Complexity of Reasoning in Residuated Kleene Algebras

Stepan Kuznetsov[1,2]

[1] Steklov Mathematical Institute of the RAS, Moscow, Russia
sk@mi-ras.ru
[2] National Research University Higher School of Economics, Moscow, Russia

Abstract

We survey algorithmic complexity results for reasoning in several classes of algebras with Kleene star, with an emphasis on the residuated ones, in the sense of their equational theories.

1 Introduction

Iteration, or Kleene star, is one of the most interesting algebraic operations which appear in theoretical computer science. Being an inductive object, iteration extends a purely propositional, algebraic logic setting with interesting features usually found in more expressive systems, like arithmetic or higher order inductive types theories.

The notion of *residuated Kleene algebra,* or *action algebra,* was introduced by V. Pratt [26] as an extension of traditional Kleene algebras [11] by means of residuation, or division operations. D. Kozen [13] further extended action algebras to *action lattices,* which include both meet and join (while Pratt's action algebras, as well as Kleene algebras are only join-semilattices). One also has to distinguish the class of all residuated Kleene algebras (lattices) from the more specific class of *-continuous ones. In the former class, the Kleene star is defined in an inductive manner, as a fixpoint; for the latter, iteration of a is, roughly speaking, the supremum of a^n.

In this essay, we survey algorithmic complexity results for reasoning in these classes of algebras in the sense of their equational theories. The choice of such a weak language (considering only generally true equations) is motivated by the fact that already the Horn theory even for Kleene algebras, without meet and residuals, reaches the maximal possible complexity. Namely, for the *-continuous case it is Π_1^1-hard, and for the general case it is Σ_1^0-hard [15], which coincides with the easily-proved upper bounds that come from the syntax of the logics axiomatising these theories.

For a quick start, we define a Kleene algebra as an algebraic structure with two binary operations, $\cdot$ (product) and $\vee$ (union, also denoted by $+$), the unit constant $\mathbf{1}$, and one unary operation, *. The product and the unit impose a monoid structure; $\vee$ makes the structure a join-semilattice. The product is distributive over join: $a \cdot (b \vee c) = (a \cdot b) \vee (a \cdot c)$ and $(a \vee b) \cdot c = (a \cdot c) \vee (b \cdot c)$. This entails monotonicity of the product w.r.t. the semilattice preorder $\prec$.

One can also impose the existence of the zero object, $\mathbf{0}$, as the smallest element of the semilattice. The zero object is also required to be the zero w.r.t. the product, $\mathbf{0} \cdot a = a \cdot \mathbf{0} = \mathbf{0}$. In other words, $\cdot$, $\vee$, $\mathbf{1}$, and $\mathbf{0}$ form a structure of an idempotent ring. Throughout this paper, we do not use the zero constant, but our results are also valid in the presence of $\mathbf{0}$.

Iteration, a^*, is defined as an element which is simultaneously the smallest (in the semilattice preorder $\preceq$) b such that $\mathbf{1} \vee ab \preceq b$ and the smallest c such that $\mathbf{1} \vee ca \preceq c$ (i.e., a^* is both the left and the right iteration of a). A Kleene algebra is *-continuous, if for any a, b, c we have $b \cdot a^* \cdot c = \sup_{\preceq} \{b \cdot a^n \cdot c \mid n \geq 0\}$.

Kleene lattices are Kleene algebras with $\wedge$, which is the lattice meet operation. They were introduced by Kozen [13] in order to gain closure under matrix formation. Finally, residuated

Kleene lattices (algebras), or action lattices (resp., algebras) are obtained by adding two division operations (residuals), $\backslash$ and $/$, obeying the following principles (which are essentially due to Lambek [20]): $b \preceq a \mathbin{\backslash} c \iff a \cdot b \preceq c \iff a \preceq c \mathbin{/} b$. Removing $\vee$ and $\wedge$ from the signature of action lattices results in residuated monoids with iteration.

In the next section, we provide sequential axiomatisations for the (in)equational theories, or algebraic logics, of these classes of structures.

The results on the complexity of these theories we feature in this paper, both previously known and new, are summarised in the following table. For comparison, we also included results on structures without iteration, namely residuated monoids and residuated lattices.

	general	***-continuous**
Kleene algebras	PSPACE-complete (D. Kozen 1994 [12])	
Kleene lattices	*open problem*	*open problem*
action algebras	Σ_1^0-complete (new)	Π_1^0-complete (W. Buszkowski, E. Palka 2007 [4, 23])
action lattices	Σ_1^0-complete (new: LICS 2019 [19])	Π_1^0-complete (W. Buszkowski, E. Palka 2007 [4, 23])
residuated monoids with iteration	*open problem*	Π_1^0-complete (new: submitted to a journal [18])
residuated monoids	NP-complete (M. Pentus 1996 [24])	
residuated lattices	PSPACE-complete (M. Kanovich 1994 [9])	

Let us comment a bit on the second line of this table, concerning Kleene lattices. To the best of the author's knowledge, the complexity for their equational theories, both in the general and in the *-continuous case, is still an open question. A conjecture could be formulated however, that the theory is decidable and belongs to EXPSPACE, based on known results for more specific classes of Kleene lattices [1, 3, 21, 6]. The lattices considered in the papers cited here, however, are distributive [6], which is not the case for Kleene lattices in general.

2 Calculi for Equational Theories

The (in)equational theory of residuated lattices (without iteration) is described by **MALC**, the multiplicative-additive Lambek calculus [22]. Sequents of **MALC** are of the form $\Pi \vdash \beta$, where β is a formula (built from variables using residuated lattice operations) and Π is a finite, possibly empty, linearly ordered sequence of formulae. The empty sequence is denoted by Λ. A sequent $\alpha_1, \ldots, \alpha_n \vdash \beta$ is interpreted as $\alpha_1 \cdot \ldots \cdot \alpha_n \preceq \beta$; $\Lambda \vdash \beta$ means $\mathbf{1} \preceq \beta$. Axioms and inference rules of **MALC** are as follows.

$$\frac{}{\alpha \vdash \alpha}\ (\mathrm{ax}) \qquad \frac{\Pi \vdash \alpha \quad \Gamma, \beta, \Delta \vdash \gamma}{\Gamma, \Pi, \alpha \mathbin{\backslash} \beta, \Delta \vdash \gamma}\ (\backslash \vdash) \qquad \frac{\alpha, \Pi \vdash \beta}{\Pi \vdash \alpha \mathbin{\backslash} \beta}\ (\vdash \backslash) \qquad \frac{\Gamma, \alpha, \beta, \Delta \vdash \gamma}{\Gamma, \alpha \cdot \beta, \Delta \vdash \gamma}\ (\cdot \vdash)$$

$$\frac{\Gamma, \Delta \vdash \gamma}{\Gamma, \mathbf{1}, \Delta \vdash \gamma}\ (\mathbf{1} \vdash) \qquad \frac{\Pi \vdash \alpha \quad \Gamma, \beta, \Delta \vdash \gamma}{\Gamma, \beta \mathbin{/} \alpha, \Pi, \Delta \vdash \gamma}\ (/ \vdash) \qquad \frac{\Pi, \alpha \vdash \beta}{\Pi \vdash \beta \mathbin{/} \alpha}\ (\vdash /) \qquad \frac{\Gamma \vdash \alpha \quad \Delta \vdash \beta}{\Gamma, \Delta \vdash \alpha \cdot \beta}\ (\vdash \cdot)$$

$$\frac{}{\Lambda \vdash \mathbf{1}}\ (\vdash \mathbf{1}) \qquad \frac{\Gamma, \alpha_i, \Delta \vdash \gamma}{\Gamma, \alpha_1 \wedge \alpha_2, \Delta \vdash \gamma}\ (\wedge \vdash)_i,\ i = 1,2 \qquad \frac{\Pi \vdash \alpha_1 \quad \Pi \vdash \alpha_2}{\Pi \vdash \alpha_1 \wedge \alpha_2}\ (\vdash \wedge)$$

$$\frac{\Gamma, \alpha_1, \Delta \vdash \gamma \quad \Gamma, \alpha_2, \Delta \vdash \gamma}{\Gamma, \alpha_1 \vee \alpha_2, \Delta \vdash \gamma}\ (\vee \vdash) \qquad \frac{\Pi \vdash \alpha_i}{\Pi \vdash \alpha_1 \vee \alpha_2}\ (\vdash \vee)_i,\ i = 1,2$$

The logic for action lattices, **ACT** (action logic), is obtained from **MALC** by adding the following rules.

$$\frac{\Lambda \vdash \beta \quad \alpha, \beta \vdash \beta}{\alpha^* \vdash \beta}\ ({}^* \vdash)_{\mathrm{fp}} \qquad \frac{}{\Lambda \vdash \alpha^*}\ (\vdash {}^*)_0 \qquad \frac{\Pi \vdash \alpha \quad \Delta \vdash \alpha^*}{\Pi, \Delta \vdash \alpha^*}\ (\vdash {}^*)_{\mathrm{fp}} \qquad \frac{\Pi \vdash \alpha \quad \Gamma, \alpha, \Delta \vdash \gamma}{\Gamma, \Pi, \Delta \vdash \gamma}\ (\mathrm{cut})$$

In this formulation of **ACT**, we formulate the rules for Kleene star without left contexts. One could add these context, making the rules more Gentzen-style, like Jipsen [8] and Pentus [25] do. However, as discussed below, this does not help with cut elimination.

The logic for *-continuous action lattices, $\mathbf{ACT}_\omega$ (infinitary action logic), is an extension of **MALC** with the following rules.

$$\frac{\bigl(\Gamma, \alpha^n, \Delta \vdash \gamma\bigr)_{n \in \omega}}{\Gamma, \alpha^*, \Delta \vdash \gamma}\ ({}^* \vdash)_\omega \qquad \frac{\Pi_1 \vdash \alpha \quad \dots \quad \Pi_n \vdash \alpha}{\Pi_1, \dots, \Pi_n \vdash \alpha^*}\ (\vdash {}^*)_n,\ n \in \omega$$

Notice that we include cut as an official rule of the system only in **ACT**. Indeed, in $\mathbf{ACT}_\omega$, as shown by Palka [23], cut is eliminable, while for **ACT** no cut-free system is known. An attempt to construct such a system was taken by P. Jipsen [8], but Buszkowski [4] showed that in Jipsen's system cut is not eliminable.

All these systems are sound and complete w.r.t. the corresponding classes of algebras, by Lindenbaum – Tarski construction.

3 Reasoning in Action Lattices

3.1 Reasoning in Action Lattices: Undecidability

In this section we prove undecidability of **ACT** as well as all logics between **ACT** and $\mathbf{ACT}_\omega$, extending Buszkowski's [4] technique of proving Π_1^0-hardness of $\mathbf{ACT}_\omega$.

Theorem 1. *Any $\mathcal{L}$, such that $\mathbf{ACT} \subseteq \mathcal{L} \subseteq \mathbf{ACT}_\omega$, is undecidable.*

The proof of this theorem is based on encoding behaviour of deterministic Turing machines via the totality property of context-free grammars. Usually, in undecidability proofs one takes care about halting vs. non-halting of a Turing machine on a given input. Here we distinguish three possible kinds of behaviour of a Turing machine $\mathfrak{M}$ on input x:

1. $\mathfrak{M}$ halts on x;
2. $\mathfrak{M}$ *trivially cycles* on x;
3. $\mathfrak{M}$, when running on x, does not halt for another reason.

The notion of trivally cycling is defined as follows. In what follows, we consider only deterministic, single-tape, single-head Turing machines.

Definition 1. A state q_c of a Turing machine $\mathfrak{M}$ is a *cycling* one, if all rules of $\mathfrak{M}$ for this state do not change the configuration (i.e., the rules are of the form $\langle q_c, a\rangle \to \langle q_c, a, N\rangle$, for any letter a of the internal alphabet; N stands for "no move").

Definition 2. A Turing machine $\mathfrak{M}$ trivially cycles on input word x, if $\mathfrak{M}$ reaches a cycling state q_c while running on x. The class of pairs $\langle \mathfrak{M}, x\rangle$ where $\mathfrak{M}$ trivially cycles on x is denoted by $\mathcal{C}$.

The notion of trivially cycling is essentially equivalent to reachability of the designated state q_c. For our exposition, however, it is more convenient to consider the case of trivially cycling as a subcase of non-halting. Therefore, we force the Turing machine to get stuck in q_c forever and thus forbid halting after reaching q_c.

There is also a more general notion of *cycling* on a given input, when $\mathfrak{M}$ returns to the same configuration (and therefore runs infinitely long).

Definition 3. A Turing machine $\mathfrak{M}$ halts on input word x, if it reaches a configuraton from which there is no next move (thus, we do not distinguish "successful" computations from those which halt by error). The class of pairs $\langle \mathfrak{M}, x \rangle$ where $\mathfrak{M}$ halts on x is denoted by $\mathcal{H}$. Its complement, i.e., the class of pairs $\langle \mathfrak{M}, x \rangle$ where $\mathfrak{M}$ does not halt on x, i.e., runs infinitely, is denoted by $\overline{\mathcal{H}}$.

Evidently, $\mathcal{C} \subset \overline{\mathcal{H}}$, and we shall use a folklore fact that $\mathcal{C}$ and $\mathcal{H}$ are recursively inseparable:

Proposition 1. *The classes $\mathcal{C}$ and $\mathcal{H}$ are recursively inseparable, i.e., there exists no decidable class $\mathcal{K}$ such that $\mathcal{C} \subseteq \mathcal{K} \subseteq \overline{\mathcal{H}}$.*

Next, we encode each configuration of a Turing machine as $a_1 \dots a_{i-1} q a_i \dots a_m$ (where q is the state, and the word in the memory is $a_1 \dots a_m$, a_i being observed), and the *protocol* of execution is $\# k_0 \# k_1 \# \dots \# k_n \#$, where $k_0 = q_0 x$ is the initial configuration, and each k_i is the successor of k_{i-1}; $\#$ is a fresh symbol. The protocol is a halting one, if k_n has no successor.

Some encodings, in order to simplify proofs a bit, make configurations in a protocol alternatingly reversed ($\# k_0 \# k_1^R \# k_2 \# k_3^R \# \dots$); however, in Kozen's textbook [14] one can find an encoding without reversions.

Let us fix $\mathfrak{M}$ and its input word x. In order to describe all the words except the halting protocol of $\mathfrak{M}$ on x by a context-free grammar $\mathcal{G}_{\mathfrak{M},x}$, we consider three classes of such words.

First, there are words beginning with # which *cannot be even a prefix* of a halting protocol. These include all words, which either

- include q_c, where q_c is a cycling state.
- or include a block between #'s, which is not a code of a configuration;
- or include a block of the form $\# k \# k' \#$, where k' is not the successor of k;

Second, there are possibly *incomplete* protocols. These include

- words whose last symbol is not #;
- words of the form $u \# k \#$, where k is a configuration which has a successor.

Third, there are words not beginning with #.

Now we are ready to construct $\mathcal{G}_{\mathfrak{M},x}$, which is going to be a context-free grammar in Greibach [7] normal form. All words of the first class form a context-free language (cf. [14], adding extra rules for words including q_c). If we remove the leftmost #, the language remains context-free. Let us write down a grammar for it in Greibach normal form and let its starting symbol be Y. For the second class, we have the same, since these words form a regular, and therefore context-free, language. The grammar for this language, with the leftmost #'s removed, has starting symbol Z. We can suppose that the sets non-terminal symbols of the two grammars are disjoint. Next, we put things all together:

$S \to bU$, for all $b \neq \#$	$U \to aU$, for any a	$Y \to \dots$
$S \to \# Y U$	$U \to a$, for any a	$Z \to \dots$
$S \to \# Z$	$S \to a$ for all a	

Notice that U, which generates all non-empty words, appears in the second, but not in the third production rule. As mentioned above, any word which has a *prefix* from the first class is necessarily not the halting protocol. For the second class, this is not the case.

By construction, $\mathcal{G}_{\mathfrak{M},x}$ generates all non-empty words if and only if there exists no halting protocol, i.e., $\mathfrak{M}$ does not halt on x.

Let us now translate $\mathcal{G}_{\mathfrak{M},x}$ to the Lambek calculus. The construction essentially resembles the translation of context-free grammars to basic categorial grammars by Gaifman [2]. Let non-terminals of $\mathcal{G}_{\mathfrak{M},x}$ be Lambek variables. For each letter a let

$$\varphi_a = \bigwedge\{(A\,/(B_1 \cdot \ldots \cdot B_\ell) \mid (A \to aB_1 \ldots B_\ell) \text{ is a production rule of } \mathcal{G}_{\mathfrak{M},x}\}$$

(in particular, for a production rule of the form $A \to a$ we have $\ell = 0$, and $A\,/(B_1 \cdot \ldots \cdot B_\ell)$ means just A), and let

$$\psi_{\mathfrak{M},x} = \bigvee_a \varphi_a.$$

Proposition 2. *A word $a_1 \ldots a_n$ is generated from non-terminal A in $\mathcal{G}_{\mathfrak{M},x}$ if and only if the sequent $\varphi_{a_1}, \ldots, \varphi_{a_n} \vdash A$ is derivable in* **MALC**. [2, 4]

This lemma is easily established by induction on the size of derivations (in both directions).

Proposition 3. *$\mathcal{G}_{\mathfrak{M},x}$ generates all words of length n iff $\psi^n_{\mathfrak{M},x} \vdash S$ is derivable in* **MALC**. [4]

Corollary 4. *$\mathcal{G}_{\mathfrak{M},x}$ generates all non-empty words iff $\psi^+_{\mathfrak{M},x} \vdash S$ is derivable in* $\mathbf{ACT}_\omega$.

Recall that $\psi^n = \psi, \ldots, \psi$ (n times) and $\psi^+ = \psi \cdot \psi^*$.

Corollary 5. *$\mathfrak{M}$ does not halt on x (i.e., $\langle \mathfrak{M}, x\rangle \in \overline{\mathcal{H}}$) iff $\psi^+_{\mathfrak{M},x} \vdash S$ is derivable in* $\mathbf{ACT}_\omega$.

Our new key lemma is as follows:

Lemma 6. *If $\mathfrak{M}$ trivially cycles on x, then $\psi^+_{\mathfrak{M},x} \vdash S$ is derivable in* **ACT**.

Proof. For simplicity we write just ψ for $\psi_{\mathfrak{M},x}$. First we show derivability of $\psi^+ \vdash U$ in **ACT**. Informally, this means that already **ACT** "knows that" U generates all non-empty words.

$$\dfrac{\dfrac{\dfrac{\dfrac{\psi \vdash U}{\Lambda \vdash \psi \setminus U}\,(\vdash \setminus) \qquad \dfrac{\psi \vdash \psi \qquad \dfrac{\dfrac{\psi \vdash U\,/\,U}{\psi, U \vdash U}\,(\vdash /),\text{ inverted}}{U \vdash \psi \setminus U}\,(\vdash \setminus)}{\psi, \psi \setminus U \vdash \psi \setminus U}\,(\setminus \vdash)}{\psi^* \vdash \psi \setminus U}\,({}^* \vdash)_{\mathrm{fp}}}{\psi, \psi^* \vdash U}\,(\vdash \setminus),\text{ inverted}}{\psi^+ \vdash U}\,(\cdot \vdash)$$

The sequents $\psi \vdash U \setminus U$ and $\psi \vdash U$ are derivable (in **MALC**, and therefore in **ACT**), since $U \setminus U$ and U are included in all φ_a conjunctions.

Next, $\psi^+ \vdash S$ is derived using the "long rule," which is admissible in **ACT** (for any fixed $n \geq 1$):

$$\dfrac{\psi \vdash S \quad \psi^2 \vdash S \quad \ldots \quad \psi^n \vdash S \quad \psi^n, \psi^+ \vdash S}{\psi^+ \vdash S}$$

Since $\mathfrak{M}$ trivially cycles on x, there is a correct (incomplete) protocol which includes q_c. Let n be the length of such a protocol, in symbols (not the number of states!).

The first n sequents, $\psi \vdash S, \ldots, \psi^n \vdash S$, are derivable by Proposition 3, since $\mathcal{G}_{\mathfrak{M},x}$ indeed generates all non-empty words. As for $\psi^n, \psi^+ \vdash S$, it is derived from all sequents of the form $\varphi_{a_1}, \ldots, \varphi_{a_n}, \psi^+ \vdash S$, by the left rule for disjunction. By cut with $\psi^+ \vdash U$, this sequent is derivable from $\varphi_{a_1}, \ldots, \varphi_{a_n}, U \vdash S$. In order to derive the latter, consider two cases.

Case 1: $a_1 \neq \#$. Then $S \mathbin{/} U$ is inside φ_{a_1}, and $U \mathbin{/} U$ is inside $\varphi_{a_2}, \ldots, \varphi_{a_n}$. Thus, our sequent is derivable from $S \mathbin{/} U, U \mathbin{/} U, \ldots, U \mathbin{/} U, U \vdash S$.

Case 2: $a_1 = \#$. Then φ_{a_1} includes $S \mathbin{/} (Y \cdot U)$, and we derive as follows:

$$\dfrac{\dfrac{\varphi_{a_2}, \ldots, \varphi_{a_n} \vdash Y \quad U \vdash U \quad S \vdash S}{S \mathbin{/} (Y \cdot U), \varphi_{a_2}, \ldots, \varphi_{a_n}, U \vdash S}\,(\vdash \cdot), (/ \vdash)}{\varphi_{a_1}, \varphi_{a_2}, \ldots, \varphi_{a_n}, U \vdash S}\,(\wedge \vdash)$$

The sequent $\varphi_{a_2}, \ldots, \varphi_{a_n} \vdash Y$ is derivable by Proposition 2, since $a_2 \ldots a_n$ includes q_c and is therefore derivable from Y in $\mathcal{G}_{\mathfrak{M},x}$. $\square$

Now, for an arbitrary logic in the language of **ACT** and $\mathbf{ACT}_\omega$, let

$$\mathcal{K}(\mathcal{L}) = \{\langle \mathfrak{M}, x\rangle \mid \psi^+_{\mathfrak{M},x} \vdash S \text{ is derivable in } \mathcal{L}\}.$$

(Formally speaking, a "logic" is just an arbitrary set of sequents, so "derivable in $\mathcal{L}$" formally means "belongs to the set $\mathcal{L}$." When a logic is defined by a calculus, like **ACT** or $\mathbf{ACT}_\omega$, this set is the set of *theorems* of the logic.)

If $\mathbf{ACT} \subseteq \mathcal{L} \subseteq \mathbf{ACT}_\omega$, then Corollary 5 and Lemma 6 yield the following:

$$\mathcal{C} \subseteq \mathcal{K}(\mathbf{ACT}) \subseteq \mathcal{K}(\mathcal{L}) \subseteq \mathcal{K}(\mathbf{ACT}_\omega) = \overline{\mathcal{H}},$$

and by inseparability of $\mathcal{C}$ and $\mathcal{H}$ (Proposition 1) $\mathcal{K}(\mathcal{L})$ is undecidable for any $\mathcal{L}$ between **ACT** and $\mathbf{ACT}_\omega$, and so is $\mathcal{L}$ itself. This finishes the proof of Theorem 1.

3.2 Cyclic Systems for Action Logic

The idea of the undecidability proof explained above is much inspired by *cyclic* calculi for action logic [5, 16] (in particular, cycling of a Turing machine reflects as a cyclic proof of its non-halting), though these calculi are not explicitly used in the proof itself.

These cyclic systems also give an interesting example of a logic strictly between **ACT** and $\mathbf{ACT}_\omega$. Let us recall the definitions of non-well-founded and cyclic proof systems for action logic. As shown by Das and Pous [5], the rules for iteration in $\mathbf{ACT}_\omega$ can be equivalently replaced by the following ones:

$$\dfrac{\Gamma, \Delta \vdash \gamma \quad \Gamma, \alpha, \alpha^*, \Delta \vdash \gamma}{\Gamma, \alpha^*, \Delta \vdash \gamma}\,({}^* \vdash)' \qquad \dfrac{}{\Lambda \vdash \alpha^*}\,(\vdash {}^*)_0 \qquad \dfrac{\Gamma \vdash \alpha \quad \Delta \vdash \alpha^*}{\Gamma, \Delta \vdash \alpha^*}\,(\vdash {}^*)_{\mathrm{fp}}$$

now allowing non-well-founded derivations. More precisely, a *preproof* is an arbitrary derivation tree, possibly with infinitely long branches. A preproof is considered a valid derivation, if it obeys a certain correctness condition. Namely, every infinite branch should contain an infinite thread of occurrences of α^*, connected by the immediate ancestry relation, which undergoes principal applications of $({}^* \vdash)'$ infinitely many times. Let us denote this system by $\mathbf{ACT}_\infty$. Cut in $\mathbf{ACT}_\infty$ is eliminable [5].

The cyclic fragment of $\mathbf{ACT}_\infty$, denoted by $\mathbf{ACT}_{\mathrm{cycle}}$ includes only regular derivations, that is, derivations with only a finite number of non-isomorphic subtrees. These derivations are allowed to use cut, which is in general not eliminable here: the cut elimination procedure could transform a regular tree to an irregular one. As shown by Das and Pous [5], $\mathbf{ACT}_{\mathrm{cycle}}$ is equivalent to $\mathbf{ACT}$.

As one can notice, rules of $\mathbf{ACT}_\infty$ are asymmetric (left-handed). Consider the dual rules:

$$\frac{\Gamma, \Delta \vdash \gamma \quad \Gamma, \alpha^*, \alpha, \Delta \vdash \gamma}{\Gamma, \alpha^*, \Delta \vdash \gamma}\ ({}^* \vdash)'_r \qquad \frac{\Gamma \vdash \alpha^* \quad \Delta \vdash \alpha}{\Gamma, \Delta \vdash \alpha^*}\ (\vdash {}^*)_{\mathrm{fp;r}}$$

The interesting fact is that adding these rules to the cyclic subsystem $\mathbf{ACT}_{\mathrm{cycle}}$ yields a system which is strictly stronger than $\mathbf{ACT}$. Let us denote the new, symmetric system by $\mathbf{ACT}_{\mathrm{bicycle}}$. This system admits the following "induction-in-the-middle" rule:

$$\frac{\Lambda \vdash \beta \quad \alpha \vdash \beta \quad \alpha, \beta, \alpha \vdash \beta}{\alpha^* \vdash \beta}$$

Using this rule, one can derive $(p \wedge q \wedge (p \,/\, q) \wedge (p \setminus q))^+ \vdash p$, which is not derivable in $\mathbf{ACT}$ [17].

On the other side, $\mathbf{ACT}_{\mathrm{bicycle}}$ is still strictly included in $\mathbf{ACT}_\omega$. On one hand, $\mathbf{ACT}_\omega$ proves "induction-in-the-middle" and thus subsumes $\mathbf{ACT}_{\mathrm{bicycle}}$. On the other hand, these systems could not coincide due to complexity reasons: $\mathbf{ACT}_{\mathrm{bicycle}}$ is recursively enumerable, while $\mathbf{ACT}_\omega$ is Π^0_1-hard (see the next section). Thus, $\mathbf{ACT}_{\mathrm{bicycle}}$ is a natural example of a system strictly between $\mathbf{ACT}$ and $\mathbf{ACT}_\omega$.

3.3 Reasoning in Action Lattices: Complexity

In this section we show that our construction actually gives more than just undecidability. Namely we provide exact complexity bounds for $\mathbf{ACT}$ and $\mathbf{ACT}_{\mathrm{bicycle}}$ by proving their Σ^0_1-completeness. The infinitary system $\mathbf{ACT}_\omega$, in its turn, is Π^0_1-complete. The lower bound, Π^0_1-hardness, follows from $\mathcal{K}(\mathbf{ACT}_\omega) = \overline{\mathcal{H}}$. The upper bound was proved by Palka [23] using her *-eliminating technique, and also follows from cut-elimination results in non-well-founded proof systems for action logic by Das and Pous [5].

We follow a general road to obtain Σ^0_1-completeness results from recursive inseparability, noticed by Speranski [30]. The idea is to use effective inseparability instead of the usual one. The methods used are rather classical; we refer to Rogers' textbook [27] for concrete statements.

Definition 4. Let W_u be the domain of the partial function computed by Turing machine with code u (i.e., "the u-th recursively enumerable set"). A disjoint pair of sets A and B is called *effectively inseparable,* if there exists a computable function f with two arguments, such that if $A \subseteq W_u$, $B \subseteq W_v$, and $W_u \cap W_v = \varnothing$, then $f(u, v)$ is defined[1] and $f(u, v) \notin W_u \cup W_v$.

Intuitively, effective inseparability means that any attempt to separate A and B by a pair of recursively enumerable sets W_u and $W_v = \overline{W}_u$ (which would then, being complements of each other, be both decidable) is uniformly falsified by the function f, which provides an element which belongs neither to W_u not to W_v.

One can easily notice that if A and B are effectively inseparable, $A \subseteq A'$, $B \subseteq B'$, and $A' \cap B' = \varnothing$, then A' and B' are also effectively inseparable (one just takes the same f).

We use the following known facts on effective inseparability.

[1] In the usual definition of efficiently inseparable sets, f is allowed to be partial. Requiring f to be total, however, yields an equivalent definition [29, Exercise 4.13(a)].

Proposition 7. *If A and B are effectively inseparable and are both recursively enumerable (belong to Σ^0_1), then they are both Σ^0_1-complete.* [27, Exercise 11-14].

Proposition 8. *Classes $\mathcal{C}$ and $\mathcal{H}$ are effectively inseparable.* [27, Exercise 7-55(d)]

These propositions immediately yield the following theorem.

Theorem 2. *If $\mathbf{ACT} \subseteq \mathcal{L} \subseteq \mathbf{ACT}_\omega$ and $\mathcal{L}$ is recursively enumerable, then $\mathcal{L}$ is Σ^0_1-complete. In particular, $\mathbf{ACT}$ and $\mathbf{ACT}_{\mathrm{bicycle}}$ are Σ^0_1-complete.*

Proof. Obviously, $\mathbf{ACT}$ and $\mathbf{ACT}_{\mathrm{bicycle}}$, as well as the $\mathcal{C}$ and $\mathcal{H}$ classes, are recursively enumerable (*i.e.,* belong to Σ^0_1).

Recall that $\mathcal{C} \subseteq \mathcal{K}(\mathbf{ACT}) \subseteq \mathcal{K}(\mathcal{L}) \subseteq \mathcal{K}(\mathbf{ACT}_\omega) = \overline{\mathcal{H}}$. Since $\mathcal{C}$ and $\mathcal{H}$ are effectively inseparable (Proposition 8), so are $\mathcal{K}(\mathcal{L})$ and $\mathcal{H}$. By Proposition 7, this yields Σ^0_1-completeness of $\mathcal{K}(\mathcal{L})$. Since $\mathcal{L}$ subsumes $\mathcal{K}(\mathcal{L})$, it is also Σ^0_1-hard; the upper Σ^0_1 bound is given. $\square$

4 Restricting the Language

4.1 Reasoning in Action Algebras

Action algebras, as defined by Pratt [26], do not include the meet operation, $\wedge$. The logics for all action algebras and for the subclass of *-continuous ones are obtained, respectively, from $\mathbf{ACT}$ and $\mathbf{ACT}_\omega$ by removing the rules for $\wedge$. We denote the resulting calculi by $\mathbf{ACT}^\sim$ and $\mathbf{ACT}^\sim_\omega$ respectively. Due to cut elimination, $\mathbf{ACT}^\sim_\omega$ is a conservative fragment of $\mathbf{ACT}_\omega$. Algebraically, if a sequent is true in all *-continuous action lattices and does not include $\wedge$, then it is true in all *-continuous action algebras. For $\mathbf{ACT}$ and $\mathbf{ACT}^\sim$, this is unknown: potentially, there could exist a $\wedge$-free formula derivable in $\mathbf{ACT}$ (via a detour using cut with $\wedge$-formulae), but not in $\mathbf{ACT}^\sim$.

Buszkowski [4] proved Π^0_1-hardness for $\mathbf{ACT}^\sim_\omega$ also. The upper Π^0_1 bound follows from that for $\mathbf{ACT}_\omega$ [23, 5] and the conservativity result mentioned above. The trick used by Buszkowski in order to get rid of using $\wedge$ in his Π^0_1-hardness proof, is the usage of "pseudo-double-negation." Let b be a fresh variable, not used in our encoding. Let $\alpha^b = b \,/\, \alpha$; we call $\alpha^{bb} = b\,/(b\,/\,\alpha)$ the pseudo-double-negation of α. Since $\alpha^{bb} \wedge \beta^{bb}$ is equivalent to $(\alpha^b \vee \beta^b)^b$, this construction allows us to replace $\wedge$ with $\vee$, where needed. Below we develop this idea more accurately.

For each symbol a let

$$\tilde{\varphi}_a = \Big(\bigvee \{(A\,/(B_1 \cdot \ldots \cdot B_\ell))^b \mid A \to aB_1 \ldots B_\ell \text{ is a production rule of } \mathcal{G}_{\mathfrak{M},x}\}\Big)^b$$

and

$$\tilde{\psi}_{\mathfrak{M},x} = \bigvee_a \tilde{\varphi}_a.$$

Now it is sufficient to establish the following two lemmata, in order to use the machinery of Section 3 and obtain Theorems 1 and 2 in the restricted language without $\wedge$.

Lemma 9. *If $\mathfrak{M}$ trivially cycles on x, then $\tilde{\psi}^+_{\mathfrak{M},x} \vdash S^{bb}$ is derivable in $\mathbf{ACT}^\sim$.*

Lemma 10. *If $\tilde{\psi}^+_{\mathfrak{M},x} \vdash S^{bb}$ is derivable in $\mathbf{ACT}^\sim_\omega$, then $\mathfrak{M}$ does not halt on x.*

Using these two lemmata, we get, for any $\mathcal{L}$ between $\mathbf{ACT}^\sim$ and $\mathbf{ACT}^\sim_\omega$,

$$\mathcal{C} \subseteq \tilde{\mathcal{K}}(\mathbf{ACT}^\sim) \subseteq \tilde{\mathcal{K}}(\mathcal{L}) \subseteq \tilde{\mathcal{K}}(\mathbf{ACT}^\sim_\omega) \subseteq \overline{\mathcal{H}},$$

where $\tilde{\mathcal{K}}(\mathcal{L}) = \{\langle \mathfrak{M}, x\rangle \mid \tilde{\psi}^+_{\mathfrak{M},x} \vdash S^{bb} \text{ is derivable in } \mathcal{L}\}$. This immediately yields analogs of Theorems 1 and 2 for the systems without $\wedge$:

Theorem 3. *If $\mathbf{ACT}^\sim \subseteq \mathcal{L} \subseteq \mathbf{ACT}^\sim_\omega$, then $\mathcal{L}$ is undecidable. Moreover, if $\mathcal{L}$ is recursively enumerable, it is Σ^0_1-complete.*

Now let us prove Lemma 9 and Lemma 10.

Proof of Lemma 9. First we notice that if $A \to aB_1 \dots B_\ell$ is a production rule of $\mathcal{G}_{\mathfrak{M},x}$, then $\tilde{\varphi}_a \vdash (A \,/\, (B_1 \cdot \ldots \cdot B_\ell))^{bb}$ is derivable in $\mathbf{ACT}^\sim$. In particular, we can derive $\tilde{\varphi}_a \vdash U^{bb}$ and $\tilde{\varphi}_a \vdash (U \,/\, U)^{bb}$, which yield $\tilde{\psi} \vdash U^{bb}$ and $\tilde{\psi} \vdash (U \,/\, U)^{bb}$. This allows us to derive $\tilde{\psi}^+ \vdash U^{bb}$, just replacing U with U^{bb} in the derivation from the proof of Lemma 6 and using the fact that $\tilde{\psi} \vdash (U \,/\, U)^{bb}$ entails $\tilde{\psi} \vdash U^{bb} \,/\, U^{bb}$, by cut with $(U \,/\, U)^{bb} \vdash U^{bb} \,/\, U^{bb}$.

Next, we use the "long rule" to derive $\tilde{\psi}^+ \vdash S^{bb}$. The first n premises are handled exactly as Buszkowski does [4], using the fact that, for a fresh b, $\xi_1^{bb}, \dots, \xi_n^{bb} \vdash \zeta^{bb}$ is equiderivable in $\mathbf{MALC}$ with $\xi_1, \dots, \xi_n \vdash \zeta$ [10, Lemma 2]. For the last premise, the interesting case is $a_1 = \#$ (see proof of Lemma 6), in which the necessary $(S \,/\, (Y \cdot U))^{bb}, \tilde{\varphi}_{a_2}, \dots, \tilde{\varphi}_{a_n}, U^{bb} \vdash S^{bb}$ is obtained, by cut, from $(S \,/\, (Y \cdot U))^{bb}, Y^{bb}, U^{bb} \vdash S^{bb}$. □

Proof of Lemma 10. This is essentially due to Buszkowski [4]; see also [10, Lemma 12]. □

4.2 The Multiplicative Lambek Calculus with Iteration

In this section we go further and consider the system without both $\vee$ and $\wedge$, i.e., the purely multiplicative Lambek calculus extended with iteration. In this setting, we consider only the infinitary calculus, denoted by $\mathbf{L}^*_\omega$, and prove its Π^0_1-completeness. Complexity of the corresponding systems with fixpoint definitions of iteration is left as an open problem (we again conjecture Σ^0_1-completeness).

Notice that for Kleene algebras their equational theory can be interpreted in its fragment without $\vee$, by distributing $\vee$ out (for Kleene star, use $(a \vee b)^* = a^*(b^*a)^*$) and applying the disjunctive property. For the residuated case, this would not work (for example, one cannot distribute $\vee$ out of $(a \vee b) \,/\, c)$, thus complexity for the purely multiplicative case is a separate issue.

The key idea which allows the removal of both $\vee$ and $\wedge$ simultaneously is based on the following theorem:

Theorem 4. *For any context-free language without the empty word there exists, and can be effectively computed, a Lambek grammar with unique type assignments. Namely, if $\Sigma = \{a_1, \dots, a_n\}$ is the terminal alphabet, then there exist formulae $\alpha_1, \dots, \alpha_n, \beta$ such that $\alpha_1, \dots, \alpha_n \vdash \beta$ is derivable in the Lambek calculus iff $a_1 \dots a_n$ belongs to the language.*

An analog of this theorem was proved by Safiullin [28] (see also [16, Appendix]), but for the variant of the Lambek calculus where left-hand sides of sequents are required to be non-empty. For the version without this restriction, used in this paper, Safiullin's construction has to be modified [18].

Using this theorem, we encode the following 2-alternation problem for context-free grammars: a grammar over alphabet $\Sigma = \{a_1, a_2\}$ is 2-alternating, if it generates all the words starting with a_1 and ending with a_2. It is easy to show that this problem is also Π^0_1-hard. Now let $\xi = (\alpha_1^+ \cdot \alpha_2^+)^+$, where α_1 and α_2, along with β, are obtained from the grammar by Theorem 4. The grammar is 2-alternating, if and only if $\xi \vdash \beta$ is derivable in $\mathbf{L}^*_\omega$. Thus, we obtain Π^0_1-hardness of this system; the upper bound follows from the upper bound for $\mathbf{ACT}_\omega$ [23, 5] by conservativity.

Theorem 5. $\mathbf{L}^*_\omega$ *is* Π^0_1*-complete.*

Acknowledgments The author is grateful to Anupam Das, Max Kanovich, Fedor Pakhomov, Andre Scedrov, Daniyar Shamkanov, and Stanislav Speranski for fruitful discussions. Being a Young Russian Mathematics award winner, the author thanks its jury and sponsors for this high honour. Research towards this paper was supported within the framework of the HSE University Basic Research Program and funded by the Russian Academic Excellence Project '5-100.'

References

[1] H. Andréka, S. Mikulás, and I. Németi. The equational theory of Kleene lattices. *Theoretical Computer Science*, 412(52):7099–7108, 2011.

[2] Y. Bar-Hillel, C. Gaifman, and E. Shamir. On categorial and phrase-structure grammars. *Bulletin of the Research Council of Israel*, 9F:1–16, 1960.

[3] P. Brunet and D. Pous. Petri automata for Kleene allegories. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2015)*, pages 68–79. IEEE, 2015.

[4] W. Buszkowski. On action logic: equational theories of action algebras. *Journal of Logic and Computation*, 17(1):199–217, 2007.

[5] A. Das and D. Pous. Non-wellfounded proof theory for (Kleene+action) (algebras+lattices). In D. Ghica and A. Jung, editors, *27th EACSL Annual Conference on Computer Science Logic (CSL 2018)*, volume 119 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 19:1–19:18. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2018.

[6] A. Doumane and D. Pous. Completeness for identity-free Kleene lattices. In *29th International Conference on Concurrency Theory (CONCUR 2018)*, volume 118 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:17. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2018.

[7] S. Greibach. A new normal-form theorem for context-free phrase structure grammars. *Journal of the ACM*, 12(1):42–52, 1965.

[8] P. Jipsen. From residuated semirings to Kleene algebras. *Studia Logica*, 76:291–303, 2004.

[9] M. Kanovich. Horn fragments of non-commutaive logics with additives are PSPACE-complete. In *1994 Annual Conference of the EACSL*, Kazimierz, Poland, 1994.

[10] M. Kanovich, S. Kuznetsov, and A. Scedrov. The complexity of the multiplicative-additive Lambek calculus: 25 years later, 2019. Accepted to WoLLIC.

[11] S. C. Kleene. Representation of events in nerve nets and finite automata. In *Automata Studies*, pages 3–41. Princeton University Press, 1956.

[12] D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Information and Computation*, 110(2):366–390, 1994.

[13] D. Kozen. On action algebras. In J. van Eijck and A. Visser, editors, *Logic and Information Flow*, pages 78–88. MIT Press, 1994.

[14] D. Kozen. *Automata and Complexity.* Springer-Verlag, New York, 1997.

[15] D. Kozen. On the complexity of reasoning in Kleene algebra. *Information and Computation*, 179:152–162, 2002.

[16] S. Kuznetsov. The Lambek calculus with iteration: two variants. In J. Kennedy and R. de Queiroz, editors, *WoLLIC 2017: Logic, Language, Information, and Computation*, volume 10388 of *Lecture Notes in Computer Science*, pages 182–198. Springer, 2017.

[17] S. Kuznetsov. *-continuity vs. induction: divide and conquer. In G. Bezhanishvili, G. D'Agostino, G. Metcalfe, and T. Studer, editors, *AiML 2018*, volume 12 of *Advances in Modal Logic*, pages 439–510, 2018.

[18] S. Kuznetsov. Complexity of the infinitary Lambek calculus with Kleene star, 2019. Submitted.

[19] S. Kuznetsov. The logic of action lattices is undecidable, 2019. Accepted to LICS.

[20] J. Lambek. The mathematics of sentence structure. *American Mathematical Monthly*, 65:154–170, 1958.

[21] Y. Nakamura. Partial derivatives on graphs for Kleene allegories. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2017)*. IEEE, 2017.

[22] H. Ono. Semantics for substructural logics. In P. Schroeder-Heister and K. Došen, editors, *Substructural Logics*, volume 2 of *Studies in Logic and Computation*, pages 259–291. Clarendon Press, Oxford, 1993.

[23] E. Palka. An infinitary sequent system for the equational theory of *-continuous action lattices. *Fundamenta Informaticae*, 78(2):295–309, 2007.

[24] M. Pentus. Lambek calculus is NP-complete. *Theoretical Computer Science*, 351(1):186–201, 2006.

[25] M. Pentus. Residuated monoids with Kleene star. Unpublished manuscript, 2010.

[26] V. Pratt. Action logic and pure induction. In J. van Eijck, editor, *JELIA 1990: Logics in AI*, volume 478 of *Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence)*, pages 97–120. Springer, 1991.

[27] H. Rogers. *Theory of recursive functions and effective computability.* McGraw-Hill, 1967.

[28] A. N. Safiullin. Derivability of admissible rules with simple premises in the Lambek calculus. *Moscow University Mathematics Bulletin*, 62(4):168–171, 2007.

[29] R. I. Soare. *Recursively enumerable sets and degrees.* Perspectives in Mathematical Logic. Springer-Verlag, 1987.

[30] S. Speranski. A note on hereditarily Π^0_1- and Σ^0_1-complete sets of sentences. *Journal of Logic and Computation*, 26(5):1729–1741, 2016.