First-order theories

Gabriele Puppis

LaBRI / CNRS
Definition

Fix a class $\mathcal{C}$ of structures (e.g. graphs) and a logic $\mathcal{L}$ (e.g. FO).

The $\mathcal{L}$-theory of $\mathcal{C}$ is the set of all formulas in $\mathcal{L}$ that can be satisfied by some structure in $\mathcal{C}$.

The theory is **decidable** if there is an algorithm that receives formulas as input and tells whether they are in the theory or not.
Definition

Fix a class $\mathcal{C}$ of structures (e.g. graphs) and a logic $\mathcal{L}$ (e.g. FO).

The **$\mathcal{L}$-theory of $\mathcal{C}$** is the set of all formulas in $\mathcal{L}$ that can be *satisfied by some structure* in $\mathcal{C}$.

The theory is **decidable** if there is an algorithm that receives formulas as input and tells whether they are in the theory or not.

Examples

- first-order theory of the class of all graphs
- monadic theory of the class of all linear orders
- monadic theory of $\mathbb{N}$
- monadic theory of the grid
Undecidability of first-order theory

One cannot decide whether a given formula of $\text{FO}[\Sigma, E_1, E_2]$ is satisfied over some labelled grid.
Undecidability of first-order theory

One cannot decide whether a given formula of $\text{FO}[\Sigma, E_1, E_2]$ is satisfied over some labelled grid.

Given a Turing machine $M$, construct $\psi_M$ defining its halting runs:
Undecidability of first-order theory

One cannot decide whether a given formula of \( \text{FO}[\Sigma, E_1, E_2] \) is satisfied over some labelled grid.

Given a Turing machine \( M \), construct \( \psi_M \) defining its halting runs:

1. encode initial configuration by top row

\[
\begin{array}{ccccccc}
q_0 & \rightarrow & q_1 & \rightarrow & q_2 & \rightarrow & q_3 & \rightarrow & \ldots \\
\downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & \ldots \\
\bullet & \rightarrow & \bullet & \rightarrow & \bullet & \rightarrow & \bullet & \rightarrow & \bullet & \rightarrow & \bullet & \rightarrow & \ldots \\
\downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & \ldots \\
\bullet & \rightarrow & \bullet & \rightarrow & \bullet & \rightarrow & \bullet & \rightarrow & \bullet & \rightarrow & \bullet & \rightarrow & \ldots \\
\downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & \ldots \\
\bullet & \rightarrow & \bullet & \rightarrow & \bullet & \rightarrow & \bullet & \rightarrow & \bullet & \rightarrow & \bullet & \rightarrow & \ldots \\
\downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & \ldots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ldots
\end{array}
\]
Undecidability of first-order theory

One cannot decide whether a given formula of $\text{FO}[\Sigma, E_1, E_2]$ is satisfied over some labelled grid.

Given a Turing machine $M$, construct $\psi_M$ defining its halting runs:

1. encode initial configuration by top row

2. encode next configurations by next rows
Undecidability of first-order theory

One cannot decide whether a given formula of $\text{FO}[\Sigma, E_1, E_2]$ is satisfied over some labelled grid.

Given a Turing machine $M$, construct $\psi_M$ defining its halting runs:

1. encode initial configuration by top row
2. encode next configurations by next rows
3. find a row with halting configuration
Undecidability of first-order theory

One cannot decide whether a given formula of $\text{FO}[\Sigma, E_1, E_2]$ is satisfied over some labelled grid.

(and equally for $\text{MSO}[E_1, E_2]$ over the grid $\mathbb{N} \times \mathbb{N}$)

Given a Turing machine $M$, construct $\psi_M$ defining its halting runs:

1. encode initial configuration by top row
2. encode next configurations by next rows
3. find a row with halting configuration

MSO can even guess the labelling!
Undecidability of first-order theory

One cannot decide whether a given formula of $\text{FO}[\Sigma, E_1, E_2]$ is satisfied over some labelled grid.

(and equally for $\text{MSO}[E_1, E_2]$ over the grid $\mathbb{N} \times \mathbb{N}$)

**QUIZ**

- Given a Turing machine $M$, construct $\psi_M$ defining its **halting runs**:
  1. Encode initial configuration by top row
  2. Encode next configurations by next rows
  3. Find a row with halting configuration

---

**MSO can even guess the labelling!**
The FO theory of the class of all finite structures is undecidable (provided that signature contains a binary predicate besides =).
Consequences (Church ’36, Turing ’37, Trakhtenbrot ’50, …)

The FO theory of the class of all finite structures is undecidable (provided that signature contains a binary predicate besides =).

The MSO theory of any class of graphs with unbounded grids as minors is undecidable.
Consequences (Church ’36, Turing ’37, Trakhtenbrot ’50, ...) 

The FO theory of the class of all finite structures is undecidable (provided that signature contains a binary predicate besides =).

The MSO theory of any class of graphs with unbounded grids as minors is undecidable.

The MSO theory of \((\mathbb{N}, +)\) is undecidable.

\[
\begin{array}{cccccccc}
\cdots & n & \cdots & n+1 & \cdots & n+2 & \cdots & n+3 & \cdots & n+4 \\
\vdots & \vdots & & \vdots & & \vdots & & \vdots & & \vdots \\
\cdots & 2n & \cdots & 2n+2 & \cdots & 2n+4 & \cdots & 2n+6 & \cdots & 2n+8 \\
\vdots & \vdots & & \vdots & & \vdots & & \vdots & & \vdots \\
\cdots & 3n & \cdots & 3n+3 & \cdots & 3n+6 & \cdots & 3n+9 & \cdots & 3n+12 \\
\vdots & \vdots & & \vdots & & \vdots & & \vdots & & \vdots \\
\cdots & 4n & \cdots & 4n+4 & \cdots & 4n+8 & \cdots & 4n+16 & \cdots & 4n+20 \\
\end{array}
\]
**Definition**

**Presburger arithmetic** is the first-order theory of \((\mathbb{N}, +)\)

**Examples of Presburger formulas**

\[
\psi_0 = \exists x. \forall y. (x + y = y)
\]

\[
\varphi_{\leq}(x, y) = \exists z. (y = x + z)
\]

\[
\varphi_{2x}(x, y) = (x + x = y)
\]

\[
\psi_\omega = \forall x. \exists y. (x \leq y \land \neg x = y)
\]
Decidability of Presburger arithmetic (Presburger ’29)

One can decide if a Presburger sentence $\psi$ holds over $(\mathbb{N}, +)$.

Originally proved by quantifier elimination. Here we use automata!
Decidability of Presburger arithmetic (Presburger '29)

One can decide if a Presburger sentence $\psi$ holds over $(\mathbb{N}, +)$.

Originally proved by *quantifier elimination*. Here we use automata!

1. Encode numbers $x \in \mathbb{N}$ by *reverse binary expansions* $[x] \in \mathbb{B}^*$

   e.g. $[4] = 001$, $[0] = \varepsilon$, ...
Decidability of Presburger arithmetic (Presburger ’29)

One can decide if a Presburger sentence $\psi$ holds over $(\mathbb{N}, +)$.

Originally proved by \textit{quantifier elimination}. Here we use automata!

1. Encode numbers $x \in \mathbb{N}$ by \textbf{reverse binary expansions} $[x] \in \mathbb{B}^*$
   e.g. $[4] = 001, \quad [0] = \varepsilon, \ldots$

2. Encode sum relation $+ \subseteq \mathbb{N} \times \mathbb{N} \times \mathbb{N}$ by language $L_+ \subseteq (\mathbb{B} \times \mathbb{B} \times \mathbb{B})^*$
   e.g. $[+(3, 1, 4)] = [3] \otimes [1] \otimes [4] = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix}$
Decidability of Presburger arithmetic (Presburger ’29)

One can decide if a Presburger sentence $\psi$ holds over $(\mathbb{N}, +)$. Originally proved by *quantifier elimination*. Here we use automata!

1. Encode numbers $x \in \mathbb{N}$ by **reverse binary expansions** $[x] \in \mathbb{B}^*$
   
e.g. $[4] = 001$, $[0] = \varepsilon$, ...

2. Encode sum relation $+ \subseteq \mathbb{N} \times \mathbb{N} \times \mathbb{N}$ by language $L_+ \subseteq (\mathbb{B} \times \mathbb{B} \times \mathbb{B})^*$
   
e.g. $[ + (3, 1, 4)] = [3] \otimes [1] \otimes [4] = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix}$

$$\mathcal{A}_+ :$$

![Graph of automaton $A_+$](image)
Decidability of Presburger arithmetic (Presburger ’29)

One can decide if a Presburger sentence $\psi$ holds over $(\mathbb{N}, +)$.

We inductively translate every Presburger formula $\varphi(x_1, \ldots, x_m)$ into a finite automaton $A_\varphi$ over $\Sigma_m = \mathbb{B}^m$ such that

$$L(A_\varphi) = \{ [x_1] \otimes \cdots \otimes [x_m] \in \Sigma_m^* \mid (\mathbb{N}, +) \models \varphi(x_1, \ldots, x_m) \}$$

so as to reduce satisfiability of $\varphi$ to emptiness of $L(A_\varphi)$.
Decidability of Presburger arithmetic (Presburger '29)

One can decide if a Presburger sentence $\psi$ holds over $(\mathbb{N}, +)$.

We inductively translate every Presburger formula $\varphi(x_1, \ldots, x_m)$ into a finite automaton $A_\varphi$ over $\Sigma_m = \mathbb{B}^m$ such that

$$L(A_\varphi) = \{ [x_1] \otimes \cdots \otimes [x_m] \in \Sigma_m^* \mid (\mathbb{N}, +) \models \varphi(x_1, \ldots, x_m) \}$$

so as to reduce satisfiability of $\varphi$ to emptiness of $L(A_\varphi)$.

- For atomic formulas $x = y$ and $+(x, y, z)$, use automata $A_\leq$ and $A_+$
One can decide if a Presburger sentence $\psi$ holds over $(\mathbb{N}, +)$.

We inductively translate every Presburger formula $\varphi(x_1, \ldots, x_m)$ into a finite automaton $A_\varphi$ over $\Sigma_m = \mathbb{B}^m$ such that

$$L(A_\varphi) = \{ [x_1] \otimes \cdots \otimes [x_m] \in \Sigma_m^* \mid (\mathbb{N}, +) \models \varphi(x_1, \ldots, x_m) \}$$

so as to reduce satisfiability of $\varphi$ to emptiness of $L(A_\varphi)$.

- For atomic formulas $x = y$ and $+(x, y, z)$, use automata $A_\leq$ and $A_+$.
- For disjunction $\varphi_1(\bar{x}) \lor \varphi_2(\bar{x})$, compute union of $A_{\varphi_1}$ and $A_{\varphi_2}$. 
Decidability of Presburger arithmetic (Presburger ’29)

One can decide if a Presburger sentence $\psi$ holds over $(\mathbb{N}, +)$.

We inductively translate every Presburger formula $\varphi(x_1, \ldots, x_m)$ into a finite automaton $A_\varphi$ over $\Sigma_m = \mathbb{B}^m$ such that

$$L(A_\varphi) = \{ [x_1] \otimes \cdots \otimes [x_m] \in \Sigma_m^* \mid (\mathbb{N}, +) \models \varphi(x_1, \ldots, x_m) \}$$

so as to reduce satisfiability of $\varphi$ to emptiness of $L(A_\varphi)$.

- For atomic formulas $x = y$ and $+(x, y, z)$, use automata $A_-$ and $A_+$
- For disjunction $\varphi_1(\bar{x}) \lor \varphi_2(\bar{x})$, compute union of $A_{\varphi_1}$ and $A_{\varphi_2}$
- For negation $\neg \varphi(\bar{x})$, compute the complement of $A_\varphi$
Decidability of Presburger arithmetic (Presburger ’29)

One can decide if a Presburger sentence $\psi$ holds over $(\mathbb{N}, +)$.

We inductively translate every Presburger formula $\varphi(x_1, \ldots, x_m)$ into a finite automaton $A_\varphi$ over $\Sigma_m = \mathbb{B}^m$ such that

$$L(A_\varphi) = \{ [x_1] \otimes \cdots \otimes [x_m] \in \Sigma^*_m \mid (\mathbb{N}, +) \models \varphi(x_1, \ldots, x_m) \}$$

so as to reduce satisfiability of $\varphi$ to emptiness of $L(A_\varphi)$.

- For atomic formulas $x = y$ and $+(x, y, z)$, use automata $A_\equiv$ and $A_+$
- For disjunction $\varphi_1(\bar{x}) \lor \varphi_2(\bar{x})$, compute union of $A_{\varphi_1}$ and $A_{\varphi_2}$
- For negation $\neg \varphi(\bar{x})$, compute the complement of $A_\varphi$
- For existential quantification $\exists y. \varphi(\bar{x}, y)$, project $A_\varphi$ from $\Sigma_{m+1}$ to $\Sigma_m$
Example of translation

Consider the (unsatisfiable) formula

\[ \psi = \exists x. \neg \exists y. (y = x + 1) \]
Example of translation

Consider the (unsatisfiable) formula

\[ \psi = \exists x. \neg \exists y. (y = x + 1) \]

1. Start from automaton \( A_{y=x+1} \)
Example of translation

Consider the (unsatisfiable) formula

$$\psi = \exists x. \neg \exists y. (y = x + 1)$$

1. Start from automaton $A_{y=x+1}$

2. Project away the encoding of $y$, thus capturing $\exists y. (y = x + 1)$
Example of translation

Consider the (unsatisfiable) formula

\[ \psi = \exists x. \neg \exists y. (y = x + 1) \]

1. Start from automaton \( A_{y=x+1} \)
2. Project away the encoding of \( y \), thus capturing \( \exists y. (y = x + 1) \)
3. Complement the det. automaton, thus capturing \( \neg \exists y. (y = x + 1) \)
Example of translation

Consider the (unsatisfiable) formula

\[ \psi = \exists x. \neg \exists y. (y = x + 1) \]

1. Start from automaton \( A_{y=x+1} \)
2. Project away the encoding of \( y \), thus capturing \( \exists y. (y = x + 1) \)
3. Complement the det. automaton, thus capturing \( \neg \exists y. (y = x + 1) \)
4. Project away the encoding of \( x \), thus getting \( \exists x. \neg \exists y. (y = x + 1) \)
Example of translation

Consider the (unsatisfiable) formula

\[ \psi = \exists x. \neg \exists y. (y = x + 1) \]

1. Start from automaton \( A_{y=x+1} \)

2. Project away the encoding of \( y \), thus capturing \( \exists y. (y = x + 1) \)

3. Complement the det. automaton, thus capturing \( \neg \exists y. (y = x + 1) \)

4. Project away the encoding of \( x \), thus getting \( \exists x. \neg \exists y. (y = x + 1) \)

What’s wrong??

Languages of encodings should be closed under padding with 0’s. After complement, keep only final states that are stable under 0.
Example of translation

Consider the (unsatisfiable) formula

$$\psi = \exists x. \neg \exists y. (y = x + 1)$$

1. Start from automaton $A_{y=x+1}$
2. Project away the encoding of $y$, thus capturing $\exists y. (y = x + 1)$
3. Complement the det. automaton, thus capturing $\neg \exists y. (y = x + 1)$
4. Project away the encoding of $x$, thus getting $\exists x. \neg \exists y. (y = x + 1)$

What’s wrong??

 Languages of encodings should be closed under padding with 0’s
After complement, keep only final states that are stable under 0.
The previous result can be generalized to many other structures:

**Definition**

An **automatic structure** is a structure that is isomorphic to

\[ \left( L, R_1, \ldots, R_n \right) \]

where

- \( L \) is a regular language of words over \( \Sigma \)
  
  (each word identifies a precise element of the structure)

- each relation \( R_i \) has arity \( k_i \) and is represented
  
  by a regular language \( L_i \) over \( (\Sigma \cup \{\#\})^{k_i} \)

  (e.g. \((ab, aab) \in R_i \) iff \( \left( \begin{array}{c} a \\ a \end{array} \right) \left( \begin{array}{c} b \\ b \end{array} \right) \left( \begin{array}{c} \# \\ b \end{array} \right) \in L_i \) )

Examples of automatic structures:

- \((\mathbb{N}, \mathbb{N}, \text{divides})\)
- \((\mathbb{N}, \times, \text{power})\)
- \((\mathbb{N} \times \mathbb{N}, \rightarrow, \downarrow)\)

Binary tree with successor, ancestor, and equi-level predicates

Unlabelled grid
The previous result can be generalized to many other structures:

**Definition**

An *automatic structure* is a structure that is isomorphic to

\[
\left( L, R_1, \ldots, R_n \right)
\]

where

- \( L \) is a regular language of words over \( \Sigma \) 
  (each word identifies a precise element of the structure)
- each relation \( R_i \) has arity \( k_i \) and is represented
  by a regular language \( L_i \) over \( \left( \Sigma \cup \{\#\} \right)^{k_i} \)

(e.g. \((ab, abb) \in R_i \iff (a\ a)(b\ b)(\#\ b) \in L_i \))

**Examples of automatic structures**

- \((\mathbb{N},+,\mid_p)\), with \( x \mid_p y \iff x = p^n \) divides \( y \)
- Binary tree with successor, ancestor, and equi-level predicates
- Unlabelled grid \((\mathbb{N} \times \mathbb{N}, \rightarrow, \downarrow)\)
Theorem (Büchi ’60, Hodgson ’76, Khoussainov & Nerode ’94)

Every automatic structure has a **decidable first-order theory**.
Theorem (Büchi '60, Hodgson '76, Khoussainov & Nerode '94)
Every automatic structure has a **decidable first-order theory**.

On the other hand...

Theorem
Some automatic structures have an **undecidable reachability problem**.

💡 The **transition graph** of a Turing machine is automatic!

- configurations are encoded by words $a_1...a_{i-1} q a_i a_{i+1}...a_n$
- transitions are of the following forms

\[
\begin{align*}
& a_1...a_{i-1} q a_i a_{i+1}...a_n \\
& a_1...a_{i-1} q a_i a_{i+1}...a_n \\
& a_1...a_{i-1} a_i q a_{i+1}...a_n \\
& a_1...a_{i-1} q' a_i a_{i+1}...a_n \\
& a_1...a_{i-1} a'_i q' a_{i+1}...a_n \\
& a_1...a_{i-1} q' a'_i a_{i+1}...a_n \\
& a_1...a_{i-1} a_i q a_{i+1}...a_n \\
& a_1...a_{i-1} q a_i a_{i+1}...a_n \\
& a_1...a_{i-1} a_i q a_{i+1}...a_n \\
& a_1...a_{i-1} a'_i q a_{i+1}...a_n \\
& a_1...a_{i-1} q' a'_i a_{i+1}...a_n
\end{align*}
\]