Kleene algebra with commutativity conditions is undecidable

Arthur Azevedo de Amorim, Marco Gaboardi, Cheng Zhang

Kleene algebra with commutativity conditions is undecidable

Arthur Azevedo de Amorim Marco Gaboardi Cheng Zhang

April 5, 2024

Abstract

We prove that the equational theory of Kleene algebra with commutativity conditions on atomic terms is undecidable, thereby settling a longstanding open question in the theory of Kleene algebra. In fact, we show that this undecidability result holds even if we drop the *induction axioms* of Kleene algebra, which leads to a simpler equational theory. This complements a recent result of Kuznetsov, who independently established a similar undecidability result, but relying on the full power of induction.

1 Introduction

Kleene algebra is an algebraic framework that generalizes the equational properties of operations on regular languages. One of the pleasant properties of Kleene algebra is that its equational theory is decidable, which enables several applications. In particular, there are several program analyses that can be defined by translating programs into Kleene algebra terms, and then checking which terms are equal.

Often, it is convenient to enrich the definition of Kleene algebra with additional axioms. One popular class of axioms are commutativity conditions of the form $e_1 e_2 = e_2 e_1$, which state that the terms e_1 and e_2 can be composed in any order. In terms of program analysis, the terms e_1 and e_2 correspond to sub-commands of a larger program, and their commutativity ensures that they can be executed in any order without affecting the final output.

Unfortunately, it is known that such conditions can pose issues for the decidability. In particular, if we are allowed to add arbitrary commutativity conditions between atomic Kleene algebra terms, it is undecidable to test if two Kleene algebra equations hold for all language models [Koz97]—in fact, it has been known that this problem is Π^0_1-complete. Such language models are particularly important for program analysis applications, which hinders the applicability of Kleene algebra in these contexts. Nevertheless, for a long time, it was unknown if this undecidability result still held if we considered *arbitrary* Kleene algebras with commutativity conditions on atoms—in other words, if we wanted to consider only equations in freely generated Kleene algebras.

This paper settles this question by proving that it is impossible to decide whether an equation between two terms holds only by using commutativity conditions on atoms and the axioms of Kleene algebra. The question has been independently settled by Kuznetsov [Kuz23], who showed that this problem is, in fact, Σ_1^0-complete (that is, equivalent to the halting problem for Turing machines). Though our techniques overlap, we show actually that the undecidability result extends to weaker version of Kleene algebra where we use only a subset of its axioms. More precisely, we show that the theory of Kleene algebra remains undecidable even if we drop its *induction axioms*, which are needed to prove many identities involving the iteration operation on Kleene algebras.

Like prior results on this question, our development views uses Kleene algebra terms to model certain transition systems—in particular, automata like Turing machines that operate on strings. Given a final state x for a transition system, we can construct an inequality on Kleene algebra terms that roughly means "if the system reaches a final state from a given initial state, the final state must be x". This inequality can be proved if x is indeed a final state, and can be refuted if the transition system reaches some other final state y. This means that we can use Kleene algebra terms to decide a language on that is reminiscent of the acceptance problem for Turing machines, which implies that it is undecidable.

Structure of the paper In Section 2, we introduce an abstract framework for stating the problem of Kleene algebra terms modulo commutativity conditions, using the language of category theory. In Section 3, we recall basic facts about Kleene algebra and related structures.

In Section 4, we show how we can view Kleene algebra terms as automata, proving an *expansion lemma* (Lemma 4.8) that guarantees that most terms can be expanded so that all of its matched strings bounded by some maximum length can be identified. This framework generalizes the usual definitions of derivative on Kleene algebra terms, but does not rely on the induction axioms of Kleene algebra.

In Section 5, we develop a framework for representing relations using Kleene algebra terms. Our results apply to so-called *bounded-output* terms, which, roughly speaking, represent relations that map a string to only finitely many next strings. This restriction allows us to represent the transitive closure of a transition relation using Kleene algebra iteration, and also to analyze the behavior of the transition relation on paths of finite length. We prove a *partial reachability* result (Theorem 5.23), which shows that we can construct an inequality on Kleene algebra terms to upper bound the set of strings that can be reached from some input configuration.

In Section 6, we use our framework of relations to represent two-counter machines, a computation formalism that is equivalent to Turing machines in expressiveness. Our partial reachability result implies that, if we could decide equalities between Kleene algebra terms, we could solve an undecidable problem on Turing machines.

2 Commutable Sets

To discuss properties of Kleene algebras with commutativity conditions, it will be useful to have an abstract notion of what it means for two elements to commute.

Definition 2.1. A *commuting relation* on a set X is a reflexive symmetric relation on X. A *commutable set* is a carrier set endowed with a commuting relation $\sim$. We say that two elements x and y commute if $x \sim y$. A commutable set is *commutative* if all two elements commute; it is *discrete* if the commuting relation is equality.

Definition 2.2. A morphism of commutable sets is a function between the carriers that preserves the commuting relation. This data defines a category Comm.

Lemma 2.3. *Let* $U_{\mathsf{Comm}} : \mathsf{Comm} \to \mathsf{Set}$ *be the forgetful functor that forgets the commuting relation of a commutable set. This functor has a left adjoint* $D : \mathsf{Set} \to \mathsf{Comm}$*, which views a set as a discrete commutable set, and a right adjoint* $K : \mathsf{Set} \to \mathsf{Comm}$*, which views a set as a commutative set.*

Definition 2.4. A *commutable subset* of a commutable set X is a commutable set Y whose carrier is a subset of X, and whose commuting relation is obtained by restricting the corresponding relation of X. We'll often abuse notation and treat a subobject $Y \hookrightarrow X$ as a commutable subset if its image in X is a commutable subset.

Definition 2.5. Given two commutable sets X and Y, their sum is given by the disjoint union $X + Y$ endowed with the commuting relation generated by the following rules:

$$\frac{x \sim x'}{\iota_1(x) \sim \iota_1(x')} \qquad \frac{y \sim y'}{\iota_2(y) \sim \iota_2(y')}$$

This yields a binary coproduct on Comm.

Definition 2.6. Let X and Y be objects of Comm. We define the object $X \oplus Y \in \mathsf{Comm}$ as follows. The carrier of $X \oplus Y$ is the disjoint union $X + Y$. We use x_l and y_r to indicate the copies of $x \in X$ and $y \in Y$ in $X \oplus Y$. The relation on $X \oplus Y$ is generated by the following rules:

$$\frac{}{x_l \sim y_r} \qquad \frac{x \sim x'}{x_l \sim x'_l} \qquad \frac{y \sim y'}{y_r \sim y'_r}$$

The canonical injections $(-)_l : X \to X \oplus Y$ and $(-)_r : Y \to X \oplus Y$ are morphisms in Comm (and present commutable subsets). We'll abbreviate $X \oplus X$ as $\ddot{X}$.

3 Basic Kleene Algebra

Here we collect some basic facts about Kleene algebra.

Definition 3.1. A (left-biased) *weak Kleene algebra* is an idempotent semiring X equipped with a star operation. Spelled out explicitly, this means that X has operations of types

$$\begin{aligned} 1 &: X \\ 0 &: X \\ (-) + (-) &: X \times X \to X \\ (-) \cdot (-) &: X \times X \to X \\ (-)^* &: X \to X. \end{aligned}$$

which are required to satisfy the following equations:

$$\begin{aligned} 1 \cdot x &= x \\ x \cdot 1 &= x \\ 0 \cdot x &= 0 \\ x \cdot 0 &= 0 \\ x \cdot (y \cdot z) &= (x \cdot y) \cdot z \\ 0 + x &= x \\ x + y &= y + x \\ x + (y + z) &= (x + y) + z \\ x \cdot (y + z) &= x \cdot y + x \cdot z \\ (x + y) \cdot z &= x \cdot z + y \cdot z \\ x^* &= 1 + x \cdot x^* \qquad \text{left unfolding.} \end{aligned}$$

We view a weak Kleene algebra as a partial order by using the usual ordering relation on idempotent monoids: $x \leq y$ means that $y + x = x$. A *Kleene algebra* is a weak Kleene algebra that satisfies the following properties:

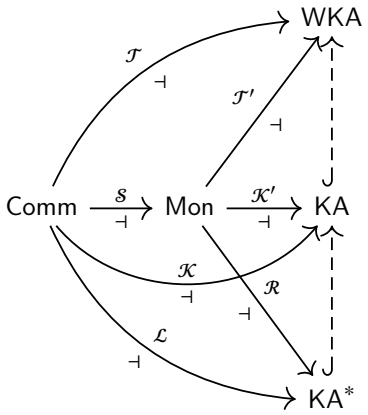
$$\begin{aligned} xy \leq y \Rightarrow x^* y \leq y \qquad & \text{left induction} \\ xy \leq x \Rightarrow xy^* \leq x \qquad & \text{right induction.} \end{aligned}$$

A $*$*-continuous Kleene algebra* is a weak Kleene algebra where $\sup_{n \geq 0} pq^n r = pq^* r$ holds for all p, q and r.

Definition 3.2. Let X and Y be weak Kleene algebras. A *morphism* of type $X \to Y$ is a function $f : X \to Y$ that commutes with all the algebra operations. In what follows, we let WKA denote the category of weak Kleene algebras. We denote by KA and KA^* the full subcategories of Kleene algebras and $*$-continuous algebras.

The booleans $\mathbb{2} \triangleq \{0 \leq 1\}$ form a Kleene algebra. The addition operation is disjunction, the multiplication operation is conjunction, and the star operation always outputs 1. This Kleene algebra is the initial object in all three categories WKA, KA and KA^*.

These categories have various relationships that can be visualized in the following diagram, where the dashed arrows indicate places where the diagram does not necessarily commute.



First, we can show that every $*$-continuous algebra is actually a Kleene algebra, so KA^* is actually a full subcategory of KA. Given a monoid X, we can view it as a commutable set by saying that $x \sim y$ if and only if $xy = yx$. This assignment can be lifted to a functor, and that functor has a left adjoint $\mathcal{S}$, which maps $X \in \mathsf{Comm}$ to the monoid of strings of X modulo the commuting relation of X. Each weak Kleene algebra can be seen as a monoid by keeping its multiplication operation and forgetting the rest of its structure. These yield forgetful functors from WKA, KA and KA^* to Mon that all have left adjoints.

For weak Kleene algebras and usual Kleene algebras, these left adjoints can be described by building terms over monoid elements, quotienting these terms by provable equalities between terms, and also by identifying the multiplicative structure of the free algebra with the monoid structure.

For $*$-continuous algebras, the left adjoint $\mathcal{R}$ is given by so-called *regular languages* [Koz97]. Explicitly, if X is a monoid, we can view the set $\mathcal{P}X$ as a $*$-continuous algebra by using the following operations:

$$\begin{aligned} 0 &\triangleq \varnothing \\ 1 &\triangleq \{1\} \\ A + B &\triangleq A \cup B \\ A \cdot B &\triangleq \{xy \mid x \in A, y \in B\} \\ A^* &\triangleq \bigcup_{n \in \mathbb{N}} A^n. \end{aligned}$$

The $*$-continuous algebra $\mathcal{R}X$ of regular sets over X is the smallest subalgebra of $\mathcal{P}X$ that contains the singletons. We write $l_X : \mathcal{T}X \to \mathcal{L}X$ for the unique algebra morphism that maps a symbol $x \in X$ to the singleton set $\{x\}$. We refer to this morphism as the *language interpretation* of $\mathcal{T}X$. By abuse of notation, we'll often identify Kleene algebra terms with their corresponding language interpretations; e.g. if e is a term, we'll write $X \subseteq e$ to mean $X \subseteq l(e)$.

To simplify the notation, we will never bother explicitly mentioning these right adjoints. In other words, we will always view $\mathcal{T}X$ as a monoid or as a commutable set by using the structure given by the right adjoints. We can also show that the unit

of all the involved adjunctions is an injection, so we'll treat $X \in \mathsf{Comm}$ as a subset of $\mathcal{S}X$, $\mathcal{T}X$, etc. For example, the unit $X \to \mathcal{L}X$ maps an element $x \in X$ to the singleton set $\{x\}$. By direct calculation, we can show that this unit factors through the unit $X \to \mathcal{T}X$, so this unit is an injection as well.

Theorem 3.3. *The following diagram commutes:*

$$\begin{array}{ccc} \mathcal{S}X & \hookrightarrow & \mathcal{T}X \\ & \searrow^{\{-\}} & \downarrow{\scriptstyle l} \\ & & \mathcal{L}X, \end{array}$$

where the diagonal arrow is the monoid morphism that sends $s \in \mathcal{S}X$ to $\{s\} \in \mathcal{L}X$.

Proof. A monoid morphism on $\mathcal{T}X$ is uniquely determined by its action on elements of X. Thus, we just have to note that both paths map $x \in X$ to $\{x\} \in \mathcal{L}X$. □

Theorem 3.4. *Let $s \in \mathcal{S}X$ be a string and $e \in \mathcal{T}X$ be an arbitrary term. The following conditions are equivalent:*

- $s \leq e$
- $s \in l(e)$.

Proof. Suppose that $s \leq e$. Then $s \in \{s\} = l_X(s) \subseteq l_X(e)$ by monotonicity (cf. Theorem 3.3). Conversely, if $s \in l(e)$, we proceed by induction on e. □

One useful fact about the fact that $\mathcal{L}$ has an adjoint is that, if Y is $*$-continuous, then every morphism of algebras $f : \mathcal{T}X \to Y$ can be factored through the language interpretation l:

$$\begin{array}{ccc} \mathcal{T}X & \xrightarrow{l} & \mathcal{L}X \\ & \searrow_{f} & \downarrow \\ & & Y. \end{array}$$

This has some pleasant consequences. For example, if $[-]_0 : \mathcal{T}X \to \mathbb{2}$ be the algebra morphism that maps every $x \in X$ to 0, then $[e]_0 = 1$ if and only if $1 \leq e$. Indeed, this morphism must factor through $\mathcal{L}X$. The corresponding factoring $\mathcal{L}X$ must map any nonempty string to 0 and the empty string to 1. Thus, $[e]_0 = 1$ if and only if $1 \in l(e)$, which is equivalent to $1 \leq e$.

Definition 3.5. A term $e \in \mathcal{T}X$ is *finite* if $l(e)$ is finite.

Theorem 3.6. *If $e \in \mathcal{T}X$ is finite, then $e = \sum l(e)$.*

Proof. By induction on e. We note that, if $l(e)$ is finite, then $l(e')$ is also finite for every immediate subterm e', which allows us to apply the relevant induction hypotheses. □

Corollary 3.7. *The language interpretation l is injective on finite terms: if $l(e_1) = l(e_2)$ is finite, then $e_1 = e_2$.*

Proof. We have $e_1 = \sum l(e_1) = \sum l(e_2) = e_2$. □

Corollary 3.8. *If $e \neq 0$, then there exists some string s such that $s \leq e$.*

Proof. Note that $l(e) \neq \varnothing$. Indeed, if $l(e) = \varnothing = l(0)$, then $e = 0$ by Corollary 3.7, which contradicts our hypothesis. Therefore, we can find some s such that $s \in l(e)$. But this is equivalent to $s \leq e$ by Theorem 3.4. □

Lemma 3.9. *$\mathcal{S}X$ is a cancellative monoid: if $ss_1 = ss_2$ or $s_1 s = s_2 s$, then $s_1 = s_2$.*

Proof. Suppose that $ss_1 = ss_2$; the other case follows analogously. We prove the result by induction on s. There are three cases to consider.

- If $s = 1$, the result follows immediately.
- If $s = s_1' s_2'$, then write

  $$ss_1 = s_1'(s_2' s_1)$$
  $$ss_2 = s_1'(s_2' s_2).$$

  The induction hypothesis applied to s_1' shows that $s_2' s_1 = s_2' s_2$. Then, another use of the induction hypothesis on s_2' allows us to conclude.
- Otherwise, s must be an element x of X. Thus, we need to show that $xs_1 = xs_2$ implies $s_1 = s_2$. TODO

□

As usual, if X is a (weak) Kleene algebra, we are going to view a finite set of elements A of X as the element $\sum_{a \in A} a \in X$.

Lemma 3.10. *Let X be a commutable set, and Y be a commutable subset of X. The following is a morphism of commutable sets:*

$$\pi_Y : X \to \mathcal{T}Y$$
$$\pi_Y = \begin{cases} c & \text{if } c \in Y \\ 1 & \text{otherwise.} \end{cases}$$

A similar result holds if we replace $\mathcal{T}$ with $\mathcal{S}$. When $X = X_l \oplus X_r$ and Y is one of the two summands, we'll use the notations π_l and π_r to refer to these projections.

Proof. We just need to show that the commuting relation is preserved. This follows because 1 commutes with anything in $\mathcal{T}Y$ or $\mathcal{S}Y$ and because the commuting relation in Y is inherited from X. □

Lemma 3.11. *Let X be a commutable set, and Y be a commutable subset of X. The canonical inclusion*

$$Y \hookrightarrow X \longrightarrow \mathcal{T}X,$$

gives rise to a morphism of algebras of type $\mathcal{T}Y \to \mathcal{T}X$. This morphism has a left inverse $\pi_Y : \mathcal{T}X \to \mathcal{T}Y$, given by lifting the morphism of the same name in Lemma 3.10.

Convention 3.12. *If X and Y are commutable sets, we lift the functions $(-)_l : X \to X \oplus Y$ and $(-)_r : Y \to X \oplus Y$ and view them as having types $\mathcal{T}X \to \mathcal{T}(X \oplus Y)$ and $\mathcal{T}Y \to \mathcal{T}(X \oplus Y)$. We'll also view them as having types $\mathcal{S}X \to \mathcal{S}(X \oplus Y)$ and $\mathcal{S}Y \to \mathcal{S}(X \oplus Y)$.*

If X is a commutable set, view a term $e \in \mathcal{T}X$ as an element $\mathcal{T}(X \oplus X)$ by mapping each symbol $x \in X$ in e to $x_l x_r$.

Any string over $X \oplus Y$ can be seen as a pair of strings over X and Y. More precisely, the monoids $\mathcal{S}(X \oplus Y)$ and $\mathcal{S}X \times \mathcal{S}Y$ are isomorphic via the mappings

$$\mathcal{S}(X \oplus Y) \ni s \mapsto (\pi_l(s), \pi_r(s)) \in \mathcal{S}X \times \mathcal{S}Y$$
$$\mathcal{S}X \times \mathcal{S}Y \ni (s_1, s_2) \mapsto (s_1)_l (s_2)_r \in \mathcal{S}(X \oplus Y).$$

4 Automata theory

Definition 4.1. Let $e \in \mathcal{T}X$ be a term, where X is finite. We say that e is *derivable* if there exists a family of terms $\{\delta_x(e)\}_{x \in X}$ such that $e = [e]_0 + \sum_x x\delta_x(e)$. We refer to the term $\delta_x(e)$ as the *derivative with respect to x*.

The family $\delta_x(e)$ is not necessarily unique. Nevertheless, we'll use the notation $\delta_x(e)$ to refer to specific derivatives of x when it is clear from the context which one we mean.

Lemma 4.2. *Derivable terms are closed under all the weak Kleene algebra operations, with the following caveats: for e^*, we also require that $[e]_0 = 0$; for $e_1 e_2$, the term is also derivable if e_2 isn't, provided that $[e_1]_0 = 0$. We have the following choices of derivatives:*

$$\begin{aligned}
\delta_x(0) &= 0 \\
\delta_x(1) &= 1 \\
\delta_x(x) &= 1 \\
\delta_x(y) &= 0 && \text{if } y \neq x \\
\delta_x(e_1 + e_2) &= \delta_x(e_1) + \delta_x(e_2) \\
\delta_x(e_1 e_2) &= [e_1]_0 \delta_x(e_2) + \delta_x(e_1) e_2 \\
\delta_x(e^*) &= \delta_x(e) e^*,
\end{aligned}$$

where, by abuse of notation, we treat $[e_1]_0 \delta_x(e_2)$ as 0 when e_2 is not necessarily derivable (since, by assumption, $[e_1]_0 = 0$ in that case).

Proof. We prove the closure property for products and star. For products, we start by expanding e_1:

$$\begin{aligned}
e_1 e_2 &= \left([e_1]_0 + \sum_x x\delta_x(e_1) \right) e_2 \\
&= [e_1]_0 e_2 + \sum_x x\delta_x(e_1) e_2.
\end{aligned}$$

If $[e_1]_0 = 0$, the first term gets canceled out, and we obtain $\sum_x x\delta_x(e_1)e_2 = [e_1]_0[e_2]_0 + \sum_x x\delta_x(e_1)e_2$. Otherwise, we know that e_2 is derivable, and we proceed as follows:

$$\begin{aligned} e_1e_2 &= [e_1]_0\left([e_2]_0 + \sum_x x\delta_x(e_2)\right) + \sum_x x\delta_x(e_1)e_2 \\ &= [e_1]_0[e_2]_0 + \sum_x [e_1]_0 x\delta_x(e_2) + \sum_x x\delta_x(e_1)e_2 \\ &= [e_1]_0[e_2]_0 + \sum_x x([e_1]_0\delta_x(e_2) + \delta_x(e_1)e_2) \qquad \text{(because } [e_1]_0 x = x[e_1]_0\text{),} \end{aligned}$$

which allows us to conclude.

For star, assuming that $[e]_0 = 0$, we note that $e^* = 1 + ee^*$, and we apply the closure properties for the other operations. □

Definition 4.3. Suppose that X is finite. A *finite-state automaton* is a finite set S of elements of $\mathcal{T}X$ (the *states*) that contains 1, is closed under finite sums and under taking derivatives (that is, every $e \in S$ is derivable, and each $\delta_x(e)$ is a state). We say that a term e is *finite state* if it if a finite sum of states of some finite-state automaton S. The *residue* of e, written $\rho(e)$, is the greatest element of S.

Remark 4.4. In the usual theory of Kleene algebra, regular expressions satisfy a *fundamental theorem* that guarantees that every term is finite state. Because we are working with weak Kleene algebra, however, not every term is finite state. For example, because we cannot show that 1^* is equal to 1, we cannot show that 1^* is finite state. Indeed, we can expand $1^* = 1 + 11^* = 1 + 1^*$, but since 1^* is not preceded by any symbol x, this is not a valid derivation of 1^*.

Lemma 4.5. *Suppose that X is finite. A* pre-automaton *is a finite set S of terms over X such that every $e \in S$ is derivable, and $\delta_x(e)$ is a sum of elements of $S \cup \{1\}$. The set $\bar{S} \triangleq \{\sum_{i=1}^n e_i \mid n \in \mathbb{N}, e \in (S \cup \{1\})^n\}$ is a finite-state automaton. We refer to $\bar{S}$ as the automaton generated by the pre-automaton S.*

Proof. It is easy to show that $\bar{S}$ is finite, contains 1 and is closed under finite sums. We just need to show that it is closed under taking derivatives. This follows from Lemma 4.2. □

Lemma 4.6. *Let S be an automaton and $S_0 \subseteq S$ be a subset of states. Let S' be the* set of states reachable from S_0*, which is the smallest set satisfying*

- $S_0 \subseteq S'$*;*
- *If $e \in S'$ and $\delta_x(e) \in S$ is some derivative, then $\delta_x(e) \in S'$.*

The set S' is a pre-automaton. We refer to its generated automaton as the automaton of states reachable from S_0.

Lemma 4.7. *Let X be a finite commutable set. Finite-state terms are preserved by all the weak Kleene algebra operations (for e^*, we additionally require that $[e]_0 = 0$). Moreover, the set of states of the corresponding automata can be effectively computed.*

Proof. Let's consider all the cases.

- The set $\{0, 1\}$ is an automaton by Lemma 4.2. Therefore, 0 and 1 are finite state.
- By Lemma 4.2, if x is a symbol, the set $S = \{x\}$ is a pre-automaton. Therefore, x is finite state because it belongs to the automaton $\bar{S}$.
- Suppose that S_1 and S_2 are finite automata. By Lemma 4.2, the set $S = \{e_1 + e_2 \mid e_1 \in S_1, e_2 \in S_2\}$ is a pre-automaton. Therefore, if we have finite-state terms e_1 and e_2 of S_1 and S_2, their sum $e_1 + e_2$ is finite state because it belongs to the automaton $\bar{S}$.
- Suppose that S_1 and S_2 are finite automata. By Lemma 4.2, the set $S = \{e_1 e_2 \mid e_1 \in S_1, e_2 \in S_2\}$ is a pre-automaton. Indeed, $\delta_x(e_1 e_2) = [e_1]_0 \delta_x(e_2) + \delta_x(e_1) e_2$ is a sum of elements of S, since

$$\begin{aligned} [e_1]_0 &\in S_1 \\ \delta_x(e_2) &\in S_2 \\ \delta_x(e_1) &\in S_1 \\ e_2 &\in S_2. \end{aligned}$$

  Therefore, if we have finite-state terms e_1 and e_2 of S_1 and S_2, their product $e_1 e_2$ is finite state because it belongs to the automaton $\bar{S}$.
- Suppose that e is a state of some automaton S such that $[e]_0 = 0$. Define $S' = \{e' e^* \mid e' \in S\}$. By Lemma 4.2, this set is a pre-automaton. Indeed,

$$\begin{aligned} \delta_x(e' e^*) &= [e']_0 \delta_x(e^*) + \delta_x(e') e^* \\ &= [e']_0 \delta_x(e) e^* + \delta_x(e') e^* \\ &= ([e']_0 \delta_x(e) + \delta_x(e')) e^*. \end{aligned}$$

  The terms $\delta_x(e)$ and $\delta_x(e')$ are in S. Thus, $[e']_0 \delta_x(e) \in S$ and $\delta_x(e' e^*)$ is a sum of terms of S'. Since $e^* = 1 e^*$ is an element of S', then it is a state of $\bar{S}'$, and e^* is finite state.

□

Lemma 4.8. *Let $e \in \mathcal{T}X$ be a state of a finite-state automaton S, and $k \in \mathbb{N}$. We can write*

$$e = \sum_{\substack{s \in \mathcal{S}X \\ s \leq e \\ |s| < k}} s + \sum_{\substack{s \in \mathcal{S}X \\ |s| = k}} s e_s,$$

where each $e_s \in S$ for all s, and the size $|s| \in \mathbb{N}$ of a string s is defined by mapping every symbol of s to $1 \in \mathbb{N}$.

Proof. By induction on k. When $k = 0$, the equation is equivalent to $e = e$, and we are done. Otherwise, suppose that the result is valid for k. We need to prove that it is also valid for $k + 1$. Write

$$e = \sum_{\substack{s \in \mathcal{S}X \\ s \leq e \\ |s| < k}} s + \sum_{\substack{s \in \mathcal{S}X \\ |s| = k}} s e_s.$$

By deriving each e_s, we can rewrite this as

$$\begin{aligned} e &= \sum_{\substack{s \in \mathcal{S}X \\ s \leq e \\ |s| < k}} s + \sum_{\substack{s \in \mathcal{S}X \\ |s| = k}} s \left([e_s]_0 + \sum_{x \in X} x \delta_x(e_s) \right) \\ &= \sum_{\substack{s \in \mathcal{S}X \\ s \leq e \\ |s| < k}} s + \sum_{\substack{s \in \mathcal{S}X \\ |s| = k}} s[e_s]_0 + \sum_{\substack{s \in \mathcal{S}X \\ |s| = k}} \sum_{x \in X} s x \delta_x(e_s). \end{aligned} \tag{1}$$

We can see that $[e_s]_0 = 1$ if and only if $s \leq e$: by taking the language interpretation of (1), we can see that a string of size k can only belong to the middle term, since the left and right terms can only account for strings of strictly smaller or larger size, respectively. Thus, we can rewrite (1) as

$$\begin{aligned} e &= \sum_{\substack{s \in \mathcal{S}X \\ s \leq e \\ |s| < k}} s + \sum_{\substack{s \in \mathcal{S}X \\ |s| = k \\ s \leq e}} s[e_s]_0 + \sum_{s, |s| = k} \sum_{x \in X} s x \delta_x(e_s) \\ &= \sum_{\substack{s \in \mathcal{S}X \\ s \leq e \\ |s| < k+1}} s + \sum_{\substack{s \in \mathcal{S}X \\ |s| = k}} \sum_{x \in X} s x \delta_x(e_s). \end{aligned} \tag{2}$$

Given some string s with $|s| = k + 1$, define

$$e'_s \triangleq \sum_{\substack{(s', x) \in \mathcal{S}X \times X \\ s = s'x}} \delta_x(e_{s'}).$$

This sum is well defined because there are only finitely many s' and $x \in X$ such that $s = s'x$: s' must be of size k, and there are only finitely many such strings. Moreover, e'_s is an element of S, since S is closed under taking derivatives and finite sums. We have

$$\begin{aligned} s e'_s &= \sum_{\substack{(s', x) \\ |s'| = k \\ s = s'x}} s \delta_x(e_{s'}) \\ &= \sum_{\substack{(s', x) \\ |s'| = k \\ s = s'x}} s' x \delta_x(e_{s'}). \end{aligned}$$

Therefore,

$$\begin{aligned}\sum_{\substack{s\\|s|=k+1}} se'_s &= \sum_{\substack{s\\|s|=k+1}} \sum_{\substack{(s',x)\\|s'|=k\\s=s'x}} s'x\delta_x(e_{s'}) \\ &= \sum_{\substack{(s',x)\\|s'|=k}} s'x\delta_x(e_{s'}) \\ &= \sum_{\substack{s'\\|s'|=k}} \sum_{x\in X} s'x\delta_x(e_{s'}).\end{aligned}$$

Putting everything together, (2) becomes

$$e = \sum_{s\le e, |s|<k+1} s + \sum_{\substack{s\\|s|=k+1}} se'_s, \tag{3}$$

which completes the inductive case. □

5 Representing Relations

We are going to develop a framework for representing relations on strings as Kleene algebra terms. More specifically, if Σ is a finite, discrete commutable set, consider the commutable set $\ddot{\Sigma} = \Sigma \oplus \Sigma$. A string $s \in \mathcal{S}\ddot{\Sigma}$ can be seen as a pair of strings s_1 and s_2 over Σ: one corresponds to the left symbols, the other one corresponds to the right symbols. Thus, via the language interpretation l, we can view an expression $e \in \mathcal{T}\ddot{\Sigma}$ as a set of pairs of strings over Σ—in other words, a relation on strings. Given a term $e \in \mathcal{T}\ddot{\Sigma}$, we will write $s \xrightarrow{e} s'$ to say that $s_l s'_r \le e$ (or, equivalently, that $s_l s'_r \in l(e)$). We write $s \xrightarrow{e}{}^{n} s'$ to denote that s' can be reached from s via e in exactly n steps, and $s \xrightarrow{e}{}^{<n} s'$ when they are related some number of steps below n. We write $s \xrightarrow{e}{}^{*} s'$ for the reflexive-transitive closure of $\xrightarrow{e}$.

Definition 5.1. Let $e \in \mathcal{T}(X \oplus Y)$ be a term. We say that e has *bounded output* if there exists some $k \in \mathbb{N}$ (the *fanout*) such that, for every string $s \le e$, $|\pi_r(s)| \le (|\pi_l(s)|+1)k$.

Lemma 5.2. *If e has fanout k and $k' \ge k$, then e has fanout k.*

Lemma 5.3. *Bounded-output terms are closed under all the weak Kleene algebra operations. For e^*, we additionally require that $|\pi_l(s)| \ge 1$ for all strings $s \le e$.*

Proof. Let's focus on the last point. Suppose that e has fanout k and that $|\pi_l(s)| \ge 1$ for every $s \le e$. We are going to show that e^* has bounded output with fanout $2k$.

Suppose that $s \le e^*$. We can write $s = s_1 \cdots s_n$ such that $s_i \le e$ for every $i \in \{1, \ldots, n\}$. We have, for every $i \in \{1, \ldots, n\}$,

$$|\pi_r(s_i)| \le (|\pi_l(s_i)| + 1)k.$$

Thus,

$$\begin{aligned}
|\pi_r(s)| &= \sum_{i=1}^{n} |\pi_r(s_i)| \\
&\le \sum_{i=1}^{n} (|\pi_l(s_i)| + 1)k \\
&\le \sum_{i=1}^{n} 2|\pi_l(s_i)|k && \text{(because } |\pi_l(s_i)| \ge 1\text{)} \\
&= \left(\sum_{i=1}^{n} |\pi_l(s_i)|\right) 2k \\
&= |\pi_l(s_0)\cdots\pi_l(s_n)|2k \\
&= |\pi_l(s_0\cdots s_n)|2k \\
&= |\pi_l(s)|2k \\
&\le (|\pi_l(s)| + 1)2k.
\end{aligned}$$

□

Lemma 5.4. *Let $e \in \mathcal{T}\ddot{\Sigma}$ be a bounded-output term that is the state of some automaton S. There exists some $k \in \mathbb{N}$ such that e has fanout k and such that, for every $n \in \mathbb{N}$, we can write*

$$e = \sum_{\substack{s \in SX \\ |s|<n \\ s \le e}} s + \sum_{\substack{s \in SX \\ |s|=n \\ |\pi_r(s)| \le (|\pi_l(s)|+1)k}} se_s,$$

where $e_s \in S$ for every s.

Proof. Let k_0 be the fanout of e. For each $e' \in S$ such that $e' \neq 0$, choose some string $w_{e'} \le e'$. Define $m \triangleq \max\{|\pi_l(w_{e'})| \mid e' \in S, e' \neq 0\}$ and $k \triangleq (m+1)k_0$. Since $k \ge k_0$, we know that e has fanout k. Moreover, by Lemma 4.8, we have

$$\begin{aligned}
e &= \sum_{\substack{s \le e \\ |s|<n}} s + \sum_{\substack{s \in SX \\ |s|=n}} se_s \\
&= \sum_{\substack{s \le e \\ |s|<n}} s + \sum_{\substack{s \in SX \\ |s|=n \\ e_s \neq 0}} se_s,
\end{aligned}$$

where each e_s is a state of S. If s is such that $|s| = n$ and $e_s \neq 0$, we have $sw_{e_s} \le e$. Therefore,

$$\begin{aligned}
|\pi_r(s)| &\le |\pi_r(sw_{e_s})| \\
&\le (|\pi_l(sw_{e_s})| + 1)k_0 \\
&= (|\pi_l(s)| + |\pi_l(w_{e_s})| + 1)k_0 \\
&\le (|\pi_l(s)| + m + 1)k_0 \\
&\le (|\pi_l(s)| + 1)(m+1)k_0 \\
&= (|\pi_l(s)| + 1)k.
\end{aligned}$$

Thus,

$$\begin{aligned} e &= \sum_{\substack{s \le e \\ |s|<n}} s + \sum_{\substack{s \\ |s|=n \\ e_s \neq 0 \\ |\pi_r(s)| \le (|\pi_l(s)|+1)k}} s e_s \\ &= \sum_{\substack{s \le e \\ |s|<n}} s + \sum_{\substack{s \\ |s|=n \\ |\pi_r(s)| \le (|\pi_l(s)|+1)k}} s e_s. \end{aligned}$$

□

Definition 5.5. A term L over Σ is *prefix free* if for all strings $s_1 \le L$ and $s_2 \le L$, if s_1 is a prefix of s_2, then $s_1 = s_2$.

Lemma 5.6. *Let s and s' be two strings over Σ such that one is not a prefix of the other, or vice versa. Then we can write $s = s_0 x s_1$ and $s' = s_0 x' s_1'$ with $x \neq x'$.*

Lemma 5.7. *Consider strings s over Σ and s' over $\ddot{\Sigma}$ such that $\pi_l(s')$ is not a prefix of s, or vice versa. Then $s_r s' \le \Sigma^* \Sigma^{\neq} \Sigma_l^* \Sigma_r^*$, where $\Sigma^{\neq} = \sum_{x,y \in \Sigma, x \neq y} x_l y_r$.*

Lemma 5.8. *Let s and s' be strings over Σ, p be a string over $\ddot{\Sigma}$, and e be a term over $\ddot{\Sigma}$. Suppose that s and s' belong to some prefix free L and that $sp \le s'e$. Then $s = s'$ and $p \le e$.*

Proof. By interpreting the inequality over languages, we see that sp must be of the form $s'p'$ for some $p' \le e$. Thus, $s\pi_l(p) = \pi_l(sp) = \pi_l(s'p') = s'\pi_l(p')$. This is an equality between strings over a discrete alphabet, so either s is a prefix of s', or the other way around. But L is prefix free, so we must have $s = s'$. Moreover, string concatenation is a cancellative operation, so we must have $\pi_l(p) = \pi_l(p')$. An analogous reasoning shows that $\pi_r(p) = \pi_r(p')$. We conclude by noting that $p = \pi_l(p)_l \pi_r(p)_r = \pi_l(p')_l \pi_r(p')_r = p'$. □

Definition 5.9. Let L_1 and L_2 be terms over Σ. A term $e \in F\ddot{\Sigma}$ is a *representable relation* of type $L_1 \to L_2$ if it is finite state, has bounded output, $\pi_l(e) \le L_1$ and $\pi_r(e) \le L_2$. We write $e : L_1 \to L_2$ to denote the type of e.

Lemma 5.10. *If $e : L_1 \to L_2$ is representable, $L_1 \le L_1'$ and $L_2 \le L_2'$, then e is also of type $L_1' \to L_2'$.*

Definition 5.11. Let $e : L_1 \to L_2$ be representable, and Λ be some set of strings. The image of Λ by e is the set $\mathsf{Next}_e(\Lambda) \subseteq L_2$ defined as follows:

$$\mathsf{Next}_e(\Lambda) \triangleq \bigcup_{s \in \Lambda} \{s' \le L_2 \mid s \xrightarrow{e} s'\}.$$

By abuse of notation, if s is a string, we write $\mathsf{Next}_e(s)$ to mean $\mathsf{Next}_e(\{s\})$. When $n \in \mathbb{N}$, we write $\mathsf{Next}_e^n(\Lambda)$ for the iterates. We also write $\mathsf{Next}_e^{<n}(\Lambda)$ for the set $\bigcup_{0 \le i < n} \mathsf{Next}_e^i(\Lambda)$, and similarly for $\mathsf{Next}_e^{\le n}(\Lambda)$.

Lemma 5.12. *If $e : L_1 \to L_2$ is representable and $\mathsf{Next}_e(s) \neq 0$, then $s \leq L_1$.*

Lemma 5.13. *If $e : L_1 \to L_2$ is representable and $s \xrightarrow{e} s'$, then $s \leq L_1$ and $s' \leq L_2$. Thus, $s' \in \mathsf{Next}_e(s)$.*

Proof. We have, by definition $s_l s'_r \leq e$, and thus $s = \pi_l(s_l s'_r) \leq \pi_l(e) \leq L_1$. An analogous reasoning yields $s' \leq L_2$. We conclude by expanding the definition of $\mathsf{Next}_e(s)$. □

Lemma 5.14. *Let $e : L_1 \to L_2$ be representable with fanout k and let Λ be finite. If $s \in \mathsf{Next}_e(\Lambda)$, then $|s| \leq (m+1)k$, where $m = \max\{|s'| \mid s' \in \Lambda\}$. Thus, the sets $\mathsf{Next}_e^n(\Lambda)$ and $\mathsf{Next}_e^{\leq n}(\Lambda)$ are finite for every $n \in \mathbb{N}$.*

Proof. The last observation follows from the bound because Σ is finite. To prove the first part, note that, if $s \in \mathsf{Next}_e(\Lambda)$, by definition, there exists $s' \in \Lambda$ such that $s'_l s_r \leq e$. Since e has fanout k, we have

$$|s| = |\pi_r(s'_l s_r)| \leq (|\pi_l(s'_l s_r)| + 1)k = (|s| + 1)k \leq (n+1)k.$$

□

In the following, we are going to fix some representable relation $e : L_1 \to L_2$ where $L_1 \leq L_2$ and L_2 is prefix free (which implies that L_1 is prefix free as well). By Lemma 5.10, we can also view it as a relation of type $e : L_2 \to L_2$.

Lemma 5.15. *If $s \in L_2$, we have*

$$s\,\mathsf{Next}_e(s) \leq s_r e \leq s\,\mathsf{Next}_e(s) + \Sigma^*\Sigma^{\neq}\Sigma_l^*\Sigma_r^*\rho(e),$$

where $\rho(e)$ is the residue of e; cf. Definition 4.3.

Proof. Let k be the constant of Lemma 5.4 for e, $n = |s|$, and let $p = (k+1)(n+1)$. Let

$$\Lambda \triangleq \{s' \in M\ddot{\Sigma} \mid |s'| = p+1, |\pi_r(s')| \leq (|\pi_l(s')|+1)k\}.$$

By applying Lemma 5.4 to e, we can write

$$\begin{aligned} e &= \sum_{\substack{s' \leq e \\ |s'|<p+1}} s' + \sum_{s' \in \Lambda} s' e_{s'} \\ &= \sum_{s' \leq e, |s'| \leq p} s' + \sum_{s' \in \Lambda} s' e_{s'} \\ &= \sum_{\substack{s' \leq e \\ |s'| \leq p \\ \pi_l(s') = s}} s' + \sum_{\substack{s' \leq e \\ |s'| \leq p \\ \pi_l(s') \neq s}} s' + \sum_{s' \in \Lambda} s' e_{s'}, \end{aligned}$$

Thus, to prove the inequality, it suffices to prove

$$s_r \sum_{\substack{s' \le e \\ |s'| \le p \\ \pi_l(s') = s}} s' = s\,\mathsf{Next}_e(s)_r \tag{4}$$

$$s_r \sum_{\substack{s' \le e \\ |s'| \le p \\ \pi_l(s') \neq s}} s' \le \Sigma^*\Sigma^{\neq}\Sigma_l^*\Sigma_r^*\rho(e) \tag{5}$$

$$s_r \sum_{s' \in \Lambda} s' e'_s \le \Sigma^*\Sigma^{\neq}\Sigma_l^*\Sigma_r^*\rho(e). \tag{6}$$

Let us start with (4). Notice that, for any string s' over $\ddot{\Sigma}$, we have $s' = \pi_l(s')_l \pi_r(s')_r$. Therefore, there is a bijection between the set of indices s' of the sum and the set of strings $\mathsf{Next}_e(s)$. The bijection is given by

$$\begin{aligned} s' &\mapsto \pi_r(s') \in \mathsf{Next}_e(s) \\ \mathsf{Next}_e(s) \ni s' &\mapsto s_l s'_r. \end{aligned}$$

To prove that this is a bijection, we must show that the inverse produces indeed a valid index. Notice that, if $s' \in \mathsf{Next}_e(s)$, by Lemma 5.14, we have $|s'| \le (n+1)k$, and thus $|s_l s'_r| = |s| + |s'| \le (n+1)(k+1) = p$.

By reindexing the sum in (4) with this bijection, we have

$$\begin{aligned} s_r \sum_{\substack{s' \le e \\ |s'| \le p \\ \pi_l(s') = s}} s' &= s_r \sum_{s' \in \mathsf{Next}_e(s)} s_l s'_r \\ &= s_r s_l \sum_{s' \in \mathsf{Next}_e(s)} s'_r \\ &= s_r s_l \left(\sum_{s' \in \mathsf{Next}_e(s)} s' \right)_r \\ &= s\,\mathsf{Next}_e(s)_r. \end{aligned}$$

Next, let us look at (5). Suppose that s' is such that $s' \le e$ and $\pi_l(s') \neq s$. Since L_1 is prefix free, and $\pi_l(s') \le L_1$, Lemma 5.7 applied to s and s' yields

$$s_l s' \le \Sigma^*\Sigma^{\neq}\Sigma_l^*\Sigma_r^* \le \Sigma^*\Sigma^{\neq}\Sigma_l^*\Sigma_r^*\rho(e).$$

Summing over all such s', we get the desired inequality.

To conclude, we must show (6). By distributivity, this is equivalent to showing that, for every $s' \in \Lambda$,

$$s_r s' e_{s'} \le \Sigma^*\Sigma^{\neq}\Sigma_l^*\Sigma_r^*\rho(e).$$

If $e_{s'} = 0$, we are done. Otherwise, by Corollary 3.8, we can find some string $s'' \le e_{s'}$. We have $s's'' \le s'e_{s'} \le e$.

Note that we must have $|\pi_l(s')| > n$. Indeed, suppose that $|\pi_l(s')| \leq n$. Since $s' \in \Lambda$, we have

$$\begin{aligned}
|s'| &= |\pi_l(s')| + |\pi_r(s')| \\
&\leq |\pi_l(s')| + (|\pi_l(s')| + 1)k \\
&\leq (|\pi_l(s')| + 1)(k+1) \\
&\leq (n+1)(k+1) \\
&< p + 1 \\
&= |s'|,
\end{aligned}$$

which is a contradiction.

Since $\pi_l(s's'') \leq \pi_l(e) \leq L_1$ and L_1 is prefix free, by Lemma 5.6, we can write $s = s_0 x s_1$ and $\pi_l(s's'') = \pi_l(s')\pi_l(s'') = s_0 x' s_1'$, with $x \neq x'$. But $|\pi_l(s')| > n = |s|$ and $|s_0| < |s|$, thus $\pi_l(s')$ must be of the form $s_0 x' s_2'$. By Lemma 5.7, we find that $s_r s' = s_r \pi_l(s')\pi_r(s') \leq \Sigma^*\Sigma^{\neq}\Sigma_l^*\Sigma_r^*$, and thus

$$s_r s' e_{s'} \leq \Sigma^*\Sigma^{\neq}\Sigma_l^*\Sigma_r^* e_{s'} \leq \Sigma^*\Sigma^{\neq}\Sigma_l^*\Sigma_r^*\rho(e).$$

□

We are now going to use this theory to reason about the reflexive-transitive closure of a relation.

Definition 5.16. Let Λ be a set of strings, and $n \in \mathbb{N}$. We define $\mathsf{Path}_e^n(\Lambda) \in L_2^{n+1}$, the set of e-paths of length n starting at Λ, as follows:

$$\begin{aligned}
\mathsf{Path}_e^0(\Lambda) &\triangleq \bigcup_{s \in \Lambda} \{(s)\} \\
\mathsf{Path}_e^{i+1}(\Lambda) &\triangleq \bigcup_{s \in \Lambda} \{(s, p) \mid p \in \mathsf{Path}_e^i(\mathsf{Next}_e(s))\}.
\end{aligned}$$

By abuse of notation, if s is a string, we write $\mathsf{Path}_e^n(s)$ to denote $\mathsf{Path}_e^n(\{s\})$. We will write $\mathsf{Path}_e^{<n}(\Lambda)$ for the set $\bigcup_{0 \leq i < n} \mathsf{Path}_e^i(\Lambda)$, and similarly for $\mathsf{Path}_e^{\leq n}(\Lambda)$.

Lemma 5.17. *If Λ is finite, then $\mathsf{Path}_e^n(\Lambda)$ (and thus $\mathsf{Path}_e^{<n}(\Lambda)$) are finite.*

Proof. By induction on n, noting that $\mathsf{Next}_e(s)$ is also finite by Lemma 5.14. □

Convention 5.18. *We view a path $(s_0, \ldots, s_n)$ as a string over $\ddot{\Sigma}$ by mapping it to $\left(\prod_{0 \leq j < n} s_j\right)(s_n)_r$. By analogy, we view a set of paths such as $\mathsf{Path}_e^n(\Lambda)$ as a set of strings over $\ddot{\Sigma}$.*

Lemma 5.19. *The following properties hold of paths.*

1. $\mathsf{Path}_e^i(\varnothing) = \varnothing$.
2. $\mathsf{Path}_e^i(\Lambda) = \bigcup_{s \in \Lambda} \mathsf{Path}_e^i(s)$.

3. *Seen as a term,* $\mathsf{Path}_e^0(\Lambda) = \Lambda_r$.

4. *Seen as a term,* $\mathsf{Path}_e^{i+1}(\Lambda) = \sum_{s\in\Lambda} s\,\mathsf{Path}_e^i(\mathsf{Next}_e(s))$.

5. *If* $\mathsf{Path}_e^i(\Lambda) = \varnothing$, *then* $\mathsf{Path}_e^j(\Lambda) = \varnothing$ *for* $j \geq i$.

6. $\mathsf{Path}_e^{<n+1}(\Lambda) = \mathsf{Path}_e^0(\Lambda) + \sum_{s\in\Lambda} s\,\mathsf{Path}_e^{<n}(\mathsf{Next}_e(s))$.

Proof. We just show the last item. We have

$$\begin{aligned}
&\mathsf{Path}_e^{<n+1}(\Lambda)\\
&= \mathsf{Path}_e^0(\Lambda) + \sum_{i<n} \mathsf{Path}_e^{i+1}(\Lambda)\\
&= \mathsf{Path}_e^0(\Lambda) + \sum_{i<n}\sum_{s\in\Lambda} s\,\mathsf{Path}_e^i(\mathsf{Next}_e(s))\\
&= \mathsf{Path}_e^0(\Lambda) + \sum_{s\in\Lambda} s\sum_{i<n} \mathsf{Path}_e^i(\mathsf{Next}_e(s))\\
&= \mathsf{Path}_e^0(\Lambda) + \sum_{s\in\Lambda} s\,\mathsf{Path}_e^{<n}(\mathsf{Next}_e(s)).
\end{aligned}$$

□

Lemma 5.20. *Let* Λ *be some finite set of strings. We have the following inequalities:*

$$\begin{aligned}
\mathsf{Path}_e^n(\Lambda) &\leq L_1^n\,\mathsf{Next}_e^n(\Lambda)_r \leq L_1^*\,\mathsf{Next}_e^n(\Lambda)_r\\
\mathsf{Path}_e^{<n}(\Lambda) &\leq L_1^*\,\mathsf{Next}_e^{<n}(\Lambda)_r.
\end{aligned}$$

Proof. We prove the first one by induction on n. If $n = 0$, then

$$\begin{aligned}
\mathsf{Path}_e^0(\Lambda) &= \Lambda_r\\
&= L_1^0\Lambda_r\\
&= L_1^0\,\mathsf{Next}_e^0(\Lambda)_r.
\end{aligned}$$

If the result holds for some arbitrary n, then

$$
\begin{aligned}
&\mathsf{Path}_e^{n+1}(\Lambda) \\
&= \sum_{s\in\Lambda} s\,\mathsf{Path}_e^n(\mathsf{Next}_e(s)) \\
&= \sum_{\substack{s\in\Lambda \\ \mathsf{Next}_e(s)=\emptyset}} s\,\mathsf{Path}_e^n(\mathsf{Next}_e(s)) + \sum_{\substack{s\in\Lambda \\ \mathsf{Next}_e(s)\neq\emptyset}} s\,\mathsf{Path}_e^n(\mathsf{Next}_e(s)) \\
&= \sum_{\substack{s\in\Lambda \\ \mathsf{Next}_e(s)\neq\emptyset}} s\,\mathsf{Path}_e^n(\mathsf{Next}_e(s)) && \text{by Lemma 5.19} \\
&\leq \sum_{\substack{s\in\Lambda \\ s\leq L_1}} s\,\mathsf{Path}_e^n(\mathsf{Next}_e(s)) && \text{by Lemma 5.12} \\
&\leq \sum_{\substack{s\in\Lambda \\ s\leq L_1}} L_1\,\mathsf{Path}_e^n(\mathsf{Next}_e(s)) \\
&\leq \sum_{s\in\Lambda} L_1\,\mathsf{Path}_e^n(\mathsf{Next}_e(s)) \\
&\leq \sum_{s\in\Lambda} L_1 L_1^n\,\mathsf{Next}_e^n(\mathsf{Next}_e(s)) && \text{by the I.H.} \\
&= \sum_{s\in\Lambda} L_1^{n+1}\,\mathsf{Next}_e^{n+1}(s) \\
&= L_1^{n+1}\,\mathsf{Next}_e^{n+1}(\Lambda).
\end{aligned}
$$

The other results follow by using $L_1^n \leq L_1^*$ and by iterating over the sums. □

Lemma 5.21. *Consider some arbitrary $p \in \mathsf{Path}_e^n(\Lambda)$ with $\Lambda \leq L_2$.*

1. *For any term e' and any $m > n$, we cannot have $p \leq \mathsf{Path}_e^m(\Lambda)e'$.*
2. *If $p \leq (L_1)^*(L_f)_r$, then the last element of p is in L_f.*
3. *For any term e', we cannot have $p \leq \Sigma^*\Sigma^{\neq}e'$.*

Proof. 1. We proceed by induction on n.

- If $n = 0$, then p is of the form s_r for some $s \leq \Lambda$. Since $m > n \geq 0$, by Lemma 5.19, we have

$$\mathsf{Path}_e^m(\Lambda)e' = \sum_{s'\in\Lambda} s'\,\mathsf{Path}_e^{m-1}(\mathsf{Next}_e(s'))e'.$$

  Since Λ is prefix free, these s' cannot be empty, and thus the language corresponding to right-hand side only contains strings that have at least one left symbol. This is not the case for p, so the inequality cannot hold.
- Otherwise, suppose that the result holds for n, and we want to prove it for $n + 1$. The path p seen as a string must be of the form sp', where

$s \in \Lambda$ is some string, and $p \leq \mathsf{Path}_e^n(\mathsf{Next}_e(s))$. Since $m > n+1 \geq 0$, by Lemma 5.19, we have

$$\mathsf{Path}_e^m(\Lambda)e' = \sum_{s' \in \Lambda} s' \, \mathsf{Path}_e^{m-1}(\mathsf{Next}_e(s'))e'.$$

Suppose that the inequality holds. Thus, there exists some $s' \in \Lambda$ such that sp belongs to the language interpretation of $s' \, \mathsf{Path}_e^{m-1}(\mathsf{Next}_e(s'))e'$—or, equivalently, $sp \leq s' \, \mathsf{Path}_e^{m-1}(\mathsf{Next}_e(s')e'$. By Lemma 5.8, we find that $p \leq \mathsf{Path}_e^{m-1}(\mathsf{Next}_e(s'))e'$, which contradicts the induction hypothesis.

2. If $p = (s_0, \ldots, s_n)$, with $s_i \leq L_2$ for every $i \leq n$, then the representation of p as a string is $\left(\prod_{i<n} s_i\right)(s_n)_r$. On the other hand, if $p \leq (L_1)^*(L_f)_r$, by taking language interpretations on both sides, we can write it as $\left(\prod_{i<m} s_i'\right)(s_m')_r$, with $s_i' \leq L_1$ for every $i < m$ and, additionally, $s_m' \leq L_f$. Thus, it suffices to show that $s_n = s_m'$, and thus that $(s_n)_r = (s_m')_r$. By projecting on both sides, we find that

$$\begin{aligned}\left(\prod_{i<n} s_i\right) &= \pi_l\left(\left(\prod_{i<n} s_i\right)(s_n)_r\right) \\ &= \pi_l\left(\left(\prod_{i<m} s_i'\right)(s_m')_r\right) \\ &= \left(\prod_{i<m} s_i'\right).\end{aligned}$$

This implies

$$\left(\prod_{i<n} s_i\right)(s_n)_r = \left(\prod_{i<n} s_i\right)(s_m')_r,$$

which yields the sought result by cancellation (Lemma 3.9).

3. Seen as a string, the path p is of the form ss_r' for two strings strings s and s' over Σ. Thus, we can write $ss_r' = s_1 x_l y_r s_2$, where $x \neq y$ are elements of Σ, $s_1 \leq \Sigma^*$ and $s_2 \leq e'$. By unfolding the language interpretation of Σ^*, we see that s_1 must be the image of some string s_1' in Σ. At this point, we can conclude by induction on the size of s.

□

Lemma 5.22. *For every $n \in \mathbb{N}$ and every finite $\Lambda \leq L_2$, we have the inequality*

$$\mathsf{Path}_e^{<n}(\Lambda) \leq \Lambda_r e^* \leq \mathsf{Path}_e^{<n}(\Lambda) + \mathsf{Path}_e^n(\Lambda)e^* + \varepsilon,$$

where $\varepsilon \triangleq \Sigma^*\Sigma^{\neq}\Sigma_l^*\Sigma_r^*\rho(e)e^*$ can be effectively computed.

Proof. By induction on n. If $n = 0$, then the goal becomes, by Lemma 5.19,

$$0 \leq \Lambda_r e^* \leq \mathsf{Path}_e^0(\Lambda)e^* + \varepsilon,$$

which holds because $\mathsf{Path}_e^0(\Lambda) = \Lambda_r$.

Otherwise, for the inductive step, suppose that the goal is valid for n. We need to prove that it is valid for $n+1$. Recall that $\mathsf{Next}_e(s) \le L_2$ for every s. Now we compute

$$\begin{aligned}
&\mathsf{Path}_e^{<n+1}(\Lambda) \\
&= \mathsf{Path}_e^0(\Lambda) + \sum_{s\in\Lambda} s\,\mathsf{Path}_e^{<n}(\mathsf{Next}_e(s)) && \text{by Lemma 5.19} \\
&\le \mathsf{Path}_e^0(\Lambda) + \sum_{s\in\Lambda} s\,\mathsf{Next}_e(s)_r e^* && \text{by the I.H.} \\
&\le \mathsf{Path}_e^0(\Lambda) + \sum_{s\in\Lambda} s_r e e^* && \text{by Lemma 5.15} \\
&= \Lambda_r + \sum_{s\in\Lambda} s_r e e^* \\
&= \Lambda_r + \Lambda_r e e^* \\
&= \Lambda_r e^*.
\end{aligned}$$

This shows the first inequality. As for the second inequality, by Lemma 5.15, we have

$$\begin{aligned}
\Lambda_r e^* &= \Lambda_r + \sum_{s\in\Lambda} s_r e e^* \\
&\le \Lambda_r + \sum_{s\in\Lambda} (s\,\mathsf{Next}_e(s)_r + \Sigma^*\Sigma^{\neq}\Sigma_l^*\Sigma_r^*\rho(e))e^* \\
&\le \Lambda_r + \sum_{s\in\Lambda} s\,\mathsf{Next}_e(s)_r e^* + \Sigma^*\Sigma^{\neq}\Sigma_l^*\Sigma_r^*\rho(e)e^* \\
&\le \Lambda_r + \sum_{s\in\Lambda} s\,\mathsf{Next}_e(s)_r e^* + \varepsilon \\
&= \mathsf{Path}_e^0(\Lambda) + \sum_{s\in\Lambda} s\,\mathsf{Next}_e(s)_r e^* + \varepsilon.
\end{aligned}$$

Let us focus on the middle term. We have

$$\begin{aligned}
&\sum_{s\in\Lambda} s\,\mathsf{Next}_e(s)_r e^* \\
&\le \sum_{s\in\Lambda} s\left[\mathsf{Path}_e^{<n}(\mathsf{Next}_e(s)) + \mathsf{Path}_e^n(\mathsf{Next}_e(s))e^* + \varepsilon\right] && \text{by the I.H.} \\
&= \sum_{s\in\Lambda} s\,\mathsf{Path}_e^{<n}(\mathsf{Next}_e(s)) + \sum_{s\in\Lambda} s\,\mathsf{Path}_e^n(\mathsf{Next}_e(s))e^* + \varepsilon \\
&= \sum_{s\in\Lambda} s\,\mathsf{Path}_e^{<n}(\mathsf{Next}_e(s)) + \mathsf{Path}_e^{n+1}(\Lambda)e^* + \varepsilon && \text{by Lemma 5.19.}
\end{aligned}$$

Thus,

$$\begin{aligned}
&\Lambda_r e^* \\
&\le \mathsf{Path}_e^0(\Lambda) + \sum_{s\in\Lambda} s\,\mathsf{Path}_e^{<n}(\mathsf{Next}_e(s)) + \mathsf{Path}_e^{n+1}(\Lambda)e^* + \varepsilon \\
&= \mathsf{Path}_e^{<n+1}(\Lambda) + \mathsf{Path}_e^{n+1}(\Lambda)e^* + \varepsilon && \text{by Lemma 5.19.}
\end{aligned}$$

□

We arrive at our final result. To make the result self-contained, we explicit state all the involved hypotheses.

Theorem 5.23. *Let Σ be a finite set, which we view as a discrete commutable set. Let $e : L_1 \to L_2$ be a representable relation, where $L_1 \leq L_2 \in T\Sigma$ and L_2 is prefix free. We can compute a term ε that expresses* partial reachability statements *for e, in the following sense.*

Let $L_i \leq L_1$ be a finite term over Σ and L_f be an arbitrary term. Suppose that the language interpretation of $\ddot{\Sigma}$ factors through some set A:

$$\begin{array}{ccc} T\ddot{\Sigma} & & \\ \downarrow{\scriptstyle f} & \searrow{\scriptstyle l} & \\ A & \xrightarrow{\;g\;} & \mathcal{L}\ddot{\Sigma}. \end{array}$$

Consider the equation

$$f((L_i)_r e^* + L_1^*(L_f)_r + \varepsilon) = f(L_1^*(L_f)_r + \varepsilon). \tag{7}$$

The following two results hold.

- Soundness*: if (7) holds and we have a sequence $s_i \overset{e}{\to}{}^{*} s_f$ of transitions, with $s_i \leq L_i$, then $s_f \leq L_f$.*
- Completeness for terminating executions*: if there is some k such that $s_i \overset{e}{\to}{}^{n} s_f$ implies $n < k$ whenever $s_i \in L_i$, and if all such sequences end in L_f, then (7) holds.*

Proof. Define $\varepsilon \triangleq \Sigma^*\Sigma^{\neq}\Sigma_l^*\Sigma_r^*\rho(e)e^*$. We reason as follows.

- *Soundness* Suppose that (7) holds. By applying g to both sides of the equation, and by unfolding the definition of the semiring order relation, we find

  $$l((L_i)_r e^*) \subseteq l(L_1^*(L_f)_r + \varepsilon).$$

  Suppose that we have some sequence of transitions $s_i \overset{e}{\to}{}^{n} s_f$ of length n. We can represent this sequence as an element p of the set $\mathsf{Path}_e^{<n+1}(L_i)$. By Lemma 5.22, this element must lie in $(L_i)_r e^*$, and thus, by (7), either in $L_1^*(L_f)_r$ or in ε. However, by Lemma 5.21, it cannot be in ε, so it must be in $L_1^*(L_f)_r$. We conclude by applying Lemma 5.21 again.

- *Completeness for terminating executions* Suppose that there is some strict upper bound k on the number of transitions starting at L_i. We can prove by induction that $\mathsf{Next}_e^k(L_i) = 0$. Moreover, if we assume that every sequence of transitions ends with an element of L_f, we can also prove by induction that $\mathsf{Next}_e^{<k}(L_i) \leq L_f$. By Lemma 5.22, we have

  $$\Lambda_r e^* \leq \mathsf{Path}_e^{<n}(\Lambda) + \mathsf{Path}_e^n(\Lambda)e^* + \varepsilon.$$

By Lemma 5.20, we have

$$\begin{aligned}\Lambda_r e^* &\leq L_1^* \,\mathsf{Next}_e^{<k}(\Lambda)_r + L_1^* \,\mathsf{Next}_e^k(\Lambda) e^* + < \\ &= L_1^* \,\mathsf{Next}_e^{<k}(\Lambda)_r + L_1^* 0 e^* + \varepsilon \\ &= L_1^* \,\mathsf{Next}_e^{<k}(\Lambda)_r + \varepsilon \\ &\leq L_1^* (L_f)_r + \varepsilon.\end{aligned}$$

□

6 Machines

We are now ready to show our undecidability result by building upon the infrastructure of the last section. More precisely, we will show that, if we can decide equations between weak Kleene algebra terms with commutativity conditions, then we can solve an undecidable problem on Turing machines. We'll use a slightly non-standard notion of reduction, which we'll introduce here. We use the notation $\langle x \rangle$ to refer to some effective encoding of the object x as a binary string.

Definition 6.1. A *bisected language* is a pair $L = (A, B)$ of disjoint sets of strings. Given a string x, we say that L *accepts* x, written $L(x) = \mathsf{y}$, if $x \in A$, and we say that L *rejects* x, written $L(x) = \mathsf{n}$, if $x \in B$.

We say that a bisected language L is *decidable* if there exists some Turing machine M such that:

- M halts on every input.
- If $L(x) = \mathsf{y}$, then M accepts x.
- If $L(x) = \mathsf{n}$, then M rejects x.

A bisected language is *total* if every string is in either A or B. In this case, the notion of decidability for bisected languages coincides with the usual notion of decidability.

Lemma 6.2. *Suppose that $E : 2^* \to 2^*$ is a computable function. We say that E is a reduction from the bisected language (A, B) to (A', B') if $E(A) \subseteq A'$ and $E(B) \subseteq B'$. In this case, if (A', B') is decidable, so is (A, B). Conversely, if (A, B) is undecidable, then (A', B') is undecidable.*

Theorem 6.3. *Define the* output (bisected) language of Turing machines L_{OUT} *as the pair (A, B), where*

$$\begin{aligned}A &\triangleq \{\langle \langle M \rangle, x \rangle \mid M \text{ outputs “yes” on } x\} \\ B &\triangleq \{\langle \langle M \rangle, x \rangle \mid M \text{ outputs “no” on } x\}.\end{aligned}$$

The language L_{OUT} is undecidable.

Proof. By adapting the usual diagonalization proof for the halting problem. □

To use Theorem 6.3, we need to find some reduction of the output language of Turing to an inequality on weak Kleene algebras. For simplicity, we'll use as a stepping stone the notion of *two-counter machine*. Roughly speaking, a two-counter machine M is an automaton that has a control state and two counters. The machine can increment each counter, test if their values are zero, and halt; we will give a formal definition of its behavior using the formalism of representable relations (Section 5).

Two-counter machines and Turing machines are equivalent in expressive power: any two-counter machine can simulate the execution of a Turing machine, and vice versa; see Hopcroft, Motwani, and Ullman [HMU01, §8.5.3, §8.5.4] for an idea of how this simulation works. In particular, given a Turing machine M, there exists a two-counter machine that halts on every input where M halts, and yields the same output for that input. To conclude, it suffices to reduce the output language for two-counter machines to the problem of solving weak Kleene algebra inequalities.

Definition 6.4. A *two-counter* machine is a tuple $M = (Q_M, \dot{q}, \iota)$, where Q is a finite set of *control states*, $\dot{q} \in Q_M$ is an initial state, and $\iota : Q_M \to I_M$ is a transition function. The set I_M is the set of *instructions* of the machine, defined as follows:

$$\begin{aligned} I_M \triangleq\ & \{\mathsf{Inc}(r, q) \mid r \in \{1, 2\}, q \in Q_M\} \\ & \cup \{\mathsf{If}(r, q_1, q_2) \mid r \in \{1, 2\}, q_1, q_2 \in Q_M\} \\ & \cup \{\mathsf{Halt}(x) \mid x \in \{\mathsf{y}, \mathsf{n}\}\}. \end{aligned}$$

Two-counter machines act on configurations, which are strings of the form $a^n b^m q$, where q is a control state and a and b are counter symbols: the number of symbol occurrences determines which number is stored in a counter. When the machine halts, it outputs either y or n to indicate whether its input was accepted or rejected.

Definition 6.5. Let M be a two-counter machine. We define the following discrete commutable sets and terms:

$$\begin{aligned} \Sigma_M &\triangleq Q_M + \{a, b, \mathsf{y}, \mathsf{n}\} && \text{symbols} \\ F\Sigma_M \ni C_M &\triangleq a^* b^* Q_M && \text{running configurations} \\ F\Sigma_M \ni T_M &\triangleq C_M + \{\mathsf{y}, \mathsf{n}\} && \text{all configurations.} \end{aligned}$$

Lemma 6.6. *We have $C_M \leq T_M$, and T_M is prefix free.*

Proof. The first point is trivial. As for the second one, we note that every string $s \leq T_M$ must be of the form $s'x$, where $x \in Q_M \cup \{\mathsf{y}, \mathsf{n}\}$ and s' does not contain any such symbols. Any proper prefix of such a string cannot lie in T_M. □

Definition 6.7 (Running a two-counter machine)**.** We interpret each instruction $i \in$

I_M as an element $(\!|i|\!) \in F\ddot{\Sigma}_M$:

$$\begin{aligned}
(\!|\mathsf{Inc}(1,q)|\!) &\triangleq a_r a^* b^* q_r \\
(\!|\mathsf{Inc}(2,q)|\!) &\triangleq a^* b_r b^* q_r \\
(\!|\mathsf{If}(1,q_1,q_2)|\!) &\triangleq b^*(q_1)_r + a_l a^* b^*(q_2)_r \\
(\!|\mathsf{If}(2,q_1,q_2)|\!) &\triangleq a^*(q_1)_r + a^* b_l b^*(q_2)_r \\
(\!|\mathsf{Halt}(x)|\!) &\triangleq x_r.
\end{aligned}$$

The *transition relation* of M is defined as

$$F\ddot{\Sigma}_M \ni R_M \triangleq \sum_{q \in Q_M} (\!|\iota(q)|\!) q_l.$$

We say that M *terminates* on n if $a^n b^0 \dot{q} \xrightarrow{R_M}{}^* x$ for some $x \in \{\mathsf{y}, \mathsf{n}\}$. We refer to x as the *output* of M on n. We say that M *accepts* n if it terminates on n outputting y, and we say that M *rejects* n if it does not accept n.

Lemma 6.8. *The term R_M is a representable relation of type $C_M \to T_M$ (Definition 5.9). Explicitly,*

- $\pi_l(R_M) \leq C_M$
- $\pi_r(R_M) \leq T_M$
- *R_M is finite state (Definition 4.3)*
- *R_M has bounded output (Definition 5.1)*

Proof. The first point is analogous to the second one, so we focus on that one. First, we prove that, for any instruction $i \in I_M$, $\pi_r((\!|i|\!)) \leq T_M$. Let us consider the example of Inc; the others are analogous:

$$\begin{aligned}
\pi_r(\mathsf{Inc}(1,q)) &= \pi_r(a_r a^* b^* q_r) \\
&= a a^* b^* q \\
&\leq a^* b^* q && \text{because } a a^* \leq a^* \\
&\leq C_M * \\
&\leq T_M.
\end{aligned}$$

Thus,

$$\begin{aligned}
\pi_r(R_M) &= \sum_{q \in Q_M} \pi_r((\!|\iota(q)|\!) q_l) \\
&= \sum_{q \in Q_M} \pi_r((\!|\iota(q)|\!)) \\
&\leq \sum_{q \in Q_M} T_M \\
&= T_M.
\end{aligned}$$

To show the remaining two points, we just have to appeal to the closure properties of finite-state and bounded-output terms (Lemmas 4.7 and 5.3). These lemmas say that these properties are always preserved by all the algebra operations, except possibly for star. For star, we need to check that the starred sub-terms do not contain 1 and that they only contain strings with at least one left symbol. The starred sub-terms are just $a_l a_r$ and $b_l b_r$, both of which satisfy this property. □

Lemma 6.9. *The relation R_M satisfies the following property: for every $a^n b^m q \leq C_M$, if $a^n b^m q \xrightarrow{R_M} s$, then $s = [\![\iota(q)]\!](n, m)$, where the function $[\![i]\!] : \mathbb{N}\times\mathbb{N} \to T_M$ is defined as follows*

$$\begin{aligned}
[\![\mathsf{Inc}(1,q)]\!](n,m) &\triangleq a^{n+1}b^m q \\
[\![\mathsf{Inc}(2,q)]\!](n,m) &\triangleq a^n b^{m+1} q \\
[\![\mathsf{If}(1,q_1,q_2)]\!](n,m) &\triangleq \begin{cases} a^n b^m q_1 & \text{if } n = 0 \\ a^p b^m q_2 & \text{if } n = p+1 \end{cases} \\
[\![\mathsf{If}(2,q_1,q_2)]\!](n,m) &\triangleq \begin{cases} a^n b^m q_1 & \text{if } m = 0 \\ a^n b^p q_2 & \text{if } m = p+1 \end{cases} \\
[\![\mathsf{Halt}(x)]\!](n,m) &\triangleq x.
\end{aligned}$$

In particular, this defines a functional relation.

Theorem 6.10. *Suppose that we have a diagram of sets:*

$$\begin{array}{ccc} T\ddot{\Sigma} & & \\ \downarrow{\scriptstyle f} & \searrow{\scriptstyle l} & \\ A & \xrightarrow{g} & \mathcal{L}\ddot{\Sigma}. \end{array}$$

Then the following is a reduction from the output language of two-counter machines to the (total) language of equations in A:

$$\langle\langle M\rangle, \langle n\rangle\rangle \mapsto \langle f((a^n b^0 \dot{q})_r R_M^* + (C_M)^*(C_M + \mathsf{y})_r + \varepsilon), f((C_M)^*(C_M + \mathsf{y})_r + \varepsilon)\rangle,$$

where ε is computed as in Theorem 5.23, by setting

$$\begin{aligned}
L_1 &\triangleq C_M \\
L_2 &\triangleq T_M \\
L_i &\triangleq a^n b^0 \dot{q} \\
L_f &\triangleq C_M + \mathsf{y}.
\end{aligned}$$

In particular, the language of equations of A is undecidable.

Proof. Suppose that $\langle\langle M\rangle, \langle n\rangle\rangle$ is accepted by L_{OUT}. Thus, $a^n b^0 \dot{q} \xrightarrow{R_M}{}^{k} \mathsf{y}$ for some k. Since y does not transition, and since the relation R_M is a (partial) function, all

transition sequences starting from L_i are strictly bounded by $k+1$. By completeness, this means that the equation holds.

Otherwise, suppose that $\langle\langle M\rangle, \langle n\rangle\rangle$ is rejected by L_{OUT}. Thus, $a^n b^0 \dot{q} \xrightarrow{R_M}{}^{*} \mathsf{n}$. Then the equation cannot hold. For suppose that it did hold. By soundness, we should have $\mathsf{n} \leq C_M + \mathsf{y}$, which is not the case. □

References

[HMU01] J.E. Hopcroft, R. Motwani, and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley series in computer science. Addison-Wesley, 2001. ISBN: 9780201441246. URL: https://books.google.com/books?id=omIPAQAAMAAJ.

[Koz97] Dexter Kozen. "On the Complexity of Reasoning in Kleene Algebra". In: *Proceedings, 12th Annual IEEE Symposium on Logic in Computer Science, Warsaw, Poland, June 29 - July 2, 1997.* IEEE Computer Society, 1997, pp. 195–202. DOI: 10.1109/LICS.1997.614947. URL: https://doi.org/10.1109/LICS.1997.614947.

[Kuz23] Stepan L. Kuznetsov. "On the Complexity of Reasoning in Kleene Algebra with Commutativity Conditions". In: *Theoretical Aspects of Computing - ICTAC 2023 - 20th International Colloquium, Lima, Peru, December 4-8, 2023, Proceedings.* Ed. by Erika Ábrahám, Clemens Dubslaff, and Silvia Lizeth Tapia Tarifa. Vol. 14446. Lecture Notes in Computer Science. Springer, 2023, pp. 83–99. DOI: 10.1007/978-3-031-47963-2_7. URL: https://doi.org/10.1007/978-3-031-47963-2%5C_7.

A Addenda on Kleene Algebra

Theorem A.1. *Let $X \in \mathsf{Comm}$ be discrete and $Y \in \mathsf{Comm}$ be commutative.*

$$l_X : \mathcal{K}X \to \mathcal{L}X$$
$$l_Y : \mathcal{K}Y \to \mathcal{L}Y$$

are isomorphisms.

Proof. These are standard results of Kleene algebra. □

Lemma A.2. *Let x be an element of a Kleene algebra X such that*

$$1 \leq x$$
$$xx \leq x.$$

Then the set $X_x \triangleq \{y \in X \mid y \leq x\}$ is a subalgebra of X.

Proof. Since all Kleene algebra operations are monotonic, they automatically preserve X_x. For example, if $y \leq x$ and $z \leq x$, then $yz \leq xx \leq x$, so $yz \in X_x$. Furthermore, notice that $x^* \leq x$ by induction. Therefore, $y^* \leq x^* \leq x$, so $y^* \in X_x$. □

As a subalgebra, when Y is finite, $\mathcal{K}Y$ is canonically isomorphic to $(\mathcal{K}X)_{Y^*}$ (cf. Lemma A.2).

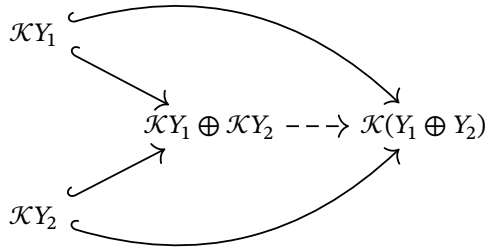
Lemma A.3. *Let X be a Kleene algebra. The set $M_{n,n}(X)$ of $n \times n$ matrices with coefficients in X is a Kleene algebra when endowed with matrix addition and multiplication. The star of a block matrix is given by the formula*

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^* = \begin{bmatrix} F^* & F^*BD^* \\ D^*CF^* & D^* + D^*CF^*BD^* \end{bmatrix},$$

where $F = A + BD^*C$. In particular, if $C = 0$ (that is, if the matrix is block upper triangular), then $F = A$, and thus

$$\begin{bmatrix} A & B \\ 0 & D \end{bmatrix}^* = \begin{bmatrix} A^* & A^*BD^* \\ 0 & D^* \end{bmatrix}.$$

Theorem A.4. *Let Y_1 and Y_2 be finite commutable sets. Given $a_1 \in \mathcal{K}Y_1$ and $a_2 \in \mathcal{K}Y_2$, we have $a_1a_2 = a_2a_1$ in $\mathcal{K}(Y_1 \oplus Y_2)$. In other words, we have the following factorization:*



Proof. By induction on a_1.

If $a_1 \in Y_1$, we proceed by induction on a_2. If $a_2 \in Y_2$, then $a_1a_2 = a_2a_1$ in $\mathcal{K}(Y_1 \oplus Y_2)$ because $a_1 \sim a_2$ in $Y_1 \oplus Y_2$. If $a_2 \in \{0, 1\}$, the result is trivial. If $a_2 = a_{21} + a_{22}$, we reason $a_1(a_{21} + a_{22}) = a_1a_{21} + a_1a_{22} = a_{21}a_1 + a_{22}a_1 = (a_{21} + a_{22})a_1$. If $a_2 = a_{21}a_{22}$, we reason $a_1a_{21}a_{22} = a_{21}a_1a_{22} = a_{21}a_{22}a_1$. Finally, if $a_2 = b_2^*$, we use the Kleene algebra theorem $xy = zx \Rightarrow xy^* = z^*x$.

The other cases for a_1 follow a similar pattern. □

Theorem A.5. *Let $Y \subseteq X$ be a commutable subset. We have the following pullback square:*

$$\begin{array}{ccc} \mathcal{K}Y & \xrightarrow{l} & \mathcal{L}Y \\ \downarrow & & \downarrow \\ \mathcal{K}X & \xrightarrow{l} & \mathcal{L}X. \end{array}$$

In other words, if $e \in \mathcal{K}X$ is such that $l(e) \subseteq Y^*$, then e must be in the image of $\mathcal{K}Y$ in $\mathcal{K}X$.

Proof. By induction on e.

- If $e = 0$ or $e = 1$, then e is trivially in the image.
- Suppose that $e = x \in X$. Since $l(e) \subseteq Y^*$, we have $x \in Y$, from which the result follows.
- Suppose that $e = e_1 + e_2$. Then $l(e_i) \subseteq Y^*$ for every i. By the induction hypotheses, this means that, for every i, e_i is in the image of $\mathcal{K}Y$. Thus, so is $e_1 + e_2$.
- Suppose that $e = e_1 e_2$. Let us begin by showing that $l(e_1) \subseteq Y^*$. We can assume that $l(e_2)$ is nonempty; otherwise, the result follows trivially, because Theorem 3.6 implies $e_2 = 0$ and thus $e = 0$, which is in the image of $\mathcal{K}Y$. Thus, we have some $s_2 \in l(e_2)$. Now, consider some arbitrary $s_1 \in l(e_1)$. Since $s_1 s_2 \in l(e_1 e_2) \subseteq Y^*$, we conclude that s_1 can only have symbols in Y. This implies that $l(e_1)$.

  By a symmetric reasoning, we can show that $l(e_2) \subseteq Y^*$. Thus, our induction hypotheses show that both e_1 and e_2 are in the image of $\mathcal{K}Y$, and so is $e_1 e_2$.
- Finally, suppose that $e = e_1^*$. We have $l(e_1) \subseteq l(e_1^*) \subseteq Y^*$. Thus, our induction hypothesis says that e_1 is in the image of $\mathcal{K}Y$, implying that so is e.

□

B Decomposition

Let X be a finite commutable set and Y and Z be two disjoint commutable subsets of X. We are going to show that every term of TX can be written uniquely as a sum of two components: one that only contains symbols from Y, and another one that contains at least one symbol from Z. Consider the following matrix:

$$U_{X,Y} \triangleq \begin{bmatrix} Y^* & Y^*ZX^* \\ 0 & X^* \end{bmatrix}.$$

We can check $1 \leq U_{X,Y}$ and $U_{X,Y}U_{X,Y} \leq U_{X,Y}$, hence we can consider the ideal Kleene algebra $M_{2,2}(\mathcal{K}X)_{U_{X,Y}}$.

Lemma B.1. *Let*

$$D_{X,Y} \triangleq \left\{ \begin{bmatrix} a & b \\ 0 & a+b \end{bmatrix} \mid a \leq Y^*, b \leq Y^*ZX^* \right\}.$$

This is a Kleene subalgebra of $M_{2,2}(\mathcal{K}X)_{U_{X,Y}}$.

Proof. It suffices to check closure for all the Kleene algebra operations. We focus on two interesting cases. Let

$$x_1 = \begin{bmatrix} a_1 & b_1 \\ 0 & a_1 + b_1 \end{bmatrix}$$
$$x_2 = \begin{bmatrix} a_2 & b_2 \\ 0 & a_2 + b_2 \end{bmatrix}.$$

Then

$$x_1 x_2 = \begin{bmatrix} a_1 a_2 & a_1 b_2 + b_1 (a_2 + b_2) \\ 0 & a_1 a_2 + (a_1 b_2 + b_1 (a_2 + b_2)) \end{bmatrix},$$

which is of the desired form.

Otherwise, suppose

$$x = \begin{bmatrix} a & b \\ 0 & a + b \end{bmatrix},$$

then

$$x^* = \begin{bmatrix} a^* & a^* b (a + b)^* \\ 0 & (a + b)^* \end{bmatrix}.$$

Since $a^* + a^* b (a + b)^* = (a + b)^*$ by a standard theorem of Kleene algebra, we are done. □

Lemma B.2. *Define a morphism of type $i : \mathcal{K}X \to D_{X,Y}$ by lifting*

$$c \mapsto \begin{cases} \begin{bmatrix} c & 0 \\ 0 & c \end{bmatrix} & \text{if } c \in Y \\ \begin{bmatrix} 0 & c \\ 0 & c \end{bmatrix} & \text{otherwise.} \end{cases}$$

This morphism has a left inverse j given by

$$j \begin{bmatrix} a & b \\ 0 & a + b \end{bmatrix} = a + b.$$

Corollary B.3 (Existence of decomposition)**.** *Every $x \in \mathcal{K}X$ must be of the form $a + b$ for some $a \leq Y^*$ and some $b \leq Y^* Z X^*$.*

Proof. By Lemma B.2, $x = j(i(x))$, which must be of this form. □

We are now going to show that the decomposition is unique.

Lemma B.4. *Suppose $a \leq Y^*$ and $b \leq Y^* Z X^*$. Then*

$$i(a) \triangleq \begin{bmatrix} a & 0 \\ 0 & a \end{bmatrix} \qquad\qquad i(b) \triangleq \begin{bmatrix} 0 & b \\ 0 & b \end{bmatrix}.$$

Proof. Notice that

$$i(Y^*) \triangleq \begin{bmatrix} Y^* & 0 \\ 0 & Y^* \end{bmatrix} \qquad\qquad i(Y^* Z X^*) \triangleq \begin{bmatrix} 0 & Y^* Z X^* \\ 0 & Y^* Z X^* \end{bmatrix}.$$

Since $i(a) \leq i(Y^*)$ and $i(b) \leq i(Y^* Z X^*)$, these matrices must be of the form

$$i(a) \triangleq \begin{bmatrix} a' & 0 \\ 0 & a' \end{bmatrix} \qquad\qquad i(b) \triangleq \begin{bmatrix} 0 & b' \\ 0 & b' \end{bmatrix}.$$

But by Lemma B.2, we must have $a' = a$ and $b' = b$, from which the result follows. □

Corollary B.5 (Decomposition is unique)**.** *Suppose that* $a_1 + b_1 = a_2 + b_2$*, when* $a_i \le Y^*$ *and* $b_i \le Y^* Z X^*$*. Then* $a_1 = a_2$ *and* $b_1 = b_2$*.*

Proof. By Lemma B.4,

$$\begin{aligned} i(a_i + b_i) &= i(a_i) + i(b_i) \\ &= \begin{bmatrix} a_i & 0 \\ 0 & a_i \end{bmatrix} + \begin{bmatrix} 0 & b_i \\ 0 & b_i \end{bmatrix} \\ &= \begin{bmatrix} a_i & b_i \\ 0 & a_i + b_i \end{bmatrix}. \end{aligned}$$

Since $i(a_1 + b_1) = i(a_2 + b_2)$, we must have $a_1 = a_2$ and $b_1 = b_2$. □

Corollary B.6. $i : \mathcal{K}X \cong D_{X,Y}$.

Proof. By the previous results, $i(j(x))$ must be equal to x, so i and j are two-sided inverses of each other. □

With this decomposition, we can define and compute the empty word property of a word. We can instantiate $Y = \emptyset$, and $Z = X$, in this case:

$$i(e) = \begin{bmatrix} a & b \\ 0 & a + b \end{bmatrix},$$

and $a \le Y^*$, and $b \le XX^*$. Since the projection of first diagonal element $\pi_{1,1}$ is a homomorphism, $\pi_{1,1} \circ i : FX \to FX$ is also a homomorphism, and thus the range of $\pi_{1,1} \circ i$ forms a KA. Since the element in this range is smaller than Y^*, the range is a sub-KA of FY, also because $Y = \emptyset$ is the initial object in the category of commutable set, thus FY is also the initial element in KA, which is 2, and such a KA contains no proper sub-KA. Therefore, the range of $\pi_{1,1} \circ i$ is 2, and we will denote the homomorphism $\pi_{1,1} \circ i$ restricted to its range as $E : FX \to 2$.

TODO: prove term under Y^* is isomorphic to FY.

Corollary B.7 (Completeness of E)**.** *For all term* $e \in FX$*:*

$$E(e) = 1 \iff \epsilon \in l(e) \iff e \ge 1.$$

Proof. Recall that e can be decomposed into $E(e) + b$, where $b \in XX^*$. We consider the language interpretation of e:

$$l(e) = l(E(e)) + l(b).$$

Notice that $b \le XX^*$, hence $\epsilon \notin l(b)$. And because $\{\epsilon\} \not\subseteq l(b)$, therefore $1 \not\le b$.

We first show $E(e) = 1 \iff \epsilon \in l(e)$. If $E(e) = 1$, then $\epsilon \in l(E(e))$, and

$$\epsilon \in l(E(e)) \subseteq l(E(e)) + l(b) = l(e).$$

If $E(e) \ne 1$, i.e. $E(e) = 0$, then $\epsilon \notin l(E(e))$. Because $\epsilon \notin l(E(e))$ and $\epsilon \notin l(b)$, therefore

$$\epsilon \notin l(E(e)) + l(b) = l(e).$$

We then show $E(e) = 1 \iff e \geq 1$. If $E(e) = 1$, then

$$e = E(e) + b \geq 1.$$

If $E(e) \neq 1$, then $E(e) = 0$,

$$e = E(e) + b = b \not\geq 1.$$

□

Corollary B.8 (Decidability of E)**.** *For all term $e \in FX$, $E(e) = 1$ is decidable.*

Proof. Because the KA 2 is decidable, that is every expression in 2 can be reduced to either 0 or 1 in finite time. Therefore, we can simply compute the result of $E(e)$ following the structure of e, and then the result of $E(e)$ to see if it equals to 1. □

TODO: I don't think the fact that "i is an isomorphism" is necessary?

Corollary B.9. *Given any $x \in \mathcal{K}X$, it is decidable whether $1 \leq x$.*

Proof. Let's instantiate the previous results with $Y = \emptyset$ and $Z = X$. Since $i : \mathcal{K}X \cong D_{X,Y}$, $1 \leq x$ is equivalent to

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \leq i(x).$$

We can determine this by checking that $1 \leq i(x)_{1,1}$. Since $\emptyset$ is the initial object of Comm, $F\emptyset$ must also be initial in KA, and thus isomorphic to $\{0 \leq 1\}$, which is a decidable Kleene Algebra. So, we just have to see if $i(x)_{1,1}$ is equal to 1 via the canonical isomorphism coming from initiality. □

C Derivative

C.1 Term Derivative

Let X be a finite commutable set. We prove that the derivative of a term $a \in \mathcal{K}X$ is well-defined with respect to any $x \in X$. Define the following commutable subsets of X:

$$Y \triangleq \{y \sim x \mid y \neq x\}$$
$$Z \triangleq \{z \not\sim x \mid z \neq x\}.$$

Define the following matrix

$$U_x \triangleq \begin{bmatrix} Y^* & Y^*ZX^* & X^* \\ 0 & X^* & 0 \\ 0 & 0 & X^* \end{bmatrix}.$$

Note that $1 \leq U_x$ and $U_xU_x \leq U_x$. Thus, we can consider the ideal Kleene algebra $M_{3,3}(\mathcal{K}X)_{U_x}$.

Lemma C.1. *The set $D_x \subseteq M_{3,3}(\mathcal{K}X)_{U_x}$ defined as*

$$D_x \triangleq \left\{ \begin{bmatrix} a & b & c \\ 0 & d & 0 \\ 0 & 0 & d \end{bmatrix} \mid d = a + b + xc \right\}$$

is a Kleene subalgebra.

Proof. We just need to check closure for the Kleene algebra operations. We focus on two cases: multiplication and star.

Suppose that we have m_1 and m_2 with

$$m_i = \begin{bmatrix} a_i & b_i & d_i \\ 0 & d_i & 0 \\ 0 & 0 & d_i \end{bmatrix},$$

where $d_i = a_i + b_i + xc_i$, and similarly for d_2. Hence

$$m_1 m_2 = \begin{bmatrix} a_1a_2 & a_1b_2 + b_1d_2 & a_1c_2 + c_1d_2 \\ 0 & d_1d_2 & 0 \\ 0 & 0 & d_1d_2 \end{bmatrix}.$$

We have

$$\begin{aligned} d_1d_2 &= a_1a_2 + [a_1b_2 + b_1a_2 + b_1b_2 + b_1xc_2] + [a_1xc_2 + xc_1a_2 + xc_1b_2 + xc_1xc_2] \\ &= a_1a_2 + [a_1b_2 + b_1a_2 + b_1b_2 + b_1xc_2] + [xa_1c_2 + xc_1a_2 + xc_1b_2 + xc_1c_2] \\ &= a_1a_2 + [a_1b_2 + b_1a_2 + b_1b_2 + b_1xc_2] + x[a_1c_2 + c_1a_2 + c_1b_2 + c_1xc_2] \\ &= a_1a_2 + [a_1b_2 + b_1d_2] + x[a_1c_2 + c_1d_2], \end{aligned}$$

which is what we seek.

On the other hand, we have

$$m_1^* \triangleq \begin{bmatrix} a_1^* & a_1^*b_1d_1^* & a_1^*c_1d_1^* \\ 0 & d_1^* & 0 \\ 0 & 0 & d_1^* \end{bmatrix}.$$

Since

$$\begin{aligned} d_1^* &= (a_1 + b_1 + xc_1)^* \\ &= a_1^* + a_1^*b_1d_1^* + a_1^*xc_1d_1^* \\ &= a_1^* + a_1^*b_1d_1^* + x(a_1^*c_1d_1^*), \end{aligned}$$

(note that $a_1^*x = xa_1^*$) we are done. □

Theorem C.2. *Define a morphism of type $i : \mathcal{K}X \to D_x$ by lifting*

$$\alpha \mapsto \begin{cases} \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & \alpha \end{bmatrix} & \text{if } \alpha \in Y \\ \begin{bmatrix} 0 & \alpha & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & \alpha \end{bmatrix} & \text{if } \alpha \in Z \\ \begin{bmatrix} 0 & 0 & 1 \\ 0 & x & 0 \\ 0 & 0 & x \end{bmatrix} & \text{if } \alpha = x \end{cases}$$

This morphism has a left inverse j given by

$$j \begin{bmatrix} a & b & c \\ 0 & d & 0 \\ 0 & 0 & d \end{bmatrix} = d.$$

Corollary C.3. *Every $e \in TX$ is of the form $a + b + xc$ for some $a \leq Y^*$, $b \leq Y^*ZX^*$ and $c \leq X^*$.*

Lemma C.4. *Let $a \leq Y^*$, $b \leq Y^*ZX^*$ and $c \leq X^*$. Then*

$$i(a) = \begin{bmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & a \end{bmatrix} \qquad i(b) = \begin{bmatrix} 0 & b & 0 \\ 0 & b & 0 \\ 0 & 0 & b \end{bmatrix} \qquad i(xc) = \begin{bmatrix} 0 & 0 & c \\ 0 & xc & 0 \\ 0 & 0 & xc \end{bmatrix}.$$

Proof. By linearity,

$$i(Y) = \begin{bmatrix} Y & 0 & 0 \\ 0 & Y & 0 \\ 0 & 0 & Y \end{bmatrix} \qquad i(Z) = \begin{bmatrix} 0 & Z & 0 \\ 0 & Z & 0 \\ 0 & 0 & Z \end{bmatrix} \qquad i(X) = \begin{bmatrix} Y & Z & 1 \\ 0 & X & 0 \\ 0 & 0 & X \end{bmatrix}.$$

Therefore,

$$
\begin{aligned}
i(Y^*) &= \begin{bmatrix} Y^* & 0 & 0 \\ 0 & Y^* & 0 \\ 0 & 0 & Y^* \end{bmatrix} \\
i(X^*) &= \begin{bmatrix} Y^* & Y^*ZX^* & Y^*1X^* \\ 0 & X^* & 0 \\ 0 & 0 & X^* \end{bmatrix} \\
&= \begin{bmatrix} Y^* & Y^*ZX^* & X^* \\ 0 & X^* & 0 \\ 0 & 0 & X^* \end{bmatrix} \\
i(Y^*ZX^*) &= \begin{bmatrix} Y^* & 0 & 0 \\ 0 & Y^* & 0 \\ 0 & 0 & Y^* \end{bmatrix} \begin{bmatrix} 0 & Z & 0 \\ 0 & Z & 0 \\ 0 & 0 & Z \end{bmatrix} \begin{bmatrix} Y^* & Y^*ZX^* & X^* \\ 0 & X^* & 0 \\ 0 & 0 & X^* \end{bmatrix} \\
&= \begin{bmatrix} 0 & Y^*Z & 0 \\ 0 & Y^*Z & 0 \\ 0 & 0 & Y^*Z \end{bmatrix} \begin{bmatrix} Y^* & Y^*ZX^* & X^* \\ 0 & X^* & 0 \\ 0 & 0 & X^* \end{bmatrix} \\
&= \begin{bmatrix} 0 & Y^*ZX^* & 0 \\ 0 & Y^*ZX^* & 0 \\ 0 & 0 & Y^*ZX^* \end{bmatrix} \\
i(xX^*) &= \begin{bmatrix} 0 & 0 & 1 \\ 0 & x & 0 \\ 0 & 0 & x \end{bmatrix} \begin{bmatrix} Y^* & Y^*ZX^* & Y^*1X^* \\ 0 & X^* & 0 \\ 0 & 0 & X^* \end{bmatrix} \\
&= \begin{bmatrix} 0 & 0 & X^* \\ 0 & xX^* & 0 \\ 0 & 0 & xX^* \end{bmatrix}.
\end{aligned}
$$

We conclude by noting that $i(a) \leq i(Y^*)$, $i(b) \leq i(Y^*ZX^*)$ and $i(xc) \leq i(xX^*)$, like we did for the proof of Lemma B.4. □

Corollary C.5. *The morphism $i : TX \to D_x$ is surjective, and thus an isomorphism. In particular, every $e \in TX$ can be uniquely written as a sum $a + b + xc$ with $a \leq Y^*$, $b \leq Y^*ZX^*$ and $c \leq X^*$.*

Proof. We have

$$
\begin{aligned}
&\begin{bmatrix} a & b & c \\ 0 & a+b+xc & 0 \\ 0 & 0 & a+b+xc \end{bmatrix} \\
&= \begin{bmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & a \end{bmatrix} + \begin{bmatrix} 0 & b & 0 \\ 0 & b & 0 \\ 0 & 0 & b \end{bmatrix} + \begin{bmatrix} 0 & 0 & c \\ 0 & xc & 0 \\ 0 & 0 & xc \end{bmatrix} \\
&= i(a) + i(b) + i(xc) \\
&= i(a + b + xc).
\end{aligned}
$$

□

Definition C.6. Suppose that

$$i(e) = \begin{bmatrix} a & b & c \\ 0 & d & 0 \\ 0 & 0 & d \end{bmatrix}.$$

We define the following terms:

$\rho_x(e) \triangleq a + b$	residue of e with respect to x
$\delta_x(e) \triangleq c$	derivative of e with respect to x.

Theorem C.7. *The derivative and the residue satisfy the following properties.*

$$\begin{aligned}
e &= \rho_x(e) + x\delta_x(e) \\
\rho_x(e) &\leq Y^* + Y^*ZX^* \\
\rho_x(xe) &= 0 \\
\rho_x(ye) &= y\rho_x(e) && y \in Y \\
\rho_x(ze) &= ze && z \in Z \\
\rho_x(1) &= 1 \\
\rho_x(0) &= 0 \\
\rho_x(e_1 + e_2) &= \rho_x(e_1) + \rho_x(e_2) \\
\delta_x(x) &= 1 \\
\delta_x(xe) &= e \\
\delta_x(y) &= 0 && y \neq x \\
\delta_x(ye) &= y\delta_x(e) && y \in Y \\
\delta_x(ze) &= 0 && z \in Z \\
\delta_x(1) &= 0 \\
\delta_x(0) &= 0 \\
\delta_x(e_1 + e_2) &= \delta_x(e_1) + \delta_x(e_2) \\
\delta_x(e_1e_2) &= \delta_x(e_1)e_2 + \pi_Y(e_1)\delta_x(e_2) \\
\delta_x(e^*) &= \pi_Y(e)^*\delta_x(e)e^* \\
\rho_x(\rho_x(e)) &= \rho_x(e) \\
\delta_x(\rho_x(e)) &= 0 \\
\delta_x(e) = 0 &\iff \rho_x(e) = e \\
e_1 \leq e_2 &\implies \delta_x(e_1) \leq \delta_x(e_2) \\
e_1 \leq e_2 &\implies \rho_x(e_1) \leq \rho_x(e_2).
\end{aligned}$$

Corollary C.8. *Given any $e, e' \in \mathcal{K}X$, we have*

$$xe \leq e' \iff e \leq \delta_x(e').$$

In particular, $x\delta_x(e) \leq e$.

Proof. We first show that δ_x is monotonic. Because i is a homomorphism, hence monotonic. Recall the order on matrix model is the point-wise order, and because δ_x is a component of i:

$$e_1 \geq e_2 \Longrightarrow i(e_1) \geq i(e_2) \Longrightarrow \delta_x(e_1) \geq \delta_x(e_2).$$

The $\Longrightarrow$ direction:

$$xe \leq e' \Longrightarrow \delta_x(xe) \leq \delta_x(e') \Longrightarrow e \leq \delta_x(e').$$

The $\Longleftarrow$ direction:

$$e \leq \delta_x(e') \Longrightarrow xe \leq x\delta_x(e') \leq e'$$

□

TODO: can we prove the minimality of $\rho_x(e)$, that is $\rho_x(e)$ is the least e' s.t. $e = e' + x\delta_x(e)$

Definition C.9. The definition of derivative and residue can be extended to words as follows:

$$\delta_\epsilon(e) \triangleq e \text{ and } \delta_{w\cdot x}(e) = \delta_w(\delta_x(e))$$
$$\rho_\epsilon(e) \triangleq 0 \text{ and } \rho_{w\cdot x}(e) = \rho_w(e) + \delta_w(\rho_x(e))$$

Theorem C.10 (soundness)**.** *The following property still holds on derivative and residue for word:*

$$e = \rho_w(e) + w \cdot \delta_w(e)$$
$$we \leq e' \Longleftrightarrow e \leq \delta_w(e')$$

Corollary C.11. *Given any string $s \in TX$ and any term $e \in TX$, it is decidable to check whether $s \leq e$.*

Proof. Write s as a product of the form $\prod x_i$, where $x_i \in X$. By Corollaries B.9 and C.8, we can check $1 \leq \delta_{x_n}(\cdots\delta_{x_1}(e)\cdots)$. □

Lemma C.12. *We have $\delta_x(e) = 0$ if and only if $e \leq Y^* + Y^*ZX^*$.*

Proof. If $e \leq Y^* + Y^*ZX^*$, then $\delta_x(e) \leq \delta_x(Y^* + Y^*ZX^*) = 0$. Conversely, suppose that $\delta_x(e) = 0$. By Corollary C.5, we can write $e = a + b + x\delta_x(e)$, with $a \leq Y^*$ and $b \leq Y^*ZX^*$, which allows us to conclude. □

We can iterate this procedure finitely many times.

Theorem C.13. *Suppose that X is a commutable set and $A \subseteq X$ is a discrete finite commutable subset. Every $e \in TX$ can be written in the form*

$$e = e_0 + \sum_{x \in A} x\delta_x(e),$$

where $\delta_x(e_0) = 0$ for every $x \in A$. Moreover, the $\delta_x(e)$ are the unique terms that enable this decomposition.

Proof. We prove this by induction on the cardinality of A. If A is empty, we can take e_0 to be e. Otherwise, assume that $A = A' \cup \{x\}$, where $x \notin A'$. By the induction hypothesis, we can write e as

$$e = e_0 + \sum_{y \in A'} y\delta_y(e),$$

where $\delta_x(e_0) = 0$ for every $x \in A'$. By Theorem C.7, we have

$$\begin{aligned}
\delta_x(e) &= \delta_x(e_0) + \delta_x\left(\sum_{y \in A'} y\delta_y(e)\right) \\
&= \delta_x(e_0) + \sum_{y \in A'} \delta_x(y\delta_y(e)) \\
&= \delta_x(e_0) + \sum_{y \in A'} 0 \\
&= \delta_x(e_0).
\end{aligned}$$

Thus,

$$\begin{aligned}
e &= \rho_x(e_0) + x\delta_x(e_0) + \sum_{y \in A'} y\delta_y(e) \\
&= \rho_x(e_0) + x\delta_x(e) + \sum_{y \in A'} y\delta_y(e) \\
&= \rho_x(e_0) + \sum_{y \in A} y\delta_y(e).
\end{aligned}$$

We have $\delta_x(\rho_x(e_0)) = 0$ and $\delta_y(\rho_x(e_0)) \leq \delta_y(e_0) = 0$ if $y \in A'$. So this decomposition has the desired form.

To show that the decomposition is unique, suppose that we can write

$$e = e' + \sum_{y \in A} ye_y = e' + xe_x + \sum_{y \in A'} ye_y.$$

where $\delta_y(e') = 0$ for $y \in A'$. This means that $\delta_y(e' + xe_x) = 0$ for every $y \in A'$. Since the decomposition with respect to A' is unique, we find that $e' + xe_x = e_0$ and $e_y = \delta_y(e)$ for every $y \in A'$. And, since $\delta_x(e') = 0$, the decomposition of e_0 with respect to x is unique, and $e' = \rho_x(e_0)$ and $e_x = \delta_x(e_0)$. □

C.2 Language Derivative

Definition C.14. Let $L \in \mathcal{L}X$ and $x \in X$. Define $\delta_x(L)$ as follows:

$$\delta_x(L) \triangleq \{s \in TX \mid xs \in L\}.$$

Theorem C.15. *The following diagram commutes for every $x \in X$:*

$$\begin{array}{ccc}
TX & \xrightarrow{\delta_x} & TX \\
\downarrow{\scriptstyle l} & & \downarrow{\scriptstyle l} \\
\mathcal{L}X & \xrightarrow{\delta_x} & \mathcal{L}X.
\end{array}$$

Proof. Let $a \in TX$ and $s \in TX$ be a string. We have

$$\begin{aligned}
& s \in l(\delta_x(a)) \\
\Leftrightarrow\ & s \leq \delta_x(a) && \text{by Theorem 3.4} \\
\Leftrightarrow\ & xs \leq a && \text{by Corollary C.8} \\
\Leftrightarrow\ & xs \in l(a) && \text{by Theorem 3.4} \\
\Leftrightarrow\ & s \in \delta_x(l(a)) && \text{by Corollary C.8.}
\end{aligned}$$

□

C.3 Fundamental Theorem And Word Decomposition

Given a commutable set X, we consider its carrier as a discrete commutable set $X_\asymp$. There is a canonical homomorphism between these commutable sets

$$[-]_\sim : X_\asymp \to X \tag{8}$$

$$[x]_\sim \triangleq x \tag{9}$$

And this map lifts to $[-]_\sim : FX_\asymp \to FX$, which simply maps each term to syntactically the same term. Furthermore, this homomorphism can be lifted into matrices by point-wise application.

Since $X_\asymp$ is a discrete commutable set, $FX_\asymp$ is simply the free Kleene Algebra over $X_\asymp$. It is well known that the fundamental theorem holds for free Kleene Algebras:

$$\forall e \in FX_\asymp, e = E(e) + \sum_{a \in X_\asymp} a \cdot \delta_a(e).$$

Thus, the fundamental theorem holds over discrete commutative set. To extend the fundamental theorem over any commutable set X, We need a simple lemma:

Lemma C.16. *For any $x \in X$ Consider the homomorphism i from the last section,*

$$[i(e)]_\sim \leq i([e]_\sim).$$

Proof. The proof is a simple inductive the structure of e, the base case is apparent by inspecting the definition of i. And the induction case is trivial by the fact that both i and $[-]_\sim$ are homomorphism, and all KA operations preserves order.

We will take the star case as an example, assume $e \triangleq e'^*$, given $[i(e')]_\sim \leq i([e']_\sim)$, by the fact that star operation preserves order:

$$[i(e'^*)]_\sim = ([i(e')]_\sim)^* \leq (i([e']_\sim))^* = i([e'^*]_\sim).$$

TODO: we can probably prove this as a general lemma, order of the homomorphism are uniquely determined by order on primitives. □

TODO: prove $[-]_\sim$ preserves E.

Since $[i(e)]_\sim \leq i([e]_\sim)$, and the derivative is a component of i, we have

$$\forall x \in X, [\delta_x(e)]_\sim \leq \delta_x([e]_\sim).$$

Theorem C.17 (Fundamental Theorem)**.** *For all term $e_\sim \in FX$, we have the following equation:*

$$e = E(e) + \sum_{a \in X} a \cdot \delta_a(e).$$

Proof. Since $[-]_\sim$ is surjective, we let $e = [e_\asymp]_\sim$ for some $e_\asymp \in FX_\asymp$.

We first show that $e \leq E(e) + \sum_{a \in X} a \cdot \delta_a(e)$. Given that the fundamental theorem hold for $e_\asymp$, we have the following inequality:

$$e_\asymp \leq E(e_\asymp) + \sum_{a \in X} a \cdot \delta_a(e_\asymp)$$

We apply the homomorphism $[-]_\sim$ to both side, and get:

$$e \leq [E(e_\asymp)]_\sim + \sum_{a \in X} a \cdot [\delta_a(e_\asymp)]_\sim.$$

Because $[E(e_\asymp)]_\sim = E([e_\asymp]_\sim)$ and $[\delta_x(e_\asymp)]_\sim \leq \delta_x([e_\asymp]_\sim)$, we have

$$e \leq E(e) + \sum_{a \in X} a \cdot \delta_a(e).$$

The other direction $e \geq E(e) + \sum_{a \in X} a \cdot \delta_a(e)$ is straightforward:

Since $e = \rho_x(e) + a\delta_a(e)$, therefore $e \geq a\delta_a(e)$ for all a. And because $e \geq E(e)$, we have

$$e \geq E(e) + \sum_{a \in X} a \cdot \delta_a(e).$$

□

Corollary C.18 (Word Decomposition)**.** *Given an expression $e \in FX$, and a word $w \in l(e)$, the following equality holds:*

$$e = w + \rho_w(e) + \sum_{x \in X} w \cdot x \cdot \delta_{w \cdot x}(e).$$

Proof.

$$\begin{aligned} e &= w \cdot \delta_w(e) + \rho_w(e) \\ &= w \cdot (1 + \sum_{x \in X} x \cdot \delta_{w \cdot x}(e)) + \rho_w(e) \\ &= w + \rho_w(e) + \sum_{x \in X} w \cdot x \cdot \delta_{w \cdot x}(e) \end{aligned}$$

□

The word decomposition theorem can be seen as a more explicit proof of the completeness of word inhabitant. We can derive a similar but less explicit result without using the fundamental theorem, by applying the decomposition in Corollary B.9. Although such result is more opaque, it is enough for proving the decidability and undecidability results.