Assignment 2
Finite Types

CS 7400: Intensive Principles of Programming Languages
Chris Martens

Due Monday, October 21, 11:59pm
70 pts

This assignment is due on the above date and must be submitted electronically on Gradescope. Please use the LaTeX template on the website to typeset your assignment and make sure to include your full name and NU ID. For the written problems, you may (alternatively) submit handwritten answers that have been scanned and are **easily legible** (if you're not sure, check with a classmate).

You should submit *one* file, `hw02.pdf`, with your written solutions to the questions.

1 Simply-Typed Lambda Calculus

For the problems in this assignment, please use the syntax, judgments, and rules given in Appendix A.

Task 1 (10 points) Fill in the blanks in the following typing judgments so the resulting judgment holds, or indicate there is no way to do so. You do not need to justify your answer or supply a typing derivation, and the types do not need to be "most general" in any sense. Remember that the function type constructor associates to the right, so that $\tau \to \sigma \to \rho = \tau \to (\sigma \to \rho)$.

(i) $\boxed{} \vdash y\,x : A$

(ii) $\boxed{} \vdash x\,x : \boxed{}$

(iii) $\cdot \vdash \boxed{} : (A \to A) \to A$

(iv) $\cdot \vdash \boxed{} : (B \to C) \to A \times B \to A \times C$

(v) $\cdot \vdash \lambda f.\,\lambda g.\,\lambda x.\,(f\,x)\,(g\,x) : (A \to \boxed{}) \to (A \to \boxed{}) \to (A \to \boxed{})$

Task 2 (5 points) Show how the following term evaluates according to the small-step evaluation semantics in Appendix A:

$$(\lambda p.\ \mathsf{split}(p, f.x.\ \lambda z.\ f\ x))\ ((\lambda x.\lambda y.x), ())$$

Write the evaluation as a series of rewrites, one per line, until the final line is a value, counting small-step evaluations and substitutions. That is, each rewrite by a small step of computation should be written on a line preceded by $\mapsto$. You may optionally explicitly write substitutions on their own line as well. Be careful to avoid variable capture in substitutions.

Task 3 (5 points) Rewrite the expression $\mathsf{split}(p, f.x.\ \lambda z.\ f\ x)$ with the lazy-pair destructors (.1, .2) instead of $\mathsf{split}()$. That is, assuming the original expression is well-typed under $p : A \times B$, write an equivalent well-typed expression under $p : A \,\&\, B$.

Task 4 (6 points) It is often stated that lazy pairs are not necessary in an eager language, because we can already define $\tau_1 \mathbin{\&} \tau_2$ and the corresponding constructors and destructors. Fill in this table.

$\tau_1 \mathbin{\&} \tau_2$	$\triangleq$	$(\mathbf{1} \to \tau_1) \times (\mathbf{1} \to \tau_2)$
$\langle e_1, e_2 \rangle$	$\triangleq$	
$e.1$	$\triangleq$	
$e.2$	$\triangleq$	

Task 5 (4 points) How would you formalize the idea that your translation in the preceding task is correct? Give a prose description of your approach, then give one or more formal theorem statements. You do not need to prove the theorems.

2 Type Isomorphisms

Task 6 (20 points) Prove the following type isomorphisms or explain why they are *not* isomorphic.

1. $A + \mathbf{1} \simeq A$
2. $A + \mathbf{0} \simeq A$

3 Programming Languages in the Wild

Task 7 (20 points) Find a programming language with an implementation (perhaps one you already use and are familiar with) with a type system that includes types similar to at least two of the type connectives from STLC ($\mathbf{1}$, $\mathbf{0}$, $+$, $\times$, $\&$, $\to$). Explain which constructors in your chosen language map most closely to which STLC connectives, and provide a small example program (or several) whose type includes those connectives. If applicable, include a machine-checked type annotation, or show the output of the language's compiler or interpreter's inferred type for each program. Finally, explain where the types in your chosen language *differ* from the idealized presentation in STLC.

A Rule Sheet: STLC/$\lambda^{\times+\mathbf{1}}$

A.1 Statics

$$\frac{}{() : \mathbf{1}}\ \textsf{ty/unit} \qquad \frac{}{x{:}\tau \vdash x : \tau}\ \textsf{ty}/x$$

$$\frac{e : \tau_1}{\textsf{in}_1\, e : \tau_1 + \tau_2}\ \textsf{ty/in}_1 \qquad \frac{e : \tau_1}{\textsf{in}_2\, e : \tau_1 + \tau_2}\ \textsf{ty/in}_2$$

$$\frac{e : \tau_1 + \tau_2 \quad x{:}\tau_1 \vdash e_1 : \tau \quad x{:}\tau_2 \vdash e_2 : \tau}{\textsf{case}(e, x.\ e_1, x.\ e_2) : \tau}\ \textsf{ty/case}$$

$$\frac{e_1 : \tau_1 \quad e_2 : \tau_2}{(e_1, e_2) : \tau_1 \times \tau_2}\ \textsf{ty/pair} \qquad \frac{e : \tau_1 \times \tau_2 \quad x{:}\tau_1, y{:}\tau_2 \vdash e' : \tau}{\textsf{split}(e, x.y.\ e') : \tau}\ \textsf{ty/split}$$

$$\frac{x : \tau_1 \vdash e : \tau_2}{\lambda x.\ e : \tau_1 \to \tau_2}\ \textsf{ty/lam} \qquad \frac{f : \tau_1 \to \tau_2 \quad e : \tau_1}{f\, e : \tau_2}\ \textsf{ty/app}$$

Lazy pairs (&):

$$\frac{e_1 : \tau_1 \quad e_2 : \tau_2}{\langle e_1, e_2 \rangle : \tau_1 \,\&\, \tau_2}\ \textsf{ty/lpair} \qquad \frac{e : \tau_1 \,\&\, \tau_2}{e.1 : \tau_1}\ \textsf{ty/proj}_1 \qquad \frac{e : \tau_1 \,\&\, \tau_2}{e.2 : \tau_2}\ \textsf{ty/proj}_2$$

Empty type ($\mathbf{0}$):

$$\frac{\Gamma \vdash e : \mathbf{0}}{\Gamma \vdash \textsf{case}(e) : \tau}\ \textsf{ty/casez}$$

A.2 Values

$$\frac{}{()\ \textsf{value}}\ \textsf{val/unit} \qquad \frac{e\ \textsf{value}}{\textsf{in}_1\, e\ \textsf{value}}\ \textsf{val/in}_1 \qquad \frac{e\ \textsf{value}}{\textsf{in}_2\, e\ \textsf{value}}\ \textsf{val/in}_2$$

$$\frac{e_1\ \textsf{value} \quad e_2\ \textsf{value}}{(e_1, e_2)\ \textsf{value}}\ \textsf{val/pair} \qquad \frac{}{\lambda x.\ e\ \textsf{value}}\ \textsf{val/lam}$$

$$\frac{}{\langle e_1, e_2 \rangle\ \textsf{value}}\ \textsf{val/lpair}$$

A.3 Small-Step Operational Semantics

Computation rules:

$$\frac{e'\ \textsf{value}}{(\lambda x.\ e)\ (e') \mapsto [e'/x']e}\ \textsf{step/app/lam}$$

$$\frac{e_1\ \textsf{value} \quad e_2\ \textsf{value}}{\textsf{split}((e_1, e_2), x.y.\ e) \mapsto [e_1/x][e_2/y]e}\ \textsf{step/split/pair}$$

$$\frac{e\ \mathsf{value}}{\mathsf{case}(\mathsf{in}_1\, e, x.e_1, y.e_2) \mapsto [e/x]e_1}\ \mathsf{step/case/in_1}$$

$$\frac{e\ \mathsf{value}}{\mathsf{case}(\mathsf{in}_2\, e, x.e_1, y.e_2) \mapsto [e/y]e_2}\ \mathsf{step/case/in_2}$$

$$\frac{}{\langle e_1, e_2\rangle.1 \mapsto e_1}\ \mathsf{step/proj_1/pair} \qquad \frac{}{\langle e_1, e_2\rangle.2 \mapsto e_2}\ \mathsf{step/proj_2/pair}$$

Congruence rules:

$$\frac{f \mapsto f'}{f\ e \mapsto f'\ e}\ \mathsf{step/lam/fn} \qquad \frac{f\ \mathsf{value} \quad e \mapsto e'}{f\ e \mapsto f\ e'}\ \mathsf{step/lam/arg}$$

$$\frac{e_1 \mapsto e_1'}{(e_1, e_2) \mapsto (e_1', e_2)}\ \mathsf{step/pair/1} \qquad \frac{e_1\ \mathsf{value} \quad e_2 \mapsto e_2'}{(e_1, e_2) \mapsto (e_1, e_2')}\ \mathsf{step/pair/2}$$

$$\frac{e \mapsto e'}{\mathsf{split}(e, x.y.\ e_2) \mapsto \mathsf{split}(e', x.y.e_2)}\ \mathsf{step/split/1}$$

$$\frac{e \mapsto e'}{\mathsf{case}(e, x.\ e_1, y.\ e_2) \mapsto \mathsf{case}(e', x.\ e_1, y.\ e_2)}\ \mathsf{step/case/1}$$

$$\frac{e \mapsto e'}{e.1 \mapsto e'.1}\ \mathsf{step/proj_1/1} \qquad \frac{e \mapsto e'}{e.2 \mapsto e'.2}\ \mathsf{step/proj_2/1}$$

$$\frac{e \mapsto e'}{\mathsf{case}(e) \mapsto \mathsf{case}(e')}\ \mathsf{step/casez}$$