1. Consider the rules:

\[
\begin{align*}
\text{zero} & \rightarrow \text{nat} \\
\text{succ}(n) & \rightarrow \text{nat} \\
\text{nil} & \rightarrow \text{list} \\
\text{cons}(n, l) & \rightarrow \text{list}
\end{align*}
\]

These rules define a set of terms \(\text{nat}\) representing natural numbers in Peano encoding and a set of terms \(\text{list}\) representing lists of such numbers.

We can inductively (i.e., recursively) define the following \(\text{append}\) function on lists:

\[
\begin{align*}
\text{append}(\text{nil}, m) &= m \\
\text{append}(\text{cons}(n, l), m) &= \text{cons}(n, \text{append}(l, m))
\end{align*}
\]

(a) Represent \(\text{append}\) as a ternary relation and give its definition inductively.

Solution:
Let the relation \(A\) be the smallest set such that

i. For every \(y\) such that \(y \text{ list}\) we have \((\text{nil}, y, y) \in A\).

ii. If \((x, y, z) \in A\) and \(a \text{ nat}\), then \((\text{cons}(a, x), y, \text{cons}(a, z)) \in A\).

(b) Write down a set of inference rules that defines the same ternary relation.

Solution:

\[
\frac{\text{y list}}{\text{append}(\text{nil}, y)} \quad \frac{\text{append}(x, y, z) \quad a \text{ nat}}{\text{append}(\text{cons}(a, x), y, \text{cons}(a, z))}
\]

(c) Prove that the so-defined relation is single-valued, i.e., that it represents a binary function.
To show:
If \(\text{append}(x, y, z)\) and \(\text{append}(x, y, z')\), then \(z = z'\).

Proof:
By induction on the derivation of \(\text{append}(x, y, z)\).

**Case 1:** Rule \(\text{r1}\) was used to derive \(\text{append}(x, y, z)\), so \(x = \text{nil}\) and \(y = z\). Since \(x = \text{nil}\), rule \(\text{r1}\) must also have been used to derive \(\text{append}(x, y, z')\). Thus, \(z' = y = z\).

**Case 2:** Rule \(\text{r2}\) was used to derive \(\text{append}(x, y, z)\). Thus, \(x = \text{cons}(a, x_0)\) for some \(a\) and \(x_0\), and \(z = \text{cons}(a, z_0)\). Furthermore, inversion of \(\text{r2}\) gives \(\text{append}(x_0, y, z_0)\). Since \(x \neq \text{nil}\), \(\text{r2}\) must also have been used to derive \(\text{append}(x_0, y, z_0')\). Thus, we have \(z' = \text{cons}(a, z_0')\) and \(\text{append}(x_0, y, z_0')\) for some \(z_0'\). Using the induction hypothesis we find that \(z_0 = z_0'\). Therefore, \(z = \text{cons}(a, z_0) = \text{cons}(a, z_0') = z'\) as required.

2. (See Chapter 2.1) Let \(s \mapsto s'\) be some arbitrary binary relation and let \(\mapsto^*\) be defined by the following two inference rules:

\[
\begin{align*}
\text{REFL} & : & s & \mapsto^* s \\
\text{TRANS} & : & s & \mapsto s' & s' & \mapsto^* s'' & \Rightarrow & s & \mapsto^* s''
\end{align*}
\]

Prove that \(\mapsto^*\) is indeed transitive, i.e., that \(\forall s, s', s''. s \mapsto^* s' \land s' \mapsto^* s'' \Rightarrow s \mapsto^* s''\).

**Solution:**
By induction on the derivation of \(s \mapsto^* s'\):

**Case 1:** Rule \(\text{REFL}\) was used last to derive \(s \mapsto^* s'\), so \(s = s'\). Thus, trivially, \(s \mapsto^* s''\).

**Case 2:** Rule \(\text{TRANS}\) was used last to derive \(s \mapsto^* s'\). Thus, by inversion of the rule there exists a \(t\) such that \(s \mapsto t\) and \(t \mapsto^* s'\). We use the IH on \(t \mapsto^* s'\) and \(s' \mapsto^* s''\), finding that \(t \mapsto^* s''\). Using rule \(\text{TRANS}\) in forward direction on \(s \mapsto t\) and \(t \mapsto^* s''\) lets us conclude that \(s \mapsto^* s''\) as required.

3. Consider a language where all values are Peano-encoded natural numbers given by the \text{nat} judgment from question 1. The expressions \(e\) of the language shall be of one of the following forms: \text{zero} representing the constant 0, \(\text{succ}(e)\) representing the operation of producing the successor of a given argument, \(\text{pred}(e)\) representing the operation of producing the natural predecessor of a given argument, and \(\text{if0}(e_1, e_2, e_3)\) representing

\[\text{The natural predecessor of } n + 1 \text{ is } n, \text{ and the natural predecessor of } 0 \text{ is taken to be } 0.\]
a test of \(e_1\) for being 0, returning the result of \(e_2\) if it is or the result of \(e_3\) if it is not.

(a) Give a definition of \(e\) in BNF style.

Solution:

\[
e ::= \text{zero} \mid \text{succ}(e) \mid \text{pred}(e) \mid \text{if0}(e, e, e)
\]

(b) Give equivalent inference rules for a judgment \(e \exp\) which holds if \(e\) is an expression of the language.

\[
\begin{array}{c}
\text{zero} \Rightarrow \text{zero} \\
\text{suc}(e) \Rightarrow \text{suc}(n) \\
\text{pred}(e) \Rightarrow \text{pred}(n)
\end{array}
\]

(c) Give a set of inference rules for judgments of the form \(e \Rightarrow n\) where \(e\) is an expression and \(n\) is a natural number (in Peano-encoding). The judgment should express the “evaluates-to” relation in the style of a big-step operational semantics and must correspond to the informal description given above.

The tricky bits are:

- We need two rules for \(\text{pred}\)—one for the case that the argument evaluates to \(\text{zero}\) and one for the case where the argument evaluates to some \(\text{suc}(n)\).

- The rules for \(\text{if0}\) require an explicit premise of the form \(e_3 \exp\) or \(e_2 \exp\) for the sub-term that does not get evaluated. Otherwise the statement to be proved in part (d) would not be true.

Here are the rules:

\[
\begin{array}{c}
\text{zero} \Rightarrow \text{zero} \quad \text{E-Z} \\
\text{suc}(e) \Rightarrow \text{suc}(n) \quad \text{E-S} \\
\text{pred}(e) \Rightarrow \text{zero} \quad \text{E-P(z)} \\
\text{pred}(e) \Rightarrow n \quad \text{E-P(s)} \\
\text{if0}(e_1, e_2, e_3) \Rightarrow v \quad \text{E-C(z)} \\
\text{if0}(e_1, e_2, e_3) \Rightarrow v \quad \text{E-C(s)}
\end{array}
\]
(d) Prove that if $e \Rightarrow n$ is derivable, then so is $e \text{ exp}$ as well as $n \text{ nat}$.

\begin{center}
\begin{tabular}{|l|}
\hline
\textbf{n nat} & The proof proceeds by induction on the derivation of $e \Rightarrow n$. The cases $E-Z$ and $E-P(Z)$ are trivial by rule \textit{zero}. Case $E-C(Z)$ follows directly from the IH for $e_2$. Similarly, case $E-C(S)$ follows from the IH for $e_3$. For case $E-S$ we use the IH on $e$ and then use rule \textit{succ}. For case $E-P(S)$ we use the IH on $e$ and then apply the inversion of rule \textit{succ}. (As discussed in class, this inversion is an admissible rule.) \\
\hline
\end{tabular}
\end{center}

\textit{e exp} Again, the proof proceeds by induction on the derivation of $e \Rightarrow n$. Case $E-Z$ is immediate by rule \textit{z}. Case $E-S$ uses the IH on $e$ and then rule \textit{s}; cases $E-P(Z)$ and $E-P(S)$ use the IH on $e$ and then rule \textit{p}. Case $E-C(Z)$ uses the IH on $e_1$ and $e_2$ and then applies rule \textit{c}. Notice that the last step requires to know that $e_3 \text{ exp}$, which is given by inversion of $E-C(Z)$. Case $E-C(S)$ is analogous to $E-C(Z)$, with the roles of $e_2$ and $e_3$ swapped.

(e) Prove that the relation $\Rightarrow$ defined by your rules is single-valued.
To show:
If $e \Rightarrow n$ and $e \Rightarrow n'$, then $n = n'$.

Proof:
By induction on the derivation of $e \Rightarrow n$.

**E-z:** We have $e = \text{zero}$ and $n = \text{zero}$. E-z must have been used to derive $e \Rightarrow n'$, so $n' = \text{zero} = n$.

**E-s:** We have $e = \text{succ}(e_0)$, $n = \text{succ}(n_0)$, and $e_0 \Rightarrow n_0$. E-s must have been used to derive $e \Rightarrow n'$, so $n' = \text{succ}(n'_0)$ and $e_0 \Rightarrow n'_0$. By IH: $n_0 = n'_0$. Thus, $n = \text{succ}(n_0) = \text{succ}(n'_0) = n'$.

**E-p(z):** We have $e = \text{pred}(e_0)$, $n = \text{zero}$ and $e_0 \Rightarrow \text{zero}$. Two sub-cases:

- **E-P(z) used for** $e \Rightarrow n'$: Here $n' = \text{zero} = n$.
- **E-P(s) used for** $e \Rightarrow n'$: Here $e_0 \Rightarrow \text{succ}(n_0)$ for some $n_0$. By IH this means that $\text{zero} = \text{succ}(n_0)$, which is a contradiction. (This means that E-P(s) could not have been used for $e \Rightarrow n'$ after all.)

**E-P(s):** We have $e = \text{pred}(e_0)$ and $e_0 \Rightarrow \text{succ}(n)$. Two sub-cases:

- **E-P(z) used for** $e \Rightarrow n'$: $e_0 \Rightarrow \text{zero}$, so by IH, $\text{zero} = \text{succ}(n)$, i.e., contradiction.
- **E-P(s) used for** $e \Rightarrow n'$: $e_0 \Rightarrow \text{succ}(n')$. By IH: $\text{succ}(n) = \text{succ}(n')$, so $n = n'$.

**E-C(z):** We have $e = \text{if0}(e_1, e_2, e_3)$ and $e_1 \Rightarrow \text{zero}$. By reasoning analogous to case E-P(z) it must be that $e \Rightarrow n'$ also uses rule E-C(z) (as opposed to E-C(s)). We use the IH on $e_2$, which gives the desired result.

**E-C(s):** We have $e = \text{if0}(e_1, e_2, e_3)$ and $e_1 \Rightarrow \text{succ}(n_1)$ for some $n_1$. By reasoning analogous to case E-P(s) it must be that $e \Rightarrow n'$ also uses rule E-C(s) (as opposed to E-C(s)). We use the IH on $e_3$, which gives the desired result.