Prof. Dr. Jürgen Giesl — Carsten Otto

Bachelor/Master Exam Version V3B

First Name: ______________________

Last Name: ______________________

Immatriculation Number: ______________________

Course of Studies (please mark exactly one):

○ **Informatik Bachelor** ○ **Mathematik Master**
○ **TK Master** ○ **Other:** ______________

	Maximal Points	Achieved Points
Exercise 1	8	
Exercise 2	16	
Exercise 3	12	
Exercise 4	10	
Exercise 5	10	
Exercise 6	4	
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): (5 + 3 = 8 points)

Let $\varphi = \mathrm{p}(\mathrm{s}^2(0), 0) \wedge \forall X\,(\mathrm{p}(\mathrm{s}^2(X), X) \to \mathrm{p}(\mathrm{s}^4(X), \mathrm{s}^2(X))) \wedge \neg\mathrm{p}(\mathrm{s}^3(0), \mathrm{s}(0))$ and $\psi = \exists Y\, \mathrm{p}(\mathrm{s}^6(0), Y)$ be formulas over the signature (Σ, Δ) with $\Sigma = \Sigma_0 \cup \Sigma_1, \Sigma_0 = \{0\}, \Sigma_1 = \{\mathrm{s}\}$, and $\Delta = \Delta_2 = \{\mathrm{p}\}$. Here, $\mathrm{s}^2(0)$ stands for $\mathrm{s}(\mathrm{s}(0))$, etc.

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.

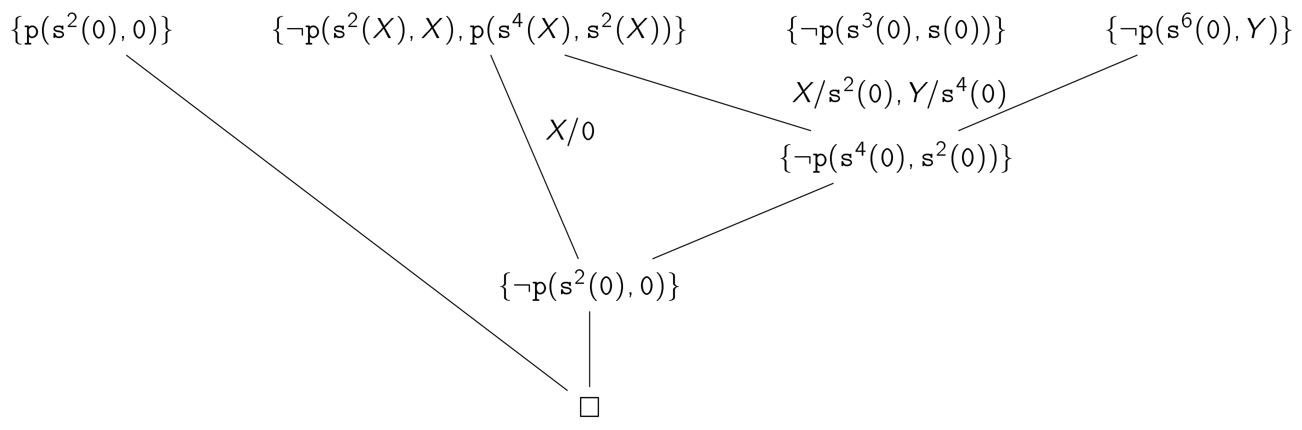
Solution:

a)

$$\begin{aligned}
\varphi \wedge \neg\psi &\Leftrightarrow \mathrm{p}(\mathrm{s}^2(0), 0) \wedge \forall X\,\big(\mathrm{p}(\mathrm{s}^2(X), X) \to \mathrm{p}(\mathrm{s}^4(X), \mathrm{s}^2(X))\big) \wedge \neg\mathrm{p}(\mathrm{s}^3(0), \mathrm{s}(0)) \wedge \neg\exists Y\,\big(\mathrm{p}(\mathrm{s}^6(0), Y)\big)\\
&\Leftrightarrow \mathrm{p}(\mathrm{s}^2(0), 0) \wedge \forall X\,\big(\neg\mathrm{p}(\mathrm{s}^2(X), X) \vee \mathrm{p}(\mathrm{s}^4(X), \mathrm{s}^2(X))\big) \wedge \neg\mathrm{p}(\mathrm{s}^3(0), \mathrm{s}(0)) \wedge \neg\exists Y\,\big(\mathrm{p}(\mathrm{s}^6(0), Y)\big)\\
&\Leftrightarrow \mathrm{p}(\mathrm{s}^2(0), 0) \wedge \forall X\,\big(\neg\mathrm{p}(\mathrm{s}^2(X), X) \vee \mathrm{p}(\mathrm{s}^4(X), \mathrm{s}^2(X))\big) \wedge \neg\mathrm{p}(\mathrm{s}^3(0), \mathrm{s}(0)) \wedge \forall Y\,\big(\neg\mathrm{p}(\mathrm{s}^6(0), Y)\big)\\
&\Leftrightarrow \forall X, Y\,\big(\mathrm{p}(\mathrm{s}^2(0), 0) \wedge (\neg\mathrm{p}(\mathrm{s}^2(X), X) \vee \mathrm{p}(\mathrm{s}^4(X), \mathrm{s}^2(X))) \wedge \neg\mathrm{p}(\mathrm{s}^3(0), \mathrm{s}(0)) \wedge \neg\mathrm{p}(\mathrm{s}^6(0), Y)\big)
\end{aligned}$$

Thus, the equivalent clause set for $\varphi \wedge \neg\psi$ is
$\{\mathrm{p}(\mathrm{s}^2(0), 0)\}, \{\neg\mathrm{p}(\mathrm{s}^2(X), X), \mathrm{p}(\mathrm{s}^4(X), \mathrm{s}^2(X))\}, \{\neg\mathrm{p}(\mathrm{s}^3(0), \mathrm{s}(0))\}, \{\neg\mathrm{p}(\mathrm{s}^6(0), Y)\}$.

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


$\{\mathrm{p}(\mathrm{s}^2(0), 0)\}$
$\{\neg\mathrm{p}(\mathrm{s}^2(X), X), \mathrm{p}(\mathrm{s}^4(X), \mathrm{s}^2(X))\}$
$\{\neg\mathrm{p}(\mathrm{s}^3(0), \mathrm{s}(0))\}$
$\{\neg\mathrm{p}(\mathrm{s}^6(0), Y)\}$
$X/\mathrm{s}^2(0), Y/\mathrm{s}^4(0)$
$X/0$
$\{\neg\mathrm{p}(\mathrm{s}^4(0), \mathrm{s}^2(0))\}$
$\{\neg\mathrm{p}(\mathrm{s}^2(0), 0)\}$
$\square$

Hence, we have proven $\{\varphi\} \models \psi$.

$\square$

b) We have $S \models \varphi$ for the Herbrand structure $S = (\mathcal{T}(\Sigma), \alpha)$ with $\alpha_0 = 0, \alpha_{\mathtt{s}}(t) = \mathtt{s}(t)$, and $\alpha_{\mathtt{p}} = \{(\mathtt{s}^{i+2}(0), \mathtt{s}^{i}(0)) \mid i \geq 0 \wedge i \neq 1\}$

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':

```
replace([],        []).
replace([b,a|XS], [z,z|YS]) :- replace(XS, YS).
replace([X|XS],   [X|YS])   :- replace(XS, YS).
```

For example, the query `?- 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:

`?- replace([a,b,b,a], Res).`

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

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

`?- replace([a,b,b,a], 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 ocurrences of 'ba' by 'zz'.

For example, now the query `?- replace([b,a,b,a], Z)` must have the only answer substitution `Z = [z,z,z,z]`.

Solution:

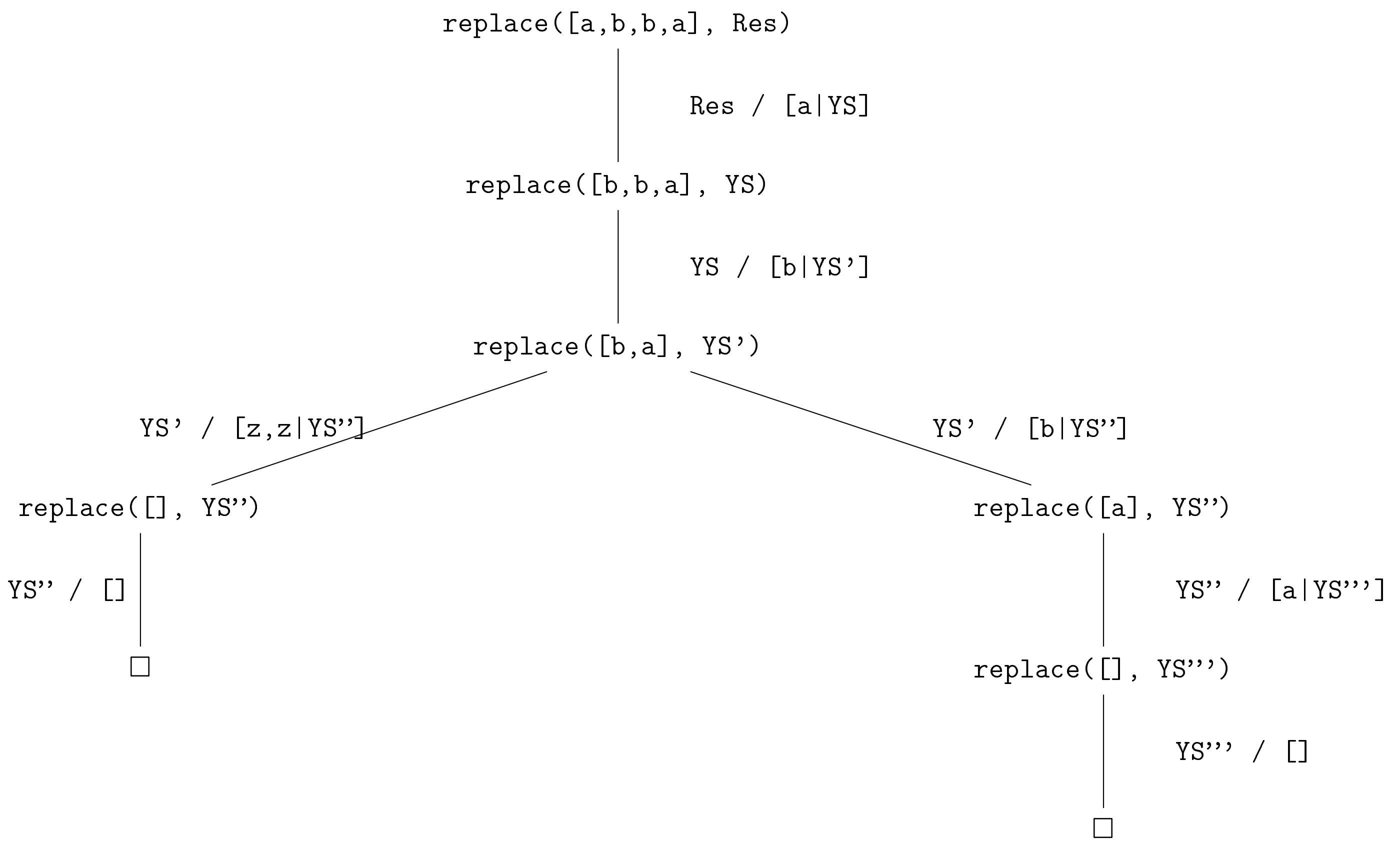
a)

$$
\begin{aligned}
& (\{\neg\texttt{replace([a,b,b,a], Res)}\}, \varnothing) \\
\vdash_{\mathcal{P}}\ & (\{\neg\texttt{replace([b,b,a], YS)}\}, \{\texttt{Res / [a|YS]}\}) \\
\vdash_{\mathcal{P}}\ & (\{\neg\texttt{replace([b,a], YS')}\}, \{\texttt{Res / [a,b|YS']}\}) \\
\vdash_{\mathcal{P}}\ & (\{\neg\texttt{replace([], YS'')}\}, \{\texttt{Res / [a,b,z,z|YS'']}\}) \\
\vdash_{\mathcal{P}}\ & (\Box, \{\texttt{Res / [a,b,z,z]}\})
\end{aligned}
$$

Alternative:

$$
\begin{aligned}
& (\{\neg\texttt{replace([a,b,b,a], Res)}\}, \varnothing) \\
\vdash_{\mathcal{P}}\ & (\{\neg\texttt{replace([b,b,a], YS)}\}, \{\texttt{Res / [a|YS]}\}) \\
\vdash_{\mathcal{P}}\ & (\{\neg\texttt{replace([b,a], YS')}\}, \{\texttt{Res / [a,b|YS']}\}) \\
\vdash_{\mathcal{P}}\ & (\{\neg\texttt{replace([a], YS'')}\}, \{\texttt{Res / [a,b,b|YS'']}\}) \\
\vdash_{\mathcal{P}}\ & (\{\neg\texttt{replace([], YS''')}\}, \{\texttt{Res / [a,b,b,a|YS''']}\}) \\
\vdash_{\mathcal{P}}\ & (\Box, \{\texttt{Res / [a,b,b,a]}\})
\end{aligned}
$$

b)



c)
```
replace([],        []).
replace([b,a|XS], [z,z|YS]) :- !, replace(XS, YS).
replace([X|XS],    [X|YS])   :- replace(XS, YS).
```

Exercise 3 (Definite Logic Programming): (12 points)

Implement the predicate `noDupl/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 `noDupl` 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 `noDupl([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:

```
noDupl(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 4 (Meta-Programming): (10 points)

Consider a set $M \subseteq \mathbb{N}$. Let $f_M : \mathbb{N} \to \{0,1\}$ be the characteristic function of M:

$$f_M(x) = \begin{cases} 1 & \text{if } x \in M \\ 0 & \text{if } x \notin M \end{cases}$$

In Prolog we encode the function f_M using the predicate `fM/1` such that `fM(`x`)` is true if $f_M(x) = 1$ and it is false if $f_M(x) = 0$.
Please implement the predicate `filter/3` that can be used to filter a list of numbers based on a characteristic function as described above.
The first argument of `filter` is a list of numbers. The second argument contains a function symbol `f/0`. Here, we assume that for this `f/0` there exists a predicate symbol `f/1` which is a characteristic function for a set as described above. The third argument is the result of filtering the list in the first argument using the predicate in the second argument: the list in the third argument contains exactly those elements of the input list (in that order) for which the predicate in the second argument is true.
For example, assuming that `fODD/1` is a predicate that is true exactly for odd numbers, the query `?- filter([1,2,3], fODD, X)` is evaluated by executing the calls `fODD(1)` (which is true), `fODD(2)` (which is false), and `fODD(3)` (which is true). It yields the answer substitution `X = [1,3]`.
Hints:

- You may use the built-in predicate `=../2`.
- When called with the empty list as input, the result always is the empty list.
- You may use the cut (`!`) or negation as failure.

Solution:

```
filter([], _, []).
filter([X|XS], F, [X|YS]) :- G =.. [F, X],
                             G,
                             filter(XS, F, YS).
filter([X|XS], F, [YS]) :- G =.. [F, X],
                           \+ G,
                           filter(XS, F, YS).
```

With cut:

```
filter([], _, []).
filter([X|XS], F, [X|YS]) :- G =.. [F, X],
                             G,
                             !,
                             filter(XS, F, YS).
filter([X|XS], F, [YS]) :- filter(XS, F, YS).
```

Exercise 5 (Difference Lists): (10 points)

Below you can find an implementation of the Quicksort algorithm that can be used to sort lists. Here, the implementations of the predicate `partition` and the pre-defined predicate `append` are not shown[1].

```
quicksort([], []).
quicksort([X|XS], Sorted) :-
  partition(XS, X, Left, Right),
  quicksort(Right, RS),
  quicksort(Left, LS),
  append(LS, [X|RS], Sorted).
```

Based on this implementation, please implement a modified variant of the Quicksort algorithm. You should implement a predicate `qs/2` where, like in the case of `quicksort/2` above, the first argument is the input list in standard list representation. In the second argument of `qs`, you should make use of difference lists.
Instead of using `append` (working on standard lists), make use of the following fact that can be used to append two difference lists in a single step: `app(A-B, B-C, A-C).`
The call `qs(`*x*`, Output - [])` for a list *x* in standard representation should compute the answer substitution `Output =` *y* where *y* is a standard list containing the entries of *x* in sorted order. If `quicksort(`*x*`,` *y*`)` holds then `qs(`*x*`,` *y* `- [])` holds. As an example, `qs([3,1,2], X)` should compute an answer substitution like `X = [1,2,3|Y] - Y` and `qs([3,1,2], X - [])` should compute the answer substitution `X = [1,2,3]`.
Hint: You do not need to change the implementation of `partition` or change the call to it.

Solution:

```
qs([], X-X).
qs([X|XS], Sorted-SR) :-
  partition(XS, X, Left, Right),
  qs(Right, RS-RSR),
  qs(Left, LS-LSR),
  app(LS-LSR, [X|RS]-RSR, Sorted-SR).
```

Simplified version:

```
qs([], X-X).
qs([X|XS], LS-RSR) :-
  partition(XS, X, Left, Right),
  qs(Right, RS-RSR),
  qs(Left, LS-[X|RS]).
```

[1] `partition(XS, X, Left, Right)` copies the elements of the list `XS` into two lists `Left` and `Right` which contain the values of `XS` that are smaller or equal resp. bigger than the pivot element `X`. As an example, `partition([7,4,6,5], 5, [4,5], [7,6])` is true.

Exercise 6 (Arithmetic): (4 points)

Tetration is the logical extension of multiplication and exponentiation:

multiplication	$a \cdot n$	$:=$	$\underbrace{a + a + \cdots + a}_{n}$
exponentation	a^n	$:=$	$\underbrace{a \cdot a \cdot \ldots \cdot a}_{n}$
tetration	$a \upuparrows n$	$:=$	$\underbrace{a^{\left(a^{\cdot^{\cdot^{(a^a)}}}\right)}}_{n}$

Examples:

- $4 \upuparrows 2 = 4^4 = 256$
- $1 \upuparrows 3 = 1^{(1^1)} = 1^1 = 1$
- $2 \upuparrows 4 = 2^{\left(2^{(2^2)}\right)} = 2^{(2^4)} = 2^{16} = 65.536$

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 $x \upuparrows y$.
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:

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