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**
- ○ **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**.

Exercise 1 (Theoretical Foundations): **(3 + 3 + 4 = 10 points)**

Let $\varphi = \mathtt{q}(0, \mathtt{s}(0)) \wedge \forall X, Y\, (\mathtt{q}(X, Y) \rightarrow \mathtt{q}(\mathtt{s}(X), \mathtt{s}(Y)))$ and $\psi = \exists Z\, \mathtt{q}(\mathtt{s}(Z), \mathtt{s}(\mathtt{s}(Z)))$ be formulas over the signature (Σ, Δ) with $\Sigma = \Sigma_0 \cup \Sigma_1, \Sigma_0 = \{0\}, \Sigma_1 = \{\mathtt{s}\}$, and $\Delta = \Delta_2 = \{\mathtt{q}\}$.

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

Hint: First transform the formula $\varphi \wedge \neg\psi$ into 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.

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 $\mathcal{P}$ which can be used to sort a list of numbers using the *bubblesort* algorithm:

```
bubble(L, R) :- swap(L, N), !, bubble(N, R).
bubble(L, L).
swap([A,B|L]), [B,A|L]) :- B < A.
swap([A|L], [A|N]) :- swap(L, N).
```

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

```
0 < 1.
1 < 2.
0 < 2.
...
```

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

```
?- bubble([2,1,0], [1,2,X]).
```

For the logic program $\mathcal{P}'$, i.e. **without the cut**, please show a successful computation for the query above (i.e., a computation of the form $(G, \varnothing) \vdash^{+}_{\mathcal{P}'} (\Box, \sigma)$ where $G = \{\neg \texttt{bubble}([2,1,0],[1,2,X])\}$). 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., **with the cut**).

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

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

```
gt(s(X), 0).
gt(s(X), s(Y)) :- gt(X, Y).
```

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

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

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

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, $\hat{f}(4, \mathbf{0}) = \mathbf{4}$, $\hat{f}(4, \mathbf{1}) = \mathbf{2}$, $\hat{f}(4, \mathbf{2}) = \mathbf{2}$.

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)

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
$[[l_1^1, l_2^1, \ldots, l_{k_1}^1], [l_1^2, l_2^2, \ldots, l_{k_2}^2], \ldots, [l_1^n, l_2^n, \ldots, l_{k_n}^n]]$
where all l_i^j are of the form `pos(X)` or `neg(X)` for some Prolog variables `X`. The list t represents a set of clauses where `pos(X)` stands for the propositional variable X while `neg(X)` stands for its negation. A call `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 `solve(`t`)` may behave arbitrarily. You **must not use** any built-in predicates in this exercise. The following example calls to `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 `pos(1)` *or* `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