Types and Programming Languages (15-814), Fall 2018
Assignment 2: Products and Sums

Contact: 15-814 Course Staff

Due Tuesday, October 2, 2018

This assignment is due by 23:59 on the above date and it must be submitted electronically as a PDF file on Canvas. Please use the attached template to typeset your assignment and make sure to include your full name and Andrew ID. As before, problems marked "WB" are subject to the whiteboard policy; all other problems must be done individually.

We also make the two following requests. Please record the time spend on the homework assignment. It is difficult for us to judge how long this assignment will take. Please mark the questions that you discussed with classmates using the whiteboard policy. We would like to see how many of you take advantage of this and for which questions.

Lazy products

In class on Tuesday, we learned about eager products. When eliminating eager products, we first reduced the components from left to right until both were values. Only then did we deem the pair to be a value. This is because we wanted to simultaneously use both components in the **case** elimination form, and in a call-by-value semantics, variables should only ever be instantiated with values.

With *lazy products*, a pair of expressions is always a value. Instead of a **case** elimination form that simultaneously extracts both components of an ordered pair, lazy products have left and right projections that extract the single corresponding component from the pair. Before we can project out of a term that is not an ordered pair, we must first reduce that term until it becomes one.

In this part of the assignment, you will explore the dynamics for lazy products and prove various properties about them. We explore lazy products in the context of the simple language we have seen so far. Its syntax, statics, and dynamics are given in the appendices.

We first extend our syntax:

$$
\begin{array}{lr}
\tau ::= \cdots & \\
\quad \mid \tau_1 \mathbin{\&} \tau_2 & \text{lazy product of } \tau_1 \text{ and } \tau_2 \\
e ::= \cdots & \\
\quad \mid \langle\!| e_1, e_2 |\!\rangle & \text{ordered pair of } e_1 \text{ and } e_2 \\
\quad \mid e \cdot l & \text{left projection} \\
\quad \mid e \cdot r & \text{right projection}
\end{array}
$$

The introduction rule for lazy pairs is:

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash \langle\!| e_1, e_2 |\!\rangle : \tau_1 \mathbin{\&} \tau_2} \text{ (I-\&)}$$

Its elimination rules are:

$$\frac{\Gamma \vdash e : \tau_1 \mathbin{\&} \tau_2}{\Gamma \vdash e \cdot l : \tau_1} \text{ (E-\&-L)} \qquad \frac{\Gamma \vdash e : \tau_1 \mathbin{\&} \tau_2}{\Gamma \vdash e \cdot r : \tau_2} \text{ (E-\&-R)}$$

Task 1 (10 points, WB)**.** Give rules capturing the intended dynamics for lazy products. That is, give rules defining the judgments e *val* and $e \mapsto e'$ for terms e of the form $\langle\!| e_1, e_2 |\!\rangle$, $e_0 \cdot l$, and $e_0 \cdot r$. They should satisfy $\langle\!| e_1, e_2 |\!\rangle \cdot l \mapsto^* e_1$ and $\langle\!| e_1, e_2 |\!\rangle \cdot r \mapsto^* e_2$ and capture the intuitions we gave above.

Task 2 (10 points, WB)**.** State and prove the canonical forms theorem for lazy pairs. The statement should be of the following form: "if $\cdot \vdash e : \tau_1 \mathbin{\&} \tau_2$ and e *val*, then e is a term of the form(s) ...".

Task 3 (10 points, WB)**.** Show that the progress property still holds after the addition of lazy pairs. In particular, prove if $\cdot \vdash e : \tau$ by (I-&), (E-&-L), or (E-&-R), then either e *val* or there exists an e' such that $e \mapsto e'$.

Task 4 (5 points, WB)**.** Show that the preservation property still holds after the addition of lazy pairs. In particular, show that if $e \mapsto e'$ by one of the rules you gave in task 1 and $\cdot \vdash e : \tau$, then $\cdot \vdash e' : \tau$.

Task 5 (5 points, WB)**.** Write down terms ξ and ξ^{-1} such that

$$\cdot \vdash \xi : \tau \otimes \sigma \to \tau \mathbin{\&} \sigma$$
$$\cdot \vdash \xi^{-1} : \tau \mathbin{\&} \sigma \to \tau \otimes \sigma$$

and for all values v of type $\tau \otimes \sigma$, $\xi^{-1}(\xi(v)) \mapsto^* v$ (you do not need to prove this). Do these terms witness the isomorphism? If so, state this. If not, briefly explain why.

Hint. Consider the dynamics of eager and lazy products in the presence of non-termination.

A billion dollar mistake

A frequent source of bugs programs is attempting to dereference null pointers. Languages subject to this class of errors feature a type π of "pointers" with a distinguished pointer **null** called the "null pointer". The intention is that the null pointer represent a lack of value, while all other pointers can be "dereferenced" to produce a value. Any attempt to dereference a null pointer is necessarily an error. This class of errors is so rampant that Hoare, the inventor of the null pointer, has called null pointers his "billion dollar mistake" [2].

The crux of the problem is a mis-identification of the type option type π **opt** with the type π. The option type τ **opt** represents optional values of type τ, where **some** e captures the presence of a value e of type τ and **none** represents the lack of value. We can test for the presence of a value using the construction "**case** e_1 $\{$**some** $x \Rightarrow e_2 \mid$ **none** $\Rightarrow e_3\}$". This checks if e_1 is **none**, in which it produces e_3, and if e_1 is **some** v_1, then we produce $[v_1/x]e_2$. This can be encoded using sum types as follows:

$$\begin{aligned}
\tau\ \mathbf{opt} &= \tau + 1 \\
\mathbf{some}\ e &= l \cdot e \\
\mathbf{none} &= r \cdot \langle\rangle \\
\mathbf{case}\ e_1\ \{\mathbf{some}\ x \Rightarrow e_2 \mid \mathbf{none} \Rightarrow e_3\} &= \mathbf{case}\ e_1\ \{l \cdot x \Rightarrow e_2 \mid r \cdot _ \Rightarrow e_3\}
\end{aligned}$$

The key point is that the type system stops us from directly using an optional value e of type τ **opt** in a context expecting a value of type τ: the type system forces us to account for the fact that e could be **none**. In contrast, π treats the null pointer **null** as a genuine pointer, allowing it to be used in any context expecting a genuine pointer.

The affected languages try to work around this problem by providing a function **isnull** $: \pi \to$ **bool** that produces **true** if it is applied to **null**, and otherwise produces **false**. To avoid accidentally dereferencing a pointer, code is then littered with checks of the form:

$$\mathbf{if}\ (\mathbf{isnull}\ e)\ \mathbf{then}\ \ldots\textit{handle error}\ldots\ \mathbf{else}\ \ldots\textit{do something}\ldots.$$

Unfortunately, it is very easy to forget such a check and attempt to dereference the potentially **null** value e. Implicitly, this work-around attempts to simulate the type τ **opt** using **bool** $\otimes\ \tau$, where a value of type τ is tagged with a boolean that signals whether or not the value is actually present. The reader is referred to [1, pp. 92f.] for more details.

Task 6 (10 points, WB, [1, Ex. 11.2])**.** Informally show that we cannot identify $\mathbf{bool} \otimes \tau$ and τ **opt** for arbitrary τ. Do so by attempting to give an implementation of τ **opt** in terms of $\mathbf{bool} \otimes \tau$ by sensibly completing the following chart:

$$\begin{aligned} \mathbf{none} &= ? \\ \mathbf{some}\ e &= ? \\ \mathbf{case}\ e_1\ \{\mathbf{some}\ x \Rightarrow e_2 \mid \mathbf{none} \Rightarrow e_3\} &= ? \end{aligned}$$

Where do you get stuck if you do not additionally assume the existence of a "null" value $\mathbf{null}_\tau$ for each type τ? Argue that even by artificially assuming the existence of such values, you cannot complete the chart in a manner that captures the semantics of τ **opt** in their absence. (Please be brief: your answer must contain at most eight lines of prose.)

For your convenience, you may assume the existence of an $\mathbf{if}\ b\ \mathbf{then}\ e_1\ \mathbf{else}\ e_2$ construct as defined in lecture 6.

Gaining types

Task 7 (5 points, WB)**.** Find terms e and e' and types τ and τ' such that $\cdot \vdash e : \tau$, $e \mapsto e'$, and $\cdot \vdash e' : \tau'$, but not $\cdot \vdash e : \tau'$.

Is zero a zero?

Task 8 (5 points, WB)**.** Is the nullary sum type $\mathbf{0}$ is the zero for $\otimes$? That is, prove or disprove that for all τ,

$$\tau \otimes \mathbf{0} \cong \mathbf{0}.$$

In the affirmative case, we only ask that you write down the two terms witnessing the isomorphism. In the negative case, you must provide a counter-example and explain why it is a counter-example.

Task 9 (0 points)**.** How long did you spend on this assignment? Please list the questions that you discussed with classmates using the whiteboard policy.

References

[1] R. HARPER, *Practical Foundations for Programming Languages*, Cambridge University Press, 2 ed., 2016.

[2] C. A. R. HOARE, *Null references: The billion dollar mistake*. Conference Talk at QCon, August 2009.

A Syntax

Types τ and terms e are given by the grammars:

$$
\begin{aligned}
\tau &::= \alpha \mid \tau_1 \to \tau_2 \mid \mathbf{0} \mid \tau_1 + \tau_2 \mid \mathbf{1} \mid \tau_1 \otimes \tau_2 \mid \tau_1 \,\&\, \tau_2 \\
e &::= x \\
&\quad \mid \lambda x.e \mid e_1 e_2 \\
&\quad \mid \textbf{case}\ e\ \{\,\} \\
&\quad \mid l \cdot e \mid r \cdot e \mid \textbf{case}\ e\ \{l \cdot x \Rightarrow e_x \mid r \cdot y \Rightarrow y_\} \\
&\quad \mid \langle\,\rangle \mid \textbf{case}\ e_1\ \{\langle\,\rangle \Rightarrow e_2\} \\
&\quad \mid \langle e_1, e_2 \rangle \mid \textbf{case}\ e_1\ \{\langle x_1, x_2 \rangle \Rightarrow e_2\} \\
&\quad \mid \langle\!| e_1, e_2 |\!\rangle \mid e \cdot l \mid e \cdot r
\end{aligned}
$$

B Statics

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

$$\frac{\Gamma \vdash e_1 : \tau \to \tau' \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1 e_2 : \tau'}\ (\textsc{App})$$

$$\frac{\Gamma \vdash e : \tau}{\Gamma \vdash l \cdot e : \tau + \tau'}\ (\text{I-+-L}) \qquad \frac{\Gamma \vdash e : \tau'}{\Gamma \vdash r \cdot e : \tau + \tau'}\ (\text{I-+-R})$$

$$\frac{\Gamma \vdash e : \tau_1 + \tau_2 \quad \Gamma, x : \tau_1 \vdash e_1 : \tau \quad \Gamma, x : \tau_2 \vdash e_2 : \tau}{\Gamma \vdash \textbf{case}\ e\ \{l \cdot x \Rightarrow e_1 \mid r \cdot x \Rightarrow e_2\} : \tau}\ (\text{E-+})$$

No (I-$\mathbf{0}$) rule. $\qquad \frac{\Gamma \vdash e : \mathbf{0}}{\Gamma \vdash \textbf{case}\ e\ \{\,\} : \tau}\ (\text{E-}\mathbf{0})$

$$\frac{}{\Gamma \vdash \langle\,\rangle : \mathbf{1}}\ (\text{I-}\mathbf{1}) \qquad \frac{\Gamma \vdash e_0 : \mathbf{1} \quad \Gamma \vdash e_1 : \tau}{\Gamma \vdash \textbf{case}\ e_0\ \{\langle\,\rangle \Rightarrow e_1\} : \tau}\ (\text{E-}\mathbf{1})$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash \langle e_1, e_2 \rangle : \tau_1 \otimes \tau_2}\ (\text{I-}\otimes)$$

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

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash \langle\!| e_1, e_2 |\!\rangle : \tau_1 \,\&\, \tau_2}\ (\text{I-}\&)$$

$$\frac{\Gamma \vdash e : \tau_1 \,\&\, \tau_2}{\Gamma \vdash e \cdot l : \tau_1}\ (\text{E-}\&\text{-L}) \qquad \frac{\Gamma \vdash e : \tau_1 \,\&\, \tau_2}{\Gamma \vdash e \cdot r : \tau_2}\ (\text{E-}\&\text{-R})$$

C Dynamics

$$\frac{}{\lambda x.e\ \mathit{val}}\ (\to\text{-VAL}) \qquad \frac{}{(\lambda x.e_1)e_2 \mapsto [e_2/x]e_1}\ (\text{APP-RED})$$

$$\frac{e_1 \mapsto e_1'}{e_1 e_2 \mapsto e_1' e_2}\ (\text{APP-STEP-L}) \qquad \frac{e_1\ \mathit{val} \quad e_2 \mapsto e_2'}{e_1 e_2 \mapsto e_1 e_2'}\ (\text{APP-STEP-R})$$

$$\frac{e\ \mathit{val}}{l \cdot e\ \mathit{val}}\ (\text{VAL/L}) \qquad \frac{e\ \mathit{val}}{r \cdot e\ \mathit{val}}\ (\text{VAL/R})$$

$$\frac{e \mapsto e'}{l \cdot e \mapsto l \cdot e'}\ (\mapsto\text{/L}) \qquad \frac{e \mapsto e'}{r \cdot e \mapsto r \cdot e'}\ (\mapsto\text{/R})$$

$$\frac{e \mapsto e'}{\textbf{case}\ e\ \{\,\} \mapsto \textbf{case}\ e'\ \{\,\}}\ (\mapsto / \textbf{case}_0)$$

$$\frac{e \mapsto e'}{\textbf{case}\ e\ \{l \cdot x \Rightarrow e_1 \mid r \cdot y \Rightarrow e_2\} \mapsto \textbf{case}\ e'\ \{l \cdot x \Rightarrow e_1 \mid r \cdot y \Rightarrow e_2\}}\ (\mapsto / \textbf{case}_1)$$

$$\frac{v_1\ \mathit{val}}{\textbf{case}\ l \cdot v_1\ \{l \cdot x \Rightarrow e_1 \mid r \cdot y \Rightarrow e_2\} \mapsto [v_1/x]e_1}\ (\mapsto / \textbf{case}/l)$$

$$\frac{v_2\ \mathit{val}}{\textbf{case}\ r \cdot v_2\ \{l \cdot x \Rightarrow e_1 \mid r \cdot y \Rightarrow e_2\} \mapsto [v_2/y]e_2}\ (\mapsto / \textbf{case}/r)$$

$$\frac{}{\langle\,\rangle\ \mathit{val}}\ (\langle\,\rangle\text{-VAL}) \qquad \frac{v_1\ \mathit{val} \quad v_2\ \mathit{val}}{\langle v_1, v_2\rangle\ \mathit{val}}\ (\text{PAIR-VAL})$$

$$\frac{e_1 \mapsto e_1'}{\langle e_1, e_2\rangle \mapsto \langle e_1', e_2\rangle}\ (\text{STEP-L}) \qquad \frac{v_1\ \mathit{val} \quad e_2 \mapsto e_2'}{\langle v_1, e_2\rangle \mapsto \langle v_1, e_2'\rangle}\ (\text{STEP-R})$$

$$\frac{e_0 \mapsto e_0'}{\textbf{case}\ e_0\ \{\langle\,\rangle \Rightarrow e_1\} \mapsto \textbf{case}\ e_0'\ \{\langle\,\rangle \Rightarrow e_1\}}\ (\text{STEP-SUBJ-1})$$

$$\frac{e_0 \mapsto e_0'}{\textbf{case}\ e_0\ \{\langle x_1, x_2\rangle \Rightarrow e_1\} \mapsto \textbf{case}\ e_0'\ \{\langle x_1, x_2\rangle \Rightarrow e_1\}}\ (\text{STEP-SUBJ-2})$$

$$\frac{}{\textbf{case}\ \langle\,\rangle\ \{\langle\,\rangle \Rightarrow e_1\} \mapsto e_1}\ (\text{STEP-CASE-1})$$

$$\frac{\langle v_1, v_2\rangle\ \mathit{val}}{\textbf{case}\ \langle v_1, v_2\rangle\ \{\langle x_1, x_2\rangle \Rightarrow e_1\} \mapsto [v_1, v_2/x_1, x_2]e_1}\ (\text{STEP-CASE-2})$$