Chair of Logic and Verification
School of Computation, Information and Technology
Technical University of Munich

Esolution
Place student sticker here

Note:

- During the attendance check a sticker containing a unique code will be put on this exam.
- This code contains a unique number that associates this exam with your registration number.
- This number is printed both next to the code and to the signature field in the attendance check list.

Lambda Calculus

Exam:	IN2358 / Retake	**Date:**	Friday 5th April, 2024
Examiner:	Prof. Tobias Nipkow	**Time:**	17:00 – 18:30

	P 1	P 2	P 3	P 4	P 5
I					

Working instructions

- This exam consists of **12 pages** with a total of **5 problems**.
  Please make sure now that you received a complete copy of the exam.
- The total amount of achievable credits in this exam is 30 credits.
- Detaching pages from the exam is prohibited.
- Allowed resources:
  - one **DIN A4 sheet** with **hand-written** notes on both sides
  - one **analog dictionary** English ↔ native language
- Subproblems marked by * can be solved without results of previous subproblems.
- **Answers are only accepted if the solution approach is documented.** Give a reason for each answer unless explicitly stated otherwise in the respective subproblem.
- Do not write with red or green colors nor use pencils.
- Physically turn off all electronic devices, put them into your bag and close the bag.

Left room from ____________ to ____________ / Early submission at ____________

Problem 1 Programming with λ-terms (12 credits)

Recall the fold encoding from the exercise sheets where a list $[x, y, z]$ is represented as $\lambda c\ n.\ c\ x\ (c\ y\ (c\ z\ n))$. Accordingly, the empty list is defined as $\mathsf{nil} := \lambda c\ n.\ n$ and we can prepend an element to a list with $\mathsf{cons}\ x\ l := \lambda c\ n.\ c\ x\ (l\ c\ n)$.

0 1 2 3 4

a)* Define a λ-term snoc that appends one element, i.e. it should hold that

- $\mathsf{snoc}\ x\ \mathsf{nil} =_\beta \mathsf{cons}\ x\ \mathsf{nil}$
- $\mathsf{snoc}\ x\ (\mathsf{cons}\ h\ t) =_\beta \mathsf{cons}\ h\ (\mathsf{snoc}\ x\ t)$

$\mathsf{snoc} := \lambda x\ l.\ \lambda c\ n.\ l\ c\ (c\ x\ n)$

0 1 2 3 4

b)* Without using a fixed point combinator, define a λ-term reverse that reverses a list, i.e. it should hold that

- $\mathsf{reverse}\ \mathsf{nil} =_\beta \mathsf{nil}$
- $\mathsf{reverse}\ (\mathsf{cons}\ h\ t) =_\beta \mathsf{snoc}\ h\ (\mathsf{reverse}\ t)$

$\mathsf{reverse} := \lambda l.\ l\ \mathsf{snoc}\ \mathsf{nil}$

0 1 2 3 4

c)* Implement reverse again using the fixed point combinator fix. Your implementation must not rely on the underlying encoding of lists but you may use the functions nil, cons, null, head, tail with the usual semantics. Additionally, you may use snoc. Remember that null returns a boolean, i.e. $\mathsf{null}\ \mathsf{nil}\ x\ y \to_\beta^* x$ and $\mathsf{null}\ (\mathsf{cons}\ h\ t)\ x\ y \to_\beta^* y$.

$\mathsf{reverse} := \mathsf{fix}\ (\lambda r.\ \lambda l.\ (\mathsf{null}\ l)\ \mathsf{nil}\ (\mathsf{snoc}\ (\mathsf{head}\ l)\ (r\ (\mathsf{tail}\ l))))$

Problem 2 Rewriting (8 credits)

Let $\to \subseteq A \times A$ be a relation. Below, all variables are implicitly assumed to be elements of A.
The set S of **direct successors** of x is defined as $S(x) = \{y \mid x \to y\}$.
We call $\to$

finitely branching if every x has only finitely many direct successors,
i.e. $S(x)$ is finite.

$\mathbb{N}$-terminating if there is a function $t : A \to \mathbb{N}$ such that
$x \to y \Longrightarrow t(x) > t(y)$.

An element x is called **bounded**, or more explicitly k-**bounded**, if the length of all reductions starting from x is bounded: there is a $k \in \mathbb{N}$ such that for every y it holds that $x \to^n y \Longrightarrow n \leq k$.

Assume that $\to$ is finitely branching and $\mathbb{N}$-terminating. Prove that all x are bounded.

Hint: Proof by induction.

The proof must be given in the standard verbal style.

We assume that $\to$ is finitely branching and $\mathbb{N}$-terminating (with function t) and prove that every x is bounded.

The proof is by (strong) induction on $t(x)$. We may assume that all y with $t(y) < t(x)$ are bounded. Let $S(x) = \{y_1, ..., y_n\}$ (because $\to$ is finitely branching). Because $x \to y_i$ we have $t(y_i) < t(x)$. Thus there are $k_1, \ldots, k_n$ such that y_i is k_i-bounded. Let $k = \mathsf{max}\{0, k_1, \ldots, k_n\} + 1$. Then x is k bounded: If $x \to^n z$ then either $n = 0$ (in which case $n \leq k$) or $x \to y_i \to^{n-1} z$. By IH $n - 1 \leq k_i$ and therefore $n \leq k$.

Problem 3 **Quiz (3 credits)**

0 ☐
1 ☐

a)* We define the set of WNF inductively with

$$\frac{t_1, \ldots, t_n \in \mathsf{WNF}}{x\ t_1\ \ldots\ t_n \in \mathsf{WNF}}$$

$$\frac{}{(\lambda x.\ t) \in \mathsf{WNF}}$$

Give a closed term t such that $t \in \mathsf{WNF}$ but $t \notin \mathsf{NF}$. Justify your choice of t!

Let $t = (\lambda x.\ (\lambda x.\ x)\ (\lambda x.\ x))$. Then, $t \in \mathsf{WNF}$ by the second rule but $t \notin \mathsf{NF}$ since

$$(\lambda x.\ (\lambda x.\ x)\ (\lambda x.\ x)) \to_\beta (\lambda x.\ (\lambda x.\ x)).$$

0 ☐
1 ☐

b)* Give a type τ such that only finitely many simply typed λ-terms t have that type, i.e. it holds that $\vdash t : \tau$.

Let $\tau = A \to B$. Then, no term has type τ.

0 ☐
1 ☐

c)* Let f be a λ-term with type $\mathsf{bool} \to \mathsf{bool}$ in System F, i.e. it holds that $\vdash f : \mathsf{bool} \to \mathsf{bool}$. Is it decidable whether $f\ x \to_\beta^* \mathsf{true}$ for every x in $\beta\eta$-normal form with type bool?
Remember that there are exactly two terms with type bool that are in $\beta\eta$-normal form, namely true and false.

Yes. Since System F is strongly normalising, we can reduce f true and f false to normal form and check whether both are equal to true.

Problem 4 **Typing with `let`-polymorphism (5 credits)**

Consider simply typed lambda calculus extended with `let` and consider the following the typing problem

$$a : A \vdash \texttt{let } x = \lambda z.\ z\ a \texttt{ in } \lambda y.\ x\ (x\ y) :?\tau$$

where A is a type variable.

a)* Find a most general type schema σ with $a : A \vdash \lambda z.\ z\ a : \sigma$. You may, but you do not need to draw a type derivation tree.

$$\sigma = \forall C.\ (A \to C) \to C$$

b) Draw the type derivation tree for

$$a : A, x : \sigma \vdash \lambda y.\ x\ (x\ y) : ?\tau$$

Of course, with the correct type for $?\tau$.
Use only the introduction and elimination rules for $\to$ and $\forall$ and the standard assumption rule

$$\Gamma \vdash x : \tau \quad \text{where} \quad \Gamma(x) = \tau.$$

Let $\Gamma := a : A, x : \sigma, y : B$. The solution is $?\tau = (A \to A \to D) \to D$.

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash x : \forall C.\ (A \to C) \to C}}{\Gamma \vdash x : (A \to D) \to D}\,\forall E \qquad \dfrac{\dfrac{\dfrac{}{\Gamma \vdash x : \forall C.\ (A \to C) \to C}}{\Gamma \vdash x : (A \to A \to D) \to (A \to D)}\,\forall E \qquad \dfrac{}{\Gamma \vdash y : A \to A \to D}}{\Gamma \vdash x\ y : A \to D}\,\to E}{\dfrac{\Gamma \vdash x\ (x\ y) : D}{a : A, x : \sigma \vdash \lambda y.\ x\ (x\ y) : (A \to A \to D) \to D}\,\to I}\,\to E$$

Problem 5 **Intuitionistic Logic (2 credits)**

In this exercise, we consider intuitionistic logic with just the rules $\wedge$I, $\wedge\mathrm{E}_1$, $\wedge\mathrm{E}_2$, $\neg$I, and $\neg$E, i.e.

$$\frac{\Gamma, A \vdash \bot}{\Gamma \vdash \neg A}\ \neg\mathrm{I}$$

$$\frac{\Gamma \vdash \neg A \qquad \Gamma \vdash A}{\Gamma \vdash \bot}\ \neg\mathrm{E}$$

In the following, explicitely annotate the rule that was used for each inference step.

0 1 a)* Prove that an assumption A can be weakened to $A \wedge B$ in intuitionistic logic, i.e. show that $\Gamma, A \vdash C$ implies $\Gamma, A \wedge B \vdash C$. Use induction on the derivation of $\Gamma, A \vdash C$. You only need to consider the cases where $\Gamma, A \vdash C$ was proved by assumption or $\wedge$I.

- Case assumption:
  We have $\Gamma, A \vdash C$ and $C \in \Gamma, A$. If $C \in \Gamma$, then we have $\Gamma, A \wedge B \vdash C$ by assumption. If $C = A$, we have the following proof tree:

$$\frac{\overline{\Gamma, A \wedge B \vdash A \wedge B}}{\Gamma, A \wedge B \vdash A}\ \wedge\mathrm{E}_1$$

- Case $\wedge$I:
  We have $\Gamma, A \vdash C \wedge D$. As induction hypotheses we obtain that $\Gamma, A \wedge B \vdash C$ and $\Gamma, A \wedge B \vdash D$. We justify our goal with the following proof tree:

$$\frac{\dfrac{}{\Gamma, A \wedge B \vdash C}\ \text{I.H.} \qquad \dfrac{}{\Gamma, A \wedge B \vdash D}\ \text{I.H.}}{\Gamma, A \wedge B \vdash C \wedge D}\ \wedge\mathrm{I}$$

b)* In order to represent implication, we view $A \to B$ as an abbreviation for $\neg(A \wedge \neg B)$. Prove the implication introduction rule, i.e. show that $\Gamma, A \vdash B$ implies $\Gamma \vdash A \to B$.

0 1

$$\dfrac{\dfrac{\wedge E_2\,\dfrac{\overline{\Gamma, A \wedge \neg B \vdash A \wedge \neg B}}{\Gamma, A \wedge \neg B \vdash \neg B} \qquad \dfrac{\overline{\Gamma, A \vdash B}\ \text{Assumption}}{\Gamma, A \wedge \neg B \vdash B}\ \text{Part a)}}{\Gamma, A \wedge \neg B \vdash \bot}\ \neg E}{\Gamma \vdash \neg(A \wedge \neg B)}\ \neg I$$

Additional space for solutions–clearly mark the (sub)problem your answers are related to and strike out invalid solutions.