Exercise 9.1. [Decidability]

1. Resolution for first-order logic is sound and complete.
2. Satisfiability and validity for first-order logic are undecidable.

How do you reconcile these two facts?

Exercise 9.2. [Ground Resolution]

Use ground resolution to prove that the following formula is valid:

$$(\forall x P(x, f(x))) \rightarrow \exists y P(c, y)$$

Exercise 9.3. [Equality]

We consider how to model equality in predicate logic. In the lecture slides, the following axiom schema for congruence is used:

$$\frac{Eq(x_i, y) \quad \cdots \quad Eq(x_n, y_n)}{Eq(f(x_1, \ldots, x_i, \ldots, x_n), f(x_1, \ldots, y, \ldots, x_n))}$$

Assume that this schema is replaced by:

$$\frac{Eq(x_1, y_1) \quad \cdots \quad Eq(x_n, y_n)}{Eq(f(x_1, \ldots, x_n), f(y_1, \ldots, y_n))}$$

Reflexivity, symmetry and transitivity stay unchanged. Show that the above modified schemas is equivalent to the schemas from the lecture.

*Hint:* Simulate the modified schema with the original one and vice versa.
**Homework 9.1.  [Restricted Resolution]**  
(8 points)
In the resolution procedure as defined in the lecture slides, we can unify arbitrarily many literals from two clauses. Consider a modified resolution procedure, where exactly one literal is picked. We add another rule ("collapsing rule"): For a clause $C = \{L_1, \ldots, L_n\}$, where $\{L_i, L_j\}$ can be unified using a mgu $\delta$, add another clause $C' = (C - L_i)\delta$.

For example, given the clause
$$C = \{\neg W(x), \neg W(f(y)), T(x, y), \neg W(f(c))\}$$
we can apply the collapsing rule as follows:

$$L_1 = \neg W(x), L_2 = \neg W(f(y)), \delta = \{x \mapsto f(y)\}, C' = \{\neg W(f(y)), T(f(y), y), \neg W(f(c))\}$$

(Note that there are multiple possible ways to apply the collapsing rule to $C$.)

Prove that our modified resolution calculus, including collapsing rule, can be simulated by the original resolution calculus, and vice versa.

**Homework 9.2.  [Resolution]**  
(6 points)
Show with resolution that:
$$f(f(f(a))) = a \rightarrow f(f(a)) = a \rightarrow f(a) = a$$
is valid. First, remove equality based on the procedure from the lecture. Then perform resolution.
Homework 9.3. [Undecidability] (6 points)
Recall the proof that validity of first-order formulas is undecidable from the lecture. It works by transforming register machine programs to formulas expressing the program behaviour.

Review the slides for the possible instructions and the execution state \((n_1, \ldots, n_r, k)\). A summary of the proof construction from the lecture follows.

Summary. In the proof, we have introduced the function symbols \(z\) and \(s\) and the predicate symbol \(R\). \(z\) and \(s\) should represent natural numbers (0 and successor), similar to Exercise 7.3. \(R(n_1, \ldots, n_r, k)\) represents that a program state with register \(R_i\) set to \(n_i\) and \(k\) as the program counter can be reached. The initial state \(R(z, \ldots, z, s(z))\) is always reachable.

We now phrase the implications that model the transitions of the program. For each instruction \(I_i\), we define a formula \(\psi_i\), depending on the kind of instruction. We use the notation \(n\) to “convert” a natural number into a corresponding term \(s(s(\ldots s(z) \ldots)))\).

- \(I_i = R_n := 0\)
  \(\psi_i = \forall x_1 \ldots x_r. (R(x_1, \ldots, x_n, \ldots, x_r, \overline{1}) \rightarrow R(x_1, \ldots, z, \ldots, x_r, s(\overline{1})))\)

- \(I_i = R_n := R_n + 1\)
  \(\psi_i = \forall x_1 \ldots x_r. (R(x_1, \ldots, x_n, \ldots, x_r, \overline{1}) \rightarrow R(x_1, \ldots, s(x_n), \ldots, x_r, s(\overline{1})))\)

- \(I_i = R_n := R_m + 1\)
  \(\psi_i = \forall x_1 \ldots x_r. (R(x_1, \ldots, x_n, \ldots, x_r, \overline{1}) \rightarrow R(x_1, \ldots, x_m, \ldots, x_r, s(\overline{1})))\)

- \(I_i = (\text{IF } R_m = R_n \text{ GOTO } p)\)
  \(\psi_i = \forall x_1 \ldots x_r. (R(x_1, \ldots, x_r, \overline{1}) \rightarrow ((x_m = x_n \rightarrow R(x_1, \ldots, x_r, \overline{p})) \land (x_m \neq x_n \rightarrow R(x_1, \ldots, x_r, s(\overline{1}))))\)

We also define \(\psi_N = (\forall xy. (s(x) = s(y) \rightarrow x = y) \land \forall x (z \neq s(x)))\).

The behaviour of an entire program is now simulated by the formula
\[
\psi = \psi_N \land R(z, \ldots, z, s(z)) \land \bigwedge_{1 \leq i \leq s} \psi_i
\]
In plain words, the conjunction of “the natural numbers”, the initial state, and the behaviour of every instruction.¹

Problem. Consider the following program \(P\), annotated with instruction numbers:
1: \(R_1 := R_1 + 1\)
2: \(\text{IF } R_0 = R_1 \text{ GOTO } 1\)

In this case, the program state can be expressed with two registers. The initial state is \(R(0, 0, \overline{0}, \overline{1}) = R(z, z, s(z))\).

Based on this program, show that \(\psi_N\) is a necessary part of the formula \(\psi\). Construct a suitable structure \(\mathcal{A}\) and a program state \(S = (n_1, n_2, k)\) such that:

1. \(\mathcal{A} \models R(z, z, s(z))\)
2. \(\mathcal{A} \models \psi_i\) for \(1 \leq i \leq s\)
3. \(S\) is reachable in \(P\) but \(\mathcal{A} \not\models R(n_1, n_2, k)\)

¹Note that the initial state had been omitted in the lecture by accident.