EPL Exam Review Session

Simon Fowler
University of Edinburgh

April 26, 2017
Today's Session

→ Two hours (but longer if you like)
→ Plan: Few words to start us off, then questions from you
→ I have slides working through two types of questions:
  → “Is this substitution correct?”
  → “Is this system sound?”
→ ...but I’ve prepared all three exams, so we can go through any of them on the board
Exam Information

→ Your exam:
  → Time: Tuesday, 16th May 2017
  → Location: Patersons Land - 1.26 (Holyrood)
  → (Be sure to check closer to the time – these sometimes change!)

→ Exam format:
  → Two hours
  → Question 1 is compulsory, then you have a choice between questions 2 and 3.

→ Revision Exercises:
  → Three papers:
    → Mock exam (on EPL course page)
    → 2015/16 exam
    → 2015/16 resit exam
  → Tutorial questions
Consider the following substitutions:

\[
\begin{align*}
&\rightarrow (\lambda x.x \ y)[x/y] = \lambda z.z \ x \\
&\rightarrow (\lambda x.\lambda y.(x, y, z))[(y, z)/x] = \lambda x.\lambda y.((y, z), y, z) \\
&\rightarrow (\lambda x.x + ((\lambda y.y) \ z))[y/z] = \lambda x.x + ((\lambda y.y) \ y) \\
&\rightarrow (\lambda x.x + ((\lambda y.y) \ z))[x/z] = \lambda x.x + ((\lambda y.y) \ x)
\end{align*}
\]

For each one, explain whether the substitution has been performed correctly or not. If not, give the correct answer for the right-hand side.

[8 marks]
(\lambda x. x y)[x/y] = \lambda z. z x
\[(\lambda x. x \ y)[x/y] = \lambda z. z \ x\]

This is correct.
→ Substituting \(x\) for \(y\) naïvely would result in \(\lambda x. x \ x\). Here, \(x\) would be captured by the \(\lambda x\) binder, changing the meaning of the program.
→ Instead, it is always safe to perform substitution by choosing fresh variables for the binders, and then performing the substitution:
→ \((\lambda z. z \ y)[x/y] = (\lambda z. z \ x)\)
\[(\lambda x. \lambda y. (x, y, z))[ (y, z)/x ] = \lambda x. \lambda y. ((y, z), y, z)\]
\[(\lambda x. \lambda y. (x, y, z))[(y, z)/x] = \lambda x. \lambda y. ((y, z), y, z)\]

→ This is incorrect.
→ Whereas the \(y\) in \((y, z)\) was free before the substitution, \(y\) has been captured by the \(\lambda y\) afterwards.
→ To correct the substitution, freshen the binders beforehand:

\[(\lambda a. \lambda b. (a, b, z))[(y, z)/x] = \lambda a. \lambda b. (a, b, z)\]
\[(\lambda x. x + ((\lambda y. y) z))[y/z] = \lambda x. x + ((\lambda y. y) y)\]
\((\lambda x. x + ((\lambda y. y) z))[y/z] = \lambda x. x + ((\lambda y. y) y)\)

→ This is correct.

→ \(z\) is not in the scope of the \(\lambda y\) binder, so \(y\) is not captured when it is substituted.
This is incorrect.

$z$ is in the scope of $\lambda x$ before the substitution, so $x$ is captured by the binder.

As ever, this can be solved by freshening the binder before substituting:
“Type soundness is often proved using two properties, called preservation and progress”. Define the preservation property.
“Type soundness is often proved using two properties, called preservation and progress”. Define the preservation property.

→ **Preservation**: Typing is preserved under reduction.
   → More formally, if $\Gamma \vdash e : \tau$ and $e \rightarrow e'$, then $\Gamma \vdash e' : \tau$.

→ **Progress**: A well-typed term is either a value, or can take a reduction step (evaluation doesn’t get “stuck”)
   → More formally, if $\Gamma \vdash e : \tau$, then either $e$ is a value $v$, or there exists some $e'$ such that $e \rightarrow e'$.

→ **Soundness**: A system is **sound** if it satisfies preservation and progress.

These seem to come up a lot – they’re worth knowing!
Consider the following rules which we might add to handle random number generation to a language that already has basic arithmetic:

\[
\begin{align*}
  e & \leadsto e' \\
  \text{randInt}(e) & \leadsto \text{randInt}(e') \\
  0 \leq n < v & \Rightarrow \text{randInt}(v) \leadsto n \\
  v \leq 0 & \Rightarrow \text{randInt}(v) \leadsto 0
\end{align*}
\]

\[
\Gamma \vdash e : \tau
\]

\[
\Gamma \vdash e : \text{int} \\
\Gamma \vdash \text{randInt}(e) : \text{int}
\]

Is this system sound? Briefly explain why or why not.
Does the system satisfy preservation? If something reduces, does it have the same type?

→ Yes: the type is \texttt{int} before and after reduction.

Does the system satisfy progress? Can we always reduce?

→ Yes: if \texttt{randInt} is evaluating a value, then all values accounted for by the last two rules. If evaluating a subexpression, we can assume it takes a step, and thus conclude with the first rule.
How would we prove this formally?

→ Preservation: by induction on $e \mapsto e'$.
→ Progress: by induction on $\Gamma \vdash e : \tau$. 

\[
\begin{align*}
\Gamma \vdash e : \tau & \quad \Gamma \vdash e : \text{int} \\
\Gamma \vdash \text{randInt}(e) : \text{int} & \quad 0 \leq n < \nu \\
& \quad \nu \leq 0
\end{align*}
\]
The system is not sound. Preservation holds: if we take a reduction step, we still end up with a float. Progress does not hold: we cannot reduce \( v_1 \div v_2 \) since no rules match, yet \( v_1 \div v_2 \) is not a value.

Is this system sound?
Is this system sound?

→ No.

→ Preservation holds: if we take a reduction step, we still end up with a float.

→ Progress does not hold: we cannot reduce \( v_1 \div 0 \) since no rules match, yet \( v_1 \div 0 \) is not a value.