MPRI 2-27-1 Exam

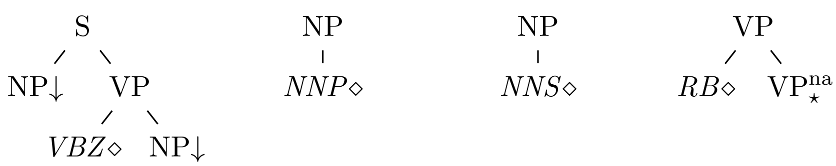
Duration: 3 hours
Paper documents are allowed. The numbers in front of questions are indicative of hardness or duration.

1 Two-level Syntax

Exercise 1 (Derivation trees)**.** In a tree adjoining grammar $\mathcal{G} = \langle N, \Sigma, T_\alpha, T_\beta, S\rangle$, the trees in $L_T(\mathcal{G})$ 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 '$^{\text{na}}$' null adjunction annotation.

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

Consider for instance the TAG $\mathcal{G}_1$ with $N \stackrel{\text{def}}{=} \{\text{S}, \text{NP}, \text{VP}\}$, $\Sigma \stackrel{\text{def}}{=} \{\mathit{VBZ}\diamond, \mathit{NNP}\diamond, \mathit{NNS}\diamond, \mathit{RB}\diamond\}$, $T_\alpha \stackrel{\text{def}}{=} \{\mathit{eats}, \mathit{Bill}, \mathit{mushrooms}\}$, $T_\beta \stackrel{\text{def}}{=} \{\mathit{possibly}\}$, and $S \stackrel{\text{def}}{=} \text{S}$, where the elementary trees are shown below:



(*eats*) (*Bill*) (*mushrooms*) (*possibly*)

Then *eats* has contents $c(\mathit{eats}) = q_{\text{S}}, q_{\text{NP}\downarrow}, q_{\text{VP}}, q_{\text{NP}\downarrow}$, $c(\mathit{Bill}) = q_{\text{NP}}$, $c(\mathit{mushrooms}) = q_{\text{NP}}$, and $c(\mathit{possibly}) = q_{\text{VP}}$.

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

$$\begin{aligned}\delta \stackrel{\text{def}}{=} &\{(q_{A\downarrow}, \alpha^{(r(\alpha))}, c(\alpha)) \mid A{\downarrow} \in N{\downarrow}, \alpha \in T_\alpha, \text{rl}(\alpha) = A\}\\ \cup\ &\{(q_A, \beta^{(r(\beta))}, c(\beta)) \mid A \in N, \beta \in T_\beta, \text{rl}(\beta) = A\}\\ \cup\ &\{(q_A, \varepsilon^{(0)}) \mid A \in N\}\end{aligned}$$

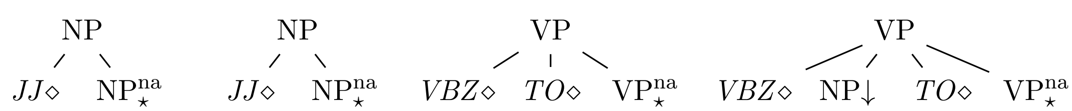
where 'rl' returns the root label of the tree.

[1] 1. Give the finite automaton $\mathcal{A}_{\mathcal{G}_1}$ associated with the example TAG $\mathcal{G}_1$.

Solution:

$$\begin{aligned}
Q &= \{q_{\mathrm{S}\downarrow}, q_{\mathrm{NP}\downarrow}, q_{\mathrm{S}}, q_{\mathrm{VP}}, q_{\mathrm{NP}}\}\ ,\\
\mathcal{F} &= \{\mathit{eats}^{(4)}, \mathit{Bill}^{(1)}, \mathit{mushrooms}^{(1)}, \mathit{possibly}^{(1)}, \varepsilon^{(0)}\}\ ,\\
\delta &= \{(q_{\mathrm{S}\downarrow}, \mathit{eats}^{(4)}, q_{\mathrm{S}}, q_{\mathrm{NP}\downarrow}, q_{\mathrm{VP}}, q_{\mathrm{NP}\downarrow}),\\
&\quad (q_{\mathrm{NP}\downarrow}, \mathit{Bill}^{(1)}, q_{\mathrm{NP}}),\\
&\quad (q_{\mathrm{NP}\downarrow}, \mathit{mushrooms}^{(1)}, q_{\mathrm{NP}}),\\
&\quad (q_{\mathrm{S}}, \varepsilon^{(0)}),\\
&\quad (q_{\mathrm{VP}}, \mathit{possibly}^{(1)}, q_{\mathrm{VP}}),\\
&\quad (q_{\mathrm{VP}}, \varepsilon^{(0)}),\\
&\quad (q_{\mathrm{NP}}, \varepsilon^{(0)})\}
\end{aligned}$$

[1] 2. Modify your automaton in order to also handle the trees *real*, *fake*, $\mathit{wants\_to}_0$, $\mathit{wants\_to}_1 \in T_\beta$ shown below, where $TO\diamond, JJ\diamond \in \Sigma$:



(*real*) (*fake*) ($\mathit{wants\_to}_0$) ($\mathit{wants\_to}_1$)

We call the resulting tree adjoining grammar $\mathcal{G}_2$.

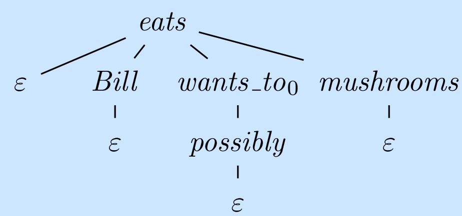
Solution: Add $\mathit{someone}^{(1)}$, $\mathit{real}^{(1)}$, $\mathit{fake}^{(1)}$, and $\mathit{wants\_to}^{(3)}$ to $\mathcal{F}$ and the rules

$$\begin{aligned}
&(q_{\mathrm{NP}}, \mathit{real}^{(1)}, q_{\mathrm{NP}})\\
&(q_{\mathrm{NP}}, \mathit{fake}^{(1)}, q_{\mathrm{NP}})\\
&(q_{\mathrm{VP}}, \mathit{wants_to}_0^{(1)}, q_{\mathrm{VP}})\\
&(q_{\mathrm{VP}}, \mathit{wants_to}_1^{(2)}, q_{\mathrm{VP}}, q_{\mathrm{NP}\downarrow})
\end{aligned}$$

to δ.

[1] 3. The intention that our finite automaton generates the *derivation* language $L_D(\mathcal{G}) \stackrel{\text{def}}{=} L(\mathcal{A}_{\mathcal{G}})$ of $\mathcal{G}$. Can you figure out what should be the derivation tree of '*Bill possibly wants to eat mushrooms*'?

Solution:



[2] 4. Give a PDL node formula φ_2 such that $L(\mathcal{A}_{\mathcal{G}_2}) = \{t \in T(\mathcal{F}) \mid t, \text{root} \models \varphi_2\}$.

Solution:

$$\begin{aligned}\varphi_1 \stackrel{\text{def}}{=} \varphi_{\mathrm{S}\downarrow} \wedge [\downarrow^*]\big(\quad & \mathit{eats} \implies \langle \downarrow; \mathsf{first}?; \varphi_{\mathrm{S}}?; \rightarrow; \varphi_{\mathrm{NP}\downarrow}?; \rightarrow; \varphi_{\mathrm{VP}}?; \rightarrow; \varphi_{\mathrm{NP}\downarrow}?\rangle \mathsf{last}\\ & \mathit{wants\_to}_0 \implies \langle \downarrow; \mathsf{first}?; \varphi_{\mathrm{VP}}?\rangle \mathsf{last}\\ & \mathit{wants\_to}_1 \implies \langle \downarrow; \mathsf{first}?; \varphi_{\mathrm{VP}}?; \rightarrow; \varphi_{\mathrm{NP}\downarrow}?\rangle \mathsf{last}\\ & \mathit{Bill} \implies \langle \downarrow; \mathsf{first}?; \varphi_{\mathrm{NP}}?\rangle \mathsf{last}\\ & \mathit{real} \implies \langle \downarrow; \mathsf{first}?; \varphi_{\mathrm{NP}}?\rangle \mathsf{last}\\ & \mathit{fake} \implies \langle \downarrow; \mathsf{first}?; \varphi_{\mathrm{NP}}?\rangle \mathsf{last}\\ & \mathit{mushrooms} \implies \langle \downarrow; \mathsf{first}?; \varphi_{\mathrm{NP}}?\rangle \mathsf{last}\\ & \mathit{possibly} \implies \langle \downarrow; \mathsf{first}?; \varphi_{\mathrm{VP}}?\rangle \mathsf{last}\\ & \varepsilon \implies \mathsf{leaf} \quad \big)\end{aligned}$$

where

$$\varphi_{\mathrm{S}\downarrow} \stackrel{\text{def}}{=} \mathit{eats} \qquad \varphi_{\mathrm{NP}\downarrow} \stackrel{\text{def}}{=} \mathit{Bill} \vee \mathit{mushrooms}$$

$$\varphi_{\mathrm{S}} \stackrel{\text{def}}{=} \varepsilon \qquad \varphi_{\mathrm{VP}} \stackrel{\text{def}}{=} \mathit{possibly} \vee \mathit{wants\_to}_0 \vee \mathit{wants\_to}_1 \vee \varepsilon \qquad \varphi_{\mathrm{NP}} \stackrel{\text{def}}{=} \mathit{real} \vee \mathit{fake} \vee \varepsilon$$

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 greatser than zero, we abuse notations and write $Q(\mathcal{X})$ for the alphabet of pairs $(q, x) \in Q \times \mathcal{X}$ with $\mathit{arity}(q, x) \stackrel{\text{def}}{=} \mathit{arity}(q) - 1$. 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 ≥ 1, $\mathcal{F}$ and $\mathcal{F}'$ are finite ranked alphabets, $I \subseteq Q_1$ is a set of root states of arity one, and Δ is a finite set of term rewriting rules of the form $q(f(x_1, \ldots, x_n), y_1, \ldots, y_p) \to e$ where $q \in Q_{1+p}$ for some $p \geq 0$, $f \in \mathcal{F}_n$ for some $n \in \mathbb{N}$, and $e \in T(\mathcal{F}' \cup Q(\mathcal{X}_n), \mathcal{Y}_p)$. 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: $t \xrightarrow{\text{IO}} t'$ if there exist a rule $q(f(x_1, \ldots, x_n), y_1, \ldots, y_p) \to e$ in Δ, a context $C \in C(\mathcal{F} \cup \mathcal{F}' \cup Q)$, and two substitutions $\sigma: \mathcal{X} \to T(\mathcal{F})$ and $\rho: \mathcal{Y} \to T(\mathcal{F}')$ such that $t = C[q(f(x_1, \ldots, x_n), y_1, \ldots, y_p)\sigma\rho]$ and $t' = C[e\sigma\rho]$. In other words, in inside-out rewriting, when applying a rewriting rule $q(f(x_1, \ldots, x_n), y_1, \ldots, y_p) \to e$, 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 $[\![\mathcal{M}]\!]_{\text{IO}}$ realised by $\mathcal{M}$ is defined through inside-out rewriting semantics:

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

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

$$\begin{aligned}
q(a(x_1)) &\to q'(x_1, \$, \$) & q'(\$, y_1, y_2) &\to f(y_1, y_1, y_2)\\
q'(a(x_1), y_1, y_2) &\to q'(x_1, a(y_1), a(y_2)) & q'(a(x_1), y_1, y_2) &\to q'(x_1, a(y_1), b(y_2))\\
q'(a(x_1), y_1, y_2) &\to q'(x_1, b(y_1), a(y_2)) & q'(a(x_1), y_1, y_2) &\to q'(x_1, b(y_1), b(y_2))
\end{aligned}$$

Then we have for instance the following derivation:

$$\begin{aligned}
q(a(a(a(\$)))) &\xrightarrow{\text{IO}} q'(a(a(\$)), \$, \$)\\
&\xrightarrow{\text{IO}} q'(a(\$), b(\$), b(\$))\\
&\xrightarrow{\text{IO}} q'(\$, a(b(\$)), b(b(\$)))\\
&\xrightarrow{\text{IO}} f(a(b(\$)), a(b(\$)), b(b(\$)))
\end{aligned}$$

showing that $(\ a(a(a(\$))),\ f(a(b(\$)), a(b(\$)), b(b(\$)))\ ) \in [\![\mathcal{M}]\!]$.

Exercise 2 (Monadic trees)**.** An NMTT $\mathcal{M}$ is called *linear* and *non-deleting* if, in every rule $q(f(x_1, \ldots, x_n), y_1, \ldots, y_p) \to e$ in Δ, 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 $\mathcal{F}' \stackrel{\text{def}}{=} \{a^{(1)}, b^{(1)}, \$^{(0)}\}$. 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 $C = a(b(a(a(\square))))$, then $C^R = a(a(b(a(\square))))$. Formally, let $n \in \mathbb{N}$ be such that $\operatorname{dom} C = \{0^m \mid m \leq n\}$; then $C(0^n) = \square$ and $C(0^m) \in \{a, b\}$ for $m < n$. Then C^R is defined by $\operatorname{dom} C^R \stackrel{\text{def}}{=} \operatorname{dom} C$, $C^R(0^n) \stackrel{\text{def}}{=} \square$, and $C^R(0^m) \stackrel{\text{def}}{=} C^R(0^{n-m})$ for all $m < n$.

[2] 1. Give a linear and non-deleting NMTT $\mathcal{M}$ from $\mathcal{F}'$ to $\mathcal{F}'$ such that $[\![\mathcal{M}]\!]_{\text{IO}} = \{(C[\$], C[C^R[\$]]) \mid C \in C(\mathcal{F}')\}$. In terms of words over $\{a, b\}^*$, this transducer maps w to the palindrome ww^R. Is $[\![\mathcal{M}]\!]_{\text{IO}}(T(\mathcal{F}))$ a recognisable tree language?

Solution: Let $\mathcal{M} \stackrel{\text{def}}{=} (Q, \mathcal{F}', \mathcal{F}', \Delta, I)$ where $Q \stackrel{\text{def}}{=} \{q_i^{(1)}, q^{(2)}\}$, $I \stackrel{\text{def}}{=} \{q_i\}$, and Δ is the set of rules

$$\begin{array}{ccc} q_i(\$) \to \$ & q_i(a(x_1)) \to a(q(x_1, a(\$))) & q_i(b(x_1)) \to b(q(x_1, b(\$))) \\ q(\$, y_1) \to y_1 & q(a(x_1), y_1) \to a(q(x_1, a(y_1))) & q(b(x_1), y_1) \to b(q(x_1, b(y_1))) \;. \end{array}$$

We leave the proof of correctness to the reader.

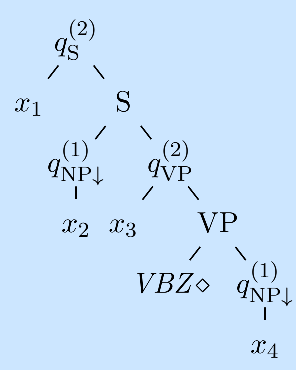
This macro tree transducer is deterministic, and complete. Because a monadic tree language over $\mathcal{F}'$ is recognisable if and only if the corresponding word language over $\{a, b\}$ is recognisable, $[\![\mathcal{M}]\!]_{\text{IO}}(T(\mathcal{F}))$ is not a recognisable tree language. In turn, 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.

[3] 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 $\text{dom}([\![\mathcal{M}_2]\!]_{\text{IO}}) = L_D(\mathcal{G}_2)$ and $[\![\mathcal{M}_2]\!]_{\text{IO}}(T(\mathcal{F})) = L_T(\mathcal{G}_2)$.

Solution: We set $\mathcal{F}' \stackrel{\text{def}}{=} N \uplus \Sigma$, $Q \stackrel{\text{def}}{=} \{q_{\text{S}\downarrow}^{(1)}, q_{\text{S}}^{(2)}, q_{\text{NP}\downarrow}^{(1)}, q_{\text{NP}}^{(2)}, q_{\text{VP}}^{(2)}\}$, $I \stackrel{\text{def}}{=} \{q_{\text{S}}^{(1)}\}$, and Δ:

$q_{\text{S}\downarrow}^{(1)}(eats(x_1, x_2, x_3, x_4)) \to$



$$q^{(2)}_{\mathrm{S}}(\varepsilon, y_1) \to y_1$$

$$q^{(1)}_{\mathrm{NP}\downarrow}(Bill(x_1)) \to q^{(2)}_{\mathrm{NP}}\big(x_1, \mathrm{NP}(NNP\diamond)\big)$$

$$q^{(1)}_{\mathrm{NP}\downarrow}(mushrooms(x_1)) \to q^{(2)}_{\mathrm{NP}}\big(x_1, \mathrm{NP}(NNS\diamond)\big)$$

$$q^{(2)}_{\mathrm{NP}}(real(x_1), y_1) \to q^{(2)}_{\mathrm{NP}}\big(x_1, \mathrm{NP}(JJ\diamond, y_1)\big)$$

$$q^{(2)}_{\mathrm{NP}}(fake(x_1), y_1) \to q^{(2)}_{\mathrm{NP}}\big(x_1, \mathrm{NP}(JJ\diamond, y_1)\big)$$

$$q^{(2)}_{\mathrm{NP}}(\varepsilon, y_1) \to y_1$$

$$q^{(2)}_{\mathrm{VP}}(possibly(x_1), y_1) \to q^{(2)}_{\mathrm{VP}}\big(x_1, \mathrm{VP}(RB\diamond, y_1)\big)$$

$$q^{(2)}_{\mathrm{VP}}(wants\_to_0(x_1), y_1) \to q^{(2)}_{\mathrm{VP}}\big(x_1, \mathrm{VP}(VBZ\diamond, TO\diamond, y_1)\big)$$

$$q^{(2)}_{\mathrm{VP}}(wants\_to_1(x_1, x_2), y_1) \to q^{(2)}_{\mathrm{VP}}\big(x_1, \mathrm{VP}(VBZ\diamond, q_{\mathrm{NP}\downarrow}(x_2), TO\diamond, y_1)\big)$$

$$q^{(2)}_{\mathrm{VP}}(\varepsilon, y_1) \to y_1$$

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.

[5] 1. Show that $L \stackrel{\text{def}}{=} [\![\mathcal{M}]\!]_{\text{IO}}(L(\mathcal{A})) = \{t' \in T(\mathcal{F}') \mid \exists t \in L(\mathcal{A}) \,.\, (t, t') \in [\![\mathcal{M}]\!]_{\text{IO}}\}$ 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 $L_{\text{IO}}(\mathcal{G}) = L$.

Solution: Let

$$N \stackrel{\text{def}}{=} (Q \times Q') \uplus \{S\}$$

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

$$\begin{aligned} R \stackrel{\text{def}}{=} & \{S \to (q, q')^{(0)} \mid q \in I, q' \in I'\} \\ & \cup \{(q, q')^{(p)}(y_1, \ldots, y_p) \to e[q_i'/x_i]_i \mid \exists n \,.\, \exists f \in \mathcal{F}_n \,.\, q^{(1+p)}(f(x_1, \ldots, x_n), y_1, \ldots, y_n) \to e \in \Delta \\ & \qquad \text{and } (q', f, q_1', \ldots, q_n') \in \delta\} \end{aligned}$$

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 $t' \in T(\mathcal{F}')$, $e \stackrel{\text{IO}}{\Longrightarrow}{}^{k}_{\mathcal{G}} t'$ if and only if $\exists t_1, \ldots, t_n \in T(\mathcal{F})$ such that $e[t_i/q_i']_{1 \le i \le n} \stackrel{\text{IO}}{\Longrightarrow}{}^{k}_{\mathcal{M}} t'$ and for all $1 \le i \le n$, $t_i \stackrel{\delta_B}{\Longrightarrow}{}^{*}_{\mathcal{A}} q_i'$.

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 $e \stackrel{\text{IO}}{\Longrightarrow}{}^{k}_{\mathcal{G}} t'$.

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

$$e = f(e_1, \ldots, e_m) \stackrel{\text{IO}}{\Longrightarrow}{}^{k}_{\mathcal{G}} f(t_1', \ldots, t_m') = t'$$

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

$$e_j \stackrel{\text{IO}}{\Longrightarrow}{}^{k_j}_{\mathcal{G}} t_j'$$

and

$$k = \sum_{1 \le j \le m} k_j \,.$$

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 $t_{j,1}, \ldots, t_{j,n_j} \in T(\mathcal{F})$ such that

$$e_j[t_{j,i}/q'_{j,i}]_{1 \leq i \leq n_j} \overset{\text{IO}}{\Longrightarrow}{}^{k_j}_{\mathcal{M}}\ t'_j$$

and

$$t_{j,i} \overset{\delta_{\mathcal{B}}}{\Longrightarrow}{}^{*}_{\mathcal{A}}\ q'_{j,i}$$

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

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

as desired.

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

$$\begin{aligned} e = (q, q')^{(p)}(e_1, \ldots, e_p) &\overset{\text{IO}}{\Longrightarrow}{}^{k'}_{\mathcal{G}}\ (q, q')^{(p)}(t'_1, \ldots, t'_p) \\ &\overset{\text{IO}}{\Longrightarrow}_{\mathcal{G}}\ e'[q'_i/x_i]_{1 \leq i \leq m}[t'_j/y_j]_{1 \leq j \leq p} \\ &\overset{\text{IO}}{\Longrightarrow}{}^{k''}_{\mathcal{G}}\ t' \end{aligned}$$

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

$$e_j \overset{\text{IO}}{\Longrightarrow}{}^{k_j}_{\mathcal{G}}\ t'_j$$

and $k' = \sum_{1 \leq j \leq m} k_j$ and $k = 1 + k' + k''$; also $N(e) = \{(q, q')\} \cup \bigcup_{1 \leq j \leq p} N(e_j)$ 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 $f \in \mathcal{F}_m$ such that there are rules $q^{(1+p)}(f(x_1, \ldots, x_m), y_1, \ldots, y_p) \to e' \in \Delta$ and $(q', f, q'_1, \ldots, q'_m) \in \delta$.

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

$$e_j[t_{j,i}/q'_{j,i}]_{1 \leq i \leq n_j} \overset{\text{IO}}{\Longrightarrow}{}^{k_j}_{\mathcal{M}}\ t'_j$$

and

$$t_{j,i} \overset{\delta_{\mathcal{B}}}{\Longrightarrow}{}^{*}_{\mathcal{A}}\ q'_{j,i}$$

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

$$e'[t'_j/y_j]_{1\leq j\leq p}[t_i/x_i]_{1\leq i\leq m} \overset{\text{IO}}{\Longrightarrow}{}^{k''}_{\mathcal{M}}\, t'$$

and

$$t_i \overset{\delta_B}{\Longrightarrow}{}^{*}_{\mathcal{A}}\, q'_i$$

for all $1 \leq i \leq m$. Note that, because $(q', f, q'_1, \ldots, q'_m) \in \delta$, the latter imply

$$f(t_1,\ldots,t_m) \overset{\delta_B}{\Longrightarrow}{}^{*}_{\mathcal{A}}\, f(q'_1,\ldots,q'_m) \overset{\delta_B}{\Longrightarrow}_{\mathcal{A}}\, q' \;.$$

Thus, in $\mathcal{M}$, we have the rewrite

$$\begin{aligned}
&e[f(t_1,\ldots,t_m)/q][t'_{j,i}/q'_{j,i}]_{1\leq j\leq m,1\leq i\leq n_i}\\
&= q^{(1+p)}(f(t_1,\ldots,t_m), e_1[t'_{1,i}/q'_{1,i}]_{1\leq i\leq n_1},\ldots,e_m[t'_{m,i}/q'_{m,i}]_{1\leq i\leq n_m})\\
&= q^{(1+p)}(f(x_1,\ldots,x_m), e_1[t'_{1,i}/q'_{1,i}]_{1\leq i\leq n_1},\ldots,e_m[t'_{m,i}/q'_{m,i}]_{1\leq i\leq n_m})[t_1/x_1,\ldots,t_m/x_m]\\
&\overset{\text{IO}}{\Longrightarrow}{}^{k'}_{\mathcal{M}}\, q^{(1+p)}(f(x_1,\ldots,x_m), t'_1,\ldots,t'_p)[t_1/x_1,\ldots,t_m/x_m]\\
&\overset{\text{IO}}{\Longrightarrow}_{\mathcal{M}}\, e'[t_i/x_i]_{1\leq i\leq m}[t'_j/y_j]_{1\leq j\leq p}\\
&\overset{\text{IO}}{\Longrightarrow}{}^{k''}_{\mathcal{M}}\, t'
\end{aligned}$$

as desired.

2 Scope Ambiguities and Propositional Attitudes

Exercise 5. One considers the two following signatures:

(Σ_{ABS})

$$\begin{aligned}
\textsc{suzy} &: NP\\
\textsc{bill} &: NP\\
\textsc{mushroom} &: N\\
\textsc{a} &: N \to (NP \to S) \to S\\
\textsc{a}_{inf} &: N \to (NP \to S_{inf}) \to S_{inf}\\
\textsc{eat} &: NP \to NP \to S_{inf}\\
\textsc{to} &: (NP \to S_{inf}) \to VP\\
\textsc{want} &: VP \to NP \to S
\end{aligned}$$

$$(\Sigma_{\text{S-FORM}}) \qquad \begin{aligned} \boldsymbol{Suzy} &: \mathit{string} \\ \boldsymbol{Bill} &: \mathit{string} \\ \boldsymbol{mushroom} &: \mathit{string} \\ \boldsymbol{a} &: \mathit{string} \\ \boldsymbol{eat} &: \mathit{string} \\ \boldsymbol{to} &: \mathit{string} \\ \boldsymbol{wants} &: \mathit{string} \end{aligned}$$

where, as usual, *string* is defined to be $o \to o$ for some atomic type o.

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

$$(\mathcal{L}_{\text{SYNT}}) \qquad \begin{aligned} NP &:= \mathit{string} \\ N &:= \mathit{string} \\ S &:= \mathit{string} \\ S_{\mathit{inf}} &:= \mathit{string} \\ VP &:= \mathit{string} \\ \text{SUZY} &:= \boldsymbol{Suzy} \\ \text{BILL} &:= \boldsymbol{Bill} \\ \text{MUSHROOM} &:= \boldsymbol{mushroom} \\ \text{A} &:= \lambda xy.\, y\,(\boldsymbol{a} + x) \\ \text{A}_{\mathit{inf}} &:= \lambda xy.\, y\,(\boldsymbol{a} + x) \\ \text{EAT} &:= \lambda xy.\, y + \boldsymbol{eat} + x \\ \text{TO} &:= \lambda x.\, \boldsymbol{to} + (x\,\epsilon) \\ \text{WANT} &:= \lambda xy.\, y + \boldsymbol{wants} + x \end{aligned}$$

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

[1] 1. Give two different terms, say t_0 and t_1, such that:

$$\mathcal{L}_{\text{SYNT}}(t_0) = \mathcal{L}_{\text{SYNT}}(t_1) = \boldsymbol{Bill} + \boldsymbol{wants} + \boldsymbol{to} + \boldsymbol{eat} + \boldsymbol{a} + \boldsymbol{mushroom}$$

Solution:

$$\begin{aligned} t_0 &= \text{WANT}\,(\text{TO}\,(\lambda x.\, \text{A}_{\mathit{inf}}\, \text{MUSHROOM}\,(\lambda y.\, \text{EAT}\, y\, x)))\, \text{BILL} \\ t_1 &= \text{A}\, \text{MUSHROOM}\,(\lambda y.\, \text{WANT}\,(\text{TO}\,(\lambda x.\, \text{EAT}\, y\, x))\, \text{BILL}) \end{aligned}$$

Exercise 6. One considers a third signature :

$$(\Sigma_{\text{L-FORM}}) \qquad \begin{aligned} \textbf{suzy} &: \mathsf{ind} \\ \textbf{bill} &: \mathsf{ind} \\ \textbf{mushroom} &: \mathsf{ind} \to \mathsf{prop} \\ \textbf{eat} &: \mathsf{ind} \to \mathsf{ind} \to \mathsf{prop} \\ \textbf{want} &: \mathsf{ind} \to \mathsf{prop} \to \mathsf{prop} \end{aligned}$$

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

$$(\mathcal{L}_{\text{SEM}}) \qquad \begin{aligned} NP &:= \mathsf{ind} \\ N &:= \mathsf{ind} \to \mathsf{prop} \\ S &:= \mathsf{prop} \\ S_{inf} &:= \mathsf{prop} \\ VP &:= \mathsf{ind} \to \mathsf{prop} \end{aligned}$$

$$\begin{aligned} \text{SUZY} &:= \textbf{suzy} \\ \text{BILL} &:= \textbf{bill} \\ \text{MUSHROOM} &:= \textbf{mushroom} \\ \text{A} &:= \lambda xy.\, \exists z.\, (x\, z) \wedge (y\, z) \\ \text{A}_{inf} &:= \lambda xy.\, \exists z.\, (x\, z) \wedge (y\, z) \\ \text{EAT} &:= \lambda xy.\, \textbf{eat}\, y\, x \\ \text{TO} &:= \lambda x.\, x \\ \text{WANT} &:= \lambda xy.\, \textbf{want}\, y\, (x\, y) \end{aligned}$$

[1] 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:

$$\begin{aligned} \mathcal{L}_{\text{SEM}}(t_0) &= \textbf{want}\,\textbf{bill}\,(\exists z.\, (\textbf{mushroom}\, z) \wedge (\textbf{eat}\,\textbf{bill}\, z)) \\ \mathcal{L}_{\text{SEM}}(t_1) &= \exists z.\, (\textbf{mushroom}\, z) \wedge (\textbf{want}\,\textbf{bill}\,(\textbf{eat}\,\textbf{bill}\, z)) \end{aligned}$$

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

$$\begin{aligned} (\Sigma_{\text{ABS}}) &\quad \text{WANT2} : \; NP \to VP \to NP \to S \\ (\mathcal{L}_{\text{SYNT}}) &\quad \text{WANT2} := \lambda xyz.\, z + \textit{\textbf{wants}} + x + y \end{aligned}$$

[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.*

Solution:

$$(\mathcal{L}_{\text{SEM}}) \quad \textsc{want2} := \lambda xyz.\, \textbf{want}\, z\, (y\, x)$$

Exercise 8. One extends Σ_{ABS} as follows:

$$(\Sigma_{\text{ABS}}) \quad \begin{aligned} \textsc{everyone} &: (NP \to S) \to S \\ \textsc{think} &: S \to NP \to S \end{aligned}$$

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

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

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

Solution:

$$(\Sigma_{\text{S-FORM}}) \quad \begin{aligned} &\textbf{\textit{everyone}} : \textit{string} \\ &\textbf{\textit{thinks}} : \textit{string} \end{aligned}$$

$$(\mathcal{L}_{\text{SYNT}}) \quad \begin{aligned} &\textsc{everyone} := \lambda x.\, x\, \textbf{\textit{everyone}} \\ &\textsc{think} := \lambda xy.\, y + \textbf{\textit{thinks}} + x \end{aligned}$$

$$(\Sigma_{\text{L-FORM}}) \quad \begin{aligned} &\textbf{human} : \mathsf{ind} \to \mathsf{prop} \\ &\textbf{think} : \mathsf{ind} \to \mathsf{prop} \to \mathsf{prop} \end{aligned}$$

$$(\mathcal{L}_{\text{SEM}}) \quad \begin{aligned} &\textsc{everyone} := \lambda x.\, \forall y.\, (\textbf{human}\, y) \to (x\, y) \\ &\textsc{think} := \lambda xy.\, \textbf{think}\, y\, x \end{aligned}$$

[2] 2. Give the several λ-terms that correspond to the different parsings of sentence (1).

Solution: There are four such terms:

$$\textsc{everyone}\,(\lambda x.\, \textsc{think}\,(\textsc{want}\,(\textsc{to}\,(\lambda z.\, \textsc{a}_{\mathit{inf}}\, \textsc{mushroom}\,(\lambda y.\, \textsc{eat}\, y\, z)))\, \textsc{bill})\, x)$$
$$\textsc{everyone}\,(\lambda x.\, \textsc{think}\,(\textsc{a}\, \textsc{mushroom}\,(\lambda y.\, \textsc{want}\,(\textsc{to}\,(\lambda z.\, \textsc{eat}\, y\, z))\, \textsc{bill}))\, x)$$
$$\textsc{everyone}\,(\lambda x.\, \textsc{a}\, \textsc{mushroom}\,(\lambda y.\, \textsc{think}\,(\textsc{want}\,(\textsc{to}\,(\lambda z.\, \textsc{eat}\, y\, z))\, \textsc{bill})\, x))$$
$$\textsc{a}\, \textsc{mushroom}\,(\lambda y.\, \textsc{everyone}\,(\lambda x.\, \textsc{think}\,(\textsc{want}\,(\textsc{to}\,(\lambda z.\, \textsc{eat}\, y\, z))\, \textsc{bill})\, x))$$