1 Course description

This course focuses on computational aspects of grammar logics. In a previous course, we have presented grammatical derivations in the Gentzen calculus and in the Natural Deduction format. From a computational point of view, these formats are problematic. Natural Deduction is good for the presentation of proofs once they have been found, but it doesn’t offer pleasant proof search facilities. Gentzen calculus does have a nice ‘backward chaining’ proof search interpretation, but it is hopelessly inefficient as a result of spurious non-determinism: there can be different derivations that compute one and the same meaning.

EXAMPLE. Irrelevant choices in the order of application of rules. Hit $\rightarrow$ for an alternative we’d like to prune.
The proof-net approach of Linear Logic has attractive properties for natural language processing. Proof nets are graphs satisfying certain well-formedness conditions. You can view a proof net as a ‘parallelized’ derivation: nets capture the essence of a derivation without redundant choice points. The original well-formedness conditions for proof nets check for resource multiplicity: every resource has to be used exactly once. The challenge, in adapting the nets for NLP purposes, is to introduce additional constraints to also check the relevant aspects of grammatical structure. One way of doing that is to decorate the nodes in a proof structure graph with terms taken from an algebra of structural labels. Alternatively, one can build in the structural factors directly in the graph-theoretic characterization of nets.

Proof nets form the subject of the first part of the course (Week 1 to Week 3). In the second part of the course, we look at some alternative computational regimes that can be used for fragments of the full grammar logic. We investigate the trade-off between expressivity and computational efficiency.

2 Sessions

2.1 Week 1. Basics

We introduce the basic concepts of proof nets for L. We consider formulas with polarities: antecedent (·)• polarity (‘given’) versus succedent (·)◦ polarity (‘to prove’). We compute the formula decomposition tree for arbitrary formulas with the following unfolding rules.

\[
\begin{align*}
(A)\cdot & \quad (B)\circ \\
(A/B)^\cdot & \quad (A/B)^\circ \\
(B/A)^\cdot & \quad (B/A)^\circ \\
(A\cdot B)^\cdot & \quad (A\cdot B)^\circ \\
(B\cdot A)^\cdot & \quad (B\cdot A)^\circ \\
(B\setminus A)^\cdot & \quad (B\setminus A)^\circ \\
\end{align*}
\]

Notice that there are two types of links (⊗, ⊕) and that the order of the subtypes is inverted in the (·)◦ unfolding:

- ⊗-type (‘tensor’) links: cf. the two-premise sequent rules /L, \L, •R
To build a proof net for a sequent \( A_1, \ldots, A_n \Rightarrow B \), proceed as follows:

1. Build a candidate proof structure. A proof structure is obtained by taking the formula decomposition trees \( (A_1)^* \ldots (A_n)^* (B)^\circ \) together with an axiom linking. An axiom linking is a pairwise matching of leaves (literals, atomic formulas) with opposite polarities.

2. Check whether the proof structure is in fact a proof net by testing the correctness criteria on its correction graphs. A correction graph is obtained from a proof structure by removing exactly one edge from every \( \oplus \) link. A proof structure is a proof net iff every correction graph for it is
   - a-cyclic and connected (linear wellformedness)
   - planar (no crossing axiom links — Lambek wellformedness, non-commutativity).

**Exercise 1.**

Give the formula decomposition trees for the Lambek formulas below, both for input \((\cdot)^*\) and for output \((\cdot)^\circ\) polarities. Don’t forget: the order of the premises is significant!

1. \( s/(np\backslash s) \)
2. \( (s/np)\backslash s \)
3. \( (n\backslash n)/(s/np) \)
4. \( (np\backslash s)/(np^* ap) \)

**Exercise 2.**

Now the reverse. Below a number of formula decomposition trees. Construct the (polarised) Lambek formulas that type these trees.

\[
\begin{align*}
(s)^* & \quad (s)^\circ \quad (np)^* \\
\downarrow & \quad ? \\
\otimes & \quad \oplus
\end{align*} & \\
\begin{align*}
(s)^\circ & \quad (s)^* \quad (np)^\circ \\
\downarrow & \quad ? \\
\oplus & \quad \otimes
\end{align*}
\]

\[
\begin{align*}
(np)^\circ & \quad (s)^* \\
\downarrow & \quad ? \\
\otimes & \quad \oplus
\end{align*} & \\
\begin{align*}
(s)^\circ & \quad (np)^* \\
\downarrow & \quad ? \\
\oplus & \quad \otimes
\end{align*}
\]

**Exercise 3.**
The proof structures for the sequents below have only one possible axiom linking. Are these proof structures in fact proof nets? If not, which condition is/are violated?

\((\dag)\) \(a \bullet (b/c) \Rightarrow (a \bullet b)/c\) \(\quad (\ddagger)\) \((a \bullet b)/c \Rightarrow a \bullet (b/c)\)

**Exercise 4.**

Below the formula decomposition trees for the sequent \((\dag)\). (Dotted lines for cotensor links.) How many axiom linkings can you find that turn this into a proof net? Which axiom linkings lead to violations of the correctness criteria?

\[(\ddagger)\] \(s/(np\ \atop \!\!\!\!\!\!s), (s/(np\ \atop \!\!\!\!\!\!s)\ \atop \!\!\!\!\!\!s) \Rightarrow \ s\)

---

### 2.2 Week 2. Proof nets and semantic readings

We present two ways of computing the semantic readings (expressed as \(\lambda\) terms) for proof nets:

- **algebra** labelling the vertices of a net with (partially instantiated) \(\lambda\) terms plus unification/matching at the axioms — the *static/declarative* method;
- **geometry** assembling the \(\lambda\) term via a traversal of an oriented version (the *dynamic graph*) of a proof net — the *dynamic/procedural* method.

Recall first the syntax of the \(\lambda\) calculus, and the correspondence between inference rules in the sequent calculus and the operations pairing/projection and abstraction/application.

\[\phi, \psi \rightarrow x\] atoms

| \(\langle \phi, \psi \rangle\) | pairing, constructor \(\bullet\) |
| \(\phi_0 \mid \phi_1\) | projections, destructors \(\bullet\) |
| \(\lambda x. \phi\) | abstraction, constructor implication |
| \(\phi \psi\) | application, destructor implication |
2.2.1 The algebraic method: labelling

Formula decomposition with $\lambda$ term decoration. Notation: $x, y, z$ ($t, u, v$) for object-level variables (terms), $M, N$ for meta-level variables. Newly introduced object-level variables and metavariables in the rules below are chosen fresh.

\[
\frac{(t \ M) : (A)^* \ M : (B)^o}{t : (A/B)^*} \otimes x : (B)^* \ N : (A)^o \oplus
\]
\[
\frac{M : (B)^o \ (t \ M) : (A)^*}{t : (B\setminus A)^*} \otimes N : (A)^o \ x : (B)^* \oplus
\]
\[
\frac{(t)_0 : (A)^* \ (t)_1 : (B)^*}{t : (A \bullet B)^*} \oplus N : (B)^o \ M : (A)^o \oplus
\]

AXIOM LINKS. (One-sided) unification/matching of the unknown $M$ at the output node with the term $t$ at the input node.

\[
\{M := t\}
\]

To compute the lambda term for a net:

1. assign fresh (object-level) variables to the terminal input formulas (including hypotheses!);
2. build the (partially instantiated) terms that go with the formula decomposition links;
3. compose the matchings found at the axiom links.

2.2.2 Illustration

Term decorated net for Lifting: $np \Rightarrow s/(np\setminus s)$. (Hit $\rightarrow$ for the axiom links and substitutions.)

\[
\begin{array}{c}
\text{AXIOM LINK} \\
0 - 1 \quad 2 - 3
\end{array}
\]\n
\[
\begin{array}{c}
\text{SUBSTITUTION} \\
M := x \\
N := (y \ M)
\end{array}
\]
EXERCISE 5.
Provide typing for the formula decomposition trees below. Build two proof nets over them, and work out the term decoration for these nets.

\[
\begin{align*}
\text{Exercise 5.} \\
\text{Provide typing for the formula decomposition trees below. Build two proof} \\
\text{nets over them, and work out the term decoration for these nets.}
\end{align*}
\]

\[\begin{align*}
\text{2.2.3 The geometric method: dynamic graphs}
\end{align*}\]

The *dynamic graph* specifies a set of travel instructions for the traversal of a proof net. For axiom links, the information flow is from input to output polarities. For the logical links, we have oriented edges as follows:

\[
\begin{align*}
(B)^o & \quad (A)^* \\
(A)^o & \quad (B)^* \\
\end{align*}
\]

De Groote-Retoré travel instructions (for the implicational fragment):

1. enter at the (unique) output conclusion;
2. follow the ascending path of output polarities until an axiom link is reached; crossing cotensor edges corresponds to $\lambda$ ABSTRACTIONS;
3. cross the axiom link from output to input vertex;
4. follow the descending path of input polarities; crossing tensor edges corresponds to successive APPLICATIONS; the path ends on
   - an input conclusion: free head variable of the corresponding term;
   - the input premise of a cotensor link: head variable bound by the $\lambda$ of the $\oplus$ link;
5. repeat (1)-(4) from every output premise of the $\otimes$ links encountered in step (4); this produces the arguments to which the head variable is applied.

2.2.4 Illustration

The dynamic graph for $s/(np\backslash s), np\backslash s, (np\backslash s)\backslash (np\backslash s) \Rightarrow s$ (‘Everybody listens carefully’). You have to provide the typing of the graph yourself! Hitting $\rightarrow$, you’ll see in turn the dynamic switches, and the stepwise visit of the vertices in the graph. (After stepping through this movie a couple of times, you may prefer to jump immediately to Exercise (6) or to §2.3 . . .)

Exercise 6.

Display the dynamic graph of §2.2.4 as a tree (growing downward). Collapse the input and output vertices of axiom links. Compute the corresponding $\lambda$ term ‘bottom up’ as follows:

- initialize the graph by decorating the leaves with
  - constants $\forall, c_1$ (‘listen’), $c_2$ (‘carefully’) for the leaves corresponding to lexical input formulas;
  - variables $x_1, x_2$ for the hypothetical inputs;
- compute complex terms for the mother nodes by performing
– application at the branching nodes;
– abstraction at the non-branching nodes

2.3 Week 3. Proof nets for structured resources

The proof net machinery we have seen so far applies to \( L \), the associative Lambek calculus. We now add extra information to obtain proof nets for the base logic (no associativity), and for logics obtained from the base logic by means of explicit structural postulates. We also extend the formula language with unary connectives: \( \Diamond, \Box \) — operators for structural control.

2.3.1 A language of structural labels

\[
\sigma, \tau \rightarrow x \quad \text{atoms}
\]

\[
\langle \sigma \rangle \quad \text{constructor } \Diamond
\]

\[
\lceil \sigma \rceil \quad \text{destructor } \Diamond
\]

\[
\lfloor \sigma \rfloor \quad \text{goal } \Box
\]

\[
(\sigma \circ \tau) \quad \text{constructor } \bullet
\]

\[
\langle (\sigma) \rangle \quad \text{left-destructor } \bullet
\]

\[
\langle (\sigma) \rangle \quad \text{right-destructor } \bullet
\]

\[
x \backslash \sigma \quad \text{goal } \backslash
\]

\[
\sigma / x \quad \text{goal } /
\]

2.3.2 Proof nets for the base logic

Nets are decorated with structural labels:

- Inputs (conclusions and hypotheticals): atomic labels \( x \)

- Output literals: meta-level structure label variables

- Exit nodes of \( \otimes \) links: structure labels \( \langle \sigma \rangle, (\sigma \circ \tau) \)

- Exit nodes of \( \oplus \) links: auxiliary labels

As with the \( \lambda \) term labeling, at an axiom link, we match the unknown at the output vertex with the structure label decorating the input vertex. As an extra condition, we require that the structure label computed for the net be normal, in the sense that all auxiliary labels must reduce. The labeling rules for logical links and the reduction relation \( \succ \) are defined on the next pages.
2.3.3 Residuation reductions: binary connectives

Notation: \(x, y, z \ (t, u, v)\) for object-level formula (structure) labels, \(\Gamma, \Delta\) for meta-level structure label variables.

\[
\frac{(t \circ \Delta) : (A)^\bullet}{t : (A/B)^\bullet} \otimes \frac{\Delta : (B)^\circ}{\Gamma/x : (A/B)^\circ} \oplus \\
(t \circ x)/x \succ t
\]

\[
\frac{\Delta : (B)^\circ \quad (\Delta \circ t) : (A)^\bullet}{t : (B \setminus A)^\bullet} \otimes \frac{\Gamma : (A)^\circ}{x \setminus (B \setminus A)^\circ} \oplus \\
x \setminus (x \circ t) \succ t
\]

\[
\frac{(t \circ (A)^\bullet \quad (t)^\circ : (B)^\bullet)}{t : (A \bullet B)^\bullet} \oplus \frac{\Delta : (B)^\circ \quad \Gamma : (A)^\circ}{(\Gamma \circ \Delta) : (A \bullet B)^\circ} \otimes \\
((t \circ (t)^\circ) \succ t
\]

2.3.4 Residuation reductions: unary connectives

\[
\frac{[t] : (A)^\bullet}{t : (\Diamond A)^\bullet} \oplus \frac{\Gamma : (A)^\circ}{(\Gamma) : (\Diamond A)^\circ} \otimes \\
([t]) \succ t
\]

\[
\frac{\langle t \rangle : (A)^\bullet}{t : (\Box A)^\bullet} \otimes \frac{\Gamma : (A)^\circ}{[\langle t \rangle] : (\Box A)^\circ} \oplus \\
[\langle t \rangle] \succ t
\]

The reduction relation captures the asymmetry of derivability. Compare the theorem (†) with the non-theorem (‡)

\[
(\dagger) A \rightarrow \Box A \quad (\ddagger) \Box A \rightarrow A
\]

\[
\frac{\Delta : (A)^\circ}{\langle \Delta \rangle : (\Diamond A)^\circ} \otimes \frac{[\langle x \rangle] : (A)^\bullet}{\langle x \rangle} \oplus \frac{\langle \langle x \rangle \rangle : (\Diamond A)^\circ}{\langle \langle x \rangle \rangle} \oplus \\
(x : (A)^\bullet \quad 1 \quad \langle \Delta \rangle : (\Diamond A)^\circ \quad \langle \langle x \rangle \rangle : (\Diamond A)^\circ \quad \Gamma : (A)^\circ \quad 4
\]

Matching 1 2 gives \(\Delta := x\). For the goal type \((\Box \Diamond A)^\circ\) we get

\[
\langle \langle \Delta \rangle \rangle = \langle \langle x \rangle \rangle \succ x
\]

In the case of (†), matching 3 4 yields

\[
\Gamma := \langle \langle x \rangle \rangle \quad \text{but} \quad \langle \langle x \rangle \rangle \not\succ x
\]
**2.3.5 Structural reasoning**

Structural postulates are translated in extra clauses for the $\succ$ relation. The general pattern: a postulate $A \rightarrow B$ corresponds to a clause $\sigma(B) \succ \sigma(A)$, where $\sigma(\cdot)$ is the structural label translation of a formula.

Here is an example: the Distributivity postulate $K$:

<table>
<thead>
<tr>
<th>POSTULATE</th>
<th>REDUCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Diamond(A \cdot B) \rightarrow \Diamond A \cdot \Diamond B$</td>
<td>$((\Gamma) \circ (\Delta)) \succ (\langle \Gamma \circ \Delta \rangle$</td>
</tr>
</tbody>
</table>

**2.3.6 Illustration**

$\Box(A/B) \Rightarrow \Box A/\Box B$

\[
\begin{array}{c}
\frac{(x) \circ \Delta : (A)^\bullet 1 \quad \Delta : (B)^\circ 2}{(x) : (A/B)^\bullet} \\
\frac{y : (B)^\bullet 3 \quad \Gamma : (A)^\circ 4}{y : (\Box B)^\bullet} \quad \frac{\boxed{\Gamma}/y : (\Box A/\Box B)^\circ}{\boxed{\Gamma}/y : (\Box A/\Box B)^\circ}
\end{array}
\]

\[
\Delta = \{y\} \quad \text{(Axiom link 3 2)} \\
\Gamma = ((x) \circ \Delta) \quad \text{(Axiom link 1 4)} \\
\Gamma = ((x) \circ \{y\}) \\
\succ \langle (x \circ y) \rangle \\
\Gamma = [\langle (x \circ y) \rangle / y] \\
\Gamma / y = (x \circ y) / y \\
\Gamma / y = x
\]

**Exercise 7.**

Consider the Dutch relative clause below (literally: ‘who(m) Alice teases’, as in ‘de jongen die Alice plaagt’/‘the boy who(m) Alice teases’):

\[
\text{die Alice plaagt}
\]

Subordinate clauses (type sb) in Dutch have the verb in final position, following its complements — compare ‘ik weet dat Alice Tweedledum plaagt’ versus ‘I know that Alice teases Tweedledum’. Unlike the English gloss, the Dutch relative clause example is ambiguous: the relative pronoun ‘die’ could relate to the subject of ‘tease’ (the boy is the one who teases Alice) or to the direct object (Alice being the one who teases the boy).

Give the formula decomposition trees for the relative clause example, and give two different ways of matching literals that express this ambiguity. The nets...
should respect the acyclicity and connectedness criteria, but you can violate planarity. Which reading (subject or object interpretation of ‘die’ is derivable in the base logic?

Exercise 8.

We now introduce a structural postulate \((P)\) that makes the direct object interpretation derivable.

\[
(P) \quad \Diamond A \bullet (B \bullet C) \rightarrow B \bullet (\Diamond A \bullet C)
\]

Give the translation of this postulate in a reduction clause for \(\succ\). Work out the structural labeling for the formula decomposition trees in the previous example. Take the literal matching for the direct object interpretation, and show that the label you can compute for \((r)^{o}\) can be reduced to the structure below, with the elementary clauses for \(\succ\) plus the clause for \((P)\).

\[
die \circ (Alice \circ plaagt)
\]
Solutions to Exercises

Exercise 1. Putting the $(\cdot)^*$ and the $(\cdot)^0$ unfoldings next to each other nicely brings out the symmetry. In fact you can see these unfoldings as proof frames for the sequents below. You can turn these proof frames into nets by adding the (planar) axiom links for the literals with opposite polarity.

\[
\begin{align*}
(1) & \quad s/(s/np) \Rightarrow s/(s/np) \\
(2) & \quad (s/np)s \Rightarrow (s/np)s \\
(3) & \quad (n\ n)/(s/np) \Rightarrow (n\ n)/(s/np) \\
(4) & \quad (np\ s)/(np\ ap) \Rightarrow (np\ s)/(np\ ap)
\end{align*}
\]

Here are the formula trees.

Exercise 1

Exercise 2.
Exercise 3. (†) Proof structure for $a \bullet (b/c) \Rightarrow (a \bullet b)/c$. This is in fact a proof net: every possible setting of the $\oplus$ switches (dotted lines) produces an acyclic, connected and planar graph.

We attempt a proof structure for (‡). Hit $\rightarrow$ for a setting of the $\oplus$ switches that makes the structure cyclic, and that disconnects the $b$ axiom.

(‡) $(a \bullet b)/c \Rightarrow a \bullet (b/c)$

This is not a net, and indeed there is no sequent derivation for (‡). $\bullet R$ is not applicable: it requires one to split up the antecedent in two non-empty parts, but there is only one formula. $\vdash/L$ is not applicable either: there is no material to the right of the antecedent formula to produce the $c$.
Exercise 4. Hit \( \rightarrow \) for an axiom linking that in fact produces a proof net: every correction graph is in fact acyclic, connected and planar.

THERE IS MORE . . . Hit \( \rightarrow \) for a second matching of literals that obeys the correctness criteria.

It’s easy to violate the correctness criteria. Below two ways of producing a cycle. You can add yourself some crossing axiom linkings violating planarity.
Exercise 5. Labels for input formulas \((x, y, z, u)\), and label propagation to the literals. You can read off the matching of the unknowns at the output vertices from the axiom links (hit \(\rightarrow\)). The second solution is on the next page.

The second solution:

\[
N := (u \, \lambda z. (x \, \lambda y. (z \, y)))
\]
Exercise 6. You can use \( \rightarrow \) and \( \leftarrow \) to (de)compose the term.

Exercise 7. Here are the formula decomposition trees for the lexical inputs. We number the leaves for the next exercise, with 8 for the goal formula \((r)^{n}\).
Literal matching. On the right the subject interpretation for the relative pronoun, which gives rise to a planar net and is in fact derivable in the base logic.

Exercise 7

Exercise 8. The $\succ$ clause for $(P)$ is

$$\Delta_1 \circ (\langle \Delta_2 \circ \Delta_3 \rangle \succ \langle \Delta_2 \rangle \circ (\Delta_1 \circ \Delta_3))$$

Labeling: for the lexical inputs, we use the words themselves as labels, for the $\Diamond \Box np$ hypothetical input, we use the atomic label $x$.

Inputs:  
1. die $\circ (x|K)$  
3. $\langle [x] \rangle$  
4. Alice  
7. $M \circ (L \circ plaagt)$

Outputs:  
2. $K$  
5. $L$  
6. $M$  
8. $G$

Axiom links:

1 $\sim$ 8  
7 $\sim$ 2  
4 $\sim$ 6  
3 $\sim$ 5

$G := \text{die} \circ (x|K)$  
$K := M \circ (L \circ plaagt)$  
$M := \text{Alice}$  
$L := \langle [x] \rangle$
Reduction of $G$. Notice that an eager reduction $\langle \lfloor x \rfloor \rangle \triangleright x$ as the first step would make the structural rewriting $(P)$ unapplicable.

$$G := \text{die} \circ (x \backslash (\text{Alice} \circ (\langle \lfloor x \rfloor \rangle \circ \text{plaagt}))) \triangleright (P)$$

$$\text{die} \circ (x \backslash (\langle \lfloor x \rfloor \rangle \circ (\text{Alice} \circ \text{plaagt}))) \triangleright$$

$$\text{die} \circ (x \backslash (x \circ (\text{Alice} \circ \text{plaagt}))) \triangleright$$

$$\text{die} \circ (\text{Alice} \circ \text{plaagt})$$

Exercise 8