to investigating more closely those connections, we have several directions for improving our labelled proof system. The first task is to lift the results of Chapter 4 and Section 5.1 to the extended labelling system of Section 5.2. Indeed, for the more restricted fragment and given a circular proof presented as a graph with back edges, we provided a method to effectively check that one can assign labels. It is therefore natural to expect extending these results to the relaxed framework. Another point we plan to investigate is whether every circular μMALL proof can be labelled. Even though this can look paradoxical given the complexity of checking validity of circular proofs, one should keep in mind that it might well be the case that, in order to label a circular proof presented as a tree with back edges, one has to unfold some of the back edges, or possibly pick a different finite representation of the proof which may result in a space blow up. Related to this question is the connection of our labelling methods with size-change termination methods. Indeed, in designing the extended labelling, one gets closer to the kind of constructions one finds in SCT-based approaches: this should be investigated

further since it may also be a key for our finitization objective. Note that the previous two directions would lead to a solution to the Brotherston-Simpson conjecture.

Our work in Chapter 5 helped closing the gap between μMALL^ω and μMALL by proposing a rich fragment of μMALL^ω that is equivalent μMALL. A very natural question for future work is to extend the still *ad hoc* finitization method presented in the last section to the whole fragment of relaxed labelled proofs, and maybe to all μMALL^ω.

We also wish to investigate the potential use of labellings in circular proof-search. Indeed, there are several different labellings for a given finite derivation with back edges where the labels are weakened. Proposition 15 characterizes least and greatest validating sets: those extremal validating sets correspond to different strategies in placing the labels, which have different properties with respect to the ability to form back edges or to validate the proof that one may exploit in proof-search.

Bibliography

Smbat Abian and Arthur B Brown. A theorem on partially ordered sets, with applications to fixed point theorems. *Canadian Journal of Mathematics*, 13:78–82, 1961.

Peter Aczel. An introduction to inductive definitions. In *Studies in Logic and the Foundations of Mathematics*, volume 90, pages 739–782. Elsevier, 1977.

Bahareh Afshari and Graham E. Leigh. Cut-free completeness for modal mu-calculus. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12. IEEE Computer Society, 2017. doi: 10.1109/LICS.2017.8005088. URL https://doi.org/10.1109/LICS.2017.8005088.

Alfred V. Aho and Jeffrey D. Ullman. The universality of data retrieval languages. In Alfred V. Aho, Stephen N. Zilles, and Barry K. Rosen, editors, *Conference Record of the Sixth Annual ACM Symposium on Principles of Programming Languages, San Antonio, Texas, USA, January 1979*, pages 110–120. ACM Press, 1979. doi: 10.1145/567752.567763. URL https://doi.org/10.1145/567752.567763.

Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009. ISBN 978-0-521-42426-4. URL http://www.cambridge.org/catalogue/catalogue.asp?isbn=9780521424264.

David Baelde. On the proof theory of regular fixed points. In Martin Giese and Arild Waaler, editors, *Automated Reasoning with Analytic Tableaux and Related Methods, 18th International Conference, TABLEAUX 2009, Oslo, Norway, July 6-10, 2009. Proceedings*, volume 5607 of *Lecture Notes in Computer Science*, pages 93–107. Springer, 2009. doi: 10.1007/978-3-642-02716-1_8. URL https://doi.org/10.1007/978-3-642-02716-1_8.

David Baelde. Least and greatest fixed points in linear logic. *ACM Trans. Comput. Log.*, 13(1):2:1–2:44, 2012. doi: 10.1145/2071368.2071370. URL https://doi.org/10.1145/2071368.2071370.

David Baelde and Dale Miller. Least and greatest fixed points in linear logic. In Nachum Dershowitz and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning, 14th International Conference, LPAR 2007, Yerevan,*

Armenia, October 15-19, 2007, Proceedings, volume 4790 of *Lecture Notes in Computer Science*, pages 92–106. Springer, 2007. doi: 10.1007/978-3-540-75560-9_9. URL https://doi.org/10.1007/978-3-540-75560-9_9.

David Baelde, Amina Doumane, and Alexis Saurin. Infinitary proof theory: the multiplicative additive case. In Jean-Marc Talbot and Laurent Regnier, editors, *25th EACSL Annual Conference on Computer Science Logic, CSL 2016, August 29 - September 1, 2016, Marseille, France*, volume 62 of *LIPIcs*, pages 42:1–42:17. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016. doi: 10.4230/LIPIcs.CSL.2016.42. URL https://doi.org/10.4230/LIPIcs.CSL.2016.42.

Stefano Berardi and Makoto Tatsuta. Equivalence of inductive definitions and cyclic proofs under arithmetic. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12. IEEE Computer Society, 2017a. doi: 10.1109/LICS.2017.8005114. URL https://doi.org/10.1109/LICS.2017.8005114.

Stefano Berardi and Makoto Tatsuta. Classical system of martin-löf's inductive definitions is not equivalent to cyclic proof system. In Javier Esparza and Andrzej S. Murawski, editors, *Foundations of Software Science and Computation Structures - 20th International Conference, FOSSACS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings*, volume 10203 of *Lecture Notes in Computer Science*, pages 301–317, 2017b. doi: 10.1007/978-3-662-54458-7_18. URL https://doi.org/10.1007/978-3-662-54458-7_18.

Stefano Berardi and Makoto Tatsuta. Intuitionistic podelski-rybalchenko theorem and equivalence between inductive definitions and cyclic proofs. In Corina Cîrstea, editor, *Coalgebraic Methods in Computer Science - 14th IFIP WG 1.3 International Workshop, CMCS 2018, Colocated with ETAPS 2018, Thessaloniki, Greece, April 14-15, 2018, Revised Selected Papers*, volume 11202 of *Lecture Notes in Computer Science*, pages 13–33. Springer, 2018. doi: 10.1007/978-3-030-00389-0_3. URL https://doi.org/10.1007/978-3-030-00389-0_3.

Stefano Berardi and Makoto Tatsuta. Classical system of martin-lof's inductive definitions is not equivalent to cyclic proofs. *Log. Methods Comput. Sci.*, 15(3), 2019. doi: 10.23638/LMCS-15(3:10)2019. URL https://doi.org/10.23638/LMCS-15(3:10)2019.

Nicolas Bourbaki. Sur le theoreme de zorn. *Archiv der Mathematik*, 2(6):434–437, 1949.

James Brotherston. *Sequent calculus proof systems for inductive definitions*. PhD thesis, University of Edinburgh. College of Science and Engineering. School of . . . , 2006.

James Brotherston and Alex Simpson. Complete sequent calculi for induction and infinite

descent. In *22nd IEEE Symposium on Logic in Computer Science (LICS 2007), 10-12 July 2007, Wroclaw, Poland, Proceedings*, pages 51–62. IEEE Computer Society, 2007. doi: 10.1109/LICS.2007.16. URL https://doi.org/10.1109/LICS.2007.16.

James Brotherston and Alex Simpson. Sequent calculi for induction and infinite descent. *J. Log. Comput.*, 21(6):1177–1216, 2011. doi: 10.1093/logcom/exq052. URL https://doi.org/10.1093/logcom/exq052.

James Brotherston, Nikos Gorogiannis, and Rasmus Lerchedahl Petersen. A generic cyclic theorem prover. In Ranjit Jhala and Atsushi Igarashi, editors, *Programming Languages and Systems - 10th Asian Symposium, APLAS 2012, Kyoto, Japan, December 11-13, 2012. Proceedings*, volume 7705 of *Lecture Notes in Computer Science*, pages 350–367. Springer, 2012. doi: 10.1007/978-3-642-35182-2_25. URL https://doi.org/10.1007/978-3-642-35182-2_25.

Pierre Clairambault. Least and greatest fixpoints in game semantics. In Luca de Alfaro, editor, *Foundations of Software Science and Computational Structures, 12th International Conference, FOSSACS 2009, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, York, UK, March 22-29, 2009. Proceedings*, volume 5504 of *Lecture Notes in Computer Science*, pages 16–31. Springer, 2009. doi: 10.1007/978-3-642-00596-1_3. URL https://doi.org/10.1007/978-3-642-00596-1_3.

Pierre Clairambault. *Logique et Interaction : une Étude Sémantique de la Totalité. (Logic and Interaction : a Semantic Study of Totality)*. PhD thesis, Paris Diderot University, France, 2010. URL https://tel.archives-ouvertes.fr/tel-00459307.

Vincent Danos and Laurent Regnier. The structure of multiplicatives. *Arch. Math. Log.*, 28(3):181–203, 1989. doi: 10.1007/BF01622878. URL https://doi.org/10.1007/BF01622878.

Brian A. Davey and Hilary A. Priestley. *Introduction to Lattices and Order (2. ed.)*. Cambridge University Press, 2002. ISBN 978-0-521-78451-1.

Anuj Dawar and Yuri Gurevich. Fixed point logics. *Bull. Symb. Log.*, 8(1):65–88, 2002. doi: 10.2178/bsl/1182353853. URL https://doi.org/10.2178/bsl/1182353853.

Christian Dax, Martin Hofmann, and Martin Lange. A proof system for the linear time μ-calculus. In S. Arun-Kumar and Naveen Garg, editors, *FSTTCS 2006: Foundations of Software Technology and Theoretical Computer Science, 26th International Conference, Kolkata, India, December 13-15, 2006, Proceedings*, volume 4337 of *Lecture Notes in Computer Science*, pages 273–284. Springer, 2006. doi: 10.1007/11944836_26. URL https://doi.org/10.1007/11944836_26.

J. W. de Bakker and Willem P. de Roever. A calculus for recursive program schemes. In Maurice Nivat, editor, *Automata, Languages and Programming, Colloquium, Paris, France, July 3-7, 1972*, pages 167–196. North-Holland, Amsterdam, 1972.

Amina Doumane. Constructive completeness for the linear-time μ-calculus. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12. IEEE Computer Society, 2017a. doi: 10.1109/LICS.2017.8005075. URL https://doi.org/10.1109/LICS.2017.8005075.

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

Amina Doumane, David Baelde, Lucca Hirschi, and Alexis Saurin. Towards completeness via proof search in the linear time μ-calculus: The case of büchi inclusions. In Martin Grohe, Eric Koskinen, and Natarajan Shankar, editors, *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 377–386. ACM, 2016. doi: 10.1145/2933575.2933598. URL https://doi.org/10.1145/2933575.2933598.

Jérôme Fortier and Luigi Santocanale. Cuts for circular proofs: semantics and cut-elimination. In Simona Ronchi Della Rocca, editor, *Computer Science Logic 2013 (CSL 2013), CSL 2013, September 2-5, 2013, Torino, Italy*, volume 23 of *LIPIcs*, pages 248–262. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013. doi: 10.4230/LIPIcs.CSL.2013.248. URL https://doi.org/10.4230/LIPIcs.CSL.2013.248.

Gerhard Gentzen. Untersuchungen über das logische schließen. i. *Mathematische zeitschrift*, 39(1):176–210, 1935a.

Gerhard Gentzen. Untersuchungen über das logische schließen. ii. *Mathematische Zeitschrift*, 39(1):405–431, 1935b.

Gerhard Gentzen. Investigations into logical deduction. *The Collected Papers of Gerhard Gentzen*, pages 68–131, 1969. URL https://ci.nii.ac.jp/naid/10007157703/en/.

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

Jean-Yves Girard. Proof theory and logical complexity. *Annals of Pure and Applied Logic*, 53(4):197, 1991.

Jean-Yves Girard. *The Blind Spot: lectures on logic*. European Mathematical Society, 2011.

Jean-Yves Girard, Paul Taylor, and Yves Lafont. *Proofs and Types*. Cambridge University Press, New York, NY, USA, 1989. ISBN 0-521-37181-3.

Nikos Gorogiannis and Reuben Rowe. The cyclist theorem prover, 2014. URL `http://www.cyclist-prover.org/`.

Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]*, volume 2500 of *Lecture Notes in Computer Science*, 2002. Springer. ISBN 3-540-00388-6. doi: 10.1007/3-540-36387-4. URL `https://doi.org/10.1007/3-540-36387-4`.

Yuri Gurevich and Saharon Shelah. Fixed-point extensions of first-order logic. *Ann. Pure Appl. Log.*, 32:265–280, 1986. doi: 10.1016/0168-0072(86)90055-2. URL `https://doi.org/10.1016/0168-0072(86)90055-2`.

Pierre Hyvernat. The size-change termination principle for constructor based languages. *Logical Methods in Computer Science*, 10(1), 2014. doi: 10.2168/LMCS-10(1:11)2014. URL `https://doi.org/10.2168/LMCS-10(1:11)2014`.

Pierre Hyvernat. The size-change principle for mixed inductive and coinductive types. *CoRR*, abs/1901.07820, 2019. URL `http://arxiv.org/abs/1901.07820`.

Neil D. Jones. *Computability and complexity - from a programming perspective*. Foundations of computing series. MIT Press, 1997. ISBN 978-0-262-10064-9.

Roope Kaivola. Axiomatising linear time mu-calculus. In Insup Lee and Scott A. Smolka, editors, *CONCUR '95: Concurrency Theory, 6th International Conference, Philadelphia, PA, USA, August 21-24, 1995, Proceedings*, volume 962 of *Lecture Notes in Computer Science*, pages 423–437. Springer, 1995a. doi: 10.1007/3-540-60218-6_32. URL `https://doi.org/10.1007/3-540-60218-6_32`.

Roope Kaivola. A simple decision method for the linear time mu-calculus. In *Structures in Concurrency Theory*, pages 190–204. Springer, 1995b.

Dexter Kozen. On induction vs. *-continuity. In Dexter Kozen, editor, *Logics of Programs, Workshop, Yorktown Heights, New York, USA, May 1981*, volume 131 of *Lecture Notes in Computer Science*, pages 167–176. Springer, 1981. doi: 10.1007/BFb0025782. URL `https://doi.org/10.1007/BFb0025782`.

Dexter Kozen. Results on the propositional mu-calculus. *Theor. Comput. Sci.*, 27: 333–354, 1983. doi: 10.1016/0304-3975(82)90125-6. URL `https://doi.org/10.1016/0304-3975(82)90125-6`.

Jean-Louis Lassez, V. L. Nguyen, and Liz Sonenberg. Fixed point theorems and semantics: A folk tale. *Inf. Process. Lett.*, 14(3):112–116, 1982. doi: 10.1016/0020-0190(82)90065-5. URL `https://doi.org/10.1016/0020-0190(82)90065-5`.

Chin Soon Lee, Neil D. Jones, and Amir M. Ben-Amram. The size-change principle for program termination. In Chris Hankin and Dave Schmidt, editors, *Conference Record of POPL 2001: The 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, London, UK, January 17-19, 2001*, pages 81–92. ACM, 2001. doi: 10.1145/360204.360210. URL `http://doi.acm.org/10.1145/360204.360210`.

Rodolphe Lepigre and Christophe Raffalli. Practical subtyping for curry-style languages. *ACM Trans. Program. Lang. Syst.*, 41(1):5:1–5:58, 2019. doi: 10.1145/3285955. URL `https://doi.org/10.1145/3285955`.

George Markowsky. Chain-complete posets and directed sets with applications. *Algebra universalis*, 6(1):53–68, 1976.

David Park. Fixpoint induction and proofs of program properties. *Machine intelligence*, 5(59-78):5–3, 1969.

Sylvain Perifel. *Complexité algorithmique*. Ellipses, 2014.

Dominique Perrin and Jean-Éric Pin. *Infinite words: automata, semigroups, logic and games*, volume 141. Academic Press, 2004.

Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*, pages 46–57. IEEE Computer Society, 1977. doi: 10.1109/SFCS.1977.32. URL `https://doi.org/10.1109/SFCS.1977.32`.

Reuben N. S. Rowe and James Brotherston. Automatic cyclic termination proofs for recursive procedures in separation logic. In Yves Bertot and Viktor Vafeiadis, editors, *Proceedings of the 6th ACM SIGPLAN Conference on Certified Programs and Proofs, CPP 2017, Paris, France, January 16-17, 2017*, pages 53–65. ACM, 2017. doi: 10.1145/3018610.3018623. URL `https://doi.org/10.1145/3018610.3018623`.

Luigi Santocanale. A calculus of circular proofs and its categorical semantics. In Mogens Nielsen and Uffe Engberg, editors, *Foundations of Software Science and Computation Structures, 5th International Conference, FOSSACS 2002. Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2002 Grenoble, France, April 8-12, 2002, Proceedings*, volume 2303 of *Lecture Notes in Computer Science*, pages 357–371. Springer, 2002. doi: 10.1007/3-540-45931-6_25. URL `https://doi.org/10.1007/3-540-45931-6_25`.

Alex Simpson. Cyclic arithmetic is equivalent to peano arithmetic. In Javier Esparza and Andrzej S. Murawski, editors, *Foundations of Software Science and Computation Structures - 20th International Conference, FOSSACS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings*, volume 10203 of *Lecture Notes in Computer Science*, pages 283–300, 2017. doi: 10.1007/978-3-662-54458-7_17. URL https://doi.org/10.1007/978-3-662-54458-7_17.

A. Prasad Sistla, Moshe Y. Vardi, and Pierre Wolper. The complementation problem for büchi automata with appplications to temporal logic. *Theor. Comput. Sci.*, 49: 217–237, 1987. doi: 10.1016/0304-3975(87)90008-9. URL https://doi.org/10.1016/0304-3975(87)90008-9.

Colin Stirling. A tableau proof system with names for modal mu-calculus. In Andrei Voronkov and Margarita V. Korovina, editors, *HOWARD-60: A Festschrift on the Occasion of Howard Barringer's 60th Birthday*, volume 42 of *EPiC Series in Computing*, pages 306–318. EasyChair, 2014. URL http://www.easychair.org/publications/?page=1932281032.

Gadi Tellez and James Brotherston. Automatically verifying temporal properties of pointer programs with cyclic proof. In Leonardo de Moura, editor, *Automated Deduction - CADE 26 - 26th International Conference on Automated Deduction, Gothenburg, Sweden, August 6-11, 2017, Proceedings*, volume 10395 of *Lecture Notes in Computer Science*, pages 491–508. Springer, 2017. doi: 10.1007/978-3-319-63046-5_30. URL https://doi.org/10.1007/978-3-319-63046-5_30.

Moshe Y. Vardi. A temporal fixpoint calculus. In Jeanne Ferrante and P. Mager, editors, *Conference Record of the Fifteenth Annual ACM Symposium on Principles of Programming Languages, San Diego, California, USA, January 10-13, 1988*, pages 250–259. ACM Press, 1988. doi: 10.1145/73560.73582. URL https://doi.org/10.1145/73560.73582.

Igor Walukiewicz. 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*, pages 136–146. IEEE Computer Society, 1993. doi: 10.1109/LICS.1993.287593. URL https://doi.org/10.1109/LICS.1993.287593.

Igor Walukiewicz. Completeness of kozen's axiomatisation of the propositional mu-calculus. In *Proceedings, 10th Annual IEEE Symposium on Logic in Computer Science, San Diego, California, USA, June 26-29, 1995*, pages 14–24. IEEE Computer Society, 1995. doi: 10.1109/LICS.1995.523240. URL https://doi.org/10.1109/LICS.1995.523240.