(my  |  a linguist’s)
categorial grammar:
syntax made meaningful
are principles of grammar ‘domain’ specific? (riny)
(is ‘merge’ a general procedure? can grammar account for linguistic variation? does grammar define the class of possible languages?)

symmetry versus anti-symmetry = nature versus symbols? (riny)

is syntax derivational? (gereon)

is grammar declarative or procedural? (the ot paradox)

phonological form is messy, logical form is clean? (riny)

is economy an attribute of grammar? (riny, hans,
upcoming themes ...

what is the weak and strong translatability between frameworks? (gereon, hans, stephan, mary)

what is the labour division between the different components of the grammar? (all)

where is the information and is there any theoretical constraint on the type of information that is stored or generated? (mary)
among the founding fathers ...

logicians:
ajuduliewicz, lesniewski, bar-hillel, lambek, geach, lewis, montague, van benthem, morrill, ....

linguists:
moortgat, zwarts, hoeksema, steedman, bach, ...

costarring:
p. levelt, t.hoekstra ....
basic observations for categorialists

language refers (by necessity): being logically meaningful is the defining characteristic of human language

the (syntactic) structure of the meaningful expression is not homomorphic to the structure of its denotation

an expression is only meaningful if it enjoys (syntactic) structure
basic hypotheses for categorial grammar

linguistic meaning is propositional: *nur im zusammenhang* ..

linguistic meaning is computable

linguistic meaning is model-wise rather than world-wise
the ‘landscape’ in time

the polish

lambekian deductive

montegovian combinatory

multimodality

lambekian: intuitionistic type-logical calculus

montegovian: combinatory ‘opportunism’
the basic configurations

\[\begin{array}{c}
\alpha \\
\downarrow \\
\alpha/\beta \\
\alpha/(\alpha/\beta)
\end{array} \quad \xrightarrow{\sim} \quad 
\begin{array}{c}
f(a) \\
\downarrow \\
f \\
a
\end{array} \quad \equiv \quad \varphi
\]
the basic components

string algebra
lambda calculus
type logic
(generalized quantifiers)
generalized quantifiers

natural language meanings have algebraic structure: partially ordered sets

\[ GQ(A_\alpha) = \{ X_\alpha \mid R(A, X) \} \] for an algebraic \( R \)

\[ \text{every}(\llbracket N \rrbracket_{<e,t>}) = \{ X_{<e,t>} \mid \llbracket N \rrbracket \subseteq X \} \]

the algebraic structure is relevant to both the semantics (entailment) and the distribution of phrases (polarity)

the structures explain the ‘easy’ propositional interpretability of language (finite domains)

the algebraic structures can be typed - they (also) live in the syntax
restricting the universe

\[(\![s\ NP\ VP\ ]\] = R( \![NP]\ , \![NP]\ \cap \![VP]\ )\]
types: $e$, $t$ and family

invented to avoid paradoxes in set and function theory

(russell paradox: if $A = \{X \mid X \notin X\}$, is $A \in A$?)

introducing elementary syntactic categories on ontological basis,
introducing fundamental asymmetry in the elementary predicative
relation $\in$

the model theory for typed structures is:

let there be a non-empty set

types thus explain how abstract categories can be combines
meaningfully without encyclopedia

unfortunately, $e$ is not a category in natural language
if $\phi$ is of type $a$ and $x$ is a variable of type $b$, $\lambda x.\phi$ is a function of type $<b,a>$ - abstraction

$\lambda x_b.\phi_a(a_b) = \phi[a/x]$ - conversion

the linguistic variant of the church-turing thesis (lambda calculus equivalent to TM): every computable meaning is a lambda-term

so (typed) lambda-terms make semantics work - iff the syntax works
categorial syntax: the string algebra

only one operation: *concatenation*

strings are classified combinatorially, not categorically

if two strings have different concatenative properties, they are in different combinatorial categories (e.g. $v_2$ vs $v$-final sentences) if the concatenative properties are dynamically acquired (compare *agreement*)

the classification can be by explicit combinatorics or by unification
concatenation is defined by reference to combinatorial classes: **categories defined with respect to each other:** np, vp, vp/np, vp\np

**rules (or theorems):**

\[
\begin{align*}
A/B \cdot B & \Rightarrow A & A/B \cdot B/C & \Rightarrow A/C
\end{align*}
\]

**string interpretation of the rule:**

a string \(w\) is concatenated with a string \(v\) to a string \(wv\) of category \(A/C\) (A) iff \(w\) is of category \(A/B\) and \(v\) is of category \(B/C\) (B)
the string algebra ...

despite the concatenation process is defined by algebraic categories
since concatenation is linear there are two division operators: left and right (concatenation) languages are not closed under permutation).

the categories are assembled by product and division, and they behave likewise (full residuation):

\[ A \cdot B \Rightarrow C \quad \text{iff} \quad B \Rightarrow C \backslash A \quad \text{iff} \quad A \Rightarrow C / B \]

\[ 3 \cdot 5 = 15 \quad \text{iff} \quad 5 = 15 \mid 3 \quad \text{iff} \quad 3 = 15 \mid 5 \]

all ‘arithmetically’ true derivations may be interesting for concatenation: \( A / B \Rightarrow (A / C) / (B / C) \) but not all instances are ‘linguistically’ relevant and/or enlightening \( A \Rightarrow A / (B \backslash B) \)

thus: the need for restrictions on concatenation = restrictions on the cancellation of dividers (modalities)
the string algebra particularities....

there is no unit:

there is no category \( E \) such that for all \( A \),
\[
A \cdot E \Rightarrow A \text{ and/or } A/E \Rightarrow A
\]

consequently, the empty string has no category and is therefore not part of the string algebra

there is no complementation

the logic or modus operandi for the string algebra is resource sensitive (constructive, intuitionistic):

\[
\n\begin{align*}
\n\not\forall_{SA} w & \iff ww \\
\not\forall_{SA} A & \iff A \cdot A
\end{align*}
\]

compare:

\[
\n\begin{align*}
\n\not\forall_{A} 3 & = 3 \ast 3 \\
\n\n\not\exists_{\text{CPC}} p & \text{ iff } p \& p
\end{align*}
\]
the nature of categories

categories are sets of strings: categories are labels for properties of strings (e.g. categorial unification grammar)
syntactic categories are defined combinatorically, not with respect to their internal properties

combinatorically relevant properties of strings are only ‘categorized’ if they are part of the concatenation grammar (compare: v-position and agreement features)

sentential categories is delilah: 

\[
\begin{array}{c|c|c|c|c}
s & s_{sub} & s_{vn} & q & q_{sub} \\
\end{array}
\ldots
\]
the power of syntax

categorial grammar is a grammar of adjacency and conservation of linear order (steedman)
beyond context-free-ness: the grammar of discontinuity

discontinuity: lexically (and semantically) related elements are not adjacent in the string

discontinuity for categorial grammar: semantically related elements are not adjacent in the string but the gap between them can be bridged by operations on adjacent strings: in \(<X \ldots A \ldots B \ldots X>'\) X and X' can only be re-united by involving A and B in the compositional process
limiting the power

natural languages (tend to) avoid the degree of context-sensitivity of the mix-languages or the de-bruyne-sequences - the context-sensitivity in which every occurrence depends on every other occurrence or no occurrence can be licensed without inspecting every other occurrence.

mix-languages: simple context-sensitive languages closed under permutation like \( \pi( a^n b^n c^n) \)

de-bruyne-sequences: the set of all sequences over \{0,1\} with length \(2^n\) in which every sequence of length \(n\) occurs exactly once.

no categorial syntax for natural languages should give rise to this type of context-sensitivity or - dependency.

no categorial syntax should be such that it can only be parsed under random memory access.
there ain’t no truth outside the lexicon

there is no grammar apart from very general algebraic manipulation of lexically given modalized categories

categories are just labels for strings, not necessarily different from labels for complex symbols (compare minimalistic feature checking)

complex phrases (e.g. multi word expressions) have to be categorized in the lexicon: where else?
a categorial syntax

two categories (the primary and the secondary) merge into a third with the primary’s head by cancelling exactly one argument in the secondary category and appending the remnant arguments of the two categories

\[ s \setminus u \sim [np] / u \sim [vp^{\text{mode}}] \cdot \ vp \setminus u \sim [np, pp] / [] \Rightarrow \]

\[ s \setminus u \sim [np, pp, np] / [] \]

arguments don’t change sides (steedman)

only one argument is cancelled

argument stacks append, but don’t mix

the \( \sim \)-mode is an index for lexicality (unaffectedness) of the category
composition and disharmonic composition

normal composition  \( \frac{a}{b} \cdot \frac{b}{c} \Rightarrow \frac{a}{c} \) is an alternative to application

\[ \frac{a}{b} \cdot \frac{b}{c} \cdot c \Rightarrow \frac{a}{b} \cdot b \text{ or } \frac{a}{b} \cdot \frac{b}{c} \cdot c \Rightarrow \frac{a}{c} \cdot c \]

disharmonious composition  \( \frac{a}{b} \cdot b \cdot c \Rightarrow \frac{a}{c} \cdot c \) introduces new linearization

\[ c \cdot \frac{a}{b} \cdot b \cdot c \Rightarrow c \cdot \frac{a}{c} \cdot c \] has no alternative

disharmony is not a theorem of the Lambek calculus

disharmony pushes the Lambek calculus into permutation closure

disharmony can only be used in (combinatory) modal categorial grammar

disharmony makes Dutch work (to be proven)
the phenomena:
unorthodox solutions to problems

wh
v2
crossing dependencies (and adjuncts!)
dutch er
coordination
extended lexical units
wh in (my) categorial grammar

problem: how to move in the lexicon?

solution: (1) assign double types: one for the argument role ‘down right’ and one for the operator in ‘spec cp’ and (2) compose the right environment (disharmonically) such that the ‘hole’ is brought to the wh double category

die \Rightarrow (n\n)/s\_vn* np \quad (operator * argument)
... man die ik zag werken
n (n\n)/s\_vn* np np s\_vn\_np/vp vp vp np \Rightarrow
n (n\n)/s\_vn* np np___ s\_vn\_np,np np \Rightarrow
n (n\n)/s\_vn* np___ s\_vn\_np \Rightarrow
wh overview

single but complex category per *wh-* element (also vermaat 2007): double type \( X^*Y \) is not itself reducable

no type raising or manipulation of categories

possibly disharmonious composition

distinct combinatorial modus, therefore: distinct lexical categories for every extractable argument (lexical multiplication)

islands and non-extractable arguments are (partially) lexically marked

double role of *wh*-arguments properly expressed

definition of movement track: a *wh*-element \( A \) can be related to an argument position \( B \) iff the string between \( A \) and \( B \) allows for proper composition
v2: leave it to the lexicon

problem: finite verbs occur in essentially different positions, their arguments occur in at different sites and the position of the finite verb determines the combinatorics of the sentence

solution: finite verbs in different positions come with different categories, varying in the head-label and the direction of the arguments

arrangement in the lexicon: utterly complex lexical rules deriving the distinct finite categories (and related templates) (lexical multiplication)

motivation to leave it to the lexicon: the categorial algebra does
dutch = crossing dependencies

problem: there is no context-free (non-discontinuous) grammar for crossing dependencies and it is not movement

solution: disharmonious composition under modal control

\[ ... \ np \ np \ np \ s\{np/vp\ \ vp\{np/vp\ \ vp\{np \Rightarrow \]
\[ ... \ np \ np \ np \ s\{np/vp\ \ vp[[np, np] \Rightarrow \]
\[ ... \ np \ np \ np \ s[[np, np, np] \Rightarrow^* \ s \]

and because the categories stand for lexical constructions the semantics are pre-established and conserved under composition

adjuncts also enjoy disharmonious composition:

\[ ... \ np \ np \ np \ s\{np/vp\ \ vp/vp\ \ vp\{np/vp\ \ vp\{np \]
"er and painful grammar"

**problem:** argumentive *er* occurs discontinuously on almost every position to the left of its licenser and it may / must collapse with other *er’s*

**the real problem for parsers and generators:** you have to solve it

er hebben drie geleerden op gereageerd

* er hebben alle geleerden op gereageeerd
er hebben drie geleerden gereageerd

**solution:** (1) relaxing the argument stacks control for argumentive *er*
constructions: combine your way into and out of the lexicon

problem: syntactic and semantic structures don’t converge

solution: generate all (instances of all) constructions in the lexicon and treat them ‘normally’ ever after

e.g. all instances of 《 reflexive een {weg, pad, ...} pp intransitive_verb》

head:log:vconcept
arg1:[...:subject, theta:agens, log:subject, ...]
arg3:[...:subcat:refl, log:subject ... ]
log: subject by doing vconcept causes subject to move according to pp
category: vp\[np,np,np]/[] (.....s\[np/[np,np], vp_d\[np,np], .....)

coordination: the original

**problem:** coordination disturbs the arity (‘count’) in syntax and coordination is not structurally marked. One of the very first motivations for categorial algebra (Zwarts, Ades and Steedman, Wittenburg, Houtman ...)

**solutions:**
(1) lift coordinators to a type that makes two out one:
\[ X/X/X \]
**problem:** tractability - how to guess the right category for an unknown string?
(2) exploit pre-coordinative structure to feed an algorithm that interprets (restructures) coordination (Cremers, Grootveld ...)
the senses of categorial grammar

lexicalism: cg cries out for lexical data. it is construction grammar in an algebraic coat. it is an effective diagnostic grammar model for lexicalistic descriptivism.

parsed and generation: cg comes with in-built parsing strategies, imposed by the formalism. they serve as the backbone for parsing and generation (!) (www.delilah.eu)

acquisition: algebra by definition lives on elementary operations. these operations may give rise to a model of first language acquisition (mother daughter machine)

algebraic cognitivism: if the algebra of generalized determiners is part of the human devices, the combinatory rules may have deep roots too.
don’t let the lambdas catch you

don’t neglect them either: they are real