Additive and Multiplicative Semiautomatic Structures with Applications to Geometry*

Ziyuan Gao[1], Sanjay Jain[2], Ji Qi[1],
Philipp Schlicht[3], Frank Stephan[12] and Jacob Tarr[4]

[1] Department of Mathematics, National University of Singapore
10 Lower Kent Ridge Road, S17, Singapore 119076, Republic of Singapore
fstephan@comp.nus.edu.sg
[2] Department of Computer Science, National University of Singapore
13 Computing Drive, COM1, Singapore 117417, Republic of Singapore
sanjay@comp.nus.edu.sg
[3] School of Mathematics, University of Bristol
University Walk, Bristol, BS8 1TW, United Kingdom
philipp.schlicht@bristol.ac.uk
[4] University of British Columbia
jacobdtarr@gmail.com

Abstract. The present work looks at semiautomatic rings with automatic addition and comparisons which are dense subrings of the real numbers and asks how these can be used to represent geometric objects such that certain operations and transformations are automatic. Furthermore, it is shown that for every multiplicatively closed countable ordinal δ, the structure $(\delta; \leqslant, =, +, \cdot)$ is semiautomatic.

Keywords: automatic groups, semiautomatic structures, geometric objects.

1 Introduction

Hodgson [6,7] as well as Khoussainov and Nerode [11] and Blumensath and Grädel [1,2] initiated the study of automatic structures. A structure, say the ordered semigroup of natural numbers $(\mathbb{N}, \circ, \leqslant)$ is then automatic iff there is an isomorphic structure $(A, \circ, \leqslant)$ where A is regular and $\circ, \leqslant, =$ are automatic in the following sense: A finite automaton reads all tuples of possible inputs and outputs with the same speed in a synchronised way and accepts these tuples which are valid tuples in the relations $\leqslant$ and $=$ or which are valid combinations (x, y, z) with $x \circ y = z$ in the case of the semigroup operation (function) $\circ$. For this, one assumes that the inputs and outputs of relations and functions are aligned with each other, like decimal numbers in addition, and for this alignment – which has to be the same for all operations – one fills the gaps with a special

* F. Stephan (PI) and S. Jain (Co-PI) are supported in part by Singapore Ministry of Education Academic Research Fund Tier 2 MOE2016-T2-1-019 / R146-000-234-112. Additionally, S. Jain was supported in part by NUS grant C252-000-087-001.

character. So words are functions with some domain $\{-m, -m+1, \ldots, n-1, n\}$ and some fixed range Σ and the finite automaton reads, when processing a pair (x, y) of inputs, in each round the symbols $(x(k), y(k))$ where the special symbol $\# \notin \Sigma$ replaces $x(k)$ or $y(k)$ in the case that these are not defined. See Example 3 below for an example of a finite automaton checking whether $x + y = z$ for numbers x, y, z; note that a finite automaton computes a function by checking whether the output matches the inputs. Note that structures might have nonunique representatives; however, in the case that the representatives are not unique, a finite automaton must recognise the equality.

The reader should note, that after Hodgson's pioneering work [6,7], Epstein, Cannon, Holt, Levy, Paterson and Thurston [5] argued that in the above formalisation, automaticity is, at least from the viewpoint of finitely generated groups, too restrictive. They furthermore wanted that the representatives of the group elements are given as words over the generators, leading to more meaningful representatives than arbitrary strings. Their concept of automatic groups led, for finitely generated groups, to a larger class of groups, though, by definition, it of course does not include groups which require infinitely many generators; groups with infinitely many generators, to some extent, were covered in the notion of automaticity by Hodgson, Khoussainov and Nerode. Nies and Thomas [13,14] provide results which contrast and compare these two notions of automaticity and give an overview on results for groups which are automatic in the sense of Hodgson, Khoussainov and Nerode. Kharlampovich, Khoussainov and Miasnikov [9] generalised the notion further to Cayley automatic groups. Here a finitely generated group $(A, \circ)$ is Cayley automatic iff the domain A is a regular set, for every group element there is a unique representative in A and, for every $a \in A$, the mapping $x \mapsto x \circ a$ is automatic.

Jain, Khoussainov, Stephan, Teng and Zou [8] investigated the general approach where, in a structure for some relations and functions, it is only required that the versions of the functions or relations with all but one variable fixed to constants is automatic. Here the convention is to put the automatic domains, functions and relations before a semicolon and the semiautomatic relations after the semicolon. For example, a semiautomatic group $(A, \circ; =)$ would be a structure where the domain A is regular, the group operation (with both inputs) is automatic and for each fixed element $a \in A$ the set $\{b \in A : b = a\}$ is regular — note that group elements might have several representatives in semiautomatic groups. The present work will focus more on structures like rings and ordinals than groups, although the field of automatic structures has a strong group theoretic component.

The present work looks at two types of additive and multiplicative structures which generalise well-known structures: (a) one and two dimensional grids where a grid is a dense semiautomatic subring of the real numbers with automatic addition, comparison and equality; (b) the ordinals which generalise the natural numbers into the transfinite. The structures obtained in (a) can be used to represent geometric objects in the plane and it depends on their expressiveness which geometric operations can be carried out.

Definition 1. The convolution of two words v, w is a mapping from the union of their domains to $(\Sigma \cup \{\#\}) \times (\Sigma \cup \{\#\})$ such that first one extends v, w to v', w', each having the domain $dom(v) \cup dom(w)$, by assigning # whenever v or w are undefined and the letting the convolution u map every $h \in dom(v) \cup dom(w)$ to the new symbol $(v'(h), w'(h))$. Similarly one defines the convolutions of three, four or more words.

A h-ary relation R is automatic iff the set of all convolutions of $(x_1, \ldots, x_h) \in R$ is regular; a h-ary function f is automatic iff the set of all convolutions of $(x_1, \ldots, x_h, y)$ with $f(x_1, \ldots, x_h) = y$ is regular. A h-ary relation P is semiautomatic iff for all indices $i \in \{1, \ldots, h\}$ and for all possible values x_j with $j \neq i$ the resulting set $\{x_i : (x_1, \ldots, x_h) \in P\}$ is regular. A h-ary function g is semiautomatic iff for all indices $i \in \{1, \ldots, h\}$ and all possible values x_j with $j \neq i$ the function $x_i \mapsto g(x_1, \ldots, x_h)$ is automatic.

A structure $(A, f_1, \ldots, f_k, R_1, \ldots, R_\ell; g_1, \ldots, g_i, P_1, \ldots, P_j)$ is semiautomatic iff (i) A is a regular set of words with domain being a finite subset of $\mathbb{Z}$, (ii) each f_h is automatic, (iii) each R_h is automatic and (iv) each g_h is semiautomatic and (v) each P_h is semiautomatic. The semicolon separates the automatic components of the structure from those which are only semiautomatic. Structures without semiautomatic items are just called automatic.

An automatic family $\{L_d : d \in E\}$ is a collection of sets such that their index set E and the set of all convolutions of (d, x) with $x \in L_d$ and $d \in E$ are regular.

Definition 2. A semiautomatic grid or, in this paper, just grid, is a semiautomatic ring $(A, +, =, <; \cdot)$ where the multiplication is only semiautomatic and the addition and comparisons are automatic such that A forms a dense subring of the reals, that is whenever p, r are real numbers with $p < r$ then there is an $q \in A$ with $p < q \wedge q < r$ and furthermore, all elements of A represent real numbers.

Example 3. The ring $(\mathbb{D}_b, +, =, <; \cdot)$ of the rational numbers in base b with only finitely many nonzero digits is a grid. Here $\mathbb{D}_b = \{n/b^m : n, m \in \mathbb{Z}\}$. Addition and comparison follow the school algorithm as in the following example of Stephan [16]. In $\mathbb{D}_{10}$, 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   Incomplete Addition
 # 2 3 5 8 . 2 2 5     3 3 3 3 . 3 3 #      9 9 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      0 0 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 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. The decimal dot is not there physically, it just indicates

the position between digit a_0 and digit a_{-1}. The domain of each string is an interval from a negative to a positive number plus an entry for the sign $-$ if needed. $x - y = z$ is checked by checking whether $x = y + z$ and then one can compare the outcome of additions of possibly negative numbers by going to $-$ when the signs of the numbers require this. Furthermore, $x < y$ iff $y - x$ is positive and $x = y$ if the two numbers are equal as strings.

For checking whether $x \cdot i/j = y$ for given rational constant i/j, one just checks whether $i \cdot x = j \cdot y$ which, as i, j are constants, can be done by i times adding x to itself and j time adding y to itself and then comparing the results. So $x \cdot 3/2 = y$ is equivalent to $x+x+x = y+y$ and the latter check is automatic. Also the set of all $x \in \mathbb{D}_{10}$ so that x is a multiple of 3 is regular, as it is first-order definable as

$$\{x : \exists y \in \mathbb{D}_{10}\, [x = y + y + y]\}$$

and 1.2 would be in this set and 1.01 not. This works for all multiples of fixed rational numbers in $\mathbb{D}_b$.

Definition 4. An ordinal structure $(A, +, \cdot, <, =, 0, 1, \omega)$ is a downward closed set A of ordinals with ordinal addition and multiplication and where $0, 1$ are the two smallest ordinals and ω the smallest infinite ordinal. Within the present work, A is always countable and represented such that the operations and comparisons on the structure are at least semiautomatic.

Delhommé [4] showed that a well-ordered automatic set $(A, <, =)$ has an automatic presentation iff the domain of A is isomorphic to some ordinal $\alpha < \omega^\omega$, that is, bounded by some ω^n with $n \in \mathbb{N}$; furthermore, he showed that $(A, <, =)$ has a tree-automatic presentation iff the domain of A is isomorphic to some ordinal $\alpha < \omega^{(\omega^\omega)}$, that is, bounded by some ordinal $\omega^{(\omega^n)}$ for some $n \in \mathbb{N}$. For simplicity, $\omega^{(\omega^\alpha)}$ is in the following just written as ω^{ω^α} with the understanding that the expression $(\omega^\alpha)^\beta$ is always written with brackets in this way.

2 Grids with Special Properties

Jain, Khoussainov, Stephan, Teng and Zou [8] showed that for every natural number c which is not a square there is a grid containing $\sqrt{c}$. Though these grids are dense subsets of the real numbers, they do not have the property that one can divide by any natural number, that is, for each $b \geqslant 2$ there is a ring element x such that x/b is not in the ring. The reason is that most of the rings considered by Jain, Khoussainov, Stephan, Teng and Zou are of the form $\mathbb{Z} \oplus \sqrt{c} \cdot \mathbb{Z}$. The following result will produce grids for which one can always divide by some number $b \geqslant 2$, if this number is composite, it might allow division by finitely many primes. Note that the number of primes cannot be infinite by a result of Tsankov [17].

Theorem 5. *Assume that $b \in \{2, 3, 4, \ldots\}$ and c is some root of an integer and let $u > 1$ be a real number chosen such that the following four polynomials p_1, p_2, p_3, p_4 and constants $\ell, \hat{c}$ exist, where all polynomials have only finitely many nonzero coefficients and all coefficients are integers:*

1. $p_1(u) = \sum_{k\in\mathbb{Z}} b_k u^k = 1/b$;
2. $p_2(u) = \sum_{k\in\mathbb{Z}} c_k u^k = c$;
3. $p_3(u) = \sum_{k=0,-1,-2,\ldots,-h+1} d_k u^k = 0$ *with* $d_0 = 1$;
4. $p_4(u) = \sum_{k\in\mathbb{Z}} e_k u^k = 0$ *with* $e_\ell > \sum_{k\neq\ell} |e_k|$ *and* $|e_k|$ *being the absolute value of* e_k.

Furthermore, the choice of the above has to be such that $\hat{c} > 3|e_\ell|$ *and one can run for every polynomial* $p = \sum_{k=-m,\ldots,n} a_k u^k$ *with every* a_k *being an integer satisfying* $|a_k| \leqslant 3|e_\ell|$ *the following algorithm* C *satisfying the below termination condition:*

Let $k = n + h$.
While $k > -m$ *and* $|a_{k'}| \leqslant \hat{c}$ *for* $k' = k, k-1, \ldots, k-h+1$
Do Begin $p = p - a_k \cdot p_3(u) \cdot u^k$ *and update the coeficients of the polynomial* p *accordingly End.*

The termination condition on C *is that whenever the algorithm terminates at some* $k > -m$ *with some* $|a_{k'}| > \hat{c}$ *then*

$$|\sum_{k'=k,k-1,\ldots,k-h+1} u^{k'} a_k| > u^{k-h}/(1-u) \cdot 3|e_\ell|.$$

If all these assumptions are satisfied then one can use the representation

$$S = \{ \sum_{k=-m,\ldots,n} a_k u^k : m, n \in \mathbb{N},\ a_k \in \mathbb{Z} \text{ and } |a_k| < |e_\ell| \}$$

to represent every member of $\mathbb{D}_b[c]$ *and the ring* $(S, +, <, =; \cdot)$ *has automatic addition and comparisons and semiautomatic multiplication. Furthermore, as* $1/b$ *is in the ring, it is a dense subset of the reals, thus the ring forms a semiautomatic grid.*

Proof. When not giving $-m, n$ explicitly in the sum, the sum $\sum_{k\in\mathbb{Z}} a_k u^k$ uses the assumption that almost all a_k are 0. For the ease of notation, let S' be the set

$$S' = \{ \sum_{k\in\mathbb{Z}} a_k u^k :\ \text{almost all } a_k \text{ are 0 and all } a_k \in \mathbb{Z} \}$$

so that $S \subseteq S'$. On members $p, q \in S'$, one defines that $p \leqslant q$ iff $p(u) \leqslant q(u)$ when the polynomial is evaluated at the real number u. Furthermore, $p = q$ iff $p \leqslant q$ and $q \leqslant p$. Addition and substraction in S' is defined using componentwise addition of coefficients.

Now one shows that for every $p \in S'$ there is a $q \in S$ with $p = q$. For this one lets initially $h = 0$ and $q_h = p$ and whenever there is a coefficient a_k of q_h with $|a_k| \geqslant |e_\ell|$ then one either lets $q_{h+1} = q_h - u^{k-\ell} \cdot p_4(u)$ (in the case that $a_k > 0$ or lets $q_{h+1} = q_h + u^{k-\ell} \cdot p_4(u)$ (in the case that $a_k < 0$). Now let $||q_h||$ be the sum of the absolute values of the coefficients; note that

$$||q_{h+1}|| \leqslant ||q_h|| - e_\ell + \sum_{k\neq\ell} |e_k| < ||q_h||$$

and as there is no infinite strictly decreasing sequence of integers, there is a h where q_h is defined but q_{h+1} not, as this update can no longer be made. Thus all coefficients of q_h are between $-|e_\ell|$ and $+|e_\ell|$ and furthermore, as each polynomial $p_4(u) \cdot u^{k-\ell}$ added or subtracted has the value 0, $q_h = p$. Now let $q = q_h$ and note that q is a member of S with the same value at u as p, so $p(u) = q(u)$.

Now let p, q, r be members of S. In order to see what the sign of $p+q-r$ is, that is, whether $p(u) + q(u) < r(u)$, $p(u) + q(u) = r(u)$ or $p(u) + q(u) > r(u)$, one adds the coefficients pointwise and to check the expression $p + q - r$ at u, one then runs the algorithm C. If C terminates with some $|a_k| > \hat{c}$, then the signs of the current values of

$$\sum_{k'=k,k-1,\ldots,k-h+1} u^{k'} a_k$$

is the sign of $p+q-r$, as the not yet processed tail-sum of $p+q-r$ is bounded by $u^{k-h}/(1-u) \cdot 3|e_\ell|$. In the case that C terminates with all $|a_k| \leqslant \hat{c}$ and $k = -m$, then only the coefficients at $k' = k, k-1, \ldots, k-h+1$ are not zero and again the sign of

$$\sum_{k'=k,k-1,\ldots,k-h+1} u^{k'} a_k$$

is the sign of the original polynomial $p+q-r$. Note that the algorithm C can be carried out by a finite automaton, as it only needs memorise the current values of $(a_k, a_{k-1}, \ldots, a_{k-h+1})$ which are $(0, 0, \ldots, 0)$ at the start and which are updated in each step by reading a_{k-h} for $k = n+h, n+h-1, \ldots, -m$; the update is just subtracting $a_{k'} = a_{k'} - a_k \cdot d_{k'-k}$ for $k' = k, k-1, \ldots, k-h+1$ and then updating $k = k-1$ which basically requires to read a_{k-h} into the window and shift the window into the front; note that the first member which goes out of the window is 0. Furthermore, during the whole runtime of the algorithm, all values in the window have at most the values $(1+\max\{|d_{k''}| : 0 \geqslant k'' \geqslant -h+1\}) \cdot \hat{c}$ and thus there are only finitely many choices for $(a_k, a_{k-1}, \ldots, a_{k-h+1})$ so that at the determination of the sign $\sum_{k'=k,k-1,\ldots,k-h+1} u^{k'} a_k$ the corresponding operation can be done by a table look-up of a finite table. Early termination of the finite automaton can be handled by not changing the state on reading new symbols, once it has gone to a state with some $|a_{k'}| > \hat{c}$. Thus comparisons and addition are automatic; note that for automatic functions, the automaton checks whether the tuple $(inputs, output)$ is correct, it does not compute $output$ from $inputs$.

For the multiplication with constants, note that multiplication with u or u^{-1} is just shifting the coefficients in the representation by one position; multiplication with -1 can be carried out componentwise on all coefficients; multiplication with integers is repeated addition with itself. This also then applies to polynomials put together from these ground operations, so $p \cdot (u^2 - 2 + u^{-1})$ can be put together as the sum of $p \cdot u \cdot u$, $-p$, $-p$, $p \cdot u^{-1}$. All four terms of the sum can be computed by concatenated automatic functions, thus there is an automatic function which also computes the sum of these terms from a single input p. □

Example 6. There is a semiautomatic grid containing $\sqrt{2}$ and $1/2$.

Proof. For $c = \sqrt{2}$ and $b = 2$, one chooses

1. $u^{-1} = 1 - c/2$ (note that $u = 1/(1 - \sqrt{1/2}) = 2/(2 - \sqrt{2}) > 1$),
2. $p_1(u) = 2u^{-1} - u^{-2} = 1/2$,
3. $p_2(u) = 2 - 2u^{-1} = c$,
4. $p_3(u) = 1 - 4u^{-1} + 2u^{-2} = 0$,
5. $p_4(u) = -u + 4 - 2u^{-1} = 0$ with $\ell = 0$,
6. $\hat{c} = 100$ (or any larger value).

While all operations above come from straight-forward manipulations of the choice of u^{-1}, one has to show the termination condition of the algorithm.

For this one uses that $u \geqslant 3.41$ and $1/(1 - 1/u) = \sum_{k \leqslant 0} u^k = \sqrt{2} \leqslant 1.4143$. Assume that the algorithm satisfies before doing the step for k that all $|a_{k'}| \leqslant \hat{c}$ and does not satisfy this after updating a_k, a_{k-1}, a_{k-2} to 0, $a' = a_{k-1} + 4a_k$ and $a'' = a_{k-2} - 2a_k$; in the following, a_k, a_{k-1}, a_{k-2} refer to the values before the update. Without loss of generality assume that $a_k > 0$, the case $a_k < 0$ is symmetric, the case $a_k = 0$ does not make the coefficients go beyond $\hat{c}$. If $a'' < -\hat{c}$ — it can only go out of the range to the negative side — then $2a_k \geqslant \hat{c} - 3(e_\ell - 1)$ and $p(u)$ is at least $a_k \cdot (4 - 2/u) \cdot u^{k-1} - \hat{c} \cdot u^{k-1}/(1 - 1/u) \geqslant ((2 - 1/u) \cdot (\hat{c} - 9) - 1.4143\hat{c})u^{k-1} \geqslant (1.7 \cdot (\hat{c} \cdot 0.9) - 1.4143\hat{c})u^{k-1} \geqslant 0.1 \cdot \hat{c} \cdot u^{k-1} > 0$. If $a'' \geqslant -\hat{c}$ and $a' > \hat{c}$ then $p(u) \geqslant (\hat{c} \cdot u - 1/(1 - 1/u)\hat{c}) \cdot u^{k-2} \geqslant \hat{c} \cdot u^{k-2} > 0$. So in both cases, one can conclude that $p(u)$ is positive. Similarly, when $a_k < 0$ and the bound $\hat{c}$ becomes violated in the updating process then $p(u) < 0$. □

Example 7. There is a grid which contains $\sqrt{3}$ and $1/2$ or, more generally, any c of the form $c = \sqrt{b^2 - 1}$ for some integer $b \geqslant 2$.

Proof. One chooses

1. $u^{-1} = 1 - c/b$ (note that $u = b/(b - c) > 1$),
2. $p_1(u) = 2bu^{-1} - bu^{-2} = 1/b$,
3. $p_2(u) = b - bu^{-1} = c$,
4. $p_3(u) = 1 - 2b^2u^{-1} + b^2u^{-2} = 0$,
5. $p_4(u) = -u + 2b^2 - b^2u^{-1} = 0$ with $\ell = 0$,
6. $\hat{c} = 1000 \cdot b^5$ (or any larger value).

While all operations above come from straight-forward manipulations of the equations, the termination condition of the algorithm needs some additional work. Note that $u > b$, as $b - c < 1$. Indeed, by $u \geqslant 1$ and $p_3(u) = 0$ and $b \geqslant 2$, one has $1 - b^2u^{-1} \geqslant 0$ and $u \geqslant b^2$ and $\sum_{k \leqslant 0} u^k \leqslant 2$. For the algorithm, one now notes that if after an update at k where, without loss of generality, $a_k > 0$, it happens that either (a) $a'' = a_{k-2} - b^2a_k < -\hat{c}$ or (b) $a'' \geqslant -\hat{c}$ and $a' = a_{k-1} + 2b^2a_k > \hat{c}$ then the following holds: In the case (a), $a_k \geqslant 1000b^3 - 6b^2$ and the value of the sum is at least

$$\begin{aligned}(a_k \cdot (2b^2u - b^2) - \hat{c} \cdot u - 12b^4) \cdot u^{k-2} > \\ (2000(b^5 - 6b^4) \cdot u - 1000b^5 \cdot u - 12b^4) \cdot u^{k-2} > \\ (1000b^5 - 18b^4) \cdot u^{k-1} > 0\end{aligned}$$

where the tail sum $12b^4$ estimates that all digits $a_{k'}$ with $k' \leqslant k-2$ are at least $-6b^2$ in the expression and the a_{k-1} is at least $-\hat{c}$ by assumption. In case (b), one just uses that the first coefficient in the sum is at greater than $\hat{c}$ while all other coefficients are of absolute value below $\hat{c}$, in particular as $\hat{c} \geqslant 6b^2$, so that, since $u \geqslant 2$,

$$\hat{c} \cdot u^{k-1} > \sum_{k'<k-1} \hat{c} u^{k'} \text{ and}$$
$$\hat{c} \cdot u^{k-1} > 2 \cdot \hat{c} \cdot u^{k-2}$$

where the third condition then verifies both above. Thus the algorithm terminates as required. □

Example 8. There is a grid which contains $\sqrt[3]{2}$ and $1/2$. For this, one chooses

1. $u^{-1} = 1 - \sqrt[3]{1/2}$ (note that $u > 1$),
2. $p_1(u) = 3u^{-1} - 3u^{-2} + u^{-3} = 1/2$,
3. $p_2(u) = 4 - 8u^{-1} + 4u^{-2} = \sqrt[3]{2}$,
4. $p_3(u) = 1 - 6u^{-1} + 6u^{-2} - 2u^{-3} = 0$,
5. $p_4(u) = -3u^5 + 15u^4 + u^3 + 6u^2 - 2 = (u^3 - 6u^2 + 6u - 2) \cdot (3u^2 + 3u + 1) = 0$ with $\ell = 4$.
6. Instead of a flat $\hat{c}$, one uses a bit different bound for the algorithm, namely $|a_k| \leqslant 8045$, $|a_{k-1}| \leqslant 5045$, $|a_{k-2}| \leqslant 2045$.

The values of the polynomials can be verified by using that $1 - u^{-1} = \sqrt[3]{1/2}$ and then $1/2$ is the third power of this giving p_1 and $\sqrt[3]{2}$ is $2 \cdot (1 - u^{-1})^2$ giving p_2 and p_3 is obtained by taking $p_1(u) - 1/2$ and multiplying it with 2 and $p_4(u) = p_3(u) \cdot u^3 \cdot (3u^2 + 3u + 1)$ which was found using trial products of $p_3(u) \cdot u^3$ with other polynomials until one coefficient was larger than the sum of the absolute values of all others.

3 Applications to Geometry

One can use the grid to represent the coordinates of geometric objects. For this, one uses in the field of automatic structures the concept of convolution which uses the overlay of constantly many words into one word. One introduces a new symbol, #, which is there to pad words onto the same length. Now, for example, if in the grid of decimal numbers, one wants to describe a point of coordinates $(1.112, 22.2895)$, this would be done with the convolution $(\#,2)(1,2).(1,2)(1,8)(2,9)(\#,5)$ where these six characters are the overlay of two characters and the dot is virtual and only marking the position where the numbers have to be alinged, that means, the position between the symbols number 0 and number -1. Instead of combining two numbers, one can also combine five numbers or any other arity.

An automatic family is a family of sets L_e with the indices e from some regular set D such that the set $\{conv(e,x) : x \in L_e\}$ is regular. Given a grid G, the set of all lines parallel to the x-axis in $G \times G$ is an automatic family: Now $D = G$ and $L_y = \{conv(x,y) : x \in G\}$. The next example shows that one cannot have an automatic family of all lines.

Example 9. The set of all lines (with arbitrary slope) is not an automatic family, independent of the definition of the semiautomatic grid. Given a grid G and assuming that $\{L_e : e \in D\}$ is the automatic family of all lines, one can first-order define the multiplication using this automatic family:

> $x \cdot y = z$ if either at least one of x, y and also z are 0 or $x = 1$ and $y = z$ or $y = 1$ and $x = z$ or all are nonzero and neither x nor y is 1 and there exist an $e \in D$ such that $conv(0,0), conv(1,y), conv(x,z)$ are all three in L_e.

As the grid G has to be dense and is a ring with automatic addition and comparison and as $G \subset \mathbb{R}$, the ring G is an integral domain and furthermore, G has an automatic multiplication by the above first-order definition. Khoussainov, Nies, Rubin and Stephan [12] showed that no integral domain is automatic, hence the collection of all lines cannot be an automatic family, independent of the choice of the grid.

Similarly one can consider the family of all triangles.

Theorem 10. *Independently of the choice of the semiautomatic grid G, the family of all triangles in the plane is not an automatic family. However, every triangle with corner points in $G \times G$ is a regular set.*

Proof. For the first result, assume that $\{L_e : e \in D\}$ is a family of all triangles – when viewed as closed subsets of $G \times G$ – which are represented in the grid and that this family contains at least all triangles with corner points in $G \times G$. Now one can define for $x, y, z > 0$ the multiplication-relation $z = x \cdot y$ using this family as follows:

> $z = x \cdot y \Leftrightarrow$ some $e \in D$ satisfies the following conditions:
> $\forall v, w \in G$ with $v \leqslant 0\, [(v,w) \in L_e \Leftrightarrow (v,w) = (0,0)]$,
> $\forall w \in G\, [(1,w) \in L_e \Leftrightarrow 0 \leqslant w \wedge w \leqslant y]$,
> $\forall w \in G\, [(x,w) \in L_e \Leftrightarrow 0 \leqslant w \wedge w \leqslant z]$.

This definition can be extended to a definition for the multiplication on full G with a straightforward case-distinction. Again this cannot happen as then the grid would form an infinite automatic integral domain which does not exist.

However, given a triangle with corner points $(x,y), (x',y'), (x'',y'')$, note that one can find that linear functions from $G \times G$ into G which are nonnegative iff the input point is on the right side of the line through (x,y) and (x',y'). So one would require that the function

$$f(v,w) = (w-y) \cdot (x'-x) - (v-x) \cdot (y'-y)$$

is either always nonpositive or always nonnegative, depending on which side of the line the triangle lies; by multiplying f with -1, one can enforce nonnegativeness. Note here that $x' - x$ and $y' - y$ are constants and multyplying with constants is automatic, as the ring has a semiautomatic multiplication. Thus a point is in the interior of the triangle iff all three automatic functions associated with the three border-lines of the triangle do not have negative values. This allows to show that every triangle with corner points in $G \times G$ is regular. □

Proposition 11. *Moving a polygone by a distance (v, w) can be done in any grid, as it only requires addiing (v, w) to the coordinates of each points. However, rotating by 30° or 45° is possible only with certain assumptions on the grid.*

Proof. Note that the formula for rotating around 30°, one needs to map each point (x, y) by the mapping $(x, y) \mapsto (\cos(30°)x - \sin(30°)y, \sin(30°)x + \cos(30°y)$ and similarly for 45° and 60°. For 30°, as $\sin(30°) = 1/2$ and $\cos(30°) = \sqrt{3}/2$, one needs a grid which allows to divide by 2 and multiply by $\sqrt{3}$, an example is given by the grid of Example 7. For rotating by 60°, as it is twice doing a rotation by 30°, the same requirements on the grid need to be there. For rotating by 45°, the grid from Example 6 can be used. However, these operations cannot be done with grids which do not have the corresponding roots and also do not have the possibility to divide by 2. Note that when $\sqrt{1/2}$ is in the ring, then also $1/2$, as it is the square of the latter. □

Remark 12. . Note that one represent a word $a_5a_4 \ldots a_1a_0.a_{-1}a_{-2} \ldots a_{-7}$ also by staring with a_0 and then putting alternatingly the digits of even and odd indices giving $a_0a_1a_{-1}a_2a_{-2} \ldots a_5a_{-5}0a_{-6}0a_{-7}$ and one can show that in this representation, the same semiautomaticity properties are valid as in the previously considered representation. However, one gets one additional relation: One can recognise whether two digits a_{-m} and a_n satisfy that $n = m + c$ for a given integer constant c. This is used in the following example.

Example 13. The family $\{E_d : d \in D\}$ of all axis-parallel rectangles is an automatic family in all grids. Furthermore, let $d \equiv d'$ denote that E_d and $E_{d'}$ have the same area. In no grid, this relation $\equiv$ is automatic. In the grid $\mathbb{D}_p, +, =, <; \cdot)$ where p is a prime power from Example 3, the relation $\equiv$ is semiautomatic using the representation given in Remark 12. However, for grids like $\mathbb{D}_6, +, =, <; \cdot)$ where b is a composite number other than a prime power, this method does not work.

4 Ordinals and Automaticity

Theorem 14. *Let A be a downward closed set of ordinals. There is a semiautomatic structure $(B, +, \cdot, <, =; \cdot)$ containing all ordinals from A iff the ordinals in A have an upper bound $\alpha < \omega^\omega$.*

Jain, Khoussainov, Stephan, Teng and Zou [8] showed that for countable downward-closed sets A of ordinals, the structure $(A; +, <, =)$ is semiautomatic; however,

none of the relations is automatic in general. The next result shows that if A is the set of all ordinals strictly below $\omega^\omega \cdot n$ for some fixed n, then one can achieve that the equality is automatic.

Example 15. For every fixed $n \in \mathbb{N}$, there is a semiautomatic structure $(A, =; <, +)$ containing all ordianals below $\omega^\omega \cdot n$ where the domain A is regular and the equality is automatic.

The following result is due to space constraints delayed to the journal version of this article.

Theorem 16. *Let A be a downward closed countable set of ordinals. There is a semiautomatic structure representing all ordinals in A in which addition, multiplication, order and equality are all semiautomatic.*

Acknowledgments. Ji Qi and Jacob Tarr have done their undergraduate research projects UROPS at NUS on topics from this paper.

References

1. Achim Blumensath. *Automatic structures.* Diploma thesis, Department of Computer Science, RWTH Aachen, 1999.
2. Achim Blumensath and Erich Grädel. Automatic structures. *Fifteenth Annual IEEE Symposium on Logic in Computer Science*, LICS 2000, pages 51–62, 2000.
3. John Case, Sanjay Jain, Samuel Seah and Frank Stephan. Automatic functions, linear time and learning. *Logical Methods in Computer Science*, 9(3), 2013.
4. Christian Delhommé. Automaticité des ordinaux et des graphes homogènes. *Comptes Rendus Mathematique*, 339(1):5–10, 2004.
5. 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.
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. Sanjay Jain, Bakhadyr Khoussainov, Frank Stephan, Dan Teng and Siyuan Zou. Semiautomatic structures. *Theory of Computing Systems*, 61(4):1254–1287, 2017.
9. Olga Kharlampovich, Bakhadyr Khoussainov and Alexei Miasnikov. From automatic structures to automatic groups. *Groups, Geometry and Dynamical Systems*, 8(1):157–198, 2014.
10. Bakhadyr Khoussainov and Mia Minnes. Three lectures on automatic structures. *Logic Colloquium 2007*, Proceedings. *Lecture Notes in Logic*, 35:132–176, 2010.
11. Bakhadyr Khoussainov and Anil Nerode. Automatic presentations of structures. *Logic and Computational Complexity, International Workshop*, LCC 1994, Indianapolis, Indiana, USA, October 13–16, 1994, Proceedings. *Springer LNCS*, 960:367–392, 1995.
12. Bakhadyr Khoussainov, Andre Nies, Sasha Rubin and Frank Stephan. Automatic structures: richness and limitations. *Logical Methods in Computer Science*, volume 3, number 2, 2007.

13. André Nies. Describing Groups. *The Bulletin of Symbolic Logic*, 13(3):305-339, 2007.
14. André Nies and Richard Thomas. FA-presentable groups and rings. *Journal of Algebra*, 320:569-585, 2008.
15. 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.
16. Frank Stephan. Automatic structures – recent results and open questions. Keynote Talk. *Third International Conference on Science and Engineering in Mathematics, Chemistry and Physics*, ScieTech 2015, *Journal of Physics: Conference Series*, 622/1 (Paper 012013), 2015.
17. Todor Tsankov. The additive group of the rationals does not have an automatic presentation. *The Journal of Symbolic Logic*, 76(4):1341–1351, 2011.

5 Appendix

The below material will only be taken into the journal version of this paper. Recall that in a semi-automatic presentation, one considers all relations and functions with all but one coordinate fixed as arbitrary elements of the structure. All these relations and functions are required to be automatic. Equality is understood as a relation – in other words the presentation can be non-injective. Moreover, equality needs to satisfy the same requirement of semiautomaticity.

Theorem 17. *If η is any multiplicatively closed countable ordinal, then the structure $(\eta; \leqslant, +, \cdot)$ is semiautomatic.*

Proof. Since η is multiplicatively closed, it equals ω^ζ for some additively closed ordinal ζ. Then $\zeta = \omega^\delta$ for some ordinal δ. Recall that any countable relational structure with finitely many relations is semiautomatic. [cite from Frank's paper] We fix such a presentation of $(\delta; \leqslant)$ in a finite language L with domain D. Note that the proof of [cite result from Frank's paper] shows that equality is automatic. Moreover, note that the presentation of $(\delta; \leqslant)$ from [cite result from Frank's paper] looks as follows. For each $w < \delta$, there is some n such that $v \leqslant w$ is decided by the fact that $v(n) = 0$ or 1, except for finitely many words v.

We add $\{\oplus_l, \oplus_r, \otimes_l, \otimes_r\}$ to the alphabet. The blocks in the language L between the symbols $\oplus_l, \oplus_r, \otimes_l, \otimes_r$ are of the form $\omega^{p_0} + \cdots + \omega^{p_n}$, where $p_i = \omega^{\alpha_0} + \ldots \omega^{\alpha_{n_i}}$ for $\alpha < \delta$. More precisely, the blocks are written as $\omega^\wedge(p_0) + \cdots + \omega^\wedge(p_n)$, Moreover, p_i is presented as $\omega^\wedge w_0 + \cdots + \omega^\wedge w_{n_i}$, where $w_i \in D$ represents α_i. For this, we add the new symbols $\omega, (,), +$ and $\wedge$ to our language.

It is clear that in this presentation, addition and multiplication are semiautomatic. One simply appends a word. So it remains to describe automata for $\leqslant \gamma$ and $\geqslant \gamma$. It is sufficient to do this for $\geqslant \gamma$, since $\beta \leqslant \gamma$ is equivalent to $\neg\beta \geqslant \gamma + 1$.

The algorithm for $\geqslant \gamma$ works as follows.

The *double Cantor normal form* of an ordinal γ is the (unique) presentation

$$\gamma = \omega^{\zeta_l} k_l + \cdots + \omega^{\zeta_0} k_0$$

where $k_i \geqslant 1$ for all $i \leqslant l$ and each ζ_i is written as

$$\zeta_i = \omega^{\alpha^i_{l_i}} m^i_{l_i} + \cdots + \omega^{\alpha^i_0} m^i_0.$$

Let A be a finite set of ordinals and k_+, l_+ natural numbers.

- We add new symbols $\{0, \ldots, k_+\}$. $k < k_+$ represents the natural number k, while k_+ represents an arbitrary natural number $\geqslant k_+$.
- We add new symbols $\{0, \ldots, m_+\}$. $m < m_+$ represents the natural number m, while m_+ represents an arbitrary natural number $\geqslant m_+$.
- Let $(\alpha_0, \ldots, \alpha_l)$ enumerate A in increasing order. We add new symbols $\{\alpha_0, \ldots, \alpha_l\}$ and $\{\beta_0, \ldots, \beta_{l+1}\}$. α_i represents the ordinal α_i, while β_i represents ordinals in the interval (α_{i-1}, α_i), where $\alpha_{-1} = -\infty$ and $\alpha_{l+1} = \infty$.

We define $\gamma \mapsto \gamma^*$ as follows.

If $\alpha < \beta < \delta$ and β is additively closed, then we can form a quotient of $(\delta, \leqslant, +)$ by mapping the interval (α, β) to a single point. $\leqslant$ is well-defined on the quotient, since (α, β) is an interval. $+$ is well-defined on the quotient, since β is additively closed.

Similarly, one can form the quotient with respect to arbitrarily many disjoint such intervals, and $\leqslant$ and $+$ will be well-defined.

One always gets a well-ordered set with an associative addition. It does not have to be the usual addition, since $x + x = x \neq 0$ is possible.

We now represent ordinals $\gamma < \delta$ in double Cantor normal form: $\gamma = \omega^{\omega^{\alpha_k} n_k + \cdots + \omega^{\alpha_0} n_0} + \ldots$.

First form quotients as above in the exponents. Certain intervals $[\omega^\alpha, \omega^\beta)$ are mapped to a point.

Given a well-ordered set A, consider the set B of expressions $\omega^{\alpha_k} n_k + \cdots + \omega^{\alpha_0} n_0$ with $\alpha_i < \eta$ and $\alpha_k > \cdots > \alpha_0$.

To Do: [inline]The order below needs to be changed a bit. $n_i = 0$ if α_i is quotiented out.

This carries a natural order, like for ordinals: the lexicographic order on tuples $(\alpha_k, n_k, \ldots, \alpha_0, n_0)$. It also has a natural addition, defined as for ordinals. **To Do:** check here that $+$ is compatible with quotient Larger terms on the **To Do:** write out right absorb smaller terms on the left.

Suppose that A has a binary function $\oplus$ (do we need compatible with the order?).

Multiplication on B: Let $\omega^\alpha \cdot \omega^\beta = \omega^{\alpha \oplus \beta}$. **To Do:** check here that $\cdot$ is compatible with quotient Moreover, let $(\omega^{\alpha_k} n_k + \cdots + \omega^{\alpha_0} n_0) \cdot \omega^\delta n = \omega^{\alpha_0} \omega^\delta n$ for $\delta > 0$ and $(\omega^{\alpha_k} n_k + \cdots + \omega^{\alpha_0} n_0) \cdot n = \omega^{\alpha_k} n_k n + \cdots + \omega^{\alpha_0} n_0$. This is extended in the obvious way.

Let $\alpha \mapsto \alpha^*$ be the quotient map.

Claim 18. $\beta \geqslant \alpha \Rightarrow \beta^* \geqslant \alpha^*$.

Proof. This holds for the quotient maps. So it holds for the blocks. Thus the claim holds by definition of the order for γ-pseudo-ordinals.

Claim 19. $\alpha < \beta = \beta^* \implies \alpha^* < \beta = \beta^*$.

Proof. Take the first block that's different. Take its exponent. Whether the quotient is applied to it or not, the strict order for this exponent is preserved.

To Do: [inline]Continue here. The following just says that the quotient is compatible with $+$ and $\cdot$.

Claim 20. $(\alpha + \beta)^* = \alpha^* + \beta^*$.

Proof. Look at the first ω-power in the CNF of γ. Take the first ω-power in the CNF of β that's $\leqslant$.

If this (in γ) is not substituted, then the claim follows from the previous claims.

Suppose this is substituted. It seems to follow from the absorption rule for ω^{β_i}.

Claim 21. $(\alpha \cdot \beta)^* = \alpha^* \cdot \beta^*$.

Proof. Look at sums of the exponents.

How does one choose the intervals for γ? $[\omega^{\alpha_i}, \omega^{\alpha_{i+1}})$ for $i \leqslant l$, $(\lambda+m_+, \lambda+\omega)$ for limits λ.

The algorithm will read the input word from left to right and add or multiply the next block between the symbols $\{\oplus_l, \oplus_r, \otimes_l, \otimes_r\}$ by applying $*$.

We compare the outcome of the computation with γ with respect to $\leqslant$. It follows from the above claims that this determines whether $\beta \geqslant \gamma$.

Let $\zeta = \omega^{\alpha_k} n_k + \cdots + \omega^{\alpha_0} n_0$.

We obtain ζ_i' by taking the Cantor normal form of ζ_i and replacing each ordinal in the interval (α_{i-1}, α_i) by β_i and each $m \geqslant m_+$ by m_+. We call ζ_i' *approximate* if the first substitution takes place, i.e. some β_i appear. We obtain γ^* from the Cantor normal form of γ by replacing ζ_i with ζ_i', and k_i with 1 if ζ_i' is approximate.

Now fix γ. Let $A = \{\alpha_j^i \mid i \leqslant l, j \leqslant l_i\}$, $k_+ = \max\{k_i \mid i \leqslant l\}$ and $m_+ = \max\{m_j^i \mid i \leqslant l, j \leqslant l_i\}$. The map $\delta \mapsto \delta^*$ is defined as above.

A $*$-*ordinal* is a formal Cantor normal form with only ordinals in A appearing as exponents, coefficients on the first level bounded by k_+ and on the second level by m_+.

The order $\leqslant$ on $*$-ordinals, the exponents and exponents of exponents in them, is defined as for ordinals in double Cantor normal form, where the order on the new symbols is given in the obvious way.

Furthermore, the same definition allows us to compare any ordinal in double Cantor normal form with a $*$-ordinal with respect to $\leqslant$.

Addition of $*$-ordinals is defined as follows. One adds the Cantor normal forms as usual, but for the coefficients let $k_+ + k = k_+ + k_+ = k_+$ for any k and $l_+ + l = l_+ + l_+ = l_+$ for any l. Moreover, $\omega^{\zeta_i^*} + \omega^{\zeta_i^*} = \omega^{\zeta_i^*}$ if ζ_i^* is approximate. Moreover, $\omega^{\zeta_i^*} + \alpha = \omega^{\zeta_i^*}$ for any $\alpha \leqslant \omega^{\zeta_i^*}$.

Multiplication of $*$-ordinals is defined via $\leqslant$ and addition.

Note: any exponent of any **To Do:** γ-pseudo-ordinal$*$-ordinal is of the form (a normal ordinal in CNF) plus (a single term $\omega^{\alpha_k^*}$). This follows from the addition defined above.

There is only a finite number of $*$-ordinals.

Note that $\gamma^* = \gamma$.

To Do: [inline]Continue here

To Do: [inline]Call $*$ "mod γ"? $\beta^* = \beta^{|\gamma}$

Claim. $\beta \geqslant \alpha \Rightarrow \beta^* \geqslant \alpha^*$.

Proof. Take the first place where the CNFs of α and β differ.

It's clear from the definition that the order of single blocks is preserved when forming $*$.

It follows that $\beta^* \geqslant \alpha^*$.

Claim 22. $\beta \geqslant \gamma \Leftrightarrow \beta^* \geqslant \gamma^* = \gamma$.

Proof. Suppose that $\beta < \gamma$. Take the first place where the CNFs of β and γ differ. Before that, nothing is substituted.

Argue that for two blocks, if the first one is $<$ and is substituted, the second one is not, then the first one ends up strictly smaller.

Argue that this is sufficient.

Claim 23. $(\alpha + \beta)^* = \alpha^* + \beta^*$.

Proof. Look at the first ω-power in the CNF of γ. Take the first ω-power in the CNF of β that's $\leqslant$.

If this (in γ) is not substituted, then the claim follows from the previous claims.

Suppose this is substituted. It seems to follow from the absorption rule for ω^{β_i}.

Claim 24. $(\alpha \cdot \beta)^* = \alpha^* \cdot \beta^*$.

Proof. Look at sums of the exponents.
The algorithm will read the input word from left to right and add or multiply the next block between the symbols $\{\oplus_l, \oplus_r, \otimes_l, \otimes_r\}$ by applying $*$.

We compare the outcome of the computation with γ with respect to $\leqslant$. It follows from the above claims that this determines whether $\beta \geqslant \gamma$.

Theorem 25. *Certain pattern languages are regular in the above presentation.*

Note that Theorem 25 implies Theorem 17.