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>11</td>
<td></td>
</tr>
<tr>
<td>Exercise 2</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>Exercise 3</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Exercise 4</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>Exercise 5</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Exercise 6</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>Grade</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

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): (4 + 4 + 3 = 11 points)

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

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.

c) Prove correctness of propositional resolution. You may assume that the following is correct: If $K$ is a set of clauses without variables, $S$ is a model of $K$, $K_1, K_2 \in K$ and $R$ is a resolvent of $K_1$ and $K_2$, then $S$ is a model of $K \cup \{R\}$.

Solution:

a) $\varphi \land \neg \psi \iff p(0, s(0)) \land \forall X, Y \left( p(X, Y) \rightarrow p(s(X), s^2(Y))) \land \neg p(s(0), s^2(0))) \land \neg \exists Z \left( p(Z, s^2(Z))) \right) \iff p(0, s(0)) \land \forall X, Y \left( \neg p(X, Y) \lor p(s(X), s^2(Y))) \land \neg p(s(0), s^2(0))) \land \neg \exists Z \left( p(Z, s^2(Z))) \right) \iff p(0, s(0)) \land \forall X, Y \left( \neg p(X, Y) \lor p(s(X), s^2(Y))) \land \neg p(s(0), s^2(0))) \land \forall Z \left( \neg p(Z, s^2(Z))) \right) \iff \forall X, Y, Z \left( p(0, s(0)) \land \left( \neg p(X, Y) \lor p(s(X), s^2(Y))) \land \neg p(s(0), s^2(0))) \land \neg p(Z, s^2(Z))) \right)$.

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

$\{p(0, s(0))\}, \{-p(X, Y) \lor p(s(X), s^2(Y)))\}, \{-p(s(0), s^2(0)))\}, \{-p(Z, s^2(Z)))\}$.

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

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 $\alpha_p = \{(s^i(0), s^{2i+1}(0)) \mid i \geq 0\}$.
c) We have to show that if $\mathcal{K}$ is a set of clauses without variables and $\square \in \text{Res}^*(\mathcal{K})$, then $\mathcal{K}$ is unsatisfiable. Let $\mathcal{K}$ be a set of clauses without variables and $\square \in \text{Res}^*(\mathcal{K})$. Then we know that $\text{Res}^*(\mathcal{K})$ is unsatisfiable. Assume there is a model $S$ for $\mathcal{K}$. By the given lemma we have that if $S$ is a model of $\mathcal{K}$, then $S$ is a model of $\text{Res}(\mathcal{K})$ and by induction it follows that $S$ is a model of $\text{Res}^*(\mathcal{K})$. This is a contradiction to $\text{Res}^*(\mathcal{K})$ being unsatisfiable. Therefore, no such model $S$ exists, so $\mathcal{K}$ is unsatisfiable.
Exercise 2 (Procedural Semantics, SLD tree):

Consider the following Prolog program \( \mathcal{P} \).

\[
\begin{align*}
  a(X, Y) & : - b(s(X)). \\
  a(X, Y) & : - b(Y), \text{!}, c(X). \\
  a(s(X), s(Y)) & : - a(X, Y). \\
  c(s(0)) & . \\
  b(0) & . \\
  b(1) & .
\end{align*}
\]

a) Consider the following query:

? - a(A, B).

For the logic program \( \mathcal{P}' \) that results by removing the cut from \( \mathcal{P} \), please show a successful computation for the query above (i.e., a computation of the form \( (G, \emptyset) \vdash_{\mathcal{P}'} (\square, \sigma) \) where \( G = \{ \neg a(A, B) \} \)). It suffices to give substitutions only for those variables which are used to define the value of the variables \( A \) and \( B \) in the query.

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

? - a(A, B).

in the program \( \mathcal{P} \) with the cut. For every part of the tree that is cut off by evaluating \( \text{!} \), please indicate the cut by marking the corresponding edge. For the cut-off parts only indicate the first cut-off goal, but do not evaluate further.

Solution:

---

a)

\[
\begin{align*}
  (\{ \neg a(A, B) \}, \emptyset) \\
  \vdash_{\mathcal{P}} (\{ \neg b(B), \neg c(A) \}, \emptyset) \\
  \vdash_{\mathcal{P}} (\{ \neg c(A) \}, \{ B/0 \}) \\
  \vdash_{\mathcal{P}} (\square, \{ A/s(0), B/0 \})
\end{align*}
\]

b)
\[
\begin{array}{c}
\text{a}(A,B) \\
\text{b}(s(A)) \quad \text{b}(B),!,\text{c}(A) \quad \text{a}(\ldots) \\
\quad \\text{A}/s(X), \text{B}/s(Y) \\
\quad \quad \text{B}/0 \quad \text{B}/1 \\
\quad \quad \quad \text{\ldots} \\
\end{array}
\]
Exercise 3 (Fixpoint Semantics): \( (5 + 2 + 3 = 10 \text{ points}) \)

Consider the following logic program \( P \) over the signature \((\Sigma, \Delta)\) with \( \Sigma = \Sigma_0 \cup \Sigma_1 \), \( \Sigma_0 = \{0\} \), \( \Sigma_1 = \{s\} \), and \( \Delta = \Delta_3 = \{p\} \).

\[
p(0, s(X), X).
p(s(X), s(Y), s(s(Z))) :- p(X, Y, Z).
\]

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

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

\(c\) Give \( FJP, \{\neg p(s(s(0)), s(s(X)), Y)\} \).

Solution:

Let \( G \) be the set of all ground terms, i.e., \( G = \{s^i(0) \mid i \in \mathbb{N}\} = T(\Sigma) \).

\(a\)

\[
\begin{align*}
\text{trans}_P^0(\emptyset) &= \emptyset \\
\text{trans}_P^1(\emptyset) &= \{p(0, s(t), t) \mid t \in G\} \\
\text{trans}_P^2(\emptyset) &= \{p(s(0), s^2(t), s^2(t)) \mid t \in G\} \cup \text{trans}_P^1(\emptyset) \\
\text{trans}_P^3(\emptyset) &= \{p(s^2(0), s^3(t), s^4(t)) \mid t \in G\} \cup \text{trans}_P^2(\emptyset) \\
&\vdots \\
\text{trans}_P^n(\emptyset) &= \{p(s^i(0), s^{i+1}(t), s^{2i}(t)) \mid t \in G, 0 \leq i < n\}
\end{align*}
\]

\(b\)

\[
\begin{align*}
\text{lfp}(\text{trans}_P) &= \{p(s^i(0), s^{i+1}(t), s^{2i}(t)) \mid t \in G, i \geq 0\} \\
&= \{p(s^i(0), s^{i+j+1}(0), s^{2i+j}(0)) \mid i, j \geq 0\}
\end{align*}
\]

\(c\)

\[
FJP, \{\neg p(s(s(0)), p(s(s(X)), Y)\}] = \{p(s^2(0), s^3(t), s^4(t)) \mid t \in G\}
\]\n
\(= \{p(s^2(0), s^{3+i}(0), s^{4+i}(0)) \mid i \geq 0\}\)
Exercise 4 (Definite Logic Programming): (8 + 6 = 14 points)

a) Implement the predicate `incr/2` in Prolog. This predicate can be used to identify the longest increasing prefix \([a_0, \ldots, a_n]\) of a list \([a_0, \ldots, a_n, a_{n+1}, \ldots, a_m]\) such that for all \(i \in \{0, \ldots, n\}\) it holds that \(a_i = a_0 + i\). The first argument of `incr` is the list to analyze. The second argument is the increasing prefix as described above.

As an example, for the list \([1, 2, 3, 2, 1]\) the result \([1, 2, 3]\) is computed (because \(a_3 = 2\) is not equal to \(1 + 3\)). In Prolog, the corresponding call

\[
\text{incr}([s(0), s(s(0)), s(s(s(0))), s(s(0)), s(0)], \text{Res})
\]

should return the only answer \(\text{Res} = [s(0), s(s(0)), s(s(s(0)))].\)

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

b) The Collatz sequence \(n_0, n_1, \ldots\) for some initial value \(n_0 > 0\) is defined as

\[
n_{i+1} = \begin{cases} 
    n_i/2, & \text{if } n_i \text{ is even} \\
    n_i \times 3 + 1, & \text{otherwise}
\end{cases}
\]

It can easily be seen that if \(n_i = 1\) then the sequence will continue: \(4, 2, 1, 4, 2, 1, \ldots\). We define the function `collatz_len(n_0)` for a start value \(n_0\) as the smallest \(i\) such that \(n_i = 1\). Note that it is a famous open problem if all start values will eventually reach 1 or if there are other loops or diverging sequences.

Some examples for the length of the Collatz sequence:

- `collatz_len(1) = 0`, since \(n_0 = 1\)
- `collatz_len(2) = 1`, since \(n_0 = 2, n_1 = 1\)
- `collatz_len(3) = 7`, due to the Collatz sequence \(3, 10, 5, 16, 8, 4, 2, 1\)
- `collatz_len(4) = 2`

Implement the predicate `collatz_len/2` in Prolog that calculates the length of the Collatz sequence for a given initial value. It may behave arbitrarily if the length of the sequence starting in \(n_0\) is not defined or \(n_0 \leq 0\).

As an example, `collatz_len(3, Z)` gives the answer substitution \(Z = 7\).

**Hints:**

- You may only use the built-in predicate `is/2`, the cut and the usual arithmetic operators such as `mod`, `+`, `*`, `-`, `/`.

**Solution:**
a) incr([],[]).
   incr([X],[X]).
   incr([X,s(X)|XS],[X|YS]) :- incr([s(X)|XS],YS).
   incr([X,Y|_],[X]) :- neq(s(X),Y).

   neq(0,s(_)).
   neq(s(_),0).
   neq(s(X),s(Y)) :- neq(X,Y).

b) collatz(1,0):-!.
   collatz(X,Y):- 0 is X mod 2, XP is X/2, collatz(XP,Z), Y is Z+1.
   collatz(X,Y):- 1 is X mod 2, XP is X*3+1, collatz(XP,Z), Y is Z+1.

____________________________
Consider a function $f : \mathbb{N}^{n+1} \rightarrow \mathbb{N}$. The function $g : \mathbb{N}^n \rightarrow \mathbb{N}$ is defined as:

$$g(k_1, \ldots, k_n) = k \text{ iff } f(k_1, \ldots, k_n, k) = f(k_1, \ldots, k_n, k+1) \text{ and }$$

for all $0 \leq k' < k$ we have $f(k_1, \ldots, k_n, k')$ is defined and $f(k_1, \ldots, k_n, k') \neq f(k_1, \ldots, k_n, k'+1)$

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

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

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

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

$$g(k_1, \ldots, k_n) = k \text{ iff } \mathcal{P} \models \underline{g}(k_1, \ldots, k_n, k).$$

**Solution:**

$$g(X_1, \ldots, X_n, Z) : \neg \underline{f}'(X_1, \ldots, X_n, 0, Z).$$
$$\underline{f}'(X_1, \ldots, X_n, Y, Y) : \neg \underline{f}(X_1, \ldots, X_n, Y, A), \underline{f}(X_1, \ldots, X_n, s(Y), A).$$
$$\underline{f}'(X_1, \ldots, X_n, Y, Z) : \neg \underline{f}(X_1, \ldots, X_n, Y, A), \underline{f}(X_1, \ldots, X_n, s(Y), B), \text{neq}(A, B),$$
$$\underline{f}'(X_1, \ldots, X_n, s(Y), Z).$$
$$\text{neq}(0, s(\_)).$$
$$\text{neq}(s(\_), 0).$$
$$\text{neq}(s(X), s(Y)) : \neg \text{neq}(X, Y).$$
Exercise 6 (Programming with CLP): (9 points)

We use Prolog to find solutions for a problem we call “3-tuple covers”. A “3-tuple cover” of the range \{1,\ldots,n\} (where \(n > 0\) is divisible by 3) is a set of 3-tuples

\[
\{(a_1, a_2, a_3), (a_4, a_5, a_6), \ldots, (a_{n-2}, a_{n-1}, a_n)\}
\]

such that all \(a_i\) are pairwise different (i.e., \(\forall i,j : a_i \neq a_j\)), all \(a_i\) are from the set \{1,\ldots,n\} (i.e., \((a_1, \ldots, a_n)\) is a permutation of \((1, \ldots, n)\)) and for all 3-tuples, the sum of the first two elements is equal to the last one (i.e, \(\forall i \in \{1, \ldots, n\} : \text{if } (i-1) \text{ is divisible by } 3, \text{ then } a_i + a_{i+1} = a_{i+2}\)). We want to write a program that finds such 3-tuple covers for any given \(n\). The program might behave arbitrarily if \(n\) is not divisible by 3 or \(n \leq 0\).

Implement a Prolog predicate \texttt{cover/2} that finds a satisfying solution for a given \(n\) and returns the list \([a_1, \ldots, a_n]\) in its second argument. It should backtrack to find all valid solutions. Make use of the Prolog \texttt{clpfd} library.

Example:

?- cover(3,X).
X = [1, 2, 3] ;
X = [2, 1, 3].

Hints:

- You may only use the built-in predicates \texttt{length/2}, \texttt{ins/2}, \texttt{in/2}, \texttt{#=/2}, \texttt{all_distinct/1}, \texttt{label/1}, the cut and the usual arithmetic operators such as \texttt{mod}, \texttt{+, *, -, /}. Additionally, you may construct \texttt{clpfd} ranges using the \texttt{../2} operator.

- The following is already given: \texttt{:- use_module(library(clpfd)).}

Solution:

\[
:- \text{use_module(library(clpfd)).}.
\]

\[
\text{cover}(N,VS) :- \text{length}(VS,N),
\quad VS \text{ ins } 1..N, \text{ all_distinct(VS), sums_match(VS),}
\quad label(VS).
\]

\[
\text{sums_match([],)}.
\]

\[
\text{sums_match([A,B,C|VS]) :- A+B #=} C, \text{ sums_match(VS).}
\]

\[
\]