Inputs, Outputs, and Composition in the Logic of Information Flows

Heba Aamer[1] , **Bart Bogaerts**[2] , **Dimitri Surinx**[1] , **Eugenia Ternovska**[3] , **Jan Van den Bussche**[1]

[1]Universiteit Hasselt, Belgium
[2]Vrije Universiteit Brussel, Belgium
[3]Simon Fraser University, Canada
{heba.mohamed,dimitri.surinx,jan.vandenbussche}@uhasselt.be, bart.bogaerts@vub.be, ter@sfu.ca

Abstract

The logic of information flows (LIF) is a general framework in which tasks of a procedural nature can be modeled in a declarative, logic-based fashion. The first contribution of this paper is to propose semantic and syntactic definitions of inputs and outputs of LIF expressions. We study how the two relate and show that our syntactic definition is optimal in a sense that is made precise. The second contribution is a systematic study of the expressive power of sequential composition in LIF. Our results on composition tie in the results on inputs and outputs, and relate LIF to first-order logic (FO) and bounded-variable LIF to bounded-variable FO.

1 Introduction

The Logic of Information Flows (LIF) (Ternovska 2017; Ternovska 2019) is a knowledge representation framework designed to model and understand how information propagates in complex systems, and to find ways to navigate it efficiently. The basic idea is that modules, that can be given procedurally or declaratively, are the atoms of a logic whose syntax resembles first-order logic, but whose semantics produces new modules. In LIF, atomic modules are modeled as relations with designated input and output arguments. Computation is modeled as propagation of information from inputs to outputs, similarly to propagation of tokens in Petri nets. The specification of a complex system then amounts to *connecting* atomic modules together. For this purpose, LIF uses the classical logic connectives, i.e., the boolean operators, equality, and existential quantification. The goal is to start from constructs that are well understood, and to address the fundamental question of *what logical means are necessary and sufficient to model computations declaratively*. The eventual goal, which goes beyond the topic of this paper, is to come up with restrictions or extensions of LIF that make the computations efficient.

In its most general form, LIF is a rich family of logics with recursion and higher-order variables. Atomic modules are given by formulae in various logics, and may be viewed as solving the task of Model Expansion (Mitchell and Ternovska 2005): the input structure is expanded to satisfy the specification of a module thus producing an output. The semantics is given in terms of pairs of structures. We can, for example, give a graph (a relational structure) on the input of a module that returns a Hamiltonian cycle on the output, and compose it sequentially with a module that checks whether the produced cycle is of even length. One can vary both the expressiveness of logics for specifying atomic modules and the operations for combining modules, to achieve desirable complexity of the computation for the tasks of interest.

Many issues surrounding LIF, however, are already interesting in a first-order setting (see, e.g., (Aamer et al. 2020)); and in fact such a setting is more generic than the higher-order setting, which can be obtained by considering *relations* as atomary data values. Thus, in this paper, we give a self-contained, first-order presentation of LIF, with a semantics defined in terms of pairs of valuations of first-order variables; the first valuation represents a situation right before applying the module, while the second represents a possible situation immediately afterwards. The results in this paper are then also applicable to the case of higher-order variables.

Our contributions can be summarized as follows.

(i) While the input and output arguments of relation atoms are specified by the vocabulary, it is not clear how to designate the input and output variables of a complex LIF formula. Actually, coming up with formal definitions of what it means for a variable to be an input or output is a technically and philosophically interesting undertaking. We propose semantic definitions, based on natural intuitions, which are, of course, open to further debate. The semantic notions of input and output turn out to be undecidable. This is not surprising, since LIF formulas subsume classical first-order logic formulas, for which most inference tasks in general are undecidable.

(ii) We proceed to give an approximate, syntactic definition of the input and output variables of a formula, which is effectively computable. Indeed, our syntactic definition is *compositional,* meaning that the set of syntactic input (or output) variables of a formula depends only on the top-level operator of the formula, and the syntactic inputs and outputs of the operands. We prove our syntactic input–output notion to be *sound:* every semantic input or output is also a syntactic input or output, and the syntactic inputs and outputs are connected by a natural property that we call *input–output determinacy*. Moreover, we prove an optimality result: our definition provides the most precise approximation to semantic input and outputs among all compositional and sound definitions.

(iii) We investigate the expressive power of sequential composition in the context of LIF. The sequential composition of two modules is fundamental to building complex systems. Hence, we are motivated to understand in detail whether or not this operation is expressible in terms of the basic LIF connectives. This question turns out to be approachable through the notion of inputs and outputs. Indeed, there turns out to be an elegant expression for the composition of *io-disjoint* modules. Here, io-disjointness means that inputs and outputs do not overlap. For example, a module that computes a function of x and returns the result in y is io-disjoint; a module that stores the result back in x, thus overwriting the original input, is not.

(iv) We then use the result on io-disjoint expressions to show that composition is indeed an expressible operator in the classical setting of LIF, where there is an infinite supply of fresh variables. (In contrast, the expression for io-disjoint modules does not need extra variables.)

(v) Finally, we complement the above findings with a result on LIF in a bounded-variable setting: in this setting, composition is necessarily a primitive operator.

Many of our notions and results are stated generally in terms of transition systems (binary relations) on first-order valuations. Consequently, we believe our work is also of value to settings other than LIF inasmuch as they involve dynamic semantics. Several such settings, where input–output specifications are important, are discussed in Related Work.

The rest of this paper is organized as follows. In Section 2, we formally introduce the Logic of Information Flows from a first-order perspective. Section 3 presents our investigation concerning the notion of inputs and outputs of complex expressions. Section 4 then presents our investigation on the expressibility of sequential composition. Section 5 discusses related work. We conclude in Section 6.

2 Preliminaries

A *(module) vocabulary* $\mathcal{S}$ is a triple $(\mathit{Names}, \mathrm{ar}, \mathrm{iar})$ where:

- Names is a nonempty set, the elements of which are called *module names*;
- ar assigns an arity to each module name in Names;
- iar assigns an input arity to each module name M in Names, where $\mathrm{iar}(M) \leq \mathrm{ar}(M)$.

We fix a countably infinite universe $\mathbf{dom}$ of data elements. An *interpretation* D of $\mathcal{S}$ assigns to each module name M in Names an $\mathrm{ar}(M)$-ary relation $D(M)$ over $\mathbf{dom}$.

Furthermore, we fix a universe of *variables* $\mathbb{V}$. This set may be finite or infinite; the size of $\mathbb{V}$ will influence the expressive power of our logic. A *valuation* is a function from $\mathbb{V}$ to $\mathbf{dom}$. The set of all valuations is denoted by $\mathcal{V}$. We say that ν_1 and ν_2 *agree on* $Y \subseteq \mathbb{V}$ if $\nu_1(y) = \nu_2(y)$ for all $y \in Y$ and that they *agree outside of* Y if they agree on $\mathbb{V} - Y$ A *partial valuation* on $Y \subseteq \mathbb{V}$ is a function from Y to $\mathbb{V}$; we will also call such a function a Y-valuation. If ν is a valuation, we use $\nu|_Y$ to denote its restriction to Y. Let ν be a valuation and let ν_1 be a partial valuation on $Y \subseteq \mathbb{V}$. Then the substitution of ν_1 into ν, denoted by $\nu[\nu_1]$ is defined as $\nu_1 \cup (\nu|_{\mathbb{V}-Y})$. We assume familiarity with the syntax and semantics of first-order logic (FO, relational calculus) over $\mathcal{S}$ (Enderton 1972) and use $:=$ to mean "is by definition".

BRVs The semantics of LIF will be defined in terms of binary relations on $\mathcal{V}$ (abbreviated BRV: Binary Relations on Valuations). Before formally introducing LIF, we define operations on BRVs corresponding to the classical logical connectives, adapted to a dynamic semantics. For boolean connectives, we simply use the standard set operations. For equality, we introduce selection operators. For existential quantification, we introduce cylindrification operators.

Let A and B be BRVs, let Z be a finite set of variables, and let x and y be variables.

- **Set operations:** $A \cup B$, $A \cap B$, and $A - B$ are well known.
- **Composition**
  $A;B := \{(\nu_1, \nu_2) \mid \exists \nu_3 : (\nu_1, \nu_3) \in A \text{ and } (\nu_3, \nu_2) \in B\}$
- **Left and Right Cylindrifications**
$$\mathrm{cyl}^l_Z(A) := \{(\nu_1, \nu_2) \mid \exists \nu'_1 : (\nu'_1, \nu_2) \in A \text{ and } \nu'_1 \text{ and } \nu_1 \text{ agree outside of } Z\}$$
$$\mathrm{cyl}^r_Z(A) := \{(\nu_1, \nu_2) \mid \exists \nu'_2 : (\nu_1, \nu'_2) \in A \text{ and } \nu'_2 \text{ and } \nu_2 \text{ agree outside of } Z\}$$
- **Left and Right Selection**
$$\sigma^l_{x=y}(A) := \{(\nu_1, \nu_2) \in A \mid \nu_1(x) = \nu_1(y)\}$$
$$\sigma^r_{x=y}(A) := \{(\nu_1, \nu_2) \in A \mid \nu_2(x) = \nu_2(y)\}$$
- **Left-To-Right Selection**
$$\sigma^{lr}_{x=y}(A) := \{(\nu_1, \nu_2) \in A \mid \nu_1(x) = \nu_2(y)\}$$

If $\bar{x}$ and $\bar{y}$ are tuples of variables of length n, we write $\sigma^{lr}_{\bar{x}=\bar{y}}(A)$ for $\sigma^{lr}_{x_1=y_1}\sigma^{lr}_{x_2=y_2}\dots\sigma^{lr}_{x_n=y_n}(A)$ and if z is a variable we write cyl^l_z for $\mathrm{cyl}^l_{\{z\}}$. Intuitively, a BRV is a dynamic system that manipulates the interpretation of variables. A pair (ν_1, ν_2) in a BRV represents that a transition from ν_1 to ν_2 is possible, i.e., that when given ν_1 as input, the values of the variables can be updated to ν_2. The different operations defined above correspond to manipulations/combinations of such dynamic systems. Union, for instance, represents a non-deterministic choice, while composition corresponds to composing two such systems. Left cylindrification corresponds, in the dynamic view to performing search before following the underlying BRV. Indeed, when given an input ν_1, alternative values for the cylindrified variables are searched for which transitions are possible. The selection operations correspond to performing checks, on the input, the output, or a combination of both in addition to performing what the underlying BRV does.

Some of the above operators are redundant, in the sense that they can be expressed in terms of others, for instance, $A \cap B = A - (A - B)$. We also have:

Lemma 1. *For any BRVs A and B, and any variables x and y, the following hold:*
$$\sigma^r_{x=y}(A) = A \cap \mathrm{cyl}^l_x \sigma^{lr}_{x=y} \sigma^{lr}_{x=x} \mathrm{cyl}^l_x(A)$$
$$\sigma^l_{x=y}(A) = A \cap \mathrm{cyl}^r_x \sigma^{lr}_{y=x} \sigma^{lr}_{x=x} \mathrm{cyl}^r_x(A)$$

The expression for $\sigma^r_{x=y}$ can be explained as follows. First, we copy x from right to left by applying cyl^l_x followed by $\sigma^{lr}_{x=x}$. Selection $\sigma^r_{x=y}$ can now be simulated by $\sigma^{lr}_{x=y}$. The original x value on the left is restored by a final application of cyl^l_x and intersecting with the original A.

The Logic of Information Flows The language of LIF expressions α over a vocabulary $\mathcal{S}$ is defined by the following grammar:

$$\alpha ::= id \mid M(\overline{z}) \mid \alpha \cup \alpha \mid \alpha \cap \alpha \mid \alpha - \alpha \mid \alpha \,;\, \alpha$$
$$\mid \mathrm{cyl}^{l}_{Z}(\alpha) \mid \mathrm{cyl}^{r}_{Z}(\alpha) \mid \sigma^{lr}_{x=y}(\alpha) \mid \sigma^{l}_{x=y}(\alpha) \mid \sigma^{r}_{x=y}(\alpha)$$

Here, M is any module name in $\mathcal{S}$, Z is a finite set of variables; $\overline{z}$ is a tuple of variables; and x, y are variables. For *atomic module expressions*, i.e., expressions of the form $M(\overline{z})$, the length of $\overline{z}$ must equal $\mathrm{ar}(M)$. In practice, we will often write $M(\overline{x};\overline{y})$ for atomic module expressions, where $\overline{x}$ is a tuple of variables of length $\mathrm{iar}(M)$ and $\overline{y}$ is a tuple of variables of length $\mathrm{ar}(M) - \mathrm{iar}(M)$.

We will define the semantics of a LIF expression α, in the context of a given interpretation D, as a BRV which will be denoted by $[\![\alpha]\!]_D$. Thus, adapting Gurevich's terminology (Gurevich 1983; Gurevich 1988), every LIF expression α denotes a *global BRV* $[\![\alpha]\!]$: a function that maps interpretations D of $\mathcal{S}$ to the BRV $\alpha(D) := [\![\alpha]\!]_D$.

For atomic module expressions, we define

$$[\![M(\overline{x};\overline{y})]\!]_D := \{(\nu_1,\nu_2) \in \mathcal{V}\times\mathcal{V} \mid \nu_1(\overline{x}) \cdot \nu_2(\overline{y}) \in D(M)$$
$$\text{and } \nu_1 \text{ and } \nu_2 \text{ agree outside of } \overline{y}\}.$$

Here, $\nu_1(\overline{x}) \cdot \nu_2(\overline{y})$ denotes the *concatenation* of tuples. Intuitively, the semantics of an expression $M(\overline{x};\overline{y})$ represents a transition from ν_1 to ν_2: the inputs of the module are "read" in ν_1 and the outputs are updated in ν_2. The value of every variable that is not an output is preserved; this important semantic principle is a realization of the commonsense law of inertia (McCarthy and Hayes 1969; Lifschitz 1987). We further define

$$[\![id]\!]_D := \{(\nu,\nu) \mid \nu \in \mathcal{V}\}.$$

The semantics of other operators is obtained directly by applying the corresponding operation on BRVs, e.g.,

$$[\![\alpha - \beta]\!]_D := [\![\alpha]\!]_D - [\![\beta]\!]_D$$
$$[\![\sigma^{lr}_{x=y}(\alpha)]\!]_D := \sigma^{lr}_{x=y}([\![\alpha]\!]_D)$$

We say that α and β are *equivalent* if $[\![\alpha]\!]_D = [\![\beta]\!]_D$ for each interpretation D, i.e., if they denote the same global BRV.

3 Inputs and Outputs

We are now ready to study inputs and outputs of LIF expressions, and, more generally, of global BRVs. We first investigate what inputs and outputs mean on the semantic level before introducing a syntactic definition for LIF expressions.

3.1 Semantic Inputs and Outputs

Intuitively, an output is a variable whose value can be changed by the expression, i.e., a variable that is not subject to inertia.

Definition 2. *A variable x is a* semantic output *for a global BRV Q if there exists an interpretation D and $(\nu_1,\nu_2) \in Q(D)$ such that $\nu_1(x) \neq \nu_2(x)$. We use $O^{\mathrm{sem}}(Q)$ denote the set of semantic output variables of Q. If α is a LIF expression, we call a variable a* semantic output *of α if it is a semantic output of $[\![\alpha]\!]$. We also write $O^{\mathrm{sem}}(\alpha)$ for the semantic outputs of α.*

Defining semantic inputs is a bit more subtle. Intuitively, a variable is an input for a BRV if its value on the left-hand side matters for determining the right-hand side (i.e., that if the value of the input would have been different, so would have been the right-hand side; which is in fact a very coarse counterfactual definition of actual causality (Lewis 1973)). However, a naive formalization of this intuition would result in a situation in which all inertial variables are inputs since their value on the right-hand side always equals to the one on the left-hand side. A slight refinement of our intuition is that the inputs are those variables whose value matters for determining the possible values of the outputs. This is formalized in the following definitions.

Definition 3. *Let Q be a global BRV and X, Y be sets of variables. We say that X determines Q on Y if for every interpretation D, every $(\nu_1,\nu_2) \in Q(D)$ and every ν_1' such that $\nu_1' = \nu_1$ on X, there exists a ν_2' such that $\nu_2' = \nu_2$ on Y and $(\nu_1',\nu_2') \in Q(D)$.*

Definition 4. *A variable x is a* semantic input *for a global BRV Q if $\mathbb{V} - \{x\}$ does not determine Q on $O^{\mathrm{sem}}(Q)$. The set of input variables of Q is denoted by $I^{\mathrm{sem}}(Q)$. A variable is a* semantic input *of a LIF expression α if it is a semantic input of $[\![\alpha]\!]$; the semantic inputs of α are denoted $I^{\mathrm{sem}}(\alpha)$.*

Rephrased, x is a semantic input if there is an interpretation D and valuations ν_1, ν_2, and ν_1' such that $(\nu_1,\nu_2) \in Q(D)$ and $\nu_1' = \nu_1$ outside of x, while there is no ν_2' such that $\nu_2' = \nu_2$ on $O^{\mathrm{sem}}(Q)$ and $(\nu_1',\nu_2') \in Q(D)$. The following proposition shows that the semantic inputs of Q are indeed exactly the variables that determine Q.

Proposition 5. *A set of variables X determines a global BRV Q on $O^{\mathrm{sem}}(Q)$ if and only if $I^{\mathrm{sem}}(Q) \subseteq X$.*

Our next property of semantic inputs and outputs expresses that they are exactly what one expects for atomic modules.

Proposition 6. *If α is an atomic LIF expressions $M(\bar{x};\bar{y})$, then $I^{\mathrm{sem}}(\alpha) = \bar{x}$ and $O^{\mathrm{sem}}(\alpha) = \bar{y}$.*

Example 7. *A variable can be both input and output of a given expression. A very simple example is an atomic module $P_1(x;x)$. To illustrate where this can be useful, assume $\mathbf{dom} = \mathbb{Z}$ and consider an interpretation D such that $D(P_1) = \{(n, n+1) \mid n \in \mathbb{Z}\}$. In that case, the expression $P_1(x;x)$ represents a dynamic system in which the value of x is incremented by 1; x is an output of the system since its value is changed; it is an input since its original value matters for determining its value in the output.*

Intuitively, the inputs and outputs are the only variables that matter for a given global BRV, similar to how in classical logic the free variables are the only ones that matter. All other variables can take arbitrary values, *but*, their values are preserved by inertia, i.e., remain unchanged by the dynamic system. We now formalize this intuition.

Definition 8. *Let Q be a global BRV and X a set of variables. We say that Q is* inertially cylindrified *on X if for every interpretation D and every $(\nu_1,\nu_2) \in Q(D)$:*
(i) *ν_1 and ν_2 are equal on X and*
(ii) *for every X-valuation ν' also $(\nu_1[\nu'],\nu_2[\nu']) \in Q(D)$.*

Proposition 9. *Every global BRV Q is inertially cylindrified outside the semantic inputs and outputs of Q.*

We will now show that the problem of deciding whether a given variable is a semantic input or output of a LIF expression is undecidable. Thereto we begin by noting that first-order logic (FO) is naturally embedded in LIF in the following manner. When evaluating FO formulas on interpretations, we agree that the domain of quantification is always $\mathbf{dom}$.

Lemma 10. *Let $\mathcal{S}$ be a vocabulary with $\mathrm{iar}(R) = 0$ for every $R \in \mathcal{S}$. Then, for every FO formula φ over $\mathcal{S}$, there exists a LIF expression α_φ such that for every interpretation D the following holds:*

$$[\![\alpha_\varphi]\!]_D = \{(\nu, \nu) \mid D, \nu \models \varphi\}.$$

Proof sketch. The proof is by structural induction on φ.

- If φ is $x = y$, take $\alpha_\varphi = \sigma^r_{x=y}(id)$.
- If φ is $R(\bar{x})$ for some $R \in \mathcal{S}$, take $\alpha_\varphi = id \cap R(;\bar{x})$.
- If φ is $\varphi_1 \vee \varphi_2$, take $\alpha_\varphi = \alpha_{\varphi_1} \cup \alpha_{\varphi_2}$.
- If φ is $\neg\varphi_1$, take $\alpha_\varphi = id - \alpha_{\varphi_1}$.
- If φ is $\exists x\, \varphi_1$, take $\alpha_\varphi = \sigma^{lr}_{x=x}(\mathrm{cyl}^l_x(\mathrm{cyl}^r_x(\alpha_{\varphi_1})))$. □

It is well known that satisfiability of FO formulas over a fixed countably infinite domain is undecidable. This leads to the following undecidability results.

Problem: **Semantic Output Membership** Given: a variable x and a LIF expression α. Decide: $x \in O^{\mathrm{sem}}(\alpha)$?

Proposition 11. *The semantic output membership problem is undecidable.*

Proof sketch. By reduction from the satisfiability of FO formulas. Let φ be an FO formula and let α_φ be the LIF expression obtained from Lemma 10. Let $\alpha := \mathrm{cyl}^l_x(\alpha_\varphi)$. It can be verified that $x \in O^{\mathrm{sem}}(\alpha)$ iff φ is satisfiable. □

Problem: **Semantic Input Membership** Given: a variable x and a LIF expression α. Decide: $x \in I^{\mathrm{sem}}(\alpha)$?

Proposition 12. *The semantic input membership problem is undecidable.*

Proof sketch. Let φ be an FO formula and let α_φ be the LIF expression obtained from Lemma 10. Let $\alpha := \sigma^l_{x=z}(\alpha_\varphi)$, where z is a variable that is not a free variable of φ and different from x. It can be verified that $x \in I^{\mathrm{sem}}(\alpha)$ iff φ is satisfiable. □

3.2 Syntactic Inputs and Outputs

Since both the membership problems for semantic inputs and outputs are undecidable, to use inputs and outputs in practice, we will need decidable approximations of these concepts. Before giving our syntactic definition, we define some properties of candidate definitions.

Definition 13. *Let I and O be functions from LIF expressions to sets of variables. We say that (I, O) is a* sound input–output definition *if the following hold:*

- *If $\alpha = M(\overline{x}; \overline{y})$, then $I(\alpha) = \overline{x}$ and $O(\alpha) = \overline{y}$,*
- *$I(\alpha) \supseteq I^{\mathrm{sem}}(\alpha)$,*
- *$O(\alpha) \supseteq O^{\mathrm{sem}}(\alpha)$, and*
- *$I(\alpha)$ determines $[\![\alpha]\!]$ on $O(\alpha)$.*

The first condition states that on atomic expressions (of which we know the inputs), I and O are defined correctly. The next two conditions state that I and O approximate the semantic notions correctly. We only allow for overapproximations; that is, false positives are allowed while false negatives are not. The reason for this is that falsely marking a variable as non-output while it is actually an output would mean incorrectly assuming the variable cannot change value. A similar argument can be made for inputs. The last condition establishes the relation between I and O, and is called *input–output determinacy*. It states that the inputs need to be large enough to determine the outputs, as such generalizing the defining condition of semantic inputs. Essentially, this means that whenever we overapproximate our outputs, we should also overapproximate our inputs to compensate for this; that correspondence is formalized in Lemma 15.

Besides requiring that our definitions be sound, we will focus on definitions that are *compositional*, in the sense that definitions of inputs and outputs of compound expressions can be given in terms of their direct subexpressions essentially treating subexpressions as black boxes. This means that the definition nicely follows the inductive definition of the syntax. Formally:

Definition 14. *Suppose I and O are functions from LIF expression to sets of variables. We say that (I, O) is* compositional *if for all LIF expressions $\alpha_1, \alpha_2, \beta_1$, and β_2 with $I(\alpha_1) = I(\alpha_2)$, $O(\alpha_1) = O(\alpha_2)$, $I(\beta_1) = I(\beta_2)$, and $O(\beta_1) = O(\beta_2)$ the following hold:*

- *For every unary operator $\square$: $I(\square\alpha_1) = I(\square\alpha_2)$ and $O(\square\alpha_1) = O(\square\alpha_2)$; and*
- *For every binary operator $\boxdot$: $I(\alpha_1 \boxdot \beta_1) = I(\alpha_2 \boxdot \beta_2)$, and $O(\alpha_1 \boxdot \beta_1) = O(\alpha_2 \boxdot \beta_2)$.*

The previous definition essentially states that in order to be compositional, the inputs and outputs of $\alpha_1 \boxdot \beta_1$ and $\square\alpha_1$ should only depend on the inputs and outputs of α_1 and β_1, and not on their inner structure. The following lemma rephrases input–output determinacy in terms of the inputs and outputs: in order to determine the output-value of an inertial variable, we need to know its input-value.

Lemma 15. *Let (I, O) be a sound input–output definition and let α be a LIF expression. If α is satisfiable,[1] then*

$$O(\alpha) - O^{\mathrm{sem}}(\alpha) \subseteq I(\alpha).$$

If (I, O) is compositional, this holds for all α.

Proof Sketch. The proof hinges on the fact that if a variable x is neither in $I^{\mathrm{sem}}(\alpha)$, nor $O^{\mathrm{sem}}(\alpha)$, Proposition 9 applies, and x is inertially cylindrified. However, in such case, all sets of variables that determine $[\![\alpha]\!]$ on $O(\alpha)$ should contain x (this can be seen by taking any D such that $[\![\alpha]\!]_D \neq \emptyset$). For the compositional case, we can always replace subexpressions by atomic expressions with the same inputs and outputs to ensure satisfiability. □

[1] By satisfiable, we here mean that there exists an interpretation D such that $[\![\alpha]\!]_D \neq \emptyset$.

We now provide a sound and compositional input–output definition. While the definition might seem complex, there is a good reason for the different cases. Indeed, as we show below in Theorem 21, our definition is optimal among the sound and compositional definitions. In the definition, the condition $x =_{\text{syn}} y$ simply means that x and y are the same variable and $\triangle$ denotes the symmetric difference of two sets.

Definition 16. *The syntactic inputs and outputs of a LIF expression α, denoted $I^{\text{syn}}(\alpha)$ and $O^{\text{syn}}(\alpha)$ respectively, are defined recursively as given in Table 1.*

While we do not have enough space to discuss the motivation for all the cases of Definition 16 (their motivation will be clarified in Theorem 21), we here discuss one of the most difficult parts, namely the case where $\alpha = \sigma^{lr}_{x=y}(\alpha_1)$. For a given interpretation D,

$$[\![\alpha]\!]_D = \{(\nu_1, \nu_2) \in [\![\alpha_1]\!]_D \mid \nu_1(x) = \nu_2(y)\}.$$

First of all, since $[\![\alpha]\!]_D \subseteq [\![\alpha_1]\!]_D$, it is clear that the outputs of α should be a subset of those of α_1 (if α_1 admits no pairs in its semantics that change the value of a variable, then neither does α). For the special case in which x and y are the same variable, this selection enforces x to be inertial, i.e., it should not be an output of α.

Secondly, all inputs of α_1 remain inputs of α. Since we select those pairs whose y-value on the righ equals the x-value on the left, clearly x must be an input of α (the special case $x =_{\text{syn}} y$ and $y \notin O^{\text{syn}}(\alpha_1)$ only covers cases where α_1 and α are actually equivalent). Whether or not y is an input depends on α_1: if $y \notin O^{\text{syn}}(\alpha_1)$, y is inertial. Since we compare the input-value of x with the output-value of y, essentially this is the same as comparing the input-values of both variables, i.e., the value of y on the input-side matters. On the other hand, if $y \in O^{\text{syn}}(\alpha_1)$, the value of y can be changed by α_1 and thus this selection does not force y to be an input.

Our syntactic definition is clearly compositional (since we only use the inputs and outputs of subexpressions). An important result is that our definition is also sound, i.e., that our syntactic concepts are overapproximations of the semantic concepts.

Theorem 17. *$(I^{\text{syn}}, O^{\text{syn}})$ is a sound and compositional input–output definition.*

Of course, since the semantic notions of inputs and outputs are undecidable and our syntactic notions clearly are decidable, expressions exist in which the semantic and syntactic notions do no coincide. We give some examples.

Example 18. *Consider the LIF expression*

$$\alpha := \sigma^{l}_{x=y}\sigma^{r}_{x=y}R(x;y)$$

In this case, $O^{\text{sem}}(\alpha) = \emptyset$. However, it can be verified that $O^{\text{syn}}(\alpha) = \{x, y\}$.

Example 19. *Consider the LIF expression*

$$\alpha := \sigma^{lr}_{x=x}\text{cyl}^{r}_{x}\text{cyl}^{l}_{x}P(x;).$$

Thus, we first cylindrify x on both sides and afterwards only select those pairs that have inertia. As such, x is inertially cylindrified in α and $x \notin O^{\text{sem}}(\alpha), x \notin I^{\text{sem}}(\alpha)$. However, $I^{\text{syn}}(\alpha) = \{x\}$.

These examples suggest that our definition can be improved. Indeed, one can probably keep coming up with adhoc but more precise approximations of inputs and outputs for various specific patterns of expressions. Such improvements would not be compositional, as they would be based on inspecting the structure of subexpressions. In the following results, we show that $(I^{\text{syn}}, O^{\text{syn}})$ is actually the most precise sound and compositional input–output definition.

Theorem 20 (Precision Theorem)**.** *Let α be a LIF expression that is either atomic, or a unary operator applied to an atomic module expression, or a binary operator applied to two atomic module expressions involving different module names. Then*

$$O^{\text{sem}}(\alpha) = O^{\text{syn}}(\alpha) \text{ and } I^{\text{sem}}(\alpha) = I^{\text{syn}}(\alpha).$$

Proof sketch. The proof is done by an extensive case analysis. For each of the different operations, and every variable z, if $z \in O^{\text{syn}}(\alpha)$, then we can construct an interpretation D such that z is not inertial in $[\![\alpha]\!]_D$ and thus $z \in O^{\text{sem}}(\alpha)$. Similarly, for every variable $z \in I^{\text{syn}}(\alpha)$, we can construct an interpretation D as a witness of the fact that $\mathbb{V} - \{z\}$ does not determine $[\![\alpha]\!]$ on $O^{\text{sem}}(\alpha)$ and thus that $z \in I^{\text{sem}}(\alpha)$. □

Now, the precision theorem forms the basis for our main result on syntactic inputs and outputs, which states that Definition 16 yields the most precise sound and compositional input–output definition.

Theorem 21 (Optimality Theorem)**.** *Suppose (I, O) is a sound and compositional input–output definition. Then for each LIF expression α:*

$$I^{\text{syn}}(\alpha) \subseteq I(\alpha) \text{ and } O^{\text{syn}}(\alpha) \subseteq O(\alpha).$$

Proof sketch. The proof is by induction on the structure of α. For atomic module expressions α, this follows directly from Theorem 20. For $\alpha = id$ this is immediate since $I^{\text{syn}}(id) = O^{\text{syn}}(id) = \emptyset$.

We only give the inductive case for inputs, and α of the form $\alpha_1 \cup \alpha_2$. We define $\alpha'_1 = M_1(\bar{x}; \bar{y})$ and $\alpha'_2 = M_2(\bar{u}, \bar{v})$ where $\bar{x} = I(\alpha_1)$, $\bar{y} = O(\alpha_1)$, $\bar{u} = I(\alpha_2)$, and $\bar{v} = O(\alpha_2)$ with M_i distinct module names of the right arity.

Since (I, O) is compositional, we know that

$$I(\alpha_1 \cup \alpha_2) = I(\alpha'_1 \cup \alpha'_2). \tag{1}$$

We also know that

$$I(\alpha'_1 \cup \alpha'_2) \supseteq I^{\text{sem}}(\alpha'_1 \cup \alpha'_2) = I^{\text{syn}}(\alpha'_1 \cup \alpha'_2), \tag{2}$$

where the inclusion holds since (I, O) is sound and the equality follows from the Precision Theorem. We now **claim** that

$$I^{\text{syn}}(\alpha'_1 \cup \alpha'_2) \supseteq I^{\text{syn}}(\alpha_1 \cup \alpha_2). \tag{3}$$

By combining Equations (1–3), we find that

$$I(\alpha_1 \cup \alpha_2) \supseteq I^{\text{syn}}(\alpha_1 \cup \alpha_2),$$

which we needed to show.

All that is left to do is to **prove our claim**. From the inductive hypothesis, we know that for $i \in \{1, 2\}$:

$$I^{\text{syn}}(\alpha_i) \subseteq I(\alpha_i) = I(\alpha'_i) = I^{\text{syn}}(\alpha'_i)$$

α	$I^{\text{syn}}(\alpha)$	$O^{\text{syn}}(\alpha)$
id	$\emptyset$	$\emptyset$
$M(\overline{x};\overline{y})$	$\{x_1,\dots,x_n\}$ where $\overline{x}=x_1,\dots,x_n$	$\{y_1,\dots,y_n\}$ where $\overline{y}=y_1,\dots,y_n$
$\alpha_1\cup\alpha_2$	$I^{\text{syn}}(\alpha_1)\cup I^{\text{syn}}(\alpha_2)\cup(O^{\text{syn}}(\alpha_1)\mathbin{\triangle}O^{\text{syn}}(\alpha_2))$	$O^{\text{syn}}(\alpha_1)\cup O^{\text{syn}}(\alpha_2)$
$\alpha_1\cap\alpha_2$	$I^{\text{syn}}(\alpha_1)\cup I^{\text{syn}}(\alpha_2)\cup(O^{\text{syn}}(\alpha_1)\mathbin{\triangle}O^{\text{syn}}(\alpha_2))$	$O^{\text{syn}}(\alpha_1)\cap O^{\text{syn}}(\alpha_2)$
$\alpha_1-\alpha_2$	$I^{\text{syn}}(\alpha_1)\cup I^{\text{syn}}(\alpha_2)\cup(O^{\text{syn}}(\alpha_1)\mathbin{\triangle}O^{\text{syn}}(\alpha_2))$	$O^{\text{syn}}(\alpha_1)$
$\alpha_1\mathbin{;}\alpha_2$	$I^{\text{syn}}(\alpha_1)\cup(I^{\text{syn}}(\alpha_2)-O^{\text{syn}}(\alpha_1))$	$O^{\text{syn}}(\alpha_1)\cup O^{\text{syn}}(\alpha_2)$
$\text{cyl}^l_x(\alpha_1)$	$I^{\text{syn}}(\alpha_1)-\{x\}$	$O^{\text{syn}}(\alpha_1)\cup\{x\}$
$\text{cyl}^r_x(\alpha_1)$	$I^{\text{syn}}(\alpha_1)$	$O^{\text{syn}}(\alpha_1)\cup\{x\}$
$\sigma^{lr}_{x=y}(\alpha_1)$	$\begin{cases} I^{\text{syn}}(\alpha_1) & \text{if } x=_{\text{syn}}y \text{ and } y\notin O^{\text{syn}}(\alpha_1)\\ I^{\text{syn}}(\alpha_1)\cup\{x,y\} & \text{if } x\neq_{\text{syn}}y \text{ and } y\notin O^{\text{syn}}(\alpha_1)\\ I^{\text{syn}}(\alpha_1)\cup\{x\} & \text{otherwise}\end{cases}$	$\begin{cases} O^{\text{syn}}(\alpha_1)-\{x\} & \text{if } x=_{\text{syn}}y\\ O^{\text{syn}}(\alpha_1) & \text{otherwise}\end{cases}$
$\sigma^{l}_{x=y}(\alpha_1)$	$\begin{cases} I^{\text{syn}}(\alpha_1) & \text{if } x=_{\text{syn}}y\\ I^{\text{syn}}(\alpha_1)\cup\{x,y\} & \text{otherwise}\end{cases}$	$O^{\text{syn}}(\alpha_1)$
$\sigma^{r}_{x=y}(\alpha_1)$	$\begin{cases} I^{\text{syn}}(\alpha_1) & \text{if } x=_{\text{syn}}y\\ I^{\text{syn}}(\alpha_1)\cup(\{x,y\}-O^{\text{syn}}(\alpha_1)) & \text{otherwise}\end{cases}$	$O^{\text{syn}}(\alpha_1)$

Table 1: Syntactic inputs and outputs for LIF expressions.

and similarly for the outputs: $O^{\text{syn}}(\alpha_i)\subseteq O^{\text{syn}}(\alpha_i')$. To show the claim, take any variable

$$\begin{aligned} x &\in I^{\text{syn}}(\alpha_1\cup\alpha_2)\\ &= I^{\text{syn}}(\alpha_1)\cup I^{\text{syn}}(\alpha_2)\cup(O^{\text{syn}}(\alpha_1)\mathbin{\triangle}O^{\text{syn}}(\alpha_2)).\end{aligned}$$

If $x\in I^{\text{syn}}(\alpha_i)$ for some i, we know $x\in I^{\text{syn}}(\alpha_i')\subseteq I^{\text{syn}}(\alpha_1'\cup\alpha_2')$. If $x\in O^{\text{syn}}(\alpha_1)\mathbin{\triangle}O^{\text{syn}}(\alpha_2)$, we can assume by symmetry that $x\in O^{\text{syn}}(\alpha_1)\subseteq O^{\text{syn}}(\alpha_1')$, while $x\notin O^{\text{syn}}(\alpha_2)$. We make a distinction in two cases:

Case 1: If $x\notin O^{\text{syn}}(\alpha_2')$, then $x\in O^{\text{syn}}(\alpha_1')\mathbin{\triangle}O^{\text{syn}}(\alpha_2')\subseteq I^{\text{syn}}(\alpha_1'\cup\alpha_2')$ and the result easily follows.

Case 2: If $x\in O^{\text{syn}}(\alpha_2')$, then $x\in O(\alpha_2')=O(\alpha_2)$. Since $x\notin O^{\text{syn}}(\alpha_2)$, $x\notin O^{\text{sem}}(\alpha_2)$ either. Since (I,O) is sound and compositional, Lemma 15 then states that $x\in I(\alpha_2)=I(\alpha_2')=I^{\text{syn}}(\alpha_2')\subseteq I^{\text{syn}}(\alpha_1'\cup\alpha_2')$, which finishes our proof. $\square$

4 Primitivity of Composition

We now turn our attention to the study of composition in LIF. We begin by showing that for "well-behaved" expressions (all subexpressions have disjoint inputs and outputs) composition is redundant in LIF: every well-behaved LIF expression is equivalent to a LIF expression that does not use composition. As a corollary, we will obtain that composition is generally redundant if there is an infinite supply of variables. In contrast, in the bounded variable case, we will show that composition is primitive in LIF. Here, we use LIFnc to denote the fragment of LIF without composition.

4.1 When Input and Output are Disjoint, Composition is Non-Primitive

Our first non-primitivity result is based on inputs and outputs. We say that a LIF expression β is *io-disjoint* if $O^{\text{sem}}(\beta)\cap I^{\text{sem}}(\beta)=\emptyset$. The following theorem implies that if α, β, and all their subexpressions are io-disjoint, we can rewrite $\alpha\mathbin{;}\beta$ into a LIFnc expression.

Theorem 22. *Let α and β be LIF expressions such that β is io-disjoint. Then, $\alpha\mathbin{;}\beta$ is equivalent to*

$$\gamma:=\text{cyl}^r_{O^{\text{sem}}(\beta)}(\alpha)\cap\text{cyl}^l_{O^{\text{sem}}(\alpha)}(\beta).$$

Intuitively, the reason why this expression works is as follows: we cylindrify α on the right. In general, this might result in a loss of information, but since we are only cylindrifying outputs of β, this means we only forget the information that would be overwritten by β anyway. Since the inputs and outputs of β are disjoint, β does not need to know what α did to those variables in order to determine its own outputs. We also cylindrify β on the left on the outputs of α, since these values will be set by α. One then still needs to be careful in showing that the intersection indeed removes all artificial pairs, by exploiting the fact that expressions are inertial outside of their output.

Proof. Let D be an interpretation. First, we show that $[\![\alpha\mathbin{;}\beta]\!]_D\subseteq[\![\gamma]\!]_D$. If $(\nu_1,\nu_2)\in[\![\alpha\mathbin{;}\beta]\!]_D$, then there is a ν_3 such that $(\nu_1,\nu_3)\in[\![\alpha]\!]_D$ and $(\nu_3,\nu_2)\in[\![\beta]\!]_D$. By definition of the outputs of β, ν_3 and ν_2 agree outside of $O^{\text{sem}}(\beta)$. Hence, $(\nu_1,\nu_2)\in[\![\text{cyl}^r_{O^{\text{sem}}(\beta)}(\alpha)]\!]_D$. Similarly, we can show that $(\nu_1,\nu_2)\in[\![\text{cyl}^l_{O^{\text{sem}}(\alpha)}(\beta)]\!]_D$.

For the other inclusion, assume that $(\nu_1,\nu_2)\in[\![\gamma]\!]_D$. Using the definition of the semantics of cylindrification, we find ν_2' such that $(\nu_1,\nu_2')\in[\![\alpha]\!]_D$ and ν_2 agrees with ν_2' outside of $O^{\text{sem}}(\beta)$ and we find a ν_1' such that ν_1' agrees with ν_1 outside of $O^{\text{sem}}(\alpha)$ and $(\nu_1',\nu_2)\in[\![\beta]\!]_D$. Using the definition of output of β, we know that also ν_1' agrees with ν_2 outside of the outputs of β, thus ν_1' and ν_2' agree outside of the outputs of β, and hence definitely on the inputs of β. Since $(\nu_1',\nu_2)\in[\![\beta]\!]_D$ and Proposition 5 guarantees that β is determined by its inputs, there exists a ν_2'' such that $(\nu_2',\nu_2'')\in[\![\beta]\!]_D$ where $\nu_2''=\nu_2$ on the outputs of β and, since β is inertial outside its outputs, $\nu_2''=\nu_2'$ outside the outputs of β. But we previously established that ν_2' agrees with ν_2 outside of the outputs of β, hence

$\nu_2'' = \nu_2$. Summarized we now found that $(\nu_1, \nu_2') \in [\![\alpha]\!]_D$ and $(\nu_2', \nu_2) \in [\![\beta]\!]_D$, hence indeed $(\nu_1, \nu_2) \in [\![\alpha \, ; \beta]\!]_D$. $\square$

Given the undecidability results of the previous section, Theorem 22 is not effective. We can however give the following syntactic variant, which is proven similarly.

Theorem 23. *Let α and β be LIF expressions such that $O^{\text{syn}}(\beta) \cap I^{\text{syn}}(\beta) = \emptyset$. Then, $\alpha \, ; \beta$ is equivalent to*

$$\text{cyl}^r_{O^{\text{syn}}(\beta)}(\alpha) \cap \text{cyl}^l_{O^{\text{syn}}(\alpha)}(\beta).$$

The expression obtained in Theorem 22 does not work if β has overlapping inputs and outputs as the following example illustrates.

Example 24 (Example 7 continued)**.** *Consider the expression*

$$\alpha := P_1(x; x) \, ; P_1(x; x).$$

with the interpretation D as before. In this case, α increments the value of x by two. However, $[\![\text{cyl}^r_x(P_1(x;x))]\!]_D$ and $[\![\text{cyl}^l_x(P_1(x;x))]\!]_D$ are both equal to

$$\{(\nu_1, \nu_2) \mid \nu_1(z) = \nu_2(z) \text{ for all } z \neq x\}.$$

Hence indeed, in this case α is not equivalent to

$$\text{cyl}^r_x(P_1(x;x)) \cap \text{cyl}^l_x(P_1(x;x)).$$

4.2 If $\mathbb{V}$ is Infinite, Composition is Non-Primitive

We know from Theorem 22 that if β is io-disjoint, α and β can be composed without using composition. If $\mathbb{V}$ is sufficiently large, we can force any expression β to be io-disjoint by having β write its output onto unused variables instead of its actual outputs. The composition can then be eliminated following Theorem 22, after which we move the variables back so that the "correct" outputs are used. What we need to show is that the "moving the variables around", as described above, is expressible without composition. As before, we define the operators on BRVs but their definition is lifted to LIF expressions in a straightforward way.

Definition 25. *Let B be a BRV and let $\bar{x}$ and $\bar{y}$ be disjoint tuples of distinct variables of the same length. The* right move *is defined as follows:*

$$\begin{aligned}\text{mv}^r_{\bar{x}\to\bar{y}}(B) := \{(\nu_1, \nu_2) \mid \nu_2(\bar{x}) = \nu_1(\bar{x}) \\ \text{and } \exists \nu_2' : (\nu_1, \nu_2') \in B \text{ and } \nu_2'(\bar{x}) = \nu_2(\bar{y}) \\ \text{and } \nu_2 = \nu_2' \text{ outside } \bar{x} \cup \bar{y}\}.\end{aligned}$$

This operation can be expressed without composition.

Lemma 26. *For any BRV B, we have*

$$\text{mv}^r_{\bar{x}\to\bar{y}}(B) = \sigma^{lr}_{\bar{x}=\bar{x}} \text{cyl}^r_{\bar{x}} \sigma^r_{\bar{x}=\bar{y}} \text{cyl}^r_{\bar{y}}(B)$$

The intuition for the expression in Lemma 26 is as follows. We first cylindrify the $\bar{y}$ variables on the right so that the subsequent selection effectively copies $\bar{x}$ to $\bar{y}$. The final two operations make sure that the $\bar{x}$ variables are inertial.

Lemma 27. *Let A and B be BRVs and let $\bar{x}$ and $\bar{y}$ be disjoint tuples of distinct variables of the same length such that all variables in $\bar{y}$ are inertially cylindrified in A and B. In that case:*

$$A \, ; B = \text{mv}^r_{\bar{y}\to\bar{x}}(A \, ; \text{mv}^r_{\bar{x}\to\bar{y}}(B)).$$

What Lemma 27 shows is that we can temporarily move certain variables (the $\bar{x}$) to unused variables (the $\bar{y}$) and then move them back. The proof of this lemma is based on the properties that **(i)** $A \, ; \text{mv}^r_{\bar{x}\to\bar{y}}(B) = \text{mv}^r_{\bar{x}\to\bar{y}}(A \, ; B)$ and that **(ii)** if the variables in $\bar{y}$ are inertially cylindrified in a BRV C, then $\text{mv}^r_{\bar{y}\to\bar{x}}(\text{mv}^r_{\bar{x}\to\bar{y}}(C)) = C$. This finally brings us to the main result of the current subsection.

Theorem 28. *If $\mathbb{V}$ is infinite, then every LIF expression is equivalent to a LIFnc expression.*

Proof Sketch. For a LIF expression $\alpha \, ; \beta$ we can choose a tuple of variables $\bar{y}$ of the same length as $O^{\text{syn}}(\beta)$, such that $\bar{y}$ does not occur in $\alpha \, ; \beta$. In that case, $\bar{y}$ is inertially cylindrified in α and in β and hence Lemma 27 yields that $\alpha \, ; \beta$ is equivalent to

$$\text{mv}^r_{\bar{y}\to O^{\text{syn}}(\beta)}(\alpha \, ; \text{mv}^r_{O^{\text{syn}}(\beta)\to\bar{y}}(\beta)).$$

Furthermore, it can be verified that $\text{mv}^r_{O^{\text{syn}}(\beta)\to\bar{y}}(\beta)$ is syntactically io-disjoint. Hence, we can apply Theorem 23 to eliminate the composition. This construction can be applied recursively to eliminate all occurrences of composition. $\square$

4.3 If $\mathbb{V}$ is Finite, Composition is Primitive

The case that remains is when $\mathbb{V}$ is finite. We will show that in this case, composition is indeed primitive by relating bounded-variable LIF to bounded-variable first-order logic.

Assume $\mathbb{V} = \{x_1, \ldots, x_n\}$. Since BRVs involve pairs of $\mathbb{V}$-valuations, we introduce a copy $\mathbb{V}_y = \{y_1, \ldots, y_n\}$ disjoint from $\mathbb{V}$. For notational convenience we also write $\mathbb{V}_x$ for $\mathbb{V}$. As usual, by FO[k] we denote the fragment of first-order logic that uses only k distinct variables. We observe:

Proposition 29. *For every LIF expression α, there exists an* FO[$3n$] *formula φ_α with free variables in $\mathbb{V}_x \cup \mathbb{V}_y$ such that*

$$(\nu_1, \nu_2) \in [\![\alpha]\!]_D \textit{ iff } D, (\nu_1 \cup \nu_2') \models \varphi_\alpha,$$

where ν_2' is the $\mathbb{V}_y$-valuation such that $\nu_2'(y_i) = \nu_2(x_i)$ for each i. Furthermore, if α is a LIFnc expression, φ_α can be taken to be a FO[$2n$] *formula.*

Proof. The proof is by induction on the structure of α (using Lemma 1, we omit redundant operators).

We introduce a third copy $\mathbb{V}_z = \{z_1, \ldots, z_n\}$ of $\mathbb{V}$. For every $u, v \in \{x, y, z\}$ we define ρ_{uv} as follows:

$$\rho_{uv} : \mathbb{V}_u \to \mathbb{V}_v : u_i \mapsto v_i$$

Using these functions, we can translate a valuation ν on $\mathbb{V} = \mathbb{V}_x$ to a corresponding valuation on $\mathbb{V}_u$ with $u \in \{y, z\}$. Clearly, $\nu \circ \rho_{ux}$ does this job.

In the proof, we actually show a stronger statement by induction, namely that for each α and for every $u \neq v \in \{x, y, z\}$ there is a formula φ^{uv}_α with free variables in $\mathbb{V}_u \cup \mathbb{V}_v$ in FO[$\mathbb{V}_x \cup \mathbb{V}_y \cup \mathbb{V}_z$] such that for every D:

$$(\nu_1, \nu_2) \in [\![\alpha]\!]_D \text{ iff } D, (\nu_1 \circ \rho_{ux} \cup \nu_2 \circ \rho_{vx}) \models \varphi^{uv}_\alpha.$$

Since the notations x, y, z, u and v are taken, we use notations a, b and c for variables.

- $\alpha = id$. For φ^{uv}_α we can take $\bigwedge_{i=1}^n u_i = v_i$.

- $\alpha = M(\overline{a}; \overline{b})$. For φ_α^{uv} we can take $M(\rho_{xu}(\overline{a}), \rho_{xv}(\overline{b})) \wedge \bigwedge_{c \notin \overline{b}} \rho_{xu}(c) = \rho_{xv}(c)$.
- $\alpha = \alpha_1 \cup \alpha_2$, we take $\varphi_\alpha^{uv} = \varphi_{\alpha_1}^{uv} \vee \varphi_{\alpha_2}^{uv}$.
- $\alpha = \mathrm{cyl}_a^l(\alpha_1)$. For φ_α^{uv} we can take $\exists \rho_{xu}(a)\ \varphi_{\alpha_1}^{uv}$.
- $\alpha = \mathrm{cyl}_a^r(\alpha_1)$. For φ_α^{uv} we can take $\exists \rho_{xv}(a)\ \varphi_{\alpha_1}^{uv}$.
- $\alpha = \alpha_1 - \alpha_2$. For φ_α^{uv} we can take $\varphi_{\alpha_1}^{uv} \wedge \neg\varphi_{\alpha_2}^{uv}$.
- $\alpha = \sigma_{a=b}^{lr}(\alpha_1)$. For φ_α^{uv} we can take $\varphi_{\alpha_1} \wedge \rho_{xu}(a) = \rho_{xv}(b)$.
- $\alpha = \alpha_1 \,;\, \alpha_2$. Let $w \in \{x, y, z\} - \{u, v\}$. Define $\varphi_\alpha^{uv} = \exists w_1 \ldots \exists w_n\ (\varphi_{\alpha_1}^{uw} \wedge \varphi_{\alpha_2}^{wv})$. ☐

Now that we have established that LIFnc can be translated into FO$[2n]$, all that is left to do is find a Boolean query that can be expressed in LIF with n variables, but not in FO$[2n]$. We find such a query in the existence of a $3n$-clique. We will first show that we can construct a LIFnc expression α_{2n} such that, given an interpretation D interpreting a binary relation R, $[\![\alpha_{2n}]\!]_D$ consists of all $2n$-cliques of R. Next, we show how α_{2n} can be used (with composition) to construct an expression $\alpha_{\exists 3n}$ such that $[\![\alpha_{\exists 3n}]\!]_D$ is non-empty if and only if R has a $3n$-clique. Since this property cannot be expressed in FO$[2n]$, we can conclude that composition must be primitive.

To avoid confusion, we recall that a set L of k data elements is a k-clique in a binary relation r, if any two distinct a and b in L, we have $(a, b) \in r$ (and also $(b, a) \in r$).

Proposition 30. *Suppose that* $|\mathbb{V}| = n$ *with* $n \geq 2$ *and let* $\mathcal{S} = \{R\}$ *with* $\mathrm{ar}(R) = \mathrm{iar}(R) = 2$. *There exists a LIF expression* α_{2n} *such that*

$$[\![\alpha_{2n}]\!]_D = \{(\nu_1, \nu_2) \mid \nu_1(\mathbb{V}) \cup \nu_2(\mathbb{V}) \text{ is a } 2n\text{-clique in } D(R)\}.$$

Proof. Throughout this proof, we identify a pair (ν_1, ν_2) of two valuations with the $2n$ tuple of data elements

$$\nu_1(x_1, \ldots, x_n) \cdot \nu_2(x_1, \ldots, x_n).$$

Before coming to the actual expression for α_{2n}, we introduce some auxiliary concepts. First, we define

$$\mathit{all} := \mathrm{cyl}_{\mathbb{V}}^l\, \mathrm{cyl}_{\mathbb{V}}^r(\mathit{id}).$$

It is clear that

$$[\![\mathit{all}]\!]_D = \{(\nu_1, \nu_2) \in \mathcal{V} \times \mathcal{V}\}.$$

A first condition for being a $2n$-clique is that all data elements are different. It is clear that the expression

$$\alpha_= := \bigcup_{x \neq y \in \mathbb{V}} \left(\sigma_{x=y}^l(\mathit{all}) \cup \sigma_{x=y}^r(\mathit{all})\right) \cup \bigcup_{x,y \in \mathbb{V}} \sigma_{x=y}^{lr}(\mathit{all})$$

has the property that $[\![\alpha_=]\!]_D$ consists of all $2n$-tuples where at least one data element is repeated. Hence

$$\alpha_{\neq} := \mathit{all} - \alpha_=$$

is such that $[\![\alpha_{\neq}]\!]_D$ consists of all $2n$-tuples of distinct data elements.

The second condition for being a $2n$-clique is that each two distinct elements are connected by R. For checking this, for each two variables x and y we define

$$R_{x,y}^l := \mathrm{cyl}_{\mathbb{V}-\{x,y\}}^l \mathrm{cyl}_{\mathbb{V}}^r(R(x,y) \cap R(y,x))$$
$$R_{x,y}^r := \mathrm{cyl}_{\mathbb{V}}^l \mathrm{cyl}_{\mathbb{V}-\{x,y\}}^r(R(x,y) \cap R(y,x))$$
$$R_{x,y}^{lr} := \mathrm{cyl}_{\mathbb{V}-\{x\}}^l \mathrm{cyl}_{\mathbb{V}-\{y\}}^r(R(x,y) \cap R(y,x)).$$

With these definitions, for instance $[\![R_{x_i,x_j}^{lr}]\!]_D$ consists of all $2n$-tuples such that the i^{th} and the $n + j^{\text{th}}$ element are connected (in two directions) in R, and similar properties hold for R^l and R^r. From this, it follows that the expression

$$\alpha_{2n} = \alpha_{\neq} \cap \bigcap_{x \neq y \in \mathbb{V}} \left(R_{x,y}^l \cap R_{x,y}^r\right) \cap \bigcap_{x,y \in \mathbb{V}} R_{x,y}^{lr}$$

satisfies the proposition; it intersects $\alpha_{\neq}$ with all the expressions stating that each two data elements must be (bidirectionally) connected by R. ☐

Notice that α_{2n} can be used to compute *all* the $2n$-cliques of the input interpretation. We now use α_{2n} to check for existence of $3n$-cliques.

Proposition 31. *Suppose that* $|\mathbb{V}| = n$ *with* $n \geq 2$ *and let* $\mathcal{S} = \{R\}$ *with* $\mathrm{ar}(R) = \mathrm{iar}(R) = 2$. *Define*

$$\alpha_{\exists 3n} := (\alpha_{2n} \,;\, \alpha_{2n}) \cap \alpha_{2n}.$$

Then, for every interpretation D, $[\![\alpha_{\exists 3n}]\!]_D$ *is non-empty if and only if* $D(R)$ *has a* $3n$*-clique.*

It is well known that existence of a $3n$-clique is not expressible in FO$[2n]$ (Ebbinghaus and Flum 1999). The above proposition thus immediately implies

Theorem 32. *Suppose that* $|\mathbb{V}| = n \geq 2$. *Then, composition is primitive in LIF. Specifically, no LIFnc expression is equivalent to the LIF expression* $\alpha_{\exists 3n}$.

5 Related Work

LIF grew out of the **Algebra of Modular Systems** (Ternovska 2015), which was developed to provide foundations for programming from available components. That paper mentions information flows, in connection with input–output behaviour in classical logic, for the first time. The paper also surveys earlier work from the author's group, as well as other closely related work.

In a companion paper (Aamer et al. 2020), we report on an application of LIF to **querying under limited access patterns**, as for instance offered by web services (Mcilraith, Son, and Zeng 2001). That work also involves inputs and outputs, but only of a syntactic nature, and for a restricted variant of LIF (called "forward" LIF) only. The property of io-disjointness turned also to be important in that work, albeit for a quite different purpose.

Our results also relate to **the evaluation problem for LIF**, which takes as input a LIF expression α, an interpretation D, and a valuation ν_1, and where the task is to find all ν_2 such that $(\nu_1, \nu_2) \in [\![\alpha]\!]_D$. From our results, it follows that only the value of ν_1 on the input variables is important, and similarly we are only interested in the values of each ν_2 on the output variables. A subtle point, however, is that D may be infinite, and moreover, even if D itself is not infinite, the output of the evaluation problem may still be. In many cases, it is still possible to obtain a finite representation, for instance by using quantifier elimination techniques as done in Constraint Databases (Kuper, Libkin, and Paredaens 2000).

We have defined the semantics of LIF algebraically, in the style of **cylindric set algebra** (Henkin, Monk, and Tarski 1971; Imielinski and Lipski 1984). An important difference is the dynamic nature of BRVs which are sets of *pairs* of valuations, as opposed to sets of valuations which are the basic objects in cylindric set algebra.

Our optimality theorem was inspired by work on **controlled FO** (Fan, Geerts, and Libkin 2014), which had as aim to infer boundedness properties of the outputs of first-order queries, given boundedness properties of the input relations. Since this inference task is undecidable, the authors defined syntactic inferences similar in spirit to our syntactic definition of inputs and outputs. They show (their Proposition 4.3) that their definitions are, in a sense, sharp. Note that our optimality theorem is stronger in that it shows that no other compositional and sound definition can be better than ours. Of course, the comparison between the two results is only superficial as the inference tasks at hand are very different.

The Logic of Information Flows is similar to **dynamic predicate logic** (DPL) (Groenendijk and Stokhof 1991), in the sense that formulas are also evaluated with respect to pairs of valuations. There is, however a key difference in philosophy between the two logics. LIF starts from the idea that well-known operators from first-order logic can be used to describe combinations and manipulations of dynamic systems, and as such provides a means for procedural knowledge in a declarative language. The dynamics in LIF are dynamics of the described system. Dynamic predicate logic, on the other hand starts from the observation that, in natural language, operators such as conjunction and existential quantification are dynamic, where the dynamics are in the process of parsing a sentence, often related to coreference analysis. To the best of our knowledge, inputs and outputs of expressions have not been studied in DPL.

Since we developed a large part of our work in the general setting of BRVs, and thus of **transition systems**, we expect several of our results to be applicable in the context of other formalisms where specifying inputs and outputs is important, such as API-based programming (Calvanese et al. 2016) and synthesis (De Giacomo, Patrizi, and Sardiña 2013; Alechina et al. 2019), privacy and security, business process modeling (Calvanese et al. 2008), and model combinators in Constraint Programming (Fontaine, Michel, and Van Hentenryck 2013).

6 Conclusion and Future Work

Declarative modeling is of central importance in the area of Knowledge Representation and Reasoning. The Logic of Information Flows provides a framework to investigate how, and to what degree, dynamic or imperative features can be modeled declaratively. In this paper we have focused on inputs, outputs, and sequential composition, as these three concepts are fundamental to modeling dynamic systems. There are many directions for further research.

Inputs and outputs are not just relevant from a theoretic perspective, but can also have ramifications on computation. Indeed, they form a first handle to parallelize computation of complex LIF expressions, or to decompose problems.

In this paper, we have worked with a basic set of operations motivated by the classical logic connectives. In order to provide a fine control of computational complexity, or to increase expressiveness, it makes sense to consider other operations.

The semantic notions developed in this paper (inputs, outputs, soundness) apply to global BRVs in general, and hence are robust under varying the set of operations. Moreover, our work delineates and demonstrates a methodology for adapting syntactic input–output definitions to other operations.

A specific operation that is natural to consider is *converse*. The converse of a BRV A is defined to be $\{(\nu_2, \nu_1) \mid (\nu_1, \nu_2) \in A\}$. In the context of LIF (Ternovska 2019) it can model constraint solving by searching for an input to a module that produces a desired outcome. When we add converse to LIF with only a single variable ($|\mathbb{V}| = 1$), and the vocabulary has only binary relations of input arity one, then we obtain the classical calculus of relations (Tarski 1941). There, converse is known to be primitive (Fletcher et al. 2015). When the number of variables is strictly more than half of the maximum arity of relations in the vocabulary, converse is redundant in LIF, as can be shown using similar techniques as used in this paper to show redundancy of composition. Investigating the exact number of variables needed for non-primitivity is an interesting question for further research.

Another direction for further research is to examine fragments of LIF for which the semantic input or output problem may be decidable, or even for which the syntactic definitions coincide with the semantic definitions.

Finally, an operation that often occurs in dynamic systems is the fixed point construct used by Ternovska (2019). It remains to be seen how our work, and the further research directions mentioned above, can be extended to include the fixpoint operation.

Acknowledgments

This research received funding from the Flemish Government under the “Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen” programme, from FWO Flanders project G0D9616N, and from Natural Sciences and Engineering Research Council of Canada (NSERC). Jan Van den Bussche is partially supported by the National Natural Science Foundation of China (61972455).

References

Aamer, H.; Bogaerts, B.; Surinx, D.; Ternovska, E.; and Van den Bussche, J. 2020. Executable first-order queries in the logic of information flows. In *Proceedings 23rd International Conference on Database Theory*, volume 155 of *Leibniz International Proceedings in Informatics*, 4:1–4:14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Alechina, N.; Brázdil, T.; De Giacomo, G.; Felli, P.; Logan, B.; and Vardi, M. Y. 2019. Unbounded orchestrations of transducers for manufacturing. In *AAAI*, 2646–2653. AAAI Press.

Calvanese, D.; De Giacomo, G.; Lenzerini, M.; Mecella, M.; and Patrizi, F. 2008. Automatic service composition and synthesis: the roman model. *IEEE Data Eng. Bull.* 31(3):18–22.

Calvanese, D.; De Giacomo, G.; Lenzerini, M.; and Vardi, M. Y. 2016. Regular open APIs. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifteenth International Conference, KR 2016, Cape Town, South Africa, April 25-29, 2016.*, 329–338.

De Giacomo, G.; Patrizi, F.; and Sardiña, S. 2013. Automatic behavior composition synthesis. *Artif. Intell.* 196:106–142.

Ebbinghaus, H.-D., and Flum, J. 1999. *Finite Model Theory*. Springer, second edition.

Enderton, H. B. 1972. *A Mathematical Introduction To Logic*. Academic Press.

Fan, W.; Geerts, F.; and Libkin, L. 2014. On scale independence for querying big data. In *Proceedings 33th ACM Symposium on Principles of Database Systems*, 51–62.

Fletcher, G.; Gyssens, M.; Leinders, D.; Surinx, D.; Van den Bussche, J.; Van Gucht, D.; Vansummeren, S.; and Wu, Y. 2015. Relative expressive power of navigational querying on graphs. *Information Sciences* 298:390–406.

Fontaine, D.; Michel, L.; and Van Hentenryck, P. 2013. Model combinators for hybrid optimization. In *Principles and Practice of Constraint Programming - 19th International Conference, CP 2013, Uppsala, Sweden, September 16-20, 2013. Proceedings*, 299–314.

Groenendijk, J., and Stokhof, M. 1991. Dynamic predicate logic. *Linguistics and Philosophy* 14:39–100.

Gurevich, Y. 1983. Algebras of feasible functions. In *Proceedings 24th Symposium on Foundations of Computer Science*, 210–214. IEEE Computer Society.

Gurevich, Y. 1988. Logic and the challenge of computer science. In Börger, E., ed., *Current Trends in Theoretical Computer Science*. Computer Science Press. 1–57.

Henkin, L.; Monk, J.; and Tarski, A. 1971. *Cylindric Algebras. Part I.* North-Holland.

Imielinski, T., and Lipski, W. 1984. The relational model of data and cylindric algebras. *Journal of Computer and System Sciences* 28:80–102.

Kuper, G.; Libkin, L.; and Paredaens, J., eds. 2000. *Constraint Databases*. Springer.

Lewis, D. 1973. Causation. *Journal of Philosophy* 70:113–126.

Lifschitz, V. 1987. Formal theories of action (preliminary report). In McDermott, J. P., ed., *Proceedings of the 10th International Joint Conference on Artificial Intelligence. Milan, Italy, August 23-28, 1987*, 966–972. Morgan Kaufmann.

McCarthy, J., and Hayes, P. J. 1969. Some philosophical problems from the standpoint of artificial intelligence. In Meltzer, B., and Michie, D., eds., *Machine Intelligence 4*. Edinburgh University Press. 463–502.

Mcilraith, S.; Son, T.; and Zeng, H. 2001. Semantic web services. *Intelligent Systems, IEEE* 16:46 – 53.

Mitchell, D. G., and Ternovska, E. 2005. A framework for representing and solving NP search problems. In *Proc. AAAI*, 430–435.

Tarski, A. 1941. On the calculus of relations. *Journal of Symbolic Logic* 6:73–89.

Ternovska, E. 2015. An algebra of combined constraint solving. In *Global Conference on Artificial Intelligence, GCAI 2015, Tbilisi, Georgia, October 16-19, 2015*, 275–295.

Ternovska, E. 2017. Recent progress on the algebra of modular systems. In Reutter, J., and Srivastava, D., eds., *Proceedings 11th Alberto Mendelzon International Workshop on Foundations of Data Management*, volume 1912 of *CEUR Workshop Proceedings*.

Ternovska, E. 2019. An algebra of modular systems: static and dynamic perspectives. In Herzig, A., and Popescu, A., eds., *Frontiers of Combining Systems: Proceedings 12th FroCos*, volume 11715 of *Lecture Notes in Artificial Intelligence*, 94–111. Springer.