A Complete Diagrammatic Calculus for Automata Simulation

Thibaut Antoine
ENS Rennes, France

Robin Piedeleu
University College London, UK

Alexandra Silva
Cornell University, Ithaca, NY, USA

Fabio Zanasi
University College London, UK

Abstract

We give a sound and complete (in)equational theory for simulation of finite state automata. Our approach uses a string diagrammatic calculus to represent automata and a functorial semantics to capture simulation in a compositional way. Interestingly, in contrast to other approaches based on regular expressions, fixpoints are a derived notion in our calculus and the resulting axiomatisation is finitary.

2012 ACM Subject Classification Theory of computation

Keywords and phrases finite-state automata, simulation, string diagrams, axiomatisation

Digital Object Identifier 10.4230/LIPIcs.CSL.2025.27

Funding Fabio Zanasi acknowledges support from EPSRC EP/V002376/1, MIUR P2022HXNSC (PRIN 2022 PNRR - Next Generation EU), and ARIA Safeguarded AI TA1.1 grant n.8777242. The other authors acknowledge support from ERC grant Autoprobe (no. 101002697), ARIA Safeguarded AI program, and EPSRC Standard Grant CLeVer (EP/S028641/1).

Acknowledgements Fabio Zanasi conducted part of this research while affiliated with the University of Bologna, Italy.

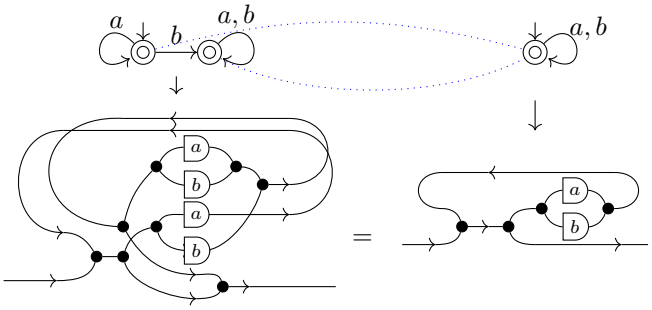
1 Introduction

Non-deterministic automata are basic models of computation which play a central role in formal verification – for instance, the Kripke frames used in the semantics of modal and temporal logics are non-deterministic automata. In verification, one often tries to answer a question of the form: does state s of the model satisfy φ? This question can be rephrased in terms of trace containment – do all the valid traces starting from s satisfy φ? To effectively answer it, formulae can be compiled to models which are then compared to the original model, using *simulation* for trace containment [2] (or, if one requires a finer semantics, bisimulation).

Another approach to reason about model behaviour is algebraic: one can design a language into which both the models and the formulae can be compiled, and reason equationally about their relationship in this common language. A very successful example of this approach is Kleene algebra and extensions thereof [13, 15, 23, 10]. In the algebraic approach, it is important that one has sufficient axioms to reason about the equivalence or refinement of behaviours. This amounts to providing a *sound and complete* axiomatization of the intermediary language with respect to model behaviour. Famously, Kozen was the first to provide a sound and complete algebraic axiomatization of language-equivalence of regular expressions [14]. The main challenge to overcome was the handling of loops (or Kleene star), a challenge which reappears in extensions of Kleene algebra and in other process calculi.

This paper builds on recent work connecting language theory and string diagrams [20] to offer a novel perspective on non-deterministic finite-state automata (NFA) and simulation. Remarkably, [20] provides a *finitary* axiomatization of NFA up to language equivalence. The key point is that automata are represented in a more modular syntax of string diagrams, in which loops/fixpoints are a derived notion. Moreover, the proposed diagrammatic language is more expressive, exploiting the underlying lattice structure of languages. This allows the authors to *(i)* encode not only the NFA themselves, but relations between the states of NFA, as well as *(ii)* an equational proof that these relations satisfy the defining properties of simulations [20]. The payoff is that it becomes possible to reason purely equationally (or inequationally, since we allow inequalities between diagrams) about fixpoints, a feature that is provably impossible to achieve in a traditional syntax [22].

Below, we give an example of a (bi)simulation (in blue) between two NFA, with their diagrammatic representation below. These correspond to $a^*(b(a+b)^*+1)$ and $(a+b)^*$; proving that they are (bi)similar in a traditional syntax requires fixpoint axioms. Our axiomatisation on the other hand will allow us to derive this fact using purely (in)equational reasoning on the diagrams themselves (Example 38).



Contributions. Our main contribution is a complete axiomatisation of NFA simulation. Contrary to language equivalence, the algebraic study of simulation has not been explored in depth in previous work, with the notable exception of Frendrup and Jensen's work on CCS expressions modulo similarity [9]. Given the clear importance of simulation for program refinement and bisimulation, it is worthwhile to develop different techniques to study it. Here, we offer a novel perspective on simulation through a string diagrammatic calculus with the same syntax as the one in [20], but a new semantics, capturing similarity instead of language equivalence. Importantly, the added expressiveness of our calculus allows the complete axiomatisation to be *finite*. Moreover, on the road to completeness, we prove several results about the algebraic structure of automata up to simulation, in particular that they form a lattice (Section 2.2, Theorem 5), and characterise fixed-points of systems of linear inequalities (Theorem 8).

Outline. After introducing some preliminary material in Section 2, we recall the diagrammatic calculus in which we encode NFA, together with a functorial (that is, compositional) semantics expressed in terms of simulation, rather than language-equivalence (Section 3). In Section 4 we provide constructions to translate between NFA and string diagrams. The main technical result appears in Section 5: we prove that the calculus is sound and complete to reason about simulation of NFA. We discuss related work and future research in Section 6, including a discussion of how our approach could extend to NFA modulo bisimilarity.

2 Background: automata and simulation

In this section we recall the necessary background regarding automata, simulation, as well as some basic properties and algebraic operations first introduced by Milner [18]. Then, we characterise the solutions of certain systems of equations (or rather, inequations, in this case), a result which will be instrumental in showing the soundness of our diagrammatic representation of NFA in later sections.

In what follows, we fix some finite alphabet Σ, over which all NFA will be defined and write $\mathcal{P}_f(\Sigma)$ for the set of finite subsets of Σ.

2.1 Automata and their algebraic operations

We recall here the notion of automaton with which we will work throughout.

▶ **Definition 1** (Nondeterministic Finite Automaton). *A nondeterministic finite automaton (NFA) A over Σ is a quadruple (Q, q_0, F, α) where Q is a finite set of states, $q_0 \in Q$ is the initial state of A, $F \subseteq Q$ is a set of final states (also seen as a function $Q \to \{0,1\}$), and $\alpha \subseteq Q \times \Sigma \times Q$ is the transition relation. We will denote $q \xrightarrow{a}_A q'$ for $(q, a, q') \in \alpha$, and will omit the sub-scripted A in case it is clear in context. Note that we do not allow ϵ-transitions.*

A path in A is a sequence $(q_{i_0}, \dots, q_{i_k}) \in Q^{k+1}$ where $q_{i_j} \to_A q_{i_{j+1}}$, denoted $q_{i_0} \rightsquigarrow_A q_{i_k}$.

We will use the following standard operations on NFA extensively: sum, (synchronous) product and composition, including prefixing. Rigorous definitions can be found in Appendix A. Let $A = (Q, q_0, F, \alpha)$ and $B = (S, s_0, G, \beta)$ be two NFA.

Sum. $A + B$ is defined by taking $Q \cup S \cup \{t_0\}$ where t_0 is a new state which can transition to the set of states reachable from q_0 and s_0.

Product. $A \times B$ is defined by taking the $Q \times S$ as set of states and allowing transitions $(q, s) \xrightarrow{a}_{A\times B} (q', s')$ iff $q \xrightarrow{a}_A q'$ and $s \xrightarrow{a}_B s'$.

Composition. $A.B$ is defined by attaching a copy of B to every final state in A and keeping as final states those of B.

Prefix. $a.A$ is the composition of $(\{q_0, q_1\}, q_0, \{q_1\}, \{(q_0, a, q_1)\})$ and A. We will also write $S.A$ for $S \in \mathcal{P}_f(\Sigma)$ as a shorthand for $\left(\sum_{a\in S} a\right).A = \sum_{a\in S} a.A$.

Finally, we define three special NFA to which we will often refer:

$$\bot = (\{q_0\}, q_0, 0, \varnothing) \qquad 1 = (\{q_0\}, q_0, 1, \varnothing) \qquad \top = (\{q_0\}, q_0, 1, \{(q_0, a, q_0) \mid a \in \Sigma\})$$

2.2 The simulation lattice

We now define (strong) *simulation* of a NFA by another, the central concept of this paper.

▶ **Definition 2** (Simulation Relation, Simulation preorder). *Let $A = (Q, q_0, F, \alpha), B = (S, s_0, G, \beta)$ be two NFAs. A simulation relation from A to B is a binary relation $R \subseteq Q \times S$ such that (1) $(q_0, s_0) \in R$; (2) if $(q, s) \in R$, then $F(q) \leq G(s)$; (3) if $(q, s) \in R$ and $q \xrightarrow{a}_A q'$, then there exists a state $s' \in S$ such that $s \xrightarrow{a}_B s'$ and $(q', s') \in R$.*

Note that condition (3) can also be phrased in terms of relational composition as $R^{-1}; \to_A \subseteq \to_B; R^{-1}$. In this case, we say that A is simulated by B (or that B simulates A) and we write $A \preceq B$, or $A \preceq_R B$ to specify the simulation relation.

It immediately follows that $\preceq$ is a preorder on NFAs: for all A, B, C NFAs we have $A \preceq_{\mathrm{id}} A$, as well as $A \preceq_R B$ and $B \preceq_{R'} C$ implies $A \preceq_{R;R'} C$.

▶ **Definition 3** (Similarity partial order). *If A and B are two NFA such that $A \preceq B$ and $B \preceq A$, we say that they are* similar *and write $A \simeq B$.*

We call Ω the set of all NFA modulo similarity. Given two equivalence classes $X = [A]$, $Y = [B]$, we write $X \leq Y$ if $A \preceq B$.

It is a standard fact about preorders that $\leq$ forms a partial order over Ω. Moreover, all previously defined operations are monotone with respect to it.

▶ **Proposition 4** (Monotony of composition). *Let $A = (Q, q_0, F, \alpha)$ and $B = (S, s_0, G, \beta)$ be two NFAs and let $A' = (Q', q'_0, F', \alpha')$, $B' = (S', s'_0, G', \beta')$ such that $A \preceq A'$, $B \preceq B'$. Then we have $A.B \preceq A'.B'$. In particular, prefixing is monotone with respect to $\preceq$.*

Proof. Assume in particular that $A \preceq_{R_A} A'$, $B \preceq_{R_B} B'$. Let $R = R_A \cup \{((i,s),(j,s')) \mid (s,s') \in R_B, (q_i, q'_j) \in R_A, F(q_i) = 1\}$. We will show that $A.B \preceq_R A'.B'$.

- $R_A \subseteq R$, thus $(q_0, q'_0) \in R$.
- Let $(t,t') \in R$, and $t \xrightarrow{a}_{A.B} u$. The interesting case happens when $t = q_i$ with $F(q_i) = 1$ and $u = (i,s)$ with $s_0 \xrightarrow{a}_B s$. Since $B \preceq_{R_B} B'$, $(s_0, s'_0) \in R_B$ and there is a $s' \in S'$ such that $s'_0 \xrightarrow{a}_B s'$ and $(s,s') \in R_B$.
  Moreover, since $t = q_i \in Q$ and $(t,t') \in R$, $t' = q'_j \in Q'$; and we also have $F(q_i) = 1 \leq F'(q'_j)$ thus q'_j is final in A'. By definition of $A'.B'$ this means that $q'_j \xrightarrow{a}_{A'.B'} (j,s')$, and by definition of R we also have $((i,s),(j,s')) \in R$, which concludes the proof. ◀

Proposition 4 allows us to lift prefixing to equivalence classes of NFA modulo similarity as a monotone operation: let $a.[A] = [a.A]$ for $[A] \in \Omega$.

The second key result of this subsection is the existence of a lattice structure on Ω. We first state the necessary properties for NFA before lifting the sum and product to Ω.

▶ **Theorem 5** (Bounded lattice operations). *Let A, B be two NFAs. We have:*

1. $\bot \preceq A \preceq \top$
2. $A \times B \preceq A, B$ *and if* $C \preceq A, B$ *then* $C \preceq A \times B$ *(that is, product of NFAs acts as a meet)*
3. $A, B \preceq A + B$ *and if* $A, B \preceq C$ *then* $A + B \preceq C$ *(that is, sum of NFAs acts as a join)*

Proof. We give the simulation relation for each:

1. The full relation suffices for each inequality.
2. $A \times B \preceq A$ through the projection, *i.e.*, the relation $\{((q,s),q) \mid q \in Q\}$. The same goes for B symmetrically. Moreover if $C \preceq_{R_A} A$ and $C \preceq_{R_B} B$ then $C \preceq_R A \times B$ with $R = \{(t,(q,s)) \mid (t,q) \in R_A \text{ and } (t,s) \in R_B\}$.
3. The simulation relations for $A, B \preceq A + B$ are given by the injections. If $A \preceq_{R_A} C$ and $B \preceq_{R_B} C$, then $A + B \preceq_R C$ with $R = \{(p_0, t_0)\} \cup \{(q,t) \mid (q,t) \in R_A\} \cup \{(s,t) \mid (s,t) \in R_B\}$, if p_0 and t_0 are the initial state of $A + B$ and C respectively. ◀

Direct consequences of this theorem are the monotony of $+$ and $\times$, their commutativity and unitality (with $\bot$ and $\top$ respectively). The monotony of $+$ and $\times$ also allows us to lift the sum and product to the set of NFA modulo similarity: for $[A], [B] \in \Omega$, let $[A] + [B] = [A + B]$ and $[A] \times [B] = [A \times B]$; finally we let $\bot$ and $\top$ denote $[\bot]$ and $[\top]$ respectively.

▶ **Corollary 6.** *Ω is a lattice, with $+$ as join, $\times$ as meet, $\bot$ as bottom, and $\top$ as top.*

In what follows, we will often use NFA to denote their equivalence class modulo similarity.

2.3 Systems of linear inequalities

In this section we introduce systems of linear inequalities and characterise their (least) solutions. This is not only an original result of independent interest, but one we will use later to show the soundness of the representation of NFA in our diagrammatic calculus.

▶ **Definition 7** (System of linear inequations). *Let $n \in \mathbb{N}$ and $K_0, \dots, K_n \in \Omega$, and for all $0 \leq i, j \leq n$, let $d_{i,j} \in \mathcal{P}_f(\Sigma)$. These define a system of $n+1$ inequations:*

$$(E) : \left\{ K_i + \sum_{j=0}^{n} d_{i,j}.X_j \leq X_i \right\}_{0 \leq i \leq n}$$

where $X_0, \dots, X_n$ are variables taking value in Ω. Using matrix multiplication and vector notations, we will write (E) as $\mathbf{K} + \mathbf{DX} \leq \mathbf{X}$.

The following theorem is the key result of this section: given a system of inequalities, we construct the (equivalence class of) NFA that is its unique smallest solution. Frendrup and Jensen prove a similar result [9, Theorem 7] for their syntax directly, but our methods are different and of independent interest.

▶ **Theorem 8.** *Let $\mathbf{D}$ be a matrix of coefficients in $\mathcal{P}_f(\Sigma)$.*

1. *For every vector $\mathbf{K}$, the system $\mathbf{K} + \mathbf{DX} \leq \mathbf{X}$ has a unique smallest solution $S(\mathbf{K})$.*
2. *Let $\mathbf{F} \in \{0,1\}^{n+1}$, K be a NFA and $A = (Q, q_0, \mathbf{F}, \alpha)$ where $Q = \{q_i\}_{0 \leq i \leq n}$, and $q_i \xrightarrow{a} q_j$ iff $a \in d_{i,j}$. Let $\mathbf{A} = (A_0, \dots, A_n)$ where $A_i = (Q, q_i, \mathbf{F}, \alpha)$. Then $S(\mathbf{F}.K) = \mathbf{A}.K$.*

Proof.

1. By monotony of prefixing and summing, $\mathbf{X} \mapsto \mathbf{K} + \mathbf{DX}$ is monotonous; by the Knaster-Tarski theorem it has a unique least fixed point, which is the solution we are looking for.
2. One can see easily that $\mathbf{A}.K$ is a solution. We show that it is the smallest. For that sake, let $\mathbf{X} = (X_0, \dots, X_n)$ be a solution of the system. In all that follows we write $x_0^{(i)}$ for the initial state of X_i. For each $0 \leq i \leq n$, let $Y_i = F_i.K + \sum_{j=0}^{n} d_{i,j}.X_j$, s_i be its initial state, and R_i be the simulation relation from Y_i to X_i. We now fix a i, and build a simulation relation S from $A_i.K$ to X_i.
   First, for all $0 \leq j \leq n$ we define P_j as follows:

$$P_j = \bigcup_{q_{i_0} \to_{A_i} \cdots \to_{A_i} q_{i_k}} R_{i_{k-1}}; \cdots; R_{i_0}$$

   where $i_0 = i$ and $i_k = j$. It is a simulation relation by union and composition. Then let

$$S' = \bigcup_{q_j \text{ s.t. } q_i \rightsquigarrow_{A_i} q_j} \{(q_j, x) \mid (x_0^{(j)}, x) \in P_j\}$$

$$S'' = \bigcup_{\substack{q_j \text{ s.t. } q_i \rightsquigarrow_{A_i} q_j \\ F(q_j)=1}} \{((j,t), x) \mid \exists x' \in X_j : (t, x') \in R_j \text{ and } (x', x) \in P_j\}$$

   and $S = \{(q_i, x_0^{(i)})\} \cup S' \cup S''$.
   Note that S'' is well defined since if q_j is final, then $K \preceq_{R_j} X_j$. We show that S is a simulation relation.

- By definition, $(q_i, x_0^{(i)}) \in S$
- Let $(q_j, x) \in S$, and $q_j \xrightarrow{a} q_k$. By definition, this means $(q_j, x) \in S'$, i.e. $(x_0^{(j)}, x) \in P_j$. By definition of R_j, we have $(s_j, x_0^{(j)}) \in R_j$. Combined to the existence of a path $q_i \rightsquigarrow q_j \to q_k$, this implies $(s_j, x) \in P_k$. Moreover, we know by construction of A that $s_j \xrightarrow{a} x_0^{(k)}$ since $q_j \xrightarrow{a} q_k$. Therefore, P_k being a simulation relation, there is a y such that $x \xrightarrow{a} y$ and $(x_0^{(k)}, y) \in P_k$ i.e. $(q_k, y) \in S'$.
- Let q_j be an accessible final state in A and $(q_j, x) \in S$ with $q_j \xrightarrow{a} (j, t)$, meaning $t_0 \xrightarrow{a}_K t$. Moreover since q_j is final, $K \preceq_{R_j} X_j$ and thus $(t_0, x_0^{(j)}) \in R_j$. By simulation there is a $x' \in X_j$ such that $x_0^{(j)} \xrightarrow{a} x'$ and $(t, x') \in R_j$. Moreover $(q_j, x) \in S$ thus $(x_0^{(j)}, x) \in P_j$. By simulation, there is a y with $(x', y) \in P_j$ such that $x \xrightarrow{a} y$, which concludes this case.
- The proof in the case $(j, t) \xrightarrow{a} (j, t')$ is direct by applying the simulation property. ◀

3 Syntax and semantics

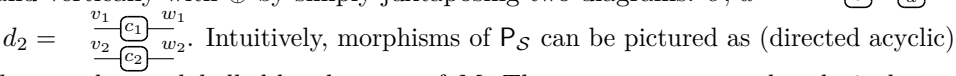
In this section, we define a diagrammatic calculus in which we can encode NFA in a natural way. The syntax has appeared in previous work [20], but the semantics is new, reflecting the focus of this work on simulation, rather than language-equivalence. We will then equip the same syntax with an (in)equational theory which we will show in Section 5 fully axiomatises the intended semantics. We proceed in three steps:

- In Section 3.1, we define our syntax as a free coloured *prop* on a number of generators, using string diagrams to depict its morphisms. For an introduction to string diagrammatic syntax, we refer the reader to [21].
- In Section 3.2, we formalise the intended semantics as a *symmetric monoidal functor* into the symmetric monoidal category (SMC) of monotone relations with the Cartesian product.
- Finally, we equip the syntax with a partial order which is sound for the intended semantics. This order is given by a finite number of (in)equalities of the same type.

3.1 Syntax

We build on a line of research that has sought to give a formal treatment of graphical models of computation of varying expressive power within the unifying language of symmetric monoidal categories. More specifically, we rely on the notion of coloured product and permutations category (prop), a mathematical structure which generalises standard multisorted algebraic theories [5]. Formally, a *prop* is a strict symmetric monoidal category (SMC) whose objects are words of a finite alphabet and where the monoidal product $\oplus$ on objects is given by concatenation. Equivalently, it is a strict SMC whose objects are all monoidal products of a finite number of generating objects.

Our syntax is a prop $\mathsf{P}_{\mathcal{S}}$, freely generated from a *signature* $\mathcal{S} = (G, M)$: a pair of a finite set of objects G and a set M of morphisms $g : v \to w$, where v and w are words over G. There are two ways of describing the morphisms of the prop $\mathsf{P}_{\mathcal{S}}$ concretely. As terms of (G^*, G^*)-sorted syntax whose constants are elements of M and whose operations are the usual categorical composition $(-);(-) : \mathsf{P}_{\mathcal{S}}(u, v) \times \mathsf{P}_{\mathcal{S}}(v, w) \to \mathsf{P}_{\mathcal{S}}(u, w)$ and the monoidal product $(-) \oplus (-) : \mathsf{P}_{\mathcal{S}}(v_1, w_1) \times \mathsf{P}_{\mathcal{S}}(v_2, w_2) \to \mathsf{P}_{\mathcal{S}}(v_1 v_2, w_1 w_2)$, quotiented by the laws of SMCs. However, this quotient is cumbersome and unintuitive to work with.

This is why we prefer a different representation: with their two forms of composition, monoidal categories admit a natural two-dimensional notation of *string diagrams*. The idea is that a morphism $c : v \to w$ of P_S is better represented as a box with $|v|$ ordered wires labelled by the elements of v on the left and $|w|$ wires labelled by the elements of w on the right. We can compose these diagrams in two different ways: horizontally with ; by connecting the right wires of the first diagram to the left wires of the second (when the types match), and vertically with $\oplus$ by simply juxtaposing two diagrams: $c ; d =$  and $d_1 \oplus d_2 =$. Intuitively, morphisms of P_S can be pictured as (directed acyclic) graphs whose nodes are labelled by elements of M. The symmetry $\sigma_{a,b} : ab \to ba$ is drawn as a wire crossing which swaps the a-and b-wires, and the unit for $\oplus$, $id_0 : 0 \to 0$, as the empty diagram (we use 0 to denote the empty word). With this representation the laws of SMCs become diagrammatic tautologies.

In this work, we start with the same diagrammatic syntax as [20], which we call Syn_Σ. It is the free two-coloured prop freely generated by the objects and morphisms below.

- Two generating objects, ▶ (right) and ◀ (left), whose identities we will depict respectively as and .
- The following generating morphisms:

(1)

(2)

Morphisms of Syn_Σ are thus vertical and horizontal compositions of these generators, potentially including wire crossings. The direction of the arrows on the generators' wires denotes their type: for example,  represents an operation of type ▶→▶▶, while has type ◀▶→ ε (where ε denotes the empty list, *i.e.*, the unit for the monoidal product).

As in [20], we will use the generators (1) to represent NFA: , , allow us to encode states by gathering incoming and outgoing transitions, while the transitions themselves are encoded with . The remaining two generators, and , allow us to form feedback loops, with which we can encode iteration, as we will see in more details in Section 4. The , , , play another important role: they will allow us to construct simulation relations directly in our syntax.

The white generators , , in (2) are not used to build automata-diagrams, but will allow us to show that the diagrammatic simulation relations we construct do satisfy the required properties, and thereby allow us to prove that one automaton simulates another using purely (in)equational reasoning. As we will see next, these are not artificial syntactic operations, but emerge naturally out of the chosen semantics.

Note that [20] also contained dual generators , which are not needed here.

3.2 Semantics

In this section, we explain how to interpret the diagrams of the previous section as relations. We will formalise the intended semantics as a SMC, and the interpretation as a symmetric monoidal functor from the syntax to the semantics.

Contrary to [20], the relations in our target semantics are not between languages, but between elements of Ω, that is, equivalence classes of NFAs *up to similarity*. As in [20], each generator is interpreted as a *monotone* relation between posets.

▶ **Definition 9** (Monotone relation). *Given two posets $(X, \leq_X)$ and $(Y, \leq_Y)$, a relation $R : X \to Y$ is* monotone *whenever for $(x, y) \in R$, if $x' \leq_X x$, $y \leq_Y y'$ then $(x', y') \in R$.*

▶ **Proposition 10** (SMC of monotone relations). *Posets and monotone relations form a SMC* $\mathsf{Prof}_{\mathbb{B}}$ *with composition given by relational composition, where the identity for an object* $(X, \leq_X)$ *is the order relation* $\leq_X$ *itself, and with monoidal product the product of posets.*

Moreover, monotone relations of the same type can be ordered by inclusion, making $\mathsf{Prof}_{\mathbb{B}}$ *into an order-enriched SMC.*

The SMC $\mathsf{Prof}_{\mathbb{B}}$ of monotone relations has also appeared in the literature under the name of Boolean(-enriched) [8, 20] and relational [11] profunctors, or weakening relations [19].

Since Syn_Σ is freely-generated, to define a symmetric monoidal functor $[\![\cdot]\!] : \mathsf{Syn}_\Sigma \to \mathsf{Prof}_{\mathbb{B}}$, it is sufficient to specify the image of each generating object and morphism.

▶ **Definition 11** (Semantics). *Let* $[\![\cdot]\!]$ *be the following mapping.*

- *For the generating objects, let* $[\![\blacktriangleright]\!] = (\Omega, \geq)$ *and* $[\![\blacktriangleleft]\!] = (\Omega, \leq)$*. By Definition 9, this fixes the interpretation of the corresponding identities to be the order relations themselves:*

$$[\![\longrightarrow\!\!\!\rightarrow\!\!\!\longrightarrow]\!] = \{(X, Y) \mid Y \leq X\} \qquad [\![\longrightarrow\!\!\!\leftarrow\!\!\!\longrightarrow]\!] = \{(X, Y) \mid X \leq Y\}$$

- *We map generating morphisms to the following relations:*

$$\left[\!\!\left[\rightarrow\!\!\bullet\!\!\begin{array}{l}\rightarrow\\ \rightarrow\end{array}\right]\!\!\right] = \left\{\big(X, (Y_1, Y_2)\big) \mid Y_i \leq X,\, i = 1, 2\right\} \qquad [\![\rightarrow\!\!\bullet]\!] = \{(X, \bullet) \mid X \in \Omega\}$$

$$\left[\!\!\left[\begin{array}{l}\rightarrow\\ \rightarrow\end{array}\!\!\bullet\!\!\rightarrow\right]\!\!\right] = \left\{\big((X_1, X_2), Y\big) \mid Y \leq X_i,\, i = 1, 2\right\} \qquad [\![\bullet\!\!\rightarrow]\!] = \{(\bullet, Y) \mid Y \in \Omega\}$$

$$\left[\!\!\left[\begin{array}{l}\rightarrow\\ \leftarrow\end{array}\right]\!\!\right] = \{(\bullet, (Y_1, Y_2)) \mid Y_2 \leq Y_1\} \qquad \left[\!\!\left[\begin{array}{l}\leftarrow\\ \rightarrow\end{array}\right]\!\!\right] = \{((X_1, X_2), \bullet) \mid X_1 \leq X_2\}$$

$$[\![\!-\boxed{a}\!-]\!] = \{(X, Y) \mid a.Y \leq X\} \qquad (a \in \Sigma)$$

$$\left[\!\!\left[\rightarrow\!\!\circ\!\!\begin{array}{l}\rightarrow\\ \rightarrow\end{array}\right]\!\!\right] = \left\{\big(X, (Y_1, Y_2)\big) \mid Y_1 \times Y_2 \leq X, \right\} \qquad [\![\rightarrow\!\!\circ]\!] = \{(\top, \bullet)\}$$

A few remarks about the semantics are in order.

- The order relation is built into the identity wires $\longrightarrow\!\!\!\rightarrow\!\!\!\longrightarrow$ and $\longrightarrow\!\!\!\leftarrow\!\!\!\longrightarrow$. The two directions represent the identities on Ω ordered by $\geq := \{(X, Y) : Y \leq X\}$ and $\leq$ respectively. This is the opposite of the convention in [20], and is imposed by the use of prefixes (as opposed to suffixes in [20]) to interpret the atomic actions $-\boxed{a}-$, *cf.* last item of this list.
- The black primitives are standard monotone relations associated with any poset. One can see them as copy, delete, and their duals, relative to the relevant partial order relation. Similarly, the wire-bending primitives are interpreted as relations that exist for any poset, making use of the fact that our semantic category is compact-closed [12].
- The white generators are interpreted as the meet and top of the lattice structure obtained in Corollary 6. Contrary to [20], we do not need generators for the join of the lattice – this is one of the distinguishing features of language equivalence and simulation.
- The action of $-\boxed{a}-$ for each letter $a \in \Sigma$, relates $a.Y$ to any NFA X simulating it.

▶ **Proposition 12.** $[\![\cdot]\!]$ *extends to a symmetric monoidal functor* $\mathsf{Syn}_\Sigma \to \mathsf{Prof}_{\mathbb{B}}$*.*

Proof. Since Syn_Σ is freely generated, we just need to check that each generator is a *monotone* relation. The only non-immediate case is the monotony of $[\![\!-\boxed{a}\!-]\!]$, which follows from Proposition 4. ◀

This means that we can compute the semantics of arbitrary diagrams in a fully compositional way, using the following rules for composition and monoidal product:

$$[\![c\,;\, d]\!] = \left[\!\!\left[\overset{u}{-}\boxed{c}\overset{v}{-}\boxed{d}\overset{w}{-}\right]\!\!\right] = \left\{(\mathbf{X}, \mathbf{Z}) \mid \exists \mathbf{Y}\, (\mathbf{X}, \mathbf{Y}) \in [\![\!-\boxed{c}\!-]\!]\,, (\mathbf{Y}, \mathbf{Z}) \in [\![\!-\boxed{d}\!-]\!]\right\}$$

$$[\![c_1 \oplus c_2]\!] = \left[\!\!\left[\begin{array}{l}\overset{v_1}{-}\boxed{c_1}\overset{w_1}{-}\\ \underset{v_2}{-}\boxed{c_2}\underset{w_2}{-}\end{array}\right]\!\!\right] = \left\{\big((\mathbf{X}_1, \mathbf{X}_2), (\mathbf{Y}_1, \mathbf{Y}_2)\big) \mid (\mathbf{X}_i, \mathbf{Y}_i) \in [\![\!-\boxed{c_i}\!-]\!]\,,\, i = 1, 2\right\}$$

3.3 Inequational theory

In Figure 1 below we introduce MDA, the theory of *Milner Diagram Algebra*, a set of axioms which will prove complete for simulation of NFA (once we have shown how to encode them into the diagrammatic syntax). Some background on symmetric monoidal (inequality) theories can be found in Appendix B.

Unsurprisingly MDA is close to the existing diagrammatic axiomatisation of NFA *up to language equivalence* of [20, Section 3] and we use the same naming scheme to highlight the similarities and differences between the two. There are two main differences: the lack of  , and and (E1-E2) which only hold *laxly* here, witnessing the simulation $a.b+a.c \preceq a.(b+c)$. This is akin to how regular expressions up to (bi)similarity do not satisfy left-distributivity of composition over the sum, but still satisfy right-distributivity (E3-E4).

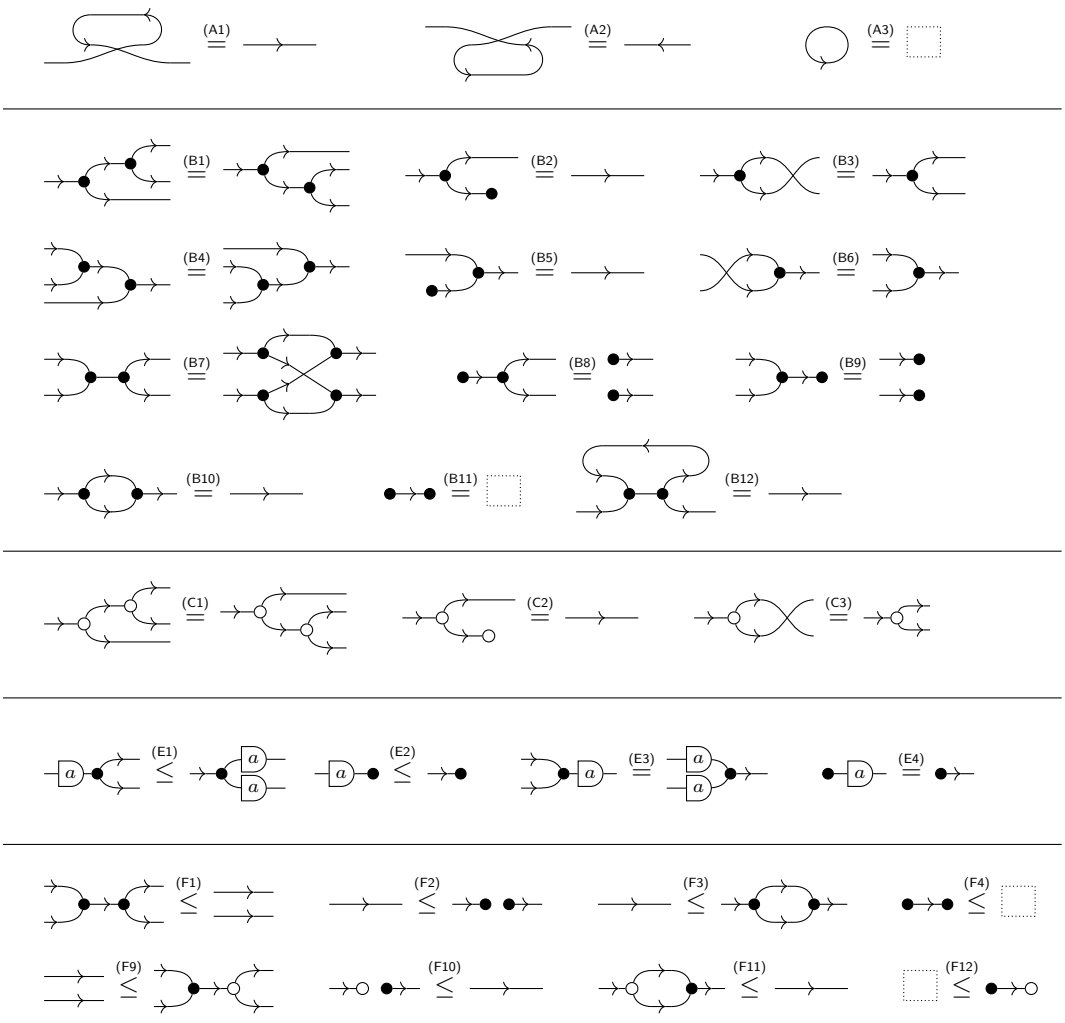
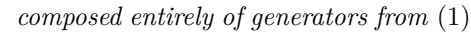


Figure 1 Theory of Milner Diagram Algebra (MDA).

▶ Remark 13. Note that this theory is not minimal: for example, the lax distributivity axiom (E1) can be proven using (F1), (B10) and (E3). A similar property holds for (E2). However, we have preferred to keep them since they highlight the main difference with previous work on language equivalence/inclusion [20].

As we will explain more thoroughly in Section 4, we are interested in the properties of diagrams that are closely related to NFA. We identify these in the following definition.

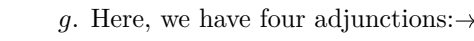
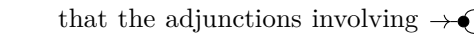
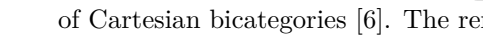
▶ **Definition 14** (Automaton-diagram). *We call* automaton-diagram *any diagram of* Syn_Σ *composed entirely of generators from* (1)*, namely*  $(a \in \Sigma)$ *In other words, automata-diagrams are the morphisms of the sub-prop freely generated by the morphisms of* (1)*. We call this sub-prop* Aut_Σ*.*

We can now state the main result of this paper.

▶ **Theorem 15** (Soundness and Completeness). *For any two automata-diagrams* $c, d : \blacktriangleright \to \blacktriangleright$*,*

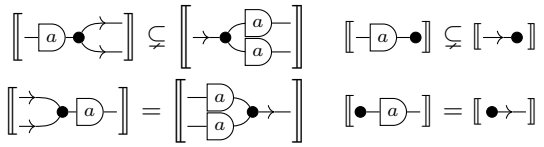
$[\![c]\!] \subseteq [\![d]\!]$ *if and only if* $c \leq_{MDA} d$*.*

The *completeness* of MDA is the most involved and will be the subject of Section 5. Its *soundness* on the other hand is straightforward and is a matter of verifying that each axiom holds in the semantics. Except for axioms of the E block (see Proposition 16), the soundness of the remaining (in)equalities follows from properties which have been proven earlier:

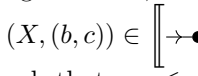
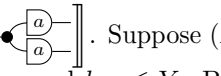
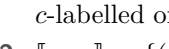
- The A and B blocks encode equalities that hold for any poset. They imply that our category of interest is compact-closed (A) and that every object is equipped with a (bi)commutative bimonoid (B1-B11), a common structure in diagrammatic calculi [16].
- The E block encodes the interaction of the atomic actions with the simulation order. As we already stated, here lies the main difference with [20] – (E1-E2) now only hold laxly.
- The C and F blocks encode the lattice structure of $(\Omega, \leq)$, and all (in)equalities follow from the existence of meets. The F block in particular encodes a number of adjunctions in the following 2-categorical sense: two morphisms $f : X \to Y$ and $g : Y \to X$ are *adjoints* if $\mathrm{id}_X \leq f\,;g$ and $g\,;f \leq \mathrm{id}_Y$. We write $f \dashv g$ and say that f is left adjoint to g. Here, we have four adjunctions:  and  . Note that the adjunctions involving  are the key defining adjunctions of Cartesian bicategories [6]. The remaining adjunctions hold whenever the supporting poset is a semi-lattice (has binary meets and top), which is the case for simulation.

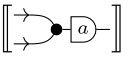
We prove here the soundness of axioms (E1-E4) and that the inequalities corresponding to axioms (E1-E2) are strict.

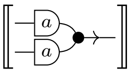
▶ **Proposition 16.** *We have the following (in-)equalities:*



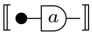
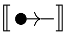
Proof.

1. The proof of the inclusion is straightforward, but the fact that it is strict is more interesting. Take $X = a.b + a.c \in \Omega$. Then $(X, (b, c)) \in$  . Suppose $(X, (b, c)) \in$  too. Then there is a $Y \in \Omega$ such that $a.y \leq x$, and $b, c \leq Y$. By the join property of $+$, we have $b + c \leq Y$ and therefore $a.(b + c) \leq X$. However $a.b + a.c$ does not simulate $a.(b + c)$ since there is no state in $a.b + a.c$ having a b-labelled out-transition as well as a c-labelled one. Thus $(X, (b, c)) \notin$.
2. $[\![\to \bullet]\!] = \{(X, \bullet) : X \in \Omega\} \supseteq \{(a.X, \bullet) : X \in \Omega\} = [\![$  $]\!]$, and it is clear that for all $X \in \Omega$, $0 \neq a.X$ because 0 has no transitions.

3. We have $[\![$  $]\!] = \{((X_1, X_2), Y) : a.Y \leq X_1, X_2\}$ and

$[\![$  $]\!]$ $$= \left\{ ((X_1, X_2), Y) : \exists U_1, U_2, \begin{array}{r} Y \leq U_1, U_2 \\ a.U_1 \leq X_1 \\ a.U_2 \leq X_2 \end{array} \right\}$$

The "$\supseteq$" inclusion follows from the monotony of prefixing, which gives us $a.Y \leq a.U_i \leq X_i$ for $i = 1, 2$. For the other inclusion, choosing $U_1 = U_2 = Y$ gives the desired result.

4. $[\![$  $]\!] = \{(\bullet, Y) : \exists X (a.Y \leq X)\}$ is clearly contained in $\{(\bullet, Y) : Y \in \Omega\} = [\![$  $]\!]$; for the other direction, we can just set $X = a.Y$. ◀

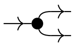
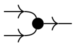
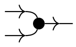
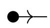
4 Automata-diagrams

This section aims to explain and justify our representation of NFA using string diagrams. Our encoding follows that of [20] – the aim of this section is not merely to reproduce it, but to show that the same syntax is able to represent automata uniformly for different semantics, and to define the notions we will use in the proof of completeness.

- In Section 4.1, we first show that the same encoding is sound for the simulation semantics, in the sense that the diagram c_A which represents a given NFA A has the appropriate semantics, *i.e.*, $[\![c_A]\!] = \{(X, Y) : A.Y \leq X\}$. This is Theorem 23 below.
- In Section 4.2, we prove a converse result: any automaton-diagram $\blacktriangleright \to \blacktriangleright$ can be thought of as a NFA, *i.e.*, for any automaton diagram c, there exists some NFA A_c such that $[\![c]\!] = \{(X, Y) : A_c.Y \leq X\}$. This is Corollary 27 below.

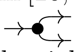
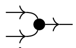
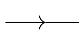
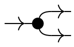
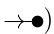
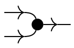
4.1 From automata to string diagrams...

First, we show how to encode relations into our calculus. We will use these to represent the initial and final states of NFA, and simulation relations in the proof of completeness.

▶ **Definition 17** (Relation-diagram). *A* relation-diagram *is a diagram of* Syn_Σ *composed entirely of*  ,  ,  ,  . *We call* RelDiag *the corresponding sub-prop. A relation-diagram is* functional *when it is composed only of*  ,  .

In what follows, we call *block* of a certain subset of the generators of Syn_Σ, any diagram made up entirely of the prescribed generators using vertical and horizontal composition (and, possibly, identities and wire crossings).

▶ **Proposition 18** (Encoding relations). *Given two finite sets* $Q = \{q_i\}_{1 \leq i \leq n}$ *and* $S = \{s_j\}_{1 \leq j \leq m}$, *and some relation* $R \subseteq Q \times S$, *there exists a relation diagram* $d_R : \blacktriangleright^n \to \blacktriangleright^m$ *such that* $[\![d_R]\!] = \{(\mathbf{X}, \mathbf{Y}) \mid X_i \geq Y_j$ *if* $(q_i, s_j) \in R\}$.

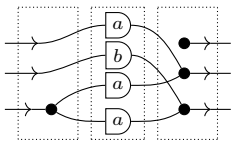
Proof. We reproduce the construction from [20, Section 4]. The relation-diagram d_R $\blacktriangleright^n \to \blacktriangleright^m$ is formed of two blocks: first, a block of  ,  , followed by a block of  ,  . The left i-th port of d_R is connected to the the right j-th port through an identity wire  precisely when $(q_i, s_j) \in R$; we use  to accommodate multiple outgoing connections from a single left port (or none, with ), and  for multiple incoming connections into a right port (or none, with ).

Finally, it is straightforward to see that we have constructed d_R so that its semantics is precisely $\{(\mathbf{X}, \mathbf{Y}) \mid X_i \geq Y_j$ if $(q_i, s_j) \in R\}$. ◀

We now show how to encode the transition relation of any given automaton.

▶ **Definition 19** (Matrix-diagram). *A* matrix-diagram *is a diagram* $\blacktriangleright^n \to \blacktriangleright^m$ *that factors as a composition of a block of*  , , *another of* *for* $a \in \Sigma$, *and a last one of* , , *such that any path from a left to right port encounters exactly one* .

The intuition behind the condition of the previous definition are to *(i)* disallow multiple transitions for the same letter between the same two states, and *(ii)* disallow ϵ-transitions since our definition of NFA does not allow them. For example, the following is is a well-formed matrix-diagram (with the three blocks highlighted), encoding the transition relation $\{(q_0, a, q_1), (q_1, b, q_2), (q_2, a, q_1), (q_2, a, q_2)\}$.

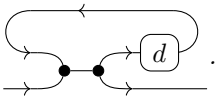


Proposition 20 shows that we can encode any transition relation as a matrix-diagram with the desired semantics.

▶ **Proposition 20** (Encoding transition-relations). *Given a set* $Q = \{q_i\}_{1 \leq i \leq n}$ *and transition relation* $\alpha \subseteq Q \times \Sigma \times Q$, *write* $\mathbf{D}$ *for the corresponding* $n \times n$ *matrix of coefficients in* $\mathcal{P}_f(\Sigma)$. *There exists a matrix-diagram* $d_\alpha : \blacktriangleright^n \to \blacktriangleright^n$ *such that* $[\![d_\alpha]\!] = \{(\mathbf{X}, \mathbf{Y}) : \mathbf{DY} \leq \mathbf{X}\}$.

Proof. Again, the idea is already present in [20, Section 4.2]. Let d_α be the matrix-diagram where the i-th input wire is linked to the j-th output wire via whenever $q_i \xrightarrow{a}_A q_j$. By construction, $[\![d_\alpha]\!] = \{(\mathbf{X}, \mathbf{Y}) : \mathbf{DY} \leq \mathbf{X}\}$, as we wanted. ◀

The last ingredient of the correspondence between automata-diagrams and NFA is iteration. As explained above, and allow us to form the diagrammatic counterpart of iteration, aka the Kleene star, defined as follows.

▶ **Definition 21** (Star). *For a diagram* $d : \blacktriangleright^n \to \blacktriangleright^n$, *let* :=  .

▶ **Proposition 22** (Stars are least fixpoints). *If* $d : \blacktriangleright^n \to \blacktriangleright^n$ *is a matrix-diagram whose corresponding matrix is* $\mathbf{D}$, *then* $[\![d^*]\!] = \{(\mathbf{X}, \mathbf{Y}) : S(\mathbf{Y}) \leq \mathbf{X}\}$, *where* $S(\mathbf{Y})$ *is the (unique) least solution of the system* $\mathbf{Y} + \mathbf{DX} \leq \mathbf{X}$.

Proof. The semantics of d^* is given by $\{(\mathbf{X}, \mathbf{Y}) : \mathbf{Y} + \mathbf{DX} \leq \mathbf{X}\}$, which is equal to $\{(\mathbf{X}, \mathbf{Y}) : S(\mathbf{Y}) \leq \mathbf{X}\}$ by Theorem 8. ◀

▶ **Theorem 23** (Encoding of NFA). *Let* $A = (Q, q_0, F, \alpha)$ *be an automaton. There exists an automaton-diagram* $c_A : \blacktriangleright \to \blacktriangleright$ *such that* $[\![c_A]\!] = \{(X, Y) : A.Y \leq X\}$.

Proof. We represent automata as in [20, Section 4.2]. First, fix some ordering of $Q = \{q_i\}_{1 \leq i \leq n}$. Then

- $e : \blacktriangleright \to \blacktriangleright^n$ is the relation-diagram encoding the singleton $\{(q_0, \bullet)\}$, using Proposition 18;
- $f : \blacktriangleright^n \to \blacktriangleright$ is the relation-diagram encoding the $\{(q_i, \bullet) : q_i \in F\}$, using Proposition 18;
- $d : \blacktriangleright^n \to \blacktriangleright^n$ is the matrix-diagram representing α, using Proposition 20. It is such that $[\![d]\!] = \{(\mathbf{X}, \mathbf{Y}) : \mathbf{DY} \leq \mathbf{X}\}$.

Then, let $c_A = e; d^*; f$ where d^* is defined as in Definition 21. By Proposition 22, $[\![d^*]\!] = \{(\mathbf{X}, \mathbf{Y}) : S(\mathbf{Y}) \leq \mathbf{X}\}$. Then we have:

$$[\![c_A]\!] = \left\{ (X, Y) : \exists \mathbf{X}, \mathbf{Y}, \begin{array}{c} \forall i, q_i \in F \Rightarrow \mathbf{Y}_i \leq Y \\ S(\mathbf{Y}) \leq \mathbf{X} \\ \mathbf{X}_0 \leq X \end{array} \right\}$$

We want to show that $[\![c_A]\!]$ is equal to $\{(X, Y) : A.Y \leq X\}$ by double-inclusion.

(⊆) Let $X, Y \in \Omega$ be such that $A.Y \leq X$. Then take $\mathbf{Y} = \mathbf{F}.Y$ and $\mathbf{X} = S(\mathbf{Y})$. By definition, if $q_i \in F$ then $\mathbf{Y}_i \leq Y$, and $S(\mathbf{Y}) \leq \mathbf{X}$.
Moreover $S(\mathbf{Y}) = S(\mathbf{F}.Y) = \mathbf{A}.Y$ by Theorem 8. Thus $\mathbf{X}_0 = (S(\mathbf{Y}))_0 = (\mathbf{A}.Y)_0 = A_0.Y = A.Y \leq X$, and therefore $(X, Y) \in [\![c_A]\!]$.

(⊇) Let $(X, Y) \in [\![c_A]\!]$ and $\mathbf{X}, \mathbf{Y}$ be two vectors of behaviours witnessing this membership. To conclude the proof of the second inclusion, we have $A.Y = (\mathbf{A}.Y)_0 = (S(\mathbf{F}.Y))_0 \leq (S(\mathbf{Y}))_0 \leq \mathbf{X}_0 \leq X$ where the third step is our hypothesis.

Finally, we need to show that this representation is independent of the choice of ordering of Q. This is the case, because any other choice of total order induces a permutations of the wires. In other words, given another ordering, from which we obtain two other relation-diagrams e' and f' encoding initial and final states, and a matrix-diagram d' encoding the transition relation of A, there exists some permutation $\pi : \blacktriangleright^n \to \blacktriangleright^n$ such that $d' = \pi\,;\, d\,;\, \pi^{-1}$, $e' = e\,;\, \pi^{-1}$, and $f' = \pi\,;\, f$. A simple diagrammatic derivation shows that $d'^* = \pi\,;\, d^*\,;\, \pi^{-1}$ and therefore $e'\,;\, d'^*\,;\, f' = e\,;\, d\,;\, f$. ◀

As a result, by soundness, an inequality between two automata-diagrams implies the existence of a simulation between the corresponding NFA *in the reverse order*:

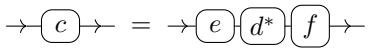
▶ **Corollary 24.** *Let A, B be two NFA and c_A, c_B the corresponding automaton-diagram obtained from Theorem 23. If $c_A \leq c_B$, then $B \leq A$.*

Proof. The proof is straightforward: by soundness, $c_A \leq c_B$ means $[\![c_A]\!] \subseteq [\![c_B]\!]$. Since $A.1 = A \leq A$, we have $(A, 1) \in [\![c_A]\!]$. Thus $(A, 1) \in [\![c_B]\!]$, *i.e.*, $1.B = B \leq A$. ◀

4.2 ...and back

Theorem 23 shows how to encode a given NFA as a string diagram into our diagrammatic syntax. Conversely, given a $\blacktriangleright \to \blacktriangleright$ automaton-diagram, we can extract the NFA it represents, by rewriting it into a form which mimics the encoding of NFA of the previous section – this is what we call a *representation*.

▶ **Definition 25** (Representation)**.** *For a diagram $c : \blacktriangleright \to \blacktriangleright$, a* representation *is a triple (e, d, f) of a matrix-diagram, $d : \blacktriangleright^\ell \to \blacktriangleright^\ell$, one functional relation-diagram $e : \blacktriangleright \to \blacktriangleright^\ell$, and one relation-diagram $f : \blacktriangleright^\ell \to \blacktriangleright$, such that*



The intuition is that d represents the transition relation of the associated automaton, e the initial state, and f its final states.

▶ **Proposition 26.** *Any automaton-diagram $\blacktriangleright \to \blacktriangleright$ has a representation.*

Proof. The proof is the same as [20, Proposition 4.7]. All axioms used in this proof are in MDA – crucially, it does not use left-distributivity, but only right-distributivity (E3-E4). ◀

From here it is easy to extract a diagrammatic representation of its initial state, final states, and transition relations.

▶ **Corollary 27** (NFA from automaton-diagram). *Given an automaton-diagram $c : \blacktriangleright \to \blacktriangleright$, there exists a NFA A such that $[\![c]\!] = \{(X, Y) \mid A.Y \leq X\}$.*

Proof. First, by Proposition 26, we can find a representation (e, d, f) of c. We construct $A = (Q, q_0, F, \alpha)$. First, if $d : \blacktriangleright^n \to \blacktriangleright^n$, let $Q = \{1, \ldots, n\}$. Then let q_0 be the only i such that $(\bullet, i) \in \mathfrak{R}(e)$ (remember that $\mathfrak{R}(e)$ is a $n \times 1$ Boolean matrix, which is moreover functional, so that it is fully characterised by a single i between 1 and n). Let $F := \{j : (j, \bullet) \in \mathfrak{R}(f)\}$. Finally, the transition relation is determined by the matrix-diagram d. Call $\mathbf{D}$ be the $n \times n$ matrix with coefficients in $\mathcal{P}_f(\Sigma)$ obtained from Proposition 33. Then, let $(i, a, j) \in \alpha$ if $\mathbf{D}_{i,j}$ contains a (remember that the coefficients are finite subsets of Σ). This is well-defined because d is assumed to be ϵ-free. ◀

5 Completeness

The main idea to tackle completeness is that simulation relations themselves can be encoded as relation-diagrams into our calculus.

We have already shown how to encode relations as relation-diagrams; now we explain how to go in the other direction. From any relation-diagram $d : \blacktriangleright^n \to \blacktriangleright^m$ we can obtain a relation between $\{1, \ldots, n\}$ and $\{1, \ldots, n\}$, *i.e.*, a matrix with Boolean coefficients. As we will need to manipulate these relations in calculations below, we formalise the correspondence between relation-diagrams and relations as a *functor* from RelDiag (the sub-prop freely generated by these diagrams) to $\mathsf{Mat}_{\mathbb{B}}$, the SMC of Boolean matrices with the disjoint sum as monoidal product. The latter has natural numbers as objects and $m \times n$ matrices with Boolean coefficients as morphisms $n \to m$. Its morphisms can also be ordered by inclusion if we think of them as relations: given two Boolean $n \times m$ matrices $A = (a_{ij})$ and $B = (b_{ij})$, we write $A \leq B$ if $a_{ij} \leq b_{ij}$ for all i and j.

▶ **Definition 28.** *Let $\mathfrak{R}(\cdot)$ be the mapping given by:*

$$\mathfrak{R}\left(\to\bullet\circlearrowright\right) = \begin{pmatrix}1\\1\end{pmatrix} \qquad \mathfrak{R}(\to\bullet) = 1 \qquad \mathfrak{R}\left(\begin{smallmatrix}\to\\\to\end{smallmatrix}\bullet\to\right) = \begin{pmatrix}1 & 1\end{pmatrix} \qquad \mathfrak{R}(\bullet\to) = 1$$

By the freeness of RelDiag, we obtain the following result immediately.

▶ **Proposition 29.** *$\mathfrak{R}(\cdot)$ extends to a symmetric monoidal functor* RelDiag $\to \mathsf{Mat}_{\mathbb{B}}$.

We will use extensively the fact that MDA is complete for relation-diagrams.

▶ **Theorem 30** (Completeness for relation-diagrams). *If c, d are two relation-diagrams, then $c \leq_{MDA} d$ iff $\mathfrak{R}(d) \leq \mathfrak{R}(c)$.*

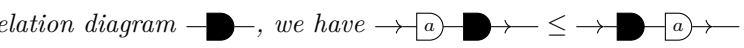
Proof. This is a standard completeness result for the symmetric monoidal theory of an idempotent (co)commutative bimonoid [7, Theorem 7.2], *i.e.* axioms (B1)-(B11). While it is usually stated for equalities, the extension to inequalities is straightforward, since inequalities can be recovered from the semi-lattice structure of the binary operation defined by $\to\bullet\circlearrowright$ and $\begin{smallmatrix}\to\\\to\end{smallmatrix}\bullet\to$, that is we can show that $c \leq d$ iff $\to\bullet\begin{smallmatrix}\to c \to\\\to d \to\end{smallmatrix}\bullet\to = c$. See [20, Propositions 5.2-5.3] for the detailed proof. ◀

We will need the fact that we can (co)copy and (co)delete any relation-diagram.

▶ **Lemma 31** (Distributivity for relation-diagrams). *Any relation-diagram* $d : \blacktriangleright^m \to \blacktriangleright^n$ *satisfies*

Proof. Since the corresponding equalities hold in $\mathsf{Mat}_{\mathbb{B}}$, we can deduce the two syntactic equalities from Theorem 30 (completeness for relations). ◀

We will need a particularly simple form of simulation below for a single letter.

▶ **Lemma 32.** *For any relation diagram*  , *we have* 

Proof. By structural induction. The two inductive cases for composition and monoidal product are straightforward. The base cases are the axioms (E1-E4). ◀

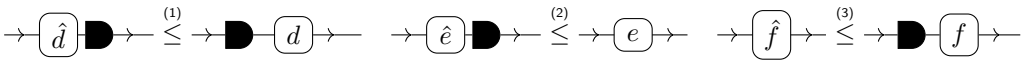
In the previous section, we have used matrix-diagrams (Definition 19) to encode the transition relations of NFA (Proposition 20). Clearly, we can also go the other way, associating a unique transition relation δ to each matrix-diagram.

▶ **Proposition 33.** *For any matrix-diagram* $d : \blacktriangleright^n \to \blacktriangleright^n$, *there exists a* $n \times n$ *matrix* $\mathbf{D}$ *with coefficients in* $\mathcal{P}_f(\Sigma)$ *such that* $[\![d]\!] = \{(\mathbf{X}, \mathbf{Y}) : \mathbf{DY} \leq \mathbf{X}\}$.

Proof. This is obvious from the way matrix-diagrams are defined. We can obtain the (i, j)-th coefficient of $\mathbf{D}$ by plugging $\bullet\!\!\rightarrow$ and $\rightarrow\!\!\bullet$ into all other left and right ports of d. ◀

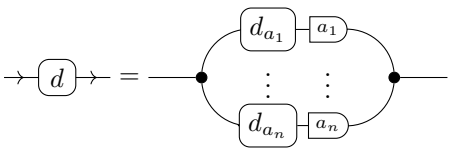
We can now show that the relation-diagram encoding a given simulation does satisfy the diagrammatic analogues of the three defining properties of simulation from Definition 2.

▶ **Lemma 34** (Simulation for representations). *Given two NFA* A *and* $\hat{A}$, *let* (e, d, f) *and* $(\hat{e}, \hat{d}, \hat{f})$ *be the representations associated to their respective encoding as automata-diagrams. If* $A \leq \hat{A}$, *there is a relation-diagram*  *such that:*

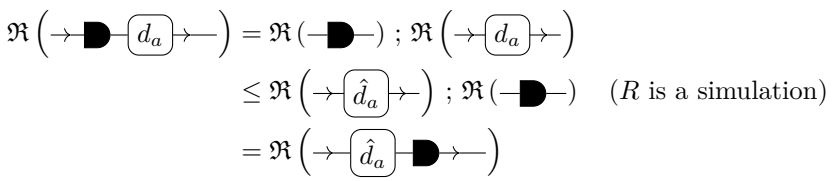


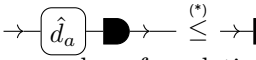
Proof. Assume that R is a simulation relation witnessing $A \leq \hat{A}$ and let be the relation-diagram encoding R^{-1}, using Proposition 18 with the ordering of the states of A and $\hat{A}$ already fixed by the choice of the representations in the statement of the lemma. Let us prove satisfies the required inequalities.

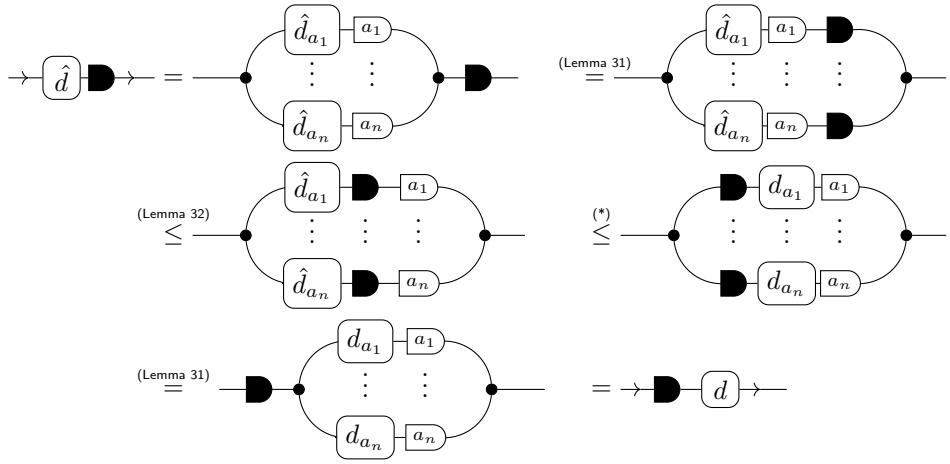
(1) For $a \in \Sigma$, let d_a be relation-diagram encoding the relation $\xrightarrow{a}_A$. Then, if $\Sigma = \{a_1, \ldots, a_n\}$, it is easy to check that



by applying the (E3-E4) axioms to merge letters, as well as the (co)unit axiom for the black (co)monoid (recall that, as a relation-diagram can be described as a block of  ,  composed with a block of  , ). Naturally, this also holds for $\hat{d}$.
Then, we show that behaves like a simulation for all d_a:



By completeness for relations (Theorem 30), we get the reverse syntactic inequality:  $\overset{(*)}{\leq}$. Finally, putting it all together and using copy and merge laws for relations, we have:



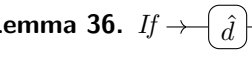
(2-3) Those two inequalities are just straightforward applications of completeness for relations and the definition of simulation. ◀

The first inequality of the previous lemma is insufficient, because automata-diagrams factor as $e; d^*; f$, not $e; d; f$, for a given representation. The rest of the proof is thus dedicated to lifting Lemma 34(1) to d^* instead of just d, using a proof similar to that of [20, Section 5] (of which we only sketch the main ideas). Crucially, this is where the white generators  , , and their associated axioms come into play.

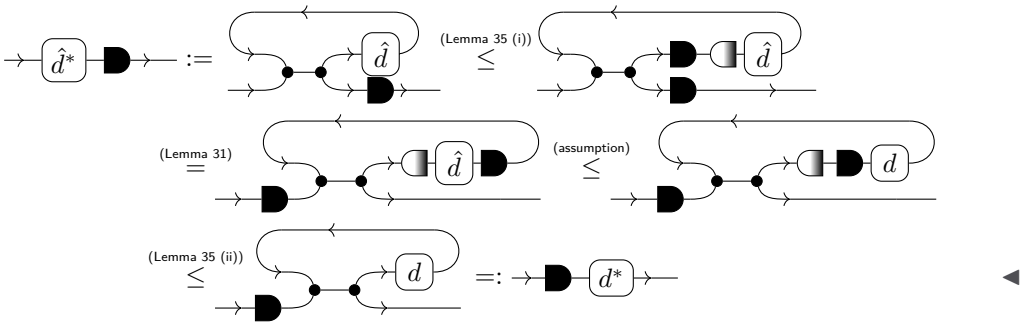
First, we build a right adjoint  to a given simulation relation(-diagram) using , (see [20, Section 5.2] for the details). Then, the F axioms of MDA are enough to show the necessary adjunction: $\dashv$, this is [20, Lemma 5.14].

▶ **Lemma 35.** *For any relation-diagram* *, there exists a diagram* *, such that the following inequalities hold: (i)* $\overset{m}{\longrightarrow} \leq$ *and (ii)* $\leq \overset{n}{\longrightarrow}$.

We can now lift the simulation relation to d^* as we wanted.

▶ **Lemma 36.** *If*  *then* .

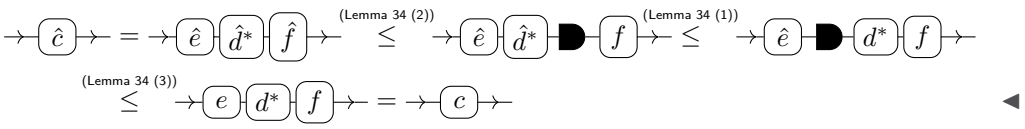
Proof. We have



◀

We are finally ready to finish our proof of the completeness of MDA.

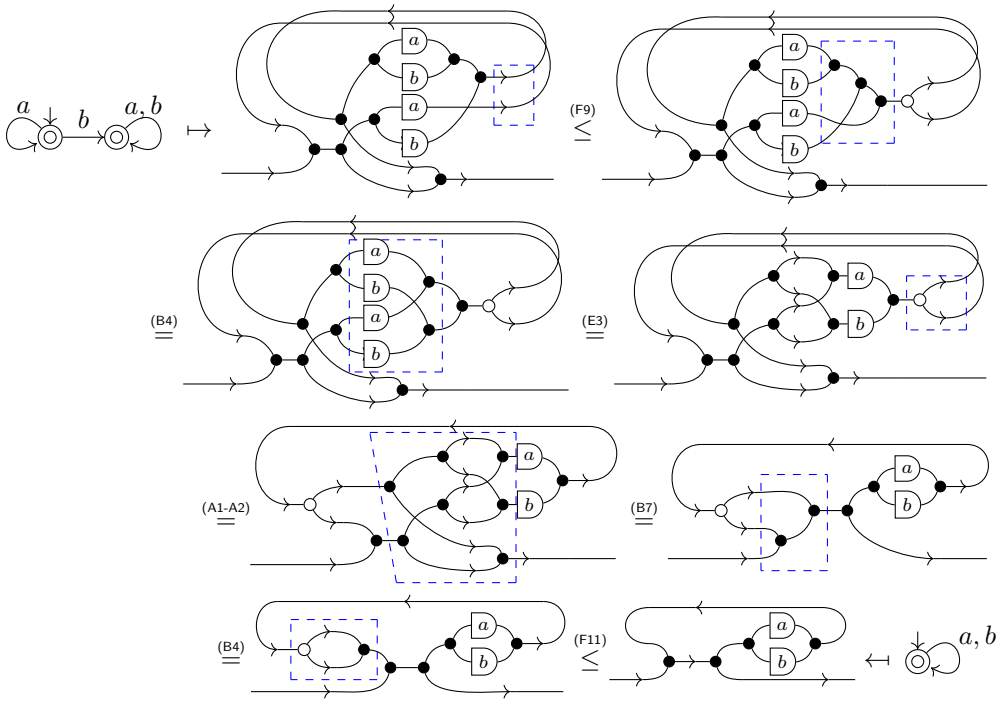
Proof of Theorem 15 (completeness). Given two automata-diagrams $c, \hat{c} : \blacktriangleright \to \blacktriangleright$, we can first extract their respective representations (e, d, f) and $(\hat{e}, \hat{d}, \hat{f})$, using Proposition 26. From Corollary 27, we can recover two a NFA A and $\hat{A}$ such that $[\![c]\!] = \{(X, Y) \mid A.Y \leq X\}$ and $[\![\hat{c}]\!] = \{(X, Y) \mid \hat{A}.Y \leq X\}$. If we also assume, as in the statement of the theorem, that $[\![\hat{c}]\!] \subseteq [\![c]\!]$, then, since $\hat{A}.1 = \hat{A} \leq \hat{A}$, we have $(\hat{A}, 1) \in [\![\hat{c}]\!]$. Thus also $(\hat{A}, 1) \in [\![c]\!]$, *i.e.*, $A.1 = A \leq \hat{A}$. In other words, $\hat{A}$ simulates A. Given such a simulation, we can encode it as a relation diagram —●— and conclude the proof using Lemma 34 as follows:



◀

▶ Remark 37 (Completeness for arbitrary automata-diagrams). The results in this paper are stated for $\blacktriangleright \to \blacktriangleright$ automata-diagrams, which correspond precisely to NFA (*cf.* Section 4). It is natural to wonder whether they extend to automata-diagrams of arbitrary type. The short answer is yes. First of all, we can always bend the wires of any given automaton-diagram $d : v \to w$ to obtain one of type $\blacktriangleright^n \to \blacktriangleright^m$. Semantically, this only amounts to changing whether a given variable appears on the left or on the right of the relation $[\![d]\!]$ [20, Proposition 5.4]. Second, it is possible to show [20, Theorem 5.5] that any automaton-diagram $\blacktriangleright^n \to \blacktriangleright^m$ distributes over  and —●—. As a result, any automaton-diagram $\blacktriangleright^n \to \blacktriangleright^m$ is fully characterised by n diagrams of type $\blacktriangleright \to \blacktriangleright^m$. Now, contrary to [20], automata-diagrams $\blacktriangleright^n \to \blacktriangleright^m$ in this paper do *not* distribute over —●  and —●— in general (they do so only laxly). This means that we cannot reduce the completeness for automata-diagrams $\blacktriangleright^n \to \blacktriangleright^m$ to that of automata-diagrams $\blacktriangleright \to \blacktriangleright$. However, it is possible to define a notion of NFA with multiple sets of final states and to adapt the definition of simulation correspondingly: if $(q, s) \in R$ for some simulation relation R between two such NFA A and B, each with m sets of final states $\{F_i\}_{1 \leq i \leq m}$ and $\{G_i\}_{1 \leq i \leq m}$, then we have $F_i(q) \leq G_i(s)$ for all $1 \leq i \leq m$. Then, all results of this paper generalise to automata-diagrams $\blacktriangleright \to \blacktriangleright^m$ (with multiple right ports, but only one left port) and we can prove that MDA is complete for all automata-diagrams. While this is a stronger result, we have preferred to state our main result for the class of diagrams that correspond more closely to plain NFA, as the more general notion is not standard and introduces distracting complications.

▶ **Example 38.** We now come back to the example from the introduction and show how to prove the similarity of the two NFA.



The other inequality can be proven entirely analogously, replacing  with  , axiom (F9) with (F1), and axiom (F11) with (F3).

6 Conclusion

In this paper, we have successfully provided a finite axiomatisation of NFA modulo similarity, using a string diagrammatic syntax.

Related work. In doing so, we have built on an earlier diagrammatic axiomatisation of NFA *up to language equivalence* [20]. We have shown here that the same syntax is able to accommodate a different semantics and can be axiomatised with a slight change of (in)equational theory: left-distributivity of —[a]— over  and is now lax. This change reflects the well-known fact that simulation implies language-inclusion. Another interesting corollary is that, *for deterministic automata(-diagrams)*, lax left-distributivity is sufficient to prove language-inclusion (*i.e.*, if we can show that $c \leq d$ using the axioms of [20], we can show it using only those of MDA).

Our axiomatisation is also closely related to an existing axiomatisation of regular CCS expressions up to similarity [9]. That work builds on Milner's axiomatisation of *bisimilarity*, adding the axiom $E \leq E+F$, an axiom which is derivable in our calculus. The main difference with our work is that this axiomatisation contains implicational axioms for fixpoints. In contrast, our axiomatisation is *finite*, as was the case in our earlier work on language equivalence [20]. We were able to achieve this by using a more expressive diagrammatic calculus, in which we can encode not only the simulation relations themselves, but the proof that they satisfy the desired properties.

Future work. First, we would like to characterise the expressiveness and give a complete axiomatisation of the full syntax (including the white generators) for the simulation semantics.

Second, we want to axiomatise the same syntax up to *bisimilarity*. Recall that a relation R is a bisimulation between two NFA when both R and R^{-1} are simulations. The present completeness result implies it is possible to check "by hand" that a relation-diagram encoding R witnesses the bisimulation between two automata-diagrams c and d: we have to prove that $c \leq d$ and $d \leq c$, as in the proof of Theorem 15, using R in one direction and R^{-1} in the other. However, this requires us to keep track of which simulation relation we use in each direction, since the relation $\leq$ itself omits this crucial piece of information.

Excitingly, this paves the way for a 2-categorical approach to the theory of bisimilarity, a perspective which would allow us to track the way in which two diagrams are related explicitly: for c, d two automata-diagrams and r a relation-diagram, $r : c \Rightarrow d$ is a 2-morphism whenever r is a simulation. We aim to show that this defines a (symmetric monoidal) 2-category and find a presentation for it, using 2-morphisms to replace the axioms of MDA and further equalities between them. Since bisimulations would be 2-isomorphisms in this setting, such a presentation would allow us to construct them as 2-morphisms and prove that they are invertible and satisfy the required properties using the additional equalities.

Finally, we would like to translate the axioms of MDA into transformations of the state-transition graph of the corresponding automata, building on the extensive work on formulating string diagram rewriting as rewriting of hypergraphs [3, 4].

References

1 John Baez, Brandon Coya, and Franciscus Rebro. Props in network theory. *Theory and Applications of Categories*, 33(25):727–783, 2018.

2 Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.

3 Filippo Bonchi, Fabio Gadducci, Aleks Kissinger, Paweł Sobociński, and Fabio Zanasi. Rewriting modulo symmetric monoidal structure. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, 2016.

4 Filippo Bonchi, Fabio Gadducci, Aleks Kissinger, Paweł Sobociński, and Fabio Zanasi. Rewriting with Frobenius. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 165–174, 2018.

5 Filippo Bonchi, Paweł Sobociński, and Fabio Zanasi. Deconstructing Lawvere with distributive laws. *Journal of logical and algebraic methods in programming*, 95:128–146, 2018. doi:10.1016/J.JLAMP.2017.12.002.

6 Aurelio Carboni and RFC Walters. Cartesian bicategories I. *Journal of pure and applied algebra*, 49(1-2):11–32, 1987.

7 Brandon Coya and Brendan Fong. Corelations are the prop for extraspecial commutative frobenius monoids. *Theory and Applications of Categories*, 32(11):380–395, 2017.

8 Brendan Fong and David I Spivak. *An invitation to applied category theory: seven sketches in compositionality*. Cambridge University Press, 2019.

9 Ulrik Frendrup and Jesper Nyholm Jensen. *A complete axiomatization of simulation for regular CCS expressions*. BRICS, Department of Computer Science, Univ., 2001.

10 CAR Hoare, Bernhard Möller, Georg Struth, and Ian Wehrman. Concurrent Kleene algebra. In *Proceedings of the 20th International Conference on Concurrency Theory (CONCUR)*, pages 399–414. Springer, 2009.

11 Martin Hyland and Andrea Schalk. Glueing and orthogonality for models of linear logic. *Theoretical Computer Science*, 294(1-2):183–231, 2003. doi:10.1016/S0304-3975(01)00241-9.

12 Gregory M Kelly and Miguel L Laplaza. Coherence for compact closed categories. *Journal of Pure and Applied Algebra*, 19:193–213, 1980.

13 Stephen C Kleene. Representation of events in nerve nets and finite automata. Technical report, RAND PROJECT AIR FORCE SANTA MONICA CA, 1951.

14 Dexter Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Information and Computation*, 110(2):366–390, 1994. doi:10.1006/INCO.1994.1037.

15 Dexter Kozen. Kleene algebra with tests. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 19(3):427–443, 1997. doi:10.1145/256167.256195.

16 Stephen Lack. Composing PROPs. *Theory and Application of Categories*, 13(9):147–163, 2004.

17 F William Lawvere. Functorial semantics of algebraic theories. *Proceedings of the National Academy of Sciences of the United States of America*, 50(5):869, 1963.

18 Robin Milner. A complete inference system for a class of regular behaviours. *Journal of Computer and System Sciences*, 28(3):439–466, 1984. doi:10.1016/0022-0000(84)90023-0.

19 Andrew M Moshier. Coherence for categories of posets with applications. *Topology, Algebra and Categories in Logic (TACL)*, page 214, 2015.

20 Robin Piedeleu and Fabio Zanasi. A finite axiomatisation of finite-state automata using string diagrams. *Logical Methods in Computer Science*, 19, 2023. doi:10.46298/LMCS-19(1:13)2023.

21 Robin Piedeleu and Fabio Zanasi. An introduction to string diagrams for computer scientists. *arXiv preprint arXiv:2305.08768*, 2023. doi:10.48550/arXiv.2305.08768.

22 Valentin N Redko. On defining relations for the algebra of regular events. *Ukrainskii Matematicheskii Zhurnal*, 16:120–126, 1964.

23 Steffen Smolka, Nate Foster, Justin Hsu, Tobias Kappé, Dexter Kozen, and Alexandra Silva. Guarded Kleene algebra with tests: verification of uninterpreted programs in nearly linear time. *Proceedings of the 47th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL)*, 4:1–28, 2020. doi:10.1145/3371129.

A Algebraic operations on NFA

We define here precisely the operation we use in the paper. Fix two NFA $A = (Q, q_0, F, \alpha)$ and $B = (S, s_0, G, \beta)$.

▶ **Definition 39** (Sum). *We define the sum of A and B, denoted $A + B$ as follows: $A + B = (T, t_0, H, \gamma)$ where:*

- $T = Q \sqcup S \sqcup \{t_0\}$, *with t_0 neither appearing in Q nor S.*
- $H(t) = F(t)$ *if $t \in Q$, $H(t) = G(t)$ if $t \in S$ and $H(t_0) = F(q_0) \vee G(t_0)$*
- $(t, a, t') \in \gamma$ *whenever $(t, a, t') \in \alpha$, or $(t, a, t') \in \beta$, or $t = t_0$ and $(q_0, a, t) \in \alpha$ or $(s_0, a, t) \in \beta$.*

Intuitively, summing A and B only consists in merging their initial states into one which mimics the behaviour of both.

▶ **Definition 40** (Synchronous Product). *The product $A \times B = (T, t_0, H, \gamma)$ of A and B is defined as:*

- $T = Q \times S$
- $t_0 = (q_0, s_0)$
- $H(q, s) = F(q) \wedge G(s)$
- $(q, s) \xrightarrow{a}_{A \times B} (q', s')$ *whenever $q \xrightarrow{a}_A q'$ and $s \xrightarrow{a}_B s'$*

Intuitively, every path in $A \times B$ is also a path in both A and B.

▶ **Definition 41** (Composition). *The composition $A.B = (T, t_0, H, \gamma)$ of A and B is defined as:*

- $T = Q \sqcup \{i \mid F(q_i) = 1\} \times (S \setminus \{s_0\})$.
- $t_0 = q_0$
- $H(q) = F(q) \wedge G(s_0)$ *for $q \in Q$, and $H(i, s) = G(s)$ for $s \in S$*

- *There are three cases for transitions in $A.B$:*
  - *For $q, q' \in Q$, $q \xrightarrow{a}_{A.B} q'$ whenever $q \xrightarrow{a}_{A} q'$,*
  - *For $s, s' \in S$, $(i, s) \xrightarrow{a}_{A.B} (i, s')$ whenever $s \xrightarrow{a}_{B} s'$*
  - *For $q_i \in Q$ such that $F(q_i) = 1$ and $s \in S$, $q_i \xrightarrow{a}_{A.B} (i, s)$ whenever $s_0 \xrightarrow{a}_{B} s$*

Intuitively, to every final state in A we "attach" an entire copy of B.

▶ **Definition 42** (Prefix). *If $a \in \Sigma$ then A prefixed by a, written $a.A$, is the composition of $(\{q_0, q_1\}, q_0, \{q_0 \mapsto 0, q_1 \mapsto 1\}, \{(q_0, a, q_1)\})$ and A.*

We will also write $S.A$ for $S \in \mathcal{P}_f(\Sigma)$ some finite subsets of Σ, as a shorthand for $\left(\sum_{a \in S} a\right).A = \sum_{a \in S} a.A$.

B Background on SMCs and props

B.1 Props, String Diagrams, and Symmetric Monoidal Theories

After having defined our syntax, the free prop $\mathsf{P}_{\mathcal{S}}$ over signature $\mathcal{S}$, we give its interpretation into Sem, a SMC that constitutes our target semantics. To guarantee a compositional interpretation, we require $[\![\cdot]\!] : \mathsf{P}_{\mathcal{S}} \to \mathsf{Sem}$, the mapping of terms to their intended semantics, to be a symmetric monoidal functor.

Once we have specified $[\![\cdot]\!] : \mathsf{P}_{\mathcal{S}} \to \mathsf{Sem}$, it is natural to look for equations to reason about semantic equality directly on the diagrams themselves. Given a set of equations E, i.e., a set containing pairs of morphisms of the same type, we write $=_E$ for the smallest congruence w.r.t. the two composition operations ; and $\oplus$. We say that $=_E$ is *sound* if $c =_E d$ implies $[\![c]\!] = [\![d]\!]$. It is moreover *complete* when the converse implication also holds. We call a pair $(\mathcal{S}, E)$ a *symmetric monoidal theory* (SMT) and we can form the prop $\mathsf{P}_{\mathcal{S},E}$ obtained by quotienting each homset of $\mathsf{P}_{\mathcal{S}}$ by $=_E$. There is then a prop morphism $q : \mathsf{P}_{\mathcal{S}} \to \mathsf{P}_{\mathcal{S},E}$ witnessing this quotient.

The reader familiar with categorical logic, may find it helpful to know that the concrete description above can be described in more abstract categorical terms, in line with Lawvere's account of algebraic theories [17]: signatures can be organised into a category and the free prop $\mathsf{P}_{\mathcal{S}}$ given as a monad structure over this category. Furthermore, the category of props and prop morphisms is equivalent to the category of algebras for this monad. Then, by standard abstract nonsense, the prop $\mathsf{P}_{\mathcal{S},E}$ and the quotient morphism q arise as a coequaliser of free props. A detailed account of this presentation can be found in [1, Appendix A.2].

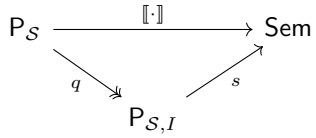
B.2 (Pre-)Ordered Props and Symmetric Monoidal Inequality Theories

Our semantic prop Sem often carries additional structure that we wish to lift to the syntax: monotone relations *qua* relations can be ordered by inclusion. The corresponding mathematical structure is that of an *ordered (or order-enriched) prop*, a prop whose homsets are also posets, with composition and monoidal product monotone maps.

In the same way that props can be presented by SMTs, an ordered prop can be presented by *symmetric monoidal inequality theory* (SMIT). Formally, the data of a SMIT is the same as that of a SMT: a signature $\mathcal{S}$ and a set I of pairs $c, d : X \to Y$ of $\mathsf{P}_{\mathcal{S}}$-arrows of the same type, that we now read as *inequalities* $c \leq d$.

As for plain props, we can construct a pre-ordered prop from a SMIT by building the free prop $\mathsf{P}_{\mathcal{S}}$ and passing to a quotient $\mathsf{P}_{\mathcal{S},I}$: we first build the pre-order on each homset by closing I under $\oplus$ and taking the reflexive and transitive closure of the resulting relation. Finally, we obtain $\mathsf{P}_{\mathcal{S},I}$ by quotienting the resulting prop by imposing anti-symmetry.

SMITs subsume SMTs, since every SMT can be presented as a SMIT, by splitting each equation into two inequalities. As a result, in the main text, we only consider SMITs, referring to them simply as *theories*, and their defining inequalities as *axioms*. When referring to a sound and complete theory, we will also use the term *axiomatisation*, as is standard in the literature. The situation for a sound and complete theory is summarised as a commutative diagram:



Soundness simply means that $[\![\cdot]\!]$ factors as $s \circ q$ through $\mathsf{P}_{\mathcal{S},E}$ and completeness means that s is a faithful prop morphism.