Extending lambda grammars

Richard Moot (CNRS, LaBRI/LIRMM)

New Landscapes In Theoretical Computational Linguistics, 16-10-2016, Columbus, Ohio

Overview

- Lambek grammars
- Lambda grammars
- Lambda grammars as a fragment of first-order linear logic
- Problems and extensions

The Lambek calculus

The Lambek calculus

- A logical calculus for natural language syntax and semantics introduced in (Lambek 1958)
- Two connectives:
  - A/B (A *over* B)
  - B\A (B *under* A)
- The Curry-Howard isomorphism allows us to combine the Lambek calculus with natural language semantics in the tradition of Montague.

The Lambek calculus

$$\frac{A/B \quad B}{A}\,/E \qquad \frac{B \quad B\backslash A}{A}\,\backslash E$$

$$\dfrac{\begin{array}{cc} \ldots & [B]^i \\ \vdots & \\ A & \end{array}}{A/B}\,/I \qquad \dfrac{\begin{array}{cc} [B]^i & \ldots \\ \vdots & \\ A & \end{array}}{B\backslash A}\,\backslash I$$

Limitations of the Lambek calculus

- Though the Lambek calculus handles the basics of Montague grammar, it has problems with non-peripheral wide scope.
- Lambek grammars generate only context-free languages; some natural languages demonstrate non-context-free phenomena like copying and multiple/crossed dependencies.
- To deal with these problems, a large number of extensions to the Lambek calculus has been proposed.

Medial extraction

- It is often claimed that the Lambek calculus cannot handle medial extraction, eg. that phrases like the following are underivable.

    1. contracts which John filed yesterday

- What do we mean exactly when we make claims like this?

Semantic types

$$
\dfrac{\dfrac{which}{(e \to t) \to (e \to t) \to e \to t}\,Lex \qquad \dfrac{\dfrac{\dfrac{John}{e}\,Lex \qquad \dfrac{\dfrac{filed}{e \to e \to t}\,Lex \qquad [e]_1}{e \to t}\,\to E}{t}\,\to E}{e \to t}\,\to I_1}{(e \to t) \to e \to t}\,\to E
$$

$$
\begin{aligned}
s^* &= t \\
np^* &= e \\
n^* &= e \to t \\
(A \multimap B)^* &= A^* \to B^*
\end{aligned}
$$

Deep structure

$$\dfrac{\dfrac{\mathit{which}}{(np \multimap s) \multimap (n \multimap n)}\mathit{Lex} \qquad \dfrac{\dfrac{\dfrac{\mathit{John}}{np}\mathit{Lex} \quad \dfrac{\dfrac{\mathit{filed}}{np \multimap (np \multimap s)}\mathit{Lex} \quad [np]_1}{np \multimap s}\multimap E}{s}\multimap E}{np \multimap s}\multimap I_1}{n \multimap n}\multimap E$$

$$\begin{aligned} s^* &= t \\ np^* &= e \\ n^* &= e \to t \\ (A \multimap B)^* = & \quad A^* \to B^* \end{aligned}$$

Lambek calculus pseudo-proof

$$\dfrac{\dfrac{\mathit{which}}{(n\backslash n)/(np \multimap s)}\,Lex \qquad \dfrac{\dfrac{\dfrac{\mathit{John}}{np}\,Lex \qquad \dfrac{\dfrac{\mathit{filed}}{(np\backslash s)/np}\,Lex \qquad [np]_1}{np\backslash s}\,/E}{s}\,\backslash E}{np \multimap s}\,\multimap I}{n\backslash n}\,/E$$

Medial extraction in L (solution 1)

contracts	n
which	$(n\backslash n)/s_{np}$
which	$(n\backslash n)/(np\backslash s)$
John	np
filed	$(np\backslash s)/np$
filed	$np\backslash s_{np}$
yesterday	$(np\backslash s)\backslash(np\backslash s)$
yesterday	$(np\backslash s_{np})\backslash(np\backslash s_{np})$

Medial extraction in L (solution 1)

contracts	n
which	$(n\backslash n)/s_{np}$
which	$(n\backslash n)/(np\backslash s)$
John	np
filed	$(np\backslash s)/np$
filed	$np\backslash s_{np}$
yesterday	$(np\backslash s)\backslash(np\backslash s)$
yesterday	$(np\backslash s_{np})\backslash(np\backslash s_{np})$

Medial extraction in L (solution 1)

contracts	n
which	$(n\backslash n)/s_{np}$
which	$(n\backslash n)/(np\backslash s)$
John	np
filed	$(np\backslash s)/np$
filed	$np\backslash s_{np}$
without	$((np\backslash s)\backslash(np\backslash s))/(np\backslash sing)$
without	$((np\backslash s_{np})\backslash(np\backslash s_{np}))/(np\backslash sing_{np})$
reading	$np\backslash sing/np$
reading	$np\backslash sing_{np}$

Medial extraction in L (solution 2)

- Emms proposes the second-order formula $\forall X. ((n\backslash n)/(X\backslash s))/(X/np)$ for extraction.
- We can select a suitable finite set of instantiations of this type to at least approximate medial extraction

Two forms of inadequacy

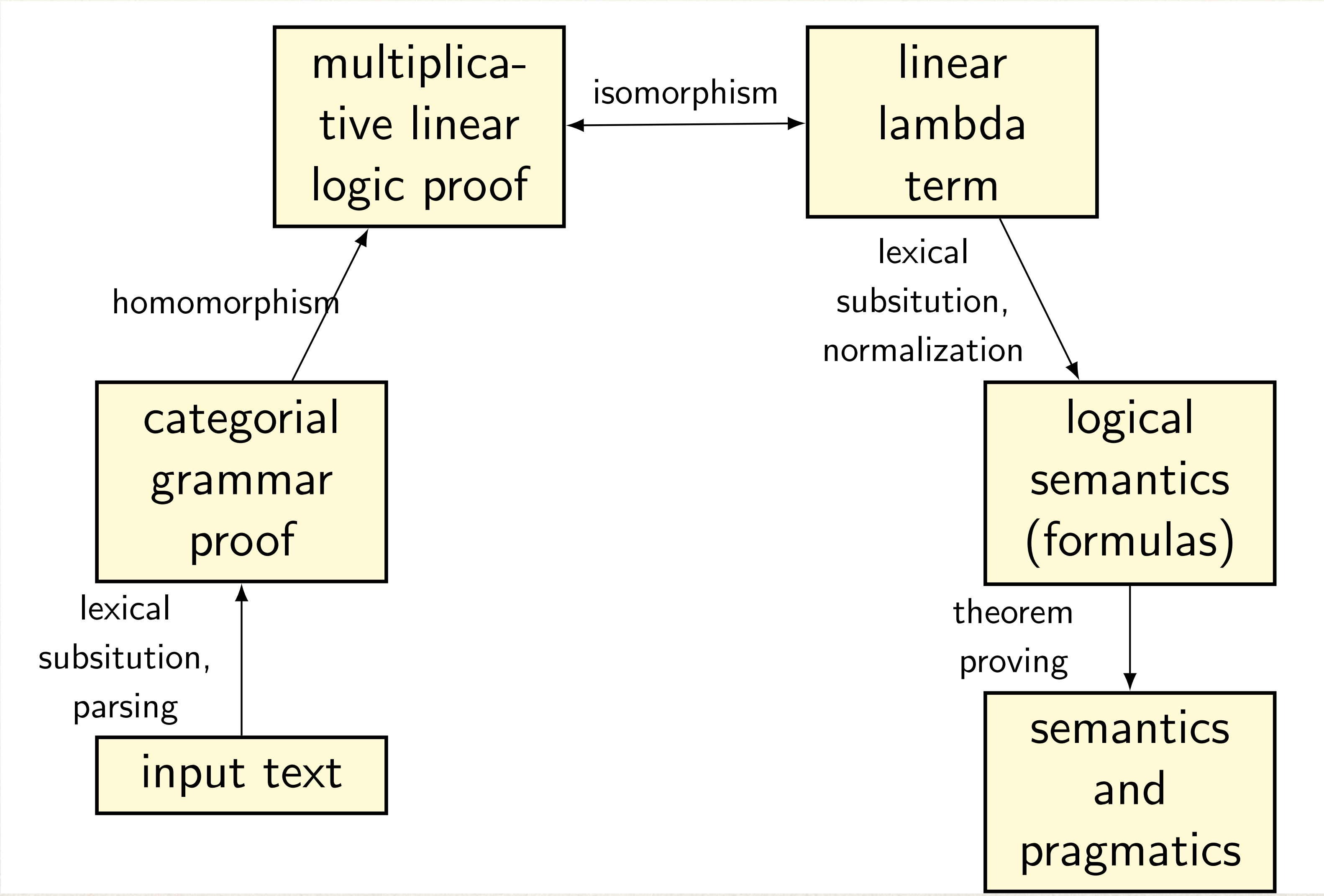
- Shieber (1988) distinguishes between the *absolute* expressiveness and the *functional* expressiveness of formal systems
    - the Lambek calculus cannot handle extraction in terms of functional expressiveness
    - the Lambek calculus cannot handle Montague-style quantifier scope in terms of absolute expressiveness (by a simple counting argument)

Descriptive adequacy

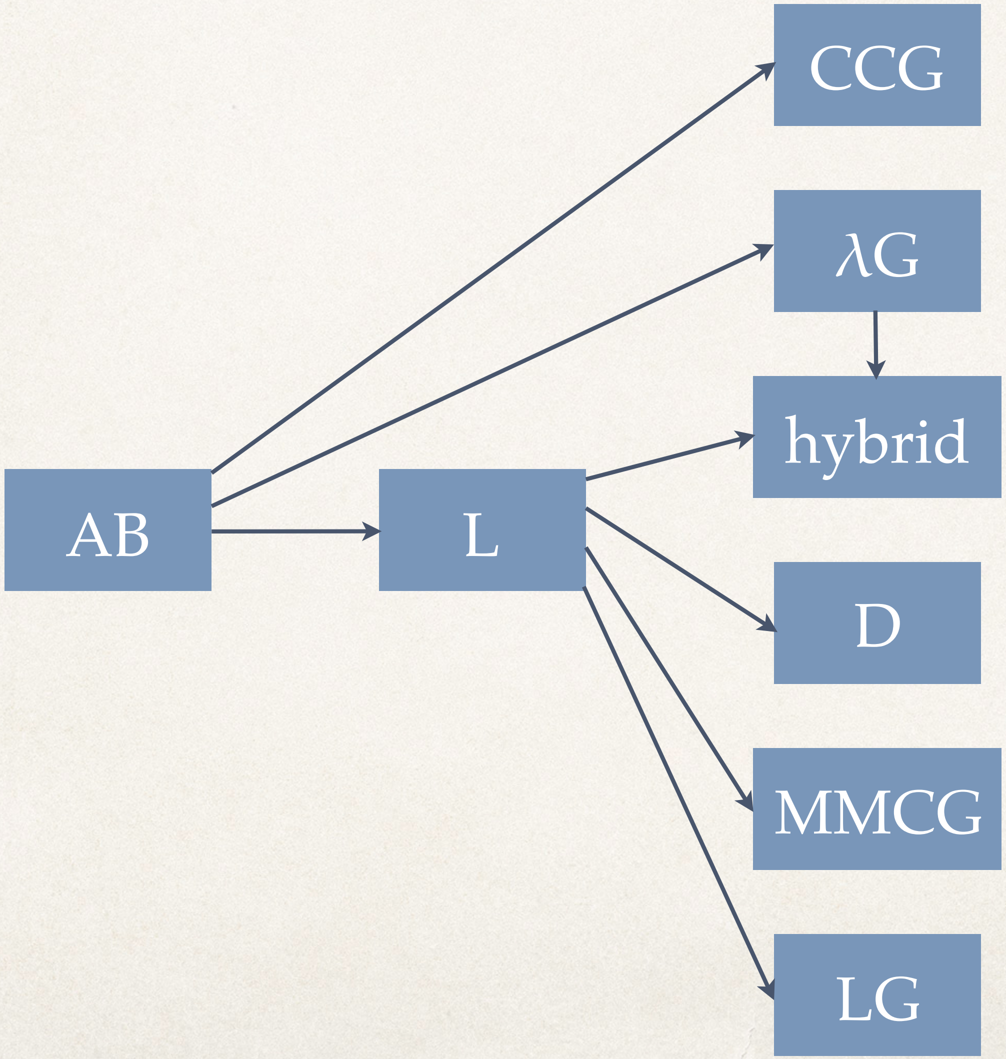
- Claims that the Lambek calculus cannot handle medial extraction *presuppose* both type-logical deep structure and a descriptive adequacy criterion (that is, avoiding lexical duplication).
- Solution 1 does not generalise very well to multiple extraction, multiple medial quantification (which can only be approximated), gapping, etc.
- If we allow duplication of lexical entries and atomic formulas, then it becomes very hard to falsify any type-logical grammar (even impossible if we allow approximation).

Extensions and variants of the Lambek calculus

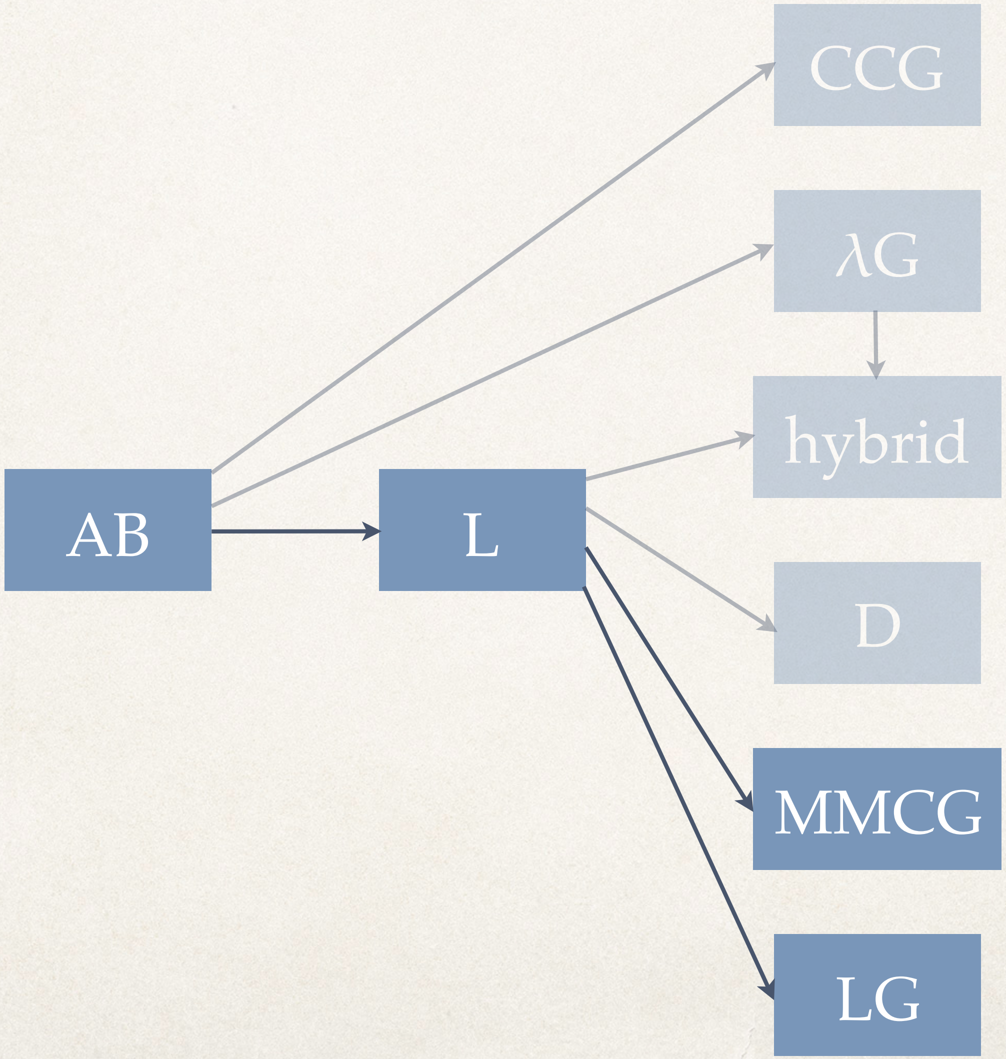
Common core architecture of type-logical grammars



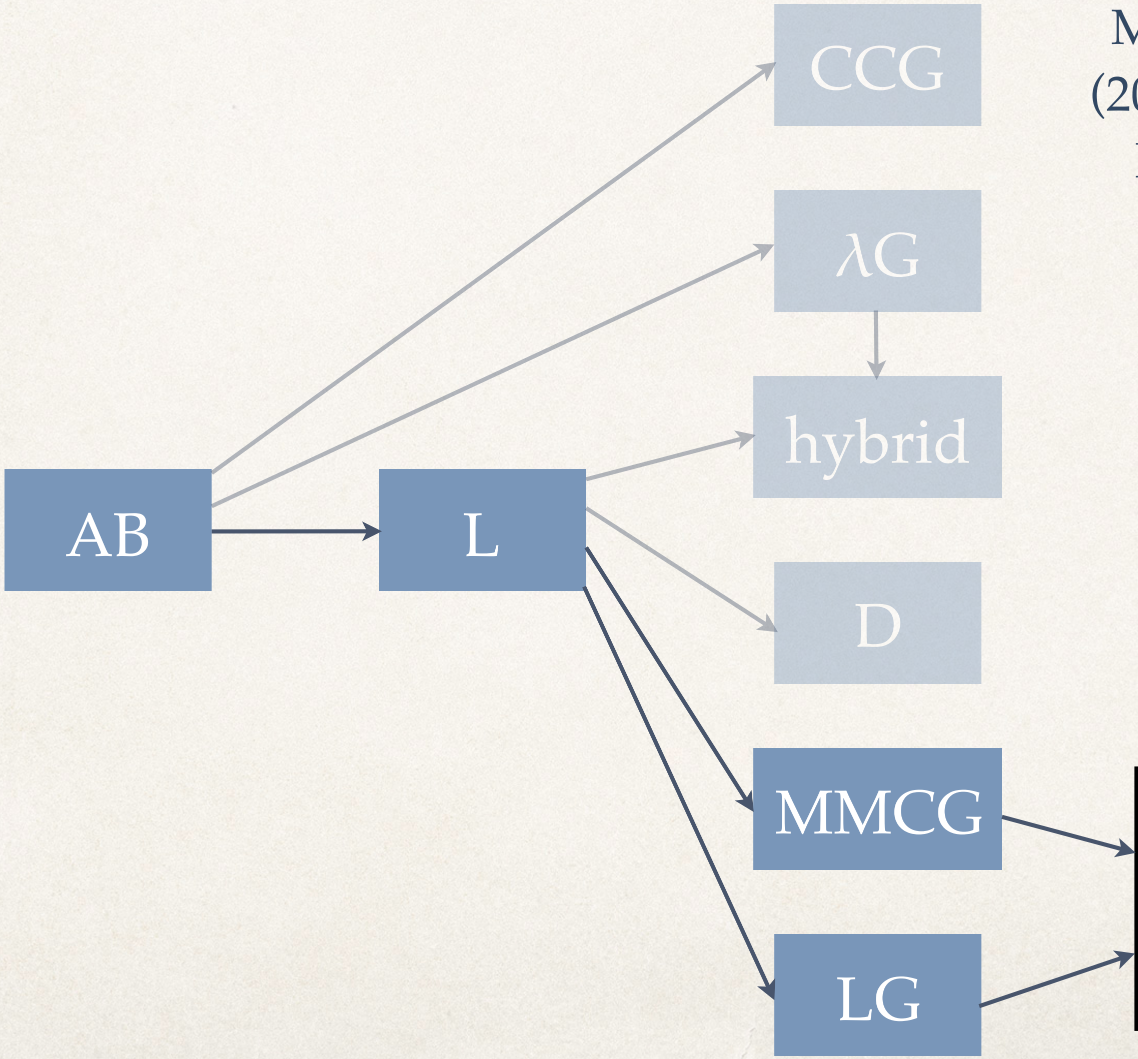
The Lambek calculus and its extensions



The Lambek calculus and its extensions



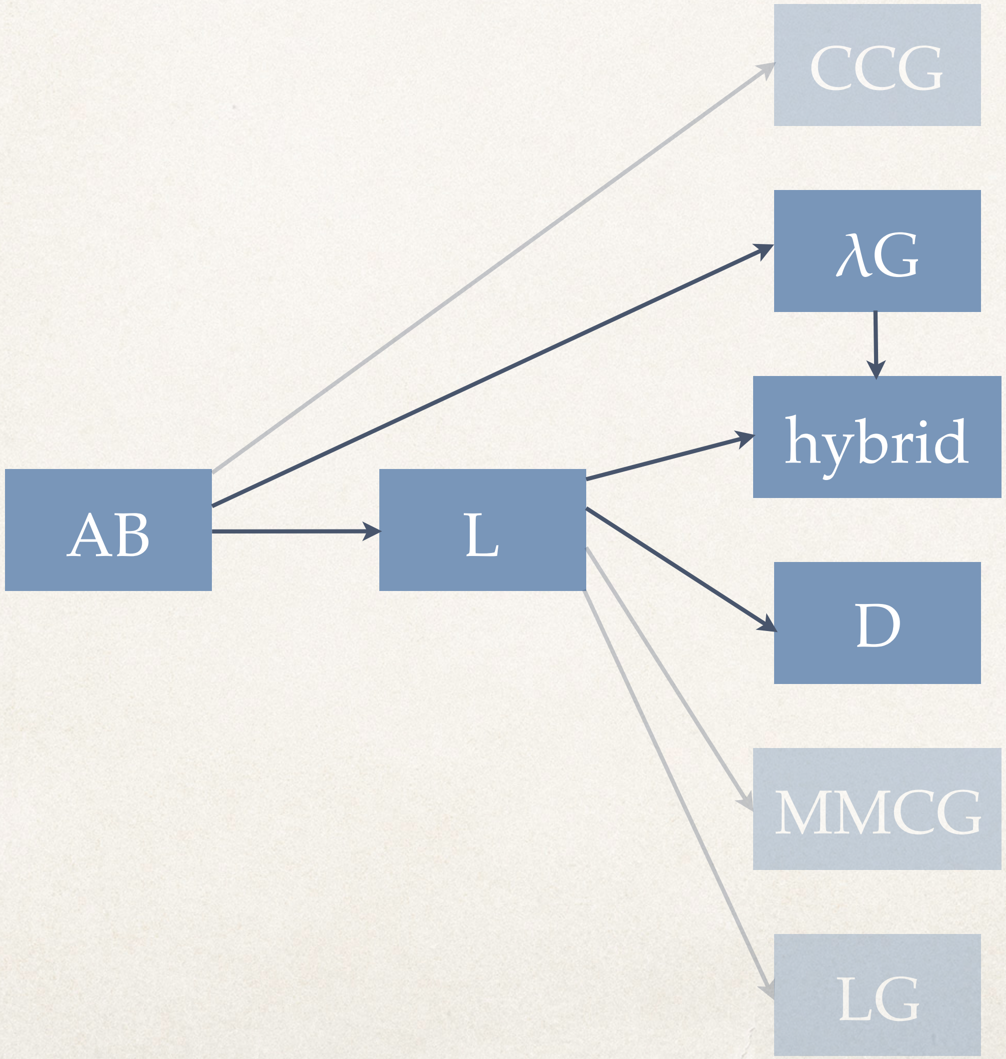
The Lambek calculus and its extensions



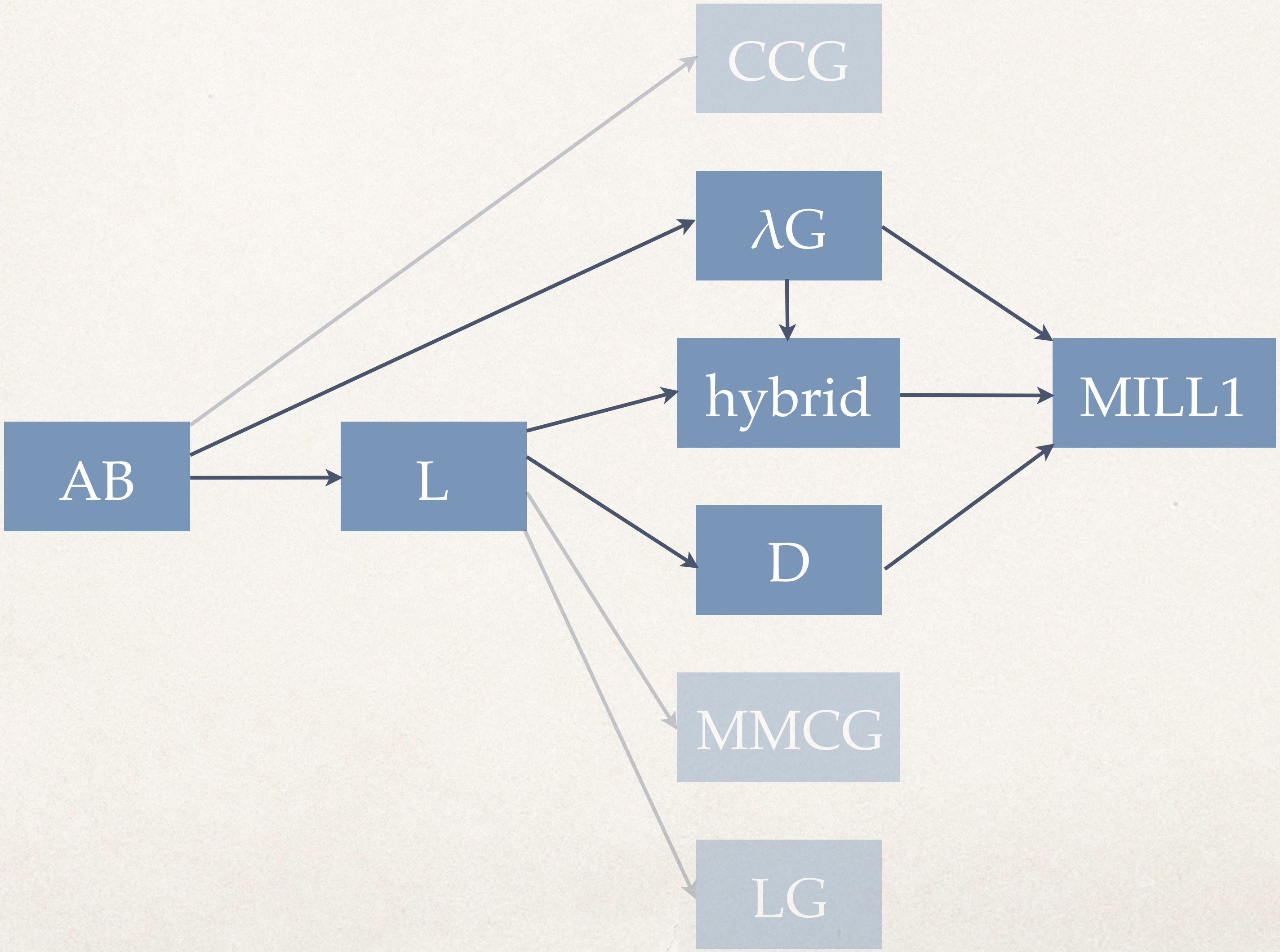
Moot & Puite (2002), Moot (2007),Moot & Retoré (2012), Moortgat & Moot (2013)

Proof nets as graph rewriting ≈ interaction nets

The Lambek calculus and its extensions



The Lambek calculus and its extensions



Multiplicative first-order linear logic

Parsing using string position pairs

0 1 2 3 4

Jim	proved	the	theorem
np	(np\s)/np	np/n	n

Parsing using string position pairs

0	1	2	3	4

Jim	proved	the	theorem
np	(np\s)/np	np/n	n
		$\forall x.n(3,x) \multimap np(2,x)$	n(3,4)

Parsing using string position pairs

0	1	2	3	4

Jim	proved	the	theorem
np	(np\s)/np	np/n	n

$$\frac{\forall x.n(3,x) \multimap np(2,x) \qquad n(3,4)}{np(2,4)}$$

Parsing using string position pairs

0 1 2 3 4

Jim	proved	the	theorem
np	(np\s)/np	np/n	n

$$\forall z.np(2,z) \multimap \forall y.np(y,1) \multimap s(y,z) \qquad \frac{\forall x.n(3,x) \multimap np(2,x) \qquad n(3,4)}{np(2,4)}$$

Parsing using string position pairs

0	1	2	3	4

Jim	proved	the	theorem
np	(np\s)/np	np/n	n

$$np(0,1) \qquad \frac{\forall z.np(2,z) \multimap \forall y.np(y,1) \multimap s(y,z) \qquad \dfrac{\forall x.n(3,x) \multimap np(2,x) \qquad n(3,4)}{np(2,4)}}{\forall y.np(y,1) \multimap s(y,4)}$$

Parsing using string position pairs

0 1 2 3 4

Jim	proved	the	theorem
np	(np\s)/np	np/n	n

$$\dfrac{np(0,1) \qquad \dfrac{\forall z.np(2,z)\multimap\forall y.np(y,1)\multimap s(y,z) \qquad \dfrac{\forall x.n(3,x)\multimap np(2,x) \qquad n(3,4)}{np(2,4)}}{\forall y.np(y,1)\multimap s(y,4)}}{s(0,4)}$$

Lambek calculus and MILL1

$$
\begin{aligned}
\|a\|^{\langle e_i, e_j \rangle} &= a(e_i, e_j) \\
\|A/B\|^{\langle e_i, e_j \rangle} &= \forall x_k . \|B\|^{\langle e_j, x_k \rangle} \multimap \|A\|^{\langle e_i, x_k \rangle} \\
\|B \backslash A\|^{\langle e_i, e_j \rangle} &= \forall x_k . \|B\|^{\langle x_k, e_i \rangle} \multimap \|A\|^{\langle x_k, e_j \rangle} \\
\|A \bullet B\|^{\langle e_i, e_j \rangle} &= \exists x_k . \|A\|^{\langle e_i, x_k \rangle} \otimes \|B\|^{\langle x_k, e_j \rangle}
\end{aligned}
$$

See Moot & Piazza (2001) for the correctness of this translation

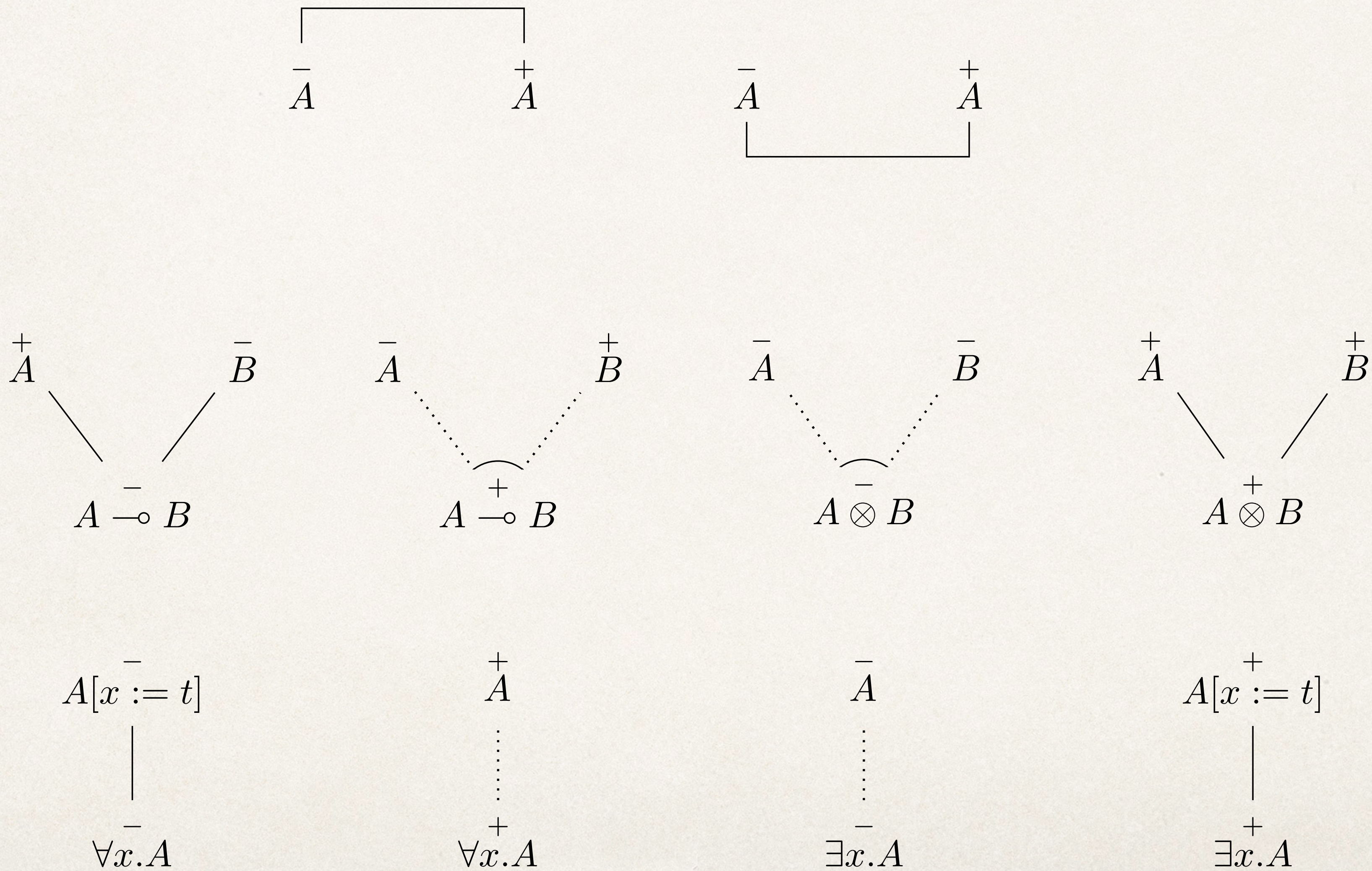
MILL1 natural deduction

$$\dfrac{A \quad A \multimap B}{B} \multimap E \qquad \dfrac{\begin{matrix}[A]^i \\ \vdots \\ B\end{matrix}}{A \multimap B} \multimap I$$

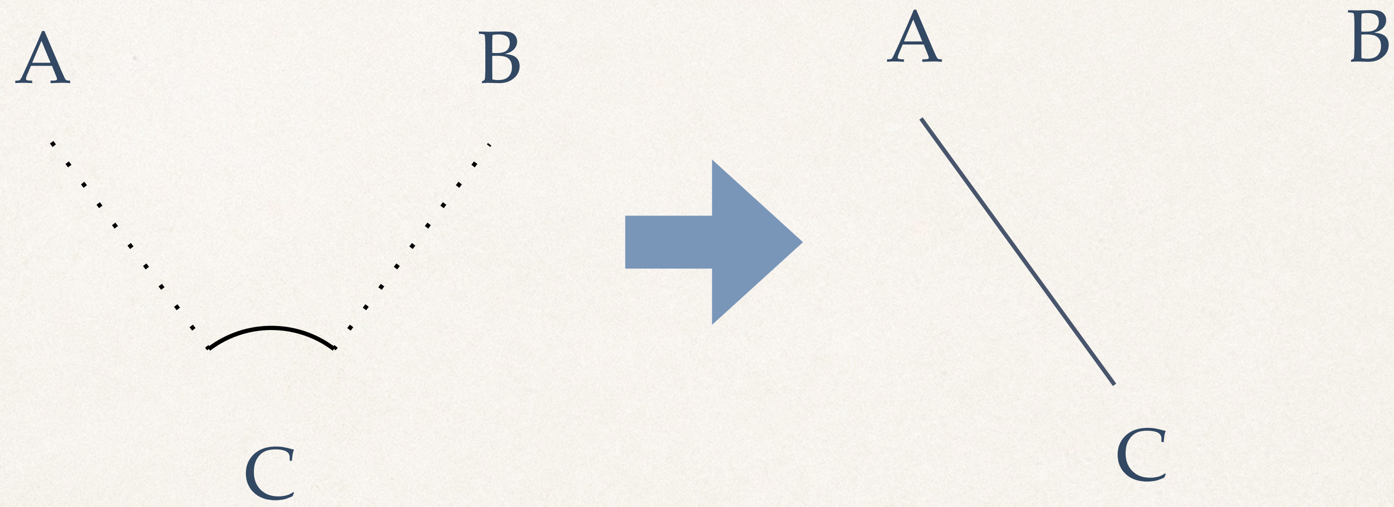
$$\dfrac{\exists x.A \quad \begin{matrix}[A]^i \\ \vdots \\ C\end{matrix}}{C} \exists E_i^* \qquad \dfrac{A[x := t]}{\exists x.A} \exists I$$

$$\dfrac{\forall x.A}{A[x := t]} \forall E \qquad \dfrac{A}{\forall x.A} \forall I^*$$

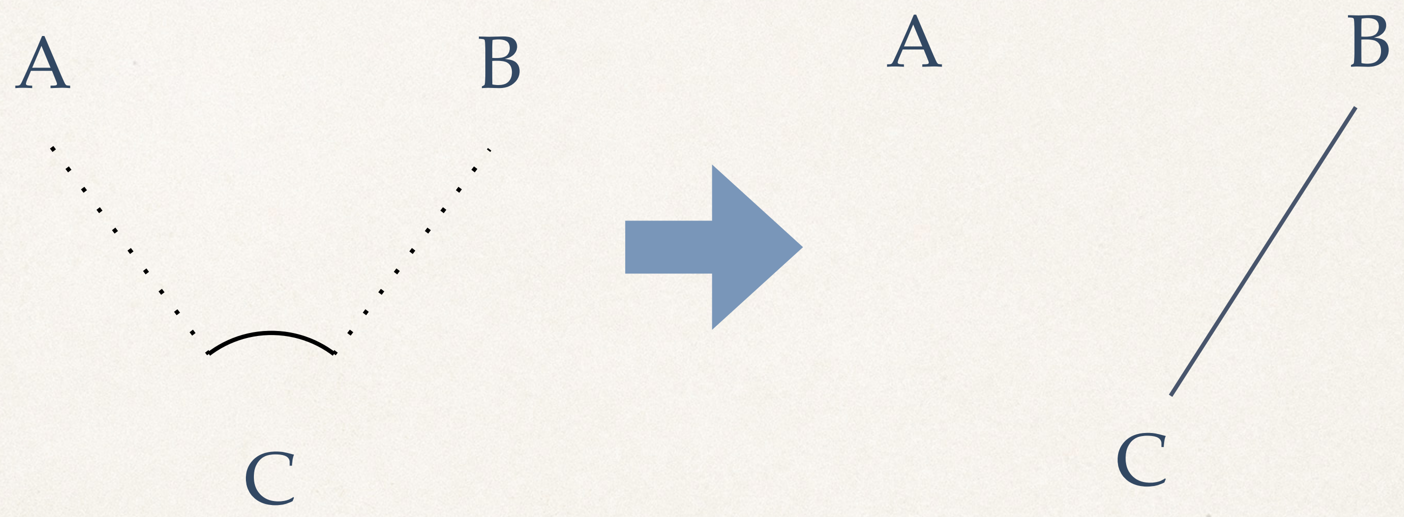
MILL1 proof structures: links



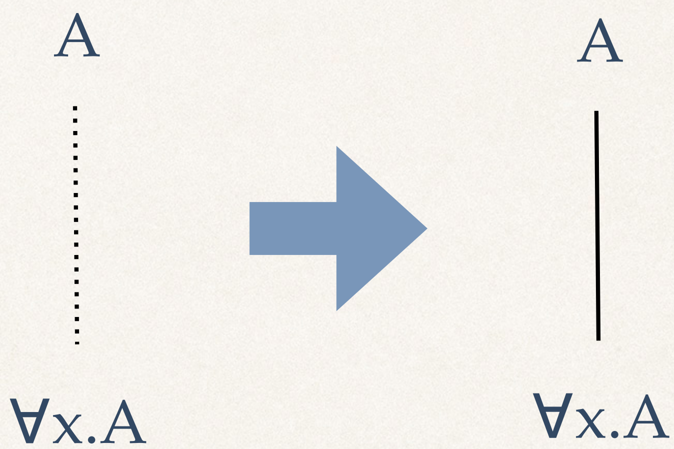
MILL1 proof nets: switchings



MILL1 proof nets: switchings



MILL1 proof nets: switchings



replace dotted link by solid link: boring (necessary only for "vacuous" quantification (no occurrences of x in A)

Interesting case: connect to formula with free occurrence of x

In addition, a switching can connect the conclusion of the link to any formula containing a free occurrence of the variable *x*

MILL1 proof nets

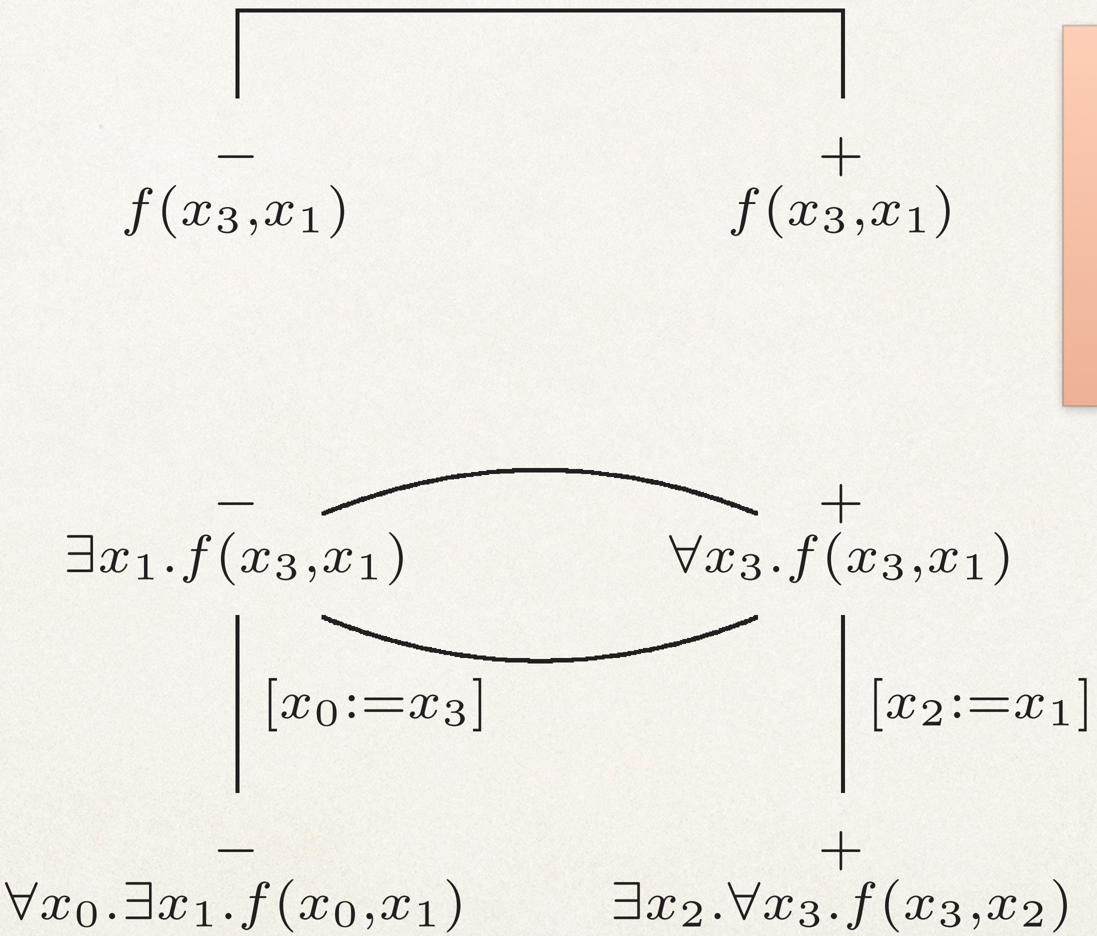
Theorem (Girard 1991)
A first-order multiplicative proof structure is a *proof net* iff all its correction graphs are acyclic and connected

Not sure if it stays quite linear, but it is easy to see this can be done in n^2 time

MILL1 proof nets

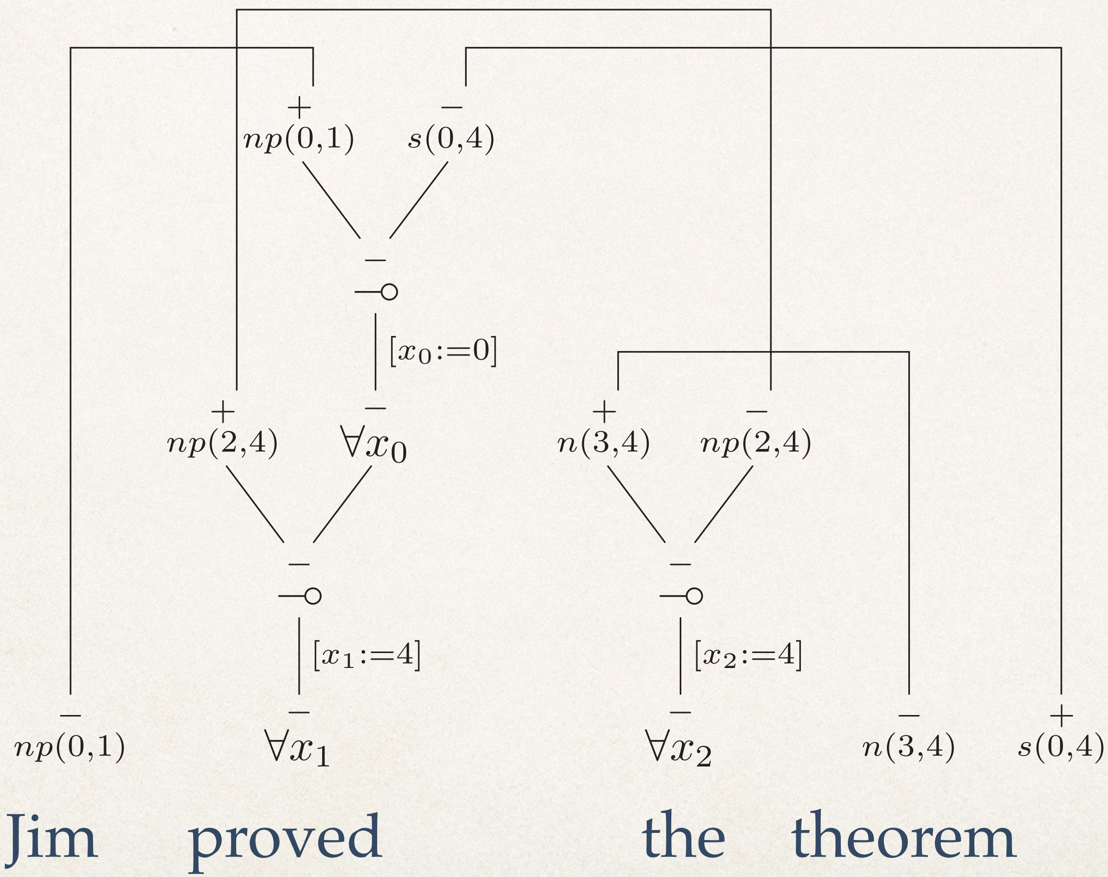
$$
\begin{tikzcd}
\overset{-}{f(x_3,x_1)} \arrow[r, dash, bend left] \arrow[d, dash, dotted] & \overset{+}{f(x_3,x_1)} \arrow[d, dash, dotted] \\
\overset{-}{\exists x_1.f(x_3,x_1)} \arrow[d, dash, "{[x_0:=x_3]}"] & \overset{+}{\forall x_3.f(x_3,x_1)} \arrow[d, dash, "{[x_2:=x_1]}"] \\
\overset{-}{\forall x_0.\exists x_1.f(x_0,x_1)} & \overset{+}{\exists x_2.\forall x_3.f(x_3,x_2)}
\end{tikzcd}
$$

MILL1 proof nets



think of f as “has a father”

Example: proof nets without planarity



Notions of complexity - order

$$\mathrm{order}(p) = 0$$
$$\mathrm{order}(A \multimap B) = \max(\mathrm{order}(A) + 1, \mathrm{order}(B))$$

- roughly speaking, the order of the formulas used in type-logical grammars is an indication of the complexity of the semantic operations
- in treebanks, order 3 or 4 seems to suffice (order 4 occurs for gapping of auxiliaries and the copula)

Notions of complexity - width

the *width* of a formula is the maximum number of free variables occurring in its subformulas

- roughly speaking, formula width corresponds to the complexity of the string operations: width 2 corresponds to operations on strings, width 4 pairs of strings etc.
- it is a more robust notion than predicate arity

λ-grammars

Lambda-grammars

- Formalism introduced by Curry (1961) and Oehrle (1994), called (depending on the authors) λ-grammars (Muskens), abstract categorial grammars (de Groote e.a.) and linear grammars (Pollard)
- Replace strings by (simply typed, linear) lambda-terms
- Lambda grammars and first-order linear logic: all the hard work is done in Kanazawa (2011), de Groote (2015).

Using pairs of string positions

0 1 2 3

Mary	loves	someone

$$\lambda z^{\sigma}(Mary^{\sigma\to\sigma}\, z) : \sigma \to \sigma$$
$$np^{*}$$

$$\lambda q^{\sigma\to\sigma}\,\lambda p^{\sigma\to\sigma}\,\lambda y^{\sigma}.(p\,(loves^{\sigma\to\sigma}(q\,y))) : (\sigma \to \sigma)\to(\sigma \to \sigma)\to\sigma \to \sigma$$
$$(np \multimap (np \multimap s))^{*}$$

$$\lambda P^{(\sigma\to\sigma)\to\sigma\to\sigma}\lambda z^{\sigma}.((P\,someone^{\sigma\to\sigma})\,z) : ((\sigma \to \sigma)\to\sigma \to \sigma)\to\sigma \to \sigma$$
$$((np \multimap s) \multimap s)^{*}$$

Using pairs of string positions

0 1 2 3

Mary	loves	someone

$$\lambda z^1 (Mary^{1 \to 0} z) : 1 \to 0$$

$$np^*$$

$$\lambda q^{B \to 2} \lambda p^{1 \to A} \lambda y^B . (p\,(loves^{2 \to 1} (q\, y))) : (B \to 2) \to (1 \to A) \to B \to A$$

$$(np \multimap (np \multimap s))^*$$

$$\lambda P^{(3 \to 2) \to D \to C} \lambda z^D . ((P\, someone^{3 \to 2})\, z) : ((3 \to 2) \to D \to C) \to D \to C$$

$$((np \multimap s) \multimap s)^*$$

Using pairs of string positions

0 1 2 3

Mary	loves	someone

$$\lambda z^1 (Mary^{1 \to 0}\, z) : 1 \to 0$$

$$np(0,1)$$

$$\lambda q^{B \to 2} \lambda p^{1 \to A} \lambda y^B.(p\,(loves^{2 \to 1}(q\,y))) : (B \to 2) \to (1 \to A) \to B \to A$$

$$np(2,B) \multimap (np(A,1) \multimap s(A,B))$$

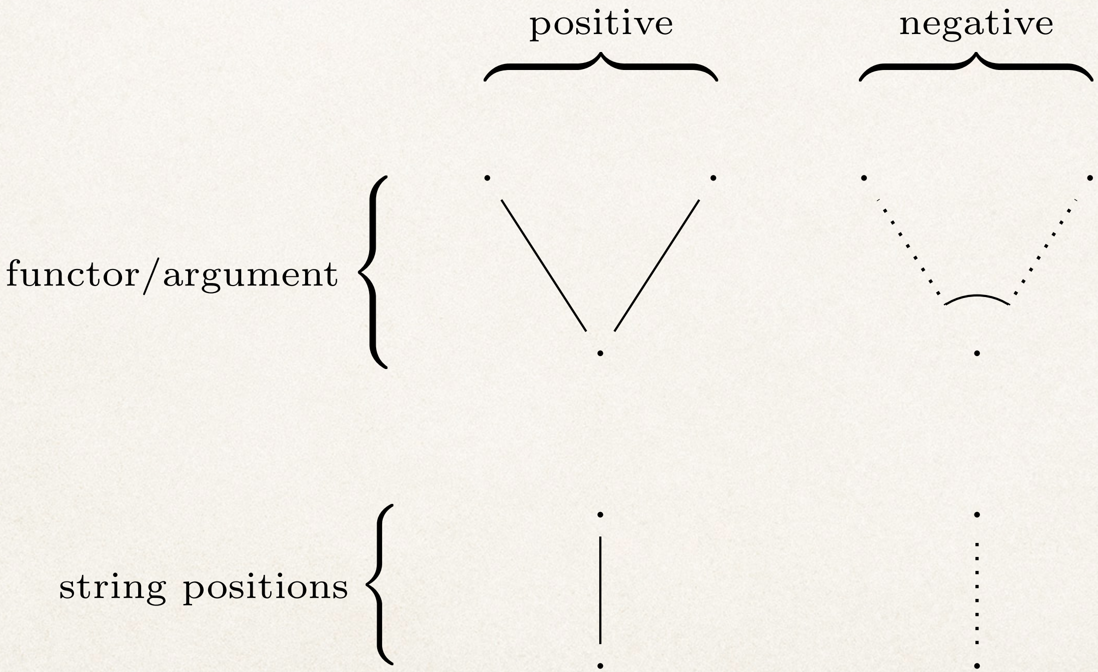
$$\lambda P^{(3 \to 2) \to D \to C} \lambda z^D.((P\, someone^{3 \to 2})\, z) : ((3 \to 2) \to D \to C) \to D \to C$$

$$(np(2,3) \multimap s(C,D)) \multimap s(C,D)$$

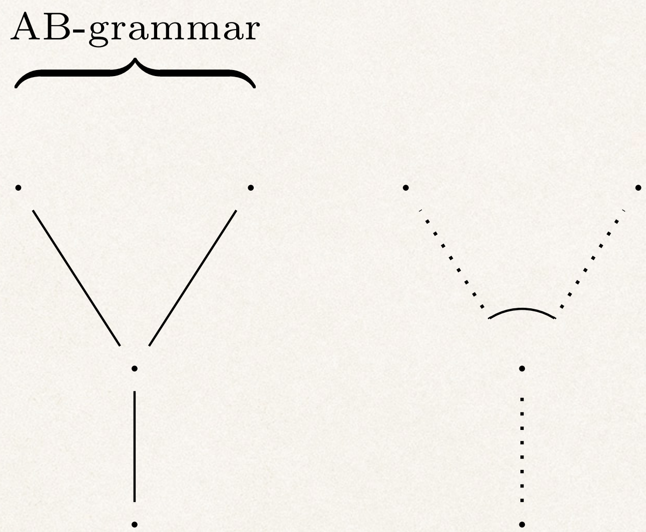
Hybrid type-logical grammar

- Lambda-grammars are a fragment of first-order linear logic which contains only negative universal and positive existential quantifiers (in terms of classical proof nets, there are no *universal* links).
- Hybrid type-logical grammar (Kubota & Levine 2013) add the Lambek connectives to lambda-grammars: we are allowed to replace atomic formulas of type $\sigma \rightarrow \sigma$ by Lambek calculus formulas. From the current point of view, this means *composing* the two translations.
- The goal of this addition is to address some of the challenges of lambda-grammars we will see later.

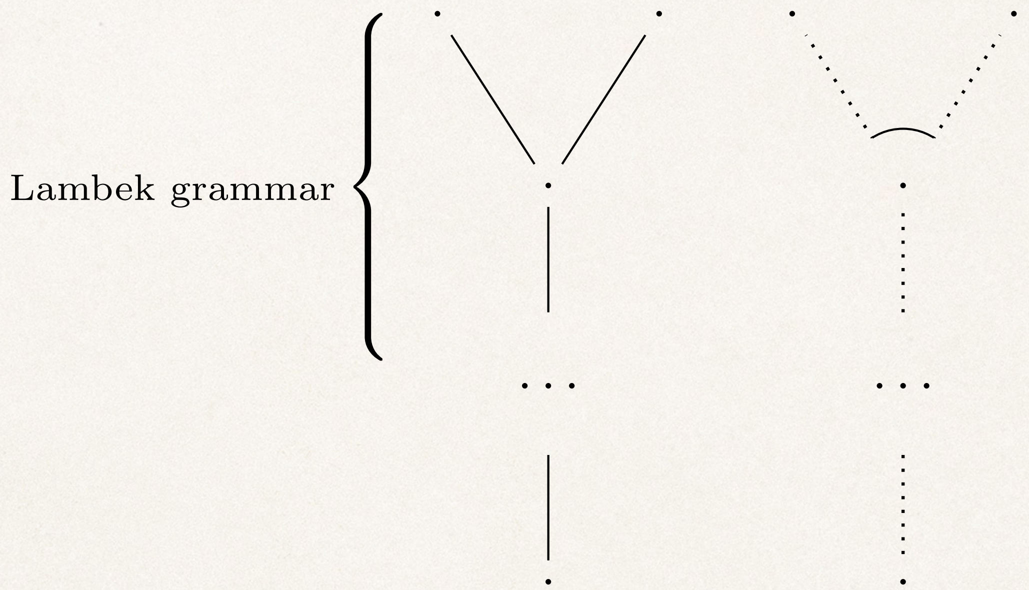
A Visual Comparison of the Different Calculi - MILL1



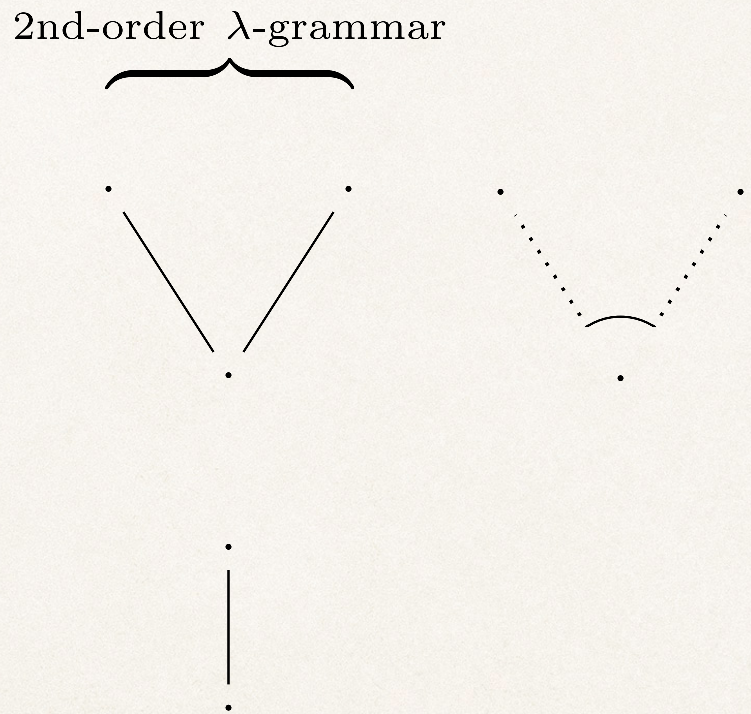
A Visual Comparison of the Different Calculi - L



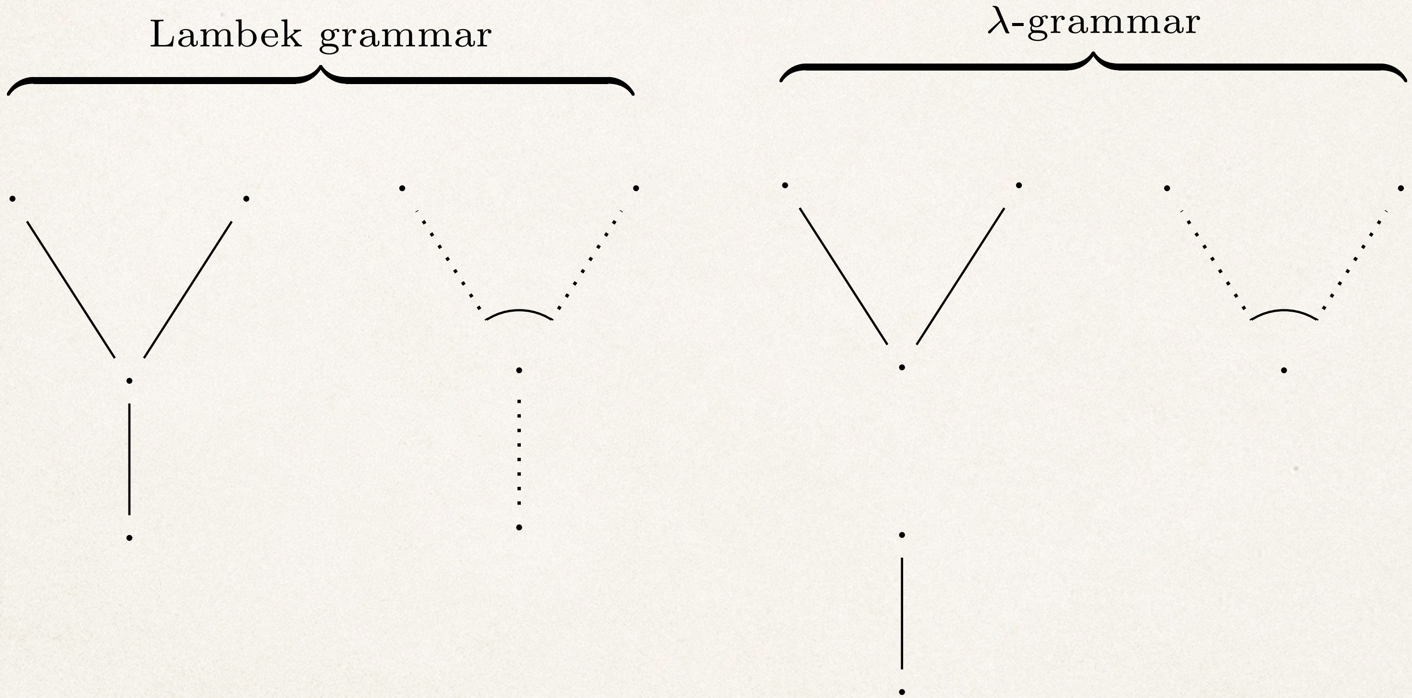
A Visual Comparison of the Different Calculi - D



A Visual Comparison of the Different Calculi - λ-grammars



A Visual Comparison of the Different Calculi - hybrid



Convergence

- In many cases (eg. for relativizers and quantifiers), analyses proposed independently for the different formalisms are *identical* on translation into MILL1

(Moot & Piazza 2001)

$$l(\mathit{someone}, c_i, c_j) = \forall x_0 \forall x_1 (np(c_i, c_j) \multimap s(x_0, x_1)) \multimap s(x_0, x_1)$$

(Morril e.a. 2011)

$$\mathit{someone} : (s \uparrow np) \downarrow s$$

(Oehrle 1994)

$$\lambda P \lambda z.((P\ \mathit{someone}), z) : (np \multimap s) \multimap s$$

Problems for lambda grammars

Using the missing universal link as a diagnostic

- Look at prototypical applications of the universal link.
- In some cases, such as Lambek formulas $s/(np\backslash s)$ and $(n\backslash n)/(s/np)$, we can sidestep the absence of the universal link and make better predictions.
- Is this true for other cases?

ACG/lambda grammar problems

1. John deliberately hit Mary. (adverbs)
2. John bought a sandwich and ran to the train. (VP coordination)
3. John caught and ate a fish. (TV coordination)
4. John loves but Mary hates Noam. (right-node raising)
5. John bought himself a present. (reflexives)
6. John gave himself and every/a pretty girl a present.
7. John studies logic and Charles, phonetics. (gapping)
8. John left before Mary did. (ellipsis)

Are these really problems?

- These are problems according to the *exact same* standards as medial extraction is a problem for the Lambek calculus.
- Hence, saying the the ACG/lambda grammar treatment of extraction is superior to the Lambek calculus treatment, means admitting other type-logical grammars have a superior treatment for many other phenomena (unless we want to evaluate ACG/lambda grammars to lower standards than we apply to other formalisms)

Are these new problems?

- A move to [lambda grammar] representations […] does not seem to be compatible with this analysis [of coordination] (Muskens, 2001)
- [with respect to adverbs] Some extra machinery therefore needs to be developed in order to get a grammar in Curry's spirit working (Muskens, 2010, p.130).
- Lambek categorial grammars essentially fail to deal with medial gaps. […] This is a direct consequence of the attempt to regulate word order on the level of the type system. In fact, a lot of research carried out within the Lambek paradigm can be seen as the invention of a series of epicycles needed to counter this architectural mistake. (Muskens, 2010, p.131)

Adverbs

1. John deliberately hit Mary.

Lambek $(np\backslash s)/(np\backslash s)$

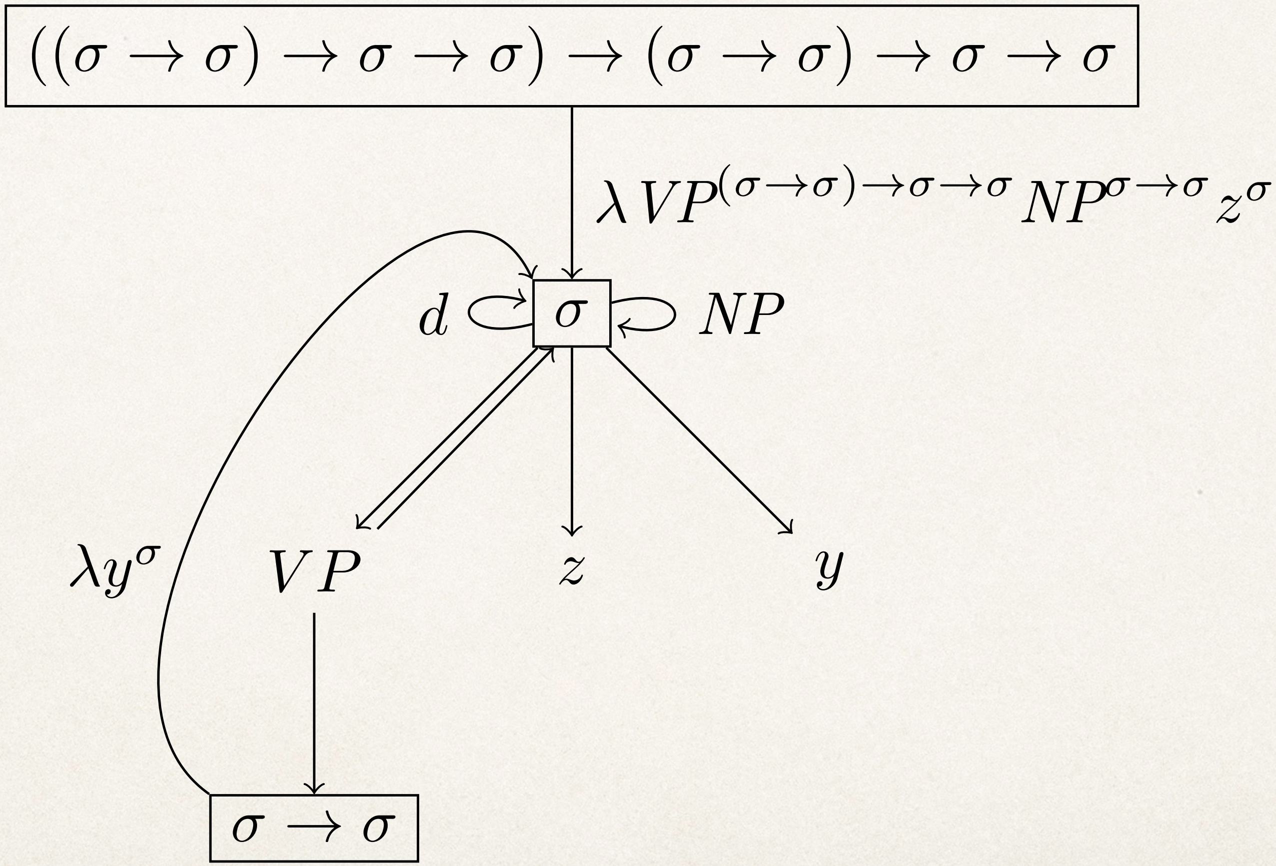
MILL1 $(\forall c.np(c,2) \multimap s(c,D)) \multimap np(E,1) \multimap s(E,D)$

$$2 \to 1 \nvdash ((2 \to c) \to D \to c) \to (1 \to E) \to D \to E$$

Enumerating lexical entries

- For the Lambek calculus, we could enumerate all possible syntactic types given a deep structure type simply by choosing "/" or "\" for each of the implications.
- For lambda grammars, we obtain a prosodic type from the deep structure type and can enumerate all (linear) lambda terms using inhabitation machines (van Benthem was the first to use these in the context of categorial grammars for enumerating possible *semantic* terms).

Inhabitation machine for the adverb



Adverbs

The candidate corresponding to the Lambek formula is uninhabited. However, we can try to approximate it, we can enumerate all possible lambda-terms for the given “deep structure” type and keep only those which can generate the correct word order. (We can use inhabitation machine, which have been used for categorial grammars at least since (van Benthem 1991)

1. John deliberately hit Mary.

$$((2 \to c) \to D \to c) \to (1 \to E) \to D \to E$$

$$a)\ \lambda VP \lambda NP \lambda z.NP\ (d\ ((VP\ \lambda y.y)\ z)) :$$
$$((C \to C) \to D \to 2) \to (1 \to E) \to D \to E$$
$$b)\ \lambda VP \lambda NP \lambda z.NP\ ((VP\ \lambda y.d\ y)\ z) :$$
$$((2 \to 1) \to D \to C) \to (C \to E) \to D \to E$$
$$c)\ \lambda VP \lambda NP \lambda z.((VP\ \lambda y.NP\ (d\ y))\ z) :$$
$$((2 \to C) \to D \to E) \to (1 \to C) \to D \to E$$

Adverbs

What does this mean? It means an adverb first selects a sentence missing an NP *anywhere*, then a noun phrase juste before it which will be the subject *semantically*.

$$a)\ \lambda VP \lambda NP \lambda z. NP\ (d\ ((VP\ \lambda y.y)\ z)) :$$
$$((C \rightarrow C) \rightarrow D \rightarrow 2) \rightarrow (1 \rightarrow E) \rightarrow D \rightarrow E$$

$$\begin{array}{rl} \textit{John} & np(0,1) \\ \textit{deliberately} & (np(C,C) \multimap s(2,D)) \multimap np(E,1) \multimap s(E,D) \\ \textit{hit} & np(A,2) \multimap np(3,B) \multimap s(A,B) \\ \textit{Mary} & np(3,4) \end{array}$$

1. John deliberately Mary hit.
2. John deliberately Mary claims likes Susan.
3. John deliberately Mary hit the sister of.

“deliberately” occupies an NP position (already a bit strange!) to form an S, we

Adverbs

$$b)\ \lambda VP \lambda NP \lambda z.NP\ ((VP\ \lambda y.d\ y)\ z):$$

$$((2 \to 1) \to D \to C) \to (C \to E) \to D \to E$$

$$\begin{array}{rl} John & np(0,1) \\ deliberately & (np(2,1) \multimap s(C,D)) \multimap np(E,C) \multimap s(E,D) \\ hit & np(A,2) \multimap np(3,B) \multimap s(A,B) \\ Mary & np(3,4) \end{array}$$

1. Mary John hit deliberately.

2. Mary the friend of deliberately left.

3. Mary John gave the friend of deliberately a book.

This could be an extraposition sentence, but it is very strange to have an *adverb* license extraposition

Meaning: tt was deliberate *on the part of Mary* that John hit her

Adverbs

"deliberately" is a sort of noun postmodifier: it can occur after any noun and the noun it modifiers will do things deliberately

$$c)\ \lambda VP \lambda NP \lambda z.((VP\ \lambda y.NP\ (d\ y))\ z):$$

$$((2 \to C) \to D \to E) \to (1 \to C) \to D \to E$$

$$\begin{array}{rl} John & np(0,1) \\ deliberately & (np(C,2) \multimap s(E,D)) \multimap np(C,1) \multimap s(E,D) \\ hit & np(A,2) \multimap np(3,B) \multimap s(A,B) \\ Mary & np(3,4) \end{array}$$

1. John hit Mary deliberately.

2. The friend of Mary deliberately left.

3. The friend of Mary deliberately who lives in Paris left.

Adverbs

- The best approximations that we can obtain all suffer from overgeneration because non-commutativity is insufficiently enforced.
- Can we work around the problem using additional lexical entries?
- We can add *many* new lexical entries, by optionally replacing all occurrences of np\s by a new atomic formula, say *vp*.
- Apart from the ad hoc nature of this solution, we would essentially *double* the number of lexical entries for adverbs, verbs and prepositions - which already have a high number of formulas - for a single type of example. And many more will follow...

Coordination

4. John caught and ate a fish.

$$(np \to np \to s) \to (np \to np \to s) \to np \to np \to s$$

"and" is a transitive verb conjunction;
we can reject many of the possible surface structure lambda-terms directly (eg. NP as argument of TV or the string "and" spanning an NP position) for reasons similar to the adverb case. However, there is a new type of term, which looks superficially correct.

$$\lambda TV2.\lambda TV1.\lambda NP2.\lambda NP1.\lambda z.$$
$$NP1\,((TV1\,\lambda x.x\;\lambda y.y)(and\,((TV2\,\lambda v.v\;\lambda w.w)(NP2\,z))))$$

Coordination

$$a1 = (np \to np \to s) \to (np \to np \to s) \to np \to np \to s$$

$$a2 = (np \to np \to s) \to np \to np \to s$$

$$\dfrac{\dfrac{John}{np} \qquad \dfrac{\dfrac{\dfrac{caught}{np \to np \to s} \qquad \dfrac{\dfrac{and}{a1} \qquad \dfrac{ate}{np \to np \to s}}{a2}}{np \to np \to s} \qquad \dfrac{a\ fish}{np}}{np \to s}}{s}$$

$$(((and\, a)\ c)\ f)\ j$$

Coordination

$$a1 = (np \to np \to s) \to (np \to np \to s) \to np \to np \to s$$

$$a2 = (np \to np \to s) \to np \to np \to s$$

$$\frac{\dfrac{John}{np} \qquad \dfrac{\dfrac{\dfrac{caught}{np \to np \to s} \qquad \dfrac{\dfrac{and}{a1} \qquad \dfrac{\dfrac{\dfrac{\dfrac{x}{[np]^1} \qquad \dfrac{\dfrac{ate}{np \to np \to s} \qquad \dfrac{y}{[np]^2}}{np \to s}}{s}}{np \to s}\ I_1}{np \to np \to s}\ I_2}{a2}}{np \to np \to s} \qquad \dfrac{a\ fish}{np}}{np \to s}}{s}$$

$$((and\, \lambda y.\lambda x.((a\, y)\, x)\, c)\ f)\ j$$

Coordination

As observed by Kubota & Levine (2013), this produces
"John caught and ate a fish"
with semantics "John caught a fish and a fish ate John"

$$\dfrac{\dfrac{John}{np} \quad \dfrac{\dfrac{\dfrac{caught}{np \to np \to s} \quad \dfrac{\dfrac{and}{a1} \quad \dfrac{\dfrac{\dfrac{\dfrac{x}{[np]^1} \quad \dfrac{\dfrac{ate}{np \to np \to s} \quad \dfrac{y}{[np]^2}}{np \to s}}{s}}{np \to s}\, I_2}{np \to np \to s}\, I_1}{a2}}{np \to np \to s} \quad \dfrac{a\ fish}{np}}{np \to s}}{s}$$

$$(((and\, \lambda x.\lambda y.((a y)\, x)\, c)\, f)\, j = (((and\, (\mathbf{C} a))\, c)\, f)\, j$$

Coordination

- Again, non-commutativity is insufficiently enforced, but this time in the form of strange *semantics*.

- The problem is that we want to say that "and" takes two *transitive verb* arguments, whereas we can only say it takes two sentences each missing two noun phrases (with no restriction as to *where* they are missing).

- Adding a new atomic lexical entry (say *tv*), is again not an attractive option, since we would need many additional entries to handle cases like "John has understood and will probably implement Dijkstra's algorithm" (no extra work is needed in the Lambek calculus for examples of this kind)

Gapping

"and" takes two sentences missing a transitive verb, then a transitive verb to form a sentence, and it does so by "plugging" the transitive verb into the leftmost sentence missing a transitive verb

7. John studies logic and Charles, phonetics.

$$((np \to np \to s) \to s) \to ((np \to np \to s) \to s) \to (np \to np \to s) \to s$$

$$\lambda STV2.\lambda STV1.\lambda TV.\lambda z.((STV1\,\lambda O1\lambda S1\lambda x.(((TV\,\lambda w.O1\,w)\,\lambda v.S1\,v))\,x)$$
$$(and\,(STV2\,\lambda O2\lambda S2\lambda y.S2\,(O2\,y))\,z))$$
$$\equiv_\eta$$
$$\lambda STV2.\lambda STV1.\lambda TV.\lambda z.((STV1\,TV)$$
$$(and\,(STV2\,\lambda O2\lambda S2\lambda y.S2\,(O2\,y))\,z))$$

Gapping

$$b1 = ((np \to np \to s) \to s) \to ((np \to np \to s) \to s) \to (np \to np \to s) \to s$$

$$b2 = ((np \to np \to s) \to s) \to (np \to np \to s) \to s$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathit{John}}{np} \quad \dfrac{\dfrac{P}{[np \to np \to s]^1} \quad \dfrac{\mathit{logic}}{np}}{np \to s}}{s}}{(np \to np \to s) \to s}\,I_1 \qquad \dfrac{\dfrac{\mathit{and}}{b1} \quad \dfrac{\dfrac{\dfrac{\mathit{Charles}}{np} \quad \dfrac{\dfrac{Q}{[np \to np \to s]^2} \quad \dfrac{\mathit{phonetics}}{np}}{np \to s}}{s}}{(np \to np \to s) \to s}\,I_2}{b2}}{(np \to np \to s) \to s} \qquad \dfrac{\mathit{studies}}{np \to np \to s}}{s}$$

$$(((\mathit{and}\ \lambda Q.((Q\,p)\,c))\ \lambda P.((P\,l)\,j))\ s)$$

Gapping

$$b1 = ((np \to np \to s) \to s) \to ((np \to np \to s) \to s) \to (np \to np \to s) \to s$$

$$b2 = ((np \to np \to s) \to s) \to (np \to np \to s) \to s$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathit{John}}{np} \quad \dfrac{\dfrac{P}{[np \to np \to s]^1} \quad \dfrac{\mathit{logic}}{np}}{np \to s}}{s}}{(np \to np \to s) \to s}\, I_1 \quad \dfrac{\dfrac{\mathit{and}}{b1} \quad \dfrac{\dfrac{\dfrac{\mathit{Charles}}{np} \quad \dfrac{\dfrac{Q}{[np \to np \to s]^2} \quad \dfrac{\mathit{phonetics}}{np}}{np \to s}}{s}}{(np \to np \to s) \to s}\, I_2}{b2}}{(np \to np \to s) \to s} \quad \dfrac{\dfrac{\dfrac{\dfrac{x}{[np]^3} \quad \dfrac{\dfrac{\mathit{studies}}{np \to np \to s} \quad \dfrac{y}{[np]^4}}{np \to s}}{s}}{np \to s}\, I_3}{np \to np \to s}\, I_4}{s}$$

$$(((\mathit{and}\ \lambda Q.((Q\, p)\, c))\ \lambda P.((P\, l)\, j))\ \lambda x.\lambda y.((s\, x)\, y))$$

Gapping

"John studies logic and Charles phonetics" with meaning
"John studies logic and phonetics studies Charles"

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{\mathit{logic}}{np}
\quad
\dfrac{\dfrac{P}{[np \to np \to s]^1} \quad \dfrac{\mathit{John}}{np}}{np \to s}
}{s}
}{(np \to np \to s) \to s}\, I_1
\quad
\dfrac{
\dfrac{\mathit{and}}{b1}
\quad
\dfrac{
\dfrac{
\dfrac{\mathit{Charles}}{np}
\quad
\dfrac{\dfrac{Q}{[np \to np \to s]^2} \quad \dfrac{\mathit{phonetics}}{np}}{np \to s}
}{s}
}{(np \to np \to s) \to s}\, I_2
}{b2}
}{(np \to np \to s) \to s}
\quad
\dfrac{
\dfrac{
\dfrac{
\dfrac{x}{[np]^3}
\quad
\dfrac{\dfrac{\mathit{studies}}{np \to np \to s} \quad \dfrac{y}{[np]^4}}{np \to s}
}{s}
}{np \to s}\, I_4
}{np \to np \to s}\, I_3
}{s}
$$

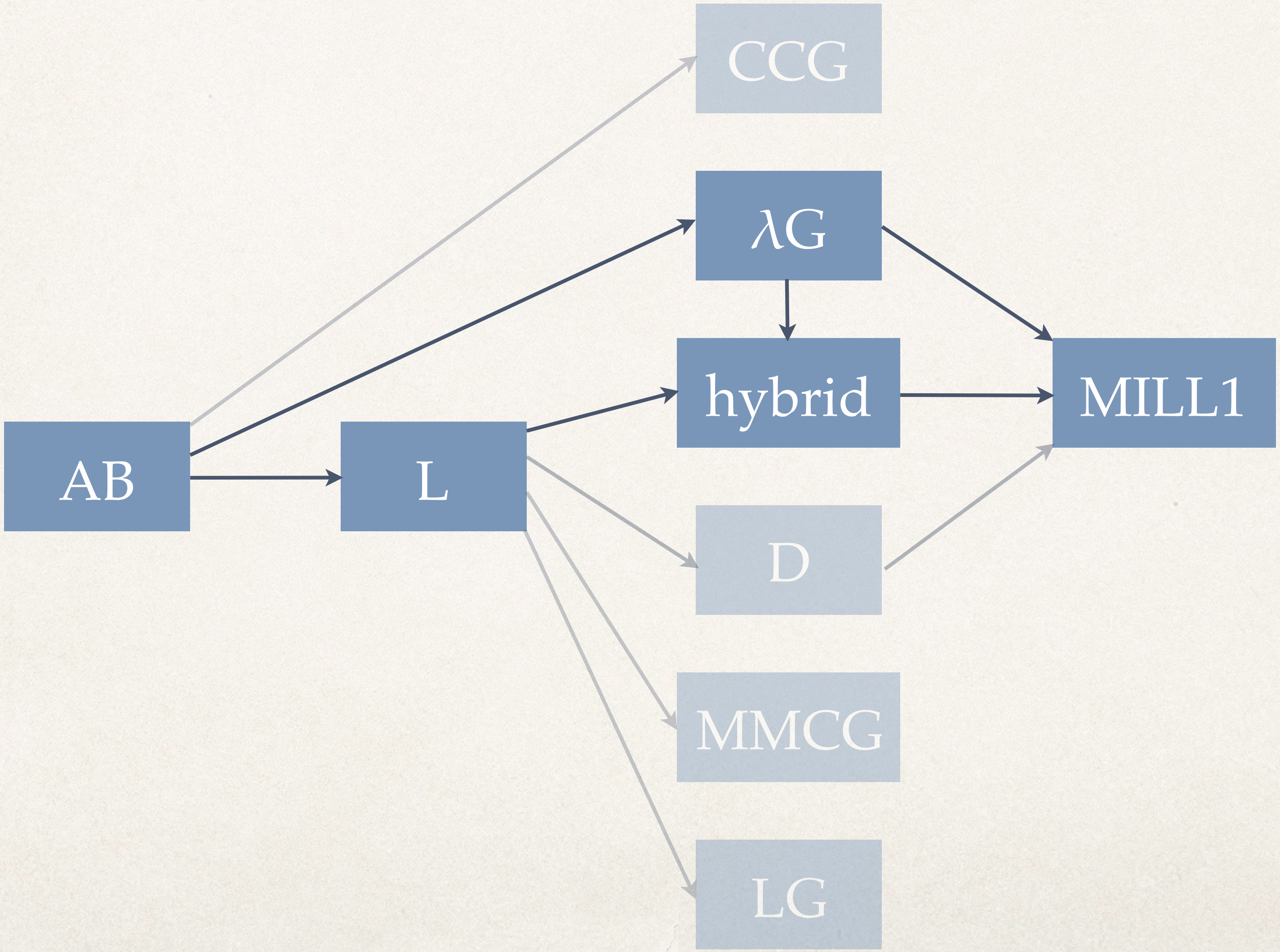
$$(((\mathit{and}\ \lambda Q.((Q\, p)\, c))\ \lambda P.((P\, j)\, l))\ \lambda x.\lambda y.((s\, y)\, x))$$

Solutions?

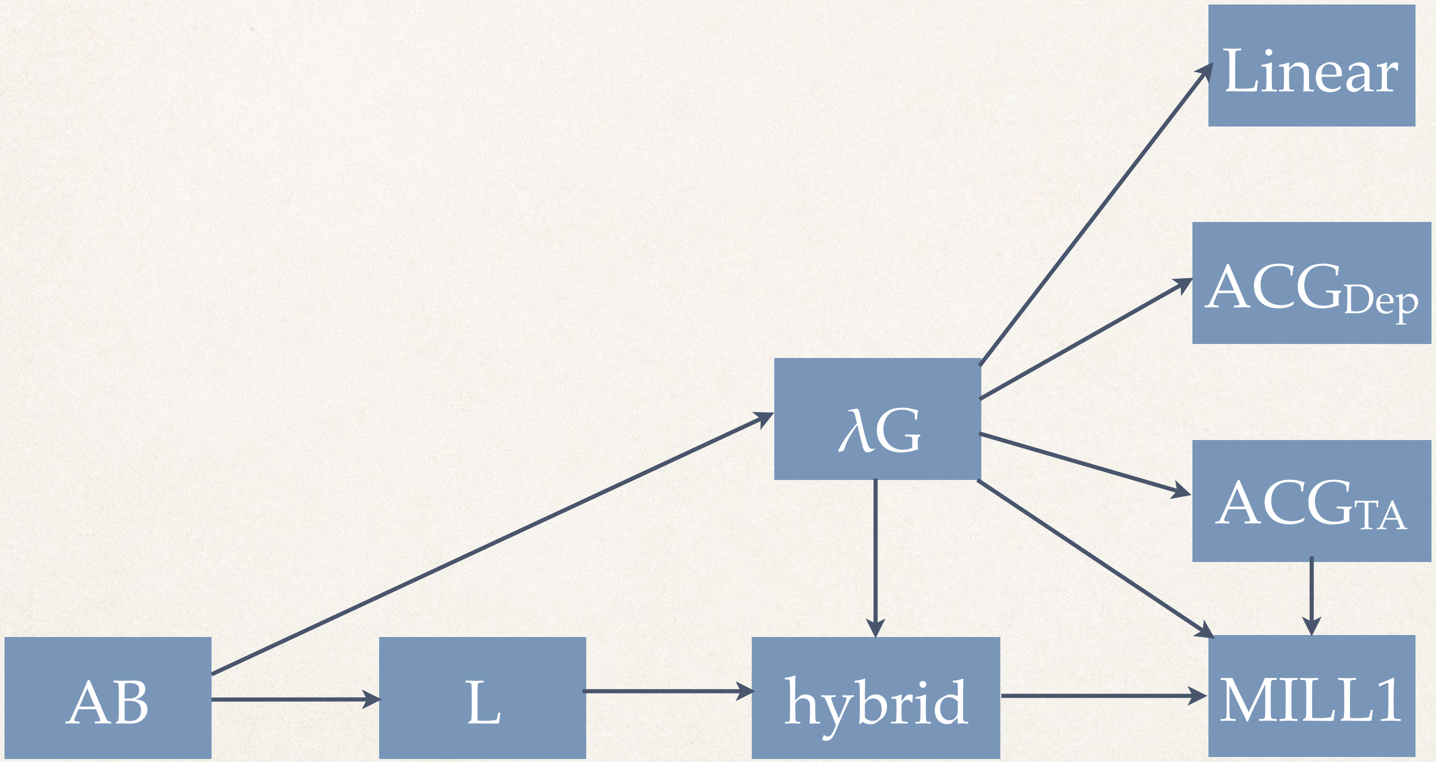
- Abandon type-logical deep structure and/or restrict the field of application of ACGs
- Hold lambda grammars to lower standards of adequacy than Lambek grammars
- Extend the formalism

Extending lambda grammars

The Lambek calculus and its extensions



Lambda grammars and variants/extensions



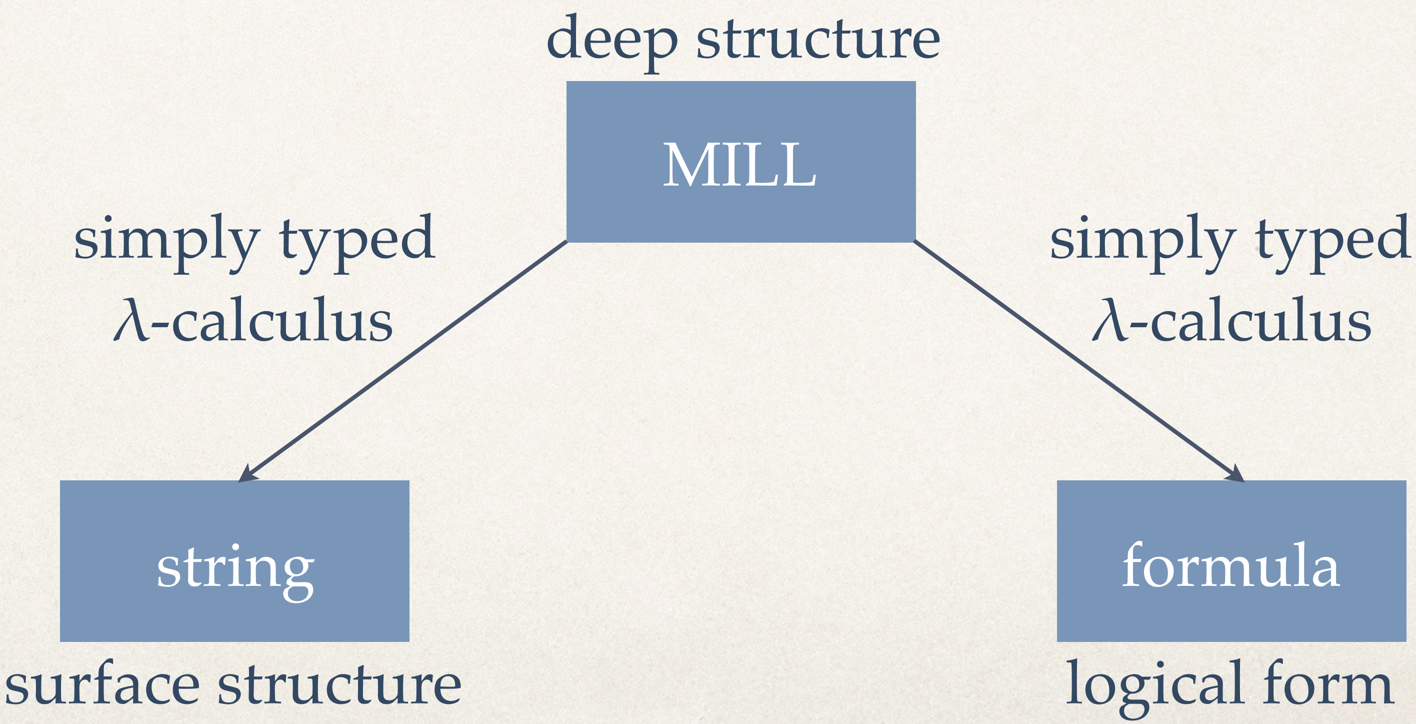
Extensions of lambda grammars

- ACG_{TA}: add tree automata (Kanazawa)
- Linear grammar: add subtyping and term constraints "phenomenators" (Worth & Pollard)
- Hybrid type-logical grammars: add Lambek calculus connectives (Kubota & Levine)
- ACG_{Dep}: add dependent types/terms (Pogodalla & Pompigne)
- First-order linear logic: add missing rules for introduction of the universal quantifier and elimination of the existential quantifier (Moot & Piazza)

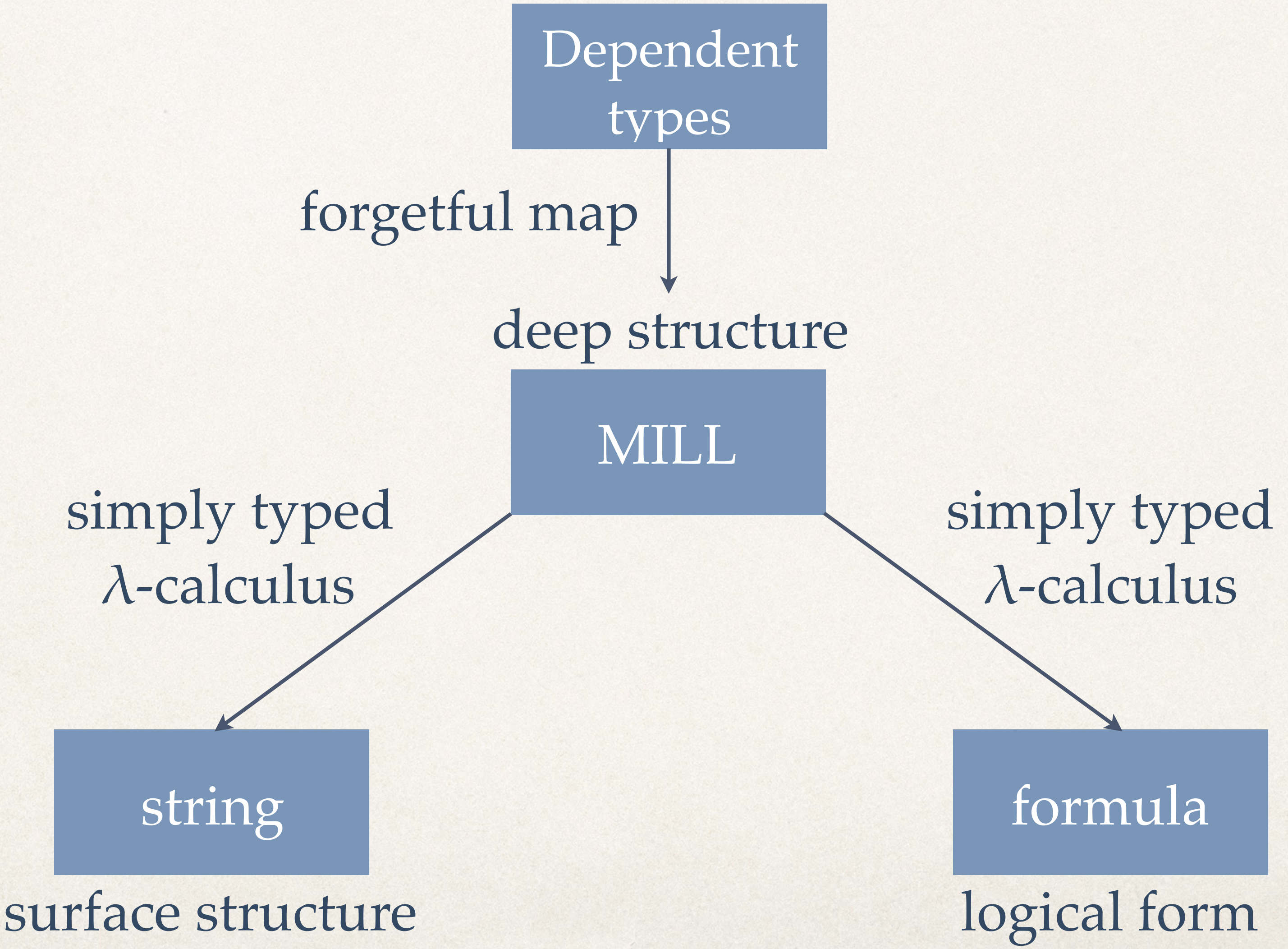
A brief comparison of the extensions

	Solves problems	Reasonable Complexity	Too powerful
ACG_{TA}	no	no?	no
Linear	yes	no?	?
Hybrid	yes?	yes	no
Dependent	yes	no?	probably
First-order	yes	yes	no

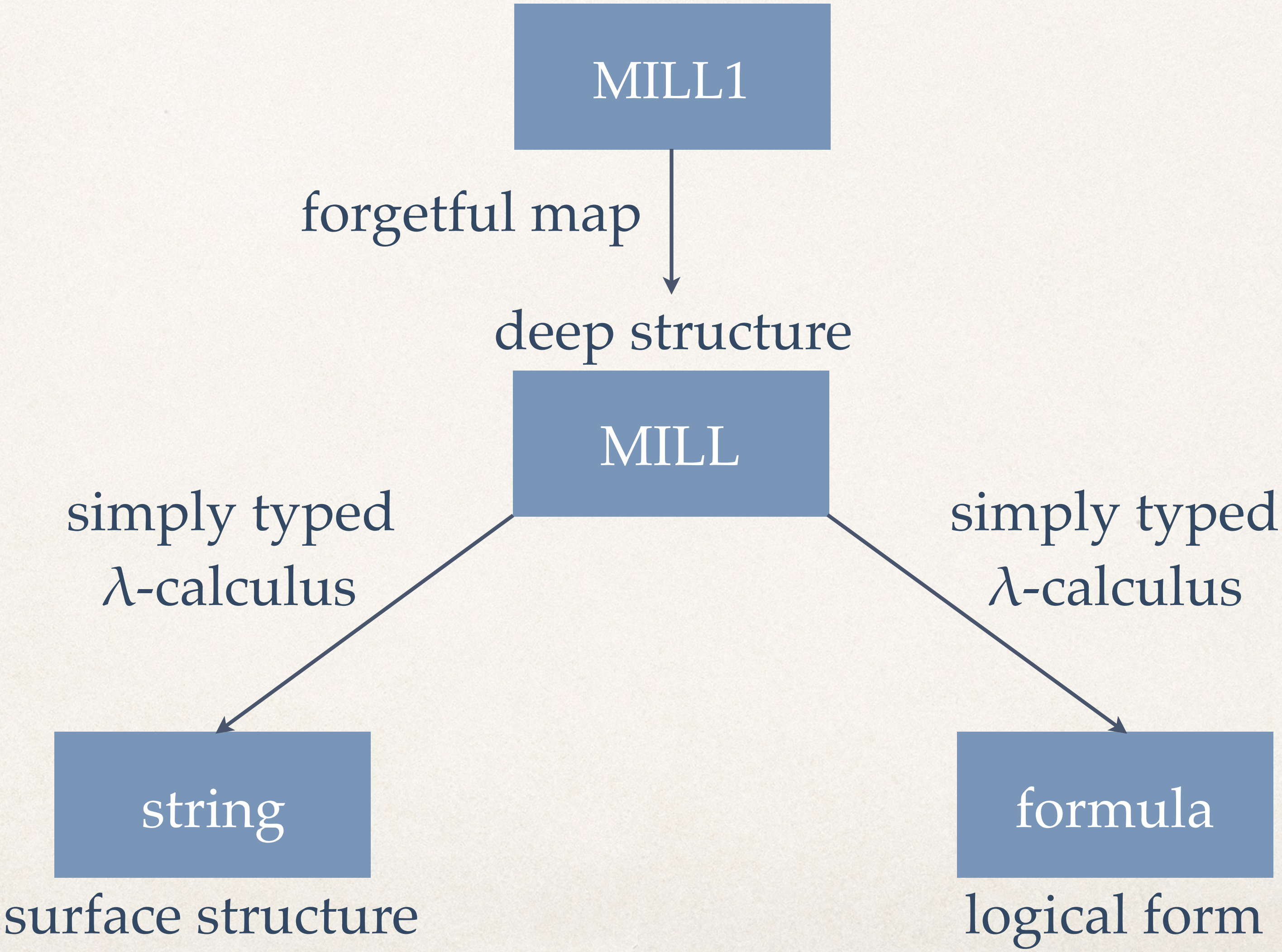
Standard architecture



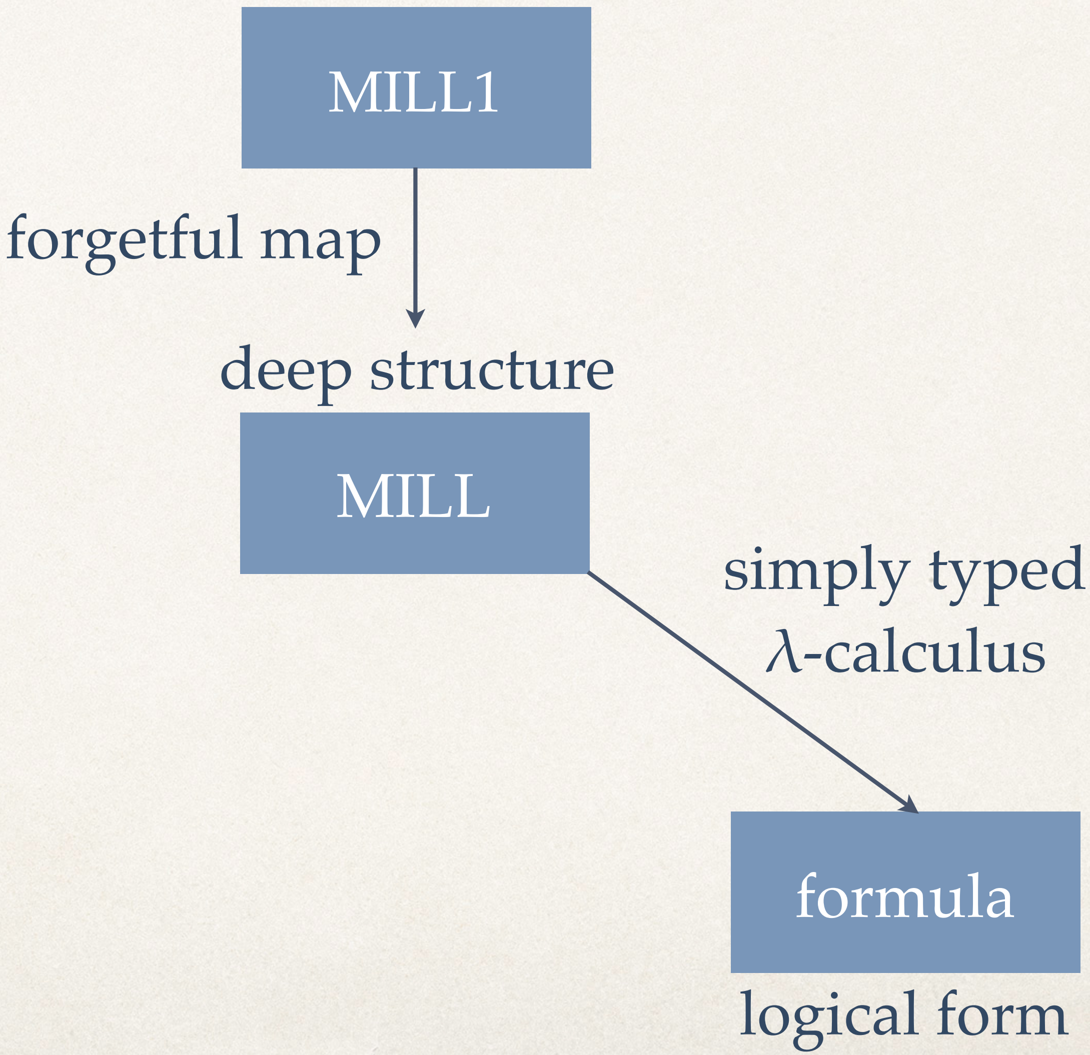
Dependent types



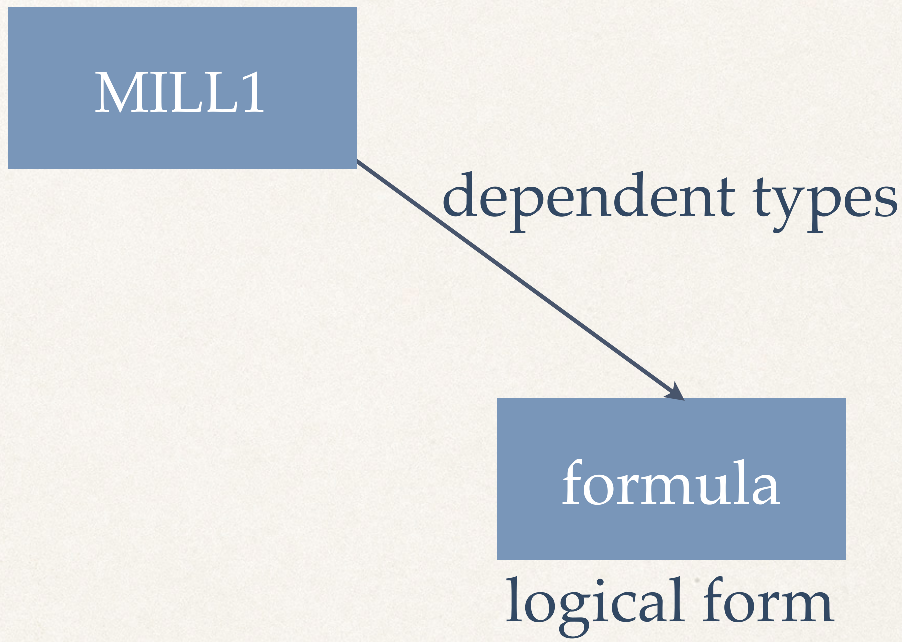
First-order linear logic



First-order linear logic



First-order linear logic



Conclusions

Open Questions

- Are there other extensions of lambda grammars which solve the problems without increasing the complexity?
- What are the relations of all these different logics to each other?
- Are there more empirical data for which these different formalisms differ in their predictions, and help us choose between them?
- What about formal language theory? We know almost no upper bounds for extended Lambek calculi, since the methodology of the Pentus proof does not extend to more complicated logics.

Conclusions

- One of the measures of the success of a theory is the number of its purported successors. In this sense the lambda grammar framework developed by Oehrle and others has been immensely successful.
- A number of potential solutions to the problems with lambda grammars has been proposed. I believe first-order linear logic is a good candidate for the underlying "machine language" of many grammatical logics.
- First-order linear logic is a natural *logical* extension of lambda grammars. Moreover it conveniently allows to to mix-and-match existing analyses from lambda grammars, hybrid type-logical grammars and the Displacement calculus.

References

- Haskell Curry (1961), Some logical aspects of grammatical structure, Structure of language and its mathematical aspects, 56-68.
- Jean-Yves Girard (2001), Quantifiers in linear logic II, `nuovi problemi della logica e della filosofia della scienza', Vol. II, CLUEB, Bologna.
- Philippe de Groote (2001), Towards abstract categorial grammars, Proceedings of the 39th annual meeting of the Association for Computational Linguistics.
- Joachim Lambek (1958), The mathematics of sentence structure, American Mathematical Monthly **65**(3), 154-170.
- Makoto Kanazawa (2011), Parsing and generation as Datalog query evaluation
- Yusuke Kubota & Robert Levine (2013), Empirical Foundations for Hybrid Type-logical Categorial Grammar, course notes, ESSLLI 2013.

References

- Michael Moortgat (2011), Categorial type logics, Handbook of logic and language, Elsevier, 95-179.
- Richard Moot (2013), Extended Lambek calculi and first-order linear logic
- Richard Moot & Mario Piazza (2001), Linguistic applications of first-order multiplicative linear logic, Journal of Logic, Language and Information **10**(2), 211-232.
- Glyn Morrill, Oriol Valentín and Mario Fadda (2001), The displacement calculus, Journal of Logic, Language and Information **20**(1), 1-48.
- Reinhard Muskens (2003), Languages, Lambdas and Logic, Resource sensitivity, binding and anaphora, Springer, 23-54.
- Richard Oehrle (1994), Term-labeled categorial type systems, Linguistics & Philosophy **17**(6), 633-678.

Why ACG_{TA} (Kanazawa, 2015) is not a solution

- ACG_{TA} suffers from overgeneration; notably it does not actually solve the problems it set out to solve
- ACG_{TA} suffers from undergeneration, even for linguistically relevant examples
- ACG_{TA} does not provide a treatment of discontinuous gapping which is superior to other type-logical grammars (eg. Kubota & Levine)

Overgeneration

1. Terry hates and Leslie likes Robin (right-node-raising)
2. What did Peter buy last week and throw away yesterday? (across-the-board extraction)
3. I wonder which song Peter composed yesterday and Susan sang today. (across-the-board extraction)

Overgeneration

1. Terry hates and Leslie likes Robin (right-node-raising)

2. What did Peter buy last week and throw away yesterday? (across-the-board extraction)

$$(np(0,0) \multimap s(0,R)) \multimap (np(0,0) \multimap s(L,0)) \multimap np(0,0) \multimap s(L,R)$$
$$\lambda P \lambda Q \lambda x.(P\,\epsilon) + and + (Q\,x)$$

Problem: the lexical assignments required for 2) and 3) have as an immediate consequence that 1) is predicted to have a reading meaning “Terry hates Robin and Robin likes Leslie”

Overgeneration

$$\dfrac{\begin{array}{l}\lambda s_0.\lambda t_0.t_0 + \text{hates} + s_0\\ \text{np}(0, V1) \multimap (\text{np}(U1, 0) \multimap \text{s}(U1, V1))\end{array}}{\begin{array}{l}\lambda s_0.\lambda t_0.t_0 + \text{hates} + s_0\\ \text{np}(0,0) \multimap (\text{np}(0,0) \multimap \text{s}(0,0))\end{array}}\ \forall E$$

$$\dfrac{\left[\begin{array}{l} r_0\\ \text{np}(0,0)\end{array}\right]^2 \quad \begin{array}{l}\lambda s_0.\lambda t_0.t_0 + \text{hates} + s_0\\ \text{np}(0,0) \multimap (\text{np}(0,0) \multimap \text{s}(0,0))\end{array}}{\begin{array}{l}\lambda t_0.t_0 + \text{hates} + r_0\\ \text{np}(0,0) \multimap \text{s}(0,0)\end{array}}\ \multimap E$$

$$\dfrac{\begin{array}{l}\text{terry}\\ \text{np}(0,0)\end{array} \quad \begin{array}{l}\lambda t_0.t_0 + \text{hates} + r_0\\ \text{np}(0,0) \multimap \text{s}(0,0)\end{array}}{\begin{array}{l}\text{terry} + \text{hates} + r_0\\ \text{s}(0,0)\end{array}}\ \multimap E$$

$$\dfrac{\begin{array}{l}\text{terry} + \text{hates} + r_0\\ \text{s}(0,0)\end{array}}{\begin{array}{l}\lambda r_0.\text{terry} + \text{hates} + r_0\\ \text{np}(0,0) \multimap \text{s}(0,0)\end{array}}\ \multimap I_2$$

$$\dfrac{\begin{array}{l}\lambda s_2.\lambda t_2.t_2 + \text{likes} + s_2\\ \text{np}(0, X1) \multimap (\text{np}(W1, 0) \multimap \text{s}(W1, X1))\end{array}}{\begin{array}{l}\lambda s_2.\lambda t_2.t_2 + \text{likes} + s_2\\ \text{np}(0,0) \multimap (\text{np}(0,0) \multimap \text{s}(0,0))\end{array}}\ \forall E$$

$$\dfrac{\begin{array}{l}\text{leslie}\\ \text{np}(0,0)\end{array} \quad \begin{array}{l}\lambda s_2.\lambda t_2.t_2 + \text{likes} + s_2\\ \text{np}(0,0) \multimap (\text{np}(0,0) \multimap \text{s}(0,0))\end{array}}{\begin{array}{l}\lambda t_2.t_2 + \text{likes} + \text{leslie}\\ \text{np}(0,0) \multimap \text{s}(0,0)\end{array}}\ \multimap E$$

$$\dfrac{\begin{array}{l}\lambda s_4.\lambda t_4.\lambda p_5.(t_4\,\epsilon) + \text{and} + (s_4\,p_5)\\ (\text{np}(0,0) \multimap \text{s}(0, Z1)) \multimap ((\text{np}(0,0) \multimap \text{s}(Y1, 0)) \multimap (\text{np}(0,0) \multimap \text{s}(Y1, Z1)))\end{array}}{\begin{array}{l}\lambda s_4.\lambda t_4.\lambda p_5.(t_4\,\epsilon) + \text{and} + (s_4\,p_5)\\ (\text{np}(0,0) \multimap \text{s}(0,0)) \multimap ((\text{np}(0,0) \multimap \text{s}(0,0)) \multimap (\text{np}(0,0) \multimap \text{s}(0,0)))\end{array}}\ \forall E$$

$$\dfrac{\begin{array}{l}\lambda t_2.t_2 + \text{likes} + \text{leslie}\\ \text{np}(0,0) \multimap \text{s}(0,0)\end{array} \quad \begin{array}{l}\lambda s_4.\lambda t_4.\lambda p_5.(t_4\,\epsilon) + \text{and} + (s_4\,p_5)\\ (\text{np}(0,0) \multimap \text{s}(0,0)) \multimap ((\text{np}(0,0) \multimap \text{s}(0,0)) \multimap (\text{np}(0,0) \multimap \text{s}(0,0)))\end{array}}{\begin{array}{l}\lambda t_4.\lambda p_5.(t_4\,\epsilon) + \text{and} + p_5 + \text{likes} + \text{leslie}\\ (\text{np}(0,0) \multimap \text{s}(0,0)) \multimap (\text{np}(0,0) \multimap \text{s}(0,0))\end{array}}\ \multimap E$$

$$\dfrac{\begin{array}{l}\lambda r_0.\text{terry} + \text{hates} + r_0\\ \text{np}(0,0) \multimap \text{s}(0,0)\end{array} \quad \begin{array}{l}\lambda t_4.\lambda p_5.(t_4\,\epsilon) + \text{and} + p_5 + \text{likes} + \text{leslie}\\ (\text{np}(0,0) \multimap \text{s}(0,0)) \multimap (\text{np}(0,0) \multimap \text{s}(0,0))\end{array}}{\begin{array}{l}\lambda p_5.\text{terry} + \text{hates} + \text{and} + p_5 + \text{likes} + \text{leslie}\\ \text{np}(0,0) \multimap \text{s}(0,0)\end{array}}\ \multimap E$$

$$\dfrac{\begin{array}{l}\text{robin}\\ \text{np}(0,0)\end{array} \quad \begin{array}{l}\lambda p_5.\text{terry} + \text{hates} + \text{and} + p_5 + \text{likes} + \text{leslie}\\ \text{np}(0,0) \multimap \text{s}(0,0)\end{array}}{\begin{array}{l}\text{terry} + \text{hates} + \text{and} + \text{robin} + \text{likes} + \text{leslie}\\ \text{s}(0,0)\end{array}}\ \multimap E$$

Undergeneration

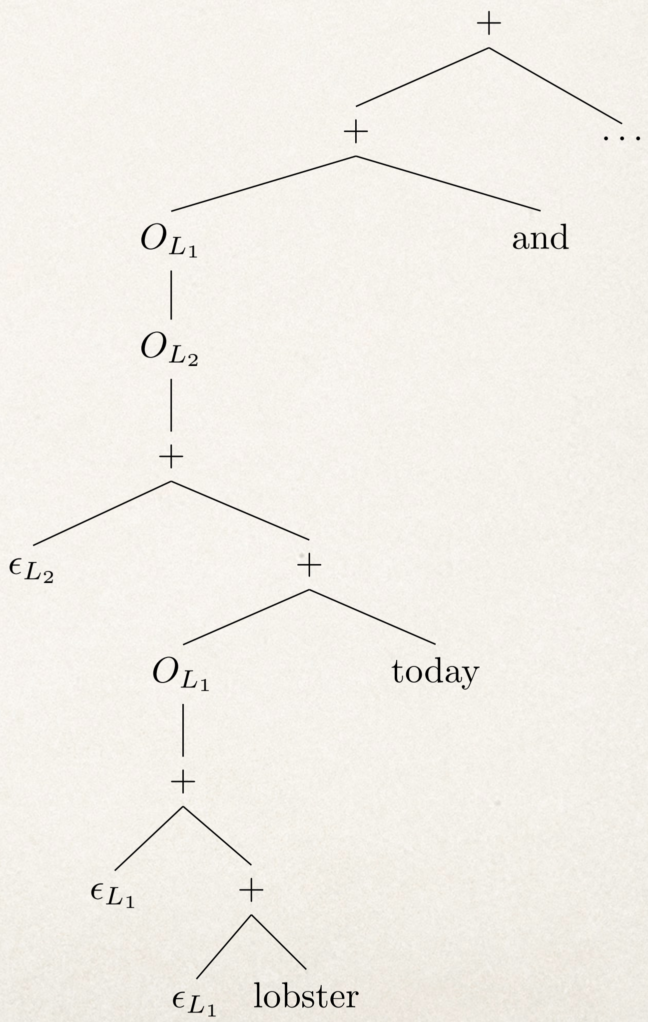
4. Captain Jack served lobster today and bananafish yesterday

$$\epsilon_{L_1} + (\epsilon_{L_1} + lobster)$$

$$\downarrow$$

$$(\epsilon_{L_1} + \epsilon_{L_1})$$

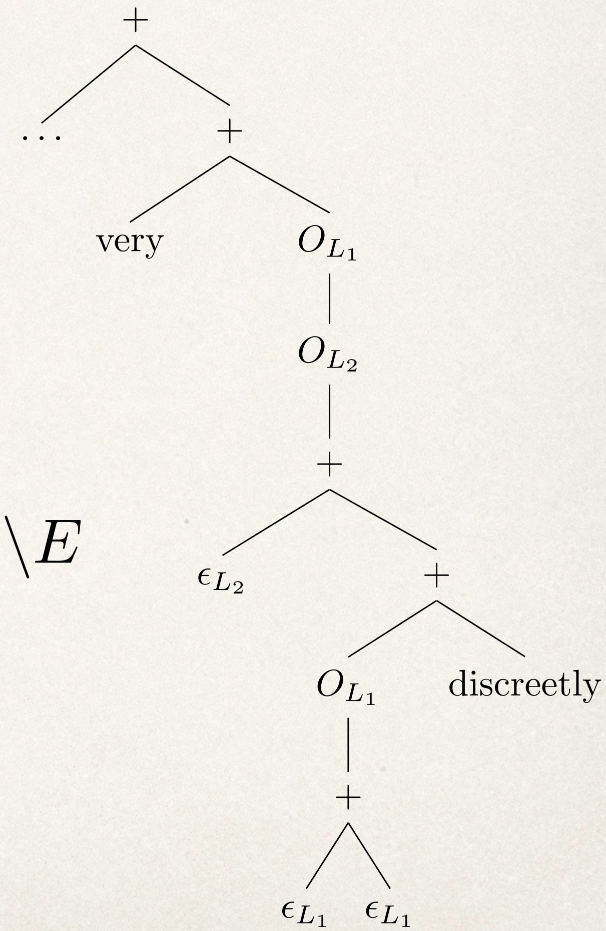
FAILS



Undergeneration

5. Happy slipped into the mansion very discreetly

$$\dfrac{\mathit{very} \quad \dfrac{\dfrac{\dfrac{[np]^2 \quad \dfrac{\dfrac{\dfrac{[np]^1 \quad [np\backslash s]^3}{s}\backslash E}{np\backslash s}\backslash I_1 \quad \dfrac{\mathit{discreetly}}{(np\backslash s)\backslash(np\backslash s)}}{np\backslash s}\backslash E}{s}\backslash E}{np\backslash s}\backslash I_2}{(np\backslash s)\backslash(np\backslash s)}\backslash I_3}{(np\backslash s)\backslash(np\backslash s)}/E$$



Gapping

1. *John met the vice-president of IBM and Betsy [met the vice-president of] Xerox. (problem for all analyses I know)

2. *John met the vice-president of IBM in France and [John met the vice-president of] Xerox [in] Italy. (problem for Kanazawa but not for others)

3. *John introduced the vice-president of IBM to the chairman of Xerox and [John introduced the vice-president of] Microsoft [the chairman of] Apple. (problem for Kanazawa but not for others)