Isomorphie von endlich präsentierten Strukturen

Markus Lohrey

Universität Siegen

Theorietag 2013

Finitely presented structures

Goal: Infinite structures that have a finite representation.

Finitely presented structures

Goal: Infinite structures that have a finite representation.

- computable structures
- (ω-)automatic structures, (ω-)tree automatic structures
- prefix recognizable graphs
- equational graphs
- pushdown graphs

Computable structures

Computable structure

A (relational) structure $\mathcal{A} = (A, R_1, \ldots, R_n)$ is computable, if

- $A \subseteq \mathbb{N}$ is a decidable set of naturals and
- every relation $R_i \subseteq A^{n_i} \subseteq \mathbb{N}^{n_i}$ is decidable too.

Computable structures

Computable structure

A (relational) structure $\mathcal{A} = (A, R_1, \ldots, R_n)$ is computable, if

- $A \subseteq \mathbb{N}$ is a decidable set of naturals and
- every relation $R_i \subseteq A^{n_i} \subseteq \mathbb{N}^{n_i}$ is decidable too.

Problem: Most interesting properties of computable structures are undecidable (e.g. has a computable graph at least one edge).

Automatic structures

Automatic structure (Büchi 1960, Khoussainov, Nerode 1995)

A structure $\mathcal{A} = (A, R_1, \ldots, R_n)$ is automatic, if there exists a finite alphabet Σ such that:

- $A \subseteq \Sigma^*$ is a regular language and
- every relation $R_i \subseteq A^{n_i} \subseteq (\Sigma^*)^{n_i}$ is synchronously regular.

Automatic structures

Automatic structure (Büchi 1960, Khoussainov, Nerode 1995)

A structure $\mathcal{A} = (A, R_1, \dots, R_n)$ is automatic, if there exists a finite alphabet Σ such that:

- $A \subseteq \Sigma^*$ is a regular language and
- every relation $R_i \subseteq A^{n_i} \subseteq (\Sigma^*)^{n_i}$ is synchronously regular.

A synchronously regular relation $R \subseteq (\Sigma^*)^n$ is a relation that can be accepted by a synchronous n-tape automaton.

Automatic structures

Automatic structure (Büchi 1960, Khoussainov, Nerode 1995)

A structure $\mathcal{A} = (A, R_1, \dots, R_n)$ is automatic, if there exists a finite alphabet Σ such that:

- $A \subseteq \Sigma^*$ is a regular language and
- every relation $R_i \subseteq A^{n_i} \subseteq (\Sigma^*)^{n_i}$ is synchronously regular.

A synchronously regular relation $R \subseteq (\Sigma^*)^n$ is a relation that can be accepted by a synchronous n-tape automaton.

A synchronous 2-tape automaton working on a pair $(u, v) \in \Sigma^* \times \Sigma^*$:

v	b_0	b_1	b_2	$\cdots$	b_{m-1}	b_m	$\#$	$\cdots$	$\#$
u	a_0	a_1	a_2	$\cdots$	a_{m-1}	a_m	a_{m+1}	$\cdots$	a_n

Automatic structures

Automatic structure (Büchi 1960, Khoussainov, Nerode 1995)

A structure $\mathcal{A} = (A, R_1, \ldots, R_n)$ is automatic, if there exists a finite alphabet Σ such that:

- $A \subseteq \Sigma^*$ is a regular language and
- every relation $R_i \subseteq A^{n_i} \subseteq (\Sigma^*)^{n_i}$ is synchronously regular.

A synchronously regular relation $R \subseteq (\Sigma^*)^n$ is a relation that can be accepted by a synchronous n-tape automaton.

A synchronous 2-tape automaton working on a pair $(u, v) \in \Sigma^* \times \Sigma^*$:

q_0 (initial state)

v	$\boxed{b_0}$	b_1	b_2	$\cdots$	b_{m-1}	b_m	$\#$	$\cdots$	$\#$
u	$\boxed{a_0}$	a_1	a_2	$\cdots$	a_{m-1}	a_m	a_{m+1}	$\cdots$	a_n

Automatic structures

Automatic structure (Büchi 1960, Khoussainov, Nerode 1995)

A structure $\mathcal{A} = (A, R_1, \ldots, R_n)$ is automatic, if there exists a finite alphabet Σ such that:

- $A \subseteq \Sigma^*$ is a regular language and
- every relation $R_i \subseteq A^{n_i} \subseteq (\Sigma^*)^{n_i}$ is synchronously regular.

A synchronously regular relation $R \subseteq (\Sigma^*)^n$ is a relation that can be accepted by a synchronous n-tape automaton.

A synchronous 2-tape automaton working on a pair $(u, v) \in \Sigma^* \times \Sigma^*$:

q_1

v	b_0	$\boxed{b_1}$	b_2	$\cdots$	b_{m-1}	b_m	$\#$	$\cdots$	$\#$
u	a_0	$\boxed{a_1}$	a_2	$\cdots$	a_{m-1}	a_m	a_{m+1}	$\cdots$	a_n

Automatic structures

Automatic structure (Büchi 1960, Khoussainov, Nerode 1995)

A structure $\mathcal{A} = (A, R_1, \ldots, R_n)$ is automatic, if there exists a finite alphabet Σ such that:

- $A \subseteq \Sigma^*$ is a regular language and
- every relation $R_i \subseteq A^{n_i} \subseteq (\Sigma^*)^{n_i}$ is synchronously regular.

A synchronously regular relation $R \subseteq (\Sigma^*)^n$ is a relation that can be accepted by a synchronous n-tape automaton.

A synchronous 2-tape automaton working on a pair $(u, v) \in \Sigma^* \times \Sigma^*$:

q_2

v	b_0	b_1	$\boxed{b_2}$	$\cdots$	b_{m-1}	b_m	$\#$	$\cdots$	$\#$
u	a_0	a_1	$\boxed{a_2}$	$\cdots$	a_{m-1}	a_m	a_{m+1}	$\cdots$	a_n

Automatic structures

Automatic structure (Büchi 1960, Khoussainov, Nerode 1995)

A structure $\mathcal{A} = (A, R_1, \dots, R_n)$ is automatic, if there exists a finite alphabet Σ such that:

- $A \subseteq \Sigma^*$ is a regular language and
- every relation $R_i \subseteq A^{n_i} \subseteq (\Sigma^*)^{n_i}$ is synchronously regular.

A synchronously regular relation $R \subseteq (\Sigma^*)^n$ is a relation that can be accepted by a synchronous n-tape automaton.

A synchronous 2-tape automaton working on a pair $(u, v) \in \Sigma^* \times \Sigma^*$:

q_{m-1}

v	b_0	b_1	b_2	$\cdots$	$\boxed{b_{m-1}}$	b_m	$\#$	$\cdots$	$\#$
u	a_0	a_1	a_2	$\cdots$	$\boxed{a_{m-1}}$	a_m	a_{m+1}	$\cdots$	a_n

Automatic structures

Automatic structure (Büchi 1960, Khoussainov, Nerode 1995)

A structure $\mathcal{A} = (A, R_1, \ldots, R_n)$ is automatic, if there exists a finite alphabet Σ such that:

- $A \subseteq \Sigma^*$ is a regular language and
- every relation $R_i \subseteq A^{n_i} \subseteq (\Sigma^*)^{n_i}$ is synchronously regular.

A synchronously regular relation $R \subseteq (\Sigma^*)^n$ is a relation that can be accepted by a synchronous n-tape automaton.

A synchronous 2-tape automaton working on a pair $(u, v) \in \Sigma^* \times \Sigma^*$:

q_m

v	b_0	b_1	b_2	$\cdots$	b_{m-1}	$\boxed{b_m}$	$\#$	$\cdots$	$\#$
u	a_0	a_1	a_2	$\cdots$	a_{m-1}	$\boxed{a_m}$	a_{m+1}	$\cdots$	a_n

Automatic structures

Automatic structure (Büchi 1960, Khoussainov, Nerode 1995)

A structure $\mathcal{A} = (A, R_1, \ldots, R_n)$ is automatic, if there exists a finite alphabet Σ such that:

- $A \subseteq \Sigma^*$ is a regular language and
- every relation $R_i \subseteq A^{n_i} \subseteq (\Sigma^*)^{n_i}$ is synchronously regular.

A synchronously regular relation $R \subseteq (\Sigma^*)^n$ is a relation that can be accepted by a synchronous n-tape automaton.

A synchronous 2-tape automaton working on a pair $(u, v) \in \Sigma^* \times \Sigma^*$:

q_{m+1}

v	b_0	b_1	b_2	$\cdots$	b_{m-1}	b_m	$\#$	$\cdots$	$\#$
u	a_0	a_1	a_2	$\cdots$	a_{m-1}	a_m	a_{m+1}	$\cdots$	a_n

Automatic structures

Automatic structure (Büchi 1960, Khoussainov, Nerode 1995)

A structure $\mathcal{A} = (A, R_1, \ldots, R_n)$ is automatic, if there exists a finite alphabet Σ such that:

- $A \subseteq \Sigma^*$ is a regular language and
- every relation $R_i \subseteq A^{n_i} \subseteq (\Sigma^*)^{n_i}$ is synchronously regular.

A synchronously regular relation $R \subseteq (\Sigma^*)^n$ is a relation that can be accepted by a synchronous n-tape automaton.

A synchronous 2-tape automaton working on a pair $(u, v) \in \Sigma^* \times \Sigma^*$:

q_n

v	b_0	b_1	b_2	$\cdots$	b_{m-1}	b_m	$\#$	$\cdots$	$\boxed{\#}$
u	a_0	a_1	a_2	$\cdots$	a_{m-1}	a_m	a_{m+1}	$\cdots$	$\boxed{a_n}$

Automatic structures

Automatic structure (Büchi 1960, Khoussainov, Nerode 1995)

A structure $\mathcal{A} = (A, R_1, \ldots, R_n)$ is automatic, if there exists a finite alphabet Σ such that:

- $A \subseteq \Sigma^*$ is a regular language and
- every relation $R_i \subseteq A^{n_i} \subseteq (\Sigma^*)^{n_i}$ is synchronously regular.

A synchronously regular relation $R \subseteq (\Sigma^*)^n$ is a relation that can be accepted by a synchronous n-tape automaton.

A synchronous 2-tape automaton working on a pair $(u, v) \in \Sigma^* \times \Sigma^*$:

q_{n+1}(final state?)

v	b_0	b_1	b_2	$\cdots$	b_{m-1}	b_m	$\#$	$\cdots$	$\#$
u	a_0	a_1	a_2	$\cdots$	a_{m-1}	a_m	a_{m+1}	$\cdots$	a_n

Automatic structures

Remarks:

Automatic structures

Remarks:

- An automatic structure $\mathcal{A} = (A, R_1, \ldots, R_n)$ can be represented by a tuple $(M, M_1, \ldots, M_n)$ of finite automata.

Automatic structures

Remarks:

- An automatic structure $\mathcal{A} = (A, R_1, \ldots, R_n)$ can be represented by a tuple $(M, M_1, \ldots, M_n)$ of finite automata.
- A structure, which is isomorphic to an automatic structure will be called automatic too.

Automatic structures

Remarks:

- An automatic structure $\mathcal{A} = (A, R_1, \ldots, R_n)$ can be represented by a tuple $(M, M_1, \ldots, M_n)$ of finite automata.
- A structure, which is isomorphic to an automatic structure will be called automatic too.
- Every automatic structure is computable but not vice versa.

Automatic structures

Remarks:

- An automatic structure $\mathcal{A} = (A, R_1, \ldots, R_n)$ can be represented by a tuple $(M, M_1, \ldots, M_n)$ of finite automata.
- A structure, which is isomorphic to an automatic structure will be called automatic too.
- Every automatic structure is computable but not vice versa.

Examples for automatic structures:

Automatic structures

Remarks:

- An automatic structure $\mathcal{A} = (A, R_1, \ldots, R_n)$ can be represented by a tuple $(M, M_1, \ldots, M_n)$ of finite automata.
- A structure, which is isomorphic to an automatic structure will be called automatic too.
- Every automatic structure is computable but not vice versa.

Examples for automatic structures:

- Every finite structure: clear

Automatic structures

Remarks:

- An automatic structure $\mathcal{A} = (A, R_1, \ldots, R_n)$ can be represented by a tuple $(M, M_1, \ldots, M_n)$ of finite automata.
- A structure, which is isomorphic to an automatic structure will be called automatic too.
- Every automatic structure is computable but not vice versa.

Examples for automatic structures:

- Every finite structure: clear
- $(\mathbb{Q}, \leq)$:

Automatic structures

Remarks:

- An automatic structure $\mathcal{A} = (A, R_1, \ldots, R_n)$ can be represented by a tuple $(M, M_1, \ldots, M_n)$ of finite automata.
- A structure, which is isomorphic to an automatic structure will be called automatic too.
- Every automatic structure is computable but not vice versa.

Examples for automatic structures:

- Every finite structure: clear
- $(\mathbb{Q}, \leq)$: $(\mathbb{Q}, \leq) \cong (\{0,1\}^* 1, \leq_{\mathsf{lex}})$

Automatic structures

Remarks:

- An automatic structure $\mathcal{A} = (A, R_1, \ldots, R_n)$ can be represented by a tuple $(M, M_1, \ldots, M_n)$ of finite automata.
- A structure, which is isomorphic to an automatic structure will be called automatic too.
- Every automatic structure is computable but not vice versa.

Examples for automatic structures:

- Every finite structure: clear
- $(\mathbb{Q}, \leq)$: $(\mathbb{Q}, \leq) \cong (\{0,1\}^*1, \leq_{\text{lex}})$
- $(\mathbb{N}, +)$, $(\mathbb{Z}, +)$

Automatic structures

Remarks:

- An automatic structure $\mathcal{A} = (A, R_1, \ldots, R_n)$ can be represented by a tuple $(M, M_1, \ldots, M_n)$ of finite automata.
- A structure, which is isomorphic to an automatic structure will be called automatic too.
- Every automatic structure is computable but not vice versa.

Examples for automatic structures:

- Every finite structure: clear
- $(\mathbb{Q}, \leq)$: $(\mathbb{Q}, \leq) \cong (\{0,1\}^*1, \leq_{\mathsf{lex}})$
- $(\mathbb{N}, +)$, $(\mathbb{Z}, +)$
- Configuration graphs of Turing machines

Automatic structures: Limitations

The following structures are not automatic:

Automatic structures: Limitations

The following structures are not automatic:

- A free monoid generated by at least two elements (Blumensath, Grädel 2000)

Automatic structures: Limitations

The following structures are not automatic:

- A free monoid generated by at least two elements (Blumensath, Grädel 2000)
- $(\mathbb{N}, \times)$ (Blumensath, Grädel 2000)

Automatic structures: Limitations

The following structures are not automatic:

- A free monoid generated by at least two elements (Blumensath, Grädel 2000)
- $(\mathbb{N}, \times)$ (Blumensath, Grädel 2000)
- $(\mathbb{Q}, +)$ (Tsankov 2009)

Automatic structures: Limitations

The following structures are not automatic:

- A free monoid generated by at least two elements (Blumensath, Grädel 2000)
- $(\mathbb{N}, \times)$ (Blumensath, Grädel 2000)
- $(\mathbb{Q}, +)$ (Tsankov 2009)

Characterizations:

Automatic structures: Limitations

The following structures are not automatic:

- A free monoid generated by at least two elements (Blumensath, Grädel 2000)
- $(\mathbb{N}, \times)$ (Blumensath, Grädel 2000)
- $(\mathbb{Q}, +)$ (Tsankov 2009)

Characterizations:

- A finitely generated group is automatic if and only if it is virtually Abelian (Oliver, Thomas 2005).

Automatic structures: Limitations

The following structures are not automatic:

- A free monoid generated by at least two elements (Blumensath, Grädel 2000)
- $(\mathbb{N}, \times)$ (Blumensath, Grädel 2000)
- $(\mathbb{Q}, +)$ (Tsankov 2009)

Characterizations:

- A finitely generated group is automatic if and only if it is virtually Abelian (Oliver, Thomas 2005).
- An ordinal α is automatic if and only if $\alpha < \omega^\omega$ ist (Delhommé 2001).

Automatic structures: Limitations

The following structures are not automatic:

- A free monoid generated by at least two elements (Blumensath, Grädel 2000)
- $(\mathbb{N}, \times)$ (Blumensath, Grädel 2000)
- $(\mathbb{Q}, +)$ (Tsankov 2009)

Characterizations:

- A finitely generated group is automatic if and only if it is virtually Abelian (Oliver, Thomas 2005).
- An ordinal α is automatic if and only if $\alpha < \omega^\omega$ ist (Delhommé 2001).
- A field is automatic if and only if it is finite (Khoussainov, Nies, Rubin, Stephan 2007).

Isomorphism

The isomorphism problem for a class $\mathcal{C}$ of finitely presented structures (e.g., computable structures, automatic structures):

INPUT: two structures $\mathcal{A}_1, \mathcal{A}_2 \in \mathcal{C}$
QUESTION: $\mathcal{A}_1 \simeq \mathcal{A}_2$ (are $\mathcal{A}_1$ and $\mathcal{A}_2$ isomorphic)?

Isomorphism

The isomorphism problem for a class $\mathcal{C}$ of finitely presented structures (e.g., computable structures, automatic structures):

INPUT: two structures $\mathcal{A}_1, \mathcal{A}_2 \in \mathcal{C}$
QUESTION: $\mathcal{A}_1 \simeq \mathcal{A}_2$ (are $\mathcal{A}_1$ and $\mathcal{A}_2$ isomorphic)?

Observation: The isomorphism problem for computable structures is undecidable:

Isomorphism

The isomorphism problem for a class $\mathcal{C}$ of finitely presented structures (e.g., computable structures, automatic structures):

INPUT: two structures $\mathcal{A}_1, \mathcal{A}_2 \in \mathcal{C}$
QUESTION: $\mathcal{A}_1 \simeq \mathcal{A}_2$ (are $\mathcal{A}_1$ and $\mathcal{A}_2$ isomorphic)?

Observation: The isomorphism problem for computable structures is undecidable: It is undecidable whether a computable graph (V, E) is isomorphic to $(\mathbb{N}, \emptyset)$.

Isomorphism

The isomorphism problem for a class $\mathcal{C}$ of finitely presented structures (e.g., computable structures, automatic structures):

INPUT: two structures $\mathcal{A}_1, \mathcal{A}_2 \in \mathcal{C}$
QUESTION: $\mathcal{A}_1 \simeq \mathcal{A}_2$ (are $\mathcal{A}_1$ and $\mathcal{A}_2$ isomorphic)?

Observation: The isomorphism problem for computable structures is undecidable: It is undecidable whether a computable graph (V, E) is isomorphic to $(\mathbb{N}, \emptyset)$.

To get more interesting results, we need a bit of recursion theory.

Arithmetical hierarchy

Arithmetical hierarchy (Kleene 1943)

A set $A \subseteq \mathbb{N}$ belongs to the class Σ^0_n (resp. Π^0_n) if there exists a quantifier-free arithmetical formula $\varphi(x, \overline{y}_1, \ldots, \overline{y}_n)$ (with $+$ and $\times$) such that

$$\begin{aligned} A &= \{x \in \mathbb{N} \mid \exists \overline{y}_1 \forall \overline{y}_2 \exists \overline{y}_3 \cdots \forall/\exists \overline{y}_n : \varphi(x, \overline{y}_1, \ldots, \overline{y}_n)\} \\ (\text{resp. } A &= \{x \in \mathbb{N} \mid \forall \overline{y}_1 \exists \overline{y}_2 \forall \overline{y}_3 \cdots \exists/\forall \overline{y}_n : \varphi(x, \overline{y}_1, \ldots, \overline{y}_n)\}) \end{aligned}$$

Arithmetical hierarchy

Arithmetical hierarchy (Kleene 1943)

A set $A \subseteq \mathbb{N}$ belongs to the class Σ_n^0 (resp. Π_n^0) if there exists a quantifier-free arithmetical formula $\varphi(x, \overline{y}_1, \ldots, \overline{y}_n)$ (with $+$ and $\times$) such that

$$\begin{aligned} A &= \{x \in \mathbb{N} \mid \exists \overline{y}_1 \forall \overline{y}_2 \exists \overline{y}_3 \cdots \forall/\exists \overline{y}_n : \varphi(x, \overline{y}_1, \ldots, \overline{y}_n)\} \\ (\text{resp. } A &= \{x \in \mathbb{N} \mid \forall \overline{y}_1 \exists \overline{y}_2 \forall \overline{y}_3 \cdots \exists/\forall \overline{y}_n : \varphi(x, \overline{y}_1, \ldots, \overline{y}_n)\}) \end{aligned}$$

Instead of a quantifier-free arithmetical formula (with $+$ and $\times$) one may take an arbitrary computable predicate $\varphi(x, \overline{y}_1, \ldots, \overline{y}_n)$.

Arithmetical hierarchy

Arithmetical hierarchy (Kleene 1943)

A set $A \subseteq \mathbb{N}$ belongs to the class Σ_n^0 (resp. Π_n^0) if there exists a quantifier-free arithmetical formula $\varphi(x, \overline{y}_1, \ldots, \overline{y}_n)$ (with $+$ and $\times$) such that

$$\begin{aligned} A &= \{x \in \mathbb{N} \mid \exists \overline{y}_1 \forall \overline{y}_2 \exists \overline{y}_3 \cdots \forall/\exists \overline{y}_n : \varphi(x, \overline{y}_1, \ldots, \overline{y}_n)\} \\ (\text{resp. } A &= \{x \in \mathbb{N} \mid \forall \overline{y}_1 \exists \overline{y}_2 \forall \overline{y}_3 \cdots \exists/\forall \overline{y}_n : \varphi(x, \overline{y}_1, \ldots, \overline{y}_n)\}) \end{aligned}$$

Instead of a quantifier-free arithmetical formula (with $+$ and $\times$) one may take an arbitrary computable predicate $\varphi(x, \overline{y}_1, \ldots, \overline{y}_n)$.

$$\begin{tikzcd}
 & \Pi_1^0 \arrow[dr, dash] & & \Pi_2^0 \arrow[dr, dash] & & \Pi_3^0 \arrow[dr, dash] & & \\
\mathrm{REC} = \Sigma_1^0 \cap \Pi_1^0 \arrow[ur, dash] \arrow[dr, dash] & & \Sigma_2^0 \cap \Pi_2^0 \arrow[ur, dash] \arrow[dr, dash] & & \Sigma_3^0 \cap \Pi_3^0 \arrow[ur, dash] \arrow[dr, dash] & & \Sigma_4^0 \cap \Pi_4^0 & \cdots \\
 & \Sigma_1^0 \arrow[ur, dash] & & \Sigma_2^0 \arrow[ur, dash] & & \Sigma_3^0 \arrow[ur, dash] & &
\end{tikzcd}$$

Analytical hierarchy

Analytical hierarchy (Kleene 1955)

A set $A \subseteq \mathbb{N}$ belongs to the class Σ^1_n (resp. Π^1_n) if there exists a first-order arithmetical formula $\varphi(x, X_1, \ldots, X_n)$ (with $+$ and $\times$ and unary predicates $X_1, \ldots, X_n$) such that

$$\begin{array}{rcl} A & = & \{x \in \mathbb{N} \mid \exists Y_1 \forall Y_2 \exists Y_3 \cdots \forall/\exists Y_n : \varphi(x, Y_1, \ldots, Y_n)\} \\ (\text{resp. } A & = & \{x \in \mathbb{N} \mid \forall Y_1 \exists Y_2 \forall Y_3 \cdots \exists/\forall Y_n : \varphi(x, Y_1, \ldots, Y_n)\}) \end{array}$$

Analytical hierarchy

Analytical hierarchy (Kleene 1955)

A set $A \subseteq \mathbb{N}$ belongs to the class Σ^1_n (resp. Π^1_n) if there exists a first-order arithmetical formula $\varphi(x, X_1, \ldots, X_n)$ (with $+$ and $\times$ and unary predicates $X_1, \ldots, X_n$) such that

$$\begin{array}{rcl} A & = & \{x \in \mathbb{N} \mid \exists Y_1 \forall Y_2 \exists Y_3 \cdots \forall/\exists Y_n : \varphi(x, Y_1, \ldots, Y_n)\} \\ (\text{resp. } A & = & \{x \in \mathbb{N} \mid \forall Y_1 \exists Y_2 \forall Y_3 \cdots \exists/\forall Y_n : \varphi(x, Y_1, \ldots, Y_n)\}) \end{array}$$

$$\begin{tikzcd}
 & & \Pi^1_1 \arrow[dr, dash] & & \Pi^0_2 \arrow[dr, dash] & & \Pi^1_3 \arrow[dr, dash] & & \\
\mathrm{AH} \arrow[r, dash] & \Sigma^1_1 \cap \Pi^1_1 \arrow[ur, dash] \arrow[dr, dash] & & \Sigma^1_2 \cap \Pi^1_2 \arrow[ur, dash] \arrow[dr, dash] & & \Sigma^1_3 \cap \Pi^1_3 \arrow[ur, dash] \arrow[dr, dash] & & \Sigma^1_4 \cap \Pi^1_4 & \cdots \\
 & & \Sigma^1_1 \arrow[ur, dash] & & \Sigma^0_2 \arrow[ur, dash] & & \Sigma^1_3 \arrow[ur, dash] & &
\end{tikzcd}$$

Isomorphism problem for computable structures

Theorem 1 (probably Kleene)

The isomorphism problem for computable structures is Σ_1^1-complete.

Isomorphism problem for computable structures

Theorem 1 (probably Kleene)

The isomorphism problem for computable structures is Σ_1^1-complete.

Intuitive meaning: In order to express that $\mathcal{A}_1 \cong \mathcal{A}_2$ for computable structures $\mathcal{A}_1$, $\mathcal{A}_2$, there is no better way than saying:
"There exists a mapping $f : \mathcal{A}_1 \to \mathcal{A}_2$, which is an isomorphism."

Isomorphism problem for computable structures

Theorem 1 (probably Kleene)

The isomorphism problem for computable structures is Σ_1^1-complete.

Intuitive meaning: In order to express that $\mathcal{A}_1 \cong \mathcal{A}_2$ for computable structures $\mathcal{A}_1$, $\mathcal{A}_2$, there is no better way than saying:
"There exists a mapping $f : \mathcal{A}_1 \to \mathcal{A}_2$, which is an isomorphism."

Theorem 2 (Goncharov, Knight 2002)

The isomorphism problem is Σ_1^1-complete for:

- *computable linear orders,*

Isomorphism problem for computable structures

Theorem 1 (probably Kleene)

The isomorphism problem for computable structures is Σ^1_1-complete.

Intuitive meaning: In order to express that $\mathcal{A}_1 \cong \mathcal{A}_2$ for computable structures $\mathcal{A}_1$, $\mathcal{A}_2$, there is no better way than saying:
"There exists a mapping $f : \mathcal{A}_1 \to \mathcal{A}_2$, which is an isomorphism."

Theorem 2 (Goncharov, Knight 2002)

The isomorphism problem is Σ^1_1-complete for:

- *computable linear orders,*
- *computable order trees (trees viewed as partial orders),*

Isomorphism problem for computable structures

Theorem 1 (probably Kleene)

The isomorphism problem for computable structures is Σ^1_1-complete.

Intuitive meaning: In order to express that $\mathcal{A}_1 \cong \mathcal{A}_2$ for computable structures $\mathcal{A}_1$, $\mathcal{A}_2$, there is no better way than saying:
"There exists a mapping $f : \mathcal{A}_1 \to \mathcal{A}_2$, which is an isomorphism."

Theorem 2 (Goncharov, Knight 2002)

The isomorphism problem is Σ^1_1-complete for:

- *computable linear orders,*
- *computable order trees (trees viewed as partial orders),*
- *computable Abelian groups, ...*

Isomorphism problem for computable structures

Theorem 1 (probably Kleene)

The isomorphism problem for computable structures is Σ_1^1-complete.

Intuitive meaning: In order to express that $\mathcal{A}_1 \cong \mathcal{A}_2$ for computable structures $\mathcal{A}_1$, $\mathcal{A}_2$, there is no better way than saying:
"There exists a mapping $f : \mathcal{A}_1 \to \mathcal{A}_2$, which is an isomorphism."

Theorem 2 (Goncharov, Knight 2002)

The isomorphism problem is Σ_1^1-complete for:

- *computable linear orders,*
- *computable order trees (trees viewed as partial orders),*
- *computable Abelian groups, ...*

Isomorphism problem for automatic structures

Theorem 3 (Khoussainov, Nies, Rubin, Stephan 2005)

The isomorphism problem is decidable for

- *automatic ordinals and*
- *automatic Boolean algebras.*

Isomorphism problem for automatic structures

Theorem 3 (Khoussainov, Nies, Rubin, Stephan 2005)

The isomorphism problem is decidable for

- *automatic ordinals and*
- *automatic Boolean algebras.*

Theorem 4 (Rubin 2004)

The isomorphism problem for automatic graphs of bounded degree is Π^0_3-complete.

Isomorphism problem for automatic structures

Theorem 3 (Khoussainov, Nies, Rubin, Stephan 2005)

The isomorphism problem is decidable for

- *automatic ordinals and*
- *automatic Boolean algebras.*

Theorem 4 (Rubin 2004)

The isomorphism problem for automatic graphs of bounded degree is Π^0_3-complete.

Theorem 5 (Khoussainov, Nies, Rubin, Stephan 2007)

The isomorphism problem is Σ^1_1-complete for automatic structures (even automatic successor trees).

Isomorphism problem for automatic structures

Theorem 3 (Khoussainov, Nies, Rubin, Stephan 2005)

The isomorphism problem is decidable for

- *automatic ordinals and*
- *automatic Boolean algebras.*

Theorem 4 (Rubin 2004)

The isomorphism problem for automatic graphs of bounded degree is Π^0_3-complete.

Theorem 5 (Khoussainov, Nies, Rubin, Stephan 2007)

The isomorphism problem is Σ^1_1-complete for automatic structures (even automatic successor trees).

Basic ideas: Configuration graphs of Turing machines

Isomorphism problems for equivalence structures

Techniques based on configuration graphs of Turing machines do not work for transitive relations and in particular for equivalence structures ($=$ set $+$ equivalence relation).

Isomorphism problems for equivalence structures

Techniques based on configuration graphs of Turing machines do not work for transitive relations and in particular for equivalence structures ($=$ set $+$ equivalence relation).

Theorem 6 (Kuske, Liu, L 2010)

The isomorphism problem for automatic equivalence structures is Π^0_1-complete.

Isomorphism problems for equivalence structures

Techniques based on configuration graphs of Turing machines do not work for transitive relations and in particular for equivalence structures ($=$ set $+$ equivalence relation).

Theorem 6 (Kuske, Liu, L 2010)

The isomorphism problem for automatic equivalence structures is Π^0_1-complete.

Theorem 7 (Calvert, Cenzer, Harizanov, Morozov 2005)

The isomorphism problem for computable equivalence structures is Π^0_4-complete.

Automatic equivalence structures

Upper bound

The isomorphism problem for automatic equivalence structures is in Π_1^0.

Automatic equivalence structures

Upper bound

The isomorphism problem for automatic equivalence structures is in Π^0_1.

Let $\mathcal{E}_1, \mathcal{E}_2$ be automatic equivalence structures.

Automatic equivalence structures

Upper bound

The isomorphism problem for automatic equivalence structures is in Π_1^0.

Let $\mathcal{E}_1, \mathcal{E}_2$ be automatic equivalence structures.

Then, $\mathcal{E}_1 \cong \mathcal{E}_2$ if and only if for all $n \in \mathbb{N} \cup \{\infty\}$:

$$\#[\mathcal{E}_1\text{-equiv. classes of size } n] \;=\; \#[\mathcal{E}_2\text{-equiv. classes of size } n]$$

Automatic equivalence structures

Upper bound

The isomorphism problem for automatic equivalence structures is in Π_1^0.

Let $\mathcal{E}_1, \mathcal{E}_2$ be automatic equivalence structures.

Then, $\mathcal{E}_1 \cong \mathcal{E}_2$ if and only if for all $n \in \mathbb{N} \cup \{\infty\}$:

$$\#[\mathcal{E}_1\text{-equiv. classes of size } n] \; = \; \#[\mathcal{E}_2\text{-equiv. classes of size } n]$$

For a fixed $n \in \mathbb{N} \cup \{\infty\}$, the number of $\mathcal{E}_i$-equivalence classes of size n can be computed effectively.

Automatic equivalence structures

Lower bound

The isomorphism problem for automatic equivalence structures is Π_1^0-hard.

Automatic equivalence structures

Lower bound

The isomorphism problem for automatic equivalence structures is Π_1^0-hard.

Proof is based on undecidability of Hilbert's 10th problem.

Automatic equivalence structures

Lower bound

The isomorphism problem for automatic equivalence structures is Π^0_1-hard.

Proof is based on undecidability of Hilbert's 10th problem.

Theorem 8 (Matiyasevich 1971)

For every Σ^0_1-set $X \subseteq \mathbb{N}$ there is a polynomial $p \in \mathbb{Z}[x_0, \ldots, x_k]$ (which can be computed effectively from an index for X) with:

$$X = \{n \in \mathbb{N} \mid \exists a_1, \ldots, a_k \in \mathbb{N} : p(n, a_1, \ldots, a_k) = 0\}.$$

Automatic equivalence structures

Lower bound

The isomorphism problem for automatic equivalence structures is Π_1^0-hard.

Proof is based on undecidability of Hilbert's 10th problem.

Theorem 8 (Matiyasevich 1971)

For every Σ_1^0-set $X \subseteq \mathbb{N}$ there is a polynomial $p \in \mathbb{Z}[x_0, \ldots, x_k]$ (which can be computed effectively from an index for X) with:

$$X = \{n \in \mathbb{N} \mid \exists a_1, \ldots, a_k \in \mathbb{N} : p(n, a_1, \ldots, a_k) = 0\}.$$

Corollary

The following problem is Π_1^0-complete:

INPUT: polynomials $p_1(x_1, \ldots, x_k), p_2(x_1, \ldots, x_k) \in \mathbb{N}[x_1, \ldots, x_k]$

QUESTION: $\forall x_1, \ldots, x_k \in \mathbb{N} : p_1(x_1, \ldots, x_k) \neq p_2(x_1, \ldots, x_k)$?

Coding polynomials by automata

Consider a polynomial $p(x_1, \ldots, x_k) \in \mathbb{N}[x_1, \ldots, x_k]$.

Coding polynomials by automata

Consider a polynomial $p(x_1, \ldots, x_k) \in \mathbb{N}[x_1, \ldots, x_k]$.

Step 1: From $p(x_1, \ldots, x_k)$ we can construct inductively a nondeterministic finite automaton (NFA) $\mathcal{A}$ over the alphabet $\{a_1, \ldots, a_k\}$ such that $L(\mathcal{A}) = a_1^* a_2^* \cdots a_k^*$ and

$$p(x_1, \ldots, x_k) \;=\; \#[\text{accepting runs of } \mathcal{A} \text{ on input } a_1^{x_1} \cdots a_k^{x_k}].$$

Coding polynomials by automata

Consider a polynomial $p(x_1, \ldots, x_k) \in \mathbb{N}[x_1, \ldots, x_k]$.

Step 1: From $p(x_1, \ldots, x_k)$ we can construct inductively a nondeterministic finite automaton (NFA) $\mathcal{A}$ over the alphabet $\{a_1, \ldots, a_k\}$ such that $L(\mathcal{A}) = a_1^* a_2^* \cdots a_k^*$ and

$$p(x_1, \ldots, x_k) \;=\; \#[\text{accepting runs of } \mathcal{A} \text{ on input } a_1^{x_1} \cdots a_k^{x_k}].$$

Let Q be the set of states of $\mathcal{A}$ and let $\Delta \subseteq Q \times \{a_1, \ldots, a_k\} \times Q$ be the set of transition triples of $\mathcal{A}$.

Coding polynomials by automata

Consider a polynomial $p(x_1, \ldots, x_k) \in \mathbb{N}[x_1, \ldots, x_k]$.

Step 1: From $p(x_1, \ldots, x_k)$ we can construct inductively a nondeterministic finite automaton (NFA) $\mathcal{A}$ over the alphabet $\{a_1, \ldots, a_k\}$ such that $L(\mathcal{A}) = a_1^* a_2^* \cdots a_k^*$ and

$$p(x_1, \ldots, x_k) \;=\; \#[\text{accepting runs of } \mathcal{A} \text{ on input } a_1^{x_1} \cdots a_k^{x_k}].$$

Let Q be the set of states of $\mathcal{A}$ and let $\Delta \subseteq Q \times \{a_1, \ldots, a_k\} \times Q$ be the set of transition triples of $\mathcal{A}$.

Step 2: We define the run automaton $\mathrm{Run}(\mathcal{A})$ as the NFA (over the alphabet Δ) that results from $\mathcal{A}$ by replacing in $\mathcal{A}$ every transition

$$q \xrightarrow{a} q' \quad \text{by the transition} \quad q \xrightarrow{(q,a,q')} q'.$$

Coding polynomials by automatic equivalence relations

Let $\pi : \Delta \to \{a_1, \ldots, a_k\}$ be the projection with $\pi(q, a, q') = a$.

Coding polynomials by automatic equivalence relations

Let $\pi : \Delta \to \{a_1, \ldots, a_k\}$ be the projection with $\pi(q, a, q') = a$.

Step 3: Let $\mathcal{E}(p)$ be the following automatic equivalence relation:

Coding polynomials by automatic equivalence relations

Let $\pi : \Delta \to \{a_1, \ldots, a_k\}$ be the projection with $\pi(q, a, q') = a$.

Step 3: Let $\mathcal{E}(p)$ be the following automatic equivalence relation:

- Domain $D = L(\mathrm{Run}(\mathcal{A}))$

Coding polynomials by automatic equivalence relations

Let $\pi : \Delta \to \{a_1, \ldots, a_k\}$ be the projection with $\pi(q, a, q') = a$.

Step 3: Let $\mathcal{E}(p)$ be the following automatic equivalence relation:

- Domain $D = L(\mathrm{Run}(\mathcal{A}))$
- Equivalence relation: $\{(u, v) \in D \times D \mid \pi(u) = \pi(v)\}$

Coding polynomials by automatic equivalence relations

Let $\pi : \Delta \to \{a_1, \ldots, a_k\}$ be the projection with $\pi(q, a, q') = a$.

Step 3: Let $\mathcal{E}(p)$ be the following automatic equivalence relation:

- Domain $D = L(\mathrm{Run}(\mathcal{A}))$
- Equivalence relation: $\{(u, v) \in D \times D \mid \pi(u) = \pi(v)\}$

Then we have for all $n \in \mathbb{N}$:

$$\mathcal{E}(p) \text{ has an equivalence class of size } n$$

$$\iff$$

$$\exists w \in a_1^* a_2^* \cdots a_k^* : n = \#[\text{accepting runs of } \mathcal{A} \text{ on input } w]$$

$$\iff$$

$$n \in \mathrm{Img}(p)$$

The reduction

Step 4: Let $p_1(x_1, \ldots, x_k)$, $p_2(x_1, \ldots, x_k)$ be two polynomials over $\mathbb{N}$.

The reduction

Step 4: Let $p_1(x_1, \ldots, x_k)$, $p_2(x_1, \ldots, x_k)$ be two polynomials over $\mathbb{N}$.

We construct two automatic equivalence structures $\mathcal{E}_1$ and $\mathcal{E}_2$ such that $\mathcal{E}_1 \cong \mathcal{E}_2$ if and only if

$$\forall x_1, \ldots, x_k \in \mathbb{N} : p_1(x_1, \ldots, x_k) \neq p_2(x_1, \ldots, x_k).$$

The reduction

Step 4: Let $p_1(x_1, \ldots, x_k)$, $p_2(x_1, \ldots, x_k)$ be two polynomials over $\mathbb{N}$.

We construct two automatic equivalence structures $\mathcal{E}_1$ and $\mathcal{E}_2$ such that $\mathcal{E}_1 \cong \mathcal{E}_2$ if and only if

$$\forall x_1, \ldots, x_k \in \mathbb{N} : p_1(x_1, \ldots, x_k) \neq p_2(x_1, \ldots, x_k).$$

Let $C(x, y) = (x + y)^2 + 3x + y$ (injective from $\mathbb{N} \times \mathbb{N}$ to $\mathbb{N}$) and

$$\begin{aligned} S_1(x_1, \ldots, x_k) &= C(p_1(x_1, \ldots, x_k), p_2(x_1, \ldots, x_k)) \\ S_2(x_1, x_2) &= C(x_1 + x_2 + 1, x_1) \\ S_3(x_1, x_2) &= C(x_1, x_1 + x_2 + 1). \end{aligned}$$

The reduction

For two equivalence structures $\mathcal{E}_1$ and $\mathcal{E}_2$ let us write $\mathcal{E}_1 \sim \mathcal{E}_2$ if:

$$\forall n \in \mathbb{N} \cup \{\infty\} : \mathcal{E}_1 \text{ has equivalence class of size } n \;\Leftrightarrow\; \mathcal{E}_2 \text{ has equivalence class of size } n$$

The reduction

For two equivalence structures $\mathcal{E}_1$ and $\mathcal{E}_2$ let us write $\mathcal{E}_1 \sim \mathcal{E}_2$ if:

$$\forall n \in \mathbb{N} \cup \{\infty\} : \mathcal{E}_1 \text{ has equivalence class of size } n \iff \mathcal{E}_2 \text{ has equivalence class of size } n$$

Then (using "$n \in \mathrm{Img}(p) \iff \mathcal{E}(p)$ has equiv. class of size n"), we have:

$$\forall x_1, \ldots, x_k \in \mathbb{N} : p_1(x_1, \ldots, x_k) \neq p_2(x_1, \ldots, x_k)$$

The reduction

For two equivalence structures $\mathcal{E}_1$ and $\mathcal{E}_2$ let us write $\mathcal{E}_1 \sim \mathcal{E}_2$ if:

$$\begin{aligned} \forall n \in \mathbb{N} \cup \{\infty\} : \mathcal{E}_1 &\text{ has equivalence class of size } n \;\Leftrightarrow \\ \mathcal{E}_2 &\text{ has equivalence class of size } n \end{aligned}$$

Then (using "$n \in \mathrm{Img}(p) \;\Leftrightarrow\; \mathcal{E}(p)$ has equiv. class of size n"), we have:

$$\forall x_1, \ldots, x_k \in \mathbb{N} : p_1(x_1, \ldots, x_k) \neq p_2(x_1, \ldots, x_k)$$

$$\Longleftrightarrow$$

$$\mathrm{Img}(S_1) \cup \mathrm{Img}(S_2) \cup \mathrm{Img}(S_3) = \mathrm{Img}(S_2) \cup \mathrm{Img}(S_3)$$

The reduction

For two equivalence structures $\mathcal{E}_1$ and $\mathcal{E}_2$ let us write $\mathcal{E}_1 \sim \mathcal{E}_2$ if:

$$\begin{aligned} \forall n \in \mathbb{N} \cup \{\infty\} : \mathcal{E}_1 & \text{ has equivalence class of size } n \;\Leftrightarrow \\ \mathcal{E}_2 & \text{ has equivalence class of size } n \end{aligned}$$

Then (using "$n \in \mathrm{Img}(p) \;\Leftrightarrow\; \mathcal{E}(p)$ has equiv. class of size n"), we have:

$$\forall x_1, \ldots, x_k \in \mathbb{N} : p_1(x_1, \ldots, x_k) \neq p_2(x_1, \ldots, x_k)$$

$$\Longleftrightarrow$$

$$\mathrm{Img}(S_1) \cup \mathrm{Img}(S_2) \cup \mathrm{Img}(S_3) = \mathrm{Img}(S_2) \cup \mathrm{Img}(S_3)$$

$$\Longleftrightarrow$$

$$\Big(\mathcal{E}(S_1) \uplus \mathcal{E}(S_2) \uplus \mathcal{E}(S_3)\Big) \sim \Big(\mathcal{E}(S_2) \uplus \mathcal{E}(S_3)\Big)$$

Final step

For an equivalence structure $\mathcal{E} = (A, \equiv)$ let $\aleph_0 \cdot \mathcal{E}$ be the equivalence structure

$$\mathcal{E} = (\mathbb{N} \times A, \equiv'),$$

where

$$(m, a) \equiv' (n, b) \quad \Longleftrightarrow \quad (m = n \text{ and } a \equiv b).$$

Final step

For an equivalence structure $\mathcal{E} = (A, \equiv)$ let $\aleph_0 \cdot \mathcal{E}$ be the equivalence structure

$$\mathcal{E} = (\mathbb{N} \times A, \equiv'),$$

where

$$(m, a) \equiv' (n, b) \quad \iff \quad (m = n \text{ and } a \equiv b).$$

The following two observations conclude the proof:

Final step

For an equivalence structure $\mathcal{E} = (A, \equiv)$ let $\aleph_0 \cdot \mathcal{E}$ be the equivalence structure

$$\mathcal{E} = (\mathbb{N} \times A, \equiv'),$$

where

$$(m, a) \equiv' (n, b) \quad \Longleftrightarrow \quad (m = n \text{ and } a \equiv b).$$

The following two observations conclude the proof:

- If $\mathcal{E}$ is automatic, then $\aleph_0 \cdot \mathcal{E}$ is automatic too.

Final step

For an equivalence structure $\mathcal{E} = (A, \equiv)$ let $\aleph_0 \cdot \mathcal{E}$ be the equivalence structure

$$\mathcal{E} = (\mathbb{N} \times A, \equiv'),$$

where

$$(m, a) \equiv' (n, b) \quad \iff \quad (m = n \text{ and } a \equiv b).$$

The following two observations conclude the proof:

- If $\mathcal{E}$ is automatic, then $\aleph_0 \cdot \mathcal{E}$ is automatic too.
- For two countable equivalence structures $\mathcal{E}_1$ and $\mathcal{E}_2$ we have:

$$\mathcal{E}_1 \sim \mathcal{E}_2 \quad \iff \quad \aleph_0 \cdot \mathcal{E}_1 \cong \aleph_0 \cdot \mathcal{E}_2.$$

Isomorphism problem for automatic structures

The technique of coding polynomials by automata can be used to get further results:

Isomorphism problem for automatic structures

The technique of coding polynomials by automata can be used to get further results:

Theorem 9 (Kuske, Liu, L 2010)

- *For all $n \geq 2$, the isomorphism problem for the class of automatic trees of height at most n is Π^0_{2n-3}-complete.*

Isomorphism problem for automatic structures

The technique of coding polynomials by automata can be used to get further results:

Theorem 9 (Kuske, Liu, L 2010)

- *For all $n \geq 2$, the isomorphism problem for the class of automatic trees of height at most n is Π^0_{2n-3}-complete.*
- *The isomorphism problems for (i) automatic order trees of finite height and (ii) automatic order trees with only countably many infinite paths are equivalent to true arithmetic $\mathrm{FOTh}(\mathbb{N}, +, \times)$.*

Isomorphism problem for automatic structures

The technique of coding polynomials by automata can be used to get further results:

Theorem 9 (Kuske, Liu, L 2010)

- *For all $n \geq 2$, the isomorphism problem for the class of automatic trees of height at most n is Π^0_{2n-3}-complete.*
- *The isomorphism problems for (i) automatic order trees of finite height and (ii) automatic order trees with only countably many infinite paths are equivalent to true arithmetic $\mathsf{FOTh}(\mathbb{N}, +, \times)$.*
- *The isomorphism problem for automatic order trees is Σ^1_1-complete.*

Isomorphism problem for automatic structures

The technique of coding polynomials by automata can be used to get further results:

Theorem 9 (Kuske, Liu, L 2010)

- *For all $n \geq 2$, the isomorphism problem for the class of automatic trees of height at most n is Π^0_{2n-3}-complete.*
- *The isomorphism problems for (i) automatic order trees of finite height and (ii) automatic order trees with only countably many infinite paths are equivalent to true arithmetic $\mathsf{FOTh}(\mathbb{N}, +, \times)$.*
- *The isomorphism problem for automatic order trees is Σ^1_1-complete.*
- *The isomorphism problem for automatic linear orders is Σ^1_1-complete.*

Isomorphism problem for automatic structures

The technique of coding polynomials by automata can be used to get further results:

Theorem 9 (Kuske, Liu, L 2010)

- *For all $n \geq 2$, the isomorphism problem for the class of automatic trees of height at most n is Π^0_{2n-3}-complete.*
- *The isomorphism problems for (i) automatic order trees of finite height and (ii) automatic order trees with only countably many infinite paths are equivalent to true arithmetic $\mathrm{FOTh}(\mathbb{N}, +, \times)$.*
- *The isomorphism problem for automatic order trees is Σ^1_1-complete.*
- *The isomorphism problem for automatic linear orders is Σ^1_1-complete.*
- *The isomorphism problem for scattered automatic linear orders can be reduced to $\mathrm{FOTh}(\mathbb{N}, +, \times)$ (but decidability is open).*

Isomorphism problem for ω-automatic structures

ω-automatic structures: Replace in the definition of automatic structures finite words by ω-words and finite automata by Büchi automata.

Isomorphism problem for ω-automatic structures

ω-automatic structures: Replace in the definition of automatic structures finite words by ω-words and finite automata by Büchi automata.

Theorem 10 (Kuske, Liu, L 2010)

- *The isomorphism problems for ω-automatic trees of finite height is at least as hard as true second-order arithmetic and hence does not belong to the analytical hierarchy.*

Isomorphism problem for ω-automatic structures

ω-automatic structures: Replace in the definition of automatic structures finite words by ω-words and finite automata by Büchi automata.

Theorem 10 (Kuske, Liu, L 2010)

- *The isomorphism problems for ω-automatic trees of finite height is at least as hard as true second-order arithmetic and hence does not belong to the analytical hierarchy.*
- *Assuming* **CH**, *the isomorphism problems for ω-automatic trees of finite height is equivalent to true second-order arithmetic.*

Isomorphism problem for tree-automatic structures

tree-automatic structures: Replace in the definition of automatic structures finite words by finite trees and finite word automata by tree automata.

Isomorphism problem for tree-automatic structures

tree-automatic structures: Replace in the definition of automatic structures finite words by finite trees and finite word automata by tree automata.

Theorem 11 (Kuske, 2012)

The isomorphism problem for scattered tree-automatic linear orders is undecidable (Π^0_1-hard).

Isomorphism problem for tree-automatic structures

tree-automatic structures: Replace in the definition of automatic structures finite words by finite trees and finite word automata by tree automata.

Theorem 11 (Kuske, 2012)

The isomorphism problem for scattered tree-automatic linear orders is undecidable (Π^0_1-hard).

Theorem 12 (Huschenbett, Kartzow, Liu, L 2012)

The isomorphism problem for well-founded tree-automatic order trees is complete for the hyperarithmetical level $\Delta^0_{\omega^\omega}$.

Complexity of isomorphisms between automatic structures

The class of hyperarithmetical sets is $\Sigma^1_1 \cap \Pi^1_1$.

Complexity of isomorphisms between automatic structures

The class of hyperarithmetical sets is $\Sigma_1^1 \cap \Pi_1^1$.

Theorem 13 (Kuske, Liu, L 2010)

There exist two isomorphic automatic order trees (automatic linear orders) $\mathcal{A}_1$ and $\mathcal{A}_2$ such that there exists no hyperarithmetical isomorphism between $\mathcal{A}_1$ and $\mathcal{A}_2$.

Complexity of isomorphisms between automatic structures

The class of hyperarithmetical sets is $\Sigma_1^1 \cap \Pi_1^1$.

Theorem 13 (Kuske, Liu, L 2010)

There exist two isomorphic automatic order trees (automatic linear orders) $\mathcal{A}_1$ and $\mathcal{A}_2$ such that there exists no hyperarithmetical isomorphism between $\mathcal{A}_1$ and $\mathcal{A}_2$.

Theorem 14 (Finkel 2010)

There exist two ω-tree automatic structures $\mathcal{A}_1$ and $\mathcal{A}_2$ and two models $\mathcal{S}_1$ and $\mathcal{S}_2$ of ZFC such that

- $\mathcal{S}_1 \models \mathcal{A}_1 \cong \mathcal{A}_2$
- $\mathcal{S}_2 \models \mathcal{A}_1 \not\cong \mathcal{A}_2$

Equational graphs, pushdown graphs

A graph is HR-equational if it can be generated by a deterministic hyperedge replacement grammar.

Equational graphs, pushdown graphs

A graph is HR-equational if it can be generated by a deterministic hyperedge replacement grammar.

Theorem 15 (Courcelle 1989)

The isomorphism problem for HR-equational graphs (and hence pushdown graphs) is decidable.

Equational graphs, pushdown graphs

A graph is HR-equational if it can be generated by a deterministic hyperedge replacement grammar.

Theorem 15 (Courcelle 1989)

The isomorphism problem for HR-equational graphs (and hence pushdown graphs) is decidable.

Courcelle proves that every HR-equational graph can be described up to isomorphism in MSO logic.

Equational graphs, pushdown graphs

A graph is HR-equational if it can be generated by a deterministic hyperedge replacement grammar.

Theorem 15 (Courcelle 1989)

The isomorphism problem for HR-equational graphs (and hence pushdown graphs) is decidable.

Courcelle proves that every HR-equational graph can be described up to isomorphism in MSO logic.

Open problem: Isomorphism problem for prefix recognizable graphs.

Regular trees and regular linear orders

Theorem 16 (L, Mathissen 2011)

It is P-complete to check for two given DFAs A_1, A_2, whether $(L(A_1), \leq_{pref}) \cong (L(A_2), \leq_{pref})$ (resp. $(L(A_1), \leq_{lex}) \cong (L(A_2), \leq_{lex})$).

Regular trees and regular linear orders

Theorem 16 (L, Mathissen 2011)

It is P-complete to check for two given DFAs A_1, A_2, whether $(L(A_1), \leq_{pref}) \cong (L(A_2), \leq_{pref})$ (resp. $(L(A_1), \leq_{lex}) \cong (L(A_2), \leq_{lex})$).

Theorem 17 (L, Mathissen 2011)

It is PSPACE-hard (and in EXPTIME) to check for two given NFAs A_1, A_2, whether $(L(A_1), \leq_{lex}) \cong (L(A_2), \leq_{lex})$.

Regular trees and regular linear orders

Theorem 16 (L, Mathissen 2011)

It is P-complete to check for two given DFAs A_1, A_2*, whether* $(L(A_1), \leq_{pref}) \cong (L(A_2), \leq_{pref})$ *(resp.* $(L(A_1), \leq_{lex}) \cong (L(A_2), \leq_{lex})$*).*

Theorem 17 (L, Mathissen 2011)

It is PSPACE-hard (and in EXPTIME) to check for two given NFAs A_1, A_2*, whether* $(L(A_1), \leq_{lex}) \cong (L(A_2), \leq_{lex})$*.*

Theorem 18 (L, Mathissen 2011)

It is EXPTIME-complete to check for two given NFAs A_1, A_2*, whether* $(L(A_1), \leq_{pref}) \cong (L(A_2), \leq_{pref})$*.*

Context-free languages

Theorem 19 (Kuske, Liu, L 2010)

It is Σ^1_1-complete to check for two given deterministic pushdown automata (even visibly pushdown automata) A_1, A_2, whether $(L(A_1), \leq_{pref}) \cong (L(A_2), \leq_{pref})$ (resp. $(L(A_1), \leq_{lex}) \cong (L(A_2), \leq_{lex})$)

Context-free languages

Theorem 19 (Kuske, Liu, L 2010)

It is Σ^1_1-complete to check for two given deterministic pushdown automata (even visibly pushdown automata) A_1, A_2, whether $(L(A_1), \leq_{pref}) \cong (L(A_2), \leq_{pref})$ (resp. $(L(A_1), \leq_{lex}) \cong (L(A_2), \leq_{lex})$)

Theorem 20 (Kuske 2013)

It is undecidable to check for deterministic real-time 1-counter automata A_1, A_2 with $L(A_1) \cap L(A_2) = \emptyset$, whether $(L(A_1) \cup L(A_2), \leq_{lex}) \cong (\mathbb{Q}, \leq)$.

Open problem

- Isomorphism problem for scattered automatic linear orders.

Open problem

- Isomorphism problem for scattered automatic linear orders.
- Isomorphism problem for prefix recognizable graphs.

Open problem

- Isomorphism problem for scattered automatic linear orders.
- Isomorphism problem for prefix recognizable graphs.
- Is the following problem Σ^1_1-complete: Is a given computable structure automatic?