PSPACE-Completeness of a Thread Criterion for Cyclic Proofs in Linear Logic with Least and Greatest Fixed Points

Rémi Nollet, Alexis Saurin, Christine Tasson

PSPACE-Completeness of a Thread Criterion for Circular Proofs in Linear Logic with Least and Greatest Fixed Points*

Rémi Nollet[1,2,3], Alexis Saurin[3,2,1,4], and Christine Tasson[1,2,3]

[1] Universit de Paris
[2] IRIF, Paris, France, irif.fr
[3] CNRS
[4] INRIA πr^2
https://www.irif.fr/∼{nollet,saurin,tasson}
{nollet,saurin,tasson}@irif.fr

Abstract. In the context of logics with least and greatest fixed points, circular (ie. non-wellfounded but regular) proofs have been proposed as an alternative to induction and coinduction with explicit invariants. However, those proofs are not wellfounded and to recover logical consistency, it is necessary to consider a validity criterion which differentiates valid proofs among all preproofs (i.e. infinite derivation trees).
The paper 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. We prove that the problem of deciding thread-validity for μMALL 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.

Keywords: sequent calculus · non-wellfounded proofs · circular proofs · induction · coinduction · fixed points · linear logic · mu-MALL · size-change · PSPACE-complete · complexity

1 Introduction

The search for proofs of formulas or theorems is one of the fundamental and difficult tasks in proof theory. In the usual setting, those proofs should be easy to check and thus finite. Induction and coinduction principles have been used in order to provide such a finite proof theory for reasoning on formulas with least or greatest fixed points (see Kozen [10, 11] and Baelde [2]). However in those finite systems, the inference rule for greatest fixed points does not preserve the subformula property. As a consequence, the proof search cannot be driven by the

* Supported by ANR RAPIDO. An extended version of this article is available at https://hal.archives-ouvertes.fr/hal-02173207.

formula that we aim to prove. This is one reason why infinite proofs have been considered for logic with fixed points. 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. However, checking the validity criterion might be complex and the purpose of this paper is to show that it is PSPACE-complete. Let us get into more details.

Circular proofs, which are infinite proofs satisfying the validity criteria, have thus been proposed as an alternative to induction and coinduction with explicit invariants. Circular 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 [3, 6].

Circular proofs have already proved useful in implementing efficient automatic provers, *e. g.* the Cyclist prover [1]. However, the complexity avoided in the search, thanks to the subformula property and the fact that we need not guess invariants, is counterbalanced by the complexity of the validity criterion at the time of proof checking.

There are already polynomial-space and exponential-time methods to decide thread validity criterions in several settings, but there was no lower bound on its complexity and the exact complexity of checking the thread criterion was still unknown.

The contribution of this work is to show that, in the setting of linear logic with least and greatest fixed point, the decidability of thread criterion is PSPACE-complete.

Thread validity and size-change termination. Our proof takes a lot of inspiration from the proof of PSPACE-completeness of size-change termination by Lee, Jones and Ben Amram [12]: 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.

Organization of the paper. In section 2 we recall the formulas and rules of linear logic with least and greatest fixed points, as well as the notions of preproofs and the thread validity criterion, and we recall that the thread criterion is effectively decidable in PSPACE. The main section of the article is section 3, in which

we show the PSPACE-completeness of the thread criterion for μMALL^ω, in theorem 1. Section 4 is devoted to a discussion of our approach and a comparison with related works. We conclude in section 5.

2 Background on Circular Proofs and Thread Validity

In this section, we recall the definition of the logic μMALL^ω.

2.1 Formulas

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

Definition 1 (μMALL^ω 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.

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 2 (μMALL^ω formulas). *A* formula *is a closed preformula. We denote by $\mathcal{F}$ the set of all formulas.*

Definition 3 (μMALL^ω negation). *An involutive negation $\cdot^\perp$ is defined on every μMALL^ω preformula, inductively specified by:*

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

Example 1. 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 1. 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 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.

2.2 Sequents and Preproofs

Proofs of $\mu\mathrm{MALL}^\omega$ are selected among a set of preproofs. Preproofs of $\mu\mathrm{MALL}^\omega$ are circular objects, defined by adding back-edges to ordinary proof-trees.

In this article, a sequent is a list of formulas. The inference rules of $\mu\mathrm{MALL}^\omega$ are given below

$$\frac{}{\vdash A, A^\perp}\,(\mathrm{id}) \qquad \frac{\vdash A, \Gamma \quad \vdash A^\perp, \Delta}{\vdash \Gamma, \Delta}\,(\mathrm{cut}) \qquad \frac{\vdash A_{\sigma(0)}, \dots, A_{\sigma(n-1)}}{\vdash A_0, \dots, A_{n-1}}\,(\mathrm{exc})$$

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

$$\frac{\vdash \Gamma, A}{\vdash \Gamma, A \oplus B}\,(\oplus^1) \qquad \frac{\vdash \Gamma, B}{\vdash \Gamma, A \oplus B}\,(\oplus^2) \qquad \frac{\vdash \Gamma, A \quad \vdash \Gamma, B}{\vdash \Gamma, A \,\&\, B}\,(\&) \qquad \frac{}{\vdash \Gamma, \top}\,(\top)$$

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

Note that, in the exchange rule (exc), σ must be a permutation of $\{0, 1, \dots, n-1\}$. The (exc) rules are generally left implicit in descriptions of proof trees.

Definition 4 ($\Pi_0(\mu\mathrm{MALL}^\omega)$: preproofs). *A $\mu\mathrm{MALL}^\omega$ preproof consists of a finite proof tree π, composed using the rules given above, and which may have open sequents*[5]*, together with a function* **back***, which associate to each occurrence s of an open sequent in π, an occurrence* $\mathbf{back}(s)$ *of the same sequent in π, such that* $\mathbf{back}(s)$ *is strictly below s in π (i. e. closer to the root).*

We denote by $\Pi_0(\mu\mathrm{MALL}^\omega)$ the set of all $\mu\mathrm{MALL}^\omega$ preproofs.

Example 2. Let π be the following proof tree, with three open sequents, and let us denote by $s_0, \dots, s_8$ its occurrences of sequents, as indicated:

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

[5] We call an occurrence of open sequent any occurrence of sequent which is not the conclusion of an inference. In Example 2, s_4, s_6 and s_8 are occurrences of open sequents.

then $(\pi, \{s_4 \mapsto s_1, s_6 \mapsto s_5, s_8 \mapsto s_7\})$ is a preproof of μMALL^ω, that we will more simply denote by:

$$\dfrac{\dfrac{\dfrac{(0)}{\vdash \nu X(X \parr X), \mu XX} \quad \dfrac{\dfrac{(1)}{\vdash \nu XX, \nu X(X \parr X), \mu XX}}{(1) \vdash \nu XX, \nu X(X \parr X), \mu XX}\,(\nu)}{\vdash \nu X(X \parr X), \nu X(X \parr X), \mu XX}\,(\text{cut})}{\dfrac{\vdash \nu X(X \parr X) \parr \nu X(X \parr X), \mu XX}{(0) \vdash \nu X(X \parr X), \mu XX}\,(\nu)}\,(\parr) \qquad \dfrac{(2)}{\vdash \nu XX}}{\vdash \nu X(X \parr X)}\,(\text{cut})$$

2.3 Proofs

The validity criterion used to distinguish proofs among preproofs is given in Definition 11 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. Note that:

- 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.
- If $(\pi, \mathbf{back})$ is a preproof, we say that an occurrence of a sequent in π is closed when it is not an open sequent *i. e.* it is the conclusion of some inference in π.

Definition 5 (G_{branch}, branch graph of a preproof). *Let $(\pi, \mathbf{back})$ be a μMALL^ω preproof. Its branch graph is the graph G_{branch} defined as follows. The vertices of G_{branch} are the occurrences 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 occurrence of sequent in π, and from s to $\mathbf{back}(s')$ if s' is an open occurrence of sequent in π.*

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

Definition 6 (G_{thread}, thread graph of a preproof). *Let $(\pi, \mathbf{back})$ be a μMALL^ω preproof. Its thread graph is the graph G_{thread} defined as follows. The vertices of G_{thread} are the occurrences 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 occurrence of formula β in s' which has an immediate descendant α in s, there is an edge in G_{thread}, from α to β if s' is a closed occurrence of sequent in π, and from α to the occurrence of the formula corresponding to β in $\mathbf{back}(s')$ if s' is an open occurrence of sequent in π.*

Definition 7 (Infinite branch). *If $(\pi, \mathbf{back})$ is a 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.*

Example 3. The infinite branches of the preproof of Example 2 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

1. 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;
2. include the implicit (exc) rules.

These details are omitted here for concision; they will cause no ambiguity on the validity of this preproof.

Definition 8 (Thread). *A thread in a preproof is simply a path (finite or infinite) in G_{thread}.*

Example 4. Let us denote by $\{\alpha, \beta, \gamma, \dots, \mu\}$ the vertices of G_{thread} for the preproof shown on Example 2, as indicated here:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(0)}{\vdash \nu X(X \parr X), \mu XX} \quad \dfrac{\dfrac{(1)}{\vdash \nu XX, \nu X(X \parr X), \mu XX}}{(1) \vdash \nu XX_\iota, \nu X(X \parr X)_\kappa, \mu XX_\lambda}(\nu)}{\vdash \nu X(X \parr X)_\zeta, \nu X(X \parr X)_\eta, \mu XX_\theta}(\text{cut})}{\vdash \nu X(X \parr X) \parr \nu X(X \parr X)_\delta, \mu XX_\epsilon}(\parr)}{(0) \vdash \nu X(X \parr X)_\beta, \mu XX_\gamma}(\nu) \quad \dfrac{\dfrac{(2)}{\vdash \nu XX}}{(2) \vdash \nu XX_\mu}(\nu)}{\vdash \nu X(X \parr X)_\alpha}(\text{cut})$$

The maximal threads of this preproof are $(\mu)^\omega$, $\gamma\epsilon\theta(\lambda)^\omega$, $(\iota)^\omega$, $\alpha(\beta\delta\zeta)^\omega$ 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 occurrences of formulas in the sequents that are hidden by the elision of the (exc) rules.

Definition 9 ($\mathbf{U}\colon G_{\text{thread}} \to G_{\text{branch}}$). *For any preproof, there is an obvious graph morphism from G_{thread} to G_{branch}, associating to every occurrence of a formula the sequent occurrence it belongs to. We denote this graph morphism by $\mathbf{U}$. If t is a path in G_{thread} (i. e. a thread), we will also denote by $\mathbf{U}(t)$ the corresponding path in G_{branch}.*

Remark 2. Even when t is an infinite thread, $\mathbf{U}(t)$ may not be an infinite branch because it may not start at the root of the preproof. However, if t is an infinite thread, then $\mathbf{U}(t)$ is a suffix of an infinite branch.

Example 5. The images, by this morphism, of the threads of Example 4 are

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

$$\mathbf{U}(\alpha(\beta\delta\zeta)^\omega) = s_0(s_1s_2s_3)^\omega$$

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

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

Lemma 1. *Let $t = (s_n)_{n\in\mathbf{N}}$ be an infinite thread in a preproof. Let $\inf(t) = \{A \in \mathcal{F} \mid \forall n_0 \in \mathbf{N}, \exists n \geqslant n_0, s_n$ is principal and $\mathbf{fml}(s_n) = A\}$ i. e. the set of formulas that are infinitely often principal in t.*

If $\inf(t) \neq \emptyset$, i. e. if t encounters infinitely often principal formulas, then it contains a smallest infinitely principal formula, and this formula is a fixed point formula : $\exists \sigma \in \{\mu, \nu\}, \exists C, \sigma XC \in \inf(t)$ and $\forall A \in \inf(t), \sigma XC$ is a subformula of A. As a minimum, this formula is unique.

Definition 10 (Valid thread). *An infinite thread t is valid if $\inf(t)$ is non-empty and the smallest formula in $\inf(t)$ is a ν-formula (cf. Lemma 1 just above).*

Example 6. Among the threads of Example 4:

- $(\mu)^\omega$ and $(\iota)^\omega$ are valid: their smallest infinitely principal formula is $\nu X X$;
- $\alpha(\beta\delta\zeta)^\omega$ is valid: its smallest infinitely principal formula is $\nu X(X \parr X)$;
- $\gamma\epsilon\theta(\lambda)^\omega$ 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 occurrence of β.

Definition 11 ($\Pi(\mu\mathrm{MALL}^\omega)$: proofs). *We say that an infinite branch b of a 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 $\mu\mathrm{MALL}^\omega$ preproof ϖ is a proof if all its infinite branches are valid.

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

Example 7. The preproof of Example 2 is a proof:

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

2.4 Deciding Thread Validity in PSPACE

In this section, we recall the fact that the problem $\Pi(\mu\mathrm{MALL}^\omega)$ is in PSPACE. Several algorithms can be used for that. Here we reduce this problem to the problem of deciding equality of languages for parity ω-automata, which is known to be in PSPACE. 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.

Let $\varpi = (\pi, \mathbf{back})$ be a preproof. Let $A = E_{\mathrm{branch}}$, the set of edges of G_{branch}; this will be the input alphabet of our automata.

The first ω-automaton is $\mathcal{A}_{\mathrm{branch}} = \langle Q_{\mathrm{branch}}, i_{\mathrm{branch}}, T_{\mathrm{branch}}\rangle$, where:

– the set of states is $Q_{\text{branch}} = V_{\text{branch}}$, the set of vertices of G_{branch}
– the initial state i_{branch} is the root of π
– the set of transitions is

$$T_{\text{branch}} = \{s \xrightarrow{e} s' \mid e \text{ is an edge from } s \text{ to } s' \text{ in } G_{\text{branch}}\}$$

and the acceptance condition is trivial: every infinite run is accepted. With that definition, the following lemma is immediate:

Lemma 2. *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 $\Omega\colon \mathcal{F} \to \mathbf{N}$ with two properties:

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

Such a function is not difficult to construct.From now on we assume that one has been chosen.

Our second automaton is a parity ω-automaton, with priorities in $\mathbf{N} \cup \{\infty\}$, defined as $\mathcal{A}_{\text{thread}} = \langle Q_{\text{thread}}, i_{\text{thread}}, T_{\text{thread}}\rangle$, where:

– the set of states is $Q_{\text{thread}} = V_{\text{thread}} + \{\bot_s \mid s \in V_{\text{branch}}\}$, *i. e.* the vertices of G_{thread} plus one extra vertex for each vertex of G_{branch}
– the initial state is $i_{\text{thread}} = \bot_r$ where r is the root of π
– the set of transitions is

$$\begin{aligned} T_{\text{thread}} = \quad & \{\bot_s \xrightarrow{e\colon \infty} \bot_{s'} \mid e \text{ is an edge from } s \text{ to } s' \text{ in } G_{\text{branch}}\} \\ & \cup \{\ \alpha \xrightarrow{\mathbf{U}(e)\colon \Omega(\alpha)} \beta \ \mid \\ & \qquad e \text{ is an edge from } \alpha \text{ to } \beta \text{ in } G_{\text{thread}} \text{ and } \alpha \text{ is principal}\} \\ & \cup \{\ \alpha \xrightarrow{\mathbf{U}(e)\colon \infty} \beta \ \mid \\ & \qquad e \text{ is an edge from } \alpha \text{ to } \beta \text{ in } G_{\text{thread}} \text{ and } \alpha \text{ is not principal}\} \\ & \cup \{\bot_s \xrightarrow{\epsilon\colon \infty} \alpha \mid s = \mathbf{U}(\alpha)\} \end{aligned}$$

where $q \xrightarrow{e\colon i} q'$ denote a transition from state $q \in Q_{\text{thread}}$ to state $q' \in Q_{\text{thread}}$ with label $e \in A$ and priority $i \in \mathbf{N} \cup \{\infty\}$.

The acceptance condition is: a run is accepted if the smallest priority appearing infinitely often is *odd* (∞ being even).

Once again, it should be clear from Definitions 10 and 11 that:

Lemma 3. *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 1. *We have the inclusion $\mathcal{L}(\mathcal{A}_{\text{thread}}) \subseteq \mathcal{L}(\mathcal{A}_{\text{branch}})$ and the pre-proof ϖ is valid iff. this inclusion is an equality.*

Deciding this equality can be done in PSPACE, and the constructions of these automata are obviously PSPACE, so:

Proposition 2. *The problem $\Pi(\mu\text{MALL}^{\omega})$ is in* PSPACE.

3 PSPACE-Completeness

3.1 Outline of the PSPACE-Completeness Proof

We now aim at proving that $\Pi(\mu\text{MALL}^\omega)$ is PSPACE-complete for LOGSPACE reductions. As it is already known that $\Pi(\mu\text{MALL}^\omega) \in \text{PSPACE}$, it remains to prove that we have $\text{PSPACE} \leqslant_{\text{L}} \Pi(\mu\text{MALL}^\omega)$.

We follow the same methodology as Lee, Jones and Ben Amram [12]: in order to prove that deciding size-change termination is PSPACE-complete, they define a notion of boolean program (see Definition 12) and use the fact that the following problem 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 decidability of size-change termination.

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

3.2 Defining the Reduction

Let us first introduce boolean programs.

Definition 12 ($\texttt{BOOLE}_{false}$ and $\mathcal{B}_{false}$). *A boolean program in* BOOLE *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 and labels* ℓ', ℓ'' *range overs* $\{0, \ldots, m\}$.

The semantics is as expected: a program is executed together with a store assigning values to variables which shall initially assign all variables to `false` *at the beginning of the execution (this is the* initial store*). More precisely, an execution is a sequence of pairs* (ℓ, s) *of a label and a store subject to the expected transitions* $(\ell_1, s_1) \to (\ell_2, s_2)$ *if* $\ell_1 \neq 0$ *and:*

- *if* $I_{\ell_1} = X := \neg X$*, then* $\ell_2 = \ell_1 + 1\ (\texttt{mod}\ m + 1)$ *and* $s_2(Y) = s_1(Y)$ *for all variable* $Y \neq X$ *and* $S_2(X) = \neg(s_1(X))$*;*
- *if* $I_{\ell_1} = \text{if } \texttt{X} \text{ then goto } \ell' \text{ else goto } \ell''$ *then* $s_2 = s_1$ *and* $\ell_2 = \ell'$ *if* $s_1(X) = \texttt{true}$ *and* $\ell_2 = \ell''$ *otherwise.*

The program terminates when the label reaches 0*, the current store at termination is the* final store*.*

A program in $\texttt{BOOLE}_{false}$ *is a program in* BOOLE *such that, if it terminates, its final store is such that all variables have value false. We also denote the set of terminating* $\texttt{BOOLE}_{false}$ *programs as:*

$$\mathcal{B}_{false} = \{b \in \texttt{BOOLE}_{false} \mid b \textit{ terminates}\}.$$

Remark 3. The constraint on the values of the variables at the end of the program will be useful when reducing it to $\Pi(\mu\mathrm{MALL}^\omega)$. This circular 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 4. *$\mathcal{B}_{false}$ is* PSPACE*-hard under* LOGSPACE*-reductions:*

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

Proof. We reduce from the problem of termination for a more expressive language, which has been defined and proved PSPACE-complete by Jones in [9], under the name of BOOLE.

The following definition will be used in the proof of Proposition 3:

Definition 13 (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*

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

Definition 14 ($[\![\cdot]\!] \colon \mathtt{BOOLE}_{false} \to \Pi_0(\mu\mathrm{MALL}^\omega)$). *For every boolean program $b \in \mathtt{BOOLE}_{false}$, 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 $(D_n), (E_n)$:*

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

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

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

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

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

where Γ^n is an abbreviation for $\underbrace{\Gamma, \dots, \Gamma}_{n}$.

The root of the preproof $[\![b]\!]$ is constructed by translating each pair $\ell\mathtt{:I}_\ell$ of a label and an instruction into a finite segment of branch of preproof, as defined in eq. (1)*, with each subderivation $[\![\ell\mathtt{:I}_\ell]\!]$ defined in Fig. 2 and each subderivation $[\![\ell\mathtt{:goto}\ \ell']\!]$ in Fig. 1.*

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

$$\dfrac{\dfrac{\dfrac{\text{Back-edge to (Root)}}{\vdash A^{2k}, B, C, D_2, D_m, E_m, \ldots, E_m}\,(\nu)^{m-1}}{\vdash A^{2k}, B, C, D_2, D_m, (\underline{\nu X \lambda_m X})^{\ell'-1}, E_m, (\underline{\nu X \lambda_m X})^{m-\ell'}}\,(\mu), (\oplus^2), (\parr)^{m-1}}{\vdash A^{2k}, B, C, D_2, D_m, \underline{E_m}}\,\big((\mu), (\oplus^1), (\bot)\big)^{m-1}$$
$$\vdash A^{2k}, B, C, D_2, D_m, \underline{E_m}^{\ell-1}, E_m, \underline{E_m}^{m-\ell}$$

Fig. 1. Back-edges of the preproof

Remark 4 (Implicit vs. explicit exchange rules). Notice that in the translation of the previous definition, our derivations make an implicit use of the exchange rule. In order to make explicit the exchange, it is enough to add an exchange rule at the conclusion of every inference in the proof, simply doubling the size of the proof. This will therefore have no impact on the forthcoming reductions and completeness proofs that will be studied in the remaining of the paper.

Remark 5 (Infinite branches of $[\![b]\!] \simeq E^\omega$). The preproof $[\![b]\!]$ constructed from b by the reduction $[\![\cdot]\!]$ of Definition 14 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 13). 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 $\overline{u} \in E^\omega$.

3.3 Main Theorem

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

Remark 6 (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 $[\![\cdot]\!]$ (Definition 14) that a valid thread in the preproof cannot pass

$$[\![\texttt{0:goto 1}]\!]$$

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

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

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

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

Fig. 2. Premises p_ℓ of the preproof

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 5. *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 (Proof sketch). By case on the instructions involved.

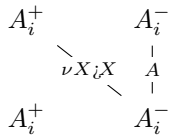
In order to prove the first part of the statement, that is that an infinite branch $\overline{u} \in E^\omega$ in the preproof contains a validating thread in the E_m group iff. no suffix of $\overline{u}$ is a valid path in G, we reason by case on the instructions involved and

remark that the E_m formulas are touched only in the $[\![\ell\texttt{:goto}\ \ell']\!]$ parts of the preproof.

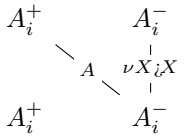
In order to prove the second part of the statement, that is that an infinite branch $\overline{u} \in E^\omega$ in the preproof contains a validating thread in the B, C group iff. 0 occurs only finitely in $\overline{u}$, we reason by case on the instructions involved.

Remark 7. Because of lemma 5, 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 $0 \xrightarrow{0} 1$, and because of the hypothesis that $\overline{u}$ is a path in G, for $n \geqslant 1$ every u_n has source and target $1 \xrightarrow{u_n}{}^+ 0$.

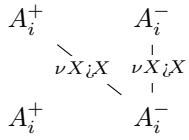
Lemma 6. *Assume $1 \xrightarrow{u}{}^+ \ell$, which does not contain the edge 0. If u is a prefix of the execution of b then the threads of $\{A_i^+, A_i^-\}$ in $0 \xrightarrow{0u}{}^+ \ell$ are*



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



Proof (Proof sketch). The proof goes by induction on the length of u.

The diagrams we use here are sketches of the thread structure of a segment of branch. For instance the first of these diagrams should be read as: the occurrence A_i^- in the conclusion sequent is a descendant of both occurrences A_i^+ and A_i^- in the sequent at the top of the segment of branch we consider. The smallest principal formula along the segment of thread from the lower A_i^- to the upper A_i^+ is $\nu X \text{¿} X$ and the smallest principal formula along the segment of thread from the lower A_i^- to the upper A_i^- is A. The occurrence A_i^+ in the lower sequent is not a descendant of any of the occurrences A_i^+ nor A_i^- in the upper sequent.

Proposition 3. $[\![\cdot]\!]$ *is a* LOGSPACE *reduction from* $\overline{\Pi(\mu\mathrm{MALL}^\omega)}$ *to* $\mathcal{B}_{false}$.

Proof. For the LOGSPACE character: the only data that need to be remembered while constructing the preproof are integers like k, m, ℓ, ℓ'. As $\ell, \ell' \leqslant m$ 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 $b \in \mathtt{BOOLE}_{false}$ and prove that $[\![b]\!] \notin \Pi(\mu\mathrm{MALL}^\omega) \Leftrightarrow b \in \mathcal{B}_{false}$. Let $G = (V, E)$ be the call-graph of b, as defined in Definition 13. Following remark 5, we denote by elements of E^ω the infinite branches of $[\![b]\!]$. There are two cases: either $b \in \mathcal{B}_{false}$ and we have to prove that $p \notin \Pi(\mu\mathrm{MALL}^\omega)$, or $b \notin \mathcal{B}_{false}$ and we have to prove that $p \in \Pi(\mu\mathrm{MALL}^\omega)$. First case: if $b \in \mathcal{B}_{false}$: 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 $b \notin \mathcal{B}_{false}$: 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 1. *The problem* $\Pi(\mu\mathrm{MALL}^\omega)$ *is* PSPACE*-hard under* LOGSPACE *reductions :*

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

Proof. We reduce from $\mathcal{B}_{false}$, which is PSPACE-complete by Lemma 4. More precisely, we reduce $\mathcal{B}_{false}$ 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 14. It is indeed a LOGSPACE reduction by Proposition 3.

Remark 8. 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.

Remark 9. Our result extends to μLJ, μLK, $\mu\mathrm{LK}\bigcirc$ and $\mu\mathrm{LK}\Box\Diamond$ and we conjecture that the method we illustrate here on μMALL can apply as well to the guarded cases of μ-calculi with modalities.

4 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 [12]. 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 $[\![\ell\texttt{:goto } \ell']\!]$ 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 $[\![\ell\texttt{:goto } \ell']\!]$ 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 tree-with-back-edges-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 PTIME. The following boolean program:

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

will be translated to a graph-shaped preproof of size $\Theta(n)$ but the unfolding of this preproof into a tree-with-back-edges-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 [12]), 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. A first step in this direction has already been done in [14].

We recalled in section 2.4 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 [12] 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 in [5] by Dax, Hofmann and Lange, who remark furthermore that this method leads to a more efficient implementation than the automata-based one.

5 Conclusion

In the present paper, 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 size-change termination even though we depart at some crucial points in order to build our reduction to take into account the specific form of circular proofs.

We conjecture that our proof adapts straightforwardly to a number of other circular proof systems based on sequent calculus such as intutionistic or classical proof systems in addition to the linear case on which we have focused here.

While our result can be seen as negative for circular proofs, it does not prevent actual implementations from being tractable and usable in many situations as exemplified by the Cyclist prover for instance. In such systems, validity checking does not seem to be the bottleneck in circular proof construction compared with the complexity that is inherent in exploring and backtracking in the search tree [4, 15, 16].

Our work suggests deep connections between thread-validity and size-change termination, which we only touched upon in the previous section. This confirms connections previously hinted by other authors [5, 7, 8, 13] that we plan to investigate further in the future.

Acknowledgements

A special thanks must go to Anupam Das and Reuben Rowe, and to the anonymous reviewer, for their very complete and most relevant comments.

References

1. The cyclist theorem prover, http://www.cyclist-prover.org/

2. Baelde, D.: Least and greatest fixed points in linear logic. ACM Trans. Comput. Log. **13**(1), 2:1–2:44 (2012). https://doi.org/10.1145/2071368.2071370, https://doi.org/10.1145/2071368.2071370
3. Baelde, D., Doumane, A., Saurin, A.: Infinitary proof theory: the multiplicative additive case. In: Talbot, J., Regnier, L. (eds.) 25th EACSL Annual Conference on Computer Science Logic, CSL 2016, August 29 - September 1, 2016, Marseille, France. LIPIcs, vol. 62, pp. 42:1–42:17. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2016). https://doi.org/10.4230/LIPIcs.CSL.2016.42, https://doi.org/10.4230/LIPIcs.CSL.2016.42
4. Brotherston, J., Gorogiannis, N., Petersen, R.L.: A generic cyclic theorem prover. In: Jhala, R., Igarashi, A. (eds.) Programming Languages and Systems - 10th Asian Symposium, APLAS 2012, Kyoto, Japan, December 11-13, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7705, pp. 350–367. Springer (2012). https://doi.org/10.1007/978-3-642-35182-2_25, https://doi.org/10.1007/978-3-642-35182-2_25
5. Dax, C., Hofmann, M., Lange, M.: A proof system for the linear time μ-calculus. In: Arun-Kumar, S., Garg, N. (eds.) FSTTCS 2006: Foundations of Software Technology and Theoretical Computer Science, 26th International Conference, Kolkata, India, December 13-15, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4337, pp. 273–284. Springer (2006). https://doi.org/10.1007/11944836_26, https://doi.org/10.1007/11944836_26
6. Doumane, A.: On the infinitary proof theory of logics with fixed points. (Théorie de la démonstration infinitaire pour les logiques à points fixes). Ph.D. thesis, Paris Diderot University, France (2017), https://tel.archives-ouvertes.fr/tel-01676953
7. Hyvernat, P.: The size-change termination principle for constructor based languages. Logical Methods in Computer Science **10**(1) (2014). https://doi.org/10.2168/LMCS-10(1:11)2014, https://doi.org/10.2168/LMCS-10(1:11)2014
8. Hyvernat, P.: The size-change principle for mixed inductive and coinductive types. CoRR **abs/1901.07820** (2019), http://arxiv.org/abs/1901.07820
9. Jones, N.D.: Computability and complexity - from a programming perspective. Foundations of computing series, MIT Press (1997)
10. Kozen, D.: On induction vs. *-continuity. In: Kozen, D. (ed.) Logics of Programs, Workshop, Yorktown Heights, New York, USA, May 1981. Lecture Notes in Computer Science, vol. 131, pp. 167–176. Springer (1981). https://doi.org/10.1007/BFb0025782, https://doi.org/10.1007/BFb0025782
11. Kozen, D.: Results on the propositional mu-calculus. Theor. Comput. Sci. **27**, 333–354 (1983). https://doi.org/10.1016/0304-3975(82)90125-6, https://doi.org/10.1016/0304-3975(82)90125-6
12. Lee, C.S., Jones, N.D., Ben-Amram, A.M.: The size-change principle for program termination. In: Hankin, C., Schmidt, D. (eds.) Conference Record of POPL 2001: The 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, London, UK, January 17-19, 2001. pp. 81–92. ACM (2001). https://doi.org/10.1145/360204.360210, http://doi.acm.org/10.1145/360204.360210
13. Lepigre, R., Raffalli, C.: Practical subtyping for curry-style languages. ACM Trans. Program. Lang. Syst. **41**(1), 5:1–5:58 (2019). https://doi.org/10.1145/3285955, https://doi.org/10.1145/3285955
14. Nollet, R., Saurin, A., Tasson, C.: Local validity for circular proofs in linear logic with fixed points. In: Ghica, D.R., Jung, A. (eds.) 27th EACSL Annual Conference on Computer Science Logic, CSL 2018, September 4-7, 2018,

Birmingham, UK. LIPIcs, vol. 119, pp. 35:1–35:23. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2018). https://doi.org/10.4230/LIPIcs.CSL.2018.35, https://doi.org/10.4230/LIPIcs.CSL.2018.35

15. Rowe, R.N.S., Brotherston, J.: Automatic cyclic termination proofs for recursive procedures in separation logic. In: Bertot, Y., Vafeiadis, V. (eds.) Proceedings of the 6th ACM SIGPLAN Conference on Certified Programs and Proofs, CPP 2017, Paris, France, January 16-17, 2017. pp. 53–65. ACM (2017). https://doi.org/10.1145/3018610.3018623, https://doi.org/10.1145/3018610.3018623
16. Tellez, G., Brotherston, J.: Automatically verifying temporal properties of pointer programs with cyclic proof. In: de Moura, L. (ed.) Automated Deduction - CADE 26 - 26th International Conference on Automated Deduction, Gothenburg, Sweden, August 6-11, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10395, pp. 491–508. Springer (2017). https://doi.org/10.1007/978-3-319-63046-5_30, https://doi.org/10.1007/978-3-319-63046-5_30