1 Two-level Syntax

Exercise 1 (Derivation trees). In a tree adjoining grammar \( G = \langle N, \Sigma, T_\alpha, T_\beta, S \rangle \), the trees in \( \mathcal{L}(G) \) are called derived trees. We are interested here in another tree structure, called a derivation tree, for which we propose a formalisation here. Let us assume for simplicity that all the foot nodes of auxiliary trees have the 'na' null adjunction annotation.

For an elementary tree \( \gamma \in T_\alpha \cup T_\beta \), we define its contents \( c(\gamma) \) to be a finite sequence over the alphabet \( Q \) defined as \( \{ q_A \mid A \in N \cup N_\downarrow \} \). Formally, we enumerate for this the labels in \( Q \) of its nodes in position order; the nodes labelled by \( \Sigma \cup N_{na} \) are ignored.

Consider for instance the TAG \( G_1 \) with \( N \) defined as \( \{ S, NP, VP \} \), \( \Sigma \) defined as \( \{ VBZ, NNP, NNS, RB \} \), \( T_\alpha \) defined as \( \{ eats, Bill, mushrooms \} \), \( T_\beta \) defined as \( \{ possibly \} \), and \( S \) defined as \( S \), where the elementary trees are shown below:

\[
\begin{array}{cccc}
S & NP & VP & \downarrow \\
\uparrow & \downarrow & \downarrow & \downarrow \\
NP & VP & NNP \circ & NNS \circ \\
\lor & \lor & \downarrow & \downarrow \\
VBZ \circ & NP \downarrow & & \downarrow \\
(eats) & (Bill) & (mushrooms) & (possibly)
\end{array}
\]

Then \( eats \) has contents \( c(eats) = q_S, q_{NP\downarrow}, q_{VP\downarrow}, q_{NP\downarrow} \), \( c(Bill) = q_{NP} \), \( c(mushrooms) = q_{NP} \), and \( c(possibly) = q_{VP} \).

We now define a finite ranked alphabet \( \mathcal{F} \) defined as \( T_\alpha \cup T_\beta \cup \{ \varepsilon^{(0)} \} \). For an elementary tree \( \gamma \in T_\alpha \cup T_\beta \), its rank is \( r(\gamma) \) defined as the length of its contents. For the symbol \( \varepsilon \), its rank is \( r(\varepsilon) \) defined as 0. For a TAG \( G = \langle N, \Sigma, T_\alpha, T_\beta, S \rangle \), we construct a finite tree automaton \( \mathcal{A}_G \) defined as \( \langle Q, \mathcal{F}, \delta, q_{S\downarrow} \rangle \) where \( Q \) and \( \mathcal{F} \) are defined as above and

\[
\delta \equiv \{(q_A, \alpha^{(r(\alpha))}, c(\alpha)) \mid A \downarrow \in N \downarrow, \alpha \in T_\alpha, rl(\alpha) = A\} \\
\cup \{(q_A, \beta^{(r(\beta))}, c(\beta)) \mid A \in N, \beta \in T_\beta, rl(\beta) = A\} \\
\cup \{(q_A, \varepsilon^{(0)}) \mid A \in N\}
\]

where 'rl' returns the root label of the tree.
1. Give the finite automaton $A_{G_1}$ associated with the example TAG $G_1$.

Solution:

\[ Q = \{ q_{S↓}, q_{NP↓}, q_S, q_{VP}, q_{NP} \}, \]
\[ F = \{ \textit{eats}^{(4)}, \textit{Bill}^{(1)}, \textit{mushrooms}^{(1)}, \textit{possibly}^{(1)}, \varepsilon^{(0)} \}, \]
\[ \delta = \{ (q_{S↓}, \textit{eats}^{(4)}, q_S, q_{NP↓}, q_{VP}, q_{NP}), \]
\[ (q_{NP↓}, \textit{Bill}^{(1)}, q_{NP}), \]
\[ (q_{NP↓}, \textit{mushrooms}^{(1)}, q_{NP}), \]
\[ (q_{S↓}, \varepsilon^{(0)}), \]
\[ (q_{VP}, \textit{possibly}^{(1)}, q_{VP}), \]
\[ (q_{VP}, \varepsilon^{(0)}), \]
\[ (q_{NP}, \varepsilon^{(0)}) \} \]

2. Modify your automaton in order to also handle the trees \textit{real}, \textit{fake}, \textit{wants_to0}, \textit{wants_to1} $\in T_\beta$ shown below, where $TO\circ, JJ\circ \in \Sigma$:

\[
\begin{array}{lcc}
\text{NP} & \text{VP} & \text{VP} \\
JJ\circ & \text{NP}^{na} & \text{VP}^{na} & \text{VP}^{na} \\
\text{NP}^{na} & \text{VP}^{na} & \text{VP}^{na} & \text{VP}^{na} \\
\end{array}
\]

(\textit{real}) (\textit{fake}) (\textit{wants_to0}) (\textit{wants_to1})

We call the resulting tree adjoining grammar $G_2$.

Solution: Add \textit{someone}^{(1)}, \textit{real}^{(1)}, \textit{fake}^{(1)}, and \textit{wants_to}^{(3)} to $F$ and the rules

\[
(q_{NP}, \textit{real}^{(1)}, q_{NP})
\]
\[
(q_{NP}, \textit{fake}^{(1)}, q_{NP})
\]
\[
(q_{VP}, \textit{wants_to0}^{(1)}, q_{VP})
\]
\[
(q_{NP}, \textit{wants_to1}^{(2)}, q_{VP}, q_{NP↓})
\]

to $\delta$.

3. The intention that our finite automaton generates the derivation language $L_D(G) \overset{\text{def}}{=} L(A_G)$ of $G$. Can you figure out what should be the derivation tree of ‘Bill possibly wants to eat mushrooms’?
Solution:

\[
\begin{array}{c}
\varepsilon & \text{eats} & \text{Bill} \\
\varepsilon & \text{wants} & \text{mushrooms} \\
\varepsilon & \text{possibly} & \varepsilon
\end{array}
\]

[2] 4. Give a PDL node formula \( \varphi_2 \) such that \( L(\mathcal{A}_{g_2}) = \{ t \in T(\mathcal{F}) \mid t, \text{root} \models \varphi_2 \} \).

Solution:

\[
\varphi_1 \overset{\text{def}}{=} \varphi_{SL} \land [_1^*](
\begin{array}{c}
eats \implies (\downarrow; \text{first}?, \varphi_S?; \rightarrow; \varphi_{NP}?; \rightarrow; \varphi_{VP}?; \rightarrow; \varphi_{NP}?))_{\text{last}} \\
wants_{to0} \implies (\downarrow; \text{first}?, \varphi_{VP}?)_{\text{last}} \\
wants_{to1} \implies (\downarrow; \text{first}?, \varphi_{VP}?; \rightarrow; \varphi_{NP}?))_{\text{last}} \\
\text{Bill} \implies (\downarrow; \text{first}?, \varphi_{NP}?))_{\text{last}} \\
\text{real} \implies (\downarrow; \text{first}?, \varphi_{NP}?))_{\text{last}} \\
\text{fake} \implies (\downarrow; \text{first}?, \varphi_{NP}?))_{\text{last}} \\
\text{mushrooms} \implies (\downarrow; \text{first}?, \varphi_{NP}?))_{\text{last}} \\
\text{possibly} \implies (\downarrow; \text{first}?, \varphi_{VP}?)_{\text{last}} \\
\varepsilon \implies \text{leaf}
\end{array}
\)

where

\[
\begin{align*}
\varphi_{SL} & \overset{\text{def}}{=} \text{eats} \\
\varphi_{NP} & \overset{\text{def}}{=} \text{Bill} \lor \text{mushrooms} \\
\varphi_S & \overset{\text{def}}{=} \varepsilon \\
\varphi_{VP} & \overset{\text{def}}{=} \text{possibly} \lor \text{wants}_{to0} \lor \text{wants}_{to1} \lor \varepsilon \\
\varphi_{NP} & \overset{\text{def}}{=} \text{real} \lor \text{fake} \lor \varepsilon
\end{align*}
\]

1.1 Macro Tree Transducers

Let \( \mathcal{X} \) be a countable set of variables and \( \mathcal{Y} \) a countable set of parameters; we assume \( \mathcal{X} \) and \( \mathcal{Y} \) to be disjoint. For \( Q \) a ranked alphabet with arities greater than zero, we abuse notations and write \( Q(\mathcal{X}) \) for the alphabet of pairs \((q, x) \in Q \times \mathcal{X} \) with \( \text{arity}(q, x) \overset{\text{def}}{=} \text{arity}(q) - 1 \). This is just for convenience, and \((q, x)(t_1, \ldots, t_n)\) is really the term \( q(x, t_1, \ldots, t_n) \).

**Syntax.** A macro tree transducer (NMTT) is a tuple \( \mathcal{M} = (Q, \mathcal{F}, \mathcal{F}', \Delta, I) \) where \( Q \) is a finite set of states, all of arity \( \geq 1 \), \( \mathcal{F} \) and \( \mathcal{F}' \) are finite ranked alphabets, \( I \subseteq Q_1 \) is a set of root states of arity one, and \( \Delta \) is a finite set of term rewriting rules of the form \( q(f(x_1, \ldots, x_n), y_1, \ldots, y_p) \rightarrow e \) where \( q \in Q_{1+p} \) for some \( p \geq 0 \), \( f \in \mathcal{F}_n \) for some \( n \in \mathbb{N} \),
and \(e \in T(\mathcal{F}^t \cup Q(\mathcal{X}_n), \mathcal{Y}_p)\). Note that this imposes that any occurrence in \(e\) of a variable \(x \in \mathcal{X}\) must be as the first argument of a state \(q \in Q\).

**Inside-Out Semantics.** Given a NMTT, the *inside-out* rewriting relation over trees in \(T(\mathcal{F} \cup \mathcal{F}' \cup Q)\) is defined by: \(t \xrightarrow{\Delta} t'\) if there exist a rule \(q(f(x_1, \ldots, x_n), y_1, \ldots, y_p) \rightarrow e\) in \(\Delta\), a context \(C \in C(\mathcal{F} \cup \mathcal{F}' \cup Q)\), and two substitutions \(\sigma: \mathcal{X} \rightarrow T(\mathcal{F})\) and \(\rho: \mathcal{Y} \rightarrow T(\mathcal{F}')\) such that \(t = C[q(f(x_1, \ldots, x_n), y_1, \ldots, y_p)\sigma\rho]\) and \(t' = C[q\sigma\rho]\). In other words, in inside-out rewriting, when applying a rewriting rule \(q(f(x_1, \ldots, x_n), y_1, \ldots, y_p) \rightarrow e\), the parameters \(y_1, \ldots, y_p\) must be mapped to trees in \(T(\mathcal{F}')\), with no remaining states from \(Q\).

Similarly to context-free tree grammars, the *inside-out* transduction \([\mathcal{M}]_{IO}\) realised by \(\mathcal{M}\) is defined through inside-out rewriting semantics:

\[
[\mathcal{M}]_{IO} \overset{\text{def}}{=} \{(t, t') \in T(\mathcal{F}) \times T(\mathcal{F'}) \mid \exists q \in I, q(t) \xrightarrow{\Delta}^* t'\}.
\]

**Example 1.** Let \(\mathcal{F} \overset{\text{def}}{=} \{a(1), s(0)\}\) and \(\mathcal{F'} \overset{\text{def}}{=} \{f(3), a(1), b(1), s(0)\}\). Consider the NMTT \(\mathcal{M} = \{q(1), q(3)\}, \mathcal{F}, \mathcal{F}', \Delta, \{q\}\) with \(\Delta\) the set of rules

\[
\begin{align*}
q(a(x_1)) & \rightarrow q'(x_1, s, s) \\
q'(a(x_1), y_2) & \rightarrow q'(x_1, y_2, a(y_2)) \\
q'(a(x_1), y_2) & \rightarrow q'(x_1, b(y_1), a(y_2))
\end{align*}
\]

Then we have for instance the following derivation:

\[
q(a(a(s))) \xrightarrow{\Delta} q'(a(a(s)), s, s) \xrightarrow{\Delta} q'(a(s), b(s), b(s)) \xrightarrow{\Delta} q'(s, a(b(s)), b(b(s))) \xrightarrow{\Delta} f(a(b(s)), a(b(s)), b(b(s)))
\]

showing that \((a(a(s))), f(a(b(s)), a(b(s)), b(b(s))) \in \mathcal{M}\).

**Exercise 2** (Monadic trees). An NMTT \(\mathcal{M}\) is called *linear* and *non-deleting* if, in every rule \(q(f(x_1, \ldots, x_n), y_1, \ldots, y_p) \rightarrow e\) in \(\Delta\), the term \(e\) is linear in \(\{x_1, \ldots, x_n\}\) and \(\{y_1, \ldots, y_p\}\), i.e. each variable and each parameter occurs exactly once in the term \(e\).

Let \(\mathcal{F} \overset{\text{def}}{=} \{a(1), b(1), s(0)\}\). Observe that trees in \(T(\mathcal{F}')\) are in bijection with contexts in \(C(\mathcal{F}')\) and words over \(\{a, b\}^*\). For a context \(C\) from \(C(\mathcal{F}')\), we write \(C^R\) for its *mirror context*, read from the leaf to the root. For instance, if \(C = a(b(a(\square))))\), then \(C^R = a(a(b(a(\square))))\). Formally, let \(n \in \mathbb{N}\) be such that \(\text{dom } C = \{0^m \mid m \leq n\}\); then \(C(0^n) = \square\) and \(C(0^m) \in \{a, b\}\) for \(m < n\). Then \(C^R\) is defined by \(\text{dom } C^R \overset{\text{def}}{=} \text{dom } C\), \(C^R(0^n) \overset{\text{def}}{=} \square\), and \(C^R(0^m) \overset{\text{def}}{=} C^R(0^{n-m})\) for all \(m < n\).
1. Give a linear and non-deleting NMTT \( \mathcal{M} \) from \( \mathcal{F}' \) to \( \mathcal{F}' \) such that \( \mathcal{M}_{10} = \{(C[\$], C[C^R[\$]]) \mid C \in C(\mathcal{F}')\} \). In terms of words over \( \{a, b\}^* \), this transducer maps \( w \) to the palindrome \( ww^R \). Is \( \mathcal{M}_{10}(T(\mathcal{F})) \) a recognisable tree language?

**Solution:** Let \( \mathcal{M} = (Q, \mathcal{F}', \mathcal{F}', \Delta, I) \) where \( Q = \{q_1, q_2\} \), \( I = \{q_1\} \), and \( \Delta \) is the set of rules

\[
q_1(\$) \rightarrow \$ \quad q_2(a(x)) \rightarrow a(q_1, a(\$)) \quad q_2(b(x)) \rightarrow b(q_1, b(\$)) \quad q_2(\$, y_1) \rightarrow y_1 \quad q_2(a(x), y_1) \rightarrow a(q_1, a(y_1)) \quad q_2(b(x), y_1) \rightarrow b(q_1, b(y_1)).
\]

We leave the proof of correctness to the reader.

This macro tree transducer is deterministic, and complete. Because a monadic tree language over \( \mathcal{F}' \) is recognisable if and only if the corresponding word language over \( \{a, b\} \) is recognisable, \( \mathcal{M}_{10}(T(\mathcal{F})) \) is not a recognisable tree language. In turn, this shows that recognisable tree languages are not closed under linear non-deleting macro transductions, not even the complete deterministic ones.

**Exercise 3** (From derivation to derived trees). Consider again the tree adjoining grammar \( \mathcal{G}_2 \) from Exercise 1.

1. Give a linear non-deleting NMTT \( \mathcal{M}_2 \) that maps the derivation trees of \( \mathcal{G}_2 \) to its derived trees. Formally, we want \( \text{dom}(\mathcal{M}_2) = L_D(\mathcal{G}_2) \) and \( \mathcal{M}_2_{10}(T(\mathcal{F})) = L_T(\mathcal{G}_2) \).

**Solution:** We set \( \mathcal{F}' = N \uplus \Sigma, Q = \{q_S^{(1)}, q_S^{(2)}, q_{NP}^{(1)}, q_{NP}^{(2)}, q_{VP}^{(2)}\}, I = \{q_S^{(1)}\} \), and \( \Delta \):

\[
q_S^{(1)}(\text{eats}(x_1, x_2, x_3, x_4)) \rightarrow q_S^{(2)}.
\]
**Exercise 4** (Context-free tree grammar). Let $\mathcal{M} = (Q, \mathcal{F}, \mathcal{F}', \Delta, I)$ be an NMTT and $\mathcal{A} = (Q', \mathcal{F}, \delta, I')$ be an NFTA.

1. Show that $L \overset{\text{def}}{=} [\mathcal{M}]_{IO}(L(\mathcal{A})) = \{ t' \in T(\mathcal{F}') \mid \exists t \in L(\mathcal{A}). (t, t') \in [\mathcal{M}]_{IO} \}$ is an inside-out context-free tree language, i.e., show how to construct a CFTG $\mathcal{G} = (N, \mathcal{F}', S, R)$ such that $L_{IO}(\mathcal{G}) = L$.

**Solution:** Let

$N \overset{\text{def}}{=} (Q \times Q') \cup \{ S \}$

where each pair $(q^{(1+p)}, q')$ from $Q \times Q'$ has arity $p$, and

$R \overset{\text{def}}{=} \{ S \to (q, q')^{(0)} \mid q \in I, q' \in I' \}$

$\cup \{ (q, q')^{(p)}(y_1, \ldots, y_p) \to e[q_i/x_i] \mid \exists n. \exists f \in \mathcal{F}_n. q^{(1+p)}(f(x_1, \ldots, x_n), y_1, \ldots, y_n) \to e \in \Delta$ and $(q', f, q'_1, \ldots, q'_n) \in \delta \}$

where we abuse notation as indicated at the beginning of the section. For a tree $e \in T(N \cup \mathcal{F}')$, we let $N(e) = \{(q_1, q'_1), \ldots, (q_n, q'_n)\}$ be the set of symbols from $N$ occurring inside $e$.

Let us show that, for all $k \in \mathbb{N}$, for all $e \in T(N \cup \mathcal{F}')$ with $N(e) = \{(q_1, q'_1), \ldots, (q_n, q'_n)\}$ and for all $t' \in T(\mathcal{F}')$, $e \overset{IO^k}{\Rightarrow} G t'$ if and only if $\exists t_1, \ldots, t_n \in T(\mathcal{F})$ such that $e[t_i/q'_i]_{1 \leq i \leq n} \overset{IO^k}{\Rightarrow} M t'$ and for all $1 \leq i \leq n, t_i \overset{\delta^*}{\Rightarrow} A q'_i$.

We prove the statement by induction, first over $k$ the number of rewriting steps in $\mathcal{G}$ and $\mathcal{M}$, and second over the term $e$. We only prove the ‘if’ direction, as the ‘only if’ one is similar.

**If** Assume $e \overset{IO^k}{\Rightarrow} G t'$.

**If** $e = f(e_1, \ldots, e_m)$ for some $m \in \mathbb{N}$ and $f \in \mathcal{F}'_m$, then this rewrite can be decomposed as

$$e = f(e_1, \ldots, e_m) \overset{IO^k}{\Rightarrow} G f(t'_1, \ldots, t'_m) = t'$$

where for all $1 \leq j \leq m, t'_j \in T(\mathcal{F}')$ is such that

$$e_j \overset{IO^{k_j}}{\Rightarrow} G t'_j$$

and

$$k = \sum_{1 \leq j \leq m} k_j$$
Let $N(e_j) = \{(q_{j,1}, q'_{j,1}), \ldots, (q_{j,n_j}, q'_{j,n_j})\}$; then $N(e) = \bigcup_{1 \leq j \leq m} N(e_j)$.

For each $1 \leq j \leq m$, by induction hypothesis on the subterms $e_j$ since $k_j \leq k$, there exist $t_{j,1}, \ldots, t_{j,n_j} \in T(F)$ such that

$$e_j[t_{j,i}/q'_{j,i}]_{1 \leq i \leq n_j} \xrightarrow{\text{IO}_{k_j}} t'_j$$

and

$$t_{j,i} \xrightarrow{\delta_{h^*}} q'_{j,i}$$

for all $1 \leq i \leq n_j$. Thus

$$f(e_1, \ldots, e_m)[t_{j,i}/q'_{j,i}]_{1 \leq j \leq m, 1 \leq i \leq n_j} \xrightarrow{\text{IO}_{k}} f(t'_1, \ldots, t'_m) = t'$$

as desired.

If $e = (q, q')(p)(e_1, \ldots, e_p)$ for some $p \in \mathbb{N}$ and $(q, q')(p) \in Q \times Q'$, then this rewrite can be decomposed as

$$e = (q, q')(p)(e_1, \ldots, e_p) \xrightarrow{\text{IO}_{k'}} (q, q')(p)(t'_1, \ldots, t'_p)$$

$$\xrightarrow{\text{IO}_G} e'[q'_i/x_i]_{1 \leq i \leq m}[t'_j/y_j]_{1 \leq j \leq p}$$

$$\xrightarrow{\text{IO}_{k''}} t'$$

where for all $1 \leq j \leq m$, $t'_j \in T(F')$ is such that

$$e_j \xrightarrow{\text{IO}_{k_j}} t'_j$$

and $k' = \sum_{1 \leq j \leq m} k_j$ and $k = 1 + k' + k''$; also $N(e) = \{(q, q')\} \bigcup \bigcup_{1 \leq j \leq p} N(e_j)$ where $N(e_j) = \{(q_{j,1}, q'_{j,1}), \ldots, (q_{j,n_j}, q'_{j,n_j})\}$. Such a rule application relies on the existence of $m \in \mathbb{N}$ and $f \in F_m$ such that there are rules $q^{(1+p)}(f(x_1, \ldots, x_m), y_1, \ldots, y_p) \rightarrow e' \in \Delta$ and $(q', f, q'_{1,p}, \ldots, q'_{m,p}) \in \delta$.

By induction hypothesis on $k_j < k$ for each $1 \leq j \leq p$, there exist $t_{j,1}, \ldots, t_{j,n_j} \in T(F)$ such that

$$e_j[t_{j,i}/q'_{j,i}]_{1 \leq i \leq n_j} \xrightarrow{\text{IO}_{k_j}} t'_j$$

and

$$t_{j,i} \xrightarrow{\delta_{h^*}} q'_{j,i}$$

for all $1 \leq i \leq n_j$. 
Furthermore, \( N(e'[t'_j/y_j]_{1 \leq j \leq p}[q'_i/x_i]_{1 \leq i \leq m}) = \{ (q_1, q'_1), \ldots, (q_m, q'_m) \} \) and by induction hypothesis over \( k'' < k \), there exist \( t_1, \ldots, t_m \in T(\mathcal{F}) \) such that
\[
e'[t'_j/y_j]_{1 \leq j \leq p}[t_i/x_i]_{1 \leq i \leq m} \overset{k''}{\Rightarrow}_M t'
\]
and
\[
t_i \overset{\delta_B^*}{\Rightarrow}_A q'_i
\]
for all \( 1 \leq i \leq m \). Note that, because \( (q', f, q'_1, \ldots, q'_m) \in \delta \), the latter imply
\[
f(t_1, \ldots, t_m) \overset{\delta_B^*}{\Rightarrow}_A f(q'_1, \ldots, q'_m) \overset{\delta_B}{\Rightarrow}_A q'.
\]
Thus, in \( M \), we have the rewrite
\[
e[f(t_1, \ldots, t_m)/q][t'_j, i/q'_j, i]_{1 \leq j \leq m, 1 \leq i \leq n_i}
\]
\[
= q^{(1+p)}(f(t_1, \ldots, t_m), e_1[t'_{1, i}/q'_1, i]_{1 \leq i \leq n_1}, \ldots, e_m[t'_{m, i}/q'_m, i]_{1 \leq i \leq n_m})
\]
\[
= q^{(1+p)}(f(x_1, \ldots, x_m), e_1[t'_{1, i}/q'_1, i]_{1 \leq i \leq n_1}, \ldots, e_m[t'_{m, i}/q'_m, i]_{1 \leq i \leq n_m})[t_1/x_1, \ldots, t_m/x_m]
\]
\[
\overset{10}{\Rightarrow}_{\mathcal{M}} q^{(1+p)}(f(x_1, \ldots, x_m), t'_1, \ldots, t'_p)[t_1/x_1, \ldots, t_m/x_m]
\]
\[
\overset{10}{\Rightarrow}_{\mathcal{M}} e'[t'_i/x_i]_{1 \leq i \leq m}[t'_j/y_j]_{1 \leq j \leq p}
\]
\[
\overset{10}{\Rightarrow}_{\mathcal{M}} t'
\]
as desired.

## 2 Scope Ambiguities and Propositional Attitudes

**Exercise 5.** One considers the two following signatures:

\[
(\Sigma_{\text{ABS}}) \quad \text{SUZY} : NP \\
\text{BILL} : NP \\
\text{MUSHROOM} : N \\
\quad \text{A} : N \rightarrow (NP \rightarrow S) \rightarrow S \\
\quad \text{A}_{\text{inf}} : N \rightarrow (NP \rightarrow S_{\text{inf}}) \rightarrow S_{\text{inf}} \\
\quad \text{EAT} : NP \rightarrow NP \rightarrow S_{\text{inf}} \\
\quad \text{TO} : (NP \rightarrow S_{\text{inf}}) \rightarrow VP \\
\quad \text{WANT} : VP \rightarrow NP \rightarrow S
\]
(Σ_{S-FORM})

\begin{align*}
\textbf{Suzy} & : \text{string} \\
\textbf{Bill} & : \text{string} \\
\textit{mushroom} & : \text{string} \\
\textit{a} & : \text{string} \\
\textit{eat} & : \text{string} \\
\textit{to} & : \text{string} \\
\textbf{wants} & : \text{string}
\end{align*}

where, as usual, \textit{string} is defined to be \( o \rightarrow o \) for some atomic type \( o \).

One then defines a morphism \((\mathcal{L} \text{_{SYNT}} : \Sigma_{\text{ABS}} \rightarrow \Sigma_{S\text{-FORM}})\) as follows:

\begin{align*}
\mathcal{L} \text{_{SYNT}}(NP) & := \text{string} \\
\mathcal{L} \text{_{SYNT}}(N) & := \text{string} \\
\mathcal{L} \text{_{SYNT}}(S) & := \text{string} \\
\mathcal{L} \text{_{SYNT}}(S_{\text{inf}}) & := \text{string} \\
\mathcal{L} \text{_{SYNT}}(VP) & := \text{string} \\
\mathcal{L} \text{_{SYNT}}(\text{SUZY}) & := \textbf{Suzy} \\
\mathcal{L} \text{_{SYNT}}(\text{BILL}) & := \textbf{Bill} \\
\mathcal{L} \text{_{SYNT}}(\text{MUSHROOM}) & := \textit{mushroom} \\
\mathcal{L} \text{_{SYNT}}(A) & := \lambda xy. y (a + x) \\
\mathcal{L} \text{_{SYNT}}(A_{\text{inf}}) & := \lambda xy. y (a + x) \\
\mathcal{L} \text{_{SYNT}}(\text{EAT}) & := \lambda xy. y + \text{eat} + x \\
\mathcal{L} \text{_{SYNT}}(\text{TO}) & := \lambda x. \text{to} + (x \epsilon) \\
\mathcal{L} \text{_{SYNT}}(\text{WANT}) & := \lambda xy. y + \textbf{wants} + x
\end{align*}

where, as usual, the concatenation operator (+) is defined as functional composition, and the empty word (\( \epsilon \)) as the identity function.

1. Give two different terms, say \( t_0 \) and \( t_1 \), such that:

\[ \mathcal{L} \text{_{SYNT}}(t_0) = \mathcal{L} \text{_{SYNT}}(t_1) = \textbf{Bill} + \textbf{wants} + \textit{to} + \textit{eat} + \textit{a} + \textit{mushroom} \]

\textbf{Solution:}

\[ t_0 = \text{WANT} \left( \text{TO} \left( \lambda x. A_{\text{inf}} \text{MUSHROOM} (\lambda y. \text{EAT} y x) \right) \right) \text{BILL} \]
\[ t_1 = \text{A MUSHROOM} (\lambda y. \text{WANT} (\text{TO} (\lambda x. \text{EAT} y x)) \text{BILL}) \]

\textbf{Exercise 6.} One considers a third signature :
suzy : ind
bill : ind
mushroom : ind → prop
eat : ind → ind → prop
want : ind → prop → prop

One then defines a morphism \( \mathcal{L}_{\text{SEM}} : \Sigma_{\text{ABS}} \rightarrow \Sigma_{\text{L-FORM}} \) as follows:

\[
\begin{align*}
NP & := \text{ind} \\
N & := \text{ind} \rightarrow \text{prop} \\
S & := \text{prop} \\
S_{\text{inf}} & := \text{prop} \\
VP & := \text{ind} \rightarrow \text{prop} \\
SUZY & := \text{suzy} \\
BILL & := \text{bill} \\
MUSHROOM & := \text{mushroom} \\
A & := \lambda xy. \exists z. (xz) \land (yz) \\
A_{\text{inf}} & := \lambda xy. \exists z. (xz) \land (yz) \\
EAT & := \lambda xy. \text{eat} \; yx \\
TO & := \lambda x. x \\
WANT & := \lambda xy. \text{want} \; y \; (xx)
\end{align*}
\]

1. Compute the different semantic interpretations of the sentence Bill wants to eat a mushroom, i.e., compute \( \mathcal{L}_{\text{SEM}}(t_0) \) and \( \mathcal{L}_{\text{SEM}}(t_1) \).

Solution:

\[
\begin{align*}
\mathcal{L}_{\text{SEM}}(t_0) & = \text{want} \; \text{bill} \; (\exists z. (\text{mushroom} \; z) \land (\text{eat} \; \text{bill} \; z)) \\
\mathcal{L}_{\text{SEM}}(t_1) & = \exists z. (\text{mushroom} \; z) \land (\text{want} \; \text{bill} \; (\text{eat} \; \text{bill} \; z))
\end{align*}
\]

Exercise 7. One extends \( \Sigma_{\text{ABS}} \) and \( \mathcal{L}_{\text{SYNT}} \), respectively, as follows:

\[
\begin{align*}
(\Sigma_{\text{ABS}}) & \quad \text{WANT2} : \quad NP \rightarrow VP \rightarrow NP \rightarrow S \\
(\mathcal{L}_{\text{SYNT}}) & \quad \text{WANT2} := \lambda yz. z + \text{wants} + x + y
\end{align*}
\]

[1] 1. Extend \( \mathcal{L}_{\text{SEM}} \) accordingly in order to allow for the analysis of a sentence such as Bill wants Suzy to eat a mushroom.
Exercise 8. One extends $\Sigma_{\text{ABS}}$ as follows:

\[ \begin{align*}
(\Sigma_{\text{ABS}}) & \quad \text{EVERYONE} : (NP \to S) \to S \\
& \quad \text{THINK} : S \to NP \to S
\end{align*} \]

in order to allow for the analysis of the following sentence:

(1) everyone thinks Bill wants to eat a mushroom.

1. Extend $\Sigma_{\text{S-FORM}}, \mathcal{L}_{\text{SYNT}}, \Sigma_{\text{L-FORM}},$ and $\mathcal{L}_{\text{SEM}}$ accordingly.

Solution:

\[ \begin{align*}
(\Sigma_{\text{S-FORM}}) & \quad \text{everyone} : \text{string} \\
& \quad \text{thinks} : \text{string} \\
(\mathcal{L}_{\text{SYNT}}) & \quad \text{EVERYONE} := \lambda x. x \text{everyone} \\
& \quad \text{THINK} := \lambda xy. y + \text{thinks} + x \\
(\Sigma_{\text{L-FORM}}) & \quad \text{human} : \text{ind} \to \text{prop} \\
& \quad \text{think} : \text{ind} \to \text{prop} \to \text{prop} \\
(\mathcal{L}_{\text{SEM}}) & \quad \text{EVERYONE} := \lambda x. \forall y. (\text{human} y) \to (x y) \\
& \quad \text{THINK} := \lambda xy. \text{think} y x
\end{align*} \]

2. Give the several $\lambda$-terms that correspond to the different parsings of sentence (1).

Solution: There are four such terms:

\[ \begin{align*}
\text{EVERYONE} (\lambda x. \text{THINK} (\text{WANT} (\text{TO} (\lambda z. \text{INF MUSHROOM} (\lambda y. \text{EAT} y z))) \text{BILL}) x) \\
\text{EVERYONE} (\lambda x. \text{THINK} (\text{A MUSHROOM} (\lambda y. \text{WANT} (\text{TO} (\lambda z. \text{EAT} y z))) \text{BILL})) x) \\
\text{EVERYONE} (\lambda x. \text{A MUSHROOM} (\lambda y. \text{THINK} (\text{WANT} (\text{TO} (\lambda z. \text{EAT} y z))) \text{BILL}) x)) \\
\text{A MUSHROOM} (\lambda y. \text{EVERYONE} (\lambda x. \text{THINK} (\text{WANT} (\text{TO} (\lambda z. \text{EAT} y z))) \text{BILL}) x))
\end{align*} \]