Statement of scientific achievements

Szymon Toruńczyk

April 29, 2019

Degrees

2011, June Ph.D. Computer Science, University of Warsaw. Title: *Languages of Profinite words and the limitedness problem* (Automata Theory). Supervisor: prof. Mikołaj Bojańczyk

2006, May M.Sc. Mathematics, University of Warsaw. Title: *$SO(2)$-characteristic geometries on low dimensional manifolds* (Differential Geometry). Supervisor: prof. Marcin Bobieński

2004, May B.S. Computer Science, University of Warsaw.

Professional career

02.2010-present (with breaks for research visits) Assistant professor, Department of Informatics, University of Warsaw

10.2016-11.2016 Research position, TU Dresden, Institute of Algebra, Dresden

10.2015-2.2016 Research position, UPC, Department of Computer Science, Barcelona

01.2011-01.2012 Postdoctoral researcher, ENS de Cachan, Laboratoire Spécification et Vérification

Scientific achievements

Title

Computation in infinite structures

Publications included in the scientific achievments

[TUR] Mikołaj Bojańczyk, Bartek Klin, Slawomir Lasota, and Szymon Toruńczyk. "Turing Machines with Atoms". In: *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013*.

[AMAL] Mikołaj Bojańczyk, Luc Segoufin, and Szymon Toruńczyk. "Verification of database-driven systems via amalgamation". In: *32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2013*.

[TRAC] Mikolaj Bojańczyk and Szymon Toruńczyk. "On computability and tractability for infinite sets". In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018*.

[LOC] Bartek Klin, Eryk Kopczyński, Joanna Ochremiak, and Szymon Toruńczyk. "Locally Finite Constraint Satisfaction Problems". In: *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015*.

[ALF] Bartek Klin, Sławomir Lasota, Joanna Ochremiak, and Szymon Toruńczyk. "Turing machines with atoms, constraint satisfaction problems, and descriptive complexity". In: *Procs. of CSL-LICS'14*.

[LOIS] Eryk Kopczyński and Szymon Toruńczyk. "LOIS: syntax and semantics". In: *44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017*.

[REG] Luc Segoufin and Szymon Toruńczyk. "Automata based verification over linearly ordered data domains". In: *28th International Symposium on Theoretical Aspects of Computer Science, STACS 2011*.

Computation in infinite structures

Description of the results

1 Introduction

Modelling very large, finite systems as infinite ones is a simple, yet powerful method used in all areas of mathematics, including computer science. For example, the working tape of a Turing machine is infinite, the set of names of program variables is modelled as an infinite set, and the set of possible values taken by a variable is often modelled as an infinite domain. Of course the underlying set alone is rather irrelevant – what matters for the studied problem is the structure that this set carries. And so, the structure of a working tape is captured by the successor function of the naturals; the relevant structure for names of program variables is equality of names; when modeling reals, the relevant structure might be the order, addition and multiplication. All the above are examples of logical structures: $(\mathbb{N}, +1)$, $(\mathbb{N}, =)$, $(\mathbb{R}, \leqslant, +, \times)$.

It is thus clear that infinite logical structures can be useful to model various scenarios arising in computer science. However, it is not obvious how to algorithmically perform computations on such infinite models. In practice, frequently infinite systems are approximated by finite ones, in order to facilitate such computations.

We argue that certain infinite structures and systems can be directly processed and manipulated by computers and algorithms. This thesis contributes to the foundations of a uniform approach to such computations, in the framework of *sets with atoms*.

Running example: register automata. As an example of infinite state systems studied in computer science, consider *register automata*, introduced by Kaminsky and Franzez [15]. A register automaton is furbished with a finite set of control states and a finite set of registers, capable of storing data values from some fixed infinite set D. A register automaton then processes a given input sequence of data values in a sequential fashion. Basing on its current control state and equalities between the values in the registers and the currently processed data value, the automaton decides to move to another control state, and whether to place the currently processed data value in one of the registers, erasing its previous content. For example, one could construct a register automaton with two registers which accepts those sequences of data values such that the first two data values do not reappear later in the sequence. Kaminsky and Francez observed that the reachability problem – does a given register automaton accept some word? – is decidable.

A related model, *rational relational automata*, introduced by Čerāns [**Cer94**], allows the automaton to store rational numbers in its registers, and in each step, to compare the values of the registers with respect to the linear order on $\mathbb{Q}$. Such an automaton can for example accept sequences of rationals which are increasing. Čerāns also considered *integer relational automata*, defined similarly, but storing integers instead of rationals. For both models, the reachability problem is decidable. More generally, for a logical structure Atoms, we may consider Atoms-register automata, where the transition relation is defined by a quantifier-free formula in the signature of Atoms. And so, $(D, =)$-register automata are the automata of Kaminsky and Francez, $(\mathbb{Q}, \leqslant)$-register automata are rational relational automata, whereas for Atoms $= (\mathbb{Z}, \leqslant)$ we get integer relational automata. As another example, $(\mathbb{Z}, +1)$-register automata, where $+1$ is the successor function, can test whether the value in one register is one larger than the value in another register. This model is essentially equivalent to *counter machines* of Minsky – and is known to have undecidable reachability.

We will provide a uniform explanation to the fact that the above models, apart from the last one have decidable reachability. In each case, the algorithm deciding reachability is described by the same pseudocode, presented in Figure 1. The configuration graph of an Atoms-register automaton is the (usually, infinite) graph with vertex set $V = Q \times \text{Atoms}^k$, where Q is the finite control space and k is the number of registers, and edge set E consisting of edges (p,q) such that the automaton can transit from configuration p to configuration q following some input letter $a \in \text{Atoms}$. I and F denote the sets of initial and accepting configurations.

Algorithm 1 Algorithm testing reachability in directed graphs

```
function isReachable($V, E, I, F$)    ▷ determine $I$-to-$F$ reachability in the directed graph $(V, E)$
                                      ▷ input: directed graph $(V, E)$ and sets $I, E \subseteq V$
                                      ▷ output: is there a directed path from $I$ to $F$?
    $R := I$                          ▷ vertices reached in current step
    $P := \emptyset$                  ▷ vertices reached in previous step
    while $(R \neq P)$ do
        $P := R$
        for $v \in R$ do
            for $w \in Q$ do
                if $(v, w) \in E$ then
                    insert$(R, w)$    ▷ add the value $w$ to the set $R$
    return $(R \cap F = \emptyset)$
```

Observe that the presented algorithm is just the standard reachability algorithm used for finite graphs. However here, all the variables represent possibly infinite sets. In particular, the algorithm constructs infinite sets, and iterates over them, using the **for** loop. We will see how algorithms like this can be executed, and how to prove their termination, using tools from model theory. We will also see how other classical algorithms lift to the case of infinite inputs, and how their computational complexity can be analysed.

1.1 Sets with atoms [LOIS]

I now introduce the framework of sets with atoms, to which this thesis contributes. The idea is to consider certain infinite sets that can be by described by finite means, and which form a well-behaved set-theoretic universe. This idea was introduced in an earlier paper [**BKL**] and was gradually developed by Bojańczyk, Klin, Kopczyński, Lasota, myself, and others. Initially, the framework of sets with atoms was based on *orbit-finite sets*. Later, in a paper with Kopczyński [LOIS], we proposed to define sets with atoms using the notion of hereditarily definable sets, vastly generalizing the previous framework, and defined below. Our paper also introduces a programming language manipulating hereditarily definable sets, which is described later below.

Hereditarily definable sets with atoms. Fix an infinite logical structure Atoms as the underlying structure of *atoms*. For example, it could be $(D, =)$, where D is a countable infinite set, or $(\mathbb{Q}, \leqslant)$. We call the former structure *the pure set* and the latter structure *the dense linear order*. These two structures will be used as running examples illustrating most of the results in this dissertation. Let us focus on Atoms being the pure set for the moment.

A *hereditarily definable set* (over Atoms) is a set defined by an expression such as:

$$\{\{\{d,e\} \mid e \in \text{Atoms}, e \neq d\} \cup \{\{f,g\} \mid f,g \in \text{Atoms}, (f=d) \vee (g=d)\} \mid d \in \text{Atoms}\}.$$

This expression defines a set of sets of subsets of atoms, according to the well-established *set-builder notation* – in this case, the same set as defined by the simpler expression $\{\{\{d,e\} \mid e \in \text{Atoms}\} \mid d \in \text{Atoms}\}$. In general, we allow expressions as above, where we can take finite unions of set-builder expressions, which can be nested arbitrarily, and in which the bound variables are required to range over the atoms, and each guard (such as $e \neq d$ above) is a first-order formula in the language of Atoms. A *hereditarily definable set* is a set defined by an expression as above. More precisely, such an expression may have some free variables (as e.g. the expression $\{d,e\}$ or d), and in order specify a hereditarily definable set completely, a valuation of the free variables in the domain of Atoms is required. In this case, the atoms in the range of the given valuation are called *parameters*, and we say that the resulting hereditarily definable set is defined with parameters. However, unless explicitly stated, we will consider hereditarily definable sets defined without parameters (i.e., those defined by expressions without free variables).

Hereditarily definable sets form a rather well-behaved set-theoretic universe, with many of the usual closure properties of sets. For example, they are closed under unions (by definition), intersections, set-differences, Cartesian products[1], and others. Due to this robustness, we may consider standard set-theoretic notions, such as equivalence relations, functions, etc. in the context of hereditarily definable sets. For example, the two projections from the set $X = \{(a,b) \mid a,b \in \text{Atoms}, a \neq b\} \subseteq \text{Atoms} \times \text{Atoms}$ of ordered pairs of distinct atoms to Atoms are hereditarily definable functions: the graph of the first projection is defined by the expression $\{((a,b),a) \mid a,b \in \text{Atoms}, a \neq b\}$, and likewise for the second projection. What makes sets with atoms quite different from usual set theory is that the powerset axiom fails, and also, the axiom of choice fails[2].

Note that (hereditarily) *finite* sets are a special case of hereditarily definable sets, e.g. the von Neumann representation of the number two, the set $\{\emptyset, \{\emptyset\}\}$, is hereditarily definable.

Similarly, for other atoms, such as $(\mathbb{Q}, \leqslant)$, we can use the structure available in Atoms to construct hereditarily definable sets, such as e.g. the set of all closed intervals in $\mathbb{Q}$,

$$\{\{c \mid c \in \text{Atoms}, a \leqslant c \leqslant b\} \mid a,b \in \text{Atoms}, a \leqslant b\}.$$

Hereditarily definable structures. Within the universe of sets with atoms, we may consider many of the notions studied in finite combinatorics and computer science. For example, *hereditarily definable automata* are defined exactly as finite nondeterministic automata, but all the components (the alphabet A, the state space Q, initial states $I \subseteq Q$, accepting states $F \subseteq Q$, set of transitions $\delta \subseteq Q \times A \times Q$) are allowed to be hereditarily definable sets, rather than finite sets. We then define runs of automata and accepted words over the input alphabet, as usual. Note that Atoms-register automata considered in the introduction are their special case, where the input alphabet is Atoms and the state space is the Cartesian product of some finite set Q and Atoms^k, where k is the number of registers, and the transition relation is defined by a quantifier-free formula with $2k+1$ free variables[3]. Note, however, that for some structure of atoms, hereditarily definable automata are more

[1] a pair (x,y) is represented by its Kuratowski encoding $\{\{x\},\{x,y\}\}$

[2] for example, the projections described have no hereditarily definable right inverses $\text{Atoms} \to X$

[3] k free variables for the register values before the transition, k for the values after the transition, and one free variable for the input letter

powerful than Atoms-register automata. For example, if Atoms is $(\mathbb{N}, \leqslant)$, then hereditarily definable automata may simulate the successor function on $\mathbb{N}$ using first-order formulas, and therefore have undecidable reachability, whereas Atoms-register automata have decidable reachability, owing to the fact that their definitions only involve quantifier-free formulas.

As we will see, not only infinite-state systems can be elegantly described in the framework of sets with atoms, but more importantly, hereditarily definable automata enjoy a theory naturally generalizing that of finite automata.

Similarly, we may consider hereditarily definable context-free grammars, pushdown automata, Turing machines, graphs, etc. Furthermore, as those structures can be represented in a finite way by set-builder expressions, we can consider various standard computational problems for these objects, such as automata reachability, graph 3-colorability, etc. It turns out that these objects and their computational problems capture some problems previously studied in computer science, and that they can be solved using methods from areas of mathematics pertinent to the study of infinite structures, e.g., model theory or topology.

1.2 Programming with atoms [LOIS]

Sets with atoms, although ostensibly infinite, are presentable by finite means, and most standard set-theoretic constructions are computable on such representations. For example, all the operations described above, such as union, intersection, difference, projections, etc. are effectively computable on the level of expressions defining them. Finally, for many structures of Atoms we may effectively determine equality of two given hereditarily definable sets. This is true for atoms being $(D, =)$, $(\mathbb{Q}, \leqslant)$, $(\mathbb{R}, +, \times, \leqslant)$, or, in fact, any structure Atoms with decidable first-order theory. The above properties where first observed in the paper [LOIS].

It is therefore a natural idea to design a programming language for algorithmic manipulation of sets with atoms. Ideally, the language should be defined so that details of the representation are transparent to the programmer, who should be able to operate under the convenient intuition of computing, searching and iterating over infinite structures and their elements.

This idea was first proposed in [4], where a functional programming language called Nλ was described, and then in the imperative fashion in [21]. In each case, this was carried out in the framwork of orbit-finite sets, which is less general than the framework of hereditarily definable sets, and also, their proposed representation led to an impractical implementation.

The main contribution of the paper [LOIS] is the development of a working language, called LOIS, in the imperative style, basing on the framework of hereditarily definable sets.

As an example, the pseudocode of the function isReachable in Figure 1 implements the reachability algorithm for hereditarily definable graphs, and is directly implementable in LOIS with minor syntactic modifications. The semantics of LOIS is defined in a natural way – intuitively, the (possibly infinitely many) branches of the **for** loop are executed in parallel, and the values of a variable resulting from parallel branches are aggregated using set union (this semantics is based on, and generalizes, my previous paper [21]).

Remarkably, the code implementing the reachability function is exactly the standard code used in the case of finite graphs. Yet, it can be evaluated whenever the inputs are sets which are hereditarily definable over atoms with decidable first-order theory. For some atoms, such as the pure set or the dense linear order, it can be proved to terminate in a finite number of steps, by a general argument, using a result from model theory, which is discussed below.

Technically, LOIS is implemented as a C++ library, so the programmer has a fully-fledged programming language at her disposal, with additional features for modeling hereditarily definable sets. Internally, these are represented by the expressions defining them. In the implementation of some programming primitives, notably in the procedure for checking emptiness of hereditarily definable sets, the LOIS interpreter needs to test whether a first-order sentence is satisfiable in the underlying structure of atoms. For this, the inner mechanism of the LOIS interpreter regularly invokes an SMT solver (in our implementation, we use [33] both an internal solver, as well as external solvers such as Microsoft's Z3). All this is, of course, completely hidden from the programmer, who does not need to know how hereditarily definable sets are represented. The implementation of LOIS is mature enough to be practically useful, as demonstrated in [33]. Furthermore, our paper has inspired a renewed, working implementation of the functional programming language Nλ [14], using the same ideas as LOIS for internal representations.

Termination. In [LOIS] we prove the following result concerning programs without iteration, i.e., recursion nor **while** loops, but possibly with **for** loops.

Theorem 1 (Theorem 10 in [LOIS]). *Assume that* Atoms *has decidable first-order theory. Then any LOIS program without iteration terminates and its outcome can be computed, assuming the underlying structure* Atoms *has decidable first-order theory.*

As an example, consider the pseudocode in Figure 2, computing the set of triples (a, b, c) of reals such that the polynomial $ax^2 + bx + c$ has a root. We assume the program is evaluated in the underlying structure of atoms Atoms $= (\mathbb{R}, +, \times)$, providing an existing solver for its theory.

Algorithm 2 An algorithm computing the set of solvable quadratic polynomials

```
$S := \emptyset$
for $a \in \mathbb{R}$ do
    for $b \in \mathbb{R}$ do
        for $c \in \mathbb{R}$ do
            for $x \in \mathbb{R}$ do
                if $ax^2 + bx + c = 0$ then
                    $insert(S, (a, b, c))$
return $S$
```

The above LOIS program terminates by Theorem 1. The resulting value of the variable S is $\{(a, b, c) \mid a, b, c \in \mathbb{R}, b^2 \geqslant 4ac\}$.

Thanks to the above theorem, the programmer may focus – as usual – on proving the correctness of the program and termination of iteration, without worrying about the internal representation of sets. Below we give an example of such a termination argument, based on a notion from model theory.

ω-categoricity. We recall the following, fundamental notion from model theory. A countable logical structure Atoms is *ω-categorical* if any countable structure satisfying the same first-order sentences as Atoms is isomorphic to Atoms. Examples include the pure set, which can be axiomatized by a sequence of sentences stating that there are at least n distinct elements, for $n = 1, 2, 3, \ldots$, and the dense linear order, axiomatized by sentences expressing that it is a dense linear order without endpoints.

A fundamental result from model theory, due to Ryll-Nardzewski, Engeler and Svenonius [12], states that a structure Atoms is ω-categorical if, and only if, for every number $n \in \mathbb{N}$, the action of the automorphism group Aut(Atoms) of Atoms induces only finitely many orbits on the set Atoms^n of n-tuples of atoms. For example, the dense linear order $(\mathbb{Q}, \leqslant)$ satisfies this property for

$n = 2$: its automorphisms are monotone bijections of $\mathbb{Q}$, and there are three orbits of $\mathbb{Q}^2$ under the action of this group, defined by the formulas $x < y$, $x = y$, and $x > y$.

Note that this characterization lifts to hereditarily definable sets, as follows. If X is such a set[4], then Aut(Atoms) acts on X in a natural way – an automorphism π can be applied to X by applying it recursively to the elements of X. If Atoms is ω-categorical, then this action has finitely many orbits[5]. In particular, X has only finitely many subsets that are invariant under the action of Aut(Atoms), as each such subset is a union of orbits of X.

This yields termination of the above algorithm for graph reachability, as follows. Consider the sequence $R_0 \subseteq R_1 \subseteq R_2 \subseteq \ldots V$, where R_i is the set of vertices reachable from I in at most n steps. By an easy induction on n, it is immediate that each sets R_n is invariant under the action of Aut(Atoms). As V has only finitely many invariant subsets, the sequence $R_0 \subseteq R_1 \subseteq R_2 \subseteq \ldots$ must stabilize after finitely many steps, proving termination of the algorithm.

This gives the following theorem, proved in [LOIS], and generalizing previous results [5].

Theorem 2 (Theorem 13 in [LOIS]). *Reachability is decidable for hereditarily definable automata over a fixed ω-categorical structure* Atoms *with decidable first-order theory.*

Running example: register automata. The notion of ω-categoricity yields a single, generic algorithm, which solves the reachability problem for various models of Atoms-register automata, such as those of Kaminsky and Francez (taking the pure set as atoms), of Čerāns (taking the dense linear order as atoms), and in fact Atoms-register automata, for any ω-categorical structure Atoms with decidable theory. Furthermore, many other standard fixpoint algorithms studied in computer science, such as automata minimisation, pushdown automata emptiness, grammar-to-pushdown conversion, etc., can be directly implemented in LOIS, yielding a program which works for infinite hereditarily definable objects, and terminates by virtue of ω-categoricity of the chosen underlying structure of atoms. Note, however, that the structure Atoms $= (\mathbb{Z}, \leqslant)$ is not ω-categorical. As we will see, reachability for Atoms-register automata is still decidable for this, and other atoms Atoms. The argument will use further tools from model theory, and will yield applications in static verification of database-driven systems.

2 Database-driven systems [AMAL, REG]

The papers [AMAL, REG] study an extension of register automata, called *database-driven systems*. This has applications in static verification of database systems, such as web services, web applications, or data-centric business processes, see [9] for an overview.

We model databases as finite logical structures. Like register automata, database-driven systems are equipped with finitely many registers. This time, the registers can store elements of a database, and their behavior is described by finitely many transition rules controlling their workflow. Each such rule may be based on the result of quantifier-free queries to the database. The database is not fixed and may vary from run to run. It is however restricted to range over a fixed class of databases typically specified using a schema (or signature) and possibly some other constraints. Moreover the system has only read access to the database and the database does not change during a run.

A more formal definition follows. First note that the syntax of a Atoms-register automaton only depends on the signature Σ of the structure Atoms, and not on its domain. Hence, we may consider

[4]for simplicity, we provide it for the sets defined without parameters, i.e., by a set-builder expressions without free variables

[5]Thus, the framework of hereditarily definable sets correspond to so-called *orbit-finite sets* [4], in the case when Atoms is ω-categorical.

Σ-register automata, for a fixed signature Σ. For a class of finite Σ-structures $\mathscr{C}$, a *$\mathscr{C}$-database-driven system* is a Σ-automaton $\mathcal{A}$. In the reachability problem, the class $\mathscr{C}$ of finite databases is fixed. Given on input a $\mathscr{C}$-database-driven system $\mathcal{A}$, the question is whether there is some database $\mathbb{D}$ in $\mathscr{C}$ such that $\mathcal{A}$, treated as a $\mathbb{D}$-register automaton, has an accepting run. In this case, we say that $\mathcal{A}$ has an accepting run *driven by* the database $\mathbb{D} \in \mathscr{C}$.

Example 1. We consider a database-driven system with two registers, which can store nodes of a finite directed (unkown) graph G, modelled as a relational structure $G = (V, E)$. Here, G plays the role of the underlying database. The goal of the system is to nondeterministically test whether there is a directed cycle in G of odd length.

The constructed system has two registers. The first register stores an initial, nondeterministically chosen vertex of the graph, and its value will remain fixed during a run. The second register will trace the vertices along the cycle. The system also has two states q_0, q_1, tracking the parity of the number of performed steps. The transitions of the system are as follows. In a configuration with state q_i and register values (a, b), the system may move to the configuration with state q_{1-i} and register values (a', b'), if $a = a'$ and $E(b, b')$ holds (implicitly, in the given graph G). Initial configurations are those with state q_0 and register values (a, b) such that $a = b$, and accepting configurations are those with state q_1 and register values (a, b) such that $a = b$.

All the above conditions relating the register values before and after a transition can be expressed by a quantifier-free formula using the edge relation and equality. Hence, they define a Σ-register automaton, where Σ is the signature consisting of E. We view this Σ-register automaton as a $\mathscr{C}$-database-driven system, where $\mathscr{C}$ is the class of all finite directed graphs.

The reachability question for the above system asks: is there some finite directed graph G for which the above system has an accepting run? In this case, the answer is positive, as witnessed by a directed cycle of length 3. $\lrcorner$

In the papers [AMAL, REG], we develop general techniques for testing reachability of such database-driven systems. These techniques encompass examples concerning relational databases (modelled as relational structures), and also XML databases (modelled as trees with certain relations and functions) and – in general – any amalgamation class, as explained below.

The starting point is the following simple observation, stemming from the fact that we only consider finite accepting runs, and that transition relations in Σ-register automata are specified by quantifier-free formulas. For a class of structures $\mathscr{C}$, define its *age* as the class of all finite (induced) substructures of structures in $\mathscr{C}$.

Lemma 1. *If $\mathscr{C}$ and $\mathscr{D}$ are two classes of Σ-structrues with equal ages, then a Σ-register automaton $\mathcal{A}$ has an accepting run driven by some $\mathbb{C} \in \mathscr{C}$ if and only if it has an accepting run driven by some $\mathbb{D} \in \mathscr{D}$.*

In particular, applying the above to the case when $\mathscr{D}$ is a class consisting of a single, infinite structure $\mathbb{D}$, whose age is equal to $\mathscr{C}$, we get the following.

Corollary 1. *If $\mathscr{C}$ is the age of some structure $\mathbb{C}$, then the reachability problem for $\mathscr{C}$-database driven systems reduces to the reachability problem for $\mathbb{C}$-register automata.*

The key point is that for some classes $\mathscr{C}$ of finite structures, there is a certain infinite structure $\mathbb{C}$ whose age is $\mathscr{C}$ and which is furthermore ω-categorical. This is for instance the case when $\mathscr{C}$ is a class of finite relational structures which is closed under *amalgamation*, a combinatorial property of a class of structures $\mathscr{C}$ studied in model theory [12], which allows to "glue" any two given structures in $\mathscr{C}$ along a given common substructure. Examples of amalgamation classes include:

- the class of finite sets (viewed as structures with the equality symbol only),

- the class of finite linear orders,
- the class of finite partial orders,
- the class of finite directed graphs.

Each amalgamation class is encompassed by a certain infinite structure with many automorphisms, as made precise by the following notion. An countable structure $\mathbb{C}$ is *homogeneous* if every partial isomorphism between two finite induced substructures of $\mathbb{C}$ extends to an automorphism of $\mathbb{C}$. For example, the pure set and the dense linear order are homogeneous. Another example is the *infinite random directed graph*, constructed as follows. Take a countable set as vertices V, and for each pair (v, w) of vertices, independently and at random with probability $1/2$ create an edge from v to w. With probability 1, any two outcomes of the above process will yield the same – up to isomorphism – directed graph, which can be characterized as the unique homogeneous directed graph containing all finite directed graphs as induced subgraphs. More generally, the following fundamental result of Fraïssé(cf. [12]) states that any amalgamation class corresponds to a unique homogeneous structure in a similar way:

Theorem 3. *Fix a finite relational signature Σ. Let $\mathscr{C}$ be a class of finite Σ-structures which is closed under amalgamation and under taking substructures. Then there is a unique, up to isomorphism, homogeneous structure $\mathbb{C}$ whose age is $\mathscr{C}$.*

Note that it follows from the result of Ryll-Nardzewski, Engeler, and Svenonius that every homogeneous structure $\mathbb{C}$ over a finite relational signature is ω-categorical[6].

To solve the reachability problem for database-driven systems, driven by a class of databases $\mathscr{C}$, in the paper [AMAL], we propose the following general approach. Suppose that $\mathscr{C}$ is a class satisfying the assumptions[7] of Theorem 3, and let $\mathbb{C}$ be the homogeneous structure obtained from the theorem. Then $\mathbb{C}$ is ω-categorical and has decidable first-order theory, assuming $\mathscr{C}$ has a decidable membership problem. By Corollary 1, the reachability problem for $\mathscr{C}$-database driven systems reduces to the reachability problem for $\mathbb{C}$-register automata, which is decidable by Theorem 2. This proves the following result, which is formulated in greater generality, and with precise complexity bounds in [AMAL].

Theorem 4 (Theorem 5 in [AMAL])**.** *Let $\mathscr{C}$ be a decidable class of finite relational structures which is closed under amalgamation and under induced substructures. Then the reachability problem for $\mathscr{C}$-database-driven systems is decidable.*

In [AMAL], we then proceed to instantiating the above result to specific classes $\mathscr{C}$ of interest. For example, taking $\mathscr{C}$ to be the class of all finite structures over a fixed finite signature, we recover a result first observed in [10] (see also [REG]).

In the main result of the paper, we consider classes of finite trees, motivated by the study of XML databases. Given a regular language L of finite trees, we define a class $\mathscr{C}$ of databases obtained from each tree $t \in L$, by viewing it as a logical structure in a suitable signature (which includes relation and function symbols, e.g. for the least common ancestor). The main combinatorial task is to prove that each such class $\mathscr{C}$ can be modified to yield a class which is closed under amalgamation. This yields the following result (see [AMAL] for a statement involving precise complexity bounds):

Theorem 5 (Theorem 3 in [AMAL])**.** *The following problem is decidable:*

[6]by homogeneity, two n-tuples of elements of $\mathbb{C}$ with the same quantifier-free type are in the same orbit, hence there are finitely many orbits of n-tuples

[7]some weaker conditions are also considered, allowing the signature to contain function symbols

Input: *A tree automaton $\mathcal{T}$, and a $\mathscr{C}$-database-driven system $\mathcal{A}$, where $\mathscr{C}$ is the class of trees accepted by $\mathcal{T}$*
Decide: *does $\mathcal{A}$ have an accepting run driven by some database in $\mathscr{C}$?*

Further results show how our general method can be applied to databases which store elements of some fixed, infinite structure Atoms, such as the dense linear order, or in fact, any relational structure Atoms whose age is closed under amalgamation, such as $(\mathbb{Z}, \leqslant)$. In particular, we generalize the above results to the case when the underlying databases stores rational numbers or integers, and the database-driven system may additionally compare those values with respect to inequality.

The paper [REG]. The paper [REG] focusses on the problem of finding *infinite* accepting runs of database-driven systems, where a run is accepting if it visits an accepting state infinitely many times. We show a general method of reducing this problem to the problem of finding finite runs, which was considered above. This is then applied to solve verification problems, where the infinite runs of a system are tested against a given formula in an extension of LTL (linear temporal logic) allowing to speak about the current configuration of the database system (which includes the state, the register values, and the underlying database). In particular, we solve a problem left open in [10], regarding database-driven systems which store integers in their nodes, and are allowed to compare them for inequalities.

Theorem 6 (Theorem 6.3 and Theorem 5.1 in [REG])**.** *Given a database-driven system $\mathcal{A}$ which can access finite relational databases whose nodes are integers which can be compared with respect to $<$, and a sentence φ of LTL($<$), it is decidable in PSPACE whether the system $\mathcal{A}$ has an infinite run satisfying φ, driven by some database $\mathbb{D}$.*

3 Towards complexity theory [TRAC, LOC]

The previous sections focussed on the reachability problem for hereditarily definable graphs and automata as the guiding example of a computation problem studied in the context of sets with atoms.

As mentioned, the presented techniques generalize to many other computational problems which can be solved by fixed-point algorithms, when considering structures which are hereditarily definable over ω-categorical atoms. In this section, based on the papers [TRAC, LOC], I give an overview of the computational complexity of various classical problems lifted to the setting of sets with atoms, such as the problem of graph colorability for hereditarily definable graphs. I will also present various applications of the obtained results.

Some classical problems. Consider the following classical computational problems, where the inputs are finite structures.

1. (Directed) graph reachability: given a graph G and two vertices s, t, decide if t is reachable from s. This problem is interreducible[8] with the automata reachability problem.
2. Deterministic automata minimisation: given a deterministic automaton, compute its corresponding minimal automaton.
3. Graph 3-colorability: is a given graph 3-colorable?

[8]via first-order reductions

4. Horn-SAT: is a given set of Horn-clauses satisfiable? This problem is interreducible with context-free grammar emptiness.
5. 2CNF-SAT.
6. 3CNF-SAT.
7. Solvability of systems of linear equations.
8. Graph homomorphism problem: given two graphs G, H, is there a homomorphism from G to H?
9. Graph isomorphism.

The above example problems have many applications in various areas of computer science, and their computational complexity is well understood.

Computational problems with hereditarily definable instances. Each of the problems listed above can be studied in the context of sets with atoms, i.e., when the input structures are hereditarily definable over some fixed atoms.

For example, in graph 3-colorability, a hereditarily definable graph G is given on input, and the question is whether it is 3-colorable. In hereditarily definable systems of linear equations, or sets of clauses, variables are required to form a hereditarily definable set, and likewise the set of equations/clauses. For example, consider the following system of equations over the two-element field, where the atoms are the pure set. Its variables are x_{ab}, for distinct atoms a, b, and equations are

$$x_{ab} + x_{ba} = 1 \qquad \text{for distinct atoms } a, b. \tag{1}$$

Given a hereditarily definable system of linear equations over a finite field such as above, we may ask whether it has a solution. Similarly, we may consider hereditarily definable sets of 3CNF-clauses, and their satisfiability problem. In general, for each of the problems listed above, and for a fixed choice of atoms, we may ask about the computational complexity of the problem. It is not even clear that the above problems are decidable, even when the atoms are the pure set. The argument from the previous section works for problems 1,2,4,5 whenever the atoms are ω-categorical (e.g. the pure set or a dense linear order): the standard fixpoint algorithms for solving the classical versions of the problem also work in the case of hereditarily definable sets, and terminate by virtue of ω-categoricity. In each case, this results in an (optimal) $PSPACE$ or $EXPTIME$ complexity algorithm, depending on the considered problem. This dichotomy in complexity will be now explained in a broader context.

We show how problems 1-8, and others, can be solved for hereditarily definable instances. Decidability of the isomorphism problem for hereditarily definable graphs remains open.

CSPs with finite instances. Problems 1,3,4,5,6,7 all fall into the category of *Constraint Satisfaction Problems* (CSP's), defined below. Fix a relational structure $\mathbb{T}$ over a finite signature, called a *template*. The classical CSP problem over the template $\mathbb{T}$ is the following decision problem:

Problem: CSP($\mathbb{T}$)
Input: a (finite) relational structure $\mathbb{I}$ over the signature of $\mathbb{T}$.
Decide: is there a homomorphism from $\mathbb{I}$ to $\mathbb{T}$?

Here, a *homomorphism* is a mapping h from the domain of $\mathbb{I}$ to the domain of $\mathbb{T}$ which preserves each relation R in the signature, i.e., for each tuple $a_1, \ldots, a_n$ of elements of $\mathbb{I}$, where n is the arity of R, if $R(a_1, \ldots, a_n)$ holds in $\mathbb{I}$, then $R(h(a_1), \ldots, h(a_n))$ holds in $\mathbb{T}$.

A structure $\mathbb{I}$ which is the input to the above problem is called a CSP *instance*. A homomorphism $h\colon \mathbb{I} \to \mathbb{T}$ is called a *solution* to the instance.

As an example, consider systems of linear equations over the two element field, where each equation contains exactly three variables. To express the solvability problem, consider a template $\mathbb{T}$ with domain $\{0, 1\}$, and which is equipped with two ternary relations, denoted R_0 and R_1, defined by the equations $x + y + z = 0$ and $x + y + z = 1$ (modulo 2), respectively. A system of linear equations S as above then defines an instance $\mathbb{I}$, whose domain consists of the variables appearing in the system, and where R_0 consists of those triples (x, y, z) of variables such that $x + y + z = 0$ is an equation in S, and R_1 is defined analogously with 0 replaced by 1. Then, a homomorphism from $\mathbb{I}$ to $\mathbb{T}$ is exactly the same as a solution to the system S. Similarly, a 3CNF-SAT instance can be viewed as a CSP instance over the template $\mathbb{T}'$ whose domain consists of $0, 1$ (viewed as logical values), equipped with eight ternary relations on $\{0, 1\}$, defined by the clauses $x \vee y \vee z$, $\neg x, \vee y, \vee z$, etc.

A wide range of problems studied in computer science can be cast as constraint satisfaction problems, over suitably chosen templates. For example, Horn-SAT, 2CNF-SAT, graph 3-colorability, graph reachability, etc., are all equivalent to CSP problems over appropriate templates.

Clearly, for any finite template $\mathbb{T}$, the problem CSP($\mathbb{T}$) is solvable in NP. However, for some finite templates, the complexity may be much lower. There is a rich and very successful theory which analyses the complexity of a CSP problem, depending on the considered template. One of the most spectacular results in this theory, conjectured by Feder and Vardi [11], and proved recently independently by Bulatov [6] and Zhuk [19], says that for each finite template $\mathbb{T}$, the problem CSP($\mathbb{T}$) is either solvable in polynomial time (such as the template $\mathbb{T}$ for systems of linear equations above), or is otherwise NP-complete (such as the template $\mathbb{T}'$ for 3CNF-SAT). In particular, no CSP template defines a problem of intermediate complexity. Moreover, there is an effective, algebraic characterization of those templates for which the CSP problem is NP-complete, and those, for which it is in P. The rich theory of constraint satisfaction problems also provides tools to analyse the cases when the considered problem is in a lower complexity class (e.g. L or NL-complete) or can be solved by a specific type of algorithm, e.g. by a fixpoint algorithm.

3.1 CSPs with hereditarily definable instances and templates [LOC]

In [LOC], we consider constraint satisfaction problems, in which the instances are allowed to be hereditarily definable over some fixed atoms. As previously, we fix a template $\mathbb{T}$, and consider the problem:

Problem: DEF-CSP($\mathbb{T}$)
Input: a hereditarily definable relational structure $\mathbb{I}$ over the signature of $\mathbb{T}$
Decide: is there a homomorphism from $\mathbb{I}$ to $\mathbb{T}$?

Above, the atoms are implicit in the definition. In the discussion below, we will assume that the atoms are the pure set, or, more generally, the dense linear order (the results in [LOC] also apply to other atoms). Note that even when the atoms are the pure set and when $\mathbb{T}$ is finite, mere decidability of the above problem is not obvious.

In our paper, we focus on the case when the template $\mathbb{T}$ is finite, or more generally, *locally finite*. A template $\mathbb{T}$ is locally finite if it has a possiblye infinite set of relations, which are however required to be finite. The motivation for studying such templates will be demonstrated later below.

We show in [LOC] that for locally finite templates, the problem Def-CSP($\mathbb{T}$) is decidable. In particular, this gives an algorithm solving graph reachability, Horn-SAT, 3-colorability, 2CFN-SAT, 3CNF-SAT, etc., where the instances are hereditarily definable.

Theorem 7 (Theorem 19 in [LOC]). *Suppose that the atoms are the pure set or the dense linear order. Fix a locally finite, hereditarily definable template $\mathbb{T}$. Then Def-CSP($\mathbb{T}$) is decidable.*

Proof. We sketch the proof in the case when $\mathbb{T}$ is finite; the case of locally finite templates is similar. We also consider the case when Atoms is the dense linear order, as the case of the pure set reduces to it[9]. For those atoms, we prove the following key lemma.

Lemma 2. *If a hereditarily definable instance $\mathbb{I}$ has a solution $h\colon \mathbb{I} \to \mathbb{T}$, then it has a solution which is invariant under all automorphisms of $(\mathbb{Q}, \leqslant)$.*

Note that a function $h\colon \mathbb{I} \to \mathbb{T}$ is invariant if and only if it is constant on every orbit of the action of $\mathrm{Aut}(\mathbb{Q}, \leqslant)$ on the domain of $\mathbb{I}$. By the Ryll-Nardzewski theorem, there are finitely many orbits of this action. Therefore, the CSP instance $\mathbb{I}'$ defined as the quotient of $\mathbb{I}$ by the action of $\mathrm{Aut}(\mathbb{Q}, \leqslant)$ is finite, and furthermore, it is computable from $\mathbb{I}$ (in *PSPACE*). By the above lemma, the instance $\mathbb{I}$ has a solution if and only if $\mathbb{I}'$ has a solution. As $\mathbb{I}'$ is finite, the problem of its solvability is clearly decidable.

Observe that Lemma 2 fails if we consider the pure set as atoms, as witnessed by the system of linear equations (1), which has no solution which is invariant under all permutations of the pure set. Indeed, there is only one orbit of variables, and only two invariant assignments – constantly 0 and constantly 1 – which are not solutions to the system (1).

The proof of Lemma 2 employs the following result from topological dynamics, which is a consequence of Ramsey's theorem.

Theorem 8 (Pestov [17]). *Any continuous[10] action of the group $\mathrm{Aut}(\mathbb{Q}, \leqslant)$ on a nonempty compact topological space has a fixpoint.*

Proof of Lemma 2. The set $\mathbb{T}^{\mathbb{I}}$ of all functions from $\mathbb{I}$ to $\mathbb{T}$ forms a topological space, with the topology of pointwise convergence. By Tychonoff's theorem, as $\mathbb{T}$ is finite, this space is compact. The group $\mathrm{Aut}(\mathbb{Q}, \leqslant)$ acts on the domain of $\mathbb{I}$ (since it is a hereditarily definable set) and this action lifts to an action on $\mathbb{T}^{\mathbb{I}}$, which is continuous. The set $\mathrm{Hom}(\mathbb{I}, \mathbb{T}) \subseteq \mathbb{T}^{\mathbb{I}}$ of solutions to the given CSP instance is topologically closed, hence compact. Furthermore, it is invariant under the group action. Hence, the group $\mathrm{Aut}(\mathbb{Q}, \leqslant)$ acts continuously on the compact space $\mathrm{Hom}(\mathbb{I}, \mathbb{T})$. By Pestov's theorem, if this space is nonempty, there is a solution which is invariant under the group action, proving the lemma. ∎

As argued earlier, the lemma gives a reduction of the CSP problem to the finite case, which is decidable (in *NP*). ∎

An analysis of the proof gives precise complexity bounds on Def-CSP($\mathbb{T}$): it is exponentially harder than the classical variant CSP($\mathbb{T}$). Formally, for a complexity class C, the class $Exp(C)$ is defined using padding. As specific examples, $Exp(P) = EXPTIME$, $Exp(L) = PSPACE$, etc.

Our results give matching lower complexity bounds:

[9]Given a CSP instance which is hereditarily definable over the pure set, we may treat the expression defining it as defining an instance over the dense linear order. The resulting instance will be isomorphic to the original one.

[10]the group $\mathrm{Aut}(\mathbb{Q}, \leqslant)$ is equipped with the topology of pointwise convergence: a sequence of functions $f_1, f_2, \ldots \colon \mathbb{Q} \to \mathbb{Q}$ converges to $f\colon \mathbb{Q} \to \mathbb{Q}$ if for every element $v \in \mathbb{Q}$, $f_n(v)$ is equal to $f(v)$ for sufficiently large n.

Theorem 9 (Theorem 22 in [LOC]). *Let $\mathbb{T}$ be a finite template, such that CSP($\mathbb{T}$) is is complete*[11] *for a complexity class C. Then Def-CSP($\mathbb{T}$) is complete for the class $Exp(C)$.*

In particular, for hereditarily definable instances, graph 3-colorolability is $NEXPTIME$-complete, graph reachabilty is $PSPACE$-complete, Horn-SAT is $NEXPTIME$-complete, etc. Moreover, from the result Bulatov and Zhuk, we obtain the following.

Corollary 2. *Fix a finite template $\mathbb{T}$. Then Def-CSP($\mathbb{T}$) is either in $EXPTIME$, or it is $NEXPTIME$-complete.*

CSPs with infinite templates. The theory of constraint satisfaction problems has been successfully applied in the context of finite instances over templates which are infinite and ω-categorical. For example, when $\mathbb{T}$ is $(\mathbb{Q}, <)$, then CSP($\mathbb{T}$) is equivalent to the problem of deciding whether a given directed graph is acyclic. Many, but not all, of the classical results of CSP theory lift to this setting. In particular, the dichotomy conjecture is still open for ω-categorical templates [2].

The paper [LOC] studies the CSP problem over locally finite templates which are definable over a fixed structure of atoms. The motivating question was whether the known characterization of finite templates of *bounded width* [1] can be generalized to locally finite templates which are hereditarily definable over the pure set. Roughly, a template $\mathbb{T}$ has bounded width if CSP($\mathbb{T}$) can be solved by a fixpoint algorithm. We prove a positive answer to this question:

Theorem 10 (Corollary 35 in [LOC]). *It is decidable whether a hereditarily definable, locally finite template has bounded width.*

The proof of Theorem 10 proceeds in two steps: first we lift the existing algebraic characterization of finite templates of bounded with to the case of locally finite templates; next, we use Theorem 7 to determine whether the given locally finite template $\mathbb{T}$ satisfies this algebraic characterization.

The above result is just one example of the type of results that we manage to lift from the case of finite templates to the case of locally finite, hereditarily definable templates.

We note that Theorem 7 is generalized in [32], where we prove decidability of the problem of existence of a homomorphism between two given hereditarily definable structures with finite signatures.

3.2 Parametrized complexity [TRAC]

The complexity bounds described in Theorem 9 show that the hereditarily-definable analogoues of many classical problems have at least $PSPACE$ complexity, including the problem of graph reachability. Even worse – mere equality of two hereditarily definable sets is a $PSPACE$-hard problem for any choice of atoms with at least two elements, by an easy reduction from QBF.

However, as we observe in [TRAC], if the *dimension* of the considered expressions is fixed, the equality problem becomes decidable in polynomial time. Here, the dimension of an expression is defined as the number of distinct variables ocurring in it, where a variable may be reused several times. This motivates the following definition. A function f which inputs and outputs hereditarily definable sets is said to be computable in *fixed-dimension polynomial* time if for each $d \in \mathbb{N}$ there is a number $d' \in \mathbb{N}$ such that the following problem is computable in polynomial time (where the polynomial may depend on d):

[11] under L-reductions

Input: an expression of dimension d defining a hereditarily definable set x,
Decide: an expression of dimension d' defining the hereditarily definable set $f(x)$.

Here, the size of the input expression is defined as the number of different subexpressions in it.

We then prove (Lemma 4.11 in [TRAC]) that if the atoms are the pure set, all problems defined by fixed-point algorithms are computable in fixed-dimension polynomial time. This includes graph reachability, context-free grammar emtpiness, Horn-SAT, and many others.

As an example, it follows that the emptiness problem for register automata is decidable in polynomial time, once the number of allowed registers is fixed to a constant. A similar result holds for context-free grammars or pushdown automata.

More generally, we impose a notion of space and time resources for a while-program with atoms (or LOIS program), and prove (Theorem 4.9 in [TRAC]) that programs for which these resources are bounded by a polynomial, for every fixed dimension of inputs d, can be evaluated in time which is fixed-dimension polynomial.

Also, from the proof of our key lemma, Lemma 4.7 in [TRAC], in combination of the proof of Theorem 7 above, it follows that for every finite template $\mathbb{T}$, if CSP($\mathbb{T}$) is solvable in polynomial time, then Def-CSP($\mathbb{T}$) is solvable in fixed-dimension polynomial time. Hence, e.g. hereditarily definable systems of linear equations over a finite field can be solved in fixed-dimension polynomial time.

4 Turing machines with atoms [TUR, ALF]

The papers [TUR] and [ALF] study hereditarily definable Turing machines. These are defined just like ordinary Turing machines, but the state space and input alphabet are allowed to be hereditarily definable sets, rather than finite sets. The classical definitions of complexity classes such as P and NP lift naturally to Turing machines with atoms. We show that for some atoms, the $P = NP$ question can be answered negatively.

As one example of such atoms, consider an infinite vector space V over the two-element field. Elements of such a vector space can be identified with infinite sequences of zeros and ones[12], where addition of two such sequences is the coordinatewise xor.

Theorem 11 (Theorem VII.2 in [TUR])**.** *Fix $(V, +)$ as atoms. The language L of those sequences $v_1 v_2 \dots v_n \in V^*$ which are linearly dependent is:*

- *recognizable in polynomial time by a hereditarily definable nondeterministic Turing machine,*
- *not recognizable in polynomial time by any deterministic hereditarily definable Turing machine.*

For the first item, observe that a sequence $v_1, v_2, \dots, v_n$ is linearly dependent over the two-element field if and only if some nonempty subsequence adds up to the zero vector. A nondeterministic Turing machine with state space V can test this in linear time, by nondeterministically guessing which vectors to select.

The main result of [TUR] shows that in the case when the atoms are the pure set, nondeterministic Turing machines are more powerful than deterministic ones, even if they have ulimited resources:

Theorem 12 (Theorem III.1 in [TUR])**.** *Let atoms be the pure set. There is a hereditarily definable alphabet A and a language $L \subseteq A^*$ which is recognized by a nondeterministic Turing machine, but is not recognized by any deterministic Turing machine.*

[12]it is enough to consider sequences with finitely many ones; this results in an ω-categorical structure

In particular, in the setting of sets with atoms, for atoms being the pure set, $P \neq NP$. In contrast, we show that if the atoms are the dense linear order, the result above fails (cf. Theorem VII.1), and the answer to $P = NP$ is the same as classically.

We call an alphabet A for which a language L as in the theorem exists *nonstandard*; otherwise it is *standard*. Every alphabet of the form Atoms^k is standard. The letters in the nonstandard alphabet A in our construction are all sets of the following form, for all atoms $a_0, a_1, b_0, b_1, c_0, c_1$:

$$\{(a_0, b_0, c_0), (a_1, b_1, c_0), (a_0, b_1, c_1), (a_1, b_0, c_1)\} = \{(a_i, b_j, c_k) \mid i + j + k \text{ is even}\}.$$

The nondeterministically recognizable language $L \subseteq A^*$ is then defined by a geometric condition involving parity. Our geometric construction is inspired by a construction from model theory [8, example on page 819]. It also turns out to be strongly connected with the well-known Cai-Furer-Immerman construction from finite model theory (see Section 5 below). We study this connection in detail in the paper [ALF], where also a link to constraint satisfaction problems is revealed. Using this link, we give an effective characterization of standard alphabets definable over the pure set:

Theorem 13 (Theorem 4.7 in [ALF]). *It is decidable whether a given a hereditarily definable alphabet A is standard.*

To prove this result, given a hereditarily definable alphabet A we construct a template $\mathbb{T}_A$ such that A is standard if and only if $\mathbb{T}_A$ has bounded width. In our original proof, the obtained template $\mathbb{T}_A$ is finite. Testing if it has bounded width is decidable by known results. Our construction of the finite template $\mathbb{T}_A$ is a bit technical, and a more straightforward construction leads to a locally finite template $\mathbb{T}'_A$, for which the same equivalence still holds (see Section V in [LOC]). Testing whether $\mathbb{T}'_A$ has bounded width can be decided, by the results of [LOC] (cf. Theorem 10 above).

As an application of our effective characterization, we show that the nonstandard alphabet A presented above is the simplest possible: it has dimension 6, and every alphabet of dimension at most 5 is standard (see Section 5 in [ALF]). Moreover, the algorithm underlying Theorem 13 has been implemented[13], yielding a complete classification of all standard alphabets of dimension up to 8.

5 Connections to descriptive complexity theory [TRAC, LOC, ALF]

As mentioned above, the theory of sets with atoms turns out to have connections with descriptive complexity theory. The central open problem in this area is the question, whether there is a logic $\mathcal{L}$ *capturing polynomial time*, in the following sense:

- for every fixed sentence φ of the logic $\mathcal{L}$, it can be decided in polynomial time whether a given finite structure satisfies φ,
- every property of finite structures which can be decided in polynomial time is defined by some sentence φ of $\mathcal{L}$.

First-order logic, and the more general least fixpoint logic, both satisfy the first condition above. However, they do not satisfy the second condition, as witnessed by the parity query: does a given structure have an even number of elements? However, least fixpoint logic does capture polynomial time, if we restrict to structures which are equipped with a linear order, among other relations. This is known as the Immerman-Vardi theorem [13, 18].

[13]M.S of Ł. Wołochowski, University of Warsaw, 2014

Least fixpoint with counting [ALF]. To circumvent the problem arising from the parity query, an extension of fixpoint logic by a counting mechanism has been studied. This logic still satisfies the first condition above. However, Cai, Furer and Immerman showed in [7] that it still does not satisfy the second condition, by exhibiting the so-called *CFI-query*, which is very tightly connected to our construction of the nonstandard alphabet A.

In [ALF] we provide a common generalization of the Immerman-Vardi theorem and of the Cai-Furer-Immerman theorem. The starting point is the observation that every nonstandard alphabet A gives rise to a class of partially ordered finite structures which is recognizable in polynomial time, but is not definable by any formula of least fixpoing logic with counting. In our generalization, we consider classes of structures which are partially ordered in a certain restricted way, and characterize those classes, for which least fixpoint logic with counting captures polynomial time.

More precisely, for a fixed, finite graph $\mathfrak{p}$, a *linearly* $\mathfrak{p}$*-patched structure* is a finite graph G, together with a linearly ordered family $\mathfrak{p}_1 < ... < \mathfrak{p}_n$ of subgraphs of G, each of which is isomorphic to $\mathfrak{p}$, and which jointly cover G. The Immerman-Vardi theorem can be expressed as stating that for $\mathfrak{p}$ being the 2-clique, least fixpoint logic with counting captures polynomial time on lineaerly $\mathfrak{p}$-patched structures, whereas the Cai-Furer-Immerman result states the that it fails to do so when $\mathfrak{p}$ is the disjoint union of two 3-cliques. We show the following result, which gives an effective characterization of those $\mathfrak{p}$ for which fixed point logic captures polynomial time.

Theorem 14 (Theorem 6.4 in [ALF])**.** *Given a graph* $\mathfrak{p}$*, it can be effectively decided whether least fixpoint with counting captures polynomial time over linearly* $\mathfrak{p}$*-patched structures.*

It also follows from our results that least fixpoint logic captures polynomial time over linearly $\mathfrak{p}$-patched structures, whenever $\mathfrak{p}$ has fewer than 6 vertices.

Choiceless Polynomial Time [TRAC]. Choiceless Polynomial Time is another candidate for a logic capturing polynomial time, proposed by Blass, Gurevich and Shelah [3]. The logic also satisfies the first condition above, but it is still open whether it satisfies the second one.

In [TRAC], we extend this logic to a logic which operates on hereditarily definable structures. We show (see Theorem 4.9 in [TRAC]) that every formula of this logic defines a problem which is decidable in fixed-dimension polynomial time (cf. Section 3.2 above). We do not know whether the converse implication also holds: is it the case that every decision problem concerning hereditarily definable sets that is decidable in fixed-dimension polynomial time, is also definable in our logic? A positive answer to this question would imply that choiceless polynomial time captures polynomial time over finite structures, answering the central question of descriptive complexity theory.

References

[1] Libor Barto and Marcin Kozik. “Constraint Satisfaction Problems of Bounded Width”. In: *Procs. FOCS’09*. IEEE Computer Society.

[2] Libor Barto and Michael Pinsker. “The Algebraic Dichotomy Conjecture for Infinite Domain Constraint Satisfaction Problems”. In: *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*.

[3] Andreas Blass, Yuri Gurevich, and Saharon Shelah. “On polynomial time computation over unordered structures”. In: *J. Symbolic Logic* 67.3 (2002).

[4] Mikołaj Bojańczyk, Laurent Braud, Bartek Klin, and Sławomir Lasota. “Towards nominal computation”. In: *Procs. POPL 2012*.

[5] Mikołaj Bojańczyk, Bartek Klin, and Sławomir Lasota. "Automata with group actions". In: *Proc. LICS'11*.

[6] Andrei A. Bulatov. "A Dichotomy Theorem for Nonuniform CSPs". In: *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017*.

[7] Jin-yi Cai, Martin Fürer, and Neil Immerman. "An optimal lower bound on the number of variables for graph identifications". In: *Combinatorica* 12.4 (1992).

[8] Gregory Cherlin and Alistair H Lachlan. "Stable finitely homogeneous structures". In: *Transactions of the American Mathematical Society* 296.2 (1986).

[9] Elio Damaggio, Alin Deutsch, Richard Hull, and Victor Vianu. "Automatic Verification of Data-Centric Business Processes". In: *Procs. Business Process Management - 9th International Conference, BPM 2011.*

[10] Alin Deutsch, Richard Hull, Fabio Patrizi, and Victor Vianu. "Automatic verification of data-centric business processes". In: *12th International Conference on Database Theory, ICDT 2009*.

[11] Tomás Feder and Moshe Y. Vardi. "The Computational Structure of Monotone Monadic SNP and Constraint Satisfaction: A Study Through Datalog and Group Theory". In: *SIAM J. Comput.* 28.1 (1999).

[12] Wilfrid Hodges. *Model Theory*. Encyclopedia of Mathematics and its Applications 42. Cambridge University Press, 1993.

[13] Neil Immerman. "Relational queries computable in polynomial time". In: *Information and Control* 68.1 (1986).

[14] Bartek Klin and Michał Szynwelski. "SMT Solving for Functional Programming over Infinite Structures". In: Proceedings 6th Workshop on *Mathematically Structured Functional Programming*.

[15] M. Kaminski and N. Francez. "Finite memory automata". In: *Theor. Comp. Sci.* 134.2 (1994).

[16] Jaroslav Nešetřil and Patrice Ossona de Mendez. *Sparsity — Graphs, Structures, and Algorithms*. Vol. 28. Algorithms and combinatorics. Springer, 2012.

[17] V.G. Pestov. "On free actions, minimal flows, and a problem by Ellis". In: *Trans. Amer. Math. Soc.* 350 (1998).

[18] Moshe Y Vardi. "The complexity of relational query languages". In: *14th annual ACM symposium on Theory of computing, STOC 1982*.

[19] Dmitriy Zhuk. "A Proof of CSP Dichotomy Conjecture". In: *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017*.

My other publications

[20] Albert Atserias and Szymon Toruńczyk. "Non-Homogenizable Classes of Finite Structures". In: *25th EACSL Annual Conference on Computer Science Logic, CSL 2016*.

[21] M. Bojańczyk and S. Toruńczyk. "Imperative Programming in Sets with Atoms". In: *Procs. FSTTCS 2012*. Vol. 18. LIPIcs.

[22] Mikołaj Bojańczyk, Eryk Kopczyński, and Szymon Toruńczyk. "Ramsey's theorem for colors from a metric space". In: *Semigroup Forum* 85.1 (2012).

[23] Mikołaj Bojańczyk, Paweł Parys, and Szymon Toruńczyk. "The MSO+U Theory of (N, <) is Undecidable". In: *33rd Symposium on Theoretical Aspects of Computer Science, STACS 2016*.

[24] Mikołaj Bojańczyk and Szymon Toruńczyk. "Deterministic Automata and Extensions of Weak MSO". In: *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2009*.

[25] Mikołaj Bojańczyk and Szymon Toruńczyk. "Weak MSO+U over infinite trees". In: *29th International Symposium on Theoretical Aspects of Computer Science, STACS 2012*.

[26] Thomas Colcombet, Denis Kuperberg, Amaldev Manuel, and Szymon Toruńczyk. "Cost Functions Definable by Min/Max Automata". In: *33rd Symposium on Theoretical Aspects of Computer Science, STACS 2016*.

[27] Aldric Degorre, Laurent Doyen, Raffaella Gentilini, Jean-François Raskin, and Szymon Toruńczyk. "Energy and Mean-Payoff Games with Imperfect Information". In: *19th EACSL Annual Conference on Computer Science Logic, CSL 2010*.

[28] Grzegorz Fabianski, Michal Pilipczuk, Sebastian Siebertz, and Szymon Toru'nczyk. "Progressive Algorithms for Domination and Independence". In: *36th International Symposium on Theoretical Aspects of Computer Science, STACS 2019*. LIPIcs.

[29] Jakub Gajarský, Stephan Kreutzer, Jaroslav Nesetril, Patrice Ossona de Mendez, Michal Pilipczuk, Sebastian Siebertz, and Szymon Toru'nczyk. "First-Order Interpretations of Bounded Expansion Classes". In: *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018*.

[30] Tomasz Gogacz and Szymon Toruńczyk. "Entropy Bounds for Conjunctive Queries with Functional Dependencies". In: *20th International Conference on Database Theory, ICDT 2017*.

[31] Szczepan Hummel, Michał Skrzypczak, and Szymon Toruńczyk. "On the Topological Complexity of MSO+U and Related Automata Models". In: *35th International Symposium on Mathematical Foundations of Computer Science, MFCS 2010*.

[32] Bartek Klin, Slawomir Lasota, Joanna Ochremiak, and Szymon Toruńczyk. "Homomorphism Problems for First-Order Definable Structures". In: *36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2016*.

[33] Eryk Kopczyński and Szymon Toruńczyk. "LOIS: An Application of SMT Solvers". In: *14th International Workshop on Satisfiability Modulo Theories affiliated with the International Joint Conference on Automated Reasoning, SMT@IJCAR 2016*.

[34] Paweł Parys and Szymon Toruńczyk. "Models of Lambda-Calculus and the Weak MSO Logic". In: *25th EACSL Annual Conference on Computer Science Logic, CSL 2016*.

[35] Michal Pilipczuk, Sebastian Siebertz, and Szymon Toru'nczyk. "On the number of types in sparse graphs". In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018*. Ed. by Anuj Dawar and Erich Grädel. ACM.

[36] Michal Pilipczuk, Sebastian Siebertz, and Szymon Toru'nczyk. "Parameterized circuit complexity of model-checking on sparse structures". In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018*. Ed. by Anuj Dawar and Erich Grädel. ACM.

[37] Michal Pilipczuk and Szymon Toruńczyk. "On Ultralimits of Sparse Graph Classes". In: *Electronic Journal of Combinatorics* 23.2 (2016).

[38] Szymon Toruńczyk. “Languages of Profinite Words and the Limitedness Problem”. In: *39th International Colloquium on Automata, Languages, and Programming, ICALP 2012*.

Other achievements

I briefly describe my remaining achievements, grouped by area of research.

Sets with atoms. The papers [21, 33] where mentioned in the achievements above, and concern the described imperative programming language, LOIS. The first paper [21] is not included in the scientific achievements since its results are vastly subsumed by the paper [LOIS]. The paper [33] is a more detailed description of certain aspects studied in [LOIS]. The paper [32] continues the investigation initiated in [LOC], and studies variants of the problem of deciding the existence of a homomorphism between two hereditarily definable structures.

Boundedness problems for automata. The papers [23–27, 31, 38] are related to the limitedness problem for automata. We consider various models of automata and logics – on finite words, infinite words, infinite trees. These automata or logics are equipped with various means of determining whether a set of elements of the input structure is of bounded size. We prove various results concerning these models. In particular, my paper [38] developes a theory of boundedness based on the notion of *profinite words*.

Sparse graphs. The papers [28, 29, 35–37] concern finite graphs which are sparse, in a certain sense made precise by the notion of nowheredenseness, introduced by Nešetřil and Ossona de Mendez [16]. We study various algorithmic problems for such graphs, related to model-checking formulas of first-order logic.

Other papers. The paper [30] concerns the query evaluation problem for databases. It proves an optimal bound for the output size of conjunctive queries, using tools from information theory. The paper [20] concerns classes of amalgamation structures. We show that certain classes arising from the theory of constraint satisfaction problems are not closed under amalgamation. The paper [22] concerns a generalization of Ramsey's theorem to the case where the set of colors comes from an infinite, compact metric space. The paper [34] concerns evaluating formulas of Weak MSO logic on models of λ-calculus.

Szymon Toruńczyk