DELTA:
DÉfis pour la Logique, les Transducteurs et les Automates

1 Scientific context and challenges

Context. Software systems are ubiquitous, and more and more complex. The automated analysis of computer-generated data and the reliability analysis of complex programs become therefore crucial, and challenging due to the increasing size and intricacy of the objects that software systems have to face. Model-checking is a successful technique for system verification, that is now mature, and used for industrial purposes: it amounts to verifying the correctness of the system with respect to some specification. The core of this approach relies on *finite-state machines*: on the one hand, they provide an operational model of systems, and on the other hand, they can be considered as a low-level description of specifications. Finite-state machines processing various kinds of objects, such as words or trees, are very appealing because they capture one of the most robust concepts in computer science: the notion of *regularity*. This notion unifies a triangle (a *Delta*!) of different approaches with longstanding tradition: an operational one (automata), a descriptive one (logic), and an equational one (algebra). Logic is a powerful and precise formalism for specifying properties of systems, whereas automata offer an effective algorithmic tool, for instance for verifying and certifying properties; algebra comes naturally into play when we ask how far we can go with a specification formalism. The interplay summarized by the perfect Δ of automata-logic-algebra is best understood for languages of words, where a rich theory has developed since the fundamental results of the sixties. Since then, the theory has grown in many directions, raising new and important challenges.

Current challenges. Modeling by classical finite-state machines suffices for very simple systems or programs whose variables range over bounded domains. But most realistic systems require to cope with values from *unbounded domains*. Let us motivate the new difficulties on three symbolic examples, where finite domains are not sufficient anymore. First, when modeling embedded software, values may represent the energy level of the system, and we may be interested in measuring maximal or average quantity of energy consumption: we need to reason about *quantities*. Second, looking for a text pattern in a mailbox can be solved very efficiently with finite-state machines, but what about searching for patterns where we must compare values, like time-stamps or process identifiers? We need to reason about values from an *unbounded domain*. A last example comes from the verification of stream-processing programs, that aims, *e.g.*, at building on-the-fly some output flow of relevant elements out of a linear sequence of input events: the difficulty comes again from unbounded data, and requirements on the memory of the on-the-fly process of the input flow.

Objectives. New theories must be devised to solve these new issues. The challenge is to retain all the advantages of the classical theory, in particular simplicity and efficiency of finite-state machines, while handling values from unbounded domains, in keeping with the needs of the aforementioned motivations. The DELTA project thus proposes a fundamental research program with motivations from automatic verification and static analysis for database systems.

Our primary objective is to extend these connections between automata, logic and algebra to richer settings that can handle values, with the two following goals in mind:

1. We first aim at **developing appropriate models** for the three facets of the Δ (operational, descriptive, and equational), in the three extensions that we want to consider: quantitative values, documents carrying data, string-processing programs (also called transducers).
2. The second objective concerns the efficiency in processing the descriptive and operational models, like **simplification** of specifications and machines, or **efficient translation** of high-level specifications into low-level machines.

Section 2 is structured by following the extensions and models we plan to develop, which will explain the first objective. Our second goal will be explained and highlighted all along the text.

2 Research Objectives

Quantitative properties and systems. An important trend in the automata theory community is to develop a robust notion of regularity in quantitative settings. A prime example is provided by T. Colcombet's theory of *regular cost functions*[1] that allows to reason about asymptotic bounds on parameters. Regular cost functions enjoy equivalent characterizations in terms of cost automata, recognizability by (stabilization) monoids and cost monadic logic: the perfect triangle is formed once more and several important algorithmic questions are decidable. However, **extensions** of the theory to more complex structures required for reasoning on reactive systems, like infinite trees, remain wide open so far. **Approximate analysis** of quantitative models of automata is another of our research goals. It is related to regular cost functions and quantitative program analysis, like estimation time of program termination.

An orthogonal direction we want to investigate is the search of *high level specification languages* to reason about quantities, and algorithms for verifying such properties against a system. The road towards such languages has taken a new direction with the investigation on weighted monadic second order logic, fragments of it, and extensions to other structures: trees, graphs, infinite structures[2]. However, such descriptive languages are still difficult to use in practice because of their heavy syntax, or the high complexity of their verification algorithms. Our objective is to **develop appealing specification languages** for quantitative properties, with expressiveness closer to the targeted applications (such as databases with XPath, or reasoning about programs for PDL), a lighter syntax to ease practical use, and nice algorithmic properties (evaluation, equivalence test) based on automata-theoretic approaches.

Documents carrying data. *Automata and logics over data words and trees* have received much attention in recent years[3]. One motivation to consider data objects is the semi-structured data format XML, where attributes are modeled as data values. Many other areas require to reason about values in a similar vein. For instance, in program verification, data comes into play when we reason about programs with dynamic data structures, or about parametrized concurrent programs. The quest for a robust theory of regularity for languages of data words and trees is still ongoing, as most of the classical tools and constructions cannot be lifted

[1] Colcombet: Regular cost functions I: logic and algebra over words, Logical Methods Comput. Science, 2013.
[2] Bollig, Gastin, Monmege, Zeitoun: Pebble weighted automata and logics. ACM Trans. Comp. Log. 2014.
[3] Bojanczyk, Muscholl, Schwentick, Segoufin: Two-variable logic on data trees and XML reasoning. JACM 2009.

easily here. In particular, we face a difficult trade-off between expressiveness of the models and complexity of the fundamental problems (*e.g.*, evaluation and equivalence). A first goal in the project is to advance the theory of automata for data languages by investigating novel **mechanisms for processing data objects**, such as fixpoint computations, restricted forms of non-determinism, two-way heads. This will result in a better understanding of the structural properties that can ensure decidability (with realistic complexities) for the basic algorithmic questions. The study will also give insights into **designing natural specification languages** based on logics that match the expressive power of the automaton models.

Input-output devices: transducers. The role of automata as an algorithmic tool is even more striking when they are used as computational devices, rather than acceptors. *Transducers* extend automata with outputs, transforming objects from a given domain into objects from another. Nowadays, transducers become increasingly important for stream processing in data management. Streaming algorithms are often based on automata for query answering and transducers for reactions on stream events selected by a query.

We plan to investigate these operational models in the perspective of resource usage, resources being computational power or memory consumption. Recent research motivated by analysis of data streams focused on the theory of *two-way* transducers, where the input can be processed in multiple passes. A two-way transducer is easier to design, but less efficient than a one-way transducer. It is decidable[4] whether a two-way transducer is equivalent to a one-way transducer, but the algorithm has non-elementary complexity. A recent result[5] shows how to solve this question for sweeping transducers in exponential space. A short-term goal is to propose an algorithm with elementary complexity for **arbitrary** two-way transducers.

The two-way model of transducers has a characterization in terms of monadic second-order logic and an equivalent one-way model with additional buffer-like memory: *streaming transducers*. There is a precise link between the number of buffers (memory consumption) and the number of passes of the two-way transducer (computational power). We are interested here in questions related to memory consumption: we aim at deciding how much **additional memory** is needed for a streaming transducer, an open question related to *minimization*.

Regarding algebra, the class of two-way transducers corresponding to first-order logic, a fundamental restriction of monadic second-order logic mentioned above, has been recently characterized by algebraic means[6]. We aim at **generalizing this type of characterization**, for capturing the expressive power of other relevant logical fragments of first-order logic.

Regarding the declarative aspect, monadic second order logic is of course very expressive, but its high complexity makes it untractable for model-checking, when applications to automatic verification of transformations are targeted: suitable logics for effective verification are still missing. One of the goals of this project is to **design convenient and efficient logics** to express properties of input-output transformations.

Finally, transducers over more complex objects such as *nested words* and *hyper-streams* are major areas of interest in our project. Nested words represent trees as data streams, and visibly pushdown transducers (VPT) provide here online tree-to-tree transformations. Some funda-

[4]Filiot, Gauwin, Reynier, Servais: From two-way to one-way finite state transducers. LICS 2013.
[5]Baschenis, Gauwin, Muscholl, Puppis: One-way definability of sweeping transducers. FSTTCS 2015.
[6]Carton, Dartois: Aperiodic two-way transducers and FO-transductions. CSL 2015.

mental questions about VPTs are still open and challenging[7]: for instance, deciding whether a functional VPT can be turned into a deterministic one, *i.e.*, whether such a transformation can be achieved by a **simpler machine**. Studying richer models of transducers that can process multiple input streams (hyper-streams), is an important task in distributed data management, such as cloud databases or distributed workflow. For these transducers we want to **identify streamable subclasses**, and study static analysis problems such as type checking.

Transducers and quantitative theories benefit of transfers from one to the other. First, advances on transducers have been transferred to quantitative models, as demonstrated, *e.g.*, by Alur: cost register automata have been adapted from streaming string transducers. Conversely, transducers can be considered as a special case of weighted automata. In particular, logical characterizations via weighted logics may be lifted to the setting of transducers.

Separation problems in the classical model. Logic is the main tool to specify properties in automated verification or to query databases. In databases, query optimization amounts to rewriting a query in a "simpler" logic, *e.g.*, a logic with lower evaluation complexity. An example is rewriting a recursive query into a non-recursive one. When such a rewriting fails, an alternate solution is to seek for an over-approximation of the query that can be expressed by the simpler logic: we consider a positive query P and a negative one N with $P \cap N = \emptyset$, and we want to compute a query Q in a "simpler" logic such that $P \subseteq Q$ and $Q \cap N = \emptyset$. This kind of problem, known as *separation*, is a variant of interpolation. It occurs in various settings. For instance, the XML streamability problem asks for a finite-state interpolation between two given schemas.

Surprisingly, the separation problem turned out to be a powerful tool for attacking a well-known open problem in logic: the decidability of the quantifier alternation hierarchy of first-order logic over words. Here, we want to determine how many quantifier alternations are necessary for expressing a first-order property. So the question amounts to rewriting a first-order property in the "simplest possible" fragment of the logic, *i.e.*, with as few as possible quantifier alternations.

There are many longstanding open questions in this area. Recently, we achieved a breakthrough[8,9] for this problem by solving separation for some levels of the hierarchy, and lifting these solutions to characterize higher levels. This approach lead us to introduce a *strengthening of the separation problem*. Solving the decidability for each level of the hierarchy is a long-term goal that requires to **define a generic framework**, in order to formulate further generalizations of separation, and to understand the relationships between these problems. This is the goal of this part of the project. Such a generic framework is also a prerequisite to simplify existing results on words and trees, with the aim of capturing new classes.

[7] Filiot, Raskin, Reynier, Servais, Talbot: Properties of visibly pushdown transducers. MFCS 2010.
[8] Place, Zeitoun: Going higher in the First-order quantifier alternation hierarchy on words. ICALP 2014.
[9] Place: Separating Regular Languages with Two Quantifier Alternations. LICS 2015.