Algorithmic theory of automatic structures

Markus Lohrey

Universität Leipzig

GAMES-EPIT 2011

Model theory

Model theory: Branch of mathematical logic that deals with the logical properties of mathematical structures.

Model theory

Model theory: Branch of mathematical logic that deals with the logical properties of mathematical structures.

Structure: $\mathcal{A} = (A, R_1, \ldots, R_n)$, where A is a non-empty set (universe) and $R_i \subseteq \underbrace{A \times A \times \cdots \times A}_{n_i \text{ many}}$ is a relation of arbitrary arity n_i $(1 \leq i \leq n)$.

Model theory

Model theory: Branch of mathematical logic that deals with the logical properties of mathematical structures.

Structure: $\mathcal{A} = (A, R_1, \ldots, R_n)$, where A is a non-empty set (universe) and $R_i \subseteq \underbrace{A \times A \times \cdots \times A}_{n_i \text{ many}}$ is a relation of arbitrary arity n_i $(1 \leq i \leq n)$.

Main focus of model theory:

Model theory

Model theory: Branch of mathematical logic that deals with the logical properties of mathematical structures.

Structure: $\mathcal{A} = (A, R_1, \ldots, R_n)$, where A is a non-empty set (universe) and $R_i \subseteq \underbrace{A \times A \times \cdots \times A}_{n_i \text{ many}}$ is a relation of arbitrary arity n_i $(1 \leq i \leq n)$.

Main focus of model theory:

- First-order logic (FO): Formulas have the form $x = y,\ R_i(x_1, \ldots, x_{n_i}),\ \neg\varphi,\ \varphi \wedge \psi,\ \varphi \vee \psi,\ \forall x : \varphi,\ \exists x : \varphi$, where variables take values from the universe A.

Model theory

Model theory: Branch of mathematical logic that deals with the logical properties of mathematical structures.

Structure: $\mathcal{A} = (A, R_1, \ldots, R_n)$, where A is a non-empty set (universe) and $R_i \subseteq \underbrace{A \times A \times \cdots \times A}_{n_i \text{ many}}$ is a relation of arbitrary arity n_i $(1 \leq i \leq n)$.

Main focus of model theory:

- First-order logic (FO): Formulas have the form $x = y,\ R_i(x_1, \ldots, x_{n_i}),\ \neg\varphi,\ \varphi \wedge \psi,\ \varphi \vee \psi,\ \forall x : \varphi,\ \exists x : \varphi$, where variables take values from the universe A.
- Mathematically interesting structures (linear orderings, groups, rings, fields, etc.)

Model theory

Model theory: Branch of mathematical logic that deals with the logical properties of mathematical structures.

Structure: $\mathcal{A} = (A, R_1, \ldots, R_n)$, where A is a non-empty set (universe) and $R_i \subseteq \underbrace{A \times A \times \cdots \times A}_{n_i \text{ many}}$ is a relation of arbitrary arity n_i $(1 \leq i \leq n)$.

Main focus of model theory:

- First-order logic (FO): Formulas have the form $x = y,\ R_i(x_1, \ldots, x_{n_i}),\ \neg\varphi,\ \varphi \wedge \psi,\ \varphi \vee \psi,\ \forall x : \varphi,\ \exists x : \varphi$, where variables take values from the universe A.
- Mathematically interesting structures (linear orderings, groups, rings, fields, etc.)
- Typical question: Is the first-order theory

$$\mathrm{FOTh}(\mathcal{A}) = \{\varphi \in \mathrm{FO} \mid \varphi \text{ has no free variables, } \mathcal{A} \models \varphi\}$$

of the structure $\mathcal{A}$ decidable?

Computable structures

Main point of criticism from computer science perspective:

Computable structures

Main point of criticism from computer science perspective:

Structures do not have a finite representation in general and cannot be processed by a machine.

Computable structures

Main point of criticism from computer science perspective:

Structures do not have a finite representation in general and cannot be processed by a machine.

Solution: Restrict to

Computable structures

Main point of criticism from computer science perspective:

Structures do not have a finite representation in general and cannot be processed by a machine.

Solution: Restrict to

- finite structures ($\rightsquigarrow$ finite model theory) or

Computable structures

Main point of criticism from computer science perspective:

Structures do not have a finite representation in general and cannot be processed by a machine.

Solution: Restrict to

- finite structures ($\rightsquigarrow$ finite model theory) or
- computable structures ($\rightsquigarrow$ computable model theory).

Computable structures

Main point of criticism from computer science perspective:

Structures do not have a finite representation in general and cannot be processed by a machine.

Solution: Restrict to

- finite structures ($\rightsquigarrow$ finite model theory) or
- computable structures ($\rightsquigarrow$ computable model theory).

Computable structure

A 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

Remarks:

Computable structures

Remarks:

- A structure, which is isomorphic to a computable structure will be called computable too.

Computable structures

Remarks:

- A structure, which is isomorphic to a computable structure will be called computable too.
- The restriction that the universe is a set of natural numbers is not essential — any set of finite objects is good as well.

Computable structures

Remarks:

- A structure, which is isomorphic to a computable structure will be called computable too.
- The restriction that the universe is a set of natural numbers is not essential — any set of finite objects is good as well.
- A computable structure $\mathcal{A} = (A, R_1, \ldots, R_n)$ can be represented by a tuple $(T, T_1, \ldots, T_n)$ of Turing machines.

Computable structures

Remarks:

- A structure, which is isomorphic to a computable structure will be called computable too.
- The restriction that the universe is a set of natural numbers is not essential — any set of finite objects is good as well.
- A computable structure $\mathcal{A} = (A, R_1, \ldots, R_n)$ can be represented by a tuple $(T, T_1, \ldots, T_n)$ of Turing machines.

Examples:

- Every finite structure

Computable structures

Remarks:

- A structure, which is isomorphic to a computable structure will be called computable too.
- The restriction that the universe is a set of natural numbers is not essential — any set of finite objects is good as well.
- A computable structure $\mathcal{A} = (A, R_1, \ldots, R_n)$ can be represented by a tuple $(T, T_1, \ldots, T_n)$ of Turing machines.

Examples:

- Every finite structure
- $(\mathbb{N}, +, \times)$

Computable structures

Remarks:

- A structure, which is isomorphic to a computable structure will be called computable too.
- The restriction that the universe is a set of natural numbers is not essential — any set of finite objects is good as well.
- A computable structure $\mathcal{A} = (A, R_1, \ldots, R_n)$ can be represented by a tuple $(T, T_1, \ldots, T_n)$ of Turing machines.

Examples:

- Every finite structure
- $(\mathbb{N}, +, \times)$
- Every free monoid Σ^* with Σ a finite alphabet.

Computable model theory

Computable model theory studies the logical properties of computable structures.

Computable model theory

Computable model theory studies the logical properties of computable structures.

Started to develop in the 50s simultanously in the western (Fröhlich, Sheperdsen, Rabin) and eastern hemisphere (Mal'cev, Ershov).

Computable model theory

Computable model theory studies the logical properties of computable structures.

Started to develop in the 50s simultanously in the western (Fröhlich, Sheperdsen, Rabin) and eastern hemisphere (Mal'cev, Ershov).

Main point of criticism from computer science perspective:

Computable model theory

Computable model theory studies the logical properties of computable structures.

Started to develop in the 50s simultanously in the western (Fröhlich, Sheperdsen, Rabin) and eastern hemisphere (Mal'cev, Ershov).

Main point of criticism from computer science perspective:

- Too general!

Computable model theory

Computable model theory studies the logical properties of computable structures.

Started to develop in the 50s simultanously in the western (Fröhlich, Sheperdsen, Rabin) and eastern hemisphere (Mal'cev, Ershov).

Main point of criticism from computer science perspective:

- Too general!
- Practically all interesting properties of computable structures are undecidable.

Computable model theory

Computable model theory studies the logical properties of computable structures.

Started to develop in the 50s simultanously in the western (Fröhlich, Sheperdsen, Rabin) and eastern hemisphere (Mal'cev, Ershov).

Main point of criticism from computer science perspective:

- Too general!
- Practically all interesting properties of computable structures are undecidable.

  Example: Has a computable graph an edge?

Computable model theory

Computable model theory studies the logical properties of computable structures.

Started to develop in the 50s simultanously in the western (Fröhlich, Sheperdsen, Rabin) and eastern hemisphere (Mal'cev, Ershov).

Main point of criticism from computer science perspective:

- Too general!
- Practically all interesting properties of computable structures are undecidable.

  Example: Has a computable graph an edge?

Solution: finite automata instead of Turing machines

Computable model theory

Computable model theory studies the logical properties of computable structures.

Started to develop in the 50s simultanously in the western (Fröhlich, Sheperdsen, Rabin) and eastern hemisphere (Mal'cev, Ershov).

Main point of criticism from computer science perspective:

- Too general!
- Practically all interesting properties of computable structures are undecidable.

  Example: Has a computable graph an edge?

Solution: finite automata instead of Turing machines

$\rightsquigarrow$ automatic structures

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 $\mathcal{A} = (A, R_1, \ldots, R_n)$ 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.

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 $(u, v) \in \Sigma^* \times \Sigma^*$:

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

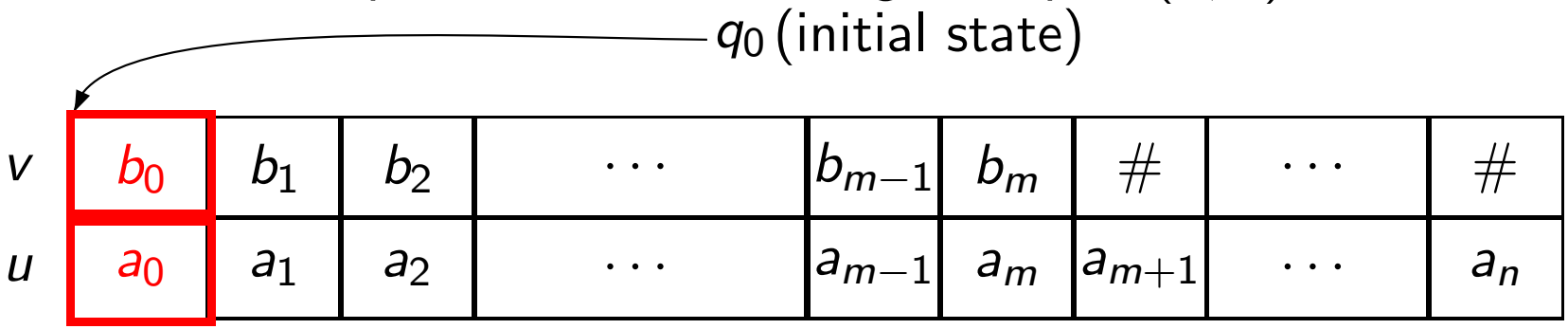
Automatic structure (Büchi 1960, Khoussainov, Nerode 1995)

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 $(u, v) \in \Sigma^* \times \Sigma^*$:



Automatic structures

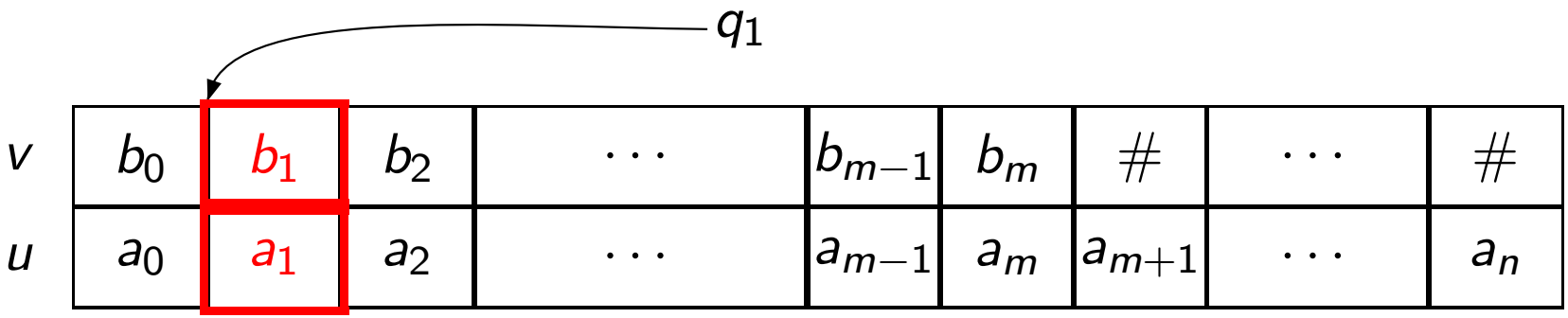
Automatic structure (Büchi 1960, Khoussainov, Nerode 1995)

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 $(u, v) \in \Sigma^* \times \Sigma^*$:



Automatic structures

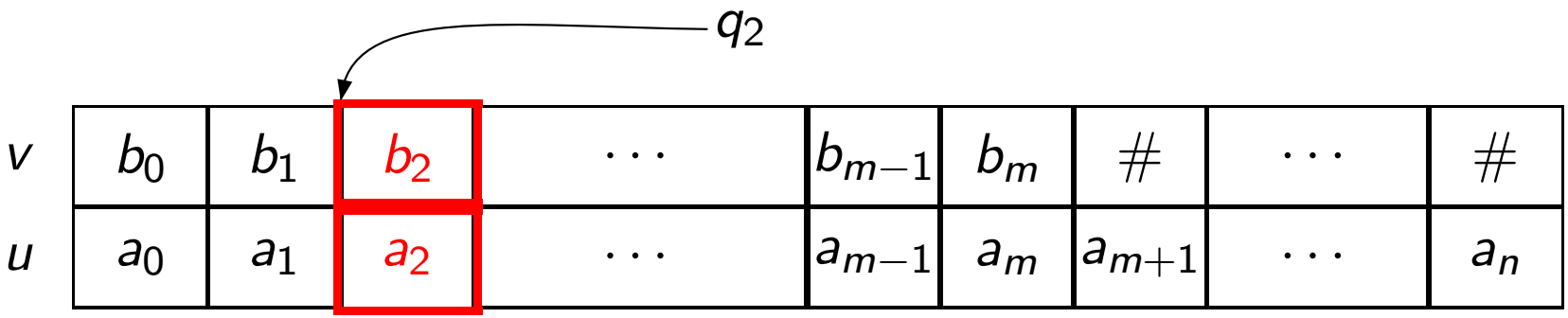
Automatic structure (Büchi 1960, Khoussainov, Nerode 1995)

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 $(u, v) \in \Sigma^* \times \Sigma^*$:



Automatic structures

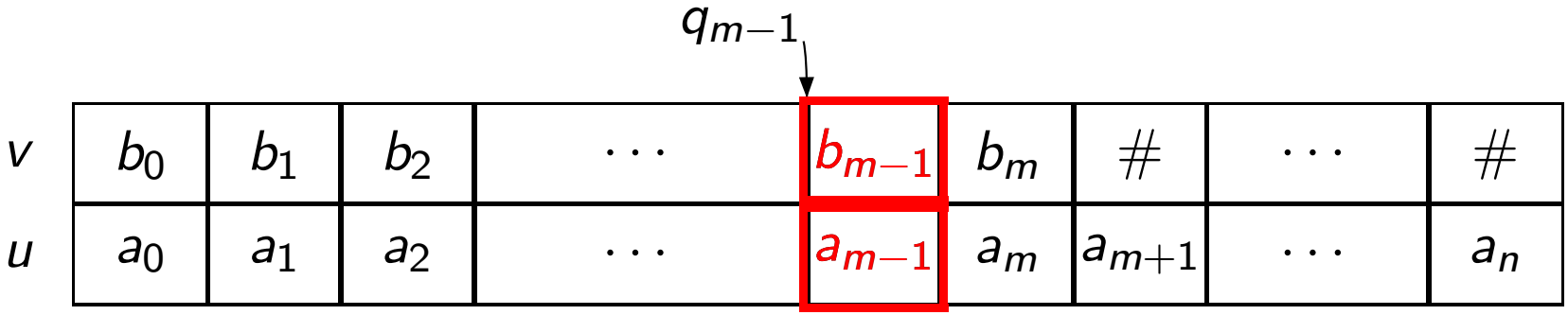
Automatic structure (Büchi 1960, Khoussainov, Nerode 1995)

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 $(u, v) \in \Sigma^* \times \Sigma^*$:



Automatic structures

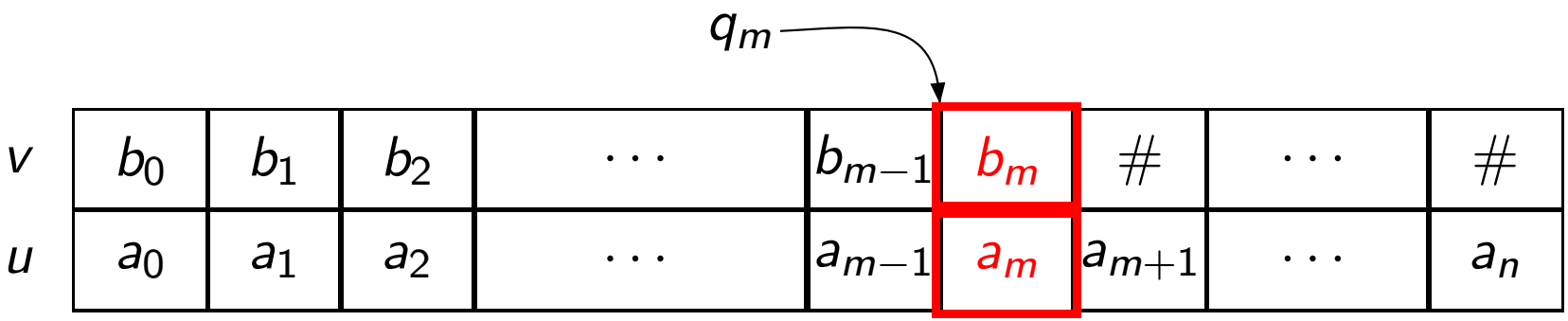
Automatic structure (Büchi 1960, Khoussainov, Nerode 1995)

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 $(u, v) \in \Sigma^* \times \Sigma^*$:



Automatic structures

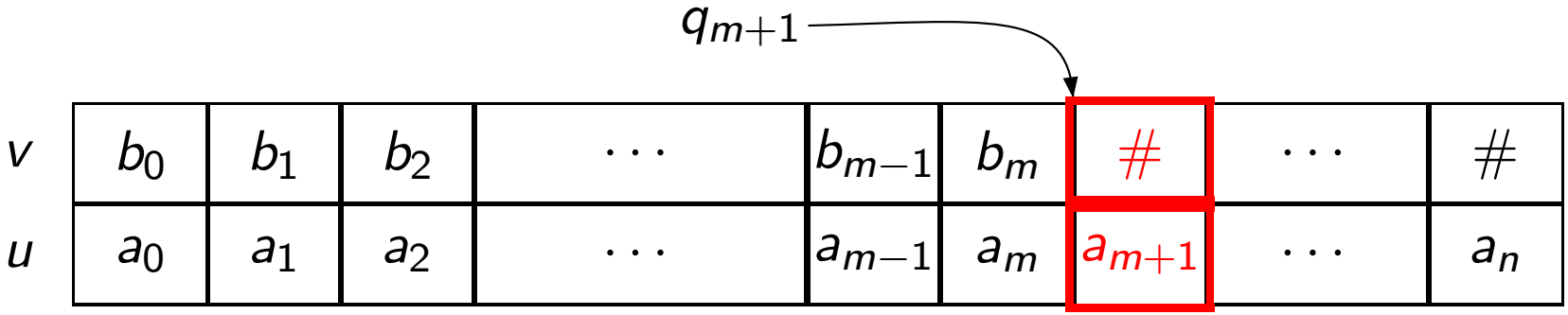
Automatic structure (Büchi 1960, Khoussainov, Nerode 1995)

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 $(u, v) \in \Sigma^* \times \Sigma^*$:



Automatic structures

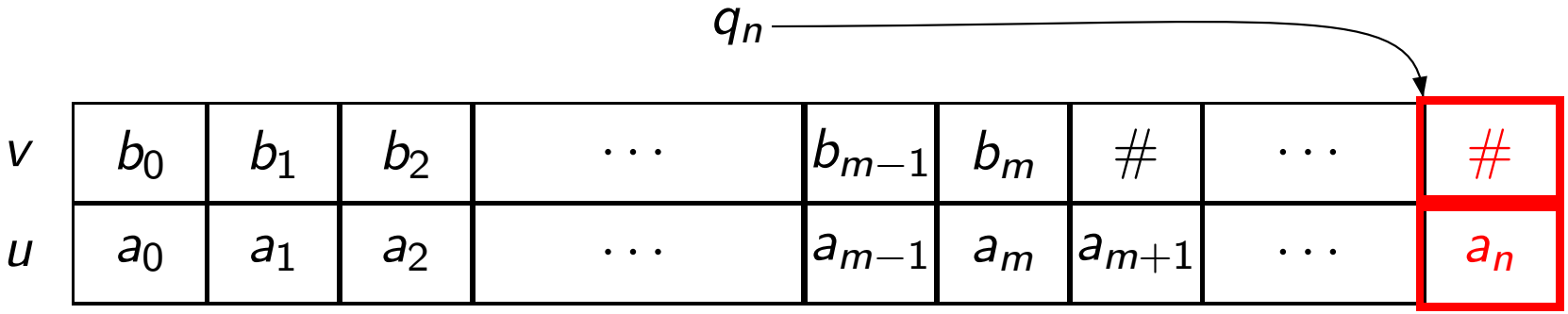
Automatic structure (Büchi 1960, Khoussainov, Nerode 1995)

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 $(u, v) \in \Sigma^* \times \Sigma^*$:



Automatic structures

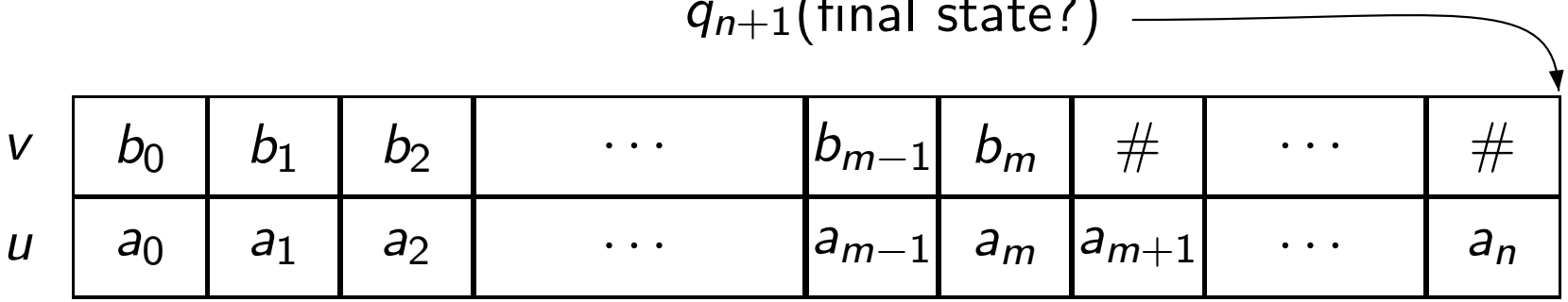
Automatic structure (Büchi 1960, Khoussainov, Nerode 1995)

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 $(u, v) \in \Sigma^* \times \Sigma^*$:



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
- $(\mathbb{Q}, \leq)$: $(\mathbb{Q}, \leq) \cong (\{0,1\}^*1, \leq_{\text{lex}})$

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

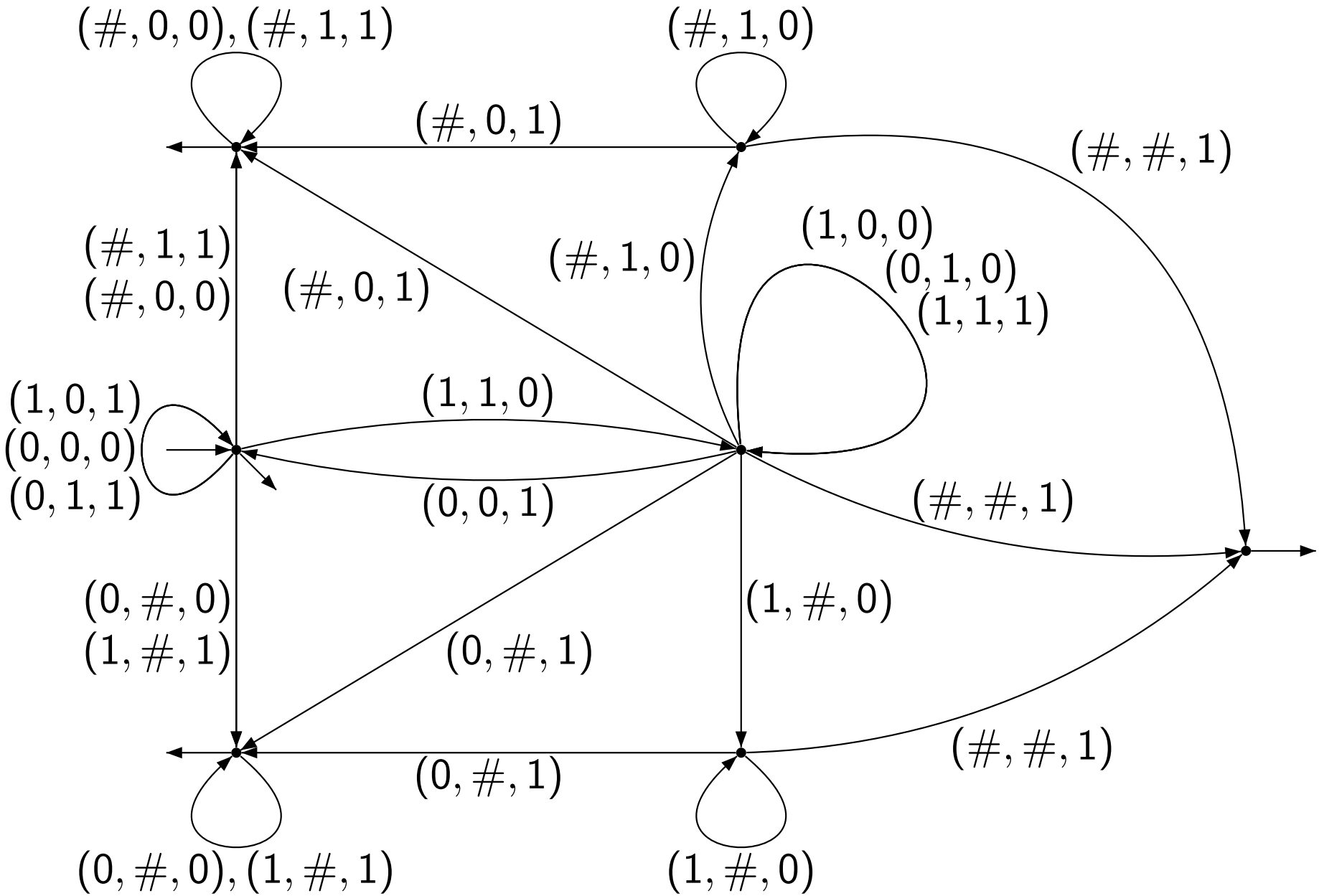
Remarks:

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

Example: $(\mathbb{N}, +)$ is automatic



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).

A universal automatic structures

Theorem 1 (Blumensath 1999)

A structure is automatic if and only if it is FO-interpretable in the structure

$$\mathcal{S} = (\{0,1\}^*, S_0, S_1, \preceq, \mathsf{el}),$$

where:

- $S_0 = \{(u, u0) \mid u \in \{0,1\}^*\}$,
- $S_1 = \{(u, u1) \mid u \in \{0,1\}^*\}$,
- $\preceq = \{(u, uv) \mid u, v \in \{0,1\}^*\}$ *(prefix relation)*
- $\mathsf{el} = \{(u, v) \mid |u| = |v|\}$ *(equal length).*

Automatic structures and FO

Remark: There are computable structures $\mathcal{A}$, for which FOTh($\mathcal{A}$) is undecidable, e.g., $\mathcal{A} = (\mathbb{N}, +, \times)$.

Automatic structures and FO

Remark: There are computable structures $\mathcal{A}$, for which FOTh($\mathcal{A}$) is undecidable, e.g., $\mathcal{A} = (\mathbb{N}, +, \times)$.

Theorem 2 (Khoussainov, Nerode 1995)

For every automatic structure $\mathcal{A}$, FOTh($\mathcal{A}$) *is decidable.*

Automatic structures and FO

Remark: There are computable structures $\mathcal{A}$, for which FOTh($\mathcal{A}$) is undecidable, e.g., $\mathcal{A} = (\mathbb{N}, +, \times)$.

Theorem 2 (Khoussainov, Nerode 1995)

For every automatic structure $\mathcal{A}$, FOTh($\mathcal{A}$) *is decidable.*

Proof:

For a given FO-formula $\varphi(x_1, \ldots, x_n)$, we build inductively over the structure of φ a synchronous n-tape automaton M_φ, which accepts the relation

$$\{(a_1, \ldots, a_n) \in \mathcal{A}^n \mid \mathcal{A} \models \varphi(a_1, \ldots, a_n)\}.$$

Automatic structures and FO

Remark: There are computable structures $\mathcal{A}$, for which FOTh($\mathcal{A}$) is undecidable, e.g., $\mathcal{A} = (\mathbb{N}, +, \times)$.

Theorem 2 (Khoussainov, Nerode 1995)

For every automatic structure $\mathcal{A}$, FOTh($\mathcal{A}$) *is decidable.*

Proof:

For a given FO-formula $\varphi(x_1, \ldots, x_n)$, we build inductively over the structure of φ a synchronous n-tape automaton M_φ, which accepts the relation

$$\{(a_1, \ldots, a_n) \in \mathcal{A}^n \mid \mathcal{A} \models \varphi(a_1, \ldots, a_n)\}.$$

For this, we use well-known closure properties of regular languages.

Automatic structures and FO

Case 1: $\varphi(x_1, \ldots, x_n)$ is atomic.

Automatic structures and FO

Case 1: $\varphi(x_1, \ldots, x_n)$ is atomic.

- Then M_φ exist since $\mathcal{A}$ is automatic.

Automatic structures and FO

Case 1: $\varphi(x_1, \ldots, x_n)$ is atomic.

- Then M_φ exist since $\mathcal{A}$ is automatic.

Case 2: $\varphi(x_1, \ldots, x_n) = \neg\psi(x_1, \ldots, x_n)$.

Automatic structures and FO

Case 1: $\varphi(x_1, \ldots, x_n)$ is atomic.

- Then M_φ exist since $\mathcal{A}$ is automatic.

Case 2: $\varphi(x_1, \ldots, x_n) = \neg\psi(x_1, \ldots, x_n)$.

- Complement the automaton M_ψ.

Automatic structures and FO

Case 1: $\varphi(x_1, \ldots, x_n)$ is atomic.

- Then M_φ exist since $\mathcal{A}$ is automatic.

Case 2: $\varphi(x_1, \ldots, x_n) = \neg\psi(x_1, \ldots, x_n)$.

- Complement the automaton M_ψ.
- Intersect with the automaton that recognizes the set of all convolutions of n strings.

Automatic structures and FO

Case 1: $\varphi(x_1, \ldots, x_n)$ is atomic.

- Then M_φ exist since $\mathcal{A}$ is automatic.

Case 2: $\varphi(x_1, \ldots, x_n) = \neg\psi(x_1, \ldots, x_n)$.

- Complement the automaton M_ψ.
- Intersect with the automaton that recognizes the set of all convolutions of n strings.

Case 3: $\varphi(x_1, \ldots, x_n) = \psi_1(x_1, \ldots, x_n) \vee \psi_2(x_1, \ldots, x_n)$.

Automatic structures and FO

Case 1: $\varphi(x_1, \ldots, x_n)$ is atomic.

- Then M_φ exist since $\mathcal{A}$ is automatic.

Case 2: $\varphi(x_1, \ldots, x_n) = \neg\psi(x_1, \ldots, x_n)$.

- Complement the automaton M_ψ.
- Intersect with the automaton that recognizes the set of all convolutions of n strings.

Case 3: $\varphi(x_1, \ldots, x_n) = \psi_1(x_1, \ldots, x_n) \vee \psi_2(x_1, \ldots, x_n)$.

- Construct automaton for the union of $L(M_{\psi_1})$ and $L(M_{\psi_2})$.

Automatic structures and FO

Case 4: $\varphi(x_1, \ldots, x_n) = \exists x_0 : \psi(x_0, x_1, \ldots, x_n)$

Automatic structures and FO

Case 4: $\varphi(x_1, \ldots, x_n) = \exists x_0 : \psi(x_0, x_1, \ldots, x_n)$

- Let $h : ((\Sigma \cup \{\#\})^{n+1})^* \to ((\Sigma \cup \{\#\})^n)^*$ be the homomorphism with

$$h(a_0, a_1, \ldots, a_n) = \begin{cases} \varepsilon & \text{if } a_1 = \cdots = a_n = \# \\ (a_1, \ldots, a_n) & \text{else} \end{cases}$$

Automatic structures and FO

Case 4: $\varphi(x_1, \ldots, x_n) = \exists x_0 : \psi(x_0, x_1, \ldots, x_n)$

- Let $h : ((\Sigma \cup \{\#\})^{n+1})^* \to ((\Sigma \cup \{\#\})^n)^*$ be the homomorphism with

$$h(a_0, a_1, \ldots, a_n) = \begin{cases} \varepsilon & \text{if } a_1 = \cdots = a_n = \# \\ (a_1, \ldots, a_n) & \text{else} \end{cases}$$

- Then we take for M_φ an automaton that recognizes $h(L(M_\psi))$.

Automatic structures and FO

Case 4: $\varphi(x_1, \ldots, x_n) = \exists x_0 : \psi(x_0, x_1, \ldots, x_n)$

- Let $h : ((\Sigma \cup \{\#\})^{n+1})^* \to ((\Sigma \cup \{\#\})^n)^*$ be the homomorphism with

$$h(a_0, a_1, \ldots, a_n) = \begin{cases} \varepsilon & \text{if } a_1 = \cdots = a_n = \# \\ (a_1, \ldots, a_n) & \text{else} \end{cases}$$

- Then we take for M_φ an automaton that recognizes $h(L(M_\psi))$.

This concludes the construction of M_φ.

Automatic structures and FO

Case 4: $\varphi(x_1, \ldots, x_n) = \exists x_0 : \psi(x_0, x_1, \ldots, x_n)$

- Let $h : ((\Sigma \cup \{\#\})^{n+1})^* \to ((\Sigma \cup \{\#\})^n)^*$ be the homomorphism with

$$h(a_0, a_1, \ldots, a_n) = \begin{cases} \varepsilon & \text{if } a_1 = \cdots = a_n = \# \\ (a_1, \ldots, a_n) & \text{else} \end{cases}$$

- Then we take for M_φ an automaton that recognizes $h(L(M_\psi))$.

This concludes the construction of M_φ.

Now, assume that we want to check whether a FO-sentence φ belongs to $\mathrm{FOTh}(\mathcal{A})$.

Automatic structures and FO

Case 4: $\varphi(x_1, \ldots, x_n) = \exists x_0 : \psi(x_0, x_1, \ldots, x_n)$

- Let $h : ((\Sigma \cup \{\#\})^{n+1})^* \to ((\Sigma \cup \{\#\})^n)^*$ be the homomorphism with

$$h(a_0, a_1, \ldots, a_n) = \begin{cases} \varepsilon & \text{if } a_1 = \cdots = a_n = \# \\ (a_1, \ldots, a_n) & \text{else} \end{cases}$$

- Then we take for M_φ an automaton that recognizes $h(L(M_\psi))$.

This concludes the construction of M_φ.

Now, assume that we want to check whether a FO-sentence φ belongs to $\mathrm{FOTh}(\mathcal{A})$.

W.l.o.g. we can assume that $\varphi = \exists x : \psi(x)$.

Automatic structures and FO

Case 4: $\varphi(x_1, \ldots, x_n) = \exists x_0 : \psi(x_0, x_1, \ldots, x_n)$

- Let $h : ((\Sigma \cup \{\#\})^{n+1})^* \to ((\Sigma \cup \{\#\})^n)^*$ be the homomorphism with

$$h(a_0, a_1, \ldots, a_n) = \begin{cases} \varepsilon & \text{if } a_1 = \cdots = a_n = \# \\ (a_1, \ldots, a_n) & \text{else} \end{cases}$$

- Then we take for M_φ an automaton that recognizes $h(L(M_\psi))$.

This concludes the construction of M_φ.

Now, assume that we want to check whether a FO-sentence φ belongs to $\mathrm{FOTh}(\mathcal{A})$.

W.l.o.g. we can assume that $\varphi = \exists x : \psi(x)$.

Construct the automaton M_ψ and check, whether $L(M_\psi) = \emptyset$. □

Complexity of FO

Main bottle neck in the construction: Negation!

Complexity of FO

Main bottle neck in the construction: Negation!

For the construction of the automaton for $\neg\psi(x_1, \ldots, x_n)$ from the automaton for $\psi(x_1, \ldots, x_n)$, we have to complement the latter. (power set construction of Rabin and Scott).

Complexity of FO

Main bottle neck in the construction: Negation!

For the construction of the automaton for $\neg\psi(x_1, \ldots, x_n)$ from the automaton for $\psi(x_1, \ldots, x_n)$, we have to complement the latter. (power set construction of Rabin and Scott).

$\rightsquigarrow$ exponential blow-up.

Complexity of FO

Main bottle neck in the construction: Negation!

For the construction of the automaton for $\neg\psi(x_1, \ldots, x_n)$ from the automaton for $\psi(x_1, \ldots, x_n)$, we have to complement the latter. (power set construction of Rabin and Scott).

$\rightsquigarrow$ exponential blow-up.

This is unavoidable:

Complexity of FO

Main bottle neck in the construction: Negation!

For the construction of the automaton for $\neg\psi(x_1, \ldots, x_n)$ from the automaton for $\psi(x_1, \ldots, x_n)$, we have to complement the latter. (power set construction of Rabin and Scott).

$\rightsquigarrow$ exponential blow-up.

This is unavoidable:

Theorem 3 (Blumensath, Grädel 2000)

There is an automatic structure $\mathcal{A}$, such that $\mathrm{FOTh}(\mathcal{A})$ *is not elementary decidable.*

Complexity of FO

Main bottle neck in the construction: Negation!

For the construction of the automaton for $\neg\psi(x_1, \ldots, x_n)$ from the automaton for $\psi(x_1, \ldots, x_n)$, we have to complement the latter. (power set construction of Rabin and Scott).

$\rightsquigarrow$ exponential blow-up.

This is unavoidable:

Theorem 3 (Blumensath, Grädel 2000)

There is an automatic structure $\mathcal{A}$, such that $\mathrm{FOTh}(\mathcal{A})$ *is not elementary decidable.*

A problem P is elementary decidable if there is $c \geq 0$ and an algorithm, which decides P in time $\underbrace{2\hat{}2\hat{} \cdots \hat{}2\hat{}n}_{c \text{ times}}$.

Complexity of the FO-theory

Example: an automatic structure, whose FO-theory is not elementarily decidable is:

$$\begin{aligned}
\mathcal{A} &= (\{0,1\}^*, S_0, S_1, \preceq), \text{ with} \\
S_0 &= \{(w, w0) \mid w \in \{0,1\}^*\} \\
S_1 &= \{(w, w1) \mid w \in \{0,1\}^*\} \\
\preceq &= \{(u, uv) \mid u, v \in \{0,1\}^*\}.
\end{aligned}$$

Complexity of the FO-theory

Example: an automatic structure, whose FO-theory is not elementarily decidable is:

$$\begin{aligned} \mathcal{A} &= (\{0,1\}^*, S_0, S_1, \preceq), \text{ with} \\ S_0 &= \{(w, w0) \mid w \in \{0,1\}^*\} \\ S_1 &= \{(w, w1) \mid w \in \{0,1\}^*\} \\ \preceq &= \{(u, uv) \mid u, v \in \{0,1\}^*\}. \end{aligned}$$

Question: Are there interesting classes of automatic structures, for which the FO-theory is elementarily decidable?

Automatic structures of bounded degree

Gaifman graph, bounded degree

The Gaifman graph of the structure $\mathcal{A} = (A, R_1, \ldots, R_n)$ is the undirected graph (A, E) with the edge relation

$$E = \{(a, b) \mid \ a \neq b,\ \exists i, \overline{c} \in R_i : a \text{ and } b \text{ appear in } \overline{c}\}.$$

Automatic structures of bounded degree

Gaifman graph, bounded degree

The Gaifman graph of the structure $\mathcal{A} = (A, R_1, \ldots, R_n)$ is the undirected graph (A, E) with the edge relation

$$E = \{(a, b) \mid \ a \neq b, \ \exists i, \overline{c} \in R_i : a \text{ and } b \text{ appear in } \overline{c}\}.$$

$\mathcal{A}$ has bounded degree if:

$$\exists \delta \in \mathbb{N} \ \forall a \in A : \ a \text{ has } \leq \delta \text{ many neighbours in the Gaifman graph}$$

Automatic structures of bounded degree

Gaifman graph, bounded degree

The Gaifman graph of the structure $\mathcal{A} = (A, R_1, \ldots, R_n)$ is the undirected graph (A, E) with the edge relation

$$E = \{(a, b) \mid \ a \neq b, \ \exists i, \overline{c} \in R_i : a \text{ and } b \text{ appear in } \overline{c}\}.$$

$\mathcal{A}$ has bounded degree if:

$$\exists \delta \in \mathbb{N} \ \forall a \in A : \ a \text{ has } \leq \delta \text{ many neighbours in the Gaifman graph}$$

- $(\mathbb{Q}, \leq)$:

Automatic structures of bounded degree

Gaifman graph, bounded degree

The Gaifman graph of the structure $\mathcal{A} = (A, R_1, \ldots, R_n)$ is the undirected graph (A, E) with the edge relation

$$E = \{(a, b) \mid \ a \neq b, \ \exists i, \overline{c} \in R_i : a \text{ and } b \text{ appear in } \overline{c}\}.$$

$\mathcal{A}$ has bounded degree if:

$$\exists \delta \in \mathbb{N} \ \forall a \in A : \ a \text{ has } \leq \delta \text{ many neighbours in the Gaifman graph}$$

- $(\mathbb{Q}, \leq)$: unbounded degree

Automatic structures of bounded degree

Gaifman graph, bounded degree

The Gaifman graph of the structure $\mathcal{A} = (A, R_1, \ldots, R_n)$ is the undirected graph (A, E) with the edge relation

$$E = \{(a, b) \mid \ a \neq b,\ \exists i, \overline{c} \in R_i : a \text{ and } b \text{ appear in } \overline{c}\}.$$

$\mathcal{A}$ has bounded degree if:

$$\exists \delta \in \mathbb{N}\ \forall a \in A : \ a \text{ has } \leq \delta \text{ many neighbours in the Gaifman graph}$$

- $(\mathbb{Q}, \leq)$: unbounded degree
- $(\mathbb{N}, +)$, $(\mathbb{Z}, +)$:

Automatic structures of bounded degree

Gaifman graph, bounded degree

The Gaifman graph of the structure $\mathcal{A} = (A, R_1, \ldots, R_n)$ is the undirected graph (A, E) with the edge relation

$$E = \{(a, b) \mid \ a \neq b,\ \exists i, \overline{c} \in R_i : a \text{ and } b \text{ appear in } \overline{c}\}.$$

$\mathcal{A}$ has bounded degree if:

$$\exists \delta \in \mathbb{N}\ \forall a \in A : \ a \text{ has } \leq \delta \text{ many neighbours in the Gaifman graph}$$

- $(\mathbb{Q}, \leq)$: unbounded degree
- $(\mathbb{N}, +)$, $(\mathbb{Z}, +)$: unbounded degree

Automatic structures of bounded degree

Gaifman graph, bounded degree

The Gaifman graph of the structure $\mathcal{A} = (A, R_1, \ldots, R_n)$ is the undirected graph (A, E) with the edge relation

$$E = \{(a, b) \mid \ a \neq b, \ \exists i, \overline{c} \in R_i : a \text{ and } b \text{ appear in } \overline{c}\}.$$

$\mathcal{A}$ has bounded degree if:

$$\exists \delta \in \mathbb{N} \ \forall a \in A : \ a \text{ has } \leq \delta \text{ many neighbours in the Gaifman graph}$$

- $(\mathbb{Q}, \leq)$: unbounded degree
- $(\mathbb{N}, +)$, $(\mathbb{Z}, +)$: unbounded degree
- Configuration graphs of Turing machines:

Automatic structures of bounded degree

Gaifman graph, bounded degree

The Gaifman graph of the structure $\mathcal{A} = (A, R_1, \ldots, R_n)$ is the undirected graph (A, E) with the edge relation

$$E = \{(a, b) \mid \ a \neq b, \ \exists i, \overline{c} \in R_i : a \text{ and } b \text{ appear in } \overline{c}\}.$$

$\mathcal{A}$ has bounded degree if:

$$\exists \delta \in \mathbb{N} \ \forall a \in A : \ a \text{ has } \leq \delta \text{ many neighbours in the Gaifman graph}$$

- $(\mathbb{Q}, \leq)$: unbounded degree
- $(\mathbb{N}, +)$, $(\mathbb{Z}, +)$: unbounded degree
- Configuration graphs of Turing machines: bounded degree

Automatic structures of bounded degree

Theorem 4 (Kuske, L 2009)

Let $\mathcal{A}$ be an automatic structure of bounded degree. Then, FOTh($\mathcal{A}$) *belongs to the class* 2EXPSPACE.

Automatic structures of bounded degree

Theorem 4 (Kuske, L 2009)

Let $\mathcal{A}$ be an automatic structure of bounded degree. Then, FOTh($\mathcal{A}$) *belongs to the class* 2EXPSPACE.

- 2EXPSPACE denotes the class of all problems that can be decided on a Turing machine in space $2^{2^{\text{poly}(n)}}$.

Automatic structures of bounded degree

Theorem 4 (Kuske, L 2009)

Let $\mathcal{A}$ be an automatic structure of bounded degree. Then, FOTh($\mathcal{A}$) *belongs to the class* 2EXPSPACE.

- 2EXPSPACE denotes the class of all problems that can be decided on a Turing machine in space $2^{2^{\text{poly}(n)}}$.
- The upper bound 2EXPSPACE holds even uniformely, i.e., if the automatic structure is part of the input.

Automatic structures of bounded degree

Theorem 4 (Kuske, L 2009)

Let $\mathcal{A}$ be an automatic structure of bounded degree. Then, FOTh($\mathcal{A}$) *belongs to the class* 2EXPSPACE.

- 2EXPSPACE denotes the class of all problems that can be decided on a Turing machine in space $2^{2^{\text{poly}(n)}}$.
- The upper bound 2EXPSPACE holds even uniformely, i.e., if the automatic structure is part of the input.
- The proof uses Gaifman's locality theorem.

Automatic structures of bounded degree

Theorem 4 (Kuske, L 2009)

Let $\mathcal{A}$ be an automatic structure of bounded degree. Then, FOTh($\mathcal{A}$) *belongs to the class* 2EXPSPACE.

- 2EXPSPACE denotes the class of all problems that can be decided on a Turing machine in space $2^{2^{\text{poly}(n)}}$.
- The upper bound 2EXPSPACE holds even uniformely, i.e., if the automatic structure is part of the input.
- The proof uses Gaifman's locality theorem.

Theorem 5 (Kuske, L 2009)

There is an automatic structure $\mathcal{A}$ of bounded degree such that FOTh($\mathcal{A}$) *is complete for the class* 2EXPSPACE.

Balls

Let $\mathcal{A} = (A, R_1, \ldots, R_n)$ be a structure, $G = (A, E)$ its Gaifman graph, and $\bar{a} = (a_1, \ldots, a_k) \in A^k$, $r \geq 0$.

Balls

Let $\mathcal{A} = (A, R_1, \ldots, R_n)$ be a structure, $G = (A, E)$ its Gaifman graph, and $\bar{a} = (a_1, \ldots, a_k) \in A^k$, $r \geq 0$.

The ball of radius r around the tuple $\bar{a}$ is

$$\mathrm{ball}_{\mathcal{A}}(r, \bar{a}) = \bigcup_{i=1}^{k} \{b \in A \mid b \text{ has distance} \leq r \text{ from } a_i \text{ in } G\}.$$

Balls

Let $\mathcal{A} = (A, R_1, \ldots, R_n)$ be a structure, $G = (A, E)$ its Gaifman graph, and $\bar{a} = (a_1, \ldots, a_k) \in A^k$, $r \geq 0$.

The ball of radius r around the tuple $\bar{a}$ is

$$\mathrm{ball}_{\mathcal{A}}(r, \bar{a}) = \bigcup_{i=1}^{k} \{b \in A \mid b \text{ has distance} \leq r \text{ from } a_i \text{ in } G\}.$$

We identify $\mathrm{ball}_{\mathcal{A}}(r, \bar{a})$ with the substructure of $\mathcal{A}$ induced by the set $\mathrm{ball}_{\mathcal{A}}(r, \bar{a})$.

Balls

Let $\mathcal{A} = (A, R_1, \ldots, R_n)$ be a structure, $G = (A, E)$ its Gaifman graph, and $\bar{a} = (a_1, \ldots, a_k) \in A^k$, $r \geq 0$.

The ball of radius r around the tuple $\bar{a}$ is

$$\mathrm{ball}_{\mathcal{A}}(r, \bar{a}) = \bigcup_{i=1}^{k} \{b \in A \mid b \text{ has distance} \leq r \text{ from } a_i \text{ in } G\}.$$

We identify $\mathrm{ball}_{\mathcal{A}}(r, \bar{a})$ with the substructure of $\mathcal{A}$ induced by the set $\mathrm{ball}_{\mathcal{A}}(r, \bar{a})$.

For $\bar{a}, \bar{b} \in A^k$ we write $\mathrm{ball}_{\mathcal{A}}(r, \bar{a}) \cong \mathrm{ball}_{\mathcal{A}}(r, \bar{b})$ if there exists an isomorphism

$$f : \mathrm{ball}_{\mathcal{A}}(r, \bar{a}) \to \mathrm{ball}_{\mathcal{A}}(r, \bar{b})$$

that maps the i^{th} entry of $\bar{a}$ to the i^{th} entry of $\bar{b}$ (for all $1 \leq i \leq k$).

Gaifman's Locality Theorem

Theorem 6 (Gaifman's Locality Theorem, 1982)

Let $\bar{a}, \bar{b} \in A^k$ *and* $d \geq 0$ *such that* $\mathsf{ball}_{\mathcal{A}}(7^d, \bar{a}) \cong \mathsf{ball}_{\mathcal{A}}(7^d, \bar{b})$.

Gaifman's Locality Theorem

Theorem 6 (Gaifman's Locality Theorem, 1982)

Let $\bar{a}, \bar{b} \in A^k$ *and* $d \geq 0$ *such that* $\mathrm{ball}_{\mathcal{A}}(7^d, \bar{a}) \cong \mathrm{ball}_{\mathcal{A}}(7^d, \bar{b})$.

Then, for every FO-formula $\varphi(x_1, \ldots, x_k)$ *of quantifier nesting depth* $\leq d$, *we have:*

$$\mathcal{A} \models \varphi(\bar{a}) \iff \mathcal{A} \models \varphi(\bar{b})$$

Realizability

For $\bar{a} = (a_1, \ldots, a_k) \in A^k$ and $k \leq d$ let

$$\mathrm{ball}^{\downarrow}_{\mathcal{A}}(7^d, \bar{a}) = \bigcup_{i=1}^{k} \{b \in \mathcal{A} \mid b \text{ has distance} \leq 7^{d-i+1} \text{ from } a_i \text{ in } G\}.$$

Realizability

For $\bar{a} = (a_1, \ldots, a_k) \in A^k$ and $k \leq d$ let

$$\mathrm{ball}^{\downarrow}_{\mathcal{A}}(7^d, \bar{a}) = \bigcup_{i=1}^{k} \{b \in \mathcal{A} \mid b \text{ has distance} \leq 7^{d-i+1} \text{ from } a_i \text{ in } G\}.$$

The notation $\mathrm{ball}^{\downarrow}_{\mathcal{A}}(7^d, \bar{a}) \cong \mathrm{ball}^{\downarrow}_{\mathcal{A}}(7^d, \bar{b})$ is defined as for ordinary balls.

Realizability

For $\bar{a} = (a_1, \ldots, a_k) \in A^k$ and $k \leq d$ let

$$\mathrm{ball}^{\downarrow}_{\mathcal{A}}(7^d, \bar{a}) = \bigcup_{i=1}^{k} \{b \in \mathcal{A} \mid b \text{ has distance} \leq 7^{d-i+1} \text{ from } a_i \text{ in } G\}.$$

The notation $\mathrm{ball}^{\downarrow}_{\mathcal{A}}(7^d, \bar{a}) \cong \mathrm{ball}^{\downarrow}_{\mathcal{A}}(7^d, \bar{b})$ is defined as for ordinary balls.

A (d, k)-ball $(k \leq d)$ is a tuple $(\mathcal{B}, \bar{b})$ such that:

- $\mathcal{B}$ is a structure over the same signature as $\mathcal{A}$ with $\bar{b} \in \mathcal{B}^k$,
- $(\mathcal{B}, \bar{b}) = \mathrm{ball}^{\downarrow}_{\mathcal{B}}(7^d, \bar{b})$.

Realizability

For $\bar{a} = (a_1, \ldots, a_k) \in A^k$ and $k \leq d$ let

$$\mathrm{ball}^{\downarrow}_{\mathcal{A}}(7^d, \bar{a}) = \bigcup_{i=1}^{k} \{b \in \mathcal{A} \mid b \text{ has distance} \leq 7^{d-i+1} \text{ from } a_i \text{ in } G\}.$$

The notation $\mathrm{ball}^{\downarrow}_{\mathcal{A}}(7^d, \bar{a}) \cong \mathrm{ball}^{\downarrow}_{\mathcal{A}}(7^d, \bar{b})$ is defined as for ordinary balls.

A (d, k)-ball $(k \leq d)$ is a tuple $(\mathcal{B}, \bar{b})$ such that:

- $\mathcal{B}$ is a structure over the same signature as $\mathcal{A}$ with $\bar{b} \in \mathcal{B}^k$,
- $(\mathcal{B}, \bar{b}) = \mathrm{ball}^{\downarrow}_{\mathcal{B}}(7^d, \bar{b})$.

The (d, k)-ball $(\mathcal{B}, \bar{b})$ is realizable in the structure $\mathcal{A}$ if

$$\exists \bar{a} \in A^k : \mathrm{ball}^{\downarrow}_{\mathcal{A}}(7^d, \bar{a}) \simeq (\mathcal{B}, \bar{b}) \ .$$

Realizability

For $\bar{a} = (a_1, \ldots, a_k) \in A^k$ and $k \leq d$ let

$$\mathrm{ball}^{\downarrow}_{\mathcal{A}}(7^d, \bar{a}) = \bigcup_{i=1}^{k} \{b \in \mathcal{A} \mid b \text{ has distance} \leq 7^{d-i+1} \text{ from } a_i \text{ in } G\}.$$

The notation $\mathrm{ball}^{\downarrow}_{\mathcal{A}}(7^d, \bar{a}) \cong \mathrm{ball}^{\downarrow}_{\mathcal{A}}(7^d, \bar{b})$ is defined as for ordinary balls.

A (d, k)-ball $(k \leq d)$ is a tuple $(\mathcal{B}, \bar{b})$ such that:

- $\mathcal{B}$ is a structure over the same signature as $\mathcal{A}$ with $\bar{b} \in \mathcal{B}^k$,
- $(\mathcal{B}, \bar{b}) = \mathrm{ball}^{\downarrow}_{\mathcal{B}}(7^d, \bar{b})$.

The (d, k)-ball $(\mathcal{B}, \bar{b})$ is realizable in the structure $\mathcal{A}$ if

$$\exists \bar{a} \in A^k : \mathrm{ball}^{\downarrow}_{\mathcal{A}}(7^d, \bar{a}) \simeq (\mathcal{B}, \bar{b}) \ .$$

A $(d, k+1)$-ball $(\mathcal{C}, \bar{b}, b_{k+1})$ extends the (d, k)-ball $(\mathcal{B}, \bar{b})$ if

$$\mathrm{ball}^{\downarrow}_{\mathcal{C}}(7^d, \bar{b}) = (\mathcal{B}, \bar{b}) \ .$$

The actual algorithm

Let $\mathcal{A}$ be an automatic structure of bounded degree.

The actual algorithm

Let $\mathcal{A}$ be an automatic structure of bounded degree.

Let $\varphi = \exists x_1 \forall x_2 \exists x_3 \cdots \forall x_d : \psi(x_1, \ldots, x_d)$ be a first-order sentence.

The actual algorithm

Let $\mathcal{A}$ be an automatic structure of bounded degree.

Let $\varphi = \exists x_1 \forall x_2 \exists x_3 \cdots \forall x_d : \psi(x_1, \ldots, x_d)$ be a first-order sentence.

We check $\mathcal{A} \models \psi$ on an alternating exponential time machine:

- Guess existentially a $(d, 1)$-ball $(\mathcal{B}_1, b_1)$, which is realizable in $\mathcal{A}$.

The actual algorithm

Let $\mathcal{A}$ be an automatic structure of bounded degree.

Let $\varphi = \exists x_1 \forall x_2 \exists x_3 \cdots \forall x_d : \psi(x_1, \ldots, x_d)$ be a first-order sentence.

We check $\mathcal{A} \models \psi$ on an alternating exponential time machine:

- Guess existentially a $(d, 1)$-ball $(\mathcal{B}_1, b_1)$, which is realizable in $\mathcal{A}$.
- Guess universally a $(d, 2)$-ball $(\mathcal{B}_2, b_1, b_2)$ extending $(\mathcal{B}_1, b_1)$, which is realizable in $\mathcal{A}$.

The actual algorithm

Let $\mathcal{A}$ be an automatic structure of bounded degree.

Let $\varphi = \exists x_1 \forall x_2 \exists x_3 \cdots \forall x_d : \psi(x_1, \ldots, x_d)$ be a first-order sentence.

We check $\mathcal{A} \models \psi$ on an alternating exponential time machine:

- Guess existentially a $(d, 1)$-ball $(\mathcal{B}_1, b_1)$, which is realizable in $\mathcal{A}$.
- Guess universally a $(d, 2)$-ball $(\mathcal{B}_2, b_1, b_2)$ extending $(\mathcal{B}_1, b_1)$, which is realizable in $\mathcal{A}$.
- Guess existentially a $(d, 3)$-ball $(\mathcal{B}_3, b_1, b_2, b_3)$ extending $(\mathcal{B}_2, b_1, b_2)$, which is realizable in $\mathcal{A}$.

The actual algorithm

Let $\mathcal{A}$ be an automatic structure of bounded degree.

Let $\varphi = \exists x_1 \forall x_2 \exists x_3 \cdots \forall x_d : \psi(x_1, \ldots, x_d)$ be a first-order sentence.

We check $\mathcal{A} \models \psi$ on an alternating exponential time machine:

- Guess existentially a $(d, 1)$-ball $(\mathcal{B}_1, b_1)$, which is realizable in $\mathcal{A}$.
- Guess universally a $(d, 2)$-ball $(\mathcal{B}_2, b_1, b_2)$ extending $(\mathcal{B}_1, b_1)$, which is realizable in $\mathcal{A}$.
- Guess existentially a $(d, 3)$-ball $(\mathcal{B}_3, b_1, b_2, b_3)$ extending $(\mathcal{B}_2, b_1, b_2)$, which is realizable in $\mathcal{A}$.

  $\vdots$
- Guess universally a (d, d)-ball $(\mathcal{B}_d, b_1, b_2, \ldots, b_d)$ extending $(\mathcal{B}_{d-1}, b_1, b_2, \ldots, b_{d-1})$, which is realizable in $\mathcal{A}$.

The actual algorithm

Let $\mathcal{A}$ be an automatic structure of bounded degree.

Let $\varphi = \exists x_1 \forall x_2 \exists x_3 \cdots \forall x_d : \psi(x_1, \ldots, x_d)$ be a first-order sentence.

We check $\mathcal{A} \models \psi$ on an alternating exponential time machine:

- Guess existentially a $(d, 1)$-ball $(\mathcal{B}_1, b_1)$, which is realizable in $\mathcal{A}$.
- Guess universally a $(d, 2)$-ball $(\mathcal{B}_2, b_1, b_2)$ extending $(\mathcal{B}_1, b_1)$, which is realizable in $\mathcal{A}$.
- Guess existentially a $(d, 3)$-ball $(\mathcal{B}_3, b_1, b_2, b_3)$ extending $(\mathcal{B}_2, b_1, b_2)$, which is realizable in $\mathcal{A}$.

  $\vdots$
- Guess universally a (d, d)-ball $(\mathcal{B}_d, b_1, b_2, \ldots, b_d)$ extending $(\mathcal{B}_{d-1}, b_1, b_2, \ldots, b_{d-1})$, which is realizable in $\mathcal{A}$.
- Check, whether $(\mathcal{B}_d, b_1, b_2, \ldots, b_d) \models \psi(b_1, \ldots, b_d)$.

Correctness of this algorithm follows from Gaifman's Theorem.

Estiminating space

Space requirement of the algorithm:

- The algorithm has to store a (d, k)-ball with $1 \leq k \leq d$.

Estiminating space

Space requirement of the algorithm:

- The algorithm has to store a (d, k)-ball with $1 \leq k \leq d$.

  Such a ball contains $2^{2^{O(|\psi|)}}$ many elements.

Estiminating space

Space requirement of the algorithm:

- The algorithm has to store a (d, k)-ball with $1 \leq k \leq d$.

  Such a ball contains $2^{2^{O(|\psi|)}}$ many elements.

- Checking realizability of a (d, k)-ball $(\mathcal{B}_k, b_1, \ldots, b_k)$:

Estiminating space

Space requirement of the algorithm:

- The algorithm has to store a (d, k)-ball with $1 \leq k \leq d$.

  Such a ball contains $2^{2^{O(|\psi|)}}$ many elements.

- Checking realizability of a (d, k)-ball $(\mathcal{B}_k, b_1, \ldots, b_k)$:

  Since $\mathcal{A}$ is automatic, we can build an automaton recognizing all tuples $(a_1, \ldots, a_k)$ such that

$$\mathrm{ball}^{\downarrow}_{\mathcal{A}}(7^d, a_1, \ldots, a_k) \simeq (\mathcal{B}_k, b_1, \ldots, b_k).$$

Estiminating space

Space requirement of the algorithm:

- The algorithm has to store a (d, k)-ball with $1 \leq k \leq d$.

  Such a ball contains $2^{2^{O(|\psi|)}}$ many elements.

- Checking realizability of a (d, k)-ball $(\mathcal{B}_k, b_1, \ldots, b_k)$:

  Since $\mathcal{A}$ is automatic, we can build an automaton recognizing all tuples $(a_1, \ldots, a_k)$ such that

  $$\mathrm{ball}^{\downarrow}_{\mathcal{A}}(7^d, a_1, \ldots, a_k) \simeq (\mathcal{B}_k, b_1, \ldots, b_k).$$

  This automaton has size at most $2^{2^{2^{O(|\psi|)}}}$ but emptiness can be checked in space $2^{2^{O(|\psi|)}}$.

Extensions

$\mathrm{FO} + \exists^{\infty} + \mathrm{MOD}$ denotes the extension of first-order logic with the following quantifiers:

Extensions

FO $+ \exists^\infty +$ MOD denotes the extension of first-order logic with the following quantifiers:

- $\exists^\infty x$: There exist infinitely many x with property ....

Extensions

FO $+ \exists^\infty +$ MOD denotes the extension of first-order logic with the following quantifiers:

- $\exists^\infty x$: There exist infinitely many x with property ....
- $\exists^{(r,q)} x$: There exist finitely many x with ... and the exact number is congruent r modulo q.

Extensions

FO $+ \exists^\infty +$ MOD denotes the extension of first-order logic with the following quantifiers:

- $\exists^\infty x$: There exist infinitely many x with property ....
- $\exists^{(r,q)} x$: There exist finitely many x with ... and the exact number is congruent r modulo q.

Theorem 7 (Blumensath 1999; Khoussainov, Rubin, Stephan 2003)

For every automatic structure $\mathcal{A}$, the (FO $+ \exists^\infty +$ MOD)*-theory is decidable.*

Extensions

$\mathrm{FO} + \exists^\infty + \mathrm{MOD}$ denotes the extension of first-order logic with the following quantifiers:

- $\exists^\infty x$: There exist infinitely many x with property ....
- $\exists^{(r,q)} x$: There exist finitely many x with ... and the exact number is congruent r modulo q.

Theorem 7 (Blumensath 1999; Khoussainov, Rubin, Stephan 2003)

For every automatic structure $\mathcal{A}$, the $(\mathrm{FO} + \exists^\infty + \mathrm{MOD})$*-theory is decidable.*

The proof extends the idea for FO: The quantifiers $\exists^\infty$ and $\exists^{(r,q)}$ preserve regularity.

Extensions

Decidability of the $(\mathrm{FO} + \exists^\infty + \mathrm{MOD})$-theory can be shown for larger classes of structures:

Extensions

Decidability of the (FO $+ \exists^\infty +$ MOD)-theory can be shown for larger classes of structures:

- ω-automatic structures (defined by Büchi automata): Kaiser, Rubin, Bárány 2008

Extensions

Decidability of the (FO $+ \exists^\infty +$ MOD)-theory can be shown for larger classes of structures:

- ω-automatic structures (defined by Büchi automata): Kaiser, Rubin, Bárány 2008
- tree automatic structures (defined by tree automata): Blumensath 1999; Colcombet 2004

Extensions

Decidability of the (FO $+ \exists^\infty +$ MOD)-theory can be shown for larger classes of structures:

- ω-automatic structures (defined by Büchi automata):
  Kaiser, Rubin, Bárány 2008
- tree automatic structures (defined by tree automata):
  Blumensath 1999; Colcombet 2004
- ω-tree automatic structures (defined by Muller tree automata):
  Bárány, Kaiser, Rabinovich 2009

Limitations

Theorem 8

The exists a fixed automatic graph G such that the reachability problem for G is undecidable. (Given, nodes u, v, is there a path from u to v?)

Limitations

Theorem 8

The exists a fixed automatic graph G such that the reachability problem for G is undecidable. (Given, nodes u, v, is there a path from u to v?)

Proof: Take the transition graph of a universal one-tape Turing machine.

Limitations

Theorem 8

The exists a fixed automatic graph G such that the reachability problem for G is undecidable. (Given, nodes u, v, is there a path from u to v?)

Proof: Take the transition graph of a universal one-tape Turing machine.

Theorem 9

For is a fixed automatic structure $\mathcal{A}$ with an undecidable MSO-theory.

Limitations

Theorem 8

The exists a fixed automatic graph G such that the reachability problem for G is undecidable. (Given, nodes u, v, is there a path from u to v?)

Proof: Take the transition graph of a universal one-tape Turing machine.

Theorem 9

For is a fixed automatic structure $\mathcal{A}$ with an undecidable MSO-theory.

First proof: Reachability can be expressed in MSO.

Limitations

Theorem 8

The exists a fixed automatic graph G such that the reachability problem for G is undecidable. (Given, nodes u, v, is there a path from u to v?)

Proof: Take the transition graph of a universal one-tape Turing machine.

Theorem 9

For is a fixed automatic structure $\mathcal{A}$ with an undecidable MSO-theory.

First proof: Reachability can be expressed in MSO.

Second proof: The 2-dimension grid

$$(\mathbb{N} \times \mathbb{N}, \{(x, y), (x + 1, y) \mid x, y \geq 0\}, \{(x, y), (x, y + 1) \mid x, y \geq 0\})$$

is automatic and has an undecidable MSO-theory.

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_n^0 (resp. Π_n^0) if there exists a quantifier-free arithmetical formula $\varphi(x, x_1, \ldots, x_n)$ (with $+$ and $\times$) such that

$$A = \{x \in \mathbb{N} \mid \exists y_1 \forall y_2 \exists y_3 \cdots \forall/\exists y_n : \varphi(x, y_1, \ldots, y_n)\}$$

$$(\text{resp. } A = \{x \in \mathbb{N} \mid \forall y_1 \exists y_2 \forall y_3 \cdots \exists/\forall y_n : \varphi(x, y_1, \ldots, y_n)\})$$

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 $\varphi(x, x_1, \ldots, x_n)$ (with $+$ and $\times$) such that

$$A = \{x \in \mathbb{N} \mid \exists y_1 \forall y_2 \exists y_3 \cdots \forall/\exists y_n : \varphi(x, y_1, \ldots, y_n)\}$$

$$(\text{resp. } A = \{x \in \mathbb{N} \mid \forall y_1 \exists y_2 \forall y_3 \cdots \exists/\forall y_n : \varphi(x, y_1, \ldots, y_n)\})$$

Instead of a quantifier-free arithmetical formula (with $+$ and $\times$) one may take an arbitrary computable predicate $\varphi(x_1, \ldots, x_n)$.

Arithmetical hierarchy

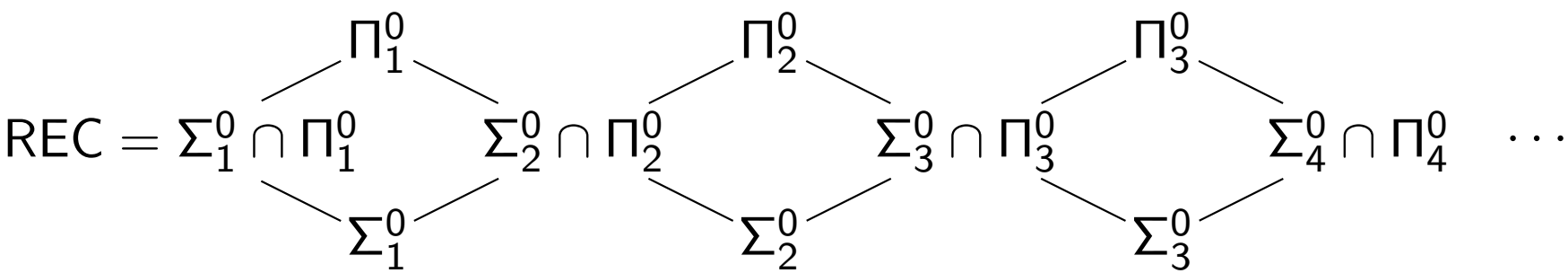
Arithmetical hierarchy (Kleene 1943)

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 $\varphi(x, x_1, \ldots, x_n)$ (with $+$ and $\times$) such that

$$A = \{x \in \mathbb{N} \mid \exists y_1 \forall y_2 \exists y_3 \cdots \forall/\exists y_n : \varphi(x, y_1, \ldots, y_n)\}$$

$$(\text{resp. } A = \{x \in \mathbb{N} \mid \forall y_1 \exists y_2 \forall y_3 \cdots \exists/\forall y_n : \varphi(x, y_1, \ldots, y_n)\})$$

Instead of a quantifier-free arithmetical formula (with $+$ and $\times$) one may take an arbitrary computable predicate $\varphi(x_1, \ldots, x_n)$.



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

$$A = \{x \in \mathbb{N} \mid \exists Y_1 \forall Y_2 \exists Y_3 \cdots \forall/\exists Y_n : \varphi(x, Y_1, \ldots, Y_n)\}$$

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

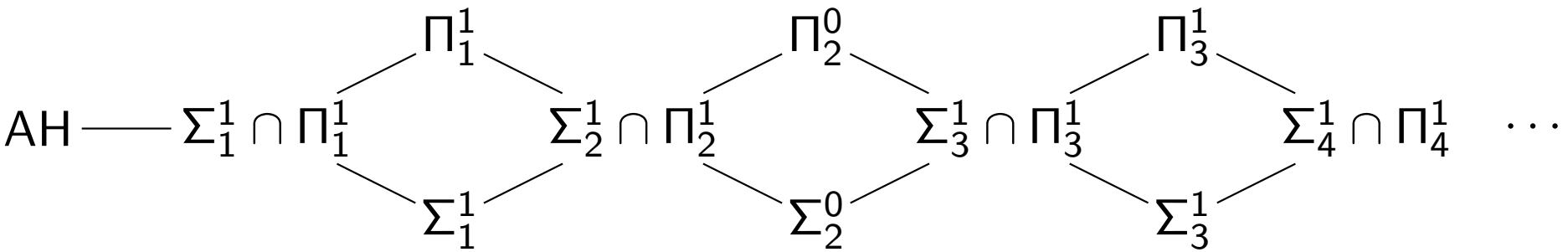
Analytical hierarchy

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

$$A = \{x \in \mathbb{N} \mid \exists Y_1 \forall Y_2 \exists Y_3 \cdots \forall/\exists Y_n : \varphi(x, Y_1, \ldots, Y_n)\}$$

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



Isomorphism problem for computable structures

Theorem 10 (probably Kleene)

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

Isomorphism problem for computable structures

Theorem 10 (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 10 (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 11 (Goncharov, Knight 2002)

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

- *computable linear orders*
  (even computable ordinals),

Isomorphism problem for computable structures

Theorem 10 (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 11 (Goncharov, Knight 2002)

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

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

Isomorphism problem for computable structures

Theorem 10 (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 11 (Goncharov, Knight 2002)

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

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

Isomorphism problem for computable structures

Theorem 10 (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 11 (Goncharov, Knight 2002)

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

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

Isomorphism problem for automatic structures

Theorem 12 (Khoussainov, Nies, Rubin, Stephan 2005)

The isomorphism problem is decidable for

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

Isomorphism problem for automatic structures

Theorem 12 (Khoussainov, Nies, Rubin, Stephan 2005)

The isomorphism problem is decidable for

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

Theorem 13 (Rubin 2004)

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

Isomorphism problem for automatic structures

Theorem 12 (Khoussainov, Nies, Rubin, Stephan 2005)

The isomorphism problem is decidable for

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

Theorem 13 (Rubin 2004)

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

Theorem 14 (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 12 (Khoussainov, Nies, Rubin, Stephan 2005)

The isomorphism problem is decidable for

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

Theorem 13 (Rubin 2004)

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

Theorem 14 (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 problem for automatic structures

Techniques based on configuration graphs of Turing machines do not work for transitive relations (e.g. equivalence relations, linear orders.)

Isomorphism problem for automatic structures

Techniques based on configuration graphs of Turing machines do not work for transitive relations (e.g. equivalence relations, linear orders.)

Theorem 15 (Kuske, Liu, L 2010)

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

Isomorphism problem for automatic structures

Techniques based on configuration graphs of Turing machines do not work for transitive relations (e.g. equivalence relations, linear orders.)

Theorem 15 (Kuske, Liu, L 2010)

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

Techniques based on configuration graphs of Turing machines do not work for transitive relations (e.g. equivalence relations, linear orders.)

Theorem 15 (Kuske, Liu, L 2010)

- *The isomorphism problem for automatic equivalence relations is* Π^0_1*-complete.*
- *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* $\mathsf{FOTh}(\mathbb{N}, +, \times)$*.*

Isomorphism problem for automatic structures

Techniques based on configuration graphs of Turing machines do not work for transitive relations (e.g. equivalence relations, linear orders.)

Theorem 15 (Kuske, Liu, L 2010)

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

Techniques based on configuration graphs of Turing machines do not work for transitive relations (e.g. equivalence relations, linear orders.)

Theorem 15 (Kuske, Liu, L 2010)

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

Automatic equivalence relations

Upper bound

The isomorphism problem for automatic equivalence relations belongs to Π^0_1.

Automatic equivalence relations

Upper bound

The isomorphism problem for automatic equivalence relations belongs to Π^0_1.

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

Automatic equivalence relations

Upper bound

The isomorphism problem for automatic equivalence relations belongs to Π^0_1.

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

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 relations

Upper bound

The isomorphism problem for automatic equivalence relations belongs to Π^0_1.

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

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 relations

Lower bound

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

Automatic equivalence relations

Lower bound

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

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

Automatic equivalence relations

Lower bound

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

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

Theorem 16 (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 relations

Lower bound

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

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

Theorem 16 (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:

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

Corollary

The following problem is Π^0_1-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 $n \in \mathbb{N}$:

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

$$\Longleftrightarrow$$

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

$$\Longleftrightarrow$$

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

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 relations $\mathcal{E}_1$ 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 relations $\mathcal{E}_1$ 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 relations $\mathcal{E}_1$ 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 relations $\mathcal{E}_1$ and $\mathcal{E}_2$ let us write $\mathcal{E}_1 \sim \mathcal{E}_2$ if:

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

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

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

The reduction

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

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

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 relations $\mathcal{E}_1$ and $\mathcal{E}_2$ let us write $\mathcal{E}_1 \sim \mathcal{E}_2$ if:

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

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

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

where

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

Final step

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

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

where

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

The following two observations conclude the proof:

Final step

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

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

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

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

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.
- For two countable equivalence relations $\mathcal{E}_1$ and $\mathcal{E}_2$ we have:

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

Other very difficult decision problems

Theorem 17 (Kuske, L 2009)

The following two problems are Σ^1_1-complete:

Other very difficult decision problems

Theorem 17 (Kuske, L 2009)

The following two problems are Σ^1_1-complete:

- *Does a given automatic (undirected) graph (of bounded degree) has a Hamilton path?*

Other very difficult decision problems

Theorem 17 (Kuske, L 2009)

The following two problems are Σ^1_1-complete:

- *Does a given automatic (undirected) graph (of bounded degree) has a Hamilton path?*
- *Does a given automatic successor tree has an infinite path?*

Other very difficult decision problems

Theorem 17 (Kuske, L 2009)

The following two problems are Σ^1_1-complete:

- *Does a given automatic (undirected) graph (of bounded degree) has a Hamilton path?*
- *Does a given automatic successor tree has an infinite path?*

Remark: It is decidable whether a given automatic order tree has an infinite path (Khoussainov, Rubin, Stephan, 2005).

Other very difficult decision problems

Theorem 17 (Kuske, L 2009)

The following two problems are Σ^1_1-complete:

- *Does a given automatic (undirected) graph (of bounded degree) has a Hamilton path?*
- *Does a given automatic successor tree has an infinite path?*

Remark: It is decidable whether a given automatic order tree has an infinite path (Khoussainov, Rubin, Stephan, 2005).

Theorem 18 (Kuske, Liu, L 2010)

The isomorphism problem for ω-automatic order trees (of finite height) does not belong to the analytical hierarchy.

Complexity of isomorphisms

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

Complexity of isomorphisms

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

Theorem 19

There exist two isomorphic automatic structures $\mathcal{A}_1$ and $\mathcal{A}_2$ such that there exists no hyperarithmetical isomorphism between $\mathcal{A}_1$ and $\mathcal{A}_2$.

Complexity of isomorphisms

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

Theorem 19

There exist two isomorphic automatic structures $\mathcal{A}_1$ and $\mathcal{A}_2$ such that there exists no hyperarithmetical isomorphism between $\mathcal{A}_1$ and $\mathcal{A}_2$.

Theorem 20 (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

- $\mathcal{S}_1 \models \mathcal{A}_1 \cong \mathcal{A}_2$
- $\mathcal{S}_2 \models \mathcal{A}_1 \ncong \mathcal{A}_2$

Open problems

Complexity of FO-theory:

Open problems

Complexity of FO-theory:

- Find other natural classes of automatic structures with elementary first-order theories.

Open problems

Complexity of FO-theory:

- Find other natural classes of automatic structures with elementary first-order theories.

Isomorphism problem:

Open problems

Complexity of FO-theory:

- Find other natural classes of automatic structures with elementary first-order theories.

Isomorphism problem:

- Is the isomorphism problem decidable for automatic ω-words?

Open problems

Complexity of FO-theory:

- Find other natural classes of automatic structures with elementary first-order theories.

Isomorphism problem:

- Is the isomorphism problem decidable for automatic ω-words?
- Is the isomorphism problem decidable for scattered linear orders?
  A linear order L is scattered if $(\mathbb{Q}, \leq)$ cannot be embedded into L.

Open problems

Complexity of FO-theory:

- Find other natural classes of automatic structures with elementary first-order theories.

Isomorphism problem:

- Is the isomorphism problem decidable for automatic ω-words?
- Is the isomorphism problem decidable for scattered linear orders?
  A linear order L is scattered if $(\mathbb{Q}, \leq)$ cannot be embedded into L.
- For which classes of automatic structures is the isomorphism problem Σ^1_1-complete?

Open problems

Complexity of FO-theory:

- Find other natural classes of automatic structures with elementary first-order theories.

Isomorphism problem:

- Is the isomorphism problem decidable for automatic ω-words?
- Is the isomorphism problem decidable for scattered linear orders?
  A linear order L is scattered if $(\mathbb{Q}, \leq)$ cannot be embedded into L.
- For which classes of automatic structures is the isomorphism problem Σ^1_1-complete?

Another problem (Khoussainov, Nerode):

Open problems

Complexity of FO-theory:

- Find other natural classes of automatic structures with elementary first-order theories.

Isomorphism problem:

- Is the isomorphism problem decidable for automatic ω-words?
- Is the isomorphism problem decidable for scattered linear orders?
  A linear order L is scattered if $(\mathbb{Q}, \leq)$ cannot be embedded into L.
- For which classes of automatic structures is the isomorphism problem Σ^1_1-complete?

Another problem (Khoussainov, Nerode):

- Is it Σ^1_1-complete to determine whether a given computable structure is automatic?