Exercise 1 (Programming in Prolog): \(0.5 + 1 + 2 + 1.5 + 2 + 1 = 8\) points

**Important:** In addition to handing in the solution on paper, please also mail your solutions for this exercise to **lp15-hiwis@i2.informatik.rwth-aachen.de**. Indicate your immatriculation numbers in the subject of the mail and inside the Prolog file.

In this exercise we investigate algorithms on lists in Prolog.

### Lists

Lists in Prolog are represented by terms built over the signature \(\Sigma = \Sigma_0 \cup \Sigma_2\) with \(\Sigma_0 = \{\}\) and \(\Sigma_2 = \{.\}\). The symbol \(\{}\) denotes the empty list, while the term \(.\(X, XS\)\) denotes the list starting with the element \(X\) (called the head of the list) and having the list \(XS\) as the remaining list (called the tail of the list). Thus, a list in Prolog containing the three numbers 2, 3 and 5 would be written as \(.\(2, .\(3, .\(5, \})\)\)). This is the standard representation internally used by Prolog.

However, Prolog also knows a more comfortable way to write lists. The term \(.\(X, XS\)\) can also be written as \([X|XS]\). With this representation, the above list is written as \([2|3|5]\)). To save brackets, the representation can be shortened by just enumerating elements in the order they appear in the list: \([2,3,5]\). Equivalently, one can also write \([2,3|5]\). While this list representation is easier to use for humans, it is equivalent to the internal representation used in Prolog. You can use both representations and even mix them within one Prolog program.

As an example for an algorithm working on lists, we write a predicate `hasLast/2` (i.e., a predicate `hasLast` of arity 2) in Prolog where `hasLast(XS, X)` is true iff \(X\) is the last element of the list \(XS\).

\[
 hasLast([], X).
 hasLast([X|XS], Y) :- hasLast(XS, Y).
\]

The following solution is also correct, but uses the less readable list representation.

\[
 hasLast(.(X, \}), X).
 hasLast(.(X, XS), Y) :- hasLast(XS, Y).
\]

You may not use any predefined Prolog predicates in this exercise (unless allowed explicitly)! However, you can define your own auxiliary predicates.

a) Implement a predicate `isNumber/1` in Prolog which determines whether its only argument is of the form \(s(\ldots s(0)\ldots)\), where \(0\) represents the number Zero and \(s(X)\) is the successor of \(X\).

b) Implement a predicate `remove/3` in Prolog. Here, `remove(X,XS,YS)` is true if \(XS\) is a list containing \(X\) as an element and \(YS\) is the list resulting by removing the first occurrence of \(X\) from \(XS\). For example, the query `remove(s(0),[s(0),0,s(0)],ZS)` should return the only answer \(ZS = [0,s(0)]\). The query `remove(s(0),[0],ZS)` should fail.

Hints:
- To test if \(X\) and \(Y\) are different, you may use the predefined predicate `\=\`. Example: \(a \= b\) is true.

c) Implement a predicate `minimum/2` which computes the minimum of a list (specified in the first argument) as its second argument. If the list in the first argument is empty or contains any element not being a number as specified in a), the predicate should fail. For example, the query `minimum([s(0),s(s(0)),s(0)],X)` should return the answer \(X = s(0)\).

Hints:
- Make use of predicates you already defined, for example the predicate `isNumber` to test if an argument is a number.
- You may use the predicate `smaller/2`, which you can find in Exercise 3.

d) Use the predicates `remove` and `minimum` to implement a predicate `selectionSort/2` which sorts the list specified in the first argument. Here, `selectionSort(XS,YS)` is true if \(YS\) contains exactly the elements...
e) Implement a predicate `insertSorted/3` in Prolog which inserts a natural number specified in the first argument into the list specified in the second argument. Here, for a sorted lists `XS`, `insertSorted(X, XS, YS)` is true iff `YS` is the list `XS` where `X` has been inserted at the correct position such that `YS` is sorted again. For example, the query `insertSorted(s(0), [s(0), s(s(0)), 0, s(0)])`, `X` should return the only answer `X = [s(0), s(s(0)), 0, s(0)]`. If `insertSorted`'s first argument is not a natural number, the query should fail.

Hints:
- Make use of predicates you already defined, for example the predicate `isNumber` to test if an argument is a number.
- You may use the predicate `smaller/2`, which you can find in Exercise 3.

f) Use the predicate `insertSorted` to implement a predicate `insertionSort/2` in Prolog which sorts the list specified in the first argument. Here, `insertionSort(XS, YS)` is true iff `YS` contains exactly the elements of `XS` in ascending order. For example, the query `insertionSort([s(0), s(s(0)), 0, s(0)], X)` should return the only answer `X = [s(0), s(s(0)), 0, s(0)]`.

Solution:

```prolog
smaller(0, s(_)).
smaller(s(X), s(Y)) :- smaller(X, Y).
smallerOrEqual(X, X) :- isNumber(X).
smallerOrEqual(X, Y) :- smaller(X, Y).

% a)
isNumber(0).
isNumber(s(X)) :- isNumber(X).

% b)
remove(X, [X|YS], YS).
remove(X, [Y|YS], [Y|ZS]) :- X \= Y, remove(X, YS, ZS).

% c)
minimum([X], X) :- isNumber(X).
minimum([X|XS], X) :- isNumber(X), minimum(XS, Y), smallerOrEqual(X, Y).
minimum([X|XS], Y) :- isNumber(X), minimum(XS, Y), smaller(Y, X).

% d)
selectionSort([], []).
selectionSort(XS, [Y|YS]) :- minimum(XS, Y), remove(Y, XS, ZS), selectionSort(ZS, YS).

% e)
insertSorted(X, [], [X]) :- isNumber(X).
insertSorted(X, [Y|YS], [X, Y|YS]) :- smallerOrEqual(X, Y).
insertSorted(X, [Y|YS], [Y|ZS]) :- smaller(Y, X), insertSorted(X, YS, ZS).

% f)
insertionSort([], []).
insertionSort([X|XS], YS) :- insertionSort(XS, ZS), insertSorted(X, ZS, YS).
```
Exercise 2 (Herbrand model): \(2 + 2 + 1.5 + 1.5 + 2 = 9\) points

Let
\[
\varphi = p(a) \\
\land \forall x \ p(x) \rightarrow q(x, f(x)) \\
\land \forall x, y \ q(x, y) \rightarrow p(y) \\
\land \exists x \lnot p(x)
\]

be a formula over the signature \((\Sigma, \Delta)\) with \(\Sigma = \Sigma_0 \cup \Sigma_1, \Sigma_0 = \{a\}, \Sigma_1 = \{f\}, \Delta = \Delta_1 \cup \Delta_2, \Delta_1 = \{p\}\) and \(\Delta_2 = \{q\}\).

Hints:
- You can use \(f^i(a)\) as an abbreviation for \(f(\ldots(f(a))\ldots)\).

a) Prove that \(\varphi\) is satisfiable. (Hint: It suffices to choose a carrier with at most two elements.)
b) Give a Herbrand model for \(\varphi\) or show why no such model exists.
c) Transform \(\varphi\) into a satisfiability-equivalent formula \(\psi\) in Skolem normal form.
d) Give a Herbrand model for \(\psi\) or show why no such model exists.
e) Are \(\varphi\) and \(\psi\) equivalent (cf. Definition 2.2.1)? Explain your answer.

Solution:

a) \(S = (A, \alpha)\) with \(A = \{0, \star\}, \alpha_a = 0, \alpha_f(0) = 0, \alpha_f(\star) = \star, \alpha_p = \{0\}\), and \(\alpha_q = \{(0, 0)\}\).
b) Assume there is a Herbrand model \(S' = (A', \alpha')\) with \(S' \models \varphi\). Then \(A' = \{a\} \cup \{f^i(a) \mid i \in \mathbb{N}, i \geq 1\}\).

Due to the first subformula, we know \(p(a)\) holds. Because of \(S' \models p(a)\) and \(S' \models \forall x p(x) \rightarrow q(x, f(x))\)

we know \(S' \models q(a, f(a))\). With \(S' \models \forall X, Y q(X, Y) \rightarrow p(Y)\) we know \(S' \models p(f(a))\). Using \(\varphi_1\) and \(\varphi_2\), by induction we have \(\{a\} \cup \{f^i(a) \mid i \in \mathbb{N}, i \geq 1\} \subseteq \alpha_p\). This contradicts the last subformula, since \(X\) can only be instantiated with terms out of \(A' = \{a\} \cup \{f^i(a) \mid i \in \mathbb{N}, i \geq 1\}\).

c) We first transform \(\varphi\) into Prenex normal form:
\[
\varphi' = \forall X, Y, Z \ \exists V \ p(a) \land (p(X) \rightarrow q(X, f(X))) \land (q(Y, Z) \rightarrow p(Z)) \land \lnot p(V)
\]

Now we transform \(\varphi'\) to Skolem normal form (where \(g\) is a new function symbol with arity 3):
\[
\psi = \forall X, Y, Z \ p(a) \land (p(X) \rightarrow q(X, f(X))) \land (q(Y, Z) \rightarrow p(Z)) \land \lnot p(g(X, Y, Z))
\]

By reordering the subformulas of \(\varphi\), the last conjunct could also be \(\lnot p(g)\), where \(g\) is a function symbol with arity 0.

d) \(S' = (A', \alpha')\) with \(A' = T(\Sigma \cup \{g\})\) and
- \(\alpha'_a = a,\)
- \(\alpha'_f(X) = f(X)\) for all \(X \in T(\Sigma \cup \{g\}),\)
- \(\alpha'_g(X, Y, Z) = g(X, Y, Z)\) for all \(X, Y, Z \in T(\Sigma \cup \{g\}),\)
- \(\alpha'_f = \{a\} \cup \{f^i(a) \mid i \in \mathbb{N}, i \geq 1\} = T(\Sigma),\)
- \(\alpha'_q = \{(a, f(a))\} \cup \{(f^i(a), f^{i+1}(a)) \mid i \in \mathbb{N}, i \geq 1\}\) (or alternatively, \(\alpha'_q = T(\Sigma) \times T(\Sigma)\)).
e) Consider the structure \( S'' = (A'', \alpha'') \) with \( A'' = \{0, \star\} \) and
\[
\begin{align*}
\alpha''_0 &= 0, \\
\alpha''(0) &= 0, \alpha''(\star) = \star, \\
\alpha''(X, Y, Z) &= 0, \\
\alpha''_Y &= \{0\}, \\
\alpha''_Z &= \{(0, 0)\},
\end{align*}
\]
which we obtain by a small modification of \( S \). Again, we have \( S'' \models \varphi \). However, we also have \( S'' \not\models \psi \) since \( S'' \not\models \neg p(g(a, a)) \). Therefore, by Definition 2.2.1 the formulas \( \varphi \) and \( \psi \) are not equivalent.

**Exercise 3 (Gilmore’s algorithm):**  
(5 points)

Consider the following logic program
\[
\begin{align*}
\text{smaller}(0, s(X)). \\
\text{smaller}(s(X), s(Y)) &\leftarrow \text{smaller}(X, Y).
\end{align*}
\]
and the query
\[
? - \text{smaller}(s(0), s(s(0))).
\]
Use Gilmore’s algorithm to show that the formulas \( \varphi_1 \) and \( \varphi_2 \) corresponding to the logic program entail the formula \( \varphi \) corresponding to the query (i.e., \( \{\varphi_1, \varphi_2\} \models \varphi \)).

**Solution:**

The formulas corresponding to the program are \( \varphi_1 = \forall X \text{smaller}(0, s(X)) \) and \( \varphi_2 = \forall X, Y \text{smaller}(X, Y) \rightarrow \text{smaller}(s(X), s(Y)) \). The formula corresponding to the query is \( \varphi := \text{smaller}(s(0), s(s(0))) \). We need to show that \( \varphi_1 \land \varphi_2 \land \neg \varphi \) is unsatisfiable. The corresponding formula in Skolem normal form is \( \forall X, Y, Z \psi \) with
\[
\psi = \text{smaller}(0, s(X)) \land (\neg \text{smaller}(Y, Z) \lor \text{smaller}(s(Y), s(Z))) \land \neg \text{smaller}(s(0), s(s(0)))
\]
With the following enumeration of the Herbrand expansion (where \( \text{smaller} \) is written as \( \text{sm} \))
\[
\begin{align*}
\psi_1 &= \psi[X/0, Y/0, Z/0] &= \text{sm}(0, s(0)) \land (\neg \text{sm}(0, 0) \lor \text{sm}(s(0), s(s(0)))) \land \neg \text{sm}(s(0), s(s(0)))) \\
\psi_2 &= \psi[X/0, Y/0, Z/s(0)] &= \text{sm}(0, s(0)) \land (\neg \text{sm}(0, 0) \lor \text{sm}(s(0), s(s(0)))) \land \neg \text{sm}(s(0), s(s(0)))) \\
\psi_3 &= \psi[X/0, Y/s(0), Z/0] &= \text{sm}(0, s(0)) \land (\neg \text{sm}(0, 0) \lor \text{sm}(s(0), s(s(0)))) \land \neg \text{sm}(s(0), s(s(0)))) \\
\psi_4 &= \psi[X/0, Y/s(0), Z/s(0)] &= \text{sm}(0, s(0)) \land (\neg \text{sm}(0, 0) \lor \text{sm}(s(0), s(s(0)))) \land \neg \text{sm}(s(0), s(s(0)))) \\
\psi_5 &= \psi[X/s(0), Y/0, Z/0] &= \text{sm}(0, s(0)) \land (\neg \text{sm}(0, 0) \lor \text{sm}(s(0), s(s(0)))) \land \neg \text{sm}(s(0), s(s(0)))) \\
\psi_6 &= \psi[X/s(0), Y/0, Z/s(0)] &= \text{sm}(0, s(0)) \land (\neg \text{sm}(0, 0) \lor \text{sm}(s(0), s(s(0)))) \land \neg \text{sm}(s(0), s(s(0)))) \\
\psi_7 &= \psi[X/s(0), Y/s(0), Z/0] &= \text{sm}(0, s(0)) \land (\neg \text{sm}(0, 0) \lor \text{sm}(s(0), s(s(0)))) \land \neg \text{sm}(s(0), s(s(0)))) \\
\psi_8 &= \psi[X/s(0), Y/s(0), Z/s(0)] &= \text{sm}(0, s(0)) \land (\neg \text{sm}(0, 0) \lor \text{sm}(s(0), s(s(0)))) \land \neg \text{sm}(s(0), s(s(0)))) \\
\psi_9 &= \psi[X/0, Y/0, Z/s(s(0))] &= \text{sm}(0, s(0)) \land (\neg \text{sm}(0, 0) \lor \text{sm}(s(0), s(s(0)))) \land \neg \text{sm}(s(0), s(s(0))))
\end{align*}
\]
and by replacing all atomic formulas with propositional variables (e.g. $A_{01}$ for $\text{smaller}(0,s(0))$) we get:

$$
\psi'_1 = A_{01} \land (\neg A_{00} \lor A_{11}) \land \neg A_{13}
$$

$$
\psi'_2 = A_{01} \land (\neg A_{01} \lor A_{12}) \land \neg A_{13}
$$

$$
\psi'_3 = A_{01} \land (\neg A_{10} \lor A_{21}) \land \neg A_{13}
$$

$$
\psi'_4 = A_{01} \land (\neg A_{11} \lor A_{22}) \land \neg A_{13}
$$

$$
\psi'_5 = A_{02} \land (\neg A_{00} \lor A_{11}) \land \neg A_{13}
$$

$$
\psi'_6 = A_{02} \land (\neg A_{01} \lor A_{12}) \land \neg A_{13}
$$

$$
\psi'_7 = A_{02} \land (\neg A_{10} \lor A_{21}) \land \neg A_{13}
$$

$$
\psi'_8 = A_{02} \land (\neg A_{11} \lor A_{22}) \land \neg A_{13}
$$

$$
\psi'_9 = A_{01} \land (\neg A_{02} \lor A_{13}) \land \neg A_{13}
$$

... 

When considering $\psi'_1 \land \ldots \land \psi'_9$, the subformula $\neg A_{13} \land A_{02} \land (\neg A_{02} \lor A_{13})$ is unsatisfiable.