Semiautomatic Structures*

Sanjay Jain[1], Bakhadyr Khoussainov[2], Frank Stephan[1,3], Dan Teng[3] and Siyuan Zou[3]

[1] Department of Computer Science, National University of Singapore
13 Computing Drive, COM1, Singapore 117417, Republic of Singapore
sanjay@comp.nus.edu.sg
[2] Department of Computer Science, University of Auckland, New Zealand
Private Bag 92019, Auckland, New Zealand
bmk@cs.auckland.ac.nz
[3] Department of Mathematics, The National University of Singapore
10 Lower Kent Ridge Road, S17, Singapore 119076, Republic of Singapore
fstephan@comp.nus.edu.sg, tengdan@u.nus.edu, zousiyuan@hotmail.com

Abstract. Semiautomatic structures generalise automatic structures in the sense that for some of the relations and functions in the structure one only requires the derived relations and functions are automatic when all but one input are filled with constants. One can also permit that this applies to equality in the structure so that only the sets of representatives equal to a given element of the structure are regular while equality itself is not an automatic relation on the domain of representatives. It is shown that one can find semiautomatic representations for the field of rationals and also for finite algebraic field extensions of it. Furthermore, one can show that infinite algebraic extensions of finite fields have semiautomatic representations in which the addition and equality are both automatic. Further prominent examples of semiautomatic structures are term algebras, any relational structure over a countable domain with a countable signature and any permutation algebra with a countable domain. Furthermore, examples of structures which fail to be semiautomatic are provided.

1 Introduction

General background. An important topic in computer science and mathematics is concerned with classifying structures that can be presented in a way that certain operations linked to the structures are computed with low computational complexity. Automatic functions and relations can, in some sense, be considered to have low complexity. The first work in this field centered on the question which sets are regular (that is, recognised by finite automata) and how one can transform the various descriptions of regular sets into each other. Later mathematicians applied

* S. Jain was supported in part by NUS grant C252-000-087-001; furthermore, S. Jain and F. Stephan were both supported in part by NUS grants R146-000-181-112 and R252-000-420-112. B. Khoussainov was supported in part by a Marsden Fund grant of the Royal Society of New Zealand. The paper was started while B. Khoussainov was on sabbatical leave to the National University of Singapore. The final work on this paper was done with support in part to S. Jain, B. Khoussainov and F. Stephan by the Singapore Ministry of Education Academic Research Fund Tier 2 grant MOE2016-T2-1-019 / R146-000-234-112.

the concept also to structures: Thurston automatic groups [4] are one of the pioneering works combining automata theory with structures. Here one has (a) a regular set of representatives A consisting of words over a finite alphabet of generators, (b) an automatic equivalence relation representing equality and (c) for every fixed group member y, an automatic mapping f_y from A to A such that $f_y(x)$ is a representative of the group member $x \circ y$. Here a function is automatic iff its graph can be recognised by a finite automaton or, equivalently, iff it is computed in linear time by a one-tape Turing machine which replaces the input by the output on the tape, starting with the same position [1]. These concepts have been generalised to Cayley automatic groups [12,19] and to automatic structures in general.

For automatic structures, one has to define how to represent the input to functions that have several inputs. This is now explained in more detail. If Σ is the alphabet used in the regular domain $A \subseteq \Sigma^*$ of the structure, one defines the convolution of two strings $a_0 a_1 \ldots a_n$ and $b_0 b_1 \ldots b_m$ to consist of combined characters $c_0 c_1 \ldots c_{\max\{m,n\}}$ where

- $c_k = \binom{a_k}{b_k}$ if $k \leqslant \min\{m, n\}$,
- $c_k = \binom{a_k}{\#}$ if $m < k \leqslant n$ and
- $c_k = \binom{\#}{b_k}$ if $n < k \leqslant m$.

Here $\#$ is a fixed character outside Σ used for padding purposes. Convolution of strings x and y is denoted by $conv(x, y)$. Now the domain of a function $f : A \times A \to A$ is the set $\{conv(x, y) : x, y \in A\}$ which might from now on be identified with $A \times A$. Similarly one can define convolutions of more than two parameters and also define that an automatic relation over A^k is an automatic function from A^k to $\{0, 1\}$ taking 1 on those tuples where the relation is true and taking 0 otherwise. A structure $\mathcal{A}$ is automatic iff it is isomorphic to a structure $\mathcal{B}$ such that the domain and all functions and relations in the structure are automatic.

Let $\mathbb{N}$ denote the set of natural numbers, $\mathbb{Z}$ the set of integers and $\mathbb{Q}$ the set of rational numbers. Now $(\mathbb{N}, +, =, <)$ is an automatic structure, as (i) there is a regular set A such that each member of $\mathbb{N}$ is represented by at least one member of A, (ii) there is an automatic function $f : A \times A \to A$ such that for each $x, y \in A$ the value $f(x, y)$ is a representative of the sum of the elements represented by x, y and (iii) the sets $\{conv(x, y) : x, y$ represent the same element of $\mathbb{N}\}$ and $\{conv(x, y) : x$ represents a number n and y represents a number m with $n < m\}$ are both regular. Automatic structures were introduced by Hodgson [6,7] and later, independently, by Khoussainov and Nerode [13]. Automatic structures have a decidable first-order theory and every function or relation first-order definable in an automatic structure (with quantification over members of the structure, say group elements and using as parameters relations from the structure or other automatic relations introduced into the representation of the structure) is again automatic. These closure properties made automatic structures an interesting field of study; however, a limitation is its expressiveness. For example, the structure $(\mathbb{N}, \cdot, =)$ is not automatic yet its first-order theory is decidable. There is a limited version of multiplication which is automatic in every automatic presentation of $(\mathbb{N}, +)$ or $(\mathbb{Z}, +)$: For every multiplication with a fixed element n, one can find an automatic function which maps every representative of a number m to a representative of the number $m \cdot n$.

Therefore, one would like to overcome the lack of expressivity of automatic structures and address the following questions: (1) Are there general ways to utilise finite automata for the representation of non-automatic structures such as $(\mathbb{Q},+)$ and $(\mathbb{N},\cdot,=)$? (2) Under such general settings, what properties of automatic structures should be sacrificed and what properties should be preserved to accommodate non-automatic structures as those mentioned above? (3) What are the limits of finite automata in representations of structures?

The present paper proposes one possible approach to address the questions above. The main concept is motivated by the notions of Thurston automaticity and Cayley automaticity for finitely generated groups [4,12]. Namely, one says that a function $f: A^k \mapsto A$ is semiautomatic iff whenever one fixes all but one of the inputs of f with some fixed elements of A, then the resulting mapping from A to A is automatic. Similarly a relation $R \subseteq A^k$ is semiautomatic, if it is a semiautomatic function when viewed as a $\{0,1\}$-valued function (mapping the members of R to 1 and the non-members of R to 0). This permits now to give the general definition using finite automata representing structures. For a structure, say $(\mathbb{N},+,<,=;\cdot)$, one says that this structure is semiautomatic iff there is a representation $(A,f,B,C;g)$ of this structure such that A is a regular set of representatives of $\mathbb{N}$, f is an automatic function representing $+$, B,C are automatic relations representing $<,=$, respectively and g is a semiautomatic function representing the multiplication. Note that the convention here is that the relations and functions before the semicolon have to be automatic while those after the semicolon need only to be semiautomatic. Hence in a structure $(\mathbb{N},+,<,=;\cdot)$ the operation $+$ and the relations $<$ and $=$ have to be automatic and $\cdot$ is only semiautomatic while in a structure $(\mathbb{Q};+,\cdot,<,=)$ not only the operations addition and multiplication are semiautomatic but also the relations $<$ and $=$, that is, only the sets which compare to a fixed element (say all representatives of numbers below $1/2$ or all representatives of 5) have to be regular. This difference is crucial, for example, $(\mathbb{Q},+;=)$ is not semiautomatic [28] and $(\mathbb{Q},=;+)$ is semiautomatic. It is of course the goal to maintain automaticity for as many operations and relations as possible, therefore one needs to pay attention to these differences. Here are some important comments on the structures.

- The condition that a basic function, say $f: A^2 \to A$, is semiautomatic requires, for all $a \in A$, merely the existence of automata recognising the sets $\{conv(x,y) : y = f(x,a)\}$ and $\{conv(x,y) : y = f(a,x)\}$. This part of the definition is kept as general as possible to accommodate a large class of structures. In particular, this part is needed to address question (3) posed above. Obviously, the requirement that the graph $\{conv(x,y) : y = f(x,a)\}$ is automatic can be made effective; namely, there is an algorithm that given any a from the domain produces a finite automaton recognising the graph $\{conv(x,y) : y = f(x,a)\}$. All the results of the paper, apart from Theorems 9, 10 and 11, satisfy this effectiveness condition. Thus, under this effectiveness condition, semiautomatic structures are still structures with finite presentations.
- For the structures $\mathcal{A}$ with no relation symbols, semiautomaticity is equivalent to saying that all algebraic polynomials with one variable (as defined in the beginning of Section 2) are automatic. Thus, semiautomaticity under the effectiveness condition, is equivalent to saying that the structure $\mathcal{A}' = (A, g_0, g_1, \ldots)$, where $g_0, g_1, \ldots$ is the list of all algebraic polynomials

with one variable, is automatic. In particular this implies that the first order theory of this structure derived from $\mathcal{A}$ is decidable. The first order theory of $\mathcal{A}'$, can naturally be embedded into the first order theory of $\mathcal{A}$. Hence, semiautomaticity of $\mathcal{A}$ under the effectiveness condition, implies that a natural fragment of the first order theory of $\mathcal{A}$ is decidable. Moreover, algebraically the structure $\mathcal{A}'$ has exactly the same set of congruences as the original structure $\mathcal{A}$.

– There is a difference between semiautomatic / automatic functions and relations when $=$ is only semiautomatic and not automatic. While a function, for each input, has to find only one representative of an output, a relation must be true for all representatives of a given tuple which is satisfied. Therefore, it can be that a function is automatic while the graph $\{conv(x,y) : y = f(x)\}$ is not automatic. However, the graph of an automatic function is semiautomatic. Given some representative z of a member of the structure, the sets $\{y : y$ and $f(z)$ represent the same element$\}$ of the representatives of the image of z and $\{x : \exists y\,[f(x)$ is mapped to y and y represents the same element as $z]\}$ of representatives of the preimage of z are regular. This difference in the effectivity of functions and relations is found in many domains where equality is not fully effective. For example there are many methods to systematically alter computer programs (for example, if p computes $x \mapsto f(x)$ then $F(p)$ computes $x \mapsto f(x) + 1$). For many programming languages, such an F can even be realised by an automatic function transforming the programs. However, it would be impossible to check whether a program q is equal to $F(p)$ in the sense that it has the same input/output behaviour: the relation $\{(p, q)$ such that q computes a function producing outputs one larger than those outputs produced by $p\}$ is indeed an undecidable set, due to Rice's Theorem [9].

Often one identifies the rationals with the set of all pairs written as a/b with $a \in \mathbb{Z}$ and $b \in \{1, 2, \ldots\}$; so one identifies "one half" with each of 1/2, 2/4, 3/6, ... and consider all of these to be equal. Similarly, in the case that the distinction is not relevant, the represented structure is often identified with its automatic or semiautomatic presentation and one denotes representatives in the automatic domain by their natural representation or vice versa and denotes the automatic functions realising these operations with the usual notation for operations of the structure represented.

History of this paper. Part of the results at this paper were presented at the conference Computability, Complexity and Randomness 2015 in Heidelberg and other parts have appeared at the Proceedings of *Computer Science – Theory and Applications* [10]. The paper includes results from the Honours Year Projects of Dan Teng [26] and Siyuan Zou [27] at the National University of Singapore. Follow-up investigations on this research were carried out by Hölzl, Jain, Khoussainov and Stephan [8,11] and targeted at the connections of automatic and semiautomatic groups to learning theory [8] and nilpotent groups [11].

Contributions of the paper. First, the paper proposes the class of semiautomatic structures that can be defined in terms of finite automata. This class contains all automatic structures. Under the effectiveness condition put on semiautomaticity, (1) these structures have finite presentations, (2) natural fragments of their first order theories are decidable and (3) the class is wide enough to contain structures with undecidable theories. The paper provides many examples of semiautomatic structures, see Section 2.

Second, the paper provides several results of a general character. For example, purely relational structures, countable ordinals and permutation algebras all have semiautomatic presentations. This provides a large class of semiautomatic structures and showcases the power of finite automata in representation of algebraic structures. Note that for these results, no effectivity constraints on the semiautomaticity are made. See Section 4.

Third, the paper proves semiautomaticity for many of the classical algebraic structures which are groups, rings and vector spaces. The main reason for this study is that most of these structures lack automatic presentations; for example, $(\mathbb{Q}, =, <; +)$, $(\mathbb{Z}, +, =, <; \cdot)$ and some infinite fields are semiautomatic but not automatic. Therefore, it is natural to ask which of these structures admit semiautomaticity. Many of these structures and in particular all concretely given examples are also semiautomatic with the effectivity condition. For instance, the ordered field $(\mathbb{Q}(\sqrt{n}); +, \cdot, <, =)$ is semiautomatic for every natural number n. There are also several counterexamples which are not semiautomatic. These examples and counterexamples are presented in Sections 5, 6 and 7.

2 Decidability theorem and examples

The first result is a simple and general decidability result about semiautomatic structures without relational symbols. So, let $\mathcal{A} = (A, f_1, \ldots, f_n)$ be a semiautomatic structure where each f_i is an operation. An *algebraic polynomial* is a unary operation g of the form $f(a_1, \ldots, a_k, x, a_{k+2}, \ldots, a_n)$, where f is a basic operation of $\mathcal{A}$, and $a_1, \ldots, a_k, a_{k+2}, \ldots, a_n$ are parameters from A. Consider the structure $\mathcal{A}' = (\mathcal{A}; g_0, g_1, \ldots)$, where $g_0, g_1, \ldots$ is a complete list of all algebraic polynomials obtained from $f_1, \ldots, f_n$. There is a close relationship between $\mathcal{A}$ and $\mathcal{A}'$ in terms of congruence relations (that is equivalence relations respected by the basic operations):

Proposition 1. *The set of congruences of the structures $\mathcal{A}$ and $\mathcal{A}'$ coincide.*

The transformation $\mathcal{A} \to \mathcal{A}'$ gives an embedding of the first order theory of $\mathcal{A}'$ (with parameters from A) into the first order theory of $\mathcal{A}$. The embedding is the identity mapping. Assuming the effectivity condition (that is there is an algorithm that given any algebraic polynomial g produces a finite automaton recognising the graph of g), the automatic structure $\mathcal{A}'$ has a decidable first order theory.

Theorem 2 (Decidability Theorem). *If $\mathcal{A}$ is semiautomatic, then under the effectivity condition the first order theory of $\mathcal{A}'$ is decidable.*

The next examples illustrate that there are many semiautomatic structures which are not automatic.

Example 3. $(\{0, 1\}^*, =; \circ)$ with $\circ$ being the string concatenation is an example of a structure which is semiautomatic but not automatic. For a fixed string v, the mappings $w \mapsto vw$ and $w \mapsto wv$ are both automatic; however, $conv(v, w) \mapsto vw$ is not an automatic mapping. Indeed, there is no automatic presentation of $(\{0, 1\}^*, =, \circ)$.

Furthermore, $(\mathbb{N}, +, <, =; \cdot)$ is an example of a semiautomatic structure which is not automatic, as there is no automatic copy of the multiplicative structure of the natural numbers $(\mathbb{N}, \cdot)$. It is known that multiplication is semiautomatic due to the fact that multiplication with a constant can be implemented as repeated addition. One can augment this example with a semiautomatic function $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that $f(x+y, x) = 3^x + 3^{x+1} \cdot y - 1$ and $f(x, x+y+1) = 2 \cdot 3^x + 3^{x+1} \cdot y - 1$. Note that f is a semiautomatic bijection and there exists no infinite regular set A for which there exists an automatic bijection $g : A \times A \to A$.

Every two-sided Cayley automatic group $(G, \circ)$ is an example of a semiautomatic structure $(G, =; \circ)$; the reason is that for finitely generated groups, multiplication with group elements is automatic, as the multiplication with each generator is automatic. Furthermore, the automaticity of $=$ follows from the definition of Cayley automatic groups. In turn, the definition of a semiautomatic structure gives that every finitely generated semiautomatic group $(G, =; \circ)$ is two-sided Cayley automatic. Miasnikov and Šunić [19] provide an example of a group $(G, \circ)$ which is one-sided but not two-sided Cayley automatic, thus one would not have that $(G, =; \circ)$ is semiautomatic but only that $(G, =; \{x \mapsto x \circ y : y \in G\})$ is semiautomatic.

Example 4. Let $\mathbb{S}$ be the set of square numbers. Then $(\mathbb{N}, \mathbb{S}, <, =; +)$ is semiautomatic. This structure is obtained by first using a default automatic representation $(A, +, <, =)$ of the additive monoid of the natural numbers and then to let $B = \{conv(a, b) : a, b \in A \wedge b \leqslant a + a\}$ be the desired structure. Here $conv(a, b)$ represents $a^2 + b$. One has now $conv(a, b) < conv(a', b')$ iff $a < a' \vee (a = a' \wedge b < b')$. Furthermore, $conv(a, b) + 1 = conv(a', b')$ iff $(a = a' \wedge b' = b + 1 \leqslant a + a) \vee (a' = a + 1 \wedge b = a + a \wedge b' = 0)$. Iterated addition with 1 defines the addition with any fixed natural number. Note that $(\mathbb{N}, \mathbb{S}, +, <, =)$ is not automatic, as the multiplication can be defined from $\mathbb{S}$: first one defines the square function using that $y = x^2$ iff $y, y + x + x + 1 \in \mathbb{S}$ and none of the numbers $y + 1, y + 2, \ldots, y + x + x$ is in $\mathbb{S}$; second one defines the product using that $x \cdot y = z$ iff $x^2 + y^2 + z + z = (x + y)^2$.

3 Term Algebras

The term algebra of a binary function f over a constant a consist of the term a and all terms $f(x, y)$ formed from previous terms x and y; for example $f(a, a)$, $f(a, f(a, a))$ and $f(f(a, a), f(a, a))$ are terms. Let T denote the set of all terms formed using the constant a and binary function f.

Theorem 5. *The term algebra $(T; f, =)$ is semiautomatic.*

Proof. Let $x_0, x_1, \ldots$ be a one-one enumeration of all terms. One now has to find a representation of the terms in which all mappings $left_k : y \mapsto f(x_k, y)$ and $right_k : y \mapsto f(y, x_k)$ are automatic. The idea is to represent a by 0 and each function $left_k$ by 01^{2k} and each function $right_k$ by 01^{2k+1}. That is, if w represents a term y, then $01^{2k}w$ denotes $left_k(y)$ and $01^{2k+1}w$ denotes $right_k(y)$. Note that each term starts with a 0 and thus, for each $w \in (01^*)^*0$, there is a unique term represented by w.

For the above representation, the functions $left_k$ and $right_k$ are clearly automatic, as each of them just inserts the prefix 01^{2k} or 01^{2k+1} in front of the input. Thus, f is semiautomatic.

Let $depth(a) = 0$ and $depth(f(x, y)) = 1 + max\{depth(x), depth(y)\}$. Now each term y has only finitely many representations, as it can only have representations which have at most $depth(y) + 1$ zeros and each $left_k$ or $right_k$ used in the representation must satisfy that x_k is a sub-term of y. Thus, $=$ is semiautomatic. $\square$

Remark 6. Using a more careful representation, one can get a semiautomatic representation for $(T, =; \{y \mapsto f(x, y)\})$; this representation has the disadvantage that only the mappings with the first parameter being the fixed one are automatic. It is known that one cannot make $(T, =, f)$ to be fully automatic and the proof actually gives that $(T, f; =)$ is also not semiautomatic. This can be shown as follows: there is a constant c such that all terms of depth n have a representative of length up to $(n+1) \cdot c$; this can be shown by induction, as f combines two inputs of length up to $(n+1) \cdot c$ to form a new term of depth $n+1$ represented by a string of length up to $(n+2) \cdot c$. Here c is the constant which bound the length of the representative a and the maximum length-increase of the automatic function f on arbitrary inputs: $|f(x, y)| \leqslant c+\max\{|x|, |y|\}$ for any two representatives x, y of some terms. Now the number of terms of depth n is doubly exponential in n. To see this, note that one can form a tree with 2^n nodes at level n, with each node at level n being either the subterm $f(a, a)$ or the subterm a; thus there are at least 2^{2^n} many terms of depth up to $n + 1$. On the other hand, there are only exponentially many strings of length up to $(n + 2) \cdot c$. Hence, for large n, not every term of depth $n + 1$ can be represented by a string of length $(n + 2) \cdot c$. It follows that f above cannot be automatic.

For the case of infinitely many constants, one can obtain that the term algebra of one binary function f and all these constants is semiautomatic with the side condition that the equality is automatic.

Theorem 7. *Let T^* be the set of all terms formed with a binary function f and constants from the natural numbers. $(T^*, =; f)$ is semiautomatic.*

Proof. The idea is that for every fixed term t represented by v there is a constant k such that, for almost all further terms t' represented by w, $1^{2k}0w$ is the representation for $f(t', t)$ and $1^{2k+1}0w$ is the representation for $f(t, t')$. However, one wants only one legal representative, thus the representatives for $f(t', t)$ and $f(t, t')$ are picked as follows, where k' is the constant associated to t':

- If $v \leqslant_{ll} w$ then $1^{2k}0w$ represents $f(t', t)$ else $1^{2k'+1}0v$ represents $f(t', t)$;
- If $v <_{ll} w$ then $1^{2k+1}0w$ represents $f(t, t')$ else $1^{2k'}0v$ represents $f(t, t')$.

This definition renders many strings from the set $\{1\}^*$ as well as infinitely many other binary strings not to represent composed terms. However, for each k there are only finitely many strings of the form $1^{2k}0w$ and $1^{2k+1}0w$ which are unused and therefore one can hard-code these finitely many exceptions in tables and so the mappings from a representative of t' to representatives of $f(t, t')$ and $f(t', t)$ are both automatic for every fixed t. Furthermore, one can use all the unused strings to find representatives for the constants $0, 1, \ldots$ from $\mathbb{N}$. The choice is that the unused string number n with respect to length-lexicographic order and starting with string number 0 represents the constant n from $\mathbb{N}$. Thus every two different binary strings represent different terms and so equality is automatic for this structure. Thus the term algebra $(T^*, =; f)$ is semiautomatic. $\square$

One can generalise the above results to term algebras with several generators and with constants. Furthermore, one can combine partial semiautomatic structures and obtain the free term-algebraic extension of a partial semiautomatic structure. This is now explained by the following example, where it is obvious how the definition generalises.

Let a semiautomatic structure $(A; f, g, =)$ be given where f is a binary function and g a ternary function. Note that a partial function like f is semiautomatic iff, for all values $a \in A$, the domains $\{w \in A : f(a, w) \text{ is defined}\}$ and $\{w \in A : f(w, a) \text{ is defined}\}$ are both regular and the partial mappings $w \mapsto f(a, w)$ and $w \mapsto f(w, a)$ are both automatic on their domains.

For the inductive definition of the free term-algebraic extension, one defines that a term has depth 0 iff it is an element of A. Furthermore, one introduces terms of depth 1 being either $f(a, b)$ or $g(a, b, c)$ with $a, b, c \in A$ and satisfying that $f(a, b)$ or $g(a, b, c)$ was originally not defined in the algebra $(A; f, g, =)$, so that these terms are introduced to represent the missing values. For $n \geqslant 1$, a term of depth $n + 1$ is either of the form $f(a, b)$ with $\max\{depth(a), depth(b)\} = n$ or of the form $g(a, b, c)$ with $\max\{depth(a), depth(b), depth(c)\} = n$. The next result shows that structures of this type are semiautomatic and provides an explicit coding for this structure, which generalises to other structures.

Theorem 8. *The free term-algebraic extension of a partial semiautomatic structure is semiautomatic.*

Proof. The proof is given for the above example $(A; dom(f), dom(g), f, g, =)$ and easily generalises to structures with other or more semiautomatic functions. Let 0 and 1 be two symbols outside the alphabet of A and let $h_0, h_1, \ldots$ be a listing of all the mappings which are represented by a term with one leaf being a variable x and the leaf sitting on the top level, say $f(a, x)$ or $g(f(a, b), x, g(c, d, e))$ where a, b, c, d, e are fixed members of A; in the second term, $f(a, b)$ and $g(c, d, e)$ would be assumed to be undefined in the original structure $(A; dom(f), dom(g), f, g, =)$, as otherwise $f(a, b)$ or $g(c, d, e)$ would have been replaced by some element of A. Now the mapping h_k would in the first case map any term w to the value of $f(a, w)$ and in the second case map w to the value of $g(f(a, b), w, g(c, d, e))$.

Any arbitrary function using f, g can be considered as composition of several h_k's, where the resulting function will be automatic. For example consider $f(a, g(x, b, c))$. For any input w from A, the output is in A iff $w \in dom(g(x, b, c))$ and the value d computed by the partial automatic function representing $x \mapsto g(x, b, c)$ satisfies that d is in the domain of $y \mapsto f(a, y)$. Thus the domain and the function $x \mapsto f(a, g(x, b, c))$ viewed as a function from A to A is regular and, on its domain, the function is automatic. It is realised as the composition of two function from the base domain. Similarly, one can realise more complicated functions by composing the base functions from the term-algebra where these base-functions satisfy that the variable sits on the top level — the composed function can be of a more general form.

Hence for every h_k there is an induced partial semiautomatic function $\tilde{h}_k : A \mapsto A$ where $\tilde{h}_k(w)$ is defined iff $w \in A$ and $h_k(w) \in A$. Furthermore, let t be a default term which is undefined in the original semiautomatic structure, such a term must exist as otherwise the whole new structure collapses to $(A; f, g, =)$ and one can assume that t has depth 1. Now one makes the following representatives:

- Members of A are represented by strings in A as before;
- For $w \in A$, if $\tilde{h}_k(w)$ is undefined then 1^k0w represents the term $h_k(w)$ else 1^k0w represents the term $h_k(t)$;
- For $w \in (1^*0)^+ \cdot A$, the term 1^k0w represents the term $h_k(w)$.

So the set $(1^*0)^* \cdot A$ of all representatives of elements of the term algebra is regular. Furthermore, for each $w \in (1^*0)^* \cdot A$, there is an automatic function which behaves as follows: If $w \in A \cap dom(\tilde{h}_k)$ then the function outputs $\tilde{h}_k(w)$ else the function outputs 1^k0w. Thus, each function h_k is automatic and the functions f, g are semiautomatic.

Furthermore, one can show by induction that for each term t' there is a finite set F of tuples $(n, k_1, k_2, \ldots, k_n, a)$ such that a is the length-lexicographic least member of $\{b \in A : a = b\}$ and $t' = \tilde{h}_{k_1}(\tilde{h}_{k_2}(\ldots \tilde{h}_{k_n}(a) \ldots))$. Hence the regular set $\{1^{k_1}01^{k_2}0 \ldots 1^{k_n}0b : b \in A \wedge (n, k_1, \ldots, k_n, a) \in F[b = a]\}$ is the set of all representatives of the term t' and equality is semiautomatic. $\square$

Note that Kozen [15] showed that every finitely presented algebra is isomorphic to a free term algebraic extension of a finite partial algebra, hence the above result also shows that every finitely presented algebra is automatic.

4 Relational Structures, Permutation Algebras and Ordinals

This section shows that when the signature of the structure is very restricted then the resulting structure is always semiautomatic. Theorem 9 says that if a structure over a countable domain consists only of relations and each of these relations has only to be semiautomatic and not automatic, then one can indeed find a representation for this structure; this first result will then be applied to show that every countable set of ordinals which is closed under $+$ and $<$ has a semiautomatic representation.

Theorem 9. *Every relational structure $(A; R_1, R_2, \ldots)$, given by at most countably many relations $R_1, R_2, \ldots$ over a countable domain A, is semiautomatic.*

Proof. In the proof, it does not matter whether the unary relations on A considered are obtained by fixing parameters from relations of higher order or whether they are unary relations themselves. Therefore, one can without loss of generality assume that $A = \mathbb{N}$ and the relations $R_0, R_1, \ldots$ are sets. The proof now proceedes by constructing a bijection $\pi : \mathbb{N} \to \{0, 1\}^*$ such that image of each R_e is a finite variant of a finite union of sets of the form $x \cdot \{0, 1\}^*$, where x ranges over finite strings. Such sets are obviously regular, hence π maps $\mathbb{N}$ to $\{0, 1\}^*$ in a way that $\pi(R_e)$ is regular for each e. Furthermore, the equality on the structure will just be the equality of strings, so the obtained copy $(\{0,1\}^*, =; \{\pi(R_e) : e \in \mathbb{N}\})$ will be semiautomatic.

The construction of π is done in stages. In stage e, (i) a new finite tree T_{e+1} extending a finite tree T_e is constructed and (ii) $\mathbb{N}$ is split into infinite sets $\{L_x : x \text{ is a leaf of } T_{e+1}\}$. The following properties will be satisfied:

(P1) π will be defined on all values below n_{e+1}, where n_{e+1} satisfies $n_{e+1} \geqslant e+1$ and $n_0 = 0$. Additionally, the range of π will contain all strings of length at most $e+1$.
(P2) π will map almost all members of L_x into extensions of the string x.
(P3) For all leaves x of T_{e+1}, either almost all $n \in L_x$ satisfy $n \in R_e$ or almost all $n \in L_x$ satisfy $n \notin R_e$.

Initially, T_0 consists only of the empty string ε as root node (which is the unique leaf of T_0) and $L_\varepsilon = \mathbb{N}$. Assume that prior to stage e the function π is defined on all inputs below n_e. Now one keeps defining $\pi(m)$ for $m = n_e, n_e+1, \ldots$ as follows until the condition mentioned below holds: let x be a leaf of T_e such that $m \in L_x$; then let $\pi(m)$ be the length-lexicographically least string extending x which is not yet in the range of π. This process continues until all strings of length up to $e+1$ are in the range of π and $m \geqslant e+1$. The number n_{e+1} is then the first n where $\pi(n)$ does not get defined in stage e. Then, one lets T_{e+1} be the union of T_e and all sets $\{x0, x1\}$ such that x is a leaf of T_e and both $L_x \cap R_e$ and $L_x - R_e$ are infinite. For each such x where $\{x0, x1\}$ are in $T_{e+1} - T_e$, define $L_{x0} = L_x - R_e$ and $L_{x1} = L_x \cap R_e$. It is easy to verify the following facts by induction:

(a) Strings in tree T_e are of length at most e;
(b) For each leaf x of T_{e+1} it holds that (i) L_x is infinite and (ii) all $m \in L_x$ for which $\pi(m)$ is defined after stage e satisfy that $\pi(m) \succeq x$ and (iii) all leaves x of T_{e+1} satisfy that either almost all members of L_x are contained in R_e or almost all members of L_x are outside of R_e;
(c) In each step only finitely many values of π are defined;
(d) Each set $\pi(R_e)$ is regular, as for all but finitely many strings y, it depends only on the first $e+1$ bits of a string y whether $y \in \pi(R_e)$ or not.

The above facts imply that the resulting copy of $(\mathbb{N}, =; \{R_e : e \in \mathbb{N}\})$ is semiautomatic. $\square$

Delhommé [3] showed that some automatic ordered set $(A, <, =)$ is isomorphic to the set of ordinals below α iff $\alpha < \omega^\omega$. Furthermore some tree-automatic set $(A, <, =)$ is isomorphic to the set of ordinals below α iff $\alpha < \omega^{\omega^\omega}$. It follows directly from Theorem 9 that every countable set of ordinals is isomorphic to a semiautomatic ordered set. The next result shows that one can combine this result also with a semiautomatic addition of ordinals.

Theorem 10. *Let α be a countable ordinal. The structure $(\{\beta : \beta < \omega^\alpha\}; +, <, =)$ is semiautomatic.*

Proof. Let $(A; <, =)$ be a semiautomatic representation of the ordinals below α and assume that the symbols ω, ^, (,), + are not in the alphabet of A. Now let B be the set of all strings of the form $\omega\hat{}(a_0) + \omega\hat{}(a_1) + \ldots + \omega\hat{}(a_n)$ representing the ordinal $\omega^{a_0} + \omega^{a_1} + \ldots + \omega^{a_n}$ with the empty string representing 0. The set of all possible representations of the ordinals below ω^α is regular. Now one can realise the addition of two non-empty strings v, w by forming $v + w$, so if $v = \omega\hat{}(5)+\omega\hat{}(3)+\omega\hat{}(3)$ and $w = \omega\hat{}(4)+\omega\hat{}(1)+\omega\hat{}(0)$ then $v+w = \omega\hat{}(5)+\omega\hat{}(3)+\omega\hat{}(3)+\omega\hat{}(4)+\omega\hat{}(1)+\omega\hat{}(0)$ representing $\omega^5+\omega^3+\omega^3+\omega^4+\omega^1+\omega^0$. Ordinals have the rules that if $a < b$ then $\omega^a + \omega^b = \omega^b$, hence the above ordinal equals $\omega^5+\omega^4+\omega^1+\omega^0$.

Following Cantor's arguments, for every ordinal $\beta < \omega^\alpha$, there is a normal form $\beta = \omega^{b_0} + \omega^{b_1} + \ldots + \omega^{b_n}$ for some n with $b_0, b_1, \ldots, b_n \in A$ and $b_0 \geqslant b_1 \geqslant \ldots \geqslant b_n$. Now, given an ordinal $w = \omega^{a_0} + \omega^{a_1} + \ldots + \omega^{a_m}$, it holds that $w = \beta$ iff there are $i_0, i_1, \ldots, i_n$ such that $i_0 < i_1 < \ldots < i_n = m$ and for all $k \leqslant m$, the least index j with $k \leqslant i_j$ must satisfy $k < i_j \Rightarrow a_k < b_j$ and $k = i_j \Rightarrow a_k = b_j$. Given the automata to check whether some $a \in A$ is below or equal b_j, one can build from these finite automata an automaton which checks whether $w = \beta$. Furthermore, $w < \beta$ iff there are $n' \leqslant n$ and $i_0, i_1, \ldots, i_{n'-1}, i_{n'}$ with $i_0 < i_1 < \ldots < i_{n'} = m+1$ and for all $k \leqslant m$, the least index j with $k \leqslant i_j$ must satisfy $k < i_j \Rightarrow a_k < b_j$ and $k = i_j \Rightarrow a_k = b_j$. Again the corresponding test can be realised by a finite automaton. □

The next result shows that permutation algebras are semiautomatic. Here a permutation algebra is a domain A plus a function f such that f is a bijection. Furthermore, the domain A is assumed to be countable and this assumption applies to all structures considered in the present paper.

Theorem 11. *Every permutation algebra $(A, f; =)$ is semiautomatic.*

Proof. Let the orbit of z be the set of all z' such that there is an n with $f^n(z') = z$ or $f^n(z) = z'$. The idea is first to pick up a set X of elements $x_0, x_1, \ldots$ such that, for each z, there is exactly one x_k in its orbit and then second to represent the set X by the set Y of its indices (which is either $\mathbb{N}$ or a finite subset of $\mathbb{N}$). Now the domain is $Y \times \mathbb{Z}$ and one uses the following semiautomatic equivalence relation $=$: $conv(k, 0)$ represents x_k and $conv(k, h) = conv(k', h')$ iff $k = k'$ and $f^{|h-h'|}(x_k) = x_k$; here $|h - h'|$ is the absolute value of $h - h'$ and if one starts at x_k, then $conv(k, h) = conv(k, h')$ iff $|h - h'|$ times applying f to x_k gives x_k again. Furthermore, $f(k, h) = (k, h + 1)$. It is easy to see that f is automatic on a suitable representation of $Y \times \mathbb{Z}$ and that also every set $\{(k', h') : (k', h') = (k, h)\}$ is regular as either it is the set $\{(k, h)\}$ itself or it is the set of all $\{(k, h + \ell \cdot c) : \ell \in \mathbb{Z}\}$ for some $c \in \{1, 2, \ldots\}$. □

Theorems 9 and 11 either use relations or a single unary function. If one has both of these concepts, then the result does no longer hold. The next result below stands in contrast with Theorem 9. Note that though relations can be made semiautomatic using the technique of Theorem 9, functions cannot: the intuitive reason is that the graphs of functions over one variable are relations over two variables (one for the input and one for the output) and the semiautomaticity requirement is that the graph of the function (which is two variable relation) is automatic.

Theorem 12. *There is a recursive subset B of $\mathbb{N}$ such that the structure $(\mathbb{N}, B, Succ; =)$ is not semiautomatic. Here $Succ$ is the successor function $x \mapsto x + 1$ on the set $\mathbb{N}$ of natural numbers.*

Proof. Let B be a recursive subset of $\mathbb{N}$ which is not exponential time computable and let $Succ : \mathbb{N} \mapsto \mathbb{N}$ be the successor function from x to $x + 1$.

If the above structure $(\mathbb{N}, B, Succ; =)$ is semiautomatic, then there exists a regular domain A (representing $\mathbb{N}$), an automaton M accepting $B \subseteq A$ and a linear time computable function $f : A \to A$ representing $Succ$, where $\{conv(x, f(x)) : x \in A\}$ is regular. Let $w \in A$ represent 0. Thus, $f^n(w)$ represents n and $f^n(w)$ has length at most $(n+1) \cdot c$ for some constant c.

Now, $B(n)$ can be decided by first computing $f^n(w)$ and then checking if $M(f^n(w))$ accepts. This can be done in time polynomial in n and thus exponential in the length of the binary representation of n. This is a contradiction, as B was chosen not to be exponential time computable. Thus, the structure $(\mathbb{N}, B, Succ; =)$ cannot be semiautomatic. □

Note that the proof of the last result needs that both $Succ$ and B are there; if the structure contains only one of B and $Succ$, then it would be automatic, as these are a predicate (characteristic function of set) and a function with only one input variable and the proof does not even use whether $=$ is automatic or semiautomatic at all.

5 Groups and Order

Khoussainov, Rubin and Stephan [14, Corollary 4.4] showed (in slightly different words) that there is a semiautomatic presentation of $(\mathbb{Z}, =; +)$ in which the order of the integers is not semiautomatic. An important question left open is whether one can improve this result such that the presentation of $(\mathbb{Z}, +, =)$ used is automatic.

Question 13. *Is there an automatic presentation of the integers such that addition and equality are automatic while the set of positive integers is not regular, that is, the ordering of the integers is not semiautomatic?*

Note that a positive answer to this question would be a strengthening of the fact that the order $<$ is not first-order definable in $(\mathbb{Z}, +, =)$. This question motivates to study the connections between automatic and semiautomatic groups and order. For this, recall the definition of ordered groups.

Definition 14. A group $(G, \circ)$ is a structure with a neutral element e such that for all $x, y, z \in G$ there is a $u \in G$ satisfying $x \circ e = e \circ x = x$, $x \circ (y \circ z) = (x \circ y) \circ z$, $u \circ x = e$ and $x \circ u = e$. Such a structure without the last statements on the existence of the inverse is called a monoid. An ordered group $(G, \circ, <)$ satisfies that $<$ is transitive, antisymmetric and that all $x, y, z \in G$ with $x < y$ satisfy $x \circ z < y \circ z$ and $z \circ x < z \circ y$. If the preservation of the order holds only for operations with z from one side, then one calls the corresponding group right-ordered or left-ordered, respectively.

The first result is that in a semiautomatic group $(G, \circ; =)$ and a semiautomatic ordered group $(G, \circ; <, =)$, the relations $=$ and $<$ are indeed automatic.

Proposition 15. *Given a semiautomatic presentation $(G, \circ; =)$ of a group, the equality in this presentation is already automatic; similarly, given any semiautomatic presentation $(G, \circ; <, =)$ of an ordered group, the equality and order in this presentation are both automatic.*

Proof. Note that there are now several members of the presentation G of the group which are equal, for ease of notation one just writes still $x \in G$ in this case.

So let the semiautomatic presentation $(G, \circ; =)$ be given and let e be the neutral element. In particular the set of all representatives of e is regular. Now one can define an automatic function neg which finds for every $x \in G$ an element $neg(x) \in G$ with $x \circ neg(x) = e$. Having this function and using that $\circ$ is automatic, one has that $x = y \Leftrightarrow x \circ neg(y) = e$, hence $=$ is automatic.

Similarly, given an automatic presentation $(G, \circ; <, =)$ of an ordered group, one shows again that $=$ is automatic. Furthermore, as the set $\{u \in G : u < e\}$ is regular, one can use that $x < y$ iff $x \circ neg(y) < e$ in order to show that $<$ is also automatic. $\square$

Example 16. (a) One might ask to which extent the above proposition generalises to monoids. The answers are in general negative. For example, if one takes an irrational number r with $1 < r < 2$ such that the first n bits of the binary expansion of r cannot be computed in time polynomial in n, then the ordered monoid $(\mathbb{N} + r \cdot \mathbb{N}, +, =; <)$ is semiautomatic; here one represents $a + rb$ by $conv(a, b)$ with component-wise addition and $a + rb = a' + rb'$ holds iff $a = a' \wedge b = b'$. The order is semiautomatic as every member of the monoid bounds only finitely many smaller members.

However, there is no presentation of the above monoid where, besides addition and equality, also the order is automatic. To see this note that, using automatic functions $x \mapsto 2x$ and $x \mapsto 2x + 1$, one can compute from n, in time polynomial in n, representatives for $r \cdot 2^n$ and any desired number $m \in \{0, 1, \ldots, 2^{n+1}\}$. Using interval search and an automatic comparison, one can then find out in time polynomial in n, the natural number m which satisfies $m \leqslant 2^n \cdot r \leqslant m+1$ and therefore obtain the first n bits of the binary expansion of r. This contradicts the choice of r.

(b) The monoid $(\mathbb{N}, +; =)$ has a semiautomatic representation in which $=$ is not automatic. The idea would be to represent every member n as $conv(a, b)$, where a is a binary and b is a ternary number and $n = a + b$. Component wise addition witnesses that $+$ is automatic. However, one cannot find out when $conv(a, 0) = conv(0, b)$, as this would need to convert binary and ternary numbers into each other, which is not automatic, hence the equality is not automatic. Equality however is semiautomatic as every n is represented by only finitely many pairs.

(c) Let $x \leqslant_{lex} y$ denote that x is before y in lexicographic order. Let $x \max_{lex} y$ denote the maximum of x, y in lexicographic order. Let $(\{0, 3\}^* \cdot \{1, 2\}, \max_{lex})$ be the semigroup so formed. Note that the sets $\{3^n1, 3^n2\}$ form maximal 2-chains, as in this structure 3^n2 is the immediate successor of 3^n1 but neither does 3^n2 have an immediate successor nor 3^n1 have an immediate predecessor. Now let $I_n = \{u \in \{0, 3\}^* \cdot \{1, 2\} : 3^n2 \leqslant_{lex} u \leqslant_{lex} 3^{n+1}1\}$. Note that compressing some of the intervals I_n to points would produce longer maximal chains, for example if I_2 and I_3 are compressed to points but no other interval then one would have the maximal 4-chain made of the equivalence classes 3^21, I_2, I_3, 3^42. For each n, one creates a maximal $n+3$-chain by compressing the intervals $I_{(n+3)^2+m}$ with $m \in \{1, 2, \ldots, n, n+1\}$ iff the n-th automatic linear order from some fixed recursive enumeration of all such orders does not have a maximal $n+3$-chain. One does not compress any other intervals. The resulting structure $(\{0, 3\}^* \cdot \{1, 2\}, \max_{lex}; <, =)$ with $=$ being defined by declaring elements in compressed intervals as equal and $x \leqslant y \Leftrightarrow \exists x', y' \, [x = x' \wedge y = y' \wedge x' \leqslant_{lex} y']$ is semiautomatic; note that each compressed interval I_n is regular and each other element is represented by a singleton; furthermore, it is easy to see that the ordering is also semiautomatic. However, by construction the linear ordering is not automatic. Furthermore, also the equality is not automatic, as the ordering is first-order definable from $\max_{lex}$ and the equality. Hence one has a semiautomatic ordered semigroup $(A, \circ; <, =)$ where the semigroup operation $\circ$ is automatic while neither $(A, \circ, =)$ nor $(A, <, =)$ have an automatic presentation.

There are numerous examples of ordered automatic groups. It is clear that such groups must be torsion-free. Examples would be the additive group of integers, $\mathbb{Z}^n$ with lexicographic order on the components and pointwise addition, subgroups of the rationals generated by elements of the form x^{-k} for some fixed rational x and k ranging over $\mathbb{N}$. So it is natural to look for further examples, in particular noncommutative ones. The next result shows noncommutative automatic ordered groups do not exist; note that the result holds even if in the group below only $(G, \circ, =)$ is automatic and the ordering exists, but is not effective.

Theorem 17. *Every ordered automatic group $(G, \circ, <, =)$ is Abelian.*

Proof. Let an automatic ordered group $(G, \circ, <, =)$ be given, as the equality is automatic, one can without loss of generality assume that every element of the group is given by a unique representative in G. Nies and Thomas [21,22] showed that due to the automaticity every finitely generated subgroup $(F, \circ)$ of G satisfies that it is Abelian by finite. In particular every two elements v, w of F satisfy that there is a power n with $v^n \circ w^n = w^n \circ v^n$. Now, following arguments of Neumann [20] and Fuchs [5, page 38, Proposition 10], one argues that the group is Abelian.

In the case that $v \circ w^n \neq w^n \circ v$, consider the element $w^n \circ v \circ w^{-n} \circ v^{-1}$ which is different from e; without loss of generality $w^n \circ v \circ w^{-n} \circ v^{-1} < e$. By multiplying from both sides inductively with $w^n \circ v \circ w^{-n}$ and v^{-1}, respectively, one gets inductively the relation $(w^n \circ v \circ w^{-n})^{m+1} \circ v^{-(m+1)} < (w^n \circ v \circ w^{-n})^m \circ v^{-m} < e$ for $m = 1, 2, \ldots, n$ and by associativity and cancellation the relation $w^n \circ v^n \circ w^{-n} \circ v^{-n} < e$ can be derived. This contradicts the assumption that v^n and w^n commute and therefore $w^n \circ v^n \circ w^{-n} \circ v^{-n} = e$.

In the case that $v \circ w^n = w^n \circ v$, one again assumes that $v \circ w \circ v^{-1} \circ w^{-1} < e$ and derives that $v \circ w^n \circ v^{-1} \circ w^{-n} < e$ contradicting the assumption that v and w^n commute. Hence one can derive that any two given elements v, w in G commute and $(G, \circ)$ is an Abelian group. $\square$

Example 18. The Klein bottle group is an example of a noncommutative left-ordered group. This is the group of all $a^i b^j$ with generators a, b and the defining equality $a \circ b = b^{-1} \circ a$. One represents the group as the set of all $conv(i, j)$ with $i, j \in \mathbb{Z}$ using an automatic presentation of $(\mathbb{Z}, +, <)$. Now the group operation $a^i b^j \circ a^{i'} b^{j'}$ is given by the mapping from $conv(i, j), conv(i', j')$ to $conv(i + i', j + j')$ in the case that i' is even and to $conv(i + i', -j + j')$ in the case that i' is odd. Thus the group is automatic. The ordering on the pairs is the lexicographic ordering, that is, $a^i b^j < a^{i'} b^{j'}$ iff $i < i'$ or $i = i' \wedge j < j'$. Using some case distinction, one can show that $a^i b^j < a^{i'} b^{j'}$ iff $a \circ a^i b^j < a \circ a^{i'} b^{j'}$ iff $b \circ a^i b^j < b \circ a^{i'} b^{j'}$ and deduce from these basic relations that the group is left-ordered.

A central motivation of Question 13 is the connection between definability and automaticity of the order in groups. The next example shows that for some semiautomatic groups, the order can be first-order defined from the group operation (which is not the case with the integers). In the example one cannot have that $\circ$ is automatic, as the group is not commutative.

Theorem 19. *There is a semiautomatic noncommutative ordered group $(G, <, =; \circ)$ such that the ordering is first-order definable from the group operation.*

Proof. The group consists of all $a^i b^j c^k$ with three generators a, b, c with the defining equalities $a \circ b = b \circ a \circ c$, $a \circ c = c \circ a$ and $b \circ c = c \circ b$. Given two elements v and w, let $r(v, w)$ denote the formula that $a \circ v = v \circ a \wedge b \circ w = w \circ b \wedge v \circ b \circ v^{-1} \circ b^{-1} = a \circ w \circ a^{-1} \circ w^{-1}$; this formula says that v is of the form $a^k c^h$ and w is of the form $b^k c^\ell$ for some k, h, ℓ. In this case $v \circ w \circ v^{-1} \circ w^{-1}$ is of the form c^{k^2}. Now the formula $p(u)$ says that u is of the form c^k for some $k > 0$ by letting

$$p(u) \Leftrightarrow \exists v_1, w_1, v_2, w_2, v_3, w_3, v_4, w_4\, [r(v_1, w_1) \wedge r(v_2, w_2) \wedge r(v_3, w_3) \wedge r(v_4, w_4) \wedge u = v_1 \circ w_1 \circ v_1^{-1} \circ w_1^{-1} \circ v_2 \circ w_2 \circ v_2^{-1} \circ w_2^{-1} \circ v_3 \circ w_3 \circ v_3^{-1} \circ w_3^{-1} \circ v_4 \circ w_4 \circ v_4^{-1} \circ w_4^{-1} \circ c],$$

using that every natural number is the sum of four squares. Now one has that

$$v < w \text{ iff } p(w^{-1} \circ v \circ b^{-1} \circ v^{-1} \circ w \circ b) \vee (b \circ v^{-1} \circ w = v^{-1} \circ w \circ b \wedge p(a \circ v^{-1} \circ w \circ a^{-1} \circ w^{-1} \circ v)) \vee p(v^{-1} \circ w)$$

which is expressing the three clauses of the lexicographic order by saying that in $v^{-1} \circ w$ there are either a positive number of a or no a and a positive number of b or only a positive number of c.

Furthermore, to show that $\circ$ is semiautomatic, one notes that $a^i b^j c^k$ is represented by $conv(i, j, k)$ and now multiplication with the generators a, b, c from the right map $conv(i, j, k)$ to $conv(i + 1, j, k - j)$, $conv(i, j + 1, k)$, $conv(i, j, k + 1)$, respectively, and multiplication with the generators a, b, c from the left map $conv(i, j, k)$ to $conv(i + 1, j, k)$, $conv(i, j + 1, -i + k)$, $conv(i, j, k + 1)$, respectively. Similar laws hold for multiplication with inverses of the generators and so all multiplications with fixed elements are realised by automatic functions. Equality is automatic, as it is identity on the given elements; the lexicographic order on the triples is automatic. Furthermore, it is easy to see that the group is an ordered group using the given lexicographic order. □

Theorem 20. *The additive ordered subgroup $(\{n \cdot 6^m : n, m \in \mathbb{Z}\}, +, <)$ of the rationals has a presentation in which the addition and equality are automatic while the ordering is not semiautomatic.*

Proof. The idea is to represent group elements as $conv(a, b, c)$ representing $a+b+c$ where $a \in \mathbb{Z}$, $b = b_1 b_2 \ldots b_n \in \{0\} \cup \{0, 1\}^* \cdot \{1\}$ represents $b_1/2 + b_2/4 + \ldots + b_n/2^n$ and $c_1 c_2 \ldots c_m \in \{0\} \cup \{0, 1, 2\}^* \cdot \{1, 2\}$ represents $c_1/3 + c_2/9 + \ldots + c_m/3^m$. The representation of $\mathbb{Z}$ is chosen such that addition is automatic. Furthermore, now one adds $conv(a, b, c)$ and $conv(a', b', c')$ by choosing $conv(a'', b'', c'')$ such that the represented values satisfy $a'' = a + a' + (b + b' - b'') + (c + c' - c'')$

and $b'' \in \{b+b', b+b'-1\}$ and $c'' \in \{c+c', c+c'-1\}$ and $0 \leqslant b'' < 1$ and $0 \leqslant c'' < 1$. It can be easily seen that the resulting operation is automatic.

Assume now by way of contradiction that one could compare the fractional parts b and c of a number in order, that is, the relation $\{(b,c) : conv(0,b,0) < conv(0,0,c)\}$ would be automatic. Then one could first-order define a function f which maps every ternary string c to the length-lexicographic shortest binary string b satisfying $conv(0,0,c1) < conv(0,b,0) < conv(0,0,c2)$. There are $3^n \cdot 2$ ternary strings c of length $n+1$ not ending with a 0 representing different values between 0 and 1 and f maps these to $3^n \cdot 2$ different binary strings representing values between 0 and 1; as the resulting strings are binary, some of these values $f(c)$ must have the length at least $n \cdot \log(3)/\log(2)$. However, this contradicts the fact the length of $f(c)$ is at most a constant longer than c for all inputs c from the domain of f (as f is first-order defined from an automatic relation and thus automatic). Thus the function f cannot be automatic and therefore the ordering can also not be automatic. It follows from Proposition 15 that the order is not even semiautomatic. $\square$

Tsankov [28] showed that the structure $(\mathbb{Q}, +, =)$ is not automatic. However, one can still get the following weaker representation.

Theorem 21. *The ordered group* $(\mathbb{Q}, <, =; +)$ *of rationals is semiautomatic.*

Proof. The idea is to represent the numbers $q \in \mathbb{Q}$ as sums $a + \sum_{n \geqslant 2} \frac{b_n}{n!}$ with $a \in \mathbb{Z}$ and $b_n \in \{0, 1, \ldots, n-1\}$. Almost all b_n will be 0 and therefore each of the sums is finite. Each b_n is taken to be of the form $1^{b_n}0$ in the case that $b_n < n-1$ and of the form 1^{n-1} in the case that $b_n = n-1$. Furthermore, let the code w consist of the concatenation of the codes for $b_2, b_3, \ldots, b_m$ up to the first m such that either $m = 2$ and all $b_n = 0$ or b_m is the last non-zero b_i. In the case that $b_m = m-1$ one appends a 0 to the representation of w in order to achieve that the set of possible representations for w is the regular set $\{0\} \cup \{0,1\}^* \cdot \{10\}$.

Note that the possible values of w are ordered by lexicographic ordering, that is, $v < w$ if either w is a proper extension of v or for some u, $u0 \preceq v$ and $u1 \preceq w$; latter only happens if for some n, the values of b_i, $2 \leqslant i < n$, are same in v, w and $b_n = 1^k0$ in v and $b_n \succeq 1^{k+1}$ in w, hence the numerical value of v is below that of w. Now, a rational number $a + \sum_{n \geqslant 2} \frac{b_n}{n!}$ is represented by the convolution of a code for a and a code w for the fractional part; the representation of the integer part is chosen such that addition and ordering are automatic. Thus, to decide the ordering, one can compare the integer parts first and then, if they are equal, decide the ordering by comparing the fractional parts with lexicographic ordering.

Furthermore, for adding a number represented as above with a fixed rational, there is a constant n such that only the values for $b_2, b_3, \ldots, b_n$ change (in the case that they are there) plus perhaps there is a carry into the integer part plus a constant addition to the integer part. Hence one only needs to decode $b_2 b_3 \ldots b_n$, do the addition and replace this prefix by the updated code, while all values $b_{n+1} b_{n+2}, \ldots$ remain unchanged, though their representation might shift forward or backward by constantly many symbols due to the fact that the space needed by the first symbols might change. These ideas show that the addition in this structure is semiautomatic. $\square$

6 Rings

Ordered rings and fields are obtained by augmenting an ordered Abelian group with a multiplication which respects the corresponding laws. The formal definition is the following.

Definition 22. Given an Abelian group $(R, +)$ (perhaps with an order $<$) with 0 being the additive identity and introducing a multiplication $\cdot$, the resulting structure is a ring iff $\cdot$ is associative and satisfies the two laws of distributivity (for all $x, y, z \in R$): $x \cdot (y \cdot z) = (x \cdot y) \cdot z$, $(x + y) \cdot z = (x \cdot z) + (y \cdot z)$, $z \cdot (x + y) = (z \cdot x) + (z \cdot y)$.

Furthermore, in an ordered ring, $(R, +, <)$ has to be an ordered group and for nonzero elements $x, y \in R$, $0 < x \cdot y$ iff either $0 < x \wedge 0 < y$ or $x < 0 \wedge y < 0$.

An (ordered) field is an (ordered) ring in which $(G - \{0\}, \cdot)$ is an Abelian group.

A vector space over a field $(F, +, \cdot)$ is a group $(G, +)$ such that there is a scalar multiplication $\odot : F \times G \to G$ with $(x \cdot y) \odot z = x \odot (y \odot z)$ and $1 \odot z = z$ and $(x + y) \odot (u + z) = x \odot u + y \odot u + x \odot z + y \odot z$ for all $x, y \in F$ and $u, z \in G$. In the case that it is clear from context, $\odot$ is denoted as $\cdot$ as well.

The ring of integers $(\mathbb{Z}, +, <, =; \cdot)$ is semiautomatic, the semiautomaticity of the multiplication stems from the fact that multiplication with fixed constants can be implemented by repeated adding or subtracting the input from 0 a fixed number of times.

However, it is not always the case if for a ring $(R, +, \cdot, =)$ the group $(R, +, =)$ is automatic implies that $(R, +, =; \cdot)$ is semiautomatic. For example if one takes the complex integers $\mathbb{Z} + i \cdot \mathbb{Z}$, one could represent the real coordinate in binary and the imaginary coordinate in ternary. This group is additive, as the addition is componentwise. However, multiplication with i would convert binary numbers into ternary ones and vice versa; this is, due to the theorem of Cobham [2] and Semenov [24] impossible, as the powers of 2 on the real axis and the powers of 3 on the imaginary axis form regular sets: If the multiplication with i would be automatic then some finite automaton would recognise the powers of 2 in ternary and the powers of 3 in binary, in contrary to the Theorem of Cobham and Semenov.

For non-trivial examples of automatic rings, one can, however, augment the ring of integers with a root of a natural number and still preserve that addition and order are automatic and multiplication is semiautomatic. The first example for this is the ordered ring $(\mathbb{Z} + \frac{1+\sqrt{5}}{2} \cdot \mathbb{Z}, +, <; \cdot)$ where the ordering $<$ is the one inherited from the real numbers. Note that $(\frac{1+\sqrt{5}}{2})^2 = 1 + \frac{1+\sqrt{5}}{2}$ and therefore $\mathbb{Z} + \frac{1+\sqrt{5}}{2} \cdot \mathbb{Z}$ is closed under multiplication, so that one can denote this set as $\mathbb{Z}[u]$ for $u = \frac{1+\sqrt{5}}{2}$, as the set $\mathbb{Z} + u \cdot \mathbb{Z}$ coincides with the polynomials over u with integer coefficients. This parameter u is called the golden ratio. Note that also $1/u = u - 1$. Thus one can use negative powers in representations of the ring elements. As the ring contains the irrational number $\sqrt{5} = 2u - 1$ and compares multiples of $\sqrt{5}$ with integers, it is not directly clear that the ring has a semiautomatic presentation. Note that the technique is similar to the one used by Nies and Semukhin [23] for an automatic representation of $(\mathbb{Z}^2, +)$ in which no subgroup generated by a single element is a regular subset. Furthermore, the basics of this representation also build on well-known work on Fibonacci numbers including Tan's automatic representation of the natural numbers with addition and the predicate that a number is a Fibonacci number [25].

Theorem 23. *Let* $u = \frac{1+\sqrt{5}}{2}$. *The ordered ring* $(\mathbb{Z}[u], +, <, =; \cdot)$ *has a semiautomatic presentation.*

Proof. The set A of representatives of a ring element a consists (of a suitably ordered) sequence $a_n a_{n-1} \ldots a_m$ of integers satisfying the following conditions:

- $n \geqslant 0 \geqslant m$;
- There is a constant k such that $-k \leqslant a_i \leqslant k$ for all i;
- $a = \sum_i a_i \cdot u^i$.

One can easily see that k can be chosen to be 2, as $3u^i = u^{i-2} + u^{i+2}$ and one uses this identity in order to update the coefficients until every a_i is in $\{-2, -1, 0, 1, 2\}$; this stage is eventually reached, as every update reduces the value $\sum_i |a_i|$ by one.

Note that automatic operations and relations need that the code as above are aligned and ordered from some starting position. Thus one represents $a = \sum_i u^i a_i$ by a string

$$a_0 a_{-1} a_1 a_{-2} a_2 \ldots a_{-\max\{n,m\}} a_{\max\{n,m\}}.$$

Linear-time one-tape Turing machines processing convoluted tuples of such strings can nevertheless go to the right end, then process the code in order from least significant to the most significant, until they reach the end again and then terminate; hence automaticity is possible with this notation (note that functions computable by linear-time one-tape Turing machines, where input and output start at the same position, are automatic [1]).

Let k' be a natural number satisfying that $2k \cdot \sum_{i \leqslant 2} u^i < k'$; this constant exists due to the fact that $\sum_{i \leqslant 2} u^i = u^3/(u-1)$. The following automatic algorithm determines the sign of a representation:

1. Let $x = 0$, $y = 0$ and $i = n$;
2. If $x + y > k'$ or $x + a_i > k'$ then terminate with output "$a > 0$";
3. If $x + y < -k'$ or $x + a_i < -k'$ then terminate with output "$a < 0$";
4. Replace (x, y) by $(x + y, x + a_i)$ and i by $i - 1$;
5. If $i \geqslant m$ then go to step 2;
6. Determine the sign of a by a finite case distinction over the values of x, y which are in $\{-k', \ldots, k'\}$.

Note that (x, y) stands during the run time for $x \cdot u^{i+2} + y \cdot u^{i+1}$ and represents the sum of $\sum_{j>i} a_j \cdot u^j$. In the fourth step of the loop, this is adjusted to a linear combination of coefficients of u^{i+1} and u^i based on the identity $u^2 = u + 1$ and the fact that now a_i becomes added into the memory. For the correctness, it is easy to see that during the whole runtime of the algorithm, the variables x, y (which represent the state of the automaton) are from the constant sized range $\{-k', \ldots, k'\}$. Furthermore, assume that the algorithm terminates early at some stage i and the algorithm terminates with output $a > 0$. Then, $x \geqslant 0$, as at least one of $x+y$ and $x+a_i$ is strictly greater than k' and both y and a_i are at most k'. Note that $-k' \leqslant y \leqslant k'$ and therefore $x+y \geqslant -k'$ and $x+a_i \geqslant -k'$. If $x+y > k'$ then $x+a_i \geqslant -k$ and if $x+a_i > k'$ then $x+y \geqslant -k$. Thus $x \cdot u^{i+2} + y \cdot u^{i+1} + \sum_{j \leqslant i} a_j \cdot u^j$ will be transformed into a sum of a number greater than $k' \cdot u^i$ and $\sum_{j<i} a_j \cdot u^j - ku^i - ku^{i+1}$ which, by the choice of k', is a positive number. Thus the termination is correct and $a > 0$. Similarly one verifies that the early termination with the output $a < 0$ is correct. It follows that the whole algorithm is correct and one can automatically determine whether a number is positive or zero or negative.

For three numbers $a = \sum_i a_i \cdot u^i$, $b = \sum_i b_i \cdot u^i$ and $c = \sum_i c_i \cdot u^i$, one can check whether $a+b=c$ by forming the number $\sum_i (a_i + b_i - c_i) \cdot u^i$ and then verify using the above algorithm whether the sign of this number (where k is now 6) is indeed zero. Similarly, one can compare two numbers in order to find whether they represent the same element or one is below the other, one just has to determine the sign of $\sum_i (a_i - b_i) \cdot u^i$. It follows that $(A, +, <, =)$ is an automatic group.

Multiplication with u or u^{-1} can be realised by shifting all coefficients in the representation by one position up or down, respectively. Multiplication with a fixed integer can be realised by a constant amount of repeated additions or subtractions. Thus one can obtain multiplication with every fixed ring element by combining finitely many of these operations.

Furthermore, one can clearly represent $0, 1$ and u in A and therefore $-1, 0, 1, -u, u$ in A. As A is closed under addition and multiplication, all ring members are represented. As $u^{-1} = u - 1$, no element is represented which is not a ring member. $\square$

Remark 24. One can indeed even show that the set of representatives of natural numbers in this ring is regular. For this, one uses that $3 \cdot u^i = u^{-2+i} + u^{2+i}$. This permits to deduce the following examples: $1 = u^0$, $3 = u^2 + u^{-2}$, $7 = u^4 + u^{-4}$, $18 = u^6 + u^{-6}$, $u^{i+2} + u^{-i-2} = 3 \cdot (u^i + u^{-i}) - (u^{i-2} + u^{-i+2})$ for even $i \geqslant 2$. This permits to represent every natural number by a string $a_0 a_{-1} a_1 a_2 a_{-2} \ldots a_{-2k} a_{2k}$ such that, for all i, $a_{-i} = a_i$, $a_i \in \{0, 1, 2\}$ and $a_i = 0$ whenever i is odd. It is easy to see that the set of the above chosen representatives is regular and, as $=$ is automatic, also the set of all representatives of natural numbers is regular. Furthermore, one can see that every finite union of sets of the form $a' \cdot \mathbb{N} + b'$ with a', b' being fixed ring elements is regular. Furthermore, there is an automatic function sending any ring-element a to the convolution of its coordinates $b, c \in \mathbb{Z}$ such that $a = b + c \cdot u$.

The construction of ordered semiautomatic rings extending the integers by the irrational factor $\frac{\sqrt{5}+1}{2}$ can be generalised as follows.

Theorem 25. *Assume that $d^2 = ne^2 + 1$ and $d > 1$ for natural numbers d, e. Let $u = d + e \cdot \sqrt{n}$. The ring $(\mathbb{Z}[u], \mathbb{Z}, +, <, =; \cdot)$ is semiautomatic.*

Proof. Note that

$$u^2 = (d + e \cdot \sqrt{n})^2 = e^2 \cdot n + 2d \cdot (d + e\sqrt{n}) - d^2 = 2du - 1$$

and $u+u^{-1}=2d$. Then all powers u^i are in the ring $\mathbb{Z}+u\cdot\mathbb{Z}$. Furthermore, one can represent the ring as done in Theorem 23 as polynomials in u with coefficients from $\{-2d,-2d+1,\ldots,-1,0,1,\ldots,2d-1,2d\}$ using that $2d>2$ and therefore every replacement $2du^i\to u^{i-1}+u^{i+1}$ reduces the sum of the absolute values of the coefficients until this sum is minimal. Writing the coefficients as a string of the form $a_0a_1a_{-1}a_2a_{-2}\ldots a_na_{-n}$ permits to identify the integers as those numbers which are equal to a number where $a_m=a_{-m}$ for all m: This can be seen by induction as one can apply a normalisation $2du^i\to u^{i-1}+u^{i+1}$ exactly at i when one does it at $-i$ and so maintain that each integer has a symmetric representation; furthermore, one can show by induction that each symmetric representation stands for an integer: $u^{i+2}+u^{-i-2}$ represents the same integer as $2du^{i+1}-u^i-u^{-i}+2du^{-i-1}$ which represents the difference of $2d$ times the integer represented by $u^{i+1}+u^{-i-1}$ and the integer represented by u^i+u^{-i}.

The order on the representatives is decided by a method similar to the one in Theorem 23 and it is automatic: One subtracts the two numbers by component-wise subtraction of the coefficients and then tests whether the result is positive, 0 or negative; so assume that all coefficients are from $\{-k,\ldots,-1,0,1,\ldots,k\}$ and that k' is so large that $(u-1)\cdot k'>2dku+k\cdot\sum_{i\leqslant 0}u^i$. Then one starts with $(x,y)=0$ and i being the maximum power with non-zero coefficient and replaces (x,y) by $(2dx+y,a_i-x)$ and i by $i-1$ until it happens that the absolute value of x or y is above k' or the last non-zero coefficient is processed.

Assume that $0<x\leqslant k'$ and $-k'\leqslant y\leqslant k'$ and the algorithm terminates in next step. If $a_i-x<-k'$ then $x\geqslant k'-k$ and $2dx+y\geqslant(2d-1)k'-2dk$ and $(2dx+y)u^{i+1}+(a_i-x)u^i+\sum_{j<i}a_j\cdot u^j\geqslant(2d-1)k'u^{i+1}-2du^{i+1}k-k'u^i-k\sum_{j\leqslant i}u^i\geqslant k'u^{i+1}-2du^{i+1}k-k'u^i-k\sum_{j\leqslant i}u^i\geqslant(u-1)k'u^i-2du^{i+1}k-k\sum_{j\leqslant i}u^j>0$, hence the number is positive. If $a_i-x\geqslant-k'$ and $2dx+y>k'$ then $(2dx+y)u^{i+1}+(a_i-x)u^i+\sum_{j<i}a_ju^j\geqslant k'u^{i+1}-k'u^i+\sum_{j<i}a_ju^j\geqslant(u-1)k'\cdot u^i-2dk\cdot u^{i+1}-k\sum_{j\leqslant i}u^j>0$, hence again the number tested is positive. These are the only cases where the algorithm terminates early for positive x. Similarly, if the algorithm terminates early for negative x then the number tested is negative. In the remaining cases, the variables (x,y) remain between $-k'$ and k' during the run-time of the algorithm and the value can be decided by a look-up in a table after the last non-zero coefficient has been processed.

The addition is automatic, as for three ring elements $\sum_i a_iu^i,\sum_i b_iu^i,\sum_i c_iu^i$, one can check whether $\sum_i(a_i+b_i-c_i)u^i$ represents the 0. Multiplication with powers of u can be realised by making a shift; multiplication with integers can be realised as repeated addition; hence multiplication with constant ring elements is automatic. $\square$

Theorem 26. *The ring $(\mathbb{Z}(\sqrt{n}),\mathbb{Z},+,<,=;\cdot)$ has for every positive natural number n a semiautomatic presentation.*

Proof. Without loss of generality, n is not a square number; the case where n is square is trivial, as it just says that the ordered ring of integers is semiautomatic. Now consider the equation $d^2=ne^2+1$; this equation is known as "Pell's Equation" and it is known that for each integer n which is not a square there are infinitely many integer pairs (d,e) with $d^2=ne^2+1$. Euler named these equations (with parameter n) after Pell and for some small n, Pell's equation was solved independently by Indian and Greek mathematicians more than 2000 years ago; later

Brahmagupta in India found a method to solve several more of these equations and Bhaskara II solved it for all non-square n. In Europe, Lagrange [16] solved the general problem first and showed that there are infinitely many solutions in the case that n is not a square. For square numbers n the only integer solution is $d = 1$ and $e = 0$, as 1 and 0 are the only integer squares with difference 1.

Given n, one can now choose d, e as above with $d > 1$. By Theorem 25 the structure $(\mathbb{Z} + u \cdot \mathbb{Z}, \mathbb{Z}, +, <, =; \cdot)$ with $u = d + e \cdot \sqrt{n}$ has an automatic presentation. Now the subset $A = \mathbb{Z} \cdot (u - d) + \mathbb{Z} \cdot e$ is a regular subset of this structure, as

$$z \in A \Leftrightarrow \exists x, y \in \mathbb{Z}\,[z = (u - d) \cdot x + e \cdot y]$$

and addition and multiplication with constants is automatic. Note that

$$A = \sqrt{n} \cdot e \cdot \mathbb{Z} + e \cdot \mathbb{Z}$$

and that this ordered additive group is — by the homomorphism $z \mapsto z/e$ — isomorphic to the ordered additive group $B = \sqrt{n} \cdot \mathbb{Z} + \mathbb{Z}$ and so one can use the element x in A to represent the element x/e of B. Furthermore, one can make B to be a ring noting that the multiplication in $(A, +, <; \cdot)$ permits through the formula $x, c \mapsto x \cdot c/e$ to define the multiplication in B. The reason is that $(x/e) \cdot (c/e) = ((x \cdot c)/e)/e$ and so one has to adjust a product in A by dividing through e in order to get the corresponding product in B. Thus, for each constant c, the multiplication with c is an automatic function in A and thus the multiplication with constants is automatic in B. The integers in B are equal to $A \cap \mathbb{Z}$, thus they form there a regular subset of B as well. $\square$

Remark 27. Note that in the above representation, the set

$$\{conv(x, y) : x, y \in \mathbb{Z} \wedge x > 0 \wedge y < \sqrt{n} \cdot x < y + 1\}$$

is regular; this permits to use finite automata to find arbitrary close approximations of $\sqrt{n}$ by quotients of natural numbers.

The next result deals with noncommutative rings where the multiplication is not commutative and where, to simplify the proof, a 1 does not need to exist.

Theorem 28. *There is a ring $(R, +, =, \cdot)$ such that $(R, +, =)$ is an automatic group and the family of functions $\{y \mapsto y \cdot x : x \in R\}$ is semiautomatic while every function $y \mapsto x \cdot y$ with $x \in R$ fixed is either constant 0 or not automatic (independent of the automatic representation chosen for the ring).*

Proof. One is considering a ring over the Boolean field $(\{0, 1\}, +, \cdot)$ where the elements of the ring are given as repetition-free sums over generators x_i; the empty sum stands for the 0 in the ring and each element is self-inverse for the addition. Furthermore, let f be a function from $\mathbb{N}$ to $\mathbb{N}$ to be defined later. Now one defines the multiplication using the equalities $x_i \cdot x_j = x_{f(j)}$

and the law of distributivity. The function f is a permutation on the natural numbers such that the length of each orbit is a prime number and the k-th prime number p_k occurs as a length of an orbit iff k is an element of some fixed undecidable set K.

If the ring would have an automatic presentation such that each mapping $x \mapsto y \cdot x$ is automatic, then one could fix y as the generator x_0 and would have that k is in the set K iff the function $x \mapsto x_0 \cdot x$ has an orbit of length p_k. As the first-order theory of this function is decidable, K would be decidable in contradiction to the assumption.

Now a presentation of the ring is given in which addition and equality are automatic, as well as multiplication with a fixed second element is automatic. For this, one represents every ring element by a binary string $a_0a_1 \ldots a_n$ representing $\sum_{m \leqslant n} a_m \cdot x_m$ and says that two such representations are the same if they only differ by trailing zeroes, hence equality is automatic. Furthermore, addition of two such strings is realised by bit-wise exclusive or, where the result has the length of the longer string (with zeroes added into the missing positions of the shorter string). Now, consider multiplication with a fixed element, say $x_3+x_8+x_9$, in the second position and let y be the sum of k generators. If k is even then $y \cdot (x_3 + x_8 + x_9) = 0$. If k is odd then $y \cdot (x_3 + x_8 + x_9) = x_{f(3)} + x_{f(8)} + x_{f(9)}$. Hence the structure $(R, +, =; \{y \mapsto y \cdot x : x \in R\})$ is semiautomatic. $\square$

7 Fields and Vector Spaces

In the following, let $(A, +, <, =; \cdot)$ be a semiautomatic ordered ring. Note that such a ring is an integral domain, as given two nonzero factors v, w, one can (after multiplication with -1 when needed) assume that $0 < v$ and $0 < w$; then it follows that $0 < v \cdot w$ and therefore $v \cdot w$ differs from 0. Hence the quotient field is always defined.

Theorem 29. *If $(A, +, <, =; \cdot)$ is a semiautomatic ordered ring then the unique quotient field F defined by the ring is an ordered semiautomatic field $(F; +, \cdot, <, =)$.*

Proof. The members of F are of the form $\frac{a}{b}$ with $a, b \in A$ and $0 < b$; they are represented by $conv(a, b)$ but for convenience in the following always written as $\frac{a}{b}$. Let $\frac{a'}{b'}$ be a fixed element of F and consider adding, multiplying and comparing with $\frac{a'}{b'}$:

- The addition $\frac{a}{b} \mapsto \frac{a \cdot b' + a' \cdot b}{b \cdot b'}$ is automatic, as multiplication with fixed ring elements a', b' is automatic and adding of ring elements is also automatic;
- The multiplication $\frac{a}{b} \mapsto \frac{a \cdot a'}{b \cdot b'}$ is automatic, for the same reasons as addition;
- The set $\{\frac{a}{b} : a \cdot b' < a' \cdot b\}$ of all representatives of members of F less than $\frac{a'}{b'}$ is regular;
- The set $\{\frac{a}{b} : a \cdot b' = a' \cdot b\}$ of all representatives of $\frac{a'}{b'}$ is regular.

Hence $(F; +, \cdot, <, =)$ is semiautomatic; the verification that the resulting structure is an ordered field follows the verification that the rationals are an ordered field when constructed from the ring of integers, this verification is left to the reader. $\square$

Corollary 30. *The ordered field $(\mathbb{Q}; +, \cdot, <, =)$ of the rationals and, for all $n \in \mathbb{N}$, the extensions $(\mathbb{Q}(\sqrt{n}); +, \cdot, <, =)$ are semiautomatic.*

Does it make a difference whether one writes $\frac{7}{3}$ or $2 + \frac{1}{3}$? The advantage of the latter is that one can directly see whether it is an integer. To see that $\frac{15133}{123} = 123 + \frac{6}{123}$ is not an integer requires already some calculations; thus one would have to choose the representation as on the right side of the equality in order to obtain a semiautomatic structure. This will then be used for Theorem 31 below.

Theorem 31 will actually give a more general result: One can even handle vectors of rational numbers as long as one maintains a common denominator for the fractional parts. For formalising this, one introduces the semilinear functions; these are called this way, as they generalise the linear functions with some additional constructs and are then closed under composition. A function $F : \mathbb{Q}^n \mapsto \mathbb{Q}^n$ is called semilinear iff there is a constant $m \geqslant n$ such that the function can be computed by program carrying out the following types of operations on vectors from $\mathbb{Q}^m$ (where coordinates with index $n+1, n+2, \ldots, m$ are initialised at 0 before the program starts and are ignored when the program terminates):

- Testing whether a coordinate satisfies some condition (is equal to a constant, less or equal than a constant or is an integer) and branching forward (not backward) if the condition is true;
- Updating a coordinate as a linear combination or the down-rounded value of the linear combination of the old values of the coordinates before doing the instruction.

Here a linear combination is the sum of constants multiplied with coordinates and of constants, for example $3x_1 + \frac{1}{5}x_2 + 8$ is a linear combination. Note that all the semilinar functions are functions from $\mathbb{Q}^n$ to $\mathbb{Q}^n$ and not from $\mathbb{Q}^n \times \mathbb{Q}^n$ to $\mathbb{Q}^n$, thus a semilinear function (or the full class of semilinear functions) is automatic iff it is semiautomatic. An example is given by the following program in which x_1, x_2 are the input / output coordinates and x_3, x_4, x_5 are auxiliary variables.

1. Function $F(x_1, x_2)$
2. $x_3 = 2x_1 + 3x_2$;
3. $x_4 = 3x_1 - 2x_2$;
4. $x_5 = round(x_1)$;
5. If $x_5 = 2$ then goto 9;
6. $x_1 = x_3$;
7. $x_2 = x_4$;
8. Goto 11;
9. $x_1 = x_4$;
10. $x_2 = x_3$;
11. End of Program.

So if $2 \leqslant x_1 < 3$ then (x_1, x_2) is mapped to $(3x_1 - 2x_2, 2x_1 + 3x_2)$ else (x_1, x_2) is mapped to $(2x_1 + 3x_2, 3x_1 - 2x_2)$.

Theorem 31. *Let $\mathcal{F}$ denote the semilinear functions. Then, for every constant n, the structure $(\mathbb{Q}^n, \mathcal{F}; +, \cdot, =, <)$ is semiautomatic, where $<$ refers to componentwise partial order.*

Proof. In the following, it is shown how the coordinates are updated for each operation. The data type consists of convoluted tuples $(a_1, b_1, a_2, b_2, \ldots, a_m, b_m, c)$ representing the numbers $a_1 + \frac{b_1}{c}$, $a_2 + \frac{b_2}{c}$, …, $a_m + \frac{b_m}{c}$ where $a_k, b_k, c \in \mathbb{Z}$ and $0 \leqslant b_k < c$ for $k = 1, 2, \ldots, m$ and m is the maximum coordinate which is used in the program; the values in the tuples are represented in any representation of the integers which permits adding, subtracting, multiplying with constants and comparing. Furthermore, one can assume that each linear mapping is computed by basic operations step by step, in order to simplify the proof. The operations are now as follows.

- Updating $a_i + \frac{b_i}{c} = a_i + \frac{b_i}{c} + a_j + \frac{b_j}{c}$: If $b_i + b_j \geqslant c$ then the new value of a_i is $a_i + a_j + 1$ and the new value of b_i is $b_i + b_j - c$ else the new value of a_i is $a_i + a_j$ and that of b_i is $b_i + b_j$.
- Multiplying $a_i + \frac{b_i}{c}$ with -1: If $b_i = 0$ then the new value of a_i is $-a_i$ and b_i remains unchanged else the new value of a_i is $-a_i - 1$ and the new value of b_i is $c - b_i$.
- Multiplying $a_i + \frac{b_i}{c}$ with constant integer $d \geqslant 1$: One determines by constantly many case distinctions the $d' \in \{0, 1, \ldots, d-1\}$ with $d'c \leqslant db_i < (d'+1)c$ and updates a_i to $a_i \cdot d + d'$ and b_i to $db_i - d'c$.
- Dividing $a_i + \frac{b_i}{c}$ by integer $d \geqslant 2$ (= multiplying with $\frac{1}{d}$): Let d' be the remainder and e' be the quotient when a_i is divided by d; so $d', e' \in \mathbb{Z}$, $a_i = de' + d'$ and $0 \leqslant d' < d$. Now one updates b_i to $b_i + d'c$ and a_i to e'; furthermore, for $j \neq i$ one leaves a_j unchanged and updates b_j to $b_j d$ and one updates c to cd.
- Adding integer constant d to a_i: This is done by updating $a_i = a_i + d$ and leaving everything else unchanged.
- Adding $\frac{1}{d}$ to a_i for integer $d \geqslant 2$: If $b_i d + c \geqslant cd$ then update a_i to $a_i + 1$ and b_i to $b_i d + c - cd$ else leave a_i unchanged and update b_i to $b_i d + c$. For all $j \neq i$, update b_j to $b_j d$ and leave a_j unchanged. Update c to cd.
- Rounding $a_i + \frac{b_i}{c}$: Let $b_i = 0$.
- Checking whether $a_i + \frac{b_i}{c}$ is an integer: Check whether $b_i = 0$.
- Checking whether $a_i + \frac{b_i}{c} \geqslant 0$: Check whether $a_i \geqslant 0$.
- Checking whether $a_i + \frac{b_i}{c} = 0$: Check whether both a_i, b_i are 0.

Using additional coordinates permits to store intermediate values and doing additions repeatedly permits to multiply with constants. For example, adding $5\frac{2}{3}$ could be broken down into adding 5, $\frac{1}{3}$ and again $\frac{1}{3}$ in the above scheme. One could create another coordinate for the target line number of jumps and then, in each command, ignore the command in the case that the target line number is larger than the current line number. Thus one can make the semilinear function a concatenation of automatic functions and this permits to make the whole function automatic.

Note that comparing of vectors with constant vectors, adding a constant vector to a vector, multiplying a vector with a constant vector and component-wise comparison with a constant vector can all be realised by semilinear programs and thus the canonical operations on the n-dimensional vector space are semiautomatic. ☐

Remark 32. Note that expanded decimals with periods (coded by overlined digits) do not give a semiautomatic presentation for the rationals. The main reason is that when adding $1.\overline{01}$ and $2.58\overline{002}$ the period gets longer and the result is $3.59\overline{012103}$. In general, when adding the fixed number $1.\overline{01}$ to a rational $1.\overline{(00)^k1}$ with period of length $2k+1$, then the resulting number is $2.\overline{(01)^k11(01)^{k-1}02}$, and thus the period has length $4k+2$ and the length of the representation doubles. Thus the addition with the fixed value $1.\overline{01}$ is not automatic in this structure and there is no easy way to overcome the problem in this type of coding.

Theorem 33. *If $(A, +, <, =; \cdot)$ is a semiautomatic ordered ring then every finite-dimensional vector space $(F^n; +, \cdot, =)$ defined from the quotient field F of the ring A has a semiautomatic representation and all linear mappings from F^n to F^n are automatic. In particular, finite algebraic extensions $(G; +, \cdot, =)$ of the field $(F, +, \cdot, =)$ are semiautomatic.*

Proof. The idea is to represent the members of the vector space by $k+1$-tuples $conv(a_1, a_2, \ldots, a_k, b)$ of members of A representing the vector $(\frac{a_1}{b}, \frac{a_2}{b}, \ldots, \frac{a_k}{b})$. Now an addition with a fixed vector $(\frac{a'_1}{b'}, \frac{a'_2}{b'}, \ldots, \frac{a'_k}{b'})$ is realised by mapping $conv(a_1, a_2, \ldots, a_k, b)$ to $(a_1 \cdot b' + b \cdot a'_1, a_2 \cdot b' + b \cdot a'_2, \ldots, a_k \cdot b' + b \cdot a'_k, b \cdot b')$ which is an automatic function as it only involved multiplication with fixed ring members $a'_1, a'_2, \ldots, a'_k, b'$, respectively, and addition. Similarly, one can do multiplication with a fixed scalar $\frac{a'}{b'}$ by multiplying all components of the vector with the corresponding constants. In the same way one can automatically compare any fixed vector for equality with elements of the vector space. For realising linear mappings, one has first to make sure that all the entries in the matrix to be multiplied with and the vector to be added afterwards have the same fixed denominator b'; furthermore, the numerators are denoted as $a'_{i,j}$ for the matrix elements and c'_i for the fixed vector involved in the linear mapping. Now $conv(a_1, a_2, \ldots, a_k, b)$ is mapped to a vector with the i-th component being $\sum_j a'_{i,j} \cdot a_j + c'_i \cdot b$ and the denominator component being $b \cdot b'$. The verification is straightforward.

Suppose the finite algebraic extension G of the F is defined from the ring A using an irreducible polynomial $c_0 + c_1x + c_2x^2 + \ldots c_{k-1}x^{k-1} + x^k$ which is set to 0. Then one can represent G as a k-dimensional vector space over F where the components stand for the coefficients of linear combinations of $1, x, x^2, \ldots, x^{k-1}$ in the field. Now every multiplication with a fixed element is a linear mapping inside the vector space using that $x^k = -(c_0 + c_1x + c_2x^2 + \ldots c_{k-1}x^{k-1})$ and hence every finite algebraic extension $(G; +, \cdot, =)$ of the field F is semiautomatic. $\square$

Note that the above result can be extended to semilinear functions except that one has to sacrifice the rounding in the case of the structure $(\mathbb{Q}^n[\sqrt{m}]; +, \cdot, <, =)$. So the commands allowed would be the following:

- A conditional or unconditional jump forward where the allowed test for the conditional jump is whether a linear combination of some variables is positive or is equal to 0;
- Replacing a variable by a linear combination of its old value and the values of other variables.

The following result shows, also in the case of the ring of rationals, that one cannot augment the programs for the semilinear functions by any of the following operations: multiplying two coordinates, mapping one coordinate to its mulitplicative inverse, forming the square of a coordinate.

Theorem 34. *Assume that $\mathcal{G}$ are the augmented semilinear functions from $\mathbb{Q}^n$ to $\mathbb{Q}^n$ where the augmented operation is any of multiplying coordinates with each other, squaring coordinates or* *taking multiplicative inverse of coordinates. Then the structure* $(\mathbb{Q}^n, \mathcal{G}; =)$ *is not semiautomatic for any fixed* $n \geqslant 1$.

Proof. The proof is based on the following general schema. Recall that one of the many equivalent definitions of a Diophantine set A is that there is a polynomial $p(x_1, \ldots, x_k)$ with integer coefficients such that

$$A = \{y \in \mathbb{N} : \exists x_1, \ldots, x_k \in \mathbb{N}\, [y = p(x_1, \ldots, x_k)]\}$$

If one uses augmented semilinear functions of any of the three types, then for any Diophantine set A of the integers, one can define a semilinear function f whose range is $B = \{(k, 0, \ldots, 0) : k \in A\}$. Assuming now by way of contradiction that f is automatic, one can use that there is also an automatic function g which realises the operation which subtracts one from the first coordinate. Thus one has for each $k \in A$ the formula

$$k \in A \Leftrightarrow \exists u \in \mathbb{Q}^n\, [g^k(f(u)) = (0, 0, \ldots, 0)]$$

which can be evaluated by the decision algorithm of the Theorem of Khoussainov and Nerode and the fact that f, g are automatic functions and that the set of representatives of $(0, 0, \ldots, 0)$ is a fixed regular subset of the domain of the semiautomatic structure. Thus the assumption that f is automatic contradicts the fact that there are nonrecursive Diophantine sets by the result of Matiyasevich [17,18].

For the main part of the proof, let a nonrecursive Diophantine set A be given and $p(x_1, \ldots, x_k)$ be the polynomial defining A as indicated above. The idea of the proof is now that one can decode one coordinate z_1 into the parameters $x_1, x_2, \ldots, x_k$ of the Diophantine set and afterwards evaluate the polynomial. The coding chosen is that one chooses a y_1 as a natural number strictly greater than each of the natural numbers $1, x_1, x_2, \ldots, x_k$ coded and z_1 is a rational satisfying

$$z_1 = y_1 + \frac{x_1}{y_1} + \frac{x_2}{y_1^2} + \frac{x_3}{y_1^3} + \ldots + \frac{x_k}{y_1^k}$$

Similarly, given a rational input z_1, one can decode $y_1, x_1, \ldots, x_k$ from it and assign some arbitrary integers $y_1, x_1, \ldots, x_k$ satisfying $y_1 \geqslant 2$ and $x_1, \ldots, x_k \in \{0, 1, \ldots, y_1 - 1\}$ in the case that z_1 is not of the given form.

Now one assumes that $z_1, \ldots, z_n$ are the main coordinates holding input and output and that $x_1, \ldots, x_k$ are auxiliary variables holding the values of the polynomial and $y_1, \ldots, y_m$ are other auxiliary variables. The computation of f can now be done as below, where first y_1 is determined, then the variables $x_1, \ldots, x_k$ are calculated and afterwards the polynomial is evaluated; the output of the polynomial is in coordinate z_1. The program for f uses the sample polynomial $p(x_1, x_2, x_3) = 8x_1^2 - 12x_1x_2 + 26x_3$; the Diophantine set defined in this example is the set of all even numbers, as $2h = 8 \cdot 1^2 - 12 \cdot 1 \cdot (2h + 18) + 26 \cdot (h + 8)$ for all h.

1. Function $f(z_1, \ldots, z_n)$ defining Diophantine set A;
2. Comment: Computing $x_1, x_2, \ldots, x_k$ from z_1; $k = 3$ in this example;

3. $y_1 = round(z_1)$;
4. If $y_1 > 2$ then goto 6;
5. $y_1 = 2$;
6. $y_2 = z_1$;
7. $y_2 = (y_2 - round(y_2)) \cdot y_1$;
8. $x_1 = round(y_2)$;
9. $y_2 = (y_2 - round(y_2)) \cdot y_1$;
10. $x_2 = round(y_2)$;
11. $y_2 = (y_2 - round(y_2)) \cdot y_1$;
12. $x_3 = round(y_2)$;
13. Comment: Setting $z_1, \ldots, z_n$ to 0; here assuming $n = 2$;
14. $z_1 = 0$;
15. $z_2 = 0$;
16. Comment: The next steps compute the polynomial value in z_1; in this example, the polynomial is $p(x_1, x_2, x_3) = 8x_1^2 - 12x_1x_2 + 26x_3$;
17. $y_3 = 8$;
18. $y_3 = y_3 \cdot x_1$;
19. $y_3 = y_3 \cdot x_1$;
20. $z_1 = z_1 + y_3$;
21. $y_3 = -12$;
22. $y_3 = y_3 \cdot x_1$;
23. $y_3 = y_3 \cdot x_2$;
24. $z_1 = z_1 + y_3$;
25. $y_3 = 26$;
26. $y_3 = y_3 \cdot x_3$;
27. $z_1 = z_1 + y_3$;
28. Comment: End of Program; Now if $z_1 \in \mathbb{N}$ then $z_1 \in A$.

One can see that a natural number is in the range of f iff there are $x_1, x_2, \ldots, x_k \in \mathbb{N}$ such that the polynomial maps $(x_1, \ldots, x_k)$ to this number.

For the cases of squares and multiplicative inverses, one can use them to express multiplication and thus adjust the above program accordingly. As

$$a \cdot b = 0.25 \cdot ((a + b)^2 - (a - b)^2)$$

and multiplication with rational constants is in the basic framework of semilinear functions, squaring allows directly to implement the multiplication. For the multiplicative inverse, one uses the formula

$$a^2 = \frac{1}{\frac{1}{a} - \frac{1}{a+1}} - a$$

which works for all a except for $a = 0$ and $a = 1$; these cases can be separated out using conditional branching. This permits to implement the multiplication indirectly in the case of the two other types of augmented semilinear functions. □

The following questions are currently open with respect to fields. In particular there is still a lack of methods to show that certain structures are not semiautomatic.

Question 35. **(a)** *Are the structures* $(\mathbb{Q}, <, =; +, \cdot)$ *or* $(\mathbb{Q}, =; +, \cdot)$ *semiautomatic? In other words, is it really needed, as done in the above default representations, that the equality and the order are not automatic?*
(b) *Is the polynomial ring* $(\mathbb{Q}[x]; +, \cdot, =)$ *semiautomatic?*
(c) *Is there a transcendental field extension of the rationals which is semiautomatic?*
(d) *Is* $(\mathbb{Q}, \mathbb{Z}; +, -, \cdot, /, <, =)$ *semiautomatic?*

In Question 35 (d), note that one can realise $(\mathbb{Q}, \mathbb{Z}; +, -, \cdot, <, =)$, so everything without the division. Furthermore, if one does not have $\mathbb{Z}$ as a subset, one can represent the rationals as $\frac{a}{b}$ with $b \geqslant 1$; here one can now form the inverse by just switching a, b and, if needed, multiplying both of them with -1; therefore division can also be realised. However, the integer test is not possible in this representation. So the question mainly asks whether there is a structure combining the advantages of both and this is unknown, most likely the answer is "no". The counterpart of Questions 35 (b) and (c) for finite fields has a positive answer.

Theorem 36. *Let* $(F, +, \cdot)$ *be a finite field. Then the following structures are semiautomatic:*

- *Every (possibly infinite) algebraic extension* $(G, +, =; \cdot)$ *of the field;*
- *The polynomial rings* $(F[x], +, =; \cdot)$ *in one variable and* $(F[x, y]; +, \cdot, =)$ *in two or more variables;*
- *The field of fractions* $(\{\frac{a}{b} : a, b \in F[x] \wedge b \neq 0\}; +, \cdot, =)$ *over the polynomial ring with one variable.*

Proof. For the first statement, let $F_0 = F$ as the starting finite field and describe the algebraic extension G by a sequence $F_1, F_2, \ldots$ of finite fields such that every field F_{n+1} is a finite algebraic extension and finite-dimensional vector space over F_n. The members of G are considered to be almost everywhere 0 functions from $\mathbb{N}$ to F_0; these can be represented by finite strings of members of F_0 (used as an alphabet) with component-wise addition; trailing zeroes are omitted and the empty string represents the everywhere 0 function. In this vector space, one groups the vector space into groups of $dim(F_n/F_0)$ entries so that each string in the representation consists of blocks of length $dim(F_n/F_0)$ of symbols representing one element in F_n, the last non-zero element might be represented by a string shorter than $dim(F_n/F_0)$ where the missing components are interpreted as zeroes. Thus the strings represent at the same time members of vectors over each F_n. One can represent a multiplication with a fixed element of F_n as a mapping which changes each block of length $dim(F_n/F_0)$ according to a fixed table, omitting trailing zeroes from the result. As a multiplication with an element from F_n is indeed a scalar multiplication in the field viewed as a vector space over F_n, it follows that the multiplication in the resulting full field G is semiautomatic.

For the second statement, one represents $F[x]$ as strings over F where a non-empty string $a_0 a_1 \ldots a_n$ represents $\sum_{m=0,\ldots,n} a_m \cdot x^m$ and representations are equal if they only differ by trailing zeroes. Addition is pointwise addition of F. For multiplication, one sees that multiplication with

x^n is just realised by the mapping $w \mapsto 0^n w$ which is automatic and multiplication with a fixed element can be realised by adding up a fixed number of such shifted versions of w. Thus $(F[x], +, =; \cdot)$ is semiautomatic.

For the second part of second statement, the semiautomaticity of the polynomial ring over several variables, the result follows from the more general result given in Theorem 38 below.

For the third statment, the field of fractions $(\mathbb{Q}; +, \cdot, =)$ of $(F[x], +, =; \cdot)$ is semiautomatic by the same arguments as in Theorem 29; here one only has to adjust the arguments relating the order of the field (which does not exist here) by the weaker property that $a, b \neq 0 \Rightarrow a \cdot b \neq 0$ for members $a, b \in F[x]$, which follows the usual standard arguments of constructing a field from a ring. □

Example 37. Let $(U, \times, <)$ be an ordered monoid with $U = \{u_0, u_1, \ldots\}$ being countable and $u_i < u_j \Leftrightarrow i < j$ and $u_i < u_j \Rightarrow u_i \times u_k < u_j \times u_k \wedge u_k \times u_i < u_k \times u_j$ for all i, j, k and u_0 being the neutral element ($u_0 \times u_i = u_i \times u_0 = u_i$ for all i). For a finite field $(F, +, \cdot)$, let a string $a_0 a_1 \ldots a_n$ in F^* with $a_n \neq 0$ represent the polynomial

$$\sum_{m \leqslant n} a_m \cdot u_m$$

of degree n and let the empty string ε represent 0 and ε having the degree -1. Addition of such polynomials is defined with component-wise addition where one first appends trailing zeroes to make the representatives having the same length and then removes the trailing zeroes from the result polynomial. For non-empty strings $a = a_0 a_1 \ldots a_n$ and $b = b_0 b_1 \ldots b_m$, one defines the product

$$(\sum_{i \leqslant n} a_i \cdot u_i) \cdot (\sum_{j \leqslant m} b_j \cdot u_j) = \sum_{i \leqslant n, j \leqslant m} a_i \cdot b_j \cdot u_i \cdot u_j$$

and notes that the degree of this product is the degree of $u_n \cdot u_m$: If $u_i < u_n$ and $u_j < u_m$ then $u_i \cdot u_j < u_i \cdot u_m < u_n \cdot u_m$ and $u_n \cdot u_j < u_n \cdot u_m$; furthermore, it follows from $a_n \neq 0 \wedge b_m \neq 0$ that the term $a_n \cdot b_m \cdot u_n \cdot u_m$ does not vanish. Thus, for all non-zero polynomials a and b, the degree of $a \cdot b$ is at least the maximum of the degrees of a and of b. One can also show that this structure is a ring. Let $(F[U], +, \cdot)$ denote this ring.

This structure is quite general. For example, one can choose U such that the ring $(F[U], +, \cdot)$ is the polynomial ring over F with a given number of variables. In the case that one wants to represent infinitely many variables, one takes $u_i \times u_j = u_{i \cdot j + i + j}$ and the k-th variable is represented by u_{p-1} where p is the k-th prime number. Furthermore, for two variables, one can consider ordered pairs $\langle i, j \rangle = (i + j) \cdot (i + j + 1)/2 + i$ and let $u_{\langle i, j \rangle} \times u_{\langle i', j' \rangle} = u_{\langle i+i', j+j' \rangle}$ with $u_{\langle i, j \rangle}$ representing the monomial $x^i y^j$. Another structure which can be realised this way is the free associative unital algebra over F in at most countably many variables.

Theorem 38. *The ring $(F[U]; +, \cdot, =)$ is semiautomatic for F, U from Example 37. In particular, polynomial rings over F with at most countably many variables and the free associative algebra over F with at most countably many variables are semiautomatic in this sense.*

Proof. One first introduces base representations of the members of $F[U]$ by strings over F ending with a non-zero element. Furthermore, let $\oplus$, $\otimes$ and $\odot$ be three symbols outside F. Now one assigns to every string w over $\{F, \oplus, \otimes, \odot\}^*$ a value $val(w)$ as follows:

- $val(\varepsilon)$ is 0, which is represented by ε;
- For $w \in F^*$, $val(w)$ is the base element v of $F[U]$ obtained by omittting trailing zeroes from w;
- $val(w \otimes \varepsilon)$ and $val(w \odot \varepsilon)$ are both 0, which is represented by ε;
- $val(w \otimes v) = val(w) \cdot val(v)$ in the case that $v \in F^*$ and v does not represent 0;
- $val(w \odot v) = val(v) \cdot val(w)$ in the case that $v \in F^*$ and v does not represent 0;
- $val(w \oplus \varepsilon) = val(w)$;
- $val(w \oplus v) = val(w) + val(v)$ in the case that $v \in F^*$ and the degree of v and w differs;
- $val(w \oplus v) = u_n$ in the case that $v \in F^*$ and v does not represent 0 and the degree of v and w is both n and $n \geqslant 0$.

Note that the function val itself is not automatic, however the function val_n is automatic where $val_n(w) = val(w)$ in the case that w represents a polynomial of degree n or less and $val_n(w) = @$ in the case that w represents a polynomial of degree above n. The reason for the automaticity is that the function can go from the start to the end over the expression and whenever the value becomes @, it remains this value unless there is a multiplication with 0 (coded as ε); thus the automaton handles only finitely many possible values and keeps in the memory one operand consisting of the value of the processed part and another one of the next value to be taken into account. These values are updated according to the operands given. Note that the nonstandard feature in the last condition of the definition of val was explicitly chosen this way in order to make the functions val_n all automatic.

Hence one can check whether a representative w has a fixed value v by evaluating $val_{deg(v)}(w)$ and comparing the result with v. One can do the addition with a fixed value v as follows. If $val_{deg(v)}(w) \neq @$ then $v + w$ is represented by the corresponding base element for that value else $v + w$ is $w \oplus v$. If $v = \varepsilon$ then the products $val(w) \cdot v$ and $v \cdot val(w)$ are represented by ε else the products $val(w) \cdot v$ and $v \cdot val(w)$ are represented by $w \otimes v$ and $w \odot v$, respectively. Thus addition and multiplication with constants is automatic and the so defined ring is semiautomatic. $\square$

The proof is relying on the fact that the field is finite. If one wants to do a similar construction for an infinite structure, a field would not do it. However, if one does not have additive inverses and still multiplication of non-zero elements are non-zero like in the structure $(\mathbb{N}, +, \cdot)$, one can show that also $(\mathbb{N}[U]; +, \cdot, =)$ is semiautomatic. The key idea is that now $val_n(w) = @$ if either the degree is above n or the value of a coefficient is above n. Thus, when evaluating an expression, one only has to deal with coefficients from $\{0, 1, \ldots, n\}$. In particular the semirings $(\mathbb{N}[x]; +, \cdot, =)$ and $(\mathbb{N}[x, y]; +, \cdot, =)$ are semiautomatic.

8 Conclusion

The present work gives an overview on initial results on semiautomatic structures and shows that many prominent structures (countable ordinals with addition, the ordered fields of rationals extended perhaps by one root of an integer, algebraic extensions of finite fields) are semiautomatic and investigates to which degree one can still have that some of the involved operators and relations are automatic. Several concrete questions are still open, in particular the following ones: Is there an automatic presentation of the integers such that addition and equality are automatic while the ordering of the integers is not semiautomatic? Are the structures $(\mathbb{Q},<,=;+,\cdot)$ or $(\mathbb{Q},=;+,\cdot)$ semiautomatic, that is, in the semiautomatic field of rationals, can the order and the equality be made automatic? The corresponding is possible for the additive group of rationals.

Additional questions might relate to the question of effectivity. For example, for a given function f in some given structure, can one effectively find from the parameter y an automaton for $x \mapsto f(x,y)$? While this is impossible for the most general results in Section 4, the concrete structures in Sections 5, 6 and 7 permit that one obtains the automata from the representatives by recursive functions. The complexity of these functions might be investigated in subsequent work for various structures.

Furthermore, one can introduce the concept of faithful semiautomatic structures. Here a semiautomatic structure, say $(\mathbb{Q};+,\cdot,<,=)$, is faithful iff there is a subset S of the set of representatives of $\mathbb{Q}$ such that for each member of $\mathbb{Q}$ there is exactly one representative in S and the semiautomatic functions realising $+$ and $\cdot$ map inputs from $S \times S$ to S. Indeed, one could determine for each rational $\frac{a}{b}$ the greatest common divisor c of a,b and then choose $s = \frac{a/c}{b/c}$. The corresponding adjustments to the semiautomatic functions will then indeed map $S \times S$ to S, although they are still defined for all legal representatives in the structure. It might indeed be interesting to get also for other structures faithful semiautomatic representations.

Acknowledgements

The authors would like to thank Anil Nerode as well as the participants of the IMS Workshop on Automata Theory and Applications in 2011 who discussed the topic and initial results with the authors.

References

1. John Case, Sanjay Jain, Samuel Seah and Frank Stephan. Automatic functions, linear time and learning. *How the World Computes - Turing Centenary Conference and Eighth Conference on Computability in Europe*, CiE 2012, Cambridge, UK, June 18–23, 2012. Proceedings. *Springer LNCS*, 7318:96–106, 2012.
2. Alan Cobham. On the base-dependence of sets of numbers recognizable by finite automata. *Mathematical Systems Theory* 3:186–192, 1969.
3. Christian Delhommé. Automaticité des ordinaux et des graphes homogènes. *Comptes Rendus Mathematique*, 339(1):5–10, 2004.
4. David B.A. Epstein, James W. Cannon, Derek F. Holt, Silvio V.F. Levy, Micheal S. Paterson and William P. Thurston. *Word Processing in Groups.* Jones and Bartlett Publishers, Boston, 1992.

5. Lázló Fuchs. *Partially Ordered Algebraic Systems.* Pergamon Press, 1963.
6. Bernard R. Hodgson. *Théories décidables par automate fini.* Ph.D. thesis, Département de mathématiques et de statistique, Université de Montréal, 1976.
7. Bernard R. Hodgson. Décidabilité par automate fini. *Annales des sciences mathématiques du Québec*, 7(1):39–57, 1983.
8. Rupert Hölzl, Sanjay Jain and Frank Stephan. Learning pattern languages over groups. *Algorithmic Learning Theory - Twentyseventh International Conference*, ALT 2016, Bari, Italy, October 19–21, 2016, Proceedings. *Springer LNCS* 9925:189–203, 2016.
9. John E. Hopcroft, Ravjeev Motwani and Jeffrey D. Ullman. Introduction to Automata Theory, Languages and Computation. Third edition. Addison Wesley, 2007.
10. Sanjay Jain, Bakhadyr Khoussainov, Frank Stephan, Dan Teng and Siyuan Zou. Semiautomatic structures. Computer Science – Theory and Applications – Ninth International *Computer Science Symposium in Russia*, CSR 2014, Moscow, Russia, June 7–11, 2014. Proceedings. *Springer LNCS* 8476:204–217, 2014.
11. Bakhadyr Khoussainov, Sanjay Jain and Frank Stephan. Finitely generated semiautomatic groups. *Pursuit of the Universal, Twelfth Conference on Computability in Europe*, CiE 2016, Paris, France, 27 June - 1 July 2016, Proceedings. *Springer LNCS* 9709:282–291, 2016.
12. Olga Kharlampovich, Bakhadyr Khoussainov and Alexei Miasnikov. *From automatic structures to automatic groups.* CoRR abs/1107.3645, 2011.
13. Bakhadyr Khoussainov and Anil Nerode. Automatic presentations of structures. *Logic and Computational Complexity, International Workshop*, LCC 1994, Indianapolis, Indiana, USA, October 13–16, 1994; *Springer LNCS*, 960:367–392, 1995.
14. Bakhadyr Khoussainov, Sasha Rubin and Frank Stephan. Definability and regularity in automatic structures. *Twentyfirst Annual Symposium on Theoretical Aspects of Computer Science*, STACS 2004, Montpellier, France, March 25-27, 2004, Proceedings; *Springer LNCS*, 2996:440-451, 2004.
15. Dexter Kozen. *Complexity of finitely presented algebras.* PhD thesis, Computer Science Department, Cornell University, May 1977.
16. Joseph-Louis Lagrange. Solution d'un probléme d'Arithmétique. J.-A. Serret (editor). *Oeuvres de Lagrange*, 1:671–731, 1867.
`http://gdz.sub.uni-goettingen.de/no_cache/dms/load/img/?IDDOC=41029`
17. Yuri V. Matiyasevich. Diofantovost' perechislimykh mnozhestv. *Doklady Akademii Nauk SSSR*, 191:297-282, 1970 (Russian). English translation: Enumerable sets are Diophantine, *Soviet Mathematics Doklady*, 11:354-358, 1970.
18. Yuri V. Matiyasevich. *Hilbert's Tenth Problem.* MIT Press, Cambridge, Massachusetts, 1993.
19. Alexei Miasnikov and Zoran Šunić. Cayley graph automatic groups are not necessarily Cayley graph biautomatic. *Language and Automata Theory and Applications - Sixth International Conference*, LATA 2012, A Coruña, Spain, March 5-9, 2012. Proceedings. *Springer LNCS*, 7183:401-407, 2012.
20. Bernhard Hermann Neumann. On ordered groups. *American Journal of Mathematics*, 71:1–18, 1949.

21. André Nies. Describing Groups. *The Bulletin of Symbolic Logic*, 13(3):305-339, 2007.
22. André Nies and Richard Thomas. FA-presentable groups and rings. *Journal of Algebra*, 320:569-585, 2008.
23. André Nies and Pavel Semukhin. Finite automata presentable Abelian groups. *Annals of Pure and Applied Logic*, 161:458–467, 2009.
24. Aleksei L. Semenov. The Presburger nature of predicates that are regular in two number systems. *Siberian Mathematical Journal* 18:289–299, 1977.
25. Wai Yean Tan. *Automatic Structures.* Honours Year Thesis, Department of Mathematics, National University of Singapore, 2008.
26. Dan Teng. *Automatic Structures.* Honours Year Thesis, Department of Mathematics, National University of Singapore, 2012.
27. Siyuan Zou. *Automatic Semigroups and Ordering.* Honours Year Thesis, Department of Mathematics, National University of Singapore, 2013.
28. Todor Tsankov. The additive group of the rationals does not have an automatic presentation. *The Journal of Symbolic Logic*, 76(4):1341–1351, 2011.