Ordered Semiautomatic Rings 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
Fry Building, Woodland Road, Bristol, BS8 1UG, UK
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. The underlying ring has always to be a countable dense subring of the real numbers and additions and comparisons and multiplications with constants need to be automatic. It is shown that the ring can be selected such that equilateral triangles can be represented and rotations by 30° are possible, while the standard representation of the b-adic rationals does not allow this.

1 Introduction

Hodgson [7,8] as well as Khoussainov and Nerode [14] and Blumensath and Grädel [1,2] initiated the study of automatic structures. A structure $(A, \circ, \leq)$ of, say, an ordered semigroup is then automatic iff there is an isomorphic copy $(B, \circ, \leq)$ where B is regular and $\circ, \leq, =$ 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 $\leq$ 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

* Frank Stephan (PI) and Sanjay Jain (Co-PI) are supported in part by Singapore Ministry of Education Academic Research Funds Tier 2 MOE2016-T2-1-019 / R146-000-234-112 and MOE2019-T2-2-121 / R146-000-304-112. Additionally, Sanjay Jain was supported in part by NUS grant C252-000-087-001. This project has also received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreements No 794020 of Philipp Schlicht (Project *IMIC: Inner models and infinite computations*). Ji Qi and Jacob Tarr worked on this paper in the framework of the Undergraduate Research Opportunities Programme at the NUS. The authors would like to thank Bakhadyr Khoussainov and Sasha Rubin for correspondence.

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 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; here a finite automaton computes a function by checking whether the output matches the inputs. Automatic functions are characterised as those computed by a position-faithful one-tape Turing machine in linear time [4]. Position-faithful means that input and output start at the same position on the Turing tape and the Turing machine overwrites the input by the output. Furthermore, note that if a structure has several representatives per element then a finite automaton must recognise the equality.

The reader should note, that after Hodgson's pioneering work [7,8], Epstein, Cannon, Holt, Levy, Paterson and Thurston [6] 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, of course it 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, Oliver and Thomas provide in several papers [16,18,19] 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 [12] 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 [9] 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 than groups, although the field of automatic and semiautomatic structures has a strong group theoretic component. The construction of these semiautomatic rings is similar to that of Nies and Semukhin [17] for a presentation of $\mathbb{Z}^2$ where no 1-dimensional subgroup is a regular subset.

The interested reader finds information about automatic structures in the surveys of Khoussainov and Minnes [13] and Rubin [20]. Related but different links between automata theory and geometry have been studied previously like, for example, the usage of weighted automata and transducers to generate fractals [5], ω-automata to represent geometric objects in the reals [3,11] and the field of reals not being ω-automatic [23,24]. The last section of the present work applies the results and methods of the current work to ω-automatic structures.

The present work looks at semiautomatic rings which can be used to represent selected points in the real plane. Addition and subtraction and comparisons as well as multiplication with constants have to be automatic; however, the full multiplication is not automatic. It depends on the structures which geometric objects and operations with such object can be represented.

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 then 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* [1,2,7,8,14] iff the set of all the convolutions of tuples $(x_1, \ldots, x_h) \in R$ 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* [9] iff for all indices $i \in \{1, \ldots, h\}$ and for all possible fixed 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* [9] iff (i) A is a regular set of words where each word maps a finite subset of $\mathbb{Z}$ to a fixed alphabet, (ii) each f_h is automatic, (iii) each R_h is automatic, (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* [10] $\{L_d : d \in E\}$ 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.

It makes sense to define density as a property of an ordered ring that is embeddable into the reals, since the embedding is unique. A necessary and sufficient criterion for the ring to be dense is that it has an element strictly between 0 and 1. If there is q with $0 < q < 1$ and p, r are any ring members with $p < r$ then $p + q \cdot (r - p)$ is strictly between p and r. Furthermore, ordered rings which are subrings of the real numbers do not have an endpoint, as adding $+1$ or -1 to any element allows finding elements strictly above and strictly below the element.

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 [21]. 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)". In the following three examples, x stands on the top, y in the second and z in the last row. The states of the automaton are for starting from the end of the string to the beginning after having processed the digits after 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.

```
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   i i n n  n n n n    c c c c c  c c c n
```

The difference $x-y=z$ 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 $\mathbb{D}_b$.

2 Grids with Special Properties

Jain, Khoussainov, Stephan, Teng and Zou [9] 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 $b \geq 2$ 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 $b \geq 2$, 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 [22].

Theorem 4. *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 in a variable x 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* $p = \sum_{k=-m,\ldots,n} a_k x^k$ *with every* a_k *being an integer satisfying* $|a_k| \leq 3|e_\ell|$ *the following algorithm* C *satisfying the below termination condition:*

Let $k = n + h$.
While $k > -m$ *and* $|a_{k'}| \leq \hat{c}$ *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 coefficients of the polynomial* p *accordingly;*
Let $k = k - 1$ *End.*

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

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

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, sums like $\sum_{k \in \mathbb{Z}} a_k u^k$ use the assumption that almost all a_k are 0. For ease of notation, let S' be the set

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

so that $S \subseteq S'$. On members $p, q \in S'$, one defines that $p \leq q$ iff $p(u) \leq q(u)$ when the polynomial is evaluated at the real number u. Furthermore, $p = q$ iff $p \leq q$ and $q \leq p$. Addition and subtraction 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 $|a_k| \geq |e_\ell|$ then one either lets $q_{h+1} = q_h - x^{k-\ell} \cdot p_4$ (in the case that $a_k > 0$) or lets $q_{h+1} = q_h + x^{k-\ell} \cdot p_4$ (in the case that $a_k < 0$). Now let $||q_h||$ be the sum of the absolute values of the coefficients; note that

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

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 sign of the current value of

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

gives the sign of $p+q-r$, as the not yet processed tail-sum of $p+q-r$ is bounded by $u^{k-h}/(1-u^{-1}) \cdot 3|e_\ell|$. In the case that C terminates with all $|a_{k'}| \leq \hat{c}$ 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 to 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 by one character; 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 $(1 + \max\{|d_{k''}| : 0 \geq k'' \geq -h+1\}) \cdot \hat{c}$ and thus there are only finitely many choices for $(a_k, a_{k-1}, \ldots, a_{k-h+1})$, and thus the determination of the sign of $\sum_{k'=k,k-1,\ldots,k-h+1} u^{k'} a_{k'}$ can be done by looking up 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 $(\mathit{inputs}, \mathit{output})$ 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 5. 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 $\ell = 0$ and $e_\ell = 4$,
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 $u \geq 3.41$ and $1/(1 - 1/u) = \sum_{k \leq 0} u^k = \sqrt{2} \leq 1.4143$. Assume that the algorithm satisfies before doing the step for k that all $|a_{k'}| \leq \hat{c}$ and does not satisfy this after updating a_k, a_{k-1}, a_{k-2} respectively 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 $2a_k \geq \hat{c} - 3(e_\ell - 1)$ and $p(u)$ is at least $a_k \cdot (4 - 2/u) \cdot u^{k-1} - \hat{c} \cdot u^{k-1}/(1 - 1/u) \geq ((2 - 1/u) \cdot (\hat{c} - 9) - 1.4143\hat{c})u^{k-1} \geq (1.7 \cdot (\hat{c} \cdot 0.9) - 1.4143\hat{c})u^{k-1} \geq 0.1 \cdot \hat{c} \cdot u^{k-1} > 0$. If $a'' \geq -\hat{c}$ and $a' > \hat{c}$ then $p(u) \geq (\hat{c} \cdot u - 1/(1 - 1/u)\hat{c}) \cdot u^{k-2} \geq \hat{c} \cdot u^{k-2} > 0$. 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 $p(u) < 0$. $\square$

Example 6. There is a grid which contains $\sqrt{3}$ and $1/2$ or, more generally, any c of the form $c = \sqrt{b^2 - 1}$ and $1/b$ for some fixed integer $b \geq 2$.

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 $\ell = 0$ and $e_\ell = 2$,
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 $u \geq 1$ and $p_3(u) = 0$ and $b \geq 2$, one has $1 - b^2u^{-1} \geq 0$ and $u \geq b^2$ and $\sum_{k \leq 0} u^k \leq 2$. 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) $a'' \geq -\hat{c}$ and $a' = a_{k-1} + 2b^2a_k > \hat{c}$ then the following holds: In the case (a), $a_k \geq 1000b^3 - 6b^2$ and the value of the sum is at least

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

where the tail sum $12b^4$ estimates that all digits $a_{k'}$ with $k' \leq k-2$ 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 greater than $\hat{c}$ while all other coefficients are of absolute value below $\hat{c}$, in particular as $\hat{c} \geq 6b^2$, so that, since $u \geq 2$,

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

Thus the algorithm terminates as required. □

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 aligned, that means, the position between the symbols at location 0 and location -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 7. 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 exists 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 [15] 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 8. *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 $v \leq 0\,[(v, w) \in L_e \Leftrightarrow (v, w) = (0, 0)]$,
$\forall w \in G\,[(1, w) \in L_e \Leftrightarrow 0 \leq w \wedge w \leq y]$,
$\forall w \in G\,[(x, w) \in L_e \Leftrightarrow 0 \leq w \wedge w \leq z]$.

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 multiplying with constants is automatic, as the ring has a semiautomatic multiplication. Thus a point is in the triangle or on its border 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. $\square$

Note that this also implies that polygons with all corner points being in $G \times G$ are regular subsets of the plane $G \times G$.

Proposition 9. *Moving a polygon by a distance (v, w) can be done in any grid, as it only requires adding (v, w) to the coordinates of each points. However, rotating by $30°$ or $45°$ is possible in some but not all grids.*

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 6. 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 5 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. In particular, the grids $\mathbb{D}_b$ do not allow to multiply by roots and those grids of the form

$\mathbb{Z} \cup \sqrt{c} \cdot \mathbb{Z}$ in prior work [9] do not allow to divide by 2. Furthermore, the authors are not aware of any grid which has both, $\sqrt{2}$ and $\sqrt{3}$ and thus allow to rotate by both, 30° and 45°. □

Remark 10. A grid allows to represent all equilateral triangles with side-length in G iff $\sqrt{3}$ and $1/2$ are both in G. This is in particular true for grids which allow to rotate by 60° and it is false for all grids of the type $\mathbb{D}_b$.

For a proof, assume that in a plane $G \times G$, an equilateral triangle is represented with all three corner points in $G \times G$. So let $(x, y), (x', y')$ be two corner points in $G \times G$. Now the third corner point (x'', y'') has either the coordinates

$$((x + x')/2 + (y - y') \cdot \sqrt{3}/2, (y + y')/2 + (x' - x) \cdot \sqrt{3}/2)$$

or the coordinates

$$((x + x')/2 - (y - y') \cdot \sqrt{3}/2, (y + y')/2 - (x' - x) \cdot \sqrt{3}/2)$$

and in either case, a scaling of either $x - x'$ or $y - y'$ by $\sqrt{3}/2$ has to be in G and at least one of these is nonzero.

This is impossible in the case of $G = \mathbb{D}_b$, as all members of G are rational.

However, it can always be done in the grid from Example 6 with $c = \sqrt{3}$ and $b = 2$, as in that grid there are automatic functions which divide the input by 2 and which multiply the input with $\sqrt{3}$, respectively. In this grid, for any base-line given by two distinct points in $G \times G$, one can find the third point in order to obtain the equilateral triangle with the three corner points.

Remark 11. Note that one represents a word $a_5 a_4 \ldots a_1 a_0.a_{-1} a_{-2} \ldots a_{-7}$ also by starting with a_0 and then putting alternatingly the digits of even and odd indices giving

$$a_0 a_1 a_{-1} a_2 a_{-2} \ldots a_5 a_{-5} 0 a_{-6} 0 a_{-7}$$

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 12. 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, as otherwise one could reconstruct the multiplication.

For p being a prime power, in the grid $(\mathbb{D}_p, +, =, <; \cdot)$ from Example 3, the relation $\equiv$ is semiautomatic using the representation given in Remark 11. To see this, for a given area $\ell \cdot p^k$, (i) one can disjunct over all factorisations $\ell_1 \cdot \ell_2$ of ℓ which are pairs of natural numbers not divisible by p, then (ii) check whether the length of the sides of a given rectangle are of the form $\ell_1 \cdot p^i$ and $\ell_2 \cdot p^j$ with $i + j = k$, where i, j are the positions of the last nonzero digits in the p-adic representation of the lengths and (iii) check, by Remark 11, whether $i + j$ are the given constant k. Note that representations using prime powers can be translated into representations based on the corresponding primes.

However, for grids such as $(\mathbb{D}_6, +, =, <; \cdot)$, where b is a composite number other than a prime power, this method does not work. This is mainly because one needs to use base 6 for the comparison $<$ and then a finite automaton cannot see whether the two sides are, for example, of lengths 2^k and 2^{-k} when recognising squares of area 1. Knowing that this method does not work, however, does not say that no other method works. It is an open problem whether one can find a semiautomatic grid which allows to divide by 6 and to represent axis-parallel rectangles in a way such that checking whether two rectangles have same area is semiautomatic. The same applies to the grids of Examples 5 and 6.

4 Cube Roots

Jain, Khoussainov, Stephan, Teng and Zou [9] did not find any example of other roots than square roots to be represented in a semiautomatic ordered ring. The following example represents a cube root.

Example 13. There is a grid which contains $\sqrt[3]{7}$. Furthermore, there is a grid which contains $\sqrt[3]{65}$ and $1/2$.

For the first, as one does not want to represent a proper rational, p_1 is not needed. For this, one chooses

1. $u^{-1} = 2 - \sqrt[3]{7}$,
2. $p_2(u) = 2 - u^{-1} = \sqrt[3]{7}$,
3. $p_3(u) = 1 - 12u^{-1} + 6u^{-2} - u^{-3} = 0$ and $p_4(u) = -p_3(u)$ with $\ell = -1$,
4. Instead of a flat $\hat{c}$, one uses a bit different bound for the algorithm, namely $|a_k| \leq 16\hat{c}$, $|a_{k-1}| \leq 4\hat{c}$ and $|a_{k-2}| \leq \hat{c}$ for $\hat{c} = 360$.

The equations for p_3, p_4 follow from $p_3(u) = (p_2(u))^3 - 7$. Note that $11 < u < 12$ and $u^{-1} + u^{-2} + \ldots \leq 1/10$. Furthermore, the coefficients in the normal form are between -12 and $+12$ and, when three numbers are added coefficientwise, between -36 and $+36$. Let $p = \sum_k a_k \cdot u^k$ be such a sum of three numbers whose sign has to be determined; all the coefficients have absolute values up to 36.

The main thing is that the algorithm can detect the sign of the number whenever the first three coefficients overshoot for the first time. Note that they start with $(0, 0, 0)$ and so one runs the updates $a_{k-k'} = a_{k-k'} - a_k \cdot d_{-k'}$ simultaneously for $k' = 1, 2, 3$ and then sets $a_k = 0$ and $k = k - 1$ Here $d.$ are coefficients of p_3. Assume that the update would make the coefficients to overshoot for the first time and let $k, a_k, a_{k-1}, a_{k-2}, a_{k-3}$ and $p = \sum_{k'} a_{k'} u^{k'}$ denote the values just before the update.

Without loss of generality, assume that $a_k > 0$ and it will be shown that this implies that the polynomial p would be positive. Note that before the update, for all $k' < k$, $|a_{k'}| \leq 4\hat{c}$ and $|\sum_{k'<k} a_{k'} u^{k'}| \leq 0.4 \cdot \hat{c} \cdot u^k$.

If a_{k-3} grows above $\hat{c}$ at the update then $a_k \geq \hat{c} \cdot 9/10$ and $p > (0.9 \cdot \hat{c} - \sum_{k' \leq -1} a_{k+k'} \cdot u^{k'}) \cdot u^k \geq (0.9 - 0.4) \cdot \hat{c} \cdot u^k > 0$.

If a_{k-2} grows below $-4\hat{c}$ at the update but a_{k-3} stays inside the bound then $a_k \geq \hat{c} \cdot 3 \cdot 1/6$ and $p > (0.5 - 0.4) \cdot \hat{c} \cdot u^k > 0$.

If a_{k-1} grows beyond $16\hat{c}$ at the update but a_{k-2} and a_{k-3} stay inside their bounds then $a_k \geq \hat{c} \cdot 12 \cdot 1/12$ and $p > (1 - 0.4) \cdot \hat{c} \cdot u^k > 0$.

So the above test can detect the sign of p by just looking at the sign of a_k in the event that the next step overshoots; if, through the runtime, it never overshoots, then one can deduce the sign of p from a table look-up of the final values of the coefficients in the tracked window. Thus there is a semiautomatic grid containing $\sqrt[3]{7}$.

Furthermore, for the grid which contains $\sqrt[3]{65}$ and $1/2$, one chooses

1. $u^{-1} = (\sqrt[3]{65} - 4)/2$, note that $96.49 \leq u \leq 96.50$,
2. $p_1(u) = 4((u^{-1} + 2)^3 - 8) = 48u^{-1} + 24u^{-2} + 4u^{-3} = 1/2$,
3. $p_2(u) = 2u^{-1} + 4 = \sqrt[3]{65}$,
4. $p_3(u) = 1 - 96u^{-1} - 48u^{-2} - 8u^{-3} = 0$ and $p_4(u) = -p_3(u)$ with $\ell = -1$,
5. Instead of a flat $\hat{c}$, one uses a bit different bound for the algorithm, namely $|a_k| \leq 19\hat{c}$, $|a_{k-1}| \leq 7\hat{c}$ and $|a_{k-2}| \leq \hat{c}$ for $\hat{c}$ very large, for example $\hat{c} = 285000$.

Note that $96.49 < u < 96.50$ and $u^{-1} + u^{-2} + \ldots \leq 1/95$. Furthermore, the coefficients in the normal form are between -95 and $+95$ (both inclusive) and, when three numbers are added coefficientwise, between -285 and $+285$ (both inclusive). Let $p = \sum_k a_k \cdot u^k$ be such a sum of three numbers whose sign has to be determined; all the coefficients have absolute values up to 285.

Again the algorithm can detect the sign of the number whenever the first three coefficients overshoot for the first time. Note that they start with $(0, 0, 0)$ and so one runs the updates $a_{k-k'} = a_{k-k'} - a_k \cdot d_{-k'}$ simultaneously for $k' = 1, 2, 3$ and then sets $a_k = 0$ and $k = k - 1$. Here $d.$ are coefficients of p_3. Assume that the update would make the coefficients to overshoot for the first time and let $k, a_k, a_{k-1}, a_{k-2}, a_{k-3}$ and $p = \sum_{k'} a_{k'} u^{k'}$ denote the values just before the update.

Without loss of generality, assume that $a_k > 0$ and it will be shown that this implies that the polynomial p would be positive. Note that before the update, for all $k' < k$, $|a_{k'}| \leq 7\hat{c}$ and $|\sum_{k' < k} a_{k'} u^{k'}| \leq 7/95 \cdot \hat{c} \cdot u^k \leq 0.074 \cdot \hat{c} \cdot u^k$.

If a_{k-3} grows above $\hat{c}$ at the update then $8a_k \geq \hat{c} \cdot 999/1000$ and $a_k \geq 0.1\hat{c}$ and $p > (0.1 \cdot \hat{c} - \sum_{k' \leq -1} a_{k+k'} \cdot u^{k'}) \cdot u^k \geq (0.1 - 0.074) \cdot \hat{c} \cdot u^k > 0$.

If a_{k-2} grows above $7\hat{c}$ at the update then $48a_k \geq (7\hat{c} - \hat{c})$ and $a_k \geq \hat{c}/8 \geq 0.1\hat{c}$ and $p > (0.1 - 0.074) \cdot \hat{c} \cdot u^k > 0$.

If a_{k-1} grows beyond $19\hat{c}$ at the update then $96a_k \geq \hat{c} \cdot (19 - 7)$ and $a_k \geq \hat{c}/8 \geq 0.1\hat{c}$ and $p > (0.1 - 0.074) \cdot \hat{c} \cdot u^k > 0$.

So the test works well and similarly the test also gives that $p < 0$ in the case that there is an overshooting with negative a_k. Thus there is a semiautomatic grid containing $\sqrt[3]{65}$ and $1/2$. The method could be used to generate other examples of grids with cube roots and fractions.

5 Representing All Reals

The authors assume for this section that the readers are familiar with the theory of ω-automata. Here ω-words are mappings $k \mapsto a_k$ from $\mathbb{Z}$ to the set of digits such that, for some k, all a_h with $h > k$ are 0. An ω-automaton starts at an arbitrary k such that all a_h with $h \geq k$ are 0 in a start state and then reads $a_k a_{k-1} a_{k-2} \ldots$ and updates, on each a_h read, its state in the same way as a finite automaton. If the ω-automaton goes infinitely often through an accepting state, then it accepts the ω-word, else it rejects the ω-word. In the case of determining the sign, there is a positive acceptance and a negative acceptance; rejecting means that the ω-word represents 0. Note that for an ω-word to be accepted, the ω-automaton needs just to have some accepting run and not all runs need to be accepting.

Now a comment on the ω-automatic approach [3,5,11]. The reals with addition and multiplication and infinite integral domains in general are not ω-automatic [23,24]. It is also clear that $(\mathbb{R}, +; \cdot)$ is not ω-semiautomatic, as one otherwise would need uncountably many different ω-automata for recognising the uncountably many functions $x \mapsto r \cdot x$ for constants $r \in \mathbb{R}$. So the best what one can expect is that $(\mathbb{R}, +, <, =)$ is ω-automatic and that there are countably many functions $f_r : x \mapsto r \cdot x$ which are ω-automatic as well. These functions certainly include, independent of the ring representation, all f_r with $r \in \mathbb{Q}$, as one only verifies the relation $x \cdot r = y$ and for $r = i/j$ this is equivalent with verifying $x \cdot i = y \cdot j$ with i, j are integers and multiplication with integers can be realised by repeated self-addition. The following result shows that one can carry over ideas of Theorem 4 to ω-automatic structures in order to get that multiplication with all constants from $\mathbb{Q}[\sqrt{b}]$ are ω-automatic for all natural numbers $b \geq 2$. As there is no need to implement the division by 2 explicitly, as that comes for free, one can use the previously known representations [9, Theorem 26].

Theorem 14. *There is an ω-automatic representation of the reals where addition, subtraction and comparisons are ω-automatic and furthermore the multiplication with any constant from $\mathbb{Q}[\sqrt{b}]$ is also an ω-automatic unary function.*

Proof. One uses the representation of [9, Theorem 26]. For $b \geq 2$ which is not a square, one can use that there are natural numbers $d, e > 3$ with $d^2 = be^2 + 1$. Now one chooses $u = d + e\sqrt{b}$ and as shown by Jain, Khoussainov, Stephan, Teng and Zou [9, Theorem 26], $u + u^{-1} = 2d$. Given any ω-word $\sum_{k \in \mathbb{Z}} a_k u^k$ where all sufficiently large k satisfy $a_k = 0$, one can always choose the largest coefficient with $|a_k| > 2d$ and let s be the sign of a_k and update $a_{k+1} = a_{k+1} + s, a_k = a_k - 2ds, a_{k-1} = a_{k-1} + s$; either this algorithm terminates or runs forever.

In the case that all coefficients are initially bounded by a constant $\hat{c}$, the algorithm produces in the limit an ω-word of the same value $\sum_k a_k u^k$ such that all k satisfy $|a_k| \leq 2d$: For the verification, one takes a value $\tilde{u} > 1$ such that $2d - \tilde{u} - 1/\tilde{u} > 0$. Starting with an ω-word $p_0(u)$, let $p_n(u)$ be the ω-word after the n-th iteration and let $q_n(\tilde{u})$ be $\sum_{k \in \mathbb{Z}} |a_k| \tilde{u}^k$ where the coefficients are taken from $p_n(u) = \sum_{k \in \mathbb{Z}} a_k u^k$. Due to the absolute bounds on the coefficients and due to $a_k = 0$ for all sufficiently large k, both $p_n(u)$ and $q_n(\tilde{u})$ converge absolutely. Note that the value $p_n(u) = \sum_k a_k u^k$ is for all n the same, as every $u^{k+1} - 2du^k + u^{k-1}$ has the value

0 and $p_{n+1}(u) = p_n(u) + s(u^{k+1} - 2du^k + u^{k-1})$ for some k and s. Furthermore,

$$q_{n+1}(\tilde{u}) \leq q_n(\tilde{u}) - (2d - \tilde{u} - 1/\tilde{u}) \cdot \tilde{u}^k$$

for the k from the update of p_n. Furthermore, $q_n(\tilde{u}) \geq 0$ for all n. For that reason, the sequence of the $q_n(\tilde{u})$ converges from above to some number. Thus, for each k there can only be finitely many updates where some digit at or above k changes. Thus the pointwise limit p_∞ of all p_n exists and this limit satisfies that all coefficients have at most the absolute value $2d$. Furthermore, all $p_n(u)$ have the same value. Furthermore, if one looks at the difference ω-words $p_n - p_\infty$ then its coefficients converge at $\tilde{u}$ absolute and therefore converge also absolute when u is taken, as $u^{-1} < \tilde{u}^{-1}$ and for almost all n, only nonzero negative-indexed coefficients occur. Thus the $p_n(u)$ converge to $p_\infty(u)$ and thus $p_\infty(u)$ has the same value as $p_n(u)$ while all its coefficients are between $-2d$ and $+2d$. Thus all real numbers can be represented in the given system.

For verifying that addition and comparison are ω-automatic, one shows that given $p = r_1 + r_2 - r_3$ computes by coefficientwise addition and subtraction of the three numbers r_1, r_2, r_3 given in the normal form, there is a deterministic Büchi automaton which recognises, on the ω-word of all a_k, the sign of $p(u) = \sum_{k \in \mathbb{Z}} a_k u^k$ where all sufficiently large k have $a_k = 0$. The digits are bounded by $6d$. The algorithm is like algorithm C in Theorem 4, when k is the largest nonzero coefficient, the algorithm starts with the memory $(a_{k+2}, a_{k+1}) = (0, 0)$ and then in each round updates on reading a_k the memory from (a_{k+2}, a_{k+1}) to $(a_{k+1} + 2da_{k+2}, a_k - a_{k+2})$, where the first number has at most the absolute value $306d^2$ and the second number has at most the absolute value $106d$. These values can only overshoot when a_{k+2} before the update has at least the absolute value $100d$. So assume, without loss of generality, that $a_{k+2} > 100d$, and that it is the first time where an overshooting will happen as a consequence of an upcoming update. As the overshooting has not yet happened, all a_h with $h < k + 2$ satisfy $|a_h| \leq 106d$. Note that

$$100d > 106d \cdot \sum_{k<0} u^k \geq 106d \cdot (1/d + 1/d^2 + \ldots),$$

as $d > 3$ and $1/d + 1/d^2 + \ldots \leq 1/3$. Thus the algorithm can always indicate that the number is positive in the case that an overshooting happens at an update where the first component of the old memory pair is positive and the algorithm can indicate that the number is negative in the case that an overshooting happens at an update where the first component of the old memory pair is negative. Furthermore, as there are only finitely many possibilities for the two components in the memory, the algorithm works overall with finite memory and can be implemented as a Büchi automaton which goes into the right accepting state (for positive or negative) at that moment that the sign of p is known and which then stays in this state forever.

While this allows directly to implement addition, subtraction and comparisons, one has still to verify that multiplications with finite polynomials $q(u)$ work. For the latter, note that a shift by a costant amount of positions, say from $\sum_k a_k u^k$ to $\sum_k a_k u^{k+h}$ for some constant h, is an ω-automatic function. Thus multiplication with fixed positive or negative powers of u is ω-automatic. Furthermore, repeated addition is ω-automatic. For multiplication with rationals involving u like $x \mapsto (3+u)/(8-u^3) \cdot x$ can be mapped back to multiplication with polynomials:

In the theory of ω-automatic functions, one does not compute the $y = (3+u)/(8-u^3) \cdot x$ directly but one compares whether y is the result of that operation. For that reason, one can just compare whether $y \cdot (8-u^3) = x \cdot (3+u)$ and this can be implemeted via multiplication with fixed polynomials in u. $\square$

This also works with the field of $\mathbb{Q}$ extended by $\sqrt[3]{7}$ or $\sqrt[3]{65}$ using the formulas and algorithms given in Example 13. So one has the following corollary.

Corollary 15. *For any semiautomatic subring $(G, +, <, =; \cdot)$ of the reals constructed by methods in the present work, there is an ω-automatic representation of the reals where addition, subtraction and comparisons are ω-automatic and furthermore the multiplication with any constant from G is also an ω-automatic unary function.*

6 Conclusion

Jain, Khoussainov, Stephan, Teng and Zou [9] studied semiautomatic structures and in particular semiautomatic ordered rings where addition, subtraction, order and equality are in fact automatic. In particular they showed that for all nonsquare positive integers c the ordered ring $(\mathbb{Z}[\sqrt{c}], +, <, =; \cdot)$ is semiautomatic. The present work shows that for certain roots one can also add in the fraction $1/2$ so that the rings $(\mathbb{D}_2[\sqrt{c}], +, <, =; \cdot)$ with $c = 2, 3$, respectively, are semiautomatic.

The case $c = 3$ has applications in geometry, as one can take the domain G of the ring as a grid in order to represent geometric objects with corner points in $G \times G$. This grid G in particular allows to represent equilateral triangles and also allows to rotate objects by $30°$ in the representation. For the case $c = 2$, one obtains a grid where one can rotate by $45°$. It is unknown whether one can make a grid which allows to rotate by $15°$, the challange would be that one has to get $\sqrt{2}$ and $\sqrt{3}$, $1/2$ all three into the grid in a way that addition, comparison, equality and multiplication with fixed elements from the grid are automatic.

Further results are investigations into which collections of geometric objects form an automatic family; while the collection of all triangles with corner points in $G \times G$ is for no grid G an automatic family, all triangles and polygones with corner points in $G \times G$ are always regular. Axis-parallel rectangles form an automatic family and for some grids, the relation saying that two axis-parallel rectangles have the same size is semiautomatic.

Finally the paper provides two semiautomatic rings which contain $\sqrt[3]{7}$ and $\sqrt[3]{65}$, respectively, and these methods can generate further such examples. Some subsequent discussions transfer the results to ω-automatic structures. For ω-automatic structures, as for every real r and every integer b the number r/b always exists, one does not need, as in the case of countable grids, to make sure that one can divide by b explicitly; it comes for free whenever addition is ω-automatic, that is, if one has an ω-automatic model of the reals with addition and order, then the multiplication with every fixed rational is automatic; so the main goal is to extend this also to multiplication with some irrationals and this can be done for all square-roots of positive integers by transferring results from prior work and also for $\sqrt[3]{7}$ and $\sqrt[3]{65}$ by applying methods in the present work. Note that enabling the multiplication of different irrationals might require different representations and therefore the current methods do not give a single ω-automatic structure where the multiplication with each algebraic number is ω-automatic.

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. Bernard Boigelot, Sebastien Jodogne and Pierre Wolper. An effective decision procedure for linear arithmetic over the integers and reals. *ACM Transactions on Computational Logic*, 6(3):614–633, 2005.
4. John Case, Sanjay Jain, Samuel Seah and Frank Stephan. Automatic functions, linear time and learning. *Logical Methods in Computer Science*, 9(3), 2013.
5. Karel Culik II and Jarkko Kari. Computational fractal geometry with WFA. *Acta Informatica*, 34:151–166, 1997.
6. 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.
7. Bernard R. Hodgson. *Théories décidables par automate fini.* Ph.D. thesis, Département de mathématiques et de statistique, Université de Montréal, 1976.
8. Bernard R. Hodgson. Décidabilité par automate fini. *Annales des sciences mathématiques du Québec*, 7(1):39–57, 1983.
9. Sanjay Jain, Bakhadyr Khoussainov, Frank Stephan, Dan Teng and Siyuan Zou. Semiautomatic structures. *Theory of Computing Systems*, 61(4):1254–1287, 2017.
10. Sanjay Jain, Yuh Shin Ong, Shi Pu and Frank Stephan. On automatic families. *Proceedings of the eleventh Asian Logic Conference* in honour of Professor Chong Chitat on his sixtieth birthday, pages 94–113, World Scientific, 2012.
11. Helmut Jürgensen, Ludwig Staiger and Hideki Yamasaki. Finite automata encoding geometric figures. *Theoretical Computer Science*, 381(2–3):20–30, 2007.
12. Olga Kharlampovich, Bakhadyr Khoussainov and Alexei Miasnikov. From automatic structures to automatic groups. *Groups, Geometry and Dynamical Systems*, 8(1):157–198, 2014.
13. Bakhadyr Khoussainov and Mia Minnes. Three lectures on automatic structures. *Logic Colloquium 2007*, Proceedings. *Lecture Notes in Logic*, 35:132–176, 2010.
14. 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.
15. Bakhadyr Khoussainov, Andre Nies, Sasha Rubin and Frank Stephan. Automatic structures: richness and limitations. *Logical Methods in Computer Science*, 3(2), 2007.
16. André Nies. Describing Groups. *The Bulletin of Symbolic Logic*, 13(3):305-339, 2007.

17. André Nies and Pavel Semukhin. Finite automata presentable Abelian groups. *Annals of Pure and Applied Logic*, 161:458–467, 2009.
18. André Nies and Richard Thomas. FA-presentable groups and rings. *Journal of Algebra*, 320:569-585, 2008.
19. 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.
20. Sasha Rubin. Automata presenting structures: a survey of the finite string case. *The Bulletin of Symbolic Logic*, 14:169–209, 2008.
21. 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.
22. Todor Tsankov. The additive group of the rationals does not have an automatic presentation. *The Journal of Symbolic Logic*, 76(4):1341–1351, 2011.
23. Faried Abu Zaid, Erich Grädel and Lukasz Kaiser. The Field of Reals is not omega-automatic. *Twenty-Ninth Symposium on Theoretical Aspects of Computer Science*, STACS 2012, pages 577–588, HAL archives-ouvertes, 2012.
24. Faried Abu Zaid, Erich Grädel, Lukasz Kaiser and Wied Pakusa. Model-theoretic properties of ω-automatic structures. *Theory of Computing Sytems*, 55:856–880, 2014.