Mid-term examination #2 — given Thursday 18th November

General instructions

Closed-book, closed-notes, closed-computer, in-class exam.

Time allowed: 75 minutes.

Total points available: 150 pts.

Answer in the spaces provided.

Your name (print):

I pledge my honor that in the preparation of this assignment I have complied with the University of New Mexico Board of Regents' Policy Manual.

Please sign and date:

2.1 Concepts in syntax and semantics of programming languages (40 pts)

Briefly explain the following concepts in the syntax and semantics of programming languages. **Illustrate each answer with an explanation of a simple example.**

1. (10 pts) Non-terminal symbols in context-free grammars.

2. (10 pts) Alpha-equivalence of lambda-terms.

3. (20 pts) Type safety.

2.2 Typing judgments (40 pts)

1. (20 pts) Using the typing rules for the typed calculus of numbers and booleans, as defined in Appendix A, identify the type T such that

$$\mathsf{iszero}\ \big(\mathsf{pred}\ (\mathsf{if}\ \mathsf{succ}\ 0\ \mathsf{then}\ 0\ \mathsf{else}\ \mathsf{succ}\ 0)\big)\ :\ T$$

holds, if such a type exists, by **constructing a full typing derivation for this judgment.**

If no such type exists, **construct a partial derivation and explain where and why the typing derivation fails.**

2. (20 pts) Using the typing rules for the typed calculus of numbers and booleans, as defined in Appendix A, identify the type T such that

$$\mathsf{if}\ (\mathsf{if}\ \mathsf{iszero}\ 0\ \mathsf{then}\ \mathsf{false}\ \mathsf{else}\ \mathsf{true})\ \mathsf{then}\ \mathsf{succ}\ 0\ \mathsf{else}\ \mathsf{pred}\ 0\ :\ T$$

   holds, if such a type exists, by **constructing a full typing derivation for this judgment.**

   If no such type exists, **construct a partial derivation and explain where and why the typing derivation fails.**

2.3 Booleans in the untyped lambda-calculus (50 pts)

1. (5 pts) Following the convention introduced in class, define closed untyped lambda-terms tru and fls that represent the boolean constants true and false, respectively.

2. (20 pts) Define an untyped lambda-term imply that represents the operation of logical implication, such that the following all hold:

$$(\mathsf{imply}\ \mathsf{fls})\ \mathsf{fls} \longrightarrow_\beta^* \mathsf{tru} \qquad (\mathsf{imply}\ \mathsf{fls})\ \mathsf{tru} \longrightarrow_\beta^* \mathsf{tru}$$

$$(\mathsf{imply}\ \mathsf{tru})\ \mathsf{fls} \longrightarrow_\beta^* \mathsf{fls} \qquad (\mathsf{imply}\ \mathsf{tru})\ \mathsf{tru} \longrightarrow_\beta^* \mathsf{tru}$$

If you use additional functions in your definition, those must also be defined.

3. (25 pts) Using your answer from the preceding part of this question, **sketch the complete reduction sequence** for the term

$$(\mathsf{imply}\ \mathsf{fls})\ \mathsf{fls}$$

using call-by-value reduction, and identify the final value obtained.

You **do not** need to construct derivations to justify each step. Just provide the sequence of reductions leading to the final value, expanding out any abbreviations for terms along the way as necessary.

2.4 Untyped call-by-value lambda-calculus—evaluation (20 pts)

Using the evaluation rules for the untyped call-by-value lambda calculus, as defined in Appendix B, **construct a full derivation for *every* step of evaluation** of the term:

$$\Big(\lambda x.\, \lambda z.\, x\Big)\ \Big((\lambda f.\, \lambda y. f\ y)\ (\lambda q.\, q)\Big)$$

Identify when you can no longer reduce the term any further, and explain whether this because you have **reduced the term to a value** or because **reduction is stuck**.

A Reference: the typed calculus of numbers and booleans

Syntax

Terms, t	$::=$	true	*constant true*
	$\mid$	false	*constant false*
	$\mid$	$\mathsf{if}\ t\ \mathsf{then}\ t\ \mathsf{else}\ t$	*conditional*
	$\mid$	0	*constant zero*
	$\mid$	$\mathsf{succ}\ t$	*successor*
	$\mid$	$\mathsf{pred}\ t$	*predecessor*
	$\mid$	$\mathsf{iszero}\ t$	*zero test*
Values, v	$::=$	true	*true value*
	$\mid$	false	*false value*
	$\mid$	nv	*numeric value*
Numeric values, nv	$::=$	0	*zero value*
	$\mid$	$\mathsf{succ}\ nv$	*successor value*
Types, T	$::=$	Bool	*type of booleans*
	$\mid$	Nat	*type of natural numbers*

Evaluation Rules

$$\frac{}{\mathsf{if}\ \mathsf{true}\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3 \longrightarrow t_2}\ (\text{E-IfTrue})$$

$$\frac{}{\mathsf{if}\ \mathsf{false}\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3 \longrightarrow t_3}\ (\text{E-IfFalse})$$

$$\frac{t_1 \longrightarrow t_1'}{\mathsf{if}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3 \longrightarrow \mathsf{if}\ t_1'\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3}\ (\text{E-If})$$

$$\frac{t_1 \longrightarrow t_1'}{\mathsf{succ}\ t_1 \longrightarrow \mathsf{succ}\ t_1'}\ (\text{E-Succ})$$

$$\frac{}{\mathsf{pred}\,0 \longrightarrow 0}\ (\text{E-PredZero})$$

$$\frac{}{\mathsf{pred}\,(\mathsf{succ}\,nv_1) \longrightarrow nv_1}\ (\text{E-PredSucc})$$

$$\frac{t_1 \longrightarrow t_1'}{\mathsf{pred}\,t_1 \longrightarrow \mathsf{pred}\,t_1'}\ (\text{E-Pred})$$

$$\frac{}{\mathsf{iszero}\,0 \longrightarrow \mathsf{true}}\ (\text{E-IsZeroZero})$$

$$\frac{}{\mathsf{iszero}\,(\mathsf{succ}\,nv_1) \longrightarrow \mathsf{false}}\ (\text{E-IsZeroSucc})$$

$$\frac{t_1 \longrightarrow t_1'}{\mathsf{iszero}\,t_1 \longrightarrow \mathsf{iszero}\,t_1'}\ (\text{E-IsZero})$$

Typing Rules

$$\frac{}{\mathsf{true} : \mathsf{Bool}}\ (\text{T-True})$$

$$\frac{}{\mathsf{false} : \mathsf{Bool}}\ (\text{T-False})$$

$$\frac{t_1 : \mathsf{Bool} \qquad t_2 : T \qquad t_3 : T}{\mathsf{if}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3 : T}\ (\text{T-If})$$

$$\frac{}{\mathsf{0} : \mathsf{Nat}} \ (\text{T-Zero})$$

$$\frac{t : \mathsf{Nat}}{\mathsf{succ}\, t : \mathsf{Nat}} \ (\text{T-Succ})$$

$$\frac{t : \mathsf{Nat}}{\mathsf{pred}\, t : \mathsf{Nat}} \ (\text{T-Pred})$$

$$\frac{t : \mathsf{Nat}}{\mathsf{iszero}\, t : \mathsf{Bool}} \ (\text{T-IsZero})$$

B Reference: untyped call-by-value lambda-calculus

Syntax

$$
\begin{array}{rcll}
\text{Terms, } t & ::= & x & \textit{variable} \\
& | & \lambda x.\, t & \textit{abstraction} \\
& | & t\ t & \textit{application} \\
\\
\text{Values, } v & ::= & \lambda x.\, t & \textit{abstraction value}
\end{array}
$$

Evaluation Rules

$$\frac{t_1 \longrightarrow t_1'}{t_1\ t_2 \longrightarrow t_1'\ t_2}\ (\text{E-APP1})$$

$$\frac{t_2 \longrightarrow t_2'}{v_1\ t_2 \longrightarrow v_1\ t_2'}\ (\text{E-APP2})$$

$$\frac{}{(\lambda x.\, t_{12})\ v_2 \longrightarrow [x \mapsto v_2]t_{12}}\ (\text{E-APPABS})$$

Capture-avoiding substitution

$$
\begin{array}{rcll}
[x \mapsto t]x & = & t & \\
[x \mapsto t]y & = & y & \text{if } x \neq y \\
[x \mapsto t](t_1\ t_2) & = & ([x \mapsto t]t_1)\ ([x \mapsto t]t_2) & \\
[x \mapsto t](\lambda x.\, t') & = & \lambda x.\, t' & \\
[x \mapsto t](\lambda y.\, t') & = & \lambda y.\, ([x \mapsto t]t') & \text{if } x \neq y
\end{array}
$$

NB: this definition is capture avoiding if we assume that t is a closed term.