Kleene Algebra with Equations

Dexter Kozen and Konstantinos Mamouras

Computer Science Department, Cornell University, Ithaca, NY 14853-7501, USA
`{kozen,mamouras}@cs.cornell.edu`

Abstract. We identify sufficient conditions for the construction of free language models for systems of Kleene algebra with additional equations. The construction applies to a broad class of extensions of KA and provides a uniform approach to deductive completeness and coalgebraic decision procedures.

1 Introduction

Kleene algebra (KA) is the algebra of regular expressions. Introduced by Stephen Cole Kleene in 1956, it is fundamental and ubiquitous in computer science. It has proven useful in countless applications, from program specification and verification to the design and analysis of algorithms [1–8].

One can augment KA with Booleans in a seamless way to obtain Kleene algebra with tests (KAT). Unlike many other related logics for program verification, KAT is classically based, requiring no specialized syntax or deductive apparatus other than classical equational logic. In practice, statements in the logic are typically universal Horn formulas

$$s_1 = t_1 \to s_2 = t_2 \to \cdots \to s_n = t_n \to s = t,$$

where the conclusion $s = t$ is the main target task and the premises $s_i = t_i$ are the verification conditions needed to prove it. The conclusion $s = t$ may encode a partial correctness assertion, an equivalence between an optimized and an unoptimized version of a program, or an equivalence between a program annotated with static analysis information and the unannotated program. The verification conditions $s_i = t_i$ are typically simple properties of the underlying domain of computation that describe how atomic actions interact with atomic assertions. They may require first-order interpreted reasoning, but are proven once and for all, then abstracted to propositional form. The proof of the conclusion $s = t$ from the premises takes place at the propositional level in KAT. This methodology affords a clean separation of the theory of the domain of computation from the program restructuring operations. It is advantageous to separate the two levels of reasoning, because the full first-order theory of the domain of computation may be highly undecidable, even though we may only need small parts of it. By isolating those parts, we can often maintain decidability and deductive completeness.

A typical form of premise that arises frequently in practice is a *commutativity condition* $pb = bp$ for an action p and a test b. This captures the idea that the action p does not affect the truth of b. For example, the action p might be an assignment $x := 3$ and b might be a test $y = 4$, where x and y are distinct variables. It is clear that the truth value of b is not affected by the action p, so it would be the same before as after. But once this is established, we no longer need to know what p and b are, but only that $pb = bp$. It follows by purely equational reasoning in KAT that $p_1b = bp_1 \to \cdots \to p_nb = bp_n \to qb = bq$, where q is any program built from atomic actions $p_1, \ldots, p_n$.

In some instances, Horn formulas with premises of a certain form can be reduced to the equational theory without loss of deductive completeness or decision efficiency using a technique known as *elimination of hypotheses* [3, 9, 10]. One important class of premises for which this is possible are those of the form $s = 0$. The universal Horn theory restricted to premises of this form is called the *Hoare theory*, because it subsumes Hoare logic: the partial correctness assertion $\{b\}p\{c\}$ can be encoded as the equation $bp\bar{c} = 0$. Other forms that arise frequently in practice are $bp = b$, which says that the action p is not necessary if b is true, useful in optimizations to eliminate redundant actions; and $pq = qp$, which says that the atomic actions p and q can occur in either order with the same effect, useful in reasoning about concurrency. Unfortunately, KAT with general commutativity assumptions $pq = qp$ is undecidable [12].

As a case in point, the NetKAT system [8] incorporates a number of such equational premises as part of the theory, which are taken as additional axioms besides those of KAT. Proofs of deductive completeness and complexity as given in [8] required extensive adaptation of the analogous proofs for KA and KAT. Indeed, this was already the case with KAT, which was an adaptation of KA to incorporate an embedded Boolean algebra.

Although each of these instances was studied separately, there are some striking similarities. It turns out that the key to progress in all of them is the identification of a suitable class of *language models* that characterize the equational theory of the system. A language model is a structure in which expressions are interpreted as sets of elements of some monoid. The language models should form the free models for the system at hand. For KA, a language model is the regular sets of strings over a finite alphabet, elements of a free monoid; for KAT, the regular sets of guarded strings; for NetKAT, the regular sets of strings of a certain reduced form. Once a suitable class of language models can be determined, this opens the door to a systematic treatment of deduction and coalgebraic decision algorithms. The question thus presents itself: Is there a general set of criteria that admit a uniform construction of language models and that would apply in a broad range of situations and subsume previous ad hoc constructions? That is the subject of this paper.

Alas, such a grand unifying framework is unlikely, given the negative results of [12] and of §2. However, we have identified a framework that goes quite far in this direction. It applies in the case in which the additional equational axioms are monoid equations or partial monoid equations (as is the case in all the examples mentioned above) and is based on a well-studied class of rewrite systems called *inverse context-free systems* [13]. We give criteria in terms of these rewrite systems that imply the existence of free language models in a wide range of previously studied instances, as well as some new ones.

This paper is organized as follows. In §2 we present preliminary definitions and our negative result limiting the applicability of the method. In §3 we establish a connection between the classical theory of string rewriting and Kleene algebra. We recall from [13] the definition of *inverse context-free rewrite systems* and the key result that they preserve regularity. The original proof involved an automata-theoretic construction, but we show that it can be carried out axiomatically in KA. In §4 we give examples of partial and total monoid equations and give a general construction that establishes completeness in those cases. The construction is a special case of the more general results of §5, but we start with it as a conceptual first step to illustrate the ideas. However, we can already derive some interesting consequences in this special case. In §5, we establish completeness for typed monoid equations. This is the most general setting covered in this paper. We give the completeness proof along with several applications. In §6 we present conclusions, future work, and open problems.

Omitted proofs can be found in the appendix.

2 Preliminaries and a Negative Result

Let Σ be a finite alphabet of symbols. The *free monoid* $(\Sigma^*, \cdot, \epsilon)$ generated by Σ is the set Σ^* of strings or words over Σ together with the operation $\cdot$ of string concatenation and the empty string ϵ as identity. To generalize this construction, we consider a *finitely presented monoid* $M = \langle a, b, \ldots \mid u_1 \equiv u_2, v_1 \equiv v_2, \ldots\rangle$ with a finite set of generators $\Sigma = \{a, b, \ldots\}$ and a finite set of relations $R = \{(u_1, u_2), (v_1, v_2), \ldots\}$. We interchangeably write a relation as an equation $u \equiv u'$ or as a pair (u, u'). Let $\leftrightarrow_R^*$ be the smallest congruence on Σ^* that contains R. The congruence class of a string u is denoted by $[u]$. The finitely presented monoid $M = \langle \Sigma \mid R\rangle = \Sigma^*/R$ has the congruence classes $\{[u] \mid u \in \Sigma^*\}$ of $\leftrightarrow_R^*$ as its carrier. Multiplication is given by $[u] \cdot [v] \mapsto [uv]$, and the identity is $[\epsilon]$.

We define *regular expressions* over the alphabet Σ to be the terms given by the grammar $e, e_1, e_2 ::= a \in \Sigma \mid 1 \mid 0 \mid e_1 + e_2 \mid e_1; e_2 \mid e^*$. We can interpret a regular expression as a subset of a finitely presented monoid $M = \langle \Sigma \mid R\rangle$ with multiplication $\cdot$ and identity $1_M = [\epsilon]$. The function $\mathcal{R}_M$, called *language interpretation* in M, sends a regular expression to a set of elements of M:

$$\begin{aligned} \mathcal{R}_M(a) &= \{[a]\} & \mathcal{R}_M(e_1 + e_2) &= \mathcal{R}_M(e_1) \cup \mathcal{R}_M(e_2) \\ \mathcal{R}_M(1) &= \{1_M\} & \mathcal{R}_M(e_1; e_2) &= \mathcal{R}_M(e_1) \cdot \mathcal{R}_M(e_2) \\ \mathcal{R}_M(0) &= \emptyset & \mathcal{R}_M(e^*) &= \textstyle\bigcup_{n \geq 0} \mathcal{R}_M(e)^n \end{aligned}$$

where $\cdot$ on sets is given by $A \cdot B = \{u \cdot v \mid u \in A, v \in B\}$, and A^n is defined inductively as $A^0 = \mathcal{R}_M(1)$ and $A^{n+1} = A^n \cdot A$. The image of the interpretation $\mathcal{R}_M$ together with the operations $\cup$, $\cdot$, *, $\emptyset$, $\{1_M\}$ is the *algebra of regular sets* over M, denoted by $\mathsf{Reg}\, M$. If M is the free monoid Σ^*, then $\mathcal{R}_M$ is the standard language interpretation of regular expressions.

It is known that the algebra of regular sets $\mathsf{Reg}\, \Sigma^*$ is the free Kleene algebra generated by Σ [11]. This is equivalent to the completeness of the axioms of KA for the standard language interpretation $\mathcal{R}$ of regular expressions. That is, for any two regular expressions e_1, e_2 over Σ, if $\mathcal{R}(e_1) = \mathcal{R}(e_2)$ then $\mathsf{KA} \vdash e_1 \equiv e_2$. The question then arises if this result extends to the general case of $\mathsf{Reg}\, M$ for a finitely presented monoid $M = \langle \Sigma \mid R \rangle$. We ask the question of whether $\mathcal{R}_M(e_1) = \mathcal{R}_M(e_2)$ implies provability of $e_1 \equiv e_2$ in a system of KA augmented with (at least) the equations corresponding to the relations R.

In general, the answer to the question posed in the previous paragraph is negative. That is, there exists a finitely presented monoid $M = \langle \Sigma \mid R \rangle$ such that the equational theory of $\mathsf{Reg}\, M$ is not recursively enumerable, and therefore not recursively axiomatizable. The *equational theory* of the Kleene algebra $\mathsf{Reg}\, M$ is the set of equations between regular expressions that are true in $\mathsf{Reg}\, M$ under the interpretation $\mathcal{R}_M$, i.e., the set $\{e_1 \equiv e_2 \mid \mathcal{R}_M(e_1) = \mathcal{R}_M(e_2)\}$. We show this negative result using the ideas developed in [12]. The proof specifies a way to construct effectively the monoid whose existence we claim.

Theorem 1. There exists a finitely presented monoid M such that the equational theory of $\mathsf{Reg}\, M$ is not recursively enumerable.

This negative result says that we can only hope to identify subclasses of finitely presented monoids M such that the algebra $\mathsf{Reg}\, M$ of regular sets over M is axiomatizable. The idea is to first restrict attention to those finite monoid presentations, where the equations can be oriented to give a confluent and terminating rewrite system. This allows one to consider as canonical representatives the irreducible strings of the congruence classes. Then, we focus on a subclass that allows two crucial algebraic constructions: a "descendants" automata-theoretic construction, and an "ancestors" construction, which is a homomorphism.

3 String Rewriting Systems

In this section we establish a connection between the classical theory of string rewriting systems and Kleene algebra. More specifically, we recall a result regarding the preservation of regularity: for every *inverse context-free* system R and a regular set L, the set of the R-descendants of L is also regular [13]. This result involves an automata-theoretic construction, which can be modeled in KA, because an automaton can be represented as an approprivate KA term [11]. The combinatorial arguments of the construction can then be replaced by equational reasoning in KA. As it turns out, this connection will allow us to obtain powerful completeness metatheorems in later sections.

A *string rewriting system* R over a finite alphabet Σ consists of rules $\ell \to r$, where ℓ and r are finite strings over Σ. This extends to the *one-step rewrite relation* $\to_R$, given by $x\ell y \to_R xry$, for strings x, y and rule $\ell \to r$ of R. If $x \to_R y$ then we say that y is an *R-successor* of x, and x is an *R-predecessor* of y. We write $\to_R^*$ for the reflexive-transitive closure of $\to_R$, which is called the *rewrite relation* for R. If u, v are strings for which $u \to_R^* v$ we say that v is an *R-descendant* of u, and that u is an *R-ancestor* of v. For a set of strings L:

$$\mathsf{Desc}_R(L) = \{v \mid \exists u \in L.\ u \to_R^* v\} \qquad \mathsf{Ance}_R(L) = \{u \mid \exists v \in L.\ u \to_R^* v\}$$

So, $\mathsf{Desc}_R(L)$ is the set of all the R-descendants of the strings in L, and similarly $\mathsf{Ance}_R(L)$ is the set of all R-ancestors of the strings in L. The *inverse system* R^{-1} of R is the system that results by taking a rule $r \to \ell$ for every rule $\ell \to r$ of R. If u is an R-ancestor of a string v, then u is an R^{-1}-descendant of v. Define $\leftrightarrow_R^*$ to be the smallest congruence on Σ^* that contains $\{(u, v) \mid u \to v \text{ is } R\text{-rule}\}$. The congruence class of a string u is denoted by $[u]$.

Lemma 1. Let R be a rewrite system consisting of rules of the form $a \to r$, where a is a letter. Assume further that every set $\mathsf{Desc}_R(a)$ is regular with $\mathcal{R}(e_a) = \mathsf{Desc}_R(a)$ for some regular expression e_a. Define the substitution θ by $a \mapsto e_a$, and extend it to all expressions. Then, $\mathsf{Desc}_R(\mathcal{R}(e)) = \mathcal{R}(\theta(e))$.

Let R be a rewrite system. We say that R is *terminating* if there is no infinite rewrite chain $x_0 \to_R x_1 \to_R x_2 \to_R \cdots$. If R has rules of the form $\ell \to r$ with $|r| < |\ell|$ then it is terminating, because every rule application strictly reduces the length of the string. A string x is called *R-irreducible* if no rule of R applies to it, that is, there is no y with $x \to_R y$. We say that R is *confluent* if $u \to_R^* x$ and $u \to_R^* y$ imply that there exists z with $x \to_R^* z$ and $y \to_R^* z$. It is said that R has the *Church-Rosser property* (we also say that "R is Church-Rosser") if for all strings x, y with $x \leftrightarrow_R^* y$ there exists z such that $x \to_R^* z$ and $x \to_R^* z$. It is a standard result that confluence and the Church-Rosser property are equivalent [13]. A system R is said to be *locally (or weakly) confluent* if for all strings u, x, y with $u \to_R x$ and $u \to_R y$, there exists a string z such that $x \to_R^* z$ and $x \to_R^* z$. If R is both locally confluent and terminating, then R is confluent [13].

Suppose that R is confluent and terminating. We map each string u to the unique R-irreducible string $\mathsf{rd}_R(u)$ that results from rewriting u as much as possible. When the rewrite system R is apparent from context, we simply write $\mathsf{rd}(u)$ instead of $\mathsf{rd}_R(u)$. For strings u, v, it holds that $u \leftrightarrow_R^* v$ iff $\mathsf{rd}(u) = \mathsf{rd}(v)$. So, two strings are congruent iff they can be rewritten to the same R-irreducible. For every congruence class $[u]$ of $\leftrightarrow_R^*$, we choose for *canonical representative* (normal form) the R-irreducible string $\mathsf{rd}(u)$.

Definition 1 (Total Fusion Product). Assume that R is confluent and terminating. We take I_R to be the set of R-irreducible strings. Define the binary operation $\diamond$ on I_R, which we call *fusion product*, by $u \diamond v = \mathsf{rd}(uv)$. The structure $(I_R, \diamond, \mathsf{rd}(\epsilon))$ is a monoid. We lift the operation of fusion product to sets of irreducible strings as $A \diamond B = \{u \diamond v \mid u \in A, v \in B\}$.

Definition 2. Let R be an arbitrary string rewriting system. For a language $L \subseteq \Sigma^*$, we define $\mathscr{C}_R(L) = \bigcup_{u \in L}[u] = \{v \mid \exists u \in L.\ v \leftrightarrow_R^* u\}$. We note that $\mathscr{C}_R(L)$, which is a set of strings, is *not* equal to $\{[u] \mid u \in L\}$, which is a set of equivalence classes of strings. Assume additionally that R is confluent and terminating, so that the function rd_R is well-defined. For $L \subseteq \Sigma^*$, we define $\mathscr{G}_R(L) = \{\mathsf{rd}_R(u) \mid u \in L\}$, which is a set of R-irreducible strings.

Lemma 2. Let R be a confluent and terminating rewrite system over Σ.
1. $\mathscr{C}_R(L) = \bigcup\{[u] \mid u \in \mathscr{G}_R(L)\}$, for a language $L \subseteq \Sigma^*$.
2. $\mathscr{G}_R(L_1) = \mathscr{G}_R(L_2)$ iff $\mathscr{C}_R(L_1) = \mathscr{C}_R(L_2)$, for languages $L_1, L_2 \subseteq \Sigma^*$.
3. $\mathscr{C}_R(L) = \mathsf{Ance}_R(\mathsf{Desc}_R(L))$, for a language $L \subseteq \Sigma^*$.

A rewrite system R is said to *preserve regularity* if for every regular language L, the R-descendants $\mathsf{Desc}_R(L)$ for a regular set. A system R is called *inverse context-free* if it only contains rules of the form $\ell \to r$, where $|r| \leq 1$. That is, every right-hand side of a rule is either a single letter or the empty string. A classical result of the theory of string rewriting is that inverse context-free systems preserve regularity (see Chapter 4 of [13] for a detailed proof). The proof of this fact uses a construction on finite automata, which we briefly present here. We will be referring to it as the *descendants construction.* Suppose that L is a regular language, recognized by an automaton $\mathcal{A}$. The automaton is possibly nondeterministic and it may have epsilon transitions. We will describe a sequence of transformations on $\mathcal{A}$. When the sequence reaches a fixpoint, we obtain an automaton (nondeterministic with epsilon transitions) that recognizes $\mathsf{Desc}_R(L)$.

– Suppose that the system R has a rule $\ell \to a$, where a is a single letter, and $\ell = \ell_1\ell_2\cdots\ell_m$ is a string of length m. We assume that there is an ℓ-path from the state q_0 to the state q_n of the automaton. That is, a sequence $q_0 \xrightarrow{x_1} q_1 \xrightarrow{x_2} q_2 \xrightarrow{x_3} \cdots \xrightarrow{x_{n-1}} q_{n-1} \xrightarrow{x_n} q_n$, where each x_i is a letter or ϵ, $x_1 \cdot x_2 \cdots x_{n-1} \cdot x_n = \ell$, and each $q_{i-1} \xrightarrow{x_i} q_i$ is a transition of the automaton. We add the transition $q_0 \xrightarrow{a} q_n$. The idea is that if the automaton accepts a string $x\ell y$, then it should also accept the R-descendant xay.

– Similarly, suppose that the system R has a rule $\ell \to \epsilon$, where ϵ is the empty string, and that there is an ℓ-path from the state q_0 to the state q_n. Then, we add the epsilon transition $q_0 \xrightarrow{\epsilon} q_n$ to the transition table of the automaton.

This process is iterated until no new transitions are added. The resulting automaton accepts exactly the set of R-descendants $\mathsf{Desc}_R(L)$.

Theorem 2. Let R be an inverse context-free rewrite system, and e be a regular expression, whose interpretation is $L = \mathcal{R}(e)$. We can construct effectively a new regular expression $\hat{e}$ such that $\mathsf{KA}_R \vdash e \equiv \hat{e}$ and $\mathcal{R}(\hat{e}) = \mathsf{Desc}_R(L)$. KA_R is the system KA augmented with an equation $\ell \equiv r$ for every rewrite rule $\ell \to r$ of R.

Theorem 2 says that the descendants construction, which is combinatorial, can be modeled algebraically in the system of KA with some extra equations. This is a central technical result that we will use for our later theorems.

4 Completeness: (Partial) Monoid Equations

In this section we present our first completeness metatheorems, from which we can prove the existence of free language models for systems of KA with extra monoid and partial monoid equations. Our metatheorems are not only a conceptual first step towards the more general typed monoid case, which we investigate in §5, but they also allow us to obtain previously unknown completeness results. As a concrete novel application, think of the assignment statement $x := c$, where c is a constant. The action $x := c$ is idempotent, meaning that the effect of $x := c; x := c$ is the same as the effect of $x := c$. We express this fact with the monoid equation $aa \equiv a$, where a is a single letter abstraction of the assignment. KA can be augmented with any number of such idempotence equations, and our metatheorem implies the existence of a free language model (see Example 1).

Definition 3 (Language Interpretation). Let R be a confluent and terminating rewrite system. The corresponding fusion product is $\diamond$. We define inductively the function $\mathcal{G}_R$ that sends a regular expression to a set of R-irreducibles:

$$\begin{aligned} \mathcal{G}_R(a) &= \{\mathsf{rd}_R(a)\} & \mathcal{G}_R(e_1 + e_2) &= \mathcal{G}_R(e_1) \cup \mathcal{G}_R(e_2) \\ \mathcal{G}_R(0) &= \emptyset & \mathcal{G}_R(e_1; e_2) &= \mathcal{G}_R(e_1) \diamond \mathcal{G}_R(e_2) \\ \mathcal{G}_R(1) &= \{\mathsf{rd}_R(\epsilon)\} & \mathcal{G}_R(e^*) &= \textstyle\bigcup_{n \geq 0} \mathcal{G}_R(e)^{\langle n \rangle} \end{aligned}$$

where, for a set A of R-irreducibles, $A^{\langle n \rangle}$ is defined by $A^{\langle 0 \rangle} = \mathcal{G}_R(1)$ and $A^{\langle n+1 \rangle} = A^{\langle n \rangle} \diamond A$. We also define the interpretation $\mathcal{C}_R(e) = \mathscr{C}_R(\mathcal{R}(e)) = \bigcup_{u \in \mathcal{R}(e)} [u]$.

Let R be a confluent and terminating system over Σ, and $M = \langle \Sigma \mid R \rangle$ be the corresponding monoid. For a regular expression e, we have that $\mathcal{R}_M(e) = \{[u] \mid u \in \mathcal{G}_R(e)\}$. The algebra $\mathsf{Reg}\, M$ is isomorphic to the algebra that is the image of $\mathcal{G}_R$. This implies that $\mathcal{R}_M(e_1) = \mathcal{R}_M(e_2)$ iff $\mathcal{G}_R(e_1) = \mathcal{G}_R(e_2)$. So, our investigations of semantic completeness can be with respect to the interpretation $\mathcal{G}_R$.

Lemma 3. Let R be a confluent and terminating string rewrite system.
1. $\mathcal{G}_R(e) = \{\mathsf{rd}_R(u) \mid u \in \mathcal{R}(e)\} = \mathscr{G}_R(\mathcal{R}(e))$, for an expression e.
2. $\mathcal{C}_R(e) = \bigcup\{[v] \mid v \in \mathcal{G}_R(e)\}$, for an expression e.
3. $\mathcal{G}_R(e_1) = \mathcal{G}_R(e_2)$ iff $\mathcal{C}_R(e_1) = \mathcal{C}_R(e_2)$, for expressions e_1, e_2.

Definition 4 (Well-Behaved Rewrite System). Let R be a rewrite system over Σ. We say that R is *well-behaved* if it consists of finitely many rules $\ell \to r$ with $|r| = 1$ and $|\ell| > 1$, and it additionally satisfies confluence and the following property: For every letter a of the alphabet, the R-ancestors of a form a regular set $\mathcal{R}(e_a)$ for some expression e_a, so that $\mathsf{KA}_R \vdash e_a \equiv a$. Recall that KA_R is the system of KA extended with equations corresponding to the rules of R.

Intuitively, we say that R is well-behaved if it allows two important algebraic constructions. First, the special form of the rules allows the automata-theoretic descendants construction (described in §3), which can be modeled in KA, because automata can be encoded as matrices. Then, the regularity requirement for the sets of R-ancestors of single letters implies that we can apply a homomorphism to obtain all the ancestors of a regular set. We can thus "close" a regular expression under the congruence induced by R.

Theorem 3 (Completeness). *Let R be a well-behaved rewrite system over Σ. For any expressions e_1 and e_2, $\mathcal{G}_R(e_1) = \mathcal{G}_R(e_2)$ implies that $\mathsf{KA}_R \vdash e_1 \equiv e_2$.*

Example 1 (Idempotence Hypotheses). We will see how the general completeness metatheorem we have shown (Theorem 3) can be used to obtain a completeness result for the regular algebra of a simple finitely presented monoid. Consider the monoid $M = \langle a, b \mid aa \equiv a \rangle$. The rewrite system R contains only the rule $aa \to a$. In order to invoke Theorem 3 we verify that R is well-behaved:
- For the only rule $\ell = aa \to a = r$ of R, we have that $|r| = 1$ and $|\ell| > 1$.
- To show confluence of R, it is sufficient to show local confluence, since R is terminating. Suppose that $u \to x$ and $u \to y$. If $x = y$, we are done. If $x \neq y$, then u, x, y must be of the following forms: $u = v_1 a^{m+1} v_2 a^{n+1} v_3$, $x = v_1 a^m v_2 a^{n+1} v_3$, and $y = v_1 a^{m+1} v_2 a^n v_3$. Notice now that $x, y \to v_1 a^m v_2 a^n v_3$, which establishes local confluence.

- For the R-ancestors of the letters a and b, we see that $\mathsf{Ance}_R(b) = \{b\}$, and $\mathsf{Ance}_R(a) = \{a^i \mid i \geq 1\} = \mathcal{R}(a^+)$, where $a^+ = a; a^*$. We put $e_b = b$ and $e_a = a^+$. Clearly, $\mathsf{KA}_R \vdash e_b \equiv b$. Reasoning in KA_R: $a \leq a^+$ and $a^+ = a; a^* \leq a \Longleftarrow a; a \leq a \Longleftarrow a; a \equiv a$. We have thus shown that $\mathsf{KA}_R \vdash e_a \equiv a$.

Since the rewrite system R satisfies the conditions of Theorem 3, we get completeness of KA together with the equation $a; a \equiv a$ for the interpretation $\mathcal{R}_M$.

We would like to generalize our result in a way that allows us to designate certain strings as being *non-well-formed* or *undefined*. Any string with a non-well-formed substring has to be discarded from the interpretation. For a string $a_1 \cdots a_k$ over the alphabet, we declare it to be non-well-formed using the equation $a_1 \cdots a_k \equiv \bot$, where $\bot$ is a special "undefined" symbol.

We define a *partial monoid* to be an algebraic structure $(M, \cdot, 1_M, \bot_M)$ satisfying the monoid axioms, as well as the equations $x \cdot \bot_M = \bot_M$ and $\bot_M \cdot x = \bot_M$. The identity is 1_M, and $\bot_M$ is called the *undefined element* of M. In a presentation of a partial monoid $M_\bot = \langle \Sigma \mid x_1 \equiv y_1, x_2 \equiv y_2, \ldots, z_1 \equiv \bot, z_2 \equiv \bot, \ldots \rangle$ we allow equations $x \equiv y$ between strings over Σ (call the collection of these R), as well as equations of the form $z \equiv \bot$, where z is a string over Σ. In order to give a concrete description of the partial monoid, we consider the strings over the extended alphabet $\Sigma \cup \{\bot\}$, and the equations $R_\bot$:

$$x_i \equiv y_i \qquad z_i \equiv \bot \qquad a\bot \equiv \bot,\ \bot a \equiv \bot \quad (a \in \Sigma) \qquad \bot\bot \equiv \bot$$

Let $\sim$ be the smallest congruence on $(\Sigma \cup \{\bot\})^*$ that contains the relations $R_\bot$. The partial monoid $M_\bot$ is the set of strings $(\Sigma \cup \{\bot\})^*$ quotiented by the congruence $\sim$, and hence equal to $\langle \Sigma \cup \{\bot\} \mid R_\bot \rangle$. The identity is the $\sim$-congruence class $[\epsilon]$, and the undefined element is the class of $[\bot]$.

Assumption 1. We collect a list of assumptions for $(\Sigma, R, R_\bot)$. First, assume that R is a confluent and terminating rewrite system over the alphabet Σ. The rewrite system $R_\bot$ extends R with rules of the form $z \to \bot$, where $z \in \Sigma^*$ and $|z| \geq 2$. Moreover, $R_\bot$ contains the rule $\bot\bot \to \bot$, as well as all the rules $a\bot \to \bot$ and $\bot a \to \bot$ for every letter $a \in \Sigma$. We further assume that $R_\bot$ is terminating, and that the *seamlessness property* is satisfied: If xzy is a string with $z \to \bot$ in $R_\bot$, then any R-successor of xzy is of the form $x'z'y'$, where $z' \to \bot$ is in $R_\bot$. Intuitively, seamlessness says that if a string contains a non-well-formed substring, then no R-rewriting can make it well-formed.

Definition 5 (Partial Fusion Product). Let $(\Sigma, R, R_\bot)$ satisfy Assumption 1. Define the partial *fusion product* operation $\diamond$ on $R_\bot$-irreducibles in Σ^*:

$$u \diamond v = \mathsf{rd}_R(uv), \text{ if } uv \not\sim \bot; \qquad u \diamond v = \text{undefined}, \text{ if } uv \sim \bot.$$

The condition $uv \not\sim \bot$ is equivalent to $\mathsf{rd}_R(uv)$ not having a substring z with $z \to \bot$. We lift the fusion product into a total operation on sets of $R_\bot$-irreducibles: $A \diamond B = \{u \diamond v \mid u \diamond v \text{ exists}, u \in A, v \in B\}$.

Definition 6 (Language Interpretation). Let $(\Sigma, R, R_\bot)$ satisfy Assumption 1. For a string u, define $[u]_\Sigma = \Sigma^* \cap [u]$. For a language $L \subseteq \Sigma^*$, put:

$$\mathscr{G}_{R_\bot}(L) = \{\mathsf{rd}_R(u) \mid u \in L\} \setminus [\bot]_\Sigma \qquad \mathscr{C}_{R_\bot}(L) = [\bot]_\Sigma \cup \bigcup_{u \in L} [u]_\Sigma$$

Now, $\mathcal{G}_{R_\bot}$ sends a regular expression to a set of $R_\bot$-irreducibles of Σ^*:

$$\mathcal{G}_{R_\bot}(a) = \{\mathsf{rd}_R(a)\} \setminus [\bot]_\Sigma \qquad \mathcal{G}_{R_\bot}(e_1 + e_2) = \mathcal{G}_{R_\bot}(e_1) \cup \mathcal{G}_{R_\bot}(e_2)$$

$$\mathcal{G}_{R_\perp}(0) = \emptyset \qquad \mathcal{G}_{R_\perp}(e_1;e_2) = \mathcal{G}_{R_\perp}(e_1) \diamond \mathcal{G}_{R_\perp}(e_2)$$
$$\mathcal{G}_{R_\perp}(1) = \{\mathsf{rd}_R(\epsilon)\} \setminus [\perp]_\Sigma \qquad \mathcal{G}_{R_\perp}(e^*) = \bigcup_{n\geq 0} \mathcal{G}_{R_\perp}(e)^{\langle n\rangle}$$

where $A^{\langle 0\rangle} = \mathcal{G}_{R_\perp}(1)$ and $A^{\langle n+1\rangle} = A^{\langle n\rangle} \diamond A$. Define $\mathcal{C}_{R_\perp}(e) = \mathscr{C}_{R_\perp}(\mathcal{R}(e))$. The interpretation $\mathcal{G}_{R_\perp}$ discards the undefined strings, but $\mathcal{C}_{R_\perp}$ adds them all in.

Definition 7 (Well-Behaved). Suppose that $(\Sigma, R, R_\perp)$ satisfies Assumption 1. We say that it is *well-behaved* if $R_\perp$ consists of finitely many rules, every rule $\ell \to r$ of R satisfies $|r| = 1$ and $|\ell| > 1$, and it satisfies the property: For every letter a of the alphabet, the R-ancestors of a form a regular set $\mathcal{R}(e_a)$ for some regular expression e_a, so that $\mathsf{KA}_R \vdash e_a \equiv a$. The empty string and the single-letter strings are $R_\perp$-irreducible. We write $\mathsf{KA}_{R_\perp}$ for the system KA_R extended with an equation $a_1; \cdots ; a_k \equiv 0$ for every rewrite rule $a_1 \cdots a_k \to \perp$ of $R_\perp$.

Lemma 4 ($\perp$-class). Suppose that $(\Sigma, R, R_\perp)$ is well-behaved. The set $\Sigma^* \cap [\perp]$ is regular. For the corresponding expression $e_\perp$ is holds that $\mathsf{KA}_{R_\perp} \vdash e_\perp \equiv 0$.

Theorem 4 (Completeness). *Suppose that $(\Sigma, R, R_\perp)$ is well-behaved. Then, $\mathcal{G}_{R_\perp}(e_1) = \mathcal{G}_{R_\perp}(e_2)$ implies that $\mathsf{KA}_{R_\perp} \vdash e_1 \equiv e_2$.*

5 Completeness: Typed Monoid Equations

We further generalize the partial monoid setting by assuming more structure on the strings and the rewrite system. One major difference from the partial monoid case is the introduction of a new category of primitive symbols, the subidentities, which allow the encoding of Booleans. We show how to cover several examples: plain KAT, KAT with simple Hoare hypotheses $b;p;c \equiv 0$, KAT with hypotheses $c;p \equiv c$, and NetKAT. There are even more applications which for lack of space we do not present here: commutativity equations $b;p \equiv p;b$ (test b, atomic action p), Boolean equations $b \equiv c$ (tests b, c), and so on. These examples attest to the generality and wide applicability of our technique.

Assumption 2. We collect a list of assumptions for $(P, Id, R, R_\perp)$. Let $\Sigma = P \cup Id$ be a finite alphabet, whose symbols are partitioned into a set P of *action symbols* and a set Id of *subidentities*. We write $p, q, r, \ldots$ to vary over actions symbols, $\alpha, \beta, \gamma, \ldots$ to vary over subidentities, and $a, b, c, \ldots$ to vary over arbitrary symbols of Σ. Let S be the subset of Σ^* consisting of all strings in which an action symbol p always appears surrounded by subidentities, as in $\alpha p \beta$. Examples of elements of S are: α, $\alpha\alpha$, $\alpha\beta$, $\alpha p \alpha$, $\alpha p \beta$, $\alpha\alpha p \beta$, $\alpha\alpha p \beta q \gamma\gamma$, $\alpha\beta p \gamma$, and so on. The set S is regular, and the corresponding regular expression is $e_S = Id \cdot (Id^* \cdot P \cdot Id)^* \cdot Id^*$. Let R be a rewrite system over Σ that includes at least the rules $\alpha\alpha \to \alpha$ for every subidentity $\alpha \in Id$, and additionally it satisfies: (1) S is closed under $\to_R$: if $x \in S$ and $x \to_R y$ then $y \in S$. Moreover, S is closed under the inverse of $\to_R$: if $y \in S$ and $x \to_R y$ then $x \in S$. (2) For every rule $\ell \to r$ of R we have that $|\ell| > |r|$. (3) R is confluent on S: For $u, x, y \in S$, $u \to_R^* x$ and $u \to_R^* y$ imply that $x \to_R^* z$ and $y \to_R^* z$ for some $z \in S$. Now, suppose that $R_\perp$ extends R with the rules $\alpha\beta \to \perp$ for all subidentities $\alpha \neq \beta$, and possibly more rules of the form $z \to \perp$, where $z \in S$ and $|z| \geq 2$. Moreover, $R_\perp$ contains all the rules $a\perp \to \perp$, $\perp a \to \perp$ (for each $a \in \Sigma$), as well as the rule

$\bot\bot \to \bot$. We assume that $R_\bot$ satisfies additionally the *seamlessness property*: For $xzy \in S$ with $z \to \bot$ in $R_\bot$, any R-successor of xzy is of the form $x'z'y'$ for some rule $z' \to \bot$ of $R_\bot$. We will use the term *irreducible* (unqualified) to mean $R_\bot$-irreducible of S. Finally, define the function cp to send every letter a of Σ to a finite subset $\mathsf{cp}(a)$ of S, called the *components* of a. For a subidentity $\alpha \in \mathit{Id}$, we put $\mathsf{cp}(\alpha) = \{\alpha\}$. For an action symbol $p \in P$, we put $\mathsf{cp}(p) = \{\alpha p \beta \mid \alpha, \beta \in \mathit{Id}\}$.

Definition 8 (Language Interpretation). Let $(P, \mathit{Id}, R, R_\bot)$ satisfy Assumption 2. For a string u, we put $[u]_S = S \cap [u]$. For a language $L \subseteq S$, we define:

$$\mathscr{G}_{R_\bot}(L) = \{\mathsf{rd}_R(u) \mid u \in L\} \setminus [\bot]_S \qquad \mathscr{C}_{R_\bot}(L) = [\bot]_S \cup \textstyle\bigcup_{u \in L} [u]_S$$

The *fusion product* of irreducibles, written $\diamond$, is defined as in Definition 5. The interpretation $\mathcal{G}_{R_\bot}$ sends a regular expression to a set of irreducibles:

$$\begin{array}{ll} \mathcal{G}_{R_\bot}(a) = \mathsf{rd}_R(\mathsf{cp}(a)) \setminus [\bot]_S & \mathcal{G}_{R_\bot}(e_1 + e_2) = \mathcal{G}_{R_\bot}(e_1) \cup \mathcal{G}_{R_\bot}(e_2) \\ \mathcal{G}_{R_\bot}(0) = \emptyset & \mathcal{G}_{R_\bot}(e_1; e_2) = \mathcal{G}_{R_\bot}(e_1) \diamond \mathcal{G}_{R_\bot}(e_2) \\ \mathcal{G}_{R_\bot}(1) = \mathit{Id} & \mathcal{G}_{R_\bot}(e^*) = \bigcup_{n \geq 0} \mathcal{G}_{R_\bot}(e)^{\langle n \rangle} \end{array}$$

Define $\mathcal{C}_{R_\bot}(e) = \mathscr{C}_{R_\bot}(\mathcal{R}(e))$, for expressions e with $\mathcal{R}(e) \subseteq S$.

Definition 9 (Well-Behaved). Let $(P, \mathit{Id}, R, R_\bot)$ be a tuple satisfying Assumption 2. We say that the tuple is *well-behaved* if $R_\bot$ consists of finitely many rules, every rule $\ell \to r$ of R satisfies $|r| = 1$ and $|\ell| > 1$, and it satisfies the following property: For every letter a of the alphabet, the R-ancestors of a form a regular set $\mathcal{R}(e_a)$ for some regular expression e_a, so that $\mathsf{KA}_R \vdash e_a \equiv a$.

We define the finite collection E of *equations associated* with the well-behaved tuple $(P, \mathit{Id}, R, R_\bot)$ to contain: (1) an equation $x \equiv y$ for every rule $x \to y$ of R, (2) an equation $z \equiv 0$ for every rule $z \to \bot$ of $R_\bot$, as well as (3) the equation $\sum_{\alpha \in \mathit{Id}} \alpha \equiv 1$. We write KA_E for the system of KA augmented with the equations E. We can prove in KA_E the equation $\sum_{x \in \mathsf{cp}(a)} x \equiv a$ for every letter a.

Lemma 5 (Interpret within S). Let $(P, \mathit{Id}, R, R_\bot)$ be well-behaved, and E be the associated equations. Define the substitution θ by: $1 \mapsto \sum_\alpha \alpha$ and $a \mapsto \sum_{x \in \mathsf{cp}(a)} x$, for every letter $a \in \Sigma$. Let e be an arbitrary expression. It holds that $\mathcal{G}_{R_\bot}(e) = \mathcal{G}_{R_\bot}(\theta(e))$. Moreover, $\mathcal{G}_{R_\bot}(e) = \mathcal{G}_{R_\bot}(e; \sum_\alpha \alpha)$. For the expression $\tilde{e} = \theta(e); \sum_\alpha \alpha$ we have that $\mathcal{G}_{R_\bot}(\tilde{e}) = \mathcal{G}_{R_\bot}(e)$, $\mathcal{R}(\tilde{e}) \subseteq S$, and $\mathsf{KA}_E \vdash \tilde{e} \equiv e$.

Lemma 6 ($\bot$-Class). Let $(P, \mathit{Id}, R, R_\bot)$ be well-behaved. The set $S \cap [\bot]$ is regular. For the corresponding expression $e_\bot$ it holds that $\mathsf{KA}_{R_\bot} \vdash e_\bot \equiv 0$.

Theorem 5 (Completeness). *Let $(P, \mathit{Id}, R, R_\bot)$ be well-behaved, and E be the associated equations. Then, $\mathcal{G}_{R_\bot}(e_1) = \mathcal{G}_{R_\bot}(e_2)$ implies that $\mathsf{KA}_E \vdash e_1 \equiv e_2$.*

5.1 Applications

Theorem 5 will give us four completeness results as corollaries. First, we will show that KAT is complete for the standard interpretation of KAT expressions as sets of guarded strings. We then extend this result to the cases where KAT is augmented with simple Hoare hypotheses $b; p; c \equiv 0$ (tests b, c, atomic action p), and with hypotheses $c; p \equiv c$ (test c, atomic action p). We conclude with a completeness proof for NetKAT.

Theorem 6. Let $\mathcal{G}_{\mathsf{KAT}}$ be the standard interpretation of KAT expressions. For any e_1 and e_2, it holds that $\mathcal{G}_{\mathsf{KAT}}(e_1) = \mathcal{G}_{\mathsf{KAT}}(e_2)$ implies $\mathsf{KAT} \vdash e_1 \equiv e_2$.

A *simple Hoare assertion* is an expression $\{b\}p\{c\}$, where b, c are tests and p is an atomic action. It can be encoded in KAT with the equation $b; p; \neg c \equiv 0$. This equation is equivalent to the conjunction of the equations $\beta; p; \gamma \equiv 0$, where β, γ are atoms with $\beta \leq b$ and $\gamma \leq \neg c$. So, w.l.o.g. we restrict attention to assertions of the form $\beta; p; \gamma \equiv 0$, where β, γ are atoms and p is an atomic action.

Theorem 7. Let Z_h be a finite collection of strings of the form $\gamma p \delta$, where γ, δ are atoms and p is an atomic action symbol. Let W be the set of strings containing some $\gamma p \delta$ in Z_h, and H be the collection of equations $\gamma; p; \delta \equiv 0$ for every $\gamma p \delta$ in Z_h. Define the interpretation $\mathcal{G}_h$ by $\mathcal{G}_h(e) = \mathcal{G}_{\mathsf{KAT}}(e) \setminus W$, which intuitively discards the guarded strings that violate the Hoare hypotheses. Then, $\mathcal{G}_h(e_1) = \mathcal{G}_h(e_2)$ implies $\mathsf{KAT} + H \vdash e_1 \equiv e_2$, where $\mathsf{KAT} + H$ is the system of KAT augmented with the Hoare hypotheses H.

We consider now another class of equations of the form $c; p \equiv c$, where c is a test and p is an atomic action. We see that $c; p \equiv c$ is equivalent to the conjunction of $\gamma; p \equiv \gamma$ for $\gamma \leq c$. So, we can restrict our attention to equations of the form $\gamma; p \equiv \gamma$, where γ is an atom, and p is an atomic action.

Theorem 8. Let X be a finite set of strings of the form γp, where γ is an atom and p is an atomic action symbol, and H be the set of equations $\gamma; p \equiv \gamma$ for every γp in X. For an atomic action symbol p, define the set of atoms $A(p) = \{\gamma \mid \gamma p \in X\}$. Intuitively, $A(p)$ is the set of atoms after which it is redundant to execute the action p. Let $\mathcal{G}_h$ be the interpretation that differs from $\mathcal{G}_{\mathsf{KAT}}$ only for the base case of atomic action symbols: $\mathcal{G}_h(p) = A(p) \cup \{\gamma p \delta \mid \gamma \notin A(p)\}$. Then, $\mathcal{G}_h(e_1) = \mathcal{G}_h(e_2)$ implies $\mathsf{KAT}{+}H \vdash e_1 \equiv e_2$, for any KAT expressions e_1, e_2.

We turn to the case of NetKAT. Fix an alphabet At of atoms. For $\alpha \in At$ we introduce an action symbol p_α, and we put $P = \{p_\alpha \mid \alpha \in At\}$. Let dup be a new action symbol, and set $\Sigma = P \cup \{\mathsf{dup}\} \cup At$. NetKAT extends KA with:

$$\begin{array}{ccc} \sum_{\alpha \in At} \alpha \equiv 1 & \alpha; \mathsf{dup} \equiv \mathsf{dup}; \alpha & p_\alpha \equiv p_\alpha; \alpha \\ \alpha; \beta \equiv 0 \ (\alpha \neq \beta) & p_\alpha; p_\beta \equiv p_\beta & \alpha \equiv \alpha; p_\alpha \end{array}$$

The axioms imply $\alpha; \alpha \equiv \alpha; p_\alpha; \alpha \equiv \alpha; p_\alpha \equiv \alpha$, for every atom α. So, NetKAT can also be defined as an extension of KAT. The following axioms

$$\begin{array}{cccc} \sum_{\alpha \in At} \alpha \equiv 1 & \alpha; \alpha \equiv \alpha & a; p_\alpha; \alpha \equiv \alpha & \alpha; \mathsf{dup}; \beta \equiv 0 \ (\alpha \neq \beta) \\ & \alpha; \beta \equiv 0 \ (\alpha \neq \beta) & p_\alpha; \alpha; p_\beta \equiv p_\beta & \alpha; p_\beta; \gamma \equiv 0 \ (\beta \neq \gamma) \end{array}$$

give an equivalent axiomatization of NetKAT (see Lemma 11 in the appendix).

Theorem 9. Let At be the subidentities (atoms), and $P' = P \cup \{\mathsf{dup}\}$ be the alphabet of action symbols, where $P = \{p_\alpha \mid \alpha \in At\}$. Define R and $R_\perp$ as:

$$\begin{array}{ccc} \alpha\alpha \to \alpha \ (\alpha \in At) & \alpha p_\alpha \alpha \to \alpha \ (\alpha \in At) & p_\alpha \alpha p_\beta \to p_\beta \ (\alpha, \beta \in At) \\ \alpha\beta \to \perp \ (\alpha \neq \beta) & \alpha \mathsf{dup} \beta \to \perp \ (\alpha \neq \beta) & \alpha p_\beta \gamma \to \perp \ (\beta \neq \gamma) \end{array}$$

$(P', At, R, R_\perp)$ is well-behaved, and NetKAT is complete for $\mathcal{G}_{R_\perp}$.

6 Conclusion

We have identified sufficient conditions for the construction of free language models for systems of Kleene algebra with additional equations. The construction provides a uniform approach to deductive completeness and coalgebraic decision procedures. The criteria are given in terms of *inverse context-free rewrite systems* [13]. They imply the existence of free language models in a wide range of previously studied instances, including KAT [6] and NetKAT [8], as well as some new ones. We have also given a negative result that establishes a limit to the applicability of the technique.

Acknowledgments

We acknowledge Bjørn Grathwohl and Stathis Zachos for helpful remarks.

References

1. Angus, A., Kozen, D.: Kleene algebra with tests and program schematology. Technical Report TR2001-1844, Computer Science Department, Cornell University (July 2001)
2. Barth, A., Kozen, D.: Equational verification of cache blocking in LU decomposition using Kleene algebra with tests. Technical Report TR2002-1865, Computer Science Department, Cornell University (June 2002)
3. Cohen, E.: Hypotheses in Kleene algebra. Technical Report TM-ARH-023814, Bellcore (1993) `http://citeseer.nj.nec.com/1688.html`.
4. Cohen, E.: Lazy caching in Kleene algebra (1994) `http://citeseer.nj.nec.com/22581.html`.
5. Cohen, E.: Using Kleene algebra to reason about concurrency control. Technical report, Telcordia, Morristown, N.J. (1994)
6. Kozen, D.: Kleene algebra with tests. Transactions on Programming Languages and Systems **19**(3) (May 1997) 427–443
7. Kozen, D., Patron, M.C.: Certification of compiler optimizations using Kleene algebra with tests. In: Proceedings of the First International Conference on Computational Logic. CL '00 (2000) 568–582
8. Anderson, C.J., Foster, N., Guha, A., Jeannin, J.B., Kozen, D., Schlesinger, C., Walker, D.: NetKAT: Semantic foundations for networks. In: Proc. 41st ACM SIGPLAN-SIGACT Symp. Principles of Programming Languages (POPL'14), San Diego, California, USA, ACM (January 2014) 113–126
9. Kozen, D., Smith, F.: Kleene algebra with tests: Completeness and decidability. In van Dalen, D., Bezem, M., eds.: Proc. 10th Int. Workshop Computer Science Logic (CSL'96). Volume 1258 of Lecture Notes in Computer Science., Utrecht, The Netherlands, Springer-Verlag (September 1996) 244–259
10. Hardin, C., Kozen, D.: On the elimination of hypotheses in Kleene algebra with tests. Technical Report TR2002-1879, Computer Science Department, Cornell University (October 2002)
11. Kozen, D.: A completeness theorem for Kleene algebras and the algebra of regular events. Information and Computation **110**(2) (1994) 366–390
12. Kozen, D.: On the complexity of reasoning in Kleene algebra. Information and Computation **179**(2) (2002) 152–162
13. Book, R.V., Otto, F.: String-Rewriting Systems. Springer-Verlag (1993)

Appendix – Omitted Proofs

*Proof (**Theorem 1**).* We consider the complement of the halting problem, the problem NotHalting: Given a Turing machine $\mathcal{M}$ (or its index) and an input u (which is a natural number), does $\mathcal{M}$ diverge on input u? The problem NotHalting is Π^0_1-complete, that is, co-r.e.-complete.

Define the Turing machine $\mathcal{M}_H$, which takes as input a triple of natural numbers (n, u, t). The number n is interpreted as the index of a Turing machine $\mathcal{M}_n$, the number u is meant to be given as input to $\mathcal{M}_n$, and t is a timeout. In order to encode (n, u, t) as a string, we take as input alphabet the set $\Sigma = \{a, b, c, \#\}$. We encode (n, u, t) as the string $a^n\#b^u\#c^t$. We describe now the algorithm that $\mathcal{M}_H$ implements. Let x be the input.

– If the string input is not of appropriate form then halt. If it is of the form $a^n\#b^u\#c^t$, then set n, u, t appropriately.
– From the index n compute the description of the Turing machine $\mathcal{M}_n$. Modify the machine to take a second input t (in addition to input u), which has the role of a timeout. The modified machine counts steps and if step t is reached before halting normally, then it halts with a special indication 'timeout'. Call this machine $\mathcal{M}'_n$.
– Simulate the execution $\mathcal{M}'_n(u, t)$. If this terminates normally, then diverge (enter an infinite loop). If it terminates with 'timeout', then halt.

Now, notice the equivalences:

$$\begin{aligned}
&\text{the pair } (n, u) \text{ belongs to NotHalting} \iff \\
&\mathcal{M}_n \text{ diverges on input } u \iff \\
&\forall t.\ \mathcal{M}'_n \text{ terminates with `timeout' on input } (u, t) \iff \\
&\forall t.\ \mathcal{M}_H \text{ halts on input } (n, u, t)
\end{aligned}$$

It is shown in [12] that for every Turing machine $\mathcal{M}$, there exists a finitely presented monoid Δ^*/E, which intuitively encodes the computations of the machine. For every input string x there exists an effectively computable equation $e_1; x; e_2 \equiv e$ such that $\mathcal{M}$ halts on input x iff $\Delta^*/E \models e_1; x; e_2 \equiv e$. All e_1, e_2, e are strings.

Suppose now that the monoid $M = \Delta^*/E$ is the one corresponding to the machine $\mathcal{M}_H$. The pair (n, u) belongs to NotHalting iff for every t, the machine $\mathcal{M}_H$ halts on input $a^n\#b^u\#c^t$ iff

$$\begin{aligned}
&\mathsf{Reg}\, M, \mathcal{R}_M \models e_1; a^n\#b^u\#c^t; e_2 \equiv e, \text{ for all } t \geq 0 \iff \\
&\mathsf{Reg}\, M, \mathcal{R}_M \models e_1; a^n\#b^u\#c^t; e_2 \leq e, \text{ for all } t \geq 0 \iff \\
&\mathsf{Reg}\, M, \mathcal{R}_M \models e_1; a^n\#b^u\#c^*; e_2 \leq e.
\end{aligned}$$

The last statement says that the equation $e_1; a^n\#b^u\#c^*; e_2 \leq e$ belongs to the equational theory of $\mathsf{Reg}\,(\Delta^*/E)$. Assume now for contradiction that this equational theory is recursively enumerable. Since NotHalting reduces to it, then NotHalting is also recursively enumerable. But NotHalting is co-r.e. and therefore decidable. This contradicts the fact that it is co-r.e.-hard. We have thus shown that the equational theory of $\mathsf{Reg}\,(\Delta^*/E)$ is not recursively enumerable.

Remark 1. Let R be a string rewrite system that has rules of the form $a \to r$, where a is a single letter. Let xy be a string. Every R-descendant of xy is of the form uv, where u (v) is an R-descendant of x (y). This can be expressed with the equation $\mathsf{Desc}_R(xy) = \mathsf{Desc}_R(x) \cdot \mathsf{Desc}_R(y)$. Using this property, we can prove its generalization $\mathsf{Desc}_R(L_1 \cdot L_2) = \mathsf{Desc}_R(L_1) \cdot \mathsf{Desc}_R(L_2)$ to sets of strings.

*Proof (**Remark 1**).* The proof for the first claim

$$\mathsf{Desc}_R(xy) = \mathsf{Desc}_R(x) \cdot \mathsf{Desc}_R(y)$$

is by a straightforward induction on the number of rule applications. The base case of no rule application is obvious. Suppose now that $xy \to_R^* uv$ with $u \in \mathsf{Desc}_R(x)$ and $v \in \mathsf{Desc}_R(y)$. Assume for the induction step that a rule $a \to r$ is further applied on a letter of u (similarly for v). So, $u \to_R u'$ for some u' and $uv \to_R u'v$. Notice that u' is an R-descendant of x. For the second part, we have

$$\begin{aligned}\mathsf{Desc}_R(L_1 \cdot L_2) &= \mathsf{Desc}_R(\{xy \mid x \in L_1, y \in L_2\}) \\ &= \textstyle\bigcup_{x \in L_1, y \in L_2} \mathsf{Desc}_R(xy) \\ &= \textstyle\bigcup_{x \in L_1, y \in L_2} \mathsf{Desc}_R(x) \cdot \mathsf{Desc}_R(y) \\ &= \mathsf{Desc}_R(L_1) \cdot \mathsf{Desc}_R(L_2).\end{aligned}$$

*Proof (**Lemma 1**).* The proof is by induction on the structure of e. Notice that there is no rule that can rewrite the empty string, therefore the only descendant of ϵ is ϵ. For the base cases we have:

$$\begin{aligned}\mathsf{Desc}_R(\mathcal{R}(a)) &= \mathsf{Desc}_R(\{a\}) = \mathcal{R}(e_a) = \mathcal{R}(\theta(a)) \\ \mathsf{Desc}_R(\mathcal{R}(1)) &= \mathsf{Desc}_R(\{\epsilon\}) = \{\epsilon\} = \mathcal{R}(1) = \mathcal{R}(\theta(1)) \\ \mathsf{Desc}_R(\mathcal{R}(0)) &= \mathsf{Desc}_R(\emptyset) = \emptyset = \mathcal{R}(0) = \mathcal{R}(\theta(0))\end{aligned}$$

For the case $e_1 + e_2$ we have:

$$\begin{aligned}\mathsf{Desc}_R(\mathcal{R}(e_1 + e_2)) &= \mathsf{Desc}_R(\mathcal{R}(e_1) \cup \mathcal{R}(e_2)) \\ &= \mathsf{Desc}_R(\mathcal{R}(e_1)) \cup \mathsf{Desc}_R(\mathcal{R}(e_2)) \\ &= \mathcal{R}(\theta(e_1)) \cup \mathcal{R}(\theta(e_2)) \\ &= \mathcal{R}(\theta(e_1) + \theta(e_2)) \\ &= \mathcal{R}(\theta(e_1 + e_2))\end{aligned}$$

For the case $e_1; e_2$ we use the equation shown in Remark 1 to obtain:

$$\begin{aligned}\mathsf{Desc}_R(\mathcal{R}(e_1; e_2)) &= \mathsf{Desc}_R(\mathcal{R}(e_1) \cdot \mathcal{R}(e_2)) \\ &= \mathsf{Desc}_R(\mathcal{R}(e_1)) \cdot \mathsf{Desc}_R(\mathcal{R}(e_2)) \\ &= \mathcal{R}(\theta(e_1)) \cdot \mathcal{R}(\theta(e_2)) \\ &= \mathcal{R}(\theta(e_1); \theta(e_2)) \\ &= \mathcal{R}(\theta(e_1; e_2))\end{aligned}$$

We handle now the case e^*. We claim that

$$\mathsf{Desc}_R(\mathcal{R}(e)^n) = \mathcal{R}(\theta(e))^n.$$

We argue by induction on n. For the base case, we have $\mathsf{Desc}_R(\mathcal{R}(e)^0) = \mathsf{Desc}_R(\{\epsilon\}) = \{\epsilon\} = \mathcal{R}(\theta(e))^n$. For the step we have:

$$\begin{aligned}\mathsf{Desc}_R(\mathcal{R}(e)^{n+1}) &= \mathsf{Desc}_R(\mathcal{R}(e)^n \cdot \mathcal{R}(e)) \\ &= \mathsf{Desc}_R(\mathcal{R}(e)^n) \cdot \mathsf{Desc}_R(\mathcal{R}(e)) \\ &= \mathcal{R}(\theta(e))^n \cdot \mathcal{R}(\theta(e)) \\ &= \mathcal{R}(\theta(e))^{n+1}.\end{aligned}$$

Finally, we obtain

$$\begin{aligned}\mathsf{Desc}_R(\mathcal{R}(e^*)) &= \mathsf{Desc}_R(\textstyle\bigcup_{n\geq 0} \mathcal{R}(e)^n) \\ &= \textstyle\bigcup_{n\geq 0} \mathsf{Desc}_R(\mathcal{R}(e)^n) \\ &= \textstyle\bigcup_{n\geq 0} \mathcal{R}(\theta(e))^n,\end{aligned}$$

which is equal to $\mathcal{R}(\theta(e)^*) = \mathcal{R}(\theta(e^*))$.

*Proof (**Lemma 2**).* We will be dropping the R subscripts freely for notational convenience. We show part (1). Using the fact $u \leftrightarrow_R^* \mathsf{rd}(u)$, we have that $[u] = [\mathsf{rd}(u)]$ and therefore

$$\mathscr{C}_R(L) = \textstyle\bigcup_{u\in L}[u] = \bigcup_{u\in L}[\mathsf{rd}(u)] = \bigcup_{v\in\mathscr{G}_R(L)}[v],$$

which establishes part (1).

We show part (2). The left-to-right direction is easy. Suppose that $\mathscr{G}(L_1) = \mathscr{G}(L_2)$. Then,

$$\mathscr{C}(L_1) = \textstyle\bigcup_{v\in\mathscr{G}(L_1)}[v] = \bigcup_{v\in\mathscr{G}(L_2)}[v] = \mathscr{C}(L_2).$$

For the right-to-left direction, it suffices by symmetry to show that $\mathscr{C}(L_1) \subseteq \mathscr{C}(L_2)$ implies $\mathscr{G}(L_1) \subseteq \mathscr{G}(L_2)$. Suppose that $\mathscr{C}(L_1) \subseteq \mathscr{C}(L_2)$, and let $\mathsf{rd}(u_1) \in \mathscr{G}(L_1)$, where $u_1 \in L_1$. We have that $[u_1] \subseteq \mathscr{C}(L_1) \subseteq \mathscr{C}(L_2)$ and therefore $\mathsf{rd}(u_1) \in \mathscr{C}(L_2)$. There exists $u_2 \in L_2$ such that $\mathsf{rd}(u_1) \in [u_2]$, and hence $\mathsf{rd}(u_1) \leftrightarrow_R^* u_2$. We conclude that $\mathsf{rd}(u_1) = \mathsf{rd}(u_2)$ is in $\mathscr{G}(L_2)$.

We show part (3). First, we see that $\mathsf{Ance}(\mathsf{Desc}(L))$ is contained in $\mathscr{C}(L)$. If $u \in \mathsf{Ance}(\mathsf{Desc}(L))$, then there is $v \in \mathsf{Desc}(L)$ with $u \to_R^* v$. There is also $u' \in L$ with $u' \to_R^* v$. It follows that $u \leftrightarrow_R^* v$ and $u' \leftrightarrow_R^* v$, therefore $u \leftrightarrow_R^* u'$. We thus obtain $u \in [u'] \subseteq \mathscr{C}(L)$. For the reverse containment, consider an arbitrary element u of $\mathscr{C}(L)$. There exists $u' \in L$ with $u \in [u']$, that is, $u \leftrightarrow_R^* u'$. But then we have that $v = \mathsf{rd}(u) = \mathsf{rd}(u')$. So, v is a descendant of both u and u'. Now, notice that $v \in \mathsf{Desc}(L)$ and since u is an ancestor of v, we conclude that $u \in \mathsf{Ance}(\mathsf{Desc}(L))$.

*Proof (**Theorem 2**).* In [11] it is shown how an arbitrary regular expression can be brought in "automaton form". The automaton is possibly nondeterministic and may have epsilon transitions. So, for the expression e there is a form $u; M^*; v$ with $\mathsf{KA} \vdash e \equiv u; M^*; v$, where u is a $1 \times n$ matrix, M is a $n \times n$ matrix, and v is a $n \times 1$ matrix. The matrix M is of the form $M = M(\epsilon) + \sum_a a \cdot M(a)$, where a ranges over the alphabet Σ and $a \cdot -$ denotes scalar multiplication. Each $n \times n$ matrix $M(a)$ encodes the transitions of the automaton on input symbol a. The entries of $M(a)$ are either 0 or 1, hence the entries of $a \cdot M(a)$ are either 0 or a.

We will show in KA_R that for a transformation step (as described in the previous section) from the automaton $u; M^*; v$ to the automaton $u; N^*; v$ we have that $\mathsf{KA}_R \vdash u; M^*; v \equiv u; N^*; v$. Suppose that $\ell \to r$ is a rule of R, $\ell = \ell_1 \ell_2 \cdots \ell_m$, and there is an ℓ-path from q_0 to q_n in the automaton:

$$q_0 \xrightarrow{x_1} q_1 \xrightarrow{x_2} \cdots \xrightarrow{x_{n-1}} q_{n-1} \xrightarrow{x_n} q_n,$$

with $x_1 \cdot x_2 \cdots x_{n-1} \cdot x_n = \ell$. Since each $q_{i-1} \xrightarrow{x_i} q_i$ is a transition of the automaton, we have that

$$\mathsf{row}(q_{i-1}); M(x_i); \mathsf{col}(q_i) \equiv 1.$$

The above equation says that the (q_{i-1}, q_i)-indexed entry of $M(x_i)$ is equal to 1. We write $\mathsf{row}(q)$ for the row vector that contains 1 at the q-indexed position and 0 in the rest of the positions. Similarly, $\mathsf{col}(q)$ is the column vector with 1 at position q and 0 elsewhere. It is easy to see that $\mathsf{row}(q); \mathsf{col}(q) \equiv 1$, and $\mathsf{col}(q_i); \mathsf{row}(q_j)$ is equal to the matrix with 1 at position (q_i, q_j) and 0 elsewhere. So, the inequality

$$\mathsf{col}(q_{i-1}); \mathsf{row}(q_i) \leq M(x_i)$$

is another way of expression the fact that $q_{i-1} \xrightarrow{x_i} q_i$ is a transition of the automaton. We define $N(a)$ so that $N(a) \equiv M(a) + \mathsf{col}(q_0); \mathsf{row}(q_n)$. This means that $N = M + a \cdot \mathsf{col}(q_0); \mathsf{row}(q_n)$. Since $M \leq N$, it follows by monotonicity of * that $M^* \leq N^*$ and hence $u; M^*; v \leq u; N^*; v$.

Now, we have to show that $u; N^*; v \leq u; M^*; v$, which is implied by $N^* \leq M^*$. In order to make our exposition more understandable, we give the proof using a specific example. Suppose we have the rule $\ell \to a$, where $\ell = ab$, and the ℓ-path we consider is $q_0 \xrightarrow{a} q_1 \xrightarrow{\epsilon} q_2 \xrightarrow{b} q_3$. We add the transition $q_0 \xrightarrow{a} q_3$ to the automaton. So, $N(a) \equiv M(a) + \mathsf{col}(q_0); \mathsf{row}(q_3)$, and $N \equiv M + a \cdot \mathsf{col}(q_0); \mathsf{row}(q_3)$. Notice:

$$\begin{aligned}
& a \cdot \mathsf{col}(q_0); \mathsf{row}(q_3) \equiv \\
& a \cdot \mathsf{col}(q_0); \mathsf{row}(q_1); \mathsf{col}(q_1); \mathsf{row}(q_3) \equiv \\
& a \cdot \mathsf{col}(q_0); \mathsf{row}(q_1); \mathsf{col}(q_1); \mathsf{row}(q_2); \mathsf{col}(q_2); \mathsf{row}(q_3) \leq \\
& a \cdot M(x_1); M(x_2); M(x_3) \equiv \\
& a; b \cdot M(a); M(\epsilon); M(b) \equiv \\
& aM(a); M(\epsilon); bM(b),
\end{aligned}$$

which is $\leq M; M; M$. It follows that $N \leq M + M; M; M$, and therefore $N^* \leq (M + M; M; M)^* \leq M^*$.

If the original automaton form is $u; M_0^*; v$, the descendants construction gives us a finite sequence of forms $u; M_0^*; v$, $u; M_1^*; v$, $\ldots$, $u; M_k^*; v$ with

$$\mathsf{KA}_R \vdash u; M_0^*; v \equiv u; M_1^*; v \equiv \cdots \equiv u; M_k^*; v.$$

No new transition can be added to the last automaton. So, the last automaton form of the sequence gives us all the descendants of $\mathcal{R}(e)$. That is, $\mathcal{R}(u; M_k^*; v) = \mathsf{Desc}_R(\mathcal{R}(u; M_0^*; v)) = \mathsf{Desc}_R(\mathcal{R}(e))$, because $\mathsf{KA} \vdash e \equiv u; M_0^*; v$. We put $\hat{e} = u; M_k^*; v$, and the proof is complete.

*Proof (**Lemma 3**).* We show part (1). The proof is by induction on the structure of e. For the bases cases we have $\mathcal{R}(a) = \{a\}$, $\mathcal{R}(0) = \emptyset$, $\mathcal{R}(1) = \{\epsilon\}$, and therefore

$$\begin{aligned}\mathcal{G}(a) &= \{a\} = \{\mathsf{rd}(a)\} = \{\mathsf{rd}(u) \mid u \in \mathcal{R}(a)\}\\ \mathcal{G}(0) &= \emptyset = \{\mathsf{rd}(u) \mid u \in \mathcal{R}(0)\}\\ \mathcal{G}(1) &= \{\epsilon\} = \{\mathsf{rd}(\epsilon)\} = \{\mathsf{rd}(u) \mid u \in \mathcal{R}(1)\}\end{aligned}$$

For the cases of choice and composition we have:

$$\begin{aligned}\mathcal{G}(e_1 + e_2) &= \mathcal{G}(e_1) \cup \mathcal{G}(e_2)\\ &= \{\mathsf{rd}(u) \mid u \in \mathcal{R}(e_1)\} \cup \{\mathsf{rd}(u) \mid u \in \mathcal{R}(e_2)\}\\ &= \{\mathsf{rd}(u) \mid u \in \mathcal{R}(e_1) \cup \mathcal{R}(e_2)\}\\ &= \{\mathsf{rd}(u) \mid u \in \mathcal{R}(e_1 + e_2)\}\\ \mathcal{G}(e_1; e_2) &= \mathcal{G}(e_1) \diamond \mathcal{G}(e_2)\\ &= \{u \diamond v \mid u \in \mathcal{G}(e_1), v \in \mathcal{G}(e_2)\}\\ &= \{\mathsf{rd}(x) \diamond \mathsf{rd}(y) \mid x \in \mathcal{R}(e_1), y \in \mathcal{R}(e_2)\}\\ &= \{\mathsf{rd}(\mathsf{rd}(x)\mathsf{rd}(y)) \mid x \in \mathcal{R}(e_1), y \in \mathcal{R}(e_2)\}\\ &= \{\mathsf{rd}(xy) \mid x \in \mathcal{R}(e_1), y \in \mathcal{R}(e_2)\}\\ &= \{\mathsf{rd}(z) \mid z \in \mathcal{R}(e_1; e_2)\}\end{aligned}$$

We handle now the case e^*. First, we claim that

$$G(e)^{\langle n \rangle} = \{\mathsf{rd}(u) \mid u \in \mathcal{R}(e)^n\}.$$

We show the claim by induction on n. For the base case, we have that $\mathcal{R}(e)^0 = \{\epsilon\}$ and therefore $G(e)^{\langle 0 \rangle} = \{\epsilon\} = \{\mathsf{rd}(\epsilon)\} = \{\mathsf{rd}(u) \mid u \in R(e)^0\}$. For the step, we have:

$$\begin{aligned}G(e)^{\langle n+1 \rangle} &= G(e)^{\langle n \rangle} \diamond G(e)\\ &= \{u \diamond v \mid u \in G(e)^{\langle n \rangle}, v \in G(e)\}\\ &= \{\mathsf{rd}(x) \diamond \mathsf{rd}(y) \mid x \in \mathcal{R}(e)^n, y \in \mathcal{R}(e)\}\\ &= \{\mathsf{rd}(xy) \mid x \in \mathcal{R}(e)^n, y \in \mathcal{R}(e)\}\\ &= \{\mathsf{rd}(z) \mid z \in \mathcal{R}(e)^{n+1}\}\end{aligned}$$

Finally, we have for the expression e^* that

$$\begin{aligned}G(e^*) &= \textstyle\bigcup_{n \geq 0} G(e)^{\langle n \rangle}\\ &= \textstyle\bigcup_{n \geq 0} \{\mathsf{rd}(u) \mid u \in \mathcal{R}(e)^n\}\\ &= \{\mathsf{rd}(u) \mid u \in \textstyle\bigcup_{n \geq 0} \mathcal{R}(e)^n\}\\ &= \{\mathsf{rd}(u) \mid u \in \mathcal{R}(e^*)\},\end{aligned}$$

thus completing the proof.

Now, using the definition of $\mathcal{C}(\cdot)$, Lemma 2(1), and part (1) of this lemma we have that:

$$\mathcal{C}(e) = \mathcal{C}_R(\mathcal{R}(e)) = \textstyle\bigcup_{v \in \mathcal{G}_R(\mathcal{R}(e))} [v] = \bigcup_{v \in \mathcal{G}(e)} [v],$$

which establishes part (2).

We show part (3). Part (1) says that $\mathcal{G}(e_i) = \mathcal{G}_R(\mathcal{R}(e_i))$. By definition of $\mathcal{C}(e_i)$ we have that $\mathcal{C}(e_i) = \mathcal{C}_R(\mathcal{R}(e_i))$. So, by Lemma 2(2) we obtain the equivalences: $\mathcal{G}(e_1) = \mathcal{G}(e_2)$ iff $\mathcal{G}_R(\mathcal{R}(e_1)) = \mathcal{G}_R(\mathcal{R}(e_2))$ iff $\mathcal{C}_R(\mathcal{R}(e_1)) = \mathcal{C}_R(\mathcal{R}(e_2))$ iff $\mathcal{C}(e_1) = \mathcal{C}(e_2)$.

*Proof (**Theorem 3**).* Consider the following transformation steps on a regular expression e:

(1) *Descendants*: As described in Theorem 2 we get an expression e' with $\mathsf{KA}_R \vdash e \equiv e'$ and $\mathcal{R}(e') = \mathsf{Desc}_R(\mathcal{R}(e))$.
(2) *Ancestors*: We describe below a transformation that gives us a new regular expression e'' with $\mathsf{KA}_R \vdash e' \equiv e''$ and $\mathcal{R}(e'') = \mathsf{Ance}_R(\mathcal{R}(e'))$.

We have $\mathsf{KA}_R \vdash e \equiv e''$ and $\mathcal{R}(e'') = \mathsf{Ance}_R(\mathsf{Desc}_R(\mathcal{R}(e)))$, which is equal to $\mathscr{C}_R(\mathcal{R}(e))$ by Lemma 2(3). It follows that $\mathcal{R}(e'') = \mathcal{C}_R(e)$.

We apply the above constructions to the expressions e_1 and e_2 to obtain the expressions e_1'' and e_2'' with:

$$\mathsf{KA}_R \vdash e_1 \equiv e_1'' \qquad\qquad \mathcal{C}_R(e_1) = \mathcal{R}(e_1'')$$
$$\mathsf{KA}_R \vdash e_2 \equiv e_2'' \qquad\qquad \mathcal{C}_R(e_2) = \mathcal{R}(e_2'')$$

From the hypothesis $\mathcal{R}_M(e_1) = \mathcal{R}_M(e_2)$ we have that $\mathcal{G}_R(e_1) = \mathcal{G}_R(e_2)$. Lemma 3 (part 3) then gives us that $\mathcal{C}_R(e_1) = \mathcal{C}_R(e_2)$. So, $\mathcal{R}(e_1'') = \mathcal{R}(e_2'')$ and by completeness of KA for the interpretation $\mathcal{R}$ we get that $\mathsf{KA} \vdash e_1'' \equiv e_2''$. Since we have proved in KA_R the equations

$$e_1 \equiv e_1'' \qquad\qquad e_1'' \equiv e_2'' \qquad\qquad e_2'' \equiv e_2$$

we conclude by transitivity that $\mathsf{KA}_R \vdash e_1 \equiv e_2$.

It remains to describe step (2) of the above transformation to complete the proof. If u is an R-ancestor of a string v, then u is an R^{-1}-descendant of v (and conversely). Since R is well-behaved, the system R^{-1} only contains rules of the form $a \to r$, where a is a letter. Moreover,

$$\mathsf{Ance}_R(a) = \mathsf{Desc}_{R^{-1}}(a) = \mathcal{R}(e_a)$$

for some regular expression e_a with $\mathsf{KA}_R \vdash e_a \equiv a$. Define the substitution θ by $a \mapsto e_a$. Lemma 1 gives us that

$$\mathsf{Ance}_R(\mathcal{R}(e')) = \mathsf{Desc}_{R^{-1}}(\mathcal{R}(e')) = \mathcal{R}(\theta(e')).$$

So, we put $e'' = \theta(e')$. We have already shown that $\mathcal{R}(e'') = \mathsf{Ance}_R(\mathcal{R}(e'))$. It remains to see that $\mathsf{KA}_R \vdash e' \equiv e'' = \theta(e')$, which is implied by $\mathsf{KA}_R \vdash e_a \equiv a$, for each letter a. The last statement is part of our hypothesis that R is well-behaved.

Lemma 7. Suppose that $(\Sigma, R, R_\perp)$ satisfies Assumption 1.
1. The system $R_\perp$ is confluent.
2. For $x \in \Sigma^*$: $x \sim \perp$ iff $\mathsf{rd}_R(x)$ contains some z with $z \to \perp$ a rule of $R_\perp$.
3. For $x, y \in \Sigma^*$, we have that $x \sim y$ iff ($x \sim \perp$ and $y \sim \perp$) or $\mathsf{rd}_R(x) = \mathsf{rd}_R(y)$.
4. $\Sigma^* \cap [\perp] = \mathsf{Ance}_R(\Sigma^* \cdot Z \cdot \Sigma^*)$, where $Z = \{z \mid z \to \perp \text{ is a rule of } R_\perp\}$.
5. $(\Sigma \cup \{\perp\})^*/{\sim}$ is isomorphic to $\Sigma^*/{\sim} = \{\Sigma^* \cap [u] \mid u \in (\Sigma \cup \{\perp\})^*\}$.

Notation: We write $\sim$ instead of $\leftrightarrow^*_{R_\perp}$, and $[x]$ for the $\sim$-class of x.

*Proof (**Lemma 7**).* We show part (1). Since $R_\perp$ is terminating, it suffices to establish local confluence of $R_\perp$. So, we suppose that $x \to_{R_\perp} y_1$ and $x \to_{R_\perp} y_2$.
- If both y_1 and y_2 contain an occurrence of $\perp$, then we have $y_1 \to^*_{R_\perp} \perp$ and $y_2 \to^*_{R_\perp} \perp$.
- If neither of y_1, y_2 contains an occurrence of $\perp$, then by confluence of R there exists $y' \in \Sigma^*$ with $y_1 \to^*_R y'$ and $y_2 \to^*_R y'$. So, $y_1 \to^*_{R_\perp} y'$ and $y_2 \to^*_{R_\perp} y'$.

- Suppose now that y_1 contains an occurrence of $\perp$, but y_2 does not. It follows that x does not contain any $\perp$ (if it did, then so would y_2). Moreover, x is of the form $x = x_1 z x_2$ for some z with $z \to \perp$ being a rule of $R_\perp$, and
$$x = x_1 z x_2 \longrightarrow_{R_\perp} x_1 \perp x_2 = y_1.$$
The string y_2 is an R-successor of x, and by the seamlessness property we have that y_2 is of the form $y_2 = x_1' z' x_2'$ for some $z' \to \perp$ in $R_\perp$. It follows that $y_1 = x_1 \perp x_2 \to^*_{R_\perp} \perp$ and $y_2 = x_1' z' x_2' \to_{R_\perp} x_1' \perp x_2' \to^*_{R_\perp} \perp$.

The case of y_2 containing $\perp$ and y_1 not containing any $\perp$ is symmetric.

Since both R and $R_\perp$ are confluent and terminating, the functions rd_R and $\mathsf{rd}_\perp$ are well-defined. We know that for all $x, y \in (\Sigma \cup \{\perp\})^*$,
$$x \sim y \iff \mathsf{rd}_\perp(x) = \mathsf{rd}_\perp(y).$$
The above equivalence is a consequence of termination and the Church-Rosser property of $R_\perp$.

Part (2): Let $x \in \Sigma^*$. Since $\perp$ is $R_\perp$-irreducible, it holds that $x \sim \perp$ iff $\mathsf{rd}_\perp(x) = \perp$. It remains to see that $\mathsf{rd}_\perp(x) = \perp$ iff $\mathsf{rd}_R(x)$ contains some z with $z \to \perp$. The right-to-left direction is obvious, because
$$x \to^*_R \mathsf{rd}_R(x) = x_1 z x_2 \to_{R_\perp} x_1 \perp x_2 \to^*_{R_\perp} \perp.$$
For the left-to-right direction, we observe that $x \to^*_{R_\perp} \mathsf{rd}_\perp(x) = \perp$ and also that $x \to^*_{R_\perp} \mathsf{rd}_R(x)$. By confluence of $R_\perp$, it must be that $\mathsf{rd}_R(x) \to^*_{R_\perp} \perp$. If $\mathsf{rd}_R(x)$ contains no z with $z \to \perp$, then $\mathsf{rd}_R(x)$ is $R_\perp$-irreducible and it is $\neq \perp$. So, it must contain such a substring z.

Part (3): The right-to-left direction of part (3) is trivial. Suppose now that $x \sim y$ and that x, y are not $\sim$-congruent to $\perp$. So, both $\mathsf{rd}_R(x)$ and $\mathsf{rd}_R(y)$ must contain no z with $z \to \perp$ (otherwise they would be $\sim$-congruent to $\perp$). It follows that $\mathsf{rd}_R(x)$ and $\mathsf{rd}_R(y)$ are $R_\perp$-irreducible. Therefore, $\mathsf{rd}_\perp(x) = \mathsf{rd}_R(x)$ and $\mathsf{rd}_\perp(y) = \mathsf{rd}_R(y)$. Now, $x \sim y$ implies that $\mathsf{rd}_\perp(x) = \mathsf{rd}_\perp(y)$, and hence $\mathsf{rd}_R(x) = \mathsf{rd}_R(y)$.

Part (4): Let $x \in \Sigma^*$ be a string in the right-hand side. There is a string $y_1 z y_2$ in $\Sigma^* \cdot Z \cdot \Sigma^*$ with $z \to \perp$ and $x \to^*_R y_1 z y_2$. But $y_1 z y_2 \to_{R_\perp} y_1 \perp y_2 \to^*_{R_\perp} \perp$. It follows that $x \to^*_{R_\perp} \perp$. So, $x \in \Sigma^* \cap [\perp]$. For the reverse containment, suppose that $x \in \Sigma^*$ and $x \sim \perp$. From part (2) we have that $\mathsf{rd}_R(x) \in \Sigma^* \cdot Z \cdot \Sigma^*$. But x is an R-ancestor of $\mathsf{rd}_R(x)$, and therefore x belongs to the right-hand size.

Part (5): We claim that the map $[u] \mapsto \Sigma^* \cap [u]$, where u is a string over $\Sigma \cup \{\perp\}$, is an isomorphism. The map is clearly surjective. It remains to see that it is injective. Suppose that $\Sigma^* \cap [u] = \Sigma^* \cap [v]$. If this is also equal to $\emptyset$, then both u and v contain an occurrence of $\perp$, and hence $[u] = [v] = [\perp]$. Assume now that $\Sigma^* \cap [u] = \Sigma^* \cap [v] \neq \emptyset$. It follows that there exists some string $w \in \Sigma^*$ with $w \sim u$ and $w \sim v$. So, $u \sim v$ and therefore $[u] = [v]$.

Lemma 8. Suppose that $(\Sigma, R, R_\perp)$ satisfies Assumption 1.
1. $\mathscr{C}_{R_\perp}(L) = [\perp]_\Sigma \cup \bigcup \{[v]_\Sigma \mid v \in \mathscr{G}_{R_\perp}(L)\}$, for a language $L \subseteq \Sigma^*$.
2. $\mathscr{G}_{R_\perp}(L_1) = \mathscr{G}_{R_\perp}(L_2)$ iff $\mathscr{C}_{R_\perp}(L_1) = \mathscr{C}_{R_\perp}(L_2)$, for languages $L_1, L_2 \subseteq \Sigma^*$.
3. $\mathscr{C}_{R_\perp}(L) = \mathsf{Ance}_R(\mathsf{Desc}_R(L)) \cup [\perp]_\Sigma$, for a language $L \subseteq \Sigma^*$.

The above are the analogue of Lemma 2. As an analogue of Lemma 3, we have:
(a) $\mathcal{G}_{R_\perp}(e) = \{\mathsf{rd}_R(u) \mid u \in \mathcal{R}(e)\} \setminus [\perp]_\Sigma = \mathscr{G}_{R_\perp}(\mathcal{R}(e))$, for an expression e.

(b) $\mathcal{C}_{R_\perp}(e) = [\perp]_\Sigma \cup \bigcup\{[v] \mid v \in \mathcal{G}_{R_\perp}(e)\}$, for an expression e.
(c) $\mathcal{G}_{R_\perp}(e_1) = \mathcal{G}_{R_\perp}(e_2)$ iff $\mathcal{C}_{R_\perp}(e_1) = \mathcal{C}_{R_\perp}(e_2)$, for expressions e_1, e_2.

*Proof (**Lemma 8**).* We will be dropping the $R_\perp$ subscripts freely for notational convenience.

We show part (1). Since $u \sim \mathsf{rd}(u)$, we have $[u] = [\mathsf{rd}(u)]$ and hence $[u]_\Sigma = [\mathsf{rd}(u)]_\Sigma$. Now,

$$\begin{aligned} \mathscr{C}(L) &= [\perp]_\Sigma \cup \textstyle\bigcup_{u\in L}[u]_\Sigma \\ &= [\perp]_\Sigma \cup \textstyle\bigcup_{u\in L, u\not\sim\perp}[u]_\Sigma \\ &= [\perp]_\Sigma \cup \textstyle\bigcup_{u\in L, u\not\sim\perp}[\mathsf{rd}(u)]_\Sigma \\ &= [\perp]_\Sigma \cup \textstyle\bigcup_{v\in\mathscr{G}(L)}[v]_\Sigma, \end{aligned}$$

which establishes part (1).

We show part (2). The left-to-right direction is easy. Suppose that $\mathscr{G}(L_1) = \mathscr{G}(L_2)$. Using part (1), we get that

$$\mathscr{C}(L_1) = [\perp]_\Sigma \cup \textstyle\bigcup_{v\in\mathscr{G}(L_1)}[v]_\Sigma = [\perp]_\Sigma \cup \textstyle\bigcup_{v\in\mathscr{G}(L_2)}[v]_\Sigma = \mathscr{C}(L_2).$$

For the right-to-left direction, it suffices by symmetry to show that $\mathscr{C}(L_1) \subseteq \mathscr{C}(L_2)$ implies $\mathscr{G}(L_1) \subseteq \mathscr{G}(L_2)$. Suppose that $\mathscr{C}(L_1) \subseteq \mathscr{C}(L_2)$, and let $\mathsf{rd}(u_1) \in \mathscr{G}(L_1)$, where $u_1 \in L_1$ and $u_1 \not\sim \perp$. We have that $[u_1]_\Sigma \subseteq \mathscr{C}(L_1) \subseteq \mathscr{C}(L_2)$ and therefore $\mathsf{rd}(u_1) \in \mathscr{C}(L_2)$. There exists $u_2 \in L_2$ such that $\mathsf{rd}(u_1) \in [u_2]_\Sigma$, and hence $\mathsf{rd}(u_1) \sim u_2$. Moreover, it holds that $u_2 \not\sim \perp$, because $u_2 \sim \perp$ would imply $\mathsf{rd}(u_1) \sim u_1 \sim \perp$, a contradiction. Using Lemma 7 we conclude that $\mathsf{rd}(u_1) = \mathsf{rd}(u_2)$ is in $\mathscr{G}(L_2)$.

Finally, we show part (3). Immediately from the definition of $\mathscr{C}(L)$ we know that $[\perp]_\Sigma \subseteq \mathscr{C}(L)$. Now, we see that $\mathsf{Ance}_R(\mathsf{Desc}_R(L))$ is contained in $\mathscr{C}(L)$. Suppose that u is in $\mathsf{Ance}_R(\mathsf{Desc}_R(L))$. Then, $u \in \Sigma^*$ and there is $v \in \mathsf{Desc}_R(L)$ with $u \to_R^* v$. There is also $u' \in L$ with $u' \to_R^* v$. It follows that $u \leftrightarrow_R^* v$ and $u' \leftrightarrow_R^* v$, therefore $u \leftrightarrow_R^* u'$. So, $u \sim u'$. We thus obtain $u \in [u']_\Sigma \subseteq \mathscr{C}(L)$.

For the reverse containment of part (3), consider an arbitrary element u of $\mathscr{C}(L)$. If $u \sim \perp$ then we are done. Assume now that $u \not\sim \perp$. There exists $u' \in L$ with $u \in [u']_\Sigma$, that is, $u \sim u'$, and $u' \not\sim \perp$. But then we have that $v = \mathsf{rd}(u) = \mathsf{rd}(u')$ (Lemma 7). So, v is a descendant of both u and u'. Now, notice that $v \in \mathsf{Desc}_R(L)$ and since u is an ancestor of v, we conclude that $u \in \mathsf{Ance}_R(\mathsf{Desc}_R(L))$.

We show part (a) by induction on the structure of e. For the bases cases we have:

$$\begin{aligned} &\mathcal{G}_\perp(a) = \{\mathsf{rd}(a)\} \setminus [\perp]_\Sigma = \{\mathsf{rd}(u) \mid u \in \mathcal{R}(a)\} \setminus [\perp]_\Sigma \\ &\mathcal{G}_\perp(0) = \emptyset = \{\mathsf{rd}(u) \mid u \in \mathcal{R}(0)\} \setminus [\perp]_\Sigma \\ &\mathcal{G}_\perp(1) = \{\mathsf{rd}(\epsilon)\} \setminus [\perp]_\Sigma = \{\mathsf{rd}(u) \mid u \in \mathcal{R}(1)\} \setminus [\perp]_\Sigma \end{aligned}$$

For the case of nondeterministic choice we have:

$$\begin{aligned}\mathcal{G}_\perp(e_1+e_2) &= \mathcal{G}_\perp(e_1)\cup\mathcal{G}_\perp(e_2)\\ &= \{\mathsf{rd}(u)\mid u\in\mathcal{R}(e_1)\}\setminus[\perp]_\Sigma\cup\\ &\qquad\qquad\{\mathsf{rd}(u)\mid u\in\mathcal{R}(e_2)\}\setminus[\perp]_\Sigma\\ &= \{\mathsf{rd}(u)\mid u\in\mathcal{R}(e_1)\cup\mathcal{R}(e_2)\}\setminus[\perp]_\Sigma\\ &= \{\mathsf{rd}(u)\mid u\in\mathcal{R}(e_1+e_2)\}\setminus[\perp]_\Sigma.\end{aligned}$$

For strings $x, y \in \Sigma^*$, it holds that $xy \sim \mathsf{rd}(x)\mathsf{rd}(y)$. It follows that $xy \sim \perp$ iff $\mathsf{rd}(x)\mathsf{rd}(y) \sim \perp$. Moreover, $xy \not\sim \perp$ implies that $x \not\sim \perp$ and $y \not\sim \perp$ (because $\sim$ is congruence). For the case of composition we have that $\mathcal{G}_\perp(e_1; e_2)$ is equal to:

$$\begin{aligned}&\mathcal{G}_\perp(e_1)\diamond\mathcal{G}_\perp(e_2) =\\ &\{u\diamond v\mid u\diamond v\text{ def., }u\in\mathcal{G}_\perp(e_1),\ v\in\mathcal{G}_\perp(e_2)\} =\\ &\{\mathsf{rd}(x)\diamond\mathsf{rd}(y)\mid\mathsf{rd}(x)\diamond\mathsf{rd}(y)\text{ defined,}\\ &\qquad\qquad\mathsf{rd}(x)\in\mathcal{G}_\perp(e_1),\ \mathsf{rd}(y)\in\mathcal{G}_\perp(e_2)\} =\\ &\{\mathsf{rd}(\mathsf{rd}(x)\mathsf{rd}(y))\mid\mathsf{rd}(x)\mathsf{rd}(y)\not\sim\perp,\\ &\qquad\qquad x\in\mathcal{R}(e_1),\ x\not\sim\perp,\ y\in\mathcal{R}(e_2),\ y\not\sim\perp\} =\\ &\{\mathsf{rd}(xy)\mid xy\not\sim\perp, x\in\mathcal{R}(e_1), y\in\mathcal{R}(e_2)\} =\\ &\{\mathsf{rd}(z)\mid z\in\mathcal{R}(e_1;e_2)\}\setminus[\perp]_\Sigma.\end{aligned}$$

We handle now the case e^*. First, we claim that

$$G_\perp(e)^{\langle n\rangle} = \{\mathsf{rd}(u)\mid u\in\mathcal{R}(e)^n\}\setminus[\perp]_\Sigma.$$

We show the claim by induction on n. For the base case, we have that $\mathcal{R}(e)^0 = \{\epsilon\}$ and therefore $G_\perp(e)^{\langle 0\rangle} = \{\epsilon\} = \{\mathsf{rd}(\epsilon)\} = \{\mathsf{rd}(u) \mid u \in R(e)^0\} \setminus [\perp]_\Sigma$. For the step, we have that $G(e)^{\langle n+1\rangle}$ is equal to:

$$\begin{aligned}&G_\perp(e)^{\langle n\rangle}\diamond G_\perp(e) =\\ &\{u\diamond v\mid u\diamond v\text{ def., }u\in G_\perp(e)^{\langle n\rangle},\ v\in G_\perp(e)\} =\\ &\{\mathsf{rd}(xy)\mid xy\not\sim\perp, x\in\mathcal{R}(e)^n, y\in\mathcal{R}(e)\} =\\ &\{\mathsf{rd}(z)\mid z\in\mathcal{R}(e)^{n+1}\}\setminus[\perp]_\Sigma,\end{aligned}$$

arguing similarly to how we did in the composition case. Finally, we have for the expression e^* that

$$\begin{aligned}G_\perp(e^*) &= \textstyle\bigcup_{n\geq 0} G_\perp(e)^{\langle n\rangle}\\ &= \textstyle\bigcup_{n\geq 0}\{\mathsf{rd}(u)\mid u\in\mathcal{R}(e)^n\}\setminus[\perp]_\Sigma\\ &= \{\mathsf{rd}(u)\mid u\in\textstyle\bigcup_{n\geq 0}\mathcal{R}(e)^n\}\setminus[\perp]_\Sigma\\ &= \{\mathsf{rd}(u)\mid u\in\mathcal{R}(e^*)\}\setminus[\perp]_\Sigma,\end{aligned}$$

thus completing the proof of part (2).

Parts (b) and (c) follow from parts (1), (2), (a), and from the definition of $\mathcal{C}_\perp$.

*Proof (**Lemma 4**).* The set Z is finite and hence regular. It follows that the set $\Sigma^* \cdot Z \cdot \Sigma^*$ is also regular, with $e = U; (\sum_z z); U$ being the corresponding regular expression. The string z ranges over the set $Z = \{z \mid z \to \perp\}$, and U is the universal expression. That is, $U = (\sum_a a)^*$, where a ranges over all the letters of the alphabet Σ.

Since R is well-behaved, the inverse system R^{-1} has rules of the form $a \to r$, where a is a letter of Σ. For every letter a we have by our hypotheses (R is well-behaved) that

$$\mathsf{Desc}_{R^{-1}}(a) = \mathsf{Ance}_R(a) = \mathcal{R}(e_a)$$

for some regular expression e_a. Moreover, $\mathsf{KA}_R \vdash e_a \equiv a$. Define the substitution θ by $a \mapsto e_a$. Lemma 1 gives us that

$$\mathsf{Ance}_R(\mathcal{R}(e)) = \mathsf{Desc}_{R^{-1}}(\mathcal{R}(e)) = \mathcal{R}(\theta(e)).$$

We put $e_\perp = \theta(e)$. From Lemma 7(4) we obtain that

$$\Sigma^* \cap [\perp] = \mathsf{Ance}_R(\Sigma^* \cdot Z \cdot \Sigma^*) = \mathsf{Ance}_R(\mathcal{R}(e)),$$

which is equal to $\mathcal{R}(\theta(e)) = \mathcal{R}(e_\perp)$. So, the set $\Sigma^* \cap [\perp] = \mathcal{R}(e_\perp)$ is regular. In order to show that $\mathsf{KA}_{R_\perp} \vdash e_\perp = \theta(e) \equiv 0$, it suffices to see that $\mathsf{KA}_{R_\perp} \vdash e \equiv 0$. Since $z \equiv 0$ for $z \to \perp$, we have $\sum_z z \equiv 0$ and therefore

$$e = U; (\textstyle\sum_z z); U \equiv 0.$$

We have thus shown that $e_\perp \equiv 0$ is provable in $\mathsf{KA}_{R_\perp}$.

*Proof (**Theorem 4**).* Consider the following transformations steps on an arbitrary regular expression e:

(1) *Descendants*: As described in Theorem 2 we get an expression e' with $\mathsf{KA}_R \vdash e \equiv e'$ and $\mathcal{R}(e') = \mathsf{Desc}_R(\mathcal{R}(e))$.
(2) *Ancestors*: Define the substitution θ by $a \mapsto e_a$, where $\mathcal{R}(e_a) = \mathsf{Ance}_R(a)$ (this can be done because R is well-behaved). Then, $\mathsf{KA}_R \vdash e' \equiv \theta(e')$ and $\mathcal{R}(\theta(e')) = \mathsf{Ance}_R(\mathcal{R}(e'))$.
(3) *Congruence class of* $\perp$: It was shown in Lemma 4 that there is an expression $e_\perp$ such that $\mathsf{KA}_{R_\perp} \vdash e_\perp \equiv 0$ and $\mathcal{R}(e_\perp) = [\perp]_\Sigma$. We put $\hat{e} = \theta(e') + e_\perp$.

We have $\mathsf{KA}_{R_\perp} \vdash e \equiv e' \equiv \theta(e') \equiv \theta(e') + e_\perp = \hat{e}$, and

$$\mathcal{R}(\hat{e}) = \mathcal{R}(\theta(e')) \cup \mathcal{R}(e_\perp) = \mathsf{Ance}_R(\mathsf{Desc}_R(\mathcal{R}(e))) \cup [\perp]_\Sigma,$$

which is equal to $\mathscr{C}_{R_\perp}(\mathcal{R}(e)) = \mathcal{C}_{R_\perp}(e)$ using Lemma 8(3). We have thus shown that $\mathcal{R}(\hat{e}) = \mathcal{C}_{R_\perp}(e)$.

We apply the above constructions to the expressions e_1 and e_2 to obtain the expressions $\hat{e}_1$ and $\hat{e}_2$ with:

$$\begin{aligned} &\mathsf{KA}_{R_\perp} \vdash e_1 \equiv \hat{e}_1 &\qquad& \mathcal{C}_{R_\perp}(e_1) = \mathcal{R}(\hat{e}_1) \\ &\mathsf{KA}_{R_\perp} \vdash e_2 \equiv \hat{e}_2 && \mathcal{C}_{R_\perp}(e_2) = \mathcal{R}(\hat{e}_2) \end{aligned}$$

From the hypothesis $\mathcal{G}_{R_\perp}(e_1) = \mathcal{G}_{R_\perp}(e_2)$ and Lemma 8 we have that $\mathcal{C}_{R_\perp}(e_1) = \mathcal{C}_{R_\perp}(e_2)$. So, $\mathcal{R}(\hat{e}_1) = \mathcal{R}(\hat{e}_2)$ and by completeness of KA for the interpretation $\mathcal{R}(\cdot)$ we get that $\mathsf{KA} \vdash \hat{e}_1 \equiv \hat{e}_2$. Since we have proved in $\mathsf{KA}_{R_\perp}$ the equations

$$e_1 \equiv \hat{e}_1 \qquad\qquad \hat{e}_1 \equiv \hat{e}_2 \qquad\qquad \hat{e}_2 \equiv e_2$$

we conclude by transitivity that $\mathsf{KA}_{R_\perp} \vdash e_1 \equiv e_2$.

Lemma 9. Suppose that $(P, Id, R, R_\perp)$ satisfies Assumption 2.
1. $R_\perp$ is terminating on $(\Sigma \cup \{\perp\})^*$ and confluent on S.
2. For $x \in S$: $x \sim \perp$ iff $\mathsf{rd}_R(x)$ contains some z, where $z \to \perp$ is a rule of $R_\perp$.
3. For $x, y \in S$, we have that $x \sim y$ iff ($x \sim \perp$ and $y \sim \perp$) or $\mathsf{rd}_R(x) = \mathsf{rd}_R(y)$.

4. $S \cap [\bot] = \mathsf{Ance}_R(\mathsf{Desc}_R(S \cdot Z \cdot S))$, where $Z = \{z \mid z \to \bot \text{ is a rule of } R_\bot\}$.

*Proof (**Lemma 9**).* The proof of the first three parts of the lemma are a straightforward adaptation of the proof given for Lemma 7.

We show now part (4). Suppose for $x \in S$ that $x \sim \bot$. Then, $\mathsf{rd}(x)$ has some substring z for which $z \to \bot$ is a rule of $R_\bot$. Since $x \to_R^* \mathsf{rd}(x)$, it suffices to show that $\mathsf{rd}(x)$ belongs to $\mathsf{Desc}_R(S \cdot Z \cdot S)$. We know that $z \in S$ is a substring of $\mathsf{rd}(x)$, and hence by the decomposition property $\mathsf{rd}(x)$ has an R-ancestor $x_1 z x_2$ such that $x_1, x_2 \in S$. Since $x_1 z x_2 \in S \cdot Z \cdot S$ and $x_1 z x_2 \to_R^* \mathsf{rd}(x)$, we obtain that $\mathsf{rd}(x) \in \mathsf{Desc}_R(S \cdot Z \cdot S)$.

For the reverse containment, let u be a string in the right-hand size of the equation. There exists v such that $u \to_R^* v$ and $v \in \mathsf{Desc}_R(S \cdot Z \cdot S)$. So, there is w such that $w \to_R^* v$ and $w \in S \cdot Z \cdot S$. From $u \to_R^* v$ and $w \to_R^* v$ we get that $u \leftrightarrow_R^* w$ and $u \sim w$. Since w is of the form $w = w_1 z w_2$ with $z \to \bot$, it holds that $w \sim \bot$. Since $Z \subseteq S$ and S is closed under concatenation, we have that $S \cdot Z \cdot S \subseteq S$. It follows that $w \in S$. Since S is additionally closed under $\to_R$ and $\to_R^{-1}$, we get that both v and u are in S. So, $u \sim \bot$ and $u \in S$.

Lemma 10. Let $(P, \mathit{Id}, R, R_\bot)$ be a tuple satisfying Assumption 2.

1. $\mathscr{C}_{R_\bot}(L) = [\bot]_S \cup \bigcup\{[v]_S \mid v \in \mathscr{G}_{R_\bot}(L)\}$, for a language $L \subseteq S$.
2. $\mathscr{G}_{R_\bot}(L_1) = \mathscr{G}_{R_\bot}(L_2)$ iff $\mathscr{C}_{R_\bot}(L_1) = \mathscr{C}_{R_\bot}(L_2)$, for languages $L_1, L_2 \subseteq S$.
3. $\mathscr{C}_{R_\bot}(L) = \mathsf{Ance}_R(\mathsf{Desc}_R(L)) \cup [\bot]_S$, for a language $L \subseteq S$.

The above are the analogue of Lemma 2. Moreover, we have:

(a) $\mathcal{G}_{R_\bot}(e) = \bigcup_{x \in \mathcal{R}(e)} \mathcal{G}_{R_\bot}(x)$, for an expression e.
(b) $\mathcal{G}_{R_\bot}(e) = \{\mathsf{rd}(u) \mid u \in \mathcal{R}(e)\} \setminus [\bot]_\Sigma = \mathscr{G}_{R_\bot}(\mathcal{R}(e))$, for an expression e.
(c) $\mathcal{C}_{R_\bot}(e) = \bigcup\{[v] \mid v \in \mathcal{G}_{R_\bot}(e)\}$, for an expression e.
(d) $\mathcal{G}_{R_\bot}(e_1) = \mathcal{G}_{R_\bot}(e_2)$ iff $\mathcal{C}_{R_\bot}(e_1) = \mathcal{C}_{R_\bot}(e_2)$, for expressions e_1, e_2.

For the parts (2), (3), and (4) we have the implicit assumption that $\mathcal{R}(e)$, $\mathcal{R}(e_1)$, $\mathcal{R}(e_2)$ are included in S. For a string $a_1 a_2 \cdots a_n$, we write $\mathcal{G}_{R_\bot}(a_1 a_2 \ldots a_n)$ to mean $\mathcal{G}_{R_\bot}(a_1; a_2; \cdots; a_n)$. Moreover, $\mathcal{G}_{R_\bot}(\epsilon)$ is notation for $\mathcal{G}_{R_\bot}(1)$.

*Proof (**Lemma 10**).* Parts (1) – (3) are essentially the same as in Lemma 8.

Suppose that e_1, e_2 are regular expressions satisfying $G(e_i) = \bigcup_{x \in \mathcal{R}(e_i)} G(x)$ for $i = 1, 2$. It follows that

$$\begin{aligned} G(e_1) \diamond G(e_2) &= \Big(\textstyle\bigcup_{x \in \mathcal{R}(e_1)} G(x)\Big) \diamond \Big(\textstyle\bigcup_{y \in \mathcal{R}(e_2)} G(y)\Big) \\ &= \textstyle\bigcup_{x \in \mathcal{R}(e_1)} \bigcup_{y \in \mathcal{R}(e_2)} G(x) \diamond G(y) \\ &= \textstyle\bigcup_{z \in \mathcal{R}(e_1) \cdot \mathcal{R}(e_2)} G(z). \\ &= \textstyle\bigcup_{z \in \mathcal{R}(e_1; e_2)} G(z). \end{aligned}$$

We show part (a) by induction on the structure of e. The base cases are straightforward. For the case of nondeterministic choice, we have:

$$\begin{aligned} G(e_1 + e_2) &= G(e_1) \cup G(e_2) \\ &= \Big(\bigcup_{x\in\mathcal{R}(e_1)} G(x)\Big) \cup \Big(\bigcup_{x\in\mathcal{R}(e_2)} G(x)\Big) \\ &= \bigcup_{x\in\mathcal{R}(e_1)\cup\mathcal{R}(e_2)} G(x) \\ &= \bigcup_{x\in\mathcal{R}(e_1+e_2)} G(x). \end{aligned}$$

The case of composition is handled immediately by the claim we showed above. For the case e^*, we first claim that

$$G(e)^{\langle n\rangle} = \bigcup_{x\in\mathcal{R}(e)^n} G(x).$$

We argue by induction on n. For the base case we have that $G(e)^{\langle 0\rangle} = \mathsf{id}$ and $\bigcup_{x\in R(e)^0} G(x) = G(\epsilon) = G(1) = \mathsf{id}$. For the step we use the claim shown above:

$$\begin{aligned} G(e)^{\langle n+1\rangle} &= G(e)^{\langle n\rangle} \diamond G(e) \\ &= \Big(\bigcup_{x\in\mathcal{R}(e)^n} G(x)\Big) \diamond \Big(\bigcup_{y\in\mathcal{R}(e)} G(y)\Big) \\ &= \bigcup_{z\in\mathcal{R}(e)^n\cdot\mathcal{R}(e)} G(z) \\ &= \bigcup_{z\in\mathcal{R}(e)^{n+1}} G(z). \end{aligned}$$

It follows that

$$\begin{aligned} G(e) &= \bigcup_{n\geq 0} G(e)^{\langle n\rangle} \\ &= \bigcup_{n\geq 0}\bigcup_{x\in\mathcal{R}(e)^n} G(x) \\ &= \bigcup_{x\in\bigcup_{n\geq 0}\mathcal{R}(e)^n} G(x), \end{aligned}$$

which is equal to $\bigcup_{x\in\mathcal{R}(e^*)} G(x)$.

For part (b), suppose that $\mathcal{R}(e) \subseteq S$. For every $x \in S$, it holds that $\mathcal{G}_\perp(x) = \{\mathsf{rd}(x)\}$ when $x \not\sim \perp$, and $\mathcal{G}_\perp(x) = \emptyset$ when $x \sim \perp$. Then,

$$\begin{aligned} \mathcal{G}_\perp(e) &= \bigcup_{x\in\mathcal{R}(e)} \mathcal{G}_\perp(x) \\ &= \{\mathsf{rd}(x) \mid x \in \mathcal{R}(e), x \not\sim \perp\}, \end{aligned}$$

which is equal to $\mathcal{G}_{R_\perp}(\mathcal{R}(e))$. Parts (c) and (d) follow easily from parts (1) – (3).

*Proof (**Lemma 5**).* For an atom α, we have $\mathcal{G}_{R_\perp}(\alpha) = \mathsf{rd}_R(\{\alpha\}) \setminus [\perp]_S = \{\alpha\}$, because the left-hand size of every rule has length ≥ 2. It follows that

$$\mathcal{G}_{R_\perp}(\textstyle\sum_\alpha \alpha) = \bigcup_\alpha \mathcal{G}_{R_\perp}(\alpha) = Id = \mathcal{G}_{R_\perp}(1).$$

For an action letter p, we observe that

$$\begin{aligned} G_{R_\perp}(p) &= \mathsf{rd}_R(\{\alpha p\beta \mid \alpha,\beta\}) \setminus [\perp]_S \\ &= \{\mathsf{rd}_R(\alpha p\beta) \mid \alpha,\beta\} \setminus [\perp]_S \\ &= \{\mathsf{rd}_R(\alpha p\beta) \mid \alpha,\beta \in At,\ \alpha p\beta \not\sim \perp\} \end{aligned}$$

Using confluence it can be shown that $\{\alpha\} \diamond G_{R_\perp}(p) \diamond \{\beta\} = \{\mathsf{rd}_R(\alpha p \beta)\} \setminus [\perp]_S$. It follows that

$$\begin{aligned}\mathcal{G}_{R_\perp}(\textstyle\sum_{\alpha,\beta} \alpha; p; \beta) &= \textstyle\bigcup_{\alpha,\beta} \mathcal{G}_{R_\perp}(\alpha; p; \beta)\\ &= \textstyle\bigcup_{\alpha,\beta} \mathcal{G}_{R_\perp}(\alpha) \diamond \mathcal{G}_{R_\perp}(p) \diamond \mathcal{G}_{R_\perp}(\beta)\\ &= \textstyle\bigcup_{\alpha,\beta} \{\alpha\} \diamond \mathcal{G}_{R_\perp}(p) \diamond \{\beta\}\\ &= \textstyle\bigcup_{\alpha,\beta} \{\mathsf{rd}_R(\alpha p \beta)\} \setminus [\perp]_S\\ &= \mathcal{G}_{R_\perp}(p).\end{aligned}$$

With a straightforward induction argument we thus have $\mathcal{G}_{R_\perp}(e) = \mathcal{G}_{R_\perp}(\theta(e))$, for every expressions e. The rest of the claim follow easily.

*Proof (**Lemma 6**).* We have assumed S to be regular with $S = \mathcal{R}(e_S)$. There are finitely many rules of the form $z \to \perp$, and we put

$$e_Z = \textstyle\sum\{z \mid z \to \perp \text{ is rule of } R_\perp\}.$$

So, $e = e_S; e_Z; e_S$ is the regular expression for the regular set $S \cdot Z \cdot S$, and clearly $\mathsf{KA}_{R_\perp} \vdash e \equiv 0$.

- *Descendants*: As described in Theorem 2 we get an expression e' with $\mathsf{KA}_R \vdash e \equiv e'$ and $\mathcal{R}(e') = \mathsf{Desc}_R(\mathcal{R}(e))$.
- *Ancestors*: Define the substitution θ by $a \mapsto e_a$, where $\mathcal{R}(e_a) = \mathsf{Ance}_R(a)$ (R is well-behaved). Then, $\mathsf{KA}_R \vdash e' \equiv \theta(e')$ and $\mathcal{R}(\theta(e')) = \mathsf{Ance}_R(\mathcal{R}(e'))$.

We put $\hat{e} = \theta(e')$. We have $\mathcal{R}(\hat{e}) = \mathsf{Ance}_R(\mathsf{Desc}_R(S \cdot Z \cdot S))$ and $\mathsf{KA}_{R_\perp} \vdash \hat{e} \equiv e' \equiv e \equiv 0$.

*Proof (**Theorem 5**).* Using Lemma 5, we see that there are expressions $\tilde{e}_1$ and $\tilde{e}_2$ such that $\mathsf{KA}_E \vdash e_1 \equiv \tilde{e}_1$, $\mathsf{KA}_E \vdash e_2 \equiv \tilde{e}_2$ and $\mathcal{R}(\tilde{e}_1), \mathcal{R}(\tilde{e}_2) \subseteq S$. Moreover, $\mathcal{G}_{R_\perp}(e_i) = \mathcal{G}_{R_\perp}(\tilde{e}_i)$ for $i = 1, 2$, and from our hypothesis $\mathcal{G}_{R_\perp}(\tilde{e}_1) = \mathcal{G}_{R_\perp}(\tilde{e}_2)$. It suffices to show that $\mathsf{KA}_E \vdash \tilde{e}_1 \equiv \tilde{e}_2$.

Consider the following transformations steps on an arbitrary regular expression e satisfying $\mathcal{R}(e) \subseteq S$:

(1) *Descendants & ancestors*: As we have done in previous proofs, we can construct an expression e' with $\mathsf{KA}_R \vdash e \equiv e'$ and $\mathcal{R}(e') = \mathsf{Ance}_R(\mathsf{Desc}_R(\mathcal{R}(e)))$.
(2) *Congruence class of* $\perp$: It was shown in Lemma 6 that there is an expression $e_\perp$ such that $\mathsf{KA}_{R_\perp} \vdash e_\perp \equiv 0$ and $\mathcal{R}(e_\perp) = [\perp]_S$. We put $\hat{e} = e' + e_\perp$.

We have $\mathsf{KA}_{R_\perp} \vdash e \equiv e' \equiv e' + e_\perp = \hat{e}$, and

$$\mathcal{R}(\hat{e}) = \mathcal{R}(e') \cup \mathcal{R}(e_\perp) = \mathsf{Ance}_R(\mathsf{Desc}_R(\mathcal{R}(e))) \cup [\perp]_S,$$

which is equal to $\mathscr{C}_{R_\perp}(\mathcal{R}(e)) = \mathcal{C}_{R_\perp}(e)$ using Lemma 10. We have thus shown that $\mathcal{R}(\hat{e}) = \mathcal{C}_{R_\perp}(e)$.

We apply the above constructions to the expressions $\tilde{e}_1$ and $\tilde{e}_2$ to obtain the expressions $\hat{e}_1$ and $\hat{e}_2$ with:

$$\begin{aligned}&\mathsf{KA}_E \vdash \tilde{e}_1 \equiv \hat{e}_1 &\qquad& \mathcal{C}_{R_\perp}(\tilde{e}_1) = \mathcal{R}(\hat{e}_1)\\ &\mathsf{KA}_E \vdash \tilde{e}_2 \equiv \hat{e}_2 && \mathcal{C}_{R_\perp}(\tilde{e}_2) = \mathcal{R}(\hat{e}_2)\end{aligned}$$

From $\mathcal{G}_{R_\perp}(\tilde{e}_1) = \mathcal{G}_{R_\perp}(\tilde{e}_2)$ and Lemma 10 we have that $\mathcal{C}_{R_\perp}(\tilde{e}_1) = \mathcal{C}_{R_\perp}(\tilde{e}_2)$. So, $\mathcal{R}(\hat{e}_1) = \mathcal{R}(\hat{e}_2)$ and by completeness of KA for the interpretation $\mathcal{R}$ we get that $\mathsf{KA} \vdash \hat{e}_1 \equiv \hat{e}_2$. It follows that $\mathsf{KA}_E \vdash \tilde{e}_1 \equiv \tilde{e}_2$, which completes the proof.

*Proof (**Theorem 6**).* We denote by P the set of the *action* letters. We write At for the set of subidentities, which we are called *atoms* in the case of KAT. We write Σ for the union of P and At. The rewrite systems R and $R_\perp$ contain only the rules stipulated in Assumption 2.

$$\alpha\alpha \to \alpha\,(\alpha \in At) \qquad \alpha\beta \to \perp (\alpha \neq \beta)$$
$$a\perp \to \perp, \perp a \to \perp (a \in \Sigma) \qquad \perp\perp \to \perp$$

Recall that $S \subseteq \Sigma^*$ is the set of non-empty strings over Σ, in which every action symbol p appears surrounded by atoms, as in $\alpha p \beta$. The set S is closed under $\to_R$, because R only collapses consecutive identical atoms. It is also closed under $\to_R^{-1}$, because the inverse system of R creates a consecutive copy of an atom. For example $\alpha p \beta \to_R^{-1} \alpha\alpha p \beta$. Every rule of R strictly reduces the length of the string. To prove that R is confluent on S, it suffices to establish local confluence, since R is terminating. The proof is similar to the one we gave in Example 1.

Seamlessness property: Suppose that $x\alpha\beta y$ is in S, and $\alpha\beta \to \perp$. If an R-rule is applied to x or y, then we are done. Suppose that $x = x'\alpha$ and we have the rewrite step $x'\alpha\alpha\beta y \to_R x'\alpha\beta y$. The only remaining case is when $y = \beta y'$ and we have the rewrite step $x\alpha\beta\beta y' \to_R x\alpha\beta y'$. So, the property holds.

The irreducibles ($R_\perp$-irreducible strings of S) are the *guarded strings* over P and At, that is, strings of the form $\alpha_0 p_1 \alpha_1 p_2 \cdots \alpha_{n-1} p_n \alpha_n$, where $n \geq 0$. The fusion product $\diamond$ of irreducibles is given by

$$x\alpha \diamond \beta y = \begin{cases} x\alpha y, & \text{if } \alpha = \beta; \\ \text{undefined}, & \text{if } \alpha \neq \beta. \end{cases}$$

The set $[\perp]_S$ is equal to $\{x \in S \mid x \text{ has some } \alpha\beta \text{ as substring with } \alpha \neq \beta\}$. The interpretation $\mathcal{G}_{R_\perp}$ is the *standard interpretation* $\mathcal{G}_{\mathsf{KAT}}$ of KAT terms as sets of guarded strings.

Now, we verify that the tuple $(P, At, R, R_\perp)$ is well-behaved. First, notice that $R_\perp$ has finitely many rules, because the set At of atoms is finite. The right-hand side of every rule $\alpha\alpha \to \alpha$ of R is a single letter. For every atom α, the R-ancestors of α are $\mathsf{Ance}_R(\alpha) = \{\alpha^i \mid i \geq 1\}$. We put $e_\alpha = \alpha^+$, and we claim that $\mathsf{KA}_R \vdash e_\alpha \equiv \alpha$. The proof is as in Example 1. For every atomic action p, the R-ancestors of p are $\mathsf{Ance}_R(p) = \{p\}$. The associated equations E are:

$$\textstyle\sum_{\alpha \in At} \alpha \equiv 1 \qquad \alpha;\alpha \equiv \alpha\ (\alpha \in At) \qquad \alpha;\beta \equiv 0\ (\alpha \neq \beta)$$

We write KAT instead of KA_E. Suppose that $\mathcal{G}_{\mathsf{KAT}}(e_1) = \mathcal{G}_{\mathsf{KAT}}(e_2)$, which is the same as $\mathcal{G}_{R_\perp}(e_1) = \mathcal{G}_{R_\perp}(e_2)$. From Theorem 5 we obtain that $\mathsf{KAT} \vdash e_1 \equiv e_2$.

*Proof (**Theorem 7**).* The language involves an alphabet P of action letters, and an alphabet At of atoms. We write Σ for the union of P and At. We define the rewrite systems R and $R_\perp$ to contain the following rules: $\alpha\alpha \to \alpha$ for every $\alpha \in At$, and

$$\alpha\beta \to \perp (\alpha \neq \beta) \qquad \gamma p \delta \to \perp (\gamma p \delta \in Z_h)$$
$$a\perp \to \perp, \perp a \to \perp (a \in \Sigma) \qquad \perp\perp \to \perp$$

Recall that $S \subseteq \Sigma^*$ is the set of non-empty strings over Σ, in which every action symbol p appears surrounded by atoms, as in $\alpha p \beta$. The set S is closed under $\to_R$ and $\to_R^{-1}$. Every rule of R strictly reduces the length of the string. To prove that R is confluent on S, it suffices to establish local confluence, since R is terminating. The seamlessness property holds, which is proved as in the case of KAT.

The irreducibles are guarded strings, that contain no occurrence of a substring $\gamma p \delta \in Z_h$. The fusion product $\diamond$ is given as in the case of KAT. The set $[\bot]_S$ is equal to $\{x \in S \mid x \text{ has a substring } \alpha\beta \text{ with } \alpha \neq \beta, \text{ or } \gamma p \delta \in Z_h\}$. A straightforward induction on the structure of e establishes that $\mathcal{G}_{R_\bot}(e) = \mathcal{G}_h(e)$.

We verify that the tuple $(P, At, R, R_\bot)$ is well-behaved. The system $R_\bot$ has finitely many rules, because At and Z_h are finite. The rest of the requirements are shown to hold as in the case of KAT. Now, E is the collection of equations:

$$\begin{array}{rr} \sum_{\alpha \in At} \alpha \equiv 1 & \alpha;\alpha \equiv \alpha \ (\alpha \in At) \\ \alpha;\beta \equiv 0 \ (\alpha \neq \beta) & \gamma;p;\delta \equiv 0 \ (\gamma;p;\delta \in Z_h) \end{array}$$

We write $\mathsf{KAT} + H$ instead of KA_E. Suppose that $\mathcal{G}_h(e_1) = \mathcal{G}_h(e_2)$, that is, $\mathcal{G}_{R_\bot}(e_1) = \mathcal{G}_{R_\bot}(e_2)$. It follows from Theorem 5 that $\mathsf{KAT} + H \vdash e_1 \equiv e_2$.

*Proof (**Theorem 8**).* The language involves an alphabet P of action letters, and an alphabet At of atoms. We write Σ for the union of P and At. We define the rewrite systems R and $R_\bot$ to contain the following rules:

$$\begin{array}{rr} \alpha\alpha \to \alpha \, (\alpha \in At) & \alpha\beta \to \bot \, (\alpha \neq \beta) \\ \gamma p \gamma \to \gamma \, (\gamma p \in X) & \gamma p \delta \to \bot \, (\gamma p \in X, \gamma \neq \delta) \\ & a\bot \to \bot, \bot a \to \bot \, (a \in \Sigma) \\ & \bot\bot \to \bot \end{array}$$

Recall that $S \subseteq \Sigma^*$ is the set of non-empty strings over Σ, in which every action symbol p appears surrounded by atoms, as in $\alpha p \beta$. The set S is closed under $\to_R$ and $\to_R^{-1}$. Every rule of R strictly reduces the length of the string. To prove that R is confluent on S, it suffices to establish local confluence, since R is terminating. The only interesting cases are the following for γp, γq in X:

$$\begin{tikzcd}[column sep=tiny] & x\gamma\gamma p\gamma y \arrow[dl] \arrow[dr] & \\ x\gamma p\gamma y \arrow[dr] & & x\gamma\gamma y \arrow[dl] \\ & x\gamma y & \end{tikzcd} \qquad \begin{tikzcd}[column sep=tiny] & x\gamma p\gamma q\gamma y \arrow[dl] \arrow[dr] & \\ x\gamma q\gamma y \arrow[dr] & & x\gamma p\gamma y \arrow[dl] \\ & x\gamma y & \end{tikzcd}$$

To see that the seamlessness property holds, observe the following cases:

$$\begin{array}{rr} x\alpha\underline{\alpha\beta}y \to x\underline{\alpha\beta}y & \text{if } \alpha p \in X : x\alpha p\underline{\alpha\beta}y \to x\underline{\alpha\beta}y \\ x\underline{\alpha\beta}\beta y \to x\underline{\alpha\beta}y & \text{if } \beta p \in X : x\underline{\alpha\beta}p\beta y \to x\underline{\alpha\beta}y \\ \text{if } \gamma p \in X,\, \gamma \neq \delta : x\gamma\underline{\gamma p\delta}y \to x\underline{\gamma p\delta}y & \text{if } \gamma p, \gamma q \in X,\, \gamma \neq \delta : x\gamma q\underline{\gamma p\delta}y \to x\underline{\gamma p\delta}y \\ \text{if } \gamma p \in X,\, \gamma \neq \delta : x\underline{\gamma p\delta}\delta y \to x\underline{\gamma p\delta}y & \text{if } \gamma p, \delta q \in X,\, \gamma \neq \delta : x\underline{\gamma p\delta}q\delta y \to x\underline{\gamma p\delta}y \end{array}$$

The irreducibles are guarded strings, that contain no occurrence of a substring $\gamma p \in X$. The fusion product $\diamond$ is given as in the case of KAT. The set $[\bot]_S$ is equal to $\{x \in S \mid x \text{ has a substring } \alpha\beta \text{ with } \alpha \neq \beta, \text{ or } \gamma p \delta \text{ with } \gamma p \in X, \gamma \neq \delta\}$. By induction on the structure of e, we can show that $\mathcal{G}_{R_\bot}(e) = \mathcal{G}_h(e)$.

We verify that the tuple $(P, At, R, R_\bot)$ is well-behaved. The system $R_\bot$ has finitely many rules, because At and X are finite. Let α be an arbitrary atom, and define $B_\alpha = \{p \mid \alpha p \in X\} = \{p_1, \dots, p_k\}$. The R-ancestors of α form a regular set equal to $\mathcal{R}(e_\alpha)$, where $e_\alpha = (\alpha^+; (p_1 + \cdots + p_k))^*; \alpha^+$. We have already seen in Example 1 how to establish that $\mathsf{KA}_R \vdash \alpha^+ \equiv \alpha$. It follows that:

$$\mathsf{KA}_R \vdash e_\alpha \equiv (\alpha; (p_1 + \cdots + p_k))^*; \alpha \equiv (\alpha; p_1 + \cdots + \alpha; p_k)^*; \alpha.$$

Reasoning in KA_R, we see that $\alpha \leq (\alpha; p_1 + \cdots + \alpha; p_k)^*; \alpha$, and also that

$$(\alpha; p_1 + \cdots + \alpha; p_k)^*; \alpha \leq \alpha \Longleftarrow (\alpha; p_1 + \cdots + \alpha; p_k); \alpha \leq \alpha,$$

which holds because every $\alpha; p_i; \alpha \equiv \alpha$ is an axiom in R. Now, E is the collection of equations:

$$\begin{array}{ll} \sum_{\alpha \in At} \alpha \equiv 1 & \gamma; p; \gamma \equiv \gamma \ (\gamma p \in X) \\ \alpha; \alpha \equiv \alpha \ (\alpha \in At) & \gamma; p; \delta \equiv 0 \ (\gamma p \in X, \gamma \neq \delta) \\ \alpha; \beta \equiv 0 \ (\alpha \neq \beta) & \end{array}$$

We observe that $\mathsf{KAT} + H$ can prove the equations E. Indeed for $\gamma p \in X$, we have that $\gamma; p; \gamma \equiv \gamma; \gamma \equiv \gamma$ and $\gamma; p; \delta \equiv \gamma; \delta \equiv 0$ $(\gamma \neq \delta)$. Suppose that $\mathcal{G}_h(e_1) = \mathcal{G}_h(e_2)$, that is, $\mathcal{G}_{R_\perp}(e_1) = \mathcal{G}_{R_\perp}(e_2)$. It follows from Theorem 5 that $\mathsf{KA}_E \vdash e_1 \equiv e_2$. So, $\mathsf{KAT} + H \vdash e_1 \equiv e_2$.

Lemma 11. Consider the language P, dup, At of NetKAT. Given the axioms of Kleene algebra, the group of axioms

$$\begin{array}{lll} \sum_{\alpha \in At} \alpha \equiv 1 & \alpha; \mathsf{dup} \equiv \mathsf{dup}; \alpha & p_\alpha \equiv p_\alpha; \alpha \\ \alpha; \beta \equiv 0 \ (\alpha \neq \beta) & p_\alpha; p_\beta \equiv p_\beta & \alpha \equiv \alpha; p_\alpha \end{array}$$

is equivalent to the group of axioms

$$\begin{array}{llll} \sum_{\alpha \in At} \alpha \equiv 1 & \alpha; \alpha \equiv \alpha & \alpha; p_\alpha; \alpha \equiv \alpha & \alpha; \mathsf{dup}; \beta \equiv 0 \ (\alpha \neq \beta) \\ & \alpha; \beta \equiv 0 \ (\alpha \neq \beta) & p_\alpha; \alpha; p_\beta \equiv p_\beta & \alpha; p_\beta; \gamma \equiv 0 \ (\beta \neq \gamma) \end{array}$$

Proof. We first show that KA and the axioms of the first group can prove the axioms of the second group:

$$\begin{aligned} \alpha; \alpha &\equiv \alpha; p_\alpha; \alpha \equiv \alpha; p_\alpha \equiv \alpha \\ \alpha; p_\alpha; \alpha &\equiv \alpha; p_\alpha \equiv \alpha \\ p_\alpha; \alpha; p_\beta &\equiv p_\alpha; p_\beta \equiv p_\beta \\ \alpha; \mathsf{dup}; \beta &\equiv \mathsf{dup}; \alpha; \beta \equiv \mathsf{dup}; 0 \equiv 0 \ (\alpha \neq \beta) \\ \alpha; p_\beta; \gamma &\equiv \alpha; p_\beta; \beta; \gamma \equiv \alpha; p_\beta; 0 \equiv 0 \ (\beta \neq \gamma) \end{aligned}$$

Now, we show that KA and the axioms of the second group can prove the axioms of the first group:

$$\begin{aligned} \alpha; \mathsf{dup} &\equiv \textstyle\sum_\beta \alpha; \mathsf{dup}; \beta \equiv \alpha; \mathsf{dup}; \alpha + \sum_{\beta \neq \alpha} \alpha; \mathsf{dup}; \beta \equiv \alpha; \mathsf{dup}; \alpha \\ \mathsf{dup}; \alpha &\equiv \textstyle\sum_\beta \beta; \mathsf{dup}; \alpha \equiv \alpha; \mathsf{dup}; \alpha + \sum_{\beta \neq \alpha} \beta; \mathsf{dup}; \alpha \equiv \alpha; \mathsf{dup}; \alpha \end{aligned}$$

and therefore $\alpha; \mathsf{dup} \equiv \mathsf{dup}; \alpha$. Now:

$$\begin{aligned} p_\alpha; p_\beta &\equiv \textstyle\sum_\gamma \left(\sum_\delta \gamma; p_\alpha; \delta; p_\beta\right) \equiv \sum_\gamma \left(\gamma; p_\alpha; \alpha; p_\beta + \sum_{\delta \neq \alpha} \gamma; p_\alpha; \delta; p_\beta\right) \\ &\equiv \textstyle\sum_\gamma \gamma; p_\alpha; \alpha; p_\beta \equiv \sum_\gamma \gamma; p_\beta \equiv p_\beta \\ p_\alpha &\equiv \textstyle\sum_\beta \left(\sum_\gamma \beta; p_\alpha; \gamma\right) \equiv \sum_\beta \left(\beta; p_\alpha; \alpha + \sum_{\gamma \neq \alpha} \beta; p_\alpha; \gamma\right) \\ &\equiv \textstyle\sum_\beta \beta; p_\alpha; \alpha \equiv p_\alpha; \alpha \end{aligned}$$

$$\alpha; p_\alpha \equiv \textstyle\sum_\beta \alpha; p_\alpha; \beta \equiv \alpha; p_\alpha; \alpha + \sum_{\beta \neq \alpha} \alpha; p_\alpha; \beta \equiv \alpha; p_\alpha; \alpha \equiv \alpha$$

*Proof (**Theorem 9**).* Let Σ be the union of P, $\{\mathsf{dup}\}$, and At. The set $S \subseteq \Sigma^*$ contains the non-empty strings over Σ, in which every action symbol (p_α or dup) appears surrounded by atoms. The set S is closed both under $\to_R$ and $\to_R^{-1}$. Every rule of R is length reducing. Since R is terminating, we only need to show local confluence in order to establish confluence. Consider, for example, the cases:

$$\begin{tikzcd}
& x\alpha\alpha p_\alpha \alpha y \arrow[dl] \arrow[dr] & \\
x\alpha p_\alpha \alpha y \arrow[dr] & & x\alpha\alpha y \arrow[dl] \\
& x\alpha y &
\end{tikzcd}
\quad
\begin{tikzcd}
& x\alpha p_\alpha \alpha\alpha y \arrow[dl] \arrow[dr] & \\
x\alpha\alpha y \arrow[dr] & & x\alpha p_\alpha \alpha y \arrow[dl] \\
& x\alpha y &
\end{tikzcd}
\quad
\begin{tikzcd}
& x\alpha p_\alpha \alpha p_\beta y \arrow[dl] \arrow[dr] & \\
x\alpha p_\beta y \arrow[r, phantom, "="] & & x\alpha p_\beta y
\end{tikzcd}
\quad
\begin{tikzcd}
& x p_\alpha \alpha p_\alpha \alpha y \arrow[dl] \arrow[dr] & \\
x p_\alpha \alpha y \arrow[r, phantom, "="] & & x p_\alpha \alpha y
\end{tikzcd}$$

For all the rules of the form $z \to \bot$, the left-hand size contains at least two letters. We have to verify that the seamlessness property holds. The only interesting case is the following:

$$\text{if } \gamma \neq \delta: \; x\alpha p_\beta \underbrace{\beta p_\gamma \delta}_{\bot} y \to_R x \underbrace{\alpha p_\gamma \delta}_{\bot} y$$

The rewrite system $R_\bot$ consists of finitely many rules, because the set At is finite. The right-hand side of every rule is a single letter.

The R-ancestors of the action symbol p_β form the regular set $\mathcal{R}(e_{p_\beta})$, where $e_{p_\beta} = (\sum_{\alpha \in At} p_\alpha; \alpha^+)^*; p_\beta$. As in Example 1, we can show that $\mathsf{KA}_R \vdash \alpha^+ \equiv \alpha$ for every atom α. Reasoning in KA_R, we see that $e_{p_\beta} \equiv (\sum_\alpha p_\alpha; \alpha)^*; p_\beta$ and $p_\beta \leq e_{p_\beta}$. To prove $e_{p_\beta} \leq p_\beta$, it suffices to show:

$$(\textstyle\sum_\alpha p_\alpha; \alpha)^*; p_\beta \leq p_\beta \Longleftarrow (\sum_\alpha p_\alpha; \alpha); p_\beta \leq p_\beta,$$

which holds because $p_\alpha; \alpha; p_\beta \leq p_\beta$ for every atom α. The R-ancestors of the atom β for the regular set $\mathcal{R}(e_\beta)$, where

$$e_\beta = \beta^+ + \beta^+; (\textstyle\sum_\alpha p_\alpha; \alpha^+)^*; p_\beta; \beta^+.$$

We have already shown in KA_R that $e_{p_\beta} \equiv p_\beta$, and therefore $e_\beta \equiv \beta + \beta; p_\beta; \beta \equiv \beta + \beta \equiv \beta$. The only ancestor of dup is dup, so we put $e_{\mathsf{dup}} = \mathsf{dup}$. Let E be the associated equations, and suppose that $\mathcal{G}_{R_\bot}(e_1) = \mathcal{G}_{R_\bot}(e_2)$. It follows that $\mathsf{KA}_E \vdash e_1 \equiv e_2$, and therefore $\mathsf{NetKAT} \vdash e_1 \equiv e_2$.

One more application

We consider the case of KAT with extra equations of the form $p; b \equiv b; p$, where b is a test and p is an atomic action. We claim that the equation $p; b \equiv b; p$ is equivalent to the conjunction of the following equations:

$$\beta; p; \gamma \equiv 0 \text{ (for } \beta \leq b, \text{ and } \gamma \leq \neg b)$$
$$\gamma; p; \beta \equiv 0 \text{ (for } \gamma \leq \neg b, \text{ and } \beta \leq b)$$

For one direction of the claim, we observe that:

$$p; b \equiv \textstyle\sum_{\gamma \in At,\, \beta \leq b} \gamma; p; \beta \equiv \left(\sum_{\gamma \leq b,\, \beta \leq b} \gamma; p; \beta\right) + \sum_{\gamma \leq \neg b,\, \beta \leq b} \gamma; p; \beta \equiv \sum_{\gamma, \beta \leq b} \gamma; p; \beta$$

$$b; p \equiv \textstyle\sum_{\beta \leq b,\, \gamma \in At} \beta; p; \gamma \equiv \left(\sum_{\beta \leq b,\, \gamma \leq b} \beta; p; \gamma\right) + \sum_{\beta \leq b,\, \gamma \leq \neg b} \beta; p; \gamma \equiv \sum_{\beta, \gamma \leq b} \beta; p; \gamma$$

It follows that $p;b \equiv b;p$. For the other direction of the claim, we have:

$$\text{for } \beta \leq b \text{ and } \gamma \leq \neg b : \beta;p;\gamma \leq b;p;\gamma \equiv p;b;\gamma \equiv 0 \implies \beta;p;\gamma \equiv 0$$

$$\text{for } \gamma \leq \neg b \text{ and } \beta \leq b : \gamma;p;\beta \leq \gamma;p;b \equiv \gamma;b;p \equiv 0 \implies \gamma;p;\beta \equiv 0$$

So, w.l.o.g. we can consider equations of the form $\beta;p;\gamma \equiv 0$, for atoms β, γ and atomic action p. This is exactly like the case of Hoare hypotheses, and so we obtain a completeness theorem.