CS 320
Computer Language Processing
Exercise Set 4

March 26, 2025

Exercise 1 For each of the following pairs of grammars, show that they are equivalent by identifying them with inductive relations, and proving that the inductive relations contain the same elements.

1. $A_1 : S ::= S + S \mid \mathbf{num}$
   $A_2 : R ::= \mathbf{num}\ R'$ and $R' ::= +R\ R' \mid \epsilon$
2. $B_1 : S ::= S(S)S \mid \epsilon$
   $B_2 : R ::= RR \mid (R)\epsilon$

Exercise 2 Consider the following expression language over naturals, and a *halving* operator:

$$expr ::= \text{half}(expr) \mid expr + expr \mid \mathbf{num}$$

where **num** is any natural number constant ≥ 0.

We will design the operational semantics of this language. The semantics should define rules that apply to as many expressions as possible, while being subjected to the following safety conditions:

- the semantics should *not* permit halving unless the argument is even
- they should evaluate operands from left-to-right

Of the given rules below, choose a *minimal* set that satisfies the conditions above. A set is *not* minimal if removing any rule does not change the set of expressions that can be evaluated by the semantics, i.e. the domain of $\leadsto$, $\{x \mid \exists y.\ x \leadsto y\}$, remains unchanged. The removed rule is said to be *redundant*.

$$\frac{e \leadsto e'}{\text{half}(e) \leadsto e'} \tag{A}$$

$$\frac{n \text{ is a value} \qquad n = 2k}{\text{half}(n) \leadsto k} \tag{B}$$

$$\frac{n \text{ is a value}}{\text{half}(n) \leadsto \lfloor \frac{n}{2} \rfloor} \tag{C}$$

$$\frac{\text{half}(e) \rightsquigarrow \text{half}(e')}{\text{half}(e) \rightsquigarrow e'} \tag{D}$$

$$\frac{e \rightsquigarrow e'}{\text{half}(e) \rightsquigarrow \text{half}(e')} \tag{E}$$

$$\frac{e' \rightsquigarrow \text{half}(e)}{half(e) \rightsquigarrow e'} \tag{F}$$

$$\frac{n_1 \text{ is a value} \qquad n_2 \text{ is a value} \qquad n_1 + n_2 = k \qquad n_1 \text{ is odd}}{n_1 + n_2 \rightsquigarrow k} \tag{G}$$

$$\frac{e \rightsquigarrow e' \qquad n \text{ is a value}}{n + e \rightsquigarrow n + e'} \tag{H}$$

$$\frac{e_2 \rightsquigarrow e_2'}{e_1 + e_2 \rightsquigarrow e_1 + e_2'} \tag{I}$$

$$\frac{n_1 \text{ is a value} \qquad n_2 \text{ is a value} \qquad n_1 + n_2 = k \qquad n_1, n_2 \text{ are even}}{n_1 + n_2 \rightsquigarrow k} \tag{J}$$

$$\frac{n_1 \text{ is a value} \qquad n_2 \text{ is a value} \qquad n_1 + n_2 = k}{n_1 + n_2 \rightsquigarrow k} \tag{K}$$

$$\frac{e_1 \rightsquigarrow e_1'}{e_1 + e_2 \rightsquigarrow e_1' + e_2} \tag{L}$$

Exercise 3 Consider a simple programming language with integer arithmetic, boolean expressions, and user-defined functions:

$$\begin{aligned} expr ::= {} & true \mid false \mid \mathbf{num} \\ & expr == expr \mid expr + expr \\ & expr \mathbin{\&\&} expr \mid if\ (expr)\ expr\ else\ expr \\ & f(expr, \ldots, expr) \mid x \end{aligned}$$

where f represents a (user-defined) function, x represents a variable, and **num** represents an integer.

1. Inductively define a substitution operation for the terms in this language, which replaces every free occurrence of a variable x with a given expression e.

   The rule for substitution in an addition is provided as an example. Here, $t[x := e]$ represents the term t, with every free occurrence of x simultaneously replaced by e.

   $$\frac{t_1[x := e] \to t_1' \qquad t_2[x := e] \to t_2'}{t_1 + t_2[x := e] \to t_1' + t_2'}$$

2. Write the rules for the operational semantics for this language, assuming *call-by-name* semantics for function calls. In call-by-name semantics, function arguments are not evaluated before the call. Instead, the parameters are merely substituted into the function body.

3. Under the following environment (with function names, parameters, and bodies):

$$(sum, [x], if\ (x == 0)\ then\ 0\ else\ x + sum(x + (-1)))$$
$$(rec, [\,], rec())$$
$$(default, [b, x], if\ b\ then\ x\ else\ 0)$$

evaluate each of the following expressions, showing the derivations:

(a) $sum(2)$

(b) $if\ (1 == 2)\ then\ 3\ else\ 4$

(c) $sum(sum(0))$

(d) $rec()$

(e) $default(false, rec())$

How would the evaluations in each case change if we used *call-by-value* semantics instead?

Exercise 4 Consider the following type system for a language with integers, conditionals, pairs, and functions:

$$\frac{n \text{ is an integer literal}}{\Gamma \vdash n : \texttt{Int}} \qquad \frac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau}$$

$$\frac{\Gamma \vdash e_1 : \texttt{Int} \qquad \Gamma \vdash e_2 : \texttt{Int}}{\Gamma \vdash e_1 + e_2 : \texttt{Int}}$$

$$\frac{\Gamma \vdash e_1 : \texttt{Int} \qquad \Gamma \vdash e_2 : \texttt{Int}}{\Gamma \vdash e_1 \times e_2 : \texttt{Int}}$$

$$\frac{b \text{ is a boolean literal}}{\Gamma \vdash b : \texttt{Bool}} \qquad \frac{\Gamma \vdash e : \texttt{Bool}}{\Gamma \vdash \texttt{not e} : \texttt{Bool}}$$

$$\frac{\Gamma \vdash e_1 : \texttt{Bool} \qquad \Gamma \vdash e_2 : \texttt{Bool}}{\Gamma \vdash e_1 \wedge e_2 : \texttt{Bool}} \qquad \frac{\Gamma \vdash e_1 : \texttt{Bool} \qquad \Gamma \vdash e_2 : \texttt{Bool}}{\Gamma \vdash e_1 \vee e_2 : \texttt{Bool}}$$

$$\frac{\Gamma \vdash e_1 : \texttt{Bool} \qquad \Gamma \vdash e_2 : \tau \qquad \Gamma \vdash e_3 : \tau}{\Gamma \vdash if\ e_1\ then\ e_2\ else\ e_3 : \tau}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \qquad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash (e_1, e_2) : (\tau_1, \tau_2)}$$

$$\frac{\Gamma \vdash e : (\tau_1, \tau_2)}{\Gamma \vdash fst(e) : \tau_1} \qquad \frac{\Gamma \vdash e : (\tau_1, \tau_2)}{\Gamma \vdash snd(e) : \tau_2}$$

$$\frac{\Gamma \oplus \{x : \tau_1\} \vdash e : \tau_2}{\Gamma \vdash x \Rightarrow e : \tau_1 \to \tau_2} \qquad \frac{\Gamma \vdash e_1 : \tau_1 \to \tau_2 \qquad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash e_1 e_2 : \tau_2}$$

1. Given the following type derivation with type variables $\tau_1, \ldots, \tau_5$, choose the correct options:

$$\dfrac{\dfrac{\dfrac{\dfrac{(x, \tau_4) \in \Gamma}{\Gamma \vdash x : \tau_4}}{\Gamma \vdash fst(x) : \tau_3} \qquad \dfrac{\dfrac{(x, \tau_4) \in \Gamma}{\Gamma \vdash x : \tau_4}}{\Gamma \vdash snd(x) : \tau_5}}{\Gamma \vdash fst(x)(snd(x)) : \tau_2}}{\Gamma' \vdash x \Rightarrow fst(x)(snd(x)) : \tau_1}$$

   (a) There are no valid assignments to the type variables such that the above derivation is valid.

   (b) In all valid derivations, $\tau_2 = \tau_5$.

   (c) There are *no* valid derivations where $\tau_2 = \texttt{Int}$.

   (d) In all valid derivations, $\tau_4 = (\tau_3, \tau_5)$

   (e) In all valid derivations, $\tau_2 = \tau_4 \to \tau_1$

   (f) There is a valid derivation where $\tau_1 = \tau_2$.

2. For each of the following pairs of terms and types, provide a valid type derivation or briefly argue why the typing is incorrect:

   (a) $x \Rightarrow x + 5$: $\texttt{Int} \to \texttt{Int}$

   (b) $x \Rightarrow y \Rightarrow x + y$: $\texttt{Int} \to \texttt{Int} \to \texttt{Int}$

   (c) $x \Rightarrow y \Rightarrow y(2) \times x$: $\texttt{Int} \to \texttt{Int} \to \texttt{Int}$

   (d) $x \Rightarrow (x, x)$: $\texttt{Int} \to \texttt{(Int, Int)}$

   (e) $x \Rightarrow y \Rightarrow if\ fst(x)\ then\ snd(x)\ else\ y$: $\texttt{(Bool, Int)} \to \texttt{(Int, Int)} \to \texttt{Int}$

   (f) $x \Rightarrow y \Rightarrow if\ y\ then\ (z \Rightarrow y)\ else\ x$: $(\texttt{Bool} \to \texttt{Bool}) \to \texttt{Bool} \to (\texttt{Bool} \to \texttt{Bool})$

   (g) $x \Rightarrow y \Rightarrow if\ y\ then\ (z \Rightarrow y)\ else\ x$: $(\texttt{Int} \to \texttt{Bool}) \to \texttt{Bool} \to (\texttt{Int} \to \texttt{Bool})$

3. Prove that there is *no* valid type derivation for the term

$$x \Rightarrow if\ fst(x)\ then\ snd(x)\ else\ x$$