Master Exam Version V3M

First Name: ____________________________________________

Last Name: ____________________________________________

Immatriculation Number: _________________________________

Course of Studies (please mark exactly one):

- Informatik Bachelor
- Informatik Master
- SSE Master
- Other: ________________________________

<table>
<thead>
<tr>
<th>Exercise</th>
<th>Maximal Points</th>
<th>Achieved Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exercise 1</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Exercise 2</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>Exercise 3</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>Exercise 4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Exercise 5</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Exercise 6</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>60</td>
<td></td>
</tr>
</tbody>
</table>

| Grade | - |

Instructions:

- On every sheet please give your first name, last name, and immatriculation number.
- You must solve the exam without consulting any extra documents (e.g., course notes).
- Make sure your answers are readable. Do not use red or green pens or pencils.
- Please answer the exercises on the exercise sheets. If needed, also use the back sides of the exercise sheets.
- Answers on extra sheets can only be accepted if they are clearly marked with your name, your immatriculation number, and the exercise number.
- Cross out text that should not be considered in the evaluation.
- Students that try to cheat do not pass the exam.
- At the end of the exam, please return all sheets together with the exercise sheets.
Exercise 1 (Theoretical Foundations): (3 + 3 + 4 = 10 points)

Let \( \varphi = q(0,s(0)) \land X,Y \left( q(X,Y) \rightarrow q(s(X),s(Y)) \right) \) and \( \psi = \exists Z q(s(Z),s(s(Z))) \) be formulas over the signature \( (\Sigma, \Delta) \) with \( \Sigma = \Sigma_0 \cup \Sigma_1, \Sigma_0 = \{0\}, \Sigma_1 = \{s\} \), and \( \Delta = \Delta_2 = \{q\} \).

a) Prove that \( \varphi \vdash \psi \) by means of resolution.

   \textit{Hint: First transform the formula } \( \varphi \land \neg \psi \) \textit{into an equivalent clause set.}

b) Explicitly give a Herbrand model of the formula \( \varphi \) (i.e., specify a carrier and a meaning for all function and predicate symbols). You do not have to provide a proof for your answer.

c) Prove or disprove that input resolution is complete for arbitrary clause sets.
Exercise 2 (Procedural Semantics, SLD tree): (7 + 9 = 16 points)

Consider the following Prolog program $P$ which can be used to sort a list of numbers using the bubblesort algorithm:

\[
\begin{align*}
\text{bubble}&(L, R) :- \text{swap}(L, N), !, \text{bubble}(N, R). \\
\text{bubble}&(L, L).
\end{align*}
\]
\[
\begin{align*}
\text{swap}([A,B|L]), [B,A|L]) &:- B < A. \\
\text{swap}([A|L], [A|N]) &:- \text{swap}(L, N).
\end{align*}
\]

*Hint:* As usual, you should treat $<$ as if it were defined by the infinitely many facts

\[
\begin{align*}
0 &< 1. \\
1 &< 2. \\
0 &< 2. \\
\ldots
\end{align*}
\]

**a)** The program $P'$ results from $P$ by removing the cut. Consider the following query:

\[
?- \text{bubble}([2,1,0], [1,2,X]).
\]

For the logic program $P'$, i.e. without the cut, please show a successful computation for the query above (i.e., a computation of the form $(G, \emptyset) \vdash^+_P (\square, \sigma)$ where $G = \{\neg \text{bubble}([2,1,0], [1,2,X])\}$). It suffices to give substitutions only for those variables which are used to define the value of the variable $X$ in the query.
b) Please give a graphical representation of the SLD tree for the query \(-\) `bubble([2, 1], X)` in the program \(\mathcal{P}\) (i.e., \textbf{with the cut}).
Exercise 3 (Fixpoint Semantics): (3 + 3 + 3 = 9 points)

Consider the following logic program \( \mathcal{P} \) over the signature \((\Sigma, \Delta)\) with \( \Sigma = \{0, s\} \) and \( \Delta = \{\text{gt}\} \).
\[
\text{gt}(s(X), 0).
\text{gt}(s(X), s(Y)) :- \text{gt}(X, Y).
\]

a) For each \( n \in \mathbb{N} \) explicitly give \( \text{trans}_n^\mathcal{P}(\emptyset) \) in closed form, i.e., using a non-recursive definition.

b) Compute the set \( \text{lfp}(\text{trans}_\mathcal{P}) \).

c) Give \( F[\mathcal{P}, \{\neg\text{gt}(s(s(X)), Y)\}] \).
Exercise 4 (Universality): (10 points)

Consider a function $f : \mathbb{N}^{n+1} \to \mathbb{N}$. The function $g : \mathbb{N}^n \to \mathbb{N}$ is defined by fixpointing of $f$:

$$g(k_1, \ldots, k_n) = k \text{ iff } f(k_1, \ldots, k_n, k) = k \text{ and } \text{for all } 0 \leq k' < k \text{ we have } f(k_1, \ldots, k_n, k') \text{ is defined and } f(k_1, \ldots, k_n, k') \neq k'$$

As an example, consider the function $\hat{f} : \mathbb{N}^2 \to \mathbb{N}$ with $\hat{f}(x, y) = y^2 - 3y + x$. The function $\hat{g} : \mathbb{N} \to \mathbb{N}$, constructed using fixpointing of $\hat{f}$ as described above, computes $\hat{g}(4) = 2$. The reason is that for $x = 4$, 2 is the smallest $y$ so that $\hat{f}(x, y) = y$. Indeed, $\hat{f}(4, 0) = 4$, $\hat{f}(4, 1) = 2$, $\hat{f}(4, 2) = 2$.

Consider a definite logic program $\mathcal{P}$ which computes the function $f$ using a predicate symbol $\mathfrak{f} \in \Delta^{n+2}$:

$$f(k_1, \ldots, k_{n+1}) = k' \text{ iff } \mathcal{P} \models \mathfrak{f}(k_1, \ldots, k_{n+1}, k').$$

Here, numbers are represented by terms built from $0 \in \Sigma_0, s \in \Sigma_1$ (i.e., $0 = 0, 1 = s(0), 2 = s(s(0)), \ldots$).

Please extend the definite logic program $\mathcal{P}$ such that it also computes the function $g$ using the predicate symbol $\mathfrak{g} \in \Delta^{n+1}$ (but without any built-in predicates):

$$g(k_1, \ldots, k_n) = k \text{ iff } \mathcal{P} \models \mathfrak{g}(k_1, \ldots, k_n, k).$$
Exercise 5 (Definite Logic Programming): (10 points)

Implement the predicate solve/1 in Prolog. This predicate can be used as a primitive SAT-solver for clause sets represented as lists of lists of literals. More precisely, a clause set is a list \( t \) of the form

\[
[[l_1^1, l_2^1, \ldots, l_k^1], [l_1^2, l_2^2, \ldots, l_k^2], \ldots, [l_1^n, l_2^n, \ldots, l_k^n]]
\]

where all \( l_j^i \) are of the form \( \text{pos}(X) \) or \( \text{neg}(X) \) for some Prolog variables \( X \). The list \( t \) represents a set of clauses where \( \text{pos}(X) \) stands for the propositional variable \( X \) while \( \text{neg}(X) \) stands for its negation.

A call \( \text{solve}(t) \) succeeds with a substitution satisfying the represented clause set \( t \) (by setting the variables to 1 or 0) if this set is satisfiable or fails if this set is unsatisfiable. If \( t \) does not represent a clause set as described above, then \( \text{solve}(t) \) may behave arbitrarily. You must not use any built-in predicates in this exercise. The following example calls to \( \text{solve}/1 \) illustrate its definition:

- ?- solve([[pos(A),pos(B)],[neg(A),neg(B)]]). has the two answer substitutions
  A = 1, B = 0 and A = 0, B = 1 (the order of the solutions is up to your implementation)

- ?- solve([[pos(A)],[neg(A)]]). fails

Hint: In this representation, a clause is satisfied if it contains at least one literal of the form \( \text{pos}(1) \) or \( \text{neg}(0) \). Moreover, a clause set is satisfied if all its clauses are satisfied. It might be useful to implement this predicate in a way that the following example calls work as described below, although this is not mandatory.

- ?- solve([[pos(1),pos(B)],[neg(1),neg(B)]]). succeeds with the answer substitution
  B = 0

- ?- solve([[pos(1),pos(0)],[neg(1),neg(0)]]). succeeds with the empty answer substitution
Exercise 6 (Arithmetic): (5 points)

Implement the predicate binomial/3 in Prolog. A call of `binomial(t_1, t_2, t_3)` works as follows. If `t_1` and `t_2` are integers with `t_1 < t_2` or at least one of `t_1` or `t_2` is negative, then it fails. If `t_1` and `t_2` are non-negative integers with `t_1 \geq t_2`, then `t_3` is unified with the integer resulting from \( \binom{t_1}{t_2} \). If `t_1` or `t_2` is no integer, `binomial/3` may behave arbitrarily.

Remember that the binomial coefficient \( \binom{n}{k} \) for non-negative integers `n` and `k` with `n \geq k` is defined as \( \binom{n}{k} = \frac{n!}{k!(n - k)!} \) with `0! = 1`.

The following example calls to `binomial/3` illustrate its definition:

- \(?- binomial(-3,2,X).\) fails
- \(?- binomial(2,3,X).\) fails
- \(?- binomial(3,2,X).\) succeeds with the answer substitution `X = 3`
- \(?- binomial(3,2,1).\) fails