Exam: Thematic Role Labeling

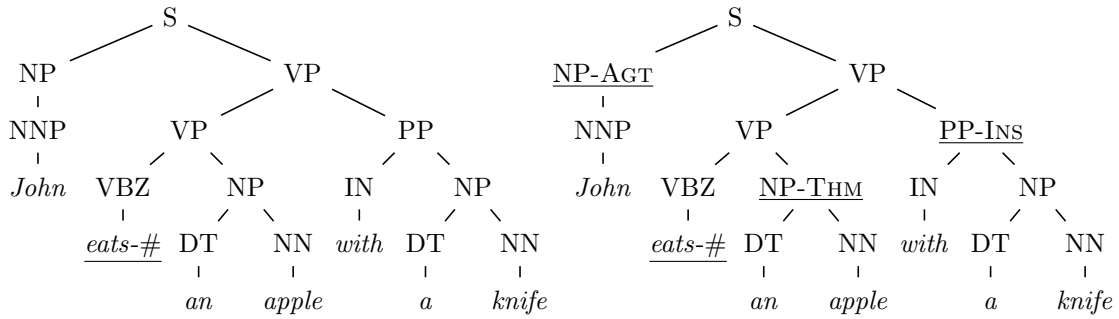
Duration: 3 hours.
Written documents are allowed. The numbers in front of questions are indicative of hardness or duration. The exam is clearly too long, but offers a variety of subjects, with independent exercises. You should pick your favorite questions and skip the rest.

The problem we address in this exam is that of semantic role labeling. It is commonplace in linguistic description to split the sentence analysis into two subparts:

syntax which provides the tree structure of a sentence, and

semantic analysis, which is here reduced to the (shallow) task of identifying thematic roles in the trees.

We consider here a very restricted setting inspired by Gildea and Jurafsky (2002). A constituent syntax analysis is provided (left-hand side tree) along with a **target word** w_t labeling one of its leaves (marked by # and underlined in the tree) and a **thematic frame** telling which thematic roles we intend to identify in this tree (in the example, this would be $\Theta = \{\textsc{Agt}, \textsc{Thm}, \textsc{Ins}\}$ standing for *agent*, *theme*, and *instrument* resp.). The output of the task is the tree annotated with thematic labels (right-hand side tree, with underlined nodes for better readability).



Formally, given a ranked alphabet Σ with such symbols as $\mathrm{S}^{(2)}$, $\mathrm{NP}^{(1)}$, $\mathit{John}^{(0)}$ and *maximal rank 2*, we want to obtain a tree labeled by $\Sigma \times \mathcal{R}$ where the additional set of labels $\mathcal{R}$ is defined as $\Theta \uplus \{\#, \bot\}$, where # is the marker for the target word, and $\bot$ is a dummy role. The alphabet $\Sigma \times \mathcal{R}$ is ranked using the arities of Σ, i.e. $(\mathrm{NP}, \textsc{Thm})$ is of arity 2 corresponding to the arity 2 of NP.

An **$\mathcal{R}$-labeled** Σ-tree is then a tree in $T(\Sigma \times \mathcal{R})$ such that:

1. each thematic role in Θ appears exactly once in the tree, and
2. the target leaf is labeled $(w_t, \#)$ and no other node is labeled by #.

A **marked** tree is a tree t in $T(\Sigma \times \{\#, \bot\})$ with a single leaf marked by $\#$, which has to be labeled $(w_t, \#)$. The left-hand side tree is a marked tree.

Given a marked tree t, a **$\mathcal{R}$-labeling** of t is an $\mathcal{R}$-labeled Σ-tree t' s.t. $\mathrm{dom}(t) = \mathrm{dom}(t')$ and for the position u in $\mathrm{dom}(t)$ s.t. $t(u) = (w_t, \#)$, $t'(u) = (w_t, \#)$, i.e. the labeling preserves the marked target leaf. For instance, the right-hand side tree is an $\mathcal{R}$-labeling of the left-hand side tree (with $\bot$ labels omitted in both cases).

In order to perform the thematic role labeling task, we intend to train a **weighted tree transducer** that inputs a marked tree and outputs every $\mathcal{R}$-labeling of it along with some weight in $\mathbb{R}$ (representing its confidence in the added labels). One way of learning this transformation is to use a fixed set of **features $\mathbf{\Phi}$**. Assume for now that you know which nodes need to be labelled. For a given node, the semantic role labeller will assign a label r to that node from the set of thematic roles Θ by voting for each label.

The vote for a label r is the inner product $V_r \stackrel{\text{def}}{=} \boldsymbol{\lambda}_r \cdot \mathbf{\Phi}$. The predicted label is the label with the highest score

$$r \stackrel{\text{def}}{=} \operatorname*{argmax}_{r \in \Theta} V_r = \operatorname*{argmax}_{r \in \Theta} \boldsymbol{\lambda}_r \cdot \mathbf{\Phi} \; . \qquad (*)$$

The vector $\mathbf{\Phi}$ is a vector of boolean valued features $\phi_1 \cdots \phi_n$ where each ϕ_i is associated to a weight $\lambda_{r,i}$ in $\mathbb{R}$. The vote for a full $\mathcal{R}$-labeled tree t is the sum $V_t \stackrel{\text{def}}{=} \sum_{r \in \Theta} V_r$ of the votes for each (unique by definition) r-labeled node in the tree t, and the predicted tree is

$$t' \stackrel{\text{def}}{=} \operatorname*{argmax}_{t' \text{ } \mathcal{R}\text{-labeling of } t} V_{t'} \; . \qquad (\dagger)$$

A feature is thus a mapping from positions in input trees to boolean values, telling if the position exhibits a particular characteristic that will help in predicting the correct label. We implement each feature ϕ_i using a PDL formula φ_i holding (or not) at the tested position. We consider two kinds of features in this exam:

head word information: given a node in the tree, recover the label (in Σ_0) of its head. For instance, NP-AGT verifies the feature $h = \textit{John}$, NP-THM verifies $h = \textit{apple}$, and PP-INS verifies $h = \textit{knife}$. Observe that this lexical information is very helpful in our example: it is unlikely for an *apple* to be either an agent or an instrument in an eating action.

path information: given a node in the tree, recover the *shortest* path to the target marked word w_t. Concretely, a path feature $p = \pi$ is defined by the syntax $\pi ::= \pi_u \uparrow a \downarrow \pi_d$ where $\pi_u \in \Sigma^+$, $\pi_d \in \Sigma^*$, and $a \in \Sigma$, denoting that the upward path to the least common ancestor (LCA) of the node and w_t is reached by reading π_u (the node label included), that this LCA is labeled a, and that we can then go downwards to w_t by reading π_d (the leaf labeled w_t excluded). For instance, for NP-AGT, $p = \text{NP} \uparrow \text{S} \downarrow \text{VP VP VBZ}$, for NP-THM, $p = \text{NP} \uparrow \text{VP} \downarrow \text{VBZ}$, and for PP-INS, $p = \text{PP} \uparrow \text{VP} \downarrow \text{VP VBZ}$. Again, this information is quite helpful: the path from the PP node is rather unlikely for an agent.

Exercise 1 (PDL Implementation)**.** It is more convenient to work in PDL with a set of atomic propositions $A = \Sigma \uplus \{\#\}$ allowing to have several label simultaneously on a single node. (We will later restrict ourselves to $\mathcal{R}$-labeled Σ-trees.)

[2] 1. Given a feature $p = \pi$, construct a PDL formula φ_π that holds at a tree node iff the shortest path from that node to the #-labeled node matches π.

Let $\pi = \pi_u \uparrow a \downarrow \pi_d$. We define inductively

$$\begin{aligned}
\varphi_{b\pi'} &= b \wedge \langle\uparrow\rangle\varphi_{\pi'} && \text{(upward path)}\\
\varphi_{\uparrow a\downarrow\pi_d} &= (\langle\rightarrow\rangle\varphi_{\pi_d} \vee \langle\leftarrow\rangle\varphi_{\pi_d}) \wedge \langle\uparrow\rangle a && \text{(upon reaching the LCA)}\\
\varphi_{b\pi'_d} &= b \wedge \langle\downarrow\rangle\varphi_{\pi'_d} && \text{(downward path)}\\
\varphi_\varepsilon &= \# \,. && \text{(checking the \# mark)}
\end{aligned}$$

As a quick justification: because we work with at most binary arities, the LCA of the node and of the target leaf is necessarily a binary node, and we merely need to go to the left or right sibling just before reaching it.

Beware that a naive implementation like

$$\varphi_{\uparrow a\downarrow\pi_d} = \langle\uparrow\rangle(a \wedge \langle\downarrow\rangle\varphi_{\pi_d})$$

does not work because we could go down using the same child of the a-labeled node as we came from, but then the a-labeled node is not the LCA.

An alternative is to check at each upward step that we have not reached the LCA yet:

$$\begin{aligned}
\varphi_{b\pi'} &= b \wedge \neg\langle\downarrow^*\rangle\# \wedge \langle\uparrow\rangle\varphi_{\pi'}\\
\varphi_{\uparrow a\downarrow\pi_d} &= \neg\langle\downarrow^*\rangle\# \wedge \langle\uparrow\rangle(a \wedge \langle\downarrow\rangle\varphi_{\pi_d})\,.
\end{aligned}$$

[5] 2. Given a feature $h = \ell$ with $\ell \in \Sigma_0$ a leaf label, construct a PDL formula φ_ℓ that holds at a tree node iff its lexical head is ℓ.

This is meant as an *open* question, though you are encouraged to reuse the **head percolation functions** described in class. Your formula should of course work on the example.

Exercise 2 (PDL and Bisimulations)**.** We consider here the restriction of PDL to the **vertical axis**: only the $\downarrow$ atomic relation is permitted. Let us consider two tree structures $\mathfrak{M} = \langle W, \downarrow, (P_a)_{a\in A}\rangle$ and $\mathfrak{M}' = \langle W', \downarrow, (P_a)_{a\in A}\rangle$. A non-empty relation $Z \subseteq W \times W'$ is a **bisimulation** (with converses) between $\mathfrak{M}$ and $\mathfrak{M}'$ if for all nodes $w \in W$ and $w' \in W'$ s.t. $w\ Z\ w'$:

(i) $\{a \in A \mid P_a(w)\} = \{a' \in A \mid P_a(w')\}$,

(ii) if $w \downarrow w_1$, then there exists w'_1 s.t. $w' \downarrow w'_1$ and $w_1 \; Z \; w'_1$,

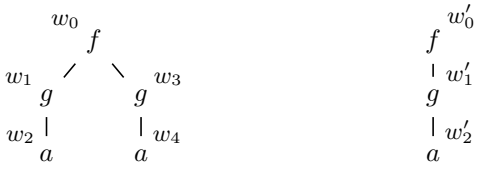
(iii) if $w' \downarrow w'_1$, then there exists w_1 s.t. $w \downarrow w_1$ and $w_1 \; Z \; w'_1$,

(iv) if $w \uparrow w_1$, then there exists w'_1 s.t. $w' \uparrow w'_1$ and $w_1 \; Z \; w'_1$,

(v) if $w' \uparrow w'_1$, then there exists w_1 s.t. $w \uparrow w_1$ and $w_1 \; Z \; w'_1$,

where $\uparrow$ denotes the converse of $\downarrow$. We say that w and w' are **bisimilar**, noted $w \underline{\leftrightarrow} w'$, if there exists a bisimulation Z s.t. $w \; Z \; w'$.

[1] 1. Show that the left node w_1 labeled g in the tree $f(g(a), g(a))$ and the unique node w'_1 labeled g in $f(g(a))$ are bisimilar:



The relation Z defined by

$$Z = \{(w_0, w'_0), (w_1, w'_1), (w_3, w'_1), (w_2, w'_2), (w_4, w'_2)\}$$

is a bisimulation between the two trees.

[4] 2. Show that, if $w \underline{\leftrightarrow} w'$ and φ is a PDL formula using only the vertical axis, then $w \in [\![\varphi]\!]_{\mathfrak{M}}$ iff $w' \in [\![\varphi]\!]_{\mathfrak{M}'}$.

Without loss of generality, we assume converse operators to be "pushed to the leaves" using the converse equivalences seen in class, so that the syntax of vertical PDL path formulæ is

$$\pi ::= \downarrow \mid \uparrow \mid \pi; \pi \mid \pi + \pi \mid \pi^* \mid \varphi?$$

where φ ranges over vertical PDL formulæ.

Let us fix a bisimulation relation Z between $\mathfrak{M}$ and $\mathfrak{M}'$. The proof proceeds by induction over PDL formulæ φ:

For a an atomic proposition in A, we have $w \in [\![a]\!]_{\mathfrak{M}}$ iff $w' \in [\![a]\!]_{\mathfrak{M}'}$ by (i).

For $\top$ we have $w \in [\![\top]\!]_{\mathfrak{M}} = W$ iff $w' \in [\![\top]\!]_{\mathfrak{M}'} = W'$.

For $\neg\varphi$ we have $w \in [\![\neg\varphi]\!]_{\mathfrak{M}}$ iff $w \notin [\![\varphi]\!]_{\mathfrak{M}}$, by ind. hyp. iff $w' \notin [\![\varphi]\!]_{\mathfrak{M}'}$, iff $w' \in [\![\neg\varphi]\!]_{\mathfrak{M}'}$.

For $\varphi \vee \varphi'$ similarly by ind. hyp.

For $\langle\pi\rangle\varphi$ we have $w \in [\![\langle\pi\rangle\varphi]\!]_{\mathfrak{M}}$ iff there exists w_1 s.t. $w_1 \in [\![\varphi]\!]_{\mathfrak{M}}$ and $(w, w_1) \in [\![\pi]\!]_{\mathfrak{M}}$. Let us show using the ind. hyp. along with (ii) and (iv) that

$$w \; Z \; w' \text{ and } (w, w_1) \in [\![\pi]\!]_{\mathfrak{M}} \text{ imply } \exists w'_1 \in W', w_1 \; Z \; w'_1 \text{ and } (w', w'_1) \in [\![\pi]\!]_{\mathfrak{M}'} \quad (\ddagger)$$

By ind. hyp. we deduce from (‡) that $w' \in [\![\langle\pi\rangle\varphi]\!]_{\mathfrak{M}'}$, and symmetrically using (iii) and (v) we can show the equivalence.

Let us prove (‡) by induction on PDL path formulæ π:

For $\downarrow$ by (ii).

For $\uparrow$ by (iv).

For $\pi;\pi'$ we have $(w, w_1) \in [\![\pi;\pi']\!]_{\mathfrak{M}}$ iff there exists w_2 s.t. $(w, w_2) \in [\![\pi]\!]_{\mathfrak{M}}$ and $(w_2, w_1) \in [\![\pi']\!]_{\mathfrak{M}}$. By ind. hyp. on π, the first implies the existence of w'_2 s.t. $w_2 \; Z \; w'_2$ and $(w', w'_2) \in [\![\pi]\!]_{\mathfrak{M}'}$. Using the ind. hyp. on π', there exists w'_1 s.t. $w_1 Z w'_1$ and $(w'_2, w'_1) \in [\![\pi']\!]_{\mathfrak{M}'}$. But then $(w', w'_1) \in [\![\pi;\pi']\!]_{\mathfrak{M}'}$.

For $\pi + \pi'$ by ind. hyp.

For π^* by ind. hyp. (similar to the $\pi;\pi'$ case, only longer...)

For $\varphi?$ by ind. hyp. on φ.

Exercise 3 (Translation to Automata)**.** Recall that, for any PDL formula φ, one can construct an equivalent tree automaton that recognizes a tree $\mathfrak{M}$ iff $\mathfrak{M}, \text{root} \models \varphi$, i.e. φ holds at the root of $\mathfrak{M}$. We apply this construction to formulæ $\langle\downarrow^*\rangle(r \wedge \varphi_i)$ where r is a role in Θ and φ_i a formula implementing some feature ϕ_i, thereby obtaining a tree automaton $\mathcal{A}_{r,i}$, which can be assumed wlog. to be bottom-up deterministic.

[2] 1. An issue is that we need to restrict ourselves to $\mathcal{R}$-labeled trees. We can do this by intersecting with a tree automaton. Give a deterministic bottom-up automaton $\mathcal{A}_{\mathcal{R}} = \langle Q_{\mathcal{R}}, \Sigma \times \mathcal{R}, \delta_{\mathcal{R}}, F_{\mathcal{R}} \rangle$ that recognizes the set of $\mathcal{R}$-labeled Σ-trees with w_t as target leaf.

Set $Q_{\mathcal{R}} = 2^{\Theta \uplus \{\#\}}$, $F_{\mathcal{R}} = \{\Theta \uplus \{\#\}\}$, and

$$\begin{aligned}\delta_{\mathcal{R}} = &\{(q_1, \ldots, q_n, (a, r)^{(n)}, q) \mid r \in \Theta, a \in \Sigma \text{ and } q_1 \uplus \cdots \uplus q_n \uplus \{r\} = q\} \\ &\cup \{(q_1, \ldots, q_n, (a, \bot)^{(n)}, q) \mid a \in \Sigma, q_1 \uplus \cdots \uplus q_n = q\} \\ &\cup \{((w_t, t)^{(0)}, \{\#\})\} \; .\end{aligned}$$

[3] 2. The translation from PDL to tree automata is however a computationally expensive procedure, and there is ample room for improvement when we know the formula in question is of a restricted form.

Let $\pi = \pi_u \uparrow a \downarrow \pi_d$ be a path feature value, r a thematic role in Θ, and w_t a target word. Give a construction for a deterministic bottom-up tree automaton

$\mathcal{A}_{r,\pi} = \langle Q_{r,\pi}, \Sigma \times \mathcal{R}, \delta_{r,\pi}, F_{r,\pi} \rangle$ s.t. the intersection automaton $\mathcal{A}_{\mathcal{R}} \cap \mathcal{A}_{r,\pi}$ accepts only $\mathcal{R}$-labeled Σ-trees with π denoting the shortest path between the node labeled (b, r) for some b and the leaf labeled $(w_t, \#)$. We wish $\mathcal{A}_{r,\pi}$ to be of size linear in $|\pi|$ (defined as $|\pi_u| + |\pi_d| + 1$).

Set $Q_{r,\pi} = \{i, f\} \uplus \mathrm{Suff}_+(\pi_u) \uplus \mathrm{Pref}(\pi_d)$ (i.e. we distinguish between proper suffixes of π_u and prefixes of π_d, even if they are the same strings), $F_{r,\pi} = \{f\}$, and

$$\begin{aligned}
\delta_{r,\pi} = {} & \{(q_1, \ldots, q_n, (b, r')^{(n)}, q) \mid b \in \Sigma, r' \neq r, r' \neq \#, q_1 = \cdots = q_n = q = i\} \\
& \cup \{((w_t, \#)^{(0)}, \pi_d)\} \\
& \cup \{(q_1, \ldots, q_n, (b, r')^{(n)}, q) \mid r' \neq r, r' \neq \#, \exists j, q_j = qb \in \mathrm{Pref}(\pi_d) \text{ and } \forall k \neq j, q_k = i\} \\
& \cup \{(q_1, \ldots, q_n, (b, r)^{(n)}, q) \mid \pi_u = bq \text{ and } \forall j, q_j = i\} \\
& \cup \{(q_1, \ldots, q_n, (b, r')^{(n)}, q) \mid r' \neq r, r' \neq \#, \exists j, q_j = bq \in \mathrm{Suff}_+(\pi_u) \text{ and } \forall k \neq j, q_k = i\} \\
& \cup \{(\varepsilon, \varepsilon, (a, r')^{(2)}, f) \mid r' \neq r, r' \neq \#\} \\
& \cup \{(q_1, \ldots, q_n, (b, r')^{(n)}, f) \mid r' \neq r, r' \neq \#, b \in \Sigma, \exists j, q_j = f \text{ and } \forall k \neq j, q_k = i\} ,
\end{aligned}$$

n ranging over $\{0, 1, 2\}$ since we consider trees with maximal arity 2.

The automaton has

$$\begin{aligned}
|\delta_{r,\pi}| &= 3|\Sigma|(|\Theta| - 1) + 1 + 3|\pi_d|(|\Theta| - 1) + 3 + 3(|\pi_u| - 1)(|\Theta| - 1) + |\Theta| - 1 + 3|\Sigma|(|\Theta| - 1) \\
&= O(|\Theta||p| + |\Sigma||\Theta|)
\end{aligned}$$

transitions.

Definition 1 (Weighted Tree Transducers)**.** A **weighted bottom-up tree transducer** (wBUTT) over a commutative semiring $\mathbb{K} = \langle K, \oplus, \odot, 0_{\mathbb{K}}, 1_{\mathbb{K}} \rangle$ is a tuple $\mathcal{T} = \langle Q, \Sigma, \Delta, R, F \rangle$ where Q is a finite set of states, Σ a ranked input alphabet, Δ a ranked output alphabet, $F \subseteq Q$ a set of final states, and R a finite set of rules of form

$$a^{(n)}(q_1(x_1), \ldots, q_n(x_n)) \to_\kappa q(t)$$

where a is an input symbol from Σ, $q_1, \ldots, q_n, q$ states in Q, $x_1, \ldots, x_n$ variables in $\mathcal{X}$, κ a weight in $\mathbb{K}$, and t a term in $T(\Delta, \mathcal{X})$. A wBUTT is **alphabetic** (an awBUTT) if in every rule t is of form $b^{(n)}(x_1, \ldots, x_n)$ with b in Δ, i.e. if the rules only relabel the tree without further change.

In order to account for weights, the usual rewrite semantics $\Rightarrow$ are refined to compute the current weight during a derivation. Write $C[t\{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}] \Rightarrow_\kappa C[t'\{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}]$ if $\rho = t \to_\kappa t'$ is a rule in R. We write $t \Rightarrow_\kappa^\rho t'$ if there exists m intermediate steps $t \Rightarrow_{\kappa_1}^{\rho_1} t_1 \cdots t_{m-1} \Rightarrow_{\kappa_m}^{\rho_m} t'$, $\rho = \rho_1 \cdots \rho_m$, and $\kappa = \bigodot_{i=1}^m \kappa_i$ (thus $\rho = \varepsilon$ and $\kappa = 1_{\mathbb{K}}$ if $m = 0$), so that the weight of a run ρ is the product of its individual weights.

The weighted transduction in $T(\Sigma) \times \mathbb{K} \times T(\Delta)$ defined by a wBUTT is

$$[\![\mathcal{T}]\!] \stackrel{\text{def}}{=} \{(t, \bigoplus_{\rho \mid \exists q \in F. t \Rightarrow^{\rho}_{\kappa_\rho} q(t')} \kappa_\rho, t') \mid t \in T(\Sigma), t' \in T(\Delta)\}$$

i.e. we sum the weights of all the possible transductions from t to t'. Thus a transduction from t to t' with no possible run gets a weight of $0_{\mathbb{K}}$.

A BUTT $\mathcal{T}$ is **unambiguous** if, for every pair of trees (t, t') in $T(\Sigma) \times T(\Delta)$, there is at most one run for it, i.e.

$$\forall (t, t') \in T(\Sigma) \times T(\Delta), |\{\rho \in R^* \mid \exists \kappa \neq 0_{\mathbb{K}}. \exists q \in F. t \Rightarrow^{\rho}_{\kappa} q(t')\}| \leq 1 \ .$$

Abusing notations, we write

$$[\![\mathcal{T}]\!] = \{(t, \kappa, t') \mid t \in T(\Sigma), t' \in T(\Delta), \exists \rho \in R^*. \exists q \in F. t \Rightarrow^{\rho}_{\kappa} q(t')\} \ ,$$

for an unambiguous BUTT, so that the pairs $(t, t') \in T(\Sigma) \times T(\Delta)$ missing in $sem\mathcal{T}$ are implicitly with $\kappa = 0_{\mathbb{K}}$.

Exercise 4 (Weigthed Tree Transducers)**.** In order to implement $(*)$ using weighted tree transducers, we work in the **max-plus** semiring $\mathbb{M} = \langle \mathbb{R}, \max, +, -\infty, 0 \rangle$.

[2] 1. Give an unambiguous awBUTT $\mathcal{T}_{\mathcal{R}}$ that translates marked trees into their $\mathcal{R}$-labelings:

$$[\![\mathcal{T}_{\mathcal{R}}]\!] = \{(t, 0, t') \mid t \text{ marked and } t' \text{ an } \mathcal{R}\text{-labeling of } t\} \ .$$

(Recall that 0 is $1_{\mathbb{M}}$.)

This is quite similar to Exercise 3.1. Let $\mathcal{T}_{\mathcal{R}} = \langle 2^{\Theta \uplus \{\#\}}, \Sigma \times \{\bot, \#\}, \Sigma \times \mathcal{R}, R_{\mathcal{R}}, \{\Theta \uplus \{\#\}\rangle$ where

$$\begin{aligned} R_{\mathcal{R}} = &\{(a, \bot)^{(n)}(q_1(x_1), \ldots, q_n(x_n)) \to_0 q((a, r)^{(n)}(x_1, \ldots, x_n) \mid r \in \Theta, a \in \Sigma, q_1 \uplus \cdots \uplus q_n \uplus \{r\} = q\} \\ &\cup \{(a, \bot)^{(n)}(q_1(x_1), \ldots, q_n(x_n)) \to_0 q((a, \bot)^{(n)}(x_1, \ldots, x_n) \mid a \in \Sigma, q_1 \uplus \cdots \uplus q_n = q\} \\ &\cup \{(w_t, \#)^{(0)}() \to_0 \{\#\}(w_t, \#)^{(0)}()\} \ . \end{aligned}$$

This wBUTT is visibly alphabetic, and unambiguous since there is a single run for each different output tree.

[2] 2. Show that alphabetic weighted bottom-up tree transductions are closed under **Hadamard product** over arbitrary commutative semirings: given two awBUTTs $\mathcal{T}_1$ and $\mathcal{T}_2$, construct an awBUTT $\mathcal{T}_1 \odot \mathcal{T}_2$ s.t.

$$[\![\mathcal{T}_1 \odot \mathcal{T}_2]\!] = \{(t, \kappa_1 \odot \kappa_2, t') \mid (t, \kappa_1, t') \in [\![\mathcal{T}_1]\!], (t, \kappa_2, t') \in [\![\mathcal{T}_2]\!]\}$$

and $\mathcal{T}_1 \odot \mathcal{T}_2$ is unambiguous if $\mathcal{T}_1$ and $\mathcal{T}_2$ are unambiguous.

Let $\mathcal{T}_1 = \langle Q_1, \Sigma, \Delta, R_1, F_1 \rangle$ and $\mathcal{T}_2 = \langle Q_2, \Sigma, \Delta, R_2, F_2 \rangle$ and construct $\mathcal{T}_1 \odot \mathcal{T}_2 = \langle Q_1 \times Q_2, \Sigma, \Delta, R, F_1 \times F_2 \rangle$ where

$$\begin{aligned} R = \{ & a^{(n)}((q_1, q_1')(x_1), \ldots, (q_n, q_n')(x_n)) \to_{\kappa_1 \odot \kappa_2} (q, q')(b^{(n)}(x_1, \ldots, x_n)) \\ & \mid a^{(n)}(q_1(x_1), \ldots, q_n(x_n)) \to_{\kappa_1} q(b^{(n)}(x_1, \ldots, x_n)) \in R_1 \\ & \text{and } a^{(n)}(q_1'(x_1), \ldots, q_n'(x_n)) \to_{\kappa_2} q'(b^{(n)}(x_1, \ldots, x_n)) \in R_2 \} \, . \end{aligned}$$

[2] 3. Assume you are provided with a deterministic bottom-up tree automaton $\mathcal{A}_{r,i}$ for the role r and the feature ϕ_i. Give an unambiguous awBUTT $\mathcal{T}_{r,i}$ s.t. $\mathcal{T}_{r,i} \odot \mathcal{T}_{\mathcal{R}}$ maps marked trees into their $\mathcal{R}$-labelings that satisfy ϕ_i at the node labeled r with weight $\lambda_{r,i}$: more formally, if we define $r(t)$ as the (unique) node labeled r in the $\mathcal{R}$-labeled tree t,

$$[\![\mathcal{T}_{r,i} \odot \mathcal{T}_{\mathcal{R}}]\!] = \{(t, \lambda_{r,i}, t') \mid t \text{ marked}, t' \text{ an } \mathcal{R}\text{-labeling of } t, \text{ and } t', r(t') \models \varphi_i\} \, .$$

Let $\mathcal{A}_{r,i} = \langle Q_{r,i}, \Sigma \times \mathcal{R}, \delta_{r,i}, F_{r,i} \rangle$ and define $\mathcal{T}_{r,i} = \langle Q_{r,i}, \Sigma \times \{\bot, \#\}, \Sigma \times \mathcal{R}, R_{r,i}, F_{r,i} \rangle$ with

$$\begin{aligned} R_{r,i} = & \{(a, \bot)^{(n)}(q_1(x_1), \ldots, q_n(x_n)) \to_0 q((a, r')^{(n)}(x_1, \ldots, x_n)) \\ & \qquad \mid r' \neq r, r' \neq \# \text{ and } (q_1, \ldots, q_n, (a, r')^{(n)}, q) \in \delta_{r,i}\} \\ & \cup \{(a, \bot)^{(n)}(q_1(x_1), \ldots, q_n(x_n)) \to_{\lambda_{r,i}} q((a, r)^{(n)}(x_1, \ldots, x_n)) \\ & \qquad \mid (q_1, \ldots, q_n, (a, r)^{(n)}, q) \in \delta_{r,i}\} \\ & \cup \{(w_t, \#)^{(0)}() \to_0 q((w_t, \#)^{(0)}) \mid ((w_t, \#)^{(0)}, q) \in \delta_{r,i}\} \, . \end{aligned}$$

Since $\mathcal{A}_{r,i}$ is deterministic, there is a single run depending on the output tree (we basically simulate $\mathcal{A}_{r,i}$ on the output tree). The value of the run is $\lambda_{r,i}$ since there is exactly one r-labeled node and thus we have to trigger the rule with weight $\lambda_{r,i}$ exactly once.

[1] 4. Give the full awBUTT $\mathcal{T}$ that performs the labeling task, i.e. that computes

$$[\![\mathcal{T}]\!] = \{(t, V_{t'}, t') \mid t \text{ marked and } t' \text{ an } \mathcal{R}\text{-labeling of } t\} \, .$$

Define

$$\mathcal{T} = \mathcal{T}_{\mathcal{R}} \odot \bigodot_{r \in \Theta} \bigodot_{i=1}^{n} \mathcal{T}_{r,i} \, .$$

Exercise 5 (Parsing with Thematic Roles)**.** Assume that you have an unweighted ε-free context-free grammar $\mathcal{G} = \langle N, T, P, S \rangle$ with $N = \Sigma_{>0}$ and $T = \Sigma_0$, and the awBUTT $\mathcal{T}$ as above. We want to define a weighted Cocke Kasami Younger (CKY) parsing algorithm that will be able to return the semantically labeled parse with the highest weight in a sentence with a marked element.

[2] 1. First extend (give the inference rules for) the classical, unweighted CKY algorithm to be able to handle context-free rules of form $A \to X$ and $A \to XY$ where A is a nonterminal and X, Y symbols in $N \cup T$.

It suffices to allow items of form $\langle i, X, j\rangle$ with $0 \leq i < j \leq n$ and $X \in N \cup T$:

$$\frac{a = a_i}{\langle i-1, a, i\rangle}\ (\mathsf{Scan})$$

$$\frac{A \to X_1 \cdots X_m \quad \langle i_0, X_1, i_1 - 1\rangle \cdots \langle i_{m-1}, X_m, i_m\rangle}{\langle i_0, A, i_m\rangle}\ (\mathsf{Complete})$$

with objective $\langle 0, S, n\rangle$ as usual.

[4] 2. Design (give the inference rules for) a weighted CKY parsing algorithm that returns the $\mathcal{R}$-labeled parse tree with the highest weight output by $\mathcal{T}$ for a given sentence $w = a_1 \cdots a_m$ with a marked terminal symbol (i.e. $a_i = (w_t, \#)$ for some i). You can modify the input context-free grammar in any way you see fit.

Let $\mathcal{G} = \langle N, T, P, S\rangle$ with $N = \Sigma_{>0}$ and $T = \Sigma_0$, and $\mathcal{T} = \langle Q, \Sigma \times \{\bot, \#\}, \Sigma \times \mathcal{R}, R, F\rangle$.

There are (at least) two ways of treating this question:

(a) either process the CFG with $\mathcal{T}$ and obtain a weighted tree automaton as output, which can be processed using a variation over weighted CKY,

(b) or give a new set of deduction rules that work on $\mathcal{G}$ and $\mathcal{T}$ synchronously.

For the first technique, define the tree automaton $\mathcal{A} = \langle (N \cup T) \times Q, \Sigma \times \mathcal{R}, \delta, \{S\} \times F\rangle$ with weighted transitions

$$\begin{aligned}\delta = \{&((X_1, q_1), \ldots, (X_m, q_m), (a, r)^{(m)}, (A, q), \kappa) \mid n > 0, A \to X_1 \cdots X_n \in P \\ &\text{and } (A, \bot)^{(m)}(q_1(x_1), \ldots, q_m(x_m)) \to_\kappa q((a, r)^{(m)}(x_1, \ldots, x_m)) \in R\} \\ \cup\ &\{((a, r)^{(0)}, (b, q), \kappa) \mid b \in T \text{ and } (b, \bot)^{(0)}() \to_\kappa q((a, r)^{(0)}()) \in R\} \\ \cup\ &\{((w_t, \#)^{(0)}, (w_t, q), \kappa) \mid (w_t, \#)^{(0)}() \to_\kappa q((w_t, \#)^{(0)}()) \in R\}\ .\end{aligned}$$

We define the deduction rules for a weighted tree automaton $\mathcal{A} = \langle Q, \Sigma, \delta, F\rangle$ (over $\mathbb{M}$) and an input word $w = a_1 \cdots a_n$ where items are of form $\langle i, q, \kappa, j\rangle$ where $0 \leq i < j \leq n$, $q \in Q$ and $\kappa \in \mathbb{R}$:

$$\frac{(a^{(0)}, q, \kappa) \in \delta \quad a_i = a}{\langle i-1, q, \kappa, i\rangle}\ (\mathsf{Scan})$$

$$\frac{(q_1, \ldots, q_m, a^{(m)}, q, \kappa) \in \delta \quad \langle i_0, q_1, \kappa_1, i_1 - 1\rangle \cdots \langle i_{m-1}, q_m, \kappa_m, i_m\rangle}{\langle i_0, q, \kappa + \sum_{i=1}^m \kappa_i, i_m\rangle}\ (\mathsf{Complete})$$

with a **best first** chart management policy: if two items $\langle i, q, \kappa, j\rangle$ and $\langle i, q, \kappa', j\rangle$ are added to the chart, keep the one with maximal weight. The objective is any item $\langle 0, q, \kappa, n\rangle$ with $q \in F$; κ will be the weight of its derivation.

For the second technique, we start directly with (more complex) derivation rules, where items are of form $\langle i, X, q, \kappa, j\rangle$ with $0 \leq i < j \leq n$, $X \in N \cup T$, $q \in Q$, and $\kappa \in \mathbb{R}$:

$$\frac{a_i = a \quad (a, \bot)^{(0)}() \to_\kappa q((b, r)^{(0)}()) \in R}{\langle i-1, a, q, \kappa, i\rangle}\ (\mathsf{Scan})$$

$$\frac{a_i = (w_t, \#) \quad (w_t, \#)^{(0)}() \to_\kappa q((w_t, \#)^{(0)}()) \in R}{\langle i-1, w_t, q, \kappa, i\rangle}\ (\#)$$

$$\frac{\begin{array}{c} A \to X_1 \cdots X_m \in P \quad (A, \bot)^{(m)}(q_1(x_1), \ldots, q_m(x_m)) \to_\kappa q((a, r)^{(m)}(x_1, \ldots, x_m)) \in R \\ \langle i_0, X_1, q_1, \kappa_1, i_1 - 1\rangle \cdots \langle i_{m-1}, X_m, q_m, \kappa_m, i_m\rangle \end{array}}{\langle i_0, A, q, \kappa + \sum_{i=1}^m \kappa_i\rangle}\ (\mathsf{Complete})$$

again with a best first chart management policy: if two items $\langle i, X, q, \kappa, j\rangle$ and $\langle i, X, q, \kappa', j\rangle$ are added to the chart, then only the one with maximal weight is kept. The objective is any item $\langle 0, S, q, \kappa, n\rangle$ with $q \in F$.

[4] 3. **Bonus question:** we mentioned in class that we could reformulate the Viterbi algorithm as the search of a shortest path in a graph as performed by Dijkstra's algorithm. Can you think about a scheme for ordering the processing of items produced by your parser such that it will make it find the best parse as quickly as possible by expressing the weighted parsing problem as a shortest path problem?

References

Gildea, D. and Jurafsky, D., 2002. Automatic labelling of semantic roles. *Computational Linguistics*, 28(3):245–288. doi:10.1162/089120102760275983.