Prof. Dr. Jürgen Giesl

Cornelius Aschermann, Jera Hensel

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	11	
Exercise 2	9	
Exercise 3	10	
Exercise 4	14	
Exercise 5	7	
Exercise 6	9	
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): **(4 + 4 + 3 = 11 points)**

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

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

Exercise 2 (Procedural Semantics, SLD tree): (5 + 4 = 9 points)

Consider the following Prolog program $\mathcal{P}$.

```
a(X,Y):-b(s(X)).
a(X,Y):-b(Y),!,c(X).
a(s(X),s(Y)):-a(X,Y).
c(s(0)).
b(0).
b(1).
```

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 $(G, \varnothing) \vdash^{+}_{\mathcal{P}'} (\Box, \sigma)$ 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 `!`, 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.

Exercise 3 (Fixpoint Semantics): **(5 + 2 + 3 = 10 points)**

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

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

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

c) Give $F[\![\mathcal{P}, \{\neg \mathtt{p(s(s(0)),\ s(s(X)),\ Y)}\}]\!]$.

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 $[a_0, \dots, a_n]$ of a list $[a_0, \dots, a_n, a_{n+1}, \dots, a_m]$ such that for all $i \in \{0, \dots, n\}$ 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

```
incr([s(0), s(s(0)), s(s(s(0))), s(s(0)), s(0)], Res)
```

should return the only answer `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 $n_0, n_1, \dots$ for some initial value $n_0 > 0$ is defined as

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

It can easily be seen that if $n_i = 1$ then the sequence will continue: $4, 2, 1, 4, 2, 1, \dots$. 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`, `+`, `*`, `-`, `/`.

Exercise 5 (Universality): (7 points)

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

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

As an example, consider the function $\hat{f} : \mathbb{N}^2 \to \mathbb{N}$ with $\hat{f}(x, y) = \max\{x - 3y, 1\}$. 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 $\hat{f}(x, y) = \hat{f}(x, y+1)$. Indeed, $\hat{f}(6, \mathbf{0}) = \mathbf{6}$, $\hat{f}(6, \mathbf{1}) = \mathbf{3}$, $\hat{f}(6, \mathbf{2}) = \hat{f}(6, \mathbf{3}) = \mathbf{1}$.

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 the cut or any other built-in predicate**):

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

Exercise 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, $\forall i \in \{1,\ldots,n\}$: if $(i-1)$ is divisible by 3, 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 `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 `clpfd` library.

Example:

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

Hints:

- You may only use the built-in predicates `length/2`, `ins/2`, `in/2`, `#=/2`, `all_distinct/1`, `label/1`, the cut and the usual arithmetic operators such as `mod`, `+`, `*`, `-`, `/`. Additionally, you may construct `clpfd` ranges using the `../2` operator.
- The following is already given: `:- use_module(library(clpfd)).`