

EPISTEMIC CLOSURE AND COMMUTATIVE, NONASSOCIATIVE RESIDUATED STRUCTURES

SEBASTIAN SEQUOIAH-GRAYSON*

Formal Epistemology Project
University of Leuven

Forthcoming in Synthese

Abstract

K-axiom-based epistemic closure for explicit knowledge is rejected for even the most trivial cases of deductive inferential reasoning on account of the fact that the closure axiom does not extend beyond a raw consequence relation. The recognition that deductive inference concerns interaction as much as it concerns consequence allows for perspectives from logics of multi-agent information flow to be refocused onto mono-agent deductive reasoning. Instead of modeling the information flow between different agents in a communicative or announcement setting, we model the information flow between different states of a single agent as that agent reasons deductively. The resource management of the database of agent states for the deductive reasoning fragment in question is covered by the residuated structure that encodes the *nonassociative Lambek Calculus with permutation, bottom, and identity*: \mathbf{NLP}_{01} .

keywords: closure, epistemic logic, interaction, information, commutation, non-association, mobiles, NLP, residuated structures, categorical grammars, Lambek.

1 Introduction

That logic concerns interaction just as much as it concerns consequence is an insight has most eagerly been adopted in various logics of multi-agent information flow, such as Public Announcement Logics, and Dynamic Epistemic Logics

*Postdoctoral Research Fellow, Centre for Logic and Analytical Philosophy, *University of Leuven* - Belgium. Senior Research Associate, IEG - Computing Laboratory, *University of Oxford*. Research Associate, GPI - *University of Hertfordshire*.

and so forth (van Benthem (2009), Baltag, Moss, and Solecki (1998), Baltag and Smetts (2008), van Ditmarsh, van der Hoek, and Kooi (2008)). What we will see, is that paying proper attention to interaction also has a rich payoff for mono-agent, deductive scenarios. The basic idea is this; instead of examining the information flow between different agents in a communicative setting, we examine the information flow between different states of a single agent as the agent reasons deductively. We will see also how this approach has its roots in the the pure function application of the Categorical Grammar literature, due to Ajdukiewicz (1935), Lambek (1958), and Lambek (1961).

The first step, undertaken in subsection 1.1, is to motivate the move to an interactive epistemic logic by looking at *epistemic closure*. This will take us into issues involving universal and existential modalities (subsection 1.2), implicit and explicit knowledge (subsection 1.3), and finally to the epistemic relevance of the distinction between external conjunction and internal conjunction (subsection 1.4). With the motivation in place, we will be in a position to put the relevant model together. This is the task undertaken in section 2.

1.1 Motivation: Epistemic Closure

As motivation, consider the K-axiom:

$$\Box(\phi \Rightarrow \psi) \Rightarrow (\Box\phi \Rightarrow \Box\psi) \tag{1}$$

interpreted epistemically, so as to get *epistemic closure*:

$$K_\alpha(\phi \Rightarrow \psi) \Rightarrow (K_\alpha\phi \Rightarrow K_\alpha\psi) \tag{2}$$

By “interpreted epistemically” we mean that $K_\alpha X$ is read as *agent α knows that X* . This is importantly distinct from faux-epistemic interpretations such as *agent α is entitled to believe that X* , in which case K would be a deontic *permissibility* operator, not an epistemic one (or more realistically, an iteration of a deontic permissibility operator and a doxastic belief operator, see (3) in subsection 1.2 below). The conceptual difference is stark enough, but the formal difference is starker still. This formal difference turns on the distinction between universal and existential modalities.

1.2 Universal and Existential Modalities

The permissibility operator (as well as the belief operator) is an existential modality whilst the knowledge operator is a universal one: Taking necessity ‘ \Box ’ and possibility ‘ \Diamond ’ (either metaphysical or logical will do) as the canonical examples, then as per the textbook exposition, a formula is necessarily true *iff* it holds at *all points* in the relevant model, whilst by contrast a formula is possibly true *iff* it holds at *some points* in the relevant model. We have similarly contrasting modality pairs with *knowledge* and *belief*, as well as *obligation* and *entitlement/permission*: the canonical semantics takes the relevant atomic formulas constructed out of the first modality in each pair to hold at all points

in the model, and takes those constructed out of the second to hold at some points in the model.

Keeping track of the universal and existential status of modalities in various multi-modal logics is complex enough so that it is common practice to ‘box’ or ‘diamond’ the various modal operator letters in order to keep track of their universal/existential status. That is, a universal modality ‘ m ’ will be written as ‘ $[m]$ ’, and an existential modality ‘ m ’ will be written as ‘ $\langle m \rangle$ ’. To reiterate, examples of universal modalities are those such as *necessity*: \Box , *knowledge*: $[k]$ and *obligation*: $[o]$, with their corresponding existential modalities being *possibility*: \Diamond , *belief*: $\langle b \rangle$ and *entitlement*: $\langle e \rangle$ respectively.¹ In this case, closure under belief entitlement comes out as follows:

$$\langle e \rangle \langle b \rangle_{\alpha} (A \Rightarrow B) \Rightarrow (\langle e \rangle \langle b \rangle_{\alpha} A \Rightarrow \langle e \rangle \langle b \rangle_{\alpha} B) \quad (3)$$

As rich a philosophical potential as (3) might have, we note that (3) is not an instance of the K-axiom. Having fixed on the K-axiom, we need to fix on the type of knowledge involved.

1.3 Implicit Knowledge and Explicit Knowledge

We take K to stand for *explicit* knowledge. If we were to take K to stand for implicit knowledge, then closure would hold trivially, and tell us nothing epistemically interesting at all. The reason for this is straightforward enough: By taking K to stand for *implicit knowledge*, we force a collapse between what follows deductively from the agent’s knowledge-base on the one hand, and what the agent knows on the basis of this knowledge-base on the other. They simply become the very same thing. Implicit knowledge does away with any requirement of awareness or realisation. For knowledge to be of any use to an agent with respect to guiding the agent’s actions, be the actions cognitive or behavioural, the agent must be aware of the information inside the scope of the knowledge operator. Simply put, knowledge is not of any actual use until it becomes explicit.

By taking $K_{\alpha}X$ to encode *agent α knows that X* under explicit knowledge, closure becomes an interesting epistemic principle. It is also false. It is not merely false in the sense that it fails to hold for suitably complex instances of deductive inference. It is false in the sense that it fails to hold for *any* instance whatsoever. Closure fails to hold for any instance of deductive inference from an agent’s knowledge-base because closure merely expresses a static consequence relation, and deductive reasoning is an essentially dynamic, non-static process.

This is *precisely* where interaction comes in: Even if α explicitly knows that $A \Rightarrow B$, and also explicitly knows that A , it never follows simply on the basis of this (where this conjunction is Boolean), that α then explicitly knows that B . At least not when this conjunction is Boolean, or *external*.

¹By “corresponding existential modalities”, we do not mean to suggest that all such modality pairs are duals.

1.4 External Conjunction and Internal Conjunction

The distinction between external, or Boolean conjunction and internal conjunction is brought out nicely via precisely these epistemic considerations. Suppose that $K_\alpha X$ and that $K_\alpha Y$, then *we* may truthfully state that $K_\alpha(X \wedge Y)$. However, it may well be the case the α has never been aware of this conjunctive proposition, despite being aware of the conjuncts independently. This situation will occur, for instance, when the knowledge-states *that X* and *that Y* have occurred to α at suitably distinct times. Boolean conjunction is known as external conjunction precisely because of this external perspective taken on a system's states (and an agent is just one type of system).

For α to come to know that B on the basis of knowing the premises, α must *merge* or *combine* the information that the premises encode. This is how the gap between what follows deductively from an agent's knowledge-base on the one hand, and what the agent comes to know explicitly on the basis of that same knowledge-base on the other, is bridged. It is here that we must acknowledge the role of *interaction* for mono-agent deductive-reasoning scenarios. To model this adequately, we need an internal conjunction that brings the interactive/combinatorial information-merging reasoning actions of the agent to the foreground. We also need a few other things (but not many).

1.5 Onwards

In section 2 we will see how a nonassociative and structure-preserving commutative logic of mobiles (half way between sequences and multisets, see Moortgat (1995)) with an operational semantics will deliver a useful model. In section 3, we will see how some residuation conditions allow us to translate between the *instructions* for a successful reasoning procedure, and the *executions* of these instructions that underpin the reasoning procedure itself.

2 Weakly Commuting Interaction Models

A semantics is *operational* if it allows an explicit representation of semantic operations on individual points in the model (Buszkowski (1986)). In the model below, the points are understood as information states. This though, is still fairly general. Since we are concerned with the information flow in deductive inference, the points are to be understood as information-bearing states of an agent as the agent reasons deductively. This information-bearing is taken to be explicit knowledge. The interplay between the semantics so understood and the resulting syntax is philosophically interesting: The information in a database will be structured in different ways, depending on the use to which the information is to be put. We can think of this structure as a type of database *grammar*. The grammar will set the constraints as to structure of the database. In short, we will specify the grammar of a fragment of the "cognitive language" of deductive reasoning.

This first thing to do is to get a weakly commuting frame up and running. This is the task of subsection 2.1. Weak commutation is simply what happens when commutation is present, but association is not. There is a small terminological point that needs to be made at this point. Commuting but non-associating logics are not all that common. A consequence of this is that terms such as “commutation”, “permutation”, “exchange” and so on, are variously used to denote both pure pair-wise one-place swaps, and also a stronger n -place commutation that follows from the one-place variety plus association. A neat heuristic is this – in commutative nonassociative logics, we are restricted to pairs, although the left and right-hand members of these pairs may trade places harmlessly. With logics that both commute and associate, we are not restricted to pairs, and members may switch as many places as they like. This is a little abstract for now, but is laid out in detail below.

With this done, in subsection 2.2 we look at the well-known case of fully non-commuting examples from natural language semantics via categorical grammar. This is not merely an aside, but will give us the conceptual framework of databases and information processing that is central to the perspective worked out and adopted for mono-agent deductive reasoning. The next step, undertaking in subsection 2.3 is to build the resulting model on the back of the frame from 2.1 via an operational semantics. With this in place, we will be in a position to make the structured information processing properties relevant to interactive epistemic logic explicit (the business of subsection 2.4). In subsection 2.5 we will see how to recover association and strong commutation. In the final subsection, 2.6, we will bring all of this together in order to specify a fragment of the grammar of deductive reasoning. With this done, we will be in a position to apply the framework explicitly to the case of mono-agent dynamic reasoning. This the business of section 3.

2.1 A Commutating, Non-Associating Information Frame

Where $A, B \dots$ are types of uninterpreted propositional formula ϕ, ψ, \dots such that $\phi : A$ is read as *formula ϕ is of type A* , the language of our logic, the *non-associative Lambek Calculus with permutation, bottom, and identity* (\mathbf{NLP}_{01}) is given as follows:

$$A ::= \phi \mid A \mid \mathbf{0} \mid \mathbf{1} \mid A \otimes B \mid A \multimap B \mid A^\perp \quad (4)$$

Take an information frame $\mathbf{F} \langle S, \sqsubseteq, \bullet \rangle$ with weak (one place) commutation and Cut, along with the two binary connectives \otimes , and \multimap , the unary connective $^\perp$, and the constants $\mathbf{0}$ and $\mathbf{1}$. S is a set of incomplete, or partial information states x, y, \dots ² The binary relation \sqsubseteq is a partial order on S of informational development/inclusion. \bullet is the (commutative and non-associative) binary composition operator on information states. \otimes is (commutative and non-associative) internal

²In doxastic cases, we would allow for inconsistency also. But since knowledge is factive if anything is, we disallow this property here.

conjunction (merge/fusion). \multimap is interactive implication, and \perp is interactive negation on account of its being defined in terms of \multimap and $\mathbf{0}$:

$$A^\perp := A \multimap \mathbf{0} \tag{5}$$

$\mathbf{0}$ is bottom, and $\mathbf{1}$ is unit/identity such that:

$$\mathbf{1} \otimes A = A = A \otimes \mathbf{1} \tag{6}$$

Hence we have weak commutation around state-combination, and (correspondingly) merge, and weak-weak commutation around identity. Weak-weak commutation around identity is standard enough. Weak commutation (around combination and fusion) allows pair-wise one-place swaps: $x \bullet y = y \bullet x$ at least insofar as informational progress is concerned, and $A \otimes B = B \otimes A$ at least insofar as derivability is concerned. That is, for our interactions, commutation is permissible *within* the parameters of the given bracketing. Hence the following are of the same type:

$$A \otimes (B \otimes C) \mid A \otimes (C \otimes B) \mid (C \otimes B) \otimes A \mid (B \otimes C) \otimes A \tag{7}$$

Whereas:

$$B \otimes (A \otimes C) \tag{8}$$

differs. Exactly why it is that this should be the case for a base-calculus in the context of a significant fragment of mono-agent deductive reasoning is made clear in section 2.4 below. There *is* a difference in other contexts, natural language grammatical ones say, where strict preservation of left-right attachment-detachment is crucial. The natural language semantics cases from categorical grammar are not merely formal relatives. They in fact give us a the starting point for the very conceptual framework that we need.

2.2 Categorical Grammars: Pure Sequences, and Pairs

This starting point is to understand a natural language lexicon as a database of sorts, and the grammar (in this case the strict ordering preservation) as a collection of constraints on the processing of that database.

This ordering preservation is due to the fact that natural languages tend to have fragments with exact rules for the ordering of words. For example, ‘Frederike peddles’ is grammatically well-formed, whereas ‘Peddles Frederike’ is not. In this manner, the intransitive verb ‘peddles’ is of type $n \rightarrow s$, since when it is applied to the right of a noun (type n), the result is a sentence (type s). By contrast, the adjective ‘happy’ is of type $n \leftarrow n$. This is due to the fact that when ‘happy’ is applied to the left of another noun, the result is another (complex) noun (phrase): ‘Happy Friederike’. In logics with no commutation (other than possible weak-weak commutation around $\mathbf{1}$ if $\mathbf{1}$ is present), the structures are at least pure sequences, and if associativity is rejected also, then they are pairs.

This much is familiar from the categorical grammar literature. What is less familiar is the following useful way of describing the relevant semantic constraints. We can think of natural language (in particular a natural language lexicon) as a *database*. In this case, a grammar may be thought of as a collection of *processing constraints* on the database. These processing constraints are operational in the strictest sense of the term. They tell us which operations are permissible insofar as generating well-formed, or meaningful, strings in the natural language is concerned. In informational terms, they tell us which operations are permissible insofar as generating information is concerned. A detailed look at grammars as processing constraints on a database may be found in Sequoia-Grayson (2009b).

For our purposes here however, all we note is that a natural language lexicon is only one type of database. A consequence of this is that we may think of the processing constraints on *any* database (and not simply natural language lexicons) as *grammars*. This is precisely what is done below with the database relevant to mono-agent deductive reasoning. The rest of this section is dedicated to developing this idea in more detail. The first step is to specify the relevant model.

2.3 The Model

A model $\mathbf{M} := \langle \mathbf{F}, \Vdash \rangle$ is an ordered pair $\mathbf{F} \langle S, \sqsubseteq, \bullet \rangle$ and \Vdash such that \Vdash is an evaluation relation that holds between members of S and formulas constructed out of our binary connectives \otimes , and \multimap , and constants $\mathbf{0}$ and $\mathbf{1}$. In what follows, we will often write $x, y, \dots \in \mathbf{F}$ as shorthand for $x, y, \dots \in S$ where $S \in \mathbf{F}$. Where A is a propositional formula, and $x, y, z \in \mathbf{F}$, \Vdash obeys the heredity or monotonicity condition:³

$$\text{For all } A, \text{ if } x \Vdash A \text{ and } x \sqsubseteq y, \text{ then } y \Vdash A. \quad (9)$$

And also obeys the following conditions for each of our connectives and constants:

$$x \Vdash A \otimes B \text{ iff for some } y, z, \in \mathbf{F} \text{ s.t. } y \bullet z \sqsubseteq x, y \Vdash A \text{ and } z \Vdash B. \quad (10)$$

$$x \Vdash A \multimap B \text{ iff for all } y, z \in \mathbf{F} \text{ s.t. } x \bullet y \sqsubseteq z, \text{ if } y \Vdash A \text{ then } z \Vdash B. \quad (11)$$

$$x \Vdash \mathbf{0} \text{ for no } x \in \mathbf{F}. \quad (12)$$

$$x \Vdash A^\perp[A \multimap \mathbf{0}] \text{ iff for all } y, z \in \mathbf{F} \text{ s.t. } x \bullet y \sqsubseteq z, \text{ if } y \Vdash A \text{ then } z \Vdash \mathbf{0}. \quad (13)$$

$$x \Vdash \mathbf{1} \text{ iff } x \in T, \text{ for all } x \in \mathbf{F}. \quad (14)$$

(14) is less straightforward than are (10)–(13). We firstly define the set of *propositions* $\text{Prop}(\mathbf{F})$ on our frame \mathbf{F} as the set of all subsets X of $S \in \mathbf{F}$ such that they are *upwardly closed*: if $x \in X$ and $x \sqsubseteq x'$, then $x' \in X$.

³We would drop this condition for certain *doxastic* scenarios where non-monotonicity is a distinctive property.

We can now see how it is that T in (14) is a *truth set*. Truth sets come in left and right versions. For any subset T of $\text{Prop}(\mathbf{F})$:

- T is a *left truth set* iff for all $y, z \in \mathbf{F}$, $y \sqsubseteq z$ iff for some $x \in T$, $x \bullet y \sqsubseteq z$.
- T is a *right truth set* iff for all $y, z \in \mathbf{F}$, $y \sqsubseteq z$ iff for some $x \in T$, $y \bullet x \sqsubseteq z$.

Given that we have commutation on our frame, our truth set T is non-directional. In other words, since $x \bullet y = y \bullet x$, the left and right truth sets collapse into a single, non-directional truth set. The converse does not hold. The “for some $x \in T$ ” constraint in the right hand clause of both truth set conditions allows us to restrict commutation to just these x . This is what allows $\mathbf{1}$ to commute around propositions in logics that are otherwise non-commutative. Intuitively, the state x carrying the information $\mathbf{1}$ behaves around other states in the same manner as does $\mathbf{1}$ around propositions (see (6) in subsection 2.1 above).

With the model conditions laid bare via the operational semantics, we can look at how the various connectives deliver with respect to information processing environments.

2.4 Databases and Information Processing

Our merge and implication connectives interrelate in the following manner:

$$A \otimes B \vdash C \text{ iff } B \vdash A \multimap C \quad (15)$$

(15) is the standard residuation condition, the algebraic counterpart to implication intro/elim. Residuation conditions encode various substructural logics, with the interesting properties of residuated structures (i.e. their conditions) being captured via the presence or absence of various structural rules. In our case we are understanding the residuation condition in (15) to specify the processing constraints on a database.

We take \vdash to be a processing-gate in the sense that $X \vdash A$ is read as *processing the information in X generates the information that A* . Exactly what it is that processing amounts to depends on the structure of the database in question, which is in turn fixed by the structural rules at work. Since the merge operation is simply *combination*, and not directional application, we get $A \vdash B \multimap C$ from $A \otimes B \vdash C$ by commuting on $A \otimes B$ so as to get $B \otimes A$. This is one sense in which we depart from Ajdukiewicz’s original contribution to categorical grammar, since we do not need to keep track of left-right attachment-detachment. The only structural rule we will admit except for Weak Commutation is Cut. Cut is important for making use of the relevant information interactions at work in basic inference. Association however, is not. To see this, set the following (labels occur inside brackets):

$$\phi \Rightarrow \psi(A), \sigma \Rightarrow \phi(B), \sigma(C), \phi(D), \psi(E) \quad (16)$$

In such a case, we have it that:

$$A \otimes (C \otimes B) \vdash E \quad (17)$$

(17) is a result of Cutting on D, since:

$$C \otimes B \vdash D, \text{ and } A \otimes D \vdash E. \quad (18)$$

It is in this sense that cut underpins the most fundamental notion of information interaction, or *processing*. However, suppose that we were to *associate* on (17). In this case, we would have it that:

$$(A \otimes C) \otimes B \vdash E \quad (19)$$

This is not good. The result of combining the information in A with the information in C is nothing such that were it to be applied to the information in B we would get the information in E . In fact, no information results from the combination of information encoded by A with information encoded by C . Such an attempt is a “dead process”, that cannot be carried out. To see this, we introduce types; given that $\phi \Rightarrow \psi(A)$ and $\sigma(C)$, it is the case that $\phi \Rightarrow \psi : C^\perp$. Hence $\phi \Rightarrow \psi : C \multimap \mathbf{0}$ via (5). For any state $x \Vdash C \multimap \mathbf{0}$, we know that it is that case that if we combine this information with any other state y s.t. $y \Vdash A$, then the result will be a state z s.t. $z \Vdash \mathbf{0}$ via (13). However, we know that $\mathbf{0}$ is not supported via any state via (12).

In information processing terms, some information is of type C^\perp iff its combination with information of type C can never generate any information. This is a conceptually parsimonious way of reading the frame condition in (13). Taking interactive implication to be *functional*, along the lines of the Lambek Calculi, then C^\perp is the type of function that can never take information of type C as an input, on account of it never outputting any information on the basis of inputs of this type. This makes perfect sense in our interactive/dynamic setting. In a static setting, negation is ruling out truth. In an interactive/dynamic setting however, negation will rule out particular interactions, or processes.⁴

Now we can see why it is that a strong, unrestricted (two-place) commutation is just as “de-railing” as associativity. This would allow us to get from $(A \otimes B) \otimes D$ to $(A \otimes D) \otimes B$ – but now just set $\phi(A), \phi \Rightarrow \psi(B), \psi(C), \psi \Rightarrow \sigma(D), \sigma(E)$. Similar results can be gotten for the other structural rules by adjusting the setup in (16) appropriately, see Sequoia-Grayson (2009c). Importantly, strong-commutation and association are not independent.

⁴Negation in a dynamic setting as *process exclusion* is a topic unto itself. We can extend the operation into a fully directional process exclusion system by dropping even weak commutation. This will allow us to split our interactive implication \multimap into a double implication pair $\langle \multimap, \multimap \rangle$, that will in turn allow us to define a split negation pair $\langle \sim, \neg \rangle$ which we may define as $\sim A := A \multimap \mathbf{0}$ and $\neg A := \mathbf{0} \multimap A$ respectively. For an examination of logic-invariant split-negation properties in terms of process exclusion, as well as an examination of its philosophical status, see Sequoia-Grayson (2009a). For a working through of a series examples of process exclusion, both directional and non-directional, as well as an examination of the connections with the related notion of negation as test-failure in *dynamic predicate logic*, see Sequoia-Grayson (2009b).

2.5 Strong-Commutation Recovery and Association Recovery

Weak-commutation with associativity recovers strong-commutation, and weak-commutation with strong-commutation recovers associativity:

- For the first recovery, start with $(A \otimes B) \otimes D$, then associate to get $A \otimes (B \otimes D)$, then weakly-commute to get $A \otimes (D \otimes B)$, then associate again to get $(A \otimes D) \otimes B$. \square
- For the second recovery, start with $A \otimes (B \otimes D)$, and strongly commute to get $B \otimes (A \otimes D)$, then weakly-commute to get $(A \otimes D) \otimes B$, then strongly-commute to get $(A \otimes B) \otimes D$. \square

In otherwords, we have generative as well as independent reasons to reject associativity and strong-commutation.

With this much done, we are in a position to specify the processing-constraints, or grammar, of a fragment of mono-agent deductive reasoning.

2.6 The Grammar of Deductive Reasoning

The *grammar* of deductive reasoning, a cognitive-language if you will, has obvious fragments with useful properties captured by a commuting, non-associating (i.e. weakly-commuting) logic. The logic corresponds to **NLP**: the *nonassociative Lambek Calculus with permutation*, where permutation is understood in the pair-wise sense such that is amounts to:

$$x \bullet y \sqsubseteq z = y \bullet x \sqsubseteq z \quad (20)$$

Specifying explicitly the pairwise nature of the permuting operation is not redundant. This is because it is commonplace in the literature to use ‘permutation’ to denote the strong-commutation that follows from commutation, or pairwise permutation, and association. This is a simple function of the fact that commuting, non-associating logics have are rare, so the resulting stronger permuting operation, allowing permutations through bracketing, or *structure*, has been the default.

The structure of the data-base on the left-hand-side of the processing gate \vdash , that is X in $X \vdash A$ that is specified by the grammar is that of *mobiles*. Mobiles are simply non-associating but bracket-sensitive-commutating structures. Since we can get strong-commutation from weak-commutation if we also have association, the addition of association would collapse the structure of our data-base into multisets (since we do not have contraction). But multisets have no structure at all, they have merely a taxonomy. Mobiles have some structure, but less than do pure sequences (where even pair-wise commutation is prohibited). If we had neither commutation nor association, we would have *static pairs*, with a fixed left component and a fixed right component. Mobile may be thought of as mobile pairs, which is what they really are after all; pairs whose left and right components may switch places.

It is important to appreciate that what is happening here is *not* simply syntactic (although it can be read off the syntax, which is part of the appeal). Merge (along with our implication) is being interpreted as a relation between *information states*. It is in this sense that *interaction structures* (so–christened due to their being constituted by interactive conjunctions) such as that on the left hand side of (17): $A \otimes (C \otimes B)$, are robustly semantic. Two interaction structures with the same form will not necessarily be equivalent, since they may be underpinned by different information states. Extracting the step–wise information state combinations across S *corresponding* to the relevant interaction structures is a straightforward mechanical task involving nothing more than successive applications of the conditions outlined by (10) above. This is most easily seen via some examples.

3 Mono–Agent Dynamic Reasoning

In subsection 3.1 we will look at information processing operations underpinning interaction structures, as well as those processing operations underpinning the corresponding *processing structures*; those conditional information processing structures got from interaction structures via the residuation conditions in (15). The task undertaken in subsection 3.2 is to *interpret* interaction structures and processing structures in terms of mono–agent deductive reasoning actions. Processing structures will be interpreted as *instructions*, and interaction structures will be interpreted as *executions* of these instructions.

3.1 Interaction Structures and Processing Structures

With respect to (17): $A \otimes (C \otimes B) \vdash E$, we have the following corresponding step–wise information state combination:

$$\begin{aligned} x \Vdash A \otimes (C \otimes B) \text{ iff for some } w, y, z \in \mathbf{F} \text{ s.t. } w \bullet (y \bullet z) \sqsubseteq x, \\ w \Vdash A, y \Vdash C, \text{ and } z \Vdash B. \end{aligned} \tag{21}$$

The information states $x, y, \dots \in S$ may be naturally interpreted as states of α as α reasons deductively. In this case, the information state combination $w \bullet (y \bullet z) \sqsubseteq x$ specifies the step–wise reasoning procedure that α must engage in in order to be truthfully said to know (on the basis of the premises at least) the result of the merged propositions, namely ψ . Since, $y \Vdash C$, and $z \Vdash B$ via (21), and $\sigma(C)$ and $\sigma \Rightarrow \phi(B)$ via (16), $y \bullet z \sqsubseteq v$, where $v \Vdash D$, and $\phi(D)$ via (16). Since $w \Vdash A$ via (21) and $\phi \Rightarrow \psi(A)$ via (16), $w \bullet v \sqsubseteq x$, where $x \Vdash E$ and $\psi(E)$ via (16). \square

Via (15), we can transform interaction structures into iterated conditional information *processing structures*. Still taking (17) as our case, via three applications of (15) and one application of (6), we generate:

$$\mathbf{1} \vdash B \multimap (C \multimap (A \multimap E)) \tag{22}$$

From $A \otimes (C \otimes B) \vdash E$ we get $C \otimes B \vdash A \multimap E$ via the first application of (15). From $C \otimes B \vdash A \multimap E$ we get $B \vdash C \multimap (A \multimap E)$ via the second application of (15). From $B \vdash C \multimap (A \multimap E)$ we get $B \otimes \mathbf{1} \vdash C \multimap (A \multimap E)$ via (6). From $B \otimes \mathbf{1} \vdash C \multimap (A \multimap E)$ we apply our third and final instance of (15) in order to get $\mathbf{1} \vdash B \multimap (C \multimap (A \multimap E))$. \square

We can understand the processing structure on the right hand side of (22) as a *typed function*. It is the type of function that takes inputs of type B , and returns another function as the output. The function that it returns as an output is the type of function that takes inputs of type C , and returns yet another function as an output. *This* function is the type of function that takes inputs of type A and returns an output of type E (which in our case is ψ , since $\psi(E)$ via (16)).

Similarly to interaction structures, the processing structures/function types are individuated by information states. With respect to (22), and via (11), we have the following:

$$\begin{aligned} x \Vdash B \multimap (C \multimap (A \multimap E)) \text{ iff for all } s, t, v, w, y, z \in \mathbf{F} \text{ s.t. } ((x \bullet y) \bullet v) \bullet t \sqsubseteq s, \\ \text{if } z \Vdash C \multimap (A \multimap E), \text{ and } y \Vdash B, \text{ and } w \Vdash A \multimap E, \text{ and } v \Vdash C, \text{ and } t \Vdash A, \\ \text{then } s \Vdash E. \end{aligned} \quad (23)$$

Since $x \bullet y \sqsubseteq z, z \bullet v \sqsubseteq w$, and $w \bullet t \sqsubseteq s$. \square

We can now turn our attention to examining the conceptual relationship between interaction structures and processing structures on the one hand, and mono-agent deductive reasoning on the other. The following section explains how it is that we may sensibly interpret processing structures and interaction structures as instructions and the executions of those instructions respectively.

3.2 Instructions and Executions

How might we think of the interaction structures such as that in (21) and their corresponding processing structures such as that in (23) with respect to our wider concern with interactive mono-agent reasoning? We can think of processing structures as *instructions*, and of their corresponding interaction structures as the result of carrying out or executing the corresponding instruction, i.e., as *executions*.

Take the instruction in (23). If α is in state x , then α is on her way to knowing explicitly *that* ψ , but she is not there yet. α 's being in state x means that α knows explicitly what is required in order that she come to know explicitly *that* ψ . Her first step is to establish $B(\sigma \Rightarrow \phi)$, which involves her being in state y . Her second step is to combine this state y with her previous state x . In other words, we read the iterated state-combination sequence corresponding to process structures such as $((x \bullet y) \bullet v) \bullet t \sqsubseteq s$ in (23) from the inside-out.

This first interaction or merge will result in two things. α will know explicitly *that* B , and also be in a new state $z \Vdash C \multimap (A \multimap E)$ which follows from this first interaction. This corresponds to the first leftwards-transfer across the processing-gate. That is, α has moved from $\mathbf{1} \vdash B \multimap (C \multimap (A \multimap E))$ to

$B \otimes \mathbf{1} \vdash (C \multimap (A \multimap E))$ with the establishing of B allowing α to get rid of $\mathbf{1}$ and arrive at $B \vdash (C \multimap (A \multimap E))$.

How might we interpret $\mathbf{1}$? That is, how might we make sense of the initial state x of α s.t. $x \Vdash \mathbf{1}$? We do so in exactly the sense stipulated above; α being in state x which carries the information that $\mathbf{1}$, is simply that state in which α knows what steps must be taken in order to establish E , and hence be in state s . But knowing the steps to take is not the same thing as having actually taken them.

The next step for α is for her to establish $\sigma(C)$, which entails that she be in state $v \Vdash C$. By combining v with z , α will then be in state $w \Vdash A \multimap E$. This corresponds to the second leftwards-transfer across the processing-gate, such that α has moved from $B \vdash (C \multimap (A \multimap E))$ to $C \otimes B \vdash A \multimap E$. The final steps for α are that she establish $\phi \Rightarrow \psi(A)$, which entails her being in state $t \Vdash A$. Then α must combine t with her previous state w . This will entail α being in state $s \Vdash E$, where $\psi(E)$. This corresponds to the third leftwards move across the processing-gate, such that α has moved from $C \otimes B \vdash A \multimap E$, to $A \otimes (C \otimes B) \vdash E$.

What has occurred is this: By following the instructions laid out in the processing structure, α has extracted the very interaction structure who's "activation" will cause her to know explicitly *that* ψ . This fact has a straightforward interpretation in terms of the data-base structure, or grammar, of the "cognitive language" of deductive reasoning. The interpretation of the end-results of applications of the residuation conditions in (15) as instructions and executions is a powerful one. The instructions tell us how to construct a well-formed sentence in the cognitive language of deductive reasoning in precisely the same manner as the types in categorical grammar instruct us on constructing well-formed terms or sentences in natural language. In natural language, if the result of one of these constructions is uttered (or conceived of) by an agent, then information is transmitted to other agents. Something meaningful, or informationally well-behaved, will have occurred.

The deductive reasoning case is the same. When the results of carrying out the instructions are executed, something informationally well-behaved has occurred. In this case, the occurrence is that the agent has accessed the information in the conclusion, or simply put; the agent is in a state such that the agent knows explicitly the information that the conclusion encodes. That deductive reasoning procedures should have their behaviour accurately described by formal tools originally developed to specify that mathematical behaviour of natural language semantics is, ultimately, not as surprising as it might first appear. After all, when we are engaged in explicit acts of deductive reasoning, it is rather like we are talking to ourselves.

Mobiles impose the relevant constraints on the grammar for the simple reason that left-right exchange *within* pairs is harmless for a significant fragment of deductive reasoning. It is by no means the case that such constraints may not be either relaxed or strengthened (more on this in the following section), but they are a useful place to start.

4 Conclusion

By paying proper attention to consequence *and* interaction in logic, we have a system of resource-management and processing on the database of mono-agent reasoning dynamics. Although what we have is far from a complete set of constraints, or a full grammar (more on this below), what do have, hopefully, is a *proof of concept*. In subsection 4.2 we will revisit the notion of epistemic closure in light of the addition of dynamic information-merging operations. The final task, undertaken in 4.3 is to sketch the direction for future research. The first step however, is a brief review.

4.1 Summary

In terms of grammatically obedient databases or “sentences” in the language of deductive reasoning, the processing structures are instructions for constructing a grammatically well-formed sentence in the language. The resulting interaction structures are the very grammatically well-formed sentences in the language of deductive reasoning that the processing structures are intended to bring about. That is, in exactly the same way that types in linguistic applications of Categorical Grammars give you the instructions for well-formed sentences in the grammar of natural language, the processing structures/types here give you the instructions for grammatically well-formed “sentences” in the cognitive-grammar of deductive reasoning.

Whether or not such sentences *are* well-formed is a function of their input/output relations. To make the point again, if $\sigma(A)$ and $\phi \Rightarrow \psi(B)$, then $A \otimes B$ is not well-formed, since it is the case that $\phi \Rightarrow \psi : A^\perp$, or $\phi \Rightarrow \psi : A \rightarrow \mathbf{0}$. In the fragment of deductive reasoning that we are considering, if we were to allow associativity or strong commutation (or any of the other well-known structural rules), then we would allow grammatical transformations, or transformations on the structure of the database, that would ruin the ability of information to flow.

The information in a database will be structured in various ways depending on the data-types it contains. The constraints turn on the use to which the information is to be put. The database structure, or grammar, of the “cognitive language” of mono-agent deductive reasoning, may be sensibly said to have many of its interesting base-properties captured by the residuated structure encoding mobiles with bottom and identity, namely $\mathbf{NLP}_{\mathbf{01}}$.

4.2 Epistemic Closure Revisited

With this much in place, where do we stand on our motivating principle from subsection 1.1? Taking $(X \otimes Y)_\alpha$ to stand for α 's merging of X and Y , we might lay out a working principle for epistemic closure:

$$K_\alpha((\phi \Rightarrow \psi) \wedge K_\alpha \phi) \Rightarrow (((\phi \Rightarrow \psi) \otimes \phi)_\alpha \Rightarrow K_\alpha \psi) \quad (24)$$

However, this would be a mistake. Until the structure of the data-base is specified (equivalently, until the grammar of the relevant fragment of the cognitive language of deductive reasoning is specified), such a dynamic information-merging principle is not particularly interesting *in itself*. In fact, one of the goals of this article has been to demonstrate that such an approach is wrong-headed from the start.

The problem with (2) is that it is a static consequence principle, and knowledge is the result of dynamic reasoning (or communicative) scenarios. Not only are counterexamples too easy to come by, but in the case of static axioms like (2), there are only counterexamples. Dynamic, or interactive axioms such as (24) do not deliver much more.

The problem is a methodological one. The interesting epistemic principles are those concerning the structural and dynamic properties of the data-bases relevant to particular epistemic contexts: mono-agent, multi-agent, empirical-information, deductive/inductive/abductive-reasoning, recall-constraints, communication-variance, and combinations of these and more. If such epistemic contexts are not specified, then for particular epistemic axioms to be true, they will need to be at such a generalised level of abstraction that it is unlikely that they will tell us anything interesting at all. Consider the T-Axiom:

$$\Box\phi \Rightarrow \phi \tag{25}$$

under an epistemic interpretation so as to get:

$$K_\alpha\phi \Rightarrow \phi \tag{26}$$

(26) is a perfect example of just this. (26) is true in absolutely any epistemic context whatsoever. However it is not telling us anything interesting about any of these contexts, nor for that matter is it telling us anything interesting about the knowledge-states of agents, apart from the definitional one – if it is true that an agent knows that A , then it is true that A . We need more than this. We also need a methodological basis that can reliably deliver more than this for an analysis of the full range of dynamic properties at work in deductive reasoning actions.

If there is one thing that dynamic/interactive epistemic logic should take from static epistemic logic, it is the knowledge that there is little to gain from top-down syntactic-axiom-driven approaches. Modern research in modal logic has long abandoned such an approach, witness Blackburn, de Rijke, and Venema (2002). For epistemic logic (that is based on modal logic after all) to contribute to our understanding of the rich tapestry of epistemic contexts that underpin our success as rational agents in the natural world, it should follow suit. Where to next?

4.3 The Future

We might think of the fragment of deductive reasoning that we have examined here as the *commutation invariant fragment*. It is a natural place to start with

an analysis, as commutative-invariant residuated structures underpin the base-calculus of mono-agent deductive reasoning at the level of abstraction of uninterpreted propositional types. Here, every process is commutation invariant.⁵ Commutation *variant* types will only arise in the presence of other structural rules, in which case they will not be pure commutation variant types, but rather commutation + rule- x variant types.

There will also be fragments of deductive reasoning that are rule- x invariant where commutation is disallowed, as well as fragments that are commutation + rule- x invariant, and so on. We should think of these invariant fragments as grammatical categories in the cognitive language of deductive reasoning, in just the same way as we do with structural-rule-invariant fragments of natural language. Specifying the full grammar of the cognitive language of mono-agent deductive reasoning will involve mapping out just these relations between various invariant and variant types. The present article has been an attempt to contribute to a fragment of such an investigation.⁶

References

- Ajdukiewicz, K. (1935): Die Syntaktische Konnexitat, *Studia Philosophica*, 1–27.
- Baltag, A., Moss, L., and Solecki, S. (1998): The logic of Public Announcements, Common Knowledge and Private Suspicions, *Proceedings of Tark'98*, 43–56, Morgan Kaufmann Publishers.
- Baltag, A, and Smetts, S. (2008): The Logic of Conditional Doxastic Actions, forthcoming in R. van Rooij and K. Apt (eds.): *Texts in Logic and Games*, Special issue on New Perspectives on Games and Interaction, Amsterdam University Press.
- van Benthem, J. (2009): Logical Dynamics of Information and Interaction, *manuscript*.
- Blackburn, P., de Rijke, M., and Venema, Y. (2002): *Modal Logic*, Cambridge University Press.

⁵In interpreted contexts, strict temporal preservation, corresponding to commutative *variance* will obviously have a roll to play.

⁶I am greatly indebted to Johan van Benthem for first suggesting to me that a deeper examination of the Categorical Grammar literature may make a positive contribution to the analysis of mono-agent dynamics. As always, his advice was good. I am also indebted to Greg Restall for answering endless questions about various commutation types. I would also like to thank Patrick Allo, Jake Chandler, Hans van Ditmarsh, Marie Duzi, Allen Mann, and Jerry Seligman for helpful comments and suggestions. Any remaining mistakes remain my own. Vincent Hendricks at *Synthese* has continually gone out of his way to make the logistics possible, a generosity for which I am grateful. Most of all, I would like to thank Igor Douven, who, in his capacity as the director of the *Formal Epistemology Project*, has created the most delightful research environment that anyone could possibly wish for.

- Busckowski, W. (1986): Completeness Results for Lambek Syntactic Calculus, *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 32, 13–28.
- van Ditmarsh, H., van der Hoek, W., and Kooi, B. (2008): *Dynamic Epistemic Logic*, Springer, The Netherlands.
- Lambek, J. (1958): The Mathematics of Sentence Structure, *American Mathematical Monthly*, 65, 154–170.
- Lambek, J. (1961): On the Calculus of Syntactic Types, in R. Jakobson (ed.): *Structure of Language and its Mathematical Aspects*, American Mathematical Society, Providence, 166–178.
- Moortgat, M. (1995): Residuation in Mixed Lambek Systems, Research Transcript no. 10, Research Institute for Language and Speech (OTS), Utrecht.
- Sequoiah-Grayson, S. (2009a): Dynamic Negation and Negative Information, *Review of Symbolic Logic*: 2.1, 233–248.
- Sequoiah-Grayson, S. (2009b): Lambek Calculi with $\mathbf{0}$ and Test–Failure in DPL, *forthcoming in Linguistic Analysis*.
- Sequoiah-Grayson, S. (2009c): A Positive Information Logic for Inferential Information, *Synthese*, 167: 2, pp. 409–431