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:

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

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:

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

[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 $\{\text{S, NP, VP, PP}\} \subseteq N$, $\{\text{PRP, VBD, DT, NN, IN}\} \subseteq \Theta$, and $\{\textit{He, hurled, the, ball, into, basket}\} \subseteq L$.

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

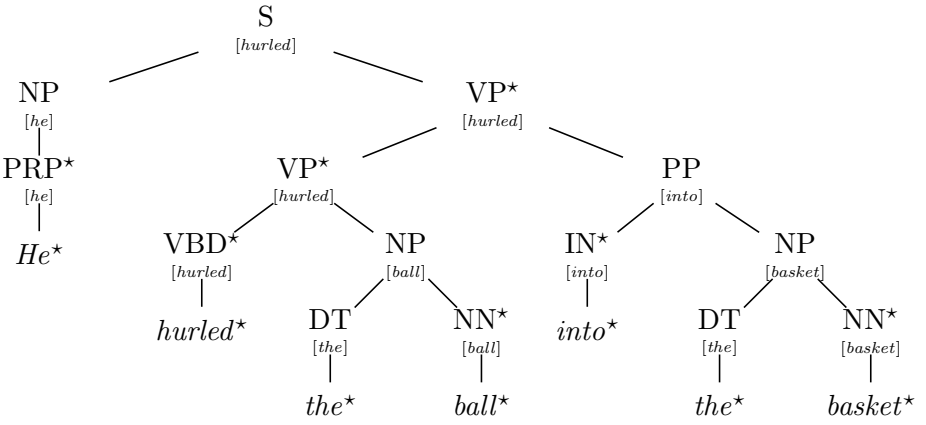

Figure 1: Example of a constituent tree. The head children are starred. The lexical heads of internal nodes are indicated inside brackets.

[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

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

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.

[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.

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

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:

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

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 $W^2 \times W^2$, 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.

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

$$[\![\rho_\ell]\!] = \{(w_1, w_1 w_2, w_1', w_1' w_2') \mid |w_2| = |w_2'| \in \mathbb{N}^*\} .$$

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

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

[4] 3. Recall from the classes that some natural languages, including Swiss German, exhibit cross-serial dependencies of the form $L_{\text{cross}} \stackrel{\text{def}}{=} \{a^n b^m c^n d^m \mid n, m > 1\}$. Provide a relational PDL node formula φ_{cross} such that $\text{yield}(L(\varphi_{\text{cross}})) = L_{\text{cross}}$.

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

Hint: Reduce from the Post Correspondence Problem.

2 Event semantics and adverbial modification

Exercise 3. One considers the three following signatures:

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

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

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

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:

$$^*\text{John did not kissed Mary}$$

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

$$\begin{gathered}
^*\text{John did not did not kissed Mary}\\
^*\text{John did not did not did not kissed Mary}\\
\vdots
\end{gathered}$$

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:

$(\mathcal{L}_{\text{SYNT}})$

$$\begin{aligned}
\text{JOHN} &:= \boldsymbol{John} \\
\text{MARY} &:= \boldsymbol{Mary} \\
\text{KISSED} &:= \lambda xy.\, y + \boldsymbol{kissed} + x \\
\text{KISSED}_\circ &:= \lambda xy.\, y + \boldsymbol{kiss} + x \\
\text{NOT} &:= \lambda fx.\, x + \boldsymbol{did} + \boldsymbol{not} + (f\,\epsilon) \\
\text{E-CLOS, E-CLOS}_\circ &:= \lambda x.\, x
\end{aligned}$$

$(\mathcal{L}_{\text{SEM}})$

$$\begin{aligned}
\text{JOHN} &:= \mathbf{j} \\
\text{MARY} &:= \mathbf{m} \\
\text{KISSED, KISSED}_\circ &:= \lambda xye.\, (\mathbf{kiss}\, e) \wedge (\mathbf{agent}\, e\, y) \wedge (\mathbf{patient}\, e\, x) \wedge (\mathbf{past}\, e) \\
\text{NOT} &:= \lambda px.\, \neg(p\, x) \\
\text{E-CLOS, E-CLOS}_\circ &:= \lambda p.\, \exists e.\, p\, e
\end{aligned}$$

These two morphisms are such that:

$$\mathcal{L}_{\text{SYNT}}(\text{E-CLOS}\,(\text{KISSED MARY JOHN})) = \boldsymbol{John} + \boldsymbol{kissed} + \boldsymbol{Mary}$$

$$\mathcal{L}_{\text{SEM}}(\text{E-CLOS}\,(\text{KISSED MARY JOHN})) = \exists e.\, (\mathbf{kiss}\, e) \wedge (\mathbf{agent}\, e\, \mathbf{j}) \wedge (\mathbf{patient}\, e\, \mathbf{m}) \wedge (\mathbf{past}\, e)$$

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

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

(Σ_{ABS})

$$\begin{array}{c}
\vdots \\
\text{NOT} : (NP \to V_\circ) \to (NP \to V) \\
\vdots
\end{array}$$

$(\mathcal{L}_{\text{SEM}})$

$$\begin{array}{c}
\vdots \\
\text{NOT} := \lambda pxe.\, \neg(p\, x\, e) \\
\vdots
\end{array}$$

What would be wrong?

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:

$$
\begin{array}{rl}
(\Sigma_{\text{ABS}}) & \begin{array}{rl}
\text{HOUR} & : N_u \\
\text{ONE} & : N_u \to NP_\tau \\
\text{FOR} & : NP_\tau \to ((V \to V) \to S) \to S \\
\text{FOR}_\circ & : NP_\tau \to ((V_\circ \to V_\circ) \to S) \to S \\
\text{FOR}_{\circ\circ} & : NP_\tau \to ((V_\circ \to V_\circ) \to S_\circ) \to S_\circ
\end{array}
\end{array}
$$

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*;

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

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

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

$$
\begin{array}{rl}
(\mathcal{L}_{\text{SYNT}}) & \begin{array}{rl}
\text{HOUR} & := \boldsymbol{hour} \\
\text{ONE} & := \lambda x.\, \boldsymbol{one} + x \\
\text{FOR, FOR}_\circ, \text{FOR}_{\circ\circ} & := \lambda x f.\, f\,(\lambda x.\, y + \boldsymbol{for} + x)
\end{array}
\end{array}
$$

$$
\begin{array}{rl}
(\mathcal{L}_{\text{SEM}}) & \begin{array}{rl}
\text{HOUR} & := \lambda xy.\, \mathbf{hour}\, x\, y \\
\text{ONE} & := \lambda pt.\, p\, t\, 1 \\
\text{FOR, FOR}_\circ, \text{FOR}_{\circ\circ} & := \lambda pq.\, \exists t.\, (p\, t) \wedge (q\, (\lambda pe.\, (p\, e) \wedge (\mathbf{duration}\, e\, t)))
\end{array}
\end{array}
$$

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

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

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

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