Exam: Abstract Categorial Grammars

Duration: 3 hours.
Written documents are allowed. The numbers in front of questions are indicative of hardness or duration. Please put your answers to sections 1 and 2 on separate sheets; do not forget to write your name on both.

Quick Course Recap. Recall from the course that a higher-order linear signature is a triple $\Sigma = \langle A, C, \tau \rangle$ where $A$ is a finite set of atomic types, $C$ is a finite set of constants, and $\tau: C \rightarrow T(A)$ is a function that assigns each constant in $C$ to a linear implicative type $\alpha$ built over $A$, according to the syntax

$$\alpha ::= a | \alpha \rightarrow \alpha$$

where $a$ ranges over $A$. By convention we consider $\rightarrow$ to be right-associative, i.e. we write $\alpha \rightarrow \beta \rightarrow \gamma$ for $\alpha \rightarrow (\beta \rightarrow \gamma)$. The order of a linear type is defined inductively as

$$\text{ord}(a) = 1 \quad \text{ord}(\alpha \rightarrow \beta) = \max(\text{ord}(\alpha) + 1, \text{ord}(\beta)).$$

Given a higher-order linear signature $\Sigma$, each linear lambda term of $\Lambda^o(\Sigma)$ can be assigned a type in $T(A)$ by the typing system

$$\begin{array}{l}
\Gamma, x : \alpha \vdash \Sigma x : \alpha \quad \text{(Var)} \\
\Gamma \vdash \Sigma \lambda x. t : \alpha \rightarrow \beta \quad \text{(Abs)} \\
\Gamma, t : \alpha \rightarrow \beta, \Delta \vdash \Sigma u : \alpha \quad \Delta \vdash \Sigma u : \alpha \\
\Gamma, \Delta \vdash \Sigma tu : \beta \quad \text{(App)}
\end{array}$$

Note that $x$ occurs free in $t$ exactly once in (Abs) and the environments $\Gamma$ and $\Delta$ are disjoint in (App).

Given two higher-order linear signatures $\Sigma_1$ and $\Sigma_2$, a linear higher-order homomorphism is generated by two functions $\eta: A_1 \rightarrow T(A_1)$ on types and $\theta: C_1 \rightarrow \Lambda^o(\Sigma_2)$ on constants such that $\Gamma \vdash \Sigma_2 \theta(c) : \eta(\tau_1(c))$ for all $c$ in $C_1$, where $\eta$ and $\theta$ are lifted in a natural way by $\eta(\alpha \rightarrow \beta) = \eta(\alpha) \rightarrow \eta(\beta)$ on the one hand, and $\theta(x) = x$, $\theta(\lambda x. t) = \lambda x. \theta(t)$, and $\theta(tu) = \theta(t)\theta(u)$ on the other hand.

An abstract categorial grammar is a tuple $G = \langle \Sigma_1, \Sigma_2, \mathcal{L}, s \rangle$ where $\mathcal{L}$ is a linear higher-order homomorphism from $\Sigma_1$ to $\Sigma_2$ and $s$ is a distinguished type in $T(A_1)$. The abstract language generated by $G$ is

$$\mathcal{A}(G) = \{ t \in \Lambda^o(\Sigma_1) \mid \vdash_{\Sigma_1} t : s \}$$

while its object language is the image of the abstract language by the homomorphism:

$$\mathcal{L}(G) = \{ t \in \Lambda^o(\Sigma_2) \mid \exists u \in \mathcal{A}(G). t = \mathcal{L}(u) \}.$$
1 Second-Order ACGs and Tree Languages

**Exercise 1** (Ground Lambda Terms). Let be a second-order linear signature, i.e. a signature such that the type of any constant is of form

\[ \tau(c) = a_1 \rightarrow \cdots \rightarrow a_n \rightarrow a_0 \]

for atomic \( a_i \)'s in \( A \). Consider the normalized typing system with a single rule

\[ \frac{\Gamma \vdash t : \tau}{\Gamma \vdash t_1 \cdots t_n : a_0} \quad (\text{App}') \]

We want to show that, for all ground terms \( t \) and atomic types \( a \), \( \vdash \Sigma t : a \) if and only if \( \Gamma \vdash \Sigma t : a \).

1. Show that, if \( \tau(c) = a_1 \rightarrow \cdots \rightarrow a_n \rightarrow a_0 \), 0 \( \leq i \leq n \), and \( \vdash \Sigma t_j : a_j \) for all \( 1 \leq j \leq i \), then \( \vdash \Sigma c t_1 \cdots t_i : a_i+1 \rightarrow \cdots \rightarrow a_n \rightarrow a_0 \). Deduce that \( \Gamma \vdash \Sigma t : a \) implies \( \vdash \Sigma t : a \) if \( t \) is ground and \( a \) atomic.

2. Show that, if \( \Gamma \vdash \Sigma t : \alpha \) for a ground term \( t \) and type \( \alpha \), then \( t = c t_1 \cdots t_i \) for some constant \( c \) with \( \tau(c) = a_1 \rightarrow \cdots \rightarrow a_n \rightarrow a_0 \), some \( 0 \leq i \leq n \), and some ground terms \( t_1, \ldots, t_i \) such that \( \alpha = a_{i+1} \rightarrow \cdots \rightarrow a_n \rightarrow a_0 \) and \( \vdash \Sigma t_j : a_j \) for \( 0 \leq j \leq i \) for some atomic types \( a_j \)'s.

3. Deduce that \( \vdash \Sigma t : a \) implies \( \Gamma \vdash \Sigma t : a \) whenever \( t \) is a ground term and \( a \) an atomic type.

**Exercise 2** (Local Tree Languages). For a second-order constant \( c \) with type \( \tau(c) = a_1 \rightarrow \cdots \rightarrow a_n \rightarrow a_0 \), we call \( n \) its arity (and thus can see \( C \) as a ranked alphabet) and associate to the ground lambda term \( t = c t_1 \cdots t_n \) the unique tree \( \bar{t} = c^{(n)}(t_1, \ldots, t_n) \). Given a second-order signature \( \Sigma \) and a distinguished atomic type \( s \), we define the tree language \( \mathcal{G}(\Sigma, s) = \{ \bar{t} \in T(C) \mid \vdash \Sigma t : s \text{ where } t \text{ is ground} \} \).

1. Consider the second-order linear signature \( \Sigma_0 \) with atomic types \( A_0 = \{np, s, c\} \), constants \( C_0 = \{\text{ALICE, BELIEVE, LEFT, SOMEONE, THAT}\} \), and typing

\[
\begin{align*}
\tau_0(\text{ALICE}) &= np \\
\tau_0(\text{BELIEVE}) &= c \rightarrow np \rightarrow s \\
\tau_0(\text{LEFT}) &= np \rightarrow s \\
\tau_0(\text{SOMEONE}) &= np \\
\tau_0(\text{THAT}) &= s \rightarrow c
\end{align*}
\]

The corresponding ranked alphabet is \( \mathcal{F}_0 = \{\text{ALICE}^{(0)}, \text{BELIEVE}^{(2)}, \text{LEFT}^{(1)}, \text{SOMEONE}^{(0)}, \text{THAT}^{(1)}\} \). Give a tree automaton over \( \mathcal{F}_0 \) for \( \mathcal{G}(\Sigma_0, s) \).
2. Let \( F \) be a ranked alphabet. A deterministic top-down tree automaton \( A = \langle Q, F, \delta, \{ q_0 \} \rangle \) is local if there exists a function \( \ell : F \to Q \) such that the rules in \( \delta \) are all of the form \( (\ell(f(n)), f(n), q_1, \ldots, q_n) \).

Show that, if \( L \) is recognized by a local deterministic top-down tree automaton, then there is a second order linear signature \( \Sigma \) and a distinguished atomic type \( s \) such that \( L = G(\Sigma, s) \).

3. Show that, conversely, given a second-order signature \( \Sigma \) and a distinguished atomic type \( s \), there exists a local top-down deterministic tree automaton \( A \) such that \( L(A) = G(\Sigma, s) \).

4. Give an example of a regular tree language, which cannot be expressed as \( G(\Sigma, s) \) for any second-order linear signature \( \Sigma \) and distinguished atomic type \( s \).

Exercise 3 (Regular Tree Languages). Fix some ranked alphabet \( F \). We define the generic tree signature \( \Sigma_F = \langle \{ \sigma \}, F, \tau_F \rangle \) by \( \tau_F(f(n)) = \sigma \circ \cdots \circ \sigma \circ \sigma = \sigma^n \circ \sigma \).

Let \( G = \langle \Sigma_1, \Sigma_F, L, s \rangle \) be an ACG with \( \Sigma_1 \) a second-order linear signature and \( s \) an atomic type of \( A_1 \). We define the tree language of \( G \) as \( T(G) = \{ \bar{t} \in T(F) \mid \exists t \text{ ground. } \exists u \text{ ground. } \vdash_{\Sigma_1} u : s \land L(u) \rightarrow^* \bar{t} \} \).

1. Give an ACG \( G \) s.t. \( T(G) = \{ f(g(a), g(b)) \} \).

Exercise 4 (Tree Adjoining Languages).

1. Give a TAG \( G \) with word language \( L(G) = \{ a^n b^m c^n d^m \mid n, m \geq 0 \} \).

2. Give an ACG \( G' \) that generates the same trees as your answer to the previous question: \( T(G') = L_T(G) \).

2 ACGs for Semantics

Exercise 5 (Covert Movements and Spurious Ambiguities). Consider again the signature of Exercise 2.1, to which we add a constant QR, i.e., \( \Sigma_0 = \langle A_0, C_0, \tau_0 \rangle \) where:

\[
A_0 = \{ np, s, c \} \quad C_0 = \{ \text{ALICE, BELIEVE, LEFT, SOMEONE, THAT, QR} \}
\]

\[
\begin{align*}
\tau_0(\text{ALICE}) &= np & \tau_0(\text{BELIEVE}) &= c \circ np \circ s \\
\tau_0(\text{LEFT}) &= np \circ s & \tau_0(\text{SOMEONE}) &= np \\
\tau_0(\text{THAT}) &= s \circ c & \tau_0(\text{QR}) &= np \circ (np \circ s) \circ s
\end{align*}
\]
Consider the signatures $\Sigma_1 = (A_1, C_1, \tau_1)$ and $\Sigma_2 = (A_2, C_2, \tau_2)$, which are respectively defined as follows:

$A_1 = \{\sigma\}$  \hspace{1cm}  $C_1 = \{/Alice/, /believes/, /left/, /someone/, /that/\}$

$\tau_1(/Alice/) = \sigma \circ \sigma$  \hspace{1cm}  $\tau_1(/believes/) = \sigma \circ \sigma$

$\tau_1(/left/) = \sigma \circ \sigma$  \hspace{1cm}  $\tau_1(/someone/) = \sigma \circ \sigma$

$\tau_1(/that/) = \sigma \circ \sigma$

$A_2 = \{\iota, o\}$  \hspace{1cm}  $C_2 = \{a, left, B, \exists\}$

$\tau_2(a) = \iota$  \hspace{1cm}  $\tau_2(left) = \iota \circ o$

$\tau_2(B) = \iota \circ o \circ o$  \hspace{1cm}  $\tau_2(\exists) = (\iota \circ o \circ o)$

Finally, define two linear higher-order homomorphisms $L_1$ and $L_2$ as follows:

$L_1(np) = \sigma \circ \sigma$  \hspace{1cm}  $L_1(s) = \sigma \circ \sigma$  \hspace{1cm}  $L_1(c) = \sigma \circ \sigma$

$L_1(ALICE) = /Alice/$  \hspace{1cm}  $L_1(BELIEVE) = \lambda xy. y + /believes/ + x$

$L_1(LEFT) = \lambda x. x + /left/$  \hspace{1cm}  $L_1(SOMEONE) = /someone/$

$L_1(THAT) = \lambda x. /that/ + x$  \hspace{1cm}  $L_1(QR) = \lambda xp. px$

where $a + b$ is defined as $\lambda x. a(bx)$,

$L_2(np) = (\iota \circ o) \circ o$  \hspace{1cm}  $L_2(s) = o$  \hspace{1cm}  $L_2(c) = o$

$L_2(ALICE) = \lambda k. k a$  \hspace{1cm}  $L_2(BELIEVE) = \lambda pk. k(\lambda x. B x p)$

$L_2(LEFT) = \lambda k. k(\lambda x. left x)$  \hspace{1cm}  $L_2(SOMEONE) = \lambda k. \exists x. k x$

$L_2(THAT) = \lambda x. x$  \hspace{1cm}  $L_2(QR) = \ldots$

1. Show that the two following terms

$t_1 = \text{BELIEVE} (\text{THAT} (\text{LEFT SOMEONE})) \text{ ALICE}$

$t_2 = \text{QR} \text{ SOMEONE} (\lambda x. \text{BELIEVE} (\text{THAT} (\text{LEFT} x)) \text{ ALICE})$

get the same interpretation under $L_1$.

1. Compute $L_2(t_1)$.

2. Define $L_2(QR)$ in such a way that $L_2(t_2)$ yields the de re interpretation (i.e., the interpretation where the existential quantifier takes wide scope over the modal operator).

3. Show that there is an infinity of terms $u_0, u_1, u_2, \ldots$ such that:

$L_1(u_i) = /Alice/ + /believes/ + /that/ + /someone/ + /left/$