Symmetric categorial grammar

Monday

Raffaella Bernardi & Michael Moortgat

Abstract

We study an extended version of the categorial base logic. In addition to the Lambek connectives (product, left and right division), it has a family of dual residuated connectives (coproduct, left and right difference). In linear logic, these are related by De Morgan duality. For linguistic purposes, a more subtle interaction is required, preserving the individual characteristics of the operators involved. The general framework of Grishin (1983) provides such interaction principles.

We investigate the relation of type similarity (aka conjoinability) for the resulting system **LG** (bi-Lambek calculus with Grishin's Type IV interactions). We show that **LG** similarity can be characterized in terms of an interpretation in the free Abelian group generated by the atomic types. This means that with respect to similarity, **LG** recovers the expressivity of **LP** (Pentus 1993), without loss of structural discrimination.

We discuss how the similarity relation can be used in the analysis of phenomena beyond the reach of **NL** and **L**.

Joint work with Mati Pentus (Moscow University).

Contents

1. Plan for today

- Two relations between types: derivability, similarity
- Characterizing **LG** similarity
- Linguistic applications:
  - long distance dependencies
  - crossed dependencies
- Comparison: LTAG embedding (Moot 2007)

2. The argument

Type similarity (Lambek 1958, notation $A \sim B$) is the reflexive, symmetric, transitive closure of the derivability relation.

Example (subject GQ) $s/(np\backslash s) \sim ((np\backslash s)/np)\backslash(np\backslash s)$ (object GQ).

- For associative and/or commutative Lambek calculi (**L**, **LP**), expressivity wrt $\sim$ is inversely proportional to structural discrimination.
- In the symmetric Lambek-Grishin calculus **LG**, the expressivity of **LP** is obtained in a structure-preserving way.

3. Type equivalence

Definition

$A \sim B$ iff $\exists\ C_1 \ldots C_n$ s.t. $C_1 = A$, $C_n = B$ and $\forall\ i{<}n$, $C_i \vdash C_{i+1}$ or $C_{i+1} \vdash C_i$

Diamond property $A \sim B$ iff one of the following equivalent statements holds

- $\exists C$ such that $A \vdash C$ and $B \vdash C$ (join)
- $\exists D$ such that $D \vdash A$ and $D \vdash B$ (meet)

$$
\begin{tikzcd}
 & D \arrow[dl] \arrow[dr] & \\
A \arrow[dr] & & B \arrow[dl] \\
 & C &
\end{tikzcd}
$$

4. Solutions for the diamond property

Lambek (1958) has a solution covering **NL** as well as **L**. The choice between lifting/lowering creates the desired derivational ambiguity. $|C| = |D| = 7$.

$$D = (A/((C/C)\backslash C)) \otimes ((C/C)\backslash B), \qquad C = (A \otimes (D\backslash D))/(B\backslash(D \otimes (D\backslash D)))$$

For **L**, one has a simpler solution with $|C| = |D| = 5$ (Pentus 93). The possibility of rebracketing the types for D and C is what makes this solution work.

$$D = (A/C) \otimes C \otimes (C\backslash B), \qquad C = (D/A)\backslash D/(B\backslash D)$$

In **LG**, we recover a length 5 solution, relying on the Grishin interactions.

$$D = (A/C) \otimes (C \oslash (B \obslash C)), \qquad C = ((D/B)\backslash D) \oplus (D \obslash A)$$

Remark $(A/-) \otimes (- \oslash (B \obslash -)) \xleftrightarrow{\infty} ((-/B)\backslash -) \oplus (- \obslash A)$.

5. Computing the meets and joins

Meet Given join c, d is a meet type for a and b:

$$
\dfrac{a \vdash a \qquad \dfrac{\dfrac{\dfrac{b \vdash c \quad c \vdash c}{(c \obslash c) \vdash (b \obslash c)}\,\obslash}{c \vdash (c \oplus (b \obslash c))}\,\blacktriangleleft'}{(c \oslash (b \obslash c)) \vdash c}\,\blacktriangleright}{\dfrac{(a/c) \vdash (a/(c \oslash (b \obslash c)))}{\underbrace{((a/c) \otimes (c \oslash (b \obslash c)))}_{d} \vdash a}\,\triangleright'}\,/
\qquad
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{b \vdash b \quad a \vdash c}{(b \obslash a) \vdash (b \obslash c)}\,\obslash}{a \vdash (b \oplus (b \obslash c))}\,\blacktriangleleft' \qquad c \vdash c}{(a/c) \vdash ((b \oplus (b \obslash c))/c)}\,/}{((a/c) \otimes c) \vdash (b \oplus (b \obslash c))}\,\triangleright'}{(((a/c) \otimes c) \oslash (b \obslash c)) \vdash b}\,\blacktriangleright}{\underbrace{((a/c) \otimes (c \oslash (b \obslash c)))}_{d} \vdash b}\,\ogreaterthan
$$

Join Given meet d, c is a join type for a and b. Take the ∞ image of the above.

$$
a \vdash \underbrace{((d/b)\backslash d) \oplus (d \obslash a)}_{c} \qquad b \vdash \underbrace{((d/b)\backslash d) \oplus (d \obslash a)}_{c}
$$

6. Models

Quasigroups (Foret 03) In **NL**, type equivalence coincides with equality in the free quasigroup generated by the atomic types, i.e. $A \sim B$ iff $[\![A]\!] =_{FQG} [\![B]\!]$.

Quasigroup equations A quasigroup is a set equiped with operations $/, \cdot, \backslash$ satisfying the equations below.

$$
\begin{array}{ll}
(x/y) \cdot y = x & y \cdot (y\backslash x) = x \\
(x \cdot y)/y = x & y\backslash(y \cdot x) = x
\end{array}
$$

Quasigroup interpretation $[\![p]\!] = p$, $[\![A/B]\!] = [\![A]\!]/[\![B]\!]$, $[\![B\backslash A]\!] = [\![B]\!]\backslash[\![A]\!]$, $[\![A \otimes B]\!] = [\![A]\!] \cdot [\![B]\!]$.

Groups (Pentus 93) In **L** $\sim$ coincides with equality of an interpretation of types in the free group generated by the atomic types (free *Abelian* group for **LP**).

Group interpretation $[\![p]\!] = p$, $[\![A \otimes B]\!] = [\![A]\!] \cdot [\![B]\!]$, $[\![A/B]\!] = [\![A]\!] \cdot [\![B]\!]^{-1}$, $[\![B\backslash A]\!] = [\![B]\!]^{-1} \cdot [\![A]\!]$.

7. Beyond NL

Restructuring In **NL**, $(a\backslash b)/c$ and $a\backslash(b/c)$ are incomparable since

$$[\![(a\backslash b)/c]\!] \neq_{FQG} [\![a\backslash(b/c)]\!]$$

In **L**, these types are comparable as a result of global $\otimes$ associativity.

In **LG**, $(a\backslash b)/c \sim a\backslash(b/c)$, this time without $\otimes$ associativity assumptions.

Reordering In **LP**, $a/b \sim b\backslash a$ as a result of global $\otimes$ commutativity. In **LG**, one obtains this similarity without $\otimes$ commutativity assumptions.

8. Recovering LP similarity in LG

Theorem In **LG**, the following notion of type equivalence obtains:

$$A \sim B \qquad \text{iff} \qquad [\![A]\!] =_{FAG} [\![B]\!]$$

where $=_{FAG}$ is equality in the free Abelian group generated by Atm $\cup \{\oplus\}$.

Proof

- We prove this first for the Frm$(/, \otimes, \backslash)$ fragment (hence, by arrow reversal, also for Frm$(\oslash, \oplus, \obslash)$.
- For the full language, we extend $[\![\cdot]\!]$ to take operator balance into account.

9. Joins for rotation variants

Rotation variants Call ρ variants slash types with the same head, and the same arguments selected with equal directionality. E.g. (with head b at depth 3):

$$\begin{array}{ll} ((a\backslash b)/c)/d & ((a\backslash b)/d)/c \\ (a\backslash (b/c))/d & (a\backslash (b/d))/c \\ a\backslash ((b/c)/d) & a\backslash ((b/d)/c) \end{array}$$

Computing joins for ρ variants The following algorithm shows that ρ variants are $\sim$ relatives. Step 1. Using co-application, expand A° to a $\oplus$ formula with yield $C_1 \ldots C_n$ where $n = \delta(A)$:

$$(\ldots(A^\circ \oplus \underbrace{(A^\circ \oslash A^\circ)) \oplus \cdots \oplus (A^\circ \oslash A^\circ)}_{n-1 \text{ times}} \ldots)$$

Step 2. Divide the factors C_i by the arguments of A° in some fixed order.

10. Example

We compute the join C for $A = (a\backslash b)/c$ and $B = a\backslash(b/c)$ according to the above recipe. $A^\circ = B^\circ = b$. $\delta(A) = \delta(B) = 2$.

$$C = (a\backslash b) \oplus ((b \obslash b)/c)$$

By the diamond property these types then also have a meet D.

$$D = (A/C) \otimes (C \oslash (B \obslash C))$$

Without abbreviations, the solution for D is ...

$$(((a\backslash b)/c)/((a\backslash b) \oplus ((b \obslash b)/c))) \otimes (((a\backslash b) \oplus ((b \obslash b)/c)) \oslash ((a\backslash(b/c)) \obslash ((a\backslash b) \oplus ((b \obslash b)/c))))$$

11. Deriving the join

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{b \vdash b \quad b \vdash b}{(b \obslash b) \vdash (b \obslash b)}\;\obslash}{b \vdash (b \oplus (b \obslash b))}\;\blacktriangleleft'}{a \vdash a \quad (a\backslash b) \vdash (a\backslash(b \oplus (b \obslash b)))}\;\backslash}{(a \otimes (a\backslash b)) \vdash (b \oplus (b \obslash b))}\;\triangleleft'}{((a \otimes (a\backslash b)) \oslash (b \obslash b)) \vdash b}\;\blacktriangleright}{(a \otimes ((a\backslash b) \oslash (b \obslash b))) \vdash b}\;\varogreaterthan}{((a\backslash b) \oslash (b \obslash b)) \vdash (a\backslash b)}\;\triangleleft}{(a\backslash b) \vdash ((a\backslash b) \oplus (b \obslash b)) \qquad c \vdash c}\;\blacktriangleright'}{((a\backslash b)/c) \vdash (((a\backslash b) \oplus (b \obslash b))/c)}\;/}{(((a\backslash b)/c) \otimes c) \vdash ((a\backslash b) \oplus (b \obslash b))}\;\triangleright'}{((a\backslash b) \obslash (((a\backslash b)/c) \otimes c)) \vdash (b \obslash b)}\;\blacktriangleleft}{(((a\backslash b) \obslash ((a\backslash b)/c)) \otimes c) \vdash (b \obslash b)}\;\varolessthan}{((a\backslash b) \obslash ((a\backslash b)/c)) \vdash ((b \obslash b)/c)}\;\triangleright}{((a\backslash b)/c) \vdash ((a\backslash b) \oplus ((b \obslash b)/c))}\;\blacktriangleleft'
$$

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{b \vdash b \quad c \vdash c}{(b/c) \vdash (b/c)}\;/}{b \vdash b \quad ((b/c) \otimes c) \vdash b}\;\triangleright'}{(b \obslash ((b/c) \otimes c)) \vdash (b \obslash b)}\;\obslash}{((b \obslash (b/c)) \otimes c) \vdash (b \obslash b)}\;\varolessthan}{(b \obslash (b/c)) \vdash ((b \obslash b)/c)}\;\triangleright}{a \vdash a \quad (b/c) \vdash (b \oplus ((b \obslash b)/c))}\;\blacktriangleleft'}{(a\backslash(b/c)) \vdash (a\backslash(b \oplus ((b \obslash b)/c)))}\;\backslash}{(a \otimes (a\backslash(b/c))) \vdash (b \oplus ((b \obslash b)/c))}\;\triangleleft'}{((a \otimes (a\backslash(b/c))) \oslash ((b \obslash b)/c)) \vdash b}\;\blacktriangleright}{(a \otimes ((a\backslash(b/c)) \oslash ((b \obslash b)/c))) \vdash b}\;\varogreaterthan}{((a\backslash(b/c)) \oslash ((b \obslash b)/c)) \vdash (a\backslash b)}\;\triangleleft}{(a\backslash(b/c)) \vdash ((a\backslash b) \oplus ((b \obslash b)/c))}\;\blacktriangleright'
$$

12. Neutral types

As in **L**, for arbitrary types A, B we have $A\backslash A \sim B/B$.

- Join type for **L**: $A\backslash((A \otimes B)/B)$ (Pentus 1993)
- Join type for **LG**: $(A\backslash((A \otimes B) \oslash B)) \oplus (B/B)$

The **LG** join is derived from the **L** formula by expanding the head and dividing by A, B (cf ρ variants):

$$
\begin{array}{lccc}
\text{(expand)} & (A \otimes B) & \vdash & ((A \otimes B) \oslash B) \oplus B \\
\text{(divide)} & \underline{A}\backslash((A \otimes B)/\underline{B}) & \vdash & (\underline{A}\backslash((A \otimes B) \oslash B)) \oplus (B/\underline{B})
\end{array}
$$

13. Symmetry

As in **LP**, for arbitrary types A, B we have $B\backslash A \sim A/B$. This time, we provide a meet type, i.e. an X such that $X \vdash B\backslash A$ and $X \vdash A/B$, which by Res means

$$B \otimes X \vdash A \qquad \text{and} \qquad X \otimes B \vdash A$$

Let us put $X := Y \oslash Z$ and solve for

$$B \otimes (Y \oslash Z) \vdash A \qquad \text{and} \qquad (Y \oslash Z) \otimes B \vdash A$$

which by Grishin mixed associativity or commutativity follows from

$$B \otimes Y \vdash A \oplus Z \qquad \text{and} \qquad Y \otimes B \vdash A \oplus Z$$

Solution $Z := (A \obslash B)$; Y the meet for C the join of $B\backslash B$ and B/B, i.e.

$$Y := ((b/b)/C) \otimes (C \oslash ((b\backslash b) \obslash C))$$

$$C := ((b\backslash((b \otimes b) \oslash b)) \oplus (b/b)))$$

14. Linguistic application

LG can analyse non-cf phenomena which, in the asymmetric Lambek calculi, require postulates that violate structure-preservation. Examples:

- Moortgat GEOCAL'06: extraction (local, and non-local via bridge predicates); crossed dependencies
- Moot 2007: copy language, counting, crossed dep's via LTAG simulation

The $\sim$ relation can be used to lexically encapsulate derivational ambiguity:

- For types satisfying $A \sim B$, lexically assign a meet type D.
- Depending on the context, D will derivationally behave as A or B
- Contrast: non-derivational meets/joins ($\cap, \cup$, Lam61, Kanazawa 92, &c)

Below an analysis of long distance filler-gap dependencies, combining the techniques of MM 06 and Moot 07.

15. Extraction

Consider relative clauses. In **NL**, subject extraction is available if one assigns the type $(n\backslash n)/(np\backslash s)$ to the relative pronoun.

...song which irritates Molly

For object extraction, the type $(n\backslash n)/(s/np)$ is useless:

...song which Molly (detests | dedicated to Leopold)

- to reach the object of a simple transitive verb $(np\backslash s)/np$, one would need associativity;
- to reach the non-peripheral object of a dative verb $((np\backslash s)/pp)/np$, one also needs a form of reordering.

Challenge Can we use $\sim$ to simulate $\diamond$ controlled extraction for a relpro assignment $(n\backslash n)/(s/\diamond\Box np)$?

16. Chameleon words

Strategy (First try) To potential gap selectors, we assign an **LG** type from which the original Lambek type is derivable.

preposition	$(pp \oslash (pp/np)) \obslash ((pp/np) \otimes np)$
transitive verb	$((np\backslash s) \oslash ((np\backslash s)/np)) \obslash (((np\backslash s)/np) \otimes np)$
	$(A \oslash (A/np)) \obslash ((A/np) \otimes np)$

Derivational behaviour The **LG** assignment adapts to its derivational context:

- Gap-free context: lowering to A/np since $\oslash, \obslash$ are dual Galois connected:

$$(A \oslash B) \obslash C \vdash B \quad \text{if} \quad C \vdash A$$

- In the presence of a gap, a saturated phrase $(A/np) \otimes np$ is left in the $\otimes$ context whereas $A \oslash (A/np)$ is sent to the rhs

17. Meet type for $\sim$

Let us write $A^{(B)}$ for $(A \oslash (A/B)) \obslash ((A/B) \otimes B)$. For the clause "which Molly thinks of highly" we now have the following schematic situation:

$$\dfrac{\dfrac{\overbrace{np \quad iv/pp \quad pp/np \quad np \quad iv\backslash iv}^{s} \vdash (pp \oslash (pp/np)) \quad (s/np)}{np \quad iv/pp \qquad pp^{(np)} \quad iv\backslash iv \ \vdash \qquad s/np} \qquad n\backslash n \vdash n\backslash n}{(n\backslash n)/(s/np) \quad np \quad iv/pp \quad pp^{(np)} \quad iv\backslash iv \ \vdash n\backslash n}$$

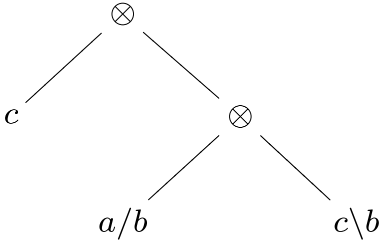
The derivation can proceed if we replace $pp \oslash (pp/np)$ in the $pp^{(np)}$ lexical assignment to prepositions by a meet type D for the similarity pair below:

$$\begin{array}{rccc} & pp \oslash (pp/np) & \sim & s \oslash (s/np) \\ \text{in general:} & A \oslash (A/B) & \sim & C \oslash (C/B) \end{array}$$

Exercise Compute a small type for D.

18. Dutch crossed dependencies

Pattern '(dat Jan) boeken wil lezen' with the order object–modal auxiliary–transitive infinitive



Challenge As with extraction, there is a double challenge:

- one wants to allow the transitive infinitive $c\backslash b$ to 'see' its direct object c across the intervening modal auxiliary a/b;
- one has to rule out the ungrammatical order $(a/b) \otimes (c \otimes (c\backslash b))$ which with the indicated types would make b derivable.

19. Crossed dependencies (cont'd)

To bring the $\sim$ strategy into play, note that

$$c\backslash b \overset{\text{(lifting)}}{\sim} c\backslash((a/b)\backslash a) \overset{\text{(rotation)}}{\sim} (a/b)\backslash(c\backslash a)$$

For the original $c\backslash b$ and the rotated $(a/b)\backslash(c\backslash a)$ we have join C and meet D:

$$C = ((a/b)\backslash a) \oplus (c\backslash(a \oslash a))$$

$$D = ((c\backslash b)/C) \otimes (C \oslash (((a/b)\backslash(c\backslash a)) \oslash C))$$

To make the ungrammatical order modal auxiliary–object–transitive infinitive underivable, we use modal decoration. We change the type of the modal auxiliary to $a/\Diamond\Box b$, and modify D accordingly, marking the rotated argument:

$$D' = ((c\backslash b)/C) \otimes (C \oslash (((a/\Diamond\Box b)\backslash(c\backslash a)) \oslash C))$$

The join type C can remain as it was since

$$(a/\Diamond\Box b)\backslash(c\backslash a) \vdash ((a/b)\backslash a) \oplus (c\backslash(a \oslash a))$$

20. Comparison: Moot 2007

As an alternative to the $\sim$ analysis of crossed dependencies, we look at the simulation of an LTAG analysis proposed in Moot (2007).

Type assignments We write v' for $(v/i) \otimes i$ (or $\Diamond\Box v$), so that $v' \longrightarrow v$.

	de nijlpaarden	np
a	zag (saw)	$(v \oslash (np\backslash(np\backslash s))) \obslash v'$
b	helpen (help)	$v\backslash((v \oslash (np\backslash v')) \obslash v')$
c	voeren (feed)	$v\backslash((v \oslash (np\backslash v')) \obslash v')$

Key & lock The assignments enforce a rule application order:

- before making $\obslash$ of (a) active, two np have to be sent to the rhs
- we can unlock (b) with $v' \longrightarrow v$
- before making $\obslash$ of (b) active, a np has to be sent to the rhs
- & c . . .

21. What have you learned today?

- type similarity is a measure for the derivational strength of a logic
- in **LG** we find the expressivity of **LP**
- this expressivity is obtained in a structure-preserving way
- meet/join types for $\sim$ can be used to lexically encapsulate derivational ambiguities
- LTAG analyses can be simulated in **LG**

Open question Complexity: $\mathbf{LG}_{\emptyset}$ has a polynomial recognition problem (Capelletti 07), **LG** with Class IV interactions recognizes non-cf patterns — where exactly does it fit in the complexity hierarchy?

22. More to Explore

- Chapter 6 of the course materials is the Moortgat & Pentus (2007) paper for FG 2007.
- Pentus (1993) is to be found at http://lpcs.math.msu.su/~pentus/, with many other nice things.
- Annie Foret, in a number of papers, has investigated the use of $\sim$ in connection with formal learning theory for categorial grammars, see her publications at http://www.irisa.fr/prive/foret/