AUTOMATIC STRUCTURES AND THEIR COMPLEXITY

BAKHADYR KHOUSSAINOV AND MIA MINNES

1. Introduction

In recent years there has been increasing interest in the study of structures that can be presented by automata. The underlying idea in this line of research consists of applying properties of automata and techniques of automata theory to decision problems that arise in logic and applications. A typical example of a decision problem is the **model checking problem**, stated as follows. For a structure $\mathcal{A}$ (e.g. a graph, a fragment of the arithmetic, the real numbers with addition) design an algorithm that, given a formula $\phi(\bar{x})$ in a formal logical system and a tuple $\bar{a}$ from the structure, decides if $\phi(\bar{a})$ is true in $\mathcal{A}$. In particular, when the formal system is the first order predicate logic or the monadic second order logic, we would like to know if the theory of the structure, that is the collection of all sentences of the logic true in the structure, is decidable. Büchi used automata to prove that the monadic second order theory of the successor function on ω is decidable ([**?**], [**?**]). Rabin, in [**?**], extended this by proving decidability of the monadic second order theory of the binary tree. There have been numerous applications and extensions of these results in logic, algebra, verification, model checking, and databases ([**?**] is an algebra application; [**?**], [**?**] treat logics and verifications; [**?**], and [**?**] give applications to databases). Using simple closure properties and the decidability of the emptiness problem for finite and tree automata, one can easily prove that the first order (and monadic second order) theories of some well-known structures are decidable. Examples of such structures are Presburger arithmetic and some of its extensions, the term algebra, real numbers under addition, finitely generated abelian groups, and the atomless Boolean algebra. Direct proofs of these results, without the use of automata, require non-trivial technical work.

A structure $\mathcal{A} = (A; R_0, \ldots, R_m)$ is **automatic** if the domain A and all the relations $R_0, \ldots, R_m$ of the structure are recognized by finite automata (precise definitions are in the next section). For instance, an automatic graph is one whose set of vertices and set of edges can be recognized by finite automata. There are several motivating results that are foundational for the development of the theory of automatic structures. Khoussainov and Nerode proved that for any given automatic structure there is an algorithm that solves the model checking problem in the first order logic (see [**?**]). In particular, the first order theory of the structure is decidable. This result is extended by adding the $\exists^{\infty}$ (there are infinitely many) and $\exists^{n,m}$ (there are m many mod n) quantifiers to the first order logic (see [**?**], [**?**]). Blumensath and Grädel proved a logical characterization theorem stating that automatic structures are exactly those definable in the fragment of arithmetic $(\omega; +, |_2, \leq, 0)$, where $+$ and $\leq$ have their usual meanings and $|_2$ is a weak divisibility predicate for which $x|_2y$

Date: July 24, 2007.

iff x is a power of 2 and divides y (see [?]). In addition, for some classes of automatic structures there are characterization theorems that have direct algorithmic implications. For example, in [?], Delhommé proved that automatic well-ordered sets are all strictly less than ω^ω. Using this characterization, [?] gives an algorithm which decides the isomorphism problem for automatic well-ordered sets. The algorithm is based on extracting the Cantor normal form for the ordinal isomorphic to the given automatic well-ordered set . Another characterization theorem of this ilk gives that automatic Boolean algebras are exactly those that are finite products of the Boolean algebra of finite and co-finite subsets of ω [?]. Again, this result can be used to show that the isomorphism problem for automatic Boolean algebras is decidable.

There is also a body of work devoted to the study of resource-bounded complexity of the model checking problem for automatic structures. One example is the following dichotomy. On the one hand, Grädel and Blumensath ([?]) constructed examples of automatic structures whose first order theories are non-elementary. On the other hand, Lohrey in [?] proved that the first order theory of any automatic graph of bounded degree is elementary. It is noteworthy that when both a first order formula ϕ and an automatic structure $\mathcal{A}$ are fixed, determining if a tuple $\bar{a}$ from $\mathcal{A}$ satisfies $\phi(\bar{x})$ can be done in linear time. There are also feasible time bounds on deciding the first order theories of automatic structures over the unary alphabet ([?], [?]).

Most of the current results about automatic structures, including the ones mentioned above, demonstrate that in various concrete senses automatic structures are not complex from a logical point of view. In this paper we probe the question of measuring the complexity of automatic structures. We use well-established concepts from both logic and model theory including ordinal heights (of well-founded relations), Scott ranks of structures, and Cantor-Bendixson ranks (of trees). We briefly describe these measures of complexity and state the results of this paper with respect to each of them.

A relation R is **well-founded** if there is no infinite sequence $x_1, x_2, x_3, \ldots$ such that $(x_{i+1}, x_i) \in R$ for $i \in \omega$. In practice, well-foundedness can be established by providing a ranking function f that associates ordinals with the elements in the set and satisfies the postulate that $f(y) < f(x)$ whenever $(y, x) \in R$. In computer science, well-founded relations are of interest due to a natural connection between well-founded sets and terminating programs (see [?]). Given a program P, we say that the program is **terminating** if every computation of P from an initial state is finite. If there is a computation from a state x to y then we say that y is reachable from x. Thus, the program is terminating if the collection of all states reachable from the initial state is a well-founded set (see [?] for more details). Since all automatic structures are computable, the obvious bound for the ranks of automatic well-founded relations is ω_1^{CK} (the first non-computable ordinal). Sections 3 to 5 study whether or not the ω_1^{CK} bound is sharp. The theorem below shows that when automatic well-founded relations are in fact partial orders then their ranks are strictly below ω^ω.

Theorem 3.5. *For each ordinal α, α is the rank an automatic well-founded partial order if and only if $\alpha < \omega^\omega$.*

In contrast, the next theorem shows that in the general case of automatic well-founded relations, the ordinal ω_1^{CK} is indeed the sharp bound.

Theorem 5.1. *For each infinite computable ordinal, $\alpha < \omega_1^{CK}$, there is an automatic well-founded relation $\mathcal{A}$ such that $\alpha \leq r(\mathcal{A}) \leq \omega + r(\mathcal{A})$.*

Section 6 is devoted to building automatic structures of high Scott ranks. The concept of Scott rank comes from the well-known theorem of Scott stating that for every countable structure $\mathcal{A}$ there exists a sentence ϕ in $L_{\omega_1,\omega}$-logic that characterizes $\mathcal{A}$ up to isomorphism [?]. The minimal ordinal rank of such a formula is called the Scott rank of $\mathcal{A}$. Informally, for those familiar with Ehrenfeucht-Fraïssé games, the Scott rank is the minimal ordinal length of a game in which Duplicator can show that two given structures $\mathcal{A}$ and $\mathcal{B}$ are isomorphic in the Ehrenfeucht-Fraïssé game between Spoiler and Duplicator [?]. A known upper bound on the Scott rank of computable structures gives that the upper bound for the Scott rank of automatic structures is $\omega_1^{CK}+1$. However, until now, all the known examples of automatic structures have had small Scott ranks. Results in [?], [?], [?], and [?] have also suggested that the Scott ranks of automatic structures could be bounded by small ordinals. Section 6 defies this intuition and leads to the following.

Corollary 6.9. *For each infinite ordinal α such that $\alpha \leq \omega_1^{CK}+1$, there is an automatic structure of Scott rank α.*

In particular, the proof of this theorem implies that the isomorphism problem for automatic structures is Σ_1^1-complete.

In the last two sections we investigate Cantor-Bendixson ranks of automatic trees. A **tree** is a partially ordered set $(T, \leq)$ such that there is a $\leq$-minimal element of T, and each subset $\{x \in T : x \leq y\}$ is finite and is linearly ordered under $\leq$. A **successor tree** is a pair (T, S) such that the reflexive and transitive closure $\leq_S$ of S produces the tree $(T, \leq_S)$. The **derivative** of a tree $\mathcal{T}$ is obtained by removing all the isolated paths of the tree. One applies the derivative operation to $\mathcal{T}$ until a fixed point is reached. The minimal ordinal that is needed to reach the fixed point is called the **Cantor-Bendixson (CB) rank** of the tree. Again, the obvious bound on CB ranks of automatic successor trees is ω_1^{CK}. The CB ranks play an important role in logic, algebra, and topology. Informally, the CB rank tells us how far the structure is from algorithmically (or algebraically) simple structures. In [?], it is proved that the CB rank of any automatic partially ordered tree is finite; moreover, there is an algorithm that computes the CB rank of the tree from the automata for the $\leq$ relation on the tree. It has been an open question whether the CB ranks of automatic successor trees can also be bounded. We answer this question in full in the following theorem.

Theorem 8.2. *For any computable ordinal $\alpha < \omega_1^{CK}$ there is an automatic successor tree of CB rank α.*

We thank Moshe Vardi who posed the question about ranks of automatic well-founded relations. We also thank Anil Nerode and Frank Stephan with whom we discussed Scott and Cantor-Bendixson ranks of automatic structures.

2. Preliminaries

A **vocabulary** V is a sequence $(\{F_i^{n_i}\}_{i\in\omega}, \{P_j^{m_j}\}_{j\in\omega}, \{c_k\}_{k\in\omega})$. Here, each $F_i^{n_i}$ is called a function symbol of arity $n_i > 0$, each $P_j^{m_j}$ is a predicate symbol of arity $m_j > 0$, and each c_k is a constant symbol. We put effectiveness condition on the vocabulary by assuming that the functions $i \to n_i$, and $j \to m_j$ are computable. If σ contains no function symbols then we say that it is a relational vocabulary.

A **structure** of the vocabulary V is a tuple $\mathcal{A} = (A; \{F_i^{\mathcal{A}}\}_{i\in\omega}, \{P_j^{\mathcal{A}}\}_{\in\omega}, \{c_k^{\mathcal{A}}\}_{k\in\omega})$, where $F_i^{\mathcal{A}}$, $P_j^{\mathcal{A}}$, and $c_k^{\mathcal{A}}$ are interpretations of the symbols of the vocabulary. As such, $F_i^{\mathcal{A}}$ and $P_j^{\mathcal{A}}$ are functions and predicates (respectively) of appropriate arities, and $c_k^{\mathcal{A}}$ are distinguished elements. When convenient, we may omit the superscripts $\mathcal{A}$. One can replace the functions of a structure with their graphs, thus turning the structure into a purely relational structure (of a new relational vocabulary). This transformation preserves important model theoretic and effective properties. Therefore, from now on, we always deal with purely relational vocabularies and structures. In this paper, all the structures we consider are infinite.

To establish notation, we briefly recall some definitions associated with finite automata. A **finite automaton** $\mathcal{M}$ over an alphabet Σ is a tuple (S, ι, Δ, F), where S is a finite set of **states**, $\iota \in S$ is the **initial state**, $\Delta \subset S \times \Sigma \times S$ is the **transition table**, and $F \subset S$ is the set of **final states**. A **computation** of $\mathcal{A}$ on a word $\sigma_1\sigma_2\ldots\sigma_n$ ($\sigma_i \in \Sigma$) is a sequence of states, say $q_0, q_1, \ldots, q_n$, such that $q_0 = \iota$ and $(q_i, \sigma_{i+1}, q_{i+1}) \in \Delta$ for all $i \in \{0, 1, \ldots, n-1\}$. If $q_n \in F$, then the computation is **successful** and we say that automaton $\mathcal{M}$ **accepts** the word $\sigma_1\sigma_2\ldots\sigma_n$. The **language** accepted by the automaton $\mathcal{M}$ is the set of all words accepted by $\mathcal{M}$. In general, $D \subset \Sigma^\star$ is **finite automaton recognisable**, or **regular**, if D is the language accepted by a finite automaton $\mathcal{M}$.

We now define the concept of n–tape automata for the purpose of defining automata recognisable relations. An n**–tape automaton** can be thought of as a one-way Turing machine with n input tapes [**?**]. Each tape is regarded as semi-infinite, having written on it a word in the alphabet Σ followed by an infinite succession of blanks, denoted by $\diamond$ symbols. The automaton starts in the initial state, reads simultaneously the first symbol of each tape, changes state, reads simultaneously the second symbol of each tape, changes state, etc., until it reads a blank on each tape. The automaton then stops and accepts the n–tuple of words if it is in a final state. The set of all n–tuples accepted by the automaton is the relation recognised by the automaton. Therefore, an n–tape automaton on Σ is a finite automaton over the alphabet $(\Sigma_\diamond)^n$.

Definition 2.1. Define $\Sigma_\diamond = \Sigma \cup \{\diamond\}$ where $\diamond \notin \Sigma$. The **convolution of a tuple** $(w_1, \cdots, w_n) \in \Sigma^{\star n}$ is the string $c(w_1, \cdots, w_n)$ of length $\max_i |w_i|$ over the alphabet $(\Sigma_\diamond)^n$ which is defined as follows. Its k'th symbol is $(\sigma_1, \ldots, \sigma_n)$ where σ_i is the k'th symbol of w_i if $k \leq |w_i|$ and $\diamond$ otherwise.

The **convolution of a relation** $R \subset \Sigma^{\star n}$ is the language $c(R) \subset (\Sigma_\diamond)^{n\star}$ formed as the set of convolutions of all the tuples in R.

Definition 2.2. An n–ary relation $R \subset \Sigma^{\star n}$ is **finite automaton recognisable**, or **regular**, if its convolution $c(R)$ is recognisable by an n–tape automaton.

We now relate n–tape automata to structures.

Definition 2.3. A structure $\mathcal{A} = (A; R_0, R_1, \ldots)$ is **automatic** over Σ if its domain $A \subset \Sigma^\star$ is finite automata recognisable, and there is an algorithm that for each i produces a finite automaton recognising the relation $R_i^A \subset (\Sigma^\star)^{n_i}$. A structure is called **automatic** if it is automatic over some finite alphabet. If $\mathcal{B}$ is isomorphic to an automatic structure $\mathcal{A}$ then we call $\mathcal{A}$ an **automatic presentation** of $\mathcal{B}$ and say that $\mathcal{B}$ is **automatically presentable**.

An example of an automatic structure is the word structure $(\{0,1\}^\star, L, R, E, \preceq)$, where for all $x, y \in \{0,1\}^\star$, $L(x) = x0$, $R(x) = x1$, $E(x, y)$ iff $|x| = |y|$, and $\preceq$ is the lexicographical order. The configuration graph of any Turing machine is another example of an automatic structure. In this example, configurations of the Turing machine are the vertices of the graph; and an edge is put from configuration c_1 to c_2 if the machine can make an instantaneous move from c_1 to c_2. Examples of automatically presentable structures are $(\mathbb{N}, +)$, $(\mathbb{N}, \leq)$, $(\mathbb{N}, S)$, the group $(\mathbb{Z}, +)$, the order on the rationals $(Q, \leq)$, and the Boolean algebra of finite or co-finite subsets of $\mathbb{N}$.

It can be shown that each automatic structure has an automatic presentation over a binary alphabet. Further, we use the following important theorem without reference. For this theorem let $(FO + \exists^\infty + \exists^{n,m})$ denote the logic that extends the first order logic with the quantifiers $\exists^\infty$ (there are infinitely many) and $\exists^{n,m}$ (there are m many mod n).

Theorem 2.4. [?] *If $\mathcal{A}$ is automatic then there exists an algorithm that applied to a $(FO + \exists^\infty + \exists^{n,m})$-definition of any relation R produces an automaton that recognizes the relation. In particular, the $(FO + \exists^\infty + \exists^{n,m})$-theory of $\mathcal{A}$ is decidable.*

As an example we provide a construction that preserves automaticity. For a given structure $\mathcal{A}$, its ω-fold disjoint union is obtained by taking ω many disjoint copies of $\mathcal{A}$.

Lemma 2.5. [?] *If $\mathcal{A}$ is automatic then its ω-fold disjoint union is automatically presentable.*

Proof. Let $\mathcal{A} = (A; R_1, R_2, \ldots)$ be automatic. Define $\mathcal{A}' = (A \times 1^\star; R'_1, R'_2, \ldots)$ by

$$\langle (x, i), (y, j) \rangle \in R'_m \qquad \Longleftrightarrow \qquad i = j \ \& \ \langle x, y \rangle \in R_m, \quad m = 1, 2, \ldots.$$

It is clear that $\mathcal{A}'$ is automatic and is isomorphic to the ω-fold disjoint union of $\mathcal{A}$. $\square$

The class of automatic structures is a proper subclass of the computable structures. The theory of computable structures is an active area of research in modern computability and model theory (see [?], [?]). In this paper, we will be coding computable structures into automatic ones. Therefore, we briefly mention some basic definitions and facts about computable structures.

Definition 2.6. A **computable structure** is a structure $\mathcal{A} = (A; R_1, R_2, \ldots)$ whose domain and relations are all uniformly computable.

Here, uniformity means that there is an effective procedure that, given an i and a tuple $(n_1, \ldots, n_{m_i})$ from the domain A, decides if the tuple is in R_i. In this paper we mostly use finite vocabularies, and hence the uniformity condition can be omitted. The domains of infinite computable structures can always be identified with the set ω of natural numbers. Under this assumption, we introduce a new constant symbol c_n for each $n \in \omega$ and interpret

c_n as the number n. We expand the vocabulary of each structure to include these new constants c_n. In this context, $\mathcal{A}$ being computable is equivalent to the **atomic diagram** of $\mathcal{A}$ (all quantifier free sentences in the extended vocabulary that are true in $\mathcal{A}$) being a computable set. If $\mathcal{A}$ is computable and $\mathcal{B}$ is isomorphic to $\mathcal{A}$ then we say that $\mathcal{A}$ is a **computable presentation** of $\mathcal{B}$. Note that if $\mathcal{B}$ has a computable presentation then $\mathcal{B}$ has ω many computable presentations.

Many of the results in this paper assume some familiarity with ordinal arithmetic and computable (recursive) ordinals (see, for example, [?] for an introduction). The least non-computable ordinal is denoted by ω_1^{CK}.

3. Ranks of automatic well-founded partial orders

In this section, we consider structures $\mathcal{A} = (A; R)$ with a single binary relation. An element x is said to be **R-minimal for a set** X if for each $y \in X$, $(y, x) \notin R$. The relation R is said to be **well-founded** if every non-empty subset of A has an R-minimal element. This is equivalent to saying that $(A; R)$ has no infinite chains $x_1, x_2, x_3, \ldots$ where $(x_{i+1}, x_i) \in R$ for all i.

A **ranking function** for $\mathcal{A}$ is an ordinal-valued function f such that $f(y) < f(x)$ whenever $(y, x) \in R$. In this case, let $ord(f)$ be the smallest ordinal larger than or equal to all values of f. The structure $\mathcal{A}$ is well-founded if and only if $\mathcal{A}$ has a ranking function. The **ordinal height** of $\mathcal{A}$ (or, the **rank** of $\mathcal{A}$), denoted $r(\mathcal{A})$, is the least ordinal α which is $ord(g)$ for some ranking function g. We now give an equivalent definition for the rank of $\mathcal{A}$. Define the function $r_{\mathcal{A}}$ by induction as follows. For the R-minimal elements x, set $r_{\mathcal{A}}(x) = 0$. Put $r_{\mathcal{A}}(z) = \sup\{r(y) + 1 : (y, z) \in R\}$. Then $r_{\mathcal{A}}$ is a ranking function admitted by $\mathcal{A}$ and $r(\mathcal{A}) = \sup\{r_{\mathcal{A}}(x) : x \in A\}$. For $B \subseteq A$, we write $r(B)$ for the ordinal height of the structure obtained by restricting R to B. The following is well-known:

Lemma 3.1. *For any computable ordinal α there is a computable well-founded relation of rank α.* □

We also use the following lemma about well-founded relations and rank. The proof follows from by induction using the well-foundedness of ordinals and of R.

Lemma 3.2. *For a structure $\mathcal{A} = (A; R)$ where R is well-founded, if $r(\mathcal{A}) = \alpha$ and $\beta < \alpha$ then there is an $x \in A$ such that $r_{\mathcal{A}}(x) = \beta$.* □

For the remainder of this section, we impose one additional condition on R, and assume that R is a partial order. For convenience, we write $\leq$ instead of R. Thus, from now on we consider automatic well-founded partial orders $\mathcal{A} = (A, \leq)$. We will see that in this restricted setting, the ranges of ranking height functions are strictly below ω^ω. We will use the notion of **natural sum of ordinals**. The natural sum of ordinals α, β (denoted $\alpha +' \beta$) is defined recursively by putting $\alpha +' \beta$ as the least ordinal strictly greater than $\gamma +' \beta$ for all $\gamma < \alpha$ and strictly greater than $\alpha +' \gamma$ for all $\gamma < \beta$. Let A_1 and A_2 be disjoint subsets of A such that $A = A_1 \cup A_2$. Consider the partially ordered sets $\mathcal{A}_1 = (A_1, \leq_1)$ and $\mathcal{A}_2 = (A_2, \leq_2)$ obtained by restricting $\leq$ to A_1 and A_2 respectively.

Lemma 3.3. *Under the assumptions above, $r(\mathcal{A}) \leq \alpha_1 +' \alpha_2$, where $\alpha_i = r(\mathcal{A}_i)$, $i = 1, 2$.*

Proof. It suffices to show that there is a ranking function on A whose range is contained in the ordinal $\alpha_1 +' \alpha_2$. The desired function f is defined as follows. For each $x \in A$ consider the partially ordered sets $\mathcal{A}_{1,x}$ and $\mathcal{A}_{2,x}$ obtained by restricting $\leq$ to $\{z \in A_1 \mid z < x\}$ and $\{z \in A_2 \mid z < x\}$, respectively. Define $f(x) = r(\mathcal{A}_{1,x}) +' r(\mathcal{A}_{2,x})$.

We claim that f is a ranking function. Indeed, assume that $x < y$. Then, since $\leq$ is transitive, it must be the case that $\mathcal{A}_{1,x} \subseteq A_{1,y}$ and $\mathcal{A}_{2,x} \subseteq A_{2,y}$. Therefore, $r(\mathcal{A}_{1,x}) \leq r(\mathcal{A}_{1,y})$ and $r(\mathcal{A}_{2,x}) \leq r(\mathcal{A}_{2,y})$. At least one of these inequalities must be strict. To see this, assume that $x \in A_1$ (the case $x \in A_2$ is similar). Then since $x \in A_{1,y}$, it is the case that $r(\mathcal{A}_{1,x}) + 1 \leq r(\mathcal{A}_{1,y})$ by the definition of ranks. By strict monotonicity of natural sum in both arguments, we have that $f(x) < f(y)$. □

Corollary 3.4. *If $r(\mathcal{A}) = \omega^n$ and $A = A_1 \cup A_2$, where $A_1 \cap A_2 = \emptyset$, then either $r(\mathcal{A}_1) = \omega^n$ or $r(\mathcal{A}_2) = \omega^n$.* □

It is clear that automatic partially ordered sets isomorphic to ω^n have rank ω^n. The next theorem shows that ω^ω is the sharp bound on ranks of all automatic well-founded partial orders. Having the corollary above, our proof follows Delhommé [?] and Rubin [?].

Theorem 3.5. *For each ordinal α, α is the rank an automatic well-founded partial order if and only if $\alpha < \omega^\omega$.*

Proof. We assume for a contradiction that there is an automatic well-founded partial order $\mathcal{A} = (A, \leq)$ with $r(\mathcal{A}) = \alpha \geq \omega^\omega$. Let $(S_A, \iota_A, \Delta_A, F_A)$ and $(S_\leq, \iota_\leq, \Delta_\leq, F_\leq)$ be finite automata over Σ recognizing A and $\leq$ (respectively).

By Lemma 3.2, for each natural number n there exists an element $u_n \in A$ such that $r_\mathcal{A}(u_n) = \omega^n$. For each $u \in A$ we define the set

$$u \downarrow = \{x \in A : x < u\}.$$

Note that if $r_\mathcal{A}(u)$ is a limit ordinal then $r_\mathcal{A}(u) = r(u \downarrow)$. We introduce another piece of notation in order to give a finite partition of each set $u_n \downarrow$. For $u, v \in \Sigma^\star$, we denote the set of extensions of v which are $\leq$-below u in $\mathcal{A}$ by X_v^u. Thus,

$$X_v^u = \{vw \in A : w \in \Sigma^\star \ \& \ vw < u\}.$$

Each set of the form $u \downarrow$ can be partitioned based on the prefixes of words as follows:

$$u \downarrow = \{x \in A : |x| < |u| \ \& \ x < u\} \cup \cup_{v \in \Sigma^\star : |v| = |u|} X_v^u.$$

(All the unions above are finite and disjoint.) Hence, applying Corollary 3.4, for each u_n there exists a v_n such that $|u_n| = |v_n|$ and $r(X_{v_n}^{u_n}) = r(u_n \downarrow) = \omega^n$.

We define the following equivalence relations on pairs of finite words of equal lengths:

$$(u, v) \sim (u', v') \iff \Delta_A(\iota_A, v) = \Delta_A(\iota_A, v') \ \& \ \Delta_\leq(\iota_\leq, \binom{v}{u}) = \Delta_\leq(\iota_\leq, \binom{v'}{u'})$$

There are at most $|S_A| \times |S_\leq|$ equivalence classes. Therefore, in the infinite sequence (u_1, v_1), (u_2, v_2), ..., (u_i, v_i), ... there are m, n such that $m \neq n$ and $(u_m, v_m) \sim (u_n, v_n)$.

Lemma 3.6. *For any $u, v, u', v' \in \Sigma^\star$, if $(u, v) \sim (u', v')$ then $r(X_v^u) = r(X_{v'}^{u'})$.*

To prove the lemma, we define the function $f : X^u_v \to X^{u'}_{v'}$ by $f(vw) = v'w$. From the definition of the equivalence relation f is well-defined, bijective, and order preserving. Hence X^u_v is isomorphic to $X^{u'}_{v'}$ as partial orders. Therefore, $r(X^u_v) = r(X^{u'}_{v'})$. This proves the lemma.

By Lemma 3.6, $\omega^m = r(X^{u_m}_{v_m}) = r(X^{u_n}_{v_n}) = \omega^n$, a contradiction with the assumption that $m \neq n$. Therefore, there is no automatic well-founded partial order of rank greater than or equal to ω^n. □

4. Configuration Spaces of Turing Machines

In all the forthcoming constructions, we embed computable structures into automatic structures via configuration spaces of underlying Turing machines. This section serves as an auxiliary technical section in which we recall relevant definitions and establish terminology to be used in later sections. Let $\mathcal{M}$ be an n-tape deterministic Turing machine. The **configuration space** of $\mathcal{M}$ is a directed graph whose nodes are configurations of $\mathcal{M}$. Thus, the nodes are n-tuples, each of whose coordinates represents the contents of a tape. Each tape is encoded as $(w\ q\ w')$, where $w, w' \in \Sigma^\star$ are the symbols on the tape before and after the location of the read/write head, and q is one of the internal states of $\mathcal{M}$. The edges of the graph are all the pairs of the form (c_1, c_2) such that there is an instruction of the machine that transforms the configuration c_1 to c_2. Since each Turing machine has finitely many states and finitely many instructions, the configuration space is an automatic graph. We denote the configuration space of $\mathcal{M}$ by $Conf(\mathcal{M})$. Note that the out-degree of every vertex in $Conf(\mathcal{M})$ is one; the in-degree may be greater than 1 and is bounded by the size of the alphabet and the number of instructions of $\mathcal{M}$.

Definition 4.1. A deterministic Turing machine $\mathcal{M}$ is **reversible** if the in-degree of each vertex in $Conf(\mathcal{M})$ is at most 1.

The configuration space of all reversible Turing machines consist of either chains of finite size, chains of type ω (the order type of the natural numbers), chains of type $\omega^\star + \omega$ (the order type of the integers), or finite cycles. The following result is well-known but we sketch its proof for the completeness of the presentation:

Lemma 4.2. [?] *Any deterministic* 1*-tape Turing machine may be simulated by a reversible* 3*-tape Turing machine.*

Proof. (Sketch) Given a deterministic Turing machine, define a 3-tape Turing machine with a modified set of instructions. The instructions are of the form $\bar{a} \to \bar{b}$, where $\bar{a}$ is referred to as the domain and $\bar{b}$ as the range of the instruction. The modified instructions have the property that neither the domains nor the ranges overlap. On the first tape, the Turing machine performs the computation as the original one would have. However, as it executes each instruction, the machine stores the index of the instruction on the second tape, forming a history. Once the machine enters a state which would have been halting for the original machine, the output of the computation is copied onto the third tape. Finally, the machine runs the computation backwards (which is possible, because of the modification on the instructions) and erases the history tape. The halting (final) configuration of this machine contains the input on the first tape, blanks on the second tape, and the output on the third tape. □

We establish the following notation for a 3-tape reversible Turing machine $\mathcal{M}$ given by the construction in this lemma. A **valid initial configuration** of the $\mathcal{M}$ is of the form $(\lambda\ \iota\ x, \lambda, \lambda)$, for some x in the domain and where λ is the empty string, ι is the initial internal state of $\mathcal{M}$. From the proof above, we observe that a **final (halting)** configuration is of the form $(x, \lambda, \lambda\ q_f\ y)$, with q_f a halting internal state of $\mathcal{M}$. The set of valid initial or final configurations is regular. It is clear from the proof of the lemma that the configuration space of $\mathcal{M}$ contains neither finite cycles nor chains of type $\omega^\star + \omega$. Thus, all the chains are either finite or ω-chains. We call elements of in-degree 0 **bases** (of chains they belong to). We classify the components of the configuration space as chains of the following types.

- **Terminating computation chains**: finite chains whose base is a valid initial configuration, that is, one of the form $(\lambda\ \iota\ x, \lambda, \lambda)$, for $x \in \Sigma^\star$ and ι the distinguished initial state of the Turing machine.
- **Non-terminating computation chains**: infinite chains whose base is a valid initial configuration.
- **Unproductive chains**: chains whose base is not a valid initial configuration.

The following fact says that, no matter how complicated the set described by the Turing machine, the configuration space does not have a very large ordinal height.

Proposition 4.3. *If $G = (A, E)$ is a locally finite graph then either E is well-founded and the rank of E is not above ω or E has an infinite chain.*

Proof. Suppose G is a locally finite graph and E is well-founded. For a contradiction, suppose $r(G) > \omega$. Then there is $v \in A$ with $r(v) = \omega$. By definition, $r(v) = \sup\{r(u) : uEv\}$. But, this implies that there are infinitely many elements E-below v, a contradiction with local finiteness of G. □

5. Ranks of well-founded automatic relations

We return to automatic well-founded relations, and prove the following theorem:

Theorem 5.1. *For each computable ordinal, $\alpha < \omega_1^{CK}$, there is an automatic well-founded relation $\mathcal{A}$ whose rank $r(\mathcal{A})$ is such that $\alpha \leq r(\mathcal{A}) \leq \omega + \alpha$.*

Proof. The proof of the theorem uses properties of Turing machines and their configuration spaces. The idea is to take a computable well-founded relation whose rank is α, and "embed" the relation into an automatic well-founded relation whose rank is close to α.

By Lemma 3.1, let $\mathcal{C} = (C, L_\alpha)$ be a computable well-founded relation of rank α. The construction involves the configuration space of a Turing machine computing the relation L_α. We assume without loss of generality that $C = \Sigma^\star$ for some finite alphabet Σ. Let $\mathcal{M}$ be the Turing machine computing the relation L_α. On each pair (x, y) from the domain, $\mathcal{M}$ halts and outputs "yes" or "no" . By Lemma 4.2, we can assume that $\mathcal{M}$ is reversible. Recall that the configuration space of $\mathcal{M}$ is an automatic graph, $Conf(\mathcal{M}) = (D, E)$.

We define the domain of our automatic structure to be $A = \Sigma^\star \cup D$. The binary relation of the automatic structure is defined to be the following:

$$R = E \cup \{(x, (\lambda\ \iota\ (x,y), \lambda, \lambda)) : x,y \in \Sigma^\star\} \cup \{(((x,y), \lambda, \lambda\ q_f\ \text{“yes”}\ ), y) : x, y \in \Sigma^\star\}.$$

Intuitively, the structure $(A;R)$ is a stretched out version of (C, L_α) with infinitely many finite pieces extending from elements of C and with disjoint pieces which are either finite chains or chains of type ω. The structure $(A;R)$ is easily seen to be automatic. We should verify, however, that R is well-founded. Let $Y \subset A$. If $Y \cap C \neq \emptyset$ then since (C, L_α) is well-founded, there is $x \in Y \cap C$ which is L_α-minimal. Then the only possible elements u in Y for which $(u,x) \in R$ are those which lie on computation chains connecting some $z \in C$ with x. Since each such computation chain is finite, there is an R-minimal u below x on each chain. Any such u is R-minimal for Y. On the other hand, if $Y \cap C = \emptyset$, then Y consists of disjoint finite chains and chains of type ω. Any such chain has a minimal element, and any of these elements are R-minimal for Y. Therefore, $(A;R)$ is an automatic well-founded structure.

We now consider the well-founded rank of $(A;R)$. For each element $x \in C$, an easy induction on $r_C(x)$, shows that

$$r_{\mathcal{C}}(x) \leq r_{\mathcal{A}}(x) \leq r_{\mathcal{C}}(x) + \omega.$$

We denote by $\ell(a,b)$ the (finite) length of the computation chain of $\mathcal{M}$ with input (a,b). For any element $a_{x,y}$ in the computation chain which represents the computation of $\mathcal{M}$ determining whether $(x,y) \in R$,

$$r_{\mathcal{A}}(x) \leq r_{\mathcal{A}}(a_{x,y}) \leq r_{\mathcal{A}}(x) + \ell(x,y).$$

For any element u in an unproductive chain of the configuration space, $0 \leq r_{\mathcal{A}}(u) \leq \omega$. Therefore, since $C \subset A$,

$$r(\mathcal{A}) = \sup\{r_{\mathcal{A}}(x) : x \in A\} \geq \sup\{r_{\mathcal{A}}(x) : x \in C\} \geq \sup\{r_{\mathcal{C}}(x) : x \in C\} = r(\mathcal{C}),$$

and

$$r(\mathcal{A}) = \sup\{r_{\mathcal{A}}(x) : x \in A\} \leq \sup\{r_{\mathcal{C}}(x) + \omega, \omega : x \in C\} \leq r(\mathcal{C}) + \omega.$$

□

6. Automatic Structures and Scott Rank

In this section, we consider automatic signatures of any signature. We use similar embedding techniques with configuration spaces to show that for each computable structure $\mathcal{C}$ there is an automatic structure $\mathcal{A}$ whose Scott rank is close to the Scott rank of $\mathcal{C}$. The Scott rank of a structure, introduced in the proof of Scott's Isomorphism Theorem [**?**], is a measure of complexity of the structure. There are several definitions of the Scott rank that are all essentially equivalent. Here we follow the definition from [**?**].

Definition 6.1. For structure $\mathcal{A}$ and tuples $\bar{a}, \bar{b} \in A^n$, define

- $\bar{a} \equiv^0 \bar{b}$ if $\bar{a}, \bar{b}$ satisfy the same quantifier free formulas in the language of $\mathcal{A}$;
- For $\alpha > 0$, $\bar{a} \equiv^\alpha \bar{b}$ if for all $\beta < \alpha$, for each $\bar{c}$ there is $\bar{d}$ such that $\bar{a}, \bar{c} \equiv^\beta \bar{b}, \bar{d}$; and for each $\bar{d}$ there is $\bar{c}$ such that $\bar{a}, \bar{c} \equiv^\beta \bar{b}, \bar{d}$.

Then, the **Scott rank** of the tuple $\bar{a}$, denoted by $\mathcal{SR}(\bar{a})$, is the least β such that for all $\bar{b} \in A^n$, $\bar{a} \equiv^\beta \bar{b}$ implies that $(\mathcal{A}, \bar{a}) \cong (\mathcal{A}, \bar{b})$. Finally, the Scott rank of $\mathcal{A}$, denoted by $\mathcal{SR}(\mathcal{A})$, is the least α greater than the Scott ranks of all tuples of $\mathcal{A}$.

Example 6.2. $\mathcal{SR}(\omega, \leq) = 2$, $\mathcal{SR}(n \cdot \omega, \leq) = n + 1$ and $\mathcal{SR}(\mathbb{Q}, \leq) = 1$.

Since the following construction uses configuration spaces of (reversible) Turing machines, we comment that these graphs are locally finite and do not have high Scott rank.

Proposition 6.3. *Let $G = (V, E)$ be a locally finite graph, then $SR(G) \leq 3$.*

The neighbourhood of diameter n of a subset U, denoted $B_n(U)$, is defined inductively as follows. First, $B_0(U) = U$. Then,

$$B_n(U) = \bigcup_{i=0,\ldots,n-1} B_i \cup \{v \in V : \exists a \in U, b_1, \ldots, b_{n-1}(aEb_1E \cdots Eb_{n-1}Ev \text{ or } vEb_1E \cdots Eb_{n-1}Ea)\}$$

Lemma 6.4. *Let $\bar{a}, \bar{b} \in V$ be such that $\bar{a} \equiv^2 \bar{b}$. Then for all n, $(B_n(\bar{a}), E, \bar{a}) \cong (B_n(\bar{b}), E, \bar{b})$ (in other words, there is bijection of the neighbourhoods which sends $\bar{a}$ to $\bar{b}$ and which respects E).*

Proof. For a given n, let $\bar{c} = B_n(\bar{a}) \setminus \bar{a}$. Note that $\bar{c}$ is a finite tuple because of the local finiteness condition. Since $\bar{a} \equiv^2 \bar{b}$, there is $\bar{d}$ such that $\bar{a}\bar{c} \equiv^1 \bar{b}\bar{d}$. If $B_n(\bar{b}) = \bar{b}\bar{d}$, we are done. Two set inclusions are needed. First, we show that $d_i \in B_n(\bar{b})$. By definition, we have that $c_i \in B_n(\bar{a})$, and let $a_j, u_1, \ldots, u_{n-1}$ witness this. Then since $\bar{a}\bar{c} \equiv^1 \bar{b}\bar{d}$, there are $v_1, \ldots, v_{n-1}$ such that $\bar{a}\bar{c}\bar{u} \equiv^0 \bar{b}\bar{d}\bar{v}$. In particular, we have that if $c_iEu_iE \cdots Eu_{n-1}Ea_j$, then also $d_iEv_iE \cdots Ev_{n-1}Eb_j$ (and likewise if the E relation is in the other direction). Hence, $d_i \in B_n(\bar{b})$. Conversely, suppose $v \in B_n(\bar{b}) \setminus \bar{d}$. Let $v_1, \ldots, v_n$ be witnesses and this will let us find a new element of $B_n(\bar{a})$ which is not in $\bar{c}$, a contradiction. □

Lemma 6.5. *Let $G = (V, E)$ be agraph. Suppose $\bar{a}, \bar{b} \in V$ are such that for all n, $(B_n(\bar{a}), E, \bar{a}) \cong (B_n(\bar{b}), E, \bar{b})$. Then there is an isomorphism between the component of G containing $\bar{a}$ and that containing $\bar{b}$ which sends $\bar{a}$ to $\bar{b}$.*

Proof. We consider a tree of partial isomorphisms of G. The nodes of the tree are bijections from $B_n(\bar{a})$ to $B_n(\bar{b})$ which respect the relation E and map $\bar{a}$ to $\bar{b}$. Node f is the child of node g in the tree if $\text{dom}(f) = B_n(\bar{a})$, $\text{dom}(g) = B_{n+1}(\bar{a})$ and $f \supset g$. Note that the root of this tree is the map which sends $\bar{a}$ to $\bar{b}$. Moreover, the tree is finitely branching and is infinite by Lemma 1. Therefore, König's Lemma gives an infinite path through this tree. The union of all partial isomorphisms along this path is the required isomorphism. □

Proof of Proposition 6.3. To prove the proposition, we note that for any $\bar{a}, \bar{b}$ in V such that $\bar{a} \equiv^2 \bar{b}$, Lemmas 1 and 2 yield an isomorphism from the component of $\bar{a}$ to the component of $\bar{b}$, and under this isomorphism $\bar{a}$ gets mapped to $\bar{b}$. Hence, if $\bar{a} \equiv^2 \bar{b}$, there is an automorphism of G that maps $\bar{a}$ to $\bar{b}$. Therefore, for each $\bar{a} \in V$, $SR(\bar{a}) \leq 2$, and $SR(G) \leq 3$. □

Let $\mathcal{C}$ be a computable structure. Our goal is to construct an automatic structure whose Scott rank is (close to) the Scott rank of $\mathcal{C}$. The construction in some sense expands $\mathcal{C}$ into an automatic structure. We comment that expansions do not necessarily preserve the Scott rank. For example, any computable structure, $\mathcal{C}$, has an expansion with Scott rank 2. The expansion is obtained by adding the successor relation into the signature.

For computable structure $\mathcal{C}$, we assume without loss of generality that $\mathcal{C}$ has finite signature $(R_1^{(m_1)}, \ldots, R_n^{(m_n)})$. Moreover, since the domain of $\mathcal{C}$ is computable, there is a one-to-one map of it onto ω. Therefore we assume that the domain is encoded as $\Sigma^\star$ for some finite alphabet Σ. The construction of the automatic structure involves connecting the configuration spaces of Turing machines computing each relation in the signature to elements in the domain. We detail the construction for each relation in the signature. Consider R_i, of arity m_i. Let $\mathcal{M}_i$ be a Turing machine for R_i. As before (see Lemma 4.2), we can assume that $\mathcal{M}_i$ is reversible. Moreover, a simple modification of the machine means that we can assume that $\mathcal{M}_i$ halts if and only if its output is "yes" (and otherwise enters an infinite loop). For the automatic structure, we again begin with the configuration space $Conf(\mathcal{M}_i)$. Recall that the set of valid initial configurations is regular. We now modify the configuration space so as to respect the isomorphism type of the structure $\mathcal{C}$. This will ensure that the construction (almost) preserves the Scott rank of the structure.

6.1. **Smoothing out unproductive parts.**

The unproductive parts of the configuration space represent features of the Turing machine computing R_i, as opposed to intrinsic properties of R_i itself. The length and number of unproductive chains is determined by the number of states of $\mathcal{M}_i$ and hence may differ even for Turing machines computing the same set. Since the unproductive part of the configuration space should play no role in determining the isomorphism type of the structure under consideration, we wish to eliminate the possible differences arising from it. To do so, we add a $\omega^\star$-chain (the successor relation on the negative integers) below each base of an unproductive chain. This procedure retains the automaticity of the structure since bases of unproductive chains can be recognized by a finite automaton and $\omega^\star$-chains are automatically presentable (e.g. with domain $1^\star$). Whereas the addition of these $\omega^\star$ tails allows us to distinguish between the unproductive and productive parts of the configuration space, we must still smooth out the structure to ensure that the unproductive section of the structure does not form a barrier to isomorphism. In other words, we wish to add enough redundant information in the unproductive section of the structure so that if two given computable structures are isomorphic, the unproductive parts of the automatic representations will also be isomorphic . Thus, to the structure we are building we add ω-many copies of $\omega^\star$ and ω-many copies of $\omega^\star + \omega$. This ensures that the (smoothed) unproductive section of the configuration space of any Turing machine will be isomorphic. It is important to note that adding this redundancy preserves automaticity since the operation is a disjoint union between automatic structures.

6.2. **Smoothing out lengths of computation chains.**

We now turn our attention to the part of the configuration space which corresponds to actual computations of the machine. These are chains (both finite and infinite) whose base is a valid initial configuration. The length of each finite chain denotes the length of computation required to return a "yes" answer. However, this detail of the computation is not relevant to the question of whether the input tuple represented in the initial configuration is in R_i. Therefore, we will smooth out these chains by adding "fans" to each base. More specifically, we connect to each base of a computation chain a structure which consists of infinitely many chains of each finite length. To guarantee that automaticity is preserved, we note that the set of bases of computations chains is recognized by a finite automaton. Also, we need to ensure that the structure we are adjoining to these bases is automatic. To do so we follow Rubin [?]: consider the structure whose domain is $0^\star 01^\star$ and whose transition relation is given by xEy if and only if $|x| = |y|$ and y is the least lexicographic successor of x in the domain. This structure has chains 0, 00 — 01, 000 — 001 — 011, 0000 — 0001 — 0011 — 0111, etc. Thus, it has a finite chain of every finite length. Moreover, it is an automatic structure. Next, as in Lemma 2.5, we take the ω-fold disjoint union of the structure and then identify the bases of all the finite chains. The resulting structure is a "fan" with infinitely many chains of each finite size.

At the end of this stage of this construction, we have that each valid initial computation state is the base for a fan. Moreover, the fan has an infinite component if and only if R_i does not hold of the input tuple corresponding to the base of the fan.

The result is a graph which extends the configuration space $Conf(\mathcal{M}_i)$ and is clearly an automatic graph. We call this graph $Smooth(R_i) = (D_i, E_i)$.

6.3. **Connecting domain symbols to the computations of the relation.** We apply the construction above to each of the relations R_i in the signature of the computable structure $\mathcal{C}$. Taking the union of the resulting automatic graphs and adding vertices for the domain, we have the structure $(\Sigma^\star \cup \cup_i D_i, E_1, \dots, E_n)$. We assume without loss of generality that each $\mathcal{M}_i$ has a different initial state, and denote it by ι_i. We add n partial functions F_1, …, F_n to the signature of the automatic structure we are building. These functions will connect the elements of the domain of the computable structure with the computations of the relations R_i. For each $1 \le i \le n$, the graph of F_i is defined as follows:

$$F_i = \{(x_0, \dots, x_{m_i-1}, (\lambda\ \iota_i\ (x_0, \dots, x_{m_i-1}), \lambda, \lambda)) \mid x_0, \dots, x_{m_i-1} \in \Sigma^\star\}$$

These functions are automatic because the set of initial configurations is recognisable by finite automaton. An important property of the construction is that for $\bar{x} \in \Sigma^\star$,

$$\begin{aligned} R_i(\bar{x}) \iff & F_i(\bar{x}, (\lambda\ \iota_i\ \bar{x}, \lambda, \lambda))\ \&\ \text{all } E_i \text{ chains} \\ & \text{emanating from } (\lambda\ \iota_i\ \bar{x}, \lambda, \lambda) \text{ are finite;} \end{aligned}$$

Thus, we have built the automatic structure $\mathcal{A} = (\Sigma^\star \cup \cup_i D_i, E_1, \dots, E_n, F_1, \dots, F_n)$. It remains to show that $\mathcal{SR}(\mathcal{A})$ is close to $\mathcal{SR}(\mathcal{C})$. To prove this, we will need the following two lemmas.

Lemma 6.6. *For $\bar{x}, \bar{y} \in \Sigma^\star$ (hence from the domain of $\mathcal{C}$), and for any ordinal α, $\bar{x} \equiv^\alpha_{\mathcal{C}} \bar{y}$ implies that $\bar{x} \equiv^\alpha_{\mathcal{A}} \bar{y}$.*

Proof. Let $X = \mathrm{dom}\mathcal{A} \setminus \Sigma^\star$. We prove the stronger result that for any ordinal α, and for all $\bar{x}, \bar{y} \in \Sigma^\star$ and $\bar{x}', \bar{y}' \in X$, if the following assumptions hold

(1) $\bar{x} \equiv^\alpha_{\mathcal{C}} \bar{y}$;
(2) $\langle \bar{x}', E_i : i = 1 \dots n\rangle_{\mathcal{A}} \cong_f \langle \bar{y}', E_i, : i = 1 \dots n\rangle_{\mathcal{A}}$ (hence the substructures in A are isomorphic) with $f(\bar{x}') = \bar{y}'$; and

(3) for each $x'_k \in \bar{x}'$, each $i = 1, \ldots, n$ and each subsequence of indices of length m_i,

$$x'_k = (\lambda\ \iota_i\ \bar{x}_j, \lambda, \lambda) \iff y'_k = (\lambda\ \iota_i\ \bar{y}_j, \lambda, \lambda)$$

then $\bar{x}\bar{x}' \equiv^{\alpha}_{\mathcal{A}} \bar{y}\bar{y}'$. The lemma follows if we take $\bar{x}' = \bar{y}' = \lambda$ (the empty string).

We show the stronger result by induction on α. If $\alpha = 0$, we need to show that for each $i, k, k', k_0, \ldots, k_{m_i-1}$, $E_i(x'_k, x'_{k'})$ if and only if $E_i(y'_k, y'_{k'})$, and that $F_i(x_{k_0}, \ldots, x_{k_{m_i-1}}, x'_{k'})$ if and only if $F_i(y_{k_0}, \ldots, y_{k_{m_i-1}}, y'_{k'})$. The first statement follows by assumption 2, since the isomorphism must preserve the E_i relations and maps $\bar{x}'$ to $\bar{y}'$. The second statement follows by assumption 3.

Assume now that $\alpha > 0$ and that the result holds for all $\beta < \alpha$. Let $\bar{x}, \bar{y} \in \Sigma^\star$ and $\bar{x}', \bar{y}' \in A$ be such that the assumptions of the lemma hold. We will show that $\bar{x}\bar{x}' \equiv^{\alpha}_{\mathcal{A}} \bar{y}\bar{y}'$. Let $\beta < \alpha$ and suppose $\bar{u} \in \Sigma^\star, \bar{u}' \in A$. By assumption 1, there is $\bar{v} \in \Sigma^\star$ such that $\bar{x}\bar{u} \equiv^{\beta}_{\mathcal{C}} \bar{y}\bar{v}$. By the construction (in particular, the smoothing steps), we can find a corresponding $\bar{v}' \in A$ such that assumptions 2, 3 hold. Applying the inductive hypothesis, we get that $\bar{x}\bar{u}\bar{x}'\bar{u}' \equiv^{\beta}_{\mathcal{A}} \bar{y}\bar{v}\bar{y}'\bar{v}'$. Analogously, given $\bar{v}, \bar{v}'$ we can find the necessary $\bar{u}, \bar{u}'$. Therefore, $\bar{x}\bar{x}' \equiv^{\alpha}_{\mathcal{A}} \bar{y}\bar{y}'$. □

In the next lemma, we use the notation X_P to mean the subset of $X = A \setminus \Sigma^\star$ which corresponds to elements on fans associated with productive chains of the configuration space. We write X_U to mean the subset of X which corresponds to the unproductive chains of the configuration space. Therefore, $A = \Sigma^\star \cup X_P \cup X_U$, a disjoint union.

Lemma 6.7. *For each $\bar{x} \in \Sigma^\star$, $\bar{x}' \in X_P$, $\bar{u} \in X_U$ there is $\bar{y} \in \Sigma^\star$ such that $\mathcal{SR}_{\mathcal{A}}(\bar{x}\bar{x}'\bar{u}) \leq 2 + \mathcal{SR}_{\mathcal{C}}(\bar{y})$.*

Proof. Given $\bar{x}, \bar{x}', \bar{u}$, let $\bar{y} \in \Sigma^\star$ be a minimal element satisfying that $\bar{x} \subset \bar{y}$ and that $\bar{x}' \subset \langle \bar{y}, E_i, F_i : i = 1 \ldots n\rangle_{\mathcal{A}}$. Then we will show that $\bar{y}$ is the desired witness. First, we observe that since the unproductive part of the structure is disconnected from the productive elements we can consider the two independently. Moreover, because the structure of the unproductive part is predetermined and simple, for $\bar{u}, \bar{v} \in X_U$, if $\bar{u} \equiv^{1}_{\mathcal{A}} \bar{v}$ then $(\mathcal{A}, \bar{u}) \cong (\mathcal{A}, \bar{v})$. It remains to consider the productive part of the structure.

Consider any $\bar{z} \in \Sigma^\star$, $\bar{z}' \in X_P$ satisfying $\bar{z}' \subset \langle \bar{z}, E_i, F_i : i = 1 \ldots n\rangle_{\mathcal{A}}$. We claim that $SR_{\mathcal{A}}(\bar{z}\bar{z}') \leq 2 + \mathcal{SR}_{\mathcal{C}}(\bar{z})$. It suffices to show that for all α, for all $\bar{w} \in \Sigma^\star, \bar{w}' \in X_P$,

$$\bar{z}\bar{z}' \ \equiv^{2+\alpha}_{\mathcal{A}} \ \bar{w}\bar{w}' \qquad \Longrightarrow \qquad \bar{z} \ \equiv^{\alpha}_{\mathcal{C}} \ \bar{w}.$$

This is sufficient for the following reason. If $\bar{z}\bar{z}' \equiv^{2+\mathcal{SR}_{\mathcal{C}}(\bar{z})}_{\mathcal{A}} \bar{w}\bar{w}'$ then $\bar{z} \equiv^{\mathcal{SR}_{\mathcal{C}}(\bar{z})}_{\mathcal{C}} \bar{w}$ and hence $(\mathcal{C}, \bar{z}) \cong (\mathcal{C}, \bar{w})$. From this automorphism, we can define an automorphism of $\mathcal{A}$ mapping $\bar{z}\bar{z}'$ to $\bar{w}\bar{w}'$ because $\bar{z}\bar{z}' \equiv^{2}_{\mathcal{A}} \bar{w}\bar{w}'$ and hence for each i, the relative positions of $\bar{z}'$ and $\bar{w}'$ in the fans above $\bar{z}$ and $\bar{w}$ are isomorphic. Therefore, $2 + \mathcal{SR}_{\mathcal{C}}(\bar{z}) \geq \mathcal{SR}_{\mathcal{A}}(\bar{z}\bar{z}')$.

So, we now show that for all α, for all $\bar{w} \in \Sigma^\star, \bar{w}' \in X_P$, $\bar{z}\bar{z}' \equiv^{2+\alpha}_{\mathcal{A}} \bar{w}\bar{w}'$ implies that $\bar{z} \equiv^{\alpha}_{\mathcal{C}} \bar{w}$. We proceed by induction on α. For $\alpha = 0$, suppose that $\bar{z}\bar{z}' \equiv^{2}_{\mathcal{A}} \bar{w}\bar{w}'$. This implies that for each i and for each subsequence of length m_i of the indices, the E_i-fan above $\bar{z}_j$ has an infinite chain if and only if the E_i-fan above $\bar{w}_j$ does. Therefore, $R_i(\bar{z}_j)$ if and only if $R_i(\bar{w}_j)$. Hence, $\bar{z} \equiv^{0}_{\mathcal{C}} \bar{w}$, as required. For the inductive step, we assume the result holds for all $\beta < \alpha$. Suppose that $\bar{z}\bar{z}' \equiv^{2+\alpha}_{\mathcal{A}} \bar{w}\bar{w}'$. Let $\beta < \alpha$ and $\bar{c} \in \Sigma^\star$. Then $2 + \beta < 2 + \alpha$ so by definition there is $\bar{d} \in \Sigma^\star, \bar{d}' \in X_P$ such that $\bar{z}\bar{z}'\bar{c} \equiv^{2+\beta}_{\mathcal{A}} \bar{w}\bar{w}'\bar{d}\bar{d}'$. However, since $2 + \beta > 1$,

$\bar{d}'$ must be empty (elements in $\Sigma^\star$ cannot be 1-equivalent to elements in X_P). Then by the induction hypothesis, $\bar{z}\bar{c} \equiv_{\mathcal{C}}^{\beta} \bar{w}\bar{d}$. The argument works symmetrically if we are given $\bar{d}$ and want to find $\bar{c}$. Thus, $\bar{z} \equiv_{\mathcal{C}}^{\alpha} \bar{w}$, as required. □

We can now prove the main theorem of this section.

Theorem 6.8. *Let $\mathcal{C}$ be a computable structure and construct the automatic structure $\mathcal{A}$ from it as above. Then $\mathcal{SR}(\mathcal{C}) \leq \mathcal{SR}(\mathcal{A}) \leq 2 + \mathcal{SR}(\mathcal{C})$.*

Proof. Let $\bar{x}$ be a tuple in the domain of $\mathcal{C}$. Then, by the definition of Scott rank, $\mathcal{SR}_A(\bar{x})$ is the least ordinal α such that for all $\bar{y} \in \mathrm{dom}(\mathcal{A})$, $\bar{x} \equiv_A^\alpha \bar{y}$ implies that $(\mathcal{A}, \bar{x}) \cong (\mathcal{A}, \bar{y})$; and similarly for $\mathcal{SR}_C(\bar{x})$. We first show that $\mathcal{SR}_A(\bar{x}) \geq \mathcal{SR}_C(\bar{x})$. Suppose $\mathcal{SR}_C(\bar{x}) = \beta$. We assume for a contradiction that $\mathcal{SR}_A(\bar{x}) = \gamma < \beta$. Consider an arbitrary $\bar{z} \in \Sigma^\star$ (the domain of $\mathcal{C}$) such that $\bar{x} \equiv_{\mathcal{C}}^{\gamma} \bar{z}$. By Lemma 6.6, $\bar{x} \equiv_{\mathcal{A}}^{\gamma} \bar{z}$. But, the definition of γ as the Scott rank of $\bar{x}$ in $\mathcal{A}$ implies that $(\mathcal{A}, \bar{x}) \cong (\mathcal{A}, \bar{z})$. Now, $\mathcal{C}$ is $L_{\omega_1,\omega}$ definable in $\mathcal{A}$ and therefore inherits the isomorphism. Hence, $(\mathcal{C}, \bar{x}) \cong (\mathcal{C}, \bar{z})$. But, this implies that $\mathcal{SR}_{\mathcal{C}}(\bar{x}) \leq \gamma < \beta = \mathcal{SR}_{\mathcal{C}}(\bar{x})$, a contradiction.

So far, we have that for each $\bar{x} \in \Sigma^\star$, $\mathcal{SR}_{\mathcal{A}}(\bar{x}) \geq \mathcal{SR}_{\mathcal{C}}(\bar{x})$. Hence, since $\mathrm{dom}(\mathcal{C}) \subset \mathrm{dom}(\mathcal{A})$,

$$\begin{aligned}\mathcal{SR}(\mathcal{A}) &= \sup\{\mathcal{SR}_{\mathcal{A}}(\bar{x}) + 1 : \bar{x} \in \mathrm{dom}(\mathcal{A})\}\\ &\geq \sup\{\mathcal{SR}_{\mathcal{A}}(\bar{x}) + 1 : \bar{x} \in \mathrm{dom}(\mathcal{C})\}\\ &\geq \sup\{\mathcal{SR}_{\mathcal{C}}(\bar{x}) + 1 : \bar{x} \in \mathrm{dom}(\mathcal{C})\} = \mathcal{SR}(C).\end{aligned}$$

In the other direction, we wish to show that $\mathcal{SR}(\mathcal{A}) \leq 2 + \mathcal{SR}(\mathcal{C})$. Suppose this is not the case. Then there is $\bar{x}\bar{x}'\bar{u} \in \mathcal{A}$ such that $\mathcal{SR}_{\mathcal{A}}(\bar{x}\bar{x}'\bar{u}) \geq 2 + \mathcal{SR}(\mathcal{C})$. By Lemma 6.7, there is $\bar{y} \in \Sigma^\star$ such that $2 + \mathcal{SR}_{\mathcal{C}}(\bar{y}) \geq 2 + \mathcal{SR}(\mathcal{C})$, a contradiction. □

Recent work in the theory of computable structures has focussed on finding computable structures of high Scott rank. Nadel [?] proved that any computable structure has Scott rank at most $\omega_1^{CK} + 1$. Early on, Harrison ([?]) showed that there is a computable ordering of type $\omega_1^{CK}(1 + \eta)$ (where η is the order type of the rational numbers). This ordering has Scott rank $\omega_1^{CK} + 1$, witnessed by any element outside the initial ω_1^{CK} set. However, it was not until much more recently that a computable structure of Scott rank ω_1^{CK} was produced (see Knight and Millar [?]). Theorem 6.8 allows us to transfer all of these results to automatic structures. Hence, we get the following corollary.

Corollary 6.9. *For any infinite computable ordinal α there is an automatic structure of Scott rank α.* □

7. Automatic Successor Trees and Cantor-Bendixson Rank

In this section, we show that there are automatic successor trees of high Cantor-Bendixson (CB) rank. This result is in contrast to one proved in [?] which states that automatic partial order trees all have finite CB rank.

Definition 7.1. A **partial order tree** is a pair $(T, \leq)$ such that $\leq$ is a partial order on T, there is a $\leq$-minimal element of T, and each subset $\{x \in T : x \leq y\}$ is finite and linearly ordered under $\leq$. A **successor tree** is a pair (T, S) such that $S \subset T \times T$ and if $\prec_S$ is the reflexive and transitive closure of S then $(T, \prec_S)$ is a partial order tree.

Let $(T,\leq)$ be automatic partial order tree. The successor relation S on T is defined by the formula $(x \leq y)$ & $\forall z(x \leq z \to y \leq z))$. Therefore (T,S) is an automatic successor tree. In this section, we will build examples of automatic successor trees (T,S) whose corresponding partial order trees $(T,\prec_S)$ have no automatic presentations. We will see later that the opposite is true for computable trees.

Definition 7.2. Given a (partial order or successor) tree T, its **derivative** $d(T)$ is the subtree of T whose domain is

$$\{x \in T : \ x \text{ lies on at least two infinite paths in } T\}.$$

By transfinite induction, we define $d^0(T)=T$, $d^{\alpha+1}(T)=d(d^{\alpha}(T))$, and for γ a limit ordinal, $d^{\gamma}(T)=\cap_{\beta<\gamma}d^{\beta}(T)$. The **CB rank** of the tree, denoted by $CB(T)$, is the least ordinal α such that $d^{\alpha}(T)=d^{\alpha+1}(T)$.

As mentioned above, the CB ranks of automatic partial order trees are finite (as proved in [**?**]). Below we provide simple examples of automatic successor trees whose ranks are small ordinals.

Example 7.3. There is an automatic partial order tree (hence a successor tree) whose CB rank is n for each $n \in \omega$.

Proof. The tree T_n is defined over the n letter alphabet $\{a_1,\ldots,a_n\}$ as follows. The domain of the tree is $a_1^{\star}\cdots a_n^{\star}$. The order $\leq_n$ is the prefix partial order. Therefore, the successor relation is given as follows:

$$S(a_1^{\ell_1}\cdots a_i^{\ell_i}) = \begin{cases} \{a_1^{\ell_1}\cdots a_i^{\ell_i+1}, a_1^{\ell_1}\cdots a_i^{\ell_i}a_{i+1}\} & \text{if } 1 \leq i < n \\ \{a_1^{\ell_0}\cdots a_i^{\ell_i+1}\} & \text{if } i=n \end{cases}$$

Note that if $n=0$ then the tree is empty, which is consistent with it having CB rank 0. It is obvious that T_n is an automatic partial order (hence successor) tree. The rank of T_n can be shown, by induction, to be equal to n. □

The following examples code the finite rank successor trees uniformly into one automaton in order to push the rank higher. The next example provides an automatic successor tree $T_{\omega+1}$ of rank $\omega+1$. We note that the CB ranks of all trees with at most countably many paths are successor ordinals. Thus, $T_{\omega+1}$ will have countably many paths. Later, we construct a tree of rank ω which must embed the perfect tree because its CB rank is a limit ordinal.

Example 7.4. There is an automatic successor tree $T_{\omega+1}$ whose CB rank is $\omega+1$.

Proof. Informally, this tree is a chain of trees of increasing finite CB ranks. Let $T_{\omega+1} = (\{0,1\}^{\star},S)$ with S defined as follows:

$$\begin{aligned} S(1^n) &= \{1^n0, 1^{n+1}\} && \text{for all } n \\ S(0u) &= 0u0 && \text{for all } u \in \{0,1\}^{\star} \\ S(1^n0u) &= \{1^n0u0, 1^{n-1}0u1\} && \text{for } n \geq 1 \text{ and } u \in \{0,1\}^{\star} \end{aligned}$$

Intuitively, the subtree of rank n is coded by the set X_n of nodes which contain exactly n 1s. By induction on the length of strings, we can show that range$(S)=\{0,1\}^{\star}$ and hence the domain of the tree is also $\{0,1\}^\star$. It is also not hard to show that the transitive closure of the relation S is a tree partial order. To see that $T_{\omega+1}$ is automatic we note that the domain of the tree is trivially automatic and that the successor relation is a finite union of relations, the convolution of each of which can be expressed as a regular expression.

Finally, we compute the rank of $T_{\omega+1}$. We note that in successive derivatives, each of the finite rank sub-trees X_n is reduced in rank by 1. Therefore

$$d^\omega(T) = 1^\star.$$

But, since each point in $1^\star$ is on exactly one infinite path, $d^{\omega+1}(T) = \emptyset$, and this is a fixed-point. Thus, $CB(T_{\omega+1}) = \omega + 1$, as required. □

We finish this section by providing an example of the tree T_ω of rank ω. The idea is to code the trees T_n provided above into the leftmost path of the full binary tree.

Example 7.5. There is an automatic successor tree T_ω whose CB rank is ω.

Proof. The tree is the full binary tree, where at each node on the leftmost branch we append trees of increasing finite CB rank. Thus, define $T_\omega = (\{0,1\}^\star \cup \{0,a\}^\star, S)$ where S is given as follows:

$$\begin{aligned}
S(u1v) &= \{u1v0, u1v1\} && \text{for all } u,v \in \{0,1\}^\star\\
S(0^n) &= \{0^{n+1}, 0^n1, 0^na\} && \text{for all } n\\
S(au) &= aua && \text{for all } u \in \{0,a\}^\star\\
S(0^nau) &= \{0^naua, 0^{n-1}au0\} && \text{for } n \geq 1 \text{ and } u \in \{0,a\}^\star
\end{aligned}$$

Proving that T_ω is an automatic successor tree is a routine check. So, we need only compute its rank. Each derivative leaves the right part of the tree (the full binary tree) fixed. However, the trees appended to the leftmost path of the tree are affected by taking derivatives. Successive derivatives decrease the rank of the protruding finite rank trees by 1. Therefore, $d^\omega(T_\omega) = \{0,1\}^\star$, a fixed point. Thus, $CB(T_\omega) = \omega$. □

To extend these examples to higher ordinals, we consider the **product** operation on trees defined as follows. Let (T_1, S_1) and (T_2, S_2) be successor trees. The product of these trees is the tree (T, S) with domain $T = T_1 \times T_2$ and successor relations given by:

$$S((x,y),(u,v)) \iff \begin{cases} y \text{ is the root of } T_2 \text{ and } (u = x, S_2(y,v)) \text{ or } (S_1(x,u), y = v)\\ y \text{ is not the root of } T_2 \text{ and } u = x,\ S_2(y,v). \end{cases}$$

The following is an easy proposition.

Proposition 7.6. *Assume that T_1 and T_2 are successor trees of CB ranks α and β, respectively, each having at most countably many paths. Then $T_1 \times T_2$ has CB rank $\alpha + \beta$. Moreover, if T_1 and T_2 are automatic successor trees then so is the product.* □

The examples and the proposition above yield tools for building automatic successor trees of CB ranks up to ω^2. However, it is not clear these methods can be applied to obtain automatic successor trees of higher CB ranks.

8. Automatic successor trees of high CB ranks

In this section we provide new methods of coding of computable successor trees into automatic successor trees. We note that every computable successor tree (T, S) is also a computable partial order tree. Indeed, in order to compute if $x \prec_S y$, we effectively find the distances of y and x from the root. If y is closer to the root or is at the same distance as x then $\neg(x \prec_S y)$; otherwise, we start computing the trees above all z at the same distance from the root as x is. Then y must appear in one of these trees. This computes if $x \prec_S y$. We point out that not every computable partial order tree is a computable successor tree.

In the proof of the main theorem of this section we use the following lemma whose proof is left to the reader.

Lemma 8.1. *For any computable ordinal α there exists a computable successor tree whose CB rank is α.* □

Theorem 8.2. *For any computable ordinal $\alpha < \omega_1^{CK}$ there is an automatic successor tree of CB rank α.*

Proof. Suppose we are given $\alpha < \omega_1^{CK}$. By the lemma above, take a computable tree R_α of CB rank α. We will embed this tree into an automatic successor tree by using the configuration space for the Turing machine computing the successor relation for R_α.

We assume that the domain of R_α is $\Sigma^\star$ for some finite alphabet Σ and $r \in \Sigma^\star$ is the root of the tree R_α. Let $\mathcal{M}$ be a deterministic Turing machine which computes the successor relation of R_α. On input (x, y), $\mathcal{M}$ halts and answers either "yes" or "no" . By Lemma 4.2, we can assume without loss of generality that $\mathcal{M}$ is reversible. Recall that for the reversible Turing machine $\mathcal{M}$ the final (halting) configurations of all computations contain both the input and the output.

Our construction begins with the configuration space $Conf(\mathcal{M})$ of $\mathcal{M}$. The idea of embedding R_α into the desired automatic successor tree is the following. Assume that y is a successor of x in R_α. In the automatic tree we wish to connect the nodes (representing) x and y by the computation of $\mathcal{M}$ on the pair (x, y).

We set the domain of the automatic successor tree to be $\Sigma^\star \cup Conf(\mathcal{M})$. The successor relation of the automatic tree contains the following edges:

$$E' = E \cup \{(x, (\lambda\ \iota\ (x, y), \lambda, \lambda)) : x, y \in \Sigma^\star\} \cup \{(((x, y), \lambda, \lambda\ q_f \text{ "yes" }), y) : x, y \in \Sigma^\star\},$$

where E is the edge relation in $Conf(\mathcal{M})$. Note that E' is still a regular binary relation. The automatic structure now consists of a stretched out version of the tree R_α, where between each two elements of the original tree we have a coding of their computation. In addition, extending from each $x \in \Sigma^\star$ we have infinitely many finite computation chains. Those chains which output "no" are not connected to any other part of the automatic structure. Finally, there is a disjoint part of the structure which corresponds to computation chains whose bases are not valid initial configurations. By the reversibility condition, the extraneous components of the configuration space are isomorphic either to a finite chain or to an ω-chain. Moreover, the set J of invalid initial configurations which are the base of such an unproductive chain is regular. Therefore, the relation

$$\tilde{E} = E' \cup \{(r, (u, v, w)) : (u, v, w) \in J\}$$

(where r is the root of R_α) is FA recognisable. We define $T_\alpha = (Conf \cup \Sigma^\star, \tilde{E})$. It is straightforward to verify that T_α is, indeed, an automatic successor tree.

We now consider the CB rank of T_α. Note that the first derivative removes all the subtrees whose roots are at distance 1 from the root and are invalid initial computations. This occurs because each of the invalid computation chains has no branching and is not connected to any other element of the tree. Next, if we consider the subtree of T_α rooted at an $x \in \Sigma^\star$, we see that all the paths which correspond to computations whose output is "no" vanish after the first derivative. Moreover, $x \in d(T_\alpha)$ if and only if $x \in d(R_\alpha)$ because the construction did not add any new infinite paths. Therefore, after one derivative, the structure is exactly a stretched out version of $d(R_\alpha)$. Likewise, for all $\beta < \alpha$, $d^\beta(T_\alpha)$ is a stretched out version of $d^\beta(R_\alpha)$. Hence, $CB(T_\alpha) = CB(R_\alpha) = \alpha$. □