Lambek–Grishin Calculus Extended to Connectives of Arbitrary Aritiy

Matthijs Melissen

Cognitive Artificial Intelligence,
Graduate School of Natural Sciences,
Universiteit Utrecht

February 13, 2009

Outline

Why formal linguistics?

Cognitive Artificial Intelligence:

- Understanding humans leads to better software
- Software leads to better understanding humans

Why formal linguistics?

Cognitive Artificial Intelligence:

- Understanding humans leads to better software
- Software leads to better understanding humans

Within formal linguistics:

- Constrains on human language can be applied in software
- Formal language theory can shine light on human language processing

Lambek calculus **NL**

Set of **types** T:

- p with $p \in \mathit{Atoms}$
- a/b with $a, b \in T$
- $b\backslash a$ with $a, b \in T$
- $a \otimes b$ with $a, b \in T$

Lambek calculus **NL**

Set of **types** T:

- p with $p \in Atoms$
- a/b with $a, b \in T$
- $b\backslash a$ with $a, b \in T$
- $a \otimes b$ with $a, b \in T$

Formulas: $a \longrightarrow b$ with a, b types

Lambek calculus **NL**

Set of **types** T:

- p with $p \in Atoms$
- a/b with $a, b \in T$
- $b\backslash a$ with $a, b \in T$
- $a \otimes b$ with $a, b \in T$

Formulas: $a \longrightarrow b$ with a, b types

Axiom:

$a \longrightarrow a$ (Axiom)

Transitivity:

$$\frac{a \longrightarrow b \qquad b \longrightarrow c}{a \longrightarrow c}$$

Residuation rules:

$$\frac{b \longrightarrow a\backslash c}{a \otimes b \longrightarrow c} \qquad \frac{a \otimes b \longrightarrow c}{a \longrightarrow c/b}$$

Lambek calculus **NL**

Set of **types** T:

- p with $p \in Atoms$
- a/b with $a, b \in T$
- $b\backslash a$ with $a, b \in T$
- $a \otimes b$ with $a, b \in T$

Formulas: $a \longrightarrow b$ with a, b types

Axiom:

$a \longrightarrow a$ (Axiom)

Transitivity:

$$\frac{a \longrightarrow b \qquad b \longrightarrow c}{a \longrightarrow c}$$

Residuation rules:

$$\frac{b \longrightarrow a\backslash c}{\dfrac{a \otimes b \longrightarrow c}{a \longrightarrow c/b}}$$

Example:

$$\frac{a\backslash b \longrightarrow a\backslash b}{a \otimes (a\backslash b) \longrightarrow b}$$

Lambek grammar

The **yield** of a formula

- $\text{yield}(a \otimes b) = \text{yield}(a), \text{yield}(b)$
- $\text{yield}(a) = a$ in other cases

Lambek grammar

The **yield** of a formula

- $\text{yield}(a \otimes b) = \text{yield}(a), \text{yield}(b)$
- $\text{yield}(a) = a$ in other cases

Lambek grammar $\mathcal{L}(\Sigma, D, \varphi)$:

- Σ Terminal symbols
- D Goal symbol
- $\varphi : \Sigma \to \mathcal{P}(T)$ assigns types to terminal symbols

Lambek grammar

The **yield** of a formula

- $\text{yield}(a \otimes b) = \text{yield}(a), \text{yield}(b)$
- $\text{yield}(a) = a$ in other cases

Lambek grammar $\mathcal{L}(\Sigma, D, \varphi)$:

- Σ Terminal symbols
- D Goal symbol
- $\varphi : \Sigma \to \mathcal{P}(T)$ assigns types to terminal symbols

The *language generated by a Lambek grammar* $\mathcal{L}(\Sigma, D, \varphi)$ is the set of expressions $t_1 \dots t_n$ over the alfabet Σ such that there is a derivable formula with yield $b_1 \dots b_n \to D$ such that $b_i \in \varphi(t_i)$ for all $i \leq n$.

Example: Alice sees the house

$$
\begin{aligned}
\varphi(\mathit{Alice}) &= \{np\} \\
\varphi(\mathit{sees}) &= \{(np\backslash s)/np\} \\
\varphi(\mathit{the}) &= \{np/n, np\} \\
\varphi(\mathit{house}) &= \{n\}
\end{aligned}
$$

Example: Alice sees the house

$$\begin{aligned} \varphi(\mathit{Alice}) &= \{np\} \\ \varphi(\mathit{sees}) &= \{(np\backslash s)/np\} \\ \varphi(\mathit{the}) &= \{np/n, np\} \\ \varphi(\mathit{house}) &= \{n\} \end{aligned}$$

With brackets: $\mathit{Alice} \otimes (\mathit{sees} \otimes (\mathit{the} \otimes \mathit{house}))$

Formula: $np \otimes (((np\backslash s)/np) \otimes ((np/n) \otimes n))$

Example: Alice sees the house

$$\begin{aligned} \varphi(\mathit{Alice}) &= \{np\} \\ \varphi(\mathit{sees}) &= \{(np\backslash s)/np\} \\ \varphi(\mathit{the}) &= \{np/n, np\} \\ \varphi(\mathit{house}) &= \{n\} \end{aligned}$$

With brackets: $\mathit{Alice} \otimes (\mathit{sees} \otimes (\mathit{the} \otimes \mathit{house}))$
Formula: $np \otimes (((np\backslash s)/np) \otimes ((np/n) \otimes n))$

$$\frac{}{np \otimes (((np\backslash s)/np) \otimes ((np/n) \otimes n)) \rightarrow s}\ \backslash E$$

Example: Alice sees the house

$$\begin{aligned} \varphi(\mathit{Alice}) &= \{np\} \\ \varphi(\mathit{sees}) &= \{(np\backslash s)/np\} \\ \varphi(\mathit{the}) &= \{np/n, np\} \\ \varphi(\mathit{house}) &= \{n\} \end{aligned}$$

$$\frac{\Gamma \to b \quad \Delta \to b\backslash a}{\Gamma \otimes \Delta \to a}\ \backslash E$$

With brackets: $\mathit{Alice} \otimes (\mathit{sees} \otimes (\mathit{the} \otimes \mathit{house}))$
Formula: $np \otimes (((np\backslash s)/np) \otimes ((np/n) \otimes n))$

$$\dfrac{np \to np \qquad \dfrac{((np\backslash s)/np) \otimes ((np/n) \otimes n) \to np\backslash s}{np \otimes (((np\backslash s)/np) \otimes ((np/n) \otimes n)) \to s}\ \backslash E}{}\ /E$$

Example: Alice sees the house

$$\begin{aligned} \varphi(\mathit{Alice}) &= \{np\} \\ \varphi(\mathit{sees}) &= \{(np\backslash s)/np\} \\ \varphi(\mathit{the}) &= \{np/n, np\} \\ \varphi(\mathit{house}) &= \{n\} \end{aligned}$$

$$\frac{\Gamma \to a/b \quad \Delta \to b}{\Gamma \otimes \Delta \to a}\ /E$$

$$\frac{\Gamma \to b \quad \Delta \to b\backslash a}{\Gamma \otimes \Delta \to a}\ \backslash E$$

With brackets: $\mathit{Alice} \otimes (\mathit{sees} \otimes (\mathit{the} \otimes \mathit{house}))$
Formula: $np \otimes (((np\backslash s)/np) \otimes ((np/n) \otimes n))$

$$\dfrac{np \to np \qquad \dfrac{(np\backslash s)/np \to (np\backslash s)/np \quad \dfrac{}{(np/n) \otimes n \to np}\ /E}{((np\backslash s)/np) \otimes ((np/n) \otimes n) \to np\backslash s}\ /E}{np \otimes (((np\backslash s)/np) \otimes ((np/n) \otimes n)) \to s}\ \backslash E$$

Example: Alice sees the house

$$\begin{aligned} \varphi(\mathit{Alice}) &= \{np\} \\ \varphi(\mathit{sees}) &= \{(np\backslash s)/np\} \\ \varphi(\mathit{the}) &= \{np/n, np\} \\ \varphi(\mathit{house}) &= \{n\} \end{aligned}$$

$$\frac{\Gamma \to a/b \quad \Delta \to b}{\Gamma \otimes \Delta \to a}\ /E \qquad \frac{\Gamma \to b \quad \Delta \to b\backslash a}{\Gamma \otimes \Delta \to a}\ \backslash E$$

With brackets: $\mathit{Alice} \otimes (\mathit{sees} \otimes (\mathit{the} \otimes \mathit{house}))$

Formula: $np \otimes (((np\backslash s)/np) \otimes ((np/n) \otimes n))$

$$\dfrac{np \to np \qquad \dfrac{(np\backslash s)/np \to (np\backslash s)/np \quad \dfrac{np/n \to np/n \;\; n \to n}{(np/n) \otimes n \to np}\ /E}{((np\backslash s)/np) \otimes ((np/n) \otimes n) \to np\backslash s}\ /E}{np \otimes (((np\backslash s)/np) \otimes ((np/n) \otimes n)) \to s}\ \backslash E$$

Lambek–Grishin calculus [Moortgat, 2007]

Set of **types** T: p with
$p \in Atoms$

- a/b
- $b\backslash a$
- $b \otimes a$
- $a \oslash b$
- $b \oslash a$
- $b \oplus a$

Lambek–Grishin calculus [Moortgat, 2007]

Set of **types** T: p with $p \in Atoms$

- a/b
- $b\backslash a$
- $b \otimes a$
- $a \oslash b$
- $b \oslash a$
- $b \oplus a$

Residuation rules:

$$\frac{b \to a\backslash c}{a \otimes b \to c} \qquad \frac{a \otimes b \to c}{a \to c/b}$$

Lambek–Grishin calculus [Moortgat, 2007]

Set of **types** T: p with $p \in Atoms$

- a/b
- $b\backslash a$
- $b \otimes a$
- $a \oslash b$
- $b \oslash a$
- $b \oplus a$

Residuation rules:

$$\frac{b \to a\backslash c}{a \otimes b \to c} \qquad \frac{a \oslash c \to b}{c \to a \oplus b}$$

$$\frac{}{a \to c/b} \qquad \frac{}{c \oslash b \to a}$$

Lambek–Grishin calculus [Moortgat, 2007]

Set of **types** T: p with $p \in Atoms$

- a/b
- $b\backslash a$
- $b \otimes a$
- $a \oslash b$
- $b \oslash a$
- $b \oplus a$

Residuation rules:

$$\frac{b \to a\backslash c}{a \otimes b \to c} \qquad \frac{a \oslash c \to b}{c \to a \oplus b}$$

$$\frac{}{a \to c/b} \qquad \frac{}{c \oslash b \to a}$$

Grishin interactions:

$(a \oslash b) \otimes c \to a \oslash (b \otimes c)$ $\quad$ $a \otimes (b \oslash c) \to (a \otimes b) \oslash c$

$a \otimes (b \oslash c) \to b \oslash (a \otimes c)$ $\quad$ $(a \oslash b) \otimes c \to (a \otimes c) \oslash b$

Motivation generalized **LG**

	Lambek calculus [Lambek, 1958]	

Motivation generalized **LG**

Assymetric (Context-free)	Lambek calculus [Lambek, 1958]	
Symmetric (Mildly context-sensitive)	Lambek–Grishin calculus [Moortgat, 2007]	

Motivation generalized **LG**

	Binary	**Arbitrary arity**
Assymetric (Context-free)	Lambek calculus [Lambek, 1958]	*n*-ary Lambek calculus [Buszkowski, 1986]
Symmetric (Mildly context-sensitive)	Lambek–Grishin calculus [Moortgat, 2007]	

Arbitrary arity: useful for ‘give him the present’ and ‘coffee or tea’

Motivation generalized **LG**

	Binary	**Arbitrary arity**
Assymetric (Context-free)	Lambek calculus [Lambek, 1958]	*n*-ary Lambek calculus [Buszkowski, 1986]
Symmetric (Mildly context-sensitive)	Lambek–Grishin calculus [Moortgat, 2007]	*n*-ary Lambek–Grishin calculus (This work)

Arbitrary arity: useful for ‘give him the present’ and ‘coffee or tea’

The types

Non-atomic types

		Multiplicative	**Implicative**
Binary	**Left**	$a \otimes b$	a/b, $a\backslash b$
	Right	$a \oplus b$	$a \oslash b$, $a \obslash b$

The types

Non-atomic types

Binary

	Multiplicative	Implicative
Left	$a \otimes b$	a/b, $a\backslash b$
Right	$a \oplus b$	$a \oslash b$, $a \obslash b$

Generalized, $n = 2$

	Multiplicative	Implicative
Left	$f_\bullet(a, b)$	$f^1_\rightarrow(a, b)$, $f^2_\rightarrow(a, b)$
Right	$g_\bullet(a, b)$	$g^1_\rightarrow(a, b)$, $g^2_\rightarrow(a, b)$

The types

Non-atomic types

Binary

	Multiplicative	Implicative
Left	$a \otimes b$	$a/b,\ a\backslash b$
Right	$a \oplus b$	$a \oslash b,\ a \obslash b$

Generalized, $n = 2$

	Multiplicative	Implicative
Left	$f_\bullet(a, b)$	$f^1_\rightarrow(a, b),\ f^2_\rightarrow(a, b)$
Right	$g_\bullet(a, b)$	$g^1_\rightarrow(a, b),\ g^2_\rightarrow(a, b)$

Generalized

	Multiplicative	Implicative
Left	$f_\bullet(a_1, \ldots, a_n)$	$f^i_\rightarrow(a_1, \ldots, a_n)$
Right	$g_\bullet(a_1, \ldots, a_n)$	$g^i_\rightarrow(a_1, \ldots, a_n)$

The axioms and rules

Identity:

$$\frac{}{a \longrightarrow a}$$

Transitivity:

$$\frac{a \longrightarrow b \qquad b \longrightarrow c}{a \longrightarrow c}$$

The axioms and rules

Identity:

$$\frac{}{a \to a}$$

Transitivity:

$$\frac{a \to b \qquad b \to c}{a \to c}$$

Residuation rules:

$$\frac{f_\bullet(a_1, \ldots, a_n) \to b}{a_i \to f^i_{\to}(a_1, \ldots, a_{i-1}, b, a_{i+1}, \ldots, a_n)}$$

$$\frac{b \to g_\bullet(a_1, \ldots, a_n)}{g^i_{\to}(a_1, \ldots, a_{i-1}, b, a_{i+1}, \ldots, a_n) \to a_i}$$

The Grishin interactions

Grishin interactions **Gr(i, j)** ($1 \leq i \leq n$, $1 \leq j \leq n$):

$$\frac{g^i_{\rightarrow}(b_1, \ldots, b_{i-1}, f_{\bullet}(a_1, \ldots, a_{j-1}, b_i, a_{j+1}, \ldots, a_n), b_{i+1}, \ldots, b_n) \rightarrow d}{f_{\bullet}(a_1, \ldots, a_{j-1}, g^i_{\rightarrow}(b_1, \ldots, b_n), a_{j+1}, \ldots, a_n) \rightarrow d}$$

The Grishin interactions

Grishin interactions $\mathbf{Gr(i,j)}$ ($1 \leq i \leq n$, $1 \leq j \leq n$):

$$\frac{g^i_{\rightarrow}(b_1, \ldots, b_{i-1}, f_\bullet(a_1, \ldots, a_{j-1}, b_i, a_{j+1}, \ldots, a_n), b_{i+1}, \ldots, b_n) \rightarrow d}{f_\bullet(a_1, \ldots, a_{j-1}, g^i_{\rightarrow}(b_1, \ldots, b_n), a_{j+1}, \ldots, a_n) \rightarrow d}$$

Example: Grishin interaction for $n = 3$, $i = 1$, $j = 2$.

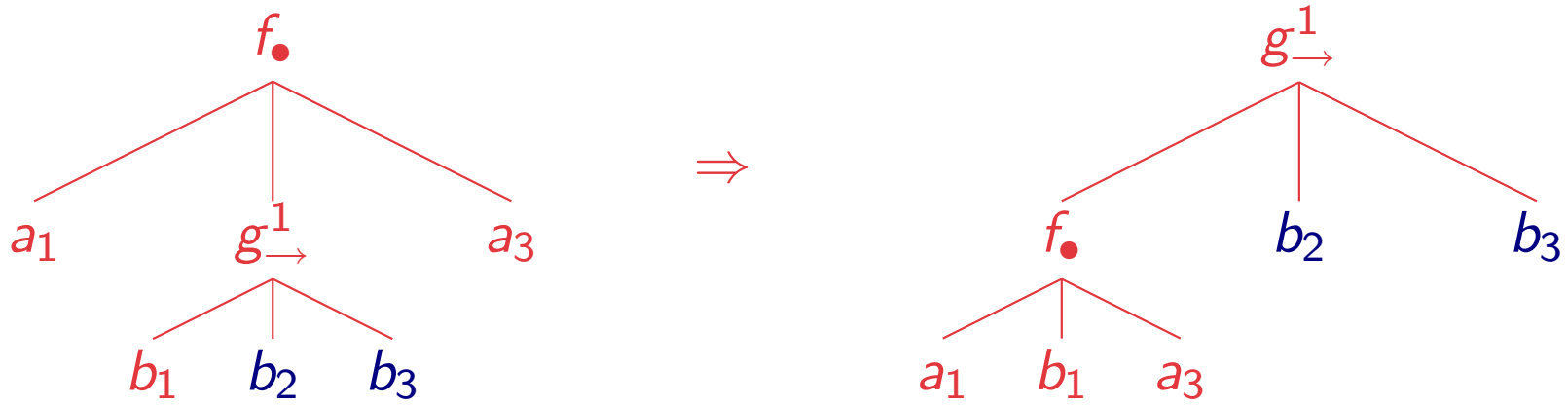
$$\frac{g^1_{\rightarrow}(f_\bullet(a_1, b_1, a_3), b_2, b_3) \rightarrow d}{f_\bullet(a_1, g^1_{\rightarrow}(b_1, b_2, b_3), a_3) \rightarrow d}$$

The Grishin interactions

Grishin interactions **Gr(i, j)** ($1 \leq i \leq n$, $1 \leq j \leq n$):

$$\frac{g^i_{\rightarrow}(b_1, \ldots, b_{i-1}, f_{\bullet}(a_1, \ldots, a_{j-1}, b_i, a_{j+1}, \ldots, a_n), b_{i+1}, \ldots, b_n) \rightarrow d}{f_{\bullet}(a_1, \ldots, a_{j-1}, g^i_{\rightarrow}(b_1, \ldots, b_n), a_{j+1}, \ldots, a_n) \rightarrow d}$$

Example: Grishin interaction for $n = 3$, $i = 1$, $j = 2$.



Properties of the calculus

Advantages:

- Branching of arbitrary order
- At least mildly context-sensitive
- Decidability
- Complete with respect to Kripke semantics
- Derivations can be interpreted using continuation semantics

Properties of the calculus

Advantages:

- Branching of arbitrary order
- At least mildly context-sensitive
- Decidability
- Complete with respect to Kripke semantics
- Derivations can be interpreted using continuation semantics

Disadvantage:

- Hard to find the right types

Generative capacity of **LG**

Chomsky hierarchy:

1. Regular
2. Context-free

4. Context-sensitive
5. Recursive
6. Recursively enumerable

Generative capacity of **LG**

Chomsky hierarchy:

1. Regular
2. Context-free (Too weak for natural language)

4. Context-sensitive (Too strong for natural language)
5. Recursive
6. Recursively enumerable

Generative capacity of **LG**

Chomsky hierarchy:

1. Regular
2. Context-free (Too weak for natural language)
3. Mildly context-sensitive
4. Context-sensitive (Too strong for natural language)
5. Recursive
6. Recursively enumerable

Generative capacity of **LG**

Chomsky hierarchy:

1. Regular
2. Context-free (Too weak for natural language)
3. Mildly context-sensitive
4. Context-sensitive (Too strong for natural language)
5. Recursive
6. Recursively enumerable

Complexity **LG**:

- At least mildly context-sensitive [Moot, 2008]
- At most recursive
- New result: stronger than context-sensitive

Generative capacity of **LG** (2)

Theorem

Any language that is the intersection of a context-free language and the permutation closure of a context-free language can be recognized by **LG**.

Examples:

- $\pi(a^n b^n c^n)$ (permutation closure of context-free language)
- $a^n b^n c^n d^n e^n$ (intersection of $a^i b^j c^k d^l e^m$ and permutation of $(abcde)^n$

Spinal Ajdukiewicz–Bar-Hillel grammar $\mathbf{AB_s}$

$\mathbf{AB_s}$ (spinal Ajdukiewicz–Bar-Hillel-grammar)

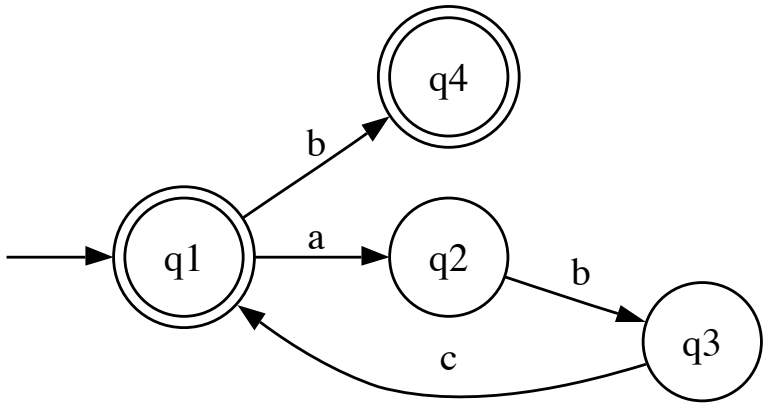
- Types: a and $a \backslash b$ where a and b atoms
- Derivation rule: only $a, (a \backslash b) \rightarrow b$
- Set of goal types instead of one goal type

All derivable sequents have the following form:
$a_0, (a_0 \backslash a_1), (a_1 \backslash a_2), \ldots, (a_{n-1} \backslash a_n) \rightarrow a_n$

Modelling of finite automata in $\mathbf{AB_s}$

Finite automata can be modelled in $\mathbf{AB_s}$

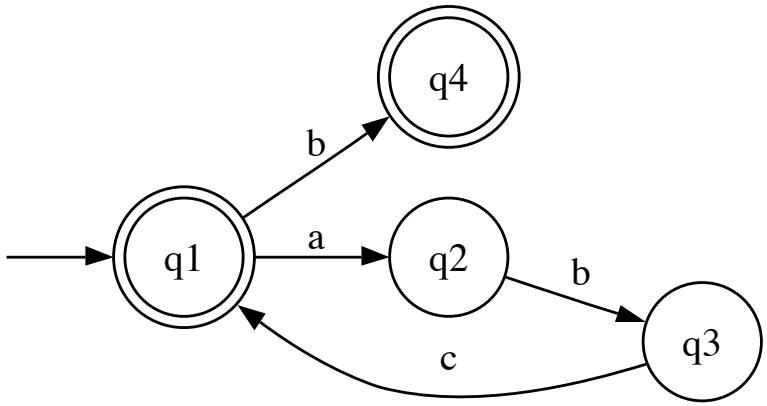
Example:



Modelling of finite automata in $\mathbf{AB_s}$

Finite automata can be modelled in $\mathbf{AB_s}$

Example:

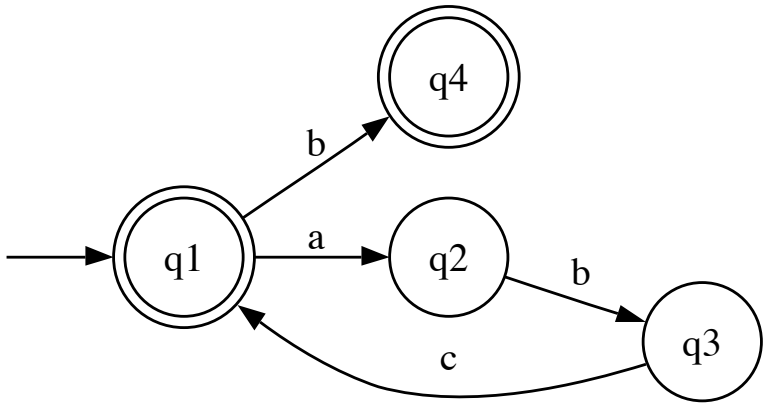


- $\Sigma = \{a, b, c\}$

Modelling of finite automata in $\mathbf{AB_s}$

Finite automata can be modelled in $\mathbf{AB_s}$

Example:

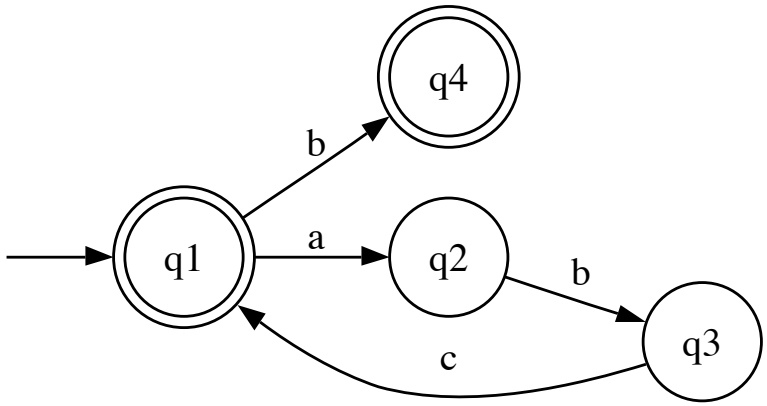


- $\Sigma = \{a, b, c\}$
- $D = \{q1, q4\}$

Modelling of finite automata in $\mathbf{AB_s}$

Finite automata can be modelled in $\mathbf{AB_s}$

Example:



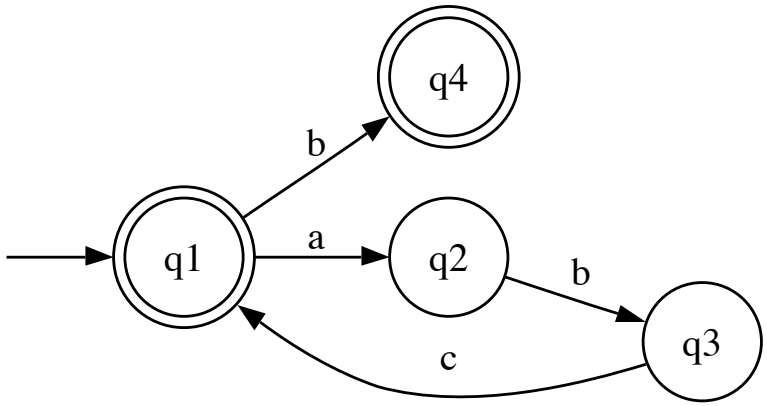
- $\Sigma = \{a, b, c\}$
- $D = \{q1, q4\}$
- φ:

t	$\varphi(t)$
a	$q1 \backslash q2$
b	$q2 \backslash q3$
c	$q3 \backslash q1$

Modelling of finite automata in $\mathbf{AB_s}$

Finite automata can be modelled in $\mathbf{AB_s}$

Example:



- $\Sigma = \{a, b, c\}$
- $D = \{q1, q4\}$
- φ:

t	$\varphi(t)$	
a	$q2$	$q1 \backslash q2$
b	$q4$	$q2 \backslash q3$
c		$q3 \backslash q1$

Conversion of permutation of $\mathbf{AB_s}$ into **LG**

$\mathbf{AB_s}$-grammar

- symbols Σ
- goal types D
- type dictionary φ_1

NL-grammar

- symbols Σ'
- goal types D
- type dictionary φ_2

Conversion of permutation of $\mathbf{AB_s}$ into **LG**

$\mathbf{AB_s}$-grammar

- symbols Σ
- goal types D
- type dictionary φ_1

NL-grammar

- symbols Σ'
- goal types D
- type dictionary φ_2

LG-grammar

- symbols $\Sigma \cap \Sigma'$
- fresh goal type d
- type dictionary φ: see below

$\varphi_1(p)$	$\varphi_2(p)$	$\varphi(p)$
a (atom)	c	$\{(a \oslash d) \oslash c\}$
$a \backslash b$	c	$\{(b \oslash a) \oslash c\}$

Example: permuation of $a^n b^n c^n$

AB_s-grammar

- symbols $\{a, b, c\}$
- goal types $\{s\}$
- type dictionary

  $\varphi_1(a) = \{a, s\backslash a\}$
  $\varphi_1(b) = \{a\backslash b\}$
  $\varphi_1(c) = \{b\backslash s\}$

NL-grammar

- symbols $\{a, b, c\}$
- goal types $\{s\}$
- type dictionary

  $\varphi_2(a) = \{s, s\backslash s\}$
  $\varphi_2(b) = \{s, s\backslash s\}$
  $\varphi_2(c) = \{s, s\backslash s\}$

Example: permuation of $a^n b^n c^n$

AB_s-grammar

- symbols $\{a, b, c\}$
- goal types $\{s\}$
- type dictionary

  $\varphi_1(a) = \{a, s\backslash a\}$
  $\varphi_1(b) = \{a\backslash b\}$
  $\varphi_1(c) = \{b\backslash s\}$

NL-grammar

- symbols $\{a, b, c\}$
- goal types $\{s\}$
- type dictionary

  $\varphi_2(a) = \{s, s\backslash s\}$
  $\varphi_2(b) = \{s, s\backslash s\}$
  $\varphi_2(c) = \{s, s\backslash s\}$

LG-grammar

- symbols $\{a, b, c\}$
- goal type d

Example: permuation of $a^n b^n c^n$

AB_s-grammar

- symbols $\{a, b, c\}$
- goal types $\{s\}$
- type dictionary

  $\varphi_1(a) = \{a, s\backslash a\}$
  $\varphi_1(b) = \{a\backslash b\}$
  $\varphi_1(c) = \{b\backslash s\}$

NL-grammar

- symbols $\{a, b, c\}$
- goal types $\{s\}$
- type dictionary

  $\varphi_2(a) = \{s, s\backslash s\}$
  $\varphi_2(b) = \{s, s\backslash s\}$
  $\varphi_2(c) = \{s, s\backslash s\}$

LG-grammar

- symbols $\{a, b, c\}$
- goal type d

$$\frac{}{\frac{}{\frac{}{(\underbrace{((b \oslash a) \obslash s)}_{\text{b}} \otimes \underbrace{((s \oslash b) \obslash (s\backslash s))}_{\text{c}}) \otimes \underbrace{((a \obslash s) \obslash (s\backslash s))}_{\text{a}} \to d}}}$$

Example: permuation of $a^n b^n c^n$

AB_s-grammar

- symbols $\{a, b, c\}$
- goal types $\{s\}$
- type dictionary

  $\varphi_1(a) = \{a, s\backslash a\}$
  $\varphi_1(b) = \{a\backslash b\}$
  $\varphi_1(c) = \{b\backslash s\}$

NL-grammar

- symbols $\{a, b, c\}$
- goal types $\{s\}$
- type dictionary

  $\varphi_2(a) = \{s, s\backslash s\}$
  $\varphi_2(b) = \{s, s\backslash s\}$
  $\varphi_2(c) = \{s, s\backslash s\}$

LG-grammar

- symbols $\{a, b, c\}$
- goal type d

$$\frac{\dfrac{(a \oslash s) \oslash ((b \oslash a) \oslash ((s \oslash b) \oslash ((s \otimes (s\backslash s)) \otimes (s\backslash s))))) \to d}{(\underbrace{((b \oslash a) \oslash s)}_{b} \otimes \underbrace{((s \oslash b) \oslash (s\backslash s))}_{c}) \otimes \underbrace{((a \oslash s) \oslash (s\backslash s))}_{a} \to d}}{}$$

Example: permuation of $a^n b^n c^n$

AB_s-grammar

- symbols $\{a, b, c\}$
- goal types $\{s\}$
- type dictionary

$\varphi_1(a) = \{a, s\backslash a\}$
$\varphi_1(b) = \{a\backslash b\}$
$\varphi_1(c) = \{b\backslash s\}$

NL-grammar

- symbols $\{a, b, c\}$
- goal types $\{s\}$
- type dictionary

$\varphi_2(a) = \{s, s\backslash s\}$
$\varphi_2(b) = \{s, s\backslash s\}$
$\varphi_2(c) = \{s, s\backslash s\}$

LG-grammar

- symbols $\{a, b, c\}$
- goal type d

$$\frac{\dfrac{\dfrac{}{(s \otimes (s\backslash s)) \otimes (s\backslash s) \to (s \oslash b) \oplus ((b \oslash a) \oplus ((a \oslash d) \oplus d))}}{(a \otimes s) \otimes ((b \oslash a) \otimes ((s \oslash b) \otimes ((s \otimes (s\backslash s)) \otimes (s\backslash s)))) \to d}}{\underbrace{(((b \oslash a) \otimes s)}_{b} \otimes \underbrace{((s \oslash b) \otimes (s\backslash s))}_{c} \otimes \underbrace{((a \otimes s) \otimes (s\backslash s))}_{a} \to d}$$

Example: permuation of $a^n b^n c^n$

AB_s-grammar

- symbols $\{a, b, c\}$
- goal types $\{s\}$
- type dictionary

  $\varphi_1(a) = \{a, s\backslash a\}$
  $\varphi_1(b) = \{a\backslash b\}$
  $\varphi_1(c) = \{b\backslash s\}$

NL-grammar

- symbols $\{a, b, c\}$
- goal types $\{s\}$
- type dictionary

  $\varphi_2(a) = \{s, s\backslash s\}$
  $\varphi_2(b) = \{s, s\backslash s\}$
  $\varphi_2(c) = \{s, s\backslash s\}$

LG-grammar

- symbols $\{a, b, c\}$
- goal type d

$$\dfrac{\dfrac{\dfrac{\dfrac{}{(s \otimes (s\backslash s)) \otimes (s\backslash s) \to (s \oslash b) \oplus ((b \oslash a) \oplus a)}}{(s \otimes (s\backslash s)) \otimes (s\backslash s) \to (s \oslash b) \oplus ((b \oslash a) \oplus ((a \oslash d) \oplus d))}}{(a \oslash s) \otimes ((b \oslash a) \otimes ((s \oslash b) \otimes ((s \otimes (s\backslash s)) \otimes (s\backslash s)))) \to d}}{\underbrace{((\underbrace{(b \oslash a)}_{b} \otimes s)}_{} \otimes \underbrace{((s \oslash b) \otimes (s\backslash s))}_{c}) \otimes \underbrace{((a \otimes s) \otimes (s\backslash s))}_{a} \to d}$$

Example: permuation of $a^n b^n c^n$

AB_s-grammar

- symbols $\{a, b, c\}$
- goal types $\{s\}$
- type dictionary

$\varphi_1(a) = \{a, s\backslash a\}$
$\varphi_1(b) = \{a\backslash b\}$
$\varphi_1(c) = \{b\backslash s\}$

NL-grammar

- symbols $\{a, b, c\}$
- goal types $\{s\}$
- type dictionary

$\varphi_2(a) = \{s, s\backslash s\}$
$\varphi_2(b) = \{s, s\backslash s\}$
$\varphi_2(c) = \{s, s\backslash s\}$

LG-grammar

- symbols $\{a, b, c\}$
- goal type d

$$\dfrac{\dfrac{\dfrac{\dfrac{s \to s}{(s \otimes (s\backslash s)) \otimes (s\backslash s) \to (s \oslash b) \oplus ((b \oslash a) \oplus a)}}{(s \otimes (s\backslash s)) \otimes (s\backslash s) \to (s \oslash b) \oplus ((b \oslash a) \oplus ((a \oslash d) \oplus d))}}{(a \otimes s) \otimes ((b \oslash a) \otimes ((s \oslash b) \otimes ((s \otimes (s\backslash s)) \otimes (s\backslash s)))) \to d}}{\underbrace{(((b \oslash a) \otimes s)}_{b} \otimes \underbrace{((s \oslash b) \otimes (s\backslash s))}_{c}) \otimes \underbrace{((a \otimes s) \otimes (s\backslash s))}_{a} \to d}$$

Proof

- Any language generated by a finite automata can be recognized by some $\mathbf{AB_s}$-grammar
- Any intersection of a permutation of an $\mathbf{AB_s}$-language and an **NL**-language can be recognized by some **LG**-grammar

Proof

- Any language generated by a finite automata can be recognized by some $\mathbf{AB_s}$-grammar
- Any intersection of a permutation of an $\mathbf{AB_s}$-language and an **NL**-language can be recognized by some **LG**-grammar

- Regular languages can be recognized by finite automata
- **CFG** languages are equal to **NL**-languages
- Therefore, any intersection of a permutation of a regular language and a context-free language can be recognized by some **LG**-grammar

Proof

- Any language generated by a finite automata can be recognized by some $\mathbf{AB_s}$-grammar
- Any intersection of a permutation of an $\mathbf{AB_s}$-language and an **NL**-language can be recognized by some **LG**-grammar

- Regular languages can be recognized by finite automata
- **CFG** languages are equal to **NL**-languages
- Therefore, any intersection of a permutation of a regular language and a context-free language can be recognized by some **LG**-grammar

- The permutation of context-free grammars is equal to the permutation of regular grammars
- Therefore, any intersection of a permutation of a context-free grammar and a context-free language can be recognized by some **LG**-grammar

Conclusions and future work

Conclusions

- We extended binary Lambek–Grishin calculus to a calculus allowing for connectives of arbitrary arity
- The calculus has some other good properties, such as decidability, completeness and a connected continuation semantics
- The binary calculus recognizes all languages that are the intersection of a context-free language and the permutation closure of a context-free language
- The generative capacity is slightly more than mildly context-sensitivity and for this reason a good candidate for modelling natural language

Conclusions and future work

Conclusions

- We extended binary Lambek–Grishin calculus to a calculus allowing for connectives of arbitrary arity
- The calculus has some other good properties, such as decidability, completeness and a connected continuation semantics
- The binary calculus recognizes all languages that are the intersection of a context-free language and the permutation closure of a context-free language
- The generative capacity is slightly more than mildly context-sensitivity and for this reason a good candidate for modelling natural language

Future work

- Apply formalism to 'real' (natural) language
- Find upper bound generative complexity

References


Buszkowski, W. (1986).
Bulletin of Polish Academy of Sciences: Mathematics 34, 507–516.


Lambek, J. (1958).
American Mathematical Monthly 65, 363–386.


Moortgat, M. (2007).
In: Proceedings WoLLIC '07, (Leivant, D. and de Quieros, R., eds) pp. 264–284, LNCS 4576. Springer.


Moot, R. (2008).
In: Proceedings of the TAG+ Conference , HAL - CCSD.