The following decision problem is undecidable:

**Given:** A sentence $\varphi$ of first-order logic

**Question:** Is $\varphi$ a tautology?
The following decision problem is undecidable:

**Given:** A sentence $\varphi$ of first-order logic

**Question:** Is $\varphi$ a tautology?

We prove that the *Entscheidungsproblem* is undecidable by a reduction from the undecidability of the *Halting problem* for Turing machines.
A Turing machine over alphabet $A$ is a tuple $M = \langle \Delta, Q, \delta, q_0, q_f \rangle$, where:

- $\Delta$ is a finite alphabet, contains $A$ and contains symbol $\Box \not\in A$ (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;
- $\delta : (Q - \{q_f\}) \times \Delta \rightarrow \Delta \times Q \times \{-1, 0, +1\}$ is a transition function.
A Turing machine over alphabet $A$ is a tuple $M = \langle \Delta, Q, \delta, q_0, q_f \rangle$, where:

- $\Delta$ is a finite alphabet, contains $A$ and contains symbol $\square \notin A$ (blank);
A Turing machine over alphabet $A$ is a tuple $M = \langle \Delta, Q, \delta, q_0, q_f \rangle$, where:

- $\Delta$ is a finite alphabet, contains $A$ and contains symbol $\square \notin A$ (blank);
- $Q$ is a finite set of states;
A Turing machine over alphabet $A$ is a tuple $M = \langle \Delta, Q, \delta, q_0, q_f \rangle$, where:

- $\Delta$ is a finite alphabet, contains $A$ and contains symbol $\square \notin A$ (blank);
- $Q$ is a finite set of states;
- $q_0 \in Q$ is an initial state;
A Turing machine over alphabet $A$ is a tuple $M = \langle \Delta, Q, \delta, q_0, q_f \rangle$, where:

- $\Delta$ is a finite alphabet, contains $A$ and contains symbol $\square \notin A$ (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;
A **Turing machine** over alphabet $A$ is a tuple $M = \langle \Delta, Q, \delta, q_0, q_f \rangle$, where:

- $\Delta$ is a finite alphabet, contains $A$ and contains symbol $\square \notin A$ (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 $\vartheta$ 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 $\vartheta$ be the conjunction of the following:

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

$\vartheta$ is satisfiable, and every model $\mathcal{A}$ of $\vartheta$ contains an infinite sequence of distinct elements $c^\mathcal{A} = a_0, a_1, a_2, \ldots$, satisfying $(a_i, a_{i+1}) \in P^\mathcal{A}$ for each $i$
Our goal – a construction to turn a Turing machine $M$ into a sentence $\varphi_M$ such that:

$M$ accepts $\varepsilon$ iff $\varphi_M$ is a tautology

$\psi_M$ is satisfiable and take $\varphi_M$ to be $\neg \psi_M$
Our goal – a construction to turn a Turing machine $M$ into a sentence $\varphi_M$ such that:

$\overline{M \text{ accepts } \varepsilon}$ iff $\varphi_M$ is a tautology

It is easier to construct a sentence $\psi_M$ such that

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

and take $\varphi_M$ 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 relation 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 relation symbol $G$;
- constant symbol $c$ and relation symbols $P$ and $R$ from $\varnothing$.
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 $y$. 

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 $y$. 
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. \emptyset
2. \begin{align*} &S_{q_0}(c) \land G(c, c) \land \forall x \ C_B(c, x); \\
&\forall x (\bigvee_{q \in Q} S_q(x)); \\
&\forall x (S_q(x) \rightarrow \neg S_p(x)), \text{ dla } q, p \in Q, q \neq p; \\
&\forall x \forall y (\bigvee_{a \in \Delta} C_a(x, y)); \\
&\forall x \forall y (C_a(x, y) \rightarrow \neg C_b(x, y)), \text{ dla } a, b \in \Delta, a \neq b; \\
&\forall x \exists y \ G(x, y); \\
&\forall x \forall y \forall z (G(x, y) \land G(x, z) \rightarrow y = z); \\
\end{align*}
\[ \forall x \forall y \forall z \left( S_q(x) \land G(x, y) \land C_a(x, y) \land P(x, z) \rightarrow S_p(z) \land C_b(z, y) \right), \text{ for } \delta(q, a) = (p, b, i); \]

\[ \forall x \forall y \forall z \left( \neg G(x, y) \land C_a(x, y) \land P(x, z) \rightarrow C_a(z, y) \right); \]

\[ \forall x \forall y \forall z \forall v \left( S_q(x) \land G(x, y) \land C_a(x, y) \land P(x, z) \land P(y, v) \rightarrow G(z, v) \right), \text{ for } \delta(q, a) = (p, b, +1); \]

\[ \forall x \forall y \forall z \forall v \left( y \neq c \rightarrow \left( S_q(x) \land G(x, y) \land C_a(x, y) \land P(x, z) \land P(v, y) \rightarrow G(z, v) \right) \right), \text{ for } \delta(q, a) = (p, b, -1); \]

\[ \forall x \forall y \forall z \forall v \left( S_q(x) \land G(x, c) \land C_a(x, y) \land P(x, z) \rightarrow G(z, c) \right), \text{ for } \delta(q, a) = (p, b, -1); \]

\[ \forall x \neg S_q_f(x). \]