Categorial Grammar at a Cross-Roads

Johan van Benthem, University of Amsterdam & Stanford University

January 2003

Abstract Categorial grammars are driven by substructural logics. These are fragments of modal logics for the structures the grammar deals with. We will discuss modal language as a means of access to families of relevant structures: formal languages, type hierarchies, relation algebras–arrow models, and vector spaces. This is the cross-roads of our title, where open directions abound.

1 From categorial proof theory to categorial model theory

Categorial grammars are driven by resource logics in a proof format (van Benthem 1991, Buszkowski 1997, Moortgat 1997). Thus, they revolve around derivation and computation, with the Curry-Howard Gestalt switch taking proofs to type-theoretic denotations for the expression analyzed. But over the past decades, categorial logics have also been analyzed model-theoretically in modal logics with standard possible worlds-style models (cf. Kurtonina 1995). Thus, e.g., a categorial product $A \bullet B$ is 'true' of some object s iff s is a concatenation, or some suitable semantic merge of two objects t, u satisfying A, B, respectively. This is a standard binary modality, which needs a ternary accessibility relation R for its abstract truth condition:

$$\boldsymbol{M}, s \models A \bullet B \quad \text{iff} \quad \exists t, u\text{: } Rs, tu \;\&\; \boldsymbol{M}, t \models A \;\&\; \boldsymbol{M}, u \models B$$

Modal logic is a world of research rather different from the usual concerns in categorial grammar. What happens when we put the two agendas side by side?

2 From categorial logic to modal logic

2.1 Categorial and modal languages The main operations of categorial grammar are product $A \bullet B$ and directed functional slashes $A \rightarrow B$ (left-looking), $B \leftarrow A$ (right-looking). Other operations include Boolean conjunctions $\wedge$ plus further imports from linear logic. The matching modal language has Boolean operations, including disjunction and negation, plus three *dyadic existential modalities*

$$\bullet_1, \bullet_2, \bullet_3$$

These stand for different views of the ternary accessibility relation R: forming a versatile triple in the sense of Venema 1991:

2.2 Semantics in abstract models General modal models $\boldsymbol{M}$ have the form (S, R, V) with a set of objects S, a ternary accessibility relation R, and a valuation V for proposition letters. One can think of S, R in various ways, directed or not:

(a) expression s is the concatenation of the strings t, u in that order
(b) transition arrow s is the composition of the arrows t, u in that order
(c) information piece s is the merge of the information pieces t, u

And many other interpretations exist: cf. Adriaans 1990, de Haas 2001 on such models in object-oriented information systems. Now, without assuming any special conditions on R, the product modalties are interpreted as follows:

$\boldsymbol{M}, s \models A\bullet_1 B$ iff $\exists tu$: Rs, tu & $\boldsymbol{M}, t \models A$ & $\boldsymbol{M}, u \models B$

$\mathbf{M}, \mathrm{s} \models \mathrm{A}\bullet_2\mathrm{B}$ iff $\exists tu$: Rt, us & $\boldsymbol{M}, u \models A$ & $\boldsymbol{M}, t \models B$

$\boldsymbol{M}, s \models A\bullet_3 B$ iff $\exists tu$: Rt, su & $\boldsymbol{M}, u \models A$ & $\boldsymbol{M}, t \models B$

2.3 The translation Now we can translate the three basic operations of categorial grammar into this modal language as follows (Kurtonina 1995):

$T(A\bullet B) = A\bullet_1 B$ $\quad T(A\rightarrow B) = \neg(\neg A\bullet_2 B)$ $\quad T(B\leftarrow A) = \neg(\neg A\bullet_3 B)$

E.g., in terms of the above truth conditions, a categorial left-implication $A\rightarrow B$ works out to a modal implication which we right in an obvious first-order form

$\forall yz\, ((Ry, zx\ \&\ Az) \rightarrow By)$

This is precisely its natural meaning in modal semantics, which allows antecedents either from the left or the right. Further Boolean operations translate as they stand. But note that if extra categorial modalities encapsulating parts of syntax (Moortgat 1997) do not translate into the above general combination modalities, but in new modal operators, introducing additional accessibility operations into the models.

But, it also easy to see that the modal language can be translated back into the categorial one, provided we enrich the latter with all Booleans. E.g., obviously:

$A\bullet_2 B$ is equivalent to $\neg\,(A\rightarrow\neg B)$

2.4 Minimal modal logic The minimal logic of our modal product language over these general models consists of

(a) all validities of classical propositional logic
(b) distribution of each modality over disjunction in both arguments
(c) a modal necessitation rule – whose exact form is irrelevant here.

This suffices for models with three separate accessibility relations for the three modalities. But as we have just one such relation, with modalties using it from three perspectives, some special axioms ensure the proper relationships between the modalities, of which we display one typical example – with '*T*' for "true":

$(C\ \&\ (A\bullet_1 B))\ \rightarrow\ (A\ \&\ (C\bullet_2 B))\bullet_1 T$ *Coherence*

The latter axioms ensure that 'triangles' *Rs, tu* cohere when looked at from any of their vertices. The result is a well-known modal modal proof theory with pleasant completeness theorems (cf. Blackburn, de Rijke, & Venema 2001).

Here is the base connection (Kurtonina 1995), where *NLC* is the non-associative Lambek calculus, and $\rightarrow$ now stands for standard Boolean implication:

Theorem $NLC \mid\!- A \Rightarrow B$ iff $T(A) \rightarrow T(B)$ is in the minimal modal logic.

2.5 Matching up stronger categorial and modal calculi Further axioms on top of the minimal base give rise to a landscape of dyadic modal logics just as for unary modal logic. In particular, one important extra axiom is the following:

$(A\bullet_1(B\bullet_1 C)) \leftrightarrow ((A\bullet_1 B)\bullet_1 C)$ *Associativity*

This corresponds to a pair of constraints on the above relation, which may be computed automatically by the usual modal techniques of frame correspondence:

$\forall xyzu$: $((Rx, yz \;\&\; Rz, uv) \rightarrow \exists s: (Rs, yu \;\&\; Rx, sv))$

$\forall xyzu$: $((Rx, yz \;\&\; Ry, uv) \rightarrow \exists s: (Rs, vz \;\&\; Rx, us))$

These are abstract associativity principles for objects, allowing recombinations. Again, there is an important categorial–modal connection, because we are now at the level of strength of the primordial categorial calculus:

Theorem The following are equivalent:

(a) $LC \mid\!- A_1,\ldots,A_k \Rightarrow B$ (derivability in the Lambek Calculus)

(b) $T(A_1 \bullet_1 \ldots \bullet_1 A_k) \rightarrow T(B)$ holds on all associative ternary models.

More generally, structural rules in the categorial tradition correspond to frame constraints and their modal axioms. One more example is the Permutation Rule of the categorial calculus *LP*, which matches a *commutativity* axiom for the product modality $_1\bullet$, and hence commutativity of the ternary relation R in its two arguments.

A final point here. One tends to take existing categorial calculi, and look for matching modal logics. But the translation also works in the opposite direction, so that existing modal logics might suggest interesting categorial counterparts, provided categorial grammarians are willing to buy into the Boolean operations.

3 The two realms compared

The preceding observations embed categorial calculi in a broader modal world. One common feature between the two, already noted, is the shared interest in what may be called a *landscape of deductive strength*, with weaker or stronger calculi for various purposes. But the modal world also has its own flavour.

3.1 Landscape of expressive strength For instance, there is also a *landscape of expressive strength*, as one can vary the vocabulary, and introduce more modal operators over models of the same signature, or richer signatures. Typical examples are the 'universal modality' quantifying over all points of a model, accessible or not, or the still more expressive 'Since/Until' modalities of temporal logic. Such expressive extensions are suggested by the fact that the modal language itself can again be translated into a larger world, viz. that of *first-order logic*. The translation doing this will be obvious from the above truth conditions, which were already written in semi-first-order style. And one can even go further, enriching modal languages with non-first-order operators, as happens in modal *fixed-point languages* like the 'μ–calculus'. One example of a fixed-point modality with a categorial flavour would involve arbitrary finite composition of objects. After all, texts are finite compositions of sentences, the core business of categorial grammar.

3.2 The balance with complexity Inside the undecidable first-order logic, modal logics strive for a balance between expressive power and *computational complexity*. For instance, the minimal dyadic modal logic is decidable and indeed, its satisfiability problem is complete for polynomial space ('*PSPACE*-complete'). Precise outcomes vary according to both deductive and expressive strength, witness Spaan 1995, who finds modal logics anywhere between polynomial time and undecidability. E.g., the modal logic of associative product is *undecidable*, as it can encode syntactic word problems. And expressive extensions with new modal operators also tend to drive up complexity. But complexity also makes sense for other logical tasks than theorem proving or satisfiability testing. With *model checking*, it is first-order logic which is PSPACE-complete, while this task can be performed much faster for most modal languages, viz. in polynomial time.

3.3 Categorial complexity Now, categorial logics were one step down from modal ones qua expressive power, lacking Booleans. We expect then that their complexity behaviour will generally be better. E.g., given the earlier embedding, *NLC* is at most as complex as the minimal modal logic, but it does not need *PSPACE*, being decidable in polynomial time. Likewise, *LC* does not inherit the undecidability of the modal logic of associative product, but stays decidable in *NP* (and even *P* is still a live option). From a first-order or modal perspective, categorial languages are 'poor man's versions' of the full language, which may get by with very low complexity. There is an incipient literature on best possible simplifications of this kind (Spaan 2000, Kerdiles 2001).

3.4 Specific comparisons We will not pursue these general issues here. Instead, we now look at a number of specific models for categorial languages – cf. the survey in van Benthem 1991. These came in several varieties. *L-models* consist of strings, and category expressions describe formal languages. This is the syntactic world behind categorial grammar. On the semantic side, there are hierarchies of type domains, where products denote Cartesian products, and functional types denote function spaces. Let's call these *D-models.* But there were also two more surprising semantics for the categorial language. *R-models* involved dynamic state transitions, letting category expressions denote binary transition relations. And finally, *N-models* let category expressions denote vectors of numbers, such as the 'polarity counts' used in pruning hopeless branches from proof search trees. These varieties of categorial models will now be considered one by one. Each genre provides its own perspective on the essential content of categorial calculi.

4 Language models

The standard model for categorial syntax obviously consists of all strings over some given alphabet as the set *S*, with syntactic concatenation $s = t \frown u$ as the intended meaning of the ternary relation *R*. This model is infinite, but we might eventually also consider submodels, consisting, e.g., of all expressions observed so far in a learning procedure (van Benthem 2003).

4.1 Language models in general logic Such models occur in general logic: axiomatizing syntax is a first step in meta-theory of formal systems. Here is an interesting early result due to Quine, rediscovered in van Benthem 1991. Consider the model ***M****{a}* of all finite strings of *a's* with a ternary relation of concatenation. ***M****{a, b}* is the same syntactic model, but now with the strings taken from a two-symbol alphabet *{a, b}*. The difference in symbols is crucial:

Theorem The complete first-order theory of ***M****{a}* is decidable.
The first-order theory of ***M****{a, b}* is equivalent to True Arithmetic.

The first assertion follows from an isomorphism between ***M****{a}* and *(N, 0, S, +),* whose first-order theory is decidable additive arithmetic. But with two symbols, there is an effective reduction between the first-order theory of ***M****{a, b}* and that of *(N, 0, S, +, x)* with multiplication. The latter is undecidable and non-axiomatizable (and worse) by Gödel's Theorem. Thus, complete logics of the most elementary syntax can be rich and complex. But, categorial languages skirt this complexity!

4.2 Categorial semantics More precisely, *language models* consist of all expressions over some finite alphabet, with concatenation for R, and with special sets of expressions assigned to atomic types, viewed as proposition letters. The following recursion assigns languages to all complex types (cf. Buskowski 1997):

$$L_{A \bullet B} = \{ x{+}y \mid x \in L_A \;\&\; y \in L_B \}$$
$$L_{A \leftarrow B} = \{ x \mid \textit{for all } y \in L_B : x{+}y \in L_A \}$$
$$L_{A \rightarrow B} = \{ x \mid \textit{for all } y \in L_A : y{+}x \in L_B \}$$

This is the earlier truth definition. For any modal formula A, L_A is $\{x \mid \boldsymbol{M}, x \models A\}$. *Validity* of categorial sequents on language models may be defined as follows:

> On every language model, the language corresponding to the product of all premises in their stated order is contained in that for the conclusion.

The following result is due to Pentus 1993:

Theorem A sequent is LC-derivable iff it is valid on language models.

This completeness theorem refers to standard models, varying only the alphabet, and denotations for atomic types. This highly constrained class of syntax models has an obvious independent interest, also for modal logic in general.

4.3 Modal logic of language models One obvious concern in the wake of the preceding result is the *complete modal logic of language models.* Syntax models are different from the usual modal frames, and that makes them a new challenge. The modal theory has some chance of being axiomatizable, even where the first-order theory was not. Here, we just discuss some of its laws. First, note some simple definable notions in the modal language over these models (with 'T' for "true"):

(a) x is a symbol if it satisfies $\neg(T \bullet_1 T)$

(b) ϕ holds somewhere if any x satisfies $T \bullet_2 \phi$

(this gives us the 'existential' and 'universal' modalities E, U)

(c) ϕ holds in some part of x if x satisfies $\phi \vee (\phi \bullet_1 T) \vee (T \bullet_1 (\phi \bullet_1 T)) \vee (T \bullet_1 \phi)$

Here are some valid principles of modal logic over language models:

(1) the minimal versatile dyadic modal logic plus associativity

(2) the following Induction Principle

$U(\neg(T\bullet_1 T) \rightarrow \phi)\ \&\ U((\phi \bullet_1 \phi) \rightarrow \phi\,) \ \rightarrow\ U\phi$

(3) a linearity principle about decompositions of sequences:

if $Rx, yz\ \&\ R\,x, uv$, then either x is an initial part of u and v a final part of y, or u is an initial part of x and y a final part of v

An explicit modal syntax for axiom (3) needs some further auxiliary notions in the earlier vein. These three laws prove modal axioms expressing the *well-foundedness* of the proper substring relation ('finite depth'), as well as a similar principle of *finite width*, allowing only finitely many decompositions for any string.

4.4 Enriching the modal language Language models also suggest richer modal languages for syntactic patterns. E.g., text processing beyond reading or writing involves further structure than concatenation. An example is *reversal* of symbols in their order. One can introduce a new modality $<rev>\phi$ for this stating what holds of the reverse of the current sequence. This, too, has valid laws, such as

idempotence $<rev><rev>\phi \leftrightarrow \phi$

distribution $<rev>(\phi \vee \psi) \leftrightarrow <rev>\phi \vee <rev>\psi.$

Also, formal language theory in linguistics or computer science suggest a non-first-order extension to all *regular operations* $\{;\ ,\ \cup\ ,\ ^*\}$. We already have composition $;$ of languages via modal product, and choice $\cup$ via ordinary Boolean disjunction. But *iteration* * requires an language extension – e.g., as follows:

$\boldsymbol{M}, s \models <^*>\phi$ iff s can be written as a concatenation of some finite number of sequences each satisfying ϕ.

This can also be formulated more abstractly, in terms of R-combination of objects in general modal models (cf. van Benthem 1996, Venema 1996). In categorial grammar terms, iteration takes us to repeated sentences, i.e., linguistic texts.

Iteration is an instance of a genre of extension, by means of *fixed-point operations*. This makes for non-first-order extensions of modal logic such as dynamic logic or μ–calculus (Harel, Kozen & Tiuryn 2000). The μ–calculus is akin to modal logic and decidable. Logics like this were devised for analyzing processes, but they make just as much sense for analyzing syntactic structure. The logic of language models in the μ–calculus seems of obvious categorial interest. It will return in Section 5.5.

5 A concrete illustration: propositional formulas

In this intermezzo, we consider the universe *PROP-L* of propositional formulas. This language model shows how categorial considerations may be a bit more complex than what was suggested above. Also, it displays some connections with more standard logical concerns of a model-theoretic nature.

5.1 The model PROP-L The domain of our model consists of all strings over alphabet $\{p, \neg, \&\}$. There is one basic type t standing for the set of propositional formulas in prefix notation. This suffices for computing the denotations of all further types, for which we only take the right-looking categorial functor $A \leftarrow B$. Modal evaluation with $\leftarrow$ as above then tells us which strings are formulas, operators on formulas, all the way upward in the hierarchy of finite types. More generally, in language models, once we have fixed the denotation of the primitive types, the model determines the more complex ones. But this is not quite the usual categorial way of thinking. E.g., when defining Polish propositional formulas, one *stipulates* beforehand that $\neg$ has type $t \leftarrow t$ and $\&$ type $(t \leftarrow t) \leftarrow t$, and then *generates* types for further expressions via categorial combination. We return to this below.

5.2 Modal model checking The most basic thing to do in a model is checking truth of formulas. As noted in Section 3.2, model checking of modal formulas takes polynomial time in the size of finite models. This may be much harder in infinite models – but what about *PROP-L*? Prima facie, verifying a categorial implication (i.e., assigning a functional type) $t \leftarrow t$ for an expression x requires search through infinitely many expressions of type t to be right-concatenated with x. But we can do better, using the following uniformity property of our model:

Fact An expression E has type $t{\leftarrow}t$ iff the concatenation Ep has type t.

Here is a folk algorithm testing if a given string has type t :

> First, assign arity numbers to symbols as follows: p: -1, $\neg$: 0, $\&$: $+1$
> For any string, left to right, compute cumulative sums of symbol numbers:
> the formulas are then exactly those strings where the value -1 is reached
> for the first time, while the intermediate computed values were all ≥ 0.

This simple procedure works because of the inductive structure of the well-formed formulas. Again, we return to this additional feature of our model below. Indeed, we conjecture that model checking for our modal language in *PROP-L* is *decidable*. The folk algorithm gives us a lot of information about the type structure of the model, and we now proceed to some further illustrations of this.

5.3 Types that are realized In logic, a formula is called *realizable* in a model if there is an object in the domain which has it. In modal logic, a realizable type of a model $\boldsymbol{M}$ is a formula satisfied by at least one point in $\boldsymbol{M}$. Not all types of a categorial grammar or its modal logic need be realized in a given language model! The latter's modal theory may rule out some. In what follows, the modal language has all three categorial operations $\bullet$, $\rightarrow$, $\leftarrow$. Then it is easy to see that

Fact Type $t{\rightarrow}t$ is empty in *PROP-L*.

The reason is that formulas cannot be extended to the right to form new formulas. By the way, here is how the latter fact shows up in the modal logic of *PROP-L*:

$$\neg(\, t \,\&\, (t \bullet_1 T))$$

In any given model, it is of interest to determine which types occur at all:

Fact The only realizable types in *PROP-L* are the *first-order* ones
t , $t \leftarrow t$, $(t \leftarrow t\,) \leftarrow t$, … plus those *LC*-derivable from them.

To prove this, one needs the following valid multiplication table for binary arrow combinations $\rightarrow$, $\leftarrow$ of three atoms: t , T ("true"), $\perp$ ("false"):

(a) any implication ending in T becomes T

(b) implications ending in $\perp$ become $\perp$, except for the next case:

(c) any implication starting with $\perp$ becomes T

(d) this leaves implications $t \rightarrow t\ (:= \perp)$, $T \rightarrow t\ (:= \perp)$, $t \leftarrow T\ (:= \perp)$

(e) and finally $t \leftarrow t$, which is an acceptable first-order combination.

Another simple observation about *PROP-L* follows at once from the folk algorithm:

Fact Each object satisfies a product type of the form $\boldsymbol{t} \bullet (t \leftarrow \boldsymbol{t})$,
where the $\boldsymbol{t}$ are finite products of types: i.e., $\boldsymbol{t} = t^*$.

Of the many further semantic peculiarities of this model, we mention just this:

Fact Each object (i.e., string) has a finite type which uniquely defines it.

Thus, objects have *principal types*, from which all their other types follow in the modal theory of the model. This reflects the principal types of categorial grammar.

5.4 Inductive syntax Logical syntax pre-assigns types to symbols:

p gets t, $\neg$ gets $t \leftarrow t$, $\&$ gets $(t \leftarrow t) \leftarrow t$.

Thus, one now stipulates beforehand that some symbols have functional types, instead of discovering this by analyzing them in an already existing model. Using repeated Modus Ponens or function application, type t is then derivable for just the well-formed propositional formulas. Our modal language still works here. Let it have modalities for all categorial operations, a proposition letter t for the basic type, and 'nominals' *'p', '¬', '&'* holding for just these symbols in *PROP-L*.. We can then express properties of symbols with modal formulas such as:

$E\,(\text{'}p\text{'} \& \neg(T \bullet_1 T) \& t)$	p is a symbol of type t
$E(\text{'}\neg\text{'} \& \neg(T \bullet_1 T) \& (t \leftarrow t))$	$\neg$ is a symbol of type $t \leftarrow t$

Moreover, the modal logic encodes basic facts about propositional syntax. E.g., the earlier $\neg(t \& (t \bullet_1 T))$ says that initial segments of formulas can never be formulas. Thus, modal logic of *PROP-L* is at the same time elementary syntax.

Finally, how to reconcile the two viewpoints on *PROP-L*? Should we *stipulate* that $\neg$ has type $t \leftarrow t$, or *discover* it by model checking? The two views coexist in practice. One arrives at some types of expressions inductively, as in categorial learning algorithms. But once a stable assignment has emerged on finite samples, we use this to parse further expressions – revising only when forced. From a logical perspective, this requires looking at finite submodels of *PROP-L*, considering their convergence to the full model as they grow: cf. van Benthem 2003.

5.5 Modal fixed-points again The usual inductive syntax of propositional formulas underlying the above phenomena requires a fixed-point extension of our language, viz. the modal μ–calculus of Section 4.4. It can be written as follows:

$$t \leftrightarrow \mu q \bullet \text{ `}p\text{'} \vee (\text{`}\neg\text{'} \bullet_1 q) \vee ((\text{`}\&\text{'} \bullet_1 q) \bullet_1 q)$$

Here 'μ' is a *smallest fixed-point operator* defining the smallest set q of objects in the model satisfying an equivalence reflecting the usual inductive syntax definition:

$$q \leftrightarrow \text{`}p\text{'} \vee (\text{`}\neg\text{'} \bullet_1 q) \vee ((\text{`}\&\text{'} \bullet_1 q) \bullet_1 q)$$

Such a smallest fixed-point always exists, as the set operator computing new approximations from successive q defined by the right-hand side of the equivalence is upward *monotone* for set inclusion. The reason is that the only occurrences of the proposition letter q in $\text{`}p\text{'} \vee (\text{`}\neg\text{'} \bullet_1 q) \vee ((\text{`}\&\text{'} \bullet_1 q) \bullet_1 q)$ are syntactically positive.

Inductive definitions like this arise more generally with *context-free grammars*. These define a bunch of predicates – not just '*S*', but also auxiliary categories – in a simultaneous recursion, and the language produced is again the smallest fixed-point of the corresponding operation. Here is the logic behind this style of definition. *CFG*-languages are computed in finite stages, terminating by stage ω. Not all fixed-point definitions show this stabilization – but here is a sufficient condition:

Fact Smallest fixed-points are reached by approximation stage ω if their definition has the special syntax $\mu q \bullet \phi$ with ϕ constructed from just (a) atoms q, (b) arbitrary q-free formulas, (c) $\&$, $\vee$, and products $\bullet_i$.

6 Domain models

6.1 Type hierarchies Montague semantics interprets expressions in semantic domains with types matching their linguistic categories. Thus, models are set-theoretic type hierarchies, or in other words, standard models for typed lambda calculus. In the above terms, think of *D-models* whose universe S consists of some type hierarchy over base domains for primitive types, built up using in particular

$D_{A\bullet B}$ is the Cartesian product $D_A \times D_B$

$D_{A\rightarrow B}$ is the full function space $D_B{}^{D_A}$

More general models have function domains which are just subsets of the full spaces $D_B{}^{D_A}$. All these are again models of our modal sort, provided we set

Rx, yz iff x is the result of applying function y to argument z

The modal logic of these models will reflect some individual features of function application. For instance, associativity was valid for concatenation, but it makes no sense for function application: if f can be applied to g and h to $f(g)$ to form $h(f(g))$, then the rebracketed $h(f)(g)$ is ill-formed by type constraints. Function *composition* is associative, but this fits better with the 'arrow models' of the next section.

But there is also a striking difference with the earlier models. Domain models support *two* natural ternary composition relations. The first is the just-mentioned function application. But the second is the formation of ordered pairs:

Sx, yz iff x is the ordered pair (y, z)

In principle, the two ternary relations R, S are independent semantic primitives. But they can be mutually constrained by computing modal frame principles for key intended validities for Cartesian product and implication. Here is an example.

Fact $A \rightarrow (B \rightarrow C) \Rightarrow (A \times B) \rightarrow C$ corresponds to
$\forall xysuv: (Ry, xs \ \&\ Ss, uv) \rightarrow \exists z: Rz, xu \ \&\ Ry, zv)$

This may be proved using standard modal frame correspondence techniques.

6.2 Categorial interpretation The motivation for type-theoretic semantics is the soundness of the Lambek Calculus with permutation *LPC* for *D*–models. But there is an issue with reading sequent validity. In language models, this was a straightforward matter of set inclusion. But, e.g., a Lambek sequent $A, A \to B \Rightarrow B$ is not valid in this sense. For, objects in the Cartesian product D_A x $D_{A\to B}$ are not in the domain D_B: they just generate an object there via application. Interpretation this time is not via inclusion, but via *construction*:

> Given any bunch of objects in the domains for the premise types, there is an object lambda-definable from these in the domain for the conclusion type.

The pure modal logic of domain models is trivial, in that syntactically different types generate different semantic domains over non-trivial base sets. One might say that the other models that we consider identify types modulo mutual derivability.

This difference does raise the issue of the semantic role of categorial *derivations*. These live at a more fine-grained category-theoretic level, where constructions themselves are semantic objects, rather than just the syntactic strings or semantic state transitions that form the output of constructions. We will return to the issue of proof semantics versus mainstream modal semantics in Section 9.

7 Relational and arrow models

7.1 Dynamics In the late 1980s a surprising different interpretation came to light for the Lambek Calculus. Any process can be viewed as a set of possible transitions in some state space, i.e., as a binary relation between states. Now, one can interpret categorial expressions A as *binary relations* R_A over some set S, with product as relational *composition*, and the two arrows as its left- and right-inverses:

$$A \,;\, B \quad = \quad \{ (s, t) \mid \exists u\colon Asu \ \&\ But \}$$
$$A \to B \quad = \quad \{ (s, t) \mid \forall u\colon Aus \to But \}$$
$$B \leftarrow A \quad = \quad \{ (s, t) \mid \forall u\colon Atu \to Bsu \}$$

Any assignment of binary relations to atomic types then lifts to relations for all categories. These dynamic models of transition relations are a special case of the above ternary models, with objects being ordered pairs of states, and

$$R\ (s, t), (x, y)\ (u, v) \qquad \text{iff} \qquad s=x\ \&\ y=u\ \&\ t=v$$

This interpretation emphasizes the process character of syntactic combination, be it at a high abstraction level. Now, in these terms, a categorial sequent is *valid* if, in every R-model, the relational composition of all premise relations in the stated order is *included* in the relation for the conclusion. Andréka & Mikulás 1993 proved the connection with the Lambek Calculus:

Theorem A sequent is LC-provable iff it is relationally valid.

This shows that categorial calculi axiomatize a little corner of relational algebra or dynamic logic. As the complete first-order theory of composition plus all Boolean operations for binary relations is *undecidable* – we again have a well-chosen fragment in the sense of Section 3.3. But there is another perspective, too.

7.2 Arrow Logic The undecidability of relational set algebra reflects the rich structure of full power sets of the available transitions SxS. But one can also allow general models having only transitions satisfying some constraint. Then, relational algebra becomes decidable (Németi 1985). This road leads to *arrow models* (van Benthem 1991, Venema 1996, Kurtonina 1995). These are structures of the form

$$\boldsymbol{M} = (A, C^3, R^2, I^1, V)$$

with A a set of objects called 'arrows' carrying three predicates:

$C^3\ x,yz$	x is a composition of y and z
$R^2\ x,y$	y is a reversal of x
$I^1\ x$	x is an identity arrow

One can think of arrows as ordered pairs, or as morphisms in a category. Of course, in the latter case, reversal will not always be defined – but what follows does not presuppose this. Distinct arrows may encode the same pair of <input, output> , but also, not every such pair need match an available arrow. A modal language for arrow models is interpreted as usual, with key clauses for two modalities:

$\boldsymbol{M}, x \models \phi \bullet \psi$ iff there are y, z with $C\, x, yz$ and $\boldsymbol{M}, y \models \phi$, $\boldsymbol{M}, z \models \psi$

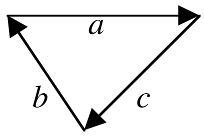
$\boldsymbol{M}, x \models \phi^{\check{}}$ iff there exists y with $R\, x, y$ and $\boldsymbol{M}, y \models \phi$

There is a minimal modal logic for such models without any constraints (cf. the cited literature for this, and what follows). On top of that, further axioms express constraints. In particular, assuming for convenience that reversal is a unary function r, two frame correspondences regulate the interaction of reversal and composition:

$(\phi \bullet \psi)^{\check{}} \rightarrow \psi^{\check{}} \bullet \phi^{\check{}}$ iff $\forall xyz\colon\ C\, x, yz \rightarrow C\, r(x), r(z)r(y)$

$\phi \bullet \neg(\phi^{\check{}} \bullet \psi) \rightarrow \neg \psi$ iff $\forall xyz\colon\ C\, x, yz \rightarrow C\, z, r(y)x$

Given this, there is no need for separate modal products any more, as we can view composition triangles like this from any arrow we please taking reversals:



Many axioms keep arrow logics decidable. An ominous threshold is *associativity*, which generally makes these logics undecidable, just as relational set algebra:

$(\phi \bullet \psi) \bullet \chi \rightarrow \phi \bullet (\psi \bullet \chi)$ iff $\forall xyzuv\colon (C\, x, yz\ \&\ C\, y, uv) \rightarrow \exists w\colon (C\, x, uw\ \&\ C\, w, vz))$

7.3 The categorial connection The second reversal/composition law above involves a sort of implication, reminiscent of the categorial law $A \bullet A \rightarrow B \Rightarrow B$. Indeed, the categorial to modal translation of Section 2.3, or the related truth condition in R-models of Section 7.1, extends to this setting. Here is the key clause:

Categorial $A \rightarrow B$ translates into arrow-logical $\neg(A^{\check{}} \bullet \neg B)$

Kurtonina 1995 shows how existing categorial logics become arrow logics in this way, and sometimes vice versa. But note how the burden of categorial explanation is now borne partly by the reversal operation for arrows. Thus, again, a dynamic interpretation of categorial grammar shifts the perspective somewhat.

7.4 Language extensions Like language models, arrow models may suggest additional operators for the modal language. For instance, there is a natural counterpart of the iteration operator to deal with unbounded repetitions of actions (cf. van Benthem 1996, chapter 8):

$\boldsymbol{M}, x \models \phi^*$ iff x is the result of some sequence of C-compositions from some finite set of arrows satisfying ϕ in $\boldsymbol{M}$.

Without Associativity, this does not say that x could be obtained by just every route of combinations from the given arrows satisfying ϕ.

Summarizing, arrow logic has its origins in concerns about the high complexity of relational algebra, which were remedied by finding broader model classes – sometimes even supporting extended vocabularies. It seems a plausible alternative modal language for general categorial analysis, whose vocabulary reflects a slightly different intuition about the source of the abstract models.

8 Vector models

8.1 Count invariants and numerical models Our final class of models take their pointy of departure in what seems a typical proof-theoretic feature of categorial derivation. Recall the computation of occurrence *count invariants* for categorial sequents (van Benthem 1991). Say, starting from two atomic types e, t , one computes 2-vectors *(e-count, t-count)* for each categorial expression:

$$e\text{-}count\ (e) = 1,\ \ e\text{-}count\ (t) = 0$$

$$e\text{-}count\ (A \bullet B) = e\text{-}count\ (A) + e\text{-}count\ (B)$$

$$e\text{-}count\ (A \rightarrow B) = e\text{-}count\ (B) - e\text{-}count\ (A)$$

Computing t-counts goes analogously. For Lambek-derivable sequents in *LPC*, all counts are the same for the product of all premises and the conclusion. Noting that computing counts is like inductively computing denotations in a domain of integer vectors, van Benthem 1991 introduced *N-models* where categorial expressions denote sets of such vectors. The rule of interpretation is this:

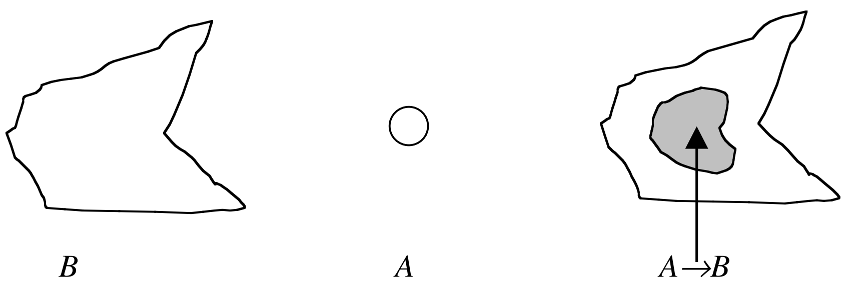
$$V_{A\bullet B} \quad = \quad \{x + y \mid x \in V_A,\ y \in V_B\}$$

$$V_{A \to B} \quad = \quad \{x \mid \forall y \in V_A: x+y \in V_B\}$$

As vector addition is commutative, the interpretation for the other arrow is similar. *LPC* is sound for interpretation in such models, with set inclusion for sequents. The converse question of completeness still seems open.

8.2 Geometrical models Vector models are like geometrical models, and this connection can be taken much further (van Benthem 1999, Aiello & van Benthem 2002). This leads to various analogies with *mathematical morphology*, the theory of basic operations on images, viewed as sets of vectors in *3*-dimensional real space IR^3. In this setting, categorial expressions denote regions in a space, and the above two operations now become vector addition of images for $A\bullet B$, and Minkowski *erosion* for $A \to B$, where the antecedent denotes a region used in 'smoothening' the image corresponding to the consequent.

Here is a picture of this sort of interpretation.



This time, categories are actual regions in space, and what used to be syntactic combination becomes geometrical transformation of figures.

But of course, these models interpret the whole language of arrow logic (Section 7.2), including arrow reversal as linear *vector inverse* $\neg X$. Moreover, they validate the stated arrow axioms. Axiomatizing the complete arrow logic of vector spaces is an open problem, though. In all, on this interpretation, categorial logic is neither about syntax nor dynamics. This time, it is all about geometrical structure!

8.3 Varying the language The geometrical perspective again suggests new modal operations. Boolean operations are also used in mathematical morphology. And iteration also makes sense. For any set of vectors denoted by A, the iteration A^* denotes all finite sums of such vectors, which amounts to taking an addition-closed subspace generated by the vectors. Of course, if we want true linear subspaces, then we must also find a modal or categorial reflection of scalar multiplications λX.

9 Comparisons and combinations

This concludes our survey of models for categorial logics. This variety itself raises some questions. Do the differences average out? Can categorial calculi such as *LC* be described as intersections of modal logics for different models? Or are the differences superficial, so that models are related, and their logics comparable?

9.1 Comparing model classes The following partial connections are from van Benthem 1991**.** From *L*-models to *R*-models, languages L_A over a set of strings can be mapped to binary relations on the domain of strings, now viewed as states:

$$R_A \quad = \quad \{(x, xy) \mid y \in L_A\}$$

This map is a homomorphism for Boolean operations, product $A \bullet B$, the right implication $B \leftarrow A$, plus the identity element. This makes the arrow logic of these operations a subset of their modal logic over language models. Next, the converse direction from *R*-models to *L*-models is harder. E.g., the following modal principle is valid on language models with strings, but not on dynamic models with binary relations (with '*Id*' for the language of just the empty string, or the identity relation):

$$(-(x \bullet x) \cap x) \bullet (-(x \bullet x) \cap x)) \cap Id = \bot$$

This has to do with the fact that operations in language models do not cut or delete, while those in dynamic models are more free. As stated earlier, the former are about reading and writing, the latter about more drastic text processing. The link between relational models and vector models is not straightforward either, as vectors in IR^k are equivalence classes of arrows between points modulo length and direction.

One might also look for coarser notions of modal similarity between the various kinds of model, on the analogy of *bisimulation* between modal models. (For specially fitted categorial bisimulations, cf. Kurtonina & de Rijke 1995, van Benthem 1996.) The above representation of strings as binary relations suggests such a connection. It induces a map F sending ordered pairs (x, xs) to strings s. It is clear that F preserves composition of transitions to concatenation of strings. Moreover, satisfies a zigzag clause: if a string $s = F((x, xs))$ is a concatenation $t^\frown u$, then the parts t, u are F-images of (x, xt), (xt, xs), respectively. This back-and-forth connection may be extended to the right implication $B{\leftarrow}A$. Thus, F is a functional surjective bisimulation for the modal language of $\{\bullet, \leftarrow\}$, and hence the complete modal theory of the relational modal is contained in that of the language model.

Nevertheless, a more systematic comparison of our different model classes seems rewarding. Arrow models might be the most promising unifying perspective here, of which the others kinds are specializations.

9.2 Combining model classes Another sense of unity arises by looking at realistic tasks. For instance, categorial grammars often do two things in tandem: syntactic analysis and semantic construction. This means that there is both an L-model and a D-model around. Strings in the language L_A have denotations in the type domain D_A. How strong is this connection? Without going into proofs and constructions, we cannot really mimick the details of a compositional semantics. But see van Benthem 2003 for more discussion of this point.

10 Conclusion

The cross-roads of our title refers to the fact that categorial grammar is a meeting point for a number of semantic worlds, such as syntax models, process models, and geometrical spaces. Its language is a small, but well-chosen fragment of the natural modal logic over these structures. Seeing things in this light places categorial grammar at an interesting place of the map, and raises a number of interesting new questions. Living at this cross-roads seems fine, provided we avoid the example of Stephen Leacocke's horseman, who "rode off madly in every direction".

11 References

P. Adriaans, 1990, 'Categoriale Modellen voor Kennissystemen', *Informatie*, 118–126.

M. Aiello & J. van Benthem, 2002, 'A Modal Walk through Space', in P. Balbiani, ed., issue of *Journal of Applied Non-Classical Logics* on spatial reasoning.

H. Andréka & S. Mikulas, 1993, 'The Completeness of the Lambek Calculus with Respect to Relational Semantics', *Journal of Logic, Language and Information* 3, 1-37.

J. Barwise & L. Moss, 1997, *Vicious Circles*, CSLI Publications, Stanford.

J. van Benthem, 1991, *Language in Action*, North-Holland, Amsterdam. Reprint with addenda, MIT Press, 1995.

J. van Benthem, 1996, *Exploring Logical Dynamics*, CSLI Publications, Stanford.

J. van Benthem, 1999, 'Logical Structures in Mathematical Morphology', manuscript, Institute for Logic, Language and Computation, Amsterdam.

J. van Benthem, 1999–2002, *Logic in Games*, lecture notes, ILLC Amsterdam & Stanford University.

J. van Benthem, 2003, 'Categorial Grammar revisited', in C. Casadio & P.J. Scott, eds., *Festschrift on the Occasion of Jim Lambek's 80th Birthday*.

P. Blackburn, M. de Rijke & Y. Venema, 2001, *Modal Logic*, Cambridge University Press, Cambridge.

W. Buszkowski, 1982, *Lambek's Categorial Grammars*, dissertation, Mathematical Institute, Adam Mickiewicz University, Poznan.

W. Buszkowski, 1986, 'Completeness Results for Lambek Syntactic Calculus', *Zeitschrift fuer mathematische Logik und Grundlagen der Mathematik* 32, 13-28.

W. Buszkowski, 1987, 'Discovery Procedures for Categorial Grammars', in E. Klein & J. van Benthem, eds., *Categories, Polymorphism and Unification*, Centre for Cognitive Science, Edinburgh & Institute for Language, Logic and Information, University of Amsterdam, 35-64.

W. Buszkowski, 1997, 'Mathematical Linguistics and Proof Theory'. In J. van Benthem & A. ter Meulen, eds., *Handbook of Logic and Language*, 638–738, Elsevier Science Publishers, Amsterdam.

M. Dalrymple, J. Lamping & V. Saraswat, 1993, 'LFG Semantics and Constraints', in S. Krauwer, M. Moortgat & L. des Tombe, eds., 1993, *Proceeedings 6th Conference of the European Chapter of the Association for Computational Linguistics*, Association for Computational Linguistics, 97-105.

H. Doets, 1996, *Basic Model Theory*, CSLI Publications, Stanford.

D. Gabbay, 1996, *Labelled Deductive Systems*, Oxford University Press, Oxford.

E. de Haas, 2001, *Logics for OO Information Systems*, dissertation DS-2001-03, Institute for Logic, Language and Computation, University of Amsterdam.

D. Harel, D. Kozen & J. Tiuryn, 2000, *Dynamic Logic*, The MIT Press, Cambridge (Mass.).

M. Kanazawa, 1994, *Learnable Classes of Categorial Grammars*, dissertation, Department of Linguistics, Stanford University. ILLC series 1994-08.

G. Kerdiles, 2001, *Saying it with Pictures, a logical landscape of conceptual graphs*, dissertation DS-2001-09, Institute for Logic, Language and Computation, University of Amsterdam.

N. Kurtonina, 1995, *Frames and Labels*, dissertation, OTS, University of Utrecht & ILLC, University of Amsterdam.

N. Kurtonina & M. de Rijke, 1997, 'Bisimulations for Temporal Logic', *Journal of Logic, Language and Information* 6, 403-425.

M. van Lambalgen, 1995, 'Natural Deduction for Generalized Quantifiers'. In J. van der Does & J. van Eyck, eds., *Quantifiers, Logic and Language*, CSLI Lecture Notes, vol. 54, Stanford University, 225–236.

M. Moortgat, 1997, 'Categorial Type Logics'. In J. van Benthem & A. ter Meulen, eds., *Handbook of Logic and Language*, 93–177, Elsevier Science Publishers, Amsterdam.

I. Németi, 1985, 'The Equational Theory of Cylindric Relativized Set Algebras is Decidable', Preprint No 63/85, Mathematical Institute, Hungarian Academy of Sciences, Budapest, 38 pp.

M. Pentus, 1993, 'Lambek Ghrammars are Context-Free, *Proceedings 8th Annual Symposium on Logic in Computer Science*, 429–433.

E. Spaan, 1993, *Complexity of Modal Logics*, dissertation, Institute for Logic, Language and Computation, University of Amsterdam.

E. Spaan, 2000, 'Poor Man's Modal Logic', in J. Gerbrandy, M. Marx, M. de Rijke & Y. Venema, eds., *JFAK, essays dedicated to Johan van Benthem on the occasion of his 50th birtday*, ILLC Amsterdam.

Y. Venema, 1991, *Many-Dimensional Modal Logic*, dissertation, Institute for Logic, Language and Computation, University of Amsterdam.

Y. Venema, 1996, 'A Crash Course in Arrow Logic', in M. Marx, M. Masuch & L. Pólos, eds., *Arrow Logic and Multimodal Logic,* Studies in Logic, Language and Information, CSLI Publications, Stanford, 3–34.