

A Logic of Information Flows

Eugenia Ternovska

Simon Fraser University, Canada
ter@sfu.ca

In this abstract, we describe a recently published paper where we introduced a Logic of Information Flows (Ternovska 2019). The paper is published in the Proceedings of the 12th International Symposium on Frontiers of Combining Systems (FroCoS), September 2019.

This paper is a part of a much larger project, where we look at Declarative Reasoning, which is at the very core of Knowledge Representation, and ask the following questions:

- *What is that that distinguishes declarative reasoning?*
- *How, and to what degree can dynamic (or imperative) features be modelled declaratively?*
- *Is it even possible to model all efficient (polynomial time) computations in a logic?*

As things currently stand, it is entirely possible that there will always be problems that are solvable by imperative programs of polynomial complexity, that are not solvable declaratively at all, in a single formalism. This is stated in a famous conjecture due to Gurevich.

We look at computation in a very general way, where there are atomic information-producing units, whose computational power is then exploited, by arranging them in a suitable manner, to produce a desirable outcome.

We introduce the Logic of Information Flows in two stages. The first stage is a *static* algebra, which is an algebraization of classical logic; and the second one is *dynamic*, or an algebra of binary relations.

We start with first-order logic with fixpoints (FO(LFP)). We use an algebraic syntax, similar to Codd's relational algebra, but the idea is the same. We redefine FO(LFP) over a vocabulary of *atomic module symbols* that replaces a relational vocabulary. In this static setting, each atomic symbol is interpreted as a set of structures rather than a relational table (set of tuples). That is, each atomic unit corresponds to a boolean query, or a decision procedure. Thus, while the syntax is first-order, the semantics is second-order because variables range over relations. This gives us the first logic.

The second stage is developing a **Logic of Information Flows (LIF)**. It is a dynamic setting where we add the ability to reason about *propagation of information from inputs to outputs*. Formally, it is given by two functions, *I* and *O* that indicate which relational variables of atomic modules are inputs and which are outputs. Semantically, modules are *binary relations on structures*. The relations describe how

information propagates. This gives us an *algebra of binary relations*, where we can reason about information flows.

Related Work Algebras of binary relations have been studied before. Such an algebra was first introduced by De Morgan. It has been extensively developed by Peirce and then Schröder. It was abstracted to relation algebra RA by Jónsson and Tarski in (Jónsson and Tarski 1952). For a historic perspective please see Pratt's informative historic overview paper (Pratt 1992). More recently, relation algebras were studied in, e.g. (Surinx, den Bussche, and Gucht 2017; Fletcher et al. 2015). The algebras of relations consider various subsets of operations on binary relations as primitive, and other as derivable. In another direction, (Jackson and Stokes 2011; McLean 2017) and others study partial functions and their algebraic equational axiomatizations.

Since, in our logic, all the variables that are not constrained by the algebraic expressions are implicitly cylindrified (i.e., interpreted arbitrarily), the closest related work is that on cylindric algebras (Henkin, Monk, and Tarski 1971). These algebras were introduced by Tarski and others as a tool in the algebraization of the first-order predicate calculus.¹ However, a fundamental difference is that, in our logic, unconstrained variables are not only cylindrified, but their interpretations, if not explicitly modified, are preserved; this important semantic principle is a realization of the commonsense law of inertia (McCarthy and Hayes 1969; Lifschitz 1987). This property gives us, mathematically, a very different formalism, suitable for reasoning about the dynamics of information flows.

Technical Contributions For essentially the same syntax, we develop two algebras, under static and dynamic semantics, respectively. The algebraic operations combine modules that communicate through common second-order variables. The dynamic algebra is called the Logic of Information Flows. Atomic modules are axiomatized in any formalism which can be interpreted over first-order structures.

An important feature of the logic is that it has a two-level syntax. Thus, there are two ways to control the expressive power: by means of restricting the algebraic operations, and by selecting the logics for axiomatizing atomic units.

¹See (Van den Bussche 2001) for a historic context in applications to Database theory.

We show that, just by adding input and output specifications to classical logic (here, in an algebraic form), we obtain multiple operations that are given as primitive in many algebras of binary relations.

We prove that, when interpreted over a pointed Kripke structure (given by the binary relations), the dynamic algebra is equivalent to a modal temporal (dynamic) logic. The logic allows one to specify patterns of execution inside (definable) modalities, similar to Dynamic Logic (see, e.g., (Harel, Kozen, and Tiuryn 2000)) and LDL_f (De Giacomo and Vardi 2013). Just like in PDL and LDL_f , the main constructs of imperative programming (e.g., *while* loops) are definable. The logic is also somewhat similar to Reiter's situation calculus (Reiter 2001) and Golog (Levesque et al. 1997). In place of Golog programs, we have algebraic terms inside the modalities that specify desired patterns of execution. However unlike the first-order successor state axioms of the situation calculus, atomic modules can be non-deterministic, and be specified in any logic, with any expressive power.

The Logic of Information Flows makes some initial steps towards answering the questions we asked. However, an important question was left open in the FroCoS paper (and conjectured to have a positive answer by the author): *can LIF define sequential composition?* Note that it is a quintessentially *dynamic* operation. We are happy to report, in a joint paper with Heba Aamer, Bart Bogaerts, Dimitri Surinx and Jan Van den Bussche (Aamer et al. 2020b) at this conference, an affirmative answer to this question. In the general setting of LIF with an unrestricted number of variables, sequential composition is *definable*. This result demonstrates the power of declarative reasoning over imperative computations. However, the operation is not definable when the number of variables is bounded by a constant. The paper also addresses the problem of characterizing inputs and outputs of compound expressions, both syntactically and semantically.

Applications As one practical application, we view LIF as a formalism for specifying and reasoning about modular systems, where one can combine reusable components, potentially written in different languages, for solving complex computational tasks.²

Recent developments on the boundary of KR and Data Management concern an application of a logic in the LIF family to **querying with limited pattern access in databases**, which is, again, a collaborative work (Aamer et al. 2020a). The work was presented at ICDT earlier this year.

As we mentioned earlier, LIF provides two mechanisms for controlling the expressive power (and thus computational complexity) — through the axiomatizations of atomic units and through the algebraic operations for combining them. Both can be restricted to provide *complexity guarantees* for the underlying computations. A lot of work is currently under way in the direction of studying the computational effects of restrictions and extensions of the LIF language.

²The heterogeneous components could be web services, knowledge bases, declarative specifications such as Integer Linear Programs, Constraint Satisfaction Problems, Answer Set Programs etc.

References

- Aamer, H.; Bogaerts, B.; Surinx, D.; Ternovska, E.; and Van den Bussche, J. 2020a. Executable first-order queries in the logic of information flows. In *Proc. ICDT'20*, 4:1–4:14.
- Aamer, H.; Bogaerts, B.; Surinx, D.; Ternovska, E.; and Van den Bussche, J. 2020b. Inputs, Outputs, and Composition in the Logic of Information Flows. In *KR'2020*.
- De Giacomo, G., and Vardi, M. Y. 2013. Linear temporal logic and linear dynamic logic on finite traces. In *IJCAI 2013*, 854–860.
- Fletcher, G.; Gyssens, M.; Leinders, D.; Surinx, D.; den Bussche, J. V.; Gucht, D. V.; Vansummeren, S.; and Wu, Y. 2015. Relative expressive power of navigational querying on graphs. *Information Sciences* 298:390–406.
- Harel, D.; Kozen, D.; and Tiuryn, J. 2000. *Dynamic Logic (Foundations of Computing)*.
- Henkin, L.; Monk, J.; and Tarski, A. 1971. *Cylindric Algebras Part I*. North-Holland, Amsterdam.
- Jackson, M., and Stokes, T. 2011. Modal restriction semigroups: towards an algebra of functions. *IJAC* 21(7):1053–1095.
- Jónsson, B., and Tarski, A. 1952. Representation problems for relation algebras. *Bull. Amer. Math. Soc.* 74:127–162.
- Levesque, H.; Reiter, R.; Lespérance, Y.; Lin, F.; and Scherl, R. 1997. GOLOG: a logic programming language for dynamic domains. *J. of Logic Programming, Special Issue on Actions* 31(1-3):59–83.
- Lifschitz, V. 1987. Formal theories of action (preliminary report). In *IJCAI'87*, 966–972.
- 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.
- McLean, B. 2017. Complete representation by partial functions for composition, intersection and anti-domain. *J. Log. Comput.* 27(4):1143–1156.
- Pratt, V. R. 1992. Origins of the calculus of binary relations. In (*LICS '92*), *Santa Cruz, California, USA, June 22-25, 1992*, 248–254.
- Reiter, R. 2001. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press.
- Surinx, D.; den Bussche, J. V.; and Gucht, D. V. 2017. The primitivity of operators in the algebra of binary relations under conjunctions of containments. In *LICS '17*.
- Ternovska, E. 2019. An algebra of modular systems: Static and dynamic perspectives. In *Proc. of the 12th Int. Symp. on Frontiers of Combining Systems (FroCoS)*. https://www2.cs.sfu.ca/ter/my_web_page/Papers_files/2019-FROCOS.pdf.
- Van den Bussche, J. 2001. Applications of Alfred Tarski's ideas in database theory. In *CSL*, volume 2142 of *Lecture Notes in Computer Science*, 20–37. Springer.