Prof. Dr. Jürgen Giesl

Carsten Fuhs, Carsten Otto, Thomas Ströder

Master Exam Version V3M

First Name: _____

Last Name: _____

Immatriculation Number: _____

Course of Studies (please mark exactly one):

- ○ **Informatik Bachelor (for Master)**
- ○ **Informatik Master**
- ○ **SSE Master**
- ○ **Other:** _____

	Maximal Points	Achieved Points
Exercise 1	10	
Exercise 2	16	
Exercise 3	9	
Exercise 4	10	
Exercise 5	10	
Exercise 6	5	
Total	60	
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**.

Name: **Immatriculation Number:**

Exercise 1 (Theoretical Foundations): (6 + 4 = 10 points)

Let

$$\varphi \;=\; \mathtt{p}(0) \;\wedge\; \exists X\, \neg \mathtt{p}(X) \;\wedge\; \forall Y\, (\,\mathtt{p}(Y) \rightarrow \mathtt{p}(\mathtt{s}(\mathtt{s}(Y)))\,) \quad \text{and}$$

$$\psi \;=\; \exists Z\, \forall U\, (\,\mathtt{p}(U) \rightarrow \mathtt{p}(\mathtt{s}(Z))\,)$$

be formulas over the signature (Σ, Δ) with $\Sigma = \Sigma_0 \cup \Sigma_1$, $\Sigma_0 = \{\mathtt{0}\}$, $\Sigma_1 = \{\mathtt{s}\}$, and $\Delta = \Delta_1 = \{\mathtt{p}\}$.

a) Prove that $\varphi \models \psi$ by means of resolution.

Hint: First transform the formula $\varphi \wedge \neg\psi$ into a satisfiability-equivalent Skolem normal form, and then convert this Skolem normal form to an equivalent clause set.

b) Explicitly give a Herbrand model of the formula φ (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.

Exercise 2 (Procedural Semantics, SLD tree): (7 + 9 = 16 points)

Consider the following Prolog program $\mathcal{P}$ which can be used to divide two numbers represented by terms built from $0 \in \Sigma_0$ and $\mathtt{s} \in \Sigma_1$ (i.e., $0 \mathrel{\hat{=}} 0, 1 \mathrel{\hat{=}} \mathtt{s}(0), 2 \mathrel{\hat{=}} \mathtt{s}(\mathtt{s}(0)), \ldots$):

```
div(X, 0, Z) :- !, fail.
div(0, s(Y), Z) :- eq(Z, 0), !.
div(X, Y, s(Z)) :- sub(X, Y, U), div(U, Y, Z).
sub(0, Y, 0).
sub(s(X), 0, s(X)).
sub(s(X), s(Y), Z) :- sub(X, Y, Z).
eq(X, X).
```

a) The program $\mathcal{P}'$ results from $\mathcal{P}$ by **removing the cuts**. Consider the following query:

```
?- div(s(0), s(0), s(s(0))).
```

For the logic program $\mathcal{P}'$, i.e., **without the cuts**, please show a successful computation for the query above (i.e., a computation of the form $(G, \varnothing) \vdash_{\mathcal{P}'}^{+} (\square, \sigma)$ where $G = \{\neg \mathtt{div}(\mathtt{s}(0), \mathtt{s}(0), \mathtt{s}(\mathtt{s}(0)))\}$). You do not need to extend the substitution, i.e., it suffices to write $\varnothing$ in each step.

b) Please give a graphical representation of the SLD tree for the query `?- div(s(s(0)), s(0), X).` in the program $\mathcal{P}$ (i.e., **with the cuts**). Also give all answer substitutions explicitly.

We repeat the program below.

```
div(X, 0, Z) :- !, fail.
div(0, s(Y), Z) :- eq(Z, 0), !.
div(X, Y, s(Z)) :- sub(X, Y, U), div(U, Y, Z).
sub(0, Y, 0).
sub(s(X), 0, s(X)).
sub(s(X), s(Y), Z) :- sub(X, Y, Z).
eq(X, X).
```

Exercise 3 (Fixpoint Semantics): (3 + 3 + 3 = 9 points)

Consider the following logic program $\mathcal{P}$ over the signature (Σ, Δ) with $\Sigma = \{0, s\}$ and $\Delta = \{plus\}$.

```
plus(X, 0, X).
plus(0, X, X).
plus(s(X), s(Y), s(s(Z))) :- plus(X, Y, Z).
```

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

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

c) Give $F[\![\mathcal{P}, \{\neg \texttt{plus(X, Y, s(s(Z)))}\}]\!]$.

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 as:

$g(k_1, \ldots, k_n) = k$ iff $f(k_1, \ldots, k_n, k)$ is odd and
for all $0 \leq k' < k$ we have that $f(k_1, \ldots, k_n, k')$ is defined and $f(k_1, \ldots, k_n, k')$ is even

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

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

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

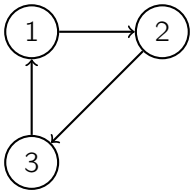
Here, numbers are represented by terms built from $\mathtt{0} \in \Sigma_0, \mathtt{s} \in \Sigma_1$ (i.e., $\underline{0} = \mathtt{0}, \underline{1} = \mathtt{s}(\mathtt{0}), \underline{2} = \mathtt{s}(\mathtt{s}(\mathtt{0})), \ldots$).

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

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

Exercise 5 (Definite Logic Programming): (10 points)

A directed graph is a pair (V, E) where V is the set of vertices and $E \subseteq V \times V$ is the set of edges. We represent such a graph by two lists where the first list contains the names of the nodes and the second list contains an element `e(`n_1`,`n_2`)` iff there is an edge from n_1 to n_2 in the represented graph. Both lists do not contain duplicates. For example, consider the following graph $(\{1, 2, 3\}, \{(1, 2), (3, 1), (2, 3)\})$:



This graph is represented by the two lists `[1,2,3]` and `[e(1,2),e(3,1),e(2,3)]`.

Implement the predicate `hamiltonian/2` in Prolog. A call `hamiltonian(`t_1`,`t_2`)` works as follows. If both t_1 and t_2 are finite lists representing a directed graph as described above, the call succeeds iff there is a hamiltonian cycle in the represented graph. If t_1 and t_2 are not of the form described above, then `hamiltonian(`t_1`,`t_2`)` may behave arbitrarily. You **must not use** any built-in predicates in this exercise. Note that meta-programming (e.g., using variables without a surrounding predicate) also uses built-in predicates (implicitly) and is, thus, not allowed in this exercise.

A hamiltonian cycle exists in a directed graph iff there is a path through the graph starting and ending in the same node (thus, visiting this node exactly twice) and visiting all other nodes exactly once. The empty graph has no hamiltonian cycle. The following example calls to `hamiltonian/2` illustrate its definition:

- `?- hamiltonian([1,2,3],[e(1,2),e(3,1),e(2,3)]).` succeeds with the empty answer substitution
- `?- hamiltonian([1,2,3],[e(1,2),e(2,1),e(1,3),e(3,1)]).` fails

Hint: Use a helper predicate `tour/4` *with the following four arguments: (1) the node where you currently are, (2) the list of nodes yet to visit, (3) the node you have to reach in the end, and (4) the list of edges in the graph. Moreover, it can be helpful to define a predicate* `choice/3` *for the non-deterministic choice of the next node to visit. Here,* `choice(`t_1`,`t_2`,`t_3`)` *is true if t_1 is a list containing t_2 and t_3 is the list t_1 where the element t_2 was deleted.*

Exercise 6 (Arithmetic): (5 points)

Implement the predicate `mean/2` in Prolog. A call of `mean(`t_1`,`t_2`)` works as follows. If t_1 is a finite non-empty list of integers, then t_2 is unified with the *rounded mean value* of all numbers in t_1. If t_1 is the empty list, `mean/2` fails. If t_1 is not a finite list of integers, `mean/2` may behave arbitrarily.

We define the *rounded mean value* of $n > 0$ integers $a_1, \ldots, a_n$ as $\left\lfloor \frac{\sum_{i=1}^{n} a_i}{n} \right\rfloor$ if $\frac{\sum_{i=1}^{n} a_i}{n}$ is positive and $\left\lceil \frac{\sum_{i=1}^{n} a_i}{n} \right\rceil$ if $\frac{\sum_{i=1}^{n} a_i}{n}$ is negative.

The following example calls to `mean/2` illustrate its definition:

- `?- mean([],X).` fails
- `?- mean([1,2,3],1).` fails
- `?- mean([1,2,3],X).` succeeds with the answer substitution `X = 2`
- `?- mean([1,2],1).` succeeds with the empty answer substitution
- `?- mean([1,-2],X).` succeeds with the answer substitution `X = 0`

Hint: In Prolog, the term `X // Y` *can be used to compute* $\left\lfloor \frac{X}{Y} \right\rfloor$ *if* $\frac{X}{Y}$ *is positive and* $\left\lceil \frac{X}{Y} \right\rceil$ *if* $\frac{X}{Y}$ *is negative.*