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	11	
Exercise 2	14	
Exercise 3	10	
Exercise 4	11	
Exercise 5	9	
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 + 4 + 4 = 11 points)

Let $\varphi = \mathrm{p}(0, \mathrm{s}(0), \mathrm{s}(\mathrm{s}(0))) \wedge \forall X, Y, Z\,(\mathrm{p}(X, Y, Z) \rightarrow \mathrm{p}(Y, Z, \mathrm{s}(\mathrm{s}(X))))$ and $\psi = \exists A\,\mathrm{p}(A, \mathrm{s}(A), \mathrm{s}(A))$ be formulas over the signature (Σ, Δ) with $\Sigma = \Sigma_0 \cup \Sigma_1, \Sigma_0 = \{0\}, \Sigma_1 = \{\mathrm{s}\}$, and $\Delta = \Delta_3 = \{\mathrm{p}\}$.

a) Prove that $\{\varphi\} \models \psi$ by means of input 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.

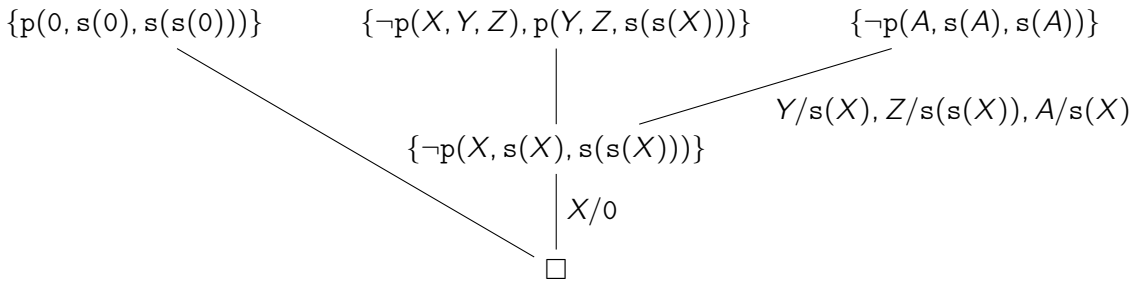
c) Let $\varphi_1, \ldots, \varphi_k, \psi' \in \mathcal{F}(\Sigma', \Delta', \mathcal{V})$ for some signature (Σ', Δ'). Prove that $\{\varphi_1, \ldots, \varphi_k\} \models \psi'$ holds if and only if $\varphi_1 \wedge \ldots \wedge \varphi_k \wedge \neg\psi'$ is unsatisfiable.

Solution:

a)

$$
\begin{aligned}
\varphi \wedge \neg\psi &\Leftrightarrow \mathrm{p}(0, \mathrm{s}(0), \mathrm{s}(\mathrm{s}(0))) \wedge \forall X, Y, Z\,(\mathrm{p}(X, Y, Z) \rightarrow \mathrm{p}(Y, Z, \mathrm{s}(\mathrm{s}(X)))) \wedge \neg\exists A\,\mathrm{p}(A, \mathrm{s}(A), \mathrm{s}(A)) \\
&\Leftrightarrow \mathrm{p}(0, \mathrm{s}(0), \mathrm{s}(\mathrm{s}(0))) \wedge \forall X, Y, Z\,(\neg\mathrm{p}(X, Y, Z) \vee \mathrm{p}(Y, Z, \mathrm{s}(\mathrm{s}(X)))) \wedge \neg\exists A\,\mathrm{p}(A, \mathrm{s}(A), \mathrm{s}(A)) \\
&\Leftrightarrow \mathrm{p}(0, \mathrm{s}(0), \mathrm{s}(\mathrm{s}(0))) \wedge \forall X, Y, Z\,(\neg\mathrm{p}(X, Y, Z) \vee \mathrm{p}(Y, Z, \mathrm{s}(\mathrm{s}(X)))) \wedge \forall A\,\neg\mathrm{p}(A, \mathrm{s}(A), \mathrm{s}(A)) \\
&\Leftrightarrow \forall X, Y, Z, A\,(\mathrm{p}(0, \mathrm{s}(0), \mathrm{s}(\mathrm{s}(0))) \wedge (\neg\mathrm{p}(X, Y, Z) \vee \mathrm{p}(Y, Z, \mathrm{s}(\mathrm{s}(X)))) \wedge \neg\mathrm{p}(A, \mathrm{s}(A), \mathrm{s}(A)))
\end{aligned}
$$

Thus, the equivalent clause set for $\varphi \wedge \neg\psi$ is
$\{\mathrm{p}(0, \mathrm{s}(0), \mathrm{s}(\mathrm{s}(0)))\}, \{\neg\mathrm{p}(X, Y, Z), \mathrm{p}(Y, Z, \mathrm{s}(\mathrm{s}(X)))\}, \{\neg\mathrm{p}(A, \mathrm{s}(A), \mathrm{s}(A))\}$.
We perform input resolution on this clause set to show $\{\varphi\} \models \psi$.



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

□

b) We have $S \models \varphi$ for the Herbrand structure $S = (\mathcal{T}(\Sigma), \alpha)$ with $\alpha_0 = 0, \alpha_{\mathrm{s}}(t) = \mathrm{s}(t)$, and

$$
\begin{aligned}
\alpha_{\mathrm{p}} = \{&(\mathrm{s}^{2i}(0), \mathrm{s}^{2i+1}(0), \mathrm{s}^{2i+2}(0)) && \mid i \geq 0\} \\
\cup\, \{&(\mathrm{s}^{2i+1}(0), \mathrm{s}^{2i+2}(0), \mathrm{s}^{2i+2}(0)) && \mid i \geq 0\} \\
\cup\, \{&(\mathrm{s}^{2i+2}(0), \mathrm{s}^{2i+2}(0), \mathrm{s}^{2i+3}(0)) && \mid i \geq 0\}
\end{aligned}
$$

Alternative solution: $\alpha_{\mathrm{p}} = \{(\mathrm{s}^i(0), \mathrm{s}^j(0), \mathrm{s}^k(0)) \mid i, j, k \geq 0\}$

c)

$\{\varphi_1, \ldots, \varphi_k\} \models \psi'$
iff for all interpretations I with $I \models \{\varphi_1, \ldots, \varphi_k\}$ we have $I \models \psi'$
iff there is no interpretation I with $I \models \{\varphi_1, \ldots, \varphi_k\}$ and $I \models \neg\psi'$
iff $\varphi_1 \wedge \ldots \wedge \varphi_k \wedge \neg\psi'$ is unsatisfiable

Exercise 2 (Procedural Semantics, SLD tree): (7 + 7 = 14 points)

Consider the following Prolog program $\mathcal{P}$ which can be used to replace the letter 'a' by the letter 'ä' in words:

```
transform([]     , []).
transform([a|XS], [ä|YS]) :- !, transform(XS, YS).
transform([X|XS], [X|YS]) :- transform(XS, YS).
```

As an example, the query `transform([b,a,n,a,n,a], Z)` would give the answer substitution `Z = [b,ä,n,ä,n,ä]`.

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

```
?- transform([a,n,a], [a|XS]).
```

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 $(G, \varnothing) \vdash^{+}_{\mathcal{P}'} (\square, \sigma)$ where $G = \{\neg$`transform([a,n,a], [a|XS])`$\}$). It suffices to give substitutions only for those variables which are used to define the value of the variable `XS` in the query.

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

```
?- transform([a,n,a], XS).
```

in the program $\mathcal{P}$ (i.e., **with the cut**). For every part of a 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.

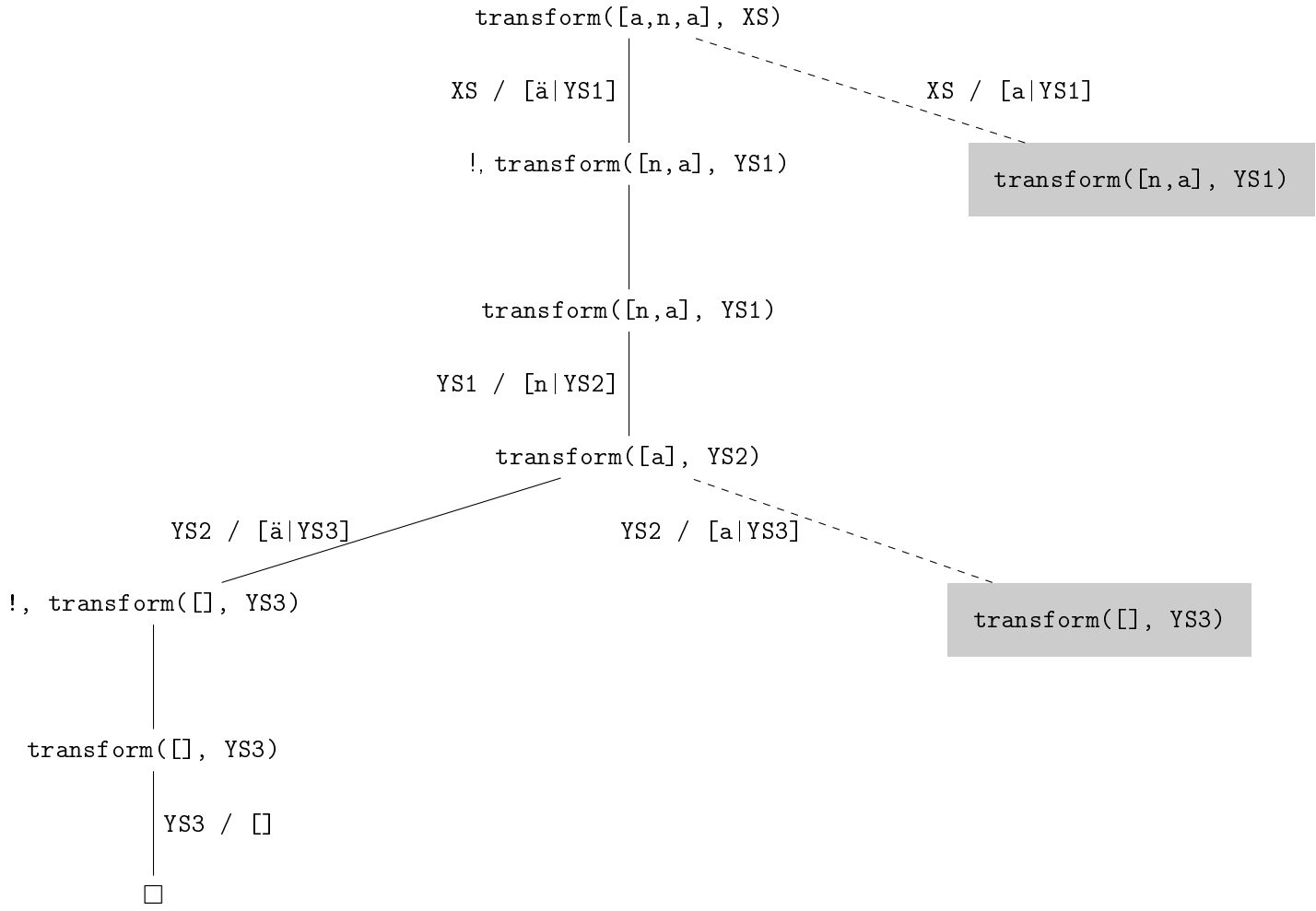
Solution:

a)

$$
\begin{aligned}
&(\{\neg\texttt{transform([a,n,a], [a|XS])}\}, \varnothing)\\
\vdash_{\mathcal{P}'}\ &(\{\neg\texttt{transform([n,a], XS)}\}, \varnothing\})\\
\vdash_{\mathcal{P}'}\ &(\{\neg\texttt{transform([a], XS')}\}, \{\texttt{XS / [n|XS']}\})\\
\vdash_{\mathcal{P}'}\ &(\{\neg\texttt{transform([], XS'')}\}, \{\texttt{XS / [n,a|XS'']}\})\\
\vdash_{\mathcal{P}'}\ &(\square, \{\texttt{XS / [n,a]}\})
\end{aligned}
$$

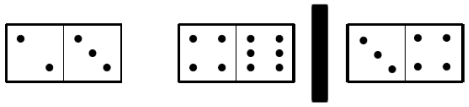
b)

In this representation, the nodes and edges deleted by the cut are shown with a gray background and dashed edges, respectively.



Exercise 3 (Definite Logic Programming): (10 points)

Implement the predicate `addDomino/2` in Prolog. This predicate can be used in the setting of the domino game to check if a given piece can be inserted in an already existing series of domino pieces.
Every domino piece consists of two numbers, a left number and a right number. A list of pieces is called *valid* only if for each domino piece α which is directly followed by another domino piece β, the right number of α is identical to the left number of β.
In the following picture you can see that the first two pieces do not form a valid list because the left number of the second piece is 4 instead of 3. However, when adding the third piece after the first piece and before the second piece, the resulting list is valid: the left number 3 of the added piece matches the right number of the first piece, and its right number 4 matches the left number of the second piece.



In Prolog we represent a single domino piece using the predicate `p/2`. The list containing the first two pieces of the picture above is represented as `[p(2,3), p(4,6)]`. The predicate `addDomino/2` is true if and only if the piece in the second argument can be inserted between two pieces of the list in the first argument (or at the end) so that the resulting list of pieces is valid. Following the example above, `addDomino([p(2,3),p(4,6)], p(3,4))` should be `true` because adding the piece `p(3,4)` after `p(2,3)` is possible.

Important: You may not use predefined predicates!

Hints:

- Pieces cannot be rotated. Because of this, `addDomino([p(2,3),p(4,6)], p(4,3))` is `false`.
- You only need to check if the piece can be added. You do not have to compute the list that results from adding the piece.
- To check if a list `XS` of pieces is valid you can check if `addDomino(XS, p(Z,Z))` for a fresh variable `Z` is `true`.
- Every list containing at most one domino piece is valid.

Solution:

```
addDomino([]                   , _      ).
addDomino([p(A, B)]            , p(B, _)).
addDomino([p(A, B),p(C, D)|XS], p(B, C)) :- addDomino([p(C, D)|XS], p(Z, Z)]).
addDomino([p(A, B),p(B, D)|XS], p(Y, Z)) :- addDomino([p(B, D)|XS], p(Y, Z)]).
```

Exercise 4 (Meta-Programming): (11 points)

Consider a list of unary functions $f_1, \ldots, f_n$. When starting with an initial value x, we first compute $f_1(x)$ and then apply the other functions $f_2, f_3, \ldots$ (in that order) to the previous result, respectively. In other words, we compute $f_n(\ldots(f_2(f_1(x)))\ldots)$.
Please implement the predicate `apply/3` that can be used for this task. The first argument of `apply` is the initial value of the computation. The second argument contains a list of function symbols `f_i/0`. Here, we assume that for each `f_i/0` there exists a predicate symbol `f_i/2`. The third argument is the result of applying the functions as described above, assuming that the first argument of each predicate `f_i/2` is the input value for the function f_i and the second argument is the result of f_i.
For example, the query `?- apply(0, [fA, fB], X).` is evaluated by executing the calls `fA(0, R1)` and `fB(R1, R2)` and unifying `X=R2`. So if `fA(0, 1)` and `fB(1, 4)` are true, then `apply(0, [fA, fB], 4)` is true.
Hints:

- You may use the built-in predicate `=../2`.
- In the case of an empty list of functions $f_1, \ldots, f_n$ the result of `apply` is the initial value, i.e. $f_n(\ldots(f_2(f_1(x)))\ldots) = x$ if $n = 0$.

Solution:

```
apply(X, []    , X).
apply(X, [F|FS], Z) :- G =.. [F, X, R],
                       G,
                       apply(R, FS, Z).
```

Exercise 5 (Difference Lists): (9 points)

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

```
q(X) :- p(X - []).

p([A|B] - B).
p([A|B] - Z) :- p(B - Z).
```

Explicitly give the set of all ground terms t for which the query `?- q(`t`).` succeeds. You do not have to provide a proof for your answer.

Solution:

$\{[t_1, \ldots, t_n] \mid n \geq 1, t_i \text{ is a ground term for all } i \in \{1, \ldots, n\}\}$

Exercise 6 (Arithmetic): (5 points)

Consider the following sequence of numbers:

$$5 \xrightarrow{-1} 4 \xrightarrow{+2} 6 \xrightarrow{-3} 3 \xrightarrow{+4} 7 \xrightarrow{-5} 2 \xrightarrow{+6} 8 \xrightarrow{-7} 1 \xrightarrow{+8} 9 \xrightarrow{-9} \ldots$$

This sequence results from alternating the subtraction and addition of a number which is incremented in each step. When starting with 5 and first subtracting 1, the result of this is the sequence shown above. Implement the predicate `nth/2` in Prolog. For a number $n > 0$ a call of `nth(`n`, X)` gives the answer substitution `X =` m where m is the nth number in the sequence described above. Here, we use 1 to describe the first element, so `nth(1, X)` gives the answer substitution `X = 5`.
As another example, `nth(5, X)` gives the answer substitution `X = 7`.

Solution:

```
nth(X, Y) :- hM(5, 1, X, Y).
hM(A, _, 1, A).
hM(A, B, X, Y) :- X > 1, T1 is A - B, T2 is B + 1, T3 is X - 1, hP(T1, T2, T3, Y).
hP(A, _, 1, A).
hP(A, B, X, Y) :- X > 1, T1 is A + B, T2 is B + 1, T3 is X - 1, hM(T1, T2, T3, Y).
```

Alternative solution:

```
nth(X, Y) :- 1 is X mod 2, Y is 5 + (X // 2).
nth(X, Y) :- 0 is X mod 2, Y is 5 - (X // 2).
```