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 Model-Theoretic Syntax

Exercise 1 (Propositional Dynamic Logic)**.** Recall that the syntax of PDL can be seen as follows. Let A be a countable set of atomic predicates. Then PDL formulæ can be defined by the abstract syntax:

$$\begin{aligned}
\varphi &::= a \mid \top \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\pi\rangle\varphi && \text{(node formulæ)}\\
\alpha &::= \varphi? \mid \downarrow \mid \uparrow \mid \rightarrow \mid \leftarrow && \text{(atomic paths)}\\
\pi &::= \alpha \mid \pi + \pi \mid \pi;\pi \mid \pi^* && \text{(path formulæ)}
\end{aligned}$$

where a ranges over A. Put differently, path formulæ are built as *rational languages* over an alphabet of atomic paths.

The *semantics* of a node formula on a tree structure $\mathfrak{M} = \langle W, \downarrow, \rightarrow, (P_a)_{a\in A}\rangle$ is a set of tree nodes $[\![\varphi]\!] = \{w \in W \mid \mathfrak{M}, w \models \varphi\}$, while the semantics of a path formula is a binary relation over W:

$$\begin{aligned}
[\![a]\!] &\stackrel{\text{def}}{=} \{w \in W \mid P_a(w)\} & [\![\downarrow]\!] &\stackrel{\text{def}}{=} \downarrow & [\![\pi_1 + \pi_2]\!] &\stackrel{\text{def}}{=} [\![\pi_1]\!] \cup [\![\pi_2]\!]\\
[\![\top]\!] &\stackrel{\text{def}}{=} W & [\![\rightarrow]\!] &\stackrel{\text{def}}{=} \rightarrow & [\![\pi_1;\pi_2]\!] &\stackrel{\text{def}}{=} [\![\pi_1]\!] \fatsemi [\![\pi_2]\!]\\
[\![\neg\varphi]\!] &\stackrel{\text{def}}{=} W \backslash [\![\varphi]\!] & [\![\uparrow]\!] &\stackrel{\text{def}}{=} (\downarrow)^{-1} & [\![\pi^*]\!] &\stackrel{\text{def}}{=} [\![\pi]\!]^\star\\
[\![\varphi_1 \vee \varphi_2]\!] &\stackrel{\text{def}}{=} [\![\varphi_1]\!] \cup [\![\varphi_2]\!] & [\![\leftarrow]\!] &\stackrel{\text{def}}{=} (\rightarrow)^{-1} & &\\
[\![\langle\pi\rangle\varphi]\!] &\stackrel{\text{def}}{=} [\![\pi]\!]^{-1}([\![\varphi]\!]) & [\![\varphi?]\!] &\stackrel{\text{def}}{=} \{(w,w) \in W \times W \mid w \in [\![\varphi]\!]\}\ , & &
\end{aligned}$$

where '$\fatsemi$' denotes relational composition: for two binary relations R and R' over W, $R \fatsemi R' = \{(w, w'') \in W \times W \mid \exists w' \in W, (w, w') \in R \wedge (w', w'') \in R'\}$.

[2] 1. Prove the following equivalences:

$$\begin{aligned}
\langle\pi_1;\pi_2\rangle\varphi &\equiv \langle\pi_1\rangle\langle\pi_2\rangle\varphi\\
\langle\pi_1+\pi_2\rangle\varphi &\equiv (\langle\pi_1\rangle\varphi) \vee (\langle\pi_2\rangle\varphi)\\
\langle\pi^*\rangle\varphi &\equiv \varphi \vee \langle\pi;\pi^*\rangle\varphi\\
\langle\varphi_1?\rangle\varphi_2 &\equiv \varphi_1 \wedge \varphi_2\ .
\end{aligned}$$

Solution: For any tree structure:

$$
\begin{aligned}
[\![\langle \pi_1 ; \pi_2 \rangle \varphi]\!] &= [\![\pi_1 ; \pi_2]\!]^{-1}([\![\varphi]\!]) \\
&= \big([\![\pi_1]\!] \fatsemi [\![\pi_2]\!]\big)^{-1}([\![\varphi]\!]) \\
&= \big([\![\pi_2]\!]^{-1} \fatsemi [\![\pi_1]\!]^{-1}\big)([\![\varphi]\!]) && (\text{⚠}) \\
&= [\![\pi_1]\!]^{-1}\big([\![\pi_2]\!]^{-1}([\![\varphi]\!])\big) && (\text{⚠}) \\
&= [\![\langle \pi_1 \rangle \langle \pi_2 \rangle \varphi]\!] \\
[\![\langle \pi_1 + \pi_2 \rangle \varphi]\!] &= [\![\pi_1 + \pi_2]\!]^{-1}([\![\varphi]\!]) \\
&= \big([\![\pi_1]\!] \cup [\![\pi_2]\!]\big)^{-1}([\![\varphi]\!]) \\
&= \big([\![\pi_1]\!]^{-1} \cup [\![\pi_2]\!]^{-1}\big)([\![\varphi]\!]) \\
&= [\![\pi_1]\!]^{-1}([\![\varphi]\!]) \cup [\![\pi_2]\!]^{-1}([\![\varphi]\!]) \\
&= [\![\langle \pi_1 \rangle \varphi \vee \langle \pi_2 \rangle \varphi]\!] \\
[\![\langle \pi^* \rangle \varphi]\!] &= [\![\pi^*]\!]^{-1}([\![\varphi]\!]) \\
&= ([\![\pi]\!]^*)^{-1}([\![\varphi]\!]) \\
&= [\![\pi]\!]^0([\![\varphi]\!]) \cup ([\![\pi]\!]^+)^{-1}([\![\varphi]\!]) \\
&= [\![\varphi]\!] \cup ([\![\pi]\!] \fatsemi [\![\pi]\!]^*)^{-1}([\![\varphi]\!]) \\
&= [\![\varphi \vee \langle \pi ; \pi^* \rangle \varphi]\!] \\
[\![\langle \varphi_1 ? \rangle \varphi_2]\!] &= [\![\varphi_1 ?]\!]^{-1}([\![\varphi_2]\!]) \\
&= \{(w, w) \mid w \in [\![\varphi_1]\!]\}^{-1}([\![\varphi_2]\!]) \\
&= \{(w, w) \mid w \in [\![\varphi_1]\!]\}([\![\varphi_2]\!]) \\
&= [\![\varphi_1]\!] \cap [\![\varphi_2]\!] \\
&= [\![\varphi_1 \wedge \varphi_2]\!] \,.
\end{aligned}
$$

[2] 2. Let us assume that we distinguish three disjoint subsets of labels: nonterminal labels in $N \subseteq A$, part-of-speech labels $\Theta \subseteq A$, and an open lexicon $L \subseteq A$. For example, in the tree in Figure 1 below, we have {S, NP, VP, PP} $\subseteq N$, {PRP, VBD, DT, NN, IN} $\subseteq \Theta$, and {*He, hurled, the, ball, into, basket*} $\subseteq L$.

Give a PDL formula ensuring that its models are labelled consistently with this style of constituent analysis.

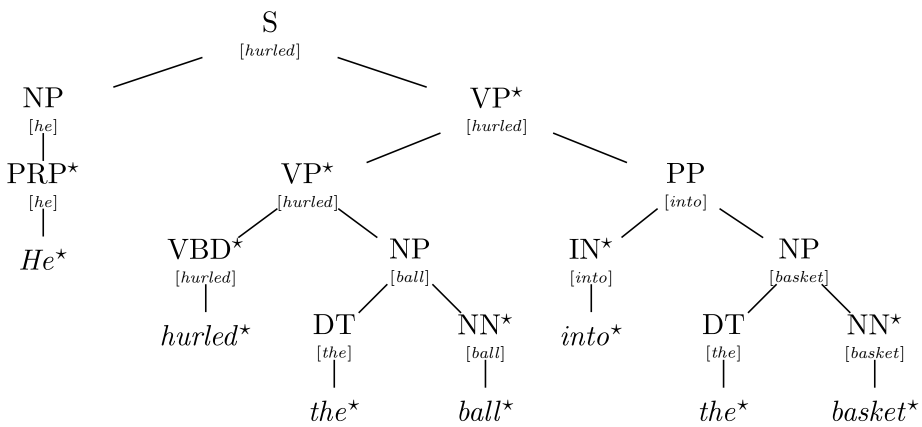

Figure 1: Example of a constituent tree. The head children are starred. The lexical heads of internal nodes are indicated between brackets.

Solution: Consider the following node formula to be evaluated at the root:

$$
\begin{aligned}
&\mathsf{root} \wedge S && \text{(the root is labelled by `S')}\\
&\wedge [\downarrow^*; \mathsf{internal}?; (\neg\langle\downarrow\rangle\mathsf{leaf})?] \bigvee_{A\in N} \left(A \wedge \bigwedge_{p\in(N\cup\Theta\cup L)\setminus\{A\}} \neg p \right) && \text{(grammatical categories)}\\
&\wedge [\downarrow^*; (\langle\downarrow\rangle\mathsf{leaf})?] \bigvee_{\theta\in\Theta} \left(\theta \wedge \bigwedge_{p\in(N\cup\Theta\cup L)\setminus\{\theta\}} \neg p \wedge [\downarrow](\mathsf{first}\wedge\mathsf{last}) \right) && \text{(part-of-speech categories)}\\
&\wedge [\downarrow^*; \mathsf{leaf}?] \bigwedge_{p\in N\cup\Theta} \neg p\\
&\wedge [\downarrow^*] \left(\bigvee_{a\in L} a \implies \mathsf{leaf} \wedge \bigwedge_{b\in L\setminus\{a\}} \neg b \right) && \text{(lexicon elements)}
\end{aligned}
$$

The formula ensures that exactly one of the propositions from $N \cup \Theta$ labels every internal node, and none labels any leaves. Furthermore, Θ-labelled nodes are exactly the ones with a leaf child, and that child must be unique. Finally, L-labelled nodes must be leaves (but not all leaves must have a label in L, so that the lexicon is open).

[3] 3. Recall from the lecture notes that a *head percolation* function $h\colon N \to \{l, r\} \times (N \uplus \Theta)^*$ provides for a given parent label $A \in N$ a pair $(d, X_1 \cdots X_n)$ consisting of a direction d and a list of potential head labels $X_1 \cdots X_n$. The intended semantics of such a function is to identify the head child of an A-labelled node $w \in W$. The direction indicates whether we should process the list of children of w left-to-right (l) or right-to-left (r). If X_1 appears among the children of w, then its leftmost (in case of l, and rightmost

in case of r) occurrence is the head child of w. Otherwise, if X_2 appears among the children, then its leftmost (in case of l, and rightmost in case of r) occurrence is the head child of w... If none of $X_1, \ldots, X_n$ appears among the children of w, then its leftmost (in case of l, and rightmost in case of r) child is considered as its head. For instance, the function

$$\begin{aligned} h(\mathrm{S}) &= (r, \mathrm{TO\ IN\ VP\ S\ SBAR} \cdots) \\ h(\mathrm{VP}) &= (l, \mathrm{VBD\ VBN\ VBZ\ VB\ VBG\ VP} \cdots) \\ h(\mathrm{NP}) &= (r, \mathrm{NN\ NNP\ NNS\ NNPS\ JJR\ CD} \cdots) \\ h(\mathrm{PP}) &= (l, \mathrm{IN\ TO\ VBG\ VBN} \cdots) \end{aligned}$$

would result in the starred head children in Figure 1.

Given a head percolation function h, provide a PDL path formula π_h s.t. $(w, w') \in [\![\pi_h]\!]$ iff w is the parent of w' and w' is the head child of w. Your formula should also consider the case where w is labelled by a part-of-speech tag in Θ.

Solution: Define the path formula

$$\pi_h \stackrel{\text{def}}{=} \left(\sum_{A \in N} A?; \pi_{h(A)} \right) + \left(\sum_{\theta \in \Theta} \theta?; \downarrow \right)$$

where we move to the first child in case of l and to the last child in case of r

$$\begin{aligned} \pi_{l, X_1 \cdots X_n} &\stackrel{\text{def}}{=} \downarrow; \mathsf{first}?; \pi'_{l, X_1 \cdots X_n} \\ \pi_{r, X_1 \cdots X_n} &\stackrel{\text{def}}{=} \downarrow; \mathsf{last}?; \pi'_{r, X_1 \cdots X_n} \end{aligned}$$

and $\pi'_{l, X_1 \cdots X_n}$ is defined by induction over n:

$$\begin{aligned} \pi'_{l, \varepsilon} &\stackrel{\text{def}}{=} \top? \\ \pi'_{l, X_1 \cdot X_2 \cdots X_n} &\stackrel{\text{def}}{=} ((\neg X_1?; \rightarrow)^*; X_1) + \big((\neg \langle \rightarrow^* \rangle X_1)?; \pi'_{l, X_2 \cdots X_n}\big) \end{aligned}$$

and $\pi'_{r, X_1 \cdots X_n}$ is defined similarly.

[1] 4. Provide a PDL path formula π_{lex} that holds between a node and its lexical head. The formula should allow to recover the lexical heads as indicated between brackets in Figure 1.

Solution: Define the formula by

$$\pi_{\text{lex}} \stackrel{\text{def}}{=} \pi_h^*; \mathsf{leaf}?$$

[1] 5. Consider $L(\varphi)$ the set of trees that satisfy the PDL node formula φ at their root. Assuming A to be finite, justify why $\text{yield}(L(\varphi))$ is a context-free word language.

Solution: As seen in class, for a PDL formula φ, one can construct a 2ATA $\mathcal{A}_\varphi$ that recognises the 'completion' of the first-child next-sibling encoding $\overline{\mathsf{fcns}(L(\varphi))}$ over $\{\#\} \cup 2^A \times \{0,1\}$, and by a theorem by Vardi this 2ATA can be converted into an equivalent nondeterministic tree automaton. Thus this encoding $\overline{\mathsf{fcns}(L(\varphi))}$ is a regular tree language.

⚠ In general for an unranked tree language L, $\text{yield}(\mathsf{fcns}(L))$ might be different from $\text{yield}(L)$; furthermore, the completion means that the yield of $\overline{\mathsf{fcns}(L(\varphi))}$ is a sequence of '#' symbols.

We can however apply a linear bottom-up tree transduction τ (which preserves regularity) to recover the yield of the original trees in $L(\varphi)$. Pick two new symbols f, g not in 2^A. Namely, the transducer has two states $q_\#$ and q and rules

$$\begin{aligned}
\#^{(0)} &\to q_\#(\#^{(0)})\\
\#^{(2)}(q(x_1), q_\#(x_2)) &\to q(x_1)\\
(a,b)^{(2)}(q_\#(x_1), q_\#(x_2)) &\to q(a^{(0)})\\
(a,b)^{(2)}(q_\#(x_1), q(x_2)) &\to q(f^{(2)}(a^{(0)}, x_2))\\
(a,b)^{(2)}(q(x_1), q_\#(x_2)) &\to q(g^{(1)}(x_1))\\
(a,b)^{(2)}(q(x_1), q(x_2)) &\to q(f^{(2)}(x_1, x_2))
\end{aligned}$$

for all $a \in 2^A$ and $b \in \{0,1\}$. The result is to replace left children labelled by '#' by their parent label—which were leaf nodes in the trees in $L(\varphi)$ before the encoding—and to remove right children labelled by '#' entirely. Thus the image $\tau(\overline{\mathsf{fcns}(L(\varphi))})$ by this linear bottom-up transduction is also a regular tree language, and therefore $\text{yield}(\tau(\overline{\mathsf{fcns}(L(\varphi))}))$ is a context-free word language. But this is exactly $\text{yield}(L(\varphi))$.

Exercise 2 (Relational PDL)**.** We extend the syntax of PDL to allow for 'relational paths'. To simplify matters, we shall only consider binary relational paths. Define $\varepsilon \stackrel{\text{def}}{=} \top?$. Then binary relational paths are defined by the following abstract syntax:

$$\begin{aligned}
\beta &::= \alpha : \varepsilon \mid \varepsilon : \alpha && \text{(atomic relations)}\\
\rho &::= \beta \mid \rho + \rho \mid \rho ; \rho \mid \rho^* && \text{(relational paths)}
\end{aligned}$$

and adding the construction $\langle \rho \rangle$ to the syntax of node formulæ. Put differently, relational paths are constructed as *rational relations* over atomic paths. The semantics of a relational path on a tree structure $\mathfrak{M} = \langle W, \downarrow, \rightarrow, (P_a)_{a \in A} \rangle$ is a 4-ary relation in W^4, i.e. a binary relation on paths, defined by:

$$[\![\alpha : \varepsilon]\!] \stackrel{\text{def}}{=} \{(w, w', w'', w'') \mid (w, w') \in [\![\alpha]\!] \wedge w'' \in W\}$$
$$[\![\varepsilon : \alpha]\!] \stackrel{\text{def}}{=} \{(w, w, w', w'') \mid w \in W \wedge (w', w'') \in [\![\alpha]\!]\}$$

for atomic relations, while the sematics for '+', ';', and '*' are the obvious ones when seeing $[\![\rho]\!]$ as a binary relation on *pairs* of nodes. Finally,

$$[\![\langle\rho\rangle]\!] \stackrel{\text{def}}{=} \{w \in W \mid \exists w', w'' \in W, (w, w', w, w'') \in [\![\rho]\!]\} ,$$

meaning that we should find two paths starting from w and related by ρ.

[1] 1. Provide a relational path formula ρ_ℓ such that

$$[\![\rho_\ell]\!] = \{(w_1, w_1', w_2, w_2') \mid \exists n \in \mathbb{N}, w_1 \downarrow^n w_1' \wedge w_2 \downarrow^n w_2'\} .$$

Intuitively, ρ_ℓ relates two paths (w_1, w_1') and (w_2, w_2'), both in $[\![\downarrow^*]\!]$, such that w_1' is as far below w_1 as w_2' is below w_2.

Solution: Define
$$\rho_\ell \stackrel{\text{def}}{=} \big((\downarrow : \varepsilon); (\varepsilon : \downarrow)\big)^*$$

[2] 2. Deduce that relational PDL allows to define some non-regular tree languages.

Solution: The tree language $\{f(g^n(a), g^n(b)) \mid n \geq 0\}$ is well-known to be non-regular (a simple pumping argument suffices). Let $A \stackrel{\text{def}}{=} \{p, p'\}$ and $f \stackrel{\text{def}}{=} p \wedge p'$, $g \stackrel{\text{def}}{=} p \wedge \neg p'$, $a \stackrel{\text{def}}{=} \neg p \wedge p'$, and $b \stackrel{\text{def}}{=} \neg p \wedge \neg p'$. This language is defined by the following formula:

$$f \wedge \langle\downarrow; (\mathsf{first} \wedge \langle\downarrow^*\rangle a)?; \rightarrow\rangle \mathsf{last}$$

ensuring the root is labelled by f and has exactly two children, the first dominating an a

$$\wedge\, [\downarrow; \downarrow^*; (\neg\mathsf{leaf})?](g \wedge [\downarrow](\mathsf{first} \wedge \mathsf{last}))$$

ensuring all the non-leaf nodes below the root are labelled by g and have a single child

$$\wedge\, \langle\rho_\ell; (a? : \varepsilon); (\varepsilon : b?)\rangle$$

ensuring the two g-branches have the same length, with one ending in an a (necessarily a leaf since otherwise it would be labelled g) and the other with a b (also a leaf).

[3] 3. Recall from the classes that some natural languages, including Swiss German, exhibit cross-serial dependencies of the form $L_{\text{cross}} \stackrel{\text{def}}{=} \{a^n b^m c^n d^m \mid n, m > 0\}$. Provide a relational PDL node formula φ_{cross} such that $\text{yield}(L(\varphi_{\text{cross}})) = L_{\text{cross}}$.

Solution: There are multiple ways around this question. Here is one solution: we define the tree language

$$L \stackrel{\text{def}}{=} \{f\big(g(a,\Box)^n \cdot a, g(b,\Box)^m \cdot b, g(c,\Box)^n \cdot c, g(d,\Box)^m \cdot d\big) \mid n, m > 0\}$$

over $\mathcal{F} \stackrel{\text{def}}{=} \{a^{(0)}, b^{(0)}, c^{(0)}, d^{(0)}, g^{(2)}, f^{(2)}\}$. Clearly $\text{yield}(L) = L_{\text{cross}}$. It remains to define L using a relational PDL formula.

We define for this

$$\rho_{ac} \stackrel{\text{def}}{=} ((g \wedge \langle\downarrow; \mathsf{first}?\rangle a)? : \varepsilon); (\downarrow : \varepsilon); (\varepsilon : (g \wedge \langle\downarrow; \mathsf{first}?\rangle c)?); (\varepsilon : \downarrow)$$
$$\rho_{bd} \stackrel{\text{def}}{=} ((g \wedge \langle\downarrow; \mathsf{first}?\rangle b)? : \varepsilon); (\downarrow : \varepsilon); (\varepsilon : (g \wedge \langle\downarrow; \mathsf{first}?\rangle d)?); (\varepsilon : \downarrow)$$

which describe 'synchronised' steps in the $g(a,\Box)$ and $g(c,\Box)$ branches and $g(b,\Box)$ and $g(d,\Box)$ ones, respectively. It remains to force the trees to be in $T(\mathcal{F})$:

$$\varphi_{\mathcal{F}} \stackrel{\text{def}}{=} [\downarrow^*] \bigvee_{f\in\mathcal{F}} f \wedge \Big(\bigwedge_{g\in\mathcal{F}\setminus\{f\}} \neg g\Big) \wedge \begin{cases} \langle\downarrow; \mathsf{first}?; \rightarrow^k; \mathsf{last}?\rangle\top & \text{if } f \in \mathcal{F}_k, k > 0 \\ [\downarrow]\bot & \text{if } f \in \mathcal{F}_0 \end{cases}$$

and finally to make sure the a, b, c, and d branches are in the correct order and are pairwise related by ρ_{ac} and ρ_{bd}:

$$\begin{aligned} \varphi_{\text{cross}} \stackrel{\text{def}}{=} {} & \varphi_{\mathcal{F}} \wedge f \wedge \big(\langle\downarrow; \mathsf{first}?; (\langle\downarrow^*\rangle a)?; \rightarrow; (\langle\downarrow^*\rangle b)?; \rightarrow; (\langle\downarrow^*\rangle c)?; \rightarrow; (\langle\downarrow^*\rangle d)?\rangle\top\big) \\ & \wedge \langle(\downarrow : \varepsilon); (\varepsilon : \downarrow); \rho_{ac}^*; (a? : \varepsilon); (\varepsilon : c?)\rangle \\ & \wedge \langle(\downarrow : \varepsilon); (\varepsilon : \downarrow); \rho_{bd}^*; (b? : \varepsilon); (\varepsilon : d?)\rangle\ . \end{aligned}$$

[4] 4. Show that the satisfiability problem for relational PDL is undecidable.

Hint: Reduce from the Post Correspondence Problem.

Solution: Only one person attempted this question (successfully). I'll let the others think about it.

2 Event Semantics and Adverbial Modification

Exercise 3. One considers the three following signatures:

$$
(\Sigma_{\mathrm{ABS}}) \qquad
\begin{aligned}
\textsc{john} &: NP \\
\textsc{mary} &: NP \\
\textsc{kissed} &: NP \to NP \to V \\
\textsc{kissed}_\circ &: NP \to NP \to V_\circ \\
\textsc{not} &: (NP \to S_\circ) \to (NP \to S) \\
\textsc{e-clos} &: V \to S \\
\textsc{e-clos}_\circ &: V_\circ \to S_\circ
\end{aligned}
$$

$$
(\Sigma_{\text{S-FORM}}) \qquad
\begin{aligned}
\boldsymbol{John} &: \mathit{string} \\
\boldsymbol{Mary} &: \mathit{string} \\
\boldsymbol{kissed} &: \mathit{string} \\
\boldsymbol{kiss} &: \mathit{string} \\
\boldsymbol{did} &: \mathit{string} \\
\boldsymbol{not} &: \mathit{string}
\end{aligned}
$$

$$
(\Sigma_{\text{L-FORM}}) \qquad
\begin{aligned}
\mathbf{j}, \mathbf{m} &: \mathsf{e} \\
\mathbf{kiss}, \mathbf{past} &: \mathsf{v} \to \mathsf{t} \\
\mathbf{agent}, \mathbf{patient} &: \mathsf{v} \to \mathsf{e} \to \mathsf{t}
\end{aligned}
$$

In Σ_{ABS}, the atomic type NP stands for the syntactic category of *noun phrases*, the atomic types S and $S_\circ$ for the syntactic category of *sentences* (positive and negative), and the atomic type V and $V_\circ$, the syntactic categories of *"open" sentences* (positive and negative). The reason for distinguishing between the categories of positive and negative (open) sentences is merely syntactic. Without such a distinction, the surface realization of a negative expression such as:

$$\textsc{not}\,(\textsc{kissed}\;\textsc{mary})\;\textsc{john}$$

would be:

*John did not kissed Mary

Without this distinction, it would also be possible to iterate negation. This would allow the following ungrammatical sentences to be generated:

*John did not did not kissed Mary
*John did not did not did not kissed Mary
⋮

In $\Sigma_{\text{S-FORM}}$, as usual, *string* is defined to be $o \to o$ for some atomic type o. This allows concatenation ($+$) to be defined as functional composition, and the empty word (ϵ) as the identity.

In $\Sigma_{\text{L-FORM}}$, the atomic type e stands for the semantic category of *entities*, the atomic types t for the semantic category of *truth values*, and the atomic types v for the semantic category of *events*.

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

$$(\mathcal{L}_{\text{SYNT}})\qquad \begin{aligned} \textsc{john} &:= \boldsymbol{John} \\ \textsc{mary} &:= \boldsymbol{Mary} \\ \textsc{kissed} &:= \lambda xy.\, y + \boldsymbol{kissed} + x \\ \textsc{kissed}_\circ &:= \lambda xy.\, y + \boldsymbol{kiss} + x \\ \textsc{not} &:= \lambda fx.\, x + \boldsymbol{did} + \boldsymbol{not} + (f\,\epsilon) \\ \textsc{e-clos},\ \textsc{e-clos}_\circ &:= \lambda x.\, x \end{aligned}$$

$$(\mathcal{L}_{\text{SEM}})\qquad \begin{aligned} \textsc{john} &:= \mathbf{j} \\ \textsc{mary} &:= \mathbf{m} \\ \textsc{kissed},\ \textsc{kissed}_\circ &:= \lambda xye.\, (\mathbf{kiss}\, e) \wedge (\mathbf{agent}\, e\, y) \wedge (\mathbf{patient}\, e\, x) \wedge (\mathbf{past}\, e) \\ \textsc{not} &:= \lambda px.\, \neg(p\, x) \\ \textsc{e-clos},\ \textsc{e-clos}_\circ &:= \lambda p.\, \exists e.\, p\, e \end{aligned}$$

These two morphisms are such that:

$$\mathcal{L}_{\text{SYNT}}(\textsc{e-clos}\,(\textsc{kissed mary john})) = \boldsymbol{John} + \boldsymbol{kissed} + \boldsymbol{Mary}$$

$$\mathcal{L}_{\text{SEM}}(\textsc{e-clos}\,(\textsc{kissed mary john})) = \exists e.\, (\mathbf{kiss}\, e) \wedge (\mathbf{agent}\, e\, \mathbf{j}) \wedge (\mathbf{patient}\, e\, \mathbf{m}) \wedge (\mathbf{past}\, e)$$

The last term may be paraphrased as follows: *there is an event e such that: e is a kissing event; the agent of this kissing event is John; the patient of this kissing event is Mary; and this event e happened in the past.*

[1] 1. Give a term t such that

$$\mathcal{L}_{\text{SYNT}}(t) = \boldsymbol{John} + \boldsymbol{did} + \boldsymbol{not} + \boldsymbol{kiss} + \boldsymbol{Mary},$$

then compute $\mathcal{L}_{\text{SEM}}(t)$.

Solution:

$t = \textsc{not}\,(\lambda x.\, \textsc{e-clos}_\circ\,(\textsc{kissed}_\circ\, \textsc{mary}\, x))\, \textsc{john}$.

$\mathcal{L}_{\text{SEM}}(t) = \neg(\exists e.\, (\mathbf{kiss}\, e) \wedge (\mathbf{agent}\, e\, \mathbf{j}) \wedge (\mathbf{patient}\, e\, \mathbf{m}) \wedge (\mathbf{past}\, e))$

[2] 2. Suppose that one modifies Σ_{ABS} and $\mathcal{L}_{\text{SEM}}$ as follows:

$$(\Sigma_{\text{ABS}})\qquad \begin{aligned} &\vdots \\ \textsc{not} &: (NP \to V_\circ) \to (NP \to V) \\ &\vdots \end{aligned}$$

$$(\mathcal{L}_{\text{SEM}})\qquad \begin{aligned} &\vdots \\ \textsc{not} &:= \lambda pxe.\, \neg(p\, x\, e) \\ &\vdots \end{aligned}$$

What would be wrong?

Solution:

Let $t = \textsc{e-clos}\,(\textsc{not}\,(\lambda x.\, \textsc{kissed}_\circ\, \textsc{mary}\, x)\, \textsc{john})$. We would have

$$\mathcal{L}_{\text{SYNT}}(t) = \boldsymbol{John} + \boldsymbol{did} + \boldsymbol{not} + \boldsymbol{kiss} + \boldsymbol{Mary},$$

and

$$\mathcal{L}_{\text{SEM}}(t) = \exists e.\, \neg((\mathbf{kiss}\, e) \wedge (\mathbf{agent}\, e\, \mathbf{j}) \wedge (\mathbf{patient}\, e\, \mathbf{m}) \wedge (\mathbf{past}\, e)).$$

This last term does not assert that there is no *past kissing event between John and Mary*, but that there is an event which is not a *past kissing event between John and Mary.* Consequently, in a situation where John kissed both Mary and Sue, we would consider "John did not kiss Mary" to be true.

Exercise 4. One extends Σ_{ABS}, $\Sigma_{\text{S-FORM}}$, $\Sigma_{\text{L-FORM}}$, $\mathcal{L}_{\text{SYNT}}$, and $\mathcal{L}_{\text{SEM}}$, respectively, as follows:

$$(\Sigma_{\text{ABS}}) \qquad \begin{aligned} \textsc{hour} &: N_u \\ \textsc{one} &: N_u \to NP_\tau \\ \textsc{for} &: NP_\tau \to ((V \to V) \to S) \to S \\ \textsc{for}_\circ &: NP_\tau \to ((V_\circ \to V_\circ) \to S) \to S \\ \textsc{for}_{\circ\circ} &: NP_\tau \to ((V_\circ \to V_\circ) \to S_\circ) \to S_\circ \end{aligned}$$

where N_u is the syntactic category of *nouns that name units of measurement*, and NP_τ is the syntactic the category of *noun phrases that denote time intervals*;

$$(\Sigma_{\text{S-FORM}}) \qquad \begin{aligned} \boldsymbol{hour} &: \mathit{string} \\ \boldsymbol{one} &: \mathit{string} \\ \boldsymbol{for} &: \mathit{string} \end{aligned}$$

$$(\Sigma_{\text{L-FORM}}) \qquad \begin{aligned} \mathbf{hour} &: \mathsf{i} \to \mathsf{n} \to \mathsf{t} \\ 1 &: \mathsf{n} \\ \mathbf{duration} &: \mathsf{v} \to \mathsf{i} \to \mathsf{t} \end{aligned}$$

where i and n stand for the semantic categories of time intervals and scalar quantities, respectively.

$$(\mathcal{L}_{\text{SYNT}}) \qquad \begin{aligned} \textsc{hour} &:= \boldsymbol{hour} \\ \textsc{one} &:= \lambda x.\, \boldsymbol{one} + x \\ \textsc{for},\ \textsc{for}_\circ,\ \textsc{for}_{\circ\circ} &:= \lambda x f.\, f\,(\lambda y.\, y + \boldsymbol{for} + x) \end{aligned}$$

$$(\mathcal{L}_{\text{SEM}}) \qquad \begin{aligned} \textsc{hour} &:= \lambda xy.\, \mathbf{hour}\, x\, y \\ \textsc{one} &:= \lambda pt.\, p\, t\, 1 \\ \textsc{for},\ \textsc{for}_\circ,\ \textsc{for}_{\circ\circ} &:= \lambda pq.\, \exists t.\, (p\, t) \wedge (q\, (\lambda pe.\, (p\, e) \wedge (\mathbf{duration}\, e\, t))) \end{aligned}$$

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

$$\mathcal{L}_{\text{SYNT}}(t_0) = \mathcal{L}_{\text{SYNT}}(t_1) = \boldsymbol{John} + \boldsymbol{did} + \boldsymbol{not} + \boldsymbol{kiss} + \boldsymbol{Mary} + \boldsymbol{for} + \boldsymbol{one} + \boldsymbol{hour}$$

Solution:

$$\begin{aligned} t_0 = {} & \textsc{for}_\circ\ (\textsc{one}\ \textsc{hour}) \\ & \quad (\lambda q.\, \textsc{not}\, (\lambda x.\, \textsc{e-clos}_\circ\, (q\, (\textsc{kissed}_\circ\, \textsc{mary}\, x)))\ \textsc{john}) \end{aligned}$$

$$\begin{aligned} t_1 = {} & \textsc{not}\ (\lambda x.\, \textsc{for}_{\circ\circ}\ (\textsc{one}\ \textsc{hour}) \\ & \qquad\qquad (\lambda q.\, \textsc{e-clos}_\circ\, (q\, (\textsc{kissed}_\circ\, \textsc{mary}\, x)))) \\ & \quad \textsc{john} \end{aligned}$$

[2] 2. Compute $\mathcal{L}_{\text{SEM}}(t_0)$ and $\mathcal{L}_{\text{SEM}}(t_1)$.

Solution:

$\mathcal{L}_{\text{SEM}}(t_0) =$
$\exists t.\, (\mathbf{hour}\, t\, 1) \wedge \neg(\exists e.\, (\mathbf{kiss}\, e) \wedge (\mathbf{agent}\, e\, \mathbf{j}) \wedge (\mathbf{patient}\, e\, \mathbf{m}) \wedge (\mathbf{past}\, e) \wedge (\mathbf{duration}\, e\, t))$

$\mathcal{L}_{\text{SEM}}(t_1) =$
$\neg(\exists t.\, (\mathbf{hour}\, t\, 1) \wedge (\exists e.\, (\mathbf{kiss}\, e) \wedge (\mathbf{agent}\, e\, \mathbf{j}) \wedge (\mathbf{patient}\, e\, \mathbf{m}) \wedge (\mathbf{past}\, e) \wedge (\mathbf{duration}\, e\, t)))$

[1] 3. Explain the difference between $\mathcal{L}_{\text{SEM}}(t_0)$ and $\mathcal{L}_{\text{SEM}}(t_1)$.

Solution: t_0 and t_1 correspond respectively to the following interpretations:

1. *For one hour, it was not the case that John kissed Mary.*
2. *It was not the case that John kissed Mary for one hour.*