PART 8: CATEGORIAL GRAMMARS

Categorial grammar. Adjukiewicz 1935, Bar-Hillel 1953

Bidirectional Categorial grammar:

CAT_B is the smallest set such that:

1. CAT_0 is a finite subset of CAT
2. If X, Y $\in$ CAT then X/Y $\in$ CAT and X\Y $\in$ CAT

Interpretation: if $\alpha \in X$ and $\beta \in X\backslash Y$ then $\alpha\beta \in Y$ left-concatenation
if $\alpha \in X/Y$ and $\beta \in Y$ then $\alpha\beta \in X$ right-concatenation

Uni-directional Categorial grammar:

CAT is the smallest set such that:

1. CAT_0 is a finite subset of CAT
2. If X, Y $\in$ CAT then X/Y $\in$ CAT

Interpretation: if $\alpha \in X/Y$ and $\beta \in Y$ then $\alpha\beta \in X$ right-concatenation

Examples: Bi-directional:

Assume basic categories: DP, N, S

D = DP/N
IV = DP\S
TV = IV/DP
ADJ = N/N
P = (IV/IV)/DP P = (N/N)/DP

Examples: IV = DP\S
IV forms an S with an NP on its left

TV = IV/DP
TV forms an IV with an NP on its right

Hence: If *Ronya* $\in$ DP, and *purrs* $\in$ DP\S then *Ronya purrs* $\in$ S.

Ronya +	*purrs*	$\rightarrow$	*Ronya purrs*
DP	DP\S		S

strokes +	*Ronya*	$\rightarrow$	*strokes Ronya*		
(DP\S)/DP	DP		(DP\S)		
		Pat +	*strokes Ronya*	$\rightarrow$	*Pat strokes Ronya*
		DP	(DP\S)		S

Uni-directional:

Assume basic categories IV, N, S

DP = S/IV
D = DP/ N
TV = IV/DP
ADJ = N/N
P = (IV/IV)/DP P = (N/N)/DP

Hence: If *Ronya* ∈ S/IV, and *purrs* ∈ IV then *Ronya purrs* ∈ S.

Ronya + *purrs* → *Ronya purrs*
S/IV IV S

strokes + *Ronya* → *strokes Ronya*
IV/(S/IV) S/IV IV

Pat + *strokes Ronya* → *Pat strokes Ronya*
S/IV IV S

Fact: (Bar-Hillel et. al. 1960)
Bi-directional categorial grammars are weakly equivalent to uni-directional categorial grammars.
Uni-directional categorial grammars are weakly equivalent to context free grammars.

Flexible interpretation and type shifting
Natural semantic interpretation for categorial grammar in type theory: correspondence between categories and types (Richard Montague, David Lewis).
e.g. IV = NP\S → <e,t>

Partee and Rooth 1983, Partee 1987, etc.: type shifting operations.
Systematic ambiguity: expressions can shift their interpretation systematically to different types via type shifting operations.
Klein and Sag: type driven interpretation: use type shifting operations to resolve type mismatches.

Example: Partee triangle for noun phrase interpretation:

type of generalized quantifiers - λP.P(*Ronya*)

$$\begin{tikzcd}
\langle\langle e,t\rangle,t\rangle & \\
 & \langle e,t\rangle \arrow[ul, "\text{EC}"'] \\
e \arrow[uu, "\text{LIFT}"] \arrow[ur, "\text{IDENT}"'] &
\end{tikzcd}$$

<e,t> **type of properties** - λx.x=*Ronya*

e
type of individuals - *Ronya*

Argument Positions	Predicate Position
e, <<e,t>,t>	<e,t>

IDENT[α] = λx.x=α   **LIFT**[α] = λP.P(α)   **EC**[α] = λP.∃x[α(x) ∧ P(x)]

Ronya	PURR	(PURR(*Ronya*))
type e	type <e,t>	type t

Noun phrases as predicates:
IDENT[Ronya] = λx.x=*Ronya*   the property of being Ronya

the	*winner*	→	(*the*(*winner*))
type <<e,t>,e>	type <e,t>		type e

be → λP.P   identity function of type <<e,t>,<e,t>>   (copula *be,* cf. Hebrew)

idea: Don't encode a special meaning into *be*: in the languages that have a copula, *be* is there for syntactic reasons, for instance, because the language does not allow noun phrases to occur as VPs without predicate marking.

is Ronya   apply λP.P to *Ronya*
type mismatch: resolve *Ronya* as **IDENT**(*Ronya*) = λx.x=*Ronya*
apply λP.P to λx.x=*Ronya*
is Ronya → λx.x=*Ronya*

The winner	*is Ronya*:
type e	type <e,t>
(*the*(*winner*))	λx.x=*Ronya*

(λx.x=*Ronya* (*the*(*winner*))

λ-conversion: $\underset{e}{(the(winner))} = \underset{e}{Ronya}$

Noun phrases as generalized quantifiers (Advanced semantics)
every → λQλP.∀x[Q(x) → P(x)]   the relation that holds between Q and P if Q is a subset of P

every lion → (λQλP.∀x[Q(x) → P(x)](LION))
λ-conversion:
every lion → λP.∀x[LION(x) → P(x)]   the set of properties that every lion has.

Ronya and every lion: **AND**$(\underset{\text{type e}}{Ronya}, \underset{\text{type <<e,t>,t>}}{λP.∀x[LION(x) → P(x)]})$

AND: type <a,<a,a>>, requires two arguments of the same type.
Type mismatch. *Ronya* resolves as **LIFT**(*Ronya*) = λP.P(*Ronya*)

Ronya and every lion: **AND**(λP.P(*Ronya*), λP.∀x[LION(x) → P(x)])
λP.P(*Ronya*) ∩ λP.∀x[LION(x) → P(x)]
λP. P(*Ronya*) ∧ ∀x[LION(x) → P(x)]
The set of properties that Ronya has and every lion has as well

Flexible categorial grammar:
-categories are a syntactic encoding of semantic categories.
-syntactic derivation represent derivations of grammatical sentences **with** a predictable meaning.
-flexibility in what fits together semantically
flexibility in what fits together syntactically.

Thus in flexible categorial grammars you extend categorial grammars with two kinds of rules:
-Category changing rules: rules that tell you that expressions of category A are also expressions of category B.
-Categorial combination operations like function composition, besides the standard or left and right concatenation. This can involve operations that are more powerful syntactically, or semantically.

Thus, Emmon Bach in the early 1980ies proposed to extend categorial grammar with wrapping rules, rules that wrap expressions into second position:

Left wrap: LWRAP[β, $a_1a_2\ldots a_{n-1}a_n$] = $a_1\beta a_2\ldots a_{n-1}a_n$
Right wrap: RWRAP[β, $a_1a_2\ldots a_{n-1}a_n$] = $a_1a_2\ldots a_{n-1}\beta a_n$

This lead to head grammars (Pollard), proven to be weakly equivalent to tree adjoining grammars by Vijay Shankar.

Extending the grammar with composition and lift operations was explored extensively in combinatory categorial grammar, e.g. Steedman 1987, 1996, 2000, Szabolcsi 1989, Jacobson 1999 (the Steedman variety has also proved to be weakly equivalent to tree adjoining languages).

Let us associate categories with lexical items:

Pat, S/IV	*strokes,* IV/(S/IV)	*Ronya,* S/IV

and think of these as **lexical axioms.**
We now think of application and composition as **proof rules**:

Application. If α, A/B and β, B occur in your proof, you can conclude αβ,A.
If α, A\B and β, B occur in your proof, you can conclude αβ, B.
(= **Modus Ponens**)

With this, we can derive:

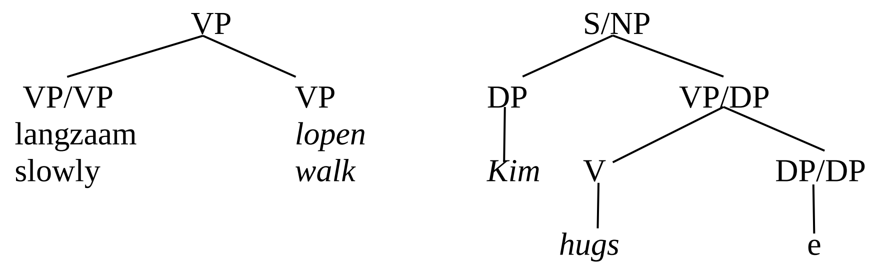
1. *strokes,* IV/(S/IV)	[axiom]
2. *Ronya,* S/IV	[axiom]
3. *strokes Ronya,* IV	[application]
4. *Pat,* S/IV	[axiom]
5. *Pat strokes Ronya,* S	[application]

The rule for composition is:

Composition: If α, A/B and β, B/C occur in your proof, you can conclude $\alpha\beta$, A/C

(See the papers in question for much discussion of the pro's and con's of different forward and backward versions of this.)

Now, there is an interesting dual perspective here on categories of the form A/B (see Landman 2006, and Advanced Semantics).
The standard tree perspective is that tree T/B is a tree that combines with tree B to form tree T, for instance, VP adverbials:



The other perspective, motivated by the semantic interpretation is that structure T/B is a T-tree with a B-tree missing in it, i.e. something that **would be** a tree T, if a tree of category B were sitting in it. And we interpret this as **a tree T with a B-gap in it.**

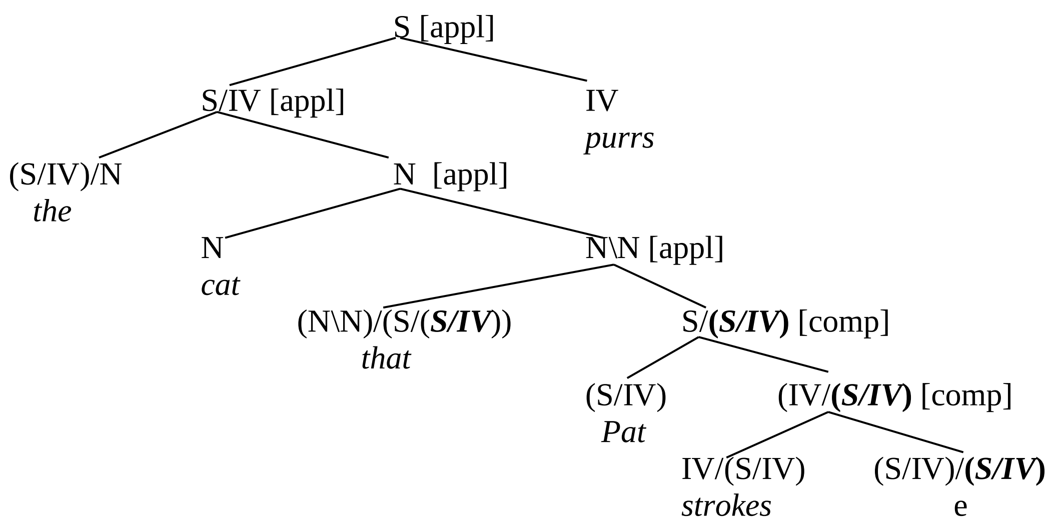
The smallest tree of this form would be [$_{B/B}$ e], a trace.
With this we can illustrate the basic treatment of gaps in combinatory categorial grammar:

Axioms:
cat, N *the*, (S/IV)/N *Pat*, S/IV *strokes*, IV/(S/IV) *purrs*, IV

that, (N\N)/(S/(S/IV)) (relativizer) e, (S/IV)/(S/IV) (gap)

We can derive:

1. *strokes*, **IV**/(S/IV)	[axiom]
2. e, (S/IV)/**(S/IV)**	[axiom] a DP gap
3. *strokes e*, (IV/(S/IV)	[composition] We derive an IV with a DP gap
4. *Pat*, S/IV	[axiom]
5. *Pat strokes* e, S/(S/IV)	[composition] We derive an S with a DP gap
6. *that*, (N\N)/(S/(S/IV))	[axiom] relativizer *that*
7 *that Pat strokes e*, (N\N)	[application] We introduce the top of the chain
8. *cat*, N	[axiom]
9. *cat that Pat strokes e*, N	[application]
10. *the*, (S/IV)/N	[axiom]
11. *the cat that Pat strokes e*, S/IV	[application]
12. *purrs*, IV	[axiom]
13. *the cat that Pat strokes e purrs*, S	[application]



Jacobson 1999 shows that this syntax has a natural semantics.
The gap [$_{DP/DP}$ e] is interpreted as the identity function at type <e,e>: λx.x.
Composition will derive a predicate for category S/DP: λx.STROKE(*Pat*,x), which is turned into a predicate modifier at the next higher level: λPλx.P(x) ∧ φ(x).
See Advanced Semantics for details.

Type Shifting Theory and Combinatory Categorial Grammar work rather bottom up with respect to the principles of the proof theory assumed: we do not know what the actual theory is, we discover in the course of studying syntactic and semantics phenomena what the operations are that we should assume.
Ideally the operations are natural operations (in the sense of Dowty 1982's grammatical operations): they have a natural mathematical interpretation, they typically apply in more than one grammatical domain, they are cross-linguistically robust, often lexicalized in one language but not in others.

Example: EC[α] = λy.∃x[α(x,y)]
love ⇒ *be loved* passivization = taking the second projection of a relation
Landman 2004 uses EC as a type shifting rule.

Proof theoretic grammar: (Lambek 1958, van Benthem 1987, see the exposition in Moortgat 1997).

Proof theoretic grammar is more ambitious. It provides not a list of typeshifting principles, or combinators, but a general logic of categories from which such principles can be proved.

Idea: a grammar is a set of formal proof rules G for proving that an expression is of category S.
Generation: α is generated by G if there is a proof in G that α is of category S.

The proof system derives a conclusion from an (ordered) set of premises:

$\alpha_{1,\ldots,}\alpha_n \Rightarrow \beta$: from premises $\alpha_1,\ldots,\alpha_n$ we derive β

The earliest proof system working this way was formulated in 1958 by Joachim Lambek. Lambek noticed the analogy between the categorial grammar principle
A, A/B ⇒ B and the principle of Modus Ponens: A, (A → B) ⇒ B. With this, he reformulated the categorial grammar of left and right concatenation as logical theory for deriving categories, and with that deriving inferences of the form:
if α is of category A, α is also of category B.
Which allows you to shift α, A to α, B which may be a category that you can combine easily with other categories. The proof theory is a Genzenstyle sequent calculus, which is close to natural deduction systems, and gives for each connective a rule for introducing it (**I**) and for exploiting it (**E**).

Lambek Calculus (Presentation and extensive discussion in Moortgat 1997)
($\alpha_1,\ldots,\alpha_n$ means: the premises come in that order but the order is assumed to be associative).

CUT	if ***X*** ⇒ A and ***Y***,A,***Z*** ⇒ B then ***Y,X,Z*** ⇒ B (cut)
ID	A ⇒ A
E/	if ***X*** ⇒ A/B and ***Y*** ⇒ B then ***X,Y*** ⇒ A
I/	If ***X*** ≠ Ø and ***X***,B ⇒ A then ***X*** ⇒ A/B
E\	if ***X*** ⇒ B and ***Y*** ⇒ B\A then ***X,Y*** ⇒ A
I\	if ***X*** ≠ Ø and B,***X*** ⇒ A then ***X*** ⇒ B\A
E•	if ***X*** ⇒ A•B and ***Y***,A,B,***Z*** ⇒ ***C*** then ***Y,X,Z*** ⇒ ***C***
I•	if ***X*** ⇒ A and ***Y*** ⇒ B then ***X,Y*** ⇒ A•B

Illustration of the principle I/.
This introduces a conditional proof.
Suppose you make a conditional assumption: e, DP
And with that assumption you derive α,S
Then you can derive α, S/DP **without** making the assumption e, DP.
But note, this is only warranted for derivations where you actually **are** making that assumption. This shows the proof theoretic approach to gaps:

1. *strokes,* IV/(S/IV)	[axiom]
2. (S/IV)	[**assumption**]
3. *strokes,* IV	[E/]
4. *Pat,* S/IV	[axiom]
5. *Pat strokes,* S	[E\]
6. *Pat strokes,* (S/(S/IV))	[I/, **withdrawl of the assumption**]
7. *that,* (N\N)/(S/(S/IV))	[axiom]
8. *that Pat strokes,* (N\N)	[E/]
9. *cat,* N	[axiom]
10. *cat that Pat strokes,* N	[E\]
10. *the,* (S/IV)/N	[axiom]
11. *the cat that Pat strokes,* S/IV	[E/]
12. *purrs,* IV	[axiom]
13. *the cat that Pat strokes purrs,* S	[E/]

All sorts of well known typeshifting rules are derivable in the Lambek calculus.

Lifting: $A \Rightarrow B/(A\backslash B)$
$A \Rightarrow (B/A)\backslash B$

We set: DP = S/IV, a DP takes an IV to its right to form a sentence.
But, of course, IV takes a DP to its left to form a sentence, which means that:

IV = DP\S = (S/IV)\S, which is what the second lifting principle says.

This has important semantic consequences, because the basic semantics associated with category IV may be of a lower type than that associated with (S/IV)\S.
See Advanced semantics for applications.

Composition: $(A/B)\bullet(B/C) \Rightarrow A/C$
$(C\backslash B)\bullet(B\backslash A) \Rightarrow C\backslash A$

Geach $A/B \Rightarrow (A/C)/(B/C)$
$B\backslash A \Rightarrow (C\backslash B)\backslash(C\backslash A)$

The Geach rule expresses compositon as (higher order) application:
Instead of **composing** α, A/B with β, B/C to get αβ, A/C,
we **apply** α, (A/C)/(B/C) to β, B/C to get αβ, A/C

See Moortgat 1997 for more principles.

Some facts:

FACT 1 (Lambek) The Lambek Calculus is decidable.

FACT 2 (Lambek) Cut-elimination: For every proof in the Lambek Calculus that uses CUT, there is a proof of the same thing that doesn't use cut.

This means that the proof theory has highly desirable logical properties like supporting interpolation normalization theorems.

Let us call a Lambek Grammar a set of lexical axioms, and for Lambek Grammar G, L(G) the set of expressions of category S derived from G by the Lambek Calculus (i.e. the intersection of the the closure of G under derivation in the Lambek Calculus with the set of expression of category S).

FACT 3 (Lambek) For every context free grammar G there is a Lambek Grammar that that is weakly equivalent to G.

Lambek, in fact, conjectured that the inverse is also true: the Lambek Calculus was **meant** to formulate the categorial grammars of left and right concatenation as a logical theory. We know that these grammars are weakly equivalent to context free grammars. This means that, **if** succesful, the Lambek Calculus would only allow Lambek grammars weakly equivalent to context free grammars.

FACT 4: (Mati Pentus, 1993) Every Lambek Grammar is weakly equivalent to a context free grammar.

So, the conjecture turned out to be true, but **remarkably hard to prove**!

FACT 5: Completeness proofs exist for the Lambek Calcules relative to various interpretations.

This is, of course, good news for semanticists who went into semantics in the first place because they couldn't deal with those endless syntactic proofs.

Thus, assuming that we interpret basic categories as sets of expressions we can define:

$[\![A\bullet B]\!] = \{\alpha\beta: \alpha \in [\![A]\!] \text{ and } \beta \in [\![B]\!]\}$
$[\![A/B]\!] = \{\alpha: \exists\beta \in [\![B]\!] \text{ such that } \alpha\beta \in [\![A]\!]\}$
$[\![B\backslash A]\!] = \{\alpha: \exists\beta \in [\![B]\!] \text{ such that } \beta\alpha \in [\![A]\!]\}$

With this semantics we can easily prove the validity of, say, the first composition principle:

$(A/B)\bullet(B/C) \Rightarrow A/C$

Proof:
Assume $\varphi \in [\![(A/B)\bullet(B/C)]\!]$.
Then $\varphi = \alpha\beta$ for some $\alpha \in [\![A/B]\!]$ and some $\beta \in [\![B/C]\!]$
Hence $\alpha \in [\![A/B]\!]$ and $\alpha\delta \in [\![A]\!]$ for some $\delta \in [\![B]\!]$
And $\beta \in [\![B/C]\!]$ and $\beta\gamma \in [\![B]\!]$ for some $\gamma \in [\![C]\!]$

Look at $\varphi\gamma$.
$\varphi\gamma = \alpha\beta\gamma$, $\alpha \in [\![A/B]\!]$ and $\beta\gamma \in [\![B]\!]$, hence $\varphi\gamma \in [\![A]\!]$
Since $\gamma \in [\![C]\!]$, it follows that $\varphi \in [\![(A/C)]\!]$.

Caveat: The sequent calculus presentation may obscure some features of the proof system that distinguish the system from more familiar proofsystems like that for classical logic. The logic is in some ways more like intuitionistic logic that like classical logic (no *ex false* inferences), in other ways it is more like linear logic (the order of the premises matters), in yet other ways it is quite unique: assumptions cannot be repeated, they can be used only once). In fact, when all the restrictions and conditions are make completely explicit, as is done in the natural deduction system given in (Ernst) Zimmermann 2006, the resulting system is surprisingly complex.

Some discussion:

The challenge of proof theoretic grammars is, of course, deriving **all and only** the wanted type shifting effects from a general proof theory. That is, it is one thing to propose a specific rule to create a specific effect where this effect is observed, and to argue that it is attractive to delegate this to a type shifting principle which is, we assume, a mathematically natural operation, and another thing to derive that principle from a much more general proof theory.
The worrysome question always is: what else do you get on the side?
The issue is highly non-trivial.
The Lambek calculus, for example, suffers from undergeneration **and** overgeneration, both syntactically and semantically.

Syntactically, we now know that indeed the Lambek Calculus is equivalent to contextfree grammars, and that means that it inherits the weakness of the latter: problems with discontinuous constituents, cross-serial dependencies, etc. But, of course, you cannot simply add some non-context free rules to deal with that: you must find a stronger logic, which does more, but not too much.

Concretely, with respect to the syntactic analysis of gaps, the combinatory categorial grammar frameworks and their extensions have developed comprehensive analyses of operator-gap constructions. The Lambek calculus actually only allows the gaps in certain peripherical positions. This means that the calculus can deal with *some* gap constructions, but not in a general enough way. Thus here too, the calculus undergeneralizes.

The calculus overgeneralizes too, because it cannot properly constrain the power that it has. That is, once you need an operation in one type of construction, it is hard to block its application in another type of construction where it is unwanted.

An instructive example based on an example by Paul Dekker is the completely ungrammatical (1):

(1) a. # The lover of and John thinks that honesty is badly missed.
b. [[[the lover of] and [John thinks that]] honesty] is badly missed.

Grammars based more on constituent structure than the Lambek calculus, will not generate sentence (1a), because they will not generate stucture (1b). But, given natural basic assumptions, the Lambek calculus **will** generate (1a) along the lines of (1b).

The reason is that the Lambek calculus gives a general account of non-constituent conjunction.
Look at (2):

```
(2) a.    the lover of   and  fighter for    honesty  is badly missed
    b. [[[the lover of] and [fighter for]] honesty] is badly missed
```

Given (2b) we need a category assignment to *the lover of*, which arguably is going to be:
(S/IV)/(S/IV) (i.e. DP/DP). This gives for (2a):

```
(2) a.    the lover of   and  fighter for    honesty  is badly missed
    b. [[[the lover of] and [fighter for]] honesty] is badly missed
          (S/IV)/(S/IV)      (S/IV)/(S/IV)
         ________________________________
                  (S/IV)/(S/IV)           (S/IV)
                  ________________________________
                                (S/IV)              IV
                                ______________________
                                          S
```

Next look at (3):

```
(3) a. John thinks that honesty and Mary thinks that prudence     is badly missed.
       [[John thinks that honesty] and [Mary thinks that prudence]] is badly missed.
        (S/IV)                                                 S/IV
        ___________________________________________________________
                                S/IV                                     IV
                                _________________________________________
                                                 S
```

The central point is this: the argument in (3) means that we cannot avoid analyzing *John thinks that honesty* as S/IV. We chose to identify DP with S/IV as well, but even if we hadn't, we can't avoid in the Lambek calculus giving DPs category S/IV as well.   You see that in the semantics: if *walks* $\in [\![\mathrm{IV}]\!]$ and *John walks* $\in [\![\mathrm{S}]\!]$, then *John* $\in \{\alpha\colon \exists\beta \in [\![\mathrm{IV}]\!]\colon \alpha\beta \in [\![\mathrm{S}]\!]\}$. This means that *John* and *Mary thinks that prudence* share a syntactic category and hence are conjoinable.

But if *Mary thinks that prudence* has category S/IV, and *prudence* has category S/IV (= DP), then *Mary thinks that* allows category (S/IV)/(S/IV) and is conjoinable with *the lover of* of category (S/IV)/(S/IV): conjoin, combine the result with S/IV *honesty*, and the result with the IV *is badly missed* and you get (1a):

```
(1) a. #  The lover of  and  John thinks that   honesty   is badly missed.
    b. [[[the lover of] and [John thinks that]] honesty] is badly missed.
         (S/IV)/(S/IV)       (S/IV)/(S/IV)
         ____________________________________
                   (S/IV)/(S/IV)              (S/IV)
                   __________________________________
                                  (S/IV)                 IV
                                  _________________________
                                               S
```

We see that the Lambek Calcucus undergenerated and overgenerates with respect to the syntax.

The same is true for the semantics as well. The Lambek calculus undergenerates, because, for instance some wide scope readings that Herman Hendrik's type shifting theory (discussed in Advanced Semantics) gets via Argument lift (lift the subject argument position of the verb first to the type of generalized quantifiers **and then** the object argument position, and combine with both arguments gives inverse scope), are beyond the capacity of the Lambek calculus.
This means that an appropriate analysis of inverse scope readings needs to be added to the Lambek calculus.

On the other hand, the Lambek calculus overgenerates because it cannot prevent application of available principles, where the reading doesn't exist.

The Lambek calculus allows lifting the Boolean operations like *and*, *not*, *or* to higher Boolean types:

Thus, we can lift the interpretation of IV *be smart* as follows:

be smart → SMART of type <e,t> lifts to: λT.T(SMART) of type <<<e,t>,t>,t> taking a DP meaning at the generalized quantifer type as argument.
Now we cannot avoid lifting negation to this type, and this means that we derive for *isn't smart*: (besides its normal meaning λx.¬SMART(x)):

isn't smart → λT.¬T(DANCE)

This allows us to deal with *all that glitters isn't gold* readings:

EVery cat isn't smART = Not every cat is smart

isn't smart	+	*every cat*	→	*Every cat isn't smart*
λT.¬T(BE SMART)		λP.∀x[CAT(x) → P(x)]		¬∀x[CAT(x) → SMART(x)]

However, the problem of the Lambek calculus is as before: if you can do it with one noun phrase, you can do it with other noun phrases as well, and hence you derive:

isn't smart	+	*some cat*	→	*Some cat isn't smart*
λT.¬T(BE SMART)		λP.∃x[CAT(x) ∧ P(x)]		¬∃x[CAT(x) → SMART(x)]
				No cat is smart

But *some cat isn't smart* doesn't have that interpretation.

And this is the problem. Since the theory derives what it allows from completely general principles, it can't distinguish different cases here, because it doesn't want to. But, of course, such distinctions are needed.

So if you want to maintain the proof theoretic perspective (and this is from a computational point of view very attractive), you must make the logic both stronger (beyond context free, and with more descriptive power), and weaker, more finegrained, distinguishing more cases from each other. But of course, the more finegrained you make it, the less it is a logic, and the more one wonders whether working from the other side (up from combinatory categorial grammar rather than down from the logical theories) may not be as attractive and maybe easier to work with.