Diagrammatic Algebra: From Linear to Concurrent Systems

FILIPPO BONCHI*, Ecole Normale Supérieure, France

JOSHUA HOLLAND, University of SouthamptonSouthampton , United Kingdom

PAWEŁ SOBOCIŃSKI†, University of Southampton, United Kingdom

ROBIN PIEDELEU, University of Oxford, United Kingdom

FABIO ZANASI‡, University College London, United Kingdom

We introduce the resource calculus, a string diagrammatic language for concurrent systems. Significantly, it uses the same syntax and operational semantics as the signal flow calculus — an algebraic formalism for signal flow graphs, which is a combinatorial model of computation of interest in control theory. Indeed, our approach stems from the simple but fruitful observation that, by replacing real numbers (modelling signals) with natural numbers (modelling resources) in the operational semantics, concurrent behaviour patterns emerge.

The resource calculus is canonical: we equip it and its stateful extension with equational theories that characterise the underlying space of definable behaviours—a convex algebraic universe of additive relations—via isomorphisms of categories. Finally, we demonstrate that our calculus is sufficiently expressive to capture behaviour definable by classical Petri nets.

Additional Key Words and Phrases: String Diagrams, props, Completeness, Petri nets, Category Theory, Concurrency

1 INTRODUCTION

The Quest for a Theory of Concurrency. In the words of Samson Abramsky [Abramsky 2014], the fundamental structures of concurrency are yet to be discovered. Unlike Turing machines or the λ-calculus, there is no yardstick formalism for concurrent computation, let alone an agreement of what are the "right" primitives to reason about distributed systems. The most widespread paradigm, process algebras, have been undeniably successful in offering powerful and flexible toolkits to reason about various concurrent behaviours found in the wild. However, as argued by Abramsky in *loc. cit.*, their versatile linguistic character is also their limit: there are too few constraints to seek intrinsic, syntax-independent concurrency primitives. The syntax-dependent nature of the research contributed to the Balkanisation of the community, and formalism-specific approaches.

At the other side of the spectrum, there is Petri's approach to concurrency. The framework of Petri nets does seek to determine grounding concepts such as causality, concurrency, process, etc. in a syntax-independent fashion. The use of a graphical formalism has the additional advantage of acknowledging the spatial structure of distributed systems, emphasising physical properties such as connectivity and resource-exchange. What makes Petri nets less appealing from a programming language perspective is that, differently from process algebras, the native form of reasoning is combinatorial. This has led to an emphasis on monolithic systems, with comparatively little attention

given to compositionality. Some research has been done to mediate between the two approaches and establish common conceptual themes, e.g. through event structures [Nielsen et al. 1993].

This paper is an effort in the quest for a canonical theory of concurrency. We depart from syntax-centric approaches in a fundamental way: our syntax is *canonical* in that it, together with its equational theory, *characterises* the underlying mathematical domains in an extremely strong sense: an isomorphism of categories. We build our way to a *complete* equational theory for the syntax, which we refer to as the *resource calculus*. We establish an analogous completeness result for its *stateful* extension. As a demonstration of the expressiveness and applicability of the calculus, we show how it acts as an assembly language for classical (Place/Transition) Petri nets.

Intriguingly, our syntax is the same that was used [Baez and Erbele 2015; Bonchi et al. 2017c; Fong et al. 2016] for linear systems, which itself goes back to Shannon [Shannon 1942] and the class of signal flow graphs [Mason 1953; Willems 2007]. The difference between the computational interpretations—from control-theoretic to concurrent—boils down to (i) *operationally*, replacing the signal domain that in control-theoretic examples is a field (e.g. $\mathbb{Q}$ or $\mathbb{R}$) with $\mathbb{N}$, since for us signals represent discrete (and non-negative) resources, and (ii) *algebraically*, different interaction patterns between the syntactic primitives. Below we expand on the principled methodology guiding us.

Linear Systems. Our syntax originates from linear systems, i.e. those whose behaviour defines a linear subspace over a field. Our departure point is *graphical linear algebra* (GLA) [Bonchi et al. 2017d; Zanasi 2015], a sound and complete theory of subspaces over a field. "Graphical" here points to the fact that, even though it is a calculus with recursively defined syntax, its terms are drawn as 2-dimensional diagrams and composed through parallel and sequential composition.

$$-\boxed{c}-, \ -\boxed{d}- \ ::= \quad \rightarrow\bullet \mid \rightarrow\!\!\!\!\prec \ \mid \ \supset\!\!- \mid \circ\!- \mid \bullet\!- \mid \ \rrangle\!\bullet- \mid -\!\circ \mid -\!\prec \tag{1}$$
$$\mid - \mid \times \mid \ -\boxed{c}-\boxed{d}- \mid \ \frac{-\boxed{c}-}{-\boxed{d}-}$$

The intended interpretation is that $\supset\!-$ is addition, $\circ\!-$ the constant zero, $\rightarrow\!\!\!\prec$ copy, $\rightarrow\!\bullet$ discard, while $\rrangle\!\bullet-$, $\bullet\!-$, $-\!\circ$ and $-\!\prec$ are the same operations right-to-left, $-$ is the identity, and $\times$ is the symmetry. This intuition is formalised via a recursively defined mapping of diagrams to linear subspaces. The syntax (1) comes with a sound and complete equational theory [Bonchi et al. 2017d]. The mathematical setting for this result is the theory of *props*, a particular kind of symmetric monoidal category [Lack 2004], suitable for reasoning algebraically about 2-dimensional syntax.

From a computational viewpoint, all the connectors of GLA are *stateless*. If one augments the syntax (1) with a generator $-\boxed{x}-$, then stateful processes can be also represented. $-\boxed{x}-$ behaves as a register (one-place buffer): at each stage of the computation, it contains a value $k \in \mathbb{R}$; on input r, it realises k as output, and r becomes the newly stored value. For example, the circuit diagram below, constructed with the syntax (1) and $-\boxed{x}-$, computes the Fibonacci rational stream function: when the stream $10000\ldots$ comes in on the left, the stream $12358\ldots$ is computed on the right.

$$fib := \tag{2}$$

By interpreting values as signals, one can think of such diagrams as *signal flow graphs*, a fundamental combinatorial model in control theory. Indeed, signal flow graphs faithfully embed into the diagrammatic theory given by (1) plus $-\boxed{x}-$. In [Baez and Erbele 2015; Bonchi et al. 2014] a complete axiomatisation, called the *signal flow calculus*, is provided for this syntax. It offers a principled, algebraic approach to a variety of tasks: circuit equivalence [Bonchi et al. 2014],

realising a specification [Bonchi et al. 2015], deadlock removal [Bonchi et al. 2015], composition of controllable systems [Fong et al. 2016], refinement [Bonchi et al. 2017a] and more.

From Linear to Concurrent Systems. The above sketch demonstrates how (1) admits an axiomatisation of a significant family of behaviours (linear systems), capturing a well-known preexisting combinatorial model (signal flow graphs). In the spirit of process algebra, the approach is compositional, allowing for the representation of open systems and emphasising interaction; in the spirit of Petri, the syntax is graphical, emphasising the connection topology of systems.

The theme of this paper is to show that the algebraic approach to linear systems can be successfully transferred to the analysis of concurrent systems. What is most striking is that the setup is essentially unaltered: the generators of the syntax are the same, and the only significant change is modelling their behaviour via a different semiring, passing from $\mathbb{R}$ to $\mathbb{N}$.

In order to explain this point, we anticipate two examples from Section 2.2.

$$c_1 := \qquad\qquad c_2 := \qquad\qquad\qquad\qquad (3)$$
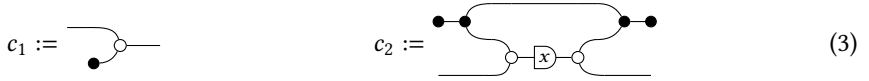
In the signal flow calculus the behaviour of these diagrams, which can be computed by following the above description for generators, is trivial: both are the full relation —● ●—, relating any input to any output. Indeed, the first diagram expresses the constraint that $r \in \mathbb{R}$ on the left is related to $r' \in \mathbb{R}$ on the right such that $r' = r + r''$ for some $r'' \in \mathbb{R}$, which is the case for all $r, r' \in \mathbb{R}$. For the second diagram, assume some value $s \in \mathbb{R}$ is currently in the register: here given input r, in order to output r' one just needs to find r'' such that $r' + r'' = s$, which is always possible.

Now, suppose that instead of giving a *linear* interpretation of these diagrams we give a *resource* interpretation: wires carry discrete tokens that cannot be borrowed (i.e. they cannot be a negative quantity). This amounts to switching the signal space from $\mathbb{R}$ to $\mathbb{N}$, the semiring of natural numbers. To quip, we move from Graphical Linear Algebra to Graphical Diophantine Algebra.

The "Aha-Erlebnis" is that, once interpreted over $\mathbb{N}$, the same diagrams with the same operational understanding now model a non-trivial (and even familiar) behaviour. In the interpretation of the first diagram, a value $r'' \in \mathbb{N}$ is now only available if $r' \geq r''$: thus it expresses an *ordering* constraint. For the second diagram, the output $r' \in \mathbb{N}$ is now forced to be a number of tokens smaller than the one $k \in \mathbb{N}$ stored in the register, and the new value in the register will be $r + (k - r')$. In other words, the diagram now models the same behaviour as a *place* in a Petri net!

Additive Relations and the Resource Calculus. The appearance of Petri nets suggests that the theory of signal flow calculus can be adapted to capture concurrent phenomena. This is the path we have taken. The first ingredient is to identify the domain of $\mathbb{N}$-valued executions of a system. In the linear case, the interpretation domain is the category LinRel where arrows are linear relations, i.e. relations between $\mathbb{R}$-vectors that are also linear subspaces. Analogously, we work in the domain of finitely generated *additive relations*, which are relations between $\mathbb{N}$-vectors: we call AddRel the resulting category. The word "additive" refers to the fact that these relations preserve the additive structure of $\mathbb{N}$. Additive relations can be thus thought of as a resource-sensitive analogue of linear relations, suitable as a mathematical universe for concurrent computational phenomena that involve consuming and producing resources, especially where "negative resources" are unreasonable.

Our first original contribution is an axiomatisation for AddRel, which we call the *resource calculus*. Foreshadowed by the above insight, the syntax of the calculus is (1), the same as the (stateless) theory of linear systems, but because of the resource interpretation in AddRel, the resulting axioms— which characterise how the basic components interact—are different: they are displayed in Figure 4. In addition to soundness and completeness, the terms of (1) suffice to express *every relation* in AddRel: we refer to this combination as soundness and *full* completeness.

THEOREM 1. *The resource calculus is sound and fully complete for* AddRel.

The equations of the resource calculus describe familiar algebraic structures: $\{$ ⊸—, ○—, —⊸, —○ $\}$ form a bimonoid, $\{$ —⊸, —●, ⊸—, —● $\}$ form a Frobenius monoid, and together they interact as another bimonoid. There are three additional equations that concern specific additive and multiplicative properties of $\mathbb{N}$. The characterisation theorem gives equational insights into the differences between the linear and the additive: for instance, in the axiomatisation of the linear case, $\{$ ⊸—, ○—, —⊸, —○ $\}$ forms a Frobenius monoid instead of a bimonoid.

The Stateful Extension. Next, we study a stateful extension that increases the expressivity of our basic calculus, while keeping the syntax anchored by a completeness result with respect to its canonical space of behaviours. In the graphical linear algebra framework, a crucial step is the addition of the register —$\boxed{x}$—, which allows to move from stateless to stateful systems. We study the same syntactic extension of the resource calculus, yielding the *stateful resource calculus*. In fact, we are able to develop a more general framework, which is of independent interest: it allows to turn a stateless theory (formally, a prop) T into a stateful one $\mathsf{St}(T)$, provided that T has a compact-closed structure. The beauty of this approach is that the semantics of the new generator —$\boxed{x}$— added by the $\mathsf{St}(\cdot)$ construction is *forced* on us: it must have the register behaviour as described above. Thus the register is a canonical way of adding state to a stateless calculus. This sheds light on previous stateful extensions (e.g. the linear case), showing that they are canonical in a precise sense.

This leads to another completeness result.

THEOREM 2. *The stateful resource calculus is sound and fully complete for* $\mathsf{St}(\mathsf{AddRel})$.

Petri nets. To showcase expressivity, we consider classical Petri nets and show that their behaviour is definable in the stateful resource calculus.

THEOREM 3. *Petri nets embed in the stateful resource calculus.*

The translation expresses the notion of place in terms of more elementary components (the rightmost diagram in (3)). As a consequence of the completeness result, we obtain a complete axiomatisation for equivalence. Thus the resource calculus and its extension can be considered as an "assembly language" in which other formalisms may be interpreted.

We cannot claim that the resource calculus and its stateful extension is the definitive answer to the question raised by Abramsky, but rather a promising step in that direction. We do not investigate fundamental concepts such as causality, conflicts, handshake communication or mobility. Moreover, the equivalence that we axiomatize for stateful systems is rather intensional, not a behavioural equivalence such as trace equivalence, bisimilarity or more refined notions [van Glabbeek 1990]. We are confident that we can make progress in these directions in future work.

Nevertheless, such translations are not mere curiosities. The microscopic analysis afforded by our theoretical framework—connecting syntax, semantics and the underlying mathematical domains—means that we arrive at a clearer demarcation of the *design space* of classical models of concurrency. As previously mentioned, the register is a canonical notion of state. Petri nets places are a different notion of state and we can isolate and give a precise explanation of *how* they define state in a broader taxonomy of possible design choices: in this sense, our theory is a yardstick for models of concurrency.

Outline. In Section 2 we introduce the syntax of circuit diagrams: the base case and its stateful extension, as well as their semantic interpretations. In Section 3 we introduce additive relations and the equational theory of the resource calculus, based on the core syntax of circuit diagrams. We then prove it sound and fully complete for additive relations. In Section 4 we use a general categorical

	[Bonchi et al. 2017d]	[Bonchi et al. 2017c]	Sec. 3	Sec. 4, 5
Theory	Graphical Linear Algebra	Signal Flow Calculus	Resource Calculus (RC)	Stateful RC
Syntax	Circ	$Circ_s$	Circ	$Circ_s$
Equations	See [Bonchi et al. 2017d]	See [Bonchi et al. 2017c]	Rc	Rc_s
Behaviours	LinRel	St(LinRel)	AddRel	St(AddRel)
Embeds	-	Signal Flow Graphs	-	Petri Nets

Fig. 1. From linear to concurrent systems. The notation follows that used in the paper.

construction that adds state and apply it to the resource calculus. This gives us a soundness and full completeness result for the stateful extension. Section 5 is our case-study and concerns Petri nets: we show how Petri nets embed into the resource calculus. We offer some concluding remarks in Section 6.

2 THE LANGUAGE OF CIRCUIT DIAGRAMS

Syntax. Our starting point is the simple grammar for a language Circ of circuit diagrams.

$$c, d \quad ::= \quad \multimap\!\bullet \mid \multimap\!\!\!\mathrel{\mathpalette\@mathrel{}{<}} \mid \mathrel{\supset}\!\!\mathrel{\circ}\!\!- \mid \circ\!- \mid \bullet\!- \mid \mathrel{\supset}\!\!\!\bullet\!- \mid \mathrel{\vdots} \mid - \mid \mathrel{\times} \mid c \mathbin{;} d \mid c \oplus d \qquad (4)$$

Although the constants of our grammar are pictures, at this point we consider them as mere symbols. We will, in due course (Section 2.1), consider terms derived from this signature as *string diagrams* [Selinger 2011], hence the pictorial rendering; indeed, the two binary operations, *sequential* ($c \mathbin{;} d$) and *parallel* ($c \oplus d$) composition, are those of monoidal categories. It is worth noting that we do not use variables; thus there is no need of assuming their countable supply, nor rules for capture-avoiding substitution. On the other hand, we need a simple sorting discipline. A sort is a pair (k, l), with $k, l \in \mathbb{N}$. Henceforth we will consider only sortable terms, according to the following rules.

$$\multimap\!\bullet : (1, 0) \quad \multimap\!\!\!\mathrel{<} : (1, 2) \quad \mathrel{\supset}\!\!\mathrel{\circ}\!- : (2, 1) \quad \circ\!- : (0, 1) \quad \bullet\!- : (0, 1) \quad \mathrel{\supset}\!\!\!\bullet\!- : (2, 1)$$

$$\mathrel{\vdots} : (0, 0) \quad - : (1, 1) \quad \times : (2, 2) \quad \frac{c : (k_1, k_2) \quad d : (k_2, k_3)}{c \mathbin{;} d : (k_1, k_3)} \quad \frac{c : (k_1, l_1) \quad d : (k_2, l_2)}{c \oplus d : (k_1 + k_2, l_1 + l_2)} \qquad (5)$$

The sorts give the number of dangling wires on each side of the diagram represented by a term. An easy induction confirms uniqueness of sorting: if $c : (k, l)$ and $c : (k', l')$, then $k = k'$ and $l = l'$.

Semantics. Define the operational meaning of terms recursively by the following structural rules

$$\multimap\!\bullet \xrightarrow{\frac{n}{\varepsilon}} \multimap\!\bullet \quad \multimap\!\!\!\mathrel{<} \xrightarrow[\substack{n}]{\frac{n}{n}} \multimap\!\!\!\mathrel{<} \quad \mathrel{\supset}\!\!\mathrel{\circ}\!- \xrightarrow{\frac{n}{m}}_{n+m} \mathrel{\supset}\!\!\mathrel{\circ}\!- \quad \circ\!- \xrightarrow{\frac{\varepsilon}{0}} \circ\!- \quad \bullet\!- \xrightarrow{\frac{\varepsilon}{n}} \bullet\!- \quad \mathrel{\supset}\!\!\!\bullet\!- \xrightarrow[\substack{n}]{\frac{n}{n}} \mathrel{\supset}\!\!\!\bullet\!-$$

$$\mathrel{\vdots} \xrightarrow{\frac{\varepsilon}{\varepsilon}} \mathrel{\vdots} \quad - \xrightarrow[\substack{n}]{\frac{n}{n}} - \quad \times \xrightarrow[\substack{n}]{\frac{n}{m}} \times \quad \frac{c \xrightarrow{\frac{a}{b}} c' \quad d \xrightarrow{\frac{b}{c}} d'}{c \mathbin{;} d \xrightarrow{\frac{a}{c}} c' \mathbin{;} d'} \quad \frac{s \xrightarrow{\frac{a_1}{b_1}} c' \quad d \xrightarrow{\frac{a_2}{b_2}} d'}{c \oplus d \xrightarrow[\substack{b_1 \\ b_2}]{\substack{a_1 \\ a_2}} d' \oplus d'} \qquad (6)$$

where m, n range over the natural numbers $\mathbb{N}$ and $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$ over natural number vectors.

The intuition is that resources (or tokens) travel along wires in bunches of size $n \in \mathbb{N}$. For each connector $c : (k, l)$, a transition $c \xrightarrow{\frac{a}{b}} c$ means that we observe resource vector $\boldsymbol{a}$ on the left, where

component a_i of $\boldsymbol{a}$ refers to the resources observed on the ith left wire, and—similarly—observe resources $\boldsymbol{b}$ on the right boundary of c. We write ε for the unique vector of length zero. The rules say that —•⟨ duplicates, —• discards and ⟩>— sums resources, whereas ∘— produces no resources. The mirror images ⟩>—, —• carry resources from right to left, with behaviour defined as for their symmetric counterpart. Finally, behaviours combine sequentially, where observations synchronise along the common boundary, or in parallel, where observations are simply concatenated.

For any term $c\colon (k,\, l)$, the rules (6) yield a labelled transition system where each transition has form $c \xrightarrow{\;a\;}_{b} c$. Given that any such transition system has precisely one state, we define the semantics of c as the following *relation*, which collects all possible observations admitted by c.

$$\llbracket c \rrbracket := \{(\boldsymbol{a}, \boldsymbol{b}) \mid c \xrightarrow{\;a\;}_{b} c\} \subseteq \mathbb{N}^k \times \mathbb{N}^l\,. \tag{7}$$

Stateful Extension. Circ is stateless. We introduce a simple stateful extension of the language of circuit diagrams, which allows for more sophisticated examples. These variations will be used to capture the familiar model of computation of Petri nets (Section 5).

For this purpose, we define the language Circ_s of *stateful* circuits that extends (4) with generator —$\boxed{x}$— $\colon (1,\, 1)$ and (6) with:

$$(—\boxed{x}—, m) \xrightarrow{\;n\;}_{m} (—\boxed{x}—, n) \tag{8}$$

Intuitively, —$\boxed{x}$— is a one-place buffer, and $(—\boxed{x}—, n)$ is the state in which it contains the value n. Thus, differently from the structural rules in (6), —$\boxed{x}$— yields transition systems with (infinitely many) different states, and the state determines possible observations at any point in the execution. In general, given a term c in Circ_s, states are pairs $(c, \boldsymbol{s})$ where $\boldsymbol{s}$ is a vector of natural numbers that stores the internal state of each buffer.

For the sake of completeness, we update the rules in (6): for the basic stateless generators, $\boldsymbol{s}$ is always ε, the unique vector of $\mathbb{N}^0$.

$$(—•, \varepsilon) \xrightarrow{\;n\;}_{\varepsilon} (—•, \varepsilon) \quad (—•⟨, \varepsilon) \xrightarrow{\;n\;}_{\substack{n\\n}} (—•⟨, \varepsilon) \quad (⟩>—, \varepsilon) \xrightarrow{\substack{n\\m}}_{n+m} (⟩>—, \varepsilon) \quad (∘—, \varepsilon) \xrightarrow{\;\varepsilon\;}_{0} (∘—, \varepsilon)$$

$$(•—, \varepsilon) \xrightarrow{\;\varepsilon\;}_{n} (•—, \varepsilon) \quad (⟨•—, \varepsilon) \xrightarrow{\substack{n\\n}}_{n} (⟨•—, \varepsilon) \quad (\boxed{}, \varepsilon) \xrightarrow{\;\varepsilon\;}_{\varepsilon} (\boxed{}, \varepsilon) \quad (—, \varepsilon) \xrightarrow{\;n\;}_{n} (—, \varepsilon)$$

$$(\times, \varepsilon) \xrightarrow{\substack{n\\m}}_{\substack{m\\n}} (\times, \varepsilon)$$

The rules for sequential and parallel composition, displayed below, deal with states of shape $(c, \boldsymbol{s})$.

$$\frac{(c, \boldsymbol{s}) \xrightarrow{\;a\;}_{b} (c', \boldsymbol{s'}) \quad (d, \boldsymbol{t}) \xrightarrow{\;b\;}_{c} (d', \boldsymbol{t'})}{(c \,;\, d, \tfrac{\boldsymbol{s}}{\boldsymbol{t}}) \xrightarrow{\;a\;}_{c} (c' \,;\, d', \tfrac{\boldsymbol{s'}}{\boldsymbol{t'}})} \qquad \frac{(c, \boldsymbol{s}) \xrightarrow{\;a_1\;}_{b_1} (c', \boldsymbol{s'}) \quad (d, \boldsymbol{t}) \xrightarrow{\;a_2\;}_{b_2} (d', \boldsymbol{t'})}{(c \oplus d, \tfrac{\boldsymbol{s}}{\boldsymbol{t}}) \xrightarrow{\substack{a_1\\a_2}}_{\substack{b_1\\b_2}} (c' \oplus d', \tfrac{\boldsymbol{s'}}{\boldsymbol{t'}})}$$

Observe that, for both $;$ and $\oplus$, the vectors $\boldsymbol{s}$ and $\boldsymbol{t}$ are simply stacked. It is important to remark that the equivalence $\sim$ (defined below (29)) makes the order over these vectors irrelevant. For instance, in the transition systems generated by $(—\boxed{x}— \,;\, —\boxed{x}—) \oplus (—\boxed{x}— \,;\, —\boxed{x}—)$ and $(—\boxed{x}— \oplus —\boxed{x}—) \,;\, (—\boxed{x}— \oplus —\boxed{x}—)$, the state is represented by a vector $\boldsymbol{s}$ in $\mathbb{N}^4$ where, for the first term, the second position of $\boldsymbol{s}$ represents the value of the top rightmost register, while in the second term, the second position represents the value of the bottom leftmost register. Nevertheless, according to the definition of $\sim$, the two terms are semantically equivalent.

Given a term $c\colon (k,\, l)$ of Circ_s with set of registers R, we define its semantics as the relation:

$$\llbracket c \rrbracket := \{(\boldsymbol{s}_1, \boldsymbol{a}, \boldsymbol{s}_2, \boldsymbol{b}) \mid (c, \boldsymbol{s}_1) \xrightarrow{\;a\;}_{b} (c, \boldsymbol{s}_2)\} \subseteq \mathbb{N}^R \times \mathbb{N}^k \times \mathbb{N}^R \times \mathbb{N}^l\,. \tag{9}$$

We equate relations (29) up-to permutations of the underlying sets of registers: we say that $S \subseteq \mathbb{N}^R \times \mathbb{N}^k \times \mathbb{N}^R \times \mathbb{N}^l$ and $S' \subseteq \mathbb{N}^{R'} \times \mathbb{N}^k \times \mathbb{N}^{R'} \times \mathbb{N}^l$ are *semantically equivalent* ($\sim$) if there is a permutation $\sigma \colon R \to R'$ such that $\sigma(S) = S'$. Analogously to α-equivalence, this quotient ensures that registers are anonymous: the name we indicated them with (as elements of S or S') does not matter. Note that for terms without registers, $\sim$ is simply equality, as relations.

It is important to note that semantic equivalence $\sim$ is not intended to be used as an *observational* equivalence (like traces or bisimilarity) because it is still quite intensional: it merely equates labelled transition systems generated by circuits through anonymising registers.
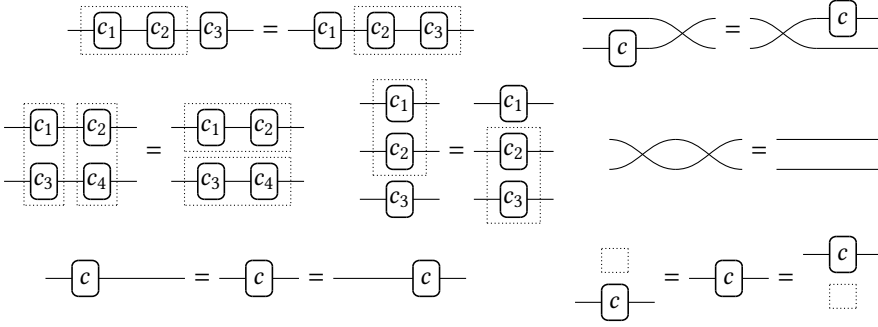
2.1 From Terms to String Diagrams

Fig. 2. Laws of Symmetric Monoidal Categories. Sort labels are omitted for readability.

Our main goal for this paper is to obtain an equational characterisation of semantic equivalence in Circ and Circ_s. We start by noting that—in each case—these equations contain those of symmetric monoidal categories (SMCs), allowing us to move from terms to string diagrams.

Even to state these initial equations, it helps to draw diagrams: the graphical notation is appealing as it highlights connectivity and the capability for resource exchange. It is for this reason that we used a graphical rendering of the components in (4). One draws terms as 2-dimensional diagrams, depicting $c \colon (k, l)$ as $\xrightarrow{k} \boxed{c} \xrightarrow{l}$, $c \mathbin{;} d$ as $\xrightarrow{k_1} \boxed{c} \xrightarrow{k_2} \boxed{d} \xrightarrow{k_3}$ and $c \oplus d$ as $\begin{smallmatrix} k_1 & \boxed{c} & l_1 \\ k_2 & \boxed{d} & l_2 \end{smallmatrix}$, where the labelled wire $\xrightarrow{k}$ stands for a stack of k wires. We often omit wire labels when it does not lead to confusion.

PROPOSITION 4. *Let $\equiv$ be the smallest congruence over the terms of* Circ *and* Circ_s *generated by the equations in Figure 2. If $c \equiv d$ then $[\![c]\!] \sim [\![d]\!]$.*

The above implies that it is harmless to consider $\equiv$ as a *structural equivalence* on terms. Indeed, henceforward we will consider the terms of each calculus as arrows of monoidal categories, which by a mild abuse of notation we will also denote Circ and Circ_s respectively. They are all examples of SMCs known as *prop*s.

DEFINITION 5. *A prop is a symmetric strict monoidal category with objects the natural numbers, with $k \oplus l$ defined by the addition $k + l$. A prop morphism $\mathcal{F} \colon \mathsf{C} \to \mathsf{D}$ is a strict symmetric monoidal functor from C to D that is the identity on objects. Props and prop morphisms form a category* **PROP**.

Thus Circ and Circ_s are props with arrows $k \to l$ sorted terms $c \colon (k, l)$ of the corresponding syntax modulo $\equiv$, with sequential composition $c \mathbin{;} d$, monoidal product $c \oplus d$, and symmetries defined by the corresponding operations in (4) ([Bonchi et al. 2017c, Definition 2.3]).

The semantic domains of our calculi are various kinds of relations. For this reason, the family of props defined below is of central importance.

DEFINITION 6 (Rel$_X$). *Given a set X, let Rel$_X$ be the prop with arrows $k \to l$ relations R from X^k to X^l, i.e. $R \subseteq X^k \times X^l$. Given $R : k_1 \to k_2$ and $S : k_2 \to k_3$, their composition $R \,;\, S : k_1 \to k_3$ is*

$$\{(\boldsymbol{x}, \boldsymbol{z}) : \boldsymbol{x} \in X^{k_1}, \, \boldsymbol{z} \in X^{k_3} \text{ and there exists } \boldsymbol{y} \in X^{k_2} \text{ such that } (\boldsymbol{x}, \boldsymbol{y}) \in R \text{ and } (\boldsymbol{y}, \boldsymbol{z}) \in S\}.$$

Given $R : k_1 \to l_1$ and $S : k_2 \to l_2$ their monoidal product is obtained by taking their cartesian product, i.e. $R \oplus S : k_1 + k_2 \to l_1 + l_2$ is the relation

$$\left\{ \left(\left(\begin{smallmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{smallmatrix} \right), \left(\begin{smallmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{smallmatrix} \right) \right) : \boldsymbol{x}_1 \in X^{k_1}, \, \boldsymbol{x}_2 \in X^{k_2}, \, \boldsymbol{y}_1 \in X^{l_1}, \, \boldsymbol{y}_2 \in X^{l_2} \text{ such that } (\boldsymbol{x}_1, \boldsymbol{y}_1) \in R \text{ and } (\boldsymbol{x}_2, \boldsymbol{y}_2) \in S \right\}.$$

Identity relations are identities and symmetries are defined in the obvious way. It is easy to see that both forms of composition are strictly associative and that '$\oplus$' is functorial wrt '$\,;\,$'.

By taking $X = \mathbb{N}$, the operational behaviours of diagrams in Circ are arrows in Rel$_\mathbb{N}$. Note that $[\![-]\!] : \text{Circ} \to \text{Rel}_\mathbb{N}$ is compositional— formally, it is a morphism of props.

Compact closed structure. Two circuits will play a special role in our exposition: $\multimap\bullet$, called cup and $\bullet\hspace{-0.3em}\multimapdotinv$, cap. Using the rules in (6), it is easy to see that their behaviour forces the two ports on the left (respectively right) to carry the same resources, thus acting as left (right) feedback:

Using these diagrams (along with — and $\times$) as building blocks, is it possible to define for each $k \in \mathbb{N}$, $\overline{k} : k+k \to 0$ and $\underline{k} : 0 \to k+k$ with operational behaviour $\overline{k} \xrightarrow[\varepsilon]{a} \overline{k}$ and $\underline{k} \xrightarrow[a]{\varepsilon} \underline{k}$ for all $\mathbf{a} \in \mathbb{N}^{\mathbf{k}}$. See e.g. [Bonchi et al. 2017c, §5.1] for details. This definition gives rise to a *self-dual compact closed structure.*

DEFINITION 7. *A (self-dual) compact closed prop is a prop such that for every $k \in \mathbb{N}$, there exist $\underline{k} : 0 \to k + k$ and $\overline{k} : k + k \to 0$ such that $\underline{k} \times = \underline{k}$, $\times \overline{k} = \overline{k}$ and*

As for identities and symmetries, also for $\underline{k}$ and $\overline{k}$ we will often omit the label k for readability. The graphical language of compact closed props allows us to bend wires at will, treating them as unoriented edges between the connection points of individual components. It also allows the introduction of "right-to-left" versions of each generator in our diagrammatic syntax. We explicitly introduce these counterparts as syntactic sugar, since they will be used in subsequent sections, where we refine our equational theories.

 (10)

The associated behaviour, in each case, is simply the opposite relation: for example,

$$[\![-\multimapinv]\!] = \{(n + m, \left(\begin{smallmatrix} n \\ m \end{smallmatrix} \right)) \mid n, m \in \mathbb{N}\}.$$

2.2 Examples

We conclude this section by taking a closer look at the examples featured in the Introduction.

Fibonacci. First, consider fib (2). As explained in the Introduction, it is a signal flow graph but it is *also* a diagram in Circ_s. Our structural rules define a transition system with states pairs $(fib, \binom{n}{m})$. Assuming that n refers to the state of the left register and m to the right one, we have:

$$(fib, \binom{n}{m}) \xrightarrow{\ i\ }_{o} (fib, \binom{n'}{m'}) \text{ iff } n' = i + m \text{ and } o = m' = n' + n. \tag{11}$$

Indeed, the content of the right register (m) is fed back through cap and cup, added to the value on the left port (i) and copied: one copy is stored as next value of the left register ($n' = i + m$) and the other is added to the current contents of the left register (n). The result ($n' + n$) is copied: one copy is sent to the right port ($o = n' + n$), the other is stored in the right register ($m' = n' + n$). The result is a Fibonacci relationship between the signal streams on the left and right wires: e.g., if both registers are initialised with 0, the stream $10000\ldots$ on the left is related to $12358\ldots$ on the right.

It is worth observing that the semantics of the signal flow calculus is the same as the one given by the rules in (6) and (8), but with n, m (and $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{v}$) ranging over (vectors of) real numbers, rather than the naturals. The behaviour described by (11) is thus a subset–containing exactly the positive integer behaviours—of the behaviours of (2) *when considered as a signal flow graph.* Next we consider three examples whose behaviour diverges conceptually from that of signal flow graphs.

Less than or equal to. Take the diagram c_1 of Circ from (3). Using (6), it is easy to see that $c_1 \xrightarrow{\ m\ }_{n} c_1$ iff there is $m' \in \mathbb{N}$ with $m + m' = n$. In other words, $c_1 \xrightarrow{\ m\ }_{n} c_1$ iff $m \le n$. Its behaviour is thus $[\![c_1]\!] = \{(m, n) \mid m \le n\}$. Observe that, if instead we interpret the operational semantics on $\mathbb{R}$ in place of $\mathbb{N}$, then since $\mathbb{R}$ has additive inverses, $[\![c_1]\!]$ collapses to the total relation $[\![-\bullet \quad \bullet-]\!] = \mathbb{R}^2$.

Petri net places. Our next example is c_2 from (3). This diagram of Circ_s captures the behaviour of a place in a Petri net: for all $i, o, m, m' \in \mathbb{N}$, $(c_2, m) \xrightarrow{\ i\ }_{o} (c_2, m')$ iff $m' = m - o + i$ and $o \le m$. The register stores the current number of available tokens (m), and, at each step, up to that number of tokens ($o \le m$) are output on the right. Those which are not output ($m - o$) are fed back around the cup and cap and summed with any incoming tokens (i) and stored back in the register. Again, the semantics of the signal flow calculus does not capture this behaviour: in the presence of additive inverses, any o may be output on the right for any input i. Concretely, if m is stored in the register, we may observe any o on the right when $m - o$ (possibly negative) is sent around the feedback.

3 ADDITIVE RELATIONS

In this section we provide an equational characterisation for semantic equivalence of Circ. The first step (Section 3.1) is to give an algebraic account of the entities in the image of $[\![\cdot]\!]$, through the concept of (finitely generated) additive relation (Definition 8). Section 3.2 proposes a (fully) complete system of equations for additive relations: paired with the syntax Circ, these equations form the *resource calculus.* In Section 3.3 we recall the diagrammatic calculus of matrices over $\mathbb{N}$ on which we rely throughout. Finally Section 3.4 is devoted to proving that the resource calculus axiomatises additive relations: as expected, *completeness* (semantic equivalence $\Rightarrow$ equational equality) is the challenging part.

3.1 The prop of finitely generated additive relations

DEFINITION 8. *An* additive relation *(over* $\mathbb{N}$*) of type* $k \to l$ *is a subset* $R \subseteq \mathbb{N}^k \times \mathbb{N}^l$ *such that (i)* $(\mathbf{0}, \mathbf{0}) \in R$ *and (ii) if* $(\mathbf{a}, \mathbf{b}), (\mathbf{a}', \mathbf{b}') \in R$ *then* $(\mathbf{a} + \mathbf{a}', \mathbf{b} + \mathbf{b}') \in R$.
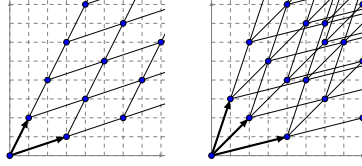
Fig. 3. Two additive relations.

If $R, R' : k \to l$ are additive relations of the same type, then both the intersection $R \cap R'$ and the Minkowski sum $R + R' = \{(\mathbf{a} + \mathbf{a}', \mathbf{b} + \mathbf{b}') \mid (\mathbf{a}, \mathbf{b}) \in R \text{ and } (\mathbf{a}', \mathbf{b}') \in R'\}$ are additive relations. Every pair $(\mathbf{a}, \mathbf{b}) \in \mathbb{N}^k \times \mathbb{N}^l$ generates an additive relation $\langle (\mathbf{a}, \mathbf{b}) \rangle = \{(p\mathbf{a}, p\mathbf{b}) \mid p \in \mathbb{N}\}$. More generally, for a finite set $G = \{(\mathbf{a}_1, \mathbf{b}_1), \cdots, (\mathbf{a}_p, \mathbf{b}_p)\}$ of points in $\mathbb{N}^k \times \mathbb{N}^l$, we write $\langle G \rangle$ for the additive relation

$$\langle G \rangle = \langle (\mathbf{a}_1, \mathbf{b}_1) \rangle + \cdots + \langle (\mathbf{a}_p, \mathbf{b}_p) \rangle = \left\{ \sum_{i=1}^{p} n_i (\mathbf{a}_i, \mathbf{b}_i) \mid n_1, \ldots, n_p \in \mathbb{N} \right\}. \tag{12}$$

We are interested in those additive relations that are finitely generated.

DEFINITION 9. *An additive relation* $R \colon k \to l$ *is* finitely generated *(f.g.) if there exists a finite set of vectors* $\{(\mathbf{a}_1, \mathbf{b}_1), \cdots, (\mathbf{a}_p, \mathbf{b}_p)\}$ *such that* $R = \langle (\mathbf{a}_1, \mathbf{b}_1) \rangle + \cdots + \langle (\mathbf{a}_p, \mathbf{b}_p) \rangle$.

EXAMPLE 10. *The two pictures of Figure 3 represents the additive relations of type* $1 \to 1$ *generated by* $\{(1, 2), (3, 1)\}$ *and* $\{(1, 3), (2, 2), (4, 1)\}$ *respectively.*

Unlike linear relations, additive relations cannot all be expressed as sets of linear combinations of a finite number of vectors: not all are *finitely generated*.

EXAMPLE 11. *The additive relation* $\{(m, n) \in \mathbb{N}^2 \mid m > n\} \cup \{(0, 0)\}$ *is not f.g.*

Henceforward, when we say "additive relation", we mean *f.g.* additive relation.

It is useful to think of additive relations as forming a sub-prop AddRel of $\mathsf{Rel}_\mathbb{N}$ (Definition 6). For this to make sense, we need to verify that they are closed under composition and monoidal product. The case of monoidal product is straightforward: if $R : k \to l$ and $R' : k' \to l'$ are f.g. additive relations with generating sets $\{(\mathbf{a}_1, \mathbf{b}_1), \cdots, (\mathbf{a}_p, \mathbf{b}_p)\}$ and $\{(\mathbf{a}'_1, \mathbf{b}'_1), \cdots, (\mathbf{a}'_q, \mathbf{b}'_q)\}$ respectively, $R \oplus R'$ has generating set $\left\{ \left(\binom{\mathbf{a}_i}{\mathbf{a}'_j}, \binom{\mathbf{b}_i}{\mathbf{b}'_j} \right) \mid 1 \le i \le p, \text{ and } 1 \le j \le q \right\}$. The case of composition also goes through, but it is non-trivial: other semirings, like the tropical semiring, do not have this closure property.

PROPOSITION 12. *The composition of two finitely generated additive relations is finitely generated.*

PROOF. Suppose that f.g. additive relations $R \colon k \to l$ and $S \colon l \to m$ have respective generating sets $\{(\mathbf{a}_1, \mathbf{b}_1), \cdots, (\mathbf{a}_p, \mathbf{b}_p)\}$ and $\{(\mathbf{c}_1, \mathbf{d}_1), \cdots, (\mathbf{c}_q, \mathbf{d}_q)\}$. We will find a generating set for $R \, ; S$. Let $U = \binom{U_k}{U_l}$ and $V = \binom{V_l}{V_m}$ be the $(k + l) \times p$ and $(l + m) \times q$ matrices whose columns are the generating vectors of R and S, respectively. By Dickson's lemma [Dickson 1913], the set $\{(\mathbf{e}, \mathbf{f}) \in \mathbb{N}^p \times \mathbb{N}^q \mid U_l \mathbf{e} = V_l \mathbf{f}\}$ has finitely many minimal elements $(\mathbf{e}_1, \mathbf{f}_1), \ldots, (\mathbf{e}_d, \mathbf{f}_d)$. Then $\{(U_k \mathbf{e}_1, V_m \mathbf{f}_1), \ldots, (U_k \mathbf{e}_d, V_m \mathbf{f}_d)\}$ generates $R \, ; S$. □

The semantics of the generating components of Circ are all additive relations. Therefore, the semantics $[\![-]\!]$ in (7) actually defines a prop morphism $[\![-]\!] \colon \mathsf{Circ} \to \mathsf{AddRel}$.

EXAMPLE 13. *There are two examples worth highlighting, because they play an important role in the axiomatisation given in Section 3.2. First, we have*

$$\llbracket \text{―} \rrbracket = \left\{ \left(\left(\begin{smallmatrix} a \\ b \end{smallmatrix} \right), \left(\begin{smallmatrix} c \\ d \end{smallmatrix} \right) \right) \mid a = c \text{ and } a + b = c + d \right\} .$$

Substituting a for c and using the cancellativity property of $\mathbb{N}$, $a + b = a + d$ *implies* $b = d$. *The above relation is, therefore, equal to the identity relation on* $\mathbb{N}^2$. *In other words*

$$\llbracket \text{―} \rrbracket = \llbracket \text{―} \rrbracket.$$

Next, we have $\llbracket \text{―} \rrbracket = \{(a, b) \mid$ *there exist* c, d *such that* $a + c = b + d\}$. *Clearly, any pair of naturals has (infinitely many) natural numbers greater or equal to both. If follows that the relation is total, in other words:* $\llbracket \text{―} \rrbracket = \llbracket \text{―}\bullet \quad \bullet\text{―} \rrbracket$.

3.2 Axiomatisation

This section illustrates the proposed axiomatisation for additive relations, as shown in Figure 4. We call the resulting theory the *resource calculus*.

DEFINITION 14. *The prop* Rc *is the quotient of* Circ *by the equations in Figure 4. For* $c, d \colon k \to l$ *in* Circ, *we write* $c =_{\mathrm{Rc}} d$ *when they are equal according to the equations, and thus the same arrow in* Rc.

Some of the equations use the syntactic sugar introduced in (10). We now remark about each of the equation blocks in Figure 4.

- In the first block, both the black and white structures are commutative monoids and comonoids, expressing fundamental properties of addition and copying.
- In the second block, the white monoid and black comonoid interact as a bimonoid. Bimonoids are one of two canonical ways that monoids and comonoids interact, as shown in [Lack 2004].
- In the third block, the black monoid and comonoid form an extraspecial Frobenius monoid. The Frobenius equations (fr 1) and (fr 2) are a famous algebraic pattern which establishes a bridge between algebraic and topological phenomena, see [Carboni and Walters 1987; Coecke and Kissinger 2017; Kock 2003]. The "extraspecial" refers to the two additional equations, the *special* equation (●-sp) and the *bone* equation (●-bo). The Frobenius equations, together with the special equation, are the another canonical pattern of interaction between monoids and comonoids identified in [Lack 2004]. Together with the bone equation, the set of four equations characterises *corelations*, see [Bruni and Gadducci 2001; Coya and Fong 2017; Zanasi 2016].
- In the fourth block, deviating from the equational theory of linear relations, the white monoid-comonoid pair forms a special bimonoid, not a Frobenius monoid. Here, the Frobenius structure—if present—would play the role of assuming the presence of additive inverses, see [Bonchi et al. 2017b; Coecke et al. 2012]. Since we are dealing with the natural numbers, the structure satisfies only the bimonoid equations. Of key interest here are (○-bi(co)un), reflecting non-negativity of the additive monoid: $a + b = 0 \implies a = b = 0$.
- In the fifth block, two more equations concern properties of addition in $\mathbb{N}$. They capture two important properties of the natural numbers, as explained in Example 13. Note that the second can be seen as a unary version of the first. In particular, (can) is one of the two equations that axiomatises the notion of complementary observables in categorical approaches to quantum mechanics [Coecke and Duncan 2011].

- Finally, the last equation is an axiom scheme, parametrised over $n \in \mathbb{N}$. It uses the following syntactic sugar, along with the obvious mirror image versions, defined recursively:

$$(13)$$

to represent the additive relations of the form $\langle (1, n) \rangle$. Equations (n-inv) are one half of the equations that concerns such sugars in graphical linear algebra [Bonchi et al. 2017d]. Their symmetric variant is *not* present since they are not sound for AddRel and rely on the ability to *divide* by non-zero scalars, as in the rational numbers.

REMARK 15. *The resource calculus is closely related to a simple model of (the multiplicative exponential fragment of) linear logic. Indeed, the coKleisli category of the multiset comonad on* Rel*, is a subprop of* AddRel*. It corresponds precisely to the theory of the white bimonoid—that constitutes the linear part of the resource calculus—with the black monoid. Additive relations or diagrams of* Rc *are more general in that they allow copying and deleting of resources through the use of the black comonoid.*

PROPOSITION 16. *There is a prop morphism* $I \colon$ Rc $\to$ AddRel *mapping* c *to* $[\![c]\!]$.

PROOF. One can readily verify—as we have already done for (can) and (up) in Example 13—that the equations are sound. □

Remarkably, I is actually an isomorphism of props. In other words, Figure 4 constitutes a sound and fully complete axiomatisation of additive relations.

THEOREM 17. $I \colon$ Rc $\to$ AddRel *is an isomorphism of props.*

The rest of the section is dedicated to proving this theorem. First, we have to take a small detour through the prop of matrices with natural number coefficients.

3.3 Picturing matrices

Our proofs exploit an existing axiomatisation of $\mathbb{N}$-matrices. Here we recall the basics.

DEFINITION 18. *Let* $\mathrm{Mat}_{\mathbb{N}}$ *be the prop in which morphisms* $k \to l$ *are* $l \times k$ *matrices with coefficients in* $\mathbb{N}$*, the monoidal product is the direct sum and composition is matrix multiplication.*

PROPOSITION 19. *There is a monoidal embedding* $\iota : \mathrm{Mat}_{\mathbb{N}} \hookrightarrow$ AddRel.

PROOF. The functor ι maps an $l \times k$ matrix A to the graph of the corresponding linear map: $\iota(A) = \{(\mathbf{x}, A\mathbf{x}) \mid \mathbf{x} \in \mathbb{N}^k\}$. Showing that ι is faithful is straightforward. □

Let Bi be the prop of *bimonoids*: arrows are the string diagrams on generators {—●⟨, —●, ⟩○—, ○—} quotiented by the equations in lines 1, 2 and 4 of Figure 4. This theory is well-known to be sound and fully complete for $\mathrm{Mat}_{\mathbb{N}}$, as the next theorem states.

THEOREM 20 (E.G. [ZANASI 2015, PROP. 3.9]). Bi *is isomorphic to* $\mathrm{Mat}_{\mathbb{N}}$.

To develop some intuition for this correspondence, let us demonstrate how matrices are represented diagrammatically. An $l \times k$ matrix A corresponds to a string diagram with k wires on the left and l wires on the right—the left ports can be interpreted as the columns and the right ports as the rows of A. The left jth port is connected to the ith port on the right through n wires whenever
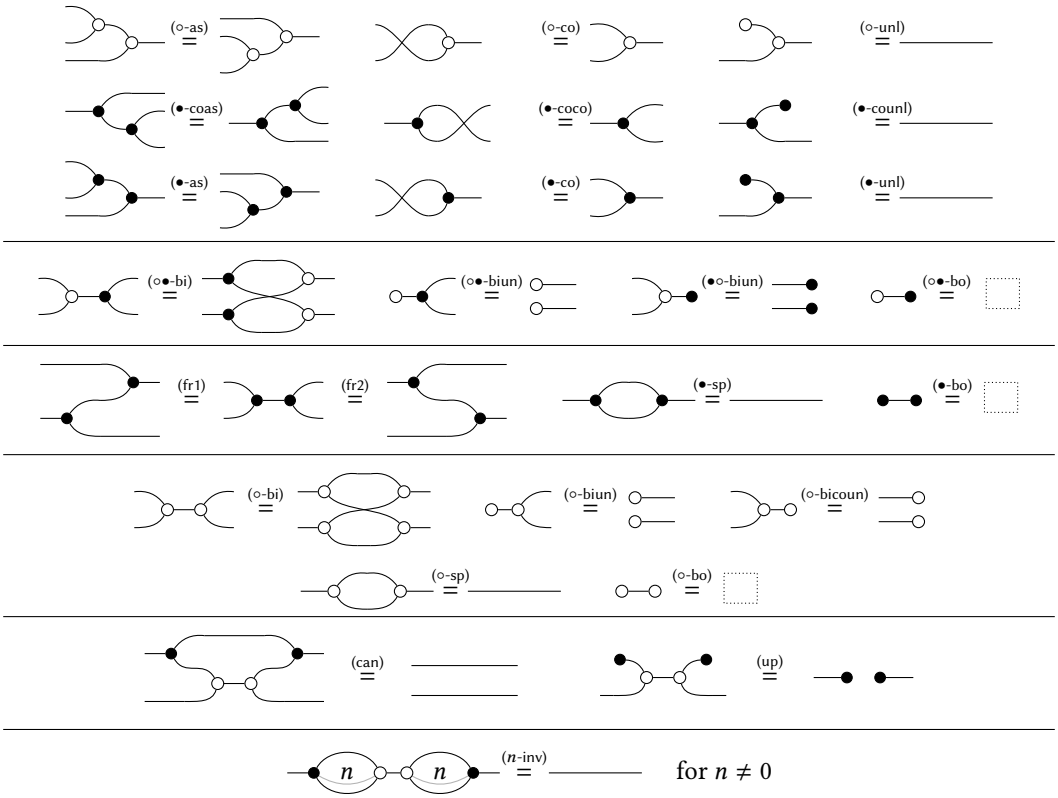
Fig. 4. Axioms of the resource calculus.

A_{ij} is a nonzero scalar $n \in \mathbb{N}$. When the A_{ij} entry is 0, they are disconnected. For example,

$$\text{The matrix } A = \begin{pmatrix} 3 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad \text{is represented by} \qquad\qquad (14)$$

Conversely, given a diagram, we recover the matrix by counting paths from left to right ports.

3.4 Full Completeness

The remainder of this section explains the proof of Theorem 17. Note, that by definition of I, we immediately obtain a characterisation of semantic equivalence for Circ:

COROLLARY 21 (FULL COMPLETENESS). *For c, d in* Circ, *$[\![c]\!] = [\![d]\!]$ if and only if $c =_{\mathrm{Rc}} d$. Also, all f.g. additive relations are in the image of $[\![\cdot]\!]$.*

Since $I\colon \mathrm{Rc} \to \mathrm{AddRel}$ is a prop morphism, it is the identity on objects. Thus in order to show that it is an isomorphism it is enough to show that it full and faithful.

Fullness. To see that I is full we simply have to notice that, for a f.g. additive relation $R\colon k \to l$ there exists $k \in \mathbb{N}$ and a $(k + l) \times p$ matrix A that represents R, in the sense that $(\mathbf{a}, \mathbf{b}) \in R$ iff there

is $\mathbf{x} \in \mathbb{N}^p$ satisfying $A\mathbf{x} = \binom{\mathbf{a}}{\mathbf{b}}$. Clearly, $\langle(\mathbf{a}_1, \mathbf{b}_1)\rangle + \cdots + \langle(\mathbf{a}_l, \mathbf{b}_l)\rangle$ is represented by $\begin{pmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_p \\ \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_p \end{pmatrix}$ and the generating set may be recovered from every matrix by taking its set of columns.

Next, as we saw in Section 3.3, any matrix can be represented in the prop of commutative bimonoids, which is a sub-theory of Rc: we can use the isomorphism of Theorem 20 and the embedding of Bi into Rc to obtain a diagram for the matrix A representing R. We use the compact structure component $\overline{k\,}$ (cf. Definition 7) to bend the last k wires, and the unit of the Frobenius monoid to "universally quantify" over the p wires on the left. The resulting diagram N_A is shown below, and it follows that $R = I(N_A)$.

$$N_A := \quad \bullet\!\!-\!\!\boxed{A}\!\!-\!\!\overset{l}{} \underset{k}{} \tag{15}$$

Faithfulness. To prove faithfulness of I (and, therefore, obtain completeness of our equational theory), we define a normal form for arrows of Rc. Normal forms are closely related to *bases* of additive relations, in a sense that we now explain. The usual notion of linear dependence can be adapted to the context of linear algebra over the semiring $\mathbb{N}$. We say that a set of vectors is *dependent* if one of them is equal to a linear combination (with coefficients from $\mathbb{N}$) of the others. Otherwise the vectors are said to be *independent*.

DEFINITION 22. *A basis of an additive relation is an independent generating set.*

Differently from linear relations, bases of additive relations are unique.

PROPOSITION 23. *Every additive relation admits a unique basis, called its Hilbert basis.*

PROOF. For an additive relation R, the elements of its Hilbert basis are the irreducible elements of R, i.e., those $(\mathbf{a}, \mathbf{b}) \in R \setminus (\mathbf{0}, \mathbf{0})$ that cannot be expressed as a non-trivial sum of two other elements. This can be easily checked by induction. □

Normal forms are N_A, as in (15), where A is a matrix arising from an independent generating set. By Proposition 23 (and an implicit appeal to the completeness result for matrices, see Theorem 20 in Section 3.3), diagrams in normal form correspond *uniquely* to additive relations—up to a permutation of the columns of their representing matrix.

By Proposition 23, $I(c) = I(d)$ if and only if $I(c)$ and $I(d)$ have the same Hilbert basis. Appealing to the completeness result for matrices (Theorem 20), this is true when the representing matrices of c and d have the same *set* of independent columns. But these two matrices may still differ by a permutation of their columns. To conclude the proof we need to show that two diagrams in normal form that differ only by a permutation of the columns of their representing matrix, are equal. This is a consequence of the commutativity of $\,\rule{0pt}{0pt}\!\!\supset\!\!\circ\!\!-\,$:

$$\tag{16}$$

$$\tag{17}$$

The general result follows from this by induction.

This proves the faithfulness of I, provided we can show that every diagram of Rc is equal to one in normal form. This procedes in two steps. First, we can show by structural induction that every diagram can be rewritten into an equal diagram of the form of n_A in (15), for an abitrary matrix A. We omit here the case by case analysis of the rewriting procedure. [More detail?] RP.

While every matrix determines a unique additive relation, the converse is not true: there may be multiple matrices representing a given additive relation as their columns may be dependent. To be able to prove that diagrams are equal to their normal form we need to make sure that redundant generators can be eliminated diagrammatically from the axioms of the prop. We need to check that columns that are sums of other columns of the same matrix can be eliminated, using only the equations of Rc. It is instructive to illustrate how redundant generators can be eliminated in the following simple example.

EXAMPLE 24. *We start with a three column matrix in which the third is the sum of the first two columns. The derivation concludes with the third column eliminated.*
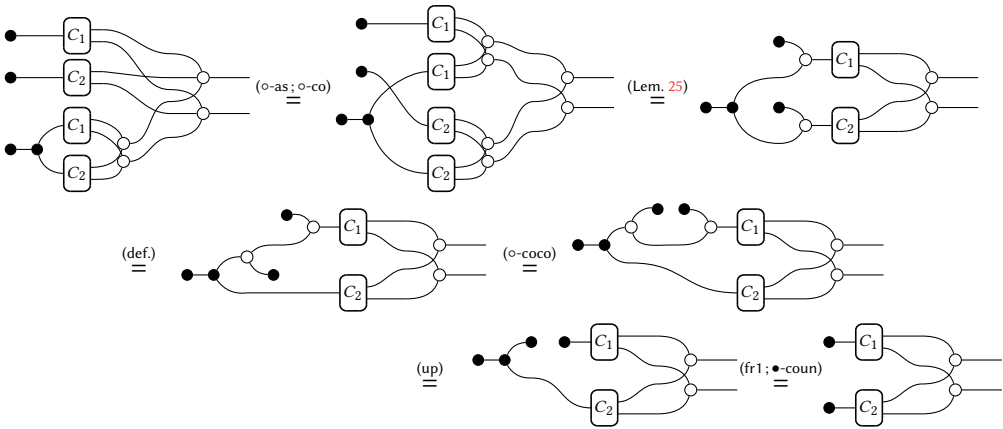
The key step is the use of the (up) equation which we had not used previously. The general case proceeds exactly as in the example above—we just need the following three lemmas to eliminate redundant sums of columns of arbitrary size.

LEMMA 25. *If A is the diagram of a $l \times k$ matrix, then*

PROOF. We could prove this by structural induction but it is easier to appeal to the completeness result of Theorem 20 and the fact that $A(\mathbf{x} + \mathbf{y}) = A\mathbf{x} + A\mathbf{y}$, for all $\mathbf{x}, \mathbf{y} \in \mathbb{N}^k$. □
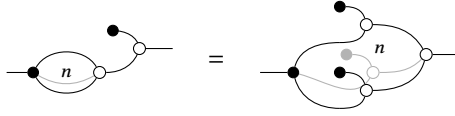
The next two lemmas are proven in Appendix A.1.

LEMMA 26.

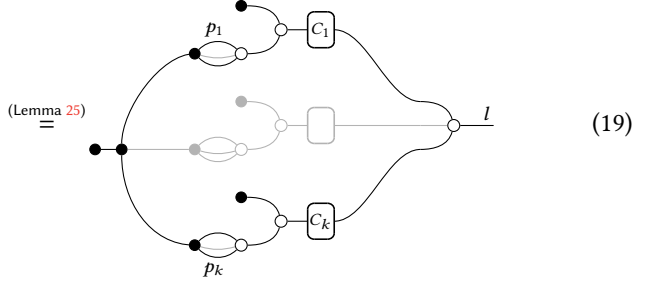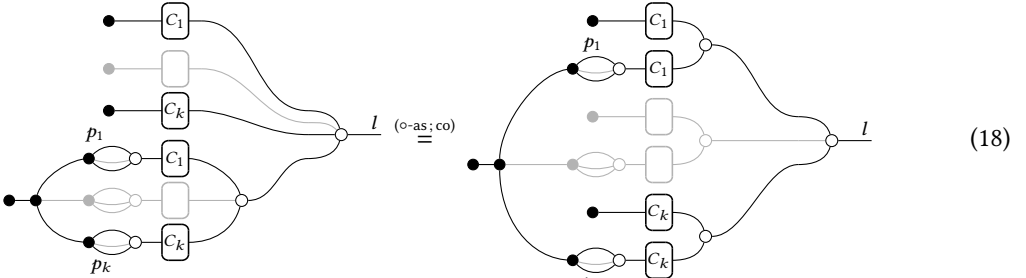The next lemma allows us to handle weighted sums by reducing them to the case of lemma 26.
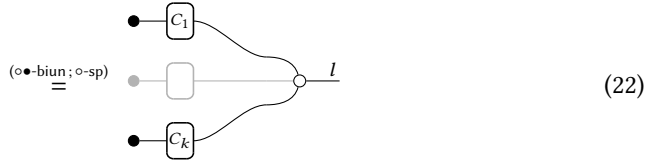
LEMMA 27. *For $n \geq 1$,*

We can now proceed exactly as in Example 24 to delete redundant generating columns in the general case.

$$(18)$$

$$(19)$$

$$(20)$$

$$(21)$$

$$(22)$$
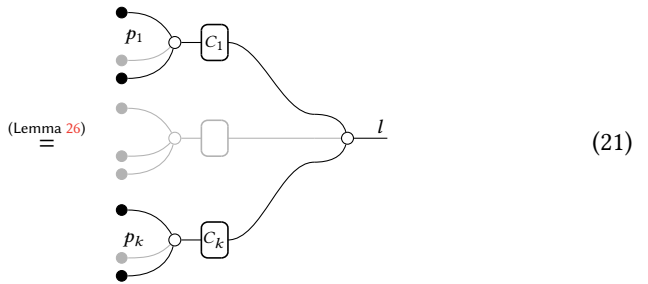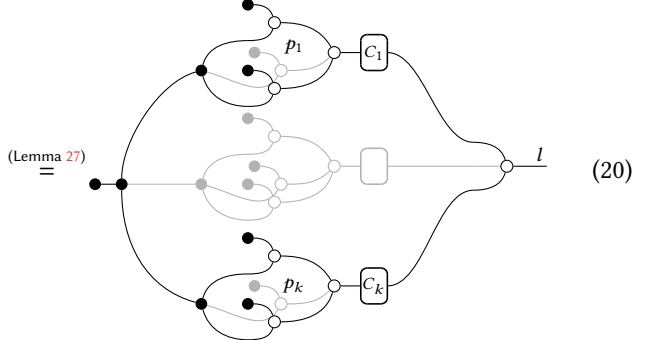
4 STATEFUL PROCESSES

This section is the conceptual bridge between the stateless calculus Rc and its stateful extension. We adopt a more abstract perspective to demonstrate that:

(a) from any prop T one may obtain a prop St(T) in which morphisms are stateful processes;

(b) in the presence of a compact closed structure on T, moving to St(T) amounts to extending T with a single generator —$\boxed{x}$— with no equations; and finally

(c) turning to the concrete case of the resource calculus Rc, the semantics of —$\boxed{x}$— given in (8), is canonical and induced by the abstract St($-$) construction.

From this, it follows that the stateful resource calculus is a sound and fully complete axiomatisations of the prop St(AddRel).

4.1 Adding state

From a given "stateless" prop, the construction St($-$) in Definition 28 below results in a prop in which the morphisms are stateful processes. This "state bootstrapping" is a well-known technique that appears in several places in the literature, e.g. in the setting of cartesian bicategories [Katis et al. 1997] and geometry of interaction [Hoshino et al. 2014].

DEFINITION 28 (PROP OF STATEFUL PROCESSES [KATIS ET AL. 1997]). *Let* T *be a prop. Define* St(T) *as the prop where:*

- *morphisms* $k \to l$ *are pairs* (s, c) *where* $s \in \mathbb{N}$ *and* $c : s + k \to s + l$ *is a morphism of* T*, quotiented by the smallest equivalence relation including every instance of*

$$\begin{array}{ccc} s & \quad c \quad & s \\ k & & l \end{array} \quad \sim \quad \begin{array}{ccc} s & \sigma \; c \; \sigma^{-1} & s \\ k & & l \end{array}$$

for a permutation $\sigma : s \to s$;

- *the composition of* $(s, c) : k \to l$ *and* $(t, d) : l \to p$ *is* $(s + t, e)$ *where* e *is the arrow of* T *given by*

- *the monoidal product of* $(s_1, c_1) : k_1 \to l_1$ *and* $(s_2, c_2) : k_2 \to l_2$ *is* $(s_1 + s_2, e)$ *where* e *is given by*

- *the identity on* j *is* $(0, id_j)$ *and the symmetry of* n, m *is* $(0, \sigma_{n,m})$.

For example, thinking of ordinary functions between sets as stateless transducers, we can construct a category of (deterministic) transducers by considering functions whose output does not just depend on their input, but also on a set of internal states that is updated at every application. In other words, a stateful function or transducer of type $A \to B$ is a function $f : S \times A \to S \times B$.

The equivalence relation of Definition 28 ensures that internal state remains anonymous. For instance, in the case of transducers, the set of states S ought to serve only as a reference for internal state: we equate transducers that only differ by a bijective relabelling of their set of states. This

is, therefore, a syntactic form of equivalence similar in flavour to α-equivalence, since it discards intensional representation details and does not affect the dynamics of stateful processes.

Let X be the prop whose arrows, as in Circ, are string diagrams arising from a signature, which in this case only contains a single generator $—\boxed{x}—$. Given a prop T, in the category **PROP** one can form the coproduct T + X of T and X, see e.g. [Zanasi 2015, §2.3] for the formal definition. Intuitively, when T has a syntactic presentation, as is the case for AddRel (Theorem 17), the prop T + X is simple to describe: it arises by freely pasting together sequentially and in parallel the string diagrams of T with the single generator $—\boxed{x}—$ of X.

In Theorem 30 below, we give a simple characterisation of St(T): it is *isomorphic* to T+X, assuming that T has compact closed structure. This result relies on the simple technical lemma below, which is a useful characterisation of the morphisms of T + X.

LEMMA 29 (TRACE CANONICAL FORM). *Suppose that* T *is a compact closed prop. For every* $d : k \to l$ *in* T + X *there exists a morphism* $c : s + k \to s + l$ *of* T *such that*

$$k —\boxed{d}— l \quad = \quad k —\overset{s}{\overgroup{\boxed{c}\,\boxed{x}}}— l \tag{23}$$

PROOF. In Appendix A.2. □

To exhibit the isomorphism St(T) ≅ T + X, we define two monoidal functors, $R: X \to St(T)$ and $Z: T \to St(T)$. For X it suffices to say where its one generator is mapped: we set $R(—\boxed{x}—) = (1, \times)$. The second functor $Z: T \to St(T)$ is defined as $Z(t) = (0, t)$ for all arrows t of T. Let

$$F := \langle Z, R \rangle : T + X \to St(T).$$

Intuitively, we can think of F as an operation on the diagrams that "cuts the $—\boxed{x}—$ out" and pulls the wires to which they were connected into state-holding wires. Indeed, since $—\boxed{x}—$ is mapped to $\times$, we have, for a morphism of T + X in trace canonical form:

$$F\left(k —\overset{s}{\overgroup{\boxed{d}\,\boxed{x}}}— l \right) = \overset{s}{\underset{k}{=}} —\boxed{d}— \overset{s}{\underset{l}{}} = \overset{s}{\underset{k}{=}} —\boxed{d}— \overset{s}{\underset{l}{}} \tag{24}$$

THEOREM 30. *If* T *is a compact closed prop then* $F: T + X \to St(T)$ *is an isomorphism.*

PROOF. We construct the inverse of F explicitly. Let $G: St(T) \to T + X$ be defined by

$$G\left(\overset{s}{\underset{k}{=}} —\boxed{d}— \overset{s}{\underset{l}{}} \right) \quad = \quad k —\overset{s}{\overgroup{\boxed{d}\,\boxed{x}}}— l \tag{25}$$

Functoriality follows from the compact structure; the argument is similar to the proof of Lemma 29. Thus, while F "cuts out the $—\boxed{x}—$", G takes a stateful d and feeds back the state guarded by $—\boxed{x}—$".

We now prove that F and G are inverse. Given d in St(T), the fact that $FG(d) = d$ is immediate by (24). Conversely, given c in T + X, we use the conclusion of Lemma 29 to obtain d in T such that

$$k —\boxed{c}— l \quad = \quad k —\overset{s}{\overgroup{\boxed{d}\,\boxed{x}}}— l$$

Thus, we have

$$GF(c) = G\left(\begin{smallmatrix} s \\ k \end{smallmatrix} \; —\!\boxed{d}\!— \; \begin{smallmatrix} s \\ l \end{smallmatrix}\right) = \quad k \; —\!\boxed{d}\,\boxed{x}\!— \; l \quad = \quad k \; —\!\boxed{c}\!— \; l$$

□

4.2 Axiomatising state

With the abstract landscape clear, we are now ready to prove soundness and full completeness of the equational theory in Figure 4 for the operational equivalence of stateful circuit diagrams.

DEFINITION 31. *The prop* Rc_s *is the quotient of* $Circ_s$ *by* $=_{Rc}$.

Since the equations of Figure 4 do not involve registers, Rc_s is simply $Rc + X$.

By Theorem 30, one has immediately that Rc_s is isomorphic to $St(Rc)$ which is in turn, by Theorem 17, isomorphic to $St(AddRel)$. The isomorphism $Rc_s \cong St(AddRel)$ is witnessed by:

$$I_s := \; Rc + X \xrightarrow{\langle Z, R \rangle} St(Rc) \xrightarrow{St(I)} St(AddRel)$$

where $St(I)$ is the obvious extension of the isomorphism $I : Rc \to AddRel$.

To conclude that $=_{Rc}$ characterises operational equivalence, we only need to check that I_s coincides with the operational semantics. To make the formulation smoother, we implicitly identify any diagram $(n, c) \in St(AddRel)[j, k]$ with the relation $c \subseteq \mathbb{N}^n \times \mathbb{N}^j \times \mathbb{N}^n \times \mathbb{N}^k$.

PROPOSITION 32. *For all diagrams c in* Rc_s, $[\![c]\!] \sim I_s(c)$.

PROOF. The proof goes by induction on Rc_s. The only non-trivial case is $—\!\boxed{x}\!—$:

$$St(I)(\langle Z, R \rangle(—\!\boxed{x}\!—)) = St(I)(R(—\!\boxed{x}\!—)) = St(I)((1, \asymp)) = (1, \{\left(\begin{smallmatrix} n \\ m \end{smallmatrix}\right), \left(\begin{smallmatrix} m \\ n \end{smallmatrix}\right) \mid n, m \in \mathbb{N}\}) \sim [\![—\!\boxed{x}\!—]\!]$$

□

We can then conclude full completeness.

COROLLARY 33 (FULL COMPLETENESS). *For c, d in* $Circ_s$, $[\![c]\!] = [\![d]\!]$ *iff $c =_{Rc_s} d$. Also, all arrows of* $St(AddRel)$ *are in the image of* $[\![\cdot]\!]$.

Notice that the results in this section justify the semantics of $—\!\boxed{x}\!—$ given in (8). The one-place buffer is a canonical operation for state, justified by the isomorphism of Theorem 30 and the relational interpretation of Proposition 32. By justifying the stateful extensions, the results of this section complement those of Section 3: we have arrived at a complete justification for the canonicity of *all* of the syntactic operations in $Circ_s$ and its operational semantics, as given in Section 2. Moreover, in the subsequent section we show that $—\!\boxed{x}\!—$ adds enough expressivity to capture a classical model of concurrency: Petri nets.

5 PETRI NETS

We use Petri nets as a test-bed for the expressiveness of the stateful resource calculus.

DEFINITION 34 (PETRI NETS AND THEIR SEMANTICS). *A Petri net $\mathcal{P} = (P, T, {}^\circ-, -^\circ)$ consists of a finite set of places P, a finite set of transitions T, and functions ${}^\circ-, -^\circ : T \to \mathbb{N}^P$. Given $\mathbf{a}, \mathbf{b} \in \mathbb{N}^P$, we write $\mathbf{a} \to \mathbf{b}$ if there exists $\mathbf{t} \in \mathbb{N}^T$ such that ${}^\circ\mathbf{t} \leq \mathbf{a}$ and $\mathbf{b} = \mathbf{a} - {}^\circ\mathbf{t} + \mathbf{t}^\circ$. The (step) operational semantics of $\mathcal{P}$ is the relation $[\![\mathcal{P}]\!] = \{(\mathbf{a}, \mathbf{b}) \mid \mathbf{a} \to \mathbf{b}\} \subseteq \mathbb{N}^P \times \mathbb{N}^P$.*

EXAMPLE 35. *Consider the Petri net displayed below with the usual graphical notation: circles represent places, and squares transitions.*
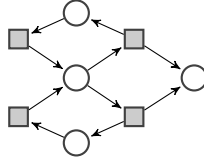
Petri nets are combinatorial objects, usually studied monolithically rather than compositionally. We introduce a syntax Circ_p that *(i)* compiles into the resource calculus in a straightforward way and *(ii)* whose *closed* diagrams (arrows in $\mathrm{Circ}_p[0, 0]$) capture precisely the behaviour of Petri nets. Moreover, arbitrary diagrams (arrows in $j \to k$) are open nets in the style of [Bruni et al. 2013].

The syntax is (4) extended with one extra generator $\rightarrow\!\bigcirc\!- : (1, 1)$ intuitively representing a Petri net place. Formally, its operational behaviour is given by the following rule.

$$\frac{o \le m}{(\rightarrow\!\bigcirc\!-, m) \xrightarrow[o]{i} (\rightarrow\!\bigcirc\!-, m - o + i)} \quad (26)$$

We use $[\![c]\!]$ to denote the resulting relation, defined analogously to (29). Like in Section 2, we consider diagrams obtained from the syntax to be morphisms of a prop Circ_p. It is useful to observe that Circ_p is the same as the coproduct $\mathrm{Circ} + \mathrm{Pl}$ in **PROP**, where Pl is the prop whose arrows are string diagrams on the signature $\{\rightarrow\!\bigcirc\!-\}$. The relevant equational theory is Petri := Rc + Pl, the quotient of Circ_p by $=_{\mathrm{Rc}}$ (Figure 4).

5.1 Encoding nets into Petri

We illustrate the role of Petri with the aid of an example.

Example 36. *The diagrams of* Petri$[0, 0]$ *corresponding to the net of Example 35 is:*
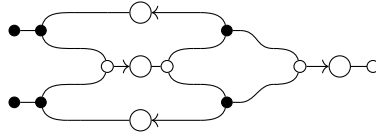
More generally, a place with multiple inputs and outputs is depicted as
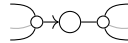
using $\rightarrow\!\!\!\bigcirc\!-$ *and* $-\!\!\bigcirc\!\!\!\leftarrow$ *, while transitions are represented with the help of* $\rightarrow\!\!\bullet\!-$ *and* $-\!\!\bullet\!\!\!\leftarrow$ *.*

Any ordinary Petri net $\mathcal{P}$ can be encoded as a diagram $d_\mathcal{P}$ in Petri$[0, 0]$. By choosing an ordering on places and transitions, the functions $^\circ\!-, -^\circ : T \to \mathbb{N}^P$ can be regarded as $\mathbb{N}$-matrices of type $|T| \to |P|$. Such matrices can be seen as special cases of additive relations (*cf.* Theorem 20, Section 3.3): let s and t be the diagrams in Rc corresponding to $^\circ\!-, -^\circ$ respectively. We let $d_\mathcal{P}$ be

That this assignement is well-defined is proven in Appendix A.3. It is easy to show that $\mathcal{P}$ and $d_\mathcal{P}$ have the same operational behaviour.

Proposition 37. *Given a Petri net $\mathcal{P}$, we have $[\![\mathcal{P}]\!] \sim [\![d_\mathcal{P}]\!]$.*

Proof. In Appendix A.3. □

Similarly, for every diagram $d \in \text{Petri}[0,0]$, one can construct a Petri net $\mathcal{P}_d$ with the following recipe: by Lemma 29, d can be written in trace canonical form, namely there exists a diagram $c \in \text{Rc}[p,p]$ such that (23) holds. The diagram c denotes an additive relation $I(c) \subseteq \mathbb{N}^p \times \mathbb{N}^p$ that, by Proposition 23, has a Hilbert basis. This basis can be represented as matrix $A \colon t \to p + p$ for some $t \in \mathbb{N}$ representing the dimension of the basis. The matrix A can be decomposed into two matrices $A_1, A_2 \colon t \to p$ such that $A = \left(\begin{smallmatrix} A_1 \\ A_2 \end{smallmatrix}\right)$. We define $\mathcal{P}_d := (p, t, A_2, A_1)$, that is p and t (seen as ordinal sets) are the set of places and transitions, A_1 and A_2 play the role of $\circ-$ and $-\circ$.

Again, it is easy to prove that the operational behaviour is preserved.

PROPOSITION 38. *For all $d \in \text{Petri}[0,0]$, $[\![d]\!] \sim [\![\mathcal{P}_d]\!]$*

PROOF. In Appendix A.3.   □

By virtue of Propositions 37 and 38 together, Petri nets and diagrams in $\text{Petri}[0,0]$ are in one-to-one correspondence modulo semantic equivalence.

5.2 Classifying stateful extensions

Recall that diagram c_2 (3) in the introduction behaves like a place of a Petri net. We now use it to embed Petri in Rc_s. The prop morphism $\mathcal{E}(-) \colon \text{Circ}_p \to \text{Circ}_s$ takes:

$$\mathcal{E}(\!\!-\!\!\bigcirc\!\!-\!\!) := \quad \text{} \tag{27}$$

and acts as the identity for the constants of Circ. Following the discussion in Section 2.2, it is immediate that $\mathcal{E}(\!\!-\!\!\bigcirc\!\!-\!\!)$ is exactly the behaviour defined by (26), that is

$$[\![\mathcal{E}(\!\!-\!\!\bigcirc\!\!-\!\!)]\!] = \{(m, i, m', o) \mid o \le m \text{ and } m' = m - o + i\} = [\![-\!\!\bigcirc\!\!-\!\!]\!] \tag{28}$$

A simple induction allows us to extend the correspondence (28) to all diagrams in Circ_p.

PROPOSITION 39. *Given arbitrary $c \in \text{Circ}_p$, $[\![\mathcal{E}(c)]\!] \sim [\![c]\!]$.*

The above proposition together with Corollary 33 guarantee the following.

COROLLARY 40. *For all diagrams c, d of Circ_p, $[\![c]\!] \sim [\![d]\!]$ if and only if $\mathcal{E}(c) =_{\text{Rc}} \mathcal{E}(d)$.*

We have shown that the stateful resources calculus is at least as expressive as Petri (and thus Petri nets). We shall now show that Petri is *strictly less expressive* than the stateful resource calculus, in the sense that not all transition systems definable by Rc_s can be expressed by diagrams in Petri. Intuitively, this is because the register of the resource calculus is *synchronous* whereas the place of Petri nets is *asynchronous*, namely it can always perform a transition that does not change its internal state. This property holds for all diagrams of the Petri calculus: Recall that we define the semantics of a diagram d of Circ_p as the relation:

$$[\![d]\!] := \{(s_1, a, s_2, b) \mid (d, s_1) \xrightarrow[b]{a} (d, s_2)\} \subseteq \mathbb{N}^R \times \mathbb{N}^k \times \mathbb{N}^R \times \mathbb{N}^l. \tag{29}$$

PROPOSITION 41. *Let d be in Circ_p. Then $(\mathbf{a}, \mathbf{0}, \mathbf{a}, \mathbf{0}) \in [\![d]\!]$ for all $\mathbf{a}$.*

PROOF. By induction. The only case of interest is $-\!\!\bigcirc\!\!-$, and $(a, 0, a, 0) \in [\![-\!\!\bigcirc\!\!-\!\!]\!]$ by (26).   □

Given that $(a, 0, a, 0) \notin [\![-\!\boxed{x}\!-]\!]$ unless $a = 0$, we immediately obtain:

COROLLARY 42. *For all diagrams d in Circ_p, $[\![d]\!] \not\sim [\![-\!\boxed{x}\!-]\!]$.*   □

The expressivity result strengthens the claim that $—\boxed{x}—$ is a "more canonical" way than $—\!\!\to\!\!\bigcirc\!—$ to introduce state to AddRel. Indeed, in Section 4.2 we have seen that Rc_s is isomorphic to St(AddRel). Now, because of Corollary 42, it cannot be the case that Petri is also isomorphic to St(Rc).

Moreover, we can use the example of $—\!\!\to\!\!\bigcirc\!—$ to show in a more precise sense how the resource calculus plays the role of a yardstick, guiding the space of design choices for stateful extensions of additive relations. Once we know that the behaviour $[\![—\!\!\to\!\!\bigcirc\!—]\!]$ of $—\!\!\to\!\!\bigcirc\!—$ is an arrow of St(AddRel), the encoding $\mathcal{E}(—\!\!\to\!\!\bigcirc\!—)$ of $—\!\!\to\!\!\bigcirc\!—$ in the resource calculus is completely determined. Using the isomorphisms $\langle Z, R \rangle \colon Rc_s \to St(Rc)$ and $St(I) \colon St(Rc) \to St(AddRel)$ from Section 4.1 we have:

$$St(I)^{-1}([\![—\!\!\to\!\!\bigcirc\!—]\!]) = \left(1, \underline{\overset{\frown}{\underset{\smile}{}}}\right) \text{ and}$$

$$\langle Z, R \rangle^{-1}\left(1, \underline{\overset{\frown}{\underset{\smile}{}}}\right) = \langle Z, R \rangle^{-1}\left(1, \underline{\overset{\frown}{\underset{\smile}{}}}\right) = \langle Z, R \rangle^{-1}\left(1, \underline{\overset{\frown}{\underset{\smile}{}}}\right)$$

$$= \langle Z, R \rangle^{-1}\left(1, \underline{\overset{\frown}{\underset{\smile}{}}}\right) = \left(\overset{\frown}{\underset{\boxed{x}}{}}\right) = \mathcal{E}(—\!\!\to\!\!\bigcirc\!—)$$

This computation evaluates $[\![—\!\!\to\!\!\bigcirc\!—]\!]$ both to a stateful diagram from Rc_s (the conclusion) and a pair in St(Circ) of a state and a stateless diagram (first equality). Notice the difference with the interpretation $(1, \times)$ that the isomorphism I_s gives to $—\boxed{x}—$ in St(Circ).

Intriguingly, the same kind of analysis can be performed to other stateful extensions of AddRel. One example from the literature is the *banking semantics* [Bruni et al. 2011, 2013] of Petri nets, which defines behaviour differently:

$$\frac{m + i = m' + o}{(—\!\!\to\!\!\boxed{b}—, m) \xrightarrow[o]{i} (—\!\!\to\!\!\boxed{b}—, m')} \, .$$

Applying the same computations to $[\![—\!\!\to\!\!\boxed{b}—]\!]$ defined in terms of the banking rule, one obtains following representations in St(Circ) and in Rc_s:

$$St(I)([\![—\!\!\to\!\!\boxed{b}—]\!]) = \left(1, \overset{\frown}{\underset{\smile}{}}\right) \text{ and } \langle Z, R \rangle^{-1} St(I)([\![—\!\!\to\!\!\boxed{b}—]\!]) = \left(\overset{\frown}{\underset{\boxed{x}}{}}\right)$$

For each notion of state, Figure 5 displays the associated diagrams in St(Rc) and Rc_s. The canonicity of $[\![—\boxed{x}—]\!]$ is witnessed by the fact that, for any possible operational semantics $[\![—\!\!\to\!\!\bigcirc\!—]\!]$ in St(AddRel) that one may want to give to $—\!\!\to\!\!\bigcirc\!—$, one obtains diagrams in St(Rc) and Rc_s.

6 CONCLUSION

The resource calculus is a language that combines a graphical rendition of systems found in combinatorial approaches such as Petri nets, and compositionality, typical of process algebraic approaches. Unlike established process algebras, where the syntax is chosen to capture a particular set of computational primitives, the syntax of the resource calculus is canonical and justified by the Diophantine linear algebraic concepts that it characterises.

The semantics is borrowed from signal flow graphs, a formalism introduced [Shannon 1942] to design linear dynamical systems. The difference is in the interpretation given to the values flowing on the wires: real numbers for signal flow graphs, natural numbers for the resource calculus. The fact that this switch introduces well-known concurrent behaviour is our starting inspiration.

	St(Rc)	Rc$_s$
$[\![-\boxed{x}-]\!]$	$(1, \asymp)$	$-\boxed{x}-$
$[\![\rightarrow\!\!\bigcirc\!\!-]\!]$		
$[\![\rightarrow\!\!\bigcirc\!\!\!b\!\!-]\!]$		

Fig. 5. For each of the semantics on the left, the central and the right columns report the corresponding arrows in St(Rc) and Rc$_s$.

Our key contribution is the presentation (by means of generators and equations) of the prop of additive relations (Theorem 17). Our axiomatisation differs substantially from graphical linear algebra [Bonchi et al. 2017d], but nevertheless features standard algebraic structures (monoids, comonoids, bimonoids and Frobenius algebra) that also appear in different fields, notably quantum foundations [Coecke and Kissinger 2017]. From a technical point of view, the presentation of additive relations is challenging, since these cannot be retrieved from the categorical *Rel*-construction: indeed, unlike matrices of integers, which are prominent in the axiomatisation of linear relations, the category of matrices of natural numbers has neither pullbacks nor a factorisation structure.

The addition of states to the language is obtained by following an abstract construction that was known in the literature [Katis et al. 1997]. Our key observation is the isomorphism (Theorem 30) that, on the one hand allows to provide an axiomatisation for stateful processes and, on the other *fixes* the semantics of the canonical stateful extension as the *synchronous buffer*, with precisely the behaviour witnessed by the registers of signal flow graphs. This is a solid argument in favour of our claims about the canonicity of our language.

We tested the expressivity of the resource calculus by giving semantic-preserving encodings of Petri nets. The semantics provided to the state of Petri nets is that of *asynchronous buffers*. We show that these can be encoded by means of synchronous buffers, but an encoding in the other direction is not possible: since *synchronous* buffers are the canonical notion arising from the aforementioned abstract construction, they are strictly more expressive.

There are many important non-additive concurrent phenomena not captured by the resource calculus (or its stateful extension). One obvious example is that of C/E nets, a type of Petri net for which each place may contain at most one token. Similarly, concurrent phenomena involving mutual exclusion or more general inhibitory patterns of behaviour are not additive but require the ability to express *bounded* resources: the underlying algebra here is not linear, but *affine*. Interestingly, affine concurrent systems can be captured with a simple extension of our approach that will be the subject of a sequel to the present paper.

We conclude by remarking once more that, for the stateful calculus, the semantics equivalence that we characterise is too intensional to be considered an observational equivalence, such as trace equivalence or bisimilarity. Devising and axiomatising a proper observational equivalence is left as future work. Nonetheless, we expect that the proximity to signal flow graphs used in control theory and the solid algebraic ground of the resource calculus will provide useful insights.

REFERENCES

Samson Abramsky. 2014. What are the fundamental structures of concurrency? We still don't know! *CoRR* abs/1401.4973 (2014).

John Baez and Jason Erbele. 2015. Categories In Control. *Theory and Applications of Categories* 30 (2015), 836–881.

Paolo Baldan, Andrea Corradini, Hartmut Ehrig, and Reiko Heckel. 2005. Compositional semantics for open Petri nets based on deterministic processes. *Mathematical Structures in Computer Science* 15, 1 (2005), 1–35.

Filippo Bonchi, Joshua Holland, Dusko Pavlovic, and Paweł Sobociński. 2017a. Refinement for Signal Flow Graphs. In *28th International Conference on Concurrency Theory, CONCUR 2017, September 5-8, 2017, Berlin, Germany*. 24:1–24:16. https://doi.org/10.4230/LIPIcs.CONCUR.2017.24

Filippo Bonchi, Dusko Pavlovic, and Pawel Sobocinski. 2017b. Functorial Semantics for Relational Theories. *arXiv preprint arXiv:1711.08699* (2017).

Filippo Bonchi, Pawel Sobocinski, and Fabio Zanasi. 2014. A Categorical Semantics of Signal Flow Graphs. In *CONCUR 2014*. 435–450.

Filippo Bonchi, Pawel Sobocinski, and Fabio Zanasi. 2015. Full Abstraction for Signal Flow Graphs. In *POPL 2015*. 515–526.

Filippo Bonchi, Pawel Sobocinski, and Fabio Zanasi. 2017c. The Calculus of Signal Flow Diagrams I: Linear relations on streams. *Inf. Comput.* 252 (2017), 2–29. https://doi.org/10.1016/j.ic.2016.03.002

Filippo Bonchi, Pawel Sobociński, and Fabio Zanasi. 2017d. Interacting Hopf Algebras. *Journal of Pure and Applied Algebra* 221, 1 (2017), 144–184.

Roberto Bruni and Fabio Gadducci. 2001. Some Algebraic Laws for Spans (and Their Connections With Multi-Relations). In *RelMiS 2001*. Elsevier.

Roberto Bruni, Hernán C. Melgratti, and Ugo Montanari. 2011. A Connector Algebra for P/T Nets Interactions. In *CONCUR 2011*. 312–326.

Roberto Bruni, Hernán C. Melgratti, Ugo Montanari, and Paweł Sobociński. 2013. Connector Algebras for C/E and P/T Nets' Interactions. *Log Meth Comput Sci* 9, 16 (2013).

Aurelio Carboni and R. F. C. Walters. 1987. Cartesian Bicategories I. *J Pure Appl Algebra* 49 (1987), 11–32.

Bob Coecke and Ross Duncan. 2011. Interacting quantum observables: categorical algebra and diagrammatics. *New Journal of Physics* 13, 4 (2011), 043016.

Bob Coecke and Aleks Kissinger. 2017. *Picturing Quantum Processes - A first course in Quantum Theory and Diagrammatic Reasoning.* Cambridge University Press.

Bob Coecke, Dusko Pavlovic, and Jamie Vicary. 2012. A new description of orthogonal bases. *Math. Struct. Comp. Sci.* 23, 3 (2012), 557–567.

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

Leonard Eugene Dickson. 1913. Finiteness of the Odd Perfect and Primitive Abundant Numbers with n Distinct Prime Factors. *American Journal of Mathematics* 35, 4 (1913), 413–422.

Brendan Fong, Paolo Rapisarda, and Paweł Sobociński. 2016. A categorical approach to open and interconnected dynamical systems. In *LICS 2016*.

Naohiko Hoshino, Koko Muroya, and Ichiro Hasuo. 2014. Memoryful geometry of interaction: from coalgebraic components to algebraic effects. In *Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. ACM, 52.

P Katis, N Sabadini, and RFC Walters. 1997. Bicategories of processes. *Journal of Pure and Applied Algebra* 115, 2 (1997), 141–178.

Joachim Kock. 2003. *Frobenius algebras and 2D topological quantum field theories.* CUP.

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

Samuel J Mason. 1953. *Feedback Theory: I. Some Properties of Signal Flow Graphs.* MIT Research Laboratory of Electronics.

Antoni Mazurkiewicz. 1987. Compositional semantics of pure place/transition systems. In *European Workshop on Applications and Theory in Petri Nets*. Springer, 307–330.

Mogens Nielsen, Lutz Priese, and Vladimiro Sassone. 1995. Characterizing behavioural congruences for Petri nets. In *International Conference on Concurrency Theory*. Springer, 175–189.

Mogens Nielsen, Vladimiro Sassone, and Glynn Winskel. 1993. Relationships Between Models of Concurrency. In *A Decade of Concurrency, Reflections and Perspectives, REX School/Symposium, Noordwijkerhout, The Netherlands, June 1-4, 1993, Proceedings*. 425–476. https://doi.org/10.1007/3-540-58043-3_25

Wolfgang Reisig. 2009. Simple composition of nets. In *International Conference on Applications and Theory of Petri Nets*. Springer, 23–42.

Peter Selinger. 2011. A survey of graphical languages for monoidal categories. *Springer Lecture Notes in Physics* 13, 813 (2011), 289–355.

[1177] Claude E. Shannon. 1942. *The Theory and Design of Linear Differential Equation Machines*. Technical Report. National Defence Research Council.

[1179] Rob J van Glabbeek. 1990. The linear time-branching time spectrum. In *International Conference on Concurrency Theory*. Springer, 278–297.

[1180] Jan C Willems. 2007. The behavioural approach to open and interconnected systems. *IEEE Contr. Syst. Mag.* 27 (2007), 46–99.

[1182] Fabio Zanasi. 2015. *Interacting Hopf Algebras: the theory of linear systems*. Ph.D. Dissertation. Ecole Normale Supérieure de Lyon.

[1184] Fabio Zanasi. 2016. The Algebra of Partial Equivalence Relations, In Mathematical Foundations of Program Semantics (MFPS). *Electr. Notes Theor. Comput. Sci.* 325 (2016), 313–333.

A APPENDIX

A.1 Additive Relations

PROOF OF LEMMA 26. By induction on n. The base case is

(30)

Assume that it holds for some positive integer n.

(31)

(32)

(33)

□

PROOF OF LEMMA 27. By induction on n. The base case is immediate so assume that the lemma holds for some positive integer $n - 1$. Then,

(34)

(35)

□

A.2 Stateful Processes

PROOF OF LEMMA 29. By induction on morphisms of T + X. For the base case, if a morphism d of T + X is in either T or X, the statement holds since

and every morphism of T is trivially in trace canonical form with the trace taken over the 0 object. There are two inductive cases to consider:

- d is given by the sequential composition of two morphisms $a\colon k \to l$ and $b\colon l \to p$ which are, by the induction hypothesis, in trace canonical form:

- d is given as the monoidal product of two morphisms $c_1\colon k_1 \to l_1$ and $c_2\colon k_2 \to l_2$, both in trace canonical form:

□

A.3 Petri nets

The following lemma ensures that the assignment $\mathcal{P} \mapsto d_{\mathcal{P}}$ is well defined, namely that it is independent from the chosen ordering on places and transitions.

LEMMA 43. Let $\sigma\colon |P| \to |P|$ and $\tau\colon |T| \to |T|$ be two permutations. Then

PROOF.

$$\tag{36}$$

$$= \quad \begin{array}{c} \text{(diagram: } s \text{—} \sigma \text{, } |P| \text{; } |T| \text{—} t \text{—} \sigma \text{—} \circ \text{)} \end{array} \tag{37}$$

$$= \quad \begin{array}{c} \text{(diagram: } s \text{, } |P| \text{; } |T| \text{—} t \text{—} \sigma \text{—} \circ \text{—} \sigma \text{)} \end{array} \tag{38}$$

$$= \quad \begin{array}{c} \text{(diagram: } s \text{, } |P| \text{; } |T| \text{—} t \text{—} \sigma \text{—} \sigma \text{—} \circ \text{)} \end{array} \tag{39}$$

$$= \quad \begin{array}{c} \text{(diagram: } s \text{, } |P| \text{; } |T| \text{—} t \text{—} \circ \text{)} \end{array} \tag{40}$$

$$\tag{41}$$

$\square$

LEMMA 44. *Let $c \in \mathrm{Rc}[p, p]$ and d be the following diagram in* Petri.

Then $[\![d]\!] = \{(\mathbf{a}, \mathbf{b}) \mid \exists(\mathbf{a}', \mathbf{b}') \in [\![c]\!] \text{ such that } \mathbf{a}' \le \mathbf{a} \text{ and } \mathbf{b} = \mathbf{a} - \mathbf{a}' + \mathbf{b}'\}.$

PROOF. Straightforward from the definition of the operational semantics of $\text{—}\!\!\!\!\multimap\!\!\bigcirc\text{—}$. $\square$

PROOF OF PROPOSITION 37. Let $\mathcal{P} = (P, T, {}^\circ\!-, -^\circ)$ be a Petri nets. Since the operational equivalence is stated modulo $\sim$, it is safe to fix an ordering on P and T. We take $\mathbf{a}, \mathbf{b} \in \mathbb{N}^{|P|}$. We denote by $I(s)$ and $I(t)$ the matrices corresponding to the diagrams $s, t \in \mathrm{Rc}[|T|, |P|]$ from the definition of $d_\mathcal{P}$. Thus we have $(\mathbf{a}, \mathbf{b}) \in [\![\mathcal{P}]\!]$ iff there exists $\mathbf{f} \in \mathbb{N}^{|T|}$ such that $I(s)(\mathbf{f}) \le \mathbf{a}$ and $\mathbf{b} = \mathbf{a} - I(s)(\mathbf{f}) + I(t)(\mathbf{f})$. Since $[\![s^{-1} \, ; t]\!] = \{(\mathbf{a}', \mathbf{b}') \mid \mathbf{a}' = I(s)(\mathbf{f}) \text{ and } \mathbf{b}' = I(t)(\mathbf{f})\}$, we have that $(\mathbf{a}, \mathbf{b}) \in [\![\mathcal{P}]\!]$ iff there exists $(\mathbf{a}', \mathbf{b}') \in [\![s^{-1} \, ; t]\!]$ such that $\mathbf{a}' \le \mathbf{a}$ and $\mathbf{b} = \mathbf{a} - \mathbf{a}' + \mathbf{b}'$. By Lemma 44, we conclude that $(\mathbf{a}, \mathbf{b}) \in [\![\mathcal{P}]\!]$ iff $(\mathbf{a}, \mathbf{b}) \in [\![c_\mathcal{P}]\!]$. $\square$

PROOF OF PROPOSITION 38. First, observe that by construction $(\mathbf{a}', \mathbf{b}') \in [\![c]\!]$ iff there exists $\mathbf{f} \in \mathbb{N}^t$ such that $A_1(\mathbf{f}) = \mathbf{a}'$ and $A_2(\mathbf{f}) = \mathbf{b}'$. Therefore, by Lemma 44, $(\mathbf{a}, \mathbf{b}) \in [\![d]\!]$ iff there exists $\mathbf{f} \in \mathbb{N}^t$ such that $A_1(\mathbf{f}) \le \mathbf{a}$ and $\mathbf{b} = \mathbf{a} - A_1(\mathbf{f}) + A_2(\mathbf{f})$. That is $(\mathbf{a}, \mathbf{b}) \in [\![d]\!]$ iff $(\mathbf{a}, \mathbf{b}) \in [\![\mathcal{P}_d]\!]$. $\square$