Master Exam Version V3M

First Name: ________________________________________________________

Last Name: ________________________________________________________

Immatriculation Number: ____________________________________________

Course of Studies (please mark exactly one):

○ Informatic Bachelor  ○ Informatic 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>8</td>
<td></td>
</tr>
<tr>
<td>Exercise 2</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>Exercise 3</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Exercise 4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Exercise 5</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>Exercise 6</td>
<td>4</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): (5 + 3 = 8 points)

Let \( \varphi = p(s^2(0), 0) \land \forall X (p(s^2(X), X) \rightarrow p(s^4(X), s^2(X))) \land \neg p(s^3(0), s(0)) \) and \( \psi = \exists Y p(s^6(0), Y) \) be formulas over the signature \( (\Sigma, \Delta) \) with \( \Sigma = \Sigma_0 \cup \Sigma_1, \Sigma_0 = \{0\}, \Sigma_1 = \{s\}, \) and \( \Delta = \Delta_2 = \{p\}. \)

Here, \( s^2(0) \) stands for \( s(s(0)) \), etc.

a) Prove that \( \{ \varphi \} \models \psi \) by means of SLD resolution.

*Hint: First transform the formula \( \varphi \land \neg \psi \) 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.

Solution:

a) 

\[
\begin{align*}
\varphi \land \neg \psi & \iff p(s^2(0), 0) \land \forall X (p(s^2(X), X) \rightarrow p(s^4(X), s^2(X))) \land \neg p(s^3(0), s(0)) \land \neg \exists Y (p(s^6(0), Y)) \\
& \iff p(s^2(0), 0) \land \forall X (\neg p(s^2(X), X) \lor p(s^4(X), s^2(X))) \land \neg p(s^3(0), s(0)) \land \neg \exists Y (p(s^6(0), Y)) \\
& \iff p(s^2(0), 0) \land \forall X (\neg p(s^2(X), X) \lor p(s^4(X), s^2(X))) \land \neg p(s^3(0), s(0)) \land \forall Y (\neg p(s^6(0), Y)) \\
& \iff \forall X, Y (p(s^2(0), 0) \land (\neg p(s^2(X), X) \lor p(s^4(X), s^2(X))) \land \neg p(s^3(0), s(0)) \land \neg p(s^6(0), Y))
\end{align*}
\]

Thus, the equivalent clause set for \( \varphi \land \neg \psi \) is

\[
\{p(s^2(0), 0)\}, \{\neg p(s^2(X), X), p(s^4(X), s^2(X))\}, \{\neg p(s^3(0), s(0))\}, \{\neg p(s^6(0), Y)\}.
\]

We perform SLD resolution on this clause set to show \( \{ \varphi \} \models \psi \).

\[
\begin{align*}
\{p(s^2(0), 0)\} & \quad \{\neg p(s^2(X), X), p(s^4(X), s^2(X))\} & \quad \{\neg p(s^3(0), s(0))\} & \quad \{\neg p(s^6(0), Y)\} \\
& \quad X/0 & \quad \text{\( X/s^2(0), Y/s^4(0) \)} & \quad \{\neg p(s^4(0), s^2(0))\} \\
& \quad \{\neg p(s^2(0), 0)\} & & \\
\end{align*}
\]

Hence, we have proven \( \{ \varphi \} \models \psi \).
b) We have $S \models \varphi$ for the Herbrand structure $S = (\mathcal{T}(\Sigma), \alpha)$ with $\alpha_0 = 0, \alpha_s(t) = s(t)$, and $\alpha_p = \{(s^{i+2}(0), s^i(0)) \mid i \geq 0 \land i \neq 1\}$.
Exercise 2 (Procedural Semantics, SLD tree): (7 + 7 + 2 = 16 points)

Consider the following Prolog program \( \mathcal{P} \) which can be used to replace the letter sequence 'ba' by 'zz':

\[
\begin{align*}
\text{replace}([], []) &. \\
\text{replace}([b, a|XS], [z, z|YS]) & :- \text{replace}(XS, YS). \\
\text{replace}([X|XS], [Y|YS]) & :- \text{replace}(XS, YS).
\end{align*}
\]

For example, the query \(- \text{replace}([b, a, b, a], Z) \) would give the answer substitution \( Z = [z, z, z, z] \). Due to backtracking it is also possible to leave (parts of) the word unchanged. Because of that the answer substitutions \( Z = [b, a, z, z] \), \( Z = [z, z, b, a] \), and \( Z = [b, a, b, a] \) are also possible.

a) Consider the following query:

\[- \text{replace}([a, b, b, a], \text{Res}). \]

For the logic program \( \mathcal{P} \) please show a successful computation for the query above (i.e., a computation of the form \( (G, \emptyset) \vdash \top \sigma \) where \( G = \{ \neg \text{replace}([a, b, b, a], \text{Res}) \} \)). It suffices to give substitutions only for those variables which are used to define the value of the variable \( \text{Res} \) in the query.

b) Please give a graphical representation of the SLD tree for the query

\[- \text{replace}([a, b, b, a], \text{Res}). \]
in the program \( \mathcal{P} \).

c) Modify the program \( \mathcal{P} \) by inserting a single cut. No other modification is allowed. Your modified program must replace all occurrences of 'ba' by 'zz'.

For example, now the query \(- \text{replace}([b, a, b, a], Z) \) must have the only answer substitution \( Z = [z, z, z, z] \).

Solution:

\[\]

\[\]

a)

\[
\begin{align*}
& (\neg \text{replace}([a, b, b, a], \text{Res}). \emptyset) \\
\vdash & (\neg \text{replace}([b, b, a], \text{YS}). \{\text{Res} / [a|\text{YS}]\}) \\
\vdash & (\neg \text{replace}([b, a], \text{YS}'). \{\text{Res} / [a, b|\text{YS}']\}) \\
\vdash & (\neg \text{replace}([], \text{YS}''). \{\text{Res} / [a, b, z, z|\text{YS}'']\}) \\
\vdash & (\emptyset. \{\text{Res} / [a, b, z, z]\})
\end{align*}
\]

Alternative:

\[
\begin{align*}
& (\neg \text{replace}([a, b, b, a], \text{Res}). \emptyset) \\
\vdash & (\neg \text{replace}([b, b, a], \text{YS}). \{\text{Res} / [a|\text{YS}]\}) \\
\vdash & (\neg \text{replace}([b, a], \text{YS}'). \{\text{Res} / [a, b|\text{YS}']\}) \\
\vdash & (\neg \text{replace}([], \text{YS}''). \{\text{Res} / [a, b, b, a|\text{YS}'']\}) \\
\vdash & (\emptyset. \{\text{Res} / [a, b, b, a]\})
\end{align*}
\]
b) replace([a,b,b,a], Res)
   Res / [a|YS]
   replace([b,b,a], YS)
   YS / [b|YS']
   replace([b,a], YS')
   YS' / [z,z|YS'']
   replace([], YS'')
   YS'' / []
   □
   □

   replace([a], YS)
   YS' / [b|YS]
   replace([a], YS)
   YS' / [b|YS']
   replace([b,a], YS')
   YS' / [z,z|YS'']
   replace([], YS'')
   YS'' / []
   □
   □

   replace([b,a|XS], [z,z|YS]) :- !, replace(XS, YS).
   replace([X|XS], [X|YS]) :- replace(XS, YS).
Exercise 3 (Fixpoint Semantics): (4 + 3 + 3 = 10 points)

Consider the following logic program $P$ over the signature $(\Sigma, \Delta)$ with $\Sigma = \{a, q\}$ and $\Delta = \{p\}$.

$$p(a, a, Z).$$
$$p(q(Y), q(X), Z) :- p(X, Y, Z).$$

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

(b) Compute the set $\text{lfp} (\text{trans}_P)$.

(c) Give $F [\lnot P, \{\neg p(X, Y, Z)\}]$.

Solution: 

Let $G$ be the set of all ground terms, i.e., $G = \{q^i(a) \mid i \in \mathbb{N}\}$.

(a) 

$$\text{trans}_0 (\emptyset) = \emptyset$$
$$\text{trans}_1 (\emptyset) = \{p(a, a, t) \mid t \in G\}$$
$$\text{trans}_2 (\emptyset) = \{p(q(a), q(a), t) \mid t \in G\} \cup \text{trans}_1 (\emptyset)$$

$$\text{trans}_3 (\emptyset) = \{p(q^2(a), q^2(a), t) \mid t \in G\} \cup \text{trans}_2 (\emptyset)$$

$$\vdots$$

$$\text{trans}_n (\emptyset) = \{p(q^i(a), q^i(a), t) \mid t \in G, 0 \leq i < n\}$$

(b) $\text{lfp} (\text{trans}_P) = \{p(q^i(a), q^i(a), t) \mid t \in G, i \geq 0\}$

(c) $F [\lnot P, \{\neg p(X, Y, Z)\}] = \{p(t, t', t) \mid t, t' \in G\}$
Exercise 4 (Universality): \hspace{1cm} (10 points)

Consider a function \( f : \mathbb{N}^{n+1} \to \mathbb{N} \). The function \( g : \mathbb{N}^n \to \mathbb{N} \) is defined as:

\[
g(k_1, \ldots, k_n) = k \text{ iff } f(k_1 + k, \ldots, k_n + k, k) = 0 \text{ and } f(k_1 + k', \ldots, k_n + k', k') \text{ is defined and } f(k_1 + k', \ldots, k_n + k', k') > 0.
\]

As an example, consider the function \( \hat{f} : \mathbb{N}^2 \to \mathbb{N} \) with \( \hat{f}(x, y) = \max\{x - 4y, 0\} \). The function \( \hat{g} : \mathbb{N} \to \mathbb{N} \), constructed as described above, computes \( \hat{g}(6) = 2 \). The reason is that for \( x = 6 \), 2 is the smallest \( y \) such that \( \hat{f}(x + y, y) = 0 \). Indeed, \( \hat{f}(6 + 0, 0) = \hat{f}(6, 0) = 6, \hat{f}(6 + 1, 1) = \hat{f}(7, 1) = 3, \hat{f}(6 + 2, 2) = \hat{f}(8, 2) = 0 \).

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

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

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 \( \mathcal{G} \in \Delta^{n+1} \) (but without the cut or any other built-in predicate):

\[
g(k_1, \ldots, k_n) = k \text{ iff } \mathcal{P} \models \mathcal{G}(k_1, \ldots, k_n, k).
\]

Solution:

\[
g(X_1, \ldots, X_n, Z) : - \mathcal{G}(X_1, \ldots, X_n, 0, Z).
\]

\[
\mathcal{F}(X_1, \ldots, X_n, Y, Y) : - \mathcal{F}(X_1, \ldots, X_n, Y, 0).
\]

\[
\mathcal{F}(X_1, \ldots, X_n, Y, Z) : - \mathcal{F}(X_1, \ldots, X_n, Y, s(U)), \mathcal{F}(s(X_1), \ldots, s(X_n), s(Y), Z).
\]
Exercise 5 (Definite Logic Programming):

Implement the predicate noDup1/2 in Prolog. This predicate can be used to identify numbers in a list that appear exactly once, i.e., numbers which are no duplicates. The first argument of noDup1 is the list to analyze. The second argument is the list of numbers which are no duplicates, as described below.

As an example, for the list [2, 0, 3, 2, 1] the result [0, 3, 1] is computed (because 2 is a duplicate). In Prolog the corresponding call noDup1([s(s(0)), 0, s(s(s(0))), s(s(0)), s(0)], Res) gives the answer substitution Res = [0, s(s(s(0))), s(0)].

In your implementation you may (only) use the following two predefined predicates:

- contained(X, XS) is true if and only if the list XS contains X.
- notContained(X, XS) is true if and only if the list XS does not contain X.

Important: You may not use the cut or any other predefined predicates in your implementation! However, you may implement auxiliary predicates.

Solution:

noDup1(XS, Res) :- help(XS, [], Res).
help([], _, []).
help([X|XS], Seen, [X|Res]) :- notContained(X, XS), notContained(X, Seen),
    help(XS, [X|Seen], Res).
help([X|XS], Seen, Res) :- contained(X, Seen), help(XS, Seen, Res).
help([X|XS], Seen, Res) :- contained(X, XS), help(XS, [X|Seen], Res).

Not part of the solution, but useful to test:

contained(X, [X|_]).
contained(X, [_|XS]) :- contained(X, XS).
notContained(_, []).
notContained(X, [Y|ZS]) :- ne(X, Y), notContained(X, ZS).
ne(0, s(_)).
ne(s(_), 0).
ne(s(X), s(Y)) :- ne(X, Y).
Exercise 6 (Arithmetic): (4 points)

Tetration is the logical extension of multiplication and exponentiation:

- **Multiplication**: \(a \cdot n := \underbrace{a + a + \cdots + a}_n\)
- **Exponentiation**: \(a^n := \underbrace{a \cdot a \cdot \cdots \cdot a}_n\)
- **Tetration**: \(a \uparrow n := \underbrace{\left(\left(a^{(\underbrace{a\cdot a}\cdots)^n}\right)\right)}_n\)

Examples:

- \(4 \uparrow 2 = 4^4 = 256\)
- \(1 \uparrow 3 = 1^{(1^1)} = 1^1 = 1\)
- \(2 \uparrow 4 = \left(2^{(2^{(2^2)})}\right) = 2^{(2^{(2^4)})} = 2^{16} = 65536\)

Implement the predicate `tetration/3` in Prolog. For numbers \(x > 0, y > 0\) the call `tetration(X, Y, Z)` gives the answer substitution \(Z = m\) where \(m\) is \(x \uparrow y\).

As an example, `tetration(2, 4, Z)` gives the answer substitution \(Z = 65536\).

Your predicate only needs to work on input values \(x > 0, y > 0\), i.e., for other input values the result of the computation is irrelevant.

**Hint**: To compute \(x^y\) in Prolog you can use `x**y`.

Solution:

```prolog
tetration(X, 1, X).
tetration(X, Y, Z) :- Y > 1, A is Y - 1, tetration(X, A, B), Z is X**B.
```

---