Advanced Logic

Markus Lohrey

Universität Siegen

Summer 2023

General comments

Informations concerning the lecture can be found at

`https://www.eti.uni-siegen.de/ti/lehre/sommer_2023/advancedlogic/advancedlogic.html`

- current version of the slides
- links to the videos
- exercise sheets

Literature:

- Schöning: Logik für Informatiker, Spektrum Akademischer Verlag 2013
  (English Edition: Logic for Computer Scientists, Birkhäuser 2008)
- Ebbinghaus, Flum, Thomas: Einführung in the mathematische Logik, Spektrum Akademischer Verlag
  (English Edition: Mathematical Logic, Springer 1994)

The **tutorials** will be organized by Louisa Seelbach.

Recapitulation from the lecture GTI

Definition (recursively enumerable)

A language $L \subseteq \Sigma^*$ is recursively enumerable, if there is an algorithm with the following properties:

For $x \in \Sigma^*$ we have:

- If $x \in L$, then the algorithm terminates with input x.
- If $x \notin L$, then the algorithm does not terminate with input x.

German term: semi-entscheidbar.

Lemma

A language $L \subseteq \Sigma^*$ is recursively enumerable if and only if there is a computable total function $f : \mathbb{N} \to \Sigma^*$ with $L = \{f(i) \mid i \in \mathbb{N}\}$.

Recapitulation from the lecture GTI

Definition (decidable and undecidable)

A language $L \subseteq \Sigma^*$ is decidable, if there is an algorithm with the following properties: for all $x \in \Sigma^*$ we have:

- If $x \in L$, then the algorithm terminates on input x with output "Yes".
- If $x \notin L$, then the algorithm terminates on input x with output "No".

A language $L \subseteq \Sigma^*$ is undecidable, if L is not decidable.

Theorem

A language $L \subseteq \Sigma^*$ is decidable if and only if L and $\Sigma^* \setminus L$ are both recursively enumerable.

Recapitulation from the lecture Logik I

We assume the following notions/definitions from Logik I

- formulas of predicate logic (Formel der Prädikatenlogik)
  Example: $G = \forall x \exists y (P(x, f(y)) \wedge \neg Q(g(z, x)))$
- sentence = formulas without free variable (Aussagen)
  Example: $F = \forall x \exists y (P(x, y) \wedge \neg P(f(x), x))$
- structure (Struktur) $\mathcal{A} = (U_{\mathcal{A}}, I_{\mathcal{A}})$, where $U_{\mathcal{A}}$ is the universe of the structure and $I_{\mathcal{A}}$ is the interpretation function (we write $f^{\mathcal{A}} = I_{\mathcal{A}}(f)$).
  Example: $U_{\mathcal{A}} = \mathbb{N}$, $f^{\mathcal{A}}(n) = n^2$, $P^{\mathcal{A}} = \{(n, m) \mid n < m\}$.
- Structure $\mathcal{A}$ is suitable (passend) for a formula F.
  Example: $\mathcal{A}$ is suitable for F, but not suitable for G.
- $\mathcal{A} \models F$: F evaluates to 1 (= true) in the structure $\mathcal{A}$.
  Whenever we write $\mathcal{A} \models F$, we implicitly assume that $\mathcal{A}$ is suitable for F.

Recapitulation from the lecture Logik I

A formula F of predicate logic is:

- satisfiable, if there is a structure $\mathcal{A}$ such that $\mathcal{A} \models F$ (F is true in the structure $\mathcal{A}$).
- valid, if $\mathcal{A} \models F$ holds for every structure $\mathcal{A}$.

Corollary of Gilmore's theorem

The set of all unsatisfiable formulas of predicate logic is recursively enumerable.

Corollary

The set of all valid formulas of predicate logic is recursively enumerable.

Proof: F is valid if and only if $\neg F$ is unsatisfiable.

Undecidability in predicate logic

We want to prove the following important result:

Church's theorem

The set of valid formulas of predicate logic is undecidable.

Corollary

The set of satisfiable formulas of predicate logic is not recursively enumerable.

Proof: The set of unsatisfiable formulas of predicate logic is recursively enumerable.

If the set of satisfiable formulas of predicate logic would be recursively enumerable, then it would be decidable.

Hence, also the set of unsatisfiable (and hence the set of valid) formulas would be decidable. $\square$

Register machines

We prove Church's theorem by a reduction from the halting problem for register machine programs.

Let $R_1, R_2, \dots$ be (names for) registers.

Intuition: Every register stores a natural number.

A register machine program (RMP for short) P is a sequence of instructions $A_1; A_2; \dots; A_l$, where A_l is the STOP instruction, and for all $1 \le i \le l-1$ the instruction A_i is of one of the following types:

- $R_j := R_j + 1$ for some $1 \le j \le l$
- $R_j := R_j - 1$ for some $1 \le j \le l$
- IF $R_j = 0$ THEN k_1 ELSE k_2 for some $1 \le j, k_1, k_2 \le l$.

Note: We assume that only the registers $R_1, \dots, R_l$ are used in an RMP with l instructions. This is no restriction.

Register machines

A configuration of P is a tuple $(i, n_1, \ldots, n_l) \in \mathbb{N}^{l+1}$ with $1 \leq i \leq l$.

Intuition: i is the number of the instruction that is executed next and n_j is the current content of register R_j.

For configurations $(i, n_1, \ldots, n_l)$ and $(i', n'_1, \ldots, n'_l)$ we write

$$(i, n_1, \ldots, n_l) \to_P (i', n'_1, \ldots, n'_l)$$

if and only if $1 \leq i \leq l-1$ and one of the following cases holds:

- $A_i = (R_j := R_j + 1)$ for some $1 \leq j \leq l$, $i' = i+1$, $n'_j = n_j + 1$, $n'_k = n_k$ for $k \neq j$.
- $A_i = (R_j := R_j - 1)$ for some $1 \leq j \leq l$, $i' = i+1$, $n_j = n'_j = 0$ or $(n_j > 0,\ n'_j = n_j - 1)$, and $n'_k = n_k$ for $k \neq j$.
- $A_i = (\text{IF } R_j = 0 \text{ THEN } k_1 \text{ ELSE } k_2)$ for some $1 \leq j, k_1, k_2 \leq l$, $n'_k = n_k$ for all $1 \leq k \leq l$, $i' = k_1$ if $n_j = 0$, $i' = k_2$ if $n_j > 0$.

Register machines

Example: The following RMP P simulates $R_1 := R_1 + R_2$:

```
1: IF $R_2 = 0$ THEN 5 ELSE 2;
2: $R_1 := R_1 + 1$;
3: $R_2 := R_2 - 1$;
4: IF $R_1 = 0$ THEN 1 ELSE 1;
5: STOP
```

More precisely: For all numbers n_1, n_2 we have

$$(1, n_1, n_2, 0, 0, 0) \rightarrow_P^* (5, n_1 + n_2, 0, 0, 0, 0).$$

Register machines

The configuration is $(1, 0, \ldots, 0)$ (all registers contain 0, first instruction is executed) is also called starting configuration.

We define

$$\begin{aligned} \mathrm{HALT} = \{P \mid {} & P = A_1; A_2; \ldots; A_l \text{ is a RMP with } l \text{ instructions}, \\ & (1, 0, \ldots, 0) \rightarrow_P^* (l, n_1, \ldots, n_l) \text{ for } n_1, \ldots, n_l \geq 0\}. \end{aligned}$$

Register machine programs exactly correspond to GOTO-programs from the GTI lecture.

In GTI we proved that Turing machines and GOTO-programs can simulate each other.

Register machines

Since the halting problem for Turing machines starting with the empty tape (does a given Turing machine finally terminate when it is started with a tape where every cell contains the blank symbol?) is undecidable, we get:

Theorem (undecidability of the halting problem for RMPs)

The set HALT is undecidable.

Remark: HALT is recursively enumerable: Simulate a given RMP on the starting configuration $(1, 0, \dots, 0)$ and stop, if the RMP reaches the STOP-instruction.

Proof of Church's theorem

We prove Church's theorem by constructing from a given RMP P a formula F_P such that:

$$F_P \text{ is valid } \iff P \in \text{HALT}$$

Let $P = A_1; A_2; \ldots; A_l$ be an RMP.

We fix the following symbols:

- $<$: binary predicate symbol
- c: constant
- f, g: unary function symbols
- R: $(l+2)$-ary predicate symbol

Proof of Church's theorem

We define a structure $\mathcal{A}_P$ by case distinction:

Case 1: $P \notin \text{HALT}$:

- Universe $U_{\mathcal{A}_P} = \mathbb{N}$
- $<^{\mathcal{A}_P} = \{(n,m) \mid n < m\}$ (the standard order on $\mathbb{N}$)
- $c^{\mathcal{A}_P} = 0$
- $f^{\mathcal{A}_P}(n) = n+1$, $g^{\mathcal{A}_P}(n+1) = n$, $g^{\mathcal{A}_P}(0) = 0$
- $R^{\mathcal{A}_P} = \{(s, i, n_1, \ldots, n_l) \mid (1, 0, \ldots, 0) \rightarrow_P^s (i, n_1, \ldots, n_l)\}$

Case 2: $P \in \text{HALT}$:

Let t such that $(1, 0, \ldots, 0) \rightarrow_P^t (l, n_1, \ldots, n_l)$ and $e = \max\{t, l\}$.

- Universe $U_{\mathcal{A}_P} = \{0, 1, \ldots, e\}$
- $<^{\mathcal{A}_P} = \{(n,m) \mid n < m\}$ (the standard order on $\{0, 1, \ldots, e\}$)
- $c^{\mathcal{A}_P} = 0$
- $f^{\mathcal{A}_P}(n) = n+1$ for $0 \le n \le e-1$ and $f^{\mathcal{A}_P}(e) = e$.
- $g^{\mathcal{A}_P}(n+1) = n$ for $0 \le n \le e-1$ and $g^{\mathcal{A}_P}(0) = 0$.
- $R^{\mathcal{A}_P} = \{(s, i, n_1, \ldots, n_l) \mid 0 \le s \le t, (1, 0, \ldots, 0) \rightarrow_P^s (i, n_1, \ldots, n_l)\}$

Proof of Church's theorem

In the following we write $\overline{m}$ for the term $f^m(c) = \underbrace{f(f(\cdots f}_{m \text{ many}}(c)\cdots))$.

We define a sentence G_P (in which $<, c, f, g$ and R are used) with the following properties:

(A) $\mathcal{A}_P \models G_P$

(B) For every model $\mathcal{A}$ of G_P the following holds:

If $(1, 0, \dots, 0) \to_P^s (i, n_1, \dots, n_l)$, then:

$$\mathcal{A} \models R(\overline{s}, \overline{i}, \overline{n_1}, \dots, \overline{n_l}) \wedge \bigwedge_{q=0}^{s-1} \overline{q} < \overline{q+1}.$$

We define $G_P = G_0 \wedge R(\overline{0}, \overline{1}, \overline{0}, \dots, \overline{0}) \wedge G_1 \wedge \cdots \wedge G_{l-1}$.

The sentences $G_0, G_1, \dots, G_{l-1}$ are defined as follows.

Proof of Church's theorem

G_0 expresses that

- $<$ is a strict linear order with smallest element c,
- $x \leq f(x)$ and $g(x) \leq x$ for all x,
- for every x, which is not the largest element with respect to $<$, $f(x)$ is the direct successor of x, and
- for every x with $x \neq c$, $g(x)$ is the direct predecessor of x.

$$\begin{aligned}
\forall x, y, z\ (\neg x < x) \wedge (x = y \vee x < y \vee y < x) \wedge ((x < y \wedge y < z) \rightarrow x < z)\\
\wedge\ (x = c \vee c < x)\\
\wedge\ (x = f(x) \vee x < f(x))\\
\wedge\ (x = g(x) \vee g(x) < x)\\
\wedge\ \big(\exists u(x < u) \rightarrow (x < f(x) \wedge \forall u(x < u \rightarrow (u = f(x) \vee f(x) < u)))\big)\\
\wedge\ \big(\exists u(u < x) \rightarrow (g(x) < x \wedge \forall u(u < x \rightarrow (u = g(x) \vee u < g(x))))\big)
\end{aligned}$$

Proof of Church's theorem

Remark: For every model $\mathcal{A}$ of G_0 we have:

- $\mathcal{A} \models g(c) = c$
- $\mathcal{A} \models \forall x\,(\exists u(x < u) \to g(f(x)) = x)$

$\mathcal{A} \models g(c) = c$: We have $g(c) = c \vee c < g(c)$ and $c = g(c) \vee g(c) < c$.

Hence, if $g(c) \neq c$ then we would obtain $c < g(c) \wedge g(c) < c$ and hence $c < c$, which is a contradiction.

$\mathcal{A} \models \forall x\,(\exists u(x < u) \to g(f(x)) = x)$: Assume that $\exists u(x < u)$.

We get $x < f(x) \wedge \forall u(x < u \to (u = f(x) \vee f(x) < u))$.

Thus, $g(f(x)) < f(x) \wedge \forall u(u < f(x) \to (u = g(f(x)) \vee u < g(f(x))))$.

Since $x < f(x)$ we obtain $x = g(f(x)) \vee x < g(f(x))$.

But $x < g(f(x)) < f(x)$ is not possible ($f(x) =$ direct successor of x).

Proof of Church's theorem

Typical models of G_0:

$$\begin{tikzcd}
c \arrow[loop left, "g"] \arrow[r, bend left, "f"] \arrow[r, phantom, "<"] & \bullet \arrow[l, bend left, "g"] \arrow[r, bend left, "f"] \arrow[r, phantom, "<"] & \bullet \arrow[l, bend left, "g"] \arrow[r, bend left, "f"] \arrow[r, phantom, "<"] & \bullet \arrow[l, bend left, "g"] & \cdots
\end{tikzcd}$$

$$\begin{tikzcd}
c \arrow[loop left, "g"] \arrow[r, bend left, "f"] \arrow[r, phantom, "<"] & \bullet \arrow[l, bend left, "g"] \arrow[r, bend left, "f"] \arrow[r, phantom, "<"] & \bullet \arrow[l, bend left, "g"] \arrow[r, bend left, "f"] \arrow[r, phantom, "<"] & \bullet \arrow[l, bend left, "g"] & \cdots & \bullet \arrow[r, bend left, "f"] \arrow[r, phantom, "<"] & \bullet \arrow[l, bend left, "g"] \arrow[loop right, "f"]
\end{tikzcd}$$

In particular: $\mathcal{A}_P$ is a model of G_0.

Proof of Church's theorem

G_i for $1 \leq i \leq l-1$ describes the effect of instruction A_i.

Case 1: $A_i = (R_j := R_j + 1)$. Define

$$G_i = \forall x \forall x_1 \cdots \forall x_l \Big(R(x, \overline{i}, x_1, \ldots, x_l) \rightarrow$$
$$(x < f(x) \wedge R(f(x), \overline{i+1}, x_1, \ldots, x_{j-1}, f(x_j), x_{j+1}, \ldots, x_l)) \Big)$$

Case 2: $A_i = (R_j := R_j - 1)$. Define

$$G_i = \forall x \forall x_1 \cdots \forall x_l \Big(R(x, \overline{i}, x_1, \ldots, x_l) \rightarrow$$
$$(x < f(x) \wedge R(f(x), \overline{i+1}, x_1, \ldots, x_{j-1}, g(x_j), x_{j+1}, \ldots, x_l)) \Big)$$

Proof of Church's theorem

Case 3: $A_i = (\text{IF } R_j = 0 \text{ THEN } k_1 \text{ ELSE } k_2)$ for a $1 \leq j, k_1, k_2 \leq l$.
Define

$$\begin{aligned} G_i \ = \forall x \forall x_1 \cdots \forall x_l \Big(& R(x, \overline{i}, x_1, \ldots, x_l) \ \to \ (x < f(x) \wedge \\ & ((x_j = c \wedge R(f(x), \overline{k_1}, x_1, \ldots, x_l)) \vee \\ & (x_j > c \wedge R(f(x), \overline{k_2}, x_1, \ldots, x_l)))) \Big) \end{aligned}$$

Statement (A) follows directly from the definition of $\mathcal{A}_P$ and G_P:

- $\mathcal{A}_P$ is a model of G_0 (slide 18).
- Since $(1, 0, \ldots, 0) \to_P^0 (1, 0, \ldots, 0)$ we have $(0, 1, 0, \ldots, 0) \in R^{\mathcal{A}_P}$.

Proof of Church's theorem

- To see that $\mathcal{A}_P$ is a model of G_i $(1 \leq i \leq l-1)$, assume that for instance $A_i = (R_j := R_j + 1)$.

  Then for all $s, n_1, \dots n_l \in U_{\mathcal{A}_P}$ with $(1, 0, \dots, 0) \rightarrow_P^s (i, n_1, \dots, n_l)$, i.e., $(s, i, n_1, \dots, n_l) \in R^{\mathcal{A}_P}$, we have:

  - $s+1, i+1, n_j+1 \in U_{\mathcal{A}_P}$,
  - $(1, 0, \dots, 0) \rightarrow_P^{s+1} (i+1, n_1, \dots, n_{j-1}, n_j+1, n_{j+1}, \dots, n_l)$ and thus $(s+1, i+1, n_1, \dots, n_{j-1}, n_j+1, n_{j+1}, \dots, n_l) \in R^{\mathcal{A}_P}$.

Statement (B) is shown by induction on s.

Induction base: $s = 0$. Let $(1, 0, \dots, 0) \rightarrow_P^0 (i, n_1, \dots, n_l)$, i.e., $i = 1$ and $n_1 = n_2 = \cdots = n_l = 0$.

$\mathcal{A} \models G_P$ implies $\mathcal{A} \models R(\overline{0}, \overline{1}, \overline{0}, \dots, \overline{0})$, i.e., $\mathcal{A} \models R(\overline{s}, \overline{i}, \overline{n_1}, \dots, \overline{n_l})$.

Proof of Church's theorem

Induction step: Let $s > 0$ and assume that statement (B) holds for $s - 1$.

Let $(1, 0, \dots, 0) \to_P^s (i, n_1, \dots, n_l)$.

There are $j \le l - 1, m_1, \dots, m_l$ with

$$(1, 0, \dots, 0) \to_P^{s-1} (j, m_1, \dots, m_l) \to_P (i, n_1, \dots, n_l)$$

The induction hypothesis implies

$$\mathcal{A} \models R(\overline{s-1}, \overline{j}, \overline{m_1}, \dots, \overline{m_l}) \wedge \bigwedge_{q=0}^{s-2} \overline{q} < \overline{q+1}.$$

We continue with a case distinction with respect to the instruction A_j. We only consider the case that A_j is of the form $R_k := R_k - 1$.

We then have $i = j + 1$, $n_1 = m_1, \dots, n_{k-1} = m_{k-1}$, $n_{k+1} = m_{k+1}, \dots, n_l = m_l$, ($n_k = m_k = 0$ or $m_k > 0$ and $n_k = m_k - 1$).

Proof of Church's theorem

Because of $\mathcal{A} \models G_j$ we have:

$$\mathcal{A} \models \forall y, y_1, \ldots, y_l \Big(R(y, \overline{j}, y_1, \ldots, y_l) \to \\ \big(y < f(y) \wedge R(f(y), \overline{j+1}, y_1, \ldots, y_{k-1}, g(y_k), y_{k+1}, \ldots, y_l)\big)\Big)$$

Since $\mathcal{A} \models R(\overline{s-1}, \overline{j}, \overline{m_1}, \ldots, \overline{m_l})$, we get

$$\mathcal{A} \models \overline{s-1} < f(\overline{s-1}) \wedge \\ R(f(\overline{s-1}), \overline{j+1}, \overline{m_1}, \ldots, \overline{m_{k-1}}, g(\overline{m_k}), \overline{m_{k+1}}, \ldots, \overline{m_l})$$

i.e., $\mathcal{A} \models \overline{s-1} < \overline{s} \wedge R(\overline{s}, \overline{i}, \overline{n_1}, \ldots, \overline{n_{k-1}}, g(\overline{m_k}), \overline{n_{k+1}}, \ldots, \overline{n_l})$.

Because of $\mathcal{A} \models \overline{s-1} < \overline{s}$, we have

$$\mathcal{A} \models \bigwedge_{q=0}^{s-1} \overline{q} < \overline{q+1}. \tag{1}$$

Proof of Church's theorem

Moreover, $\mathcal{A} \models G_0$ implies $\mathcal{A} \models g(\overline{m_k}) = \overline{n_k}$:

- If $n_k = m_k = 0$ then $\overline{m_k} = \overline{n_k} = c$.

  Since every model of G_0 satisfies $g(c) = c$ (slide 17) we get $\mathcal{A} \models g(\overline{m_k}) = \overline{n_k}$.

- If $m_k > 0$ and $n_k = m_k - 1$ then $\overline{m_k} = f(\overline{n_k})$.

  Since every model of G_0 satisfies $\forall x\, (\exists u (x < u) \rightarrow g(f(x)) = x)$ (slide 17) and $\mathcal{A} \models \overline{n_k} < \overline{n_k + 1} = \overline{m_k}$ by (1), we get

$$\mathcal{A} \models g(\overline{m_k}) = g(f(\overline{n_k})) = \overline{n_k}.$$

Therefore we have $\mathcal{A} \models R(\overline{s}, \overline{i}, \overline{n_1}, \ldots, \overline{n_l})$.

This shows (A) and (B).

Proof of Church's theorem

Proof of Church's theorem:

Define $F_P = (G_P \to \exists x \exists x_1 \cdots \exists x_l R(x, \overline{l}, x_1, \ldots, x_l))$.

Claim: F_P is valid $\iff P \in \mathrm{HALT}$.

If F_P is valid, then $\mathcal{A}_P \models F_P$.

(A) yields $\mathcal{A}_P \models \exists x \exists x_1 \cdots \exists x_l R(x, \overline{l}, x_1, \ldots, x_l)$.

Hence, there are $s, n_1, \ldots, n_l \geq 0$ with $(s, l, n_1, \ldots, n_l) \in R^{\mathcal{A}_P}$.

We obtain $P \in \mathrm{HALT}$.

Now assume that $P \in \mathrm{HALT}$.

Assume that $(1, 0, \ldots, 0) \to_P^s (l, n_1, \ldots, n_l)$.

Let $\mathcal{A}$ be a structure with $\mathcal{A} \models G_P$.

(B) implies $\mathcal{A} \models R(\overline{s}, \overline{l}, \overline{n_1}, \ldots, \overline{n_l})$.

Hence, F_P is valid. $\square$

Trachtenbrot's theorem

A formula F is finitely satisfiable if F has a model with a finite universe. If such a model does not exist then F is called finitely unsatisfiable.

Lemma

The set of finitely satisfiable formulas of predicate logic is recursively enumerable.

Proof:

Let $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \ldots$ be a systematic enumeration of all finite structures (we assume that the interpretation function $I_{\mathcal{A}_i}$ is only defined on those predicate and function symbols that appear in F).

The following algorithm terminates if and only if F is finitely satisfiable:

```
$i := 1$;
while true do
    if $\mathcal{A}_i \models F$ then STOP else $i := i + 1$
end
```

□

Trachtenbrot's theorem

A formula F is finitely valid if every finite structure is a model of F.

Example: The formula

$$\forall x \forall y (f(x) = f(y) \rightarrow x = y) \ \leftrightarrow \ \forall y \exists x (f(x) = y)$$

is finitely valid but not valid.

Trachtenbrot's theorem

The set of finitely satisfiable formulas is undecidable.

Corollary

The set of finitely unsatisfiable formulas and the set of finitely valid formulas are not recursively enumerable.

Trachtenbrot's theorem

Proof of Trachtenbrot's theorem:

We use the construction from the proof of Church's theorem.

Claim: G_P is finitely satisfiable $\iff P \in \mathsf{HALT}$.

(1) Assume that $P \in \mathsf{HALT}$.

Then $\mathcal{A}_P$ is finite and $\mathcal{A}_P \models G_P$ by statement (A).

Hence, G_P is finitely satisfiable.

Trachtenbrot's theorem

(2) Let G_P be finitely satisfiable.

Let $\mathcal{A}$ be a finite structure with $\mathcal{A} \models G_P$.

Assume that $P \notin$ HALT.

Hence, for every $s \geq 0$ there exist $i, n_1, \ldots, n_l$ with $(1, 0, \ldots, 0) \rightarrow_P^s (i, n_1, \ldots, n_l)$.

Statement (B) implies that $\mathcal{A} \models \overline{q} < \overline{q+1}$ for all $q \geq 0$.

Since $<^{\mathcal{A}}$ is a strict linear order (because $\mathcal{A} \models G_0$), the set $\{\mathcal{A}(\bar{i}) \mid i \geq 0\}$ must be infinite, which is a contradiction. $\square$

(Un)decidable theories

Let $\mathcal{A}$ be a structure such that the domain of $I_\mathcal{A}$ is finite and contains no variables.

Let the domain of $I_\mathcal{A}$ consist of $f_1, \ldots, f_n, R_1, \ldots, R_m$.

We identify $\mathcal{A}$ with the tuple $(U_\mathcal{A}, f_1^\mathcal{A}, \ldots, f_n^\mathcal{A}, R_1^\mathcal{A}, \ldots, R_m^\mathcal{A})$ for which we also write $(U_\mathcal{A}, f_1, \ldots, f_n, R_1, \ldots, R_m)$.

Definition

The theory of $\mathcal{A}$ is the set of formulas

$$\mathrm{Th}(\mathcal{A}) = \{F \mid F \text{ is a sentence and } \mathcal{A} \models F\}.$$

We are interested in the question whether a given structure has a decidable theory.

(Un)decidable theories

Theorem

Let $\mathcal{A}$ be an arbitrary structure. Then, $\mathrm{Th}(\mathcal{A})$ *is decidable if and only if* $\mathrm{Th}(\mathcal{A})$ *is recursively enumerable.*

Proof: Let $\mathrm{Th}(\mathcal{A})$ be recursively enumerable and let F be an arbitrary sentence.

We either have $F \in \mathrm{Th}(\mathcal{A})$ or $\neg F \in \mathrm{Th}(\mathcal{A})$.

Therefore we can enumerate $\mathrm{Th}(\mathcal{A})$ until we either produce F or $\neg F$.

Exactly one of the formulas F or $\neg F$ will be produced after a finite number of steps. $\square$

(Un)decidable theories

For the question whether a theory is decidable, we can restrict to so-called relational structures.

A structure $\mathcal{A} = (A, f_1, \dots, f_n, R_1, \dots, R_m)$ is relational if $n = 0$.

For an arbitrary structure $\mathcal{A} = (A, f_1, \dots, f_n, R_1, \dots, R_m)$ we define

$$\mathcal{A}_{\mathrm{rel}} = (A, P_1, \dots, P_n, R_1, \dots, R_m),$$

where $P_i = \{(a_1, \dots, a_k, a) \in A^{k+1} \mid f_i(a_1, \dots, a_k) = a\}$.

Lemma

$\mathrm{Th}(\mathcal{A})$ *is decidable* $\iff$ $\mathrm{Th}(\mathcal{A}_{rel})$ *is decidable.*

Proof: For $\Longleftarrow$ we construct from a sentence F that contains the symbols f_i, R_j a sentence F' that only contains the symbols P_i, R_j and such that:

$$\mathcal{A} \models F \iff \mathcal{A}_{\mathrm{rel}} \models F'$$

(Un)decidable theories

Consider a subformula $R_i(t_1, \ldots, t_k)$ in F, where $t_1, \ldots, t_k$ are terms, and replace it by

$$\exists x_1 \cdots \exists x_k \ (R_i(x_1, \ldots, x_k) \wedge \bigwedge_{i=1}^{k} x_i = t_i).$$

for new variables $x_1, \ldots, x_k$.

We now replace equations $y = f_j(s_1, \ldots, s_l)$ with $l \geq 0$ by

$$\exists y_1 \cdots \exists y_l \ (P_j(y_1, \ldots, y_l, y) \wedge \bigwedge_{i=1}^{l} y_i = s_i)$$

for new variables $y_1, \ldots, y_l$ until only equations of the form $y = y'$ for variables y, y' remain.

The direction $\Longrightarrow$ from the lemma is very easy (Excercise). $\square$

Undecidability of arithmetics

Theorem (Gödel 1931)

$\mathrm{Th}(\mathbb{N}, +, \cdot)$ is undecidable.

Corollary

$\mathrm{Th}(\mathbb{N}, +, \cdot)$ is not recursively enumerable.

For the proof we reduce the set HALT of terminating RMPs to $\mathrm{Th}(\mathbb{N}, +, \cdot)$.

We follow the proof from the book of Ebbinghaus, Flum and Thomas.

In order make the proof less technical we consider $\mathrm{Th}(\mathbb{N}, +, \cdot, s, 0)$ with $s(n) = n + 1$.

Undecidability of arithmetics

We then have: $\mathrm{Th}(\mathbb{N}, +, \cdot, s, 0)$ decidable $\iff$ $\mathrm{Th}(\mathbb{N}, +, \cdot)$ decidable:

- If $\mathrm{Th}(\mathbb{N}, +, \cdot, s, 0)$ is decidable, then clearly $\mathrm{Th}(\mathbb{N}, +, \cdot)$ is decidable.
- Assume that $\mathrm{Th}(\mathbb{N}, +, \cdot)$ is decidable.

  We transform a sentence F that contains $+, \cdot, s, 0$ into a sentence F' that only contains $+, \cdot$ and such that $F \in \mathrm{Th}(\mathbb{N}, +, \cdot, s, 0)$ if and only if $F' \in \mathrm{Th}(\mathbb{N}, +, \cdot)$.

  Step 1: Replace F by

  $$\exists x_0 \, \exists x_1 \, (x_0 + x_0 = x_0 \wedge x_1 \cdot x_1 = x_1 \wedge x_1 \neq x_0 \wedge F)$$

  Step 2: Replace in the resulting sentence every occurrence of the constant 0 by x_0 and every term $s(t)$ by $t + x_1$.

Undecidability of arithmetics

Now assume that $P = A_1; A_2; \cdots; A_l$ is an RMP which uses the registers $R_1, \dots, R_l$.

We construct an arithmetic formula F_P with the free variables $x, x_1, \dots, x_l$ such that for all $1 \le i \le l$ and all $n_1, \dots, n_l \in \mathbb{N}$ the following statements are equivalent:

- $(\mathbb{N}, +, \cdot, s, 0)_{[x/i,\, x_1/n_1, \dots, x_l/n_l]} \models F_P$
- $(1, 0, \dots, 0) \to_P^* (i, n_1, \dots, n_l)$

This implies $P \in \mathrm{HALT} \iff (\mathbb{N}, +, \cdot, s, 0) \models \exists x_1 \cdots \exists x_l \; F_P[x/s^l(0)]$.

Undecidability of arithmetics

Intuitively, F_P expresses the following:

There exists $t \geq 0$ and configurations $C_0, C_1, \ldots, C_t$ with:

- $C_0 = (1, 0, \ldots, 0)$
- $C_t = (x, x_1, \ldots, x_l)$
- $C_i \to_P C_{i+1}$ for all $0 \leq i \leq t-1$

We encode the $(l+1)$-tuples $C_0, C_1, \ldots, C_t$ by an $(t+1) \cdot (l+1)$-tuple.
It remains to express the following, where $k = l + 1$:

There exist $t \geq 0$ and a tuple
$(y_0, y_1, \ldots, y_{k-1},\ y_k, y_{k+1}, \ldots, y_{2k-1}, \ldots, y_{tk}, y_{tk+1}, \ldots, y_{tk+k-1})$ with:

- $y_0 = 1,\ y_1 = 0, \ldots, y_{k-1} = 0$
- $y_{tk} = x,\ y_{tk+1} = x_1, \ldots, y_{tk+k-1} = x_l$
- $(y_{ik}, \ldots, y_{ik+k-1}) \to_P (y_{(i+1)k}, \ldots, y_{(i+1)+k-1})$ for all $0 \leq i \leq t-1$

Undecidability of arithmetics

If one tries to express this with an arithmetic formula, one encounters the problem that one cannot quantify over arbitrary sequences of numbers in predicate logic ($\exists y \exists x_1 \exists x_2 \cdots \exists x_y$ is not allowed).

In order to simulate quantification of sequences of arbitrary length, we need Gödel's β-function.

Lemma

There is a function $\beta : \mathbb{N}^3 \to \mathbb{N}$ with:

- For every sequence $(a_0, \ldots, a_q)$ over $\mathbb{N}$ there exist $p, r \in \mathbb{N}$ such that $\beta(p, r, i) = a_i$ for all $0 \leq i \leq q$.
- There is an arithmetic formula B with free variables v, x, y, z such that for all $p, r, i, a \in \mathbb{N}$ we have:

  $$(\mathbb{N}, +, \cdot, s, 0)_{[v/p,\ x/r,\ y/i,\ z/a]} \models B \iff \beta(p, r, i) = a$$

  One also says that β is arithmetically definable.

Undecidability of arithmetics

Proof of the lemma:

Let $(a_0, \ldots, a_q)$ be an arbitrary sequence over $\mathbb{N}$.

Let p be a prime number with $p > q$ and $p > a_i$ for all i.

Furthermore, define

$$r = 0p^0 + a_0 p^1 + 1p^2 + a_1 p^3 + \cdots + ip^{2i} + a_i p^{2i+1} + \cdots + qp^{2q} + a_q p^{2q+1}.$$

In other words: $(0, a_0, 1, a_1, \ldots, i, a_i, \ldots, q, a_q)$ is the base-p expansion of r (least significant digit on the left).

Note: since p is prime, we have the following for every $x \in \mathbb{N}$:
There exists $m \in \mathbb{N}$ with $x = p^{2m}$ if and only if:

- x is a square ($\exists y : x = y^2$) and
- for all $d \geq 2$ with $d | x$ we have $p | d$.

Here, $x|y$ stands for "x divides y" ($\exists z : x \cdot z = y$).

Undecidability of arithmetics

Claim 1: For all $a \in \mathbb{N}$ and all $0 \le i \le q$ we have: $a = a_i$ if and only if there exist $b_0, b_1, b_2 \in \mathbb{N}$ with:

(a) $r = b_0 + b_1(i + ap + b_2p^2)$

(b) $a < p$

(c) $b_0 < b_1$

(d) b_1 is a square and $p|d$ holds for all $d \ge 2$ with $d|b_1$.
(equivalently: $\exists m : b_1 = p^{2m}$)

$\Longrightarrow$: If $a = a_i$ then we can choose b_0, b_1, b_2 as follows:

$$\begin{aligned} b_0 &= 0p^0 + a_0p^1 + 1p^2 + a_1p^3 + \cdots + (i-1)p^{2i-2} + a_{i-1}p^{2i-1} \\ b_1 &= p^{2i} \\ b_2 &= (i+1) + a_{i+1}p + \cdots + qp^{2(q-i-1)} + a_qp^{2(q-i)-1} \end{aligned}$$

Undecidability of arithmetics

$\Longleftarrow$: Assume that (a)-(d) hold, i.e.,

$$\begin{aligned} r &= b_0 + b_1(i + ap + b_2p^2) \\ &= b_0 + ip^{2m} + ap^{2m+1} + p^{2m+2}b_2. \end{aligned}$$

where $b_0 < b_1 = p^{2m}$, $a < p$ and $i < p$.

Comparing this with

$$r = 0p^0 + a_0p^1 + 1p^2 + a_1p^3 + \cdots + ip^{2i} + a_ip^{2i+1} + \cdots + qp^{2q} + a_qp^{2q+1}$$

and using the uniqueness of the base-p expansion of numbers yields $m = i$ and $a = a_i$.

This shows Claim 1.

Undecidability of arithmetics

We can now define Gödel's β-function:

For all $p, r, i \in \mathbb{N}$ we define $\beta(p, r, i)$ as

(i) the smallest number $a \in \mathbb{N}$ such that there are $b_0, b_1, b_2 \in \mathbb{N}$ with the properties (a)–(d) from Slide 40, respectively

(ii) 0 if numbers $a, b_0, b_1, b_2 \in \mathbb{N}$ with the properties (a)–(d) do not exist.

Remarks:

- The choice of 0 in (ii) is not important (any other number would be also fine).
- Also the choice of the minimum for a in point (i) is not important. It is only important that we select a unique number a having the properties (a)–(d) (one could for instance take the largest number a with these properties).

Undecidability of arithmetics

Claim 2: For every sequence $(a_0, \ldots, a_q)$ over $\mathbb{N}$ there exist $p, r \in \mathbb{N}$ such that $\beta(p, r, i) = a_i$ holds for all $0 \leq i \leq q$.

Let $(a_0, \ldots, a_q)$ be a sequence over $\mathbb{N}$.

Define p and r as on Slide 39.

Take an arbitrary number $0 \leq i \leq q$.

Due to Claim 1 (direction $\Rightarrow$) there are $a, b_0, b_1, b_2 \in \mathbb{N}$ such that (a)–(d) hold (take $a = a_i$ for this).

By definition of the function β there are $b_0, b_1, b_2 \in \mathbb{N}$ such that (a)–(d) also hold with $a = \beta(p, r, i)$.

By Claim 1 (direction $\Leftarrow$) we must have $\beta(p, r, i) = a_i$.

Undecidability of arithmetics

Claim 3: β is arithmetically definable.

All four properties (a)–(d) on Slide 40 are arithmetically definable.

For instance, property (d) can be expressed by the formula

$$\exists x : b_1 = x^2 \wedge \forall x : ((\exists y : s(s(x)) \cdot y = b_1) \to \exists z : (p \cdot z = s(s(x))).$$

Here, $s(s(x))$ stands for the number d in property (d) (the two applications of the successor function s ensure that $s(s(x)) \geq 2$ holds).

With Claims 2 and 3, the proof of the lemma is complete. $\square$

Undecidability of arithmetics

We can now conclude the undecidability proof for arithmetics.

We have to express the following statement (with free variables $x, x_1, \ldots, x_l$) by an arithmetic formula:

There is a number t and a tuple

$$(y_0, y_1, \ldots, y_{k-1},\ y_k, y_{k+1}, \ldots, y_{2k-1}, \ldots, y_{tk}, y_{tk+1}, \ldots, y_{tk+k-1})$$

such that:

- $y_0 = 1,\ y_1 = 0, \ldots, y_{k-1} = 0$
- $y_{tk} = x,\ y_{tk+1} = x_1, \ldots, y_{tk+k-1} = x_l$
- $(y_{ik}, \ldots, y_{ik+k-1}) \rightarrow_P (y_{(i+1)k}, \ldots, y_{(i+1)k+k-1})$ for all $0 \leq i \leq t-1$

Note: $k = l + 1$ is a constant that is determined by the RMP P.

Undecidability of arithmetics

This is equivalent to: there are t, p, r with:

- $\beta(p,r,0) = 1$, $\beta(p,r,1) = 0$, $\ldots$, $\beta(p,r,k-1) = 0$
- $\beta(p,r,tk) = x$, $\beta(p,r,tk+1) = x_1, \ldots, \beta(p,r,tk+k-1) = x_l$
- for all $0 \le i \le t-1$ the following holds:

$$\Big(\beta(p,r,ik), \ldots, \beta(p,r,ik+k-1)\Big) \to_P$$
$$\Big(\beta(p,r,(i+1)k), \ldots, \beta(p,r,(i+1)k+k-1)\Big)$$

It is straightforward to construct an arithmetic formula for

$$(y, y_1, \ldots, y_l) \to_P (z, z_1, \ldots, z_l)$$

as a disjunction over all instructions A_i of the RMP P (excercise). $\square$

Automatic structures

In this section we will introduce automatic structures.

Our main results concerning automatic structures are:

- Every automatic structure has a decidable theory.
- $(\mathbb{N}, +)$ is automatic.
- $(\mathbb{Q}, \leq)$ is automatically presentable.

Convolution of words

Let $n \geq 1$, let Σ be a finite alphabet and let $\# \notin \Sigma$ be a dummy symbol.

Let $\Sigma_\# = \Sigma \cup \{\#\}$ in the following.

For $n \geq 1$ we consider the alphabet $\Sigma_\#^n$ that contains all n-tuples over $\Sigma_\#$.

For words $w_1, w_2 \ldots, w_n \in \Sigma^*$ we define the convolution

$$w_1 \otimes w_2 \otimes \cdots \otimes w_n \in \left(\Sigma_\#^n\right)^*$$

as follows:

- Let $w_i = a_{i,1} a_{i,2} \cdots a_{i,\ell_i}$, (thus, $\ell_i = |w_i|$).
- Let $\ell = \max\{\ell_1, \ldots, \ell_n\}$.
- For all $1 \leq i \leq n$ and $\ell_i < j \leq \ell$ let $a_{i,j} = \#$.
- $w_1 \otimes w_2 \otimes \cdots \otimes w_n := (a_{1,1}, \ldots, a_{n,1})(a_{1,2}, \ldots, a_{n,2}) \cdots (a_{1,\ell}, \ldots, a_{n,\ell})$.

Convolution of words

Using the convolution, we encode an n-tuple $(w_1, w_2 \ldots, w_n)$ of words by the single word $w_1 \otimes w_2 \otimes \cdots \otimes w_n$.

Examples:

$$\begin{aligned} abba \otimes babaaa &= (a,b)(b,a)(b,b)(a,a)(\#,a)(\#,a) \\ abcd \otimes bcdab \otimes a &= (a,b,a)(b,c,\#)(c,d,\#)(d,a,\#)(\#,b,\#) \end{aligned}$$

Note: The tuple $(\#, \#, \ldots, \#)$ does not appear in a convolution.

In particular: $\varepsilon \otimes \varepsilon \otimes \cdots \otimes \varepsilon = \varepsilon$ (multiple convolution of the empty word yields the empty word)

Synchronous multi-tape automata

A synchronous n-tape automaton A over the alphabet Σ is an arbitrary finite automaton over the alphabet $\Sigma_{\#}^{n}$.

Hence, A accepts a language $L(A) \subseteq \left(\Sigma_{\#}^{n}\right)^{*}$.

Note: for an automaton A we denote the accepted language with $L(A)$ whereas in the GTI lecture we used $T(A)$.

The synchronous n-tape automaton accepts the n-ary relation

$$K(A) := \{(w_1, \ldots, w_n) \mid w_1, \ldots, w_n \in \Sigma^*, w_1 \otimes \cdots \otimes w_n \in L(A)\}.$$

An n-ary relation R over Σ^* is synchronously rational if there is a synchronous n-tape automaton A with $K(A) = R$.

Synchronous multi-tape automata

Words in $L(A)$ that do not belong to $\{w_1 \otimes \cdots \otimes w_n \mid w_1, \ldots, w_n \in \Sigma^*\}$ have no influence on the relation $K(A)$ (they are garbage in some sense).

On the other hand, from A one can easily construct a synchronous n-tape automaton B with $L(B) = L(A) \cap \{w_1 \otimes \cdots \otimes w_n \mid w_1, \ldots, w_n \in \Sigma^*\}$.

Note: $\{w_1 \otimes \cdots \otimes w_n \mid w_1, \ldots, w_n \in \Sigma^*\} \subseteq \left(\Sigma_{\#}^{n}\right)^*$ is regular.

Illustration of a synchronous 2-tape automaton:

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

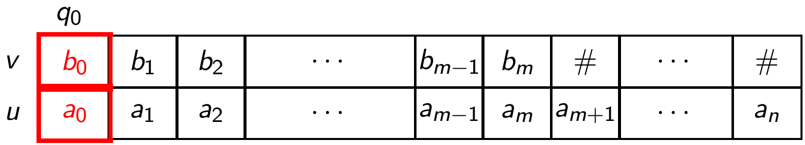
Synchronous multi-tape automata

Words in $L(A)$ that do not belong to $\{w_1 \otimes \cdots \otimes w_n \mid w_1, \ldots, w_n \in \Sigma^*\}$ have no influence on the relation $K(A)$ (they are garbage in some sense).

On the other hand, from A one can easily construct a synchronous n-tape automaton B with $L(B) = L(A) \cap \{w_1 \otimes \cdots \otimes w_n \mid w_1, \ldots, w_n \in \Sigma^*\}$.

Note: $\{w_1 \otimes \cdots \otimes w_n \mid w_1, \ldots, w_n \in \Sigma^*\} \subseteq \left(\Sigma_\#^n\right)^*$ is regular.

Illustration of a synchronous 2-tape automaton:


q_0
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

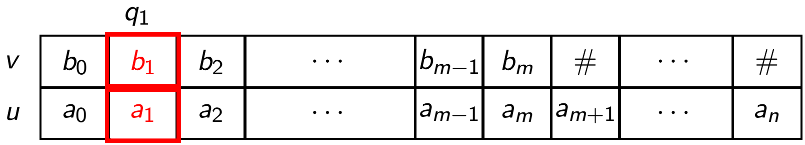
Synchronous multi-tape automata

Words in $L(A)$ that do not belong to $\{w_1 \otimes \cdots \otimes w_n \mid w_1, \ldots, w_n \in \Sigma^*\}$ have no influence on the relation $K(A)$ (they are garbage in some sense).

On the other hand, from A one can easily construct a synchronous n-tape automaton B with $L(B) = L(A) \cap \{w_1 \otimes \cdots \otimes w_n \mid w_1, \ldots, w_n \in \Sigma^*\}$.

Note: $\{w_1 \otimes \cdots \otimes w_n \mid w_1, \ldots, w_n \in \Sigma^*\} \subseteq \left(\Sigma_{\#}^n\right)^*$ is regular.

Illustration of a synchronous 2-tape automaton:


q_1
v
u
b_0 b_1 b_2 $\cdots$ b_{m-1} b_m $\#$ $\cdots$ $\#$
a_0 a_1 a_2 $\cdots$ a_{m-1} a_m a_{m+1} $\cdots$ a_n

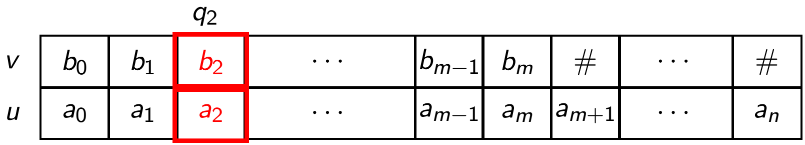
Synchronous multi-tape automata

Words in $L(A)$ that do not belong to $\{w_1 \otimes \cdots \otimes w_n \mid w_1, \ldots, w_n \in \Sigma^*\}$ have no influence on the relation $K(A)$ (they are garbage in some sense).

On the other hand, from A one can easily construct a synchronous n-tape automaton B with $L(B) = L(A) \cap \{w_1 \otimes \cdots \otimes w_n \mid w_1, \ldots, w_n \in \Sigma^*\}$.

Note: $\{w_1 \otimes \cdots \otimes w_n \mid w_1, \ldots, w_n \in \Sigma^*\} \subseteq \left(\Sigma_{\#}^n\right)^*$ is regular.

Illustration of a synchronous 2-tape automaton:



q_2
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

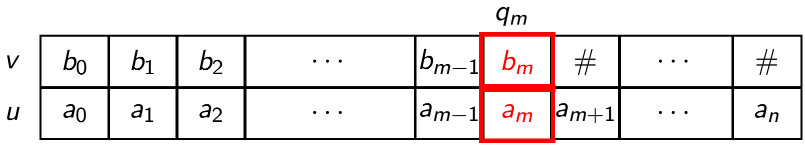
Synchronous multi-tape automata

Words in $L(A)$ that do not belong to $\{w_1 \otimes \cdots \otimes w_n \mid w_1, \ldots, w_n \in \Sigma^*\}$ have no influence on the relation $K(A)$ (they are garbage in some sense).

On the other hand, from A one can easily construct a synchronous n-tape automaton B with $L(B) = L(A) \cap \{w_1 \otimes \cdots \otimes w_n \mid w_1, \ldots, w_n \in \Sigma^*\}$.

Note: $\{w_1 \otimes \cdots \otimes w_n \mid w_1, \ldots, w_n \in \Sigma^*\} \subseteq \left(\Sigma_\#^n\right)^*$ is regular.

Illustration of a synchronous 2-tape automaton:



q_m
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

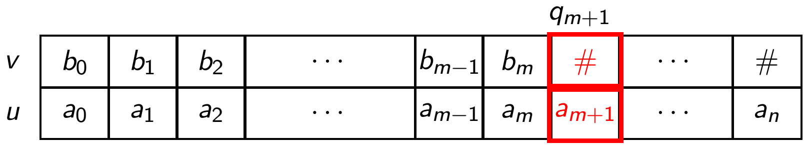
Synchronous multi-tape automata

Words in $L(A)$ that do not belong to $\{w_1 \otimes \cdots \otimes w_n \mid w_1, \ldots, w_n \in \Sigma^*\}$ have no influence on the relation $K(A)$ (they are garbage in some sense).

On the other hand, from A one can easily construct a synchronous n-tape automaton B with $L(B) = L(A) \cap \{w_1 \otimes \cdots \otimes w_n \mid w_1, \ldots, w_n \in \Sigma^*\}$.

Note: $\{w_1 \otimes \cdots \otimes w_n \mid w_1, \ldots, w_n \in \Sigma^*\} \subseteq \left(\Sigma^n_\#\right)^*$ is regular.

Illustration of a synchronous 2-tape automaton:



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

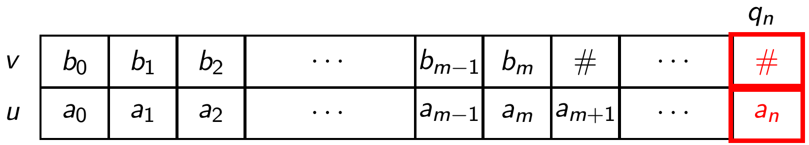
Synchronous multi-tape automata

Words in $L(A)$ that do not belong to $\{w_1 \otimes \cdots \otimes w_n \mid w_1, \ldots, w_n \in \Sigma^*\}$ have no influence on the relation $K(A)$ (they are garbage in some sense).

On the other hand, from A one can easily construct a synchronous n-tape automaton B with $L(B) = L(A) \cap \{w_1 \otimes \cdots \otimes w_n \mid w_1, \ldots, w_n \in \Sigma^*\}$.

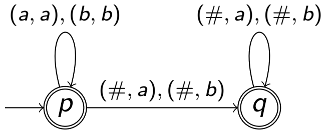
Note: $\{w_1 \otimes \cdots \otimes w_n \mid w_1, \ldots, w_n \in \Sigma^*\} \subseteq \left(\Sigma_\#^n\right)^*$ is regular.

Illustration of a synchronous 2-tape automaton:


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

Synchronous multi-tape automata

Example: Let A be the following synchronous 2-tape automaton:



We have $K(A) = \{(u,v) \mid u,v \in \{a,b\}^*, \exists w \in \{a,b\}^* : v = uw\}$ (the prefix relation).

On the other hand, the suffix relation $\{(u,v) \mid \exists w \in \{a,b\}^* : v = wu\}$ is not synchronously rational.

Automatic structures

Definition

A relational structure $\mathcal{A} = (A, R_1, \ldots, R_m)$ (with R_i an n_i-ary relation) is automatic if there exist a finite alphabet Σ, a finite automaton B over the alphabet Σ, and synchronous n_i-tape automata B_i over the alphabet Σ $(1 \leq i \leq m)$ such that:

- $L(B) = A$
- $K(B_i) = R_i$ for $1 \leq i \leq m$

Definition

A structure $\mathcal{A}$ is automatically presentable if $\mathcal{A}$ is isomorphic to an automatic structure.

Automatic structures

Excursion: isomorphic structures

Let $\mathcal{A} = (A, R_1, \ldots, R_m)$ and $\mathcal{B} = (B, P_1, \ldots, P_m)$ be relational structures, where R_i and P_i are both n_i-ary (for all $1 \leq i \leq m$).

We say that $\mathcal{A}$ and $\mathcal{B}$ are isomorphic if there is a bijection $h : A \to B$ such that for all $1 \leq i \leq m$ and all tuples $(a_1, \ldots, a_{n_i}) \in A^{n_i}$ we have:

$$(a_1, \ldots, a_{n_i}) \in R_i \iff (h(a_1), \ldots, h(a_{n_i})) \in P_i.$$

Intuitively: $\mathcal{B}$ can be obtained from $\mathcal{A}$ by renaming the elements from the universe of $\mathcal{A}$.

If $\mathcal{A}$ and $\mathcal{B}$ are isomorph then $\mathrm{Th}(\mathcal{A})$ is decidable if and only if $\mathrm{Th}(\mathcal{B})$ is decidable (predicate logic cannot refer to the names of elements in the universe).

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

Theorem

$(\mathbb{N}, +)$ with $+ = \{(a, b, c) \mid a + b = c\}$ is automatically presentable.

Proof: Let A be a finite automaton with $L(A) = \{0\} \cup \{0, 1\}^*1$.

Then, the following function $h : L(A) \to \mathbb{N}$ is a bijection:

$$\begin{aligned} h(0) &= 0 \\ h(a_0 a_1 \cdots a_{n-1} 1) &= \sum_{i=0}^{n-1} a_i 2^i + 2^n \end{aligned}$$

Let B_+ be the synchronous 3-tape automaton from the next slide.

B_+ "almost" recognizes the relation

$$\{(u, v, w) \in L(A)^3 \mid h(u) + h(v) = h(w)\}.$$

We have for instance $(00, 0000, 0000) \in K(B_+)$.

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

$$\begin{tikzcd}[column sep=huge, row sep=huge]
 & q_1 \arrow[loop above, "{(\#,0,0),(\#,1,1)}"] & q_4 \arrow[l, "{(\#,0,1)}"'] \arrow[loop above, "{(\#,1,0)}"] \arrow[dr, bend left, "{(\#,\#,1)}"] & \\
{} \arrow[r] & q_0 \arrow[loop left, "{(1,0,1)\\(0,0,0)\\(0,1,1)}"] \arrow[u, "{(\#,0,0)\\(\#,1,1)}"] \arrow[r, bend left, "{(1,1,0)}"] \arrow[d, "{(0,\#,0)\\(1,\#,1)}"'] & q_3 \arrow[l, bend left, "{(0,0,1)}"] \arrow[ul, "{(\#,0,1)}"'] \arrow[u, "{(\#,1,0)}"] \arrow[loop above, "{(1,0,0)\\(0,1,0)\\(1,1,1)}"] \arrow[r, "{(\#,\#,1)}"'] \arrow[dl, "{(0,\#,1)}"] \arrow[d, "{(1,\#,0)}"'] & q_f \\
 & q_2 \arrow[loop below, "{(0,\#,0),(1,\#,1)}"] & q_5 \arrow[l, "{(0,\#,1)}"] \arrow[loop below, "{(1,\#,0)}"] \arrow[ur, bend right, "{(\#,\#,1)}"'] &
\end{tikzcd}$$

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

Let A_+ be a synchronous 3-tape automaton with

$$L(A_+) = L(B_+) \cap \{u \otimes v \otimes w \mid u, v, w \in L(A)\}.$$

We then get $K(A_+) = \{(u, v, w) \in L(A)^3 \mid h(u) + h(v) = h(w)\}$. $\square$

Intuition: The automaton from Slide 56 checks with the school method for addition whether the number on tape 3 is the sum of the numbers on tapes 1 and 2.

For this, the automaton stores the current carry in its state.

States q_0, q_1, q_2 correspond to carry 0 whereas states q_3, q_4, q_5 correspond to carry 1.

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

Three states are needed since the numbers on tapes 1 and 2 may have a different bit lengths.

States q_1, q_4 (q_2, q_5) are needed for the situation where the number on tape 1 (2) is shorter than the number on tape 2 (1).

State q_f is a failure state.

One can slightly extend the theorem on Slide 55: For every $p > 1$ the structure $(\mathbb{N}, +, |_p)$ with

$$x \mid_p y \iff \exists n, k \in \mathbb{N} : x = p^n \wedge y = k \cdot x$$

is automatically presentable.

Linear orders

Our second example for an automatic structure is a linear order.

Recall (from the lecture DMI): a linear order is a structure (A, R), where R is a binary relation with the following properties:

- $\forall a \in A : (a, a) \in R$ (R is reflexive)
- $\forall a, b \in A : (a, b) \in R \wedge (b, a) \in R \rightarrow a = b$ (R is anti-symmetric)
- $\forall a, b, c \in A : (a, b) \in R \wedge (b, c) \in R \rightarrow (a, c) \in R$ (R is transitive)
- $\forall a, b \in A : (a, b) \in R \vee (b, a) \in R$ (R is linear)

Instead of R we denote the binary relation of a linear order always with $\leq$ (possibly with an index).

An element $a \in A$ is a smallest (resp., largest) element of the linear order $(A, \leq)$ if: $\forall b \in A : a \leq b$ (resp., $\forall b \in A : b \leq a$).

Linear orders

Theorem

The linear order $(\mathbb{Q}, \leq)$ (where $\leq$ is the standard order on $\mathbb{Q}$) is automatically presentable.

For the proof we use a famous theorem of Cantor.

It uses another property of linear orders (we write $x < y$ for $x \leq y \wedge x \neq y$): A linear order $(A, \leq)$ is dense if:

$$\forall x \forall y (x < y \to \exists z (x < z < y)).$$

Intuitively: between two different elements of A there is always a third element.

Cantor's theorem

Let $(A, \leq_A)$ and $(B, \leq_B)$ be countable dense linear orders without a smallest and largest element. Then $(A, \leq_A)$ and $(B, \leq_B)$ are isomorphic.

Cantor's theorem

Proof of Cantor's theorem:

We construct enumerations

$$a_1, a_2, a_3, a_4, \ldots \text{ and } b_1, b_2, b_3, b_4, \ldots$$

with the following properties:

- $a_i \neq a_j$ and $b_i \neq b_j$ for $i \neq j$
- $A = \{a_i \mid i \geq 1\}$ and $B = \{b_i \mid i \geq 1\}$
- $a_i < a_j$ if and only if $b_i < b_j$ for all i, j.

Then, $f : A \to B$ with $f(a_i) = b_i$ is an isomorphism.

Since A and B are countable and infinite, we can enumerate these sets:

$$A = \{x_1, x_2, x_3, \ldots\} \text{ and } B = \{y_1, y_2, y_3, \ldots\}$$

The following "algorithm" constructs enumerations with the above properties:

Cantor's theorem

```
$L_A := [x_1, x_2, x_3, \dots]$;  $L_B := [y_1, y_2, y_3, \dots]$
for all $i \geq 1$ do                    ($a_1, \dots a_{i-1}$, $b_1, \dots b_{i-1}$ are already defined)
    if $i$ is odd then
        let $x$ be the first element from $L_A$
        remove $x$ from the list $L_A$
        let $y$ be an element from $L_B$ with the following properties:
        $\forall 1 \leq j \leq i-1 : a_j < x \longleftrightarrow b_j < y$                    (†)
        remove $y$ from the list $L_B$
        $a_i := x$; $b_i := y$
    else
        let $y$ be the first element from $L_B$
        remove $y$ from the list $L_B$
        let $x$ be an element from $L_A$ with the following properties:
        $\forall 1 \leq j \leq i-1 : a_j < x \longleftrightarrow b_j < y$                    (‡)
        remove $x$ from the list $L_A$
        $a_i := x$; $b_i := y$
endfor
```

Cantor's theorem

Remarks:

- The element y with the property $(\dagger)$ exists, since $(B, \leq_B)$ is dense and neither has a smallest nor largest element.

  This ensures that we find for x an element y that has the same position relative to $b_1, \ldots, b_{i-1}$ as x to $a_1, \ldots, a_{i-1}$.

  For the same reason, the element x with the property $(\ddagger)$ exists.

- Since the correspondence $a_i \mapsto b_i$ must be bijective, we have to pair every element from the list L_A with exactly one element from the list L_B. Thereby, we also have to ensure that every element from the list L_B is paired.

  This will be enforced by the case distinction between i odd and i even. $\square$

Cantor's theorem

Illustration of the proof of Cantor's theorem:

Cantor's theorem

Illustration of the proof of Cantor's theorem:

$$L_A := [x_1, x_2, x_3, x_4, x_5, x_6, x_7, \ldots];\ \ L_B := [y_1, y_2, y_3, y_4, y_5, y_6, y_7, \ldots]$$

Cantor's theorem

Illustration of the proof of Cantor's theorem:

$$L_A := [x_1, x_2, x_3, x_4, x_5, x_6, x_7, \ldots];\ L_B := [y_1, y_2, y_3, y_4, y_5, y_6, y_7, \ldots]$$

$$x_1$$

Cantor's theorem

Illustration of the proof of Cantor's theorem:

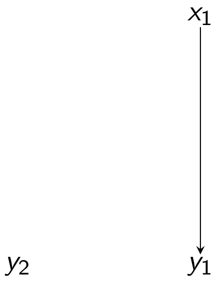
$$L_A := [x_1, x_2, x_3, x_4, x_5, x_6, x_7, \ldots];\ L_B := [y_1, y_2, y_3, y_4, y_5, y_6, y_7, \ldots]$$

$$\begin{tikzcd} x_1 \arrow[d] \\ y_1 \end{tikzcd}$$

Cantor's theorem

Illustration of the proof of Cantor's theorem:

$$L_A := [x_1, x_2, x_3, x_4, x_5, x_6, x_7, \ldots];\ L_B := [y_1, y_2, y_3, y_4, y_5, y_6, y_7, \ldots]$$


$$\begin{tikzcd} & x_1 \arrow[d] \\ y_2 & y_1 \end{tikzcd}$$

Cantor's theorem

Illustration of the proof of Cantor's theorem:

$$L_A := [x_1, x_2, x_3, x_4, x_5, x_6, x_7, \ldots]; \; L_B := [y_1, y_2, y_3, y_4, y_5, y_6, y_7, \ldots]$$

$$\begin{tikzcd}
x_6 \arrow[d] & x_1 \arrow[d] \\
y_2 & y_1
\end{tikzcd}$$

Cantor’s theorem

Illustration of the proof of Cantor’s theorem:

$$L_A := [x_1, x_2, x_3, x_4, x_5, x_6, x_7, \ldots];\ \ L_B := [y_1, y_2, y_3, y_4, y_5, y_6, y_7, \ldots]$$

$$\begin{tikzcd}
x_6 \arrow[d] & x_2 & x_1 \arrow[d] \\
y_2 & & y_1
\end{tikzcd}$$

Cantor's theorem

Illustration of the proof of Cantor's theorem:

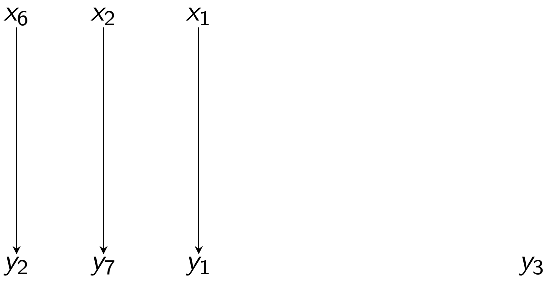
$$L_A := [x_1, x_2, x_3, x_4, x_5, x_6, x_7, \ldots];\ \ L_B := [y_1, y_2, y_3, y_4, y_5, y_6, y_7, \ldots]$$

$$\begin{tikzcd}
x_6 \arrow[d] & x_2 \arrow[d] & x_1 \arrow[d] \\
y_2 & y_7 & y_1
\end{tikzcd}$$

Cantor's theorem

Illustration of the proof of Cantor's theorem:

$$L_A := [x_1, x_2, x_3, x_4, x_5, x_6, x_7, \ldots];\ L_B := [y_1, y_2, y_3, y_4, y_5, y_6, y_7, \ldots]$$



$$\begin{tikzcd} x_6 \arrow[d] & x_2 \arrow[d] & x_1 \arrow[d] & \\ y_2 & y_7 & y_1 & y_3 \end{tikzcd}$$

Cantor's theorem

Illustration of the proof of Cantor's theorem:

$$L_A := [x_1, x_2, x_3, x_4, x_5, x_6, x_7, \ldots]; \; L_B := [y_1, y_2, y_3, y_4, y_5, y_6, y_7, \ldots]$$

$$\begin{tikzcd}
x_6 \arrow[d] & x_2 \arrow[d] & x_1 \arrow[d] & & & x_7 \arrow[d] \\
y_2 & y_7 & y_1 & & & y_3
\end{tikzcd}$$

Cantor's theorem

Illustration of the proof of Cantor's theorem:

$$L_A := [x_1, x_2, x_3, x_4, x_5, x_6, x_7, \ldots];\ L_B := [y_1, y_2, y_3, y_4, y_5, y_6, y_7, \ldots]$$

$$\begin{tikzcd}
x_3 & x_6 \arrow[d] & x_2 \arrow[d] & x_1 \arrow[d] & & x_7 \arrow[d] \\
 & y_2 & y_7 & y_1 & & y_3
\end{tikzcd}$$

Cantor's theorem

Illustration of the proof of Cantor's theorem:

$$L_A := [x_1, x_2, x_3, x_4, x_5, x_6, x_7, \ldots];\ L_B := [y_1, y_2, y_3, y_4, y_5, y_6, y_7, \ldots]$$

$$\begin{tikzcd}
x_3 \arrow[d] & x_6 \arrow[d] & x_2 \arrow[d] & x_1 \arrow[d] & x_7 \arrow[d] \\
y_6 & y_2 & y_7 & y_1 & y_3
\end{tikzcd}$$

$(\mathbb{Q}, \leq)$ is automatically presentable

Proof that $(\mathbb{Q}, \leq)$ is automatic:

Due to Cantor's theorem, it suffices to construct a countable dense automatic linear order which neither has a smallest nor a largest element.

Let $L = \{0,1\}^*1$.

Let $\leq$ be the lexicographic order on L. That means, for $x, y \in L$ we have $x \leq y$ if and only if one of the following cases holds:

- There exists $u \in \{0,1\}^*$ with $y = xu$ (x is a prefix of y)
- There exist $z, u, v \in \{0,1\}^*$ with $x = z0u$ and $y = z1v$.

Then $(L, \leq)$ is a linear order (easy to check).

- $(L, \leq)$ has no largest element:

  Let $x \in L$ be arbitrary. Then $x < x1 \in L$.

$(\mathbb{Q}, \leq)$ is automatically presentable

- $(L, \leq)$ has no smallest element:

  Let $x = u1 \in L$ be arbitrary. Then $u01 < u1 = x$

- $(L, \leq)$ is dense:

  Let $x, y \in L$ with $x < y$ be arbitrary.

  Case 1: $x = u1$, $y = u1v1$:

  Then we have $x = u1 < u10^{|v|+1}1 < u1v1 = y$.

  Case 2: $x = u0v1$, $y = u1w$:

  Then we have $x = u0v1 < u01^{|v|+2} < u1w = y$.

- $(L, \leq)$ is automatic: easy excercise $\square$

Structures that are not automatically presentable

For the following structures one can show that they are not automatically presentable:

- $(\mathbb{R}, +)$ (because every automatic structure is countable)
- every structure with an undecidable theory (see next slide).

  Examples for this:
  - $(\mathbb{N}, +, \cdot)$ (Gödel's theorem)
  - $(\Sigma^*, \circ)$ (the free monoid over Σ) for $|\Sigma| > 1$ (Quine 1946)
- $(\mathbb{N}, \cdot)$ and $(\mathbb{N}, |)$
- $(\mathbb{Q}, +)$ (Tsankov 2009)

Theory of an automatic structure

Our main result on automatic structures is:

Theorem (Khoussainov, Nerode 1994)

For every automatically presentable structure $\mathcal{A}$, $\mathsf{Th}(\mathcal{A})$ is decidable.

Corollary (Presburger 1929)

$\mathsf{Th}(\mathbb{N}, +)$ is decidable.

Corollary

$\mathsf{Th}(\mathbb{Q}, \leq)$ is decidable.

Theory of an automatic structure

For the proof of the theorem of Khoussainov and Nerode we need some facts on regular languages.

From GTI we know that the regular languages are closed under all boolean operations (complement, union, intersection).

Moreover: From finite automata A and B over an input alphabet Γ one can construct finite automata for the languages $\Gamma^* \setminus L(A)$, $L(A) \cap L(B)$ and $L(A) \cup L(B)$.

We need two further closure properties for the regular languages.

Theory of an automatic structure

A homomorphism is a function $h : \Gamma^* \to \Sigma^*$ such that:

- Γ and Σ are finite alphabets.
- $h(\varepsilon) = \varepsilon$ (the empty word is mapped to the empty word)
- For all words $u, v \in \Gamma^*$ we have $h(uv) = h(u)h(v)$.

In particular, for every word $u = a_1 a_2 \cdots a_n$ $(a_1, \ldots, a_n \in \Gamma)$:

$$h(a_1 a_2 \cdots a_n) = h(a_1)h(a_2) \cdots h(a_n).$$

In order to specify a homomorphism $h : \Gamma^* \to \Sigma^*$, it suffices to specify all words $h(a)$ for $a \in \Gamma$.

Example: Let $h : \{a, b\}^* \to \{b, c\}^*$ be the homomorphism with $h(a) = bcc$ and $h(b) = cbc$.

Then we have $h(abba) = bcc\,cbc\,cbc\,bcc$.

Theory of an automatic structure

Lemma (closure regular languages under homomorphisms)

From a finite automaton A with input alphabet Γ and a homomorphism $h : \Gamma^* \to \Sigma^*$ one can construct a finite automaton B with

$$L(B) = h(L(A)) = \{h(w) \mid w \in L(A)\}.$$

Proof: Every transition $p \xrightarrow{a} q$ in the automaton A with $a \in \Gamma$ and $h(a) = b_1 b_2 \cdots b_n$ ($b_1, \dots, b_n \in \Sigma$) is replaced by the sequence of transitions

$$p \xrightarrow{b_1} r_1 \xrightarrow{b_2} r_2 \cdots \xrightarrow{b_{n-1}} r_{n-1} \xrightarrow{b_n} q.$$

Here, $r_1, \dots, r_{n-1}$ are new states that do not appear in other transitions.

For all $v \in \Sigma^*$ we have:

$$v \in L(B) \iff \exists w \in L(A) : v = h(w) \iff v \in h(L(A)).$$

Theory of an automatic structure

But: What happens when if $n = 0$ and hence $h(a) = \varepsilon$ holds?

Then we replace the transition $p \xrightarrow{a} q$ by the ε-transition $p \xrightarrow{\varepsilon} q$.

ε-transitions do not increase the power of finite automata:

From a finite automaton with ε-transitions one can construct an equivalent finite automaton without ε-transitions.

See e.g. slides 72 & 73 from the GTI lecture in summer semester 2020 (`https://www.eti.uni-siegen.de/ti/lehre/ss20/gti/folien.pdf`) or slides 31 & 32 from the Compiler Construction lecture in summer semester 2020 (`https://www.eti.uni-siegen.de/ti/lehre/ss20/compilerbau/cb.pdf`). $\square$

Theory of an automatic structure

Lemma (closure regular languages under inverse homomorphisms)

From a finite automaton B over the input alphabet Σ and a homomorphism $h : \Gamma^* \to \Sigma^*$ one can construct a finite automaton A with

$$L(A) = h^{-1}(L(B)) = \{w \in \Gamma^* \mid h(w) \in L(B)\}.$$

Proof: The automaton A has the same set of states and the same initial/final states as B.

In the automaton A there is a transition $p \xrightarrow{a} q$ if and only if in the automaton B one can go with the word $h(a) \in \Sigma^*$ from state p to state q.

For all $w \in \Gamma^*$ we have:

$$w \in L(A) \iff h(w) \in L(B) \iff w \in h^{-1}(L(B)).$$

$\square$

Theory of an automatic structure

Now we come to the

Proof of the theorem of Khoussainov and Nerode:

Let $\mathcal{A} = (L, R_1, \ldots, R_m)$ be an automatic structure with $L \subseteq \Sigma^*$.

Goal: For every formula F which contains only free variables from the set $\{x_1, \ldots, x_n\}$ (not all variables $x_1, \ldots, x_n$ have to be free in F) we construct by induction on the structure of F a synchronous n-tape automaton B_F such that

$$K(B_F) = \{(w_1, \ldots, w_n) \in L^n \mid \mathcal{A}_{[x_1/w_1]\cdots[x_n/w_n]} \models F\}.$$

Theory of an automatic structure

Case 1: $F = R_i(x_{i_1}, \dots, x_{i_k})$, where $1 \leq i_1, \dots, i_k \leq n$:

Define the homomorphism $f : \left(\Sigma_{\#}^{n}\right)^{*} \to \left(\Sigma_{\#}^{k}\right)^{*}$ as follows, where $a_1, \dots, a_n \in \Sigma_{\#}$:

$$f(a_1, \dots, a_n) = \begin{cases} \varepsilon & \text{if } a_{i_1} = \cdots = a_{i_k} = \# \\ (a_{i_1}, \dots, a_{i_k}) & \text{otherwise} \end{cases}$$

Note: $f(w_1 \otimes \cdots \otimes w_n) = w_{i_1} \otimes \cdots \otimes w_{i_k}$ for all $w_1, \dots, w_n \in \Sigma^*$.

Let B_i be the synchronous k-tape automaton for R_i.

From B_i we construct, using the lemma from slide 73, an n-tape automaton B_F with

$$L(B_F) = f^{-1}(L(B_i)) \cap \{w_1 \otimes \cdots \otimes w_n \mid w_1, \dots, w_n \in L\}.$$

Theory of an automatic structure

We have for all $w_1, \ldots, w_n \in \Sigma^*$:

$$\begin{aligned}
& (w_1, \ldots, w_n) \in K(B_F) \\
\iff\quad & w_1 \otimes \cdots \otimes w_n \in L(B_F) \\
\iff\quad & (w_1, \ldots, w_n) \in L^n \text{ and } w_1 \otimes \cdots \otimes w_n \in f^{-1}(L(B_i)) \\
\iff\quad & (w_1, \ldots, w_n) \in L^n \text{ and } f(w_1 \otimes \cdots \otimes w_n) \in L(B_i) \\
\iff\quad & (w_1, \ldots, w_n) \in L^n \text{ and } w_{i_1} \otimes \cdots \otimes w_{i_k} \in L(B_i) \\
\iff\quad & (w_1, \ldots, w_n) \in L^n \text{ and } (w_{i_1}, \ldots, w_{i_k}) \in K(B_i) \\
\iff\quad & (w_1, \ldots, w_n) \in L^n \text{ and } (w_{i_1}, \ldots, w_{i_k}) \in R_i^{\mathcal{A}} \\
\iff\quad & (w_1, \ldots, w_n) \in L^n \text{ and } \mathcal{A}_{[x_1/w_1]\cdots[x_n/w_n]} \models R_i(x_{i_1}, \ldots, x_{i_k})
\end{aligned}$$

Theory of an automatic structure

Case 2: $F = (x_i = x_j)$, where $1 \leq i, j \leq n$:

Analogously to Case 1, since $\{(v, v) \mid v \in L\}$ is synchronously rational.

Case 3: $F = \neg G$:

By induction hypothesis there exists an n-tape automaton B_G for G.

We construct B_F such that:

$$L(B_F) = \{w_1 \otimes \cdots \otimes w_n \mid w_1, \ldots, w_n \in L\} \setminus L(B_G)$$

Case 4: $F = G \vee H$, where F contains only free variables from $x_1, \ldots, x_n$:

By the induction hypothesis there exist n-tape automata B_G and B_H for G and H, respectively.

We construct B_F such that $L(B_F) = L(B_G) \cup L(B_H)$.

Theory of an automatic structure

Case 5: $F = \exists x_{n+1} : G(x_1, \ldots, x_n, x_{n+1})$:

By the induction hypothesis there exists an $(n+1)$-tape automaton B_G for G.

Define the homomorphism $f : \left(\Sigma_{\#}^{n+1}\right)^* \to \left(\Sigma_{\#}^{n}\right)^*$ as follows, where $a_1, \ldots, a_n, a_{n+1} \in \Sigma_{\#}$:

$$f(a_1, \ldots, a_n, a_{n+1}) = \begin{cases} \varepsilon & \text{if } a_1 = \cdots = a_n = \# \\ (a_1, \ldots, a_n) & \text{otherwise} \end{cases}$$

Note: $f(w_1 \otimes \cdots \otimes w_n \otimes w_{n+1}) = w_1 \otimes \cdots \otimes w_n$ for all $w_1, \ldots, w_{n+1} \in \Sigma^*$.

We then construct for B_F an n-tape automaton with $L(B_F) = f(L(B_G))$

By the lemma from slide 71 we can construct such an automaton B_F.

Theory of an automatic structure

We have for all $w_1, \ldots, w_n \in \Sigma^*$:

$$
\begin{aligned}
& (w_1, \ldots, w_n) \in K(B_F) \\
\iff\quad & w_1 \otimes \cdots \otimes w_n \in L(B_F) \\
\iff\quad & w_1 \otimes \cdots \otimes w_n \in f(L(B_G)) \\
\iff\quad & \exists w_{n+1} : w_1 \otimes \cdots \otimes w_n \otimes w_{n+1} \in L(B_G) \\
\iff\quad & \exists w_{n+1} : (w_1, \ldots, w_n, w_{n+1}) \in K(B_G) \\
\iff\quad & \exists w_{n+1} : (w_1, \ldots, w_n, w_{n+1}) \in L^{n+1} \text{ and } \mathcal{A}_{[x_1/w_1]\cdots[x_{n+1}/w_{n+1}]} \models G \\
\iff\quad & (w_1, \ldots, w_n) \in L^n \text{ and } \exists w_{n+1} \in L : \mathcal{A}_{[x_1/w_1]\cdots[x_{n+1}/w_{n+1}]} \models G \\
\iff\quad & (w_1, \ldots, w_n) \in L^n \text{ and } \mathcal{A}_{[x_1/w_1]\cdots[x_n/w_n]} \models F
\end{aligned}
$$

This completes the construction of B_F.

Theory of an automatic structure

Assume now that F is a sentence (no free variables) and set $n = 1$ in our goal on slide 74.

We then have:

$$L(B_F) = \begin{cases} \Sigma^* \text{ if } \mathcal{A} \models F \\ \emptyset \text{ if } \mathcal{A} \not\models F \end{cases}$$

Therefore, it suffices to construct the automaton B_F and check whether it accepts a non-empty language (see lecture GTI). $\square$

Theory of an automatic structure

Remarks on the complexity: Our algorithm for checking $F \in \mathrm{Th}(\mathcal{A})$ is not very efficient.

Reason: For every negation $\neg$ we construct an automaton for the complement of the language of a previously constructed automaton. This increases the automaton size exponentially (power set construction)!

The running time of our algorithm is roughly $f_{|F|}(O(1))$, where $f_0(n) = n$ and $f_{i+1}(n) = 2^{f_i(n)}$ for $i \geq 0$ and $|F|$ = length of the formula F.

This is not avoidable:

Let $T_2 = (\{0,1\}^*, S_0, S_1, \leq)$ (the infinite binary tree) where:

- $S_0 = \{(w, w0) \mid w \in \{0,1\}^*\}$
- $S_1 = \{(w, w1) \mid w \in \{0,1\}^*\}$
- $\leq = \{(w, wu) \mid w, u \in \{0,1\}^*\}$

Note: T_2 is an automatic structure.

Theory of an automatic structure

Meyer 1974

There do not exist an $i \in \mathbb{N}$ and an algorithm that correctly decides $\mathrm{Th}(T_2)$ and whose running time is bounded by $f_i(n)$ (for an input formula of length n).

One also says: there is no elementary algorithm for $\mathrm{Th}(T_2)$.

But for many particular automatic structures one can come up with an elementary algorithm for the theory, for instance:

Oppen 1978

There is an algorithm that decides $\mathrm{Th}(\mathbb{N}, +)$ in time $2^{2^{2^{O(n)}}}$.

Oppen's algorithm uses the technique of quantifier elimination, which we will apply in the next section for another structure.

Decidability of real arithmetic

We want to prove the following famous theorem of Alfred Tarski:

Satz (Tarski 1948)

$\mathsf{Th}(\mathbb{R}, +, \cdot)$ is decidable.

Note: $(\mathbb{R}, +, \cdot)$ is not an automatic structure, since $\mathbb{R}$ is uncountable.

The proof of Tarski's theorem is quite long.

First, we extend the structure $(\mathbb{R}, +, \cdot)$ to $(\mathbb{R}, +, \cdot, <, 0, 1, -1)$.

Note: If $\mathsf{Th}(\mathbb{R}, +, \cdot, <, 0, 1, -1)$ is decidable then also $\mathsf{Th}(\mathbb{R}, +, \cdot)$ is decidable.

In fact, also the reverse implication can be shown but this is not important for us.

We will show that $\mathsf{Th}(\mathbb{R}, +, \cdot, <, 0, 1, -1)$ is decidable.

Decidability of real arithmetic

In the following, we will simply write $\mathbb{R}$ for $(\mathbb{R}, +, \cdot, <, 0, 1, -1)$.

For our decidability proof we will apply the method of quantifier elimination:

Let F be a formula with the free variables $y_0, \ldots, y_n$.

We construct a quantifier-free formula F' (that means, neither $\exists$ nor $\forall$ occurs in F') with the free variables $y_0, \ldots, y_n$ such that

$$\forall a_0, \ldots, a_n \in \mathbb{R} : \mathbb{R}_{[y_0/a_0, \ldots, y_n/a_n]} \models F \iff \mathbb{R}_{[y_0/a_0, \ldots, y_n/a_n]} \models F'$$

We do this by induction over the structure of the formula F.

Since $\forall x G \equiv \neg\exists x \neg G$, the only difficult case is where F has the form $F = \exists x\, G$.

By induction, we can assume that G is already quantifier-free.

Decidability of real arithmetic

Example: Let $F = \exists x : y = x \cdot x$.

The formula F expressed that y is a square.

A real number is a square if and only if it is not negative.

Hence, for every real number $a \in \mathbb{R}$ we have:

$$\mathbb{R}_{[y/a]} \models \exists x : y = x \cdot x \iff \mathbb{R}_{[y/a]} \models (y = 0 \vee 0 < y)$$

Thus, $F' = (y = 0 \vee 0 < y)$ is the desired quantifier-free formula.

Decidability of real arithmetic

Assume for a moment that we have already proved the quantifier elimination property.

Let F be a sentence (formula without free variables).

We want to check whether $\mathbb{R} \models F$ holds.

We apply quantifier elimination to F and construct a quantifier-free sentence F' such that:

$$\mathbb{R} \models F \quad \iff \quad \mathbb{R} \models F'.$$

Since F' is quantifier-free, we can easily check whether $\mathbb{R} \models F'$ holds:

- F' is a boolean combination of formulas $a = b$ and $a < b$.
- Here, a and b are terms that are constructed from the constants $0, 1, -1$ using the operations $\cdot$ and $+$.
- We can evaluate these terms a and b in $\mathbb{R}$ and then check whether $a = b$, resp. $a < b$, holds.

Decidability of real arithmetic

Back to quantifier elimination for $F = \exists x G$ with G quantifier-free.

Let the free variables of G be $x, y_0, \dots, y_n$; then the free variables of F are $y_0, \dots, y_n$.

For variables $x_1, \dots, x_n$ let $\mathbb{Z}[x_1, \dots, x_n]$ denote the set of all polynomials in the variables $x_1, \dots, x_n$ with coefficients from $\mathbb{Z}$.

Example: $-3x_1^4 x_2^2 x_3 + 7x_1 x_2^6 x_3^8 - 8x_2^4 x_3 + 12x_1 - 17$

Atomic subformulas of G are of the form $s = t$ and $s < t$, where s and t are terms that are constructed with $+$ and $\cdot$ from variables $(x, y_0, \dots, y_n)$ and the constants $-1, 0, 1$.

Such terms s and t can be evaluated to polynomials from $\mathbb{Z}[x, y_0, \dots, y_n]$.

In the following, we assume that $s, t \in \mathbb{Z}[x, y_0, \dots, y_n]$.

Decidability of real arithmetic

Finally, we can bring G into the following form:

$$G \; = \; s(x, y_0, \ldots, y_n) = 0 \wedge \bigwedge_{i=1}^{m} t_i(x, y_0, \ldots, y_n) > 0, \qquad (2)$$

where $s, t_1, \ldots, t_m \in \mathbb{Z}[x, y_0, \ldots, y_n]$.

First we eliminate all negations in G using the following equivalences:

- $s_1 = s_2 \iff s_1 - s_2 = 0$
- $s_1 < s_2 \iff s_2 - s_1 > 0$
- $\neg(s = 0) \iff (s > 0 \vee -s > 0)$
- $\neg(s > 0) \iff (s = 0 \vee -s > 0)$

Then we bring G into disjunctive normal form (thereby we do not introduce new negations).

Decidability of real arithmetic

Conjunctions of the form $\bigwedge_{i=1}^{k} s_i = 0$ can be replaced by a single equation using the following equivalence:

$$\bigwedge_{i=1}^{k} s_i = 0 \iff \sum_{i=1}^{k} s_i^2 = 0$$

Then we pull out the outermost disjunction using the following equivalence:

$$\exists x \Big(\bigvee_{i=1}^{k} G_i \Big) \equiv \bigvee_{i=1}^{k} \exists x\, G_i$$

Every formula $\exists x\, G_i$ has the desired form (2), and it suffices to apply quantifier elimination for each formula $\exists x\, G_i$.

Decidability of real arithmetic

We finally make a simple syntactic simplification.

Every polynomial $s, t_1, \dots, t_m \in \mathbb{Z}[x, y_0, \dots, y_n]$ in (2) can be uniquely written as a sum

$$\sum_{i=0}^{d} p_i \cdot x^{a_i}$$

with $0 \leq a_0 < a_1 < \cdots < a_d$ and $p_0, \dots, p_d \in \mathbb{Z}[y_0, \dots, y_n]$.

Example:

$$\begin{aligned} & 7 - 4y_0y_1^2y_3^4 + x^2 + y_1^5y_3x + 6y_0y_1^2y_3^4 - 2y_0^2y_1y_3^3x + 17y_0^3x^2 \\ = {} & (7 - 4y_0y_1^2y_3^4 + 6y_0y_1^2y_3^4) + (y_1^5y_3 - 2y_0^2y_1y_3^3) \cdot x + (17y_0^3 + 1) \cdot x^2 \end{aligned}$$

Decidability of real arithmetic

We can now replace every coefficient polynomial p_i by a new coefficient variable z_i, which appears in the formula only once.

After transforming the resulting formula into a quantifier-free formula, we can replace each of the new coefficient variables z_i by the original polynomial p_i.

Example: a possible formula $F = \exists x\, G$ that one might obtain in this way is

$$\exists x : z_0 + z_1 x^2 + z_2 x^3 = 0 \wedge z_3 x + z_4 x^2 > 0 \wedge z_5 + z_6 x^3 > 0.$$

In the following, let $z_0, \ldots, z_n$ be all coefficient variables in the formula G.

Recall that we want to construct a quantifier-free formula F' with:

$$\forall a_0, \ldots, a_n \in \mathbb{R} : \mathbb{R}_{[z_0/a_0, \ldots, z_n/a_n]} \models F \iff \mathbb{R}_{[z_0/a_0, \ldots, z_n/a_n]} \models F'$$

Decidability of real arithmetic

We show that it suffices to assume that F' satisfies the above equivalence only for all $a_0, \ldots, a_n \in \mathbb{R} \setminus \{0\}$.

For a subset $I \subseteq \{0, \ldots, n\}$ let G_I be the formula that is obtained from G by replacing for every $i \in I$ the variable z_i (and hence the term $z_i x^a$) by the constant 0.

Example: For our formula

$$G = (z_0 + z_1 x^2 + z_2 x^3 = 0 \wedge z_3 x + z_4 x^2 > 0 \wedge z_5 + z_6 x^3 > 0)$$

and $I = \{1, 3, 5\}$ we get

$$G_I = (z_0 + z_2 x^3 = 0 \wedge z_4 x^2 > 0 \wedge z_6 x^3 > 0).$$

Decidability of real arithmetic

We then replace the formula $\exists x\, G$ by the formula

$$\bigwedge_{I \subseteq \{0,\dots,n\}} \Big(\bigwedge_{i \in I} z_i = 0 \wedge \bigwedge_{i \notin I} z_i \neq 0 \ \rightarrow\ \exists x\, G_I \Big).$$

Here, the outer conjunction runs over all subsets $I \subseteq \{0,\dots,n\}$.

Assume we have constructed for every formula $F_I := \exists x\, G_I$ a quantifier-free formula F_I' with:

$$\forall a_0,\dots,a_n \in \mathbb{R} \setminus \{0\} : \mathbb{R}_{[z_0/a_0,\dots,z_n/a_n]} \models F_I \iff \mathbb{R}_{[z_0/a_0,\dots,z_n/a_n]} \models F_I'.$$

Then, for all $a_0,\dots,a_n \in \mathbb{R}$ the following statements are equivalent:

- $\mathbb{R}_{[z_0/a_0,\dots,z_n/a_n]} \models \exists x\, G$
- $\mathbb{R}_{[z_0/a_0,\dots,z_n/a_n]} \models \bigwedge_{I \subseteq \{0,\dots,n\}} \big(\bigwedge_{i \in I} z_i = 0 \wedge \bigwedge_{i \notin I} z_i \neq 0 \ \rightarrow\ \exists x\, G_I \big)$
- $\mathbb{R}_{[z_0/a_0,\dots,z_n/a_n]} \models \bigwedge_{I \subseteq \{0,\dots,n\}} \big(\bigwedge_{i \in I} z_i = 0 \wedge \bigwedge_{i \notin I} z_i \neq 0 \ \rightarrow\ F_I' \big)$

Decidability of real arithmetic

Remaining goal: For a formula $F = \exists x : s = 0 \wedge \bigwedge_{i=1}^{m} t_i > 0$ we have to construct a quantifier-free formula F' such that:

$$\forall a_0, \ldots, a_n \in \mathbb{R} \setminus \{0\} : \mathbb{R}_{[z_0/a_0,\ldots,z_n/a_n]} \models F \iff \mathbb{R}_{[z_0/a_0,\ldots,z_n/a_n]} \models F'.$$

Here, $s, t_1, \ldots, t_m$ are polynomials in the variables x, and the coefficients are parameters $z_0, \ldots, z_n$ that only take values $\neq 0$. Every parameter z_i appears in F only once.

Moreover, we can assume that:

- $t_i \neq 0$ for all $1 \leq i \leq m$ and
- $s = 0$ or x appears in s.

For this note that:

- If e.g. $t_1 = 0$ (the zero polynomial), then F is always wrong (we can therefore output the quantifier-free formula $0 = 1$).
- If $s \neq 0$ and x does not appear in s, then, again, F is always wrong.

Decidability of real arithmetic

We distinguish three cases:

- Case 1: x appears in s and $m = 1$.
- Case 2: x appears in s and $m > 1$
- Case 3: $s = 0$.

Case 1: $G = (s = 0 \wedge t > 0)$, where x appears in the polynomial s.

Notation: For $k \geq 0$ let $(\#x : G) = k$ be a new formula with the following semantics:

For all $a_0, \ldots, a_n \in \mathbb{R} \setminus \{0\}$: $\mathbb{R}_{[z_0/a_0,\ldots,z_n/a_n]} \models (\#x : G) = k$ if and only if

$$|\{a \in \mathbb{R} \mid \mathbb{R}_{[x/a,z_0/a_0,\ldots,z_n/a_n]} \models G\}| = k.$$

Strictly speaking, we extend here predicate logic by a new construct (the so-called counting quantifier $\#$).

Decidability of real arithmetic

Intuition: $(\#x : G) = k$ expresses that exactly k many x have the property G (i.e., $s = 0 \wedge t > 0$).

Note: if $d \geq 1$ is the x-degree of s (the largest number a such that x^a appears in s), then $\exists x\, G$ is equivalent in $\mathbb{R}$ to

$$(\#x : G) = 1 \vee (\#x : G) = 2 \vee \cdots \vee (\#x : G) = d,$$

because a polynomial $p(x)$ of degree d has at most d roots.

New goal: Find a quantifier-free formula which is equivalent to $(\#x : G) = k$ in $\mathbb{R}$.

For this we need some tools: polynomial division, Euclid's algorithm, Sturm sequences, formal derivates.

Decidability of real arithmetic

For $\overline{a} = (a_1, \ldots, a_n) \in (\mathbb{R} \setminus \{0\})^n$ let $\mathrm{Var}(\overline{a}) = |\{i < n \mid a_i a_{i+1} < 0\}|$.
(number of sign flips).

For $\overline{a} \in \mathbb{R}^n$ let $\mathrm{Var}(\overline{a}) = \mathrm{Var}(\overline{b})$, where $\overline{b}$ results from $\overline{a}$ by removing all zeros.

Example: $\mathrm{Var}(0, 2, 4, 0, -3, 0, 0, 2, 5) = \mathrm{Var}(2, 4, -3, 2, 5) = 2$.
(the red commas mark the sign flips)

For $\overline{f} = (f_1, \ldots, f_n) \in (\mathbb{R}[x])^n$ (an n-tuple of polynomials in the variables x) and $a \in \mathbb{R}$ let

$$\mathrm{Var}_a(\overline{f}) = \mathrm{Var}(f_1(a), \ldots, f_n(a)).$$

Decidability of real arithmetic

Recall from DMI: polynomial division with remainder

For polynomials $f, g \in \mathbb{R}[x]$ with $g \neq 0$ there exist unique polynomials $q, r \in \mathbb{R}[x]$ with

- $\deg(r) < \deg(g)$ or $r = 0$ and
- $f = q \cdot g + r$.

By replacing the remainder polynomial r by $-r$, we obtain $f = q \cdot g - r$.

Note: If $\deg(g) = 0$, i.e., $g \in \mathbb{R} \setminus \{0\}$, then $r = 0$ holds.

Decidability of real arithmetic

Example: We divide $(x^5 + x)$ by $(2x^2 + 1)$:

Decidability of real arithmetic

Example: We divide $(x^5 + x)$ by $(2x^2 + 1)$:

$$\begin{array}{l} (x^5 + x) : (2x^2 + 1) = \frac{1}{2}x^3 \\ -(x^5 + \frac{1}{2}x^3) \end{array}$$

Decidability of real arithmetic

Example: We divide $(x^5 + x)$ by $(2x^2 + 1)$:

$$\begin{array}{l} (x^5 + x) : (2x^2 + 1) = \frac{1}{2}x^3 \\ \underline{-(x^5 + \frac{1}{2}x^3)} \\ \qquad (-\frac{1}{2}x^3 + x) \end{array}$$

Decidability of real arithmetic

Example: We divide $(x^5 + x)$ by $(2x^2 + 1)$:

$$\begin{array}{l} (x^5 + x) : (2x^2+1) = \frac{1}{2}x^3 - \frac{1}{4}x \\ \underline{-(x^5 + \frac{1}{2}x^3)} \\ \quad (-\frac{1}{2}x^3 + x) \\ \quad -(-\frac{1}{2}x^3 - \frac{1}{4}x) \end{array}$$

Decidability of real arithmetic

Example: We divide $(x^5 + x)$ by $(2x^2 + 1)$:

$$
\begin{array}{l}
(x^5 + x) : (2x^2 + 1) = \frac{1}{2}x^3 - \frac{1}{4}x \\
\underline{-(x^5 + \frac{1}{2}x^3)} \\
\qquad (-\frac{1}{2}x^3 + x) \\
\qquad - \underline{(-\frac{1}{2}x^3 - \frac{1}{4}x)} \\
\qquad\qquad \frac{5}{4}x \text{ (remainder)}
\end{array}
$$

Decidability of real arithmetic

Example: We divide $(x^5 + x)$ by $(2x^2 + 1)$:

$$
\begin{array}{rl}
 & (x^5 + x) : (2x^2 + 1) = \frac{1}{2}x^3 - \frac{1}{4}x \\
 - & \underline{(x^5 + \frac{1}{2}x^3)} \\
 & (-\frac{1}{2}x^3 + x) \\
 - & \underline{(-\frac{1}{2}x^3 - \frac{1}{4}x)} \\
 & \frac{5}{4}x \text{ (remainder)}
\end{array}
$$

We get:

$$(x^5 + x) = (2x^2 + 1) \cdot (\frac{1}{2}x^3 - \frac{1}{4}x) + \frac{5}{4}x = (2x^2 + 1) \cdot (\frac{1}{2}x^3 - \frac{1}{4}x) - (-\frac{5}{4}x).$$

Decidability of real arithmetic

Euclid's algorithm for polynomials:

Let $f, g \in \mathbb{R}[x] \setminus \{0\}$ be non-zero polynomials.

Define the polynomials $h_0(x), \ldots, h_n(x) \in \mathbb{R}[x] \setminus \{0\}$ uniquely by:

$$\begin{aligned}
h_0(x) &= f(x) & & \\
h_1(x) &= g(x) & & \\
h_0(x) &= q_1(x)h_1(x) - h_2(x) & \qquad & \deg(h_2) < \deg(h_1) \\
h_1(x) &= q_2(x)h_2(x) - h_3(x) & & \deg(h_3) < \deg(h_2) \\
&\vdots & & \vdots \\
h_{n-2}(x) &= q_{n-1}(x)h_{n-1}(x) - h_n(x) & & \deg(h_n) < \deg(h_{n-1}) \\
h_{n-1}(x) &= q_n(x)h_n(x) & &
\end{aligned}$$

$h_{i+2} =$ division remainder if we divide h_i by h_{i+1}.

Decidability of real arithmetic

Remarks:

- Since $\deg(h_{i+1}) < \deg(h_i)$ for all $1 \leq i \leq n$, the division remainder must be finally 0.
- $h_n(x) = \gcd(f, g)$ (greatest common divisor of f and g)
- For all $0 \leq i \leq n$, the polynom $h_n(x)$ divides $h_i(x)$.

We define $[f, g] = (h_0(x), h_1(x), \ldots, h_n(x))$ as the Sturm sequence of f and g.

The reduced Sturm sequence of f and g is

$$\left(\frac{h_0(x)}{h_n(x)}, \frac{h_1(x)}{h_n(x)}, \ldots, \frac{h_{n-1}(x)}{h_n(x)}, \frac{h_n(x)}{h_n(x)}\right) = \left(\frac{h_0(x)}{h_n(x)}, \frac{h_1(x)}{h_n(x)}, \ldots, \frac{h_{n-1}(x)}{h_n(x)}, 1\right).$$

Decidability of real arithmetic

Example: We compute the Sturm sequence $[x^5 + x, x^2 + 2]$.

Successive polynomial division yields:

$$\begin{aligned}
x^5 + x &= (x^2 + 2) \cdot (x^3 - 2x) + 5x = (x^2 + 2) \cdot (x^3 - 2x) - (-5x) \\
x^2 + 2 &= (-5x) \cdot (-\frac{1}{5}x) + 2 = (-5x) \cdot (-\frac{1}{5}x) - (-2) \\
-5x &= (-2) \cdot \frac{5}{2}x
\end{aligned}$$

We therefore obtain

$$[x^5 + x, x^2 + 2] = (x^5 + x, x^2 + 2, -5x, -2).$$

Decidability of real arithmetic

For a polynomial $f(x) \in \mathbb{R}[x]$ we denote with f' the formal derivative of the polynomial f.

It is computed using the well-known rules for derivates.

Let $f, g \in \mathbb{R}[x]$, $a \in \mathbb{R}$:

- $a' = 0$
- $(a \cdot f)' = a \cdot f'$
- $(f + g)' = f' + g'$
- $(x^n)' = n \cdot x^{n-1}$ for $n \geq 1$

Also the product rule holds: $(f \cdot g)' = f' \cdot g + f \cdot g'$.

Example: For $f(x) = 4x^3 - 2x^2 + 5x - 3$ we have $f' = 12x^2 - 4x + 5$.

Decidability of real arithmetic

Let $f \in \mathbb{R}[x]$ be a polynomial with $f \neq 0$.

A real number $a \in \mathbb{R}$ is a root of f (i.e. $f(a) = 0$) if and only if $(x - a)$ divides f (i.e. $f = (x - a) \cdot g$ for some polynomial g).

Proof: If $f = (x - a) \cdot g$, then $f(a) = (a - a) \cdot g(a) = 0$.

Now assume that $f(a) = 0$.

Polynomial division of f by $x - a$: $f = (x - a) \cdot q + r$ with $\deg(r) < \deg(x - a) = 1$, i.e., $r \in \mathbb{R}$.

Because of $0 = f(a) = (a - a) \cdot q(a) + r = r$ we get $f = (x - a) \cdot q$.

This observation yields:

Lemma

Let $f, g \in \mathbb{R}[x] \setminus \{0\}$. If $\gcd(f, g) = 1$, then f and g have no common root.

Decidability of real arithmetic

A root a of a polynomial f is a multiple root of f, if $(x-a)^2$ divides f.

The following lemmas can be shown as simple exercises:

Lemma
A root a of f is a multiple root of f if and only if $f'(a) = 0$.

Lemma
If $\gcd(f, f') = 1$, then the polynomial f has no multiple root.

Decidability of real arithmetic

For a quantifier-free formula H with the only free variable x and $a, b \in \mathbb{R}$ with $a < b$ let

$$(\#x : H)_a^b = |\{c \in (a, b) \mid \mathbb{R}_{[x/c]} \models H\}|.$$

Here, $(a, b) = \{c \in \mathbb{R} \mid a < c < b\}$ is the open interval between a and b.

Hence, $(\#x : H)_a^b$ is the number of real values $c \in (a, b)$ for which H holds.

Example: $(\#x : x^2 - 2 = 0)_{-2}^2 = 2$, because there are two real roots of the polynomial $x^2 - 2$ ($-\sqrt{2}$ and $\sqrt{2}$) and both belong to $(-2, 2)$.

Moreover, $(\#x : x^2 - 2 = 0 \wedge x > 0)_{-2}^2 = 1$.

Decidability of real arithmetic

We now come to the central theorem for our further considerations:

Theorem of Sturm and Tarski

Let $f, g \in \mathbb{R}[x] \setminus \{0\}$, $f' \neq 0$, $\gcd(f, g) = \gcd(f, f') = 1$, $a, b \in \mathbb{R}$, $a < b$, $f(a) \neq 0 \neq f(b)$. Then the following identity holds:

$$(\#x : f(x) = 0 \wedge g(x) > 0)_a^b - (\#x : f(x) = 0 \wedge g(x) < 0)_a^b =$$
$$\mathrm{Var}_a([f, f'g]) - \mathrm{Var}_b([f, f'g]).$$

For the proof of the theorem of Sturm and Tarski we need two lemmas (Lemma A and Lemma B).

Decidability of real arithmetic

Lemma A

Let $f, g \in \mathbb{R}[x] \setminus \{0\}$, $a, b \in \mathbb{R}$, $a < b$, and $\forall c \in [a, b] : f(c) \neq 0$.
We have $\mathrm{Var}_a([f, g]) = \mathrm{Var}_b([f, g])$.

Proof of Lemma A: Let

$$[f, g] = S = (h_0, h_1, \dots, h_s)$$

and let

$$\tilde{S} = (\tilde{h}_0, \tilde{h}_1, \dots, \tilde{h}_s)$$

be the reduced Sturm sequence, i.e., $\tilde{h}_s = 1$ and $\tilde{h}_i = \frac{h_i}{h_s}$.

Let $N = \{c \in [a, b] \mid \exists 0 \leq i \leq s : \tilde{h}_i(c) = 0\}$.

The set N is finite (a polynomial $\neq 0$ has only finitely many roots).

Let $[a', b'] \subseteq [a, b]$ be an interval with $|N \cap [a', b']| \leq 1$.

Decidability of real arithmetic

It suffices to show: $\mathrm{Var}_{a'}(S) = \mathrm{Var}_{b'}(S)$.

Then we write $[a, b]$ as

$$[a, b] = [a_0, a_1] \cup [a_1, a_2] \cup [a_2, a_3] \cup \cdots \cup [a_{k-1}, a_k]$$

with $a_0 = a$, $a_k = b$ and $|N \cap [a_i, a_{i+1}]| \leq 1$ for all $0 \leq i \leq k-1$.

We obtain $\mathrm{Var}_{a_i}(S) = \mathrm{Var}_{a_{i+1}}(S)$ for all $0 \leq i \leq k-1$ and hence

$$\mathrm{Var}_a(S) = \mathrm{Var}_{a_0}(S) = \mathrm{Var}_{a_k}(S) = \mathrm{Var}_b(S).$$

So, let us show that $\mathrm{Var}_{a'}(S) = \mathrm{Var}_{b'}(S)$ if $|N \cap [a', b']| \leq 1$.

Since $f(a') \neq 0 \neq f(b')$ (because $\forall c \in [a, b] : f(c) \neq 0$) and $h_s = \gcd(f, g)$ divides f, we have $h_s(a') \neq 0 \neq h_s(b')$.

This implies $\mathrm{Var}_{a'}(S) = \mathrm{Var}_{a'}(\tilde{S})$ and $\mathrm{Var}_{b'}(\tilde{S}) = \mathrm{Var}_{b'}(S)$.

We show that $\mathrm{Var}_{a'}(\tilde{S}) = \mathrm{Var}_{b'}(\tilde{S})$.

Decidability of real arithmetic

Case 1: No $\tilde{h}_i$ has a root in $[a', b']$.

Since every polynomial $\tilde{h}_i$ is continuous and, by the intermediate value theorem, $\tilde{h}_i([a', b'])$ contains all values between $\tilde{h}_i(a')$ and $\tilde{h}_i(b')$, we must have

$$\tilde{h}_i(a') \cdot \tilde{h}_i(b') > 0$$

for all $0 \leq i \leq s$ ($\tilde{h}_i$ does not change its sign on $[a', b']$).

We obtain $\mathrm{Var}_{a'}(\tilde{S}) = \mathrm{Var}_{b'}(\tilde{S})$.

Case 2: At least one $\tilde{h}_i$ has a root $c \in [a', b']$.

By the choice of $[a', b']$ we have $N \cap [a', b'] = \{c\}$.

Decidability of real arithmetic

Since $\tilde{h}_s = 1$ and, by the assumption from the lemma, $f = h_0$ has no root in $[a, b]$ (then, also $\tilde{h}_0$ has no root in $[a, b]$), we must have $1 \leq i \leq s - 1$.

We have $\tilde{h}_{i-1}(c) = q_i(c)\tilde{h}_i(c) - \tilde{h}_{i+1}(c) = -\tilde{h}_{i+1}(c)$.

If $\tilde{h}_{i+1}(c) = 0 = \tilde{h}_i(c)$ would hold, then $\tilde{h}_j(c) = 0$ for all $j \geq i$ (since $\tilde{h}_{j+2}(c) = q_{j+1}(c)\tilde{h}_{j+1}(c) - \tilde{h}_j(c)$), which contradicts $\tilde{h}_s = 1$.

Therefore, we have $\tilde{h}_{i+1}(c) \neq 0$ and hence

$$\tilde{h}_{i-1}(c)\tilde{h}_{i+1}(c) = -(\tilde{h}_{i+1}(c))^2 < 0,$$

i.e., $\tilde{h}_{i-1}(c)$ and $\tilde{h}_{i+1}(c)$ have different signs.

Since $\tilde{h}_{i-1}$ and $\tilde{h}_{i+1}$ have no root in $[a', b']$ (c would have been the only possibility), the intermediate value theorem implies

$$\tilde{h}_{i-1}(a')\tilde{h}_{i+1}(a') < 0 \quad \text{und} \quad \tilde{h}_{i-1}(b')\tilde{h}_{i+1}(b') < 0.$$

Decidability of real arithmetic

We obtain:

$$\begin{aligned}
\mathrm{Var}_{a'}(\tilde{S}) &= \mathrm{Var}(\tilde{h}_0(a'), \ldots, \tilde{h}_{i-1}(a'), \tilde{h}_i(a'), \tilde{h}_{i+1}(a'), \ldots, \tilde{h}_s(a')) \\
&= \mathrm{Var}(\tilde{h}_0(a'), \ldots, \tilde{h}_{i-1}(a'), \tilde{h}_{i+1}(a'), \ldots, \tilde{h}_s(a'))
\end{aligned}$$

$$\begin{aligned}
\mathrm{Var}_{b'}(\tilde{S}) &= \mathrm{Var}(\tilde{h}_0(b'), \ldots, \tilde{h}_{i-1}(b'), \tilde{h}_i(b'), \tilde{h}_{i+1}(b'), \ldots, \tilde{h}_s(b')) \\
&= \mathrm{Var}(\tilde{h}_0(b'), \ldots, \tilde{h}_{i-1}(b'), \tilde{h}_{i+1}(b'), \ldots, \tilde{h}_s(b'))
\end{aligned}$$

In this way we can eliminate for every j with $\tilde{h}_j(c) = 0$ the entries $\tilde{h}_j(a')$ and $\tilde{h}_j(b')$.

We therefore obtain

$$\begin{aligned}
\mathrm{Var}_{a'}(\tilde{S}) &= \mathrm{Var}(g_0(a'), \ldots, g_t(a')) \\
\mathrm{Var}_{b'}(\tilde{S}) &= \mathrm{Var}(g_0(b'), \ldots, g_t(b'))
\end{aligned}$$

where the polynomials $g_0, \ldots, g_t$ have no root in $[a', b']$.

Decidability of real arithmetic

Hence, $g_i(a') \neq 0$ and $g_i(b') \neq 0$ have the same sign for all $0 \leq i \leq t$, which implies

$$\begin{aligned} \mathrm{Var}_{a'}(\tilde{S}) &= \mathrm{Var}(g_0(a'), \ldots, g_t(a')) \\ &= \mathrm{Var}(g_0(b'), \ldots, g_t(b')) \\ &= \mathrm{Var}_{b'}(\tilde{S}). \end{aligned}$$

$\square$

Lemma B

Let $f, g \in \mathbb{R}[x] \setminus \{0\}$, $f' \neq 0$, $\gcd(f, g) = \gcd(f, f') = 1$, $a, b, c \in \mathbb{R}$, $a < c < b$, $f(c) = 0$, $\forall d \in [a, b] \setminus \{c\} : f(d) \neq 0$. We have

$$\mathrm{Var}_a([f, f'g]) - \mathrm{Var}_b([f, f'g]) = \begin{cases} 1 & \text{if } g(c) > 0 \\ -1 & \text{if } g(c) < 0 \end{cases}$$

Decidability of real arithmetic

Proof of Lemma B:

Since $\gcd(f,g) = \gcd(f,f') = 1$, f and g have no common root, and f has no multiple root (Slides 104 and 105).

In particular, we have $g(c) \neq 0$ and there is a polynomial $h(x)$ with $f(x) = (x-c)\cdot h(x)$ and $h(c) \neq 0$.

We obtain $f\,f'\,g = f \cdot (h + (x-c)h') \cdot g = (x-c) \cdot \underbrace{(h^2 g + (x-c)\,h\,h'\,g)}_{u(x)}$.

Let $[f, f'g] = (f, f'g, h_2, \ldots, h_s)$ with $s \geq 1$.

Assume that $g(c) > 0$ (the case $g(c) < 0$ can be analyzed analogously).

We have $u(c) = (h(c))^2 g(c) > 0$ and $f'(c)g(c) \neq 0$.

Since $u(x)$ is continuous and $a < c < b$, there are $a' < b'$ with $a \leq a' < c < b' \leq b$ and $\forall x \in [a', b'] : u(x) > 0$ and $f'(x)g(x) \neq 0$.

With $f\,f'\,g = (x-c)\cdot u(x)$ we get $f(a')f'(a')g(a') < 0 < f(b')f'(b')g(b')$.

Decidability of real arithmetic

If $s \geq 2$ (in which case h_2 exists) we obtain:

$$\begin{aligned}
\mathsf{Var}_a([f, f'g]) &\overset{\text{Lemma A}}{=} \mathsf{Var}_{a'}([f, f'g]) \\
&= 1 + \mathsf{Var}_{a'}([f'g, h_2]) \\
&\overset{\text{Lemma A}}{=} 1 + \mathsf{Var}_{b'}([f'g, h_2]) \\
&= 1 + \mathsf{Var}_{b'}([f, f'g]) \\
&\overset{\text{Lemma A}}{=} 1 + \mathsf{Var}_b([f, f'g])
\end{aligned}$$

If $s = 1$ (i.e., $[f, f'g] = (f, f'g)$) we get

$$\begin{aligned}
\mathsf{Var}_a([f, f'g]) &\overset{\text{Lemma A}}{=} \mathsf{Var}_{a'}([f, f'g]) \\
&= 1 \\
&= 1 + \mathsf{Var}_{b'}([f, f'g]) \\
&\overset{\text{Lemma A}}{=} 1 + \mathsf{Var}_b([f, f'g])
\end{aligned}$$

Decidability of real arithmetic

Proof of the theorem of Tarski and Sturm:

Assume that $f, g \in \mathbb{R}[x] \setminus \{0\}$, $f' \neq 0$, $\gcd(f,g) = \gcd(f,f') = 1$, $a, b \in \mathbb{R}$, $a < b$, $f(a) \neq 0 \neq f(b)$.

Let $N = \{c \in (a,b) \mid f(c) = 0\}$ (a finite set).

If $N = \emptyset$ we obtain with Lemma A:

$$(\#x : f(x) = 0 \wedge g(x) > 0)_a^b - (\#x : f(x) = 0 \wedge g(x) < 0)_a^b \ = \ 0 \ = \mathrm{Var}_a([f, f'g]) - \mathrm{Var}_b([f, f'g]).$$

Now assume that $N = \{c_1, c_2, \ldots, c_n\}$ with $n \geq 1$.

Choose points $a = a_0 < c_1 < a_1 < c_2 < a_2 < \cdots < a_{n-1} < c_n < a_n = b$.

Decidability of real arithmetic

With Lemma B we obtain for all $1 \leq i \leq n$:

$$\mathsf{Var}_{a_{i-1}}([f, f'g]) - \mathsf{Var}_{a_i}([f, f'g]) = \begin{cases} 1 & \text{if } g(c_i) > 0 \\ -1 & \text{if } g(c_i) < 0 \end{cases}$$

Summing over all i yields:

$$\mathsf{Var}_a([f, f'g]) - \mathsf{Var}_b([f, f'g]) =$$
$$(\#x : f(x) = 0 \wedge g(x) > 0)_a^b - (\#x : f(x) = 0 \wedge g(x) < 0)_a^b$$

This concludes the proof of the theorem of Tarski and Sturm. □

Decidability of real arithmetic

Corollary of the theorem of Tarski and Sturm

Let $f, g \in \mathbb{R}[x] \setminus \{0\}$, $f' \neq 0$, $\gcd(f,g) = \gcd(f,f') = 1$, $a, b \in \mathbb{R}$, $a < b$, $f(a) \neq 0 \neq f(b)$. We then have

$$\begin{aligned}&\#x(f(x) = 0 \wedge g(x) > 0)_a^b \\ &= \frac{1}{2}(\mathsf{Var}_a([f, f'g]) - \mathsf{Var}_b([f, f'g]) + \mathsf{Var}_a([f, f']) - \mathsf{Var}_b([f, f'])).\end{aligned}$$

Proof: The theorem of of Tarski and Sturm yields

$$\begin{aligned}&(\#x : f(x) = 0 \wedge g(x) > 0)_a^b - (\#x : f(x) = 0 \wedge g(x) < 0)_a^b \\ &= \mathsf{Var}_a([f, f'g]) - \mathsf{Var}_b([f, f'g]).\end{aligned}$$

Decidability of real arithmetic

as well as (since f and g have no common root due to $\gcd(f,g)=1$)

$$\begin{aligned}
&(\#x : f(x) = 0 \wedge g(x) > 0)_a^b + (\#x : f(x) = 0 \wedge g(x) < 0)_a^b \\
&= (\#x : f(x) = 0)_a^b = \\
&= (\#x : f(x) = 0 \wedge 1 > 0)_a^b - (\#x : f(x) = 0 \wedge 1 < 0)_a^b \\
&= \mathsf{Var}_a([f, f']) - \mathsf{Var}_b([f, f']).
\end{aligned}$$

Adding both equalities gives:

$$\begin{aligned}
&2 \cdot (\#x : f(x) = 0 \wedge g(x) > 0)_a^b \\
&= \mathsf{Var}_a([f, f'g]) - \mathsf{Var}_b([f, f'g]) + \mathsf{Var}_a([f, f']) - \mathsf{Var}_b([f, f'])
\end{aligned}$$

$\square$

Decidability of real arithmetic

We also need Cauchy's bound for the roots of a polynomial:

Lemma (Cauchy's bound)

Let $f(x) = a_m x^m + \cdots + a_1 x + a_0 \in \mathbb{R}[x]$, $a_m \neq 0$. All real roots of the polynomial f belong to the interval $(-c, c)$ with

$$c = 1 + \frac{\max\{|a_0|, \ldots, |a_{m-1}|\}}{|a_m|}.$$

Decidability of real arithmetic

Proof of the Cauchy bound:

Assume that we have already proved the Cauchy bound for the case $a_m = 1$.

Then we get the general statement as follows:

Let α be a root of $f(x) = a_m x^m + a_{m-1} x^{m-1} + \cdots + a_1 x + a_0$ with $a_m \neq 0$.

Then α is also a root of the polynomial $x^m + \frac{a_{m-1}}{a_m} x^{m-1} + \cdots + \frac{a_1}{a_m} x + \frac{a_0}{a_m}$.

The Cauchy bound for the case $a_m = 1$ yields

$$|\alpha| < 1 + \max\{|\frac{a_i}{a_m}| \mid 0 \leq i \leq m-1\} = 1 + \frac{\max\{|a_i| \mid 0 \leq i \leq m-1\}}{|a_m|}.$$

Decidability of real arithmetic

It remains to prove the Cauchy bound for a polynomial

$$f(x) = x^m + a_{m-1}x^{m-1} + \cdots + a_1 x + a_0.$$

Let $h = \max\{|a_i| \mid 0 \le i \le m-1\}$.

Assume that $f(\alpha) = \alpha^m + a_{m-1}\alpha^{m-1} + \cdots + a_1\alpha + a_0 = 0$, i.e.,

$$\alpha^m = -a_{m-1}\alpha^{m-1} - \cdots - a_1\alpha - a_0. \tag{3}$$

We show that $|\alpha| < 1 + h$.

If $|\alpha| \le 1$, we have $|\alpha| < 1 + h$ (if $h = 0$ then we have $\alpha^m = 0$, i.e., $\alpha = 0$).

Now assume that $|\alpha| > 1$.

Decidability of real arithmetic

Using (3) and the laws $|a+b| \leq |a| + |b|$, $|a \cdot b| = |a| \cdot |b|$ for all $a, b \in \mathbb{R}$, we get

$$\begin{aligned}|\alpha|^m &\leq |a_{m-1}| \cdot |\alpha|^{m-1} + \cdots + |a_1| \cdot |\alpha| + |a_0| \\ &\leq h \cdot (|\alpha|^{m-1} + \cdots + |\alpha| + 1) \\ &= h \cdot \frac{|\alpha|^m - 1}{|\alpha| - 1}.\end{aligned}$$

Since $|\alpha| > 1$, we obtain:

$$|\alpha| - 1 \leq h \cdot \frac{|\alpha|^m - 1}{|\alpha|^m} < h \qquad \square$$

Decidability of real arithmetic

We now come back to Case 1 (see Slide 95):

Recall: We want to find a quantifier-free arithmetic formula for $(\#x : s = 0 \wedge t > 0) = k$, where

$$\begin{aligned} s &= z_0 + z_1 x + \cdots + z_m x^m \\ t &= z_{m+1} + z_{m+2} x + \cdots + z_n x^{n-m-1} \end{aligned}$$

and $1 \leq m < n$ (if $m = n$ then $t = 0$ and $(\#x : s = 0 \wedge t > 0) = 0$).

The desired quantifier-free formula has the free variables $z_0, \dots, z_n$. Moreover we can restrict to the case that all z_i only take values $\neq 0$.

Let y and z be two new variables.

By Cauchy's bound it suffices to find for $(\#x : s = 0 \wedge t > 0)^z_y = k$ (see Slide 106) a quantifier-free formula with free variables $y, z, z_0, \dots, z_m$.

Decidability of real arithmetic

In this formula we can replace the interval borders y and z by

$$-\frac{|z_m| + \max\{|z_0|, \ldots, |z_{m-1}|\}}{|z_m|}, \text{ resp. } \frac{|z_m| + \max\{|z_0|, \ldots, |z_{m-1}|\}}{|z_m|}.$$

Applications of $|\cdot|$ and max can be eliminated by case distinctions (similar to Slide 92).

Examples:
$|z_i| + y = z$ becomes $(z_i \geq 0 \rightarrow z_i + y = z \wedge z_i < 0 \rightarrow y = z + z_i)$.
$\max\{z_i, z_j\} = x$ becomes $(z_i \geq z_j \rightarrow x = z_i \wedge z_i < z_j \rightarrow x = z_j)$.

Applications of $\frac{\cdot}{|z_m|}$ (in case $z_m \neq 0$) can be eliminated by multiplication with sufficiently large powers of z_m.

Decidability of real arithmetic

Using the corollary of the Sturm-Tarski theorem from Slide 118 we construct a quantifier-free formula for

$$(\#x : s = 0 \wedge t > 0)_y^z = k$$

with free variables y, z, $z_0, \dots, z_n$.

But: Are all the assumptions for $f = s$ and $g = t$ from Slide 118 satisfied?

- $s \neq 0$ and $s' \neq 0$, since $m \geq 1$ on Slide 124 and all z_i are $\neq 0$.
- $t \neq 0$, since all z_i are $\neq 0$ and the case $m < n$ on Slide 124.
- $\gcd(s, t) = \gcd(s, s') = 1$: does not hold for all $z_i \neq 0$!

Decidability of real arithmetic

How do we ensure the assumption $\gcd(s,t) = \gcd(s,s') = 1$?

We have:

- $(\#x : s = 0 \wedge t > 0)_y^z = (\#x : s/\gcd(s,t) = 0 \wedge t > 0)_y^z$:

  By replacing s by $s/\gcd(s,t)$ we eliminate for s only common roots of s and t (for which $t > 0$ does not hold).

- $(\#x : s = 0 \wedge t > 0)_y^z = (\#x : s/\gcd(s,s') = 0 \wedge t > 0)_y^z$:

  s and $s/\gcd(s,s')$ have the same roots (only the multiplicity of the roots of s is reduced to one when dividing by $\gcd(s,s')$).

Decidability of real arithmetic

In this way we can reduce the degree of s until finally $\gcd(s, t) = \gcd(s, s') = 1$ holds.

The gcd-computations have to be done symbolically, since the coefficients of s and t are parameters $z_i \neq 0$.

Example: $m = 2$, $n = 4$, i.e.,

$$s(x) = z_0 + z_1 x + z_2 x^2 \text{ and } t(x) = z_3 + z_4 x$$

We first compute symbolically

$$\gcd(s, t) = \gcd(z_0 + z_1 x + z_2 x^2, z_3 + z_4 x).$$

Decidability of real arithmetic

In order to make the computation more convenient, we multiply s with $z_4^2 \neq 0$.

We have $(\#x : s = 0 \wedge t > 0)^z_y = (\#x : z_4^2 \cdot s = 0 \wedge t > 0)^z_y$.

Division with remainder:

$$\begin{array}{l}
\quad (z_2 z_4^2 x^2 + z_1 z_4^2 x + z_0 z_4^2) : (z_4 x + z_3) = z_2 z_4 x + (z_1 z_4 - z_2 z_3) \\
-\underline{(z_2 z_4^2 x^2 + z_2 z_4 z_3 x)} \\
\quad ((z_1 z_4^2 - z_2 z_4 z_3)x + z_0 z_4^2) \\
-\underline{((z_1 z_4^2 - z_2 z_4 z_3)x + (z_1 z_4 z_3 - z_2 z_3^2))} \\
\quad z_0 z_4^2 - z_1 z_4 z_3 + z_2 z_3^2 \text{ (remainder)}
\end{array}$$

Decidability of real arithmetic

Therefore:

- If $z_0 z_4^2 - z_1 z_4 z_3 + z_2 z_3^2 \neq 0$, then $\gcd(z_4^2 s, t) = \gcd(s, t) = 1$.
- If $z_0 z_4^2 - z_1 z_4 z_3 + z_2 z_3^2 = 0$, then $\gcd(z_4^2 s, t) = t = (z_4 x + z_3)$ und $\frac{z_4^2 s}{t} = z_2 z_4 x + (z_1 z_4 - z_2 z_3)$.

  Moreover:

  $$(\# x : s = 0 \wedge t > 0)_y^z = (\# x : z_2 z_4 x + z_1 z_4 - z_2 z_3 = 0 \wedge t > 0)_y^z$$

  At this point we do not necessarily have $\gcd(z_2 z_4 x + z_1 z_4 - z_2 z_3, t) = 1$, but the x-degree of $z_2 z_4 x + z_1 z_4 - z_2 z_3$ is smaller than the x-degree of s.

  We therefore can continue in the same way.

Decidability of real arithmetic

In our concrete situation this is quite easy:

The only root of $z_2 z_4 \cdot x + z_1 z_4 - z_2 z_3$ is $\frac{z_2 z_3 - z_1 z_4}{z_2 z_4}$.

The only root of $t = z_4 \cdot x + z_3$ is $-\frac{z_3}{z_4}$.

Hence, if $\frac{z_2 z_3 - z_1 z_4}{z_2 z_4} \neq -\frac{z_3}{z_4}$ (i.e., $z_1 z_4 \neq 2 z_2 z_3$) then

$$\gcd(z_2 z_4 x + z_1 z_4 - z_2 z_3, t) = 1.$$

On the other hand, if $z_1 z_4 = 2 z_2 z_3$ then $(\# x : s = 0 \wedge t > 0)^z_y = 0$.

Decidability of real arithmetic

To sum up, for $s(x) = z_0 + z_1 x + z_2 x^2$ and $t(x) = z_3 + z_4 x$ we have:

- If $z_0 z_4^2 - z_1 z_4 z_3 + z_2 z_3^2 \neq 0$ then $\gcd(s,t) = 1$.
- If $z_0 z_4^2 - z_1 z_4 z_3 + z_2 z_3^2 = 0$ and $z_1 z_4 \neq 2 z_2 z_3$ then
$$(\#x : s = 0 \wedge t > 0)^z_y = (\#x : z_2 z_4 x + z_1 z_4 - z_2 z_3 = 0 \wedge t > 0)^z_y$$
  and $\gcd(z_2 z_4 x + z_1 z_4 - z_2 z_3, t) = 1$.
- If $z_0 z_4^2 - z_1 z_4 z_3 + z_2 z_3^2 = 0$ and $z_1 z_4 = 2 z_2 z_3$ then
$$(\#x : s = 0 \wedge t > 0)^z_y = 0.$$

In the same way we can ensure the assumption $\gcd(s, s') = 1$.

Decidability of real arithmetic

Under the assumptions $\gcd(s,t) = \gcd(s,s') = 1$ and $s(y) \neq 0 \neq s(z)$, $(\#x : s = 0 \wedge t > 0)_y^z = k$ is equivalent to

$$\mathrm{Var}_y([s, s't]) - \mathrm{Var}_z([s, s't]) + \mathrm{Var}_y([s, s']) - \mathrm{Var}_z([s, s']) = 2k$$

This can be expressed as a boolean combination of statements of the form $\mathrm{Var}_y([s, s't]) = i_1$, $\mathrm{Var}_z([s, s't]) = i_2$, $\mathrm{Var}_y([s, s']) = i_3$, $\mathrm{Var}_z([s, s']) = i_4$.

Finally, a statement $\mathrm{Var}_y([s, s't]) = i$ (analogously for the other polynomials) can be expressed by a quantifier-free formula.

For this we execute the Euclidean algorithm symbolically for s and $s't$ and thereby compute symbolically the Sturm sequence $[s, s't]$ and then replace the variable of the polynomials in the Sturm sequence by y.

This concludes case 1. We continue with case 2 from Slide 95.

Decidability of real arithmetic

Case 2: $G = (s = 0 \wedge \bigwedge_{i=1}^{m} t_i > 0)$, $m \geq 1$, and x appears in s.

Induction over m:

Induction base: $m = 1$. See case 1.

Induction step: let $m \geq 2$.

Let $G' = (s = 0 \wedge \bigwedge_{i=1}^{m-2} t_i > 0)$. We have:

$$\begin{aligned} &\#x(G' \wedge t_{m-1} > 0 \wedge t_m > 0) + \\ &\#x(G' \wedge t_{m-1} > 0 \wedge t_m < 0) \;=\; \#x(G' \wedge t_{m-1} t_m^2 > 0) \end{aligned} \tag{4}$$

$$\begin{aligned} &\#x(G' \wedge t_{m-1} > 0 \wedge t_m > 0) + \\ &\#x(G' \wedge t_{m-1} < 0 \wedge t_m > 0) \;=\; \#x(G' \wedge t_{m-1}^2 t_m > 0) \end{aligned} \tag{5}$$

$$\begin{aligned} &\#x(G' \wedge t_{m-1} > 0 \wedge t_m < 0) + \\ &\#x(G' \wedge t_{m-1} < 0 \wedge t_m > 0) \;=\; \#x(G' \wedge t_{m-1} t_m < 0) \end{aligned} \tag{6}$$

Decidability of real arithmetic

(4) + (5) - (6) yields:

$$\begin{aligned} 2 \cdot \#x\, G &= 2 \cdot \#x \cdot (G' \wedge t_{m-1} > 0 \wedge t_m > 0) \\ &= \#x(G' \wedge t_{m-1} t_m^2 > 0) \; + \\ &\quad\; \#x(G' \wedge t_{m-1}^2 t_m > 0) \; - \\ &\quad\; \#x(G' \wedge -t_{m-1} t_m > 0) \end{aligned}$$

Case 3 from Slide 95: $s = 0$, i.e., $G = \bigwedge_{i=1}^{m} t_i > 0$ with $t_i \neq 0$.

Let $t = t_1 t_2 \cdots t_m$.

Claim: $\exists x\ G$ is equivalent in $\mathbb{R}$ to

$$\exists x_0 \forall x \le x_0 : G \ \vee\ \exists x_0 \forall x \ge x_0 : G \ \vee\ \exists x\, (t'(x) = 0 \wedge G). \qquad (7)$$

The implication (7) $\Rightarrow \exists x\ G$ is clear.

Decidability of real arithmetic

Now assume that $\mathbb{R} \models \exists x\, G$.

We obtain

$$\begin{aligned}\mathbb{R} \ \models\ & \exists x_0 \forall x \leq x_0 : G \ \vee\ \exists x_0 \forall x \geq x_0 : G\ \vee \\ & \exists x_1\, \exists x\, \exists x_2\, (x_1 < x < x_2 \ \wedge\ \neg G[x/x_1] \wedge G \wedge \neg G[x/x_2]).\end{aligned}$$

Assume that

$$\mathbb{R} \models \exists x_1\, \exists x\, \exists x_2\, (x_1 < x < x_2 \ \wedge\ \neg G[x/x_1] \wedge G \wedge \neg G[x/x_2])$$

Then there are $x_1, x, x_2 \in \mathbb{R}$ and $i, j \in \{1, \ldots, m\}$ with

- $x_1 < x < x_2$,
- $t_i(x_1) \leq 0$,
- $t_j(x_2) \leq 0$,
- $t_k(x) > 0$ for alle $1 \leq k \leq m$

Decidability of real arithmetic

Then there are also $x_1', x_2' \in \mathbb{R}$ with $x_1' < x < x_2'$ and $t_i(x_1') = t_j(x_2') = 0$.

We get $t(x_1') = t(x_2') = 0$.

Since t has only finitely many roots, we can choose for x_1' (x_2') the greatest (smallest) root of t, which is smaller (greater) than x.

We get $x_1' < x < x_2'$, $t(x_1') = 0 = t(x_2')$ and $t_k(y) > 0$ for all $y \in (x_1', x_2')$ and $1 \le k \le m$.

We obtain $t(y) > 0$ for all $y \in (x_1', x_2')$.

By Rolle's theorem (`https://de.wikipedia.org/wiki/Satz_von_Rolle`) there exists an x with $t'(x) = 0$ and $t_i(x) > 0$ for all $1 \le i \le m$, i.e.,

$$\mathbb{R} \models \exists x_0 \forall x \le x_0 : G \;\vee\; \exists x_0 \forall x \ge x_0 : G \;\vee\; \exists x \, (t'(x) = 0 \wedge G).$$

This shows the claim.

Decidability of real arithmetic

It now suffices to find a quantifier-free formula for

$$\exists x_0 \forall x \leq x_0 : G \ \vee \ \exists x_0 \forall x \geq x_0 : G \ \vee \ \exists x \, (t'(x) = 0 \wedge G).$$

For the formulas $\exists x_0 \forall x \leq x_0 \; G$ and $\exists x_0 \forall x \geq x_0 \; G$ one can easily find quantifier-free formulas.

To see this, note that for a polynomial $a_n x^n + \cdots + a_1 x + a_0$ with $a_n \neq 0$ we have

$$\exists x_0 \forall x \leq x_0 \; (a_n x^n + \cdots + a_1 x + a_0 > 0)$$

if and only if one of the following cases holds:

- n is even and $a_n > 0$,
- n is odd and $a_n < 0$.

Decidability of real arithmetic

The formula $\exists x\,(t'(x) = 0 \wedge G)$ can be made quantifier-free using case 1, respectively case 2, if x appears in $t'(x)$.

If x does not appear in $t'(x)$, then the x-degree of $t(x) = t_1(x) \cdots t_m(x)$ is at most 1.

This means that x appears in at most one of the polynomials t_i, without loss of generality assume that x appears in t_1.

Moreover, the x-degree of $t_1(x)$ is at most one.

If $t_i = z_i$ for $1 \le i \le m$, then $\exists x \bigwedge_{i=1}^{m} t_i > 0$ is equivalent to $\bigwedge_{i=1}^{m} z_i > 0$.

If $t_1 = z_1 \cdot x + z_0$ and $t_i = z_i$ for $2 \le i \le m$, then $\exists x \bigwedge_{i=1}^{m} t_i > 0$ is equivalent to $\bigwedge_{i=2}^{m} t_i > 0$

This concludes our proof of Tarski's theorem. $\square$

Decidability of real arithmetic

Tarski's theorem implies that there is no arithmetical formula $F(x)$ with a single free variable x such that for all $r \in \mathbb{R}$:

$$(\mathbb{R}, +, \cdot)_{[x/r]} \models F(x) \iff r \in \mathbb{N}$$

If there would exist such a formula $F(x)$, then together with Gödel's theorem ($\mathrm{Th}(\mathbb{N}, +, \cdot)$ is undecidable) the undecidability of $\mathrm{Th}(\mathbb{R}, +, \cdot)$ would follow.

Surprisingly, Julia Robinson found in 1949 such a formula for $\mathbb{Q}$ instead of $\mathbb{R}$:

Theorem (Robinson 1949)

There exists an arithmetical formula $F(x)$ with a single free variable x such that for all rational numbers $r \in \mathbb{Q}$:

$$(\mathbb{Q}, +, \cdot)_{[x/r]} \models F(x) \iff r \in \mathbb{N}$$

Consequence: $\mathrm{Th}(\mathbb{Q}, +, \cdot)$ is undecidable.

Monadic second order logic

Monadic second order logic (MSO for short) is an extension of predicate logic (which is also denoted as first order logic), where quantification over subsets of the universe is allowed.

For this we fix two set of variables:

- first order variables: $\mathsf{Var}_0 = \{x_1, x_2, x_3, \ldots\}$
- second order variables: $\mathsf{Var}_1 = \{X_1, X_2, X_3, \ldots\}$

We have $\mathsf{Var}_0 \cap \mathsf{Var}_1 = \emptyset$.

Variables from Var_0 are denoted with $x, y, z, x', x_0, \ldots$, whereas variables from Var_1 are denoted with $X, Y, Z, X', X_0, \ldots$.

As in predicate logic (see Logik I) we have predicate symboles P_i^k (k-ary) and function symbols f_i^k (k-ary).

Terms are defined as in predicate logic using function symbols and variables from Var_0.

Monadic second order logic

The set MSO of all MSO-formulas is the smallest set with:

- if t_1, t_2 are terms and $X \in \mathrm{Var}_1$, then $(t_1 = t_2), (t_1 \in X) \in \mathrm{MSO}$;
- if $t_1, t_2, \ldots, t_k$ are terms and P is a k-ary predicate symbol, then $P(t_1, \ldots, t_k) \in \mathrm{MSO}$;
- if $F, G \in \mathrm{MSO}$, then $\neg F,\ F \wedge G,\ F \vee G \in \mathrm{MSO}$.
- if $F \in \mathrm{MSO}$ and $x \in \mathrm{Var}_0, X \in \mathrm{Var}_1$, then $\exists x F,\ \exists X F,\ \forall x F,\ \forall X F \in \mathrm{MSO}$.

The set $\mathrm{free}(F) \subseteq \mathrm{Var}_0 \cup \mathrm{Var}_1$ of all free variables of $F \in \mathrm{MSO}$ is defined as in predicate logic.

For $F \in \mathrm{MSO}$ we also write $F(x_1, \ldots, x_n, X_1, \ldots, X_m)$ in order to express $\mathrm{free}(F) = \{x_1, \ldots, x_n, X_1, \ldots, X_m\}$.

A formula $F \in \mathrm{MSO}$ with $\mathrm{free}(F) = \emptyset$ is called an MSO-sentence.

Monadic second order logic

A structure is now a pair $\mathcal{A} = (U_{\mathcal{A}}, I_{\mathcal{A}})$, where $U_{\mathcal{A}}$ is a non-empty set (the universe) and $I_{\mathcal{A}}$ is a partially defined mapping, which assigns

- to every k–ary predicate symbol P from the domain of $I_{\mathcal{A}}$ a k–ary relation $I_{\mathcal{A}}(P) \subseteq U_{\mathcal{A}}^k$,
- to every k–ary function symbol f from the domain of $I_{\mathcal{A}}$ a k–ary function $I_{\mathcal{A}}(f) : U_{\mathcal{A}}^k \to U_{\mathcal{A}}$,
- to every variable $x \in \mathrm{Var}_0$ from the domain of $I_{\mathcal{A}}$ an element $I_{\mathcal{A}}(x) \in U_{\mathcal{A}}$, and
- to every variable $X \in \mathrm{Var}_1$ from the domain of $I_{\mathcal{A}}$ a subset $I_{\mathcal{A}}(X) \subseteq U_{\mathcal{A}}$.

A structure $\mathcal{A}$ is suitable for a formula $F \in \mathrm{MSO}$, if $I_{\mathcal{A}}$ is defined for every predicate symbol, function symbol and free variable that appears in F.

Monadic second order logic

Let $\mathcal{A}$ be suitable for F. We write $\mathcal{A} \models F$ if one of the following cases holds (the evaluation $\mathcal{A}(t) \in U_{\mathcal{A}}$ of a term t is defined as in predicate logic):

- $F = (t_1 = t_2)$ and $\mathcal{A}(t_1) = \mathcal{A}(t_2)$
- $F = (t \in X)$ and $\mathcal{A}(t) \in I_{\mathcal{A}}(X)$
- $F = P(t_1, \dots, t_k)$ and $(\mathcal{A}(t_1), \dots, \mathcal{A}(t_k)) \in I_{\mathcal{A}}(P)$
- $F = \neg G$ and $\mathcal{A} \models G$ does not hold.
- $F = G \wedge H$ and ($\mathcal{A} \models G$ and $\mathcal{A} \models H$)
- $F = G \vee H$ and ($\mathcal{A} \models G$ or $\mathcal{A} \models H$)
- $F = \exists x \; G$ and there exists $a \in U_{\mathcal{A}}$ with $\mathcal{A}_{[x/a]} \models G$
- $F = \forall x \; G$ and for all $a \in U_{\mathcal{A}}$ we have $\mathcal{A}_{[x/a]} \models G$
- $F = \exists X \; G$ and there exists $B \subseteq U_{\mathcal{A}}$ with $\mathcal{A}_{[X/B]} \models G$
- $F = \forall X \; G$ and for all $B \subseteq U_{\mathcal{A}}$ we have $\mathcal{A}_{[X/B]} \models G$.

Monadic second order logic

Conventions:

- In the following we identify a symbol P with its interpretation $I_{\mathcal{A}}(P)$.
- A structure $\mathcal{A} = (U_{\mathcal{A}}, I_{\mathcal{A}})$ with $\mathrm{dom}(I_{\mathcal{A}}) = \{P_1, \ldots, P_n, f_1, \ldots, f_m\}$ is also written as $(U_{\mathcal{A}}, I_{\mathcal{A}}(P_1), \ldots, I_{\mathcal{A}}(P_n), I_{\mathcal{A}}(f_1), \ldots, I_{\mathcal{A}}(f_m))$ or just $(U_{\mathcal{A}}, P_1, \ldots, P_n, f_1, \ldots, f_m)$.
- For an MSO-formula $F = F(x_1, \ldots, x_n, X_1, \ldots, X_m)$ and $a_1, \ldots, a_n \in U_{\mathcal{A}}$, $A_1, \ldots, A_m \subseteq U_{\mathcal{A}}$ we also write $\mathcal{A} \models F(a_1, \ldots, a_n, A_1, \ldots, A_m)$ for $\mathcal{A}_{[x_1/a_1, \ldots, x_n/a_n, X_1/A_1, \ldots, X_m/A_m]} \models F$.

The MSO-theory of a structure $\mathcal{A}$ is the set of all MSO-sentences F with $\mathcal{A} \models F$.

Monadic second order logic: Example

An example for a useful MSO-formula:

Let $\mathcal{G} = (V, E)$ be a directed graph, which is the structure with universe V and the binary relation $E \subseteq V \times V$.

The following formula reach(x, y) expresses that in $\mathcal{G}$ there is a path from vertex x to vertex y:

$$\text{reach}(x,y) = \forall X \Big(\big(x \in X \wedge \forall u \forall v (E(u,v) \wedge u \in X \ \rightarrow v \in X)\big) \rightarrow y \in X \Big)$$

Proof: We say that a subset $U \subseteq V$ of vertices is closed under the edge relation E if for every $(u, v) \in E$ the following holds: if $u \in U$ then $v \in U$.

The formula reach(x, y) says that every subset $U \subseteq V$ that is closed under the edge relation E and that contains x must also contain y.

Monadic second order logic: Example

This is indeed equivalent to the fact that there is a path from x to y, i.e., $(x, y) \in E^*$:

- Assume that there is no such a path from x to y.

  Let $U = \{v \in V \mid (x, v) \in E^*\}$ be the set of all vertices that can be reached from x.

  Then U is closed under the edge relation E and $x \in U$, $y \notin U$.

- Assume that there is a path $(u_1, u_2, \dots, u_n)$ from x to y, i.e., $u_1 = x$, $u_n = y$ and $(u_i, u_{i+1}) \in E$ for all $i \in \{1, \dots, n-1\}$.

  Let $U \subseteq V$ be set of vertices that is closed under E with $x \in U$.

  Induction along i shows that $u_i \in U$ for all $i \in \{1, \dots, n\}$.

  Hence, we have $y = u_n \in U$.

MSO-definable languages

We want to use MSO-sentences in order to define (formal) languages.

For this, we first have to represent finite words by structures.

Let Σ be a finite alphabet.

A non-empty word $w = a_1 a_2 \cdots a_n$ ($n \geq 1$, $a_i \in \Sigma$) is identified with the structure

$$\mathcal{A}_w = (\{1, 2, \ldots, n\}, <, (P_a)_{a \in \Sigma}),$$

such that:

- $<$ is the ordinary order on $\{1, 2, \ldots, n\}$
- P_a is the unary relation $P_a = \{i \mid 1 \leq i \leq n, a_i = a\}$ (the set of all positions in the word w carrying the letter a)

MSO-definable languages

For the following example let the alphabet be $\Sigma = \{a, b\}$.

Example: For the word $w = abbaa$ we have

$$\mathcal{A}_w = (\{1,2,3,4,5\}, <, \underbrace{\{1,4,5\}}_{P_a}, \underbrace{\{2,3\}}_{P_b}).$$

In the following we identify the structure $\mathcal{A}_w$ with the word w.

A language $L \subseteq \Sigma^+$ of non-empty words is MSO-definable if there is an MSO-sentence F with $L = \{w \in \Sigma^+ \mid w \models F\}$.

MSO-definable languages

Example 1: The MSO-sentence

$$\exists x \exists y \exists z (\forall u (x \leq u \wedge u \leq z) \wedge P_a(x) \wedge P_b(y) \wedge P_a(z))$$

defines the language $a\Sigma^* b \Sigma^* a$.

Here, $x \leq u$ is an abbreviation for $x < u \vee x = u$.

Example 2: The MSO-sentence

$$\exists X \, (\exists x \exists y (\forall u (x \leq u \wedge u \leq y) \wedge x \in X \wedge \neg y \in X) \wedge \\ \forall x \forall y (y = x + 1 \rightarrow (x \in X \leftrightarrow y \notin X)))$$

defines the language $\{w \in \{a, b\}^+ \mid |w| \text{ is even}\}$.

Here, $y = x + 1$ is an abbreviation for the formula
$x < y \wedge \forall z (x \leq z \leq y \rightarrow (x = z \vee y = z))$.

Büchi's theorem

Theorem (Büchi, Elgot 1958 and Trachtenbrot 1958)

A language $L \subseteq \Sigma^+$ is MSO-definable if and only if L is regular.

Proof:

1. Let $L \subseteq \Sigma^+$ be regular. We show that L is MSO-definable.

Let $A = (Q, \Sigma, \delta, q_0, F)$ be a deterministic finite automaton (DFA) with $L(A) = L$, where

- Q is the finite set of states,
- $\delta : Q \times \Sigma \to Q$ is the transition mapping,
- $q_0 \in Q$ is the initial state, and
- $F \subseteq Q$ is the set of final states.

Büchi's theorem

Without loss of generality assume that $Q = \{1, \ldots, n\}$.

Then the following MSO-sentence defines the language $L = L(A)$:

$$
\begin{aligned}
&\exists X_1 \exists X_2 \cdots \exists X_n \\
&\quad \bigwedge_{p \neq q} X_p \cap X_q = \emptyset \ \wedge \ \forall x \bigvee_{q \in Q} x \in X_q \ \wedge \\
&\quad \exists x (\forall y (x \leq y) \wedge \bigvee_{a \in \Sigma} (P_a(x) \wedge x \in X_{\delta(q_0,a)})) \ \wedge \\
&\quad \exists x (\forall y (y \leq x) \wedge \bigvee_{q \in F} x \in X_q) \ \wedge \\
&\quad \forall x \forall y (y = x + 1 \rightarrow \bigvee_{q \in Q} \bigvee_{a \in \Sigma} (x \in X_q \wedge P_a(y) \wedge y \in X_{\delta(q,a)}))
\end{aligned}
$$

Here, $X_p \cap X_q = \emptyset$ is an abbreviation for $\neg \exists x (x \in X_p \wedge x \in X_q)$.

Büchi's theorem

Idea behind this formula: The existentially quantified set X_q is the set of all positions in the word where the DFA A arrives in state $q \in \{1, \ldots, n\}$.

- $\bigwedge_{p \neq q} X_p \cap X_q = \emptyset \ \wedge\ \forall x \bigvee_{q \in Q} x \in X_q$:

  At every position, the DFA arrives in exactly one state.

- $\exists x (\forall y (x \leq y) \wedge \bigvee_{a \in \Sigma} (P_a(x) \wedge x \in X_{\delta(q_0,a)}))$:

  If position 1 of the word ($x = 1$ in the above formula) carries the letter a, then the DFA arrives at position 1 in the state $\delta(q_0, a)$ (q_0 is the initial state).

Büchi's theorem

- $\exists x(\forall y(y \leq x) \wedge \bigvee_{q \in F} x \in X_q)$:

  At the last position of the word the DFA arrives in a final state.

- $\forall x \forall y(y = x + 1 \rightarrow \bigvee_{q \in Q} \bigvee_{a \in \Sigma} (x \in X_q \wedge P_a(y) \wedge y \in X_{\delta(q,a)}))$:

  If x and $y = x + 1$ are two successive positions in the word, where position y carries the letter a, and the DFA arrives at position x in state q, then the DFA arrives at position y in state $\delta(q, a)$.

Büchi's theorem

2. Let $L \subseteq \Sigma^+$ be MSO-definable. We show that L is regular.

Let $V \subseteq \mathrm{Var}_0 \cup \mathrm{Var}_1$ be a finite set of variables.

A non-empty word

$$w = (a_1, V_1)(a_2, V_2) \cdots (a_k, V_k) \in (\Sigma \times 2^V)^+$$

($k \geq 1$, $a_i \in \Sigma$, $V_k \subseteq V$) is called valid if for every variable $x \in V \cap \mathrm{Var}_0$ there is exactly one $1 \leq i \leq k$ with $x \in V_i$.

For a valid word w we define the mapping $f_w : V \to \{1, \ldots, k\} \cup 2^{\{1,\ldots,k\}}$ by

- $f_w(x) = i$ if $x \in V_i \cap \mathrm{Var}_0$ and
- $f_w(X) = \{i \mid X \in V_i\}$ for $X \in V \cap \mathrm{Var}_1$.

Büchi's theorem

We identify a valid word $w = (a_1, V_1)(a_2, V_2) \cdots (a_k, V_k)$ with the structure $\mathcal{A}_w = (\{1, \dots, k\}, I_w)$ where

- $I_w(x) = f_w(x)$ for $x \in V \cap \mathrm{Var}_0$,
- $I_w(X) = f_w(X)$ for $X \in V \cap \mathrm{Var}_1$,
- and I_w is defined for the predicate symbols $<$ and P_a $(a \in \Sigma)$ in the same way as in the structure $\mathcal{A}_v$ for $v = a_1 a_2 \cdots a_k$.

Hence, a valid word w defines an ordinary word $a_1 a_2 \dots a_k$ and in addition a valuation of the variables from V, where

- to every $x \in V \cap \mathrm{Var}_0$ a position $i \in \{1, \dots, k\}$ is assigned to and
- to every variable $X \in V \cap \mathrm{Var}_1$ a set of positions is assigned to.

Büchi's theorem

For an MSO-formula F with the free variables $\mathsf{free}(F)$ let $L(F)$ be the set of all non-empty valid words w over the alphabet $\Sigma \times 2^{\mathsf{free}(F)}$ such that $w \models F$.

Proof strategy: we construct for every formula F a finite automaton A_F for the language $L(F)$ using induction over the structure of F. At the end, we are only interested in the case $\mathsf{free}(F) = \emptyset$.

First, one can construct for every finite set of variables $V \subseteq \mathsf{Var}_0 \cup \mathsf{Var}_1$ an automaton A_V which accepts exactly the valid words from $(\Sigma \times 2^V)^*$.

The automaton A_V only has to check that every variable $x \in V \cap \mathsf{Var}_0$ appears at exactly one position of the input word.

For this, A_V stores in its state those variables from $V \cap \mathsf{Var}_0$ that were already seen.

Büchi's theorem

Now we come to the construction of the automaton A_F:

Case 1: $F = (x = y)$. Construct A_F such that

$$L(A_F) = (\Sigma \times \{\emptyset\})^* \, (\Sigma \times \{x, y\}) \, (\Sigma \times \{\emptyset\})^*.$$

Case 2: $F = (x < y)$. Construct A_F such that

$$L(A_F) = (\Sigma \times \{\emptyset\})^* \, (\Sigma \times \{x\}) \, (\Sigma \times \{\emptyset\})^* \, (\Sigma \times \{y\}) \, (\Sigma \times \{\emptyset\})^*.$$

Case 3: $F = P_a(x)$. Construct A_F such that

$$L(A_F) = (\Sigma \times \{\emptyset\})^* \, (a, \{x\}) \, (\Sigma \times \{\emptyset\})^*.$$

Case 4: $F = (x \in X)$. Construct A_F such that

$$L(A_F) = (\Sigma \times \{\emptyset, \{X\}\})^* \, (\Sigma \times \{x, X\}) \, (\Sigma \times \{\emptyset, \{X\}\})^*.$$

Büchi's theorem

For the following cases we use the known closure properties for regular languages (closure under boolean operations, homomorphisms and inverse homomorphisms) that we also used in the proof of the theorem of Khoussainov and Nerode (Slides 69–80).

Case 5: $F = \neg G$. Let $V = \text{free}(G)$. Construct A_F such that

$$L(A_F) = L(A_V) \setminus L(A_G).$$

Case 6: $F = G \vee H$.

Let $V_G = \text{free}(G)$, $V_H = \text{free}(H)$ and $V = \text{free}(F) = V_G \cup V_H$.

Büchi's theorem

Define homomorphisms $g : (\Sigma \times 2^V)^* \to (\Sigma \times 2^{V_G})^*$ and $h : (\Sigma \times 2^V)^* \to (\Sigma \times 2^{V_H})^*$ by

$$\begin{aligned} g(a, S) &= (a, S \cap V_G), \\ h(a, S) &= (a, S \cap V_H). \end{aligned}$$

Next, construct the automata A'_G and A'_H such that

$$\begin{aligned} L(A'_G) &= L(A_V) \cap g^{-1}(L(A_G)), \\ L(A'_H) &= L(A_V) \cap h^{-1}(L(A_H)). \end{aligned}$$

The automaton A_F is now constructed such that $L(A_F) = L(A'_G) \cup L(A'_H)$.

Büchi's theorem

Case 7: $F = \exists x\ G$.

Let $V = \mathrm{free}(G)$ and hence $\mathrm{free}(F) = V \setminus \{x\}$.

Define the homomorphism $f : (\Sigma \times 2^V)^* \to (\Sigma \times 2^{V\setminus\{x\}})^*$ by

$$f(a, S) = (a, S \setminus \{x\}).$$

Construct the automaton A_F such that $L(A_F) = f(L(A_G))$.

Case 8: $F = \exists X\ G$.

Let $V = \mathrm{free}(G)$ and hence $\mathrm{free}(F) = V \setminus \{X\}$.

Define the homomorphism $f : (\Sigma \times 2^V)^* \to (\Sigma \times 2^{V\setminus\{X\}})^*$ by

$$f(a, S) = (a, S \setminus \{X\}).$$

Then, construct the automaton A_F such that $L(A_F) = f(L(A_G))$.

This concludes the proof of Büchi's theorem. $\square$

Extensions of Büchi's theorem and applications

Büchi extended his result from finite words to infinite words, also known as ω-words.

An ω-word over the alphabet Σ is an infinite sequence $w = a_0 a_1 a_2 a_3 \cdots$ with $a_i \in \Sigma$ for all $i \geq 0$. It can be identified with the function $w : \mathbb{N} \to \Sigma$ with $w(i) = a_i$.

With Σ^ω we denote the set of all ω-words over the alphabet Σ.

An ω-language is a subset of Σ^ω.

We can identify the ω-word $w = a_0 a_1 a_2 a_3 \cdots$ with the structure

$$\mathcal{A}_w = (\mathbb{N}, <, (P_a)_{a \in \Sigma}),$$

where $P_a = \{i \in \mathbb{N} \mid a = a_i\}$.

Büchi automata

Syntactically, a Büchi automaton is exactly the same thing is a nondeterministic finite automaton.

Definition (nondeterministic Büchi automaton, NBA for short)

A nondeterministic Büchi automaton (over the alphabet Σ) is a tuple $B = (S, \Sigma, \delta, s_0, F)$ such that:

- S is a finite set of states,
- Σ is a finite alphabet,
- $\delta \subseteq S \times \Sigma \times S$ is the transition relation,
- s_0 is the initial state, and
- $F \subseteq S$ is the set of final states.

Büchi automata

Definition (runs and accepting runs)

Let $B = (S, \Sigma, \delta, s_0, F)$ be an NBA and $w = (a_1 a_2 a_3 \cdots) \in \Sigma^\omega$.

A run of B on w is an ωword $(s_0 s_1 s_2 \cdots) \in S^\omega$ with $(s_i, a_{i+1}, s_{i+1}) \in \delta$ for all $i \geq 0$.

This runs is an accepting run of B in w, if there are infinitely many $i \geq 0$ with $s_i \in F$ (or equivalently: there is a $q \in F$ such that $s_i = q$ for infinitely many i).

Definition (language accepted by an NBA)

The ω-language accepted by the NBA $B = (S, \Sigma, \delta, s_0, F)$ is

$$L(B) = \{w \in \Sigma^\omega \mid \text{ there is an accepting run of } B \text{ on } w\}.$$

Büchi automata and MSO

Let $L \subseteq \Sigma^*$ be an ω-language.

- L is MSO-definable if there exists an MSO-sentence F such that $L = \{w \in \Sigma^\omega \mid \mathcal{A}_w \models F\}$.
- L is ω-regular if there is an NBA B with $L = L(B)$.

Theorem (Büchi 1960)

An ω-language $L \subseteq \Sigma^\omega$ is MSO-definable if and only if L is ω-regular.

This result can be used to show:

Corollary

The MSO-theory of $(\mathbb{N}, <)$ is decidable.

Tree automata and MSO

The results of Büchi can be extended to infinite trees.

A Σ-labelled (binary) ω-tree is a structure

$$\mathcal{T} = (\{0,1\}^*, S_0, S_1, (P_a)_{a \in \Sigma})$$

where

- $S_0 = \{(w, w0) \mid w \in \{0,1\}^*\}$, $S_1 = \{(w, w1) \mid w \in \{0,1\}^*\}$, and
- every $P_a \subseteq \{0,1\}^*$ is a unary relation such $(P_a)_{a \in \Sigma}$ is a partition of $\{0,1\}^*$.

One can define a suitable automaton model for running on such trees, which yields the class of ω-regular tree languages.

The class of ω-regular tree languages then coincides with the class of MSO-definable ω-tree languages.

More decidable MSO theories

A corollary is then the following famous (and very difficult) result:

Corollary (Rabin 1969)

The MSO-theory of the infinite binary tree $T_2 = (\{0,1\}^*, S_0, S_1, \leq)$ (see slide 81) is decidable.

From this result one easily obtains:

Corollary

The MSO-theory of $(\mathbb{Q}, \leq)$ is decidable.

Motivation

Recall that

finite automata $=$ ($\exists$)MSO (on words and trees)

Motivation

Recall that

$$\text{finite automata} = (\exists)\text{MSO (on words and trees)}$$

The basic message of the rest of the lecture:

$$\text{NP} = \exists\text{SO (on structures over any relational signature)}$$

Motivation

Recall that

$$\text{finite automata} = (\exists)\text{MSO (on words and trees)}$$

The basic message of the rest of the lecture:

$$\text{NP} = \exists\text{SO (on structures over any relational signature)}$$

Instead of arbitrary structures, we will restrict to graphs; the generalization to arbitrary relational structures is just a technicality.

Graphs and their encodings

Conventions:

- Graphs will be always finite and directed.
- The set of nodes will be an initial segment of the natural numbers.

Graphs and their encodings

Conventions:

- Graphs will be always finite and directed.
- The set of nodes will be an initial segment of the natural numbers.

Hence, a graph is a pair $\mathcal{G} = (\{1, \ldots, n\}, E)$ with $E \subseteq \{1, \ldots, n\} \times \{1, \ldots, n\}$.

Graphs and their encodings

Conventions:

- Graphs will be always finite and directed.
- The set of nodes will be an initial segment of the natural numbers.

Hence, a graph is a pair $\mathcal{G} = (\{1, \ldots, n\}, E)$ with $E \subseteq \{1, \ldots, n\} \times \{1, \ldots, n\}$.

$\mathcal{G}$ can be represented by its adjacency matrix $M_{\mathcal{G}} = (a_{i,j})_{1 \le i,j \le n}$, where

$$a_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{else} \end{cases}$$

Graphs and their encodings

Conventions:

- Graphs will be always finite and directed.
- The set of nodes will be an initial segment of the natural numbers.

Hence, a graph is a pair $\mathcal{G} = (\{1, \dots, n\}, E)$ with $E \subseteq \{1, \dots, n\} \times \{1, \dots, n\}$.

$\mathcal{G}$ can be represented by its adjacency matrix $M_{\mathcal{G}} = (a_{i,j})_{1 \le i,j \le n}$, where

$$a_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{else} \end{cases}$$

We can encode $M_{\mathcal{G}}$ and hence $\mathcal{G}$ by the following bit string of length n^2:

$$\text{code}(\mathcal{G}) = a_{1,1} a_{1,2} \cdots a_{1,n} a_{2,1} a_{2,2} \cdots a_{2,n} \cdots a_{n,1} a_{n,2} \cdots a_{n,n}$$

Graph properties

A graph property is a set of graphs $\mathcal{A}$ that is closed under isomorphism:

$$\mathcal{G}_1 \cong \mathcal{G}_2 \quad \Rightarrow \quad (\mathcal{G}_1 \in \mathcal{A} \Leftrightarrow \mathcal{G}_2 \in \mathcal{A})$$

Graph properties

A graph property is a set of graphs $\mathcal{A}$ that is closed under isomorphism:

$$\mathcal{G}_1 \cong \mathcal{G}_2 \quad \Rightarrow \quad (\mathcal{G}_1 \in \mathcal{A} \Leftrightarrow \mathcal{G}_2 \in \mathcal{A})$$

We will present two formalism for specifying graph properties:

- (Existential) second-order logic
- (Nondeterministic polynomial time) Turing machines.

Second-order logic over graphs

Let us fix countably infinite sets of variables $\mathsf{Var}_0, \mathsf{Var}_1, \mathsf{Var}_2, \ldots$.

Second-order logic over graphs

Let us fix countably infinite sets of variables $\mathsf{Var}_0, \mathsf{Var}_1, \mathsf{Var}_2, \ldots$.

- Var_0 is the set of first-order variables $x, y, z, x_1, x_2, \ldots$, ranging over nodes of a graph.

Second-order logic over graphs

Let us fix countably infinite sets of variables $\mathrm{Var}_0, \mathrm{Var}_1, \mathrm{Var}_2, \ldots$.

- Var_0 is the set of first-order variables $x, y, z, x_1, x_2, \ldots$, ranging over nodes of a graph.
- Var_k for $k \geq 1$ is the set of k-ary second-order variables $R, P, Q, R_1, R_2, \ldots$, ranging over k-ary relations on nodes.

Second-order logic over graphs

Let us fix countably infinite sets of variables $\mathrm{Var}_0, \mathrm{Var}_1, \mathrm{Var}_2, \ldots$.

- Var_0 is the set of first-order variables $x, y, z, x_1, x_2, \ldots$, ranging over nodes of a graph.
- Var_k for $k \geq 1$ is the set of k-ary second-order variables $R, P, Q, R_1, R_2, \ldots$, ranging over k-ary relations on nodes.

The set of second-order formulas (SO-formulas) is inductively defined as follows, where E denotes the edge relation of a graph:

Second-order logic over graphs

Let us fix countably infinite sets of variables $\mathsf{Var}_0, \mathsf{Var}_1, \mathsf{Var}_2, \ldots$.

- Var_0 is the set of first-order variables $x, y, z, x_1, x_2, \ldots$, ranging over nodes of a graph.
- Var_k for $k \geq 1$ is the set of k-ary second-order variables $R, P, Q, R_1, R_2, \ldots$, ranging over k-ary relations on nodes.

The set of second-order formulas (SO-formulas) is inductively defined as follows, where E denotes the edge relation of a graph:

- For all $x, y \in \mathsf{Var}_0$, $(x = y)$ and $E(x, y)$ are SO-formulas.

Second-order logic over graphs

Let us fix countably infinite sets of variables $\mathrm{Var}_0, \mathrm{Var}_1, \mathrm{Var}_2, \ldots$.

- Var_0 is the set of first-order variables $x, y, z, x_1, x_2, \ldots$, ranging over nodes of a graph.
- Var_k for $k \geq 1$ is the set of k-ary second-order variables $R, P, Q, R_1, R_2, \ldots$, ranging over k-ary relations on nodes.

The set of second-order formulas (SO-formulas) is inductively defined as follows, where E denotes the edge relation of a graph:

- For all $x, y \in \mathrm{Var}_0$, $(x = y)$ and $E(x, y)$ are SO-formulas.
- For all $R \in \mathrm{Var}_k$ and all $x_1, \ldots, x_k \in \mathrm{Var}_0$, $R(x_1, \ldots, x_k)$ is an SO-formula.

Second-order logic over graphs

Let us fix countably infinite sets of variables $\mathrm{Var}_0, \mathrm{Var}_1, \mathrm{Var}_2, \ldots$.

- Var_0 is the set of first-order variables $x, y, z, x_1, x_2, \ldots$, ranging over nodes of a graph.
- Var_k for $k \geq 1$ is the set of k-ary second-order variables $R, P, Q, R_1, R_2, \ldots$, ranging over k-ary relations on nodes.

The set of second-order formulas (SO-formulas) is inductively defined as follows, where E denotes the edge relation of a graph:

- For all $x, y \in \mathrm{Var}_0$, $(x = y)$ and $E(x, y)$ are SO-formulas.
- For all $R \in \mathrm{Var}_k$ and all $x_1, \ldots, x_k \in \mathrm{Var}_0$, $R(x_1, \ldots, x_k)$ is an SO-formula.
- If F and G are SO-formulas, then also $\neg F$, $F \wedge G$, and $F \vee G$ are SO-formulas.

Second-order logic over graphs

Let us fix countably infinite sets of variables $\mathrm{Var}_0, \mathrm{Var}_1, \mathrm{Var}_2, \ldots$.

- Var_0 is the set of first-order variables $x, y, z, x_1, x_2, \ldots$, ranging over nodes of a graph.
- Var_k for $k \geq 1$ is the set of k-ary second-order variables $R, P, Q, R_1, R_2, \ldots$, ranging over k-ary relations on nodes.

The set of second-order formulas (SO-formulas) is inductively defined as follows, where E denotes the edge relation of a graph:

- For all $x, y \in \mathrm{Var}_0$, $(x = y)$ and $E(x, y)$ are SO-formulas.
- For all $R \in \mathrm{Var}_k$ and all $x_1, \ldots, x_k \in \mathrm{Var}_0$, $R(x_1, \ldots, x_k)$ is an SO-formula.
- If F and G are SO-formulas, then also $\neg F$, $F \wedge G$, and $F \vee G$ are SO-formulas.
- If F is an SO-formula and $x \in \mathrm{Var}_0$ then also $\exists x : F$ and $\forall x : F$ are SO-formulas.

Second-order logic over graphs

Let us fix countably infinite sets of variables $\mathsf{Var}_0, \mathsf{Var}_1, \mathsf{Var}_2, \ldots$.

- Var_0 is the set of first-order variables $x, y, z, x_1, x_2, \ldots$, ranging over nodes of a graph.
- Var_k for $k \geq 1$ is the set of k-ary second-order variables $R, P, Q, R_1, R_2, \ldots$, ranging over k-ary relations on nodes.

The set of second-order formulas (SO-formulas) is inductively defined as follows, where E denotes the edge relation of a graph:

- For all $x, y \in \mathsf{Var}_0$, $(x = y)$ and $E(x, y)$ are SO-formulas.
- For all $R \in \mathsf{Var}_k$ and all $x_1, \ldots, x_k \in \mathsf{Var}_0$, $R(x_1, \ldots, x_k)$ is an SO-formula.
- If F and G are SO-formulas, then also $\neg F$, $F \wedge G$, and $F \vee G$ are SO-formulas.
- If F is an SO-formula and $x \in \mathsf{Var}_0$ then also $\exists x : F$ and $\forall x : F$ are SO-formulas.
- If F is an SO-formula and $R \in \mathsf{Var}_k$ for some $k \geq 1$ then also $\exists R : F$ and $\forall R : F$ are SO-formulas.

Second-order logic over graphs

An SO-sentence is an SO-formula without free (first-order or second-order) variables.

Second-order logic over graphs

An SO-sentence is an SO-formula without free (first-order or second-order) variables.

A first-order formula (FO-formula) is an SO-formula without quantifications over second-order variables.

Second-order logic over graphs

An SO-sentence is an SO-formula without free (first-order or second-order) variables.

A first-order formula (FO-formula) is an SO-formula without quantifications over second-order variables.

An existential second-order formula (∃SO-formula) is an SO-formula of the form

$$\exists R_1 \exists R_2 \cdots \exists R_k : F,$$

where $R_1, R_2 \dots, R_k$ are second-order variables (of arbitrary arity) and F is an FO-formula.

Second-order logic over graphs

An SO-sentence is an SO-formula without free (first-order or second-order) variables.

A first-order formula (FO-formula) is an SO-formula without quantifications over second-order variables.

An existential second-order formula ($\exists$SO-formula) is an SO-formula of the form

$$\exists R_1 \exists R_2 \cdots \exists R_k : F,$$

where $R_1, R_2 \dots, R_k$ are second-order variables (of arbitrary arity) and F is an FO-formula.

For an SO-sentence F and a graph $\mathcal{G}$ we write $\mathcal{G} \models F$ if F is true in $\mathcal{G}$.

Second-order logic over graphs

An SO-sentence is an SO-formula without free (first-order or second-order) variables.

A first-order formula (FO-formula) is an SO-formula without quantifications over second-order variables.

An existential second-order formula ($\exists$SO-formula) is an SO-formula of the form

$$\exists R_1 \exists R_2 \cdots \exists R_k : F,$$

where $R_1, R_2 \ldots, R_k$ are second-order variables (of arbitrary arity) and F is an FO-formula.

For an SO-sentence F and a graph $\mathcal{G}$ we write $\mathcal{G} \models F$ if F is true in $\mathcal{G}$.

Note: $\{\mathcal{G} \mid \mathcal{G} \models F\}$ is a graph property.

Second-order logic over graphs

An SO-sentence is an SO-formula without free (first-order or second-order) variables.

A first-order formula (FO-formula) is an SO-formula without quantifications over second-order variables.

An existential second-order formula ($\exists$SO-formula) is an SO-formula of the form

$$\exists R_1 \exists R_2 \cdots \exists R_k : F,$$

where $R_1, R_2 \ldots, R_k$ are second-order variables (of arbitrary arity) and F is an FO-formula.

For an SO-sentence F and a graph $\mathcal{G}$ we write $\mathcal{G} \models F$ if F is true in $\mathcal{G}$.

Note: $\{\mathcal{G} \mid \mathcal{G} \models F\}$ is a graph property.

Let $\mathcal{L}$ be any logic (e.g. FO, SO, $\exists$SO). A graph property $\mathcal{A}$ is $\mathcal{L}$-definable if there exists an $\mathcal{L}$-sentence F such that $\mathcal{A} = \{\mathcal{G} \mid \mathcal{G} \models F\}$.

Examples

The following $\exists$SO-sentence states that a graph (with edge relation E) can be colored with 3 colors:

$$
\begin{aligned}
&\exists C_1 \, \exists C_2 \, \exists C_3 : \\
&\qquad \bigwedge_{1 \le i < j \le 3} \neg \exists x (C_i(x) \wedge C_j(x)) \; \wedge \\
&\qquad \forall x \bigvee_{1 \le i \le 3} C_i(x) \; \wedge \\
&\qquad \forall x \, \forall y : E(x,y) \to \bigwedge_{1 \le i \le 3} \neg (C_i(x) \wedge C_i(y))
\end{aligned}
$$

Examples

The following ∃SO-sentence states that a graph has an even number of nodes:

$$
\begin{aligned}
&\exists \leq \exists S \; \exists A : \\
&\quad \forall x, y, z : x \leq x \wedge (x \leq y \leq x \rightarrow x = y) \wedge \\
&\qquad\qquad (x \leq y \leq z \rightarrow x \leq z) \wedge (x \leq y \vee y \leq x) \wedge \\
&\quad \forall x, y : S(x, y) \leftrightarrow (x < y \wedge \neg \exists z (x < z < y)) \wedge \\
&\quad \forall x : (\forall y : x \leq y) \rightarrow A(x) \wedge \\
&\quad \forall x : (\forall y : y \leq x) \rightarrow \neg A(x) \wedge \\
&\quad \forall x, y : S(x, y) \rightarrow (A(x) \leftrightarrow \neg A(y))
\end{aligned}
$$

Turing machines: Definition

A nondeterministic 1-tape Turing machine is a 6-tuple $M = (Q, \Gamma, \delta, q_0, q_Y, q_N)$, where:

Turing machines: Definition

A nondeterministic 1-tape Turing machine is a 6-tuple $M = (Q, \Gamma, \delta, q_0, q_Y, q_N)$, where:

- Q is the finite set of states,

Turing machines: Definition

A nondeterministic 1-tape Turing machine is a 6-tuple $M = (Q, \Gamma, \delta, q_0, q_Y, q_N)$, where:

- Q is the finite set of states,
- $q_0 \in Q$ is the initial state,

Turing machines: Definition

A nondeterministic 1-tape Turing machine is a 6-tuple $M = (Q, \Gamma, \delta, q_0, q_Y, q_N)$, where:

- Q is the finite set of states,
- $q_0 \in Q$ is the initial state,
- $q_Y \in Q$ (resp., q_N) is the accepting (resp., rejecting) state,

Turing machines: Definition

A nondeterministic 1-tape Turing machine is a 6-tuple $M = (Q, \Gamma, \delta, q_0, q_Y, q_N)$, where:

- Q is the finite set of states,
- $q_0 \in Q$ is the initial state,
- $q_Y \in Q$ (resp., q_N) is the accepting (resp., rejecting) state,
- Γ is the finite tape alphabet with $\{0, 1, \Box\} \subseteq \Gamma$,

Turing machines: Definition

A nondeterministic 1-tape Turing machine is a 6-tuple $M = (Q, \Gamma, \delta, q_0, q_Y, q_N)$, where:

- Q is the finite set of states,
- $q_0 \in Q$ is the initial state,
- $q_Y \in Q$ (resp., q_N) is the accepting (resp., rejecting) state,
- Γ is the finite tape alphabet with $\{0, 1, \Box\} \subseteq \Gamma$,
- $\delta : (Q \setminus \{q_Y, q_N\}) \times \Gamma \to 2^{Q \times \Gamma \times \{-1,0,1\}} \setminus \{\emptyset\}$ is the transition function.

Turing machines: Definition

A nondeterministic 1-tape Turing machine is a 6-tuple $M = (Q, \Gamma, \delta, q_0, q_Y, q_N)$, where:

- Q is the finite set of states,
- $q_0 \in Q$ is the initial state,
- $q_Y \in Q$ (resp., q_N) is the accepting (resp., rejecting) state,
- Γ is the finite tape alphabet with $\{0, 1, \Box\} \subseteq \Gamma$,
- $\delta : (Q \setminus \{q_Y, q_N\}) \times \Gamma \to 2^{Q \times \Gamma \times \{-1,0,1\}} \setminus \{\emptyset\}$ is the transition function.

M is equipped with an infinite tape, whose cells are indexed with $1, 2, 3, \ldots$.

Turing machines: Definition

A nondeterministic 1-tape Turing machine is a 6-tuple $M = (Q, \Gamma, \delta, q_0, q_Y, q_N)$, where:

- Q is the finite set of states,
- $q_0 \in Q$ is the initial state,
- $q_Y \in Q$ (resp., q_N) is the accepting (resp., rejecting) state,
- Γ is the finite tape alphabet with $\{0, 1, \Box\} \subseteq \Gamma$,
- $\delta : (Q \setminus \{q_Y, q_N\}) \times \Gamma \to 2^{Q \times \Gamma \times \{-1,0,1\}} \setminus \{\emptyset\}$ is the transition function.

M is equipped with an infinite tape, whose cells are indexed with $1, 2, 3, \ldots$.

Every cell contains a symbol from Γ.

Turing machines: Definition

A nondeterministic 1-tape Turing machine is a 6-tuple $M = (Q, \Gamma, \delta, q_0, q_Y, q_N)$, where:

- Q is the finite set of states,
- $q_0 \in Q$ is the initial state,
- $q_Y \in Q$ (resp., q_N) is the accepting (resp., rejecting) state,
- Γ is the finite tape alphabet with $\{0, 1, \Box\} \subseteq \Gamma$,
- $\delta : (Q \setminus \{q_Y, q_N\}) \times \Gamma \to 2^{Q \times \Gamma \times \{-1,0,1\}} \setminus \{\emptyset\}$ is the transition function.

M is equipped with an infinite tape, whose cells are indexed with $1, 2, 3, \ldots$.

Every cell contains a symbol from Γ.

$\Box$ is the blank symbol, 0 and 1 are the input symbols.

Turing machines: Definition

That $(q, b, d) \in \delta(p, a)$ means: If current state is p and the current tape cell $k \in \mathbb{N}$ to which the head points contains symbol a, then M:

1. changes the symbol of cell k to b,
2. moves the tape head to cell $k + d$, and
3. enters state q.

Turing machines: Definition

That $(q, b, d) \in \delta(p, a)$ means: If current state is p and the current tape cell $k \in \mathbb{N}$ to which the head points contains symbol a, then M:

1. changes the symbol of cell k to b,
2. moves the tape head to cell $k + d$, and
3. enters state q.

If the transition function δ is required to be of type

$$\delta : (Q \setminus \{q_Y, q_N\}) \times \Gamma \to Q \times \Gamma \times \{-1, 0, 1\}$$

then M is a deterministic Turing machine.

Turing machines: Configurations

Configurations of M can be represented by words $u(q,a)v$ with $u, v \in \Gamma^*$, $q \in Q$, and $a \in \Gamma$:

Turing machines: Configurations

Configurations of M can be represented by words $u(q,a)v$ with $u, v \in \Gamma^*$, $q \in Q$, and $a \in \Gamma$:

- If $u = a_1 a_2 \cdots a_n$ and $v = b_1 b_2 \cdots b_m$ $(n, m \geq 0)$ then the current tape content is

$$a_1 a_2 \cdots a_n \, a \, b_1 \cdots b_m \, \Box \, \Box \, \Box \cdots$$

Turing machines: Configurations

Configurations of M can be represented by words $u(q,a)v$ with $u, v \in \Gamma^*$, $q \in Q$, and $a \in \Gamma$:

- If $u = a_1 a_2 \cdots a_n$ and $v = b_1 b_2 \cdots b_m$ $(n, m \geq 0)$ then the current tape content is

$$a_1 a_2 \cdots a_n \, a \, b_1 \cdots b_m \, \Box \, \Box \, \Box \cdots$$

- The tape head points to cell $n + 1$ (which contains a).

Turing machines: Configurations

Configurations of M can be represented by words $u(q,a)v$ with $u, v \in \Gamma^*$, $q \in Q$, and $a \in \Gamma$:

- If $u = a_1 a_2 \cdots a_n$ and $v = b_1 b_2 \cdots b_m$ $(n, m \geq 0)$ then the current tape content is

$$a_1 a_2 \cdots a_n \, a \, b_1 \cdots b_m \, \Box \, \Box \, \Box \cdots$$

- The tape head points to cell $n+1$ (which contains a).
- q is the current state.

Turing machines: Configurations

Configurations of M can be represented by words $u(q, a)v$ with $u, v \in \Gamma^*$, $q \in Q$, and $a \in \Gamma$:

- If $u = a_1 a_2 \cdots a_n$ and $v = b_1 b_2 \cdots b_m$ $(n, m \geq 0)$ then the current tape content is
$$a_1 a_2 \cdots a_n \, a \, b_1 \cdots b_m \, \Box \, \Box \, \Box \cdots$$
- The tape head points to cell $n + 1$ (which contains a).
- q is the current state.

Note: $u(q, a)v, u(q, a)v\Box, u(q, a)v\Box\Box, \ldots$ all represent the same configuration of M.

Turing machines: Configurations

Configurations of M can be represented by words $u(q,a)v$ with $u, v \in \Gamma^*$, $q \in Q$, and $a \in \Gamma$:

- If $u = a_1 a_2 \cdots a_n$ and $v = b_1 b_2 \cdots b_m$ $(n, m \geq 0)$ then the current tape content is

$$a_1 a_2 \cdots a_n \, a \, b_1 \cdots b_m \, \Box \, \Box \, \Box \cdots$$

- The tape head points to cell $n+1$ (which contains a).
- q is the current state.

Note: $u(q,a)v, u(q,a)v\Box, u(q,a)v\Box\Box, \ldots$ all represent the same configuration of M.

The initial configuration for an input $w = a_1 \cdots a_n \in \{0,1\}^*$ is:

$$\text{init}(w) = \begin{cases} (q_0, a_1)a_2 \cdots a_n & \text{if } n \geq 1 \\ (q_0, \Box) & \text{if } n = 0. \end{cases}$$

Turing machines: Computations

For two configurations α and β we write $\alpha \vdash_M \beta$ if M can transform α into β.

Turing machines: Computations

For two configurations α and β we write $\alpha \vdash_M \beta$ if M can transform α into β.

An M-computation of length t on input w is a sequence of configurations

$$\mathsf{init}(w) \vdash_M \alpha_1 \vdash_M \alpha_2 \vdash_M \cdots \vdash_M \alpha_t.$$

Turing machines: Computations

For two configurations α and β we write $\alpha \vdash_M \beta$ if M can transform α into β.

An M-computation of length t on input w is a sequence of configurations

$$\mathrm{init}(w) \vdash_M \alpha_1 \vdash_M \alpha_2 \vdash_M \cdots \vdash_M \alpha_t.$$

This computation is accepting if $\alpha_t = u(q_Y, a)v$.

Turing machines: Computations

For two configurations α and β we write $\alpha \vdash_M \beta$ if M can transform α into β.

An *M-computation of length t on input w* is a sequence of configurations

$$\mathrm{init}(w) \vdash_M \alpha_1 \vdash_M \alpha_2 \vdash_M \cdots \vdash_M \alpha_t.$$

This computation is *accepting* if $\alpha_t = u(q_Y, a)v$.

The input word $w \in \{0,1\}^*$ is accepted by M if there exists an accepting M-computation on input w.

Turing machines: Computations

For two configurations α and β we write $\alpha \vdash_M \beta$ if M can transform α into β.

An M-computation of length t on input w is a sequence of configurations

$$\mathrm{init}(w) \vdash_M \alpha_1 \vdash_M \alpha_2 \vdash_M \cdots \vdash_M \alpha_t.$$

This computation is accepting if $\alpha_t = u(q_Y, a)v$.

The input word $w \in \{0,1\}^*$ is accepted by M if there exists an accepting M-computation on input w.

Let $f : \mathbb{N} \to \mathbb{N}$ be monotone. The machine M is $f(n)$-time bounded if every M-computation on an input w with $|w| = n$ has length at most $f(n)$.

The classes P and NP

The class NP is the set of all languages $L \subseteq \{0,1\}^*$ for which there exists a nondeterministic Turing machine M with:

- M is $p(n)$-time bounded for some polynomial $p(n)$.
- For every word $w \in \{0,1\}^*$: w is accepted by M if and only if $w \in L$.

The classes P and NP

The class NP is the set of all languages $L \subseteq \{0,1\}^*$ for which there exists a nondeterministic Turing machine M with:

- M is $p(n)$-time bounded for some polynomial $p(n)$.
- For every word $w \in \{0,1\}^*$: w is accepted by M if and only if $w \in L$.

The class P is the set of all languages $L \subseteq \{0,1\}^*$ for which there exists a deterministic Turing machine M with:

- M is $p(n)$-time bouned for some polynomial $p(n)$.
- For every word $w \in \{0,1\}^*$: w is accepted by M if and only if $w \in L$.

The classes P and NP

The class NP is the set of all languages $L \subseteq \{0,1\}^*$ for which there exists a nondeterministic Turing machine M with:

- M is $p(n)$-time bounded for some polynomial $p(n)$.
- For every word $w \in \{0,1\}^*$: w is accepted by M if and only if $w \in L$.

The class P is the set of all languages $L \subseteq \{0,1\}^*$ for which there exists a deterministic Turing machine M with:

- M is $p(n)$-time bouned for some polynomial $p(n)$.
- For every word $w \in \{0,1\}^*$: w is accepted by M if and only if $w \in L$.

Clearly, $\mathsf{P} \subseteq \mathsf{NP}$. Whether $\mathsf{P} = \mathsf{NP}$ holds, is the most important open question in TCS (and one of the most important open problems in Mathematics).

Accepting graph properties by Turing machines

We say that a Turing machine M accepts a graph property if the following holds:

- If $w \in \{0,1\}^*$ and $|w|$ is not a square, then w is not accepted by M.
- If $\mathcal{G}_1$ and $\mathcal{G}_2$ are isomorphic graphs then $\text{code}(\mathcal{G}_1)$ is accepted by M if and only if $\text{code}(\mathcal{G}_2)$ is accepted by M.

Fagin's Theorem

Theorem (Ronald Fagin, 1974)

Let $\mathcal{A}$ be a graph property. Then $\mathcal{A} \in \mathsf{NP}$ if and only if $\mathcal{A}$ is $\exists$SO-definable.

Fagin's Theorem

Theorem (Ronald Fagin, 1974)

Let $\mathcal{A}$ be a graph property. Then $\mathcal{A} \in \mathsf{NP}$ if and only if $\mathcal{A}$ is $\exists\mathsf{SO}$-definable.

Proof:

Fagin's Theorem

Theorem (Ronald Fagin, 1974)

Let $\mathcal{A}$ be a graph property. Then $\mathcal{A} \in \mathsf{NP}$ if and only if $\mathcal{A}$ is $\exists$SO-definable.

Proof:

(1) Assume that $\mathcal{A} = \{\mathcal{G} \mid \mathcal{G} \models F\}$, where F is an $\exists$SO-sentence.

Fagin's Theorem

Theorem (Ronald Fagin, 1974)

Let $\mathcal{A}$ be a graph property. Then $\mathcal{A} \in \mathsf{NP}$ if and only if $\mathcal{A}$ is $\exists$SO-definable.

Proof:

(1) Assume that $\mathcal{A} = \{\mathcal{G} \mid \mathcal{G} \models F\}$, where F is an $\exists$SO-sentence.

Let $F = \exists R_1 \exists R_2 \cdots \exists R_k Q_1 x_1 \cdots Q_\ell x_\ell : G$ where R_i is a k_i-ary second order variable, x_i is a first-order variable, $Q_i \in \{\forall, \exists\}$ and G is quantifier-free.

Fagin's Theorem

Theorem (Ronald Fagin, 1974)

Let $\mathcal{A}$ be a graph property. Then $\mathcal{A} \in \mathsf{NP}$ if and only if $\mathcal{A}$ is $\exists$SO-definable.

Proof:

(1) Assume that $\mathcal{A} = \{\mathcal{G} \mid \mathcal{G} \models F\}$, where F is an $\exists$SO-sentence.

Let $F = \exists R_1 \exists R_2 \cdots \exists R_k Q_1 x_1 \cdots Q_\ell x_\ell : G$ where R_i is a k_i-ary second order variable, x_i is a first-order variable, $Q_i \in \{\forall, \exists\}$ and G is quantifier-free.

For a graph $\mathcal{G} = (V, E)$ with $|V| = n$, a nondeterministic Turing machine can guess in time $\sum_{i=1}^{k} n^{k_i}$ relations R_i of arity k_i $(1 \le i \le k)$.

Fagin's Theorem

Theorem (Ronald Fagin, 1974)

Let $\mathcal{A}$ be a graph property. Then $\mathcal{A} \in \mathsf{NP}$ if and only if $\mathcal{A}$ is $\exists$SO-definable.

Proof:

(1) Assume that $\mathcal{A} = \{\mathcal{G} \mid \mathcal{G} \models F\}$, where F is an $\exists$SO-sentence.

Let $F = \exists R_1 \exists R_2 \cdots \exists R_k Q_1 x_1 \cdots Q_\ell x_\ell : G$ where R_i is a k_i-ary second order variable, x_i is a first-order variable, $Q_i \in \{\forall, \exists\}$ and G is quantifier-free.

For a graph $\mathcal{G} = (V, E)$ with $|V| = n$, a nondeterministic Turing machine can guess in time $\sum_{i=1}^{k} n^{k_i}$ relations R_i of arity k_i $(1 \leq i \leq k)$.

Then, $Q_1 x_1 \cdots Q_\ell x_\ell : G$ can be checked in time $n^\ell \cdot \mathrm{poly}(|G|, n)$.

Fagin's Theorem: From NP to $\exists$SO

(2) Assume that $\mathcal{A}$ belongs to NP.

Fagin's Theorem: From NP to $\exists$SO

(2) Assume that $\mathcal{A}$ belongs to NP.

Let $M = (Q, \Gamma, \delta, q_0, q_Y, q_N)$ be a nondeterministic $p(n)$-time bounded Turing machine that accepts $\mathcal{A}$, where $p(n)$ is a polynomial.

Fagin's Theorem: From NP to $\exists$SO

(2) Assume that $\mathcal{A}$ belongs to NP.

Let $M = (Q, \Gamma, \delta, q_0, q_Y, q_N)$ be a nondeterministic $p(n)$-time bounded Turing machine that accepts $\mathcal{A}$, where $p(n)$ is a polynomial.

W.l.o.g. we can assume that for all $(q, a) \in (Q \setminus \{q_Y, q_N\}) \times \Gamma$ we have $|\delta(q, a)| = 2$.

Fagin's Theorem: From NP to $\exists$SO

(2) Assume that $\mathcal{A}$ belongs to NP.

Let $M = (Q, \Gamma, \delta, q_0, q_Y, q_N)$ be a nondeterministic $p(n)$-time bounded Turing machine that accepts $\mathcal{A}$, where $p(n)$ is a polynomial.

W.l.o.g. we can assume that for all $(q, a) \in (Q \setminus \{q_Y, q_N\}) \times \Gamma$ we have $|\delta(q, a)| = 2$. Let

$$\begin{aligned}\delta(q, a) = \{&(\rho_0(q, a), \alpha_0(q, a), \delta_0(q, a)),\\ &(\rho_1(q, a), \alpha_1(q, a), \delta_1(q, a))\}.\end{aligned}$$

Fagin's Theorem: From NP to $\exists$SO

(2) Assume that $\mathcal{A}$ belongs to NP.

Let $M = (Q, \Gamma, \delta, q_0, q_Y, q_N)$ be a nondeterministic $p(n)$-time bounded Turing machine that accepts $\mathcal{A}$, where $p(n)$ is a polynomial.

W.l.o.g. we can assume that for all $(q, a) \in (Q \setminus \{q_Y, q_N\}) \times \Gamma$ we have $|\delta(q, a)| = 2$. Let

$$\begin{aligned}\delta(q, a) = \{&(\rho_0(q, a), \alpha_0(q, a), \delta_0(q, a)),\\ &(\rho_1(q, a), \alpha_1(q, a), \delta_1(q, a))\}.\end{aligned}$$

For technical reasons, we set for all $a \in \Gamma$ and $i \in \{0, 1\}$:

$$\rho_i(q_Y, a) = q_Y, \quad \alpha_i(q_Y, a) = a, \qquad \delta_i(q_Y, a) = 0$$

Fagin's Theorem: From NP to ∃SO

It suffices to come up with an ∃SO-sentence F such that:

$$\exists c > 0\ \forall n \geq c\ \forall \text{ graphs } \mathcal{G} \text{ with } n \text{ nodes}: \\ \mathcal{G} \models F \Leftrightarrow M \text{ accepts } \mathrm{code}(\mathcal{G})$$

Fagin's Theorem: From NP to $\exists$SO

It suffices to come up with an $\exists$SO-sentence F such that:

$$\begin{aligned}\exists c > 0\ \forall n \geq c\ \forall \text{ graphs } \mathcal{G} \text{ with } n \text{ nodes} :\\ \mathcal{G} \models F \Leftrightarrow M \text{ accepts } \mathrm{code}(\mathcal{G})\end{aligned}$$

There are constants $c, k > 0$ such that $\forall n \geq c : p(n^2) \leq n^k - 1$.

Fagin's Theorem: From NP to $\exists$SO

It suffices to come up with an $\exists$SO-sentence F such that:

$$\exists c > 0\ \forall n \geq c\ \forall \text{ graphs } \mathcal{G} \text{ with } n \text{ nodes} : \\ \mathcal{G} \models F \Leftrightarrow M \text{ accepts code}(\mathcal{G})$$

There are constants $c, k > 0$ such that $\forall n \geq c : p(n^2) \leq n^k - 1$.

Note: $p(n^2)$ bounds the running time of M on an input code($\mathcal{G}$) for a graph $\mathcal{G} = (V, E)$ with $n = |V|$ nodes.

Fagin's Theorem: From NP to ∃SO

It suffices to come up with an ∃SO-sentence F such that:

$$\exists c > 0\ \forall n \geq c\ \forall \text{ graphs } \mathcal{G} \text{ with } n \text{ nodes}: \\ \mathcal{G} \models F \Leftrightarrow M \text{ accepts code}(\mathcal{G})$$

There are constants $c, k > 0$ such that $\forall n \geq c : p(n^2) \leq n^k - 1$.

Note: $p(n^2)$ bounds the running time of M on an input code($\mathcal{G}$) for a graph $\mathcal{G} = (V, E)$ with $n = |V|$ nodes.

Let $\mathcal{G} = (V, E)$ be a graph with $n = |V| \geq c$ nodes.

Fagin's Theorem: From NP to $\exists$SO

It suffices to come up with an $\exists$SO-sentence F such that:

$$\exists c > 0 \, \forall n \geq c \, \forall \text{ graphs } \mathcal{G} \text{ with } n \text{ nodes}: \\ \mathcal{G} \models F \Leftrightarrow M \text{ accepts code}(\mathcal{G})$$

There are constants $c, k > 0$ such that $\forall n \geq c : p(n^2) \leq n^k - 1$.

Note: $p(n^2)$ bounds the running time of M on an input code($\mathcal{G}$) for a graph $\mathcal{G} = (V, E)$ with $n = |V|$ nodes.

Let $\mathcal{G} = (V, E)$ be a graph with $n = |V| \geq c$ nodes.

Let $w = a_1 a_2 \cdots a_{n^2} = \text{code}(\mathcal{G})$.

Fagin's Theorem: From NP to $\exists$SO

It suffices to come up with an $\exists$SO-sentence F such that:

$$\exists c > 0 \; \forall n \geq c \; \forall \text{ graphs } \mathcal{G} \text{ with } n \text{ nodes}: \\ \mathcal{G} \models F \Leftrightarrow M \text{ accepts code}(\mathcal{G})$$

There are constants $c, k > 0$ such that $\forall n \geq c : p(n^2) \leq n^k - 1$.

Note: $p(n^2)$ bounds the running time of M on an input code($\mathcal{G}$) for a graph $\mathcal{G} = (V, E)$ with $n = |V|$ nodes.

Let $\mathcal{G} = (V, E)$ be a graph with $n = |V| \geq c$ nodes.

Let $w = a_1 a_2 \cdots a_{n^2} = \text{code}(\mathcal{G})$.

Hence, every M-computation for input w has length at most $n^k - 1$.

Fagin's Theorem: From NP to $\exists$SO

An accepting M computation

$$\text{init}(w) = \gamma_0 \vdash_M \gamma_1 \vdash_M \gamma_2 \vdash_M \cdots \vdash_M \gamma_m$$

$(m \leq n^k - 1)$ on input w can be represented by an $(n^k \times n^k)$-matrix with entries from $\Gamma \cup (Q \times \Gamma)$:

Fagin's Theorem: From NP to $\exists$SO

An accepting M computation

$$\mathrm{init}(w) = \gamma_0 \vdash_M \gamma_1 \vdash_M \gamma_2 \vdash_M \cdots \vdash_M \gamma_m$$

$(m \leq n^k - 1)$ on input w can be represented by an $(n^k \times n^k)$-matrix with entries from $\Gamma \cup (Q \times \Gamma)$:

- For $1 \leq i \leq m+1$, the i-th row represents the configuration γ_{i-1}.

Fagin's Theorem: From NP to $\exists$SO

An accepting M computation

$$\text{init}(w) = \gamma_0 \vdash_M \gamma_1 \vdash_M \gamma_2 \vdash_M \cdots \vdash_M \gamma_m$$

$(m \leq n^k - 1)$ on input w can be represented by an $(n^k \times n^k)$-matrix with entries from $\Gamma \cup (Q \times \Gamma)$:

- For $1 \leq i \leq m+1$, the i-th row represents the configuration γ_{i-1}.
- For $m+1 < i \leq n^k$, the i-th row represents the configuration γ_m.

Fagin's Theorem: From NP to $\exists$SO

An accepting M computation

$$\mathrm{init}(w) = \gamma_0 \vdash_M \gamma_1 \vdash_M \gamma_2 \vdash_M \cdots \vdash_M \gamma_m$$

$(m \leq n^k - 1)$ on input w can be represented by an $(n^k \times n^k)$-matrix with entries from $\Gamma \cup (Q \times \Gamma)$:

- For $1 \leq i \leq m+1$, the i-th row represents the configuration γ_{i-1}.
- For $m+1 < i \leq n^k$, the i-th row represents the configuration γ_m.

Fagin's Theorem: From NP to $\exists$SO

An accepting M computation

$$\mathrm{init}(w) = \gamma_0 \vdash_M \gamma_1 \vdash_M \gamma_2 \vdash_M \cdots \vdash_M \gamma_m$$

$(m \leq n^k - 1)$ on input w can be represented by an $(n^k \times n^k)$-matrix with entries from $\Gamma \cup (Q \times \Gamma)$:

- For $1 \leq i \leq m+1$, the i-th row represents the configuration γ_{i-1}.
- For $m+1 < i \leq n^k$, the i-th row represents the configuration γ_m.

Our $\exists$SO-sentence F will express the existence of such a matrix.

Fagin's Theorem: From NP to ∃SO

Our ∃SO-sentence F will have the form

$$\exists \leq \exists (S_i)_{1 \leq i \leq k}\ \exists (T_x)_{x \in Q \cup \Gamma}\ \exists C_0\ \exists C_1\ \exists z_0\ \exists z_1 : \bigwedge_{i=1}^{5} F_i \wedge \bigwedge_{i=1}^{k} G_i$$

for first-order formulas $F_1, \ldots, F_5$, $G_1, \ldots, G_k$.

Fagin's Theorem: From NP to ∃SO

Our ∃SO-sentence F will have the form

$$\exists \leq \exists (S_i)_{1 \leq i \leq k} \ \exists (T_x)_{x \in Q \cup \Gamma} \ \exists C_0 \ \exists C_1 \ \exists z_0 \ \exists z_1 : \bigwedge_{i=1}^{5} F_i \wedge \bigwedge_{i=1}^{k} G_i$$

for first-order formulas $F_1, \dots, F_5$, $G_1, \dots, G_k$.

Formula G_1 expresses that (i) $\leq$ is a linear order on the node set V, (ii) S_1 is the associated successor relation, (iii) z_0 (resp., z_1) is the first (resp., last) element of $\leq$:

$$\begin{aligned} &\forall x, y, z : x \leq x \wedge (x \leq y \leq x \to x = y) \wedge \\ &\qquad\qquad (x \leq y \leq z \to x \leq z) \wedge (x \leq y \vee y \leq x) \wedge \\ &\forall x, y : S_1(x, y) \leftrightarrow (x < y \wedge \neg \exists z (x < z < y)) \wedge \\ &\forall y : z_0 \leq y \leq z_1 \end{aligned}$$

Fagin's Theorem: From NP to ∃SO

Formula G_i ($2 \leq i \leq k$) expresses that S_i is a successor relation on i-tuples of nodes:

$$\forall x_1, \ldots, x_i, y_1, \ldots, y_i : S_i(x_1, \ldots, x_i, y_1, \ldots, y_i) \leftrightarrow$$
$$\Big(\big(S_1(x_1, y_1) \wedge \bigwedge_{j=2}^{i} x_j = y_j\big) \vee$$
$$\big(x_1 = z_1 \wedge y_1 = z_0 \wedge S_{i-1}(x_2, \ldots, x_i, y_2, \ldots, y_i)\big)\Big)$$

Fagin's Theorem: From NP to $\exists$SO

Two abbreviations:

- $\text{first}(x_1, \ldots, x_k) = \bigwedge_{i=1}^{k} x_i = z_0$
- $\text{last}(x_1, \ldots, x_k) = \bigwedge_{i=1}^{k} x_i = z_1$

Fagin's Theorem: From NP to ∃SO

The second-order variables C_0 and C_1 are k-ary.

Fagin's Theorem: From NP to ∃SO

The second-order variables C_0 and C_1 are k-ary.

F_1 expresses that at every "time instance" $\overline{t} = (t_1, \ldots, t_k)$ either C_0 or C_1 holds:

$$\forall \overline{t} : \Big(C_0(\overline{t}) \vee C_1(\overline{t}) \Big) \wedge \Big(\neg C_0(\overline{t}) \vee \neg C_1(\overline{t}) \Big)$$

Fagin's Theorem: From NP to $\exists$SO

The second-order variables C_0 and C_1 are k-ary.

F_1 expresses that at every "time instance" $\overline{t} = (t_1, \dots, t_k)$ either C_0 or C_1 holds:

$$\forall \overline{t} : \Big(C_0(\overline{t}) \vee C_1(\overline{t}) \Big) \wedge \Big(\neg C_0(\overline{t}) \vee \neg C_1(\overline{t}) \Big)$$

Intuition: If $C_0(\overline{t})$ (resp., $C_1(\overline{t})$) then at time $\overline{t}$ the Turing machine takes choice 0 (resp. 1).

Fagin's Theorem: From NP to ∃SO

The second-order variables C_0 and C_1 are k-ary.

F_1 expresses that at every "time instance" $\overline{t} = (t_1, \ldots, t_k)$ either C_0 or C_1 holds:

$$\forall \overline{t} : \Big(C_0(\overline{t}) \vee C_1(\overline{t}) \Big) \wedge \Big(\neg C_0(\overline{t}) \vee \neg C_1(\overline{t}) \Big)$$

Intuition: If $C_0(\overline{t})$ (resp., $C_1(\overline{t})$) then at time $\overline{t}$ the Turing machine takes choice 0 (resp. 1).

Hence, the predicates C_0 and C_1 encode a certain computation path.

Fagin's Theorem: From NP to $\exists$SO

Every second-order variable T_x for $x \in \Gamma \cup Q$ is $2k$-ary.

Fagin's Theorem: From NP to $\exists$SO

Every second-order variable T_x for $x \in \Gamma \cup Q$ is $2k$-ary.

Formula F_2 expresses that

- every time-position pair $(\overline{t}, \overline{p})$ is labelled (via T_a) with a unique tape symbol $a \in \Gamma$, and
- that for every time $\overline{t}$, there exists a unique position $\overline{p}$ such that the time-position pair $(\overline{t}, \overline{p})$ is labelled (via T_q) with a unique state state $q \in Q$.

$$\forall \overline{t}\ \forall \overline{p} : \bigvee_{a \in \Gamma} \left(T_a(\overline{t}, \overline{p}) \wedge \bigwedge_{b \in \Gamma \setminus \{a\}} \neg T_b(\overline{t}, \overline{p}) \right) \wedge$$

$$\forall \overline{t}\ \exists \overline{p} : \bigvee_{q \in Q} \left(T_q(\overline{t}, \overline{p}) \wedge \bigwedge_{q' \in Q \setminus \{q\}} \neg T_{q'}(\overline{t}, \overline{p}) \right) \wedge$$

$$\forall \overline{p}' : \overline{p}' \neq \overline{p} \rightarrow \bigwedge_{q \in Q} \neg T_q(\overline{t}, \overline{p}')$$

Fagin's Theorem: From NP to ∃SO

Formula F_3 expresses that the predicates T_x ($x \in Q \cup \Gamma$) encode a valid computation table of the Turing machine that corresponds to the computation path encoded by C_0 and C_1.

$$\begin{aligned}\forall \overline{p}\, \forall \overline{t} : \bigwedge_{(q,a) \in Q \times \Gamma} \bigwedge_{i \in \{0,1\}} & (T_a(\overline{t}, \overline{p}) \wedge T_q(\overline{t}, \overline{p}) \wedge C_i(\overline{t}) \wedge \neg \mathrm{last}(\overline{t})) \to \\ & (T_{\alpha_i(q,a)}(\overline{t}+1, \overline{p}) \wedge T_{\rho_i(q,a)}(\overline{t}+1, \overline{p}+\delta_i(q,a)) \wedge \\ & \forall \overline{p}' : \overline{p} \neq \overline{p}' \to \bigwedge_{b \in \Gamma} : T_b(\overline{t}, \overline{p}') \leftrightarrow T_b(\overline{t}+1, \overline{p}'))\end{aligned}$$

Here, $\overline{t}+1$ is the unique $\overline{t}'$ such that $S_k(\overline{t}, \overline{t}')$ holds (similarly, for $\overline{p} \pm 1$).

Fagin's Theorem: From NP to $\exists$SO

Formula F_4 expresses that the first row of the computation table encoded by the predicates T_x ($x \in Q \cup \Gamma$) is the initial configuration for input $w = a_1 a_2 \cdots a_{n^2}$.

$$\forall \overline{t}\ \forall \overline{p} : (\text{first}(\overline{t}) \wedge \text{first}(\overline{p})) \to T_{q_0}(\overline{t}, \overline{p}) \ \wedge$$

$$\forall \overline{t}\ \forall \overline{p} : \left(\text{first}(\overline{t}) \wedge \bigwedge_{i=3}^{k} p_i = z_0\right) \to$$

$$\big(E(p_1, p_2) \to T_1(\overline{t}, \overline{p}) \wedge \neg E(p_1, p_2) \to T_0(\overline{t}, \overline{p})\big) \ \wedge$$

$$\forall \overline{t}\ \forall \overline{p} : \left(\text{first}(\overline{t}) \wedge \neg \bigwedge_{i=3}^{k} p_i = z_0\right) \to T_{\Box}(\overline{t}, \overline{p})$$

Fagin's Theorem: From NP to $\exists$SO

Formula F_5 expresses that the accepting state q_Y appears in the computation table:

$$\exists \overline{t}\ \exists \overline{p} : T_{q_Y}(\overline{t}, \overline{p})$$

Fagin's Theorem: From NP to ∃SO

Formula F_5 expresses that the accepting state q_Y appears in the computation table:

$$\exists \overline{t}\ \exists \overline{p} : T_{q_Y}(\overline{t}, \overline{p})$$

This concludes the description of the ∃SO-sentence F.

Fagin's Theorem: From NP to $\exists$SO

Formula F_5 expresses that the accepting state q_Y appears in the computation table:

$$\exists \overline{t}\ \exists \overline{p} : T_{q_Y}(\overline{t}, \overline{p})$$

This concludes the description of the $\exists$SO-sentence F.

I hope, you are convinced that it works. $\square$

Descriptive Complexity

Fagin's theorem was the first result in descriptive complexity theory. There are many other characterizations:

Descriptive Complexity

Fagin's theorem was the first result in descriptive complexity theory. There are many other characterizations:

- $\mathsf{FO} = \mathsf{AC}^0$

Descriptive Complexity

Fagin's theorem was the first result in descriptive complexity theory. There are many other characterizations:

- FO = AC^0
- SO = PH (the polynomial time hierarchy).

Descriptive Complexity

Fagin's theorem was the first result in descriptive complexity theory. There are many other characterizations:

- $\mathsf{FO} = \mathsf{AC}^0$
- $\mathsf{SO} = \mathsf{PH}$ (the polynomial time hierarchy).
- SO-Horn $= \mathsf{P}$ on structures with a successor relation

Descriptive Complexity

Fagin's theorem was the first result in descriptive complexity theory. There are many other characterizations:

- FO = AC^0
- SO = PH (the polynomial time hierarchy).
- SO-Horn = P on structures with a successor relation
  SO-Horn is the set of all SO-formulas of the form

$$Q_1 R_1 \cdots Q_n R_n \forall \overline{x} \bigwedge_{i=1}^{k} D_i, \tag{8}$$

  where every D_i is a disjunction of literals containing at most one occurrence of a positive literal $R_j(\overline{y})$.

Descriptive Complexity

Fagin's theorem was the first result in descriptive complexity theory. There are many other characterizations:

- $\mathrm{FO} = \mathrm{AC}^0$
- $\mathrm{SO} = \mathrm{PH}$ (the polynomial time hierarchy).
- SO-Horn $=$ P on structures with a successor relation
  SO-Horn is the set of all SO-formulas of the form

$$Q_1 R_1 \cdots Q_n R_n \forall \overline{x} \bigwedge_{i=1}^{k} D_i, \tag{8}$$

  where every D_i is a disjunction of literals containing at most one occurrence of a positive literal $R_j(\overline{y})$.
- SO-Krom $=$ NL on structures with a successor relation:

Descriptive Complexity

Fagin's theorem was the first result in descriptive complexity theory. There are many other characterizations:

- FO = AC^0
- SO = PH (the polynomial time hierarchy).
- SO-Horn = P on structures with a successor relation
  SO-Horn is the set of all SO-formulas of the form

$$Q_1 R_1 \cdots Q_n R_n \forall \overline{x} \bigwedge_{i=1}^{k} D_i, \tag{8}$$

  where every D_i is a disjunction of literals containing at most one occurrence of a positive literal $R_j(\overline{y})$.
- SO-Krom = NL on structures with a successor relation:
  SO-Krom is the set of all SO-formulas of the form (8), where every D_i is a disjunction of literals containing at most two literals of the form $(\neg)R_j(\overline{y})$.

Descriptive Complexity

Why is it good to have logical characterizations of complexity classes?

Descriptive Complexity

Why is it good to have logical characterizations of complexity classes?

- Machine independent characterizations

Descriptive Complexity

Why is it good to have logical characterizations of complexity classes?

- Machine independent characterizations
- Independent of concrete encoding of input structures

Descriptive Complexity

Why is it good to have logical characterizations of complexity classes?

- Machine independent characterizations
- Independent of concrete encoding of input structures

Do these logical characterizations help solving the big open problems of complexity theory?