Kleene Star, Subexponentials without Contraction, and Infinite Computations

Stepan L. Kuznetsov

Steklov Mathematical Institute of RAS

Abstract

We present an extension of intuitionistic non-commutative linear logic with Kleene star and subexponentials which allow permutation and/or weakening, but not contraction. Subexponentials which allow contraction are useful for specifying correct terminating of computing systems (e.g., Turing machines). Dually, we show that Kleene star axiomatized by an omega-rule allows modelling infinite (never terminating) behaviour. For our system we prove that it is in the Π_1^0 complexity class. Actually, it is Π_1^0-complete due to Buszkowski (2007). We also show Π_1^0-hardness of the unidirectional fragment of this logic with two subexponentials and Kleene star (this result does not follow from Buszkowski's construction). The omega-rule axiomatization can be equivalently reformulated as calculus with non-well-founded proofs (Das & Pous, 2018). We also consider the fragment of this calculus with circular proofs. This fragment is capable of modelling looping of a Turing machine, but, interestingly enough, some non-cyclic computations can also be captured by this circular fragment.

1 Introduction

We study non-commutative intuitionistic linear logic, or the multiplicative-additive Lambek calculus (MALC) [8], extended *at the same time* with subexponentials and the Kleene star, axiomatized in an infinitary way. We show, on some model examples, how this system can be used for specifying *infinite* computations ("survival"). This is dual to the more traditional use of subexponentials for modelling finite (terminating) computations. For finite computations, the necessary number of copies of the machine instructions (transition rules) are obtained by applying the contraction rule for one of the subexponentials. Here, dually, we reduplicate instructions using the Kleene star, and contraction in our system is disallowed.

Notice that we consider infinite computations of standard, finite systems, operating on finite (at each time) data, in contrast to frameworks with ω-words, like the one proposed in [30]. Unlike the use of situation calculus for specifying non-terminating processes [5], our approach here is purely propositional, without quantifiers and predicates. If compared to the algebraic framework for infinite computations proposed in [7], which is purely equational, our system is *inequational* in its nature, making the direction of computation process more explicit. To this extent, our approach is close to μMALL [1, 19], which is an extension of linear logic without (sub)exponentials with fixpoint operators. These fixpoint operations also allow replication of formulae, and thus encoding of (co-)induction. The system discussed in this paper is in a sense more specific: Kleene star is a concrete (but very important) example of least fixpoint. However, our system is non-commutative, that is why we need subexponentials locally allowing permutation.

We start with a Gentzen-style sequent calculus for MALC. Formulae of MALC are built from a countable set of variables and constant $\mathbf{1}$ using three binary connectives: $\cdot$ (product), $\backslash$ (left division), and $/$ (right division). Division operations are non-commutative linear implications. Sequents are intuitionistic-style, of the form $\Gamma \to C$, where C is a formula and Γ is a sequence of formulae (possibly empty).

In this sequential formulation of MALC, axioms and rules are as follows:

$$\frac{}{F \to F}\,\mathrm{Id} \qquad \frac{\Gamma, \Delta \to C}{\Gamma, \mathbf{1}, \Delta \to C}\,\mathbf{1}_{\mathrm{L}} \qquad \frac{}{\to \mathbf{1}}\,\mathbf{1}_{\mathrm{R}} \qquad \frac{\Pi \to F \quad \Gamma, F, \Delta \to C}{\Gamma, \Pi, \Delta \to C}\,\mathrm{Cut}$$

$$\frac{\Pi \to F \quad \Gamma, G, \Delta \to C}{\Gamma, \Pi, F \backslash G, \Delta \to C}\,\backslash_{\mathrm{L}} \qquad \frac{F, \Pi \to G}{\Pi \to F \backslash G}\,\backslash_{\mathrm{R}} \qquad \frac{\Gamma, F, G, \Delta \to C}{\Gamma, F \cdot G, \Delta \to C}\,\cdot_{\mathrm{L}}$$

$$\frac{\Pi \to F \quad \Gamma, G, \Delta \to C}{\Gamma, G / F, \Pi, \Delta \to C}\,/_{\mathrm{L}} \qquad \frac{\Pi, F \to G}{\Pi \to G / F}\,/_{\mathrm{R}} \qquad \frac{\Gamma \to F \quad \Delta \to G}{\Gamma, \Delta \to F \cdot G}\,\cdot_{\mathrm{R}}$$

$$\frac{\Gamma, F, \Delta \to C}{\Gamma, F \wedge G, \Delta \to C}\,\wedge_{\mathrm{L}} \qquad \frac{\Gamma, G, \Delta \to C}{\Gamma, F \wedge G, \Delta \to C}\,\wedge_{\mathrm{L}} \qquad \frac{\Pi \to F \quad \Pi \to G}{\Pi \to F \wedge G}\,\wedge_{\mathrm{R}}$$

$$\frac{\Gamma, F, \Delta \to C \quad \Gamma, G, \Delta \to C}{\Gamma, F \vee G, \Delta \to C}\ \vee_{\mathrm{L}} \qquad \frac{\Pi \to F}{\Pi \to F \vee G}\ \vee_{\mathrm{R}} \qquad \frac{\Pi \to G}{\Pi \to F \vee G}\ \vee_{\mathrm{R}}$$

Linear logic, as introduced by Girard [6], includes a special unary connective called the exponential. Under this connective, structural rules (contraction and weakening) are allowed, while they cannot be applied to arbitrary formulae. A further extension of linear logic with *subexponentials* [17] allows a more fine-grained control of structural rules. Each subexponential allows a specific subset of structural rules to be applied.

One of the major applications of logical frameworks based on linear logic and its subexponential extensions is connected to specifications of operational semantics [31, 18, 20, 25]. Ordered logical frameworks [26, 25], or frameworks based on non-commutative linear logic [9], allow specification of systems where order matters. The proof theory of subexponential in non-commutative settings is presented in [10].

In order to introduce subexponentials, one first fixes a *subexponential signature* of the form $\mathcal{S} = \langle \mathcal{I}, \preceq, \mathcal{W}, \mathcal{P}, \mathcal{C} \rangle$. Here $\mathcal{I}$ is the set of subexponential labels, that is, for each $s \in \mathcal{I}$ we introduce a unary connective $!^s$; $\preceq$ is a preorder on $\mathcal{I}$. Finally, $\mathcal{W}$, $\mathcal{P}$, and $\mathcal{C}$ are subsets of $\mathcal{I}$, closed upwards under $\preceq$ (if $s \in \mathcal{W}$ and $s \preceq s'$, then $s' \in \mathcal{W}$, and the same for $\mathcal{P}$ and $\mathcal{C}$). If s belongs to $\mathcal{W}$, $\mathcal{P}$, or $\mathcal{C}$, then $!^s$ allows weakening, permutation, or non-local contraction, respectively. Since non-local contraction and weakening (see below) yield permutation, we require $\mathcal{W} \cap \mathcal{C} \subseteq \mathcal{P}$.

The rules for subexponentials are as follows:

$$\frac{\Gamma, F, \Delta \to C}{\Gamma, !^s F, \Delta \to C}\ !_{\mathrm{L}} \qquad \frac{!^{s_1} F_1, \ldots, !^{s_n} F_n \to F}{!^{s_1} F_1, \ldots, !^{s_n} F_n \to !^s F}\ !_{\mathrm{R}},\ s_i \succeq s$$

$$\frac{\Gamma, \Delta \to C}{\Gamma, !^{\mathsf{w}} F, \Delta \to C}\ !_{\mathrm{Weak}},\ \mathsf{w} \in \mathcal{W}$$

$$\frac{\Gamma, G, !^{\mathsf{p}} F, \Delta \to C}{\Gamma, !^{\mathsf{p}} F, G, \Delta \to C}\ !_{\mathrm{Perm}},\ \mathsf{p} \in \mathcal{P} \qquad \frac{\Gamma, !^{\mathsf{p}} F, G, \Delta \to C}{\Gamma, G, !^{\mathsf{p}} F, \Delta \to C}\ !_{\mathrm{Perm}},\ \mathsf{p} \in \mathcal{P}$$

$$\frac{\Gamma, !^{\mathsf{c}} F, \Phi, !^{\mathsf{c}} F, \Delta \to C}{\Gamma, !^{\mathsf{c}} F, \Phi, \Delta \to C}\ !_{\mathrm{NContr}},\ \mathsf{c} \in \mathcal{C} \qquad \frac{\Gamma, !^{\mathsf{c}} F, \Phi, !^{\mathsf{c}} F, \Delta \to C}{\Gamma, \Phi, !^{\mathsf{c}} F, \Delta \to C}\ !_{\mathrm{NContr}},\ \mathsf{c} \in \mathcal{C}$$

On the other hand, we have the Kleene star. Extensions of the multiplicative-additive Lambek calculus with the Kleene star are variants of *action logic.* Action logic with inductive-style axiomatization for Kleene star is due to Pratt [27] and Kozen [12]. In this paper, however, we consider a stronger system, *infinitary action logic* [3], where the Kleene star is axiomatized by an ω-rule:

$$\frac{\Gamma, \Delta \to C \quad \Gamma, F, \Delta \to C \quad \Gamma, F, F, \Delta \to C \quad \Gamma, F, F, F, \Delta \to C \quad \ldots}{\Gamma, F^*, \Delta \to C}\ {*}_{\mathrm{L}}$$

Dually, the Kleene star on the left is introduced by one of the following rules:

$$\frac{\Pi_1 \to F \quad \ldots \quad \Pi_n \to F}{\Pi_1, \ldots, \Pi_n \to F^*}\ {*}_{\mathrm{R}}\ (n \geq 0) \qquad \text{(in particular, } \to F^* \text{ is an axiom).}$$

As a matter of notation, we always suppose that unary operations have higher priority, then binary ones.

The system with both Kleene star and subexponentials, denoted by $!_{\mathcal{S}}\mathbf{ACT}_\omega$, however, appears to be of very high complexity. Namely, as shown in [16] using results of Kozen [13], if $\mathcal{C} \neq \varnothing$, then $!_{\mathcal{S}}\mathbf{ACT}_\omega$ is Π^1_1-complete. In order to overcome this issue, we disallow contraction for our subexponentials, thus, require $\mathcal{C} = \varnothing$. This yields a subsystem denoted by $!^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\omega$, which is going to be considered throughout the paper.

We shall show that $!^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\omega$ has the same complexity as $\mathbf{ACT}_\omega$. This problem is Π^0_1-complete. Π^0_1-hardness, dually to Σ^0_1-hardness, deals with non-halting, or infinite computations. The Π^0_1-hardness proof for $\mathbf{ACT}_\omega$ by Buszkowski [2] encodes the non-halting problem for Turing machines, but in an indirect way, via the totality problem for context-free grammars. In contrast, our subexponential extension of $\mathbf{ACT}_\omega$, the system $!^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\omega$, allows modelling infinite ("surviving") computations in a natural and straightforward way.

The rest of this paper is organized as follows.

In Section 2 we establish several proof-theoretic properties of $!^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\omega$. Most notably, we prove the *-elimination theorem, extending the corresponding result of Palka [21] to include subexponentials. This yields the upper Π^0_1 complexity bound for $!^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\omega$.

In Section 3 we show how to model infinite computations ("survival" of a computational system) in $!^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\omega$. Our main example is the infinite run of a deterministic Turing machine.

In Section 4, we present a fragment of $!^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\omega$ with circular proofs. We show that derivations in this fragment model circular (looping) runs of Turing machines. However, some non-looping infinite runs are also captured by this circular fragment.

In Section 5, we present a refined encoding of Turing machine infinite computations, which uses only $/$, $\cdot$, *, and two subexponentials: $!^{\mathsf{w}}$, $!^{\mathsf{p}}$, where $\mathsf{w} \in \mathcal{W}$, $\mathsf{p} \in \mathcal{P}$, $\mathsf{p} \notin \mathcal{W}$. This is a new result: Buszkowski's construction [2] used to establish Π_1^0-hardness of $\mathbf{ACT}_\omega$ essentially uses, besides * and $/$, at least one of the two additive operations ($\vee$ or $\wedge$). A more recent Π_1^0-hardness proof [14] for the multiplicative-only non-commutative linear logic (the Lambek calculus) with Kleene star works without $\vee$ and $\wedge$, but the tradeoff is that now we have to use both divisions, $\backslash$ and $/$, and the product, $\cdot$. In our result, the tradeoff is that we have to use two subexponentials and the product. Complexity of the minimalist fragment of $\mathbf{ACT}_\omega$ itself, with only *, $/$, and maybe $\cdot$, remains an open question.

Section 6 is for concluding remarks.

The novelty and difference of this paper from other recent author's papers are as follows.

- In [14], we consider a system without subexponentials and additive connectives, only $/$, $\backslash$, $\cdot$, and *, and establish its Π_1^0-hardness. This result is obtained by encoding the totality problem for context-free languages. Here we present direct encodings of Turing computations and manage to prove Π_1^0-hardness with only one division.

- In [16], together with the Kleene star, we have a subexponential which allows contraction. This yields to a much higher complexity, Π_1^1.

- In [15], we also consider a full-power exponential which allows all structural rules, including contraction. However, the use of exponential is restricted to formulae of depth 1, which allows (using an essentially different method than the one presented here) to obtain the upper Π_1^0 bound.

- Finally, the systems in [9, 10] include only subexponentials, no Kleene star, and are used for modelling finite computations.

2 Proof-Theoretic Properties of $!^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\omega$

In this section we quickly discuss proof-theoretic properties of $!^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\omega$. The first one is cut elimination. We start with eliminating one cut rule:

Theorem 1. *If* $\Pi \to F$ *and* $\Gamma, F, \Delta \to C$ *are derivable in* $!^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\omega$ *without using cut, then so is* $\Gamma, \Pi, \Delta \to C$.

This theorem follows from a more general result of [16, Thm. 4.1]; however, in the absence of contraction the proof becomes much simpler and actually follows the line of Palka's proof [21]. The proof goes by nested transfinite induction on the following parameters:

- the complexity (length) of the formula being cut;
- the rank, or the ordinal representing the height, of the derivation of $\Pi \to F$;
- the rank of the derivation of $\Gamma, F, \Delta \to C$.

Corollary 1. *Any sequent provable in* $\mathbf{!}^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\omega$ *can be proved without cut.*

This corollary is proved, again, by transfinite induction on the rank.

Cut elimination yields invertibility of some rules, most notably the ω-rule, $*_{\mathrm{L}}$, and the polarized subformula property. Namely, each formula in a cut-free proof is a subformula of the goal sequent, with the same polarity.

Now, aiming to establish complexity bounds for $\mathbf{!}^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\omega$, we prove *-elimination. Following Palka [21], we define the *n-th approximation* of a sequent. Informally, we replace each negative occurrence of F^* with $\mathbf{1} \vee F \vee F^2 \vee \ldots \vee F^n$. The positive occurrences remain in their places, since they do not yield undecidability. This replacement has to be done in a nested way, and formally is captured by the following mutual definition of two transformations on formulae, P_n and N_n (corresponding to positive and negative polarities):

$$\begin{array}{ll}
P_n(p_i) = p_i & N_n(p_i) = p_i \\
P_n(\mathbf{1}) = \mathbf{1} & N_n(\mathbf{1}) = \mathbf{1} \\
P_n(E \cdot F) = P_n(E) \cdot P_n(F) & N_n(E \cdot F) = N_n(E) \cdot N_n(F) \\
P_n(F \setminus G) = N_n(F) \setminus P_n(G) & N_n(F \setminus G) = P_n(F) \setminus N_n(G) \\
P_n(G \,/\, F) = P_n(G) \,/\, N_n(F) & N_n(G \,/\, F) = N_n(G) \,/\, P_n(F) \\
P_n(F \wedge G) = P_n(F) \wedge P_n(G) & N_n(F \wedge G) = N_n(F) \wedge N_n(G) \\
P_n(F \vee G) = P_n(F) \vee P_n(G) & N_n(F \vee G) = N_n(F) \vee N_n(G) \\
P_n(!^s F) = !^s P_n(F) & N_n(!^s F) = !^s N_n(F) \\
P_n(F^*) = \bigl(P_n(F)\bigr)^* & N_n(F^*) = \mathbf{1} \vee N_n(F) \vee \bigl(N_n(F)\bigr)^2 \vee \ldots \vee \bigl(N_n(F)\bigr)^n.
\end{array}$$

The n-th approximation of a sequent $F_1, \ldots, F_m \to G$ is defined as

$$N_n(F_1), \ldots, N_n(F_m) \to P_n(G).$$

Theorem 2. *A sequent is derivable in* $\boldsymbol{!}^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\omega$ *if and only if so are its n-th approximations for all $n = 0, 1, 2, \ldots$*

This theorem is called *-elimination, because in the n-th approximation there are no more *negative* occurrences of Kleene star (which need the scary ω-rule in a cut-free derivation).

Proof. The proof of *-elimination for $\boldsymbol{!}^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\omega$ resembles Palka's [21] one for $\mathbf{ACT}_\omega$ without subexponentials. The only principal difference is that in $\boldsymbol{!}^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\omega$ we have permutation rules, which do not reduce (going upwards) complexity of the sequent. However, they can be dealt with quite easily (see below). Notice that here it is crucial that $\mathcal{C} = \varnothing$: contraction ($!_{\mathrm{NContr}}$) *increases* complexity, which makes the *-elimination argument fail. And indeed, $\boldsymbol{!}_{\mathcal{S}}\mathbf{ACT}_\omega$ does not enjoy *-elimination due to complexity reasons.

We start with establishing (by structural inducton on F) derivability of the following sequents (here $k \le n$):

$$F \to P_n(F); \qquad N_n(F) \to F; \qquad P_n(F) \to P_k(F); \qquad N_k(F) \to N_n(F).$$

The first two immediately yield the easy "only if" implication: if $F_1, \ldots, F_m \to G$ is derivable, then $N_n(F_1), \ldots, N_n(F_m) \to P_n(G)$ is derivable using the cut rule with $N_n(F_1) \to F_1$, ..., $N_n(F_m) \to F_m$, and $G \to P_n(G)$.

For the "if" direction, define $\kappa(F)$ to be the complexity of F, counted as the total number of connectives (including subexponentials and Kleene star) and constants ($\mathbf{1}$) in F. Next, for a sequent $\Gamma \to C$ let $c_i(\Gamma \to C)$ be the number of subformula occurrences in $\Gamma \to C$ with $\kappa(F) = i$. Now the complexity of $\Gamma \to C$ is measured by an infinite vector $(c_0, c_1, c_2, \ldots)$, and, obviously, for some i_0 we have $c_i = 0$ if $i \ge i_0$. Thus, we can define a lexicographic well-ordering on such vectors: $(c_0, c_1, c_2, \ldots) \ll (c'_0, c'_1, c'_2, \ldots)$, if for some i_0 we have $c_{i_0} < c'_{i_0}$ and $c_i = c'_i$ if $i > i_0$. Let us call such vectors c-vectors.

We proceed by well-ordered (transfinite) induction on $\ll$ and prove the contraposition. Suppose $F_1, \ldots, F_m \to G$ is not derivable in $\boldsymbol{!}^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\omega$ and prove that so is $N_n(F_1), \ldots, N_n(F_m) \to P_n(G)$ for some n. Consider two cases.

Case 1. F_i, for some i, is of the form E^*. Then $F_1, \ldots, F_{i-1}, E^k, F_{i+1}, \ldots, F_m \to G$ is not derivable for some k: otherwise the original sequent could be derived by $*_{\mathrm{L}}$. This new sequent has a smaller (in the sense of $\ll$) c-vector than the original one. Thus, by induction $N_l(F_1), \ldots, N_l(F_{i-1}), (N_l(E))^k, N_l(F_{i+1}), \ldots, N_l(F_m) \to P_l(G)$ is not derivable for some l. Let $n = \max\{l, k\}$. The sequent

$$N_n(F_1), \ldots, N_n(F_{i-1}), (N_n(E))^k, N_n(F_{i+1}), \ldots, N_n(F_m) \to P_n(G)$$

is also not derivable: otherwise we could apply cut with $N_l(F_i) \to N_n(F_i)$ and $P_n(G) \to P_l(G)$. Finally, since $(N_n(E))^k \to N_n(E^*)$ is derivable, we get non-derivability of $N_n(F_1), \ldots, N_n(F_{i-1}), N_n(E^*), N_n(F_{i+1}), \ldots, N_n(F_m) \to P_n(G)$, which is the n-th approximation of our original sequent.

Case 2. No F_i is of the form E^*. Following Palka, we perform *one step of proof search* in order to apply induction. However, now we *do not count* $!_{\mathrm{Perm}}$ *as a step*, and proceed upwards to find a "real" rule application.

Let us non-deterministically construct a set $\mathcal{A}$ of sequents as follows. Given the original sequent $F_1, \ldots, F_m \to G$, apply $!_{\mathrm{Perm}}$ several times (which is possible if some F_j are of the form $!^{\mathsf{p}}E$, $p \in \mathcal{W}$), and then attempt to derive it using one of the other rules. The premises of this rule form the set $\mathcal{A}$. Notice that this rule cannot be $*_{\mathrm{L}}$ (otherwise we are in Case 1), but could be $*_{\mathrm{R}}$. Thus, $\mathcal{A}$ is always finite. Also, recall that we can restrict ourselves to cut-free derivations, so the last rule is not cut.

Let $\mathfrak{A}$ be the set of all sets $\mathcal{A}$ that could be generated by the non-deterministic procedure described above. Each $\mathcal{A} \in \mathfrak{A}$ includes a non-derivable sequent (otherwise we could derive the original one), and the c-vector of this sequent is smaller (in the sense of $\ll$) than the c-vector of the original sequent. Thus, we can apply induction hypothesis and conclude that, for some $k_{\mathcal{A}}$, the $k_{\mathcal{A}}$-th approximation of this sequent is also non-derivable.

The set $\mathfrak{A}$ itself is finite. Indeed, $!^{\mathsf{p}}$-formulae in the original sequent allow only a finite number of permutations, and for each of them there is a finite number of rules which could possibly be applied (thanks to cut-freeness). Let $n = \max\{k_{\mathcal{A}} \mid \mathcal{A} \in \mathfrak{A}\}$. By monotonicity of approximations, in each $\mathcal{A}$ there is a sequent whose n-th approximation is not derivable.

Now take the n-th approximation of the original sequent. Suppose it is derivable. In the bottom of the derivation, there could be a series of $!_{\mathrm{Perm}}$ applications (permutations of !-formulae), preceded by some other rule. The n-th approximation does not include negative occurrences of *, so this rule is not $*_{\mathrm{L}}$. Thus, premises of this rule are exactly n-th approximations of sequents from some $\mathcal{A} \in \mathfrak{A}$. However, in each $\mathcal{A}$ there exists a sequent whose n-th approximation is not derivable. Contradiction. $\square$

Corollary 2. *The derivability problem in* $!^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\omega$ *belongs to* Π^0_1.

Proof. Derivability of sequents without negative occurrences of * is decidable by exhaustive proof search. The external "$\forall n$" quantifier (over all n-th approximations) yields Π^0_1. $\square$

Actually, since $!^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\omega$ includes $\mathbf{ACT}_\omega$, it is also Π^0_1-hard [2].

3 Modelling Infinite Computations

In this section we show how $!^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\omega$ can be used as a framework for specifying infinite behaviour of computation systems. Our main—and universal—example is the specification of infinite run of a Turing machine on a given input.

We consider only single-tape, single-head Turing machines and allow the tape to grow only to the right. (If the machine reaches the left end of the tape and attempts to move further to the left, it terminates.) Turing machines considered here are allowed to be non-deterministic.

A Turing machine is denoted by $\mathfrak{M}$. Let Σ be its internal alphabet and Q be the set of states. We denote letters of $\Sigma = \{a_1, \ldots, a_n\}$ by a, b, c (maybe with subscripts) and elements of Q (states) by $q_0, q_1, \ldots$; q_0 is the initial state. We suppose that Σ and Q are disjoint. A special *blank* symbol $\llcorner\!\lrcorner \in \Sigma$ is added to the tape when $\mathfrak{M}$ reaches the right end and moves further to the right.

Configurations of Turing machines are encoded in a standard way. If the machine is in state q, observing $a_{i_1}, \ldots, a_{i_k}$ observing letter a_{i_j}, then its configuration is encoded by $a_{i_1} \ldots a_{i_{j-1}} q a_{i_j} \ldots a_{i_k}\$$. The special character $\$ \notin \Sigma \cup Q$ denotes the right end of the tape. A slightly different encoding is used when the machine starts with an empty word on the tape. In this situation we add a blank: $q\llcorner\!\lrcorner\$$, in order to have some symbol being "observed."

Next, we suppose that all elements of $\Sigma \cup Q \cup \{\$\}$ are variables of $!^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\omega$ and represent configurations of $\mathfrak{M}$ as elementary products: $a_{i_1} \cdot \ldots \cdot a_{i_{j-1}} \cdot q \cdot a_{i_j} \cdot \ldots \cdot a_{i_k} \cdot \$$.

Transition rules of $\mathfrak{M}$ are of the form $(q_1, a) \to (q_2, b, d)$, where $q_1, q_2 \in Q$, $a, b \in \Sigma$, and $d \in \{N, L, R\}$. This rule is applicable in the case when $\mathfrak{M}$ is in state q_1 observing a on the tape, and instructs it to replace a by b, change the state to q_2, and perform a movement according to d: if $d = L$, move one cell left, if $d = R$, move one cell right, if $d = N$, do not move.

We encode each transition rule by a Lambek formula (that is, a formula without subexponentials and Kleene star), according to the following table.

Rule	**Formulae**
$(q_1, a) \to (q_2, b, N)$	$(q_2 \cdot b) \,/ (q_1 \cdot a)$
$(q_1, a) \to (q_2, b, L)$	$(q_2 \cdot c \cdot b) \,/ (c \cdot q_1 \cdot a)$, for any $c \in \Sigma$
$(q_1, a) \to (q_2, b, R)$	$(b \cdot q_2 \cdot c) \,/ (q_1 \cdot a \cdot c)$, for any $c \in \Sigma$ $(b \cdot q_2 \cdot \llcorner\!\lrcorner \cdot \$) \,/ (q_1 \cdot a \cdot \$)$

By $E_{\mathfrak{M}}$ we denote the additive conjunction ($\wedge$) of the formulae mentioned in the table above for all rules of $\mathfrak{M}$: $E_{\mathfrak{M}} = A_1 \wedge \ldots \wedge A_M$, where M is the number of transition rules.

Lemma 1. *Let N be a natural number. The sequent*

$$(!^{\mathsf{p}} E_{\mathfrak{M}})^N, q_0, a_{i_1}, \ldots, a_{i_k}, \$ \to (a_1 \vee \ldots \vee a_n)^* \cdot (q_0 \vee q_1 \vee \ldots \vee q_m) \cdot (a_1 \vee \ldots \vee a_n)^+ \cdot \$$$

is derivable in $\boldsymbol{!}^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\omega$ if and only if $\mathfrak{M}$ can make N moves, starting from state q_0 observing $a_{i_1} \ldots a_{i_k}$ on its tape.

The sequent in this lemma has the following informal reading: rules of $\mathfrak{M}$ can be successfully applied N times, starting from configuration $q_0 a_{i_1} \ldots a_{i_k} \$$, yielding again a valid configuration. The fact that the configuration is valid is expressed by the regular expression on the right; here we use the standard shortcut $F^+ = F \cdot F^*$. The subexponential $!^{\mathsf{p}}$ is used for delivering commands of $\mathfrak{M}$ to appropriate places in the configuration.

Proof. The "if" direction is trivial: for each move of $\mathfrak{M}$, we take one formula from the instances of $!^{\mathsf{p}} E_{\mathfrak{M}}$, move it to the location of q (the state) using $!_{\mathrm{Perm}}$, decompose by $!_{\mathrm{L}}$ and $\wedge_{\mathrm{L}}$, and apply $/_{\mathrm{L}}$ followed by $\cdot_{\mathrm{L}}$ to execute the rule. In the end, we get a derivable sequent of the form

$$a'_{i_1}, \ldots, a'_{i_{j-1}}, q', a'_{i_j}, \ldots, a'_{i_{k'}}, \$ \to (a_1 \vee \ldots \vee a_n)^* \cdot (q_0 \vee q_1 \vee \ldots \vee q_m) \cdot (a_1 \vee \ldots \vee a_n)^+.$$

For the interesting "only if" direction, we start with some rearrangings the proof, which we call "disbalancing." The transformations are essentially the same as used by Pentus [22, 23] for his calculus **Lcut**, the only rule of which is cut. Here we apply these transformations to logical rules in a cut-free derivation.

Consider a cut-free derivation of the sequent in question. We claim that the derivation can be reorganized so that left premises of all applications of ${/}_{\mathrm{L}}$ are trivial, that is, of the form $p_1, \ldots, p_\ell \to p_1 \cdot \ldots \cdot p_\ell$, where p_i are variables.

Indeed, due to the polarized subformula property, the only rules which can be applied in the derivation of the left premise of ${/}_{\mathrm{L}}$ are the following ones: $!_{\mathrm{L}}$, $!_{\mathrm{Perm}}$, $\wedge_{\mathrm{L}}$, ${/}_{\mathrm{L}}$, $\cdot_{\mathrm{L}}$, $\cdot_{\mathrm{R}}$. (Recall that the $/$-formula introduced by this rule came from $E_{\mathfrak{M}}$, thus, it is of the form $(r_1 \cdot \ldots \cdot r_\mu) \,/\, (s_1 \cdot \ldots s_\nu)$, where r_i and s_j are variables.)

The $\cdot_{\mathrm{R}}$ rule is interchangable upwards will other rules from this list operating on the left. For example, this is how it gets exchanged with ${/}_{\mathrm{L}}$:

$$\cfrac{\cfrac{\Pi \to F \quad \Gamma, G, \Delta \to B}{\Gamma, G \,/\, F, \Pi, \Delta \to B}\ {/}_{\mathrm{L}} \qquad \Phi \to C}{\Gamma, G \,/\, F, \Pi, \Delta, \Phi \to B \cdot C}\ \cdot_{\mathrm{R}}$$

transforms into

$$\cfrac{\Pi \to F \qquad \cfrac{\Gamma, G \,/\, F, \Pi, \Delta \to B \quad \Phi \to C}{\Gamma, G, \Delta, \Phi \to B \cdot C}\ \cdot_{\mathrm{R}}}{\Gamma, G \,/\, F, \Pi, \Delta, \Phi \to B \cdot C}\ {/}_{\mathrm{L}}$$

Thus, we can suppose that, unless the left premise of ${/}_{\mathrm{L}}$ is not of the form $p_1, \ldots, p_\ell \to p'_1 \cdot \ldots \cdot p'_{\ell'}$, then the lowermost rule applied in its derivation is not $\cdot_{\mathrm{R}}$. In such situations, we apply *disbalancing* transformations:

$$\cfrac{\cfrac{\Phi \to H \quad \Pi_1, E, \Pi_2 \to F}{\Pi_1, E \,/\, H, \Phi, \Pi_2 \to F}\ {/}_{\mathrm{L}} \qquad \Gamma, G, \Delta \to C}{\Gamma, G \,/\, F, \Pi_1, E \,/\, H, \Phi, \Pi_2, \Delta \to C}\ {/}_{\mathrm{L}}$$

transforms into

$$\cfrac{\Phi \to H \qquad \cfrac{\Pi_1, E, \Pi_2 \to F \quad \Gamma, G, \Delta \to C}{\Gamma, G \,/\, F, \Pi_1, E, \Pi_2, \Delta \to C}\ {/}_{\mathrm{L}}}{\Gamma, G \,/\, F, \Pi_1, E \,/\, H, \Phi, \Pi_2, \Delta \to C}\ {/}_{\mathrm{L}}$$

and if $\mathscr{R}$ is one of $!_{\mathrm{L}}$, $!_{\mathrm{Perm}}$, $\wedge_{\mathrm{L}}$, or $\cdot_{\mathrm{L}}$, then the transformation is as follows:

$$\dfrac{\dfrac{\widetilde{\Pi} \to F}{\Pi \to F}\,\mathscr{R} \qquad \Gamma, G, \Delta \to C}{\Gamma, G \,/\, F, \Pi, \Delta \to C}\,/_{\mathrm{L}} \qquad \rightsquigarrow \qquad \dfrac{\dfrac{\widetilde{\Pi} \to F \quad \Gamma, G, \Delta \to C}{\Gamma, G \,/\, F, \widetilde{\Pi}, \Delta \to C}\,/_{\mathrm{L}}}{\Gamma, G \,/\, F, \Pi, \Delta \to C}\,\mathscr{R}$$

Notice that $p_1, \ldots, p_\ell \to p'_1, \ldots, p'_{\ell'}$ could only be derivable if $\ell = \ell'$ and $p'_i = p_i$ for each i.

In the "main branch" of the derivation, which goes always to the right at $/_{\mathrm{L}}$, we can also suppose that $\cdot_{\mathrm{R}}$, $*_{\mathrm{R}}$, and $\vee_{\mathrm{R}}$ we applied immediately after the axiom: otherwise, they can be shifted upwards.

Thus, our derivation is now just a sequence of applications of $!_{\mathrm{L}}$, $!_{\mathrm{Perm}}$, $\wedge_{\mathrm{L}}$, $/_{\mathrm{L}}$, and $\cdot_{\mathrm{L}}$, ending at a sequent of the form

$$a'_{i_1}, \ldots, a'_{i_{j-1}}, q', a'_{i_j}, \ldots, a'_{i_{k'}}, \$ \to (a_1 \vee \ldots \vee a_n)^* \cdot (q_0 \vee q_1 \vee \ldots \vee q_m) \cdot (a_1 \vee \ldots \vee a_n)^+.$$

Indeed, the right-hand side remains the same (all right rules shifted to the top), and the form of the left-hand side is guaranteed by the formulae in $E_{\mathfrak{M}}$: after applying $/_{\mathrm{L}}$ with such a formula, we replace one element of Q with another, thus maintaining the fact that there is exactly one such letter in the sequence; \$ is always the last one.

Each $/_{\mathrm{L}}$ application, with consequential applications $\cdot_{\mathrm{L}}$, which can be done immediately by invertibility of this rule, exactly corresponds to one step of a Turing machine computation. The total number is exactly N, since each of the $E_{\mathfrak{M}}$'s gives one /-formula after its decomposition.

Thus, derivability of our sequent yields possibility to perform N steps of $\mathfrak{M}$ on the given input. $\square$

This lemma immediately yields the following corollary:

Corollary 3. *If $\mathfrak{M}$ is deterministic, then it runs forever, starting from state q_0 and input word $a_{i_1} \ldots a_{i_k}$, if and only the sequent*

$$(!^{\mathsf{p}} E_{\mathfrak{M}})^*, q_0, a_{i_1}, \ldots, a_{i_k}, \$ \to (a_1 \vee \ldots \vee a_n)^* \cdot (q_0 \vee q_1 \vee \ldots \vee q_m) \cdot (a_1 \vee \ldots \vee a_n)^+ \cdot \$$$

is derivable in $!^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\omega$.

Indeed, a deterministic Turing machine can perform N moves for any N if and only if it runs infinitely (initial configuration being fixed).

Notice that the right-hand side of this sequent, $(a_1 \vee \ldots \vee a_n)^* \cdot (q_0 \vee q_1 \vee \ldots \vee q_m) \cdot (a_1 \vee \ldots \vee a_n)^+ \cdot \$$, is actually a *regular expression* which describes the language of all correct configurations of $\mathfrak{M}$. One can make it more restrictive, and thus express a more subtle specification of $\mathfrak{M}$'s infinite run. For example, one can impose restrictions like "in the infinite run, whenever $\mathfrak{M}$ is in state q_2, it should observe letter a_6," which is expressed by $\bigl((a_1 \vee \ldots \vee a_n)^* \cdot (q_0 \vee q_1 \vee q_3 \vee \ldots \vee q_m) \cdot (a_1 \vee \ldots \vee a_n)^+ \cdot \$\bigr) \vee \bigl((a_1 \vee \ldots \vee a_n)^* \cdot q_2 \cdot a_6 \cdot (a_1 \vee \ldots \vee a_n)^* \cdot \$\bigr)$. Notice that this restriction can only apply to an individual configuration. For such more restrictive specifications we get corresponding variants of Lemma 1 and Corollary 3:

Theorem 3. *Let $\mathfrak{M}$ be a deterministic Turing machine with $\Sigma = \{a_1, \ldots, a_n\}$ and $Q = \{q_0, q_1, \ldots, q_m\}$, and let $\mathscr{R}$ be a regular expression which generates a subset of the language generated by $(a_1 \vee \ldots \vee a_n)^* \cdot (q_0 \vee q_1 \vee \ldots \vee q_m) \cdot (a_1 \vee \ldots \vee a_n)^+ \cdot \$$. Consider $\mathscr{R}$ as a formula in the language of $\boldsymbol{!}^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\omega$. Then the sequent*

$$(!^{\mathsf{p}} E_{\mathfrak{M}})^*, q_0, a_{i_1}, \ldots, a_{i_k}, \$ \to \mathscr{R}$$

is derivable in $\boldsymbol{!}^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\omega$ if and only if $\mathfrak{M}$, starting from state q_0 and input word $a_{i_1} \ldots a_{i_k}$, runs forever and its configuration at each step satisfies $\mathscr{R}$.

Proof. Exactly as Lemma 1 and Corollary 3. □

Corollary 3, by the way, gives another proof of Π^0_1-hardness of $\boldsymbol{!}^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\omega$. However, Buszkowski's [2] result is stronger, since it establishes Π^0_1-hardness already in the fragment without subexponentials.

For non-deterministic machines the situation is a bit trickier.

Theorem 4. *For an arbitrary Turing machine $\mathfrak{M}$, the sequent*

$$(!^{\mathsf{p}} E_{\mathfrak{M}})^*, q_0, a_{i_1}, \ldots, a_{i_k}, \$ \to (a_1 \vee \ldots \vee a_n)^* \cdot (q_0 \vee q_1 \vee \ldots \vee q_m) \cdot (a_1 \vee \ldots \vee a_n)^+ \cdot \$$$

is derivable in $\boldsymbol{!}^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\omega$ if and only if there exists an infinite sequence of configurations of $\mathfrak{M}$ such that each configuration (except the first one) is a successor of the previous one.

In other words, derivability of the sequent is equivalent to the *existence* of an infinite execution trajectory of $\mathfrak{M}$, starting from the given initial configuration.

Proof. The "if" part is trivial. Existence of an infinite computation implies that for any concrete N machine $\mathfrak{M}$ can make N moves. By Lemma 1 this yields derivability of

$$(!^{\mathsf{p}} E_{\mathfrak{M}})^N, q_0, a_{i_1}, \ldots, a_{i_k}, \$ \to (a_1 \vee \ldots \vee a_n)^* \cdot (q_0 \vee q_1 \vee \ldots \vee q_m) \cdot (a_1 \vee \ldots \vee a_n)^+ \cdot \$.$$

The necessary sequent is now derived by the ω-rule $*_{\mathrm{L}}$.

The proof of the "only if" part involves Kőnig's lemma [11]. Consider all pairs of the form $(\mathfrak{c}, N)$, where N is a natural number and $\mathfrak{c}$ is a configuration of $\mathfrak{M}$. Let us draw an edge between pairs $(\mathfrak{c}, N)$ and $(\mathfrak{c}', N+1)$, if $\mathfrak{c}'$ is a successor of $\mathfrak{c}$. This yields a forest.

From this forest, let us pick up the tree rooted by $(\mathfrak{c}_0, 0)$, where $\mathfrak{c}_0$ is the initial configuration. Since any configuration $\mathfrak{c}$ has only finitely many successors, this tree is finitely branching.

On the other hand, for any N there exists a vertex $(\mathfrak{c}, N)$ inside this tree. Indeed, by cut with $(!^{\mathsf{p}} E_{\mathfrak{M}})^N \to (!^{\mathsf{p}} E_{\mathfrak{M}})^*$ we get derivability of

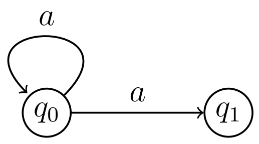
$$(!^{\mathsf{p}} E_{\mathfrak{M}})^N, q_0, a_{i_1}, \ldots, a_{i_k}, \$ \to (a_1 \vee \ldots \vee a_n)^* \cdot (q_0 \vee q_1 \vee \ldots \vee q_m) \cdot (a_1 \vee \ldots \vee a_n)^+ \cdot \$.$$

By Lemma 1 this implies that $\mathfrak{M}$ can make N moves, starting from $\mathfrak{c}_0$. In our tree, this is a path of the form $(\mathfrak{c}_0, 0) \to (\mathfrak{c}_1, 1) \to \ldots \to (\mathfrak{c}_N, N)$.

Thus, our tree is infinite. Being finitely branching, by Kőnig's lemma it should include an infinite path starting from the root: $(\mathfrak{c}_0, 0) \to (\mathfrak{c}_1, 1) \to (\mathfrak{c}_2, 2) \to \ldots$ This path corresponds to an infinite execution trajectory of $\mathfrak{M}$, starting from the given initial configuration $\mathfrak{c}_0$. $\square$

Unfortunately, this argument based on Kőnig's lemma happens to be not robust, that is, fails when we try to make our specification more restrictive. Our specification for infinite computation is encoded in the right-hand of the sequent, as a regular expression, $(a_1 \vee \ldots \vee a_n)^* \cdot (q_0 \vee q_1 \vee \ldots \vee q_m) \cdot (a_1 \vee \ldots \vee a_n)^+$, and merely states that after N steps of the computation the machine comes (in the non-deterministic setting: can come) to a correctly formed configuration. The following trivial example shows what happens if we impose a more restrictive specification.

Example 1. Consider a non-deterministic finite automaton $\mathfrak{A}$ with two states, q_0 and q_1, whose transitions are as follows:



The labels ("a") on the transition edges are considered as *output* of the automaton. Thus, its configuration, after N moves, can be represented as $a^N q_0$ or $a^N q_1$. The initial configuration is q_0. (Here we omit the sentinel \$, since q_i is always the rightmost symbol of the configuration. The blank symbol $\textvisiblespace$, after q_i, is also not used.)

Following our encoding, the E formula for $\mathfrak{A}$ is $E_{\mathfrak{A}} = ((a \cdot q_0) \,/\, q_0) \wedge ((a \cdot q_1) \,/\, q_0)$. Now let us consider the following specification formula (regular expression) in the right-hand side of the sequent: $q_0 \vee (a^+ \cdot q_1)$. The meaning is as follows: $\mathfrak{A}$ should be in state q_1, except for the initial configuration, where it is in state q_0.

It is of course impossible to maintain this specification for an infinite execution of $\mathfrak{A}$: once we reach q_1, no further move is possible. The sequent $(!^{\mathsf{p}} E_{\mathfrak{A}})^*, q_0 \to q_0 \vee (a^+ \cdot q_1)$, however, is derivable in $\mathbf{!}^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\omega$. Indeed, for $N > 0$ one can derive $!^{\mathsf{p}}((a \cdot q_1) \,/\, q_0), (!^{\mathsf{p}}((a \cdot q_0) \,/\, q_0))^{N-1}, q_0 \to a^n \cdot q_0$, thus $(!^{\mathsf{p}} E_{\mathfrak{A}})^N, q_0 \to a^+ \cdot q_1$. For $N = 0$ we have just $q_0 \to q_0$.

Non-determinism of $\mathfrak{A}$ is crucial. The automaton, for any N, can perform a computation which finishes at a configuration which obeys our specification, but these "partial" computations do not form an infinite one.

In contrast, in Theorem 4 the infinite computation was obtained from "partial" ones by Kőnig's lemma. The reason why it became possible is as follows: the specification $(a_1 \vee \ldots \vee a_n)^* \cdot (q_0 \vee q_1 \vee \ldots \vee q_m) \cdot (a_1 \vee \ldots \vee a_n)^+$ is automatically maintained not only for the last step of the computation, but also for all intermediate steps. Thus, a variant of Theorem 4 will hold for a special class of regular specifications:

Theorem 5. *Let $\mathfrak{M}$ be a non-deterministic Turing machine and let $\mathscr{R}$ be a regular expression specification (as in Theorem 3), with the following additional property: for any run of $\mathfrak{M}$, if one configuration satisfies $\mathscr{R}$, then so do all the previous ones. Then $(!^{\mathsf{p}} E_{\mathfrak{M}})^*, q_0, a_{i_1}, \ldots, a_{i_k}, \$ \to \mathscr{R}$ is derivable in $\mathscr{R}$ if and only if there exists an infinite run of $\mathfrak{M}$ starting from state q_0 and input word $a_{i_1} \ldots a_{i_k}$.*

Proof. Lemma 1 holds for an arbitrary $\mathscr{R}$. In the proof of Theorem 4, we now consider only vertices of the form $(\mathfrak{c}, N)$, where $\mathfrak{c}$ satisfies specification $\mathscr{R}$. For each N, Lemma 1 yields a sequence (path) of configurations

$(\mathfrak{c}_0, 0) \to (\mathfrak{c}_1, 1) \to \ldots \to (\mathfrak{c}_N, N)$ and guarantees that the last configuration, $\mathfrak{c}_N$, satisfies $\mathscr{R}$. On the other hand, our condition on $\mathscr{R}$ guarantees that so do all $\mathfrak{c}_i$'s, for $i = 0, 1, \ldots, N-1, N$. We finalize the proof by Kőnig's lemma, which yields an infinite path in which all configurations satisfy $\mathscr{R}$. $\square$

Though in this section we discussed only Turing machine computations, both deterministic and non-deterministic, the framework based on $!^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\omega$ is capable of specifying infinite behaviour in a broader context. Let us give just one simple example.

Example 2. Suppose we have several letter "a" in the memory, the state is always q (thus, the configuration is $qa \ldots a\$$), but now we have two controlling actors. One does nothing (just retains the q), thus, its rule can be formalized as $q \,/\, q$. The second one is evil, it removes one letter a at each step: $q \,/ (q \cdot a)$.

The following specification requires both of them to perform arbitrary many moves: $(q \,/\, q)^*, (q \,/ (q \cdot a))^*$ (compare with $((q \,/\, q) \wedge (q \,/ (q \cdot a)))^*$, which requires only the total number of moves to be arbitrarily large). Thus, the sequent $(!^{\mathsf{p}}(q \,/\, q))^*, (!^{\mathsf{p}}(q \,/ (q \cdot a)))^*, q, a, \ldots, a, \$ \to q \cdot a^* \cdot \$$ is not derivable, since the evil actor will eventually remove all a's.

Notice that here the process is non-deterministic (we do not impose any order of operations), so the equivalence between derivability and surviving of computation is proved via Kőnig's lemma, as in Theorem 4.

Finally, the term "evil" for the second actor here is not exactly appropriate: the game is actually cooperative, and we seek for a joint "winning strategy" for both players. For example, the following sequent is derivable: $(!^{\mathsf{p}}((q \cdot a) \,/\, q))^*, (!^{\mathsf{p}}(q \,/ (q \cdot a)))^*, q, a, \$ \to q \cdot (a \vee \mathbf{1}) \cdot \$$. The meaning of it is as follows: two actors can interleave their moves, when one removes the a and the other puts it back. The configuration after each step is either $qa\$$ or $q\$$.

4 Circularity in Proofs and Computations

In this section, we consider a fragment of $!^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\omega$ which is defined using circular proofs, and show the relations between derivations in this fragment and circular behaviour of infinite Turing computations.

We start with a reformulation of $!^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\omega$ as a system with non-well-founded proofs, which we denote by $!^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\infty$. This calculus forms a natural subexponential extension of the corresponding non-well-founded system for infinitary action logic by Das and Pous [4]. In this system, the rules for

Kleene star are formulated as follows:

$$\frac{\Gamma, \Delta \to C \quad \Gamma, E, E^*, \Delta \to C}{\Gamma, E^*, \Delta \to C}\ {*}'_{\mathrm{L}} \qquad \frac{}{\to E^*}\ {*}_{\mathrm{R1}} \qquad \frac{\Gamma \to E \quad \Delta \to E^*}{\Gamma, \Delta \to E^*}\ {*}_{\mathrm{R2}}$$

All other rules are the same as in $\mathbf{!}^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\omega$. Notice that now all rules are finitary; however, in $\mathbf{!}^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\infty$ derivations are allowed to have infinite branches, that is, be non-well-founded. In order to avoid vicious circles (say, an infinite branch consisting only of applications of $!_{\mathrm{Perm}}$), the following *progressivity condition* is imposed: for each infinite branch, there exists an occurrence of a formula of the form E^* which inherits upwards the branch and undergoes ${*}'_{\mathrm{L}}$ infinitely often.

Theorem 6. *A sequent has a correct non-well-founded proof in* $\mathbf{!}^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\infty$ *if and only if it is derivable in* $\mathbf{!}^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\omega$*.*

Proof. We give only a sketch of proof. For the easy "if" part, it is sufficient to show that the ${*}_{\mathrm{L}}$ and ${*}_{\mathrm{R}}$ rules of $\mathbf{!}^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\omega$ are derivable in $\mathbf{!}^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\infty$ (for ${*}_{\mathrm{L}}$, which is the ω-rule, we get an infinite branch obeying the progressivity condition). For the "only if" part, we first perform cut elimination, which is done as in [4, Thm. 15]: subexponentials without contraction do not alter the cut elimination procedure. Next, for a fixed n, we replace each sequent in the cut-free proof with its n-th approximation. The result can be transformed into a valid derivation. The only interesting case is ${*}'_{\mathrm{L}}$, which translates into $\vee_{\mathrm{L}}$, reducing n, or just disappears, if $n = 0$. Thus, in this derivation all infinite branches get cut off. Next, one simulates ${*}_{\mathrm{R1}}$ and ${*}_{\mathrm{R2}}$ in $\mathbf{!}^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\omega$. This gives a derivation for each n-th approximation. Applying Theorem 2 yields derivability of the original sequent in $\mathbf{!}^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\omega$. □

A non-well-founded proof is called *regular,* if it includes only a finite number of non-isomorphic subtrees. In an equivalent notation, one can locate each occurrence of a subtree root, which already appeared (as an isomorphic copy) lower in the proof tree, and replace the subderivation above it with a *backtrack.* This yields a finite object which represents a regular proof and is called a *circular* proof. The fragment of $\mathbf{!}^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\infty$ which allows only circular (regular) proofs is denoted by $\mathbf{!}^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_{\mathrm{circ}}$.

As noticed in [4], unlike $\mathbf{!}^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\infty$, the circular fragment $\mathbf{!}^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_{\mathrm{circ}}$ does not enjoy cut elimination: the counter-example is $A, A^* \to A^* \cdot A$.

Let us consider the circular fragment $\mathbf{!}^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_{\mathrm{circ}}$ from the point of view of Corollary 3, that is, from the point of view of modelling Turing machine behaviour. We shall consider the same encoding, *i.e.,* the sequent

$$(!^{\mathsf{p}} E_{\mathfrak{M}})^*, q_0, a_{i_1}, \ldots, a_{i_k}, \$ \to (a_1 \vee \ldots \vee a_n)^* \cdot (q_0 \vee q_1 \vee \ldots \vee q_m) \cdot (a_1 \vee \ldots \vee a_n)^+ \cdot \$,$$

which we shall denote by $(*)$, expresses the statement "$\mathfrak{M}$ runs forever on input word $a_{i_1} \ldots a_{i_k}$". (For simplicity, here we consider only deterministic machines.)

Since any sequent provable in $!^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_{\mathrm{circ}}$ is also provable in $!^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_{\omega}$, derivability of $(*)$ implies an infinite run of $\mathfrak{M}$ on $a_{i_1} \ldots a_{i_k}$. The converse does not hold, since $!^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_{\mathrm{circ}}$ has a recursively enumerable (Σ^0_1) set of theorems.

Quite naturally, derivations in the circular fragment can represent circular infinite runs of $\mathfrak{M}$:

Theorem 7. *Let $\mathfrak{M}$ be a deterministic Turing machine and have a circular run on input word $a_{i_1} \ldots a_{i_k}$, that is, starting from configuration $\mathfrak{c}_0 = q_0, a_{i_1} \ldots a_{i_k}$, it returns to some configuration $\mathfrak{c}'$ in which it already was before. Then the sequent $(*)$ is derivable in $!^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_{\mathrm{circ}}$.*

Proof. The proof is performed by a slight modification of the argument of Lemma 1. At each step of the infinite run, from configuration $\mathfrak{c}$ to configuration $\mathfrak{c}'$, we do the following (here and further we write just E for $E_{\mathfrak{M}}$):

$$\dfrac{\mathfrak{c}, \$ \to \mathscr{R} \qquad \dfrac{\ldots \quad (!^{\mathsf{p}}E)^*, \mathfrak{c}', \$ \to \mathscr{R}}{!^{\mathsf{p}}E, (!^{\mathsf{p}}E)^*, \mathfrak{c}, \$ \to \mathscr{R}}\ !_{\mathrm{Perm}}, \wedge_{\mathrm{L}}, /_{\mathrm{L}}}{(!^{\mathsf{p}}E)^*, \mathfrak{c}, \$ \to \mathscr{R}}$$

Here $\mathscr{R} = (a_1 \vee \ldots \vee a_n)^* \cdot (q_0 \vee q_1 \vee \ldots \vee q_m) \cdot (a_1 \vee \ldots \vee a_n)^+ \cdot \$$ and $\mathfrak{c}, \$ \to \mathscr{R}$ is derivable since $\mathfrak{c}$ is a valid configuration code. The left premise of $/_{\mathrm{L}}$ is an axiom-like sequent like $q_1, a \to q_1 \cdot a$, which validates applicability of a concrete transition rule taken from E by $\wedge_{\mathrm{L}}$. Finally, the progressivity condition is maintained by the fact that we decompose the same $(!^{\mathsf{p}}E)^*$ each time.

Now, if $\mathfrak{M}$ runs into a loop, the same sequent $(!^{\mathsf{p}}E)^*, \mathfrak{c}', \$ \to \mathscr{R}$ appears in the infinite derivation branch twice, and we replace the rest of the derivation with a backtrack. $\square$

In the non-deterministic case, this theorem also holds. However, in this case the original run could have been non-circular: after returning to the same configuration $\mathfrak{c}'$, the machine is not obliged to follow the old trajectory. When the derivation gets truncated via the backtrack, however, the corresponding run gets circularized.

Interestingly enough, however, there exist also non-circular runs for which $(*)$ is provable in $!^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_{\mathrm{circ}}$.

Example 3. Consider a very simple Turing machine which just infinitely populates its tape with copies of one letter a. Namely, let $\Sigma = \{a\}$, $Q = \{q\}$, $q_0 = q$, and the only transition rule be $(q, \textvisiblespace) \to (q, a, R)$. The infinite run of $\mathfrak{M}$, starting with an empty word (more precisely, $\textvisiblespace$), is definitely non-circular: the words on the tape are of the form $a^n\textvisiblespace$, and they are all different.

However, for this Turing machine the sequent $(*)$ has a circular proof! The E formula here is very simple: $E_{\mathfrak{M}} = (a \cdot q \cdot \textvisiblespace \cdot \$) \,/ (q \cdot \textvisiblespace \cdot \$)$. In order to construct the desired circular derivation, we first show that $(!^{\mathsf{p}}E)^*$ can be exchanged to the right with a (Figure 1(a)) and then prove $(*)$ itself, as shown on Figure 1(b). (The regular expression on the right-hand side here is more restrictive, than the original $\mathscr{R}$. The sequent $(*)$ itself is obtained by cut with $a^* \cdot q \cdot \textvisiblespace \cdot \$ \to (a \vee \textvisiblespace)^* \cdot q \cdot (a \vee \textvisiblespace)^+ \cdot \$$.)

Notice that, unlike the natural encoding in Theorem 7, the derivation on Figure 1 is quite sophisticated and essentially uses cut. Possibly, a version of $!^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_{\mathrm{circ}}$ without cut (which is weaker than $!^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_{\mathrm{circ}}$ itself) would capture exactly circular infinite executions. However, systems with inadmissible cut are weird from the logical point of view.

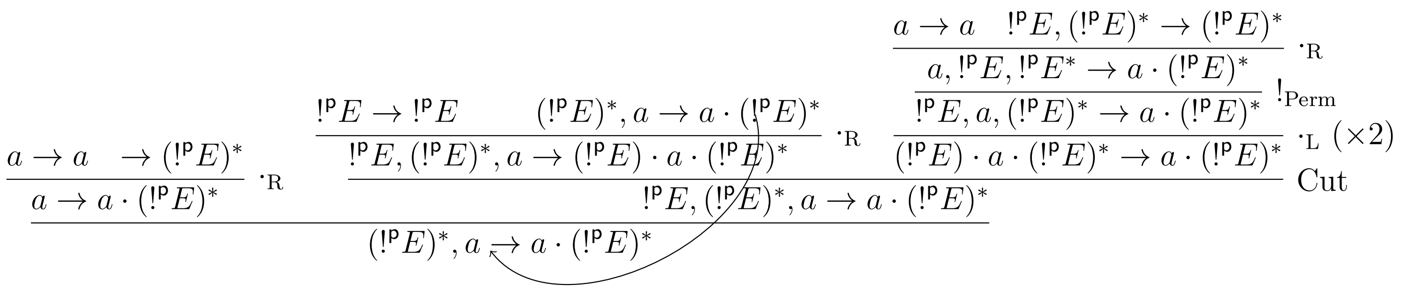
5 Undecidability with One Division

In this section we prove that $!^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_{\omega}$ is Π^0_1-hard in the smaller fragment, with only $/$, $\cdot$, *, and two subexponentials left in the language. We start with a well-known fact from the theory of regular expressions (Kleene algebra).

Lemma 2. *For any formula of the form $(B_1 \vee \ldots \vee B_n)^*$ there exists an equivalent (in $!^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_{\omega}$) formula which uses only $\cdot$ and *.*

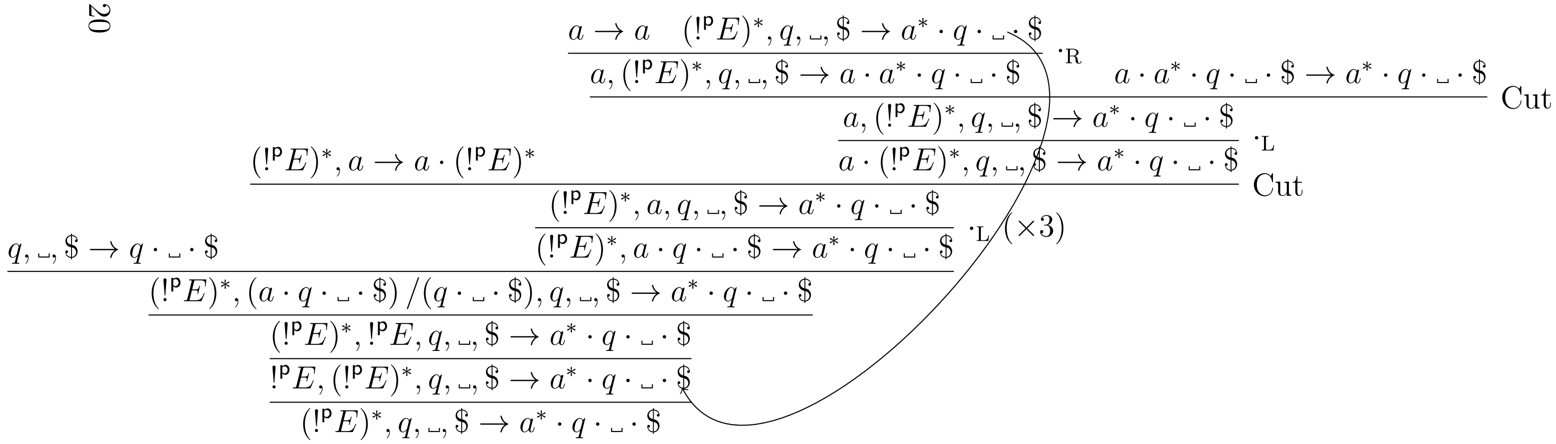
Proof. Induction on n. If $n = 1$, then we have just B_1^*, which is already in the desired language. Now let $B_1 \vee \ldots \vee B_{n-1} = C$ and $B_n = D$. Recall that $(C \vee D)^* = (C^* \cdot D)^* \cdot C^*$ is generally true in Kleene algebra, and therefore derivable in $!^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_{\omega}$. By induction hypothesis, $C^* = (B_1 \vee \ldots \vee B_{n-1})^*$ is equivalent to a formula C' in the language of $\cdot$ and *. Then $(B_1 \vee \ldots \vee B_n)^* = (C \vee D)^*$ is equivalent to $(C' \cdot D)^* \cdot C'$. $\square$

(a)



$a \to a \quad \to (!^{\mathsf{p}}E)^*$
$a \to a \cdot (!^{\mathsf{p}}E)^*$
$\cdot_{\mathrm{R}}$
$!^{\mathsf{p}}E \to !^{\mathsf{p}}E \qquad (!^{\mathsf{p}}E)^*, a \to a \cdot (!^{\mathsf{p}}E)^*$
$!^{\mathsf{p}}E, (!^{\mathsf{p}}E)^*, a \to (!^{\mathsf{p}}E) \cdot a \cdot (!^{\mathsf{p}}E)^*$
$\cdot_{\mathrm{R}}$
$a \to a \quad !^{\mathsf{p}}E, (!^{\mathsf{p}}E)^* \to (!^{\mathsf{p}}E)^*$
$a, !^{\mathsf{p}}E, !^{\mathsf{p}}E^* \to a \cdot (!^{\mathsf{p}}E)^*$
$\cdot_{\mathrm{R}}$
$!^{\mathsf{p}}E, a, (!^{\mathsf{p}}E)^* \to a \cdot (!^{\mathsf{p}}E)^*$
$!_{\mathrm{Perm}}$
$(!^{\mathsf{p}}E) \cdot a \cdot (!^{\mathsf{p}}E)^* \to a \cdot (!^{\mathsf{p}}E)^*$
$\cdot_{\mathrm{L}}$ $(\times 2)$
Cut
$!^{\mathsf{p}}E, (!^{\mathsf{p}}E)^*, a \to a \cdot (!^{\mathsf{p}}E)^*$
$(!^{\mathsf{p}}E)^*, a \to a \cdot (!^{\mathsf{p}}E)^*$

(b)



$a \to a \quad (!^{\mathsf{p}}E)^*, q, \llcorner\!\lrcorner, \$ \to a^* \cdot q \cdot \llcorner\!\lrcorner \cdot \$$
$a, (!^{\mathsf{p}}E)^*, q, \llcorner\!\lrcorner, \$ \to a \cdot a^* \cdot q \cdot \llcorner\!\lrcorner \cdot \$$
$\cdot_{\mathrm{R}}$
$a \cdot a^* \cdot q \cdot \llcorner\!\lrcorner \cdot \$ \to a^* \cdot q \cdot \llcorner\!\lrcorner \cdot \$$
Cut
$a, (!^{\mathsf{p}}E)^*, q, \llcorner\!\lrcorner, \$ \to a^* \cdot q \cdot \llcorner\!\lrcorner \cdot \$$
$a \cdot (!^{\mathsf{p}}E)^*, q, \llcorner\!\lrcorner, \$ \to a^* \cdot q \cdot \llcorner\!\lrcorner \cdot \$$
$\cdot_{\mathrm{L}}$
$(!^{\mathsf{p}}E)^*, a \to a \cdot (!^{\mathsf{p}}E)^*$
Cut
$(!^{\mathsf{p}}E)^*, a, q, \llcorner\!\lrcorner, \$ \to a^* \cdot q \cdot \llcorner\!\lrcorner \cdot \$$
$(!^{\mathsf{p}}E)^*, a \cdot q \cdot \llcorner\!\lrcorner \cdot \$ \to a^* \cdot q \cdot \llcorner\!\lrcorner \cdot \$$
$\cdot_{\mathrm{L}}$ $(\times 3)$
$q, \llcorner\!\lrcorner, \$ \to q \cdot \llcorner\!\lrcorner \cdot \$$
$(!^{\mathsf{p}}E)^*, (a \cdot q \cdot \llcorner\!\lrcorner \cdot \$)\,/(q \cdot \llcorner\!\lrcorner \cdot \$), q, \llcorner\!\lrcorner, \$ \to a^* \cdot q \cdot \llcorner\!\lrcorner \cdot \$$
$(!^{\mathsf{p}}E)^*, !^{\mathsf{p}}E, q, \llcorner\!\lrcorner, \$ \to a^* \cdot q \cdot \llcorner\!\lrcorner \cdot \$$
$!^{\mathsf{p}}E, (!^{\mathsf{p}}E)^*, q, \llcorner\!\lrcorner, \$ \to a^* \cdot q \cdot \llcorner\!\lrcorner \cdot \$$
$(!^{\mathsf{p}}E)^*, q, \llcorner\!\lrcorner, \$ \to a^* \cdot q \cdot \llcorner\!\lrcorner \cdot \$$

Figure 1: Derivations for Example 3

Now let $\mathfrak{M}$ be a deterministic Turing machine. Recall that in Section 3 we encoded transition rules of $\mathfrak{M}$ as formulae $A_1, \ldots, A_M$ and combined these formulae using $\wedge$: $!^{\mathsf{p}} E_{\mathfrak{M}} = !^{\mathsf{p}}(A_1 \wedge \ldots \wedge A_M)$. Formulae A_i are built using only $\cdot$ and $/$. The $\wedge$ connective here, according to the $\wedge_{\mathrm{L}}$ rule, implements the choice of one transition rule out of M possible at each computation step.

It happens that the same can be implemented using $\cdot$ and $!^{\mathsf{w}}$, where $\mathsf{w} \in \mathcal{W}$ (that is, allows weakening). Namely, let t be a new variable and define $\tilde{E}_{\mathfrak{M}} = !^{\mathsf{w}}(!^{\mathsf{p}} A_1 \,/\, t) \cdot \ldots \cdot !^{\mathsf{w}}(!^{\mathsf{p}} A_M \,/\, t) \cdot t$. Here t is the "tick counter." The use of t guarantees that exactly one of $!^{\mathsf{p}} A_i \,/\, t$ gets introduced by $!_{\mathrm{L}}$, and other $M-1$ ones get weakened. (Otherwise one could just weaken all these formulae immediately after decomposing *, and $\mathfrak{M}$ would not be forced to perform N steps, as desired.) An analog of Lemma 1 is now as follows:

Lemma 3. *Let N be a natural number. The sequent*

$$\tilde{E}_{\mathfrak{M}}^N, q_0, a_{i_1}, \ldots, a_{i_k}, \$ \to (a_1 \vee \ldots \vee a_n \vee q_0 \vee q_1 \vee \ldots \vee q_m \vee \$)^*,$$

is derivable in $\mathbf{!}^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_\omega$ if and only if $\mathfrak{M}$ can perform N computation steps, starting from state q_0 with input word $a_{i_1} \ldots a_{i_k}$.

Notice that here we have relaxed the regular expression on the right: now we accept any word in the language of $\Sigma \cup Q \cup \{\$\}$.

Proof. The "if" part is easy: we again directly encode each rule application, but now instead of using $\wedge_{\mathrm{L}}$ we apply $!_{\mathrm{Weak}}$ for all formulae in $\tilde{E}_{\mathfrak{M}}$, except for the necessary $!^{\mathsf{w}}(!^{\mathsf{p}} A_i \,/\, t)$. For this formula, we apply $!_{\mathrm{L}}$ removing $!^{\mathsf{w}}$, then apply $/_{\mathrm{L}}$ which "cancels" the t instances, and then use permutation to deliver the formula to the necessary place.

For the "only if" part, we disbalance the proof (as explained in the proof of Lemma 1). (Disbalancing works with $!_{\mathrm{Weak}}$ also.) Now we see that each application of $/_{\mathrm{L}}$ which decomposes $!^{\mathsf{p}} A_i \,/\, t$ has $t \to t$ as its left premise. Moreover, the number of t's should be balanced: this means that exactly N instances of $!^{\mathsf{w}}(!^{\mathsf{p}} A_i \,/\, t)$ are introduced by $!_{\mathrm{L}}$, while others get weakened. Now we are exactly in the setting of Lemma 1: we have N instances of $!^{\mathsf{p}} A_i$ (with different i's), each of which, in the disbalanced proof, implements one step of $\mathfrak{M}$. $\square$

Corollary 4. *Let $\mathfrak{M}$ be a deterministic Turing machine. The sequent*

$$\tilde{E}_{\mathfrak{M}}^*, q_0, a_{i_1}, \ldots, a_{i_k}, \$ \to (a_1 \vee \ldots \vee a_n \vee q_0 \vee q_1 \vee \ldots \vee q_m \vee \$)^*,$$

is derivable in $\mathbf{!}^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_{\omega}$ *if and only if* $\mathfrak{M}$ *runs forever, starting from state* q_0 *and input word* $a_{i_1} \ldots a_{i_k}$.

Now, using Lemma 2, we replace $(a_1 \vee \ldots \vee a_n \vee q_0 \vee q_1 \vee \ldots \vee q_m \vee \$)^*$ with a formula using $\cdot$ and *, and get the following theorem.

Theorem 8. *The derivability problem for fragment of* $\mathbf{!}^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_{\omega}$ *with only* $/$, $\cdot$, *, *and two subexponentials,* $!^{\mathsf{w}}$ *and* $!^{\mathsf{p}}$, *where* $\mathsf{w} \in \mathcal{W}$, $\mathsf{p} \in \mathcal{P}$, $\mathsf{p} \notin \mathcal{W}$, *is* Π^0_1*-complete.*

6 Conclusion and Future Work

In this paper, we have presented an extension of multiplicative-additive Lambek calculus (that is, multiplicative-additive non-commutative intuitionistic linear logic) with the Kleene star (as in infinitary action logic) and a family of subexponentials, for which we can allow permutation and/or weakening, but not contraction. We have shown that the complexity remains the same as of infinitary action logic itself: the system is Π^0_1-complete. (If we allow contraction, complexity rises up to Π^1_1.) The system presented here can be used as a framework for modelling non-terminating computations, dually to the usage of subexponential extensions of linear logic for modelling terminating computations. This works perfectly well for deterministic computations; for the non-deterministic case, some precautions are required. Finally, we prove a new Π^0_1-hardness result, in a fragment with only one division, product, Kleene star, and two subexponentials.

The following questions are left open for future research. First, the complexity of the minimalist fragment of $\mathbf{ACT}_{\omega}$ with only $/$ and * is yet unknown. A potentially easier question is to prove Π^0_1-hardness of the corresponding fragment of $\mathbf{!}^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_{\omega}$, with $/$, *, and also two subexponentials, $!^{\mathsf{w}}$ and $!^{\mathsf{p}}$. We hope that this can be done using the techniques of Section 5. The motivation for studying one-division fragments comes from the fact that in the purely multiplicative language such a fragment is polynomial-time decidable [28], while richer fragments are NP-complete [24, 29]. Second, it would be interesting to build a focused proof system for $\mathbf{!}^{\mathsf{wp}}_{\mathcal{S}}\mathbf{ACT}_{\omega}$ (like the one presented in [9], but with infinite proofs). The disbalancing technique used in the proof of Lemma 1 is actually an *ad hoc* attempt of proof reorganization, which should be done in a more uniform manner.

Acknowledgements The author is grateful to Anupam Das, Max Kanovich, Vivek Nigam, Daniyar Shamkanov, and Andre Scedrov for fruitful discussions.

References

[1] David Baelde and Dale Miller. Least and greatest fixed points in linear logic. In *LPAR 2007: Logic for Programming, Artificial Intelligence, and Reasoning*, volume 4790 of *Lecture Notes in Computer Science*, pages 92–106. Springer, 2007. URL: `https://doi.org/10.1007/978-3-540-75560-9_9`.

[2] Wojciech Buszkowski. On action logic: equational theories of action algebras. *Journal of Logic and Computation*, 17(1):199–217, 2007. URL: `https://doi.org/10.1093/logcom/exl036`.

[3] Wojciech Buszkowski and Ewa Palka. Infinitary action logic: complexity, models and grammars. *Studia Logica*, 89(1):1–18, 2008. URL: `https://doi.org/10.1007/s11225-008-9116-7`.

[4] Anupam Das and Damien Pous. Non-wellfounded proof theory for (Kleene+action) (algebras+lattices). In *27th EACSL Annual Conference on Computer Science Logic (CSL 2018)*, volume 119 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 19:1–19:18, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. URL: `http://drops.dagstuhl.de/opus/volltexte/2018/9686`, `doi:10.4230/LIPIcs.CSL.2018.19`.

[5] Guiseppe De Giacomo, Eugenia Ternovska, and Ray Reiter. Nonterminating processes in the situation calculus. *Annals of Mathematics and Artificial Intelligence*, 2019. Published online. URL: `https://doi.org/10.1007/s10472-019-09643-9`.

[6] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987. `doi:https://doi.org/10.1016/0304-3975(87)90045-4`.

[7] Walter Guttmann. Algebras for iteration and infinite computations. *Acta Informatica*, 49:343–359, 2012. URL: `https://doi.org/10.1007/s00236-012-0162-2`.

[8] Makoto Kanazawa. The Lambek calculus enriched with additional connectives. *Journal of Logic, Language and Information*, 1:141–171, 1992. URL: `https://doi.org/10.1007/BF00171695`.

[9] Max Kanovich, Stepan Kuznetsov, Vivek Nigam, and Andre Scedrov. A logical framework with commutative and non-commutative subexponentials. In *IJCAR 2018: Automated Reasoning*, volume 10900 of *Lecture Notes in Artificial Intelligence*, pages 228–245. Springer, 2018. URL: `https://link.springer.com/chapter/10.1007/978-3-319-94205-6_16`.

[10] Max Kanovich, Stepan Kuznetsov, Vivek Nigam, and Andre Scedrov. Subexponentials in non-commutative linear logic. *Mathematical Structures in Computer Science*, 29(8):1217–1249, 2019. URL: `https://www.cambridge.org/core/journals/mathematical-structures-in-computer-science/article/subexponentials-in-noncommutative-linear-logic/198AED235060784373045941E5A70241`.

[11] Dénes Kőnig. Über eine Schlussweise aus dem Endlichen ins Unendliche. *Acta Scientiarum Mathematicarum (Szeged)*, 3(2–3):121–130, 1927.

[12] Dexter Kozen. On action algebras. In J. van Eijck and A. Visser, editors, *Logic and Information Flow*, pages 78–88. MIT Press, 1994. URL: `https://www.cs.cornell.edu/~kozen/Papers/act.pdf`.

[13] Dexter Kozen. On the complexity of reasoning in Kleene algebra. *Information and Computation*, 179:152–162, 2002. URL: `https://www.sciencedirect.com/science/article/pii/S0890540101929608`.

[14] Stepan Kuznetsov. Complexity of the infinitary Lambek calculus with Kleene star. *Review of Symbolic Logic*, 2020. Published Online. URL: `https://www.cambridge.org/core/journals/review-of-symbolic-logic/article/complexity-of-the-infinitary-lambek-calculus-with-kleene-star/79EA7FE05F244185D5554FD5BAF3C072`, `arXiv:2005.00404`.

[15] Stepan Kuznetsov. A Π_1^0-bounded fragment of infinitary action logic with exponential, 2020. Submitted.

[16] Stepan L. Kuznetsov and Stanislav O. Speranski. Infinitary action logic with exponentiation, 2020. Submitted. `arXiv:2001.06863`.

[17] Vivek Nigam and Dale Miller. Algorithmic specifications in linear logic with subexponentials. In *11th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming (PPDP '09)*, pages 129–140. ACM, 2009. URL: `http://www.lix.polytechnique.fr/~dale/papers/ppdp09.pdf`.

[18] Vivek Nigam, Carlos Olarte, and Elaine Pimentel. A general proof system for modalities in concurrent constraint programming. In *CONCUR 2013 – Concurrency Theory*, volume 8052 of *Lecture Notes in Computer Science*, pages 410–424. Springer, 2013. URL: `https://link.springer.com/chapter/10.1007/978-3-642-40184-8_29`.

[19] Rémi Nollet, Alexis Saurin, and Christine Tasson. Local validity for circular proofs in linear logic with fixed points. In *27th EACSL Annual Conference on Computer Science Logic (CSL 2018)*, volume 119 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 35:1–35:23. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. URL: `https://drops.dagstuhl.de/opus/volltexte/2018/9702/pdf/LIPIcs-CSL-2018-35.pdf`.

[20] Carlos Olarte, Elaine Pimentel, and Vivek Nigam. Subexponential concurrent constraint programming. *Theoretical Computer Science*, 606:98–120, 2015. URL: `https://www.sciencedirect.com/science/article/pii/S0304397515005411`.

[21] Ewa Palka. An infinitary sequent system for the equational theory of *-continuous action lattices. *Fundamenta Informaticae*, 78(2):295–309, 2007. URL: `https://content.iospress.com/articles/fundamenta-informaticae/fi78-2-06`.

[22] Mati Pentus. Lambek grammars are context-free. In *8th Annual IEEE Symposium on Logic in Computer Science (LICS 1993)*, pages 429–433. IEEE Computer Society Press, 1993. URL: `https://ieeexplore.ieee.org/document/287565`.

[23] Mati Pentus. Product-free Lambek calculus and context-free grammars. *Journal of Symbolic Logic*, 62(2):648–660, 1997. URL: `http://lpcs.math.msu.su/~pentus/ftp/papers/prfree.pdf`.

[24] Mati Pentus. Lambek calculus is NP-complete. *Theoretical Computer Science*, 357(1):186–201, 2006. URL: `https://www.sciencedirect.com/science/article/pii/S0304397506002702`.

[25] Frank Pfenning and Robert J. Simmons. Substructural operational semantics as ordered logic programming. In *24th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2009)*, pages 101–110. IEEE, 2009. URL: `https://www.cs.cmu.edu/~fp/papers/lics09.pdf`.

[26] Jeff Polakow. Linear logic programming with an ordered context. In *PPDP 2000: Proceedings of the 2nd ACM SIGPLAN international conference on Principles and practice of declarative programming*, pages 68–79. ACM, 2000. URL: `https://dl.acm.org/doi/abs/10.1145/351268.351277`.

[27] Vaughan Pratt. Action logic and pure induction. In *JELIA 1990: Logics in AI*, volume 478 of *Lecture Notes in Artificial Intelligence*, pages 97–120. Springer, 1991. URL: `https://link.springer.com/chapter/10.1007/BFb0018436`.

[28] Yury Savateev. Unidirectional Lambek grammars in polynomial time. *Theory of Computing Systems*, 46(4):662–672, 2010. URL: `https://link.springer.com/article/10.1007/s00224-009-9208-4`.

[29] Yury Savateev. Product-free Lambek calculus is NP-complete. *Annals of Pure and Applied Logic*, 163(7):775–788, 2012. URL: `https://www.sciencedirect.com/science/article/pii/S0168007211001394`.

[30] Wolfgang Thomas and Helmut Lescow. Logical specifications of infinite computations. In *REX 1993: A Decade of Concurrency Reflections and Perspectives*, volume 803 of *Lecture Notes in Computer Science*, pages 583–621. Springer, 1994. URL: `https://link.springer.com/chapter/10.1007/3-540-58043-3_29`.

[31] Kevin Watkins, Iliano Cervesato, Frank Pfenning, and David Walker. A concurrent logical framework: the propositional fragment. In *TYPES 2003: Types for Proofs and Programs*, volume 3085 of *Lecture Notes in Computer Science*, pages 355–377. Springer, 2003. URL: `https://link.springer.com/chapter/10.1007/978-3-540-24849-1_23`.