Canonical proof-objects for coinductive programming: infinets with infinitely many cuts

Abhishek De, Luc Pellissier, Alexis Saurin

Canonical proof-objects for coinductive programming: infinets with infinitely many cuts

Abhishek De*
Université de Paris, IRIF, CNRS
Paris, France
abhishek.de@irif.fr

Luc Pellissier
LACL, Université Paris Est Créteil
Créteil, France
luc.pellissier@lacl.fr

Alexis Saurin
Université de Paris, IRIF, CNRS
Paris, France
alexis.saurin@irif.fr

ABSTRACT

Non-wellfounded and circular proofs have been recognised over the past decade as a valuable tool to study logics expressing (co)inductive properties, *e.g.* μ-calculi. Such proofs are non-wellfounded sequent derivations together with a global validity condition expressed in terms of *progressing threads*. While the cut-free fragment of circular proofs is satisfactory, cuts are poorly treated and the non-canonicity of sequent proofs becomes a major issue in the non-wellfounded setting. The present paper develops for μMLL (multiplicative linear logic with fixed points) the theory of infinets – proof-nets for non-wellfounded proofs. Our structures handles infinitely many cuts therefore solving a crucial shortcoming of the previous work [19]. We characterise correctness, define a more complete cut-reduction system and proving a cut-elimination theorem. To that end, we also provide an alternate cut reduction for non-wellfounded sequent calculus.

KEYWORDS

circular proofs, non-wellfounded proofs, fixed points, muMALL, linear logic, proof-nets, induction and coinduction

ACM Reference Format:
Abhishek De, Luc Pellissier, and Alexis Saurin. 2021. Canonical proof-objects for coinductive programming: infinets with infinitely many cuts. In *Proceedings of the 23rd Symposium on Principles and Practice of Declarative Programming, PPDP 2021, Tallinn, Estonia, September 6–8, 2021.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/NNNNNNN.NNNNNNN

1 INTRODUCTION

Coinductive programming. Computation over finitary data can be handled in declarative programming by inductive types: *termination* of computation is then guaranteed by the fact that the program is typable. Coinductive programming is a generalization of programming that provides a natural way to reason about coinductive data-types, lazy predicates, concurrent communicating predicates, *etc.* In such programs, termination is replaced by *productivity*: while the computation is not guaranteed to terminate, arbitrarily large prefixes of the result can nonetheless be computed in a finite number of steps. Proof assistants such as Agda and Coq which supports coinductive programs traditionally employ a strict type checker that checks for a guard condition. Guardedness is a sufficient but not necessary condition to ensure productivity: productivity is undecidable, hence designing tractable guard conditions that can accept more and more productive programs is an important area of research. In Figure 1, we consider several Coq coinductive definitions, whose behaviour varies wildly:

```
CoInductive Stream := Cons : nat→ Stream→ Stream.
CoFixpoint f0 (n : nat) : Stream := Cons n (f0 (n+1)).
CoFixpoint f1 (n : nat) : Stream := let s := f1 (n+1) in
    Cons n (match s with Cons h t ⇒ Cons h t end).
CoFixpoint f2 (n : nat) : Stream := let s := f2 (n+1) in
    (match s with Cons h t ⇒ Cons n (Cons h t) end).
CoFixpoint f3 (n : nat) : Stream := let s := f3 (n+1) in
    (match s with Cons h t ⇒ Cons h (Cons n t) end).
```

Figure 1: Some productive and non-productive definitions

- `f0` is the only valid Coq coinductive definition; `(f0 n)` computes the stream of natural numbers starting from `n`.
- `f1` is a productive term, even though it is rejected by Coq typechecker as it fails to pass its guard condition. It computes the same stream as `f0`.
- `f2` is not productive, but one can introduce a commutation rule: $\texttt{match}\ e_1\ \texttt{with}\ p \Rightarrow \texttt{Cons}\ (h, t) \rightsquigarrow \texttt{Cons}\ (h, \texttt{match}\ e_1\ \texttt{with}\ p \Rightarrow t)$ (if pattern p does not occur free in h and symmetrically with t) to make it so; it is then equivalent to `f1`.
- `f3` is not productive: producing the first element of `(f3 n)` requires to already have produced the first element of each stream `(f3 k)` for `k` > `n`.

As we can see, even minor differences in the code, which seemingly makes no logical difference, can cause a type checker to behave very differently. Our goal (G) is two-fold:

- extend the guard condition so that more programs are accepted;
- provide a more canonical representation of programs so that productivity is more robust.

Proof theory of fixed point logics can tell us about the computational behaviour of these programs: following the guiding principles of the Curry-Howard correspondence, co·inductive types can be encoded

*This author has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 754362.

PPDP '21, September 6–8, 2021, Tallinn, Estonia

ACM ISBN 978-1-4503-8689-0…$15.00
https://doi.org/10.1145/NNNNNNN.NNNNNNN

as formulas of the μ-calculus, and programs as proofs in such a logic.

To the notable exception of Santocanale, circular proofs were originally introduced [9, 11, 35–37] with provability and verification purposes in mind, yielding systems such as μLK. These systems allow for non-wellfounded branches in sequent derivations. Allowing unrestricted use of those non-wellfounded branches leads to logical inconsistency, as any sequent can be derived (see fig. 8a): a global validity criterion is needed to sieve the logically valid proofs from the unsound ones, and it corresponds to the aforementioned guard condition for coinductive sequent proofs. Such sequent proof systems are not well-suited to study the equality of proofs, hence we need to move to a more structural system. A natural candidate would be intuitionistic natural deduction with fixed points, another one is linear logic proof nets. We will use linear logic, as its connectives are rich enough to allow to encode many types purely logically.

Productivity, from a proofs/programs perspective. In μMALL (linear logic extended with least and greatest fixed points), one can represent the type of nats and streams as $\mathsf{N} = \mu X.1 \oplus X$ and $\mathsf{S} = \nu Y.\mathsf{N} \otimes Y$.[1] We can represent natural numbers as well as the successor function as cut-free proofs: respectively $\pi_k, k \in \mathbb{N}$ of type $\vdash \mathsf{N}$ and π_{succ} of type $\mathsf{N} \vdash \mathsf{N}$ presented in fig. 2. Also, naturals are duplicable resources *i.e.* there is a proof π_{dup} of $\mathsf{N} \vdash \mathsf{N} \otimes \mathsf{N}$. Recall that cuts reduction corresponds to computation so π_{dup} is cut with π_k reduces to the pair $\pi_k \otimes \pi_k$.

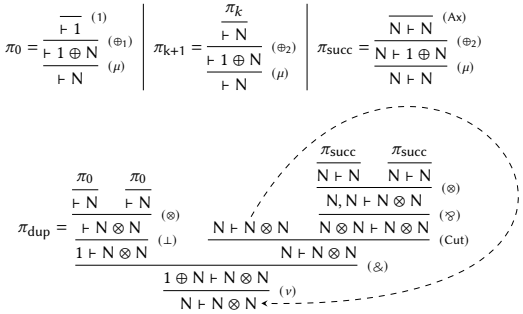


Figure 2: μMALL$^\infty$ encodings of nat and their basic functions

Similarly, one can represent streams of nats as cut-free *non-wellfounded* derivations of $\vdash \mathsf{S}$ (*e.g.* in fig. 3 π^n_{from} represents the streams of successive nats starting from n) and functions on streams (*e.g.* π_{Cons} represents the cons on streams): in those derivations, we may have infinitely deep branches as for π^n_{from} whose right-most branch is infinite.

One can encode the above coinductive programs as ***circular*** derivations[2] of $\mathsf{N} \vdash \mathsf{S}$, as shown in fig. 4: Φ_0, Φ_1, Φ_2 and Φ_3 represent respectively `f0`, `f1`, `f2` and `f3`. To compute the value of `f0(n)` one would need to consider the proof obtained by cutting Φ_0 with π_n

[1]While we use a one-sided presentation of the sequent calculus in the technical developments of this paper, as common with classical LL, we show the encoding of the above programs using two-sided sequents for clarity: $F_1, \dots, F_n \vdash G_1, \dots, G_m$ should be read as $\vdash F_1^\perp, \dots, F_n^\perp, G_1, \dots, G_m$ as usual and left inference rules are written via the right rule of their dual connective.

[2]Notice that derivations for the Φ_is contain back-edges, denoting the fact that the derivation tree is infinite but regular.

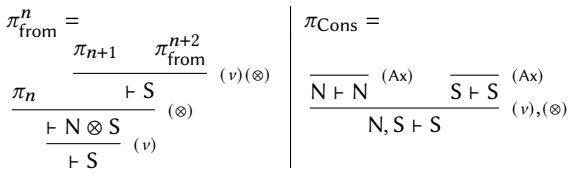


Figure 3: μMALL$^\infty$ encodings of basic functions on streams

which will induce an infinite cut-reduction sequence ***converging*** to π^n_{from}; the same happens when reducing a cut between Φ_1 and π_n: those are productive. On the other hand, if π' is obtained by cutting Φ_2 with π_n every derivation that is reached by reduction sequence from π' will have a cut as its last inference: cut cannot be eliminated from that proof, it is a non-productive computation.[3] Interestingly, the difference between Φ_1 and Φ_2 is limited to the relative order of the $(\nu)(\otimes)$ inferences and the $(\mu)(\parr)$ depicted in green and red on fig. 4 and that difference, altogether with non-wellfoundedness of the sequent derivations, will make the difference between a productive and a non-productive cut-elimination. A simpler example of the same phenomenon will be given in fig. 8b and 8c and discussed below. Therefore the desideratum is a proof paradigm such that the representations of `f1` and `f2` are the same (say $Rep(\mathtt{f1}) = Rep(\mathtt{f2}) = K$) and cut elimination productive in K cut against $Rep(\mathtt{n})$ for all n.

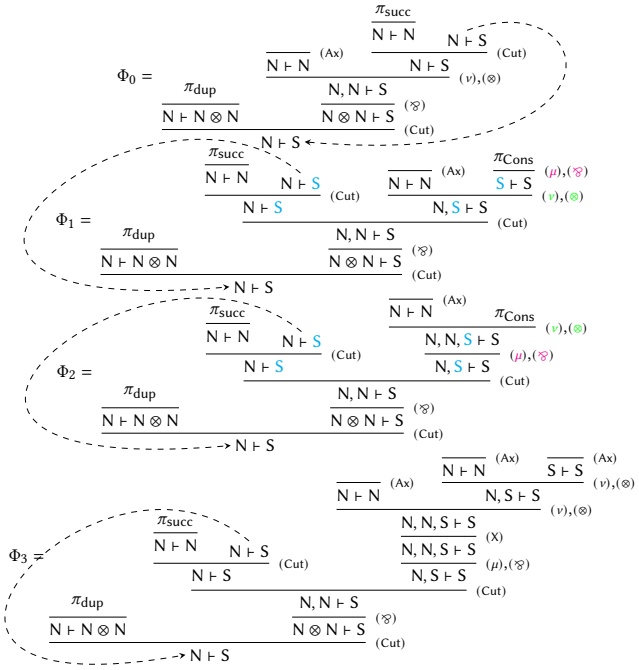


Figure 4: μMALL$^\infty$ encodings of `f0`, `f1`, `f2` and `f3` from fig. 1

[3]Indeed, to produce the first element of the stream the green $(\nu)(\otimes)$ inferences, one first needs to make the red $(\mu)(\parr)$ inferences interact over the cut but this requires first partially eliminating the cut of the derivation at the source of the back-edge, which is essentially the proof we started with: this infinite chain of dependencies causes the non-productivity.

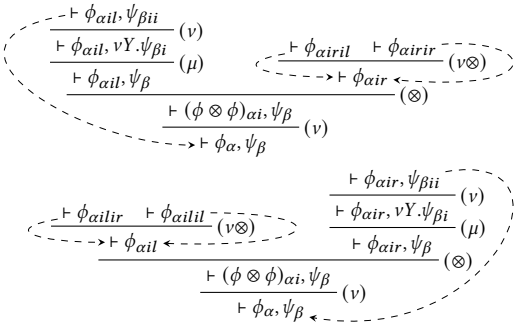


Figure 5: Two proofs of $\vdash \nu X.X \otimes X, \mu X.\nu Y.X$

Proof-nets. As we have seen, some seemingly irrelevant differences (the relative order of the application of rules) induce widely varying behaviour in μMALL sequent calculus. This phenomenon is related to the fact that the sequent calculus for LL is non-canonical: a LL proof may be reduced to two cut-free proofs π_1 and π_2 which are different but guaranteed to be equal up to irrelevant permutations of inference rules[4]. In other words, the permutations are denotationally trivial *i.e.* $[\![\pi_1]\!] = [\![\pi_2]\!]$ in any semantics. Proof-nets [28] were devised to overcome this sequentiality. A proof-net can be seen as a graph whose nodes are inference rules, which are thus not ordered, and consequently less sequential than sequent calculus proofs. As they are canonical, proof-nets are well-suited to represent computation.

Infinets. In [19], the authors defined *infinets*, canonical objects that capture exactly the equivalence classes of pre-proofs under the equivalence by (possibly infinite) permutation of inferences (*a.k.a.* permutative equivalence). Compared to MLL, more structure is needed in order to have suitable non-wellfounded proof structures for μMLL$^\infty$.

Consider the two μMLL$^\infty$ proofs in fig. 5, omitting the indices for the time being.They are not permutatively equivalent: no permutation will change the contents of the premises of a tensor. However, if we try by simply forgetting the order of inferences and keeping only the subformula ordering and the back-edges, we end up in the same structure (cf. fig. 6); which means this proof-net equivalence would be coarser than permutation equivalence. Indeed, the fact that ψ resides with one of the many possible infinite branch is lost in translation: in order to be faithful, more structure in the form of "infinite axioms" is present in infinets. Just as usual axioms encapsulate the information which formulas end up in which leaf of the proof tree, infinite axioms encapsulate the information which formulas end up in which infinite branch of the non-wellfounded proof tree.

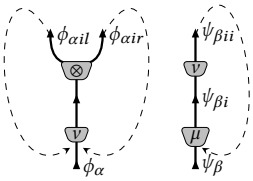


Figure 6: First attempt at infinets

[4]Normalisation for LL sits thus in the middle between classical sequent calculus LK — in which a proof (Lafont's critical pair) can be reduced to any two proofs of the same sequent — and natural deduction [27, 33] or λ-calculus [13] normalisation which are confluent.

The non-canonicity of sequent calculus manifests itself more critically in the non-wellfounded setting: productivity of cut-elimination is not preserved by permutative equivalence [5], as already noticed with fig. 4. The two pre-proofs in figs. 8b and 8c witness the same phenomenon with simpler proof objects (they use neither additive nor multiplicative connectives, only fixed points): they are permutatively equivalent but cut-elimination is productive only in the latter (fig. 11). However they have the same infinet on which the cut-reduction rules that we propose can be applied (fig. 20). Consequently, we believe that infinets are the proper framework for dealing with unrestricted cuts and more expressive validity conditions (such as bouncing-validity). Understanding the impact of those permutations and how to quotient them properly is a deep motivation for this work and for our investigation of proof-nets for non-wellfounded proofs: we aim at benefiting from the canonicity of proof-nets to improve the dynamics of non-wellfounded derivation *wrt.* cut-elimination.

Infinitely many cuts. The handling of the cuts in [19] has been rudimentary: only finitely many cuts are considered and cut-elimination is basically interpreted as an infinitary abstract rewriting system with a metric: at first, one guesses the normal form (*a.k.a.* big-step) then, a transfinite reduction sequence of small steps is shown to converge to the big-step in the limit. To guess the limit, one has to sacrifice some structure *viz.* η-expand all axioms rendering the calculus without atoms. This is a strong limitation when one see that examples as simple as Φ_0, Φ_1 and Φ_2 contain infinitely many cuts. We construe $\textcircled{\mathcal{G}}$ in terms of proof-nets:

(1) Devise proof-nets for μMLL$^\infty$.
(2) Devise cut-elimination rules for them.
(3) Provide cut-elimination result which would correspond to standard guard condition.
(4) Extend the validity condition preserving productivity of cut-elimination.
(5) Restrict the canonical proof object in such a way that they are sufficiently expressive and yet the extended validity is decidable.

The present paper provides a full treatment of cuts and axioms for non-wellfounded proof-nets thereby achieving the first three of the above stated goals.

Organisation of the contributions. This work strengthens the definition for non-wellfounded proof structures—and their correctness criterion—to accommodate infinitely many cuts: in this situation, some infinite axioms are only virtually present, and made explicit through cut-elimination. Our main contribution is the cut-elimination result for infinets with atoms and infinitely many cuts by reconciling the locality of the big-step and non-locality of the small-step. To prove that result we need to provide an alternate cut reduction system of μMLL$^\infty$ sequent calculus. The contributions are summarised in fig. 7.

In section 2, we recall the necessary background on non-wellfounded proof theory and proof-nets. In section 3, we provide a new cut-elimination result for μMLL$^\infty$ sequent calculus which is an alternative to Baelde et al [5, 6] cut-elimination and is better suited for

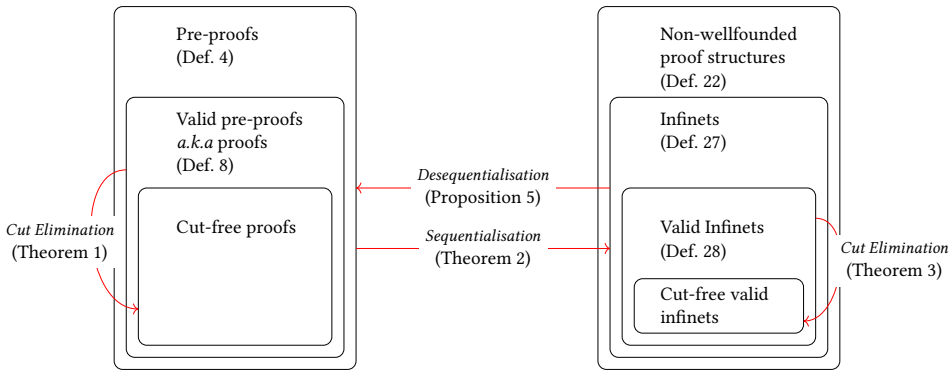


Figure 7: Schemata of the contributions

proof-nets. In section 4, we informally discuss the necessary structure (visitable paths, infinite real axioms, infinite virtual axioms) that have to be added to MLL proof-nets in order to faithfully represent $\mu\mathsf{MLL}^\infty$ pre-proofs. In section 5, we formalize this intuition and define non-wellfounded proof structures. We state the correctness criterion in section 6, which leads us to develop and adapt the theory of kingdoms to non-wellfounded structures, an additional contribution of the work. This allows us to provide a sequentialisation procedure extending that of [19]. We introduce a new cut reduction system in section 7 and establish the cut-elimination theorem by showing productivity of cut-reduction for valid infinets. Section 8 concludes with future directions. A full version with all the proofs is available at [18].

2 BACKGROUND

Before we begin, we will introduce some notations that will be used throughout the paper.

Language theoretic notations. We denote the empty word by ϵ. Let x, y be two words. The greatest common prefix of x and y is denoted by $x \cap y$. For any finite set of alphabets, Σ, Σ^∞ denotes the set of finite and infinite words made of letters from Σ. Given a set of words, L, the prefix-closure of L is denoted by $\overline{L}$.

Graph theoretic terms. A ***multigraph*** is graph is a graph which can have multiple edges between two vertices. A ***hypergraph*** is a generalization of a graph in which an edge can join any number of vertices *i.e.* an edge is a non-empty subset of of the set of vertices. A ***hybridgraph*** is a hypergraph where the normal edges *i.e.* edges joining exactly two vertices are distinguished from the rest of the edges. We will now recall some terms from the infinite graph theory.

An infinite graph (V, E) with $V = \{x_0, x_1, x_2, \dots\}$ and $E = \{x_0x_1, x_1x_2, \dots\}$ is called a ***ray***. The subrays of a ray are called its ***tails***. An ***end*** of an infinite graph $G = (V, E)$ is an equivalence class of the rays in G, where two rays are considered equivalent if, for every finite set $S \subseteq V$, both have a tail in the same component of $G - S$.

2.1 Multiplicative linear logic with fixed points

$\mu\mathsf{MALL}^\infty$, the non-wellfounded extension of MALL, the multiplicative additive fragment of linear logic, with least and greatest fixed points operators, was introduced in [6, 23]. In this paper, we only consider the unit-free multiplicative fragment which we call $\mu\mathsf{MLL}^\infty$. In this section, we recall some basic definitions.

Definition 1. *Given two disjoint infinite sets of atoms $\mathcal{A} = \{A, B, \dots\}$, and of propositional variables $\mathcal{V} = \{X, Y, \dots\}$, $\mu\mathsf{MLL}^\infty$* **pre-formulas** *are given by the following grammar (where $A \in \mathcal{A}$, $X \in \mathcal{V}$):*

$$\phi, \psi ::= A \mid A^\perp \mid X \mid \phi \parr \psi \mid \phi \otimes \psi \mid \mu X.\phi \mid \nu X.\phi$$

μ, ν bind the variable X in ϕ. Free and bound variables, as well as capture-avoiding substitution are defined as usual. The subformula ordering is denoted $\leq$. A closed pre-formula (i.e. no free variables), is called a **formula**.

We define negation, $(\bullet)^\perp$, as a meta-operation on pre-formulas (with $X^\perp = X$) and will use it only on formulas. As it is not part of the syntax, we do not need any positivity condition on the fixed-point expressions. As expected, the least and greatest fixed point are the dual of each other.

The system is classical, hence, it is enough to consider a one-sided proof system. However, in order to keep track of *progressing* (*a.k.a. valid*) *threads* and also while translating into proof nets, it is useful to distinguish occurrences of the same formula within a sequent. A $\mu\mathsf{MLL}^\infty$ ***sequent*** is an expression $\vdash \Delta$ where Δ is a finite set of pairwise *disjoint formula occurrences*. We will now define these terms introduced.

Definition 2. *An* **(in)finite address** *is an (in)finite word in $\{l, r, i\}^\infty$. Negation extends over addresses as the morphism satisfying $l^\perp = r$, $r^\perp = l$, and $i^\perp = i$. If β is a prefix of α then α is* **sub-address** *of β. Finally, α and β are said to be* **disjoint** *if $\alpha \cap \beta$ is not equal to α or β.*

Definition 3. *A* **formula occurrence** *(denoted by $F, G, \dots$) is given by a formula ϕ and a finite address α, written ϕ_α. Let $\mathsf{addr}(\phi_\alpha) = \alpha$. Operations on formulas extend to occurrences: ${\phi_\alpha}^\perp = \phi^\perp_\alpha$; for $\star \in \{\parr, \otimes\}$, $F \star G = (\phi \star \psi)_\alpha$ if $F = \phi_{\alpha l}$ and $G = \psi_{\alpha r}$; for $\sigma \in \{\mu, \nu\}$, $\sigma X.F = (\sigma X.\phi)_\alpha$ if $F = \phi_{\alpha i}$. Substitution of occurrences forgets addresses i.e. $(\phi_\alpha)[\psi_\beta/X] = (\phi[\psi/X])_\alpha$. We say that occurrences are* **disjoint** *when their addresses are. Given two occurrences ϕ_α, ψ_β, ϕ_α is called a* **suboccurrence** *of ψ_β (written $G \sqsubseteq F$) if ϕ is FL-subformula of ψ and α is a subaddress of β. Finally, $\lceil \bullet \rceil$ denotes the address erasure operation on occurrences.*

We are now ready to define the derivation trees of $\mu\mathsf{MLL}^\infty$.

Definition 4. *A* **pre-proof** *of $\mu\mathsf{MLL}^\infty$ is a possibly infinite tree generated from the inferences of unit-free multiplicative linear logic and the following rules:*

$$\frac{\vdash G[\mu X.G/X], \Delta}{\vdash \mu X.G, \Delta}\,(\mu) \qquad \frac{\vdash G[\nu X.G/X], \Delta}{\vdash \nu X.G, \Delta}\,(\nu)$$

Given a pre-proof, π, $\mathsf{addr}(\pi) \subseteq \{l, r, i\}^\infty$ is largest set of addresses s.t. if a finite address $\alpha \in \mathsf{addr}(\pi)$ then for some ϕ, ϕ_α either occurs in an axiom or occurs infinitely often in an infinite branch in π with $\mathsf{addr}(F) = \alpha$; and if an infinite address $\alpha \in \mathsf{addr}(\pi)$ then there is an infinite branch β of π such that every finite prefix of α is an address of an occurrence appearing in β.

Example 1. *The three derivations in fig. 8 are pre-proofs, as are the two in fig. 5. We draw back-edges between two sequents with the same underlying formulas as a way to represent regular proofs: the derivation above the pointing sequent is equal to the one above the pointed one, up to address renaming. If we call π the first proof in fig. 5, we have that $\mathsf{addr}(\pi) = \alpha.(i(l + r))^\omega \cup \beta i^\omega$.*

Infinitary proof systems depart peculiarly from their wellfounded counterparts: in spite of the rules being locally sound, it is possible to derive any sequent, as in Figure 8a. We impose a global validity criterion on pre-proofs. Valid pre-proofs are simply called ***proofs***.

Definition 5. *Let $\gamma = (s_i)_{i\in\omega}$ be an infinite branch of a pre-proof. A* **thread** *of γ is a sequence $\tau = \{F_i\}_{i\in\omega}$ such that there exists $j \geq 0$ such that for all $i < \omega$, we have $F_i \in s_{i+j}$ and either F_i is suboccurrence of F_{i+1} or $F_i = F_{i+1}$. We denote the sequence of formulas $\{\lceil F_i \rceil\}_{i\in I}$ by $\lceil \tau \rceil$.*

Definition 6. *A thread is said to be* **straight** *if it is not ultimately constant. A straight thread is said to be* **valid** *if the set $\mathsf{Inf}(\tau)$ of formulas occurring infinitely often in τ, admits a minimum F_{min} wrt the subformula ordering and F_{min} is a ν-formula.*

Definition 7. *An infinite branch of a pre-proof is called* **real** *if it has at least one straight thread and* **virtual** *otherwise. It is* **valid** *if it has a valid thread.*

Definition 8. *Let π be a $\mu\mathsf{MLL}^\infty$ pre-proof. It is* **straight-thread valid** *if all its infinite branches are valid.*

Remark 1. *Observe that a proof does not have virtual branches. Furthermore, if a pre-proof has virtual branches, then it has infinitely many cuts.*

Example 2. *Consider the first pre-proof (say π) in fig. 5. The left-most branch contains two threads: one following the formula ϕ, the other the formula ψ. The first one is straight and valid, the second is not: accordingly, this branch is supported by a valid thread. As it is also the case of the other infinite branches, π is a proof.*

2.2 Proof-nets

Proof-nets are geometrical proof objects introduced by Girard that eliminates two forms of bureaucracy which differentiates sequent proofs: irrelevant syntactical features and the order of inference rules.

Proof-nets are usually defined as vertex labelled, edge labelled directed multigraphs. In this section we present MLL proof-nets in a formalism due to Curien [14] which is useful to lift proof-nets to $\mu\mathsf{MLL}^\infty$.

We begin by recalling that the ***syntax tree*** of an MLL formula occurrence F induces a prefix closed language, $\mathcal{L}_F \subset \{l, r, i\}^*$ *s.t.* there is a natural bijection between $\mathcal{L}_F$ and the branches of the tree.

Definition 9. *A* **partial syntax tree**, *F^U, is a subtree of the syntax tree of the formula occurrence, F, s.t. $U \subseteq \mathcal{L}_F$ and U represents a bar of the syntax tree of F i.e. any $u, u' \in U$ are pairwise disjoint and for every $uav \in U$, there is a v' s.t. $ua^\perp v' \in U$. For $u \in \overline{U}$, we denote by (F, u) the unique suboccurrence of F with the address $\mathsf{addr}(F).u$.*

We illustrate a schematic partial syntax tree in fig. 9a. MLL proof nets without cuts can be seen as a forest of partial syntax trees of the occurrences in the conclusion sequent and axiom links between their leaves (cf. fig. 9b). To incorporate cuts we need to add the partial syntax tree of the cut occurrences (along with the axioms links involving their leaves) and links between dual cut occurrences.

Definition 10. *An* MLL **proof-structure** *is a 3-tuple $(\{F_i^{U_i}\}_{i\in I}, \mathfrak{K}, \Theta)$ where:*

- *for all $i \in I$, $F_i^{U_i}$ is a partial syntax tree; $\{F_i\}_{i\in I}$ is called the set of* **doors**.
- *$\mathfrak{K}$ is the set of cuts i.e. a (possibly empty) set of disjoint subsets of $\{F_i\}_{i\in I}$ of the form $\{C, C^\perp\}$; and,*
- *Θ is the set of axiom links i.e. a partition of the set of leaves, $\mathcal{L} = \bigcup_{i\in I}\{\alpha_i u_i \mid \mathsf{addr}(F_i) = \alpha_i, u_i \in U_i\}$ such that each cell is pair of dual addresses i.e. of the form $\{\alpha_i u_i, \alpha_j u_j\}$ such that $\lceil (F_i, u_i) \rceil = \lceil (F_j, u_j) \rceil^\perp$.*

Not all proof-structures are meaningful and we need to impose a correctness criterion. For the rest of the section fix a proof-structure $\mathcal{R} = (\{F_i^{U_i}\}_{i\in I}, \mathfrak{K}, \Theta)$.

Definition 11. *A* **switching**, *sw, of $\mathcal{R}$ is set of functions $\{sw_i : P_i \to \{l, r\}\}_{i\in I}$ s.t. for every $i \in I$, $P_i \subseteq \overline{U_i}$ and $\lceil (F_i, p) \rceil$ is a $\parr$-formula for all $p \in P_i$.*

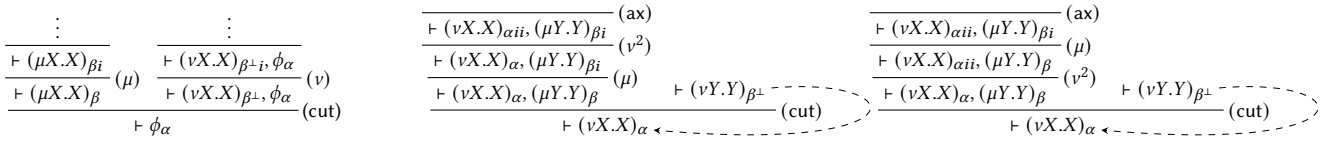
Definition 12. *Let sw be a switching of $\mathcal{R}$. Let u be a substring of a word w in $\overline{U_i}$. Then, u is said to be* **unbroken** *if for all $j \in \{1, \ldots, n-1\}$, $sw_i(vu_1 \ldots u_j) \neq u_{j+1}$ where $u = u_1 \ldots u_n$ and vu is a prefix of w for some word v.*

Fix a switching, sw, of $\mathcal{R}$. Let $\mathrm{SW} \subseteq \mathcal{L}^2$ such that $(x, y) \in \mathrm{SW}$ iff either $x = y$ or one of the following holds:

- $w = x \cap y \neq \epsilon$. Let $wu = x$ and $wv = y$. Then, u and v are unbroken;
- $x = \alpha u$ and $y = \alpha' v$ such that $\mathsf{addr}(C) = \alpha, \mathsf{addr}(C^\perp) = \alpha'$, $\{C, C^\perp\} \in \mathfrak{K}$ and u, v are unbroken.

Observe that SW is an equivalence. If we see the elements of $\mathcal{L}$ as the collection of leaves of the partial syntax trees of a proof net, cells of SW are the connected components of that proof net under the switching sw and without axiom links.

Definition 13. *The* **orthogonal graph** *of $\mathcal{R}$ for the switching, sw, (denoted $G^{sw}(\mathcal{R})$) is the undirected bipartite (multi)graph, $(\Theta, [\mathrm{SW}], E)$, where Θ is the set of axioms of $\mathcal{R}$, $[\mathrm{SW}]$ is the set of equivalence classes of* SW *and $(x, y) \in E$ iff $x \cap y \neq \emptyset$.*



(a) An unsound pre-proof, ϕ is arbitrary. **(b) Non-productive cut-elimination** **(c) Productive cut-elimination**

Figure 8: Non-wellfounded derivations. α and β are arbitrary addresses.

(a) A schematic partial syntax tree **(b) A schematic proof-net**

Figure 9: Illustration of partial syntax trees and proof-nets

Definition 14. *A proof-structure $\mathcal{R}$ is said to be* **DR-correct** *if for switchings sw of $\mathcal{R}$, $G^{sw}(\mathcal{R})$ is connected and acyclic. A DR-correct proof-structure is called a* **proof-net**.

The process of translating a proof-net into a proof is called *sequentialisation*. The process of translating a proof into a proof-net is called *desequentialisation*. Note that the former is a non-deterministic procedure.

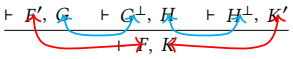
Finally we recall the notion of kingdoms from [8]. Viewed as graphs, a sub-net is a subgraph of a proof-net that is also a proof-net. The kingdom of a formula occurrence, F, in a proof-net is the upward-closed sub-net starting from F.

Definition 15. *Let $\mathcal{R}$ be a proof-net. Given F_i and $u \in \overline{U_i}$, the* **kingdom**, *$k(F_i, u)$, of (F_i, u) is the smallest sub-net of $\mathcal{R}$ with (F_i, u) as one of its doors. We define a relation on the set of occurrences $\{(F_i, u) \mid u \in \overline{U_i}, i \in I\}$ by $X \ll Y$ iff $X \in k(Y)$.*

Proposition 1 ([7]). *The relation $\ll$ is a partial order for any proof-net.*

3 CUT-ELIMINATION IN μMLL$^\infty$ SEQUENT CALCULUS

In finitary proof theory, cut elimination may proceed by reducing topmost cuts but there is no such thing, in general, as a topmost cut in non-wellfounded proof-theory. Previous cut-elimination results relied on reduction of bottom-most cuts [5, 6, 23, 25] using a generalized cut-rule, the multicut, which abstracts over a finite subtree made of cut (and axiom) rules. In those approaches, when two cuts are immediately above one another, they are merged instead of being permuted. The following is an example of a multicut rule: the red lines indicate the context and the blue lines indicate two cuts that have been merged.



A less sequential approach to cut-elimination. While the multicut brings uniformity in the treatment of cut-elimination in sequent calculus, it is not well-suited for our purpose of developing a canonical

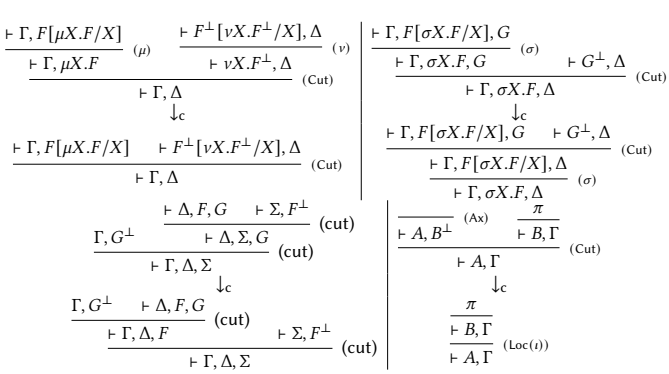


Figure 10: Main cases for μMLL$^\infty$ cut reduction, $\longrightarrow_c$. (With $\sigma \in \{\mu, \nu\}$ and ι st. $\iota(A) = B$, $\iota(H) = H$ for $H \in \Gamma$.)

and parallel treatment of cuts in non-wellfounded proof systems. It is indeed better-suited to use the usual cut-rule to draw a comparison between cut-reductions in sequent systems and in proof-nets, as we will do in the last sections of the paper. To serve this purpose, we develop here an alternative approach to cut-elimination for non-wellfounded proof which avoids the use of the multicut but on the standard cut instead and we will prove a new cut-elimination result in this case. We shall simply retain however a degenerated case of the multi-cut, the *unary* case, used to perform lazily the cut-axiom reduction and relocation of addresses. Indeed, as we work with explicit occurrences, the cut/ax case is as follows:

$$\dfrac{\dfrac{}{\vdash F, G^\perp}\,(\text{ax}) \qquad \dfrac{\pi}{\vdash G, \Gamma}}{\vdash F, \Gamma}\,(\text{cut})$$

with $\lceil F \rceil = \lceil G \rceil$, which cannot simply be reduced to $\dfrac{\pi}{\vdash F, \Gamma}$ as the occurrences do not match (in fact, the addresses of F and G are disjoint). Instead of substituting occurrences in π (which is a non-wellfounded object), we treat this substitution lazily, in the form of an explicit substitution [1] adding the following unary inference rule: $\dfrac{s'}{s}\,(\text{Loc}(\iota))$ where ι is a one-to-one map from s to s' such that for all $F \in s$, $\iota(F) \equiv F$. In the rest of the paper, when writing μMLL$^\infty$, we mean μMLL$^\infty$ extended with $(\text{Loc}(\iota))$.

Definition 16 (μMLL$^\infty$ cut reduction). *The reduction system $\longrightarrow_c$ is obtained by extending the usual* MLL *reduction system with fixed points reductions, cut-commutation rules and a new cut-axiom rules introducing* Loc(ι) *depicted in fig. 10 on p.6*

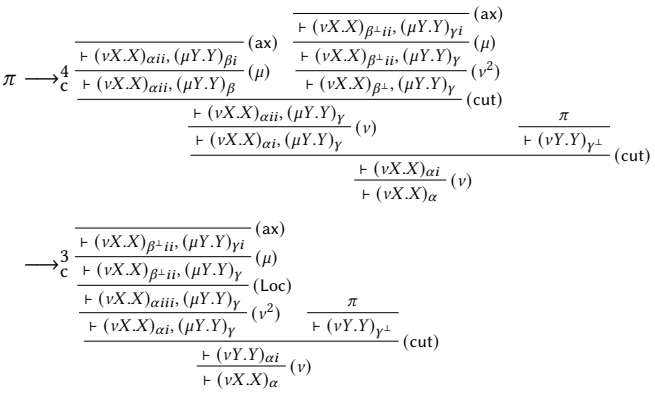


Figure 11: A productive sequence of cut-elimination

Example 3. *Consider the sequent-calculus pre-proof π in fig. 8c. We draw a finite reduction sequence of it in fig. 11. We see that after finitely many steps, it reduces to a proof that is a reduction of π above a ν rule. Note that this is impossible to do with the pre-proof in fig. 8b, as any commutation would necessitate to place a μ below the cut.*

The previous cut-reduction system admits infinitary cut-elimination:

Theorem 1. *If π_0 is a $\mu\mathsf{MLL}^\infty$ proof, there is sequence of $\mu\mathsf{MLL}^\infty$ proofs $(\pi_i)_{i\in\omega}$ with $\pi_i \longrightarrow_{\mathsf{c}} \pi_{i+1}$ (strongly) converging to a cut-free $\mu\mathsf{MLL}^\infty$ proof π'.*

4 TRIPS AND VISITABLE PATHS

In this section, by looking at examples, we study what structure present in sequent-calculus pre-proofs needs to be computed so as to translate them as proof structures.

Infinite axioms. Proof-nets are customarily obtained from a proof by forgetting the order of inference rules. In section 2.2 we exposed a proof-net as a set of formula occurrences, together with an order (the subformula ordering, representing which formula was deduced from which other formulas) and distinguished sets of formulas, representing the conclusions, axioms, and cuts. In particular, an axiom is just a set of two dual formulas. As discussed in section 1, this set of data is not sufficient for desequentializing infinite proofs (cf fig. 5): indeed, in the sequent calculus, an arbitrary formula F can be added to a proof of a sequent $\vdash \nu X.X$; F can be seen as justified by the infinite sequence of unfoldings. As such, we need to introduce *infinite axioms* as in [19] which are sets of finitely many formulas and sequences of unfoldings. Before defining our proof-objects, we thus need to define these sequences of unfolding and how they can participate in infinite axioms. We will thus define two mirror objects: *s-trips* in pre-proofs and *visitable paths* in proof-structures.

As a first approximation, infinite axioms are the infinite branches of pre-proofs, which we can picture graphically as a cell "above" an undirected ray of ν nodes (which corresponds to a straight thread in the corresponding proof) which can also be above an arbitrary number of formulas. We have drawn such a picture for the pre-proofs in figs. 12a, 12d and 12g respectively in figs. 12b, 12e and 12h. As we will see, this approximation does not allow to faithfully represent all pre-proofs, and we need to consider infinite axioms of more general kind.

Visitable paths. Consider the proof structure in fig. 12b, the naive desequentialisation of the pre-proof in fig. 12a. There is an infinite axiom "above" the undirected ray of ν nodes (which corresponds to a straight thread) and there is an undirected ray, ρ, of alternating axioms and cuts. Observe that every A introduced by a cut resides with the straight thread in the only infinite branch in fig. 12a: this is lost in translation. Since infinite axioms capture the invariant of an infinite branch, ρ should be included in the infinite axiom in a correct desequentialisation as in fig. 12c. Paths like ρ alternating through axioms and cuts are called *visitable paths*.

Visitable paths can be formed using tensor nodes as well: a similar situation as above can be reproduced using tensors and a fixed point formula. Consider the proof structure in fig. 12e, the naive desequentialisation of the pre-proof in fig. 12d. Here as well the visitable path of alternating axioms and tensors (in red) should be included in the infinite axiom above the undirected ray of ν nodes.

Trips. When translating pre-proofs into proof structures, one needs to compute the visitable paths on pre-proofs. We therefore introduce the sibling notion of s-trips.

Definition 17. *Given a pre-proof π, a* **trip** starting from F_1 *is a sequence $\tau = \{(s_i, F_i, d_i)\}_{i\in\omega}$ where s_i is a sequent in π, $F_i \in s_i$ and $d_i \in \{\uparrow, \downarrow\}$ such that $d_1 = \uparrow$ and for every $i < \omega$ exactly one of the following holds:*

- *$d_i = d_{i+1} = \uparrow$, s_{i+1} is a premise of s_i and $F_{i+1} \sqsubseteq F_i$.*
- *$d_i = d_{i+1} = \downarrow$, s_i is a premise of s_{i+1} and $F_i \sqsubseteq F_{i+1}$.*
- *$d_i = \uparrow, d_{i+1} = \downarrow, s_i = s_{i+1} = \{F_i, F_{i+1}\}$ and s_i is conclusion of a* (ax) *rule.*
- *$d_i = \downarrow, d_{i+1} = \uparrow, s_i$ and s_{i+1} are the premises of a* (cut) *rule on F_i and F_{i+1}.*
- *$d_i = \downarrow, d_{i+1} = \uparrow, s_i$ and s_{i+1} are the premises of a $(\otimes)$ rule on F_i and F_{i+1}.*

Furthermore τ satisfies that for every $i, j < \omega$, there does not exist a sequent, s, in π such that $F_i \parr F_j \in s$.

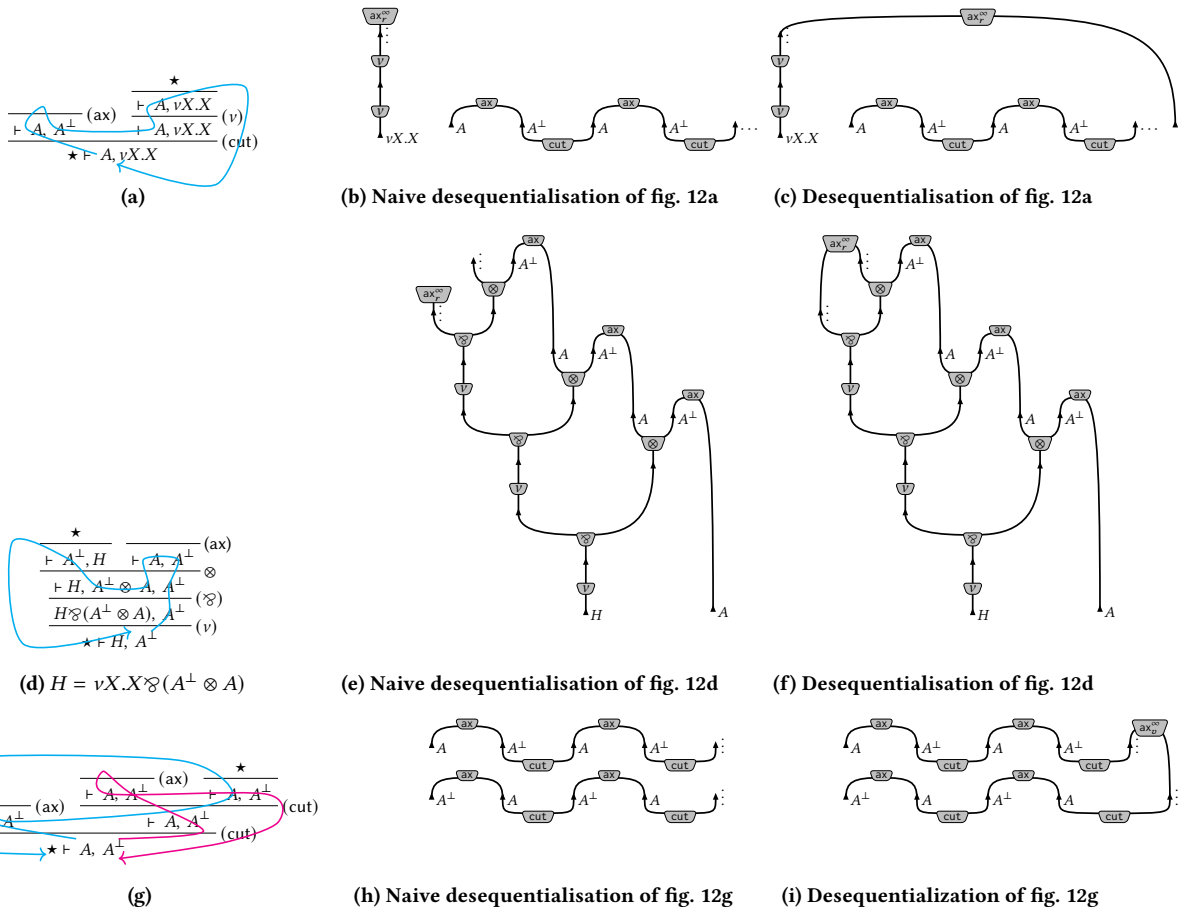
Informally, a trip is sequence of pointed sequents (*i.e.* sequents with a principal formula occurrence) with directions tracing a path bouncing on axioms and cuts or tensors. Furthermore, it does not go through two premises of a $\parr$ rule: which stated in terms of proof-nets corresponds to switching paths. Observe that straight threads are basically trips which have finitely many terms of the form $(s, F, \downarrow)$.

Definition 18. *An* **s-trip** *is a trip which has infinitely many terms of the form $(s, F, \downarrow)$. A pre-proof is said to be* **simple** *if it does not contain any s-trips.*

The coloured lines in the pre-proofs of fig. 12 indicate s-trip.

Proposition 2. *Let $\{(s_i, F_i, d_i)\}_{i\in\omega}$ be a trip of a pre-proof π. Then there exists an infinite branch that has infinitely many common terms with $(s_i)_{i\in\omega}$. Furthermore, if there exist two such infinite branches γ_1 and γ_2 then $\gamma_1 = \gamma_2$ after finitely many terms.*

From proposition 2 we can associate a unique maximal infinite branch with a trip. An s-trip correspond to a visitable path and the



(a) (b) Naive desequentialisation of fig. 12a (c) Desequentialisation of fig. 12a

(d) $H = \nu X.X \parr (A^\perp \otimes A)$ (e) Naive desequentialisation of fig. 12d (f) Desequentialisation of fig. 12d

(g) (h) Naive desequentialisation of fig. 12g (i) Desequentialization of fig. 12g

Figure 12: Naive and faithful desequentialisations of μMLL$^\infty$ simple pre-proofs. Back-edges are depicted using pointers ($\star$). Red and blue curves indicate trips.

infinite branch associated with it corresponds to the infinite axiom above the corresponding visitable path.

Virtual infinite axioms. Consider the pre-proof π in fig. 12g. It proves a sequent $\vdash A, A^\perp$ by never operating on these formulas but delaying infinitely this treatments using cuts. From this perspective, it could be desequentialised naively as in fig. 12h. The pre-proof has two maximal s-trips (following each formula as they are cut and introduced by finite axioms), just as the proof-structure has two visitable paths.

Nonetheless, we can argue as before that an infinite axiom should be atop the two visitable paths, representing that the two formulas A and $A^\perp$ are infinitely pushed away together: viewed in this way, the pre-proof π represents an infinitely cut-expanded axiom.

However we have no infinite branch in the proof-structure of fig. 12h to support an infinite axiom. We need to introduce a new kind of infinite axiom as in fig. 12i which is not "above an infinite ray" — we call it a *virtual axiom*. We will thus distinguish between infinite axioms that are supported by a straight thread (which we will call real axioms) and infinite axioms supported by visitable paths (virtual axioms). Just as real infinite axioms, virtual axioms can also contain formula occurrences with finite addresses (indeed, consider π with an arbitrarily formula added in the conclusion sequent and pushed through all the cuts). In both cases, an infinite axiom is the invariant (under permutation of inference rules) of an infinite branch (in a pre-proof), but while real axioms are invariant of straight branches, virtual one are invariants of virtual branches.

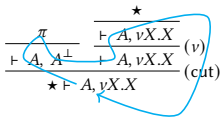


Figure 13: Let π be the proof in fig. 12g

Higher order. A final difficulty arises in the process of inventing infinitary proof structures. Consider the pre-proof in fig. 13: it consists of an infinite sequence of unfolding of a fixpoint such that, between two unfoldings, a cut introduces the infinite pre-proof π of conclusion $\vdash A, A^\perp$ studied in the last paragraph: as said there, it can be interpreted as an infinitely expanded axiom.

Let us imagine what the procedure to desequentialise the pre-proof in fig. 13 would look like, in particular to compute the visitable

paths and the infinite axioms. Typically one can imagine starting by tracing the sequents in such a way that they mimic the dynamics of an s-trip thereby recognising the infinitely many visitable paths from the infinitely many s-trips (each occurrence of π generating two maximal s-trips).

We introduced s-trips (in pre-proofs) so as to provide a counterpart to visitable paths (in proof structures) hence the proof structure desequentialised from the pre-proof in fig. 13 would also contain another visitable path: its corresponding s-trip (drawn in blue in fig. 13) which bounces through each copy of π by going up the blue trip and down the red trip of fig. 12g, and keeps going up. This visitable path resides with the undirected ν-ray in a real axiom.

Such a trip is not encompassed by def. 17: it would be an *higher order trip* that interleaves regular progression between sequents with full trips.

Definition 19. *Given a pre-proof π, an* **higher order trip** starting from F_1 *is a sequence $\tau = \{(s_i, F_i, d_i)\}_{i\in\omega}$ where s_i is a sequent in π, $F_i \in s_i$ and $d_i \in \{\uparrow, \downarrow, \uparrow^\omega, \downarrow^\omega\}$ such that $d_1 = \uparrow$ and for every $i < \omega$ exactly one of the following holds:*

- *either $d_i = \uparrow^\omega$, $d_{i+1} = \downarrow^\omega$, $s_i = s_{i+1}$ and there exist trips σ starting from F_i and σ' starting from F_{i+1} that have the same infinite branch associated with them; or,*
- *it falls in one of the conditions of def. 17.*

Furthermore for every i, $j < \omega$, there does not exist a sequent, s, in π such that $F_i \parr F_j \in s$.

To such trips correspond *higher order visitable paths* in proof-structures. Although our results scale to proof-structure containing such paths, to keep the presentation simple, we will not consider them, and thus consider only consider pre-proofs without higher order trips.

We bookend this discussion by summarizing the terms introduced: they go by pair, one in a pre-proof, and its corresponding notion in proof-structures.

Pre-proofs	Non-wellfounded Proof Structures
Axioms	Finite axiom links
Real branches (def. 7)	Real axioms
Virtual branches (def. 7)	Virtual axioms
Trips (def. 17)	Visitable paths
Higher order trips (def. 19)	Higher order visitable paths

In the next section, we will define the notions for proof-structures.

Remark 2. *The infinets introduced in [19] do not have this crucial additional structure of visitable paths (rendering the original definition of μMLL$^\infty$ proof structures in that paper incomplete), which can feature even in proof structures with finitely many cuts.*

5 NON-WELLFOUNDED PROOF STRUCTURES

In section 4 we have exposed the concepts necessary to define non-wellfounded proof structures (nwfps) and to desequentialize pre-proofs as if our proof-structures were infinite graphs. We formalise the definitions in this section.

Simple nwfps. In section 2.2 we formally expose MLL proof-nets as 3-tuples of the form $(\{F_i^{U_i}\}_{i\in I}, \mathfrak{R}, \Theta)$. These objects generalises to the non-wellfounded setting in the following way:

Firstly, the syntax tree of a μMALL formula occurrence F is the (possibly infinite) unfolding tree of the Fischer-Ladner graph of F. So the partial syntax trees $F_i^{U_i}$ could be potentially infinite.

Secondly, we might have infinitely many cut occurrences; so I could potentially be infinite.

Thirdly, recall from section 4 that we have three kinds of axioms: finite, real, and virtual. So, in μMLL$^\infty$, the set Θ of axioms is partitioned as $\Theta_f \uplus \Theta_r \uplus \Theta_v$ where, if we set $\mathcal{L} = \bigcup_{i\in I}\{\alpha_i u_i \mid \mathsf{addr}(F_i) = \alpha_i, u_i \in U_i\}$:

- Θ_f is the set of *finite axioms*: its elements are pairs of finite dual addresses from $\mathcal{L}$. They are links between leaves of partial syntax trees denoted by ax in fig. 12.
- Θ_r is the set of *real axioms*: the invariant (under permutation of inferences) of an infinite branch of a pre-proof supported by a straight thread. Its elements necessarily contain at least one infinite address and might contain visitable paths. They are denoted by ax_r^∞ in figs. 12c and 12f.
- Θ_v is the set of *virtual axioms*: the invariant (under permutation of inferences) of an infinite branch of a pre-proof supported by a visitable thread. Its elements necessarily contain visitable paths and might contain finite addresses from $\mathcal{L}$. They are denoted by ax_v^∞ in fig. 12i.

Hence Θ is a partition over $\mathcal{L} \cup V$ where V is the set of visitable paths.

An nwfps is defined in two steps: first, we define a *simple nwfps*. Then, an nwfps is nothing but a simple nwfps and its visitable paths.

Definition 20. *A* **simple nwfps** *is a 5-tuple $(\{F_i^{U_i}\}_{i\in I}, \mathfrak{R}, \Theta_f, \Theta_r, \Theta_v)$ which respectively comprises of a set of* partial syntax trees*, a set of* cuts*, a set of* finite axioms*, a set of* real axioms *such that:*

- *$\{F_i\}_{i\in I} \setminus \bigcup_{\kappa\in\mathfrak{R}} \kappa$ is finite.*
- *$\Theta_f \uplus \Theta_r \uplus \Theta_v$ is a partition of the set $\mathcal{L} = \bigcup_{i\in I}\{\alpha_i u_i \mid \mathsf{addr}(F_i) = \alpha_i, u_i \in U_i\}$ such Θ_f has only elements containing finite dual addresses, elements of Θ_r necessarily contain an infinite address and elements of Θ_v do not contain an infinite address.*

Example 4. *Consider the structure in fig. 12f. It defines a simple nwfps $(\{H^{U_1}, (A_\beta^\perp)^{U_2}\}, \emptyset, \{\theta_n\}_{n\geq 0}, \{\{(il)^\omega\}\}, \emptyset)$ such that α and β are disjoint, $U_1 = (il)^*.ir(l + r) + (il)^\omega$, U_2 is simply $\{\epsilon\}$, $\theta_0 = \{\alpha ir^2, \beta\}$ and for every $n > 0$, $\theta_n = \{(il)^{n-1}ir^2, (il)^{n-1}irl\}$.*

Visitable ends. The simple nwfps correspond to [19, def. 18]. On top of it we add the visitable paths and augment the axioms to make general nwfps.

Observe that permutation of inference rules in a pre-proof can induce a permutation of terms in its trips. However, if we only take the principal formulas of a trip (without repetition), this sequence is invariant. Further observe that this sequence is fixed by only stating the points of alternation of directions *i.e.* the cuts and tensors for $\downarrow$ to $\uparrow$ and the finite axioms for $\uparrow$ to $\downarrow$. Hence V is a set of sequences of alternating tensors (or cuts) and finite axioms. We must impose a sanity condition since any such sequence cannot be a visitable path.

Definition 21. *Given a simple nwfps, a* **visitable path** *is an infinite sequence $\{t_i\}_{i\in\mathbb{N}}$ such that if i is odd then t_i is either a tensor formula occurrence (F_j, u) for some $j \in I$ and $u \in \overline{U_j}$ or an element of*

$\mathfrak{K}$ *and if* i *is even then* $t_i \in \Theta_f$. *Further, there exists a pair of infinite sequences of addresses,* $(\{l_i\}_{i\in\mathbb{N}}, \{r_i\}_{i\in\mathbb{N}})$ *such that for every* $i \in \mathbb{N}$:

- *If* $t_{2i-1} = (F_j, u)$ *is a tensor formula, then* u *is a prefix of* l_i *and* r_i.
- *If* $t_{2i-1} = \{C, C^\perp\}$, *then* $\mathrm{addr}(C)$ *and* $\mathrm{addr}(C^\perp)$ *are prefixes of* l_i *and* r_i *respectively.*
- $\{r_i, l_{i+1}\} \subseteq t_{2i}$.

Two visitable paths $\{t_i\}_{i\in\mathbb{N}}$ *and* $\{t'_i\}_{i\in\mathbb{N}}$ *are* equivalent *if there exists* $m \in \mathbb{N}$ *such that for all* $i \geq m$, $t_i = t'_i$. *Their equivalence classes are called* **visitable ends**.

Remark 3. *In def. 21, the choice to start from tensors or cuts instead of axioms is arbitrary.*

Definition 22. *A* nwfps *is a 6-tuple* $\mathcal{R} = (\{F_i^{U_i}\}_{i\in I}, \mathfrak{K}, \Theta_f, V, \Theta_r, \Theta_v)$ *which respectively comprises of a set of* partial syntax trees, *a set of* cuts, *a set of* finite axioms, *a set of* real axioms, *a set of* visitable ends, *a set of* real axioms, *and a set of* virtual axioms *such that the following holds, where* $\mathcal{L} = \bigcup_{i\in I}\{\alpha_i u_i \mid \mathrm{addr}(F_i) = \alpha_i, u_i \in U_i\}$:

- $\mathcal{S} = (\{F_i^{U_i}\}_{i\in I}, \mathfrak{K}, \Theta_f, \{\theta \cap \mathcal{L} \mid \theta \in \Theta_r\}, \{\theta \cap \mathcal{L} \mid \theta \in \Theta_v\}\backslash\{\varnothing\})$ *is a simple* nwfps, *the* underlying simple nwfps *of* $\mathcal{R}$.
- V *is the set of visitable ends of* $\mathcal{S}$.
- $\Theta_f \uplus \Theta_r \uplus \Theta_v$ *is a partition of the set* $\mathcal{L} \cup V$ *where elements of* Θ_r *contain at least one infinite address, and, elements of* Θ_v *do not contain infinite addresses and contain at least one visitable end.*

The visible paths that are used to specify the infinite axioms of a nwfps are defined on the underlying simple nwfps (which is the simple nwfps with same components, and the paths removed in all the real and virtual axioms). In the same way, higher-order paths can be defined on nwfps[5].

Example 5. *In fig. 12f, we have the visitable path* $\rho = \{t_n\}_{n\in\mathbb{N}^*}$ *such that*

$$t_n = \begin{cases} \theta' & n = 1; \\ \theta_{\lfloor \frac{n}{2} \rfloor} & n \text{ is odd;} \\ (B_1, (il)^{\lfloor \frac{n}{2} \rfloor} ir) & n \text{ is even.} \end{cases}$$

Check that every other visitable path we can produce is a suffix of this. Hence there is only one visitable end (say, $[\rho]$*). Observe that we need to augment* θ_0 *by adding this visitable end i.e.* $\theta_0 = \{\{(il)^\omega, [\rho]\}$. *There are no virtual axioms, hence* $\Theta_v = \emptyset$. *Check that* $\Theta_f \cup \Theta_r \cup \Theta_v$ *is a partition of* $\alpha U_1 \cup \beta U_2 \cup \{[\rho]\}$.

Figure 14: loc **changes the address of** ϕ **from** α **to** β.

Recall from section 3 we are in an extended system *i.e.* $\mu\mathsf{MLL}^\infty$ with $(\mathrm{Loc}(\iota))$. Hence we need add new relocation cells with one premise and one conclusion, changing the addresses (as illustrated in fig. 14). Formally nwfps have one more component loc: a bijection between a finite subset L of $\mathcal{L}$ and a finite subset C of doors such that the underlying formulæ of the image and the antecedent are equal. However, since the geometry of nwfps is completely unaffected by the presence of finitely many loc nodes we will ignore them.

Desequentialization. Desequentialization of a $\mu\mathsf{MLL}^\infty$ pre-proof cannot be done inductively as in MLL since the objects we are constructing are potentially infinite. We translate each component sequentially. Careful readers can observe that some of those procedures could be infinitary.

Definition 23. *Let* π *be a pre-proof of the* $\mu\mathsf{MLL}^\infty$ *sequent* $\vdash \Gamma$. *The* **desequentialization** *of* π *is a* nwfps $(\{F_i^{U_i}\}_{i\in I}, \mathfrak{K}, \Theta_f, V, \Theta_r, \Theta_v)$ *satisfying the following conditions and is denoted by* $\mathrm{Deseq}(\pi)$.

- *for any cut in* π *that introduces two occurrences,* C *and* $C^\perp$, $\{C, C^\perp\} \in \mathfrak{K}$.
- $\{F_i\}_{i\in I} = \Gamma \cup \bigcup_{\kappa\in\mathfrak{K}} \kappa$.
- *for every* $i \in I$, $U_i = \mathrm{addr}(F_i)^{-1}\mathrm{addr}(\pi)$.
- *for every axiom linking* (F_i, u_i) *to* (F_j, u_j), $\{\mathrm{addr}(F_i).u_i, \mathrm{addr}(F_j).u_j\} \in \Theta_f$.
- *for every maximal s-trips in* π, *we collect the points of alternation of directions which gives us a sequence of cuts or tensor and axioms. This gives us a set of visitable path, which in turn will give us a set of visitable ends,* V.
- *for every real infinite branch* γ *in* π, $\theta \in \Theta$ *is the largest subset of* $\mathcal{L} \cup V$ *such that:*
  - *for every* $\mathrm{addr}(F)u \in \theta$, *either* $u = u_1u_2\ldots$ *is an infinite word and* $\{(F, u_1 \ldots u_i)\}_{i\in\omega}$ *is a straight thread of* γ; *or* u *is a finite word and* (F, u) *occurs in infinitely many sequents along* γ.
  - *for every* $v \in \theta \cap V$ *there exists a trip* ρ *associated with* γ *such that* v *is obtained from* ρ.
- *For every virtual infinite branch,* γ, *in* π, $\theta \in \Theta$ *is the largest subset of* $\mathcal{L} \cup V$ *such that:*
  - *for every* $\mathrm{addr}(F)u \in \theta$, u *is a finite word and* (F, u) *occurs in infinitely many sequents along* γ.
  - *for every* $v \in \theta \cap V$ *there exists a trip* ρ *associated with* γ *such that* v *is obtained from* ρ.

Proposition 3. *The desequentialisation of a simple pre-proof is a simple* nwfps *with no virtual axioms.*

Example 6. *If we desequentialize the proof in fig. 12d we get the* nwfps *described in examples 4 and 5.*

6 CORRECTNESS CRITERION

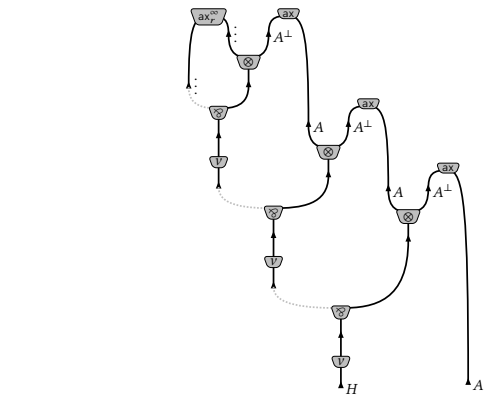
In this section, we develop a correctness criterion on nwfps, strengthening the correctness criterion in [19] to account for visitable paths. It has two conditions:

DR-correctness: The orthogonal graph is acyclic and connected.
Lock-freeness: No infinite set of nodes forms an ideal in kingdom inclusion order.

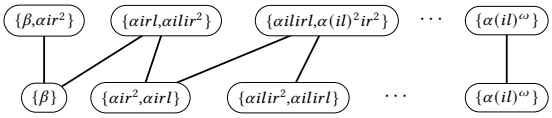
DR-correctness. Because $\mu\mathsf{MLL}^\infty$ contains MLL, DR-correctness is necessary. The reason why it is not sufficient is more subtle: the presence of infinitely many vertices in a nwfps leads to pathological cases where in order to sequentialise a certain vertex needs to wait for infinitely many vertices to sequentialise.

Def. 11 to 14 lifts to simple nwfps. Therefore for a nwfps with no visitable paths, DR-correctness as stated in [19] means that, for every switching, the orthogonal graph is acyclic and connected.

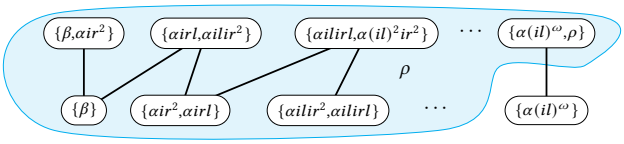
[5] These paths can then participate in real and virtual axioms in more general notions of structure. The nwfps studied here are just the second level of a hierarchy that starts with simple nwfps.



(a) The result of the switching sw



(b) The simple orthogonal graph *wrt.* sw



(c) The orthogonal graph *wrt.* sw

Figure 15: Illustrating DR-correctness on fig. 12f

However, visitable paths need to be incorporated into orthogonal graphs; so, just as we first defined simple NWFPS on which visitable paths are defined, information needed for general NWFPS, we first define simple orthogonal graphs of simple NWFPS, and enrich them into orthogonal graphs. Given a NWFPS $\mathcal{R}$ we call an orthogonal graph of its underlying simple NWFPS a ***simple orthogonal graph*** of $\mathcal{R}$.

Proposition 4. *Let $G_0^{sw}(\mathcal{R})$ be a simple orthogonal graph of $\mathcal{R}$) for some switching sw. There is a one-one correspondence between $E_{\mathcal{R}} = \bigcup_{sw}\{\rho \mid \rho$ is an end in $G_0^{sw}(\mathcal{R})\}$ and the set of visitable ends of $\mathcal{R}$.*

Definition 24. *Given a simple orthogonal graph $G_0^{sw}(\mathcal{R}) = (\Theta, [\mathrm{SW}], E_0)$ of $\mathcal{R}$, the* **orthogonal graph** *(denoted $G^{sw}(\mathcal{R})$) is the undirected hybridgraph $(\Theta', [\mathrm{SW}], E_0, E_1)$ such that Θ' is Θ augmented with visitable paths (hence E_0 is unchanged) and for every $\theta \in \Theta_r \cup \Theta_v$ that contains a visitable end, $\{\theta\} \cup S \in E_1$ where $S \subseteq \Theta \cup [\mathrm{SW}]$ is the set of all nodes appearing in every end of θ. A* **pure path** *in $G^{sw}(\mathcal{R})$ is path comprised of only E_0 or E_1 but not both.*

Definition 25. *A NWFPS, $\mathcal{R}$, is said to be* **DR-correct** *if for any switching sw, between any two nodes of $G^{sw}(\mathcal{R})$ there is exactly one pure path.*

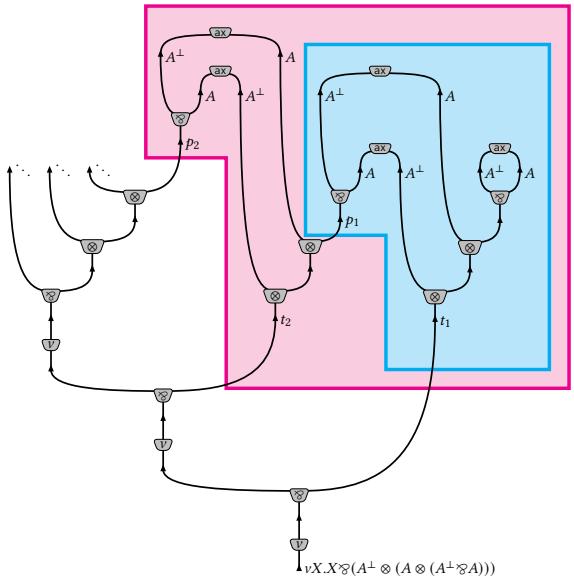


Figure 16: A DR-correct NWFPS exhibiting kingdoms

Example 7. *We illustrate DR-correctness on the NWFPS depicted in fig. 12f. There are infinitely many switchings but they can be boiled down to three cases: (i) only finitely many ⅋s switch to the left (ii) only finitely many ⅋s switch to the right (iii) infinitely many ⅋s switches to both left and right. We illustrate the particular case when all the ⅋s are switched to the right. This switching sw is depicted in fig. 15a.*

The corresponding simple orthogonal graph is depicted in fig. 15b. It has two connected components and one of them has an infinite path (denoted by ρ). The orthogonal graph has an extra hyperedge illustrated by the cyan region in fig. 15c. Observe that there is exactly one pure path between two nodes.

Lock-freeness. The lock-freeness condition as stated in [19] (for finitely many cuts) lifts straighforwardly from simple NWFPS to NWFPS. We reformulate the condition through non-wellfounded substructures (sub-NWFPS) and kingdoms.

Firstly we note that def. 15 and proposition 1 can easily be reformulated for DR-correct NWFPS.

Example 8. *The NWFPS in fig. 16 is DR-correct. The kingdom $K(p_1)$ and $K(t_2)$ are drawn in cyan and magenta. Observe that $t_1 \ll p_1 \ll t_2 \ll p_2 \ll \cdots$*

Definition 26. *A NWFPS is* **lock-free** *if $\{m \mid n \ll m\}$ is finite for all n.*

The kingdom of a node in a NWFPS is the set of nodes that are sequentialised after it in every sequentialisation. Hence, the lock-freeness condition states that, every node is included in finitely many kingdoms *i.e.* there is only finitely many nodes that need to be desequentialised before it.

Definition 27. *A NWFPS is an* **infinet** *if it is DR-correct and lock-free.*

Proposition 5. *Let π be a pre-proof.* $\mathrm{Deseq}(\pi)$ *is an infinet.*

Sequentialisation. We now give an informal description of the corecursive definition of sequentialisation. The technique in [19] basically follows the standard procedure for MLL but with a guarantee of *fairness* preventing a situation where the exploration of a branch is forgotten since the sequentialisation of another branch is forever prioritised. Fairness is ensured by time-stamping the doors of the infinet with elements of $\mathbb{N} \cup \{\infty\}$, which dictates that at any particular step the node with the least time-stamp is to be sequentialised.

We strengthen this time-stamping to account for infinitely many cuts. Given an infinet, we treat cuts as tensors: a "quasi" infinet (say $\mathcal{R}$) with potentially infinitely many conclusions. We carefully initialize the time-stamping such that infinitely many numbers are free to be used as time-stamps at later stages of the sequentialisation. Consider $t_{\mathcal{R}}$ which injectively time-stamps every maximal door in the $\ll$ ordering by powers of two[6] and every other door by ∞. $\textsc{Sequentialise}(\mathcal{R}, t_{\mathcal{R}})$ chooses the door, F, with least time-stamp, applies the corresponding rule on the finite prefix of the sequentialisation being built, and relaunches $\textsc{Sequentialise}(\mathcal{R}', t_{\mathcal{R}'})$ for every sub-infinet, $\mathcal{R}'$ and the time-stamping that results from removing F from $\mathcal{R}$.

Theorem 2. *Given an infinet, $\mathcal{R}$, and a proper time-stamping, $t_{\mathcal{R}}$,* $\mathsf{Deseq}(\textsc{Sequentialise}(\mathcal{R}, t_{\mathcal{R}})) = \mathcal{R}$.

7 CUT-ELIMINATION FOR VALID INFINETS

We now provide the main result of this paper: cut-elimination on infinets. As discussed in section 1, validity is sufficient (but not necessary, see figs. 8b and 8c) for its productivity. We retain the notion of validity in the sequent calculus and simply lift def. 8 to NWFPS.

Definition 28. *A NWFPS $\mathcal{R}$ is* **valid** *if $\Theta_v = \emptyset$ and for $\theta \in \Theta_r$, there is $\alpha u \in \theta$ such that* $\mathsf{addr}(F_i) = \alpha$ *and (F_i, v) is a ν-formula for infinitely many prefixes v of u.*

Proposition 6. *π is a proof iff* $\mathsf{Deseq}(\pi)$ *is a valid infinet.*

Cut-reduction rules. The cut-elimination procedure for infinets is adapted from MLL: during cut-elimination, finite axioms interact with cuts by annihilating one another, replaced by a wire. To satisfy our stated goal, we thus need to define cut-elimination rules also for infinite axioms. Consider the infinet, $\mathcal{R}$, in fig. 17a. The straightforward adaptation of the finitary rule makes no sense, as it would result in reducing $\mathcal{R}$ to the object in fig. 17b which is not an infinet: first, it would require to put a structure $\mathcal{S}$ atop of an infinite path of ν-cells; second, the types of this infinite path do not match[7]. To justify a better rule, let us see the situation in sequent calculus: consider a sequentialisation π as in fig. 18 of $\mathcal{R}$ (where $\mathsf{Deseq}(\pi') = \mathcal{S}$ and $\Gamma = \{A_1, \dots, A_n\}$).

The infinite axiom is represented in π by the infinite branch and the only way to make it interact with π' (in the way $\mathcal{S}$ interacts with the infinite axiom) using the rules in def. 16 is by commuting the cut with one ν-rule. Iterating such permutations builds the infinite sequence of proofs of fig. 18 which all desequentialise to $\mathcal{R}$.

[6] This labelling is arbitrary: any cofinite sequence in place of powers of two works.
[7] Along the undirected ν-ray, the types ought to remain equal to $\nu X.X$, but change to $A^{\perp}$ above.

This sequence converges (as a tree) to the proof in fig. 17c, where π' has been deleted and Γ is supported by the infinite branch. Desequentialised, this yields the proof-structure in fig. 17d. So, an infinitary axiom and a cut interact by removing the whole subinfinet "above" the cut. Now recall that the kingdom of an occurrence is the subinfinet that is always sequentialised above it . Hence, the subinfinet that has to be erased is indeed a kingdom. Although this operation will be represented by a single rule it does not correspond to one step of cut-elimination in the sequent calculus but to an infinite sequence of permutations.

Definition 29. *The cut-reduction rules are the ones illustrated in fig. 19 and the usual $\otimes/\parr$ rule of* MLL *cut reduction. A sequence of infinets, $(\mathcal{R}_i)_{i\geqslant 0}$, is called a* **reduction sequence** *if for all i, $\mathcal{R}_i \to_\kappa \mathcal{R}_{i+1}$ for some cut κ in $\mathcal{R}_i$.*

Proposition 7. *Cut-reduction on infinets is confluent.*

Limits of reduction sequences. To prove that an infinite sequence of these reductions converges to some infinet, it is possible to define a topology on the set of infinets, which accounts for the cuts moving upwards during the cut-reduction procedure, by giving weights to cuts. One way to achieve it is to consider the heights of the cuts in a sequentialisation: basically, we use a sequentialisation to give a tree-like ordering to a proof-structure, and hence, a notion of distance compatible with the reduction. This method works for straight thread valid infinets, as we have theorem 1 for the sequent calculus.

Thus, infinitary cut-elimination is carried out in **valid** and **correct** NWFPS: correctness to use the tree topology of the sequentialisations (π and π' are at a distance $\leq 2^{-h}$ if they coincide up to height h); validity to ensure productivity.

As the reductions we introduced for infinets do not correspond to a single step of cut-reduction in the sequent calculus, we introduce a new reduction in the sequent calculus. We define the family of relations $\{\Rightarrow_h |\ h \in \mathbb{N}\}$ on $\mu\mathsf{MLL}^{\infty}$ proofs such that $\pi_0 \Rightarrow_h \pi'$ if the restrictions of π_0 and π' below height h coincide and

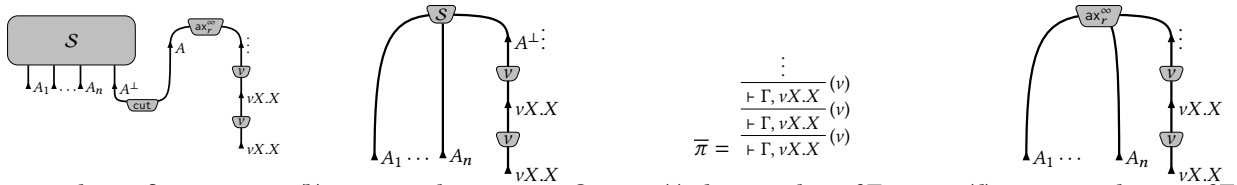
- either π' is the limit of an infinite sequence $(\pi_i)_{i\geqslant 0}$ such that for all $i \geq 0$, π_{i+1} is obtained from π_i by a permutation
- or there exists a finite sequence $(\pi_i)_{i\leqslant n}$ such that for all $i \leq n-1$, π_{i+1} is obtained from π_i by a permutation of an inference rule, and π' can be obtained from π_n by an external cut-reduction.

Definition 30. *We say that $\pi \Rightarrow_h \pi'$ is a* **sequentialisation of a reduction**, *$\mathcal{R} \to_\kappa \mathcal{R}'$, if* $\mathsf{Deseq}(\pi) = \mathcal{R}$ *and* $\mathsf{Deseq}(\pi') = \mathcal{R}'$ *and h is maximal (i.e. for every $h' > h$, $\pi \not\Rightarrow_{h'} \pi'$).*

We also extend Def. 30 to define a sequentialisation of a reduction sequence.

Lemma 1. *Let $\mathcal{R}_0$ be valid and $(\mathcal{R}_i)_{i\geqslant 0}$ be a reduction sequence. Every sequentialisation of $(\mathcal{R}_i)_{i\geqslant 0}$ has a limit which is a proof. Furthermore, the limits π and π' of two sequentialisations of the reduction sequence satisfy* $\mathsf{Deseq}(\pi) = \mathsf{Deseq}(\pi')$.

Definition 31. *Let $\mathcal{R}_0$ be valid and $(\mathcal{R}_i)_{i\geqslant 0}$ be a reduction sequence.The* **limit** *of $(\mathcal{R}_i)_{i\geqslant 0}$ is the desequentialisation of the limit of a sequentialisation of $(\mathcal{R}_i)_{i\geqslant 0}$.*



(a) $\mathcal{R}$: a cut with an infinite axiom (b) Naive cut-elimination in $\mathcal{R}$ (c) The normal proof $\overline{\pi}$ (d) Desequentialisation of $\overline{\pi}$

Figure 17: Exhibiting the necessity of the kingdom erasure rule

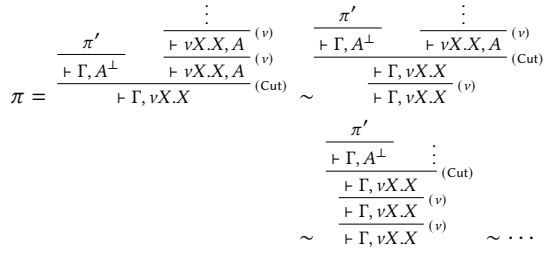


Figure 18: A productive infinite reduction

Fair sequences. Just as in the sequent calculus, not every reduction sequences converge towards a cut-free infinet: it is possible to never reduce some cuts.

Definition 32. *A reduction sequence $(\mathcal{R}_i)_{i\geqslant 0}$ is* **fair** *if for every $i \geq 0$ and r such that $\mathcal{R}_i \to_r \mathcal{R}'$, there is some $j \geq i$, such that r cannot be reduced in $\mathcal{R}_j$, i.e. there is no infinet, $\mathcal{R}''$ such that $\mathcal{R}_j \to_r \mathcal{R}''$.*

Example 9. *Consider the desequentialisation of the pre-proofs in figs. 8b and 8c and the steps of cut reduction on it in fig. 20 using the rules in fig. 19. Observe it closely resembles the reduction sequence in example 3.*

Theorem 3. *Let $\mathcal{R}_0$ be a valid infinet and $(\mathcal{R}_i)_{i\geqslant 0}$ be a fair reduction sequence. Its limit is a valid cut-free infinet.*

8 CONCLUSION

We have developed the parallel syntax of non-wellfounded proof theory by generalizing the non-wellfounded proof structures in [19] to account for the presence of infinitely many cuts; and provided a cut-elimination result on these structures. Indeed the requirement of containing only finitely many cuts in [19] prevents one to consider any circular pre-proof having a cut between the target and the source of a back-edge, as in fig. 8c. With our present generalisation and confluent cut-elimination, we hope to contribute to a better understanding of non-wellfounded proofs, from both syntactic and semantic points of view.

Related works. Several strands of research around non-wellfounded and circular proofs and their computational correspondence with (co)recursive programming are related to the present work:

Expressing theories with circular proofs is a fruitful research direction: various logical or computational settings can be expressed with circular proofs, to formalise some sort of infinite behaviour [16, 17, 21]. In addition, circular proofs also help understanding the meta-theory of traditional fixed-point logics such as the linear-time or modal μ-calculus [3, 22, 23].

Validity criteria are the subject of active investigations. For instance, Baelde et al.[5] introduced *bouncing-validity* in which the geometry of threads is more complexvalidating more non-wellfounded derivations. On the other hand, more restricted criteria can be interesting, even though they validate fewer circular derivations, as they can be less costly to check [32, 38].

The denotational semantics of circular proofs is yet to be fully understood and is a challenging direction [25, 29, 35]. Recent results on the relational semantics of finitary proofs of μLL [24, 29] allowed to make progress in interpreting circular proofs as well.

Interactive and automated theorem proving are impacted by circular proofs – which use a form of implicit induction– [4, 12, 26, 30, 34], be it for building automated proofs of (co)inductive statements or for easing the manipulation of coinduction in proof assistants such as Coq. This active research topic can be impacted by advances on validity conditions or on the fine-grained structure of circular proofs [10, 20] or approaches such as copatterns [2].

Alternative styles for handling non-wellfounded proofs are actively researched to provide solutions to some of the limitations of sequent proofs: various logical frameworks could provide solutions to some of the limitations of sequent proofs. In addition to our line of work on proof-nets, an interesting direction is that of Das on designing deep-inference proof systems for circular reasoning in his StrIP project [15].

The present work – devising proof-nets for non-wellfounded proof theory – nestles itself naturally in the last category, but is also related to other approaches: (i) proof-nets are closer to semantics than sequent proofs and infinets can potentially impact the understanding of the denotational invariants of non-wellfounded proofs and the design of their denotational semantics; (ii) the bouncing-validity condition will strongly benefit from less sequential proof structures and (iii) infinets would serve as better models of coinductive programs.

Future Work. In order to model coinductive programs and provide less stringent guard conditions we have successfully overcome the first three steps: devise the canonical proof-objects and their cut-rules, and provide a cut-elimination result on these objects. There are two directions from here: (i) improve our cut-elimination result by handling more flexible validity conditions (ii) define the circular counterpart to infinets which would both be sufficiently expressive

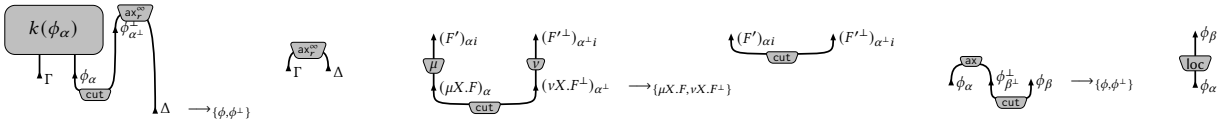


Figure 19: Cut reduction rules for $\mu\mathsf{MLL}^\infty$ infinets. $k(\phi_\alpha)$ denotes the kingdom of the occurrence ϕ_α.

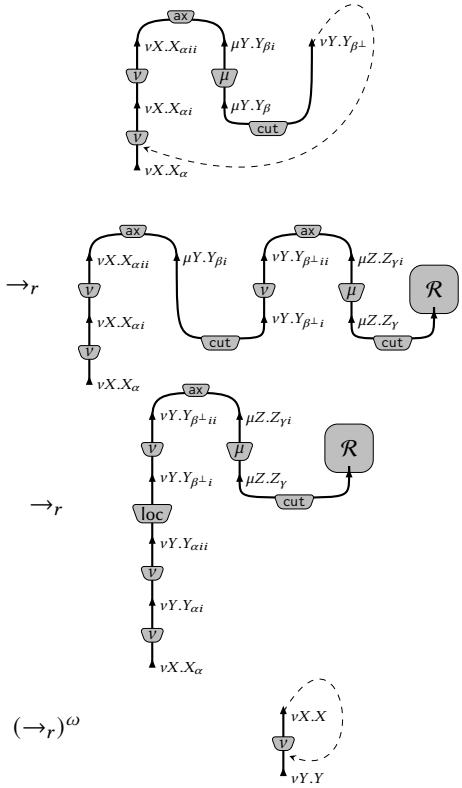


Figure 20: Cut reduction on the desequentialisation of the pre-proofs in figs. 8b and 8c

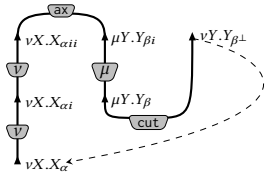


Figure 21: A circular infinet

and have decidable validity. A synthesis of these two will give us the desideratum. We will sketch a few ideas using an example.

Consider the desequentialisation of the pre-proofs in figs. 8b and 8c. They are same infinet fig. 20. Pre-empting *circular infinets* we represent the recurring subnet using a back-edge in fig. 21. However this formulation is hypothetical and there are two impediments to our desired goal:

– In order to satisfactorily adapt bouncing-validity one would need to provide a cut-elimination result independent of cut-elimination in the seqeunt calculus. This will possibly require new proof techniques and one approach will be to investigate other cut-elimination proof methods developed in the infinitary setting, for instance Mints' *continuous cut-elimination* [31].
– Circular infinets are not exactly the class of nets that are the desequentialisation of some circular proof since circularity of proofs is not preserved by permutations of inferences. Furthermore, if one imposes restrictions on infinets like allowing only regular partial syntax trees, they lose a lot of expressiveness since a finite description of visitable paths is not clear. In future work, we plan to do a comparative study of various finitely representable classes of infinets and their decidable questions.

REFERENCES

[1] Martín Abadi, Luca Cardelli, Pierre-Louis Curien, and Jean-Jacques Lévy. 1991. Explicit Substitutions. *Journal of Functional Programming* 1, 4 (1991), 375–416.

[2] Andreas Abel and Brigitte Pientka. 2013. Wellfounded recursion with copatterns: a unified approach to termination and productivity. In *ACM SIGPLAN International Conference on Functional Programming, ICFP'13, Boston, MA, USA - September 25 - 27, 2013*, Greg Morrisett and Tarmo Uustalu (Eds.). ACM, 185–196. https://doi.org/10.1145/2500365.2500591

[3] Bahareh Afshari, Gerhard Jäger, and Graham E. Leigh. 2019. An Infinitary Treatment of Full Mu-Calculus. In *Logic, Language, Information, and Computation*, Rosalie Iemhoff, Michael Moortgat, and Ruy de Queiroz (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 17–34.

[4] Jürgen Avenhaus, Ulrich Kühler, Tobias Schmidt-Samoa, and Claus-Peter Wirth. 2003. How to Prove Inductive Theorems? QUODLIBET!. In *Automated Deduction - CADE-19, 19th International Conference on Automated Deduction Miami Beach, FL, USA, July 28 - August 2, 2003, Proceedings (Lecture Notes in Computer Science, Vol. 2741)*. Springer, 328–333. https://doi.org/10.1007/978-3-540-45085-6_29

[5] David Baelde, Amina Doumane, Denis Kuperberg, and Alexis Saurin. 2020. Bouncing threads for circular and non-wellfounded proofs. (2020). https://arxiv.org/abs/2005.08257.

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

[7] Marc Bagnol, Amina Doumane, and Alexis Saurin. 2015. On the Dependencies of Logical Rules. In *Foundations of Software Science and Computation Structures - 18th International Conference, FoSSaCS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings*. 436–450. https://doi.org/10.1007/978-3-662-46678-0_28

[8] G. Bellin and J. van de Wiele. 1995. Subnets of Proof-Nets in MLL˙. In *Proceedings of the Workshop on Advances in Linear Logic*. Cambridge University Press, USA, 249–270.

[9] James Brotherston. 2005. Cyclic Proofs for First-Order Logic with Inductive Definitions. In *Automated Reasoning with Analytic Tableaux and Related Methods*, Bernhard Beckert (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 78–92.

[10] James Brotherston, Nikos Gorogiannis, and Rasmus Lerchedahl Petersen. 2012. A Generic Cyclic Theorem Prover. In *Programming Languages and Systems - 10th Asian Symposium, APLAS 2012, Kyoto, Japan, December 11-13, 2012. Proceedings (Lecture Notes in Computer Science, Vol. 7705)*. Springer, 350–367. https://doi.org/10.1007/978-3-642-35182-2_25

[11] James Brotherston and Alex Simpson. 2010. Sequent calculi for induction and infinite descent. *Journal of Logic and Computation* 21, 6 (10 2010), 1177–1216. https://doi.org/10.1093/logcom/exq052 arXiv:https://academic.oup.com/logcom/article-pdf/21/6/1177/2787531/exq052.pdf

[12] Alan Bundy. 2001. The Automation of Proof by Mathematical Induction. In *Handbook of Automated Reasoning (in 2 volumes)*. Elsevier and MIT Press, 845–911.

[13] Alonzo Church and J. B. Rosser. 1936. Some Properties of Conversion. *Trans. Amer. Math. Soc.* 39, 3 (1936), 472–482. http://www.jstor.org/stable/1989762

[14] Pierre-Louis Curien. 2006. Introduction to linear logic and ludics, part II. , 44 pages.

[15] Anupam Das. 2019. Structure vs. Invariants in Proofs: project announcement. (2019). Talk at CiSS-19, http://www.cse.chalmers.se/~bahafs/CiSS2019/programme.html.

[16] Anupam Das, Amina Doumane, and Damien Pous. 2018. Left-Handed Completeness for Kleene algebra, via Cyclic Proofs. In *LPAR (EPiC Series in Computing, Vol. 57)*. EasyChair, 271–289.

[17] Anupam Das and Damien Pous. 2017. A Cut-Free Cyclic Proof System for Kleene Algebra. In *Automated Reasoning with Analytic Tableaux and Related Methods*, Renate A. Schmidt and Cláudia Nalon (Eds.). Springer International Publishing, Cham, 261–277.

[18] Abhishek De, Luc Pellissier, and Alexis Saurin. [n. d.]. Eliminating infinitely many cuts in non-wellfounded MLL proof-nets. ([n. d.]). https://hal.archives-ouvertes.fr/hal-03235591 Available at: `https://hal.archives-ouvertes.fr/hal-03235591`.

[19] Abhishek De and Alexis Saurin. 2019. Infinets: The Parallel Syntax for Non-wellfounded Proof-Theory. In *TABLEAUX 2019*, Serenella Cerrito and Andrei Popescu (Eds.). Springer International Publishing, 297–316. https://doi.org/10.1007/978-3-030-29026-9_17

[20] Farzaneh Derakhshan and Frank Pfenning. 2019. Circular Proofs as Session-Typed Processes: A Local Validity Condition. *arXiv e-prints*, Article arXiv:1908.01909 (Aug. 2019), arXiv:1908.01909 pages. arXiv:1908.01909 [cs.LO]

[21] Simon Docherty and Reuben N. S. Rowe. 2019. A Non-wellfounded, Labelled Proof System for Propositional Dynamic Logic. In *Automated Reasoning with Analytic Tableaux and Related Methods*, Serenella Cerrito and Andrei Popescu (Eds.). Springer International Publishing, Cham, 335–352.

[22] Amina Doumane. 2017. Constructive completeness for the linear-time μ-calculus. In *LICS*. IEEE Computer Society, 1–12.

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

[24] Thomas Ehrhard and Farzad Jafar-Rahmani. 2021. Categorical models of Linear Logic with fixed points of formulas. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*. IEEE, 1–13. https://doi.org/10.1109/LICS52264.2021.9470664

[25] Jérôme Fortier and Luigi Santocanale. 2013. Cuts for circular proofs: semantics and cut-elimination. In *CSL*.

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

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

[28] Jean-Yves Girard. 1996. Proof-nets: The parallel syntax for proof-theory. In *Logic and Algebra*. Marcel Dekker, 97–124.

[29] Farzad Jafarrahmani. 2018. *Denotational semantics of linear logic with least and greatest fixpoint.* Master's thesis. Université Paris Diderot.

[30] Ralph Matthes. 1999. Monotone Fixed-Point Types and Strong Normalization. In *Computer Science Logic*, Georg Gottlob, Etienne Grandjean, and Katrin Seyr (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 298–312.

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

[32] Rémi Nollet, Alexis Saurin, and Christine Tasson. 2018. Local Validity for Circular Proofs in Linear Logic with Fixed Points. In *CSL (LIPIcs, Vol. 119)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 35:1–35:23.

[33] Dag Prawitz. 1965. *Natural Deduction: A Proof-Theoretical Study.* Dover Publications.

[34] Martin Protzen. 1994. Lazy generation of induction hypotheses. In *Automated Deduction — CADE-12*, Alan Bundy (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 42–56.

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

[36] Ulrich Schöpp and Alex K. Simpson. 2002. Verifying Temporal Properties Using Explicit Approximants: Completeness for Context-free Processes. In *FoSSaCS (Lecture Notes in Computer Science, Vol. 2303)*. Springer, 372–386.

[37] Christoph Sprenger and Mads Dam. 2003. On the Structure of Inductive Reasoning: Circular and Tree-Shaped Proofs in the μCalculus. In *Foundations of Software Science and Computation Structures*, Andrew D. Gordon (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 425–440.

[38] Sorin Stratulat. 2017. Cyclic Proofs with Ordering Constraints. In *Automated Reasoning with Analytic Tableaux and Related Methods - 26th International Conference, TABLEAUX 2017, Brasília, Brazil, September 25-28, 2017, Proceedings*. 311–327.