Theorem (Entscheidungsproblem)

The following decision problem is undecidable:
Given: A sentence φ of first-order logic
Question: Is φ a tautology?

Theorem (Entscheidungsproblem)

The following decision problem is undecidable:
Given: A sentence φ of first-order logic
Question: Is φ a tautology?

We prove that the **Entscheidungsproblem** is undecidable by a reduction from the undecidability of the **Halting problem** for Turing machines

Turing machine

A Turing machine over alphabet A is a tuple $M = \langle \Delta, Q, \delta, q_0, q_f \rangle$, where:

Turing machine

A Turing machine over alphabet A is a tuple $M = \langle \Delta, Q, \delta, q_0, q_f \rangle$, where:

- Δ is a finite alphabet, contains A and contains symbol $\Box \notin A$ (blank);

Turing machine

A Turing machine over alphabet A is a tuple $M = \langle \Delta, Q, \delta, q_0, q_f \rangle$, where:

- Δ is a finite alphabet, contains A and contains symbol $\Box \notin A$ (blank);
- Q is a finite set of states;

Turing machine

A Turing machine over alphabet A is a tuple $M = \langle \Delta, Q, \delta, q_0, q_f \rangle$, where:

- Δ is a finite alphabet, contains A and contains symbol $\Box \notin A$ (blank);
- Q is a finite set of states;
- $q_0 \in Q$ is an initial state;

Turing machine

A Turing machine over alphabet A is a tuple $M = \langle \Delta, Q, \delta, q_0, q_f \rangle$, where:

- Δ is a finite alphabet, contains A and contains symbol $\Box \notin A$ (blank);
- Q is a finite set of states;
- $q_0 \in Q$ is an initial state;
- $q_f \in Q$ is a final or accepting state;

Turing machine

A Turing machine over alphabet A is a tuple $M = \langle \Delta, Q, \delta, q_0, q_f \rangle$, where:

- Δ is a finite alphabet, contains A and contains symbol $\Box \notin A$ (blank);
- Q is a finite set of states;
- $q_0 \in Q$ is an initial state;
- $q_f \in Q$ is a final or accepting state;
- $\delta : (Q - \{q_f\}) \times \Delta \to \Delta \times Q \times \{-1, 0, +1\}$ is a transition function.

We utilize the following version of the halting problem:
Given: (An encoding of) a Turing machine M

We utilize the following version of the halting problem:
Given: (An encoding of) a Turing machine M
Question: Does M halt on the empty word?

Let ϑ be the conjunction of the following:

- $\forall y \neg P(y, c)$
- $\forall x \exists y P(x, y)$
- $\forall x \forall y (P(x, y) \rightarrow R(x, y))$
- $\forall x \forall y \forall z (R(x, y) \rightarrow (R(y, z) \rightarrow R(x, z)))$
- $\forall x \neg R(x, x)$

Let ϑ be the conjunction of the following:

- $\forall y \neg P(y, c)$
- $\forall x \exists y P(x, y)$
- $\forall x \forall y (P(x, y) \to R(x, y))$
- $\forall x \forall y \forall z (R(x, y) \to (R(y, z) \to R(x, z)))$
- $\forall x \neg R(x, x)$

ϑ is satisfiable, and every model $\mathfrak{A}$ of ϑ contains an infinite sequence of distinct elements $c^{\mathfrak{A}} = a_0, a_1, a_2, \ldots$, satisfying $(a_i, a_{i+1}) \in P^{\mathfrak{A}}$ for each i

Our goal – a construction to turn a Turing machine M into a sentence φ_M such that:
$\underline{M \text{ accpets } \varepsilon}$ iff $\underline{\varphi_M \text{ is a tautology}}$

Our goal – a construction to turn a Turing machine M into a sentence φ_M such that:

$\underline{M \text{ accpets } \varepsilon}$ iff $\underline{\varphi_M \text{ is a tautology}}$

It is easier to construct a sentence ψ_M such that

$\underline{M \text{ loops forever on } \varepsilon}$ iff $\underline{\psi_M \text{ is satisfiable}}$

and take φ_M to be $\neg\psi_M$

Signature:

Signature:

- Unary relation symbols S_q for all states $q \in Q$;

Signature:

- Unary relation symbols S_q for all states $q \in Q$;
- Binary relation symbols C_a for all letters $a \in \Delta$;

Signature:

- Unary relation symbols S_q for all states $q \in Q$;
- Binary relation symbols C_a for all letters $a \in \Delta$;
- Binary relatoin symbol G;

Signature:

- Unary relation symbols S_q for all states $q \in Q$;
- Binary relation symbols C_a for all letters $a \in \Delta$;
- Binary relatoin symbol G;
- constant symbol c and relation symbols P and R from ϑ

- The formula $S_q(x)$ is read: after x steps of computation the machine is in state q.

- The formula $S_q(x)$ is read: after x steps of computation the machine is in state q.
- The formula $G(x, y)$ is read: after x steps of computation the head occupies position y.

- The formula $S_q(x)$ is read: after x steps of computation the machine is in state q.
- The formula $G(x, y)$ is read: after x steps of computation the head occupies position y.
- The formula $C_a(x, y)$ is read: after x steps of computation symbol a is in cell y

1. ϑ
2. $S_{q_0}(c) \wedge G(c,c) \wedge \forall x\, C_{\mathrm{B}}(c,x)$;
3. $\forall x(\bigvee_{q \in Q} S_q(x))$;
4. $\forall x(S_q(x) \to \neg S_p(x))$, dla $q, p \in Q$, $q \neq p$;
5. $\forall x \forall y(\bigvee_{a \in \Delta} C_a(x,y))$;
6. $\forall x \forall y(C_a(x,y) \to \neg C_b(x,y))$, dla $a, b \in \Delta$, $a \neq b$;
7. $\forall x \exists y\, G(x,y)$;
8. $\forall x \forall y \forall z(G(x,y) \wedge G(x,z) \to y = z)$;

9. $\forall x \forall y \forall z (S_q(x) \wedge G(x,y) \wedge C_a(x,y) \wedge P(x,z) \rightarrow S_p(z) \wedge C_b(z,y))$, for $\delta(q,a) = (p,b,i)$;
10. $\forall x \forall y \forall z (\neg G(x,y) \wedge C_a(x,y) \wedge P(x,z) \rightarrow C_a(z,y))$;
11. $\forall x \forall y \forall z \forall v (S_q(x) \wedge G(x,y) \wedge C_a(x,y) \wedge P(x,z) \wedge P(y,v) \rightarrow G(z,v))$, for $\delta(q,a) = (p,b,+1)$;
12. $\forall x \forall y \forall z (S_q(x) \wedge G(x,y) \wedge C_a(x,y) \wedge P(x,z) \wedge C_a(x,y) \rightarrow G(z,y))$, for $\delta(q,a) = (p,b,0)$;
13. $\forall x \forall y \forall z \forall v (y \neq c \rightarrow (S_q(x) \wedge G(x,y) \wedge C_a(x,y) \wedge P(x,z) \wedge P(v,y) \rightarrow G(z,v)))$, for $\delta(q,a) = (p,b,-1)$;
14. $\forall x \forall y \forall z \forall v (S_q(x) \wedge G(x,c) \wedge C_a(x,y) \wedge P(x,z) \rightarrow G(z,c))$, for $\delta(q,a) = (p,b,-1)$;
15. $\forall x \neg S_{q_f}(x)$.