On Series-Parallel Pomset Languages: Rationality, Context-Freeness and Automata[☆☆]

Tobias Kappé[a], Paul Brunet[a], Bas Luttik[b], Alexandra Silva[a], Fabio Zanasi[a]

[a] *University College London, London, United Kingdom*
[b] *Eindhoven University of Technology, Eindhoven, The Netherlands*

Abstract

Concurrent Kleene Algebra (CKA) is a formalism to study programs exhibiting concurrent behaviour. As with previous Kleene Algebra extensions, developing a correspondence between denotational and operational perspectives is important, for both foundations and applications. This paper takes an important step towards this, by precisely relating bi-Kleene Algebra (BKA), a relaxed version of CKA, to a novel type of automata called *pomset automata*.

We show that pomset automata can faithfully represent the BKA-semantics of series-parallel rational expressions, and that a (structurally restricted) class of pomset automata can be translated back to these expressions. Furthermore, we characterise the behavior of general pomset automata in terms of context-free pomset grammars, thus yielding strong undecidability results.

Keywords: Concurrency, Series-Rational Expressions, Kleene Algebra, Pomsets, Pomset Automata, Brzozowski derivatives, Kleene theorem

1. Introduction

In their CONCUR'09 paper [2], Hoare, Möller, Struth, and Wehrman introduced *Concurrent Kleene Algebra* (CKA) as a mathematical framework suitable for the study of concurrent programs, in the hope of achieving the same elegance that Kozen did when using Kleene Algebra (KA) and extensions to provide a verification platform for sequential programs.

[☆] This paper is an extended version of a paper published at CONCUR'17 [1].

[☆☆] This work was partially supported by the ERC Starting Grant ProFoundNet (grant code 679127). F. Zanasi acknowledges support from EPSRC grant n. EP/R020604/1.

Email address: tkappe@cs.ucl.ac.uk (Tobias Kappé)

CKA is a seemingly simple extension of KA: it adds parallel analogues to the sequential composition and Kleene star operators, as well as the *exchange law*, which encodes the possibility of interleaving. Extending the KA toolkit, however, is challenging; in particular, an operational perspective is missing. In contrast, the correspondence between denotational and operational aspects of KA is well-understood through Kleene's theorem [3], which provided a pillar for characterising the free model [4] and establishing a decision procedure [5].

With this in mind, we pursue a version of Kleene's theorem for CKA. Specifically, we study *series-parallel rational expressions* (or *spr-expressions*), with a denotational model in terms of pomset languages. Our main contribution is a theorem which faithfully relates these expressions to a newly defined automaton model, called *pomset automata* (*PAs*). In a nutshell, PAs are automata where traces from certain states may branch into parallel threads; these threads contribute to the language when both reach an accepting state.

We are not the first to attempt such a characterisation. However, earlier works [6, 7] fall short of giving a precise correspondence between the denotational and operational models, due to the lack of a structural restriction on automata ensuring that only valid behaviours are accepted. In contrast, we propose such a restriction, which guarantees the soundness of a translation from the operational to denotational model. Furthermore, we propose a generalisation of Brzozowski derivatives [8] in the translation from expressions to automata, avoiding unnecessary ϵ-transitions and non-determinism that would result from a construction in the style of Thompson [9].

Since our denotational model does not take interleaving into account (and hence is not sound for the exchange law), our work is most accurately described as an operational model for *bi-Kleene Algebra* (BKA) [10]. We leave it to future work to incorporate the exchange law.

This work extends the conference paper [1] published at CONCUR'17 with all the previously omitted proofs and two new results. The first is the extension of the main theorem to incorporate the parallel variant of the Kleene star operator. The second is a characterisation of the behaviors of finite pomset automata in terms of context-free grammars (CFGs) [11].

The paper is organised as follows. We recall preliminaries in Section 2, and introduce PAs in Section 3. We translate a class of PAs to equivalent spr-expressions in Section 4, and describe the reverse construction in Section 5. We characterise finite PAs in terms of CFGs in Section 6. We discuss related work in Section 7; directions for further work appear in Section 8.

To preserve the flow of the narrative, some proofs appear in the appendices; proofs that are entirely routine are omitted altogether.

2. Preliminaries

Let S be a set; we write 2^S for the set of all subsets of S. We refer to a relation $\prec$ on S as *well-founded* if there are no infinite descending $\prec$-chains, i.e., no $\{s_n\}_{n\in\mathbb{N}} \subseteq S$ such that for all $n \in \mathbb{N}$ it holds that $s_{n+1} \prec s_n$.

Throughout the paper we fix a finite set Σ called the *alphabet*, whose elements are symbols usually denoted by a, b, etc. Lastly, if $\rightarrow \subseteq X \times Y \times Z$ is a ternary relation, we write $x \xrightarrow{y} z$ instead of $\langle x, y, z\rangle \in \rightarrow$.

2.1. Pomsets

Partially-ordered multisets, or *pomsets* [12, 13] for short, generalise words to a setting where actions (elements from Σ) may take place not just sequentially, but also in parallel. We recall a rigorous definition of pomsets, as well as some useful fundamental notions from literature [12, 13, 6, 14, 10].

Definition 2.1. A *labelled poset* is a tuple $\langle C, \leq_C, \lambda_C\rangle$ consisting of a *carrier* set C, a partial order $\leq$ on C and a *labelling* $\lambda : C \to \Sigma$.

A *labelled poset isomorphism* is a bijection between carriers that bijectively preserves labels and ordering. A *pomset* is an isomorphism class of labelled posets; we write $\langle\!|C, \leq, \lambda|\!\rangle$ to denote the pomset represented by $\langle C, \leq, \lambda\rangle$.

For instance, suppose a recipe for caramel-glazed cookies tells us to (i) *prepare* cookie dough, (ii) *bake* cookies in the oven, (iii) *caramelise* sugar, (iv) *glaze* the finished cookies. Here, step (i) precedes steps (ii) and (iii). Furthermore, step (iv) succeeds both steps (ii) and (iii). A pomset representing this process could be $U = \langle\!|C_U, \leq_U, \lambda_U|\!\rangle$, where $C_U = \{\text{(i)}, \text{(ii)}, \text{(iii)}, \text{(iv)}\}$ and $\leq_U$ is such that (i) $\leq_U$ (ii) $\leq_U$ (iv) and (i) $\leq_U$ (iii) $\leq_U$ (iv); λ_U associates with the elements of C_U the corresponding steps in the recipe.

We use Pom to denote the collection of all pomsets. Labelled posets and pomsets with a countable carrier suffice for our purposes. For this reason, we can (w.l.o.g.) adopt the convention that the carrier of a labelled poset representing a pomset is a subset of $\mathbb{N}$, which makes Pom a proper set.

Words over Σ are identified with totally ordered pomsets; multisets over Σ are similarly identified with pomsets having a discrete (diagonal) order. We write 1 for the empty pomset, and use $a \in \Sigma$ to refer to the *primitive* pomset with a single point labelled a (and the obvious order). Finally, we use the symbols $U, V, \ldots$ to denote pomsets.

Definition 2.2. Let $U = \langle\!|C_U, \leq_U, \lambda_U|\!\rangle$ and $V = \langle\!|C_V, \leq_V, \lambda_V|\!\rangle$ be pomsets. Without loss of generality, we can assume that C_U and C_V are disjoint.

The *sequential composition* of U and V, denoted $U \cdot V$, is the pomset

$$\langle\!|C_U \cup C_V,\ \leq_U \cup \leq_V \cup (C_U \times C_V),\ \lambda_U \cup \lambda_V|\!\rangle$$

The *parallel composition* of U and V, denoted $U \parallel V$, is the pomset

$$\langle\!|C_U \cup C_V,\ \leq_U \cup \leq_V,\ \lambda_U \cup \lambda_V|\!\rangle$$

Here, $\lambda_U \cup \lambda_V : C_U \cup C_V \to \Sigma$ agrees with λ_U on C_U, and with λ_V on C_V.

Sequential composition forces the events in the left pomset to be ordered before those in the right pomset. We note that these operators are well-defined modulo isomorphism of labelled posets, and that 1 is the unit for both sequential and parallel composition.

Definition 2.3. The set of *series-parallel* pomsets [12, 13], $\mathsf{Pom}^{\mathsf{sp}}$, is the smallest subset of Pom that includes the empty and primitive pomsets and is closed under sequential and parallel composition.

In this paper we concern ourselves with series-parallel pomsets. For inductive reasoning, it is useful to recall part of [12, Theorem 3.1].

Lemma 2.4. *Let $U \in \mathsf{Pom}^{\mathsf{sp}}$. If U is non-empty, then* exactly one *of the following is true: (i) $U = a$ for some $a \in \Sigma$, or (ii) $U = V \cdot W$ for non-empty $V, W \in \mathsf{Pom}^{\mathsf{sp}}$, strictly smaller than U, or (iii) $U = V \parallel W$ for non-empty $V, W \in \mathsf{Pom}^{\mathsf{sp}}$, strictly smaller than U.*

We can quantify the degree of nesting of parallel and sequential composition of a series-parallel pomset as follows.

Definition 2.5. The *depth* of a series-parallel pomset U [14], denoted $\partial(U)$, is defined inductively, as follows. First, if $U = 1$, then $\partial(U) = 0$. Second, if $U = a$ for some $a \in \Sigma$, then $\partial(U) = 1$. Third, if $U = U_0 \cdots U_{n-1}$ or $U = U_0 \parallel \cdots \parallel U_{n-1}$ for non-empty pomsets $U_0, \ldots, U_{n-1}$, then

$$\partial(U) = \max(\partial(U_0), \ldots, \partial(U_{n-1})) + 1$$

Note that depth is always well-defined, as a consequence of Lemma 2.4.

2.2. Pomset languages

We can group the words that represent traces arising from a sequential program into a set called a *language*. By analogy, we can group the pomsets that represent the traces arising from a parallel program into a *pomset language*. We use calligraphic symbols $\mathcal{U}, \mathcal{V}, \ldots$ to denote pomset languages.

For instance, suppose that the recipe for glazed cookies has an optional fifth step where chocolate sprinkles are spread over the cookies. In that case, there are *two* pomsets that describe a trace arising from the recipe, U^+ and U^-, either with or without the chocolate sprinkles. The pomset language $\mathcal{U} = \{U^-, U^+\}$ contains the traces that arise from the new recipe.

The composition operators for pomsets can be lifted to pomset languages. We also define two types of Kleene closure operator, similar to the one defined on languages of words, for both parallel and sequential composition.

Definition 2.6. Let $\mathcal{U}$ and $\mathcal{V}$ be pomset languages. We define:

$$\mathcal{U} \cdot \mathcal{V} = \{U \cdot V : U \in \mathcal{U}, V \in \mathcal{V}\} \qquad \mathcal{U}^* = \bigcup_{n \in \mathbb{N}} \mathcal{U}^n$$

$$\mathcal{U} \parallel \mathcal{V} = \{U \parallel V : U \in \mathcal{U}, V \in \mathcal{V}\} \qquad \mathcal{U}^\dagger = \bigcup_{n \in \mathbb{N}} \mathcal{U}^{(n)}$$

in which $\mathcal{U}^0 = \mathcal{U}^{(0)} = \{1\}$, and for all $n \in \mathbb{N}$ we define

$$\mathcal{U}^{n+1} = \mathcal{U} \cdot \mathcal{U}^n \qquad \mathcal{U}^{(n+1)} = \mathcal{U} \parallel \mathcal{U}^{(n)}$$

Sequential Kleene closure models indefinite repetition. For instance, if our cookie recipe has a final step "repeat as necessary", the pomset language $\mathcal{U}^*$ represents all possible traces of repetitions of the recipe; e.g., $U^+ \cdot U^+ \cdot U^- \in \mathcal{U}^*$ is the trace where first two batches of sprinkled cookies are made, followed by one without sprinkles. In contrast, parallel Kleene closure models unbounded parallelism; in this case, $\mathcal{U}^\dagger$ represents all possible traces of parallel executions of the recipe; e.g., $U^+ \parallel U^- \in \mathcal{U}^\dagger$ is the trace where we make two batches of cookies in parallel, one with and one without sprinkles.

2.3. Series-parallel rational expressions

Just as a rational expression can be used to describe a rational structure of sequential events, so too can a series-parallel rational expression be used to describe a rational structure of possibly parallel events. Series-parallel rational expressions can be thought of as rational expressions with parallel composition, as well as a parallel analogue to the Kleene star.

Definition 2.7. We use $\mathcal{T}$ to denote the set of *series-parallel rational expressions* (*spr-expressions*, for short) [6], formed by the grammar

$$e, f ::= 0 \mid 1 \mid a \in \Sigma \mid e + f \mid e \cdot f \mid e \parallel f \mid e^* \mid e^\dagger$$

For a closed propositional formula Φ, we write $[\Phi]$ as shorthand for 1 if Φ is satisfied, and 0 otherwise. We use e, f, g and h to denote spr-expressions.

Series-rational expressions have a semantics in terms of pomset languages.

Definition 2.8. The function $[\![-]\!] : \mathcal{T} \to 2^{\mathsf{Pom}^{\mathsf{sp}}}$ is defined [6] inductively:

$$[\![0]\!] = \emptyset \qquad [\![a]\!] = \{a\} \qquad [\![e \cdot f]\!] = [\![e]\!] \cdot [\![f]\!] \qquad [\![e^*]\!] = [\![e]\!]^*$$
$$[\![1]\!] = \{1\} \qquad [\![e + f]\!] = [\![e]\!] \cup [\![f]\!] \qquad [\![e \parallel f]\!] = [\![e]\!] \parallel [\![f]\!] \qquad [\![e^\dagger]\!] = [\![e]\!]^\dagger$$

If $\mathcal{U} \subseteq \mathsf{Pom}^{\mathsf{sp}}$ such that $\mathcal{U} = [\![e]\!]$ for some $e \in \mathcal{T}$, then $\mathcal{U}$ is said to be a *series-parallel rational language*, or *spr-language* for short.

To illustrate, the pomset language $\mathcal{U}^* = \{U^+, U^-\}^*$ describes the possible traces arising from indefinitely repeating the cookie recipe, optionally adding chocolate sprinkles at every repetition. We can describe the pomset languages $\{U^-\}$ and $\{U^+\}$ with the series-parallel rational expressions

$$e^- = \mathsf{prepare} \cdot (\mathsf{bake} \parallel \mathsf{caramelise}) \cdot \mathsf{glaze} \qquad e^+ = e^- \cdot \mathsf{sprinkle}$$

which yields the spr-expression $e = e^- + e^+$ for $\mathcal{U}$; hence, $[\![e^*]\!] = \mathcal{U}^*$.

Note that spr-expressions without $\parallel$ and $\dagger$ are rational expressions, and spr-expressions without $\cdot$ and $*$ are commutative rational expressions. To see that spr-expressions are a proper extension of rational and commutative rational expressions, we observe the following.

Lemma 2.9. *If $e \in \mathcal{T}$. The following are true. (i) If $[\![e]\!]$ consists of words, then $[\![e]\!]$ is a rational language. (ii) If $[\![e]\!]$ consists of multisets, then $[\![e]\!]$ is a commutative rational language.*

We conclude our discussion of pomset languages by recalling the following lemma, which is useful when analysing the series-parallel rationality of a language. For details, refer to [6, 10].

Lemma 2.10. *If $\mathcal{U}$ is an spr-language, then there exists an $n \in \mathbb{N}$ such that for all $U \in \mathcal{U}$ it holds that $\partial(U) \leq n$.*

More specifically, the above lemma tells us that when we want to show that a pomset language $\mathcal{U}$ is not series-parallel rational, it suffices to find a sequence $\{U_n\}_{n \in \mathbb{N}} \subseteq \mathcal{U}$ such that for $n \in \mathbb{N}$ it holds that $\partial(U_n) < \partial(U_{n+1})$.

3. Pomset Automata

We now describe an automaton model to recognise pomset languages.

Definition 3.1. A *pomset automaton (PA)* is a tuple $\langle Q, \delta, \gamma, F \rangle$ where Q is a set of *states*, with $F \subseteq Q$ the *accepting states*, $\delta : Q \times \Sigma \to Q$ is a function called the *sequential transition function*, and $\gamma : Q \times Q \times Q \to Q$ is a function called the *parallel transition function*.

We do not fix an initial state, and thus a PA does not define a single pomset language but rather a mapping from states to pomset languages. This mapping is defined in terms of a trace relation arising from δ and γ, as follows.

Definition 3.2. Let $A = \langle Q, \delta, \gamma, F \rangle$ be a PA. We define $\to_A \subseteq Q \times \mathsf{Pom}^{\mathsf{sp}} \times Q$ as the smallest relation that satisfies the rules

$$\frac{q \in Q}{q \xrightarrow{1}_A q} \qquad \frac{q \in Q \qquad a \in \Sigma}{q \xrightarrow{a}_A \delta(q,a)} \qquad \frac{q \xrightarrow{U}_A q'' \qquad q'' \xrightarrow{V}_A q'}{q \xrightarrow{U \cdot V}_A q'} \qquad \frac{r \xrightarrow{U}_A r' \in F \qquad s \xrightarrow{V}_A s' \in F}{q \xrightarrow{U \parallel V}_A \gamma(q,r,s)}$$

The *language* of A at $q \in Q$ is $L_A(q) = \left\{ U \in \mathsf{Pom}^{\mathsf{sp}} : \exists q' \in F.\ q \xrightarrow{U}_A q' \right\}$. We say that A *accepts* the pomset language $\mathcal{U}$ if $L_A(q) = \mathcal{U}$ for some $q \in Q$.

In the above, δ plays the same role as in classic finite automata: given a state and a symbol, it returns the new state after reading that symbol. The function γ warrants a bit more explanation: given states $q, r, s \in Q$, it tells us the state that is reached from q after reading two pomsets in parallel starting at states r and s, and having both "subprocesses" reach an accepting state.

For the remainder of this section, we fix a PA $A = \langle Q, \delta, \gamma, F \rangle$. Individual triplets in the trace relation are referred to as *traces*. It is useful to establish some terminology when referring to traces. Specifically:

- If $q \xrightarrow{U}_A q'$ with $q = q'$ and $U = 1$, then $q \xrightarrow{U}_A q$ is a *trivial* trace.
- If $q \xrightarrow{U}_A q'$ such that $U = a$ for some $a \in \Sigma$ and $q' = \delta(q, a)$, we say that $q \xrightarrow{U}_A q'$ is a *δ-trace*.
- If $U = V \parallel W$ and there exist $r, s \in Q$ and $r', s' \in F$ such that $r \xrightarrow{V}_A r'$ and $s \xrightarrow{W}_A s'$ and $q' = \gamma(q, r, s)$, this trace is a *γ-trace*.

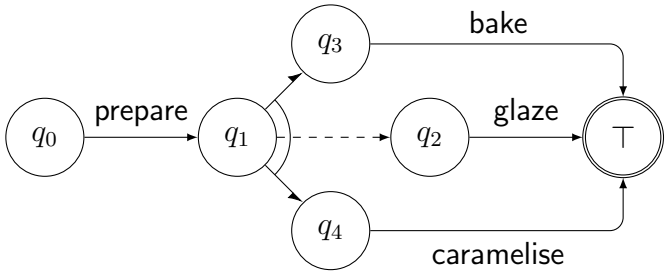


Figure 1: Pomset automaton accepting U.

The δ-traces and γ-traces are collectively known as *unit* traces.

To simplify matters later on, we assume that every PA A has states $\bot \in Q \setminus F$ and $\top \in F$ such that (i) for all $a \in \Sigma$, it holds that $\delta(\bot, a) = \delta(\top, a) = \bot$, and (ii) for all $r, s \in Q$, it holds that $\gamma(\bot, r, s) = \gamma(\top, r, s) = \bot$.

The state $\bot$ is useful when defining γ: for a fixed $q \in Q$, not all $r, s \in Q$ may give a value of $\gamma(q, r, s)$ that contributes to $L_A(q)$; in such cases, we set $\gamma(q, r, s) = \bot$.[1] The state $\top$ fulfills a similar role: it is used to signal that the target of a parallel transition accepts, but allows no further continuation of the trace; this will be important in Section 4, when we describe a class of pomset automata that admit a translation back to spr-expressions.

Lemma 3.3. *Let $q \xrightarrow{U}_A q'$ be non-trivial. If $q = \bot$ or $q = \top$, then $q' = \bot$.*

We draw a PA in a way similar to finite automata: each state (except $\bot$) is a vertex, and accepting states are marked by a double border. To represent sequential transitions, we draw labelled edges; for instance, in Figure 1, $\delta(q_0, \mathsf{prepare}) = q_1$. To represent parallel transitions, we draw hyper-edges; for instance, in Figure 1, $\gamma(q_1, q_3, q_4) = q_2$. To avoid clutter, we do not draw either of these edge types when the target state is $\bot$. It is not hard to verify that the pomset U of the earlier example is accepted by the PA in Figure 1.

3.1. Finite support

Deterministic automata with infinitely many states can accept non-rational languages. Since spr-languages extend rational languages by Lemma 2.9, and

[1] Alternatively, we could have allowed γ to be a partial function; including $\bot$ as a state, however, will simplify part of our construction in Section 5.

PAs obviously extend deterministic automata, it follows that allowing PAs with infinitely many states would dash our hopes of a Kleene theorem.

On the other hand, it is useful to work with PAs that have infinitely many states, as we shall see in Section 5. To strike a middle ground, we identify a class of PAs with possibility infinitely many states that, for any state, allow a restriction to a PA with finitely many states accepting the same language.

Definition 3.4. The *trace dependency relation* of A, denoted $\preceq_A$, is the smallest preorder on Q that satisfies the rules

$$\frac{}{\bot \preceq_A q} \qquad \frac{q, r, s \in Q \qquad \gamma(q, r, s) \neq \bot}{r, s \preceq_A q} \qquad \frac{a \in \Sigma \qquad q \in Q}{\delta(q, a) \preceq_A q} \qquad \frac{q, r, s \in Q}{\gamma(q, r, s) \preceq_A q}$$

It should be emphasised that, in general, $\preceq_A$ is not a partial order — anti-symmetry may fail because of loops in the transition structure.

We write $\prec_A$ for the *strict trace dependency relation*, which is the strict order that arises by setting $q \prec_A q'$ if and only if $q \preceq_A q'$ and $q' \not\preceq_A q$.

Definition 3.5. We say that $Q' \subseteq Q$ is *closed* in A when Q' is downward-closed with respect to $\preceq_A$ — that is, for all $q \in Q'$ and $r \in Q$ such that $r \preceq_A q$, it follows that $r \in Q'$. We write $\pi_A(q)$ for the *support* of q in A, which is the smallest closed subset of Q that contains q. We say that A is *finitely supported* if for all $q \in Q$ it holds that $\pi_A(q)$ is finite.

With this definition, the following is not hard to see.

Lemma 3.6. *If A is finitely supported, then for every $q \in Q$ there exists a finite pomset automaton A_q with a state q', such that $L_A(q) = L_{A_q}(q')$.*

Finite support is also useful in that it ensures well-foundedness of the strict trace dependency relation.

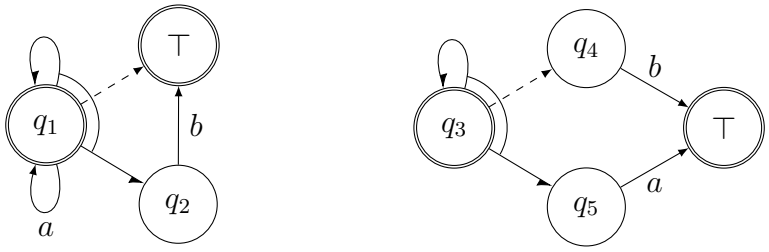
Lemma 3.7. *If A is finitely supported, then $\prec_A$ is well-founded.*

3.2. Trace length

We conclude this section with the following technical lemma, which gives us an alternative inductive handle for the lemmas to come.

Lemma 3.8. *If $q \xrightarrow{U}_A q'$, then there exist $q_0, \ldots, q_\ell \in Q$ with $q = q_0$ and $q_\ell = q'$, and $U = U_0 \cdots U_{\ell-1}$ such that for $0 \leq i < \ell$ it holds that $q_i \xrightarrow{U_i}_A q_{i+1}$. Furthermore, each of these traces is a unit trace.*

The minimal ℓ for a given trace as obtained from the above lemma is known as the *length* of the trace. Note that a trace of zero length is necessarily trivial, and a trace of unit length is necessarily a unit trace.



(a) A_0 (b) A_1

Figure 2: Finitely supported pomset automata that accept languages of unbounded depth.

4. Automata to expressions

Let us fix a finitely supported PA $A = \langle Q, \delta, \gamma, F \rangle$. We set out to find for every $q \in Q$ an $e_q \in \mathcal{T}$ such that $L_A(q) = [\![e_q]\!]$. Before we get started, however, it should be noted that not all finitely supported (or even finite) pomset automata admit such a translation. This is because δ and γ can conspire to create a state with a language of unbounded depth; Lemma 2.10 then tells us that the corresponding spr-expression cannot exist.

Example 4.1. Consider the PA A_0 in Figure 2a. Here, we have that

$$q_1 \xrightarrow{a}_{A_0} \delta(q_1, a) = q_1 \qquad q_2 \xrightarrow{b}_{A_0} \delta(q_2, b) = \top$$

Since $q_1, \top \in F$, we have $q_1 \xrightarrow{a \| b}_{A_0} \gamma(q_1, q_1, q_2) = \top$ and $q_1 \xrightarrow{a \cdot (a \| b)}_{A_0} \top$. Hence,

$$q_1 \xrightarrow{a \cdot (a \| b) \| b}_{A_0} \gamma(q_1, q_1, q_2) = \top \qquad q_1 \xrightarrow{a \cdot (a \cdot (a \| b) \| b)}_{A_0} \top$$

We can repeat this indefinitely, thereby showing that

$$\{1, a, a \parallel b, a \cdot (a \parallel b), a \cdot (a \parallel b) \parallel b, \ldots\} \subseteq L_{A_0}(q_1)$$

Consequently, there is no $e \in \mathcal{T}$ such that $L_{A_0}(q_1) = [\![e]\!]$, by Lemma 2.10.

Example 4.2. Consider the PA A_1 in Figure 2b. Here,

$$q_5 \xrightarrow{a}_{A_1} \delta(q_5, a) = \top \qquad q_4 \xrightarrow{b}_{A_1} \delta(q_4, b) = \top \qquad q_3 \xrightarrow{1}_{A_1} q_3$$

Since $q_3 \in F$, we have $q_3 \xrightarrow{1 \| a}_{A_1} \gamma(q_3, q_3, q_5) = q_4$ and $q_3 \xrightarrow{a \cdot b}_{A_1} \top$. Hence,

$$q_3 \xrightarrow{a \cdot b \| a}_{A_1} \gamma(q_3, q_3, q_5) = q_4 \qquad q_3 \xrightarrow{(a \cdot b \| a) \cdot b}_{A_1} \top$$

We can repeat the above to show that $\{1, a \cdot b, ((a \cdot b) \parallel a) \cdot b, \ldots\} \subseteq L_{A_1}(q_3)$. By Lemma 2.10, we then find that there is no $e \in \mathcal{T}$ with $L_{A_1}(q_3) = [\![e]\!]$.

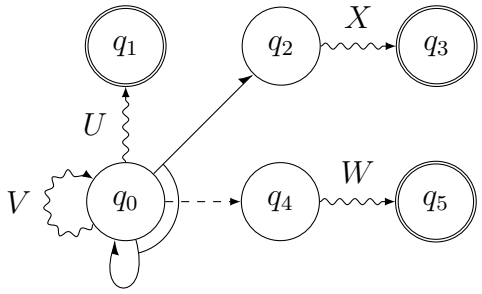


Figure 3: A template for a state with a language of unbounded depth.

To get around the problem of unbounded languages, we structurally restrict pomset automata in such a way that such behavior is excluded. To do this, we need to get a handle on the constellation of states and transitions common to the examples above that allows the depth of the pomset languages accepted by A_0 and A_1 to run amok; this is done in the following lemma.

Lemma 4.3. *Let $\{q_0, q_2, q_4\} \subseteq Q$ and $\{q_1, q_3, q_5\} \subseteq F$. Furthermore, let $U, V, W, X \in \mathsf{Pom}^{\mathsf{sp}}$ be such that the following (c.f. Figure 3) hold:*

$$q_0 \xrightarrow{U}_A q_1 \qquad q_0 \xrightarrow{V}_A q_0 \qquad q_2 \xrightarrow{X}_A q_3 \qquad q_4 \xrightarrow{W}_A q_5 \qquad \gamma(q_0, q_2, q_0) = q_4$$

If $X \neq 1$, and moreover $W \neq 1$ or $V \neq 1$, then $L_A(q_0)$ has unbounded depth.

Proof. Suppose that $Y \in L_A(q_0)$, i.e., $q_0 \xrightarrow{Y}_A q'$ for some $q' \in F$. Given the traces and the fork transition in the premises, we can then derive that

$$q_0 \xrightarrow{(X \parallel (V \cdot Y)) \cdot W} q_5 \in F$$

and hence $(X \parallel (V \cdot Y)) \cdot W \in L_A(q_0)$. Thus, $L_A(q_0)$ is closed under

$$f : \mathsf{Pom}^{\mathsf{sp}} \to \mathsf{Pom}^{\mathsf{sp}} \quad \text{given by} \quad f(-) = (X \parallel (V \cdot -)) \cdot W$$

By the premise that $X \neq 1$ as well as $W \neq 1$ or $V \neq 1$, it follows that for $Y \in \mathsf{Pom}^{\mathsf{sp}}$ we have $\partial(Y) < \partial(f(Y))$. Because $U \in L_A(q_0)$, we can point to $\{U, f(U), f^2(U), \ldots\}$ as a set of unbounded depth contained in $L_A(q_0)$, and thus conclude that this pomset language has unbounded depth. □

To counteract the pattern summarised above, we propose the following.

Definition 4.4. We say that $q \in Q$ is *sequential* if for all $r, s \in Q$ with $\gamma(q, r, s) \neq \bot$, it holds that $r, s \prec_A q$. We say that $q \in F$ is *recursive* if (i) it is not sequential, and (ii) for all $a \in \Sigma$ we have $\delta(q, a) = \bot$, and (iii) if $r, s \in Q$ and $\gamma(q, r, s) \neq \bot$, then $s = q$ and $r \prec_A q$, and $\gamma(q, r, s) = \top$.

We write Q_{seq} (resp. Q_{rec}) for the set of states in Q that are sequential (resp. recursive), and say that A *well-nested* if $Q = Q_{\mathsf{seq}} \cup Q_{\mathsf{rec}}$.

One easily sees that A_0 and A_1 are not well-nested: neither q_1 nor q_3 is sequential, because of their self-forks, but q_1 is not recursive because $\delta(q_1, a) = q_1 \neq \bot$, and q_3 is not recursive because $\gamma(q_3, q_5, q_3) = q_4 \notin \{\top, \bot\}$.

As a matter of fact, Definition 4.4 is slightly overzealous — strictly speaking, there are non-well-nested PAs which accept spr-languages exclusively. We will show in Section 6 that checking for series-parallel rationality of a finite PA is undecidable, and must therefore accept that any decidable restriction that enforces series-parallel rationality will forbid certain valid automata.

In Section 5, we shall associate with every spr-expression e a finitely supported *and* well-nested PA that accepts $[\![e]\!]$. The bi-directional correspondence between spr-expressions and pomset automata is therefore maintained.

To ease notation, we assume for the remainder of this section that A is well-nested. We shall establish that for every state q of A there exists an spr-expression e_q such that $L_A(q) = [\![e_q]\!]$. Since $\prec_A$ is well-founded, we can proceed by induction on $\prec_A$, i.e., the induction hypothesis for q is that for all $r \in Q$ with $r \prec_A q$ we can construct an $e_r \in \mathcal{T}$ such that $[\![e_r]\!] = L_A(r)$.

The language of a recursive state is not very hard to characterise.

Lemma 4.5. *If $q \in Q_{\mathsf{rec}}$, then*

$$L_A(q) = \left(\bigcup\nolimits_{\gamma(q,r,q)=\top} L_A(r) \right)^{\dagger}$$

The languages of sequential states for which our induction hypothesis holds can also be characterised. To do this, we modify the procedure for finding a rational expression for a state in a finite automaton [15].

Definition 4.6. Let $S \subseteq Q_{\mathsf{seq}}$, and suppose that for all $s \in S$, the induction hypothesis for s holds. For $q \in S$ and $q' \in Q$, we define $e^S_{qq'} \in \mathcal{T}$, as follows. If $q' = \bot$, we set $e^S_{qq'} = 0$. For the remaining cases, we define $e^S_{qq'}$ inductively. If $S = \emptyset$, then

$$e^S_{qq'} = [q = q'] + \sum_{\delta(q,a)=q'} a + \sum_{\gamma(q,r,s)=q'} e_r \parallel e_s$$

otherwise, if $S' = S \setminus \{q''\}$, then

$$e^S_{qq'} = e^{S'}_{qq'} + e^{S'}_{qq''} \cdot \left(e^{S'}_{q''q''}\right)^* \cdot e^{S'}_{q''q'}$$

Note that $e^{\emptyset}_{qq'}$ is well-defined, for if $\gamma(q,r,s) = q' \neq \bot$, then $r, s \prec_A q$ by the fact that q is sequential, and thus e_r and e_s exist. Also, the second sum is finite by the fact that A is finitely supported.

Lemma 4.7. *Let $S \subseteq Q_{\mathsf{seq}}$, and suppose that for all $s \in S$, the induction hypothesis holds. Let $q \in S$ and $q' \in Q$. Then $U \in [\![e^S_{qq'}]\!]$ if and only if $q' \neq \bot$ and there exist $q_0, \ldots, q_{\ell-1} \in S$, and $U = U_0 \cdots U_{\ell-1}$ with*

$$q = q_0 \xrightarrow{U_0}_A q_1 \xrightarrow{U_1}_A \cdots \xrightarrow{U_{\ell-2}}_A q_{\ell-1} \xrightarrow{U_{\ell-1}}_A q_\ell = q'$$

and, furthermore, for $0 \leq i < \ell$ we have that $q_i \xrightarrow{U_i}_A q_{i+1}$ is a unit trace.

With all this in hand, we are finally ready to construct series-parallel rational expressions from pomset automata.

Lemma 4.8. *If the induction hypothesis for q holds, then we can find $e_q \in \mathcal{T}$ such that $[\![e_q]\!] = L_A(q)$.*

Proof. More generally, we show that for $q' \in Q$ with $q \preceq_A q' \preceq_A q$, we can find $e_{q'} \in \mathcal{T}$ such that $L_A(q') = e_{q'}$. We partition these states as follows

$$R = \{q' \in Q \cap Q_{\mathsf{rec}} : q \preceq_A q' \preceq_A q\}$$
$$S = \{q' \in Q \cap Q_{\mathsf{seq}} : q \preceq_A q' \preceq_A q\}$$

Note that the induction hypothesis holds for all states in $R \cup S$: if $r \prec_A q' \preceq_A q$, then $r \prec_A q$; hence, for r we can find an expression e_r, such that $[\![e_r]\!] = L_A(r)$. Furthermore, R and S are finite, for A is finitely supported.

We carry on to find expressions for the languages of states in R. To this end, we define for $q' \in R$ that

$$e_{q'} = \left(\sum\nolimits_{\gamma(q',r,q')=\top} e_r\right)^\dagger$$

The above is well-defined, for if $\gamma(q',r,q') = \top$, then $r \prec_A q'$, and thus $e_r \in \mathcal{T}$ exists. Since A is finitely supported, the sum is finite. By Lemma 4.5, we find

$$[\![e_{q'}]\!] = \left(\bigcup\nolimits_{\gamma(q',r,q')=\top} [\![e_r]\!]\right)^\dagger = \left(\bigcup\nolimits_{\gamma(q',r,q')=\top} L_A(r)\right)^\dagger = L_A(q')$$

We now consider the states in S. For $q' \in S$, we define

$$e_{q'} = \sum_{r \in S \cap F} e^S_{q'r} + \sum_{r \in R} e^S_{q'r} \cdot e_r + \sum_{r \prec_A q'} e^S_{q'r} \cdot e_r$$

This expression is again well-defined, for all sums are finite, and e_r exists when $r \in R$ or $r \prec_A q'$ by the above, and furthermore the induction hypothesis holds for all $r \in S$ by the observation above.

It remains to show that, for $q' \in S$, it holds that $[\![e_{q'}]\!] = L_A(q')$. For the inclusion from left to right, suppose that $U \in [\![e_{q'}]\!]$. There are two cases.

- If $U \in [\![e^S_{q'r}]\!]$ for $r \in S \cap F$, then by Lemma 4.7 we find that $q' \xrightarrow{U}_A r$. Since $r \in F$, also $U \in L_A(q')$.
- If $U \in [\![e^S_{q'r} \cdot e_r]\!]$ for some $r \in R$ or $r \in Q$ with $r \prec_A q'$, then $U = V \cdot W$ such that $V \in [\![e^S_{q'r}]\!]$ and $W \in [\![e_r]\!]$. By Lemma 4.7, we find that $q' \xrightarrow{V}_A r$; also, we find that $r \xrightarrow{W}_A r'$ for some $r' \in F$. Together, this implies that $q' \xrightarrow{V \cdot W}_A r$, and since $r \in F$ also $U = V \cdot W \in L_A(q')$.

For the other inclusion, suppose that $U \in L_A(q')$, i.e., $q' \xrightarrow{U} r$ for some $r \in F$. By Lemma 3.8, there exist $q_0, \ldots, q_n \in Q$ with $q' = q_0$ and $r = q_n$, and $U = U_0 \cdots U_{n-1}$, such that for $1 \leq i < n$ it holds that $q_i \xrightarrow{U_i}_A q_{i+1}$. Furthermore, each of these traces is a unit trace. If $q_1, \ldots, q_n \in S$, then $U \in [\![e^S_{q'r}]\!] \subseteq [\![e_{q'}]\!]$ by Lemma 4.7.

Otherwise, i.e., when $q_i \notin S$ for some $0 < i \leq n$, let m be the smallest such i, and note that $U_m \cdots U_{n-1} \in L_A(q_m)$. Furthermore, for $0 \leq i < m$ we have that $q_i \in S$, and thus $U_0 \cdots U_{m-1} \in [\![e^S_{q'q_m}]\!]$, by Lemma 4.7. There are two cases to consider.

- If $q_m \in R$, then $L_A(q_m) = [\![e_{q_m}]\!]$ by the above. We conclude that
$$U = U_0 \cdots U_{m-1} \cdot U_m \cdots U_{n-1} \in \left[\!\!\left[e^S_{q'q_m} \cdot e_{q_m}\right]\!\!\right] \subseteq [\![e_{q'}]\!]$$
- Otherwise, if $q_m \notin R$, then since also $q_m \notin S$, we know that $q \not\preceq_A q_m$ or $q_m \not\preceq_A q$. The latter case can be excluded, for $q_m \preceq_A q' \preceq_A q$. We thus know that $q \not\preceq_A q_m$, and since $q \preceq_A q'$, also $q' \not\preceq_A q_m$. Together with $q_m \preceq_A q'$, it follows that $q_m \prec_A q'$; an argument similar to the previous case completes the proof. □

The above establishes the main result of this section.

Theorem 4.9. *Let A be a well-nested and finitely supported pomset automaton. For all states q of A, we can find $e_q \in \mathcal{T}$ such that $[\![e_q]\!] = L_A(q)$.*

5. Expressions to automata

We now turn our attention to the task of constructing a pomset automaton A that accepts the semantics of a given expression e. Since our algorithm for obtaining expressions from a pomset automaton is sound for finitely supported and well-nested PAs only, A should also satisfy these constraints. Our approach follows Brzozowski's method for constructing a deterministic finite automaton that accepts the semantics of a rational expression [8]. More precisely, we construct a finitely supported and well-nested automaton A_Σ, such that for every spr-expression e there exists a state q_e such that $L_A(q_e) = [\![e]\!]$. Intuitively, the transition structure of A_Σ is set up such that the automaton can transition from the state representing e to the state representing e' while reading a if and only if e' is what "remains" of e after consuming a — classically, this e' is called the *a-derivative* of e.

The encoding of spr-expressions into states requires some care. Specifically, if we choose to have a state for every spr-expression, it turns out that the resulting automaton is not finitely supported. This is not surprising; indeed, Brzozowski dealt with the same problem [8]. The solution is to encode spr-expressions into states by representing them as the equivalence classes of a congruence that is sound with respect to their semantics.

Definition 5.1. We define $\simeq$ as the smallest congruence on $\mathcal{T}$ such that:

$$e + 0 \simeq e \qquad e + e \simeq e \qquad e + f \simeq f + e$$

$$e + (f + g) \simeq (e + f) + g \qquad (e + f) \cdot g \simeq e \cdot g + f \cdot g$$

Thus, when $e \simeq f$, we know that e is equal to f, modulo associativity, commutativity and idempotence of $+$, and left-distributivity of $+$ over $\cdot$.

The set of equivalence classes of $\mathcal{T}$ modulo $\simeq$ is written $\mathcal{T}_\simeq$. To lighten notation, we represent the equivalence class of $e \in \mathcal{T}$ up to $\simeq$ by simply writing e; it will always be clear from the context whether we intend e as an element of $\mathcal{T}$ or $\mathcal{T}_\simeq$. We elide lemmas showing that our definitions are sound w.r.t. $\simeq$; arguments of this nature appear in Appendix C.6.

In analogy to Brzozowski's construction, where the accepting states are the rational expressions accepting the empty word, we characterise spr-expressions accepting the empty pomset to use as accepting states of our PA.

Definition 5.2. We define the set $\mathcal{F}$ as the smallest subset of $\mathcal{T}$ satisfying

$$\frac{}{1 \in \mathcal{F}} \qquad \frac{e \in \mathcal{F} \qquad f \in \mathcal{T}}{e + f, f + e \in \mathcal{F}} \qquad \frac{e, f \in \mathcal{F}}{e \cdot f \in \mathcal{F}} \qquad \frac{e, f \in \mathcal{F}}{e \parallel f \in \mathcal{F}} \qquad \frac{e \in \mathcal{T}}{e^*, e^\dagger \in \mathcal{F}}$$

Lemma 5.3. *Let $e \in \mathcal{T}$; then $e \in \mathcal{F}$ if and only if $1 \in [\![e]\!]$.*

We write $\mathcal{F}_{\simeq}$ to denote the set of congruence classes in $\mathcal{F}$ w.r.t. $\simeq$. Having identified the accepting states, we move on to the transition functions.

Definition 5.4. Let $e, f \in \mathcal{T}$. We use $e \star f$ to denote f when $e \in \mathcal{F}_{\simeq}$, and 0 otherwise; similarly, we write $e \fatsemi f$ for 0 when $e \simeq 0$, and $e \cdot f$ otherwise.

We define the function $\delta_\Sigma : \mathcal{T}_{\simeq} \times \Sigma \to \mathcal{T}_{\simeq}$ as follows:

$$\begin{aligned}
\delta_\Sigma(0,a) &= 0 & \delta_\Sigma(e \cdot f, a) &= \delta_\Sigma(e,a) \fatsemi f + e \star \delta_\Sigma(f,a) \\
\delta_\Sigma(1,a) &= 0 & \delta_\Sigma(e \parallel f, a) &= 0 \\
\delta_\Sigma(b,a) &= [a = b] & \delta_\Sigma(e + f, a) &= \delta_\Sigma(e,a) + \delta_\Sigma(f,a) \\
\delta_\Sigma(e^*,a) &= \delta_\Sigma(e,a) \fatsemi e^* & \delta_\Sigma(e^\dagger, a) &= 0
\end{aligned}$$

Furthermore, the function $\gamma_\Sigma : \mathcal{T}_{\simeq} \times \mathcal{T}_{\simeq} \times \mathcal{T}_{\simeq} \to \mathcal{T}_{\simeq}$ is defined as follows:

$$\begin{aligned}
\gamma_\Sigma(0,g,h) &= 0 & \gamma_\Sigma(e \cdot f, g, h) &= \gamma_\Sigma(e,g,h) \fatsemi f + e \star \gamma_\Sigma(f,g,h) \\
\gamma_\Sigma(1,g,h) &= 0 & \gamma_\Sigma(e \parallel f, g, h) &= [g \simeq e \wedge h \simeq f] \\
\gamma_\Sigma(b,g,h) &= 0 & \gamma_\Sigma(e + f, g, h) &= \gamma_\Sigma(e,g,h) + \gamma_\Sigma(f,g,h) \\
\gamma_\Sigma(e^*,g,h) &= \gamma_\Sigma(e,g,h) \fatsemi e^* & \gamma_\Sigma(e^\dagger, g, h) &= [g \simeq e \wedge h \simeq e^\dagger]
\end{aligned}$$

We write A_Σ for the *syntactic pomset automaton*, which is $\langle \mathcal{T}_{\simeq}, \delta_\Sigma, \gamma_\Sigma, \mathcal{F}_{\simeq} \rangle$. In this PA, the states 0 and 1 fill the roles of $\bot$ and $\top$ respectively.

We refer to δ_Σ (respectively γ_Σ) as the *sequential* (respectively *parallel*) *derivative* functions. The (strict) trace dependency relation of A_Σ is denoted $\preceq_\Sigma$ (respectively $\prec_\Sigma$). Similarly, the trace relation of A_Σ is denoted by $\to_\Sigma$, and we write $L_\Sigma(e)$ for the language of $e \in \mathcal{T}_{\simeq}$ in A_Σ.

We now claim that, first, A_Σ is finitely supported and well-nested, and that, second, for $e \in \mathcal{T}$ it holds that $L_\Sigma(e) = [\![e]\!]$. The following two sections are devoted to showing that both of these hold, respectively.

5.1. Structural properties

Let us start by arguing that the syntactic PA is finitely supported. To this end, we should show that for every $e \in \mathcal{T}$, the set $\pi_\Sigma(e)$ is finite; since this set is the smallest closed set (w.r.t. $\preceq_\Sigma$) containing e, it suffices to find a finite closed set containing e. To shorten the proof, however, it is useful to introduce the following, more general notion.

Definition 5.5. Let $E \subseteq \mathcal{T}$ and $e \in \mathcal{T}$. If there exist $e_0, \ldots, e_{n-1} \in E$ such that $e \simeq e_0 + \cdots + e_{n-1}$, we say that E *covers* e. Furthermore, we say that E is *cover-closed* if, whenever $f \in E$ and $g \preceq_\Sigma f$, it holds that E covers g.

Cover-closed sets then give us a way to show finite support, as follows.

Lemma 5.6. *Let $e \in \mathcal{T}$. If e is covered by a finite and cover-closed set, then e is contained in a finite and closed set (and hence $\pi_\Sigma(e)$ is finite).*

Showing finite support then comes down to finding a finite and cover-closed set for every expression, which can be done by induction on the expression.

Lemma 5.7. *The syntactic PA is finitely supported.*

To argue well-nestedness, we need need to show that $e \prec_\Sigma f$ for some (congruence classes of) spr-expressions. To this end, we will actually need to argue that $f \not\preceq_\Sigma e$; since it is hard to prove this directly from the inductive definition of $\preceq_\Sigma$, we introduce the following to argue $f \not\preceq_\Sigma e$ indirectly.

Definition 5.8. We define $d_\parallel : \mathcal{T} \to \mathbb{N}$ inductively, as follows:

$$\begin{aligned} d_\parallel(0) &= 0 & d_\parallel(e_0 \cdot e_1) &= \max(d_\parallel(e_0), d_\parallel(e_1)) \\ d_\parallel(1) &= 0 & d_\parallel(e_0 \parallel e_1) &= \max(d_\parallel(e_0), d_\parallel(e_1)) + 1 \\ d_\parallel(a) &= 0 & d_\parallel(e_0 + e_1) &= \max(d_\parallel(e_0), d_\parallel(e_1)) \\ d_\parallel(e_0^*) &= d_\parallel(e_0) & d_\parallel(e_0^\dagger) &= d_\parallel(e_0) \end{aligned}$$

We also define $d_\dagger : \mathcal{T} \to \mathbb{N}$ inductively, as follows:

$$\begin{aligned} d_\dagger(0) &= 0 & d_\dagger(e_0 \cdot e_1) &= \max(d_\dagger(e_0), d_\dagger(e_1)) \\ d_\dagger(1) &= 0 & d_\dagger(e_0 \parallel e_1) &= \max(d_\dagger(e_0), d_\dagger(e_1)) \\ d_\dagger(a) &= 0 & d_\dagger(e_0 + e_1) &= \max(d_\dagger(e_0), d_\dagger(e_1)) \\ d_\dagger(e_0^*) &= d_\dagger(e_0) & d_\dagger(e_0^\dagger) &= d_\dagger(e_0) + 1 \end{aligned}$$

A straightforward series of inductive proofs on the structure of spr-expressions then gives us the following:

Lemma 5.9. *If $e \preceq_\Sigma f$, then $d_\parallel(e) \leq d_\parallel(f)$ and $d_\dagger(e) \leq d_\dagger(f)$.*

Thus, if we want to show that $e \prec_\Sigma f$, it suffices to show that $e \preceq_\Sigma f$ and $d_\parallel(e) < d_\parallel(f)$ or $d_\dagger(e) < d_\dagger(f)$. With this in hand, we can show the following:

Lemma 5.10. *Let $e, g, h \in \mathcal{T}$ with $\gamma_\Sigma(e, g, h) \not\simeq 0$. Then $g \prec_\Sigma e$. Furthermore, either $h \prec_\Sigma e$ or there exists an $f \in \mathcal{T}$ such that $e \simeq f^\dagger$.*

Hence, we argue that all states A_Σ are sequential or recursive, as follows.

Lemma 5.11. *The syntactic PA is well-nested.*

Proof. Let $e \in \mathcal{T}$; by Lemma 5.10 we already know that for all $g, h \in \mathcal{T}$ such that $\gamma_\Sigma(e, g, h) \not\simeq 0$ it holds that $g \prec_\Sigma e$. If furthermore for all $g, h \in \mathcal{T}$ with $\gamma_\Sigma(e, g, h) \not\simeq 0$ it holds that $h \prec_\Sigma e$, then e is sequential. Otherwise, it follows by Lemma 5.10 that $e \simeq f^\dagger$ for some $f \in \mathcal{T}$.

We now claim that e is recursive. To see this, first note that for all $a \in \Sigma$ we have $\delta_\Sigma(e, a) \simeq \delta_\Sigma(f^\dagger, a) = 0$. Furthermore, if $g, h \in \mathcal{T}$ such that $\gamma_\Sigma(e, g, h) \simeq \gamma_\Sigma(f^\dagger, g, h) \not\simeq 0$, then $\gamma_\Sigma(e, g, h) \simeq 1$ and $g \simeq f$ and $h \simeq f^\dagger$ by definition of γ_Σ; hence $g \prec_\Sigma e$ and $h \simeq e$ by Lemma 5.9. □

5.2. Language equivalence

We now set out to prove that, for $e \in \mathcal{T}$, we have that $L_\Sigma(e) = [\![e]\!]$; to this end, we first need to discuss a number of auxiliary lemmas that help us analyse and reason about the traces in A_Σ.

For the inclusion of $L_\Sigma(e)$ in $[\![e]\!]$, it is useful to be able to take a trace labelled with some pomset and turn it into one or more traces labelled with parts of that pomset. We refer to such an action as a *deconstruction* of the starting trace. The first deconstruction lemma that we will consider concerns traces that originate in a state that represents a sum of spr-expressions.

Lemma 5.12. *Let $e_0, e_1 \in \mathcal{T}$, $f \in \mathcal{F}$ and $U \in \mathsf{Pom}^{\mathsf{sp}}$, such that $e_0 + e_1 \xrightarrow{U}_\Sigma f$ of length ℓ. There exists an $f' \in \mathcal{F}$ with $e_0 \xrightarrow{U}_\Sigma f'$ or $e_1 \xrightarrow{U}_\Sigma f'$ of length ℓ.*

Proof. The proof proceeds by induction ℓ. In the base, where $\ell = 0$, we have that $e_0 + e_1 \xrightarrow{U}_\Sigma f$ is a trivial trace. In that case, $f = e_0 + e_1$, and so $e_0 \in \mathcal{F}$ or $e_1 \in \mathcal{F}$; in the former case, choose $f' = e_0$, in the latter case, choose $f' = e_1$. In either case, the claim is satisfied.

For the inductive step, let $e_0 + e_1 \xrightarrow{U}_\Sigma f$ be of length $\ell + 1$, and assume that the claim holds for ℓ. We find that $U = V \cdot U'$ and a $g \in \mathcal{T}$ such that $e_0 + e_1 \xrightarrow{V}_\Sigma g$ is a unit trace, and $g \xrightarrow{U}_\Sigma f$ is of length ℓ. If, on the one hand, $e_0 + e_1 \xrightarrow{V}_\Sigma g$ is a δ-trace, then $V = a$ for some $a \in \Sigma$, and $g = \delta_\Sigma(e_0 + e_1, a) = \delta_\Sigma(e_0, a) + \delta_\Sigma(e_1, a)$. By induction, we then find $f' \in \mathcal{F}$ such that $\delta_\Sigma(e_0, a) \xrightarrow{U}_\Sigma f'$ or $\delta_\Sigma(e_1, a) \xrightarrow{U}_\Sigma f'$, of length ℓ. Putting this together, we have that $e_0 \xrightarrow{U}_\Sigma f'$ or $e_1 \xrightarrow{U}_\Sigma f'$, of length $\ell + 1$.

The case where $e_0 + e_1 \xrightarrow{V}_\Sigma g$ is a γ-trace is similar. □

The proofs of the other deconstruction lemmas follow a similar pattern; these appear in Appendix C.3.

Another deconstruction lemma arises when the starting state is a sequential composition. In this case, we find multiple traces: one originating in the left subterm, and another originating in the right subterm.

Lemma 5.13. *Let $e_0, e_1 \in \mathcal{T}$, $f \in \mathcal{F}$ and $U \in \mathsf{Pom}^{\mathsf{sp}}$, such that $e_0 \cdot e_1 \xrightarrow{U}_\Sigma f$ is of length ℓ. There exist $f_0, f_1 \in \mathcal{F}$ such that $U = U_0 \cdot U_1$, as well as $e_0 \xrightarrow{U_0}_\Sigma f_0$ of length ℓ_0 and $e_1 \xrightarrow{U_1}_\Sigma f_1$ of length ℓ_1, such that $\ell_0 + \ell_1 = \ell$.*

The last deconstruction lemma that we record concerns the Kleene star; here, we find a number of traces, each of which originates form the subterm below the Kleene star, and reaches an accepting state.

Lemma 5.14. *Let $e \in \mathcal{T}$ and $f \in \mathcal{F}$ and $U \in \mathsf{Pom}^{\mathsf{sp}}$ be such that $e^* \xrightarrow{U}_\Sigma f$. There exist $f_0, \ldots, f_{n-1} \in \mathcal{F}$ such that $U = U_0 \cdots U_{n-1}$ and for $0 \leq i < n$ it holds that $e \xrightarrow{U_i}_\Sigma f_i$.*

To show the other inclusion, i.e., that $[\![e]\!] \subseteq L_\Sigma(e)$, we need *construction* lemmas to compose traces of pomsets into a trace of a composition of those pomsets. To keep the lemmas concise, we need the following

Definition 5.15. We write $\lesssim$ for the smallest relation on $\mathcal{T}$ such that $e \lesssim f$ when $e + f \simeq f$; note that this makes $\lesssim$ a preorder on $\mathcal{T}$.

The first construction lemma that we encounter allows us to use $+$ to add additional terms to the starting trace, such that the target state of the new trace contains the old target state.

Lemma 5.16. *Let $e_0, e_1, f_0 \in \mathcal{T}$ and $U \in \mathsf{Pom}^{\mathsf{sp}}$ be such that $e_0 \xrightarrow{U}_\Sigma f_0$. There exists an $f \in \mathcal{T}$ such that $e_0 + e_1 \xrightarrow{U}_\Sigma f$ and $f_0 \lesssim f$.*

Proof. The proof proceeds by induction on the length ℓ of $e_0 \xrightarrow{U}_\Sigma f_0$. In the base, where $\ell = 0$, we have $f_0 = e_0$ and $U = 1$. We then choose $f = e_0 + e_1$.

For the inductive step, let $e_0 \xrightarrow{U}_\Sigma f_0$ be of length $\ell + 1$, and assume that the claim holds for ℓ. We then find $e_0' \in \mathcal{T}$ and $U = V \cdot U'$ such that $e_0 \xrightarrow{V}_\Sigma e_0'$ is a unit trace, and $e_0' \xrightarrow{U'}_\Sigma f_0$ is of length ℓ. If $e_0 \xrightarrow{V}_\Sigma e_0'$ is a δ-trace, then $V = a$ for some $a \in \Sigma$, and $e_0' = \delta_\Sigma(e_0, a)$. We choose $e_1' = \delta_\Sigma(e_1, a)$; by induction, we find $f \in \mathcal{T}$ such that $f_0 \lesssim f$ and $e_0' + e_1' \xrightarrow{U'}_\Sigma f$. Since $\delta_\Sigma(e_0 + e_1, a) = \delta_\Sigma(e_0, a) + \delta_\Sigma(e_1, a) = e_0' + e_1'$, we have that $e_0 + e_1 \xrightarrow{V}_\Sigma e_0' + e_1'$. Putting this together, we find that $e_0 + e_1 \xrightarrow{U}_\Sigma f$.

The case where $e_0 \xrightarrow{V}_\Sigma e_0'$ is a γ-trace is similar. □

Like deconstruction lemmas, the proofs of construction lemmas follow a similar pattern. Further proofs of lemmas like this appear in Appendix C.4.

The construction lemma for sequential composition consists of two parts. First, we need to be able to append an expression, in such a way that the appended expression is carried into the target state of the new trace.

Lemma 5.17. *Let $e_0, e_1 \in \mathcal{T}$ and $f_0 \in \mathcal{F}$ and $U \in \mathsf{Pom}^{\mathsf{sp}}$ be such that $e_0 \xrightarrow{U}_\Sigma f_0$. Then there exists an $f \in \mathcal{T}$ such that $f_0 \cdot e_1 \lesssim f$, and $e_0 \cdot e_1 \xrightarrow{U}_\Sigma f$.*

Second, we need to be able to prepend an expression in $\mathcal{F}$ to get a new trace with a target state that contains the old target state. The intuition here is that the constructed trace simply disregards the prepended expression (which is possible because it is in $\mathcal{F}$) and continues by imitating the old trace.

Lemma 5.18. *Let $e_0 \in \mathcal{T}$ and $f_0, f_1 \in \mathcal{F}$ and $V \in \mathsf{Pom}^{\mathsf{sp}}$ be such that $e_1 \xrightarrow{V}_\Sigma f_1$. There exists an $f \in \mathcal{F}$ such that $f_0 \cdot e_1 \xrightarrow{V}_\Sigma f$.*

The construction lemma for sequential composition is then a simple consequence of the preceding construction lemmas.

Lemma 5.19. *Let $e_0, e_1 \in \mathcal{T}$, $f_0, f_1 \in \mathcal{F}$ and $U, V \in \mathsf{Pom}^{\mathsf{sp}}$ such that $e_0 \xrightarrow{U}_\Sigma f_0$ and $e_1 \xrightarrow{V}_\Sigma f_1$. There exists an $f \in \mathcal{F}$ with $e_0 \cdot e_1 \xrightarrow{U \cdot V}_\Sigma f$.*

The final construction lemma shows how to construct a trace originating in a state of the form e^*, given a number of traces that originate in e. The intuition here is that the constructed trace mimics the traces that originate in e, while carrying a factor e^* to restart the next trace.

Lemma 5.20. *Let $e \in \mathcal{T}$ and $f_0, \ldots, f_{n-1} \in \mathcal{F}$ and $U_0, \ldots, U_{n-1} \in \mathsf{Pom}^{\mathsf{sp}}$ be such that for $0 \leq i < n$ it holds that $e \xrightarrow{U_i}_\Sigma f_i$. There exists an $f \in \mathcal{F}$ such that $e^* \xrightarrow{U_0 \cdots U_{n-1}}_\Sigma f$.*

With all of these facts about constructing and deconstructing traces in the syntactic PA, we are finally able to show correctness of our translation from expressions to automata, as witnessed by the following lemma.

Lemma 5.21. *If $e \in \mathcal{T}$, then $L_\Sigma(e) = [\![e]\!]$.*

The equality follows from using the deconstruction lemmas (for the inclusion from left to right) and the construction lemmas (to show the inclusion from right to left); as before, a full proof can be found in Appendix C.

This establishes the main result of this section.

Theorem 5.22. *For $e \in \mathcal{T}$, we can find a well-nested and finitely supported PA A that accepts $[\![e]\!]$, i.e., A has a state q_e such that $L_A(q_e) = [\![e]\!]$.*

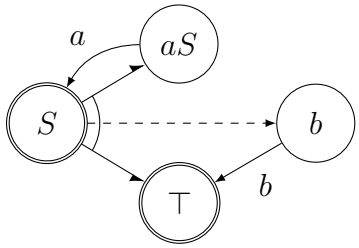


Figure 4: The PA A_2, recognising $a^n b^n$

6. Context-free pomset languages

In this section, we characterise the class of languages accepted by finite PA, with no restrictions. These turn out to be languages of pomsets generated by finite context-free grammars [11] using series-parallel terms. A pomset automaton whose language is not rational is displayed in Figure 4.

A *context-free pomset grammar* G (*CFG* in the following) is given by a triple $\langle \Gamma, S, R \rangle$, where Γ is a finite set of non-terminals, $S \in \Gamma$ is a distinguished start symbol, and R is a finite set of production rules, i.e., pairs of a non-terminal and a term built out of sequential products, parallel products, and symbols chosen from $\Gamma \cup \Sigma \cup \{\epsilon\}$. Using the production rules as usual starting from the symbol S, we define the pomset language $[\![G]\!]$ generated by a CFG G. A pomset language is called *context-free* (*CF*) if it is generated by some CFG.

Theorem 6.1. *A pomset language is accepted by a PA if and only if it is CF.*

Proof. The automaton to grammar direction is straightforward. Given a finite PA $A = \langle Q, \delta, \gamma, F \rangle$ and $q_0 \in Q$, we will build a CFG G_{A,q_0} with non-terminals Q and start symbol q_0. For every state q and letter $a \in \Sigma$ we produce a rule $q \to a \cdot \delta(q, a)$; we add for every triple of states (q, r, s) a production $q \to (r \parallel s) \cdot \gamma(q, r, s)$; finally for every accepting state $f \in F$ we add a rule $f \to \epsilon$. The fact that $[\![G_{A,q_0}]\!] = L_A(q_0)$ is straightforward from this definition: clearly there is an isomorphism between the accepting runs of A starting from q_0 and the derivations in G_{A,q_0}.

We now construct an automaton from a CFG $G = \langle \Gamma, S, R \rangle$ to recognise $[\![G]\!]$. Let $\mathcal{T}$ be the set of subterms of the right-hand sides of rules in R, s, t will range over $\mathcal{T}$ in the following; this set is clearly finite. We define a PA

$A_G = \langle \Gamma \cup \mathcal{T} \cup \{\top, \bot\}, \delta, \gamma, \{\top, \epsilon\}\rangle$, such that $L_A(S) = [\![G]\!]$ where:

$$\delta(a, a) := \top; \qquad \gamma(s \parallel t, s, t) := \top;$$

$$\gamma(s \cdot t, s, \top) := t; \qquad \gamma(X, s, \top) := \begin{cases} \top & \text{if } X \to s \in R \\ \bot & \text{otherwise} \end{cases}$$

We complete δ and γ into functions by assigning every undefined value to $\bot$. There is a straightforward correspondence between runs in A_G and derivations from G, therefore they are language equivalent. □

As usual when in the presence of context-free languages [16, 17], we obtain a host of undecidability results for PA with no restriction other than finiteness; we call out two important ones below.

Corollary 6.2. *Let A be a PA. The following are undecidable:*

(i) Given states q and q' of A, does $L_A(q) = L_A(q')$ hold?

(ii) Given a state q of A, is $L_A(q)$ an spr-language?

The second undecidability result justifies our "well-nestedness" condition for the automata-to-expression direction of our Kleene theorem, since one needs strict restrictions on PA to guarantee the rationality of its language.

7. Related work

If a PA is fork-acyclic in the sense of [1], then it is well-nested; thus, finitely supported and well-nested PAs are a superset of the PAs considered in [1]. Relaxing fork-acyclicity to well-nestedness is necessary for finitely supported pomset automata to capture spr-expressions that contain †.

Lodaya and Weil proposed another automaton formalism for pomsets, called *branching automata* [6]. These define the states where parallelism can start (*fork*) or end (*join*) in two relations; pomset automata condense this information in a single function. In op. cit., we also find a translation of spr-expressions to branching automata, based on Thompson's construction [9], which relies on the fact that transitions of branching automata are encoded as *relations*. Our Brzozowski-style [8] translation, in contrast, directly constructs transition *functions* from the expressions. Lastly, translation of branching automata to series-parallel expressions in [6] is sound only for a *semantically* restricted class of automata, whereas our restriction is *structural*.

Jipsen and Moshier [7] provided an alternative formulation of the automata proposed by Lodaya and Weil, also called *branching automata*. Their method to encode parallelism is conceptually dual to pomset automata: branching automata distinguish based on the target states of traces to determine the join state, whereas pomset automata distinguish based on the origin states of traces. The translations of series-parallel expressions to branching automata and vice versa suffer from the same shortcomings as those by Lodaya and Weil, i.e., transition relations rather than functions and a semantic restriction on automata for the translation of automata to expressions.

Lodaya and Weil observed [6] that the behaviour of a fragment of their branching automata corresponds to 1-safe Petri nets. This fragment can be matched with a fragment of pomset automata (discussed in [1]). We opted to treat semantics of spr-expressions in terms of automata instead of Petri nets to find more opportunities to extend to a coalgebraic treatment. The present paper does not reach this goal, but we believe that our formulation in terms of states and transition functions offers some hope of getting there.

Also related are *parenthesising automata* as proposed by Ésik and Németh in [14], which recognise *series-parallel n-posets*, a generalisation of words where events are partially ordered by n partial orders. Like sp-pomsets, series-parallel 2-posets can be composed using two associative operators, but unlike sp-pomsets, the second "parallel" composition operator is not commutative. This does not rule out a connection to programs with parallelism, but it does require specialisation of the model. On the other hand, parenthesising automata have an advantage over pomset automata in that they are pleasingly symmetric in how composition operators are treated, which simplifies a lot of proofs. The correspondence between automata and expressions described in op. cit. also requires restricting the class of automata. Unlike our work, however, this restriction tightly characterises the automata for which the translation is possible, and is furthermore decidable. Since parenthesising automata cannot, in general, recognise context-free languages, this does not contradict our earlier remarks about decidability of such a property.

Prisacariu introduced *Synchronous Kleene Algebra* (SKA) [18], extending Kleene Algebra with a *synchronous composition* operator. SKA differs from our model in that it assumes that all basic actions are performed in unit time, and that actors responsible for individual actions never idle. In contrast, our (BKA-like) model makes no synchrony assumptions: expressions can be composed in parallel, and the relative timing of basic actions within those expressions is irrelevant for the semantics. Prisacariu axiomatised SKA

and extended it to *Synchronous Kleene Algebra with Tests* (SKAT); others proposed Brzozowski-style derivatives of SKA- and SKAT-expressions [19].

8. Further work

Language equivalence of rational expressions can be axiomatised using Kleene's original theorem [20].[2] More precisely, the proof in op. cit. relies on encoding a minimised finite automaton for a rational expression back into a rational expression (using both directions of Kleene's theorem) to obtain an equivalent canonical representation. We hope to apply the work put forward in the present paper to axiomatise spr-expressions in the same fashion. In particular, the correspondence of expressions to states and the structural nature of well-nestedness may prove useful in validating such a canonicalisation. For this technique to work, one would need to devise a canonical form for PAs, analogous to the minimal finite automaton.

A different result axiomatises equivalence of sr-expressions (i.e., spr-expressions without the parallel star) with respect to the downward-closed pomset semantics [21, 22]. The algorithm in [22] for constructing the downward closure of an sr-expression is particularly relevant as it can be used to extend the direction from expressions to automata of our Kleene theorem. More precisely, it establishes pomset automata as an operational model for *weak CKA*, that is, BKA without the parallel star † but with the exchange law. Extending the result even further to spr-expressions is not possible with the methodology used in [22] or [21], see the conclusions of [22].

Brzozowski derivatives for classic rational expressions induce a coalgebra on rational expressions that corresponds to a finite automaton. We aim to study spr-expressions coalgebraically. The first step would be to find the coalgebraic analogue of pomset automata such that language acceptance is characterised by the homomorphism into the final coalgebra. Ideally, such a view of pomset automata would give rise to a decision procedure for equivalence of spr-expressions based on coalgebraic bisimulation-up-to [23].

Rational expressions can be extended with tests to reason about imperative programs equationally [4]. In the same vein, one can extend sr-expressions with tests [24, 7] to reason about parallel imperative programs equationally. We are particularly interested in employing such an extension to extend the

[2] A similar result exists for spr-expressions [10], but this does not rely on Kleene's theorem for canonicalisation.

network specification language NetKAT [25] with primitives for concurrency so as to model and reason about concurrency within networks.

[1] T. Kappé, P. Brunet, B. Luttik, A. Silva, F. Zanasi, Brzozowski goes concurrent — a Kleene theorem for pomset languages, in: Proc. Concurrency Theory (CONCUR), 2017, pp. 25:1–25:16. doi:10.4230/LIPIcs.CONCUR.2017.25.

[2] T. Hoare, B. Möller, G. Struth, I. Wehrman, Concurrent Kleene algebra, in: Proc. Concurrency Theory (CONCUR), 2009, pp. 399–414. doi:10.1007/978-3-642-04081-8_27.

[3] S. C. Kleene, Representation of events in nerve nets and finite automata, Automata Studies (1956) 3–41.

[4] D. Kozen, Kleene algebra with tests, ACM Trans. Program. Lang. Syst. 19 (3) (1997) 427–443. doi:10.1145/256167.256195.

[5] J. E. Hopcroft, R. M. Karp, A linear algorithm for testing equivalence of finite automata, Tech. Rep. TR71-114 (December 1971).

[6] K. Lodaya, P. Weil, Series-parallel languages and the bounded-width property, Theoretical Computer Science 237 (1) (2000) 347–380. doi:10.1016/S0304-3975(00)00031-1.

[7] P. Jipsen, M. A. Moshier, Concurrent Kleene algebra with tests and branching automata, J. Log. Algebr. Meth. Program. 85 (4) (2016) 637–652. doi:10.1016/j.jlamp.2015.12.005.

[8] J. A. Brzozowski, Derivatives of regular expressions, J. ACM 11 (4) (1964) 481–494. doi:10.1145/321239.321249.

[9] K. Thompson, Regular expression search algorithm, Commun. ACM 11 (6) (1968) 419–422. doi:10.1145/363347.363387.

[10] M. R. Laurence, G. Struth, Completeness theorems for bi-Kleene algebras and series-parallel rational pomset languages, in: Proc. Relational and Algebraic Methods in Computer Science (RAMiCS), 2014, pp. 65–82. doi:10.1007/978-3-319-06251-8_5.

[11] N. Chomsky, Three models for the description of language, IRE Transactions on Information Theory 2 (3) (1956) 113–124. doi:10.1109/TIT.1956.1056813.

[12] J. L. Gischer, The equational theory of pomsets, Theor. Comput. Sci. 61 (1988) 199–224. doi:10.1016/0304-3975(88)90124-7.

[13] J. Grabowski, On partial languages, Fundam. Inform. 4 (2) (1981) 427.

[14] Z. Ésik, Z. L. Németh, Higher dimensional automata, Journal of Automata, Languages and Combinatorics 9 (1) (2004) 3–29.

[15] R. McNaughton, H. Yamada, Regular expressions and state graphs for automata, IRE Trans. Electronic Computers 9 (1) (1960) 39–47. doi:10.1109/TEC.1960.5221603.

[16] Y. Bar-Hillel, M. Perles, E. Shamir, On formal properties of simple phrase structure grammars, Sprachtypologie und Universalienforschung 14 (1961) 143–172.

[17] S. Greibach, A note on undecidable properties of formal languages, Mathematical systems theory 2 (1) (1968) 1–6. doi:10.1007/BF01691341.

[18] C. Prisacariu, Synchronous Kleene algebra, J. Log. Algebr. Program. 79 (7) (2010) 608–635. doi:10.1016/j.jlap.2010.07.009.

[19] S. Broda, S. Cavadas, M. Ferreira, N. Moreira, Deciding synchronous Kleene algebra with derivatives, in: Proc. Implementation and Application of Automata (CIAA), 2015, pp. 49–62. doi:10.1007/978-3-319-22360-5_5.

[20] D. Kozen, A completeness theorem for Kleene algebras and the algebra of regular events, Inf. Comput. 110 (2) (1994) 366–390. doi:10.1006/inco.1994.1037.

[21] M. R. Laurence, G. Struth, Completeness theorems for pomset languages and concurrent Kleene algebras. arXiv:abs/1705.05896.

[22] T. Kappé, P. Brunet, A. Silva, F. Zanasi, Concurrent Kleene algebra: Free model and completeness, in: Proc. European Symp. on Programming (ESOP), 2018, pp. 856–882. doi:10.1007/978-3-319-89884-1_30.

[23] J. Rot, M. M. Bonsangue, J. J. M. M. Rutten, Coalgebraic bisimulation-up-to, in: Proc. Current Trends in Theory and Practice of Comp. Sci. (SOFSEM), 2013, pp. 369–381. doi:10.1007/978-3-642-35843-2_32.

[24] P. Jipsen, Concurrent Kleene algebra with tests, in: Proc. Relational and Algebraic Methods in Computer Science (RAMiCS) 2014, 2014, pp. 37–48. doi:10.1007/978-3-319-06251-8_3.

[25] C. J. Anderson, N. Foster, A. Guha, J. Jeannin, D. Kozen, C. Schlesinger, D. Walker, NetKAT: semantic foundations for networks, in: Proc. Principles of Programming Languages (POPL), 2014, pp. 113–126. doi:10.1145/2535838.2535862.

Appendix A. Proofs about pomset automata

Lemma 3.3. *Let $q \xrightarrow{U}_A q'$ be non-trivial. If $q = \bot$ or $q = \top$, then $q' = \bot$.*

Proof. The proof proceeds by induction on the construction of $\to_A$. In the base, $U = a$ for some $a \in \Sigma$, and $q' = \delta(q, a)$. It follows that $q' = \bot$.

For the inductive step, there are two cases to consider.

- If $q \xrightarrow{U}_A q'$ because $U = V \cdot W$ and there exists a $q'' \in Q$ such that $q \xrightarrow{V}_A q''$ and $q'' \xrightarrow{W}_A q'$, then we can assume without loss of generality that at least one of these traces is non-trivial — if this were not the case, then $q = q'' = q$ and $V = W = 1$, meaning that $q = q'$ and $U = 1$, and so $q \xrightarrow{U}_A q'$ would be trivial as well.

  If, on the one hand, $q \xrightarrow{V}_A q''$ is non-trivial, then $q'' = \bot$ by induction. In that case, $q' = \bot$, regardless of whether $q'' \xrightarrow{W}_A q'$ is trivial. If, on the other hand, $q'' \xrightarrow{W}_A q'$ is non-trivial, then $q' = \bot$, also by induction.

- If $q \xrightarrow{U}_A q'$ is a γ-trace, i.e., $U = V \parallel W$ and there exist $r, s \in Q$ as well as $r', s' \in Q$ such that $r \xrightarrow{V}_A r'$ and $s \xrightarrow{W}_A s'$ and $\gamma(q, r, s) = q'$, then $q' = \bot$ immediately. □

Lemma 3.6. *If A is finitely supported, then for every $q \in Q$ there exists a finite pomset automaton A_q with a state q', such that $L_A(q) = L_{A_q}(q')$.*

Proof. We define A_q as the automaton $\langle \pi_A(q), \delta, \gamma, F \cap \pi_A(q) \rangle$. Here, δ and γ are well-defined as functions of type $\pi_A(q) \times \Sigma \to \pi_A(q)$ and $\pi_A(q)^3 \to \pi_A(q)$ respectively, by definition of $\pi_A(q)$. Furthermore, since A is finitely supported, we know that A_q has finitely many states. It remains to show that $L_{A_q}(q) = L_A(q)$.

For the inclusion from left to right, we prove the more general claim that if $r \xrightarrow{U}_{A_q} r'$ with $r' \neq \bot$, then $r \xrightarrow{U}_A r'$. The proof proceeds by induction on the construction of $\to_{A_q}$. In the base, there are two cases to consider.

- If $U = 1$ and $r = r'$, then $r \xrightarrow{U}_A r'$ immediately.
- If $U = a$ for some $a \in \Sigma$ and $r' = \delta(r, a)$, then it follows that $r \xrightarrow{U}_A r'$.

For the inductive step, there are two cases to consider.

- Suppose that $r \xrightarrow{U}_{A_q} r'$ because $U = V \cdot W$ and there exists an $r'' \in \pi_A(q)$ with $r \xrightarrow{V}_{A_q} r''$ and $r'' \xrightarrow{W}_{A_q} r'$. Since $r' \neq \bot$, also $r'' \neq \bot$ by Lemma 3.3. By induction, we find that $r \xrightarrow{V}_A r'' \xrightarrow{W}_A r'$, and thus $r \xrightarrow{U}_A r'$.

- Suppose that $r \xrightarrow{U}_{A_q} r'$ because $U = V \parallel W$ and there exist $s, t \in \pi_A(q)$ and $s', t' \in \pi_A(q) \cap F$ such that $s \xrightarrow{V}_{A_q} s'$ and $t \xrightarrow{W}_{A_q} t'$ and $\gamma(r, s, t) = r'$. First, note that $s', t' \neq \bot$, since $s', t' \in F$. By induction, we find that $s \xrightarrow{V}_{A} s' \in F$ and $t \xrightarrow{W}_{A} t' \in F$. We can then conclude that $q \xrightarrow{U}_{A} q'$.

For the other inclusion, we prove the more general claim that if $r \in \pi_A(q)$ and $r \xrightarrow{U}_{A} r'$ with $r' \neq \bot$, then $r' \in \pi_A(q)$ and $r \xrightarrow{U}_{A_q} r'$. We proceed by induction on the construction of $\to_A$. In the base, there are two cases to consider.

- If $U = 1$ and $r = r'$, then $r' \in \pi_A(q)$ and $r \xrightarrow{U}_{A_q} r'$ immediately.
- If $U = a$ for some $a \in \Sigma$, and $r' = \delta(r, a)$, then note that $r' \in \pi_A(q)$ by definition of $\pi_A(q)$. Furthermore, we find that $r \xrightarrow{U}_{A_q} r'$.

For the inductive step, there are two cases to consider.

- Suppose $r \xrightarrow{U}_{A} r'$ because $U = V \cdot W$, and there exists an $r'' \in Q$ such that $r \xrightarrow{V}_{A} r''$ and $r'' \xrightarrow{W}_{A} r'$. By induction, we then find that $r'' \in \pi_A(q)$ and $r \xrightarrow{V}_{A_q} r''$. Again by induction, we also find that $r' \in \pi_A(q)$ and $r'' \xrightarrow{W}_{A_q} r'$. We then conclude that $r \xrightarrow{U}_{A_q} r'$.
- Suppose $r \xrightarrow{U}_{A} r'$ because $U = V \parallel W$, and there exist $s, t \in Q$ and $s', t' \in F$ such that $s \xrightarrow{V}_{A} s'$ and $t \xrightarrow{W}_{A} t'$ and $\gamma(r, s, t) = r'$. By the premise that $r' \neq \bot$ we have that $s, t \preceq_A q$, and thus $s, t \in \pi_A(q)$. By induction, we then find that $s', t' \in \pi_A(q)$, and $s \xrightarrow{V}_{A_q} s'$ as well as $t \xrightarrow{W}_{A_q} t'$. We can then conclude that $r \xrightarrow{U}_{A_q} r'$. □

Lemma 3.7. *If A is finitely supported, then $\prec_A$ is well-founded.*

Proof. Suppose, towards a contradiction, that $\{q_n\}_{n \in \mathbb{N}} \subseteq Q$ is such that for $n \in \mathbb{N}$ it holds that $q_{n+1} \prec_A q_n$. Since $\{q_n\}_{n \in \mathbb{N}} \subseteq \pi_A(q_0)$ and the latter is finite, it follows that $q_n = q_m$ for some $n > m$. But then we find that, $q_n \prec_A q_m$, which contradicts that $\prec_A$ is a strict order, and therefore irreflexive. □

Lemma 3.8. *If $q \xrightarrow{U}_{A} q'$, then there exist $q_0, \ldots, q_\ell \in Q$ with $q = q_0$ and $q_\ell = q'$, and $U = U_0 \cdots U_{\ell-1}$ such that for $0 \leq i < \ell$ it holds that $q_i \xrightarrow{U_i}_{A} q_{i+1}$. Furthermore, each of these traces is a unit trace.*

Proof. The proof proceeds by induction on the construction of $q \xrightarrow{U}_{A} q'$. In the base, there are two cases to consider.

- If $q \xrightarrow{U}_A q'$ is a trivial trace, then the claim holds immediately; simply choose $\ell = 0$ and $q_0 = q = q'$.
- If $q \xrightarrow{U}_A q'$ is a δ-trace, i.e., $U = a$ for some $a \in \Sigma$ and $q' = \delta(q, a)$, then choose $\ell = 1$ and $U_0 = U = a$ to satisfy the claim.

In the inductive step, there are again two cases to consider.

- Suppose that $q \xrightarrow{U}_A q'$ because $U = V \cdot W$ and there exists a $q'' \in Q$ such that $q \xrightarrow{V}_A q''$ and $q'' \xrightarrow{W}_A q'$. By induction, we find $q_0, \ldots, q_n \in Q$ with $q_0 = q$ and $q_n = q''$, and $V = V_0 \cdots V_{n-1}$ such that for $0 \leq i < \ell'$ it holds that $q_i \xrightarrow{V_i}_A q_{i+1}$. Also by induction, we find $q'_0, \ldots, q'_m$ with $q'_0 = q''$ and $q'_m = q'$, and $W = W_0 \cdots W_{m-1}$ such that for $0 \leq i < \ell''$ it holds that $q'_i \xrightarrow{W_i}_A q'_{i+1}$. We then choose for $0 \leq i < \ell'$ that $U_i = V_i$, and for $\ell' \leq i < \ell' + \ell''$ that $q_{i+\ell'} = q'_i$ and $U_{i+\ell'} = W_i$ to satisfy the claim.
- Suppose that $q \xrightarrow{U}_A q'$ is a γ-trace, i.e., $U = V \parallel W$ and there exist $r, s \in Q$ and $r', s' \in F$ such that $r \xrightarrow{V}_A r'$ and $s \xrightarrow{W}_A s'$ and $\gamma(q, r, s) = q'$. In that case, we can choose $n = 1$ and $U_0 = U = V \parallel W$ to satisfy the claim. □

Appendix B. Automata to expressions

Lemma 4.5. *If $q \in Q_{\mathsf{rec}}$, then*

$$L_A(q) = \left(\bigcup\nolimits_{\gamma(q,r,q)=\top} L_A(r) \right)^\dagger$$

Proof. For brevity, we write L' for the right-hand side of the claimed equality.

For the inclusion from left to right, we prove the more general claim that if $q \xrightarrow{U}_A q'$ for some $q' \in F$, then $U \in L'$ and furthermore $q' \in \{\top, q\}$. The proof proceeds by induction on the construction of $\to_A$. In the base, there are two cases to consider.

- If $q \xrightarrow{U}_A q'$ because $U = 1$ and $q = q'$, then the claim follows immediately.
- If $q \xrightarrow{U}_A q'$ because $U = a$ for some $a \in \Sigma$ and $q' = \delta(q, a)$, then $q' = \bot$ by the premise that q is recursive. Therefore, we can disregard this case, because it contradicts the premise that $q' \in F$.

For the inductive step, there are again two cases to consider.

- If $q \xrightarrow{U}_A q'$ because $U = V \cdot W$ and there exists a $q'' \in Q$ with $q \xrightarrow{V}_A q''$ and $q'' \xrightarrow{W}_A q'$, then there are two subcases to consider.
  - If $q \xrightarrow{V}_A q''$ is trivial, then the claim follows by applying the induction hypothesis to $q = q'' \xrightarrow{W}_A q'$, noting that $U = V \cdot W = W$.
  - If $q \xrightarrow{V}_A q''$ is non-trivial, then $q'' \in \{\bot, \top\}$ by the premise that q is recursive. Since $q' \in F$, it then follows that $q'' = \bot$ and $q'' \xrightarrow{W}_A q'$ is trivial, by Lemma 3.3. The claim then follows by applying the induction hypothesis to $q \xrightarrow{V}_A q''$, since $U = V \cdot W = V$.
- If $q \xrightarrow{U}_A q'$ because $U = V \parallel W$ and there exist $r, s \in Q$ and $r', s' \in F$ such that $r \xrightarrow{V}_A r'$ and $s \xrightarrow{W}_A s'$ and $\gamma(q, r, s) = q'$, then, since $q' \neq \bot$, it follows that $q' = \top$ and $s = q$ by the premise that q is recursive. By induction, we then know that $W \in L'$. Furthermore, $V \in L_A(r)$ by definition. Consequently, $U = V \parallel W \in L_A(r) \parallel L' \subseteq L'$.

For the inclusion from right to left, let $U \in L'$. Then $U = U_0 \parallel \cdots \parallel U_{n-1}$ and for $0 \leq i < n$ there exists an $r_i \in Q$ such that $\gamma(q, r_i, q) = \top$ and $U_i \in L_A(r_i)$. We need to prove that $U \in L_A(q)$, which we do by induction on n. In the base, where $n = 0$, we have that $U = 1$, and thus $U \in L_A(q)$ immediately, for $q \in F$. For the inductive step, assume that $n > 0$ and that the claim holds for $n - 1$; then $U' = U_1 \parallel \cdots \parallel U_{n-1} \in L_A(q)$ by induction. We thus find a $q' \in F$ such that $q \xrightarrow{U'}_A q'$. Furthermore, since $U_0 \in L_A(r_0)$, we find an $r_0' \in F$ such that $r_0 \xrightarrow{U_0} r_0'$. We then know that $q \xrightarrow{U_0 \| U'} \gamma(q, r_0, q) = \top$, and hence $U = U_0' \parallel U' \in L_A(q)$. □

Lemma 4.7. *Let $S \subseteq Q_{\mathsf{seq}}$, and suppose that for all $s \in S$, the induction hypothesis holds. Let $q \in S$ and $q' \in Q$. Then $U \in [\![e^S_{qq'}]\!]$ if and only if $q' \neq \bot$ and there exist $q_0, \ldots, q_{\ell-1} \in S$, and $U = U_0 \cdots U_{\ell-1}$ with*

$$q = q_0 \xrightarrow{U_0}_A q_1 \xrightarrow{U_1}_A \cdots \xrightarrow{U_{\ell-2}}_A q_{\ell-1} \xrightarrow{U_{\ell-1}}_A q_\ell = q'$$

and, furthermore, for $0 \leq i < \ell$ we have that $q_i \xrightarrow{U_i}_A q_{i+1}$ is a unit trace.

Proof. For the direction from left to right, first note that if $q' = \bot$, then $e^S_{qq'} = 0$, meaning $[\![e^S_{qq'}]\!] = \emptyset$; consequently, $q' \neq \bot$. For the remainder, we proceed by induction on S. In the base, where $S = \emptyset$, we have three cases.

- If $U = 1$ and $q = q'$, then we can choose $\ell = 0$ to satisfy the claim.

- If $U = a$ for $a \in \Sigma$ with $\delta(q, a) = q'$, then we choose $\ell = 1$ and $U_0 = a$.
- If $U = V \parallel W$ and $V \in [\![e_r]\!]$ and $W \in [\![e_s]\!]$ with $\gamma(q, r, s) = q'$, then $r, s \prec_A q$. By induction, $V \in L_A(r)$ and $W \in L_A(s)$; therefore, there exist $r', s' \in F$ such that $r \xrightarrow{V}_A r'$ and $s \xrightarrow{W}_A s'$. We then again choose $\ell = 1$ and $U_0 = U = V \parallel W$ to find that $q \xrightarrow{U}_A q'$.

For the inductive step, let $S = S' \cup \{q''\}$, and assume the claim holds for S'. There are two cases to consider.

- If $U \in [\![e^{S'}_{qq'}]\!]$, then the claim follows by induction.
- If $U \in [\![e^{S'}_{qq''} \cdot \left(e^{S'}_{q''q''}\right)^* \cdot e^{S'}_{q''q'}]\!]$, then $U = V \cdot W_0 \cdots W_{m-1} \cdot X$ with

  $$V \in [\![e^{S'}_{qq''}]\!] \qquad W_0 \in [\![e^{S'}_{q''q''}]\!] \qquad \cdots \qquad W_{m-1} \in [\![e^{S'}_{q''q''}]\!] \qquad X \in [\![e^{S'}_{q''q'}]\!]$$

  It should be obvious how to construct the desired trace.

For the other direction, first note that since $q' \neq \bot$, we know by Lemma 3.3 that $q_0, \ldots, q_{\ell-1} \neq \bot$. The proof proceeds by induction on ℓ. In the base, where $\ell \leq 1$, there are three cases to consider.

- If $q \xrightarrow{U} q'$ is trivial, then $U = 1$ and $q = q'$; thus $U \in [\![e^{\emptyset}_{qq'}]\!] \subseteq [\![e^{S}_{qq'}]\!]$.
- If $q \xrightarrow{U} q'$ is a δ-trace, then $U = a$ for some $a \in \Sigma$ and $\delta(q, a) = q'$. We find that $U = a \in [\![e^{\emptyset}_{qq'}]\!] \subseteq [\![e^{S}_{qq'}]\!]$.
- If $q \xrightarrow{U} q'$ is a γ-trace, then $U = V \parallel W$ with $r, s \in Q$ and $r', s' \in F$ such that $r \xrightarrow{V}_A r'$ and $s \xrightarrow{W}_A s'$ and $\gamma(q, r, s) = q'$, then $r, s \prec_A q$. By induction we have $V \in [\![e_r]\!]$ and $W \in [\![e_s]\!]$. Therefore,

  $$U = V \parallel W \in [\![e_r \parallel e_s]\!] \subseteq [\![e^{\emptyset}_{qq'}]\!] \subseteq [\![e^{S}_{qq'}]\!]$$

For the inductive step, assume that $\ell > 1$; in that case, it must be that $S \neq \emptyset$. We write $S = S' \cup \{q''\}$ and $I = \{0 \leq i < \ell : q_i = q''\}$. If $I = \emptyset$, then $U \in [\![e^{S'}_{qq'}]\!]$ by induction. Since $[\![e^{S'}_{qq'}]\!] \subseteq [\![e^{S}_{qq'}]\!]$, the claim follows. Otherwise, if $I \neq \emptyset$, then write $I = \{i_0, \ldots, i_{k-1}\}$ with $i_0 < i_1 < \cdots < i_{k-1}$. Then, by induction, we know that for $1 \leq j < k$ it holds that

$$U_{i_j} \cdot U_{i_j+1} \cdots U_{i_{j+1}-1} \in [\![e^{S'}_{q''q''}]\!]$$

Moreover, also by induction, we know that

$$U_0 \cdots U_{i_1-1} \in [\![e^{S'}_{qq''}]\!] \qquad\qquad U_{i_{k-1}} \cdot U_{i_{k-1}+1} \cdots U_\ell \in [\![e^{S'}_{q''q'}]\!]$$

Putting this together, we have

$$U \in \left[\!\!\left[e^{S'}_{qq''} \cdot \left(e^{S'}_{q''q''} \right)^* \cdot e^{S'}_{q''q'} \right]\!\!\right] \subseteq [\![e^{S}_{qq'}]\!] \qquad \square$$

Appendix C. Expressions to automata

Appendix C.1. Finite support

Lemma 5.6. *Let $e \in \mathcal{T}$. If e is covered by a finite and cover-closed set, then e is contained in a finite and closed set (and hence $\pi_\Sigma(e)$ is finite).*

Proof. We choose $F = \{e_0 + \cdots + e_{n-1} : e_0, \ldots, e_{n-1} \in E\}$; since E covers e, it follows that F contains e. To see that F is closed, let $e' \simeq e'_0 + \cdots + e'_{n-1} \in F$ for $e'_0, \ldots, e'_{n-1} \in E$, and suppose $f \preceq_\Sigma e'$. To see that $f \in F$, it suffices to validate the claim for the pairs generating $\preceq_\Sigma$:

- If $f = \delta_\Sigma(e', a)$ for $a \in \Sigma$, then $f \simeq f_0 + \cdots + f_{n-1}$ where for $0 \leq i < n$ we have $f_i = \delta_\Sigma(e'_i, a)$. Each of these f_i is covered by E; hence, the sum of terms covering these is in F, and congruent to f. The case where $f = \gamma_\Sigma(e', g, h)$ for $g, h \in \mathcal{T}$ can be argued similarly.
- If $f \preceq_\Sigma e'$ because $\gamma_\Sigma(e', f, g) \not\simeq 0$ or $\gamma_\Sigma(e', g, f) \not\simeq 0$ for some $g \in \mathcal{T}$, then (without loss of generality) assume the former. We then know that $\gamma_\Sigma(e'_i, f, g) \not\simeq 0$ for some $0 \leq i < n$, and hence $f \preceq_\Sigma e'_i$, meaning that there exist $f_0, \ldots, f_{m-1} \in E$ such that $f \simeq f_0 + \cdots + f_{m-1}$, by cover-closure of E. It then follows that $f \in F$. $\square$

Lemma 5.7. *The syntactic PA is finitely supported.*

Proof. As discussed, it suffices to find for every $e \in \mathcal{T}_\simeq$ a finite and cover-closed set $E(e)$ covering e. We proceed inductively. In the base, there are three cases to consider.

- If $e = 0$, then $E(0) = \emptyset$ suffices, since 0 is covered by the empty sum.
- If $e = 1$, then $E(1) = \{1\}$ suffices.
- If $e = a$ for some $a \in \Sigma$, then $E(a) = \{1, a\}$ suffices.

For the inductive step, there are five cases to consider.

- If $e = f + g$, we choose $E(e) = E(f) + E(g)$. This set is cover-closed, because $E(f)$ and $E(g)$ both are. Furthermore, this set covers e, because we can get terms to cover f and g from $E(f)$ and $E(g)$ respectively.

- If $e = f \cdot g$, we choose $E(e) = E(f) \cup E(g) \cup \{f' \cdot g : f' \in E(f)\}$. To see that this set covers $f \cdot g$, let $f_0, \ldots, f_{n-1} \in E(f)$ be such that $f \simeq f_0 + \cdots f_{n-1}$. We can then choose $f_0 \cdot g, \ldots, f_{n-1} \cdot g \in E(e)$ to find that $e \simeq f_0 \cdot g + \cdots f_{n-1} \cdot g$.

  To see that $E(e)$ is cover-closed, we need only consider $f' \cdot g$ for $f' \in E(f)$.

  (i) If $a \in \Sigma$, let $f'_0, \ldots, f'_{n-1} \in E(f)$ with $\delta_\Sigma(f', a) \simeq f'_0 + \cdots + f'_{n-1}$. Also, let $g_0, \ldots, g_{m-1} \in E(g)$ with $\delta_\Sigma(g, a) \simeq g_0 + \cdots + g_{m-1}$. We can then derive as follows:

  $$\begin{aligned}\delta_\Sigma(e, a) &= \delta_\Sigma(f', a) \fatsemi g + f' \star \delta_\Sigma(g, a) \\ &\simeq (f'_0 + \cdots + f_{n-1}) \fatsemi g + f' \star (g_0 + \cdots + g_{m-1}) \\ &\simeq f'_0 \cdot g + \cdots + f'_{n-1} \cdot g + f' \star g_0 + \cdots f' \star g_{m-1}\end{aligned}$$

  All of the non-zero terms in the last form can be found in $E(e)$, and thus $E(e)$ covers $\delta_\Sigma(e, a)$.

  (ii) If $h_1, h_2 \in \mathcal{T}_\simeq$, then $\gamma_\Sigma(f' \cdot g, h_1, h_2)$ is covered by $E(e)$ by an argument similar to the above.

  (iii) If $h_1, h_2 \in \mathcal{T}_\simeq$ such that $\gamma_\Sigma(f' \cdot g, h_1, h_2) \not\simeq 0$, then

  $$\gamma_\Sigma(f', h_1, h_2) \cdot g + f' \star \gamma_\Sigma(g, h_1, h_2) \not\simeq 0$$

  Hence, we know that either $\gamma_\Sigma(f', h_1, h_2) \not\simeq 0$ or $\gamma_\Sigma(g, h_1, h_2) \not\simeq 0$. In the former case, h_1 and h_2 are covered by $E(f)$, while in the latter case h_1 and h_2 are covered by $E(g)$.

- If $e = f \parallel g$, we choose $E(g) = E(e) \cup E(f) \cup \{1, f \parallel g\}$. Immediately, $E(e)$ covers e. For cover-closure of $E(e)$, we need only consider $f \parallel g$.

  (i) If $a \in \Sigma$, then $\delta_\Sigma(e, a) = 0$, and so $E(e)$ covers $\delta_\Sigma(e, a)$.

  (ii) If $h_1, h_2 \in \mathcal{T}_\simeq$, then $\gamma_\Sigma(e, h_1, h_2) \in \{0, 1\}$ by definition of γ_Σ. Consequently, $E(e)$ covers $\gamma_\Sigma(e, h_1, h_2)$.

(iii) If $h_1, h_2 \in \mathcal{T}_{\simeq}$ and $\gamma_\Sigma(e, h_1, h_2) \not\simeq 0$, then $f \simeq h_1$ and $g \simeq h_2$. Since $E(f)$ covers f and $E(g)$ covers g, $E(e)$ covers both.

- If $e = f^*$, we choose $E(e) = E(f) \cup \{f^*\} \cup \{f' \cdot f^* : f' \in E(f)\}$. Immediately, $E(e)$ covers e. For cover-closure of $E(e)$, we need only consider $f' \cdot f^*$ for $f' \in E(f)$.

  (i) If $a \in \Sigma$, let $f'_0, \ldots, f'_{n-1} \in E(f)$ be such that $f' \simeq f'_0 + \cdots + f'_{n-1}$. We can then derive that

$$\begin{aligned}\delta_\Sigma(f' \cdot f^*, a) &= \delta_\Sigma(f', a) \fatsemi f^* + f' \star f^* \\ &\simeq (f'_0 + \cdots + f'_{n-1}) \fatsemi f^* + f' \star f^* \\ &\simeq f'_0 \fatsemi f^* + \cdots + f'_{n-1} \fatsemi f^* + f' \star f^*\end{aligned}$$

  All of the non-zero terms in the last form can be found in $E(e)$, and thus $E(e)$ covers $\delta_\Sigma(f' \cdot f^*, a)$.

  (ii) If $h_1, h_2 \in \mathcal{T}_{\simeq}$, then $\gamma_\Sigma(f' \cdot f^*, h_1, h_2)$ is covered by $E(e)$ by an argument similar to the above.

  (iii) If $h_1, h_2 \in \mathcal{T}_{\simeq}$ and $\gamma_\Sigma(f' \cdot f^*, h_1, h_2) \not\simeq 0$, then $\gamma_\Sigma(f', h_1, h_2) \not\simeq 0$. Thus h_1 and h_2 are covered by $E(f)$, and hence by $E(e)$.

- If $e = f^\dagger$, we choose $E(e) = E(f) \cup \left\{e^\dagger, 1\right\}$. Once more, $E(e)$ covers e trivially. For cover-closure of $E(e)$, we need only consider $e^\dagger$.

  (i) If $a \in \Sigma$, then $\delta_\Sigma(e, a) = 0$, and hence $E(e)$ covers $\delta_\Sigma(e, a)$.

  (ii) If $h_1, h_2 \in \mathcal{T}_{\simeq}$, then $\gamma_\Sigma(e, h_1, h_2) \in \{0, 1\}$ by definition of γ_Σ, and thus $E(e)$ covers $\gamma_\Sigma(e, h_1, h_2)$.

  (iii) If $h_1, h_2 \in \mathcal{T}_{\simeq}$ and $\gamma_\Sigma(e, h_1, h_2) \not\simeq 0$, then $h_1 \simeq f$ and $h_2 \simeq e$. Since $E(f)$ covers f, we conclude that $E(e)$ covers h_1 and h_2. □

Appendix C.2. Well-nestedness

Lemma 5.9. *If $e \preceq_\Sigma f$, then $d_\|(e) \leq d_\|(f)$ and $d_\dagger(e) \leq d_\dagger(f)$.*

Proof. It suffices to verify the claim for the generating pairs of $\preceq_\Sigma$; in each case, we proceed by induction on e.

- Suppose $e \preceq_\Sigma f$ because $f = \delta_\Sigma(e, a)$ for some $a \in \Sigma$. In the base, where $e \in \{0, 1\} \cup \Sigma$, we have that $\delta_\Sigma(e, a) \in \{0, 1\}$, and hence $d_\|(\delta_\Sigma(e, a)) = d_\dagger(\delta_\Sigma(e, a)) = 0$ — the claim then holds immediately.

  For the inductive step, there are five cases to consider. Let $\circ \in \{\|, \dagger\}$.

– If $e = e_0 + e_1$, then $\delta_\Sigma(e, a) = \delta_\Sigma(e_0, a) + \delta_\Sigma(e_1, a)$. By induction, we know that $d_\circ(\delta_\Sigma(e_0, a)) \leq d_\circ(e_0)$ and $d_\circ(\delta_\Sigma(e_1, a)) \leq d_\circ(e_1)$. We can then derive that

$$\begin{aligned} d_\circ(\delta_\Sigma(e, a)) &= \max(d_\circ(\delta_\Sigma(e_0, a)), d_\circ(\delta_\Sigma(e_1, a))) \\ &\leq \max(d_\circ(e_0), d_\circ(e_1)) = d_\circ(e) \end{aligned}$$

The cases where $e = e_0 \cdot e_1$ or $e = e_0^*$ can be argued similarly.

– If $e = e_0 \parallel e_1$ or $e = e_0^\dagger$, then $\delta_\Sigma(e, a) = 0$, thus $d_\circ(\delta_\Sigma(e, a)) = 0$. The claim then follows immediately.

- Suppose $e \preceq_\Sigma f$ because $f = \gamma_\Sigma(e, g, h)$ for some $g, h \in \mathcal{T}$. In the base, where $e \in \{0, 1\} \cup \Sigma$, we have that $\gamma_\Sigma(e, a) = 0$, and hence $d_\parallel(\delta_\Sigma(e, g, h)) = d_\dagger(\gamma_\Sigma(e, g, h)) = 0$ — the claim then holds immediately.

  For the inductive step, there are two cases to consider. Let $\circ \in \{\parallel, \dagger\}$.

  – If $e \in \{e_0 + e_1, e_0 \cdot e_1, e_0^*\}$, then the proof is similar to the corresponding case above.

  – If $e \in e_0 \parallel e_1$ or $e = e_0^\dagger$, then $\gamma_\Sigma(e, g, h) \in \{0, 1\}$, and hence $d_\circ(\delta_\Sigma(e, a)) = 0$. The claim then follows.

- Suppose $e \preceq_\Sigma f$ because $\gamma_\Sigma(f, e, h) \not\simeq 0$ or $\gamma_\Sigma(f, h, e) \not\simeq 0$ for a $h \in \mathcal{T}$. In the base, where $e \in \{0, 1\} \cup \Sigma$, the claim holds vacuously.

  For the inductive step, there are five cases to consider. Let $\circ \in \{\parallel, \dagger\}$.

  – If $e = e_0 + e_1$, then $\gamma_\Sigma(e, g, h) \not\simeq 0$ implies that $\gamma_\Sigma(e_0, g, h) \not\simeq 0$ or $\gamma_\Sigma(e_1, g, h) \not\simeq 0$; w.l.o.g., we assume the former. By induction, we have $d_\circ(g), d_\circ(h) \leq d_\circ(e_0)$. Since $d_\circ(e_0) \leq d_\circ(e)$, the claim follows. The cases where $e = e_0 \cdot e_1$ or $e = e_0^*$ can be argued similarly.

  – If $e = e_0 \parallel e_1$, then $\gamma_\Sigma(e, g, h) \not\simeq 0$ implies that $e_0 \simeq g$ and $e_1 \simeq h$. We also have $d_\parallel(e_0), d_\parallel(e_1) < d_\parallel(e)$, as well as $d_\dagger(e_0), d_\dagger(e_1) \leq d_\dagger(e)$. Since $d_\circ(e_0) = d_\circ(g)$ and $d_\circ(e_1) = d_\circ(h)$, the claim then follows.

  – If $e = e_0^\dagger$, then $\gamma_\Sigma(e, g, h) \not\simeq 0$ implies that $g \simeq e_0$ and $h \simeq e$. We also have $d_\parallel(e_0) \leq d_\parallel(e)$, as well as $d_\dagger(e_0) < d_\dagger(e)$. Since $d_\circ(g) = d_\circ(e_0)$ and $d_\circ(h) = d_\circ(e)$, the claim then follows. □

Lemma 5.10. *Let $e, g, h \in \mathcal{T}$ with $\gamma_\Sigma(e, g, h) \not\simeq 0$. Then $g \prec_\Sigma e$. Furthermore, either $h \prec_\Sigma e$ or there exists an $f \in \mathcal{T}$ such that $e \simeq f^\dagger$.*

Proof. By Lemma 5.9, we need only show (i) $d_{\|}(g) < d_{\|}(e)$ or $d_{\dagger}(h) < d_{\dagger}(e)$, as well as (ii) $d_{\|}(h) < d_{\|}(e)$ or $d_{\dagger}(h) < d_{\dagger}(e)$, or $h \simeq f^{\dagger}$ for some $f \in \mathcal{T}$. The proof for both claims proceeds by induction on e. In the base, where $e \in \{0, 1\} \cup \Sigma$, we have $\gamma_{\Sigma}(e, g, h) = 0$, and hence the claim holds vacuously.

For the inductive step, there are three cases to consider.

- If $e = e_0 + e_1$, then $\gamma_{\Sigma}(e_0, g, h) \not\simeq 0$ or $\gamma_{\Sigma}(e_1, g, h) \not\simeq 0$; w.l.o.g. we assume the former. We then have that $d_{\|}(g) < d_{\|}(e_0)$ or $d_{\dagger}(g) < d_{\dagger}(e_0)$ by induction; the first claim then follows by definition of $d_{\|}$ and $d_{\dagger}$. We also know that $d_{\|}(h) < d_{\|}(e)$ or $d_{\dagger}(h) < d_{\dagger}(e)$ or $h \simeq f^{\dagger}$ for some $f \in \mathcal{T}$ by induction. In the latter case, the second claim follows immediately; otherwise, the claim follows by definition of $d_{\|}$ and $d_{\dagger}$ again.

  The cases where $e = e_0 \cdot e_1$ or $e = e_0^*$ can be argued similarly.

- If $e = e_0 \parallel e_1$, then $g \simeq e_0$ and $h \simeq e_1$ by definition of γ_{Σ}. We then have $d_{\|}(g) = d_{\|}(e_0) < d_{\|}(e)$, and $d_{\|}(h) < d_{\|}(e)$, satisfying both claims.

- If $e = e_0^{\dagger}$, then $g \simeq e_0$ and $h \simeq e$. The second claim holds immediately. For the first claim, observe that $d_{\dagger}(g) = d_{\dagger}(e_0) < d_{\dagger}(e)$. □

Appendix C.3. Deconstruction lemmas

Lemma 5.13. *Let $e_0, e_1 \in \mathcal{T}$, $f \in \mathcal{F}$ and $U \in \mathsf{Pom}^{\mathsf{sp}}$, such that $e_0 \cdot e_1 \xrightarrow{U}_{\Sigma} f$ is of length ℓ. There exist $f_0, f_1 \in \mathcal{F}$ such that $U = U_0 \cdot U_1$, as well as $e_0 \xrightarrow{U_0}_{\Sigma} f_0$ of length ℓ_0 and $e_1 \xrightarrow{U_1}_{\Sigma} f_1$ of length ℓ_1, such that $\ell_0 + \ell_1 = \ell$.*

Proof. The proof proceeds by induction on the length ℓ of $e_0 \cdot e_1 \xrightarrow{U}_{\Sigma} f$. In the base, where $\ell = 0$, we have that $f = e_0 \cdot e_1$ and $U = 1$. We can then choose $f_0 = e_0$ and $f_1 = e_1$ as well as $U_0 = U_1 = 1$ to satisfy the claim.

For the inductive step, let $e_0 \cdot e_1 \xrightarrow{U}_{\Sigma} f$ be of length $\ell + 1$. We find that $U = V \cdot U'$, and a $g \in \mathcal{T}$ such that $e_0 \cdot e_1 \xrightarrow{V}_{\Sigma} g$ is a unit trace, and $g \xrightarrow{U}_{\Sigma} f$ is of length ℓ. If $e_0 \cdot e_1 \xrightarrow{V}_{\Sigma} g$ is a δ-trace, then $V = a$ for some $a \in \Sigma$, and $g = \delta_{\Sigma}(e_0 \cdot e_1, a) = \delta_{\Sigma}(e_0, a) \fatsemi e_1 + e_0 \star \delta_{\Sigma}(e_1, a)$. By Lemma 5.12, we find $f' \in \mathcal{F}$ such that $\delta_{\Sigma}(e_0, a) \fatsemi e_1 \xrightarrow{U'}_{\Sigma} f'$ or $e_0 \star \delta_{\Sigma}(e_1, a) \xrightarrow{U'}_{\Sigma} f'$, of length ℓ. This gives us two cases.

- If $\delta_{\Sigma}(e_0, a) \fatsemi e_1 \xrightarrow{U'}_{\Sigma} f'$, then first note that $\delta_{\Sigma}(e_0, a) \not\simeq 0$, by Lemma 3.3, and hence $\delta_{\Sigma}(e_0, a) \cdot e_1 \xrightarrow{U'}_{\Sigma} f'$. By induction we find $f_0, f_1 \in \mathcal{F}$ and $U' = U'_0 \cdot U'_1$ such that $\delta_{\Sigma}(e_0, a) \xrightarrow{U'_0}_{\Sigma} f_0$ and $e_1 \xrightarrow{U'_1}_{\Sigma} f_1$, and the total length of these traces is ℓ. We can then choose $U_1 = V \cdot U'_0$ and $U_1 = U'_1$

to find that $U = V \cdot U' = V \cdot U'_0 \cdot U'_1 = U_0 \cdot U_1$, as well as $e_0 \xrightarrow{U_0}_\Sigma f_0$ and $e_1 \xrightarrow{U_1}_\Sigma f_1$, of total length $\ell + 1$.

- If $e_0 \star \delta_\Sigma(e_1, a) \xrightarrow{U'}_\Sigma f'$, then first note that $e_0 \star \delta_\Sigma(e_1, a) \not\approx 0$ by Lemma 3.3, and so $e_0 \in \mathcal{F}$. We choose $U_0 = 1$ and $U_1 = U$ as well as $f_0 = e_0$ and $f_1 = f'$ to find that $U = 1 \cdot U = U_0 \cdot U_1$ as well as $e_0 \xrightarrow{U_0}_\Sigma f_0$. Furthermore, $e_1 \xrightarrow{V}_\Sigma \delta_\Sigma(e_1, a) = e_0 \star \delta_\Sigma(e_1, a) \xrightarrow{U'}_\Sigma f'$, meaning that $e_1 \xrightarrow{U}_\Sigma f'$. The total length of these traces is again $\ell + 1$.

The case where $e_0 \cdot e_1 \xrightarrow{V}_\Sigma g$ is a γ-trace can be treated similarly. □

Lemma 5.14. *Let $e \in \mathcal{T}$ and $f \in \mathcal{F}$ and $U \in \mathsf{Pom}^{\mathsf{sp}}$ be such that $e^* \xrightarrow{U}_\Sigma f$. There exist $f_0, \ldots, f_{n-1} \in \mathcal{F}$ such that $U = U_0 \cdots U_{n-1}$ and for $0 \leq i < n$ it holds that $e \xrightarrow{U_i}_\Sigma f_i$.*

Proof. The proof proceeds by induction on the length ℓ of $e^* \xrightarrow{U}_\Sigma f$. In the base, where $\ell = 0$, we have that $f = e^*$ and $U = 1$; it suffices to choose $n = 0$.

For the inductive step, let $e^* \xrightarrow{U}_\Sigma f$ be of length $\ell + 1$, and assume that the claim holds for ℓ. We then find $g \in \mathcal{T}$ and $U = V \cdot U'$ such that $e^* \xrightarrow{V}_\Sigma g$ is a unit trace, and $g \xrightarrow{U'}_\Sigma f$ of length ℓ. If $e^* \xrightarrow{V}_\Sigma g$ is a δ-trace, then $V = a$ for some $a \in \Sigma$, and $g = \delta_\Sigma(e^*, a) = \delta_\Sigma(e, a) \fatsemi e^*$. By Lemma 3.3, and the fact that $g \xrightarrow{U'}_\Sigma f \in \mathcal{F}$, we then know that $\delta_\Sigma(e, a) \not\approx 0$, and hence $\delta_\Sigma(e, a) \cdot e^* \xrightarrow{U'}_\Sigma f$. By Lemma 5.13, we find $f'', f' \in \mathcal{F}$ such that $U' = W \cdot X$ as well as $\delta_\Sigma(e, a) \xrightarrow{W}_\Sigma f''$ and $e^* \xrightarrow{X}_\Sigma f'$ of total length ℓ.

Then, by induction, we find $f_1, \ldots, f_{n-1} \in \mathcal{T}$ such that $X = U_1 \cdots U_{n-1}$, and for $1 \leq i < n$ it holds that $e \xrightarrow{X_i}_\Sigma f_i$. We then choose $f_0 = f''$ and $U_0 = V \cdot W$. For these choices, $U = U_0 \cdot U' = V \cdot W \cdot X = U_0 \cdots U_{n-1}$. Since $e \xrightarrow{V}_\Sigma \delta_\Sigma(e, a) \xrightarrow{W}_\Sigma f''$, we also find that $e \xrightarrow{U_0}_\Sigma f_0$; this completes the proof.

The case where $e^* \xrightarrow{V}_\Sigma g$ is a γ-trace is similar. □

Appendix C.4. Construction lemmas

Lemma 5.17. *Let $e_0, e_1 \in \mathcal{T}$ and $f_0 \in \mathcal{F}$ and $U \in \mathsf{Pom}^{\mathsf{sp}}$ be such that $e_0 \xrightarrow{U}_\Sigma f_0$. Then there exists an $f \in \mathcal{T}$ such that $f_0 \cdot e_1 \lesssim f$, and $e_0 \cdot e_1 \xrightarrow{U}_\Sigma f$.*

Proof. The proof proceeds by induction on the length ℓ of $e_0 \xrightarrow{U}_\Sigma f_0$. In the base, where $\ell = 0$, we can choose $f = e_0 \cdot e_1$ to satisfy the claim.

For the inductive step, let $e_0 \xrightarrow{U}_\Sigma f_0$ be of length $\ell + 1$, and assume the claim holds for traces of length ℓ. We then find $e'_0 \in \mathcal{T}$ and $U = V \cdot U'$ such that $e_0 \xrightarrow{V}_\Sigma e'_0$ is a unit trace, and $e'_0 \xrightarrow{U'}_\Sigma f_0$ is of length ℓ. By Lemma 3.3 and the fact that $e'_0 \xrightarrow{U'}_\Sigma f_0 \in \mathcal{F}$, we know that $e'_0 \not\approx 0$, and thus $e'_0 \fatsemi e_1 = e'_0 \cdot e_1$.

By induction, we find $f' \in \mathcal{T}$ such that $f_0 \cdot e_1 \lesssim f'$, and $e_0' \cdot e_1 \xrightarrow{U'}_\Sigma f'$. If $e_0 \xrightarrow{V}_\Sigma e_0'$ is a δ-trace, then $V = a$ for some $a \in \Sigma$, and $e_0' = \delta_\Sigma(e_0, a)$. Then, by Lemma 5.16, we find $f \in \mathcal{T}$ such that $f' \lesssim f$ and $\delta_\Sigma(e_0 \cdot e_1, a) = e_0' \cdot e_1 + e_0 \star \delta_\Sigma(e_1, a) \xrightarrow{U'}_\Sigma f$. Putting these traces together, we find that $e_0 \cdot e_1 \xrightarrow{U}_\Sigma f$, as well as $f_0 \cdot e_1 \lesssim f' \lesssim f$.

The case where $e_0 \xrightarrow{V}_\Sigma e_0'$ is a γ-trace is similar. □

Lemma 5.18. *Let $e_0 \in \mathcal{T}$ and $f_0, f_1 \in \mathcal{F}$ and $V \in \mathsf{Pom}^{\mathsf{sp}}$ be such that $e_1 \xrightarrow{V}_\Sigma f_1$. There exists an $f \in \mathcal{F}$ such that $f_0 \cdot e_1 \xrightarrow{V}_\Sigma f$.*

Proof. The proof proceeds by induction on the length ℓ of $e_1 \xrightarrow{V}_\Sigma f_1$. If $\ell = 0$, we know that $f_1 = e_1$ and $V = 1$. We can then choose $f = f_0 \cdot e_1$.

For the inductive step, let $e_1 \xrightarrow{V}_\Sigma f_1$ be of length $\ell + 1$, and assume the claim holds for traces of length ℓ. We then find $e_1' \in \mathcal{T}$ and $V = W \cdot V'$ such that $e_1 \xrightarrow{W}_\Sigma e_1'$ is a unit trace, and $e_1 \xrightarrow{V'}_\Sigma f_1$ is of length ℓ. If $e_1 \xrightarrow{W}_\Sigma e_1'$ is a δ-trace, then $W = a$ for some $a \in \Sigma$, and $e_1' = \delta_\Sigma(e_1, a)$. By Lemma 5.16, we find $f \in \mathcal{F}$ such that $\delta_\Sigma(f_0, a) \fatsemi e_1 + e_1' \xrightarrow{V'}_\Sigma f$. Since $\delta_\Sigma(f_0 \cdot e_1, a) = \delta_\Sigma(f_0, a) \fatsemi e_1 + f_0 \star \delta_\Sigma(e_1, a) = \delta_\Sigma(f_0, a) \fatsemi e_1 + e_1'$ we find that $f_0 \cdot e_1 \xrightarrow{W}_\Sigma \delta_\Sigma(f_0, a) \fatsemi e_1 + e_1'$. We conclude that $f_0 \cdot e_1 \xrightarrow{V}_\Sigma f$.

The case where $e_1 \xrightarrow{W}_\Sigma e_1'$ is a γ-trace is similar. □

Lemma 5.19. *Let $e_0, e_1 \in \mathcal{T}$, $f_0, f_1 \in \mathcal{F}$ and $U, V \in \mathsf{Pom}^{\mathsf{sp}}$ such that $e_0 \xrightarrow{U}_\Sigma f_0$ and $e_1 \xrightarrow{V}_\Sigma f_1$. There exists an $f \in \mathcal{F}$ with $e_0 \cdot e_1 \xrightarrow{U \cdot V}_\Sigma f$.*

Proof. By Lemma 5.17, we find $f' \in \mathcal{T}$ such that $f_0 \cdot e_1 \lesssim f'$ and $e_0 \cdot e_1 \xrightarrow{U}_\Sigma f'$. By Lemma 5.18, we find $f'' \in \mathcal{F}$ such that $f_0 \cdot e_1 \xrightarrow{V}_\Sigma f''$. By Lemma 5.16, we find $f \in \mathcal{F}$ such that $f' \simeq f_0 \cdot e_1 + f' \xrightarrow{V}_\Sigma f$. In total, we have $e_0 \cdot e_1 \xrightarrow{U \cdot V}_\Sigma f$. □

Lemma 5.20. *Let $e \in \mathcal{T}$ and $f_0, \ldots, f_{n-1} \in \mathcal{F}$ and $U_0, \ldots, U_{n-1} \in \mathsf{Pom}^{\mathsf{sp}}$ be such that for $0 \leq i < n$ it holds that $e \xrightarrow{U_i}_\Sigma f_i$. There exists an $f \in \mathcal{F}$ such that $e^* \xrightarrow{U_0 \cdots U_{n-1}}_\Sigma f$.*

Proof. Without loss of generality, we can assume that for $0 \leq i < n$ it holds that $e \xrightarrow{U_i}_\Sigma f_i$ is non-trivial. The proof proceeds by induction on n. In the base, where $n = 0$, we can choose $f = e^*$ to satisfy the claim.

For the inductive step, assume that $n > 0$ and that the claim holds for $n - 1$. By induction, we can find $f' \in \mathcal{F}$ such that $e^* \xrightarrow{U_1 \cdots U_{n-1}}_\Sigma f'$. Since $e \xrightarrow{U_1}_\Sigma f_1$ is non-trivial, we find $e' \in \mathcal{T}$ and $U_0 = V \cdot U_0'$ such that $e \xrightarrow{V}_\Sigma e'$ is a unit trace, and $e' \xrightarrow{U_0'}_\Sigma f_0$. We note that by Lemma 3.3, this implies

that $e' \not\approx 0$. By Lemma 5.19, we find $f \in \mathcal{F}$ such that $e' \cdot e^* \xrightarrow{U_0' \cdot U_1 \cdots U_{n-1}}_\Sigma f$. If $e \xrightarrow{V}_\Sigma e'$ is a δ-trace, then $V = a$ for some $a \in \Sigma$, and $e' = \delta_\Sigma(e, a)$. In that case, $e^* \xrightarrow{V}_\Sigma \delta_\Sigma(e^*, a) = \delta_\Sigma(e, a) \fatsemi e^* = e' \cdot e^*$. Consequently, $e^* \xrightarrow{V}_\Sigma e' \cdot e^* \xrightarrow{U_0' \cdot U_1 \cdots U_{n-1}}_\Sigma f$ and therefore $e^* \xrightarrow{U_0 \cdots U_{n-1}}_\Sigma f$.

The case where $e \xrightarrow{V}_\Sigma e'$ is a γ-trace is similar. □

Appendix C.5. Soundness of the translation

Lemma 5.21. *If $e \in \mathcal{T}$, then $L_\Sigma(e) = [\![e]\!]$.*

Proof. We proceed by induction on e. In the base, there are two cases to consider. On the one hand, if $e \in \{0, 1\}$, then the claim follows from Lemma 3.3. On the other hand, if $e = a$ for some $a \in \Sigma$, then the inclusion from left to right is simple: $a \xrightarrow{a}_\Sigma \delta_\Sigma(a, a) = 1 \in \mathcal{F}$, and therefore we can conclude that $a \in L_\Sigma(a)$. For the inclusion from right to left, suppose that $a \xrightarrow{U}_\Sigma f$ for some pomset U and $f \in \mathcal{F}$. In that case, $f \neq a$ (for $a \notin \mathcal{F}$), and thus $e \xrightarrow{U}_\Sigma f$ must be non-trivial. We therefore find that $U = U_0 \cdot U'$ and a $g \in \mathcal{T}$ such that $a \xrightarrow{U_0}_\Sigma g$ is a unit trace, and $g \xrightarrow{U'}_\Sigma f$ holds as well. Whether $a \xrightarrow{U_0}_\Sigma g$ is a γ-trace or δ-trace, we have that $g \in \{0, 1\}$. Furthermore, by Lemma 3.3 and the fact that $f \in \mathcal{F}$, we know that $g \xrightarrow{U'}_\Sigma f$ must be trivial (for otherwise $f = 0 \notin \mathcal{F}$), meaning that $g = f = 1$. It then follows that $a \xrightarrow{U_0}_\Sigma g$ was a δ-trace with $U_0 = a$, and $U = U_0 \cdot U' = a \cdot 1 = a$.

For the inductive step, suppose the claim holds for all strict subterms of e. There are five cases to consider.

- If $e = e_0 + e_1$, then first suppose that $U \in L_\Sigma(e)$. By Lemma 5.12 we know that $U \in L_\Sigma(e_0)$ or $U \in L_\Sigma(e_0)$. By induction, we find that $U \in [\![e_0]\!] \cup [\![e_1]\!] = [\![e_0 + e_1]\!]$.

  For the other inclusion, let $U \in [\![e_0 + e_1]\!]$. If $U \in [\![e_0]\!]$, then $U \in L_\Sigma(e_0)$ by induction; then $U \in L_\Sigma(e_0 + e_1)$ by Lemma 5.12. The case where $U \in [\![e_1]\!]$ is similar.

- If $e = e_0 \cdot e_1$ (resp. $e = e_0^*$), then the equality follows from Lemma 5.13 and Lemma 5.19 (resp. Lemma 5.14 and Lemma 5.20) by argument analogous to the previous case.

- If $e = e_0 \parallel e_1$, then first suppose that $U \in L_\Sigma(e_0 \parallel e_1)$. A simple look at the sequential and parallel derivatives for $e_0 \parallel e_1$ shows that $U = V \parallel W$ such that $V \in L_\Sigma(e_0)$ and $W \in L_\Sigma(e_1)$. By induction, $V \in [\![e_0]\!]$ and $W \in [\![e_1]\!]$, and thus $U = V \parallel W \in [\![e_0 \parallel e_1]\!]$.

For the other inclusion, suppose that $U \in [\![e_0 \parallel e_1]\!]$. Then $U = V \parallel W$ such that $V \in [\![e_0]\!]$ and $W \in [\![e_1]\!]$. By induction, we find that $V \in L_\Sigma(e_0)$ and $W \in [\![e_1]\!]$. Another look at the parallel derivatives for $e_0 \parallel e_1$ then tells us that $U = V \parallel W \in L_\Sigma(e_0 \parallel e_1)$.

- If $e = f^\dagger$, then first note that $f^\dagger$ is a recursive state by Lemma 5.11. By Lemma 4.5 and induction, we can then conclude that

$$L_\Sigma(f^\dagger) = L_\Sigma(f)^\dagger = [\![f]\!]^\dagger = [\![f^\dagger]\!] \qquad \square$$

Appendix C.6. Soundness modulo congruence

For technical completeness, justify our notation in Section 5 by arguing that the constructs used are well-defined modulo $\simeq$.

Lemma C.1. *Let $e, f \in \mathcal{T}$. The following hold:*

(i) If $e \simeq f$, then $[\![e]\!] = [\![f]\!]$, and

(ii) if $e \simeq f$, then $e \in \mathcal{F}$ if and only if $f \in \mathcal{F}$, and

Proof. For the first part, it suffices to show that the claim holds for the pairs generating $\simeq$. This gives us four cases to consider.

- If $e = f + 0$, then $[\![e]\!] = [\![f]\!] \cup [\![0]\!] = [\![f]\!] \cup \emptyset = [\![f]\!]$.
- If $e = f + f$, then $[\![e]\!] = [\![f]\!] \cup [\![f]\!] = [\![f]\!]$.
- If $e = g_0 + g_1$ and $f = g_1 + g_0$, then $[\![e]\!] = [\![g_0]\!] \cup [\![g_1]\!] = [\![g_1]\!] \cup [\![g_0]\!] = [\![f]\!]$.
- If $e = g_0 + (g_1 + g_2)$ and $f = (g_0 + g_1) + g_2$, then

$$[\![e]\!] = [\![g_0]\!] \cup ([\![g_1]\!] \cup [\![g_2]\!]) = ([\![g_0]\!] \cup [\![g_1]\!]) \cup [\![g_2]\!] = [\![f]\!]$$

- If $e = (g_0 + g_1) \cdot g_2$ and $f = g_0 \cdot g_2 + g_1 \cdot g_2$, then

$$[\![e]\!] = ([\![g_0]\!] \cup [\![g_1]\!]) \cdot [\![g_2]\!] = [\![g_0]\!] \cdot [\![g_2]\!] \cup [\![g_1]\!] \cdot [\![g_2]\!] = [\![f]\!]$$

For the second part, it suffices to verify that the claim holds for the pairs generating $\simeq$. This gives us again four cases to consider.

- Suppose $e = f + 0$. If $e \in \mathcal{F}$, then either $f \in \mathcal{F}$ or $0 \in \mathcal{F}$. Since the latter is false, $f \in \mathcal{F}$. Also, if $f \in \mathcal{F}$, then $e = f + 0 \in \mathcal{F}$ immediately.

- Suppose $e = f + f$. If $f + f \in \mathcal{F}$, then $f \in \mathcal{F}$; if $f \in \mathcal{F}$, then $f + f \in \mathcal{F}$.
- Suppose $e = g_0 + g_1$ and $f = g_1 + g_0$. If $g_0 + g_1 \in \mathcal{F}$, then $g_0 \in \mathcal{F}$ or $g_1 \in \mathcal{F}$; in either case, $g_1 + g_0 \in \mathcal{F}$. The other direction is analogous.
- Suppose $e = g_0 + (g_1 + g_2)$ and $f = (g_0 + g_1) + g_2$. If $e \in \mathcal{F}$, then $g_0 \in \mathcal{F}$ or $g_1 + g_2 \in \mathcal{F}$, and thus one of g_0, g_1, g_2 must be in $\mathcal{F}$. But then $g_0 + g_1$ or g_2 must be in $\mathcal{F}$, and thus $f = (g_0 + g_1) + g_2 \in \mathcal{F}$. The proof in the other direction is similar.
- Suppose $e = (g_0 + g_1) \cdot g_2$ and $f = g_0 \cdot g_2 + g_1 \cdot g_2$. If $e \in \mathcal{F}$, then $g_0 + g_1 \in \mathcal{F}$ and $g_2 \in \mathcal{F}$, meaning that g_0 or g_1 can be found in $\mathcal{F}$, and g_2 too. In that case, either g_0 and g_2, or g_1 and g_2 can be found in $\mathcal{F}$, and thus $f \in \mathcal{F}$. The proof in the other direction is similar. □

Lemma C.2. *Let $e, f \in \mathcal{T}$ such that $e \simeq f$. The following hold:*

(i) If $a \in \Sigma$, then $\delta_\Sigma(e, a) \simeq \delta_\Sigma(f, a)$.

(ii) If $g, h, g', h' \in \mathcal{T}$ with $g \simeq g'$ and $h \simeq h'$, then $\gamma_\Sigma(e, g, h) = \gamma_\Sigma(f, g', h')$.

Proof. For the first part, it suffices to verify the claim for the pairs generating $\simeq$. This gives us four cases to consider.

- If $e = f + 0$, then $\delta_\Sigma(e, a) = \delta_\Sigma(f, a) + \delta_\Sigma(0, a) = \delta_\Sigma(f, a) + 0 \simeq \delta_\Sigma(f, a)$.
- If $e = f + f$, then $\delta_\Sigma(e, a) = \delta_\Sigma(f, a) + \delta_\Sigma(f, a) \simeq \delta_\Sigma(f, a)$.
- If $e = g_0 + g_1$ and $f = g_1 + g_0$, then

$$\begin{aligned}\delta_\Sigma(e, a) &= \delta_\Sigma(g_0, a) + \delta_\Sigma(g_1, a) \\ &\simeq \delta_\Sigma(g_1, a) + \delta_\Sigma(g_0, a) = \delta_\Sigma(f, a)\end{aligned}$$

- If $e = g_0 + (g_1 + g_2)$ and $f = (g_0 + g_1) + g_2$, then

$$\begin{aligned}\delta_\Sigma(e, a) &= \delta_\Sigma(g_0, a) + (\delta_\Sigma(g_1, a) + \delta_\Sigma(g_2, a)) \\ &\simeq (\delta_\Sigma(g_0, a) + \delta_\Sigma(g_1, a)) + \delta_\Sigma(g_2, a) \\ &= \delta_\Sigma(f, a)\end{aligned}$$

- If $e = (g_0 + g_1) \cdot g_2$ and $f = g_0 \cdot g_2 + g_1 \cdot g_2$, then

$$\begin{aligned}\delta_\Sigma(e, a) &= \delta_\Sigma(g_0 + g_1, a) \fatsemi g_2 + (g_0 + g_1) \star \delta_\Sigma(g_2, a) \\ &= (\delta_\Sigma(g_0, a) + \delta_\Sigma(g_1, a)) \fatsemi g_2 + (g_0 + g_1) \star \delta_\Sigma(g_2, a) \\ &\simeq \delta_\Sigma(g_0, a) \fatsemi g_2 + \delta_\Sigma(g_1, a) \fatsemi g_2 + g_0 \star \delta_\Sigma(g_2, a) + g_1 \star \delta_\Sigma(g_2, a) \\ &\simeq \delta_\Sigma(g_0 \cdot g_2 + g_1 \cdot g_2, a) = \delta_\Sigma(f, a)\end{aligned}$$

  in which we make use of the fact that $e + f \simeq 0$ if and only if $e \simeq 0$ and $f \simeq 0$. The implication from right to left follows from $e + f \simeq 0 + 0 \simeq 0$, and the other implication from the fact that $e \simeq e + 0 \simeq e + e + f \simeq e + f \simeq 0$, and similarly for f.

For the second part, note that $\gamma_\Sigma(f, g, h) \simeq \gamma_\Sigma(f, g', h')$ by construction of γ_Σ. It therefore suffices to verify that $\gamma_\Sigma(e, g, h) \simeq \gamma_\Sigma(f, g, h)$ for the pairs generating $\simeq$. This gives us four cases to consider, all of which go through in the same manner as above. □

Lemma C.3. *Let $e \simeq f$. Then $d_\parallel(e) = d_\parallel(f)$ and $d_\dagger(e) = d_\dagger(f)$.*

Proof. Let $\circ \in \{\parallel, \dagger\}$. It suffices to verify the claim for the generating pairs.

- If $e = f + 0$, then $d_\circ(e) = \max(d_\circ(f), d_\circ(0)) = d_\circ(f)$.
- If $e = f + f$, then $d_\circ(e) = \max(d_\circ(f), d_\circ(f)) = d_\circ(f)$.
- If $e = e_0 + e_1$ and $f = e_1 + e_0$, then

$$\begin{aligned}d_\circ(e) &= \max(d_\circ(e_0), d_\circ(e_1)) \\ &= \max(d_\circ(e_1), d_\circ(e_0)) = d_\circ(f)\end{aligned}$$

- If $e = e_0 + (e_1 + e_2)$ and $f = (e_0 + e_1) + e_2$, then

$$\begin{aligned}d_\circ(e) &= \max(d_\circ(e_0), \max(d_\circ(e_1), d_\circ(e_2))) \\ &= \max(\max(d_\circ(e_0), d_\circ(e_1)), d_\circ(e_2)) = d_\circ(f)\end{aligned}$$

- If $e = e_0 \cdot (e_1 + e_2)$ and $f = e_0 \cdot e_1 + e_0 \cdot e_2$, then

$$\begin{aligned}d_\circ(e) &= \max(d_\circ(e_0), \max(d_\circ(e_1), d_\circ(e_2))) \\ &= \max(\max(d_\circ(e_0), d_\circ(e_1)), \max(d_\circ(e_0), d_\circ(e_2))) = d_\circ(f)\end{aligned}$$ □