Midterm Exam

15-814 Types and Programming Languages
Frank Pfenning

October 18, 2018

Name: Andrew ID:

Instructions

- This exam is closed-book, closed-notes.
- You have 80 minutes to complete the exam.
- There are 4 problems.
- For reference, on pages 10–12 there is an appendix with sections on the syntax, statics, and dynamics.

	Combinators	Type Isomorphisms	Suspensions	Subtyping	
	Prob 1	Prob 2	Prob 3	Prob 4	Total
Score					
Max	30	30	55	35	150

1 Combinators (30 pts)

Instead of explicit λ-expressions we can base computation entirely on so-called *combinators* and *combinatory reduction*. We explore here the *simply-typed* version of combinators. The language of combinators is given by

$$\begin{array}{lrcl} \text{Types} & \tau & ::= & \alpha \mid \tau_1 \to \tau_2 \\ \text{Terms} & t & ::= & x \mid c \mid t_1\, t_2 \end{array}$$

where x are variables, c are constant combinators, and $t_1\, t_2$ is application. Each constant combinator c has a type and a reduction rule.

> *It is important to remember that application is left associative, so for example $S\, x\, y\, z$ stands for $(((S\, x)\, y)\, z)$.*

As an example, consider the combinator I. In the λ-calculus it is defined by $I = \lambda x.\, x$. In the combinatory calculus we instead define it by its reduction behavior.

$$\begin{array}{llcl} \text{Type} & I & : & \alpha \to \alpha \\ \text{Reduction} & I\, x & \mapsto & x \end{array}$$

The type of combinators are schematic in their type variables. This means, for example, that I is also typed by all instances $\tau \to \tau$ of $\alpha \to \alpha$.

Task 1 (15 pts). Fill in the following table. When there are multiple possible types, for full credit provide the most general one using type variables α, β, and γ.

Type	K	:	$\alpha \to (\beta \to \alpha)$
Reduction	$K\, x\, y$	$\mapsto$	
Type	K^*	:	
Reduction	$K^*\, x\, y$	$\mapsto$	y
Type	S	:	
Reduction	$S\, x\, y\, z$	$\mapsto$	$(x\, z)\, (y\, z)$

A remarkable property of the combinatory calculus is that all functions of the λ-calculus can be expressed with just two combinators, S and K in an operationally correct way. This is true for the untyped as well as the simply-typed version. In Tasks 2 and 3 we are not concerned with typing.

Task 2 (5 pts). Show that I can be defined as $S\, K\, K$ by showing each step in a computation $S\, K\, K\, x \mapsto^* x$.

Task 3 (5 pts). Show how to define K^* as a combinatory term using K, I, and S as needed (and **not** as a λ-expression).

Definition $K^* \;=\; \boxed{}$

Task 4 (5 pts). Verify that your definition in Task 3 is correct by showing a step-by-step reduction of $t\,x\,y \mapsto^* y$ assuming you defined $K^* = t$.

2 Type Isomorphism (30 pts)

Recall that two types τ and σ are *isomorphic* if we can supply a pair of functions $f : \tau \to \sigma$ and $g : \sigma \to \tau$ such that $f \circ g$ and $g \circ f$ are both equal to the identity function. We take here an extensional point of view, that is, two functions are equal if applied to an arbitrary value v of the correct type yield equal results, and two lazy pairs are equal if their first and second projections are equal.

In each of the following cases, provide the bodies of functions going in both directions and state whether they form an isomorphism for arbitrary types τ, σ, and ρ. You do not need to prove isomorphisms nor provide counterexamples. For ease of writing, we have already decomposed the outermost λ-abstraction in the two functions in each case.

Task 1 (10 pts). $\tau \mathbin{\&} \sigma \cong \sigma \mathbin{\&} \tau$?

$$p : \tau \mathbin{\&} \sigma \vdash \boxed{} : \sigma \mathbin{\&} \tau$$
$$q : \sigma \mathbin{\&} \tau \vdash \boxed{} : \tau \mathbin{\&} \sigma$$

Do these form an isomorphism? YES / NO

Task 2 (10 pts). $\tau \mathbin{\&} \tau \cong \tau$?

$$p : \tau \mathbin{\&} \tau \vdash \boxed{} : \tau$$
$$x : \tau \vdash \boxed{} : \tau \mathbin{\&} \tau$$

Do these form an isomorphism? YES / NO

Task 3 (10 pts). $\tau \to (\sigma \mathbin{\&} \rho) \cong (\tau \to \sigma) \mathbin{\&} (\tau \to \rho)$?

$$f : \tau \to (\sigma \mathbin{\&} \rho) \vdash \boxed{} : (\tau \to \sigma) \mathbin{\&} (\tau \to \rho)$$
$$p : (\tau \to \sigma) \mathbin{\&} (\tau \to \rho) \vdash \boxed{} : \tau \to (\sigma \mathbin{\&} \rho)$$

These form an isomorphism? YES / NO

3 Suspensions (55 pts)

In this problem with start from our call-by-value functional language with simple types and add a new type constructor $\uparrow\tau$ representing the suspension of an expression of type τ. We add a *constructor* $\mathbf{freeze}(e)$ and a *destructor* $\mathbf{thaw}(e)$ for this new type. $\mathbf{freeze}(e)$ is intended to suspend evaluation of e until it is liberated with $\mathbf{thaw}$.

Task 1 (10 pts). Give the typing rules for $\mathbf{freeze}(e)$ and $\mathbf{thaw}(e)$.

Task 2 (15 pts). Provide the rules for the judgments $e \mapsto e'$ and e *val* pertaining to the new constructs.

Task 3 (10 pts). Fill in the gaps and one case in the proof of preservation.

Theorem (Preservation)
If $\cdot \vdash e : \tau$ and $e \mapsto e'$ then $\cdot \vdash e' : \tau$.

Proof: By

Case: When the destructor meets the constructor we have

□

Task 4 (5 pts). State the case in the canonical forms theorem for the type $\uparrow\tau$. You do not need to prove it.

Theorem (Canonical Forms)
If $\cdot \vdash v : \uparrow\tau$ and v *val* then

Task 5 (10 pts). Fill in the gaps and one case in the proof of progress.

Theorem (Progress)
If $\cdot \vdash e : \tau$ then either $e \mapsto e'$ for some e' or e *val*.

Proof: By [ ]

Case: For the destructor we have

□

Task 6 (5 pts). Show how the type $\uparrow\tau$ may be encoded in the language already by giving an appropriate translation for the type, constructors, and destructors. You do not need to prove its correctness.

$$\begin{array}{lcl} \uparrow\tau & = & \boxed{} \\ \mathbf{freeze}(e) & = & \boxed{} \\ \mathbf{thaw}(e) & = & \boxed{} \end{array}$$

4 Subtyping (35 pts)

In this problem we explore *subtyping*. We define $\tau \leq \sigma$ to mean that every value of type τ also has type σ. Using n-ary labeled sum (including the case for $n = 1$), we define

$$\begin{array}{lcl} \mathit{nat} & = & \rho\alpha.\,(\mathsf{z} : 1) + (\mathsf{s} : \alpha) \\ \mathit{pos} & = & \rho\beta.\,(\mathsf{s} : \mathit{nat}) \\ \mathit{zro} & = & \rho\gamma.\,(\mathsf{z} : 1) \end{array}$$

Task 1 (5 pts). Complete the following typing derivation of $\mathbf{fold}(\mathsf{s} \cdot \mathbf{fold}(\mathsf{z} \cdot \langle\rangle)) : \mathit{pos}$.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \langle\rangle : \boxed{}}}{\vdash \mathsf{z} \cdot \langle\rangle : \boxed{}}}{\vdash \mathbf{fold}(\mathsf{z} \cdot \langle\rangle) : \boxed{}}}{\vdash \mathsf{s} \cdot \mathbf{fold}(\mathsf{z} \cdot \langle\rangle) : \boxed{}}$$

$$\vdash \mathbf{fold}(\mathsf{s} \cdot \mathbf{fold}(\mathsf{z} \cdot \langle\rangle)) : \mathit{pos}$$

Task 2 (10 pts). Prove that $\mathit{pos} \leq \mathit{nat}$, that is, for every $v :: \mathit{pos}$ we also have $v :: \mathit{nat}$. The rules for closed value typing are provided in Appendix C for reference.

Similarly (you do not have to show this), we also have $\mathit{zro} \leq \mathit{nat}$.

Task 3 (10 pts). Fill in the following table of functions with their types. When there are multiple different types, provide the most informative. For example, we have $zero : nat$, but $zero : zro$ is more informative since $zro \leq nat$.

$zero$	$:$	zro
$zero$	$=$	
$succ$	$:$	
$succ$	$=$	
$pred$	$:$	$pos \to nat$
$pred$	$=$	

Task 4 (10 pts). Give a definition *even* of the even numbers in unary form such that $even \leq nat$. You do not need to prove this proposition.

$even =$

Appendix: Some Inference Rules

A Syntax

Types τ and terms e are given by the following grammars, where I ranges over finite index sets. We present disjoint sums in their n-ary form and lazy pairs in their binary form, because it is these forms we use in this exam.

$$
\begin{array}{rcll}
\tau & ::= & \alpha \mid \tau_1 \to \tau_2 \mid \tau_1 \otimes \tau_2 \mid 1 \mid \sum_{i \in I}(i : \tau_i) \mid \tau_1 \,\&\, \tau_2 \mid \rho(\alpha.\tau) & \\
e & ::= & x & \text{(variables)} \\
 & \mid & \lambda x.\, e \mid e_1\, e_2 & (\to) \\
 & \mid & i \cdot e \mid \textbf{case}\ e\ \{i \cdot x_i \Rightarrow e_i\}_{i \in I} & (+) \\
 & \mid & \langle e_1, e_2 \rangle \mid \textbf{case}\ e_0\ \{\langle x_1, x_2 \rangle \Rightarrow e'\} & (\otimes) \\
 & \mid & \langle\,\rangle \mid \textbf{case}\ e_0\ \{\langle\,\rangle \Rightarrow e'\} & (1) \\
 & \mid & \langle\!|e_1, e_2|\!\rangle \mid e \cdot l \mid e \cdot r & (\&) \\
 & \mid & \textbf{fold}(e) \mid \textbf{unfold}(e) & (\rho) \\
 & \mid & \textbf{fix}(x.e) & \text{(recursion)}
\end{array}
$$

B Statics, Expressions: $\Gamma \vdash e : \tau$

$$\frac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau}\ (\textsc{Var}) \qquad \frac{\Gamma, x : \tau \vdash e : \tau'}{\Gamma \vdash \lambda x.e : \tau \to \tau'}\ (\text{I-}\to)$$

$$\frac{\Gamma \vdash e_1 : \tau \to \tau' \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1\, e_2 : \tau'}\ (\text{E-}\to)$$

$$\frac{\Gamma \vdash e : \tau_j \quad (j \in I)}{\Gamma \vdash j \cdot e : \sum_{i \in I}(i : \tau_i)}\ (\text{I-}+)$$

$$\frac{\Gamma \vdash e : \sum_{i \in I}(i : \tau_i) \quad \Gamma, x_i : \tau_i \vdash e_i : \tau \quad (\forall i \in I)}{\Gamma \vdash \textbf{case}\ e\ \{i \cdot x_i \Rightarrow e_i\}_{i \in I} : \tau}\ (\text{E-}+)$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash \langle e_1, e_2 \rangle : \tau_1 \otimes \tau_2}\ (\text{I-}\otimes) \qquad \frac{\Gamma \vdash e_0 : \tau_1 \otimes \tau_2 \quad \Gamma, x_1 : \tau_1, x_2 : \tau_2 \vdash e' : \tau}{\Gamma \vdash \textbf{case}\ e_0\ \{\langle x_1, x_2 \rangle \Rightarrow e'\} : \tau}\ (\text{E-}\otimes)$$

$$\frac{}{\Gamma \vdash \langle\,\rangle : 1}\ (\text{I-}1) \qquad \frac{\Gamma \vdash e_0 : 1 \quad \Gamma \vdash e' : \tau}{\Gamma \vdash \textbf{case}\ e_0\ \{\langle\,\rangle \Rightarrow e'\} : \tau}\ (\text{E-}1)$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash \langle\!|e_1, e_2|\!\rangle : \tau_1\ \&\ \tau_2}\ (\text{I-}\&) \qquad \frac{\Gamma \vdash e : \tau_1\ \&\ \tau_2}{\Gamma \vdash e \cdot l : \tau_1}\ (\text{E-}\&_l) \frac{\Gamma \vdash e : \tau_1\ \&\ \tau_2}{\Gamma \vdash e \cdot r : \tau_2}\ (\text{E-}\&_r)$$

$$\frac{\Gamma \vdash e : [\rho(\alpha.\tau)/\alpha]\tau}{\Gamma \vdash \textbf{fold}(e) : \rho(\alpha.\tau)}\ (\text{I-}\rho) \qquad \frac{\Gamma \vdash e : \rho(\alpha.\tau)}{\Gamma \vdash \textbf{unfold}(e) : [\rho(\alpha.\tau)/\alpha]\tau}\ (\text{E-}\rho)$$

$$\frac{\Gamma, x : \tau \vdash e : \tau}{\Gamma \vdash \textbf{fix}(x.e) : \tau}\ (\textsc{Fix})$$

C Statics, Closed Values: $v :: \tau$

$$\frac{x : \tau \vdash e : \tau'}{\lambda x.e :: \tau \to \tau'}\ (\text{IV-}\to) \qquad \frac{v :: \tau_j \quad (j \in I)}{j \cdot v :: \sum_{i \in I}(i : \tau_i)}\ (\text{IV-}+)$$

$$\frac{v_1 :: \tau_1 \quad v_2 :: \tau_2}{\langle v_1, v_2 \rangle :: \tau_1 \otimes \tau_2}\ (\text{IV-}\otimes) \qquad \frac{}{\langle\,\rangle :: 1}\ (\text{IV-}1)$$

$$\frac{\cdot \vdash e_1 : \tau_1 \quad \cdot \vdash e_2 : \tau_2}{\langle\!|e_1, e_2|\!\rangle :: \tau_1\ \&\ \tau_2}\ (\text{IV-}\&) \qquad \frac{v :: [\rho(\alpha.\tau)/\alpha]\tau}{\textbf{fold}(v) :: \rho(\alpha.\tau)}\ (\text{IV-}\rho)$$

D Dynamics: $e \mapsto e'$ and v *val*

$$\frac{}{\lambda x.e\ \mathit{val}}\ (\text{V-}\to) \qquad \frac{v_2\ \mathit{val}}{(\lambda x.\, e_1)\, v_2 \mapsto [v_2/x]e_1}\ (\text{R-}\to)$$

$$\frac{e_1 \mapsto e_1'}{e_1\, e_2 \mapsto e_1'\, e_2}\ (\text{CE-}\to_1) \qquad \frac{v_1\ \mathit{val} \quad e_2 \mapsto e_2'}{v_1\, e_2 \mapsto e_1\, e_2'}\ (\text{CE-}\to_2)$$

$$\frac{v\ \mathit{val}}{i \cdot v\ \mathit{val}}\ (\text{V-}+) \quad \frac{e \mapsto e'}{i \cdot e \mapsto i \cdot e'}\ (\text{CI-}+) \quad \frac{e \mapsto e'}{\textbf{case}\ e\ \{i \cdot x_i \Rightarrow e_i\}_{i \in I} \mapsto \textbf{case}\ e'\ \{i \cdot x_i \Rightarrow e_i\}_{i \in I}}\ (\text{CE-}+)$$

$$\frac{v_j\ \mathit{val}}{\textbf{case}\ (j \cdot v_j)\ \{i \cdot x_i \Rightarrow e_i\}_{i \in I} \mapsto [v_j/x_j]e_j}\ (\text{R-}+)$$

$$\frac{v_1\ \mathit{val} \quad v_2\ \mathit{val}}{\langle v_1, v_2 \rangle\ \mathit{val}}\ (\text{V-}\otimes) \qquad \frac{e_1 \mapsto e_1'}{\langle e_1, e_2 \rangle \mapsto \langle e_1', e_2 \rangle}\ (\text{CI-}\otimes_1) \qquad \frac{v_1\ \mathit{val} \quad e_2 \mapsto e_2'}{\langle v_1, e_2 \rangle \mapsto \langle v_1, e_2' \rangle}\ (\text{CI-}\otimes_2)$$

$$\frac{e_0 \mapsto e_0'}{\textbf{case}\ e_0\ \{\langle x_1, x_2 \rangle \Rightarrow e'\} \mapsto \textbf{case}\ e_0'\ \{\langle x_1, x_2 \rangle \Rightarrow e'\}}\ (\text{CE-}\otimes)$$

$$\frac{v_1\ \mathit{val} \quad v_2\ \mathit{val}}{\textbf{case}\ \langle v_1, v_2 \rangle\ \{\langle x_1, x_2 \rangle \Rightarrow e'\} \mapsto [v_1/x_1, v_2/x_2]e'}\ (\text{R-}\otimes)$$

$$\frac{}{\langle\,\rangle\ \mathit{val}}\ (\text{V-1}) \qquad \frac{e_0 \mapsto e_0'}{\textbf{case}\ e_0\ \{\langle\,\rangle \Rightarrow e'\} \mapsto \textbf{case}\ e_0'\ \{\langle\,\rangle \Rightarrow e'\}}\ (\text{CE-1}) \qquad \frac{}{\textbf{case}\ \langle\,\rangle\ \{\langle\,\rangle \Rightarrow e'\} \mapsto e'}\ (\text{R-1})$$

$$\frac{}{\langle\!|e_1, e_2|\!\rangle\ \mathit{val}}\ (\text{V-}\&) \qquad \frac{e \mapsto e'}{e \cdot l \mapsto e' \cdot l}\ (\text{CI-}\&_l) \qquad \frac{e \mapsto e'}{e \cdot r \mapsto e' \cdot r}\ (\text{CI-}\&)_r$$

$$\frac{}{\langle\!|e_1, e_2|\!\rangle \cdot l \mapsto e_1}\ (\text{R-}\&_l) \qquad \frac{}{\langle\!|e_1, e_2|\!\rangle \cdot r \mapsto e_2}\ (\text{R-}\&_r)$$

$$\frac{v\ \mathit{val}}{\textbf{fold}(v)\ \mathit{val}}\ (\text{V-}\rho) \qquad \frac{e \mapsto e'}{\textbf{fold}(e) \mapsto \textbf{fold}(e')}\ (\text{CI-}\rho)$$

$$\frac{e \mapsto e'}{\textbf{unfold}(e) \mapsto \textbf{unfold}(e')}\ (\text{CE-}\rho) \qquad \frac{v\ \mathit{val}}{\textbf{unfold}(\textbf{fold}(v)) \mapsto v}\ (\text{R-}\rho)$$

$$\frac{}{\textbf{fix}(x.e) \mapsto [\textbf{fix}(x.e)/x]e}\ (\text{R-FIX})$$