Parsing Lambek calculus using partial composition

Denis Béchet

Denis.Bechet@univ-nantes.fr

LINA, University of Nantes
Introduction

Lambek languages: Context-free languages [Pentus 92]

⇒ CYK parser: $n^3$ (n = nb of words)

Lambek calculus: NP-complete [Pentus 03]

⇒ proof search: $C^n$ (n = nb of primitive types)

Parsing of “whom have you seen ?”

whom have you seen

$Q'/(Q/NP) \quad Q/P_2/\Pi_2 \quad \Pi_2 \quad P_2/NP$

Motivation: Lambek calculus with a bounded size on types is $n^3$ (n = nb of primitive types)
Parsing with word separation

Parsing of “whom have you seen?”

whom have you seen

\[ Q'/(Q/NP) \quad Q/P_2/\Pi_2 \quad \Pi_2 \quad P_2/NP \]

We use:

- Flat Interfaces of Modules of Proof-Nets for partial compositions
- Outerplanar graph properties to find a CYK algorithm
PLAN

- Introduction
- Background
  - Lambek calculus
  - Module and proof net
  - Interface calculus
- Parsing with word separation
  - Rewriting
  - Partial composition
  - Majority partial composition
- Conclusion
Lambek calculus

Categorial Grammars

Each word is associated to one or more types:

- **Primitive types** \( Pr \):  
  - \( N \) for common nouns “cat”, “mice”  
  - \( NP \) for noun phrases or names “John”, “Mary”  
  - \( S \) for correct sentences

- **Composed types** \( Tp ::= Pr | Tp/Tp | Tp\backslash Tp | Tp \cdot Tp \):  
  - \( NP/N \) for articles “a”, “the”, “this”  
  - \( NP\backslash S \) for intransitive verbs (3rd person) “sleeps”  
  - \( (NP\backslash S)/NP \) for transitive verbs (3rd person) “eats”  
  - \( S/(NP\backslash S) \) for pronouns “he”, “she” and “it”

**Notation** \([\text{word} \mapsto \text{type}] : [a \mapsto NP/N]\)
Lambek calculus

Derivation rules for Lambek calculus $\Gamma \vdash A$

1. **Cut**
   \[ \frac{\Gamma, A, \Gamma' \vdash C \quad \Delta \vdash A}{\Gamma, \Delta, \Gamma' \vdash C} \]

2. **Ax**
   \[ \frac{}{A \vdash A} \]

3. **/L**
   \[ \frac{\Gamma \vdash A \quad \Delta, B, \Delta' \vdash C}{\Delta, B/A, \Gamma, \Delta' \vdash C} \]

4. **/R**
   \[ \frac{\Gamma, A \vdash B}{\Gamma \vdash B/A} \]

5. **\L**
   \[ \frac{\Gamma \vdash A \quad \Delta, B, \Delta' \vdash C}{\Delta, \Gamma, A\setminus B, \Delta' \vdash C} \]

6. **\R**
   \[ \frac{A, \Gamma \vdash B}{\Gamma \vdash A\setminus B} \]

7. **\L**
   \[ \frac{\Delta, A, B, \Delta' \vdash C}{\Delta, A \bullet B, \Delta' \vdash C} \]

8. **\R**
   \[ \frac{\Gamma \vdash A \quad \Gamma' \vdash B}{\Gamma, \Gamma' \vdash A \bullet B} \]
Lambek calculus

“a cat” is a noun phrase

- We have: \([a \mapsto NP/N]\) and \([cat \mapsto N]\)
- Let us prove that \([a \mapsto NP/N], [cat \mapsto N] \vdash NP:\)

\[
\begin{align*}
N & \vdash N & \text{Ax} \\
NP & \vdash NP & \text{Ax} \\
NP / N, N & \vdash NP & /L
\end{align*}
\]

/
“he eats” is equivalent to $S/NP$

We have: $[he \mapsto S/(NP\backslash S)]$ and $[eats \mapsto (NP\backslash S)/NP]$

Let us prove that $S/(NP\backslash S), (NP\backslash S)/NP \vdash S/NP$:

\[
\begin{align*}
S \vdash S & \hspace{1cm} \text{Ax} \hspace{1cm} \text{Ax} \hspace{1cm} \text{Ax} \\
NP\backslash S \vdash NP\backslash S & \hspace{1cm} NP \vdash NP \\
S/(NP\backslash S), NP\backslash S \vdash S & \hspace{1cm} \vdash L \\
S/(NP\backslash S), (NP\backslash S)/NP \vdash S & \hspace{1cm} \vdash L \\
S/(NP\backslash S), (NP\backslash S)/NP \vdash S/NP & \hspace{1cm} \vdash R
\end{align*}
\]
Proof-nets

Links

\( A^- \quad B^- \quad B^+ \quad A^+ \)

\( (A \bullet B)^- \quad (A \bullet B)^+ \)

\( A^+ \quad B^- \quad B^+ \quad A^- \)

\( (A \setminus B)^- \quad (A \setminus B)^+ \quad (A \backslash B)^- \quad (A \backslash B)^+ \)

Ax

\( A^+ \quad A^- \)

Cut
Proof-nets

Proof-net: module that corresponds to a sequential proof in Lambek calculus.

Remark: Not every module without hypothesis is a proof-net \(\Rightarrow\) We need a correctness criterion
Proof-nets

Switching links

\[
\begin{align*}
&A \otimes B \\
\end{align*}
\]

\[
\begin{align*}
&A \bowtie B \\
\end{align*}
\]

\[
\begin{align*}
&A \bowtie B \\
\end{align*}
\]
Proof-nets

Switching links

Theorem [Girard, Danos&Regnier, Roorda]. A planar module without hypothesis is a proof-net iff for every switching, the graph is acyclic and connected.
Proof-nets

A proof-net corresponding to “Peter saw briefly Mary dancing”
Two orthogonal modules = a splitting of a proof-net in two parts

Equivalent modules:

\[ M_1 \equiv M_2 \] iff they have the same orthogonal modules
Two equivalent modules
Modules and interfaces

Interface links

$k$-ary $\otimes$-link and one of its switching positions
Modules and interfaces

From module links to interface

Ax

Cut
Modules and interfaces

From module to interface

The natural interface of a module
Interface normalization

From interfaces to flat interfaces

\[ \Gamma = \emptyset \text{ No other edge} \]

\[ \begin{align*}
\n\Gamma_1 & \quad \ldots \quad \Gamma_n \\
\Gamma & \quad \rightarrow \\
\end{align*} \]

\[ \begin{align*}
\n\Gamma_1 & \quad \ldots \quad \Gamma_n \\
\Delta_1 & \quad \ldots \quad \Delta_k \\
\Gamma & \quad \rightarrow \\
\end{align*} \]

\[ \begin{align*}
\n\Gamma_1 & \quad \ldots \quad \Gamma_n \\
\Delta_1 & \quad \ldots \quad \Delta_k \\
\Delta & \quad \rightarrow \\
\end{align*} \]
Interface normalization

Flat interface normalization

![Diagram of interface normalization with nodes and arrows representing propagation, merging, and completion.](image)
Theorems

Theorem: Normalisation is confluent and terminating

Theorem: Two modules are equivalent iff their normalized flat interfaces are equal

Remark: There exists only a finite number of (normalized) flat interfaces corresponding to a given frontier
Key transformation

\[ \begin{array}{c}
\Gamma_1 \\
\Gamma_2 \\
\Delta_1, \Delta_2 \\
\end{array} \quad \xrightarrow{\text{splitting}} \quad \begin{array}{c}
\Gamma_1 \\
\Gamma_2 \\
\Delta_1 \\
\Delta_2 \\
\end{array} \]
Flat interface calculus

2 operations:

- $I|J$: $I$ and $J$ are juxtaposed

- for $I[\ldots, A, A^\bot, \ldots]$, if the addition of an axiom on $A$ and $A^\bot$ gives a correct interface, we note $I[\ldots, \bullet, \bullet, \ldots]$ the flat interface given after normalization
$v_1 \cdots v_n \in \mathcal{L}(G)$ iff for $1 \leq i \leq n$, $\exists I_i \in I(v_i)$ such that

$I_1 | \cdots | I_n \xrightarrow{(1)^*} S$

$(1)^* :$ the reflexive and transitive closure of $(1):$

$I[A_1, \ldots, A_{i-1}, B, B^\perp, A_{i+2}, \ldots, A_n] \xrightarrow{(1)}$

$I[A_1, \ldots, A_{i-1}, \bullet, \bullet, A_{i+2}, \ldots, A_n]$
Parsing with word separation (list)

Three rewriting rules \((\Gamma, \Delta \in \mathcal{I}^*, I, J \in \mathcal{I}, A, B, A_i \in Pr)\):

- **[M]** (merge): \(\Gamma, I, J, \Delta \xrightarrow{M} \Gamma, I \mid J, \Delta\).

- **[I]** (internal):
  \[
  \begin{align*}
  \Gamma, I[\ldots, A, A^\perp, \ldots], \Delta & \xrightarrow{I} \Gamma, I[\ldots, \bullet, \bullet, \ldots], \Delta, \\
  \end{align*}
  \]

- **[C\_k]** (k-partial composition):
  \[
  \begin{align*}
  \Gamma, I[\ldots, A_k, \ldots, A_1], J[A_1^\perp, \ldots, A_k^\perp, \ldots], \Delta & \xrightarrow{C_k} \\
  \Gamma, I \mid J[\ldots, \bullet, \ldots, \bullet, \bullet, \ldots, \bullet, \ldots], \Delta,
  \end{align*}
  \]

Remark: merge = 0-partial composition
Lemma: Parsing can be done using $(MIC_k)^*$:

$v_1 \cdots v_n \in \mathcal{L}(G)$ iff for $1 \leq i \leq n$, $\exists I_i \in I(v_i)$ such that

$I_1, \cdots, I_n \xrightarrow{(MIC_k)^*} S$
Internal before Merge/Composition

Lemma: $\rightarrow I$ can be performed before $\rightarrow M$ and $\rightarrow C_k$:

$v_1 \cdots v_n \in \mathcal{L}(G)$ iff

for $1 \leq i \leq n$, $\exists I_i \in I(v_i)$, $\exists Y_i \in \mathcal{I}$ such that:

\[
\begin{align*}
\begin{cases}
\text{for } 1 \leq i \leq n, I_i & \xrightarrow{I^*} J_i \\
J_1, \ldots, J_n & \xrightarrow{(MC_k)^*} S
\end{cases}
\end{align*}
\]

Remark: Partial composition does not give a polynomial parsing algorithm because the result of partial composition is not bounded by the lexicon:

$\implies$ We need a restricted partial composition
Majority partial composition

A partial composition $C_k \rightarrow$ is a majority partial composition (noted $\rightarrow@$ or $@_k \rightarrow$) if the width of the result is less or equal to the maximum of the widths of the arguments.

A non majoritory partial composition:

$$\Gamma, I[Q', O, Q^\perp], J[Q, P_2^\perp, \pi_2^\perp], \Delta \xrightarrow{C_1} \Gamma, I[Q', O, \bullet], J[\bullet, P_2^\perp, \pi_2^\perp], \Delta$$

A majoritory partial composition:

$$\Gamma, I[Q', O, Q^\perp], J[Q, O^\perp, \pi_2^\perp], \Delta$$

$$\xrightarrow{\@_2} \Gamma, I[Q', \bullet, \bullet], J[\bullet, \bullet, \pi_2^\perp], \Delta$$
Parsing using majority composition

Main theorem:
\( v_1 \cdots v_n \in \mathcal{L}(G) \) iff for \( 1 \leq i \leq n \), \( \exists I_i \in \mathcal{R}_{I^*}(I)(v_i) \) such that:

\[
I_1, \cdots, I_n \xrightarrow{a^*} S
\]

where \( \mathcal{R}_{I^*}(I) \) is the completion of \( I \) by \( I^* \)

Proof: Property of outerplanar graphs applied to planar modules. There exists a conclusion in \( \Gamma \) that is only linked to its immediate neighbour(s)
Lemma: In an outerplanar graph, there exists a vertex that is only connected to its immediate neighbour(s)
Proof-nets

Proof-nets are graphical representations of logical proofs. They are used in proof theory to represent the structure of proofs in a more visual and intuitive way. In this diagram, we have a proof-net representing the sentence "Peter saw Mary dancing briefly.

The diagram shows the logical structure of the sentence, with nodes representing logical connectives and links representing the flow of information. The labels on the edges correspond to the words in the sentence, and the structure of the proof-net reflects the logical relationships between these words.

The proof-net for this sentence is constructed as follows:

1. NP (Peter)
2. (NP \ S) / VP (saw)
3. (S / VP) \ (S / VP) (briefly)
4. NP \ (NP / VP) (Mary)
5. NP / (NP / VP) (dancing)
6. S

The proof-net shows how the sentence is constructed from its constituent parts, with the logical structure reflecting the order and relationships of the words in the sentence.
Polynomial parsing using @

Partial composition gives a polynomial parsing algorithm because the size of the result of majority partial composition is bounded by the maximum width of the types of the lexicon.

For a grammar $G$ and a list of words $v_1, \cdots, v_n \in \Sigma^+$, we compute for $1 \leq i \leq j \leq n$, $\mathcal{I}^G_{v_1,\cdots,v_n}(i,j) \subset \mathcal{I}$, the possible interfaces associated to the sublist of words $v_i, \cdots, v_j$ using majority partial composition:

\[
\begin{align*}
    i = j : & \quad \mathcal{I}^G_{v_1,\cdots,v_n}(i,j) = \mathcal{R}_{I^*}(I)(v_i) \\
    i < j : & \quad \mathcal{I}^G_{v_1,\cdots,v_n}(i,j) = \bigcup_{k=i}^{j-1} \left\{ Z \mid \exists X \in \mathcal{I}^G_{v_1,\cdots,v_n}(i,k) \quad \exists Y \in \mathcal{I}^G_{v_1,\cdots,v_n}(k+1,j) \quad X, Y \xrightarrow[@]{} Z \right\}
\end{align*}
\]
Polynomial parsing using \( @ \)

\[ v_1, \ldots, v_n \in \mathcal{L}(G) \iff S \in I^G_{v_1, \ldots, v_n}(1, n) \]

Algorithm:

1. Search the flat interfaces associated by \( G \) to each word
2. Add the flat interfaces deduced by \( I^* \) (internal axioms)
3. Compute recursively the possible types associated to a contiguous segment of words of the string using \( @ \)
4. Look at \( \bullet[S] \) in the final set of flat interfaces
Conclusion

- Polynomial parsing algorithm using majority partial composition of the flat interfaces associated to the words of a string
- Need a completion of the lexicon with types deduced using “internal” rewriting