Lambek Categorial Grammars
Day 2

Stepan Kuznetsov

Steklov Mathematical Institute, RAS
for ESSLLI '15 in Barcelona

August 4, 2015

Dependent Clauses

$\text{THE} : (e \to t) \to e$
"if a predicate has exactly one object on which it is true, return this object" (partial function)

Dependent Clauses

$\text{THE} : (e \to t) \to e$
"if a predicate has exactly one object on which it is true, return this object" (partial function)

$\text{AND} : t \to (t \to t)$, $\text{OR} : t \to (t \to t)$
(written $u \,\&\, v$, $u \vee v$)

Dependent Clauses

THE : $(e \to t) \to e$
"if a predicate has exactly one object on which it is true, return this object" (partial function)

AND : $t \to (t \to t)$, OR : $t \to (t \to t)$
(written $u \,\&\, v$, $u \vee v$)

Mapping syntactic types to semantic ones:

noun	$n \mapsto (e \to t)$
noun phrase	$np \mapsto e$
sentence	$s \mapsto t$

The Cut Rule

$$\frac{\Gamma \to A \quad \Gamma\, A\, \Delta \to C}{\Gamma\, \Pi\, \Delta \to C}$$

(corresponds to substitution of λ-terms)

Dependent Clauses

The	book	fell.	
$(np \,/\, n)$	n	$(np \,\backslash\, s)$	$\rightarrow s$
THE	BOOK	FALL	

Dependent Clauses

The	book	fell.	
$(np\,/\,n)$	n	$(np\,\backslash\,s)$	$\rightarrow s$
THE	BOOK	FALL	

the	book	which	fell	
$(np\,/\,n)$	n	$(n\,\backslash\,n)\,/(np\,\backslash\,s)$	$np\,\backslash\,s$	$\rightarrow np$
THE	BOOK	$\lambda x^{e\rightarrow t}.\lambda y^{e\rightarrow t}.\lambda z^{e}.(x(z)\,\&\,y(z))$	FALL	

Semantic value: $\text{THE}\,\lambda z.(\text{BOOK}(z)\,\&\,\text{FALL}(z))$

Dependent Clauses

John	loves	Mary.	
np	$(np \backslash s) / np$	np	$\rightarrow s$
JOHN	LOVE	MARY	

Dependent Clauses

John	loves	Mary.	
np	$(np \backslash s) / np$	np	$\to s$
JOHN	LOVE	MARY	

the	girl	which	John	loves	
(np / n)	n	$(n \backslash n) / (s / np)$	np	$(np \backslash s) / np$	$\to np$
THE	GIRL	$\lambda x^{e \to t}.\lambda y^{e \to t}.\lambda z^{e}.(x(z) \,\&\, y(z))$	JOHN	LOVE	

Semantic value: $\textsc{the}\ \lambda z.(\textsc{girl}(z) \,\&\, \textsc{love}(z)(\textsc{john}))$.

Dependent Clauses

John	loves	Mary.	
np	$(np \backslash s) / np$	np	$\to s$
JOHN	LOVE	MARY	

the	girl	which	John	loves	
(np / n)	n	$(n \backslash n) / (s / np)$	np	$(np \backslash s) / np$	$\to np$
THE	GIRL	$\lambda x^{e \to t}.\lambda y^{e \to t}.\lambda z^{e}.(x(z) \,\&\, y(z))$	JOHN	LOVE	

Semantic value: $\text{THE}\, \lambda z.(\text{GIRL}(z) \,\&\, \text{LOVE}(z)(\text{JOHN}))$.

Actually, $\mathrm{L} \vdash np\ (np \backslash s) / np \to s / np$ ("John loves" reduces to s / np).
Not derivable in AB.

Dependent Clauses: Limitations

John read *Ulysses* a month ago.
the book which John read [] a month ago

Dependent Clauses: Limitations

John read *Ulysses* a month ago.
the book which John read [] a month ago

John sings and loves Mary.
*the girl which John sings and loves

Coordination

John	sings	and	Mary	dances.	
np	$np \backslash s$	$(s \backslash s) / s$	np	$np \backslash s$	$\rightarrow s$
JOHN	SING	AND	MARY	DANCE	

Semantic value: SING(JOHN) & DANCE(MARY).

Coordination: Going Polymorphic

John	sings	or	dances.	
np	$np \setminus s$	$((np \setminus s) \setminus (np \setminus s)) / (np \setminus s)$	$(np \setminus s)$	$\rightarrow s$
$\textsc{john}$	$\textsc{sing}$	$\lambda x^{e \rightarrow t}.\lambda y^{e \rightarrow t}.\lambda z^{e}.(x(z) \vee y(z))$	$\textsc{dance}$	

Semantic value: $\textsc{sing}(\textsc{john}) \vee \textsc{dance}(\textsc{john})$.

Coordination: Going Polymorphic

In general:

$$T = T_1 \rightarrow \ldots \rightarrow (T_k \rightarrow t)$$

$$\mathfrak{or}_T : T \rightarrow (T \rightarrow T)$$

$$\mathfrak{or}_T = \lambda x^T.\lambda y^T.\lambda z_1^{T_1} \ldots \lambda z_k^{T_k}.(x z_1 \ldots z_k \vee y z_1 \ldots z_k)$$

Coordination Between Noun Phrases

John and Pete sing.

John or Pete sings.

Coordination Between Noun Phrases: AND

np^* — plural noun phrase.
$np^* \mapsto (e \to t)$ (in other words, $\mathcal{P}(e)$)

John	and	Pete	sing.	
np	$(np \setminus np^*) / np$	np	$np^* \setminus s$	$\to s$
JOHN	PAIR	PETE	SING*	

$$\{x^e, y^e\}^{e \to t} = \lambda z^e.(z = x \vee z = y)$$
$$\text{PAIR} = \lambda x^e.\lambda y^e.\{x, y\}$$
$$\text{SING}^* = \lambda w^{e \to t}.\forall^{(e \to t) \to t}\, \lambda x^e.(w(x) \Rightarrow \text{SING}(x)).$$

Coordination Between Noun Phrases: AND

np^* — plural noun phrase.
$np^* \mapsto (e \to t)$ (in other words, $\mathcal{P}(e)$)

John	and	Pete	sing.	
np	$(np \setminus np^*) / np$	np	$np^* \setminus s$	$\to s$
JOHN	PAIR	PETE	SING*	

$$\{x^e, y^e\}^{e\to t} = \lambda z^e.(z = x \vee z = y)$$
$$\text{PAIR} = \lambda x^e.\lambda y^e.\{x, y\}$$
$$\text{SING}^* = \lambda w^{e\to t}.\forall^{(e\to t)\to t}\, \lambda x^e.(w(x) \Rightarrow \text{SING}(x)).$$

This solution needs some first-order logic ($\forall$)...

Coordination Between Noun Phrases: AND

np^* — plural noun phrase.
$np^* \mapsto (e \to t)$ (in other words, $\mathcal{P}(e)$)

John	and	Pete	sing.	
np	$(np \backslash np^*) / np$	np	$np^* \backslash s$	$\to s$
JOHN	PAIR	PETE	SING^*	

$\{x^e, y^e\}^{e \to t} = \lambda z^e.(z = x \vee z = y)$
$\text{PAIR} = \lambda x^e.\lambda y^e.\{x, y\}$
$\text{SING}^* = \lambda w^{e \to t}.\forall^{(e \to t) \to t}\, \lambda x^e.(w(x) \Rightarrow \text{SING}(x))$.

This solution needs some first-order logic ($\forall$)... and doesn't work for OR.

Type Raising

$$\mathrm{L} \vdash p \to q\,/(p \setminus q)$$

Type Raising

$$\mathrm{L} \vdash p \to q \,/\, (p \,\backslash\, q)$$

$$x^{\sigma(p)} \mapsto \lambda f^{\sigma(p) \to \sigma(q)}.f(x)$$

Type Raising

$$\mathrm{L} \vdash p \to q\,/(p\,\backslash\,q)$$

$$x^{\sigma(p)} \mapsto \lambda f^{\sigma(p)\to\sigma(q)}.f(x)$$

Now we can transform $x : e$ into $\lambda f^{e\to t}.f(x) : (e \to t) \to t$.

Type Raising

$$\mathrm{L} \vdash p \to q\,/(p\,\backslash\,q)$$

$$x^{\sigma(p)} \mapsto \lambda f^{\sigma(p)\to\sigma(q)}.f(x)$$

Now we can transform $x : e$ into $\lambda f^{e\to t}.f(x) : (e \to t) \to t$.
Again, this doesn't work in AB.

Coordination of Noun Phrases: OR

$np^s = s / (np \backslash s)$.

John	or	Pete	sings.
np	$(np^s \backslash np^s) / np^s$	np	$np \backslash s$
JOHN	$\lambda x^{(e \to t) \to t}.\lambda y^{(e \to t) \to t}.\lambda z^{e \to t}.(x(z) \vee y(z))$	PETE	SING

Coordination of Noun Phrases: OR

$np^s = s/(np \backslash s)$.

John	or	Pete	sings.
np	$(np^s \backslash np^s) / np^s$	np	$np \backslash s$
$\textsc{john}$	$\lambda x^{(e \to t) \to t}.\lambda y^{(e \to t) \to t}.\lambda z^{e \to t}.(x(z) \vee y(z))$	$\textsc{pete}$	$\textsc{sing}$

Perform type raising for "John" and "Pete":

John	or	Pete	
np^s	$(np^s \backslash np^s) / np^s$	np^s	$\to np^s$
$\lambda f.f(\textsc{john})$	$\lambda x.\lambda y.\lambda z.(x(z) \vee y(z))$	$\lambda f.f(\textsc{pete})$	

Coordination of Noun Phrases: OR

$np^s = s \,/(np \backslash s)$.

John	or	Pete	sings.
np	$(np^s \backslash np^s) \,/\, np^s$	np	$np \backslash s$
JOHN	$\lambda x^{(e\to t)\to t}.\lambda y^{(e\to t)\to t}.\lambda z^{e\to t}.(x(z) \vee y(z))$	PETE	SING

Perform type raising for "John" and "Pete":

John	or	Pete	
np^s	$(np^s \backslash np^s) \,/\, np^s$	np^s	$\to np^s$
$\lambda f.f(\text{JOHN})$	$\lambda x.\lambda y.\lambda z.(x(z) \vee y(z))$	$\lambda f.f(\text{PETE})$	

John or Pete	sings.	
$s \,/(np \backslash s)$	$np \backslash s$	$\to s$
$\lambda f.(f(\text{JOHN}) \vee f(\text{PETE}))$	SING	

Final semantic value: $\text{SING}(\text{JOHN}) \vee \text{SING}(\text{PETE})$.

Gentzen-style Product-Free Lambek Calculus

$$A \to A$$

$$\frac{A\,\Pi \to B}{\Pi \to A \setminus B}\text{, } \Pi \text{ is not empty} \qquad \frac{\Pi\, A \to B}{\Pi \to B\,/\,A}\text{, } \Pi \text{ is not empty}$$

$$\frac{\Pi \to A \quad \Gamma\, B\, \Delta \to C}{\Gamma\, \Pi\, (A \setminus B)\, \Delta \to C} \qquad \frac{\Pi \to A \quad \Gamma\, B\, \Delta \to C}{\Gamma\, (B\,/\,A)\, \Pi\, \Delta \to C}$$

$$\frac{\Pi \to A \quad \Gamma\, A\, \Delta \to C}{\Gamma\, \Pi\, \Delta \to C}\ (\mathrm{cut})$$

Cut Elimination

Theorem (Lambek '58)

Every sequent derivable in L *can be derived without using the* (cut) *rule.*

Cut Elimination

Theorem (Lambek '58)

Every sequent derivable in L *can be derived without using the* (cut) *rule.*

Proof.

Routine induction on the complexity of the sequent. □

Cut Elimination

Theorem (Lambek '58)

Every sequent derivable in L *can be derived without using the* (cut) *rule.*

Proof.

Routine induction on the complexity of the sequent. □

Benefits:

- Subformula property
- Decidability (more precisely, the derivability problem for L belongs to NP)

Lambek Grammars and Context-Free Grammars

Theorem (Gaifman '61)

Every context-free language without the empty word can be generated by an AB*-grammar.*

Theorem (Pentus '92)

Every language generated by a Lambek categorial grammar is context-free.

Greibach Normal Form

A context-free grammar is in Greibach normal from if every rule of this grammar has one of the following forms:

- $A \to a$
- $A \to aB$
- $A \to aBC$

Theorem (Greibach '65)

Every context-free language without the empty word can be generated by a context-free grammar in Greibach normal form.

From Context-Free Grammars to AB-grammars: Proof

$N \ni A \rightsquigarrow p_A \in \mathrm{Pr}.$

Categorial vocabulary:

$A \to a$	$\langle p_A, a \rangle$
$A \to aB$	$\langle p_A \,/\, p_B, a \rangle$
$A \to aBC$	$\langle (p_A \,/\, p_C) \,/\, p_B, a \rangle$

$H = p_S$

Web Page

http://www.mi.ras.ru/~sk/lehre/esslli2015