Automatic Structures – Recent Results and Open Questions

Frank Stephan

Department of Mathematics and Department of Computer Science, National University of Singapore, 10 Lower Kent Ridge Road, Block S17, Singapore 119076, Republic of Singapore

E-mail: `fstephan@comp.nus.edu.sg`

Abstract. Regular languages are languages recognised by finite automata; automatic structures are a generalisation of regular languages where one also uses automatic relations (which are relations recognised by synchronous finite automata) and automatic functions (which are functions whose graph is an automatic relation). Functions and relations first-order definable from other automatic functions and relations are again automatic. Automatic functions coincide with the functions computed by position-faithful one-tape Turing machines in linear time. This survey addresses recent results and open questions on topics related to automatic structures: How difficult is the isomorphism problem for various types of automatic structures? Which groups are automatic? When are automatic groups Abelian or orderable? How can one overcome some of the limitations to represent rings and fields by weakening the automaticity requirements of a structure?

1. Introduction

Automata theory [13] deals to a large extent with finite automata. These devices process an input symbol by symbol and have a finite memory which they update in reach round of processing an input symbol; the memory also indicates whether the word – as seen so far – satisfies the condition to be checked. For example, the well-known algorithm to check whether a number is a multiple of 3 consists of adding up each digit one after the other and in between only to store the remainder by 3; thus the memory consists only of taking one of the choices 0, 1 and 2. If after the processing, the result is 0 then 3 divides the number just processed. So one can write the algorithm as a table or diagramme of its states and the memory transitions on each input symbol.

state	type	succ at 0,3,6,9	succ at 1,4,7	succ at 2,5,8
0	start, accepting	0	1	2
1	rejecting	1	2	0
2	rejecting	2	0	1

For example, when processing input 22876, the automaton is initially in the state 0, after reading the first 2 in state 2, after reading the next 2 in state 1, after reading the 8 in state 0, after reading the 7 in state 1 and after reading the 6 in state 1; thus the number is not a multiple of 3 and the input is "rejected". Such mechanisms allow to check various properties of integers x (given as sequences of their decimal digits): does x have remainder a when divided by b (for constants a, b); does x have an even number of decimal digits; does x have exactly five decimal digits with value 3; is x the sum of two powers of 10? One can also make easy syntactic checks and also process other information than numbers.

This concept of a finite automaton turned out to be very fruitful in the theory of computation and is also used in many applications: mainly due to the fact that it is though primitive still quite expressive and that automata can be compared and manupulated easily by computer programs without running into decidability-problems.

Important mathematical concepts are often independently discovered by various researchers at various places in the world. For example the concept of a 0 is due to Indian mathematicians as well as to mathematicians of the ancient Maya population in Mezoamerica. Similarly, automatic structures which are applying automata theory to algebraic structures were formalised and developed independently by three groups of researchers: The first work is due to Hodgson [11, 12]; the second team of researchers where Khoussainov and Nerode [16] and the third team of discoverers are Blumensath and Grädel [4, 5]. These teams all spotted the natural idea to carry over the ideas of automata theory from sets to relations and functions; all teams also had in common the idea of going a more restrictive path compared to the alternative path of finite transducers and rational relations and functions [3]; one of the reasons for their choice is to maintain the decidability of the first-order theory of the so defined structures what is not the case for finite transducers and their structures. Khoussainov and Minnes [15] as well as Rubin [22] provide surveys about automatic structures.

There are several equivalent though similar approaches to formalise automatic relations: One way is to consider a finite automaton reading in parallel several input-words, each word by one symbol in each step, where exhauseted input-words provide a special symbol # from outside the alphabet; the automaton accepts iff it is in an accepting state after having processed all the tuple of inputs not entirely consisting of #. The second approach makes this a bit more implicit: One first defines the convolution of several strings and then says that a relation R of several, say three, inputs is automatic iff the set $\{\text{conv}(x, y, z) : (x, y, z) \in R\}$ is regular. Here the convolution of strings combines the symbols of x, y, z at the same position into a new character from the convoluted alphabet where, again, exhausted strings are coded using a special symbol like #. For example, $\text{conv}(0010, 0123456, \varepsilon) = \langle 0, 0, \#\rangle, \langle 0, 1, \#\rangle, \langle 1, 2, \#\rangle, \langle 0, 3, \#\rangle, \langle \#, 4, \#\rangle, \langle \#, 5, \#\rangle, \langle \#, 6, \#\rangle$.

For automatic functions one can either define that their graph is an automatic function or one can say that there is a non-deterministic automaton which reads from each input one symbol in each round and produces for each output one symbol in each round and which goes into an accepting state after all inputs and outputs have become a #; such a function must be unique in the sense that no tuple of inputs has two different tuples of output; due to the pumping lemma it therefore holds that there is a constant c such that no output can be longer than $n + c$ if the longest member of the input tuple has length n. Here non-determinism is essential, as certain automatic functions cannot be computed by a deterministic finite automaton. An example of such a function is the function which exchanges the first and last symbol: $a_1a_2a_3 \ldots a_{n-1}a_n \mapsto a_na_2a_3 \ldots a_{n-1}a_1$: when processing a_1, the automaton guesses and memorises a_n and outputs it; it then reads and copies $a_2, \ldots, a_{n-1}$ until — when receiving the input a_n — it non-deterministically outputs a_1 and compares checks that the swap of a_1 and a_n was executed correctly. If so the dfa goes into an accepting else into a rejecting state; if symbols different from # follow after this swap then the automaton goes into a rejecting state forever and invalidates the complete computation. Brough, Khoussainov and Nelson [7] studied the more restrictive model of automatic functions computed by a dfa in this way and the corresponding structures.

In general, a structure $(A, f_1, \ldots, f_n, r_1, \ldots, r_m)$ of n functions and m relations is *automatic* iff A is a regular set and $f_1, \ldots, f_n$ are automatic functions and $r_1, \ldots, r_m$ are automatic relations. Here the domains of these functions and relations are either A or, for some k, the k-fold Cartesian product $\{\mathrm{conv}(x_1, x_2, \ldots, x_k) : x_1, x_2, \ldots, x_k \in A\}$. Furthermore, to simplify notation, one also calls structures which are isomorphic to automatic structures just "automatic".

Normally one assumes that the inputs in automatic relations of structures have a joint begin and end depending on the string-length; one can also code structures such that they have an aligned middle (say the position whether the dollars are separted from the cent) and then the numbers start somewhere (after perhaps some #) and end somewhere (with trailing #). This second presentation is in some cases useful and can be mapped, by coding, back to the first one. One can show that, for example, the structure of positive finite fractional decimal numbers (some digits before and some digits after the decimal dot, but not an infinite sequence of digits) with addition and $\leq$ form an automatic structure. For example, given three numbers x, y, z, an automaton to check whether $x + y = z$ would process from the back to the front and the states would be "correct and carry to next digit (c)", "correct and no carry to next digit (n)" and "incorrect (i)":

Correct Addition	Incorrect Addition	Missing Leading Digit
`# 2 3 5 8 . 2 2 5`	`3 3 3 3 . 3 3 #`	`1 2 3 . 4 5 6`
`# 9 1 1 2 . # # #`	`# # 2 2 . 2 2 2`	`9 8 7 . 6 5 4`
`1 1 4 7 0 . 2 2 5`	`# 1 5 5 . 5 5 2`	`1 1 1 . 1 1 #`
`n c n n c n n n n n`	`i i n n n n n n n`	`c c c c c c c n`

In these three examples, x stands in the top, y in the second and z in the last row. The states of the automaton are from the back to the front, they have processed the digits behind them but not those before them. The filling symbol # is identified with 0. As long as all dots are aligned, they are just ignored by the automaton and do not cause a state-change. The state "n" is the starting and accepting state, the state "c" can go to an accepting state if the next column is correctly aligned. Once a state "i" is reached, the automaton will reject the computation as incorrect, independently of what will come next. One can also easily make an automaton which goes from the front to the back and compares two numbers and which ends in an accepting state whenever the first number is larger than the second.

This concept can easily be generalised to deal with other digit systems (like binary) or to deal with positive and negative numbers (including 0). Similarly the integers with addition and $\leq$ form an automatic structure. One can also combine both structures by having an extra predicate which tells whether a number is an integer, that is, has no nonzero digits after the decimal dot. These type of structures are, roughly, the most complicated subgroups of the rationals which are automatic; in particular Tsankov [24] showed that the structures of the rationals with addition itself is not automatic. Other non-automatic structures are the ring of integers $(\mathbb{Z}, +, \cdot)$, any infinite field and the free group generated by two or more generators. Finite structures as well as their eventually constant sequences (which component-wise operations on these sequences) are automatic. There are infinite automatic rings: Taking a finite ring $(R, +, \cdot)$, one now considers $(R_\omega, +, \cdot)$ to be the set of all eventually constant sequences $a_0 a_1 a_2 \ldots$ with component-wise operations: $(a + b)_k = a_k + b_k, (a \cdot b)_k = a_k \cdot b_k$. Such a sequence $a_0 a_1 a_2 \ldots$ can be represented by the shortest prefix $a_0 a_1 \ldots a_n$ such that $a_m = a_n$ for all $m \geq n$ in order to get a finite representation. In the following, a few topics on automatic structures will be highlighted more: The characterisation when a function is automatic, orderings, groups, ordered groups, semiautomatic structures.

2. Characterising Automatic Functions

Turing [25, 26] developped a machine model which – now named after him – became a basic model for computations in complexity theory, as it permitted to formalise complexity classes in an easy and clear way. A one-tape Turing machine is a Turing machine which has finitely many states (like a finite automaton) and which has a read-write-head on a tape of symbols where, in each step, depending on the symbol and the state, it replaces the current symbol by a new one, it updates to a new state and it moves the tape by -1, 0 or $+1$ positions. A position-faithful Turing machine has on the tape a start-position with a special symbol which it never crosses. The input is written initially starting after this start-position and the computation replaces it by the output starting also after this start-position. After the end of the input / output are on the tape only blanc symbols which are not used for any other purpose. The run-time of the Turing machine is the number of steps it uses to convert the input into the output; "linear time" means that there is a constant c such that the machine runs on input of length n for at most $(n+1) \cdot c$ steps and has then replaced the input by the output and reached a halting state after which it does not carry out any further activity. Note that a Turing machine can use an alphabet much larger than just the alphabet used for input and output; these additional symbols might be used to store some data on the tape. This is essential for the theorem below. Furthermore, a Turing machine is deterministic iff it in each situation can do only one activity and there is no choice between several alternatives how to proceed. A non-deterministic Turing machine, however, can choose between several alternatives and those can lead to various runs; as different runs might result in different output, the machine might do at the end some checks. In the case that the checks go through, the machine goes into a special "accepting halting state" and the computation is valid; in the case that the checks do not go through, the machine goes into a special "rejecting halting state" and the computation is invalid. For the same input, all valid computations have to give the same output; furthermore, in the case of a time-bound, all computations, whether valid or invalid, have to keep this time-bound, that is, must terminate within $(n+1) \cdot c$ steps for a linear-time Turing machine and input of length n. The next theorem gives a Turing machine characterisation of automatic functions – here with one input mapped to one output; if several inputs or outputs are involved, one has to reduce them to one by using convolutions.

Theorem 2.1 (Case, Jain, Seah and Stephan [8]). *For a function f from strings to strings, the following conditions are equivalent:*

- *f is automatic;*
- *f is computed in deterministic linear time by a position-faithful one-tape Turing machine;*
- *f is computed in non-deterministic linear time by a position-faithful one-tape Turing machine.*

This result works for deterministic and non-deterministic linear time. It would be interesting to know whether it also works for position-faithful machines using alternating linear time; here the alternating machine first would write non-deterministically the output onto the tape which consists of at each time of a convolution of three words (x, y, z) where x is the input, y is the guessed output and z is some work-content; in the case that the machine accepts the pair (x, y) according to the convention of alternating Turing machines then y is a valid output. Furthermore the machine is required to keep the time-limit on all computation pathes involved, even on those for wrongly guessed output.

Note that in Theorem 2.1, the notion of position-faithfulness is needed. To see this consider the function which erases all leading 0 of binary strings, that is, which maps $x1y$ to $1y$ and x to ε for $x \in 0^*$ and $y \in \{0,1\}^*$. This function can be computed by a one-tape Turing machine which is not position-faithful; however, the function is not automatic as the corresponding position-faithful Turing machine would have to move a large amount of information by a non-constant amount of cells. Similarly, two-tape Turing machines can in linear-time do things which automatic functions cannot do, for example to realise the above movement of cells, thus also those are more powerful than automatic functions.

3. Decidability Questions

One important property of automatic structures is that one can use first-order logic to define new relations and functions from given automatic relations and functions; this construction is effective and hence one can even use the well-known automata-theoretic tools in order to decide whether a subset of the domain of an automatic structure defined such a way is empty, finite or coincides with the domain. This permits to decide the first-order theory of a given automatic structure [16]. An application is to get an alternative proof of the decidability of Presburger arithmetics (first-order formulas on the integers involving addition and order) and on the other hand, one also knows that structures with undecidable first-order theory are not automatic. Another important question is the question whether one can effectively decide whether two automatic structures are isomorphic. This is in general impossible, but for some special cases, it is possible: One can decide whether two automatic Boolean algebras are isomorphic and one can decide whether two automatic ordinals (viewed as regular sets of lower ordinals with the corresponding automatic order-relation) are isomorphic [22]. This stands in contrast with the following results.

Theorem 3.1 (Kuske, Liu and Lohrey [17]). *It is impossible to decide whether two given automatic structures are isomorphic for the following types of automatic structures:*

- *It is Π^0_1-complete to decide whether two automatic equivalence relations are isomorphic;*
- *It is Σ^1_1-complete to decide whether two automatic linear orders are isomorphic.*

These and similar results show that only few families of structures have a decidable isomorphism problem. Barany asked whether the following family of structures has a decidable isomorphism problem, where succ is the successor function on the natural numbers and $A, B \subseteq \mathbb{N}$ (in the corresponding representation).

Question 3.2 (Barany [6]). *Given two structures $(\mathbb{N}, A, succ)$ and $(\mathbb{N}, B, succ)$, can one decide whether the structures are isomorphic, that is, whether $A = B$?*

A further natural question is on how difficult it is to know whether a given structure (described by computer programs computing the corresponding operations) has an automatic copy which is isomorphic to it.

Question 3.3 (Kuske). *Given a recursive structure by Turing machines or computer programs deciding the domain and relations and graphs of functions involved, is it a Σ^1_1-complete problem to decide whether the so-described structure is isomorphic to any automatic structure?*

As the corresponding isomorphism problem is Σ^1_1-complete, this problem might also be a candidate for this complexity level.

4. Groups

Groups have been studied under various constraints of automaticity. One of the first frameworks to study groups within automatic structures is the approach by Epstein et al. [10] who taylormade their definition in order to capture a large class of finitely generated group while maintaining a natural coding of the representatives of the group elements. Their approach is orthogonal to the automaticity notion of Khoussainov and Nerode [16], though automaticity by the latter on finitely generated groups implies automaticity by the first, however the concept of automatic groups by Khoussainov and Nerode works also for groups which are not finitely generated. In the present paper, "automatic" always means "automatic as defined by Khoussainov and Nerode"; the reader should be aware that also Epstein et al. [10] call their groups just "automatic".

Recall that a group is a set with a binary operation such that this operation is associative, that there is a neutral element and that every member of the group has an inverse; groups are quite common in mathematics with the integers $(\mathbb{Z},+)$, rationals $(\mathbb{Q},+)$ and reals $(\mathbb{R},+)$ being examples for infinite groups. Such groups where $x+y=y+x$ for all x,y are called Abelian or commutative. There are also a multitude of finite groups. When these groups are small, they can be given by a table as in these examples (addition modulo 4, multiplication modulo 5 of non-zero numbers, double-bit exclusive or):

$+$	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

$\cdot$	1	2	3	4
1	1	2	3	4
2	2	4	1	3
3	3	1	4	2
4	4	3	2	1

$\oplus$	00	01	10	11
00	00	01	10	11
01	01	00	11	10
10	10	11	00	01
11	11	10	01	00

Groups do not need to be Abelian. For example, the group of permutations of members of a set X with at least 3 elements has the group operation $\circ$ with $(f\circ g)(x)=g(f(x))$ for all $x\in X$. This group is not Abelian; in the case that X is finite, also the group of permutations of X is finite. All finite groups are automatic.

A group is called Abelian-by-finite iff it has a subgroup of finite index which is Abelian. Finitely generated automatic groups are closely related to those which are Abelian-by-finite and to a certain degree, one can also say something about groups which itself are not finitely generated.

Theorem 4.1 (Oliver and Thomas [21]). *A finitely generated group $(G,\circ)$ is automatic iff it is Abelian-by-finite.*

Theorem 4.2 (Nies and Thomas [20]). *If $(G,\circ)$ is an automatic group then each finitely generated subgroup is Abelian-by-finite.*

These results can, for example, be used to show that Thompson's group F is not automatic (in the sense of Khoussainov and Nerode); a famous open problem [1] is whether F is automatic in the sense of Epstein et al. [10].

Example 4.3. Thompson's group F is given by generators $x_0,x_1,\ldots$ satisfying $x_nx_m=x_mx_{n+1}$ whenever $m<n$. Hence $x_1(x_0)^n=(x_0)^nx_{n+1}$ and $x_{n+1}=(x_0)^{-n}x_1(x_0)^n$, thus Thompson's group F is finitely generated. Thompson's group F can indeed be described by only finitely many equations:

$$\begin{aligned}(x_0x_1^{-1})\circ((x_0)^{-1}x_1x_0) &= ((x_0)^{-1}x_1x_0)\circ(x_0x_1^{-1});\\ (x_0x_1^{-1})\circ((x_0)^{-2}x_1(x_0)^2) &= ((x_0)^{-2}x_1(x_0)^2)\circ(x_0x_1^{-1}).\end{aligned}$$

So it is a finitely presented group. Now Thompson's group F is not Abelian-by-finite: Assume by way of contratdiction that F has an Abelian subgroup K of finite index k. Then for every $y\in F$, two powers y^i,y^j with $1\le i<j\le k+1$ go into the same set of the form zK where $z\in F$ – index k means that there are only k of these sets. Then $y^i=zv$ and $y^j=zw$ for $v,w\in K$ and $y^{-i+j}=v^{-1}\circ z^{-1}\circ z\circ w=v^{-1}\circ w\in K$. So $y^{k!}\in K$ for every $y\in F$. In particular, $(x_1)^{k!}$ and $(x_0)^{k!}$ are in K and

$$(x_{k!+1})^{k!}=((x_0)^{-k!}x_1(x_0)^{k!})^{k!}=(x_0)^{-k!}\circ(x_1)^{k!}\circ(x_0)^{k!}=(x_1)^{k!}$$

what contradicts known facts about this group: The $k!$-th powers of different x_i and x_j are different. Hence Thompson's group F is not automatic.

Two important questions related to automatic groups are still open.

Question 4.4 (Nies [19]). Is there an infinite and simple automatic group?

Question 4.5 (Nies [19]). Is every torsion-free automatic group Abelian-by-finite?

Here a simple group is a group $(G, \circ)$ which does not have a proper subgroup H of at least two elements such that $x \circ H = H \circ x$ for all $x \in G$. Infinite Abelian groups are never simple.

A group $(G, \circ)$ with neutral element e is torsion-free when $x \circ x$, $x \circ x \circ x$, ... are all different from e for every $x \in G - \{e\}$. Note that in Nies' question the restriction to "torsion-free" is needed; for example if one takes a finite group which is not Abelian (like the permutation group of a set with 3 elements), then one can just consider all functions f from $\mathbb{N}$ into this finite group which are eventually constant with a component-wise operation for each member; this group is also automatic, but not Abelian-by-finite.

The above questions mainly asked on how to describe the automatic groups; an important and well-known question is also on when two automatic groups are isomorphic. In particular, the question whether one can decide the isomorphy of Abelian automatic groups.

Question 4.6. *Is there an algorithm which decides whether two given automatic Abelian groups are isomorphic?*

Given a group $(G, \circ)$, one can fix a subset $F \subseteq G$ of generators of G and consider the edge set $E = \{(x, y) \in G : \exists z \in F\,[x \circ z = y]\}$. Such a graph is connected. Furthermore, the graph is *transitive*, that is, for every $x, y \in G$ there is a graph-isomorphism which maps (G, E) to itself and maps x to y. Berdinsky and Khoussainov [2] investigated connected and transitive automatic graphs and showed that some of them are not derived from a group in the above mentioned way.

5. Groups and Order

An important topic is to combine groups with orders. There are two notions: the ordered automatic group which satisfy both implications $x < y \Rightarrow x \circ z < y \circ z$ and $x < y \Rightarrow z \circ x < z \circ y$ for all x, y, z and the right-ordered / left-ordered groups which satisfy only the respective condition.

Theorem 5.1 (Jain, Khoussainov, Stephan, Teng and Zou [14])**.** *Every automatic ordered group is Abelian, even if only the group operation and not the ordering is automatic. However, there is a non-Abelian left-ordered group.*

This non-Abelian automatic left-ordered group is the Klein bottle group with two generators a, b where $a \circ b = b^{-1} \circ a$ and $a^i b^j < a^{i'} b^{j'}$ iff $i < i'$ or ($i = i'$ and $j < j'$). It should be noted that there are many ways to introduce an order on $(\mathbb{Z}^2, +)$ and one would be to choose real numbers a, b and to look at the ordering which $a \cdot \mathbb{Z} + b \cdot \mathbb{Z}$ inherits from the real numbers. The next result shows that there are non-trivial choices where the resulting ordered group is automatic.

Theorem 5.2 (Jain, Khoussainov, Stephan, Teng and Zou [14])**.** *For every square-root $\sqrt{n}$ of a non-square natural number n, the ordered group $(\mathbb{Z} + \sqrt{n} \cdot \mathbb{Z}, +, <)$ with the ordering $<$ inherited from the real numbers is an automatic ordered group where both, $+$ and $<$, are automatic.*

The following arguments give an idea on how to prove the automaticity of $(\mathbb{Z} + \sqrt{3} \cdot \mathbb{Z}, +, <)$; this case is easier than many others. Choose a real number $u = 2 + \sqrt{3}$, this number is a member of the given group and satisfies $u^2 = 4u - 1$. Now one represents every member of the group by a polynomial of the form

$$a_n u^n + a_{n-1} u^{n-1} + \ldots + a_1 u + a_0 + a_{-1} u^{-1} + \ldots + a_{-m} u^{-m}$$

where $a_k \in \{-3, -2, -1, 0, 1, 2, 3\}$ for all k. These polynomials are always members of the group: By $u^{k+1} = 4u^k - u^{k-1}$, it follows from u^{k-1}, u^k being in the group that also u^{k+1} is in the group and from u^k, u^{k+1} being in the group that also u^{k-1} is in the group. As $u^0 = 1$ and u are in the group, one can then conclude using induction that also all larger and all smaller powers of u are also in the group. Furthermore, due to replacing each $4u^k$ by $u^{k+1} + u^{k-1}$ and $-4u^k$ by $-u^{k+1} - u^{k-1}$ until no coefficient has an absolute value above 3, one can reduce every polynomial into this form without changing the value it represents; the process terminates as every replacement reduces the sum of the absolute values of the coefficients.

Note, that one can check with a finite automaton whether such a polynomial is positive, negative or equal to 0: Starting in the front, one has two variables v, w initialised with 0. When processing a_k, $vu+w$ represents $a_n u^{n-(k+1)} + \ldots + a_{k+1}$ and v, w are updated to values v', w' by $v' = 4v+w$ and $w' = -v+a_k$ so that $v'u+w' = (vu+w)u+a_k = a_n u^{n-k} + \ldots a_{k+1}u + a_k$. These updates are aborted when it is secure that the number is properly positive or properly negative; this is the case when $v'u+w' > 3 \cdot (1+u^{-1}+u^{-2}+\ldots)$ or $v'u+w' < -3 \cdot (1+u^{-1}+u^{-2}+\ldots)$ (while $vu+w$ was in the interval). Indeed, one can chose a constant c such that whenever one of the new values v', w' is outside the interval $\{-c, -c+1, \ldots, c-1, c\}$ for the first time then $vu+w$ passes the threshold to indicate that the number is positive or negative. Note that when $w' > c$ then $w' \leq c+3$ and $v < 3-c$ and $v' < 4v+w \leq 9-3c$ so that $vu+w < 6-2c$ due to $u > 1$. Hence one can argue that whenever one of v', w' is out of the range while v, w were in the range then v' is out of the range itself and w' is at most 3 away from the range, thus $v'u+w'$ is, for c large enough, so large that the sign of v' determines the outcome of the test. In the case that the threshold is not passed through the whole processing of the polynomial, then there are only finitely many choices left and a table tells the automaton which of these choices correspond to a positive number, negative number or to 0.

This key idea can also be adjusted to check whether a sum of two or three polynomials is 0 and so one can compare polynomials or check whether one polynomial is the sum of two others. Furthermore, using the length-lexicographic ordering permits to identify among the various representatives for any number in the group a least representative which can be taken into the domain of the structure.

More precisely, if one has an automatic structure $(A, +, <)$ with $\equiv$ being an automatic equivalence relation representing equality, then one can make the set $B = \{x \in A : \forall y \in A\ [y \equiv x \Rightarrow x \leq_{length-lexicographic} y]\}$ and $(B, +, <)$ is then an automatic structure with exactly one representative for each class of equivalent elements in A. This completes the proof-sketch that the ordered group $(\mathbb{Z} + \sqrt{3}\mathbb{Z}, +, <)$ is automatic.

For the natural numbers, one can define the order from the group operation: $x < y \Leftrightarrow \exists z\,[x+1+z = y]$. Therefore whenever $+$ is automatic, so is $<$. For the integers, this is no longer true, leaving the following open question.

Question 5.3 (Jain, Khoussainov, Stephan, Teng and Zou [14]). *Is there a automatic copy of $(\mathbb{Z}, +)$ such that the ordering $<$ in this copy is not automatic?*

Although this main question is open, there is a related result obtainable by storing fractions between 0 and 1 with denominators being a power of two and a power of three, respectively, in different components of convoluted numbers where the first one uses binary and the second one ternary digits. While an automatic addition can survive this split representation, no automaton for the order can deal with it. This idea permits to prove the following result.

Theorem 5.4 (Jain, Khoussainov, Stephan, Teng and Zou [14]). *There is an automatic copy of $(\{x \cdot 2^y \cdot 3^z : x, y, z \in \mathbb{Z}\}, +)$ in which the addition is automatic but not the order.*

6. Semiautomatic Structures

Automatic structures are limited in the way that they can neither be used to represent infinite fields nor some important groups like $(\mathbb{Q}, +, <)$. Therefore, efforts have been undertaken to overcome this problem by making the structures more general. The key idea is to relax the automaticity requirement on some of the functions and relations involved. Here one says that a function f is semiautomatic iff every induced function obtained by fixing all but one inputs is automatic; similarly for relations. One might also consider to allow multiple representatives for elements of the structure and only require that each equivalence class in the representation is regular, but no longer that the equivalence relation induced by the equality of structure elements is automatic. As one might want to relax this only for some and not all notions, at semiautomatic structures one separates by a semicolon the automatic relations and functions from those which are only semiautomatic. An example is the ring $(\mathbb{Z}, +, <, =; \cdot)$ where the multiplication $\cdot$ is only semiautomatic while the other operations and relations are automatic; as the equality is only semiautomatic in some semiautomatic structures, it is here indicated that here the equality is automatic as well. This structure can be made using any of the usual representations of the additive groups of integer with order, as the multiplication with a constant c can then be realised by c times adding the input to 0. Tsankov [24] provides a result which can be formulated as follows in the language of semiautomatic structures; subrings contain $0, 1$ and are closed under addition and multiplication.

Theorem 6.1 (Tsankov [24]). *$(A, +, <, =; \cdot)$ is a semiautomatic subring of the rationals iff $A = \{x \cdot p^y : x, y \in \mathbb{Z}\}$ for some $p \in \{1, 2, 3, \ldots\}$; the case $p = 1$ captures the ring of integers.*

If one abstains from requiring $+$ to be automatic but only requires it to be semiautomatic, then one gets quite a multitude of examples of semiautomatic groups.

Theorem 6.2 (Jain, Khoussainov, Stephan, Teng and Zou [14]). *The groups $(\mathbb{Q}, =; \cdot)$ and $(\mathbb{Q}, <, =; +)$ are semiautomatic.*

Furthermore, when allowing that all relations and functions are only semiautomatic, one can get ordered semiautomatic fields.

Theorem 6.3 (Jain, Khoussainov, Stephan, Teng and Zou [14]). *The field $(\mathbb{Q}+\sqrt{n}\mathbb{Q}; +, \cdot, <, =)$ is semiautomatic for every non-square integer n.*

However, not every countable structure is semiautomatic. For example, if one looks at vector spaces of dimension 16 over the vector space of the rationals and postulates that all linear functions from the vectorspace to itself are automatic, then this structure is automatic, however, it becomes non-automatic, when the following additional operations are required to be automatic: inverting one component (mapping x to $1/x$ for the case that $x \neq 0$ and mapping 0 to 0); testing whether the coordinates are actually from $\mathbb{Z}^{16}$. Note that if one looks just at the field of rationals itself, then $(\mathbb{Q}; +, \cdot, /, <, =)$ and $(\mathbb{Q}, \mathbb{Z}; +, \cdot, <, =)$ are both semiautomatic. On one hand, the ways to obtain the two semiautomatic structures use different representations, they cannot be combined. On the other hand, the result on the non-semiautomaticity of $\mathbb{Q}^{16}$ with the above mentioned operations was proven by coding Matiyasevich' theorem [18] that it is undecidable to check whether a polynomial in several integer variables takes the value 0. This coding cannot be done when using only one in place of 16 dimensions and therefore the question whether the corresponding structure is semiautomatic is left open, see item (d) below. The other items also stem from limitations of the known constructions where there is also no known proof that these limitations cannot be overcome.

Question 6.4 (Jain, Khoussainov, Stephan, Teng and Zou [14]).

(a) *Are the structures $(\mathbb{Q}, <, =; +, \cdot)$ or $(\mathbb{Q}, =; +, \cdot)$ semiautomatic?*

(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?*

7. Conclusion

The present survey gives an overview over various recent results from the field of automatic structures and also points to open questions which might be interesting for further study.

These questions include decidability questions of isomorphism problems or, more general, their complexity in the arithmetical or analytical hierarchy; the question to which extent automatic groups are Abelian-by-finite and how automaticity of groups relates to the order of these – a line of research initiated by Nies, Oliver and Thomas in three papers [19, 20, 21] – and on the general limits of automata-theory when dealing with structures.

Acknowledgments
The author would like to thank Bakhadyr Khoussainov, Dietrich Kuske, André Nies, Sasha Rubin, Pavel Semukhin and Yue Yang for correspondence; he thanks the NUS and MOE for support through grants R146-000-181-112 and R146-000-184-112.

References

[1] James Michael Belk. *Thompson's Group F.* PhD Thesis, Cornell University, 2004.

[2] Dmitry Berdinsyk and Bakhadyr Khoussainov. On automatic transitive graphs. *Developments in language theory – Eightteens International Conference*, DLT 2014, Ekaterinburg, Russia, August 26–29 2014, Proceedings. *Springer LNCS* 8633:1–12, 2014.

[3] Jean Berstel. *Transductions and Context-free Languages.* Teubner, 1979. `http://www-igm.univ-mlv.fr/~berstel/LivreTransductions/LivreTransductions.html`

[4] Achim Blumensath. *Automatic structures.* Diploma thesis, RWTH Aachen, 1999.

[5] Achim Blumensath and Erich Grädel. Automatic structures. *Fifteenth Annual IEEE Symposium on Logic in Computer Science, Santa Barbara*, LICS 2000, pages 51-62. IEEE Computer Society Press, Los Alamitos, CA, 2000.

[6] Vince Barany. *Automatic Presentations of Infinite Structures.* PhD Thesis, Rheinisch-Westfälische Technische Hochschule Aachen, 2007.

[7] Michael Brough, Bakhadyr Khoussainov and Peter Nelson. Sequential automatic algebras. *Logic and Theory of Algorithms, Fourth Conference on Computability in Europe*, CiE 2008, Greece, June 15–20, 2008, Proceedings. *Springer LNCS* 5028:84–93, 2008.

[8] John Case, Sanjay Jain, Samuel Seah and Frank Stephan. Automatic functions, linear time and learning. *Logical Methods in Computer Science*, 9(3), 2013.

[9] Christian Delhommé. Automaticité des ordinaux et des graphes homogènes. *Comptes Rendus Mathematique*, 339(1):5–10, 2004.

[10] David Epstein, James Cannon, Derek Holt, Silvio Levy, Michael Paterson and William Thurston. *Word Processing in Groups.* Jones and Bartlett Publishers, Boston, Massachusetts, 1992.

[11] Bernard R. Hodgson. *Théories décidables par automate fini.* Ph.D. thesis, University of Montréal, 1976.

[12] Bernard R. Hodgson. Décidabilité par automate fini. *Annales des sciences mathématiques du Québec*, 7(1):39–57, 1983.

[13] John E. Hopcroft, Rajeev Motwani and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Third Edition, Addison-Wesley Publishing, Reading Massachusetts, 2007.

[14] 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.

[15] Bakhadyr Khoussainov and Mia Minnes. Three lectures on automatic structures. *Proceedings of Logic Colloquium 2007. Lecture Notes in Logic*, 35:132–176, 2010.

[16] Bakhadyr Khoussainov and Anil Nerode. Automatic presentations of structures. *Logical and Computational Complexity*, (International Workshop LCC 1994). Springer LNCS 960:367–392, 1995.

[17] Dietrich Kuske, Jiamou Liu and Markus Lohrey. The isomorphism problem on classes of automatic structures with transitive relations. *Transactions of the American Mathematical Society*, 365:5103–5151, 2013.

[18] Yuri V. Matiyasevich. Diofantovost' perechislimykh mnozhestv. *Doklady Akademii Nauk SSSR*, 191:279–282, 1970 (Russian). English translation: Enumerable sets are Diophantine, *Soviet Mathematics Doklady*, 11:354–358, 1970.

[19] André Nies. Describing groups. *Bulletin of Symbolic Logic*, 13:305–339, 2007.

[20] André Nies and Richard M. Thomas. FA-presentable groups and rings. *Journal of Algebra*, 320:569–585, 2008.

[21] Graham Oliver and Richard M. Thomas. Automatic presentations for finitely generated groups. *Twentysecond Annual Symposium on Theoretical Aspects of Computer Science* (STACS 2005), Stuttgart, Germany, Proceedings. *Springer LNCS*, 3404:693–704, 2005.

[22] Sasha Rubin. Automata presenting structures: a survey of the finite string case. *The Bulletin of Symbolic Logic*, 14:169–209, 2008.

[23] Wai Yean Tan. *Reducibilities between regular languages.* Master Thesis, Department of Mathematics, National University of Singapore, 2010.

[24] Todor Tsankov. The additive group of the rationals does not have an automatic presentation. *The Journal of Symbolic Logic*, 76(4):1341–1351, 2011.

[25] Alan Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society 2*, 42:230–265, 1936.

[26] Alan Turing. On Computable Numbers, with an Application to the Entscheidungsproblem: A correction. *Proceedings of the London Mathematical Society 2*, 43:544–546, 1937.