Preface

Welcome to the proceedings of the 10th International Conference on Logical Aspects of Computational Linguistics, which was held December 14, in Montpeller France. The conference was held fully online, as part of the larger Mathematical Linguistics week, conjointly with Mathematics of Language (MOL, December 13) and Logic and Algorithms for Computational Linguistics (LACompLing, December 15-17).

Among the 15 papers submitted to LACL, the program committee selected the 9 high-quality contributions included in this volume.

We would like to thank all authors who submitted papers, and all conference participants. We are grateful for the members of the program committee and the additional reviewers for their thorough efforts reviewing all submissions. We also thank Larry Moss for his keynote talk on "Monotonicity in Natural Language Inference: Theory and Practice".

Finally, we would like to thank the members of the local organising committee who made the meeting possible.

November 2021

Annie Foret
Richard Moot

Organisation

Program Committee

Maxime Amblard	Université de Lorraine
Denis Béchet	LINA, Univesity of Nantes
Daisuke Bekki	Ochanomizu University
Heather Burnett	CNRS, Université Paris VII
Wojciech Buszkowski	Adam Mickiewicz University in Poznan
Stergios Chatzikyriakidis	CLASP, University of Gothenburg
Annie Foret	IRISA, Rennes University (Chair)
Nissim Francez	Technion - IIT
Philippe de Groote	INRIA Lorraine/LORIA
Gregory Kobele	Universität Leipzig
Yusuke Kubota	National Institute for Japanese Language and Linguistics
Hans Leiss	Universität München, CIS
Robert Levine	Ohio State University
Zhaohui Luo	Royal Holloway, University of London
Alda Mari	Institut Jean Nicod, CNRS/ENS/EHESS/PSL
Michael Moortgat	Utrecht University
Richard Moot	CNRS, LIRMM, Montpellier University (Chair)
Glyn Morrill	Universitat Politècnica de Catalunya
Larry Moss	Indiana University Bloomington
Valeria de Paiva	Samsung Research America and University of Birmingham
Sylvain Pogodalla	LORIA/INRIA Lorraine
Carl Pollard	Ohio State University
Jean-Philippe Prost	Université Publique de France
Mehrnoosh Sadrzadeh	University College London
Serguei Soloviev	IRIT, Université Toulouse 3
Mark Steedman	University of Edinburgh
Jakub Szymanik	University of Amsterdam
Oriol Valentín	Universitat Politécnica de Catalunya
Marek Zawadowski	University of Warsaw

Organising Committee

Virginie Feche	LIRMM, Montpellier University
Mégane Miquel	LIRMM, Montpellier University
Richard Moot	CNRS, LIRMM, Montpellier University
Christian Retoré	CNRS, LIRMM, Montpellier University

Additional Reviewers

Giuseppe Greco	Utrecht University
Gijs Wijnholds	Utrecht University

Table of Contents

An inquisitive account of *wh*-questions through event semantics

Maxime Amblard, Maria Boritchev, and Philippe de Groote

Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France
maxime.amblard@loria.fr, maria.boritchev@loria.fr,
philippe.degroote@loria.fr

Abstract. We give a compositional account of the semantics of *wh*-questions. This account is based on the alliance of two semantic theories: neo-Davidsonian event semantics [22], on the one hand, and inquisitive semantics [6], on the other hand. The resulting system is implemented in the framework of the abstract categorial grammars [13].

1 Introduction

Modeling the meaning of sentences in natural languages is a task that can be approached from different perspectives, ranging from distributional semantics to formal semantics. The study presented in this paper is conducted from the point of view of and in the scope of formal semantics.

There are multiple formal accounts of the semantics of declarative sentences, which mainly derive from Montague's seminal work [18, 19, 20]. Most of these accounts are truth-conditional: they characterize the semantics of a declarative sentence by specifying the conditions under which that declarative sentence can be considered as true. In the present work, our object of study is interrogative sentences, and for this type of sentence, truth-conditional approaches do not apply immediately. It is indeed not clear what assigning a truth value to a question would mean [12]. Is a yes/no-question true or false? What is a negative answer to a yes/no-question? What does it mean for a *wh*-question to be true or false?

Soon enough following Montague's work, Hamblin proposed a solution to this problem [16]. His proposal consists in characterizing the semantic content of a question by specifying its set of possible answers. This gives rise to a semantic framework known as *alternative semantics*. The more recent theory of Inquisitive semantics [6], which belongs to this tradition, makes several technical improvements over other Hamblin-like theories. It provides a logic that handles interrogative and declarative sentences without differentiation, and that is amenable to a compositional treatment, as shown in [7].

Following classic literature on formal semantics of interrogative sentences (see [11, 24], for a survey), we investigate questions through the types of the queries they raise. This leads us to envision an approach based on a semantic framework that makes extensive use of thematic roles: neo-Davidsonian event semantics [22]. Then, the main idea behind our proposal is to see, at the semantic level, a *wh*-extraction as an inquisitive existential quantification. This quantification binds

the variable that serves as an argument to the thematic role corresponding to the *wh*-word that triggers the *wh*-extraction.

The paper is organized as follows. The next section presents some useful linguistic and mathematical preliminaries. In particular, it encompasses a discussion about the notion of thematic role, and brief introductions to neo-Davidsonian semantics, inquisitive logic, and abstract categorial grammars. Section 3 is the core of the paper, in which we formally present our proposal. We parallel *wh*-extraction with quantifier raising, which brings us to consider *wh*-words as generalized quantifiers. The resulting system is then formalized in terms of abstract categorial grammars. These grammars are kept as simple as possible with the aim of capturing the gist of the semantic interpretation of *wh*-questions. This simplicity, however, is not without its drawbacks. It is indeed the case that the grammatical system defined in Section 3 overgenerates and introduces spurious ambiguities. In order to remedy this, we define in section 4 a device that allows us to control overgeneration. This device takes the form of an additional abstract categorial grammar that we add to our grammatical architecture. Finally, in Section 5, we conclude.

2 Linguistic and mathematical preliminaries

2.1 *Wh*-questions and thematic roles

As defined in [10], *wh*-questions, in English, are questions that give rise to answers whose semantic sorts match those of the *wh*-phrase contained in the interrogative. A *wh*-phrase is introduced by a *wh*-word: `what, when, where, who, whom, which, whose, why, how` [1].

For this definition to be operational, it is necessary to systematically define the semantic sorts. A way to do so is by using *thematic roles*, as inspired by [8, 17]. This raises many discussions related to the interpretation and definition of thematic roles. To tighten up, we use the following list of thematic roles, which is inspired by Fillmore's and Gruber's works [9, 15]: `participant`, `actor`, `cause`, `agent`, `undergoer`, `instrument`, `theme`, `pivot`, `patient`, `attribute`, `location` (see the definitions in Table 1). This list is in fact adapted from [3], without thematic roles specific to events with symmetrical participants, events of perception, or events of communication, and with the addition of the `Location` role from [4].

We note that a more detailed list of thematic roles is presented in the DIT++ schema [4], a semantically based framework for the analysis and annotation of dialogue. Following the statement in [2] that no fixed list of thematic roles can be established (nor crosslinguistic, nor for English only), we choose to showcase our method on a shorter list for the sake of readability. Our model can be easily tailored to a different list of thematic roles.

Once the thematic roles are set, asking a *wh*-question corresponds to interrogating the content of a thematic role. Therefore, in order to express the semantics of *wh*-questions, we need a formalism that gives an explicit access to terms or variables corresponding to the content of thematic roles. neo-Davidsonian event semantics [22] is such a formalism.

Table 1. List of thematic roles

Thematic role	**Definition**
Participant	Entity involved in a state or event
Actor	Participant that is the instigator of an event
Cause	Actor (animate or inanimate) in an event, that initiates the event, without intentionality or consciousness, existing independently of the event
Agent	Actor in an event who initiates and carries out the event intentionally or consciously, existing independently of the event
Undergoer	Participant in a state or event that is not an instigator of the event or state
Instrument	Undergoer that is central to an event or state that is not an instigator of the event or state
Theme	Undergoer that is central to an event or state that does not have control over the way the event occurs, is not structurally changed by the event, and/or is characterized as being in a certain position or condition throughout the state
Pivot	Theme that participates in an event with another theme unequally but is central to the event
Patient	Undergoer in an event that experiences a change of state, location, or condition, that is causally involved or directly affected by other participants, and exists independently of the event
Attribute	Undergoer that is a property of an entity or entities, as opposed to the entity itself
Location	Place where an event occurs or a state is true

2.2 Neo-Davidsonian event semantics

Neo-Davidsonian event semantics (NDES) is a formalism in which every sentence is considered in terms of occurring events and ways the sentence semantic constituents relate to this event. Recent updates such as [5, 23] present compositional versions of NDES.

Neo-Davidsonian semantics can be formalized using a simple type theory with three atomic types:

$$\begin{aligned} &\mathbf{e} \text{ (entities)} \\ &\mathbf{t} \text{ (truth values)} \\ &\mathbf{v} \text{ (events)} \end{aligned}$$

e and **t** are inherited from Montague [20], while **v** is introduced as the type of events. To illustrate the approach, let us give a simple example taken from [5]. Consider the following sentence:

(1) John kissed Mary

A classical Montague grammar would express the semantics of (1) by a simple atomic formula akin to the following one:

(2) $\mathbf{kiss}\,\mathbf{john}\,\mathbf{mary}$

where **kiss** is of type $\mathbf{e} \to \mathbf{e} \to \mathbf{t}$, and **john** and **mary** are of type **e**.

By contrast, a Montague grammar based on NDES would interpret (1) as follows:

(3) $\exists e.\,(\mathbf{kiss}\,e) \wedge (\mathbf{agent}\,e\,\mathbf{john}) \wedge (\mathbf{patient}\,e\,\mathbf{mary})$

whith **kiss** of type $\mathbf{v} \to \mathbf{t}$, and **agent** and **patient** of type $\mathbf{v} \to \mathbf{e} \to \mathbf{t}$. The intuition behind this neo-Davidsonian interpretation may be grasped by paraphrasing (3) as follows: *there is a kissing event the agent of which is John and the patient of which is Mary.*

One of the main arguments in support of NDES is its flexibility with regard to the treatment of the optional arguments of the verbs. To exemplify it, let us add to Example (1) an adverbial modifier.

(4) John kissed Mary in the garden

In Montague semantics, a verb phrase is interpreted as (the intension of) a set of entities. Accordingly, an adverbial modifier is interpreted as a set transformer. With such an approach, the semantic interpretation of (4) might be as below:

(5) $\mathbf{in}\,\mathbf{the\_garden}\,(\lambda x.\,\mathbf{kiss}\,x\,\mathbf{mary})\,\mathbf{john}$

where **in** is of type $\mathbf{e} \to (\mathbf{e} \to \mathbf{t}) \to \mathbf{e} \to \mathbf{t}$.

Using a neo-Davidsonian approach, the semantic interpretation of (4) would be rather different:

(6) $\exists e.\,(\mathbf{kiss}\,e) \wedge (\mathbf{agent}\,e\,\mathbf{john}) \wedge (\mathbf{patient}\,e\,\mathbf{mary}) \wedge (\mathbf{location}\,e\,\mathbf{the\_garden})$

It then appears that the entailment relation existing between (4) and (1) is semantically accounted for by the purely logical entailment of (3) by (6). This is not the case with the traditional Montagovian approach, where the entailment of (2) by (5) would necessitate some meaning postulates.

For the issue at hand in this paper, namely the semantic treatment of *wh*-questions, a neo-Davidsonian approach will allow us to interrogate the different thematic roles using a unique *interrogative quantifier*. Consider the three *wh*-questions one may derive from (4):

(7) a. Who did kiss Mary in the garden?

b. Whom did John kiss in the garden?

c. Where did John kiss Mary?

Using a unique interrogative quantifier, say "WHICH", our semantic account of (7a-c) will amount to a logical translation of the following respective paraphrases:

(8) a. WHICH *is the* agent *of the kissing event whose patient is Mary and whose location is the garden?*

b. WHICH *is the* patient *of the kissing event whose agent is John and whose location is the garden?*

c. WHICH *is the* location *of the kissing event whose agent is John and whose patient is Mary?*

Now, as it will appear in the sequel, this unique interrogative quantifier, WHICH, is in fact the existential quantifier of inquisitive semantics.

2.3 Inquisitive semantics

In Montague's intensional logic [20], as in modal logic, a declarative proposition is semantically interpreted as a set of possible worlds. In Hamblin-like logics of questions and answers, a question is identified with its set of possible answers. Therefore, since an answer is itself modeled by a declarative proposition, a question must be modeled by a set of declarative propositions. At the semantic levels, it means that a question must be interpreted as a set of sets of possible worlds.

Inquisitive semantics elaborates on this idea and stipulates, in addition, that both declarative and interrogative propositions must be interpreted as non-emtpty sets of sets of possible worlds that are downward closed by set inclusion. The consequences of this principle are twofold. On the one hand, inquisitive semantics provides a framework in which both declarative and interrogative expressions are treated in a uniform way. It is even the case that there is no neat separation between interrogative and declarative forms. In fact, in inquisitive semantics, every proposition has both an informative and an inquisitive content. On the

other hand, interpretating an inquisitive proposition as a set of sets that is downward closed allows conjunction and disjunction to be defined in a standard way, i.e., as intersection and union, respectively. The same is true of quantifiers: universal quantification is interpreted as an arbitrary intersection while existential quantification is interpreted as an arbitrary union. Let us illustrate all this with examples.

Let us posit a domain of interpretation, $D = \{a, b\}$, with two individuals (which we will call *Alice* and *Bob*), and a set of four possible worlds $W = \{\textsc{aa}, \textsc{as}, \textsc{sa}, \textsc{ss}\}$. The intended meaning of these four possible worlds is as follows: AA is the world where both Alice and Bob are awake; AS is the world where Alice is awake and Bob is sleeping; SA is the world where Alice is sleeping and Bob is awake; SS is the world where they are both sleeping. Then, the proposition φ_1 that *Alice sleeps* and the proposition φ_2 that *Bob sleeps* are interpreted as follows:

$$[\![\varphi_1]\!] = \{\{\textsc{sa}, \textsc{ss}\}, \{\textsc{sa}\}, \{\textsc{ss}\}, \varnothing\}$$
$$[\![\varphi_2]\!] = \{\{\textsc{as}, \textsc{ss}\}, \{\textsc{as}\}, \{\textsc{ss}\}, \varnothing\}$$

The inquisitive conjunction of φ_1 and φ_2 is interpreted as the intersection of their interpretations:

$$\begin{aligned}[\![\varphi_1 \wedge \varphi_2]\!] &= [\![\varphi_1]\!] \cap [\![\varphi_2]\!] \\ &= \{\{\textsc{sa}, \textsc{ss}\}, \{\textsc{sa}\}, \{\textsc{ss}\}, \varnothing\} \cap \{\{\textsc{as}, \textsc{ss}\}, \{\textsc{as}\}, \{\textsc{ss}\}, \varnothing\} \\ &= \{\{\textsc{ss}\}, \varnothing\}\end{aligned}$$

It corresponds to the assertion that both Alice and Bob are sleeping.

The inquisitive disjunction of φ_1 and φ_2 is interpreted as the union of their interpretations:

$$\begin{aligned}[\![\varphi_1 \vee \varphi_2]\!] &= [\![\varphi_1]\!] \cup [\![\varphi_2]\!] \\ &= \{\{\textsc{sa}, \textsc{ss}\}, \{\textsc{sa}\}, \{\textsc{ss}\}, \varnothing\} \cup \{\{\textsc{as}, \textsc{ss}\}, \{\textsc{as}\}, \{\textsc{ss}\}, \varnothing\} \\ &= \{\{\textsc{as}, \textsc{ss}\}, \{\textsc{sa}, \textsc{ss}\}, \{\textsc{as}\}, \{\textsc{sa}\}, \{\textsc{ss}\}, \varnothing\}\end{aligned}$$

This disjunction does not correspond to a proposition asserting that Alice or Bob is sleeping, but rather to the question *whether Alice or Bob is sleeping*. The mere assertion, φ_3, that *Alice or Bob is sleeping* is interpreted in a different way:

$$[\![\varphi_3]\!] = \{\{\textsc{as}, \textsc{sa}, \textsc{ss}\}, \{\textsc{as}, \textsc{sa}\}, \{\textsc{as}, \textsc{ss}\}, \{\textsc{sa}, \textsc{ss}\}, \{\textsc{as}\}, \{\textsc{sa}\}, \{\textsc{ss}\}, \varnothing\}$$

The proposition, φ_4 asserting that *Alice does not sleep* is interpreted as follows:

$$[\![\varphi_4]\!] = \{\{\textsc{aa}, \textsc{as}\}, \{\textsc{aa}\}, \{\textsc{as}\}, \varnothing\}$$

Then, the inquisitive disjunction of φ_1 and φ_4 corresponds to the polar question *whether Alice is sleeping*:

$$\begin{aligned}[\![\varphi_1 \vee \varphi_4]\!] &= [\![\varphi_1]\!] \cup [\![\varphi_4]\!] \\ &= \{\{\textsc{sa}, \textsc{ss}\}, \{\textsc{sa}\}, \{\textsc{ss}\}, \varnothing\} \cup \{\{\textsc{aa}, \textsc{as}\}, \{\textsc{aa}\}, \{\textsc{as}\}, \varnothing\} \\ &= \{\{\textsc{aa}, \textsc{as}\}, \{\textsc{sa}, \textsc{ss}\}, \{\textsc{aa}\}, \{\textsc{as}\}, \{\textsc{sa}\}, \{\textsc{ss}\}, \varnothing\}\end{aligned}$$

The interpretation of an inquisitive proposition being downward closed, it is completely characterized by its maximal elements. In our last example, these maximal elements are $\{\text{AA}, \text{AS}\}$ and $\{\text{SA}, \text{SS}\}$. They respectively corresponds to the set of worlds where Alice does not sleep, and to the one where she sleeps. In other words, these two maximal elements correspond to two modal propositions: one asserting that *Alice does not sleep*, and the other one, that *Alice sleeps*. These two propositions are precisely the two possible answers to the polar question *whether Alice is sleeping or not*. This illustrates that these maximal elements are, in fact, the counterpart of Hamblin's alternatives.

In inquisitive semantics, a proposition has both an informative and an inquisitive content. Its inquisitive content, which corresponds to Hamblin's alternatives, is given by the maximal elements, while its informative content is given by the merging of these maximal elements. For instance, the informative content of proposition $\varphi_1 \vee \varphi_2$ is that somebody (Alice or Bob) is sleeping, and its inquisitive content is the issue whether Alice or Bob is sleeping. The proposition may then be paraphrased as follows: *knowing that somebody is sleeping, one wonders whether Alice or Bob is sleeping*. The interpretation of a mere assertion such as φ_1 has only one maximal element. Accordingly, its inquisitive content is trivial, and its paraphrase would be: *knowing that Alice is sleeping, one wonders whether she is sleeping*. Similarly, a mere question such as $\varphi_1 \vee \varphi_4$ has a trivial informative content: *knowing that Alice sleeps or does not sleep, one wonders whether she is sleeping*. This absence of an actual informative content corresponds to the fact that the set of maximal elements of the proposition covers the set of possible worlds.

Inquisitive semantics features two projection operators, ! and ?, that respectively trivialize the inquisitive content and the informative content of a proposition. These operators may be defined as follows:

$$[\![!\varphi]\!] = \mathscr{P}(\bigcup [\![\varphi]\!])$$
$$[\![?\varphi]\!] = [\![\varphi]\!] \cup \mathscr{P}(W \setminus \bigcup [\![\varphi]\!])$$

where W is the set of possible worlds. Then, for any proposition φ, one has:

$$\varphi = !\varphi \wedge ?\varphi$$

Interestingly enough, these projection operators allow the interpretation of the logical connectives to be refined by providing them with different possible meanings. For instance, proposition φ_3, which corresponds to a non-inquisitive disjunction of φ_1 and φ_2, may be expressed as $!(\varphi_1 \vee \varphi_2)$.

We complete this brief introduction to inquisitive semantics by defining first-order inquisitive logic.

Let $\langle \mathcal{F}, \mathcal{R} \rangle$ be the signature of a first-order language, where $\mathcal{F}$ is the set of function symbols, and $\mathcal{R}$ is the set of relation symbols. From this signature together with a set $\mathcal{X}$ of first-order variables, the notions of terms and of first-order formulas are defined in the standard way.

The notion of a model is as usual in modal logic, i.e., a triple $\langle D, W, \mathcal{I} \rangle$, where D is the domain of interpretation, W is the set of possible worlds, an $\mathcal{I}$ is an

interpretation function ranging over $\mathcal{F} \cup \mathcal{R}$, such that:

$$\begin{aligned} \mathcal{I}(F) &\in D^{D^n} && \text{for } F \in \mathcal{F} \text{ of arity } n \\ \mathcal{I}(R) &\in \mathscr{P}(W)^{D^n} && \text{for } R \in \mathcal{R} \text{ of arity } n \end{aligned}$$

Given a valuation ξ from $\mathcal{X}$ into D, the interpretation $[\![t]\!]_\xi$ of a term t is defined as usual, and the interpretation of a first-order formula is then given by the following equations:

$$\begin{aligned} [\![R(t_1, \ldots, t_n)]\!]_\xi &= \mathscr{P}(\mathcal{I}(R)([\![t_1]\!]_\xi, \ldots, [\![t_n]\!]_\xi)) \\ [\![\neg\varphi]\!]_\xi &= \{s \mid \forall t \in [\![\varphi]\!]_\xi.\, s \cap t = \varnothing\} \\ [\![\varphi \wedge \psi]\!]_\xi &= [\![\varphi]\!]_\xi \cap [\![\psi]\!]_\xi \\ [\![\varphi \vee \psi]\!]_\xi &= [\![\varphi]\!]_\xi \cup [\![\psi]\!]_\xi \\ [\![\varphi \to \psi]\!]_\xi &= \{s \mid \forall t \subseteq s.\, t \in [\![\varphi]\!]_\xi \to t \in [\![\psi]\!]_\xi\} \\ [\![\forall x.\, \varphi]\!]_\xi &= \textstyle\bigcap_{d \in D} [\![\varphi]\!]_{\xi[x:=d]} \\ [\![\exists x.\, \varphi]\!]_\xi &= \textstyle\bigcup_{d \in D} [\![\varphi]\!]_{\xi[x:=d]} \end{aligned}$$

As for the projection operators ! and ?, they may be added as defined connectives:

$$\begin{aligned} !\varphi &= \neg\neg\varphi \\ ?\varphi &= \varphi \vee \neg\varphi \end{aligned}$$

In order to illustrate the difference between inquisitive first-order logic and usual classical first-order logic let us work out the inquisitive interpretation of an existential formula such as $\exists x.\, \mathbf{sleep}\, x$.

For the purpose of our example, we consider a model with the same domain of interpretation and the same set of possible worlds as previously. Then, we let the interpretation function $\mathcal{I}$ be such that:

$$\begin{aligned} \mathcal{I}(\mathbf{sleep})(a) &= \{\textsc{sa}, \textsc{ss}\} \\ \mathcal{I}(\mathbf{sleep})(b) &= \{\textsc{as}, \textsc{ss}\} \end{aligned}$$

In this setting, we have:

$$\begin{aligned} [\![\exists x.\, \mathbf{sleep}\, x]\!]_\xi &= \textstyle\bigcup_{d \in D} [\![\mathbf{sleep}\, x]\!]_{\xi[x:=d]} \\ &= ([\![\mathbf{sleep}\, x]\!]_{\xi[x:=a]}) \cup ([\![\mathbf{sleep}\, x]\!]_{\xi[x:=b]}) \\ &= (\mathscr{P}(\mathcal{I}(\mathbf{sleep})(a))) \cup (\mathscr{P}(\mathcal{I}(\mathbf{sleep})(b))) \\ &= \{\{\textsc{sa}, \textsc{ss}\}, \{\textsc{sa}\}, \{\textsc{ss}\}, \varnothing\} \cup \{\{\textsc{as}, \textsc{ss}\}, \{\textsc{as}\}, \{\textsc{ss}\}, \varnothing\} \\ &= \{\{\textsc{as}, \textsc{ss}\}, \{\textsc{sa}, \textsc{ss}\}, \{\textsc{as}\}, \{\textsc{sa}\}, \{\textsc{ss}\}, \varnothing\} \end{aligned}$$

This example shows that an inquisitive existential quantification, in general, has both an actual informative and an actual inquisitive content. This double content, which in the present case is the fact that somebody is sleeping and the issue whether it is Alice or Bob, may be adjusted using the projection operators. This gives rise to three kinds of existential quantifications, which in our example correspond to the following formulas:

(9) a. $!\,\exists x.\,\mathbf{sleep}\,x$

b. $?\,\exists x.\,\mathbf{sleep}\,x$

c. $\exists x.\,\mathbf{sleep}\,x$

Intuitively, these three formulas are the logical counterpart the three following utterances, respectively:

(10) a. Somebody is sleeping.

b. Who is sleeping? (*nobody* being a possible answer)

c. Somebody is sleeping. Who is it?

2.4 Abstract Categorial Grammars

The account of *wh*-question semantics that we give in the next sections is formalized using the framework of abstract categorial grammars [13]. For the sake of self-containedness, we give here the definitions necessary to understand this formalism.

We assume from the reader some acquaintance with the simply typed λ-calculus. Nevertheless, in order to fix the terminology, we briefly reminds the main definitions that we will be need in the sequel. In particular, we review the notions of *simple types*, *higher-order signature*, and *linear λ-terms* built upon a higher-order linear signature.

Let A be a set of atomic types. The set $\mathscr{T}(A)$ of *simple types* built upon A is inductively defined as follows:

1. if $a \in A$, then $a \in \mathscr{T}(A)$;
2. if $\alpha, \beta \in \mathscr{T}(A)$, then $(\alpha \to \beta) \in \mathscr{T}(A)$.

Given two sets of atomic types, A and B, a mapping $h : \mathscr{T}(A) \to \mathscr{T}(B)$ is called a *type homomorphism* (or a *type substitution*) if it satisfies the following condition:

$$h(\alpha \to \beta) = h(\alpha) \to h(\beta)$$

A *higher-order signature* consists of a triple $\Sigma = \langle A, C, \tau \rangle$, where:

1. A is a finite set of atomic types;
2. C is a finite set of constants;
3. $\tau : C \to \mathscr{T}(A)$ is a function that assigns to each constant in C a linear implicative type in $\mathscr{T}(A)$.

Let X be a infinite countable set of λ-variables. The set $\Lambda(\Sigma)$ of *linear λ-terms* built upon a higher-order linear signature $\Sigma = \langle A, C, \tau \rangle$ is inductively defined as follows:

1. if $c \in C$, then $c \in \Lambda(\Sigma)$;
2. if $x \in X$, then $x \in \Lambda(\Sigma)$;

3. if $x \in X$, $t \in \Lambda(\Sigma)$, and x occurs free in t exactly once, then $(\lambda x.\, t) \in \Lambda(\Sigma)$;
4. if $t, u \in \Lambda(\Sigma)$, and the sets of free variables of t and u are disjoint, then $(t\, u) \in \Lambda(\Sigma)$.

Let Σ_1 and Σ_2 be two signatures. We say that a mapping $h : \Lambda(\Sigma_1) \to \Lambda(\Sigma_2)$ is a λ-term homomorphism if it satisfies the following conditions:

$$\begin{aligned} h(x) &= x \\ h(\lambda x.\, t) &= \lambda x.\, h(t) \\ h(t\, u) &= h(t)\, (h(u)) \end{aligned}$$

Given a higher-order linear signature $\Sigma = \langle A, C, \tau \rangle$, each linear λ-term in $\Lambda(\Sigma)$ may possibly be assigned a linear implicative type in $\mathscr{T}(A)$. This type assignment obeys the following typing rules:

$$\vdash_\Sigma \; c : \tau_\Sigma(c) \quad \text{(CONS)}$$

$$x : \alpha \vdash_\Sigma \; x : \alpha \quad \text{(VAR)}$$

$$\frac{\Gamma, x : \alpha \vdash_\Sigma \; t : \beta}{\Gamma \vdash_\Sigma \; (\lambda x.\, t) : (\alpha \to \beta)} \text{ (ABS)}$$

$$\frac{\Gamma \vdash_\Sigma \; t : (\alpha \to \beta) \qquad \Delta \vdash_\Sigma \; u : \alpha}{\Gamma, \Delta \vdash_\Sigma \; (t\, u) : \beta} \text{ (APP)}$$

where $\mathrm{dom}(\Gamma) \cap \mathrm{dom}(\Delta) = \varnothing$.

Let $\Sigma_1 = \langle A_1, C_1, \tau_1 \rangle$ and $\Sigma_2 = \langle A_2, C_2, \tau_2 \rangle$ be two higher-order signatures. A *lexicon*, $\mathscr{L} : \Sigma_1 \to \Sigma_2$, is defined to be a realization of Σ_1 into Σ_2, i.e., an interpretation of the atomic types of Σ_1 as types built upon A_2, together with an interpretation of the constants of Σ_1 as linear λ-terms built upon Σ_2. These two interpretations must be such that their homomorphic extensions commute with the typing relations. More formally, a *lexicon* $\mathscr{L}$ from Σ_1 to Σ_2 is defined to be a pair $\mathscr{L} = \langle F, G \rangle$ such that:

1. $F : A_1 \to \mathscr{T}(A_2)$ is a function that interprets the atomic types of Σ_1 as linear implicative types built upon A_2;
2. $G : C_1 \to \Lambda(\Sigma_2)$ is a function that interprets the constants of Σ_1 as linear λ-terms built upon Σ_2;
3. the interpretation functions are compatible with the typing relation, i.e., for any $c \in C_1$, the following typing judgement is derivable:

$$\vdash_{\Sigma_2} \; G(c) : \hat{F}(\tau_1(c)) \qquad (\dagger)$$

where $\hat{F}$ is the unique homomorphic extension of F.

Remark that Condition (†) compels $G(c)$ to be typable with respect to the empty typing environment. This means that G interprets each constant c as a closed linear λ-term. Now, writing $\mathscr{L}$ for both the homomorphic extensions F and G, Condition (†) ensures that the following commutation property holds for every $t \in \Lambda(\Sigma_1)$:

$$\text{if} \quad \vdash_{\Sigma_1} t : \alpha \quad \text{then} \quad \vdash_{\Sigma_2} \mathscr{L}(t) : \mathscr{L}(\alpha)$$

We now define an *abstract categorial grammar* (ACG, for short) as a quadruple, $\mathscr{G} = \langle \Sigma_1, \Sigma_2, \mathscr{L}, S \rangle$, where:

1. Σ_1 and Σ_2 are two higher-order linear signatures; they are called the *abstract vocabulary* and the *object vocabulary*, respectively;
2. $\mathscr{L} : \Sigma_1 \to \Sigma_2$ is a lexicon from the abstract vocabulary to the object vocabulary;
3. S is an atomic type of the abstract vocabulary; it is called the *distinguished type* of the grammar.

Every ACG $\mathscr{G}$ generates two languages: an *abstract language*, $\mathcal{A}(\mathscr{G})$, and an object language $\mathcal{O}(\mathscr{G})$.

The abstract language, which may be seen as a set of abstract parse structures, is the set of closed linear λ-terms built upon the abstract vocabulary and whose type is the distinguished type of the grammar. It is formally defined as follows:

$$\mathcal{A}(\mathscr{G}) = \{t \in \Lambda(\Sigma_1) \,:\, \vdash_{\Sigma_1} t\colon S \text{ is derivable}\}$$

The *object language*, which may be seen as the set of surface forms generated by the grammar, is defined to be the image of the abstract language by the term homomorphism induced by the lexicon.

$$\mathcal{O}(\mathscr{G}) = \{t \in \Lambda(\Sigma_2) \,:\, \exists u \in \mathcal{A}(\mathscr{G}).\ t =_{\beta\eta} \mathscr{L}(u)\}$$

Both the abstract language and the object language generated by an ACG are sets of linear λ-terms. This allows more specific data structures such as strings, trees, or first-order terms to be represented. A string of symbols, for instance, can be encoded as a composition of functions. Consider an arbitrary atomic type s, and define $\sigma \triangleq s \to s$ to be the type of strings. Then, a string such as '*abbac*' may be represented by the linear λ-term:

$$\lambda x.\, a\,(b\,(b\,(a\,(c\,x)))),$$

where the atomic strings 'a', 'b', and 'c' are declared to be constants of type σ. In this setting, the empty word is represented by the identity function:

$$\epsilon \triangleq \lambda x.\, x$$

and concatenation is defined to be functional composition:

$$\_ + \_ \triangleq \lambda\alpha.\, \lambda\beta.\, \lambda x.\, \alpha\,(\beta\, x),$$

which is indeed an associative operator that admits the identity function as a unit.

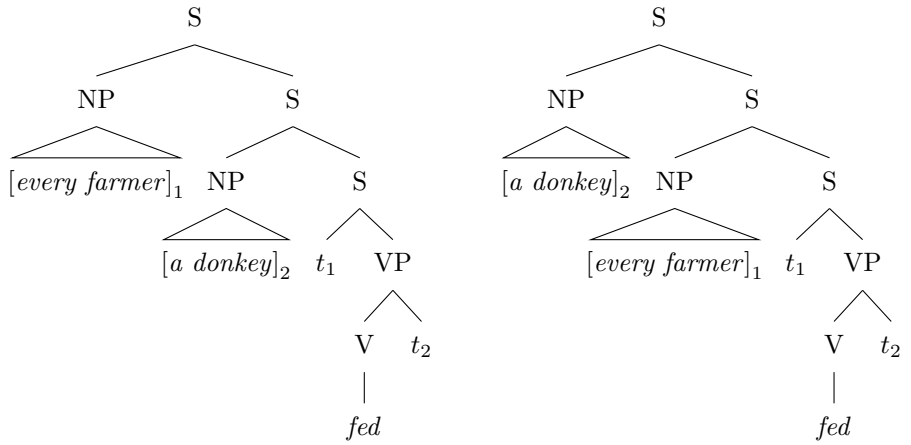
3 A categorial formalisation of the syntax and semantics of *wh*-interrogatives

It is usual in the categorial grammar tradition to distinguish between the quantified noun phrases and the mere noun phrases, at the syntactic level. While the latter are assigned a simple atomic type np, the former are assigned the functional type $(np \to s) \to s$, which reflects the fact that a quantified expression takes a scope. This allows for a smooth treatment of scope ambiguities.

Let us illustrate this approach by considering the following sentence:

(11) Every farmer fed a donkey.

the two possible readings of which are captured by the following two syntactic structures:



In order to give an abstract categorial account of sentence (11), one may declare abstract constants of the following types:

$$\begin{aligned} \textsc{farmer, donkey} &: n \\ \textsc{a, every} &: n \to (np \to s) \to s \\ \textsc{fed} &: np \to np \to s \end{aligned}$$

Then, the above syntactic structures are encoded in an almost straightforward way by the λ-terms given in Figure 1.

This categorial treatment of scope ambiguities, which directly derives from Montague [20], might be problematic when the targeted semantic formalism is Davidson's event semantics. It has indeed been argued that Montague's treatment of quantification does not combine smoothly with event semantics. The problem is that, in event semantics, a declarative sentence that is ultimately interpreted as a truth value ($\mathbf{t}$) is first interpreted as a set of events ($\mathbf{v} \to \mathbf{t}$). Then, switching from the latter interpretation of a sentence to the former necessitates an existential-closure operator, which may badly interact with the quantifiers that occur in the

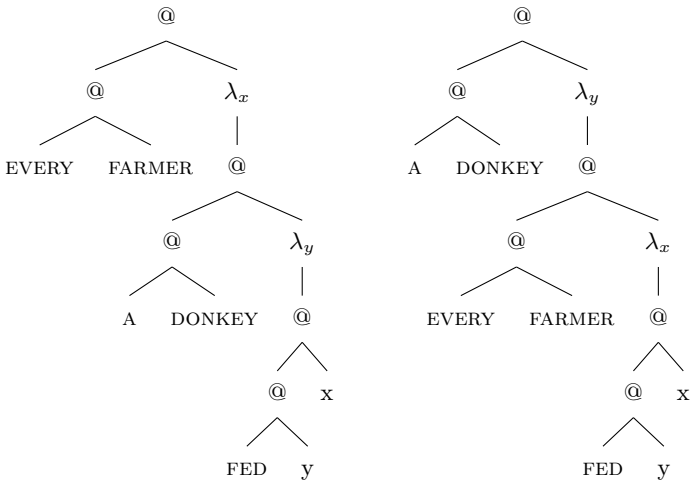


Fig. 1. Syntactic structures as linear λ-terms

interpretation of the sentence. Fortunately, the literature provides at least two solutions to this problem. A first one is due to Champollion [5], and a second one to Winter and Zwarts [23]. We follow this second solution since it is in line with the categorial tradition that we are advocating.

Winter and Zwarts' solution consists in assigning two different syntactic types to the sentences. On the one hand, a first type (s_0) is used for the "open" sentences, i.e., the sentences that are semantically interpreted as sets of events, and, on the other hand, a second syntactic type (s) is used for the sentences that are interpreted as truth values. Then, the existential closure operator allows values of type s_0 to be coerced into values of type s. Accordingly, the abstract signature we have sketched above is transformed as follows:

$$\begin{aligned} \textsc{farmer, donkey} &: n \\ \textsc{a, every} &: n \to (np \to s) \to s \\ \textsc{fed} &: np \to np \to s_0 \\ \textsc{e-clos} &: s_0 \to s \end{aligned}$$

This ensures that the existential closure operator will always take a narrower scope with respect to the other quantifiers.

Now, it is well known that there is a strong analogy between quantifier raising and *wh*-extraction. Following this analogy suggests that we should assign to a *wh*-noun phrase the type that we assign to a quantified noun phrase. Typically:

$$\textsc{who} : (np \to s) \to s$$

Similarly, a *wh*-determiner must be assigned the same type as a quantificational determiner:

$$\text{WHICH} : n \to (np \to s) \to s$$

Finally, *wh*-adverbs must be assigned the same type as the quantified adverbial modifiers, which are assigned $((s_0 \to s_0) \to s) \to s$ [14]. Accordingly, we have:

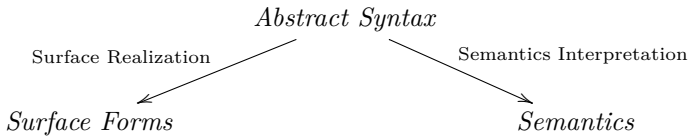
$$\text{WHERE} : ((s_0 \to s_0) \to s) \to s$$

Putting everything together, we end up with an abstract syntax specified by the signature given in Table 2, where E-CLOS and Q are syntactically silent operators. The first one allows an open sentence to be turned into a closed one. The second allows a declarative proposition to be turned into a polar question.

Table 2. Abstract syntax signature

Abstract Syntax	
FARMER, DONKEY, MEADOW	$: n$
THE	$: n \to np$
A, SOME, EVERY, WHICH	$: n \to (np \to s) \to s$
IN	$: np \to s_0 \to s_0$
FED, DID-FEED	$: np \to np \to s_0$
WHO	$: (np \to s) \to s$
WHERE	$: ((s_0 \to s_0) \to s) \to s$
E-CLOS	$: s_0 \to s$
Q	$: s \to s$

In abstract categorial grammars, the language generated by an abstract signature (such as the one given in Table 2) acts as a pivot language between surface forms and semantic interpretations. This is typically the way an abstract categorial grammar models the syntax-semantics interface:



Consequently, in order to complete the picture, it remains to give the syntactic and the semantic translations of the abstract syntax specified by the signature of Table 2. These are given in Table 3 and Table 4, respectively.

We may now illustrate the overall approach by treating the following example:

(12) Where did every farmer feed a donkey?

The above sentence contains three binding expression: a *wh*-adverb (*where*), and two quantified noun phrases (*every farmer* and *a donkey*). The relative scope of

Table 3. Surface realisation lexicon

Surface Realization	
FARMER	$:= \boldsymbol{farmer}$
DONKEY	$:= \boldsymbol{donkey}$
MEADOW	$:= \boldsymbol{meadow}$
THE	$:= \lambda x.\, \boldsymbol{the} + x$
A	$:= \lambda xp.\, p\,(\boldsymbol{a} + x)$
SOME	$:= \lambda xp.\, p\,(\boldsymbol{some} + x)$
EVERY	$:= \lambda xp.\, p\,(\boldsymbol{every} + x)$
WHICH	$:= \lambda xy.\, \boldsymbol{which} + x + (y\,\epsilon)$
IN	$:= \lambda xy.\, y + \boldsymbol{in} + x$
FED	$:= \lambda xy.\, y + \boldsymbol{fed} + x$
DID-FEED	$:= \lambda xy.\, \boldsymbol{did} + y + \boldsymbol{feed} + x$
WHO	$:= \lambda x.\, \boldsymbol{who} + (x\,\epsilon)$
WHERE	$:= \lambda q.\, \boldsymbol{where} + (q\,(\lambda x.\, x))$
E-CLOS	$:= \lambda x.\, x$
Q	$:= \lambda x.\, x$

Table 4. Semantic interpretation lexicon

Semantic Interpretation	
FARMER	$:= \lambda x.\, \mathbf{farmer}\, x$
DONKEY	$:= \lambda x.\, \mathbf{donkey}\, x$
MEADOW	$:= \lambda x.\, \mathbf{meadow}\, x$
THE	$:= \lambda p.\, \mathbf{the}\,(\lambda x.\, p\, x)$
A, SOME	$:= \lambda pq.\, !(\exists x.\, (p\, x) \wedge (q\, x))$
EVERY	$:= \lambda pq.\, \forall x.\, (p\, x) \rightarrow (q\, x)$
WHICH	$:= \lambda pq.\, \exists x.\, (p\, x) \wedge (q\, x)$
IN	$:= \lambda xp.\, \lambda e.\, (p\, e) \wedge (\mathbf{location}\, e\, x)$
FED, DID-FEED	$:= \lambda xy.\, \lambda e.\, (\mathbf{fed}\, e) \wedge (\mathbf{patient}\, e\, x) \wedge (\mathbf{agent}\, e\, y)$
WHO	$:= \lambda p.\, \exists x.\, p\, x$
WHERE	$:= \lambda p.\, \exists x.\, p\,(\lambda q.\, \lambda e.\, (q\, e) \wedge (\mathbf{location}\, e\, x))$
E-CLOS	$:= \lambda p.\, !(\exists e.\, p\, e)$
Q	$:= \lambda x.\, ?x$

these binding expressions must obey the constraint that the *wh*-expression takes the widest scope. Consequently, we are only left with two possible readings: one where the relative scope of the quantified noun phrases follows the surface order; another one where *a donkey* takes scope over *every farmer*.

These two readings correspond to two different syntactic structures, which are modelled as λ-terms built upon the signature given in Table 2. These λ-terms are the following ones:

(13) $\textsc{q}\,(\textsc{where}\ (\lambda f.\,\textsc{every farmer}\ (\lambda x.\,\textsc{a donkey}\ (\lambda y.\,\textsc{e-clos}\ (f\ (\textsc{did-feed}\ y\ x))))))$

(14) $\textsc{q}\,(\textsc{where}\ (\lambda f.\,\textsc{a donkey}\ (\lambda x.\,\textsc{every farmer}\ (\lambda y.\,\textsc{e-clos}\ (f\ (\textsc{did-feed}\ x\ y))))))$

Both (13) and (14) yield the same result when the surface realization lexicon given in Table 3 is applied to them:

(15) $\boldsymbol{where} + \boldsymbol{did} + \boldsymbol{every} + \boldsymbol{farmer} + \boldsymbol{feed} + \boldsymbol{a} + \boldsymbol{donkey}$

By contrast, when the semantic interpretation given in Table 4 is applied to them, (13) and (14) yield the two expected different readings:

(16) $?\,\exists x.\,\forall y.\,(\mathbf{farmer}\ y) \rightarrow\ !\,(\exists z.\,(\mathbf{donkey}\ z) \wedge\ !\,(\exists e.\,(\mathbf{fed}\ e) \wedge (\mathbf{patient}\ e\ z) \wedge (\mathbf{agent}\ e\ y) \wedge (\mathbf{location}\ e\ x)))$

(17) $?\,\exists x.\,!\,(\exists y.\,(\mathbf{donkey}\ y) \wedge\ (\forall z.\,(\mathbf{farmer}\ z) \rightarrow\ !\,(\exists e.\,(\mathbf{fed}\ e) \wedge (\mathbf{patient}\ e\ y) \wedge (\mathbf{agent}\ e\ z) \wedge (\mathbf{location}\ e\ x))))$

4 Controlling *wh*-extraction and quantifier raising

The grammar we have sketched in the previous section is quite simple, and has the advantage of highlighting the parallel that exists between declarative and interrogative sentences. In particular, it is based on a uniform treatment of quantification raising and *wh*-extraction. This simplicity, however, is not without its drawbacks. These are threefold. Firstly, our grammar assigns the same syntactic categories to both the declarative and the interrogative forms (for instance, EVERY and WHICH are both assigned $n \rightarrow (np \rightarrow s) \rightarrow s$). This gives rise to a grammar that generates ungrammatical surface forms such as:

(18) *Every farmer fed which donkey in which meadow.

Secondly, allowing the quantifiers to take any possible scope results in spurious ambiguities. For instance, a sentence such as:

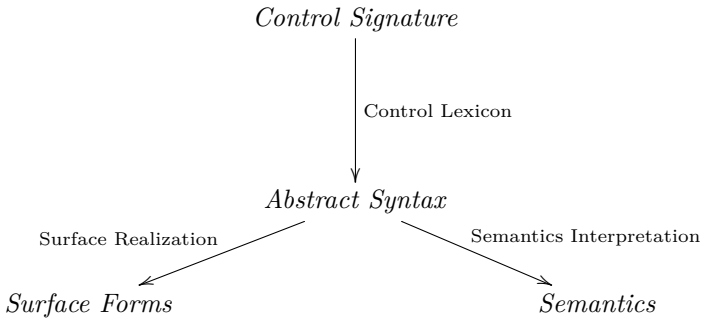
(19) A farmer fed a donkey in a meadow.

will give rise to six different abstract syntactic structures the semantic interpretations of which are all logically equivalent. Finally, the interactions between *wh*-extraction and quantifier raising must obey some constraints. For instance, in a *wh*-question, the *wh*-quantifier must always take the wider scope. Consider again example (12). The signature given in Table 2 allows one to built syntactic structures, such as the following one, that do not respect the *wh*-quantifier wider scope constraint:

(20) $\textsc{every farmer}\ (\lambda x.\ \textsc{where}\ (\lambda f.\ \textsc{a donkey}\ (\lambda y.\ \textsc{e-clos}\ (f\ (\textsc{did-feed}\ y\ x)))))$

Consequently, our grammar might allow nonsensical semantic interpretations to be derived.

The three kinds of defects that our grammar presents are all the consequence of a same fact: the *abstract syntax* signature of Table 2 allows too many abstract syntactic structures to be derived. In order to overcome this difficulty, we should be able to select among the λ-terms that can be built upon the signature of Table 2 the ones that correspond to legitimate abstract syntactic structures. A modular and efficient solution to this problem consists in using an additional abstract categorial grammar (which we will call the *control grammar*) in order to rule out the illegitimate abstract syntactic structures. This idea results in the following grammatical architecture:



In this architecture, the control signature may be seen as a type refinement of the abstract syntax. Typically, it distinguishes between different types of verb phrases and sentences, e.g., declarative verb phrase (*vp*) and interrogative verb phrase (*vpq*). This is useful, for instance, to prevent the grammar from assigning a declarative meaning to an interrogative sentence. It also distinguishes between different types of noun phrases, e.g, existentially quantified noun phrases (*npe*) and universally quantified noun phrases (*npu*). This is used to prevent a quantified noun phrase to be raised over another quantified noun phrase of the same quantificational force.

We have developed such a control grammar, which is unfortunately too large to be presented here[1]. Just to give a flavor of it, by way of illustration, we give in Tables 5 and 6 the excerpt that allows example (12) to be dealt with.

Table 5. Control signature (excerpt)

Control Signature	
a	$n \to npe$
every	$n \to npu$
donkey	n
farmer	n
Inpe	$npe \to np$
Inpu	$npu \to np$
did-feed	$np \to vpq$
did-feed$_2$	vpq_{np}
where	$sq_0 \to s$
SQ4	$np \to vpq \to sq_0$
SQ14	$npu \to vpq_{np} \to sq_{0npe}$
QRq13	$npe \to sq_{0npe} \to sq_0$

Table 6. Control lexicon (excerpt)

Control Lexicon	
a	$:= \lambda p.\, p\,(\text{A})$
every	$:= \lambda p.\, p\,(\text{EVERY})$
donkey	$:= \text{DONKEY}$
farmer	$:= \text{FARMER}$
Inpe	$:= \lambda x.\, x$
Inpu	$:= \lambda x.\, x$
did-feed	$:= \lambda pxf.\, p\,(\lambda y.\, f\,(\text{DID-FEED}\, y\, x))$
did-feed$_2$	$:= \lambda xyf.\, f\,(\text{DID-FEED}\, x\, y)$
where	$:= \lambda p.\, \text{Q}\,(\text{WHERE}\,(\lambda f.\, p\,(\lambda s.\, \text{E-CLOS}\,(f\, s))))$
SQ4	$:= \lambda pqf.\, p\,(\lambda x.\, q\, x\, f)$
SQ14	$:= \lambda pqxf.\, p\,(\lambda y.\, q\,(x\, y)\, f)$
QRq13	$:= \lambda pqf.\, p\,(\lambda x.\, q\, x\, f)$

[1] It is about three times larger than the grammar presented in Section 3.

5 Conclusion and future work

The semantic analysis of *wh*-questions that we have presented in this paper is based on a strong parallel between *wh*-words and thematic roles. This parallel, however, is not always as obvious as it might seems. Let us discuss the cases of some *wh*-words that might be problematic: *whose*, *how*, *what*, and *why*.

whose raises the problem of modeling the possessive relation. It is well known indeed that the possessive relation is multiple and that it does not correspond to a single thematic relation. In many cases, the type of the relation and the thematic role played by the possessor can be determined from the lexical semantics of the possessed entity. In some other cases, however, the nature of the possessive relation also depends on the nature of the possessor. These difficulties, in fact, are not specific to the use of the *wh*-word *whose*. They are related to the semantics of possessives, which is a subject on its own.

Regarding *how*, the difficulty is also contextual: the meaning of the *wh*-word *how* depends on the expression *how* is paired with. Consider the difference between *how long* and *how far*. In the first case, the interrogated thematic role might be time-related, while in the second case, it is location-related.

In many cases, *what* appears to be close in behavior either to *who* and *whom*, or to *which*. The difference between *what* and *which* seems to come from pragmatic considerations: the interpretation of *what* hugely depends on the context in which this interrogative word is used, while *which* is restrained in its interpretation by the definition of the set from which the choice of the response is made. *what* may also occur in a generic question such as *what did the farmer do*. This question does not interrogate a thematic role but rather the nature of an event. It could be paraphrase as *of which kind of event was the farmer the agent*. In the current state of our model, this cannot be treated because it would require a second-order quantification.

Finally, the difficulty with *why* is that it does not interrogate a thematic role but rather the argument of a discourse relation. Consequently, in order to propose a treatment of *why*, we would need to extend our model with a theory of discourse, including a theory of discursive relations.

All the possible extensions that we have discussed above will be subject to further work.

Bibliography

[1] Aarts, B., Chalker, S., Weiner, E.: The Oxford dictionary of English grammar. Oxford University Press (2014)

[2] Bender, E.M., Lascarides, A.: Linguistic fundamentals for natural language processing ii: 100 essentials from semantics and pragmatics. Synthesis Lectures on Human Language Technologies **12**(3), 1–268 (2019)

[3] Bonial, C., Corvey, W., Palmer, M., Petukhova, V.V., Bunt, H.: A hierarchical unification of lirics and verbnet semantic roles. In: 2011 IEEE Fifth International Conference on Semantic Computing. pp. 483–489. IEEE (2011)

[4] Bunt, H.: The dit++ taxonomy for functional dialogue markup. In: AAMAS 2009 Workshop, Towards a Standard Markup Language for Embodied Dialogue Acts. pp. 13–24 (2009)

[5] Champollion, L.: The interaction of compositional semantics and event semantics. Linguistics and Philosophy **38**(1), 31–66 (2015)

[6] Ciardelli, I., Groenendijk, J., Roelofsen, F.: Inquisitive semantics. Oxford University Press (2018)

[7] Ciardelli, I., Roelofsen, F., Theiler, N.: Composing alternatives. Linguistics and Philosophy **40**(1), 1–36 (2017)

[8] Dowty, D.R.: Thematic roles and semantics. In: Annual Meeting of the Berkeley Linguistics Society. vol. 12, pp. 340–354 (1986)

[9] Fillmore, C.J., Fillmore, S.: Case grammar. Universals in Linguistic Theory. Holt, Rinehart, and Winston: New York pp. 1–88 (1968)

[10] Ginzburg, J., Sag, I.: Interrogative investigations. Stanford: CSLI publications (2000)

[11] Groenendijk, J., Stokhof, M.: Questions. In: van Benthem, J., ter Meulen, A. (eds.) Handbook of Logic and Language. Elsevier, second edn. (2011)

[12] Groenendijk, J., Stokhof, M.: Studies on the Semantics of Questions and the Pragmatics of Answers. Ph.D. thesis, Univ. Amsterdam (1984)

[13] de Groote, Ph.: Towards Abstract Categorial Grammars. In: Association for Computational Linguistics, 39th Annual Meeting and 10th Conference of the European Chapter, Proceedings of the Conference. pp. 148–155 (2001)

[14] de Groote, Ph., Winter, Y.: A type-logical account of quantification in event semantics. In: Murata, T., Mineshima, K., Bekki, D. (eds.) LENLS, JURISIN, and GABA, Kanagawa, Japan, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9067, pp. 53–65. Springer (2014)

[15] Gruber, J.S.: Studies in lexical relations. Ph.D. thesis, Massachusetts Institute of Technology (1965)

[16] Hamblin, C.L.: Questions in Montague English. Foundations of Language **10**(1), 41–53 (1973)

[17] Kratzer, A., Heim, I.: Semantics in generative grammar, vol. 1185. Blackwell Oxford (1998)

[18] Montague, R.: English as a formal language. In: et al., B.V. (ed.) Linguaggi nella Società e nella Tecnica. Edizioni di Communità, Milan (1970), reprinted: [21, pages 188–221]

[19] Montague, R.: Universal grammar. Theoria **36**, 373–398 (1970), reprinted: [21, pages 222–246]
[20] Montague, R.: The proper treatment of quantification in ordinary english. In: Hintikka, J., Moravcsik, J., Suppes, P. (eds.) Approaches to natural language: proceedings of the 1970 Stanford workshop on Grammar and Semantics. Reidel, Dordrecht (1973), reprinted: [21, pages 247–270]
[21] Montague, R.: Formal Philosophy: selected papers of Richard Montague, edited and with an introduction by Richmond Thomason. Yale University Press (1974)
[22] Parsons, T.: Events in the Semantics of English, vol. 5. MIT press Cambridge, MA (1990)
[23] Winter, Y., Zwarts, J.: Event semantics and Abstract Categorial Grammar. In: Kanazawa, M., Kornai, A., Kracht, M., Seki, H. (eds.) Proceedings of Mathematics of Language, MOL12. Lecture Notes in Artificial Intelligence, LNAI, vol. 6878, pp. 174–191. Springer-Verlag, Berlin (2011)
[24] Wiśniewski, A.: Semantics of questions. In: Lappin, S., Fox, C. (eds.) The Handbook of Contemporary Semantic Theory, chap. 9, pp. 271–313. John Wiley & Sons, Ltd (2015)

Covariant Subtyping Applied to Semantic Predicate Calculi

William Babonnaud

Loria, Université de Lorraine, CNRS, Inria Nancy Grand-Est, Nancy, France
william.babonnaud@loria.fr

Abstract. Manipulating type hierarchies in formal semantic frameworks is often performed through a subtyping relation which obeys the contravariant rule for the left argument of a function type, due to the traditional representation of predicates as functions. This approach has however serious drawbacks when handling modifiers for first-order predicates. The present paper adopts an opposite view on subtyping by introducing a predicate calculus with a covariant behaviour, endowed with a categorical semantics in which subtyping coercions behave as generalisations of injective functions, and predicates are assimilated to powerobjects. This calculus is type safe in the sense that it prevents unwanted term applications, and is shown to provide a solution for the difficulties faced by a contravariant subtyping.

1 Subtyping in Natural Language Semantics

Integrating lexical semantics into formal frameworks, in the form of complex type systems using large hierarchies of base types, has been a challenging step in natural language semantics, opening new questions on the interaction between these hierarchies and compositionality. As in programming languages, the two (possibly overlapping) main strategies introduced in semantic calculi to deal with this multitude of types are *type polymorphism* (e.g. in MGL [27]), and *subtyping* (e.g. in UTT [19] and TCL [1]). The comparison with programming languages is relevant here to the extent that theories of polymorphism and subtyping have emerged firstly in this field, through the works of Milner [22] for the former, and Reynolds [28] and Cardelli [7, 8] for the latter. In the present paper, we shall focus on subtyping.

Roughly speaking, subtyping is comparable to the subset relation for sets: if an entity is of type a and a is a subtype of b, then this entity can also be seen as being of type b. Actually, Cardelli [7] proposed a set-theoretic semantics of his calculus in which subtyping corresponds exactly to set inclusion on the semantic side. This approach relies on the idea of a large domain of values V whose type interpretations are subsets, and function types are interpreted as subsets of $V \to V$ in such a way that if α is a subtype of β, then the interpretation of $\beta \to \gamma$ is a subset of the interpretation of $\alpha \to \gamma$, thus licensing the *contravariant rule* for function types, that is, the fact that the subtyping order for functions is reversed w.r.t. the domain type. However, large domains of values of this kind

are better known as denotational semantics for interpreting systems such as the untyped lambda-calculus [31]. When it comes to typed lambda-calculus, a better set-theoretic interpretation of terms of type $\alpha \to \beta$ is given by the set B^A of (continuous) functions from A to B [5, 4]; but in such a semantics, the inclusion relation from B^C to B^A when $A \subset C$ does not hold anymore, and the correspondence between contravariant subtyping and set inclusion is lost.[1] Actually, we have almost the converse semantic relation: the domain-restriction function $f \mapsto f|_A$ from B^C to B^A is generally surjective.[2]

Instead of inclusion, a more general idea conveyed by subtyping is related to type safety: whenever α is a subtype of β, a term of type α can be used at any place where a term of type β is expected without threatening the behaviour of the system. An even stronger caracterisation of the subtyping relation has been proposed by Liskov and Wing [16]: they require that whenever α is a subtype of β and ϕ is a property provable for all objects $x : \beta$, then ϕ must be provable for all objects $y : \alpha$, where the intended properties are related to safety. These accounts of safety naturally result in the contravariance rule; it is thus not surprising that applications of subtyping in formal semantic frameworks follow this approach. These applications are most commonly *coercive* [26, 17, 19], that is, the conversion of a term from a type to its supertype is done explicitly with a specific functional term called *coercion*. Such a choice is grounded in type-theoretical considerations (see e.g. [19, §4]), and in terms of set interpretation, amounts to replace inclusions by injective functions between disjoint domains. If $c : \alpha \to \beta$ is such a coercion, the coercion corresponding to the contravariant rule for functions is easily built as $\lambda f.\lambda x.f(cx) : (\beta \to \gamma) \to \alpha \to \gamma$ for any γ. This ease of usage, along with the amount of theoretical studies that supports it, explain why contravariant subtyping has been naturally imported to computational semantics.

However, as in programming languages [9], contravariance is the source of many difficulties in the interactions between subtyping and classical predicate interpretations inherited from Montague [23]. The following example is adapted from Luo [18]: suppose we are provided with a type v of vehicles and a type p of physical objects, with the subtyping relation $v \leq p$. Let $\mathbf{car} : v \to t$ and $\mathbf{heavy} : (p \to t) \to p \to t$ be predicates corresponding the noun "car" and the adjective "heavy". Then, contravariance implies $p \to t \leq v \to t$, which means that **car** is not an acceptable argument for **heavy**. Besides, contravariance also implies that general type schemes for adjective representation $(\alpha \to t) \to \alpha \to t$ and $(\beta \to t) \to \beta \to t$ are not comparable even if α and β are, since they appear both positively and negatively in the schemes. These remarks have notably led

[1] A different behaviour shows up whenever considering the set of *partial* functions: covariance of domain types is obtained instead of contravariance. However, using partial functions in natural language semantics may produce new kinds of difficulties to cope with.

[2] We may also build an injective function from B^A to B^C which extends the domain of its argument and sets a defaut value on this extension, but if no specific value in B is accessible, the axiom of choice is required to select this defaut value. We will see in Sect. 5 that our proposal uses a similar idea for predicates, since their codomain has precisely an accessible defaut value which must be "false".

to criticisms against Montagovian-based predicate models, for instance in [10]. In this paper, we redirect criticms against contravariance subtyping, which seems inadapted to the requirements of semantic-modelling calculi.

The idea of *covariant subtyping* have been pointed out as generally unsafe in programming languages.[3] However, formal frameworks for natural language semantics do not share the object-oriented perspective on types inherited from programming, where objects are thought as collections of methods in records [7, 11]. In linguistic applications, an important conceptual shift occurs by replacing objects by *entities*, which do not carry methods of their own: predicates—which generally arise from words through their part of speech, following the initial proposal of Montague [23]—exist independently of entities, and simply may or may not be applicable to them depending on their types. Even if we are right when saying that physical entities have a mass, **mass** is not a method of physical entities but a predicate which is meaningfully applicable only to physical entities.

The flexibility of language is such that predicates may be applied to entities that does not match their type requirements, as for instance in the sentence "the table talks". If predicates are interpreted as functions from their meaningful argument type to propositions or truth values, this means that they may be extended to other arguments without losing their sense nor their distinction from other predicates—an idea which is similar to covariance.[4] The key to convert this remark in a more general framework is to note that, in a semantic calculus, the only functions whose codomain is not the proposition type t are base type coercions themselves. We may thus take a sharper distinction between predicates and other functions in order to provide a well-founded covariant subtyping for predicate calculus. By doing so, we may also retrieve the correspondence between subtyping relations and injective functions, provided that we are able to extend the domain of a predicate in a satisfying way. This property would ensure that whenever coerced to their supertypes, entities and predicates remain distinct, which corresponds to what happens in natural language.[5]

We shall thus introduce the basis of a predicate calculus specifically designed for applications in natural language semantics, abstracting predicate types by new type constructors rather than using the traditional function types of codomain t. Besides, this calculus will be completed with a covariant approach to subtyping, and will be given a general semantics in category theory. The types and terms of this calculus are introduced in Sect. 2, and its categorical interpretation is presented in Sect. 3. Through a semantic-driven construction, Sect. 4 develops the rules of subtyping and shows that subtyping coercions thus defined are

[3] In a specific setting, Castagna [9] showed that contravariance and covariance describe different mechanisms which may be adapted to coexist in a type-safe way, but the actual mechanism he defines is not exactly relevant to our purposes.

[4] It also underlies the solution proposed by Asher to overcome the difficulties presented above with the covariant type $\exists x \sqsubseteq \alpha.x \to t$ [1, §4.2].

[5] This has to be opposed to some accounts of subtyping in programming languages where distinct objects may be identified through subtyping coercion, as in the example of stacks as subtypes of bags discussed in [16, §2]: in these accounts, subtyping is surjective rather than injective.

interpreted as generalisation of injective functions. Sect. 5 supplies additional results on the interaction between covariant subtyping and logical operators, and gives more intuition on the behaviour of coerced predicates. Sect. 6 discusses possible extensions and future perspectives.

2 A Predicate Calculus with Abstracted Functions

Rather than introducing a new version of a full lambda-calculus with some slight changes compared to other proposals of Montagovian inspiration, we propose a system which may appear weaker at first glance, but which actually abstracts upon the simply-typed lambda-calculus. Motivation for such a proposal comes from the fact that the expressive power of a semantic framework, viewed as a system which ultimately aims at building logical formulae to represent the intended semantics of sentences, relies mainly on three ingredients: predicates, logical connectives, and construction rules. The first two of these ingredients are rendered as constants in almost every calculus, and the last one is provided by general rules for constructing lambda-terms, *application* being the most important thereof.

It is particularly notable that the lambda-terms appearing in such frameworks are closed ones, and that variables are mainly used for composition purposes as an intermediate step before applying a rule of λ-abstraction. For instance, combining two first-order predicates $u, v : e \to t$ with the logical connective $\wedge : t \to t \to t$ use several applications and an abstraction to result in $\lambda x.u(x) \wedge v(x)$.[6] Provided that we have access to closed extensions of $\wedge$ for use with other types, for instance with the polymorphic AND definable in MGL [27], the same result could be obtained up to η-expansion as $\wedge^{e\to t}\, u\, v$ in only two applications. We shall generalise this reasoning by designing a "weaker" calculus which has no direct access to term variables and no abstraction rule.

Following this idea and the discussions on subtyping from Sect. 1, we shall therefore introduce the covariant subtyping calculus $C\Sigma$. It can be seen as a higher-level lambda-calculus, in the same way as programming languages such as Python are high-level languages w.r.t. machine code or assembly language: its types will be abstractions of common types, its terms will be constants and combinations thereof, and its rules will be restricted to application and other combining rules—but all of these components will stay definable within a more general typed lambda-calculus. As such, $C\Sigma$ is intended to provide the minimal setting for semantic purposes, and to be cleared of any other unnecessary feature.

We will successively define the three main parts of the type system of $C\Sigma$, namely the *type ontology*, the *type constructors*, and the *coercion types*. Altogether, these pieces build the set $\mathcal{T}_{C\Sigma}$ of types of our calculus.

Definition 1. *A* ***type ontology*** *is a pair* $(\mathbb{B}, \leq)$ *where* $\mathbb{B}$ *is a set of* base types *and* $\leq \subset \mathbb{B} \times \mathbb{B}$ *is a partial order endowed with a greatest element* $e \in \mathbb{B}$.

[6] Throughout this paper usual logical connectives will be assumed to carry their usual infix notation.

While any type hierarchy may satisfy this definition, the term *ontology* is reminiscent of Fred Sommers' theory of ontological types [32, 33], and hints how we would expect the base types to be constructed. In particular, we think the hierarchical structure underlying $\leq$ to be almost a tree.[7]

The next step is to introduce the type constructors. Leaving aside coercions, we will use a unit type as well as products and predicates.

Definition 2. *Let $(\mathbb{B}, \leq)$ be a type ontology. The set $\Xi_{C\Sigma}$ of* ***complex types upon*** *$\mathbb{B}$ is defined recursively by the grammar*

$$\Xi_{C\Sigma} ::= 1 \mid \mathbb{B} \mid \Xi_{C\Sigma} \times \Xi_{C\Sigma} \mid \mathcal{P}\Xi_{C\Sigma}$$

where 1 is the unit type, and $\mathcal{P}$ is the predicate constructor.

We may assume the product constructor $\times$ to be associative, so that we can avoid parentheses in successive applications. Compared to common type theories in semantic frameworks, one may notice that we did not introduce a type t for truth values or propositions. It is actually hidden in predicate types: for any $\alpha \in \Xi_{C\Sigma}$, the type $\mathcal{P}\alpha$ of predicates upon α corresponds conceptually to the simple type $\alpha \to t$. Furthermore, a predicate upon a product type corresponds to the uncurried version of a n-ary function of codomain t, and as 1 stands for the empty product, the type t is actually $\mathcal{P}1$. The choice of $\mathcal{P}$ rather than $\to$ is intended to highlight the conceptual differences between predicates and other functions, as will be clarified when introducing subtyping rules in Sect. 4. The arrow constructor is however not ruled out, as it is used for defining coercions.

Definition 3. *Let $\Xi_{C\Sigma}$ be a set of complex types upon some type ontology. A* ***$\Xi_{C\Sigma}$-coercion*** *is a type of the form $\alpha \to \beta$, where $\alpha, \beta \in \Xi_{C\Sigma}$.*

As to be seen in Sect. 4, we will generally use exactly one coercion term for each $\Xi_{C\Sigma}$-coercion, that is, we will not allow to use two different coercions for the same arrow type. This assumption enforces the coherence of the system of coercions, and henceforth enables us to identify bijectively each $\Xi_{C\Sigma}$-coercion, which is a type, to a corresponding term which is also called *coercion*. The simplest version of the $C\Sigma$ calculus can use the full set of coercions $\Xi_{C\Sigma} \to \Xi_{C\Sigma}$, but the resulting theory shall be degenerate. Therefore, we will assume to have a specific subset $\mathcal{K} \subset \Xi_{C\Sigma} \to \Xi_{C\Sigma}$ of coercions, whose construction will be constrained by the guidelines in Sect. 4.

Definition 4. *Let $(\mathbb{B}, \leq)$ be a type ontology and $\mathcal{K}$ a set of $\Xi_{C\Sigma}$-coercions. The set $\mathcal{T}_{C\Sigma}$ of* ***$C\Sigma$-types based upon*** *$\mathbb{B}$* ***and*** *$\mathcal{K}$ is defined as:*

$$\mathcal{T}_{C\Sigma} = \Xi_{C\Sigma} \cup \mathcal{K}$$

We hinted earlier that the $C\Sigma$ calculus should be definable within a lower-level typed lambda-calculus. Being given the set $\mathcal{T}$ of types of such a calculus,

[7] The reader may also consult [2, 29] for recent accounts of Sommers' theory.

including the base types of $\mathbb{B}$, a proposition type t, a unit type 1, products and arrows, we can define an encoding $\theta : \mathcal{T}_{C\Sigma} \to \mathcal{T}$ in the following way:

$$\begin{aligned} \theta(1) &= 1 & \\ \theta(b) &= b & \text{for } b \in \mathbb{B} \\ \theta(\alpha \times \beta) &= \theta(\alpha) \times \theta(\beta) & \text{for } \alpha, \beta \in \Xi_{C\Sigma} \\ \theta(\mathcal{P}(\alpha_1 \times \cdots \times \alpha_n)) &= \theta(\alpha_1) \to \cdots \to \theta(\alpha_n) \to t & \\ & & \text{for } \alpha_1, \ldots, \alpha_n \in \Xi_{C\Sigma} \\ \theta(\alpha \to \beta) &= \theta(\alpha) \to \theta(\beta) & \text{for } \alpha \to \beta \in \mathcal{K} \end{aligned} \tag{1}$$

We now turn to terms. As previously suggested, $C\Sigma$ is mainly constant-based and will not use variables nor lambda-abstraction, at least directly; yet some terms introduced here as constants may be definable as operators in a lower-level calculus precisely by using those hidden constructions. In the rest of this paper we fix a type ontology and a coercion set, so that the set of $C\Sigma$-types is also fixed. We call *constant signature* a pair $(\mathcal{Q}, \tau)$ where $\mathcal{Q}$ is a set of constants and τ is a function $\mathcal{Q} \to \Xi_{C\Sigma}$, and *coercion signature* a pair (K, σ) where K is a set of coercions and σ a bijective function $K \to \mathcal{K}$.

Definition 5. *Let $(\mathcal{Q}, \tau)$ and (K, σ) be constant and coercion signatures. The set $\Lambda_{C\Sigma}$ of **untyped terms of** $C\Sigma$ is recursively defined by the grammar:*

$$\Lambda_{C\Sigma} ::= * \mid \mathcal{Q} \mid K\Lambda_{C\Sigma} \mid \langle \Lambda_{C\Sigma}, \Lambda_{C\Sigma} \rangle \mid \Lambda_{C\Sigma}\Lambda_{C\Sigma} \mid \pi_1\Lambda_{C\Sigma} \mid \pi_2\Lambda_{C\Sigma}$$

Through misuse of language, we will consider $\mathcal{K}$ itself to be a coercion signature, and denote by $c : \alpha \to \beta \in \mathcal{K}$ that c is the symbol representing the coercion from α to β.

The restriction to constants within $C\Sigma$ also makes typing judgements simpler, since variable environments are not needed anymore. As usual, a *typing judgement* will be a sequent $\vdash u : \alpha$ where u is a term and α is a type in $\Xi_{C\Sigma}$. We shall introduce the typing rules in two batches, starting with the more direct rules and postponing the rules using coercions to Sect. 4. The first batch contains the basic rules for constants and pairs, as well as direct application:

$$\frac{}{\vdash * : 1}\,(\textsc{unit}) \qquad \frac{u \in (\mathcal{Q}, \tau)}{\vdash u : \tau(u)}\,(\textsc{const}) \qquad \frac{\vdash u : \alpha \quad \vdash v : \beta}{\vdash \langle u, v \rangle : \alpha \times \beta}\,(\textsc{pair})$$

$$\frac{\vdash u : \mathcal{P}(\alpha \times \beta) \quad \vdash v : \alpha}{\vdash uv : \mathcal{P}\beta}\,(\textsc{app}) \qquad \frac{\vdash u : \mathcal{P}\alpha \quad \vdash v : \alpha}{\vdash uv : \mathcal{P}1}\,(\textsc{app}') \tag{2}$$

$$\frac{\vdash u : \alpha \times \beta}{\vdash \pi_1 u : \alpha}\,(\textsc{proj}_1) \qquad \frac{\vdash u : \alpha \times \beta}{\vdash \pi_2 u : \beta}\,(\textsc{proj}_2)$$

Similarly to product types, we may assume that pairs extend to tuples of any size. As for direct application, the rules (app) and (app′) distinguish the cases of "partial" and "total" application of predicates. We may have considered an equivalent statement by using only the rule (app) and an isomorphic operator

$\mathcal{P}\alpha \cong \mathcal{P}(\alpha \times 1)$, thus enabling us to derive (APP′) as an admissible rule; yet we will keep the latter rule as such for clarity. One may observe that no rule is provided to build predicate terms: all predicates have to be introduced as constants. This implies of course that the constant signature is non-empty and contains at least one predicate, which is consistent with the semantic aim of the calculus: predicates in natural language semantics are generally introduced as constants, and we argue that $C\Sigma$ provides enough material to make explicit construction of predicates as lambda-abstractions unnecessary. In particular, the next section will introduce the logical operators that enables one to build complex predicates from predicate constants.

We conclude the present section by stating the mandatory term equalities. Besides the traditional rules of congruence (REFL), (SYM) and (TRANS), we assert in (ONE) that $*$ is the only term of type 1, in (EQPR) and (EQPAR) that equalities propagate to products and applications, in (PRJ_1) and (PRJ_2) that projections work on pairs as expected, and in (AP*) that successive applications of a predicate to several arguments amount to applying it to the tuple of these arguments. Furthermore, the rule (ASSOC) states the tuple equality which licenses the use of $\times$ as an associative type constructor. Both (EQAP) and (AP*) correspond to the previous typing rule (APP); the definition of their counterparts for the rule (APP′) is left as an exercise to the reader. Here again, the rules involving coercions are postponed to Sect. 4.

$$\frac{\vdash u : \alpha}{\vdash u = u : \alpha}\,(\text{REFL}) \qquad \frac{\vdash u = v : \alpha}{\vdash v = u : \alpha}\,(\text{SYM}) \qquad \frac{\vdash u : 1}{\vdash u = * : 1}\,(\text{ONE})$$

$$\frac{\vdash u = v : \alpha \quad \vdash v = w : \alpha}{\vdash u = w : \alpha}\,(\text{TRANS}) \qquad \frac{\vdash u = u' : \alpha \quad \vdash v = v' : \beta}{\vdash \langle u, v\rangle = \langle u', v'\rangle : \alpha \times \beta}\,(\text{EQPR})$$

$$\frac{\vdash u : \alpha \quad \vdash v : \beta}{\vdash \pi_1\langle u, v\rangle = u : \alpha}\,(\text{PRJ}_1) \qquad \frac{\vdash u = u' : \mathcal{P}(\alpha \times \beta) \quad \vdash v = v' : \alpha}{\vdash uv = u'v' : \beta}\,(\text{EQAP}) \tag{3}$$

$$\frac{\vdash u : \alpha \quad \vdash v : \beta}{\vdash \pi_2\langle u, v\rangle = v : \beta}\,(\text{PRJ}_2) \qquad \frac{\vdash u : \mathcal{P}(\alpha \times \beta \times \gamma) \quad \vdash v : \alpha \quad \vdash w : \beta}{\vdash (uv)w = u\langle v, w\rangle : \mathcal{P}\gamma}\,(\text{AP}^*)$$

$$\frac{\vdash u : \alpha \quad \vdash v : \beta \quad \vdash w : \gamma}{\vdash \langle u, \langle v, w\rangle\rangle = \langle\langle u, v\rangle, w\rangle : \alpha \times \beta \times \gamma}\,(\text{ASSOC})$$

3 Categorical Model of $C\Sigma$

In order to study more efficiently its underlying type theory, we define an interpretation of $C\Sigma$ in category theory. Many equivalences between type systems and classes of categories have been identified, notable examples including the correspondences between simply-typed λ-calculi and cartesian closed categories [14], as well as between Martin-Löf type theories and locally cartesian closed categories [30]. As $C\Sigma$ is "high-level", we may expect its categorical model to

have weaker assumptions; however, this will not actually be the case: we need all the power of the underlying "low-level" model to interpret efficiently our calculus.

The rest of this paper assumes preliminary knowledge of category theory, including the notions of categories, initial and terminal objects, products, exponentials, pullbacks, functors, natural transformations, and adjunctions.[8] The reader may consult [20, 25] for an introduction; useful elements (with increasing difficulty) can also be found in [12, 21, 13]. Only three definitions will be recalled here: *monomorphisms*, *subobject classifiers*, and *toposes*.

Definition 6. *A morphism $f : A \to B$ is a* ***monomorphism*** *(or "is mono", noted $f : A \rightarrowtail B$) if for any pair of morphisms $g, h : C \to A$, $f \circ g = f \circ h$ implies $g = h$. A is a* ***subobject*** *of B if there is a monomorphism $A \rightarrowtail B$.*

Definition 7. *In any category with a terminal object, a* ***subobject classifier*** *is an object Ω along with a morphism $\top : 1 \to \Omega$ such that for any monomorphism $m : A \rightarrowtail B$, there is a unique morphism $\chi_m : B \to \Omega$ such that the following diagram is a pullback:*

$$\begin{array}{ccc} B & \xrightarrow{\chi_m} & \Omega \\ {\scriptstyle m}\uparrow & & \uparrow{\scriptstyle \top} \\ A & \xrightarrow{!_A} & 1 \end{array}$$

Definition 8. *A* ***topos*** *is a cartesian closed category with a subobject classifier.*

The type ontology $(\mathbb{B}, \leq)$ can be seen as a category $\mathcal{B}$ by taking elements of $\mathbb{B}$ as objects, and by introducing a morphism $a \to b$ whenever $a \leq b$. We consider from now on a topos $\mathcal{C}$ containing $\mathcal{B}$ as a subcategory, and we note $\zeta : \mathcal{B} \to \mathcal{C}$ the associated faithful functor. In particular, we define the object E of entities as ζe, where e is the greatest element of $\mathcal{B}$. Besides, we require the image ζf of each morphism f in $\mathcal{B}$ to be a monomorphism. Notice then that $\mathcal{B}$ as a poset is contained in $\mathrm{Sub}(E)$, the set of subobjects of E.

For any object A of $\mathcal{C}$, we note $\mathcal{P}A$ the exponential Ω^A, which is called the *powerobject of A*. The associated *evaluation map* is $\mathrm{ev}_A : \mathcal{P}A \times A \to \Omega$. Notice that, by a well-known property of exponentials, we have the isomorphism $\mathcal{P}(A \times B) \cong \mathcal{P}A \times \mathcal{P}B$ for any objects A and B. If $f : A \times B \to \Omega$ is a morphism, we call *name* of f the morphism $\mathrm{name}(f) : A \to \mathcal{P}B$ obtained from f by the universal property of exponentials. As $1 \times B \cong B$, this definition extends up to isomorphism to any morphism $B \to \Omega$. As a consequence, a predicate in a topos has three equivalent modes of presentation: as an arrow $f : B \to \Omega$, as its name $1 \to \mathcal{P}B$, or as the subobject $A \rightarrowtail B$ obtained by pullback of $\top$ along f.

We have now enough theoretical support to properly define an interpretation of $C\Sigma$ in the topos $\mathcal{C}$. This interpretation will be denoted as a map $[\![\cdot]\!]$ which assigns to each type in $\Xi_{C\Sigma}$ an object of $\mathcal{C}$, and to each well-typed term a *global*

[8] The composition of $f : A \to B$ and $g : B \to C$ will be noted $g \circ f$ or gf whenever no ambiguity may occur. The identity on A is id_A. The product map of $f' : A \to B$ and $g' : A \to C$ will be noted $\langle f', g' \rangle : A \to B \times C$. The initial and terminal objects are noted 0 and 1, and the associated morphisms are $0_A : 0 \to A$ and $!_A : A \to 1$.

element in $\mathcal{C}$, that is, a morphism of the form $1 \to A$. For types the definition of $[\![\cdot]\!]$ is straightforward since every constructor has an obvious categorical counterpart:

$$\begin{aligned} [\![1]\!] &= 1 && \\ [\![b]\!] &= \zeta b && \text{for } b \in \mathbb{B} \\ [\![\alpha \times \beta]\!] &= [\![\alpha]\!] \times [\![\beta]\!] && \text{for } \alpha, \beta \in \Xi_{C\Sigma} \\ [\![\mathcal{P}\alpha]\!] &= \mathcal{P}[\![\alpha]\!] && \text{for } \alpha \in \Xi_{C\Sigma} \end{aligned} \tag{4}$$

Terms however need a few more assumptions to be correctly interpreted. First, even if by Definition 5 coercions are not considered as terms, they are given an interpretation by $[\![\cdot]\!]$. We will thus assume that each coercion $c : \alpha \to \beta \in \mathcal{K}$ is assigned a morphism $\kappa(c) : [\![\alpha]\!] \to [\![\beta]\!]$. Moreover, we require that each constant $u \in (\mathcal{Q}, \tau)$ has a corresponding morphism $\rho(u) : 1 \to [\![\tau(u)]\!]$, and that the induced map ρ from Q to the morphisms of $\mathcal{C}$ is injective. Then, any well-typed term $u : \alpha$ is interpreted as a morphism $1 \to [\![\alpha]\!]$ according to the following definition:

$$\begin{aligned} [\![*]\!] &= \mathrm{id}_1 : 1 \to 1 && \\ [\![u]\!] &= \rho(u) : 1 \to [\![\tau(u)]\!] && \text{for } u \in (Q, \tau) \\ [\![c]\!] &= \kappa(c) : [\![\alpha]\!] \to [\![\beta]\!] && \text{for } c : \alpha \to \beta \in \mathcal{K} \\ [\![cu]\!] &= [\![c]\!] \circ [\![u]\!] : 1 \to [\![\beta]\!] && \text{for } c : \alpha \to \beta \in \mathcal{K} \text{ and } u : \alpha \\ [\![\langle u, v \rangle]\!] &= \langle [\![u]\!], [\![v]\!] \rangle : 1 \to [\![\alpha]\!] \times [\![\beta]\!] && \text{for } u : \alpha \text{ and } v : \beta \\ [\![uv]\!] &= \mathrm{comp}_{[\![\alpha]\!],[\![\beta]\!]} \circ \langle [\![u]\!], [\![v]\!] \rangle : 1 \to \mathcal{P}[\![\beta]\!] && \text{for } u : \mathcal{P}(\alpha \times \beta) \text{ and } v : \alpha \\ [\![\pi_1 u]\!] &= \pi_1^{[\![\alpha]\!],[\![\beta]\!]} \circ [\![u]\!] : 1 \to [\![\alpha]\!] && \text{for } u : \alpha \times \beta \\ [\![\pi_2 u]\!] &= \pi_2^{[\![\alpha]\!],[\![\beta]\!]} \circ [\![u]\!] : 1 \to [\![\beta]\!] && \text{for } u : \alpha \times \beta \end{aligned} \tag{5}$$

where, for any objects A, B, $\pi_1^{A,B}$ and $\pi_2^{A,B}$ are the usual projections $A \times B \to A$ and $A \times B \to B$, and $\mathrm{comp}_{A,B}$ is defined to be the name of the composite:[9]

$$(\mathcal{P}(A \times B) \times A) \times B \xrightarrow{\sim} \mathcal{P}(A \times B) \times (A \times B) \xrightarrow{\mathrm{ev}_{A \times B}} \Omega$$

Then, we need to ensure that this interpretation respects the term equalities introduced in (3). This is straightforward for the congruence rules and for the propagation rules (EQPR) and (EQAP), and the well-foundedness of the rule (ONE) is guaranteed by the fact that the unit type is interpreted as the terminal object 1, for which id_1 is the unique morphism $1 \to 1$. Moreover, the projection rules (PRJ_1) and (PRJ_2) are easily retrieved by the definition of categorical products. The major subtlety lies in the translation of the last two rules, stating the associativity of the product and its consequence on direct application. For any terms u, v and w, the interpretations of $\langle u, \langle v, w \rangle \rangle$ and $\langle \langle u, v \rangle, w \rangle$ are only equal up to some isomorphism which belongs to the class of isomorphisms of the form $\iota_{A,B,C} : (A \times B) \times C \to A \times (B \times C)$. These isomorphisms also propagates to n-ary

[9] Notice in particular that, up to isomorphism, $\mathrm{comp}_{A,1} = \mathrm{ev}_A$. This is consistent because the isomorphism $\mathcal{P}1 \cong \Omega$ holds in any topos.

predicates through isomorphisms $\exists\iota_{A,B,C} : \mathcal{P}((A \times B) \times C) \to \mathcal{P}(A \times (B \times C))$, whose notation will become clearer at the end of this section. As a result, the term equalities may be correctly retrieved by considering the equivalence classes of the categorical interpretations for the relation on morphisms of being equal up to composition with some isomorphism of the form ι, $\exists\iota$, or combination thereof.[10]

So far, in the construction of $C\Sigma$ and its interpretation above we only discussed two of the three main ingredients of a semantic calculus as presented at the beginning of Sect. 2: predicates and rules. What about logical connectives? Obviously we may introduce them as constants as well, but since we do not use directly a proposition type nor lambda-abstractions, the usual definition of logical constants does not fit in $C\Sigma$: we would be unable to connect predicates before all their arguments are provided, whereas construction of lambda-abstracted compounds is common in natural language semantics. Furthermore, the introduction of these connectives should be carefully studied from a categorical perspective because toposes have their own internal logic: we then expect the logical constants of $C\Sigma$ to reflect this logic. Toposes are powerful enough to model higher-order logic [15], and depending on the properties of the given topos, this logic can be classical or intuitionistic [12].

The formal basis of topos logic lies on morphisms $\neg : \Omega \to \Omega$ and $\wedge, \vee, \Rightarrow$: $\Omega \times \Omega \to \Omega$. For any A, we can extend these arrows to "generalised" operators $\neg_A : \mathcal{P}A \to \mathcal{P}A$ and $\wedge_A, \vee_A, \Rightarrow_A$: $\mathcal{P}A \times \mathcal{P}A \to \mathcal{P}A$ as names of composites of the above morphisms with evaluation maps. For instance, $\wedge_A$ names the composite:

$$\begin{array}{ccc} (\mathcal{P}A \times \mathcal{P}A) \times A & & \Omega \times \Omega \xrightarrow{\quad\wedge\quad} \Omega \\ \downarrow{\scriptstyle \mathrm{id} \times \Delta_A} & & \uparrow{\scriptstyle \mathrm{ev}_A \times \mathrm{ev}_A} \\ (\mathcal{P}A \times \mathcal{P}A) \times (A \times A) & \xrightarrow{\sim} & (\mathcal{P}A \times A) \times (\mathcal{P}A \times A) \end{array}$$

where $\Delta_A : A \to A \times A$ is the diagonal map $\langle \mathrm{id}_A, \mathrm{id}_A \rangle$. Other logical connectives are obtained in the same way. In $C\Sigma$, we may think of $\neg$, $\wedge$, $\vee$ and $\Rightarrow$ as polymorphic constants of respective types $\forall\alpha.\mathcal{P}(\mathcal{P}\alpha \times \alpha)$ for the former and $\forall\alpha.\mathcal{P}(\mathcal{P}\alpha \times \mathcal{P}\alpha \times \alpha)$ for the others. They are however distinct from other constants and more generally from other terms in the sense that their interpretations are not global elements: to keep things consistent, we have for instance to posit for each type α in $\Xi_{C\Sigma}$ the interpretation $[\![\wedge^\alpha]\!] = \wedge_{[\![\alpha]\!]}$. Thus, logical connectives play a very specific role in $C\Sigma$, and may be added to the calculus as a separate set of constants.

The treatment of quantifiers is slightly more complex, as it requires additional constructions of topos theory. As pulling back along a morphism $f : A \to B$ preserves monomorphisms, it induces a pullback function $f^{-1} : \mathrm{Sub}(B) \to \mathrm{Sub}(A)$ which has both a left adjoint $\exists_f$ and a right adjoint $\forall_f$. As $\mathcal{P}A$ is the internal

[10] As suggested at the end of Sect. 2, we may extend this reasoning to the class of isomorphisms of the form $A \times 1 \to A$ to avoid the typing rule (APP′) and its counterparts in term equalities.

version of the set $\mathrm{Sub}(A)$ for any object A, those adjoints themselves induce internal morphisms $\exists f : \mathcal{P}A \to \mathcal{P}B$ and $\forall f : \mathcal{P}A \to \mathcal{P}B$ [21, §IV.9]. To produce models of the usual quantifiers, we use versions of these morphisms obtained from product projections: if $\pi : A \times B \to B$ is such a projection, $\exists\pi$ and $\forall\pi$ are morphisms $\mathcal{P}(A \times B) \to \mathcal{P}B$ which can serve as interpretations for polymorphic constants $\exists^{\alpha,\beta}$ and $\forall^{\alpha,\beta}$ of type $\mathcal{P}(\mathcal{P}(\alpha \times \beta) \times \beta)$ in $C\Sigma$. Other kinds of generalised quantifiers may also be introduced as morphisms of this family of types; their own properties would depend on their respective definitions, and they would differs from the universal and existential quantifiers only by the absence of obvious relation to the pullback function. This completes the introduction of logical connectives in the calculus.

4 Monomorphic Subtyping Discipline

Monomorphism is the categorical generalisation of the notion of injective function; in particular, both notions coincide in the category Set (see e.g. [12, §3.1]). As discussed in Sect. 1, a set-based interpretation of the subtyping relation in natural language semantics should be injective as well, for entities and for predicates. From a categorical perspective, we come naturally to think of the subtyping relation in terms of monomorphisms. The present section details how we can construct such a "monic" subtyping relation in $C\Sigma$, and how such a relation is necessarily a covariant one.

In Sect. 2, we hinted that defining the subtyping relation amounts to give constraints for building the set of coercions $\mathcal{K}$. If α and β are types in $\Xi_{C\Sigma}$, we note $\alpha \sqsubseteq \beta$ to express that α is a subtype of β, and whenever it is the case, we enforce the constraint $\alpha \to \beta \in \mathcal{K}$. We shall generate subtyping relations and categorical interpretations of the corresponding coercions simultaneously. As a starting point, recall that the construction of $C\Sigma$ is based on a type ontology $(\mathbb{B}, \leq)$ which includes the foundations of the subtyping relation. For every pair $a, b \in \mathbb{B}$, we naturally posit $a \sqsubseteq b$ whenever $a \leq b$ holds.[11] By assumptions made on the topos $\mathcal{C}$ in Sect. 3, such a pair provides also a monomorphism $\zeta a \rightarrowtail \zeta b$ which is exactly the interpretation of the related coercion $a \to b \in \mathcal{K}$. This sets up the basis of the subtyping relation, and we have now to investigate how it propagates to type constructors.

To avoid inconsistency, the unit type should only be comparable with itself, that is, $1 \sqsubseteq 1$ is the only valid subtyping relation involving 1. To see how to deal with products and predicates, we turn to their categorical interpretations. The topos $\mathcal{C}$, as a cartesian category, is equipped with a bifunctor $\times : \mathcal{C} \otimes \mathcal{C} \to \mathcal{C}$, where $\otimes$ stands for the product of categories. The following lemma is a common result whose proof is easy to retrieve:

[11] Actually, we may remove superfluous coercions by limiting $\mathcal{K}$ to the strict part of the subtyping relation, that is, $\alpha \to \beta$ in $\mathcal{K}$ only if $\alpha \sqsubset \beta$ holds. Indeed, the coercion corresponding to $\alpha \to \alpha$ is the identity map $\mathrm{id}_{[\![\alpha]\!]}$, which brings no useful additional information.

Lemma 1. *The bifunctor $\times$ preserves monomorphisms, that is, if $f : A \rightarrowtail C$ and $g : B \rightarrowtail D$ are monos, then so is $f \times g : A \times B \rightarrowtail C \times D$.*

A similar result can be established for powerobjects through a less common view than usual. Indeed, the operator $\mathcal{P}$ is generally extended to a contravariant functor $\mathcal{C}^{\mathrm{op}} \to \mathcal{C}$ by taking $\mathcal{P}f$ to be the internal counterpart of the map f^{-1} introduced in Sect. 3, but this definition does not satisfy our requirements. We shall use instead the *covariant powerobject functor* $\mathcal{P}^+ : \mathcal{C} \to \mathcal{C}$ defined on any object A by $\mathcal{P}^+A = \mathcal{P}A$ and on any morphism f by $\mathcal{P}^+f = \exists f$. This definition is correct since $\exists(gf) = \exists g \circ \exists f$ for any morphisms f, g, and $\exists\,\mathrm{id}_A = \mathrm{id}_{\mathcal{P}A}$ for any object A [13, §A2.3]. The following lemma appears in [13, Cor. A2.2.5] and [21, Cor. 3, §IV.3]:

Lemma 2. *If $m : A \rightarrowtail B$ is mono, then $\mathcal{P}m \circ \exists m = \mathrm{id}_{\mathcal{P}A}$, i.e. $\exists m$ is split mono.*

A weaker way to put it is to say that $\mathcal{P}^+$ preserves monomorphisms. Nevertheless we shall see below that the existence of the retraction $\mathcal{P}m$ will be useful for a special extension of $C\Sigma$. Altogether, the lemmata above provide the keys to complete the definition of the subtyping relation for all the type constructors. Hence we assert the following subtyping rules:

$$\frac{a, b \in \mathbb{B} \quad a \leq b}{a \sqsubseteq b}\,(\textsc{base}) \qquad \frac{\alpha \sqsubseteq \gamma \quad \beta \sqsubseteq \delta}{\alpha \times \beta \sqsubseteq \gamma \times \delta}\,(\textsc{prod}) \qquad \frac{\alpha \sqsubseteq \beta}{\mathcal{P}\alpha \sqsubseteq \mathcal{P}\beta}\,(\textsc{pred}) \quad (6)$$

The coercion set $\mathcal{K}$ must be built consequently. The map κ introduced in Sect. 3, which sends each coercion in $\mathcal{K}$ to a corresponding morphism in $\mathcal{C}$, can also be properly defined using similar rules: if c, c' are coercions, the corresponding product coercion introduced by the rule (PROD), noted $c \times c'$, shall be interpreted by $\kappa(c \times c') = \kappa(c) \times \kappa(c')$; and similarly, if c is a coercion, the predicate coercion introduced by the rule (PRED), noted $\mathcal{P}c$, shall be interpreted by $\kappa(\mathcal{P}c) = \exists\kappa(c)$. Recall then that we put $[\![c]\!] = \kappa(c)$ for a coercion c. Lemmata 1 and 2, as well as the construction of $\mathcal{C}$ above the type ontology, can be used to prove inductively the following result, which puts monomorphisms at the heart of the subtyping interpretation.

Proposition 1. *For any coercion $c \in \mathcal{K}$, if c has been constructed using the rules in* (6)*, then $[\![c]\!]$ is a monomorphism.*

As promised in Sect. 2, we now turn to the second batch of typing rules for $C\Sigma$, which shall constrain the use of coercions. Our main concern when defining these rules is to provide a strong discipline on coercion uses in order to preserve *type safety*, even if we take this notion in a weaker understanding compared to its definition for programming languages. Our objective is to prevent overgeneration in $C\Sigma$ by constraining the use of coercions: for instance, we may forbid the application of coercions both to a predicate and its argument at the same time, because it would allow any predicate to take any argument. Moreover, we may need to prevent n-ary predicates with argument-places related in types to accept arguments coerced from different types if the underlying semantics intends to link them, as it could lead to unwanted semantic interpretations.

Assume we add a base type $\square$ to $C\Sigma$, and define a *type context* to be a type in $\Xi^{\square}_{C\Sigma}$, that is, a type with at least one occurrence of $\square$. If d is such a context, define for any base type $a \neq \square$ the type $d[a]$ to be d where all occurrences of $\square$ are replaced by a. We may generally think of a type $d[a]$ for a term to represent a product of types in $\Xi_{C\Sigma}$, some of them involving the type a. We propose the following formulation:

Definition 9. *A $C\Sigma$ calculus is **type safe** if:*

1. *for all pairs a, b of incompatible base types, type context d, term $u : \mathcal{P}d[a]$ and constant $v : d[b]$, and coercions c and c' of respective domains $\mathcal{P}d[a]$ and $d[b]$, no combinations of typing rules make $(cu)(c'v)$ a well-typed term ;*
2. *for all type a, pair of incompatibles types b, c, type contexts d, d', d'' and γ such that $d = d' \times d'' \times \gamma$, term $u : \mathcal{P}d[a]$, constants $v : d'[b]$ and $w : d''[c]$, and coercions $c' : d'[b] \to d'[a]$ and $c'' : d''[c] \to d''[a]$, no combinations of typing rules make $u((c' \times c'')\langle v, w\rangle)$ a well-typed term.*

This formulation is intended to translate into formal conditions the discussion of the previous paragraph: the first condition forbids to coerce predicates and arguments at the same time, and the second condition forbids the simultaneous coercion of two incompatible types to the same type between arguments of a given predicate. Recall that, by the equality rule (AP*) and the definition of the product of coercion, the second condition also applies to terms of the form $u(c'v)(c''w)$. In practice, the first condition states for instance that the following rule of *free coerced application* (FCA) is *not* acceptable for type safety:

$$\text{unsafe!} \qquad \frac{\vdash u : \mathcal{P}\alpha \qquad \vdash v : \beta \qquad c : \alpha \to \gamma \in \mathcal{K} \qquad c' : \beta \to \gamma \in \mathcal{K}}{\vdash ((\mathcal{P}c)u)\,(c'v) : \mathcal{P}1}\,(\text{FCA}) \tag{7}$$

In a similar vein, the second condition states for instance that the rule of *partial coerced application* (PCA) defined below is also *not* type safe:

$$\text{unsafe!} \qquad \frac{\vdash u : \mathcal{P}(\beta \times \gamma) \qquad \vdash v : \alpha \qquad c : \alpha \to \beta \in \mathcal{K}}{\vdash u\,(cv) : \mathcal{P}\gamma}\,(\text{PCA}) \tag{8}$$

Indeed, consider base types h, v and p standing for humans, vehicles and physical entities respectively, with $h \leq p$, $v \leq p$ and h, v incompatible, and assume constant predicates $\mathbf{j} : h$, $\mathbf{car} : \mathcal{P}v$ and $\mathbf{heavy} : \mathcal{P}(\mathcal{P}p \times p)$. By construction, we have coercions $c : h \to p$ and $c' : \mathcal{P}v \to \mathcal{P}p$. Now, by two successive applications of the rule (PCA), the term $\mathbf{heavy}(c'\,\mathbf{car})(c\,\mathbf{j})$ is well-typed, which explicitly breaks the second condition, hence the unsafeness of (PCA).[12] This term could be the semantics of a category-mistaken sentence such as "?John is a heavy car". Of course, we might want to have semantic representation of such sentences in our calculus, but it should not be obtained by means of the subtyping relation.

To preserve type safety, we need a weaker version of the rule (PCA), which will be called *restricted total coerced application* (RTCA). The idea behind this

[12] The same reason explains why we did not introduce a typing rule to build directly coerced terms of the form cu in Sect. 2.

rule is to force all the arguments of a given predicate to be gathered before application and to restrict coercion uses so that the arguments filling slots with the same expected base type are themselves of the same base type, thus avoiding simultaneous application of a predicate to terms such as **car** and **j**.

$$\frac{\vdash u : \mathcal{P}d[b] \qquad \vdash v : d[a] \qquad c : d[a] \to d[b] \in \mathcal{K}}{\vdash u\,(cv) : \mathcal{P}1}\,(\textsc{rtca}) \tag{9}$$

It follows directly from Definition 9 that (RTCA) is type safe. It can even be safely extended for simultaneous subtyping of several base types: generalise the notion of type context for finite number of holes $\square_1, \ldots, \square_n$, the corresponding extended rule is defined for $\vdash u : \mathcal{P}d[b_1, \ldots, b_n]$ and $\vdash v : d[a_1, \ldots, a_n]$ with the additional condition that all b_i must be pairwise distinct. This extension is useful when dealing with n-ary predicates, for instance transitive verbs.

We acknowledge however that (RTCA), even in its extended form, may be too restrictive for semantic uses, in particular in a calculus like $C\Sigma$ where no variables and λ-abstractions are available. What if we need to perform the partial application of **heavy** on **car** as the semantic representation of the phrase "a heavy car" requires? A solution to retrieve the flexibility of partial coercion application is to exploit the retraction of the corresponding predicate coercion. For each predicate coercion $c : \mathcal{P}\alpha \to \mathcal{P}\beta$ in $\mathcal{K}$, define its *reverse coercion* $\bar{c} : \mathcal{P}\beta \to \mathcal{P}\alpha$ with $[\![\bar{c}]\!] = \mathcal{P}m$, where m is the mono $[\![\alpha]\!] \to [\![\beta]\!]$ such that $[\![c]\!] = \exists m$. Instead of using coercions to embed the type of the argument term into the type expected by the predicate, reverse coercions enable us to specialise the type of the predicate to the type of its argument. Once again, predicate specialisation must be performed on all occurrences of a given base type to prevent unsafe applications as examplified with (PCA). However, this global specialisation makes subsequent partial applications possible. If we extend our set of coercions with reversed ones, the *specialised partial coerced application* rule (SPCA) is given by:

$$\frac{\vdash u : \mathcal{P}(d[b] \times d'[b]) \qquad \vdash v : d[a] \qquad c : a \to b \in \mathcal{K}}{\vdash (\bar{c}'\,u)\,v : \mathcal{P}d'[a]}\,(\textsc{spca}) \tag{10}$$

where $\bar{c}'$ is the reverse of the coercion $c' : \mathcal{P}(d[a] \times d'[a]) \to \mathcal{P}(d[b] \times d'[b])$ built upon c. This rule is again type safe since it guarantees that after applying it, all subsequent applications will be done with base types which are subtypes of a.[13]

We insist on the fact that (SPCA), while ressembling a contravariant subtyping rule—justified by the fact that the retraction $\mathcal{P}m$ of a mono comes indeed from a contravariant functor—, is actually a direct consequence of the covariant property of our subtyping relation. As highlighted in Sect. 1, a contravariant subtyping

[13] As an anonymous reviewer pointed out, the rule (SPCA) is similar in spirit to a typing rule for *bounded polymorphism*, using a predicate of type $\forall b.\mathcal{P}(d[b] \times d'[b])$. The connections between this kind of polymorphism and covariant subtyping may be even deeper and may be worth investigating, but are beyond the scope of this paper. Incidentally, the author also took part in the developpment of another framework using bounded polymorphism and record types [3].

is unable to compare the types $(a \to t) \to a \to t$ and $(b \to t) \to b \to t$, even if a and b are comparable base types, while the covariant subtyping is able to perform the corresponding comparison between $\mathcal{P}(\mathcal{P}a \times a)$ and $\mathcal{P}(\mathcal{P}b \times b)$, and provides coercions in both directions depending on the situation. Notice however that in this case the interpretation of the corresponding reverse coercion is not a monomorphism, but has the dual property of *epimorphism*, as another consequence of Lemma 2.[14]

The last remark to make on the previous typing rules is to observe that (RTCA) is actually a special case of (SPCA) with $d'[a] = d'[b] = 1$.[15] To understand why, consider the coercion $c : d[a] \to d[b]$. Using (SPCA), from $u : \mathcal{P}d[b]$ and $v : d[a]$ we can construct the term $(\bar{c}'u)\,v$ with $[\![\bar{c}']\!] = \mathcal{P}[\![c]\!]$. In the categorical model, the following diagram commutes by general property of the contravariant powerobject functor [21, §IV.1]:

$$\begin{array}{ccccc} 1 & \xrightarrow{\langle [\![u]\!], [\![v]\!] \rangle} & \mathcal{P}[\![d[b]]\!] \times [\![d[a]]\!] & \xrightarrow{\mathcal{P}[\![c]\!] \times \mathrm{id}} & \mathcal{P}[\![d[a]]\!] \times [\![d[a]]\!] \\ & & \downarrow{\scriptstyle \mathrm{id} \times [\![c]\!]} & & \downarrow{\scriptstyle \mathrm{ev}_{[\![d[a]]\!]}} \\ & & \mathcal{P}[\![d[b]]\!] \times [\![d[b]]\!] & \xrightarrow{\mathrm{ev}_{[\![d[b]]\!]}} & \Omega \end{array}$$

In other words, $[\![(\bar{c}'u)v]\!] = [\![u(cv)]\!]$. This semantic result licenses the introduction on the syntactic side of the following coerced equational rule:

$$\frac{\vdash u : \mathcal{P}d[b] \qquad \vdash v : d[a] \qquad c : d[a] \to d[b]}{\vdash (\bar{c}'u)v = u(cv) : \mathcal{P}1}\,(\text{CEQ}) \tag{11}$$

It is then clear that with (CEQ), the rule (SPCA) entails the rule (RTCA). We can even extend the rule (CEQ) to a more general form for partial application, provided that we take care of all the details in order to preserve type safety. The resulting rule, call it (PCEQ), is given by:

$$\frac{\vdash u : \mathcal{P}(d[b] \times d'[b]) \qquad \vdash v : d[a] \qquad c : a \to b}{\vdash (\bar{c}'u)v = \bar{c}''(u(c'''v)) : \mathcal{P}d'[a]}\,(\text{PCEQ}) \tag{12}$$

where c' is as before, and c'' and c''' are the coercions $\mathcal{P}d'[a] \to \mathcal{P}d'[b]$ and $d[a] \to d[b]$ built upon c. It captures the previous rule when $d'[a] = d'[b] = 1$, by noticing that c'' and its reverse are then the coercion $\mathcal{P}1 \to \mathcal{P}1$, an identity which can be safely removed from the term. Moreover, the right-hand side of the equality does not fall under the forbidden terms of Definition 9. To complete the construction of $C\Sigma$, we shall add the rules (SPCA) and (PCEQ) to the first batch of rules introduced in Sect. 2.

[14] For this reason and despite the name of "coercion", reverse coercions must not be part of $\mathcal{K}$. Their use is only licensed by explicit involvement of the bar notation in the typing rules.

[15] This fact is more precisely verified under the assumption of interpreting terms up to the isomorphism $A \times 1 \cong A$, cf. footnote 10.

5 Understanding Predicate Covariance

The aim of this section is to provide further intuition on covariance for predicates, as the formalisms of the previous sections have been kept rather abstract. A few more results on the interactions between predicates and logical connectives will also be stated. But for now, let us start with this simple question: if $u : \mathcal{P}\alpha$ is a predicate constant and $c : \mathcal{P}\alpha \to \mathcal{P}\beta$ is a coercion, what does the predicate $cu : \mathcal{P}\beta$ actually describe? To answer it, we turn once again to the categorical model of $C\Sigma$. If c' is the coercion for $\alpha \to \beta$, we have $[\![cu]\!] = \exists [\![c']\!] \circ [\![u]\!]$ by (5) and (6). Thus, we need to explain how the covariant powerobject functor $\mathcal{P}^+$ works in $\mathcal{C}$.

Let $m : A \rightarrowtail B$ be a mono in $\mathcal{C}$. As the powerobject $\mathcal{P}A$ is an internal representation of the set $\mathrm{Sub}(A)$, it will be sometimes convenient to study the external function $\exists_m : \mathrm{Sub}(A) \to \mathrm{Sub}(B)$ instead of $\exists m$ in order to derive internal properties as an application of the Yoneda lemma (see [21, §IV.9] for details). To give even more intuition, suppose in this paragraph only that A and B are sets. Then $\mathrm{Sub}(A)$ and $\mathrm{Sub}(B)$ are their respective powersets, and if $U \in \mathrm{Sub}(A)$, then $\exists_m(U) = \{m(x) \mid x \in U\}$. Besides, if we further assume that the injective function m actually stands for the inclusion $A \subseteq B$, then $\exists_m(U)$ is U itself viewed as a subset of B. The category Set has $\{0, 1\}$ as subobject classifier, and the characteristic χ_U^A of U in A is the classical one, with $\chi_U^A(x) = 1$ if $x \in U$, and $\chi_U^A(x) = 0$ otherwise. As there is a one-to-one correspondence between subsets of A and characteristic functions on A, $\exists_m$ induces a map sending each function χ_U^A to χ_U^B, and it is clear that for all $x \in A$, $\chi_U^A(x) = \chi_U^B(x)$.

Now, abstracting over these set-theoretic considerations, we can generalise some observed properties to any topos $\mathcal{C}$. For any object X, define the morphism $\mathrm{true}_X : X \to \Omega$ as the composite $\top \circ !_X$. By definition of the subobject classifier, any subobject $k : U \rightarrowtail A$ is classified by some $f : A \to \Omega$, such that $fk = \mathrm{true}_U$. The following result appears in [21, Prop. 1, §IV.3] with slight differences in notation:

Proposition 2. *Let $g : B \to \Omega$ be the map such that* $\mathrm{name}(g) = \exists m \circ \mathrm{name}(f)$. *Then, g classifies the subobject $mk : U \rightarrowtail B$.*

In other words, mk is the pullback of $\top$ along g, which also means that g shares the same conditions as f to be evaluated to true. In $C\Sigma$, this means that, being given terms $u : \mathcal{P}\alpha$, $v : \beta$ and the coercion $c : \alpha \to \beta$, the term $((\mathcal{P}c)u)v : \mathcal{P}1$, as a logical formula, is true if and on only if there is a term $w : \alpha$ such that $v = cw$ and uw is true.[16] It is legitimate to ask whether the same result holds for conditions of falsity. We shall investigate this question from an external point of view, and study several results related to logical connectives in the way to answer it.

[16] This property, grounded in topos theory, ignores the requirements of type safety as given in Definition 9, according to which the term $((\mathcal{P}c)u)v$ cannot be typed. It shows nonetheless that terms of the form given in the first conditions are not necessary from a truth-theoretical point of view, since it brings "true" application of a coerced form of a predicate u back to a direct application of u.

Recall that for any X, $\langle \mathrm{Sub}(X), \leq, 0, X, \cup, \cap, \Rightarrow \rangle$ is a Heyting algebra (see e.g. [12, §8.3]), where $\cup$, $\cap$ and $\Rightarrow$ are external counterparts to $\wedge_X$, $\vee_X$ and $\Rightarrow_X$ introduced in Sect. 3. If $U \in \mathrm{Sub}(X)$, write $\overline{U}$ for the pseudo-complement $U \Rightarrow 0$. Being given any mono $m : A \rightarrowtail B$, we would like to know whether these logical morphisms, as well as the quantifiers $\exists_{A,X}, \forall_{A,X} : \mathcal{P}(A \times X) \to \mathcal{P}X$, are preserved by the predicate subtyping $\exists m$. The following proposition states a stronger property for some of these connectives: they are actually natural transformations from $\mathcal{P}^+$ to itself.

Proposition 3. *The transformations $\wedge_A$, $\vee_A$ are natural in A, and $\exists_{A,X}$ is natural in A and X.*

Due to the lack of space, we omit proofs of this proposition and of the following results below. The naturality of conjunction, disjunction and existential quantifier amounts to say that $C\Sigma$ can be completed with new equational rules describing the good interaction of these connectives with subtyping coercions. For the first two, the equational rules are the following:

$$\begin{array}{c} \dfrac{\vdash u : \mathcal{P}\alpha \qquad \vdash v : \mathcal{P}\alpha \qquad c : \mathcal{P}\alpha \to \mathcal{P}\beta \in \mathcal{K}}{\vdash \wedge^{\beta}\,(c \times c)\langle u, v\rangle = c\,(\wedge^{\alpha}\langle u, v\rangle) : \mathcal{P}\beta}\,(\wedge\text{-EQ}) \\[2ex] \dfrac{\vdash u : \mathcal{P}\alpha \qquad \vdash v : \mathcal{P}\alpha \qquad c : \mathcal{P}\alpha \to \mathcal{P}\beta \in \mathcal{K}}{\vdash \vee^{\beta}\,(c \times c)\langle u, v\rangle = c\,(\vee^{\alpha}\langle u, v\rangle) : \mathcal{P}\beta}\,(\vee\text{-EQ}) \end{array} \tag{13}$$

As for the existential quantifier, there are two corresponding rules to account for the double naturality:

$$\begin{array}{c} \dfrac{\vdash u : \mathcal{P}(\alpha \times \gamma) \qquad c' : \mathcal{P}(\alpha \times \gamma) \to (\beta \times \gamma) \in \mathcal{K}}{\vdash \exists^{\beta,\gamma}(c'\,u) = \exists^{\alpha,\gamma} u : \mathcal{P}\gamma}\,(\exists\text{-EQ}_1) \\[2ex] \dfrac{\vdash u : \mathcal{P}(\alpha \times \gamma) \qquad c : \mathcal{P}\gamma \to \mathcal{P}\delta \qquad c' : \mathcal{P}(\alpha \times \gamma) \to (\alpha \times \delta)}{\vdash \exists^{\alpha,\delta}(c'\,u) = c\,(\exists^{\alpha,\gamma} u) : \mathcal{P}\delta}\,(\exists\text{-EQ}_2) \end{array} \tag{14}$$

In terms of truth-theoretical interpretation, these rules mean that coercions interacts well with the truth conditions of conjunctions, disjunctions and existential quantifiers applied to predicates: in each of the equalities above, the left-hand and right-hand sides have the same conditions of truth and falsity.[17]

However, the naturality of logical connectives w.r.t. $\mathcal{P}^+$ does not propagate to the relative pseudo-complement, nor to the universal quantifier. There is a strict entailment between logical formulae using these connectives through the subtyping relation, under the sufficient condition that the codomain of the subtyping coercion contains the disjoint union of the domain with a *non zero* object, where X non zero means $X \ncong 0$. Within the type ontologies used for formal semantics, this condition generally holds: for instance, the types p, h and v from the example in Sect. 4 are such that $p \geq h \vee v$. The key properties are stated below:

[17] Notice that this property applies regardless of the ambient logic, be it classical or intuitionistic.

Proposition 4. *Let $f : A \to B$ be any morphism. If $\overline{\exists_f(A)}$ is non zero, then for all $U, V \in \mathrm{Sub}(A)$ we have the strict inclusion $\exists_f(U \Rightarrow V) < \exists_f(U) \Rightarrow \exists_f(V)$.*

Corollary 1. *If $\overline{\exists_f(A)}$ is non zero, $\exists_f(\overline{U}) < \overline{\exists_f(U)}$.*

Proposition 5. *Let $m : A \rightarrowtail B$ be a monomorphism and X any object, and suppose $m' = m \times \mathrm{id}_X$. If $\overline{\exists_m(A)}$ is non zero, then for all $U \in \mathrm{Sub}(A \times X)$ we have the strict inclusion $\forall_{\pi_{B,X}}(\exists_{m'}(U)) < \forall_{\pi_{A,X}}(U)$.*

These propositions show in particular that neither the equational rules in (13) nor ($\exists$-EQ$_1$) have counterparts for implication, negation, and universal quantifier on its first parameter. In term of truth conditions, Prop. 4 shows for instance that there are arguments on which the predicate $\Rightarrow^\beta (c \times c)\langle u, v\rangle : \mathcal{P}\beta$ can be proved true, whereas $c\,(\Rightarrow^\alpha \langle u, v\rangle) : \mathcal{P}\beta$ will be proved false, the arguments in question being those on which $\Rightarrow^\alpha \langle u, v\rangle$ fails to have a truth value due to type mismatch—and similar results hold for negation and universal quantifier. However, we have for universal quantifiers the following proposition which is weaker than naturality on the second parameter, but suffices for interaction with coercions:

Proposition 6. *Let $m : X \rightarrowtail Y$ be a monomorphism and A any object. Then, $\forall_{\pi_{A,Y}} \exists_{(\mathrm{id}_A \times m)} = \exists_m \forall_{\pi_{A,X}}$.*

The consequence of the latter result is the following counterpart to ($\exists$-EQ$_2$):

$$\frac{\vdash u : \mathcal{P}(\alpha \times \gamma) \qquad c : \mathcal{P}\gamma \to \mathcal{P}\delta \qquad c' : \mathcal{P}(\alpha \times \gamma) \to (\alpha \times \delta)}{\vdash \forall^{\alpha,\delta}(c'\,u) = c\,(\forall^{\alpha,\gamma} u) : \mathcal{P}\delta}\,(\forall\text{-EQ}) \tag{15}$$

All the previous properties are expected to provide a better understanding of the covariant subtyping. Prop. 2 showed in particular that if $u : \mathcal{P}a$ is a first-order predicate and $c : \mathcal{P}a \to \mathcal{P}b$ a predicate coercion, then $c\,u$ is a predicate which is true on the same entities as u. By Cor. 1, we can affirm that if u is false on some entity, then $c\,u$ is also false on that very entity. However, $c\,u$ can be false on an entity without it even being in the span of u at all, that is, even if it is not of type α. In terms of a lower-level lambda-calculus, $c\,u$ can be expressed by $\lambda x{:}b.\, \exists y{:}a.\, (c'(y) = x) \wedge u(y)$, where $c' : a \to b$ is the coercion that underlies c, that is, $[\![c]\!] = \exists [\![c']\!]$. Thus, the behaviour of $c\,u$ is simply explained: if x is an entity of type β, then either x is not in the image of c', in which case $c\,u\,x$ is false, or there is an antecedent y of x through c' and $c\,u\,x = u\,y$.

It turns out that the complex coercion c can be itself interpreted as an operator $\lambda u.\lambda x.\exists y.\, (c'(y) = x) \wedge u(y)$, which resembles the functor used by Asher in [1, §6.1] to transform a first-order predicate on a dot type to another one whose argument type is an aspect of the initial one. Even if dot type projections are not exactly subtyping relations as intended in this paper—neither it is for Asher, incidentally—, this similitude should not be surprising since both operators carry the same idea of a covariant transformation of predicates. Moreover, the reverse coercion $\bar{c}$ coincides with the usual contravariant subtyping for first-order predicates as the operator $\lambda u \lambda x.u(c'x)$, and as c' is intended to be mono, the

composition $\lambda u.\bar{c}(cu)$ amounts to the identity up to isomorphism, which is also a consequence of Lemma 2. Thus, a last rule of term equality can be added:

$$\frac{\vdash u : \mathcal{P}\alpha \qquad c : \mathcal{P}\alpha \to \mathcal{P}\beta}{\vdash u = \bar{c}(c\,u) : \mathcal{P}\alpha}\,(\textsc{retract}) \tag{16}$$

However, for $v : \mathcal{P}\beta$, we have $v \neq c(\bar{c}\,v)$ in general.

6 Conclusion and Future Works

We introduced $C\Sigma$, a general semantic predicate calculus using a constructor $\mathcal{P}$ instead of the traditional functions of codomain t, and completed by a covariant subtyping. The typing rules constraining the use of subtyping coercions ensure a property of type safety which is sufficient for semantic purposes. The general typing rules of $C\Sigma$ are obtained by gathering the rules in (2) and (10), and the term equality rules consist in those given in (3), to which we add the equalities given in (12–16). The typing rule in (9) as well as the term equality rule in (11) are also admissible. Altogether, these rules enable $C\Sigma$ to be as efficient as other proposals for natural language semantics. However, $C\Sigma$ gains in flexibility by its covariant approach of subtyping, which enables us to deal easily with second-order types that pose difficulties to other semantic frameworks.

In the previous section, we generalised reasoning in Set to any topos in such a way that the interpretation of $C\Sigma$ is not restricted to sets—even if toposes have a general "set-like" behaviour. It may be relevant then to ask whether interpreting $C\Sigma$ directly in Set would not have sufficed for our purposes, instead of carrying on with the generalisation we presented. There are actually two reasons for such a move. Firstly, in spite of its specific status amongst toposes (and in mathematics in general), Set has a few restrictive properties which we may want to dismiss, the most important one being the fact that, as a Boolean topos, the internal logic of Set is always classical: we may want to use another topos with intuitionistic logic, for instance. Secondly, some semantic phenomena, such as vagueness [6], suggests that predicate modelling might need to go beyond the set-theoretical basis: if this happens to be the case, we believe that the generalisation we presented offers more flexibility to help accounting for any new proposal.

We ought to add that the genericity of topos theory, as compared to set theory, make the choice of a canonical covariant subtyping coercion more difficult. We may however think of such a canonical coercion as similar to an inclusion in Set, in the sense that a coerced entity is roughly the same entity viewed in another perspective. It may be hard however to systematise this idea in a general topos, unless we introduce an entity constant by a global element for each supertype the entity has, and then choose coercions which preserve the resulting families of global elements. Nevertheless, this issue may be less harmful than it seems, to the extent that covariant coercions ensure at least that two distinct entities remain distinct when coerced, thus preserving the relations of entities w.r.t. each other.

Finally, we may extend the calculus with additional non-subtyping coercions to improve its abilities, in the spirit of the reverse coercions introduced in Sect. 4. The projection maps from product terms π_1 and π_2 have been introduced here as term constructors, but could actually be added as such coercions, provided that we enforce a strong distinction between subtyping coercions and other ones when defining the rules of our calculus. Other possible coercions—whose introduction did unfortunately not fit in those pages although studied by the author—are what we could call *transstructural coercions*, that is, coercions that do not preserve the structure of type constructors. This case includes the introduction of *dot types* [26] as subtypes of products (as proposed in [1, 2]), and other transformations such as *type shifts* [24]. Finally, creative uses of language and other transfers of meaning may be handled by more general coercions. Overall, future studies on $C\Sigma$-like frameworks will be devoted to explore these potential extensions of the calculus.

Acknowledgments. The author is grateful to the three anonymous reviewers for their helpful and valuable comments.

Bibliography

[1] Asher, N.: Lexical Meaning in Context: A Web of Words. Cambridge University Press (2011)

[2] Babonnaud, W.: A topos-based approach to building language ontologies. In: Bernardi, R., Kobele, G.M., Pogodalla, S. (eds.) Formal Grammar. 24th International Conference, FG 2019, Riga, Latvia, August 11, 2019, Proceedings. pp. 18–34. Springer, Berlin (2019)

[3] Babonnaud, W., de Groote, P.: Lexical selection, coercion, and record types. In: LENLS17: Logic & Engineering of Natural Language Semantics, Online, November 15-17, 2020 (2020)

[4] Barendregt, H.P.: The Lambda-Calculus: Its Syntax and Semantics. Elsevier (1984)

[5] Berry, G.: Stable models of typed λ-calculi. In: Ausiello, G., Böhm, C. (eds.) Automata, Languages and Programming. pp. 72–89. Springer (1978)

[6] Burnett, H., Sutton, P.: Vagueness and natural language semantics. In: Gutzmann, D., Matthewson, L., Meier, C., Rullmann, H., Zimmermann, T.E. (eds.) The Wiley Blackwell Companion to Semnatics. John Wiley & Sons (2020)

[7] Cardelli, L.: A semantics of multiple inheritance. In: Kahn, G., MacQueen, D.B., Plotkin, G. (eds.) Semantics of Data Types. pp. 51–67. Springer, Berlin (1984)

[8] Cardelli, L.: Structural subtyping and the notion of power type. In: Ferrante, J., Mager, P. (eds.) POPL '88: Proceedings of the 15th ACM SIGPLAN-SIGACT symposium on Principles of programming languages. pp. 70–79. Association for Computing Machinery, New York (1988)

[9] Castagna, G.: Covariance and contravariance: Conflict without a cause. ACM Transactions on Programming Languages and Systems **17**(3), 431–447 (1995)

[10] Chatzikyriakidis, S., Luo, Z.: On the interpretation of common nouns: Types versus predicates. In: Chatzikyriakidis, S., Luo, Z. (eds.) Modern Perspectives in Type-Theoretical Semantics, Studies in Linguistics and Philosophy, vol. 98, pp. 43–70. Springer (2017)

[11] Cook, W.R., Hill, W., Canning, P.S.: Inheritance is not subtyping. In: Allen, F.E. (ed.) POPL '90: Proceedings of the 17th ACM SIGPLAN-SIGACT symposium on Principles of programming languages. pp. 125–135. Association for Computing Machinery, New York (1990)

[12] Goldblatt, R.: Topoi: The Categorial Analysis of Logic, Studies in logic and the foundations of mathematics, vol. 98. North-Holland Publishings (1979)

[13] Johnstone, P.T.: Sketches of an Elephant: A Topos Theory Compendium. Oxford University Press (2002)

[14] Lambek, J.: From λ-calculus to cartesian closed categories. In: Hindley, J.R., Seldin, J.P. (eds.) To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism, pp. 375–402. Academic Press, London (1980)

[15] Lambek, J., Scott, P.J.: Introduction to Higher Order Categorical Logic. Cambridge University Press (1986)
[16] Liskov, B.H., Wing, J.M.: A behavioral notion of subtyping. ACM Transactions on Programming Languages and Systems **16**(6), 1811–1841 (1994)
[17] Luo, Z.: Coercive subtyping. Journal of Logic and Computation **9**(1), 105–130 (1999)
[18] Luo, Z.: Type-theoretical semantics with coercive subtyping. In: Li, N., Lutz, D. (eds.) Proceedings of SALT 20. pp. 38–56 (2010)
[19] Luo, Z., Soloviev, S., Xue, T.: Coercive subtyping: Theory and implementation. Information and Computation **223**, 18–42 (2013)
[20] MacLane, S.: Categories for the Working Mathematician. Springer (1971)
[21] MacLane, S., Moerdijk, I.: Sheaves in Geometry and Logic: A First Introduction to Topos Theory. Springer, New York (1992), corr. 2nd edition 1994
[22] Milner, R.: A theory of type polymorphism in programming. Journal of Computer and System Sciences **17**(3), 348–375 (1978)
[23] Montague, R.: The proper treatment of quantification in ordinary english. In: Suppes, P., Moravcsik, J., Hintikka, J. (eds.) Approaches to Natural Language, pp. 221–242. Reidel, Dordrecht (1973)
[24] Partee, B.H.: Noun phrase interpretation and type-shifting principles. In: Groenendijk, J., de Jongh, D., Stokhof, M. (eds.) Studies in discourse representation theory and the theory of generalized quantifiers, pp. 115–143. De Gruyter (1986)
[25] Pierce, B.C.: Basic Category Theory for Computer Scientists. The MIT Press, Cambridge (1991)
[26] Pustejovsky, J.: The Generative Lexicon. MIT Press, Cambridge (1995)
[27] Retoré, C.: The montagovian generative lexicon ΛTy_n: a type theoretical framework for natural language semantics. In: Matthes, R., Schubert, A. (eds.) Proceedings of the 19$^{\text{th}}$ International Conference on Types for Proofs and Programs. LIPICS, vol. 26, pp. 202–229 (2014)
[28] Reynolds, J.C.: Using category theory to design implicit conversions and generic operators. In: Jones, N.D. (ed.) Semantics-Directed Compiler Generation. pp. 211–258. Springer (1980)
[29] Saba, W.S.: Language and its commonsense: Where formal semantics went wrong, and where it can (and should) go. Journal of Knowledge Structures and Systems **1**(1), 40–62 (2020)
[30] Seely, R.A.G.: Locally cartesian closed categories and type theory. Mathematical Proceedings of the Cambridge Philosophical Society **95**(1), 33–48 (1984)
[31] Smyth, M.B., Plotkin, G.: The category-theoretic solution of recursive domain equations. SIAM Journal on Computing **11**(4), 761–783 (1982)
[32] Sommers, F.: The ordinary language tree. Mind **68**(2), 160–185 (1959)
[33] Sommers, F.: Type and ontology. The Philosophical Review **72**(3), 327–363 (1963)

Replacing Implications with Negation in Non-associative Lambek Calculus

Arno Bastenhof

Abstract. We present a variation $\mathbf{NL}_{\otimes\neg}$ on the non-associative Lambek calculus that replaces directional implications with linear negation. Application-wise, we argue for its suitability to both the analysis of natural language syntax (bar displacement) ánd non-local quantifier scope. This is to be compared with the incompatibility observed between said goals within van Benthem's substructural hierarchy ([39]), which previously was resolved, e.g., by weakening compositionality to a relation ([22]) or through controlled use of structural rules ([32]). As a second application we consider the use of CPS translations for defining the derivational semantics of classical type-logical grammars. Previous work in this area (e.g., [7]) gave expression to such translations using the proof terms of minimal logic or of the commutative Lambek calculus, raising the question as to whether associativity and commutativity are required of the target language even if absent in the source. We here show this not to be the case by providing a double negation translation for classical non-associative Lambek calculus ([15]) with $\mathbf{NL}_{\otimes\neg}$ as the target. Besides applications, we present the proof theory of $\mathbf{NL}_{\otimes\neg}$ by exhibiting an axiomatic formulation as well as natural deduction and a sequent calculus. Finally, we provide a constructive model-theoretic completeness proof for Cut-free derivations and discuss the algorithm extracted therefrom through its formalization in Coq.

Keywords: Lambek calculus, linear negation, scopal ambiguities, continuation-passing style (CPS), Coq

1 Introduction

Lambek's syntactic calculus ([28, 29]) offers a deductive account of grammatical structure whereby constituents are 'proven' well-formed. At the same time it facilitates a compositional semantics through the Curry-Howard correspondence ([6]), in particular reducing quantifier scope ambiguities in natural language to a question of proof identity. In practice, however, the non-commutativity, and, arguably, non-associativity of syntactic composition was found to be at odds with the ability, in general, to derive all combinatorially possible scopal readings for a given sentence. This is particularly well exemplified by van Benthem's substructural hierarchy ([39]), where the gradual reintroduction of structural rules to a base logic entirely devoid thereof makes available additional proofs at the expense of applications to grammatical analysis. Several solution directions have been explored, among which we find the weakening of compositionality to a relation ([22]) and the use of *controlled* structural rules ([32]).

The apparent mismatch in the choice of structural postulates for syntactic vs. semantic analysis is observed as well in the use of continuation-passing style (CPS) translations for defining the derivational semantics of classical type-logical grammars. Previous work in this area, such as that of Bernardi and Moortgat [7], chose target languages for this purpose like the proof terms of the commutative Lambek calculus, raising the question as to whether similar such translations may be devised that make no demands as to the presence of structural postulates in the target other than what is available in the source.

In this paper we argue that one *can* fully dispense with associativity and commutativity *without* compromising applications to the analysis of scopal ambiguities or to the definition of CPS translations. To this end, we introduce a variation $\mathbf{NL}_{\otimes\neg}$ on the non-associative Lambek calculus **NL** where we replace the latter's directional implications with linear negation. We first explore an axiomatic formulation in S2, followed by the presentation of natural deduction and a sequent calculus in S3. The applications mentioned above are fleshed out in S4, while S5 provides a model-theoretic completeness proof for Cut-free derivations. The latter proof is, in particular, constructive, and so exhibits an algorithm for transforming arbitrary derivations into normal form. The automatic extraction thereof has been carried out through a formalization of our reasoning in the Coq theorem proving assistant ([8]) and we briefly report on its behavior when applied to a concrete example. Finally, in S6 we attempt to more firmly situate $\mathbf{NL}_{\otimes\neg}$ within the wider landscape of categorial type logics by briefly touching on some areas of study neglected in S2–S5, including relational semantics and the complexity and expressivity of $\mathbf{NL}_{\otimes\neg}$. Some of the (expected) 'results' in this section will be stated as conjectures and left to future research, while in other cases we will provide a proof sketch.

Finally, a note on presentation. The contents of S2–S4 primarily concern proof formalisms whose applications to type-logical grammar have been the subject of extensive prior research, and so we will largely eschew the usual definition-theorem-proof format in favor of a more informal mode of exposition.[1] We shift gears in S5, however, where we provide a completeness proof for Cut-free derivations, the presentation of which we believe will benefit from a more careful elaboration of the definitions and lemmata upon which our argument is based.

2 Axiomatic Presentation

In this section we present a formulation of $\mathbf{NL}_{\otimes\neg}$ based on an infinite set of axioms, similar to that found in [9] for the associative Lambek calculus. Since, however, the 'meanings' of the logical connectives are more readily apparent from the algebraic point of view, we begin our exposition by defining $\mathbf{NL}_{\otimes\neg}$-*algebras*, serving, when combined with a valuation for atomic propositions, as our models. Thus, let us define by an $\mathbf{NL}_{\otimes\neg}$-algebra any tuple $(M, \otimes, \neg, \leq)$ s.t.

[1] Although we aim to make our presentation self-contained, the reader may benefit from the tutorial overview found in [33].

(i) $(M, \leq)$ is a poset; i.e., $\leq$ is reflexive, transitive and antisymmetric;
(ii) $\otimes$ is a binary operation on M that is isotone in both arguments; i.e., for all $u, v, x, y \in M$, if $x \leq u$ and $y \leq v$, then $x \otimes y \leq u \otimes v$;
(iii) $\neg$ is a unary operation on M forming a(n antitone) Galois connection with itself; i.e., for all $x, y \in M$, $x \leq \neg y$ implies $y \leq \neg x$; and
(iv) For all $x, y, z \in M$, if $x \otimes y \leq \neg z$, then $y \otimes z \leq \neg x$.

Here, condition (iii) is equivalent to stating (v) $x \leq \neg\neg x$ for all $x \in M$ and (vi) that $\neg$ is antitone, meaning $\neg y \leq \neg x$ for all $x, y \in M$ s.t. $x \leq y$ ([17, 34]). To see, e.g., that said properties are implied by (iii), we may advance the following arguments, abbreviated here informally as derivations:

$$\dfrac{\dfrac{}{\neg x \leq \neg x}\,(i)}{x \leq \neg\neg x}\,(iii) \qquad\qquad \dfrac{\dfrac{x \leq y \quad \dfrac{}{y \leq \neg\neg y}\,(v)}{x \leq \neg\neg y}\,(i)}{\neg y \leq \neg x}\,(iii)$$

E.g., the left derivation is to be taken as shorthand for saying that, for arbitrary $x \in M$, since $\neg x \leq \neg x$ by (i), it follows that $x \leq \neg\neg x$ by (iii). Next, we consider condition (iv), which informally we may think of as expressing a form of cyclicity ([40]). In particular, starting from $x \otimes y \leq \neg z$, we find that repeated applications will iterate through the cyclic permutations of x, y, z, 'projected' onto the inequality $(-) \otimes (-) \leq \neg(-)$. As consequences we have that (vii) $y \otimes \neg(x \otimes y) \leq \neg x$ and (viii) $\neg(y \otimes x) \otimes y \leq \neg x$:

$$\dfrac{\dfrac{}{x \otimes y \leq \neg\neg(x \otimes y)}\,(i)}{y \otimes \neg(x \otimes y) \leq \neg x}\,(iv) \qquad\qquad \dfrac{\dfrac{\dfrac{}{y \otimes x \leq \neg\neg(y \otimes x)}\,(i)}{x \otimes \neg(y \otimes x) \leq \neg y}\,(iv)}{\neg(y \otimes x) \otimes y \leq \neg x}\,(iv)$$

We now come to our axiomatization, based, as mentioned, on that found in [9]. To start with, we shall assume our logical vocabulary to comprise, besides the usual brackets, a countably infinite supply of atoms p along with logical constants '$\otimes$' (tensor, or multiplicative conjunction) and '$\neg$' (intuitionistic linear negation), leading to the following notion of (well-formed) formula:

$$A, B, C, \ldots ::= \; p \mid (A \otimes B) \mid \neg A \tag{1}$$

Per custom, we usually omit outer brackets; a convention we shall similarly apply to future definitions that rely on bracketing for exhibiting structure. Derivable statements take the form of *arrows* $A \to B$, subject to the single rule of inference that from $A \to B$ and $B \to C$ we may infer $A \to C$, henceforth denoted (R1), and based on axioms defined as the smallest set s.t.: (i) $A \to A$, $A \to \neg\neg A$ and $B \otimes \neg(A \otimes B) \to \neg A$ are axioms; and (ii) if $A \to B$ is an axiom, then so are $\neg B \to \neg A$, $C \otimes A \to C \otimes B$ and $A \otimes C \to B \otimes C$. Figure 1 repeats this definition using the two judgement forms '$A \to B$ is an axiom' and simply '$A \to B$,' the latter a lazy substitute for '$A \to B$ is a theorem.'

By a *model* we shall understand any pair of an $\mathbf{NL}_{\otimes\neg}$-algebra M together with a *valuation* v, mapping atoms p to elements of M. In particular, v may

Axioms.

A1. $A \to A$ is an axiom
A2. $A \to \neg\neg A$ is an axiom
A3. $B \otimes \neg(A \otimes B) \to \neg A$ is an axiom

$$\frac{A \to B \text{ is an axiom}}{\begin{array}{r} C \otimes A \to C \otimes B \text{ is an axiom} \\ A \otimes C \to B \otimes C \text{ is an axiom} \\ \neg B \to \neg A \text{ is an axiom} \end{array}}$$

Rules of inference.

$$\frac{A \to B \text{is an axiom}}{A \to B}\ Ax \qquad \frac{A \to B \quad B \to C}{A \to C}\ R1$$

Fig. 1. Axiomatic presentation of $\mathbf{NL}_{\otimes\neg}$.

be extended to arbitrary formulas by defining $v(A \otimes B) := v(A) \otimes v(B)$ and $v(\neg A) := \neg v(A)$. We seek to prove that $A \to B$ iff $v(A) \leq v(B)$ for all models; i.e., that our axiomatization of $\mathbf{NL}_{\otimes\neg}$ is sound and complete. Soundness is an easy consequence of our prior demonstrations that, for any $\mathbf{NL}_{\otimes\neg}$-algebra M and arbitrary $x, y \in M$, (v) $x \leq \neg\neg x$, (vi) $\neg$ is antitone and (vii) $y \otimes \neg(x \otimes y) \leq \neg x$. It thus remains to prove completeness. As preparation, we first establish a number of admissible inference rules for $\mathbf{NL}_{\otimes\neg}$, collected together in Figure 2.

To prove the admissibility of (R2), (R3) and (R4), one may proceed by induction on the derivation of their premise $A \to B$. In the base case, the latter is an axiom, and it follows immediately that the conclusions are then also axioms, and hence theorems, as well. Otherwise, $A \to B$ was derived by application of (R1), and we apply the induction hypothesis to the premises, $A \to D$ and $D \to B$, say. E.g., for (R2) we would proceed as in

$$\frac{\dfrac{A \to D}{C \otimes A \to C \otimes D}\ IH \quad \dfrac{D \to B}{C \otimes D \to C \otimes B}\ IH}{C \otimes A \to C \otimes B}\ R1$$

with a similar reasoning applying to handling the induction steps for (R3) and (R4). (R5) is now an easy consequence of (R1) with instances of (R2) and (R3) as premises, while (R6) may be established as follows:

$$\frac{\dfrac{}{B \to \neg\neg B}\ Ax \quad \dfrac{A \to \neg B}{\neg\neg B \to \neg A}\ R4}{B \to \neg A}\ R1$$

Finally, we have the following derivation for (R7):

$$\frac{\dfrac{\dfrac{A \otimes B \to \neg C}{C \to \neg(A \otimes B)}\ R6}{B \otimes C \to B \otimes \neg(A \otimes B)}\ R2 \quad \dfrac{}{B \otimes \neg(A \otimes B) \to \neg A}\ Ax}{B \otimes C \to \neg A}\ R1$$

Returning now to the proof of completeness, if $v(A) \leq v(B)$ for every model, then in particular this inequality holds in any one model that we can come up

$$\frac{A \to B}{C \otimes A \to C \otimes B}\ R2 \qquad \frac{A \to B}{A \otimes C \to B \otimes C}\ R3 \qquad \frac{A \to B}{\neg B \to \neg A}\ R4$$

$$\frac{A \to C \quad B \to D}{A \otimes B \to C \otimes D}\ R5 \qquad \frac{B \to \neg A}{A \to \neg B}\ R6 \qquad \frac{A \otimes B \to \neg C}{B \otimes C \to \neg A}\ R7$$

Fig. 2. Admissible rules of inference for $\mathbf{NL}_{\otimes\neg}$.

with for which we are able to prove $A \to B$ as a consequence. A first attempt might take the underlying set M of our algebra to be the set of all formulas and read $\leq$ as $\to$, but this scheme runs afoul of the derivability relation not being antisymmetric and so does not define an $\mathbf{NL}_{\otimes\neg}$-algebra. E.g., although $\neg A \leftrightarrows \neg\neg\neg A$, certainly not $\neg A = \neg\neg\neg A$. Instead, we may adapt the well-known Lindenbaum-Tarski construction,[2] as follows. Let $A \sim B$ iff $A \leftrightarrows B$, constituting an equivalence relation, and define by M the set of equivalence classes $[A]_\sim$. If $A \to B$, $C \sim A$ and $B \sim D$, then also $B \to D$, and so we may define $[A]_\sim \leq [B]_\sim$ iff $A \to B$. Clearly, $(M, \leq)$ is a poset. Next, define $[A]_\sim \otimes [B]_\sim := [A \otimes B]_\sim$ and $\neg[A]_\sim := [\neg A]_\sim$. The admissibility of (R5), (R6) and (R7) implies this gives us an $\mathbf{NL}_{\otimes\neg}$-algebra. Finally, defining $v(p) := [p]_\sim$, we can easily prove that $v(A) = [A]_\sim$ for all A, from which completeness readily follows: $v(A) \leq v(B)$ implies $[A]_\sim \leq [B]_\sim$, iff $A \to B$ by definition.

Other examples of $\mathbf{NL}_{\otimes\neg}$-algebras and of their usage in completeness proofs may be found in S5.1 and S6.3. We conclude this section by observing there exists a left-right symmetry for $\mathbf{NL}_{\otimes\neg}$, in the sense that $A \to B$ iff $A^\bowtie \to B^\bowtie$, where $p^\bowtie := p$, $(C \otimes D)^\bowtie := D^\bowtie \otimes C^\bowtie$ and $(\neg C)^\bowtie := \neg C^\bowtie$.

3 Natural Deduction and Sequent Calculus

Whereas our axiomatic presentation treated derivability '$A \to B$' as a relation between a single hypothesis A and conclusion B, the natural deduction and sequent calculus formulations that we will take up next accept a possible multitude of hypotheses. In keeping, however, with the non-associativity and non-commutativity of $\otimes$, these are combined not into a (multi)set, but rather appear as the leaves of a binary-branching tree. We further adopt the point of view of interpreting formulas-as-types and pair hypotheses with (distinct) variables, out of which a (proof) term is then constructed for labeling the derived statement. Assume, now, to have at our disposal a countably infinite number of variable letters $x, y, z, \ldots$. By a *structure* $\Gamma, \Delta, \ldots$ we shall then understand any binary bracketing of variable-formula pairs (or hypotheses) A^x:

$$\Gamma, \Delta, \ldots ::= \ A^x \mid (\Gamma, \Delta) \tag{2}$$

[2] See, for example, [16] for a previous explication of this concept within the context of substructural logics.

$$\frac{}{x : A^x \vdash A}\ Ax$$

$$\frac{M : \Gamma \vdash {\sim}\Delta}{M : \Delta \vdash {\sim}\Gamma}\ P1 \qquad \frac{M : \Gamma, \Delta \vdash {\sim}\Theta}{M : \Delta, \Theta \vdash {\sim}\Gamma}\ P2$$

$$\frac{M : \Gamma \vdash \neg A \quad N : \Delta \vdash A}{(M\ N) : \Gamma \vdash {\sim}\Delta}\ \neg E \qquad \frac{M : \Gamma \vdash {\sim}A^x}{(\lambda x M) : \Gamma \vdash \neg A}\ \neg I$$

$$\frac{M : \Gamma \vdash A \otimes B \quad N : \Delta[(A^x, B^y)] \vdash \Xi}{(\mathbf{case}\ M\ \mathbf{of}\ \langle x, y\rangle N) : \Delta[\Gamma] \vdash \Xi}\ \otimes E \qquad \frac{M : \Gamma \vdash A \quad N : \Delta \vdash B}{\langle M, N\rangle : \Gamma, \Delta \vdash A \otimes B}\ \otimes I$$

Fig. 3. Natural deduction for $\mathbf{NL}_{\otimes\neg}$.

Here A^x may be read as declaring x to be a variable of (syntactic) type A under the aforementioned formulas-as-types interpretation, while $(\cdot,\cdot)$ serves as a counterpart for $\otimes$ operating on hypotheses instead of formulas. A *context* $\Gamma[]$ is a structure Γ with exactly one occurrence of a leaf A^x having been replaced by $[]$. In writing $\Gamma[\Delta]$, we then mean to substitute Δ for $[]$ in $\Gamma[]$.

We next will consider *sequents* $M : \Gamma \vdash \Xi$, where Ξ is either a single formula or of the form ${\sim}\Delta$. Here, two additional concepts are introduced, to wit that of *raw terms* M and the symbol $\sim$. The latter, to start with, relates to $\neg$ as $(\cdot,\cdot)$ does to $\otimes$; i.e., whereas $\otimes$ and $\neg$ derive new formulas, in contrast $(\cdot,\cdot)$ and $\sim$ assign structure to the hypotheses of a sequent. Next, (raw) terms M are defined as below. Their intended interpretation is to serve as the proof terms for $\mathbf{NL}_{\otimes\neg}$, and hence are not to be confused with the ordinary λ-calculus (as should also be clear from the appearance of a matching construct).

$$M, N, \ldots ::= x \mid (\lambda x M) \mid \langle M, N\rangle \mid (M\ N) \mid (\mathbf{case}\ M\ \mathbf{of}\ \langle x, y\rangle N) \tag{3}$$

Figure 3 defines natural deduction for $\mathbf{NL}_{\otimes\neg}$. Here, in applications of ($\neg$E) and ($\otimes$I), the free variables in Γ and in Δ must be disjoint, with the same restriction applying to $\Delta[]$ and Γ in ($\otimes$E), while in addition x and y are not to occur free in $\Delta[]$ in the latter case either. It follows that if $M : \Gamma \vdash \Xi$ (i.e., if the latter sequent is derivable), then no variable occurs free more than once in M, and further that any subexpressions of the form $(\lambda x M')$ or $(\mathbf{case}\ M'\ \mathbf{of}\ \langle x, y\rangle N)$ in M bind exactly one occurrence of x in M', resp. of x and y in N. Note further the appearance of rules (P1) and (P2), resembling (Perm) and (L-Shift) from [15] (see also S4.1). Their function is much the same as those of Belnap's display postulates ([5]), in that we may use them to prove the following *display property*: if $M : \Gamma[A^x] \vdash {\sim}\Delta$ or $M : \Gamma \vdash {\sim}\Delta[A^x]$, then $M : A^x \vdash {\sim}\Theta$ for some Θ.

Before introducing normal forms, we first mention some admissible inference rules that will prove useful in the sequel. First, using (P1) and (P2), we derive

$$\frac{M : \Delta, \Theta \vdash {\sim}\Gamma}{M : \Gamma, \Delta \vdash {\sim}\Theta}\ P3$$

$$\frac{}{x : p^x \Rightarrow p}\ Ax$$

$$\frac{M : \Gamma \Rightarrow {\sim}\Delta}{M : \Delta \Rightarrow {\sim}\Gamma}\ P1 \qquad \frac{M : \Gamma, \Delta \Rightarrow {\sim}\Theta}{M : \Delta, \Theta \Rightarrow {\sim}\Gamma}\ P2$$

$$\frac{N : \Delta \Rightarrow A}{(x\ N) : {\neg}A^x \Rightarrow {\sim}\Delta}\ {\neg}L \qquad \frac{M : \Gamma \Rightarrow {\sim}A^x}{(\lambda x M) : \Gamma \Rightarrow {\neg}A}\ {\neg}R$$

$$\frac{N : \Delta[(A^x, B^y)] \Rightarrow \Xi}{(\mathbf{case}\ z\ \mathbf{of}\ \langle x, y\rangle N) : \Delta[A \otimes B^z] \Rightarrow \Xi}\ {\otimes}L \qquad \frac{M : \Gamma \Rightarrow A \quad N : \Delta \Rightarrow B}{\langle M, N\rangle : \Gamma, \Delta \Rightarrow A \otimes B}\ {\otimes}R$$

Fig. 4. Cut-free sequent calculus for $\mathbf{NL}_{\otimes\neg}$.

Next, we note derivations in natural deduction are closed under substitution. I.e., writing $N[x := M]$ to denote the substitution of M for the free occurrences of x in N,[3] we may establish (S1) and (S2) below by a straightforward mutual induction on the right premises:

$$\frac{M : \Gamma \vdash A \quad N : \Delta[A^x] \vdash \Xi}{N[x := M] : \Delta[\Gamma] \vdash \Xi}\ S1 \qquad \frac{M : \Gamma \vdash A \quad N : \Delta \vdash {\sim}\Theta[A^x]}{N[x := M] : \Delta \vdash {\sim}\Theta[\Gamma]}\ S2$$

We define *normal forms* for raw terms and derivations inductively, rather than using a rewrite relation. To this end, it suffices simply to exhibit a Cut-free sequent calculus, which we do in Figure 4, and where, to avoid confusion, we now write sequents using $\Rightarrow$ instead of $\vdash$. Note that its formulation may be obtained from that of natural deduction by instantiating the left premises of elimination rules with axioms, while restricting applications of (Ax) to atoms.

Having defined three proof formalisms for $\mathbf{NL}_{\otimes\neg}$, it remains to show their equivalence in terms of derivability. This we will do presently for the axiomatic presentation on the one hand, and natural deduction and sequent calculus on the other. Specifically, S3.1 proves $A \to B$ implies $M : A^x \vdash B$ for some M, while S3.2 shows $A \to B$ whenever $M : A^x \Rightarrow B$ for any M. On the other hand, the proof that $M : \Gamma \vdash \Xi$ implies $N : \Gamma \Rightarrow \Xi$ for some N is postponed until S5, after a brief reprive from our proof-theoretical investigations with an exploration in S4 of the applications of $\mathbf{NL}_{\otimes\neg}$ to the study of natural language semantics.

3.1 Axiomatic Derivability Implies Natural Deduction Derivability

For the remainder of this article, we omit all terms and variables from derivations (whether natural deduction or sequent calculus) whenever we are only interested in derivability, as opposed to proof identity. We are now to show $A \to B$ implies $A \vdash B$. Proceeding by induction on the derivation for $A \to B$, we note (A1) and

[3] Here, we assume an occurrence of x is free in N if it does not occur as part of M' inside a subexpression of the form $(\lambda x M')$, $(\mathbf{case}\ N'\ \mathbf{of}\ \langle x, y\rangle M')$ or $(\mathbf{case}\ N'\ \mathbf{of}\ \langle y, x\rangle M')$.

(R1) from Figure 1 follow readily from (Ax) and (S1,2) respectively, while (A2) is shown thus:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\neg A \vdash \neg A}\,Ax \quad \dfrac{}{A \vdash A}\,Ax}{\neg A \vdash {\sim} A}\,\neg E}{A \vdash {\sim}\neg A}\,P1}{A \vdash \neg\neg A}\,\neg I$$

Next consider (A3), omitting the final application of ($\otimes$E) for reasons of space:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\neg(A \otimes B) \vdash \neg(A \otimes B)}\,Ax \quad \dfrac{\dfrac{}{A \vdash A}\,Ax \quad \dfrac{}{B \vdash B}\,Ax}{A, B \vdash A \otimes B}\,\otimes I}{\neg(A \otimes B) \vdash {\sim}(A, B)}\,\neg E}{A, B \vdash {\sim}\neg(A \otimes B)}\,P1}{B, \neg(A \otimes B) \vdash {\sim} A}\,P2}{B, \neg(A \otimes B) \vdash \neg A}\,\neg I$$

It remains to show (R2), (R3) and (R4), from which the derivability of all axioms readily follows. We here show only (R2) and (R4), with (R3) following similarly.

$$\dfrac{\dfrac{\dfrac{}{\neg B \vdash \neg B}\,Ax \quad A \vdash B}{\neg B \vdash {\sim} A}\,\neg E}{\neg B \vdash \neg A}\,\neg I \qquad\qquad \dfrac{\dfrac{}{A \otimes C \vdash A \otimes C}\,Ax \quad \dfrac{A \vdash B \quad \dfrac{}{C \vdash C}\,Ax}{A, C \vdash B \otimes C}\,\otimes I}{A \otimes C \vdash B \otimes C}\,\otimes E$$

3.2 Sequent Calculus Derivability implies Axiomatic Derivability

We intend to prove $\Gamma \Rightarrow \Xi$ implies $\Gamma^\bullet \leq \Xi^\bullet$, where $\Gamma^\bullet$ and $\Xi^\bullet$ are formulas, defined $A^\bullet := A$, $(\Gamma, \Delta)^\bullet := \Gamma^\bullet \otimes \Delta^\bullet$ and $({\sim}\Gamma)^\bullet := \neg\Gamma^\bullet$. Proceeding by induction on the derivation of $\Gamma \Rightarrow \Xi$, we note ($\neg$R) is immediate, while (Ax), (P1), (P2), ($\neg$L) and ($\otimes$R) trivially reduce to (A1), (R6), (R7), (R4) and (R5) resp. This leaves only ($\otimes$L). Note $\Delta[\Gamma]^\bullet = \Delta[\Gamma^\bullet]^\bullet$ by induction on $\Delta[]$, implying $\Delta[(A, B)]^\bullet = \Delta[A \otimes B]^\bullet$, from which the desired result follows immediately.

4 Applications

4.1 $\mathbf{NL}_{\otimes\neg}$ as a Target Language for CPS Translations

The use of CPS translations (known in proof theory as double-negation translations) for defining the derivational semantics of type-logical grammars was previously discussed by Bernardi and Moortgat [7], among others. Said works, however, took as their target language either the simply-typed λ-calculus or the proof terms of the commutative Lambek calculus. A natural question to ask is whether instead the target language may be so chosen as to not require the inclusion of any structural postulates other than those present in the source as well, and $\mathbf{NL}_{\otimes\neg}$ has been devised to serve as an answer in the affirmative. In this sense, $\mathbf{NL}_{\otimes\neg}$ may be considered as the base logic for a substructural hierarchy

$$\frac{}{\Rightarrow p, \bar{p}}\ Ax$$

$$\frac{\Rightarrow \Gamma, \Delta}{\Rightarrow \Delta, \Gamma}\ Perm \qquad \frac{\Rightarrow (\Gamma, \Delta), \Theta}{\Rightarrow \Gamma, (\Delta, \Theta)}\ L\text{-}Shift$$

$$\frac{\Rightarrow A, \Gamma \quad \Rightarrow B, \Delta}{\Rightarrow (A \otimes B), (\Delta, \Gamma)}\ \otimes \qquad \frac{\Rightarrow (A, B), \Gamma}{\Rightarrow A \oplus B, \Gamma}\ \oplus$$

Fig. 5. (Cut-free) sequent calculus for **CNL**.

mirroring that of van Benthem [39], replacing the latter's directional implications with intuitionistic negation. As shown in the next subsection, however, $\mathbf{NL}_{\otimes\neg}$ does not suffer from the same limitations as its counterpart **NL** in van Benthem's original hierarchy, in the sense that even without commutativity and associativity it can account for scopal ambiguities.[4]

As illustration, we exhibit a double-negation translation for classical non-associative Lambek calculus **CNL** ([15]), constituting a strongly conservative extension of its intuitionsitic counterpart **NL** ([10]). We will follow closely the presentation of **CNL** found in [15] using a one-sided sequent calculus, although alternative sequent calculus-based formulations may be found in [10].

In short, assuming, again, a countably infinite number of atoms p, the formulae of **CNL** may be defined as below, where $\bar{p}$ represents the negation of p. Note we use the same metavariables $A, B, \ldots$ for referring both to formulas of $\mathbf{NL}_{\otimes\neg}$ and of **CNL**, though context should serve amply for disambiguation.

$$A, B, C, \ldots ::= \ p \mid \bar{p} \mid (A \otimes B) \mid (A \oplus B) \tag{4}$$

Here, $\oplus$ (the 'par') refers to *multiplicative* disjunction. Although classical linear negation $A^{\perp}$ is not part of the formula language for the particular presentation of **CNL** adopted here, it may be defined instead at the meta-level using the following clauses ([10]):

$$p^{\perp} := \bar{p} \qquad \bar{p}^{\perp} := p$$
$$(A \otimes B)^{\perp} := B^{\perp} \oplus A^{\perp} \qquad (A \oplus B)^{\perp} := B^{\perp} \otimes A^{\perp}$$

Various choices of double-negation translations may be explored, though a simple one (albeit certainly not the most economical) is the following:

$$\lceil p \rceil := p \qquad \lceil \bar{p} \rceil := \neg p$$
$$\lceil A \otimes B \rceil := \neg\neg\lceil A \rceil \otimes \neg\neg\lceil B \rceil \qquad \lceil A \oplus B \rceil := \neg(\neg\lceil B \rceil \otimes \neg\lceil A \rceil)$$

Structures $\Gamma, \Delta, \ldots$ for **CNL** are defined as for $\mathbf{NL}_{\otimes\neg}$, and we specify the extension $\lfloor \Gamma \rfloor$ of $\lceil \cdot \rceil$ to structures by $\lfloor A \rfloor := \neg\lceil A \rceil$ and $\lfloor (\Gamma, \Delta) \rfloor := (\lfloor \Delta \rfloor, \lfloor \Gamma \rfloor)$. Sequents of **CNL** take the form $\Rightarrow \Gamma, \Delta$, with derivability defined as depicted

[4] That associativity and commutativity of $\otimes$ are indeed not derivable in $\mathbf{NL}_{\otimes\neg}$ will be proved at the end of S5.1.

$$\frac{}{(y\ \lambda k(x\ k)) : \neg p^x \Rightarrow \sim\neg\neg p^y}\ Ax$$

$$\frac{M : \lfloor \Gamma \rfloor \Rightarrow \sim\lfloor \Delta \rfloor}{M : \lfloor \Delta \rfloor \Rightarrow \sim\lfloor \Gamma \rfloor}\ Perm$$

$$\frac{M : \lfloor \Delta \rfloor, \lfloor \Gamma \rfloor \Rightarrow \sim\Theta}{M : \lfloor \Gamma \rfloor \Rightarrow \sim(\lfloor \Theta \rfloor, \lfloor \Delta \rfloor)}\ L\text{-}Shift$$

$$\frac{M : \neg\lceil A \rceil^x \Rightarrow \sim\lfloor \Gamma \rfloor \quad N : \neg\lceil B \rceil^y \Rightarrow \sim\lfloor \Delta \rfloor}{(z\ \langle \lambda x M, \lambda y N \rangle) : \neg(\neg\neg\lceil A \rceil \otimes \neg\neg\lceil B \rceil)^z \Rightarrow \sim(\lfloor \Gamma \rfloor, \lfloor \Delta \rfloor)}\ \otimes$$

$$\frac{M : \neg\lceil B \rceil^u, \neg\lceil A \rceil^v \Rightarrow \sim\lfloor \Gamma \rfloor}{(x\ \lambda y(\mathbf{case}\ y\ \mathbf{of}\ \langle u, v \rangle M)) : \neg\neg(\neg\lceil B \rceil \otimes \neg\lceil A \rceil)^x \Rightarrow \sim\lfloor \Gamma \rfloor}\ \oplus$$

Fig. 6. Proof terms for the double-negation translation of **CNL** to $\mathbf{NL}_{\otimes\neg}$.

in Figure 5. We now claim $\Rightarrow \Gamma, \Delta$ implies $\lfloor \Gamma \rfloor \Rightarrow \sim\lfloor \Delta \rfloor$ in $\mathbf{NL}_{\otimes\neg}$. The proof proceeds by induction on the derivation of $\Rightarrow \Gamma, \Delta$, and we show in Figure 6 the proof terms for each of the cases involved.

We conclude with a few more remarks. First, note it follows from the above discussion that there also exists a double-negation translation from **NL** to $\mathbf{NL}_{\otimes\neg}$, given the aforementioned result in [10] that **CNL** is a strongly conservative extension of the former. To make this a little more concrete, define the directional implications $B\backslash A$ and A/B by $B^{\perp}\oplus A$ and $A\oplus B^{\perp}$ resp. Thus, e.g., the type assignment $(np\backslash s)/np$ commonly used for transitive verbs in linguistic applications then translates to $(\overline{np} \oplus s) \oplus \overline{np}$, where np and s are atoms categorizing noun phrases and sentences resp. It follows that $\lceil (np\backslash s)/np \rceil = \neg(\neg\neg np \otimes \neg\neg(\neg s \otimes \neg\neg np)$. In practice we could do better by defining a translation that uses fewer negations, although the present one accords with that used in Figure 6.

We further note that other (more involved) types of double negation translations can be defined as well, like an adaptation of that of Girard [19]. The latter, in particular, enjoys the benefit of minimizing the number of negations by taking into account the *polarity* of a formula, constituting a property related to focused proof search ([2]) and being defined in terms of the (non)invertibility of the corresponding left and right introduction rules (see also S6.4).[5]

4.2 Quantifier Scope Ambiguities

Despite our claims as to the non-associativity and -commutativity of $\otimes$ (proven in S5.1), $\mathbf{NL}_{\otimes\neg}$ nonetheless readily accounts for scopal ambiguities, as we will

[5] One anonymous referee inquired whether such different double negation translations may be compared on the basis of their applications to the study of natural language semantics. A possible starting point for such an investigation could be the literature on the linguistic applications of CPS translations, such as found in a series of publications by Barker and Shan (see, e.g., their book treatment [3]).

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{s^k \Rightarrow s}\,Ax}{\neg s^\gamma \Rightarrow \sim s^k}\,\neg L}{\neg s^\gamma \Rightarrow \neg s}\,\neg R \quad \dfrac{}{np^x \Rightarrow np}\,Ax}{\neg s^\gamma, np^x \Rightarrow \neg s \otimes np}\,\otimes R}{np^y, (\neg s^\gamma, np^x) \Rightarrow np \otimes (\neg s \otimes np)}\,\otimes R}{\neg(np \otimes (\neg s \otimes np))^R \Rightarrow \sim(np^y, (\neg s^\gamma, np^x))}\,\neg L}{(\neg(np \otimes (\neg s \otimes np))^R, np^y), \neg s^\gamma \Rightarrow \sim np^x}\,P*}{(\neg(np \otimes (\neg s \otimes np))^R, np^y), \neg s^\gamma \Rightarrow \neg np}\,\neg R}{\neg\neg np^X \Rightarrow \sim((\neg(np \otimes (\neg s \otimes np))^R, np^y), \neg s^\gamma)}\,\neg L}{(\neg s^\gamma, \neg\neg np^X), \neg(np \otimes (\neg s \otimes np))^R \Rightarrow \sim np^y}\,P*}{(\neg s^\gamma, \neg\neg np^X), \neg(np \otimes (\neg s \otimes np))^R \Rightarrow \neg np}\,\neg R}{\neg\neg np^Y \Rightarrow \sim((\neg s^\gamma, \neg\neg np^X), \neg(np \otimes (\neg s \otimes np))^R)}\,\neg L}{\neg\neg np^X, (\neg(np \otimes (\neg s \otimes np))^R, \neg\neg np^Y) \Rightarrow \sim\neg s^\gamma}\,P*}{\neg\neg np^X, (\neg(np \otimes (\neg s \otimes np))^R, \neg\neg np^Y) \Rightarrow \neg\neg s}\,\neg R$$

Fig. 7. Deriving the object-wide scope reading for "everyone loves someone."

presently illustrate. To this end, consider the oft-quoted example, "Everyone loves someone," admitting both an object wide scope reading where some person is being loved by everyone, and a subject wide scope reading where everone may love a potentially different person. In **NL**, only the reading with subject wide scope would be accounted for, and associativity is required to derive the object wide scope reading as well. More complicated sentences may easily be imagined where associativity would not suffice either, though our treatment below of the present example should make clear such problems do not occur within $\mathbf{NL}_{\otimes\neg}$.

Adopting primitive types np (categorizing noun phrases) and s (sentences), we propose the type assignments of $\neg\neg np$ to "everyone" and "someone" and of $\neg(np \otimes (\neg s \otimes np))$ to "loves." To start with, the latter type is a simplification of that derived in S4.1 as illustration of our double negation translation when applied to the **NL** type $(np\backslash s)/np$. Next, the assignment of $\neg\neg np$ to the quantified noun phrases, instead of np, may be compared to the assignment of the *semantic* type $(e \to t) \to t$ common in the literature on Montagovian semantics, as opposed to simply e, where e and t stand for the types of individuals and truth values respectively. Finally, as the goal type we take $\neg\neg s$ instead of s, noting the only derivable sequent with righthand side instantiated by s to be the axiom $s \Rightarrow s$. Given these type assignments, we now seek M s.t.

$$M : \neg\neg np^X, (\neg(np \otimes (\neg s \otimes np))^R, \neg\neg np^Y) \Rightarrow \neg\neg s \tag{5}$$

We claim there to exist derivations in normal form for the following choices of M, corresponding to the subject wide scope and object wide scope readings resp.

$$\lambda\gamma(X\ \lambda x(Y\ \lambda y(R\ \langle y, \langle \lambda k(\gamma\ k), x\rangle\rangle))) \tag{6}$$

$$\lambda\gamma(Y\ \lambda y(X\ \lambda x(R\ \langle y, \langle \lambda k(\gamma\ k), x\rangle\rangle))) \tag{7}$$

Figure 7 derives the second reading, abbreviating subsequent applications of (P1) and (P2) by (P*). As remarked before, the latter rules are similar to Belnap's display postulates ([5]), in the sense that if $\Gamma[A] \Rightarrow {\sim}\Delta$ or $\Gamma \Rightarrow {\sim}\Delta[A]$, then $A \Rightarrow {\sim}\Theta$ for some Θ, where we say A is then *displayed*. Thus, if, from the perspective of top-down proof search, we first display the occurrence of $\neg\neg np$ for the subject (followed by ($\neg$L)), we obtain the subject-wide scope reading, whereas the object-wide scope reading is found by first displaying the object instead. This works regardless of the context surrounding the formula to be displayed, explaining why associativity and commutativity play no part in deriving all possible scopal readings.

5 Completeness of the Cut-Free Sequent Calculus

We next prove in S5.1 that if $M : \Gamma \vdash \Xi$, then $N : \Gamma \Rightarrow \Xi$ for some N. Our line of reasoning follows closely that of Okada [36], who proved Cut admissibility for linear logic by composing soundness of arbitrary derivations w.r.t. Girard's phase spaces together with completeness of Cut-free derivations. In comparison, we here exclusively define what is ordinarily referred to as the particular case of a *syntactic* model, constituting another example of an $\mathbf{NL}_{\otimes\neg}$-algebra, although its formulation is primarily inspired by that of intuitionistic phase spaces ([1]).

The main limitation of our proof, shared with Okada's, is that it concerns only (Cut-free) provability and makes no claims as to the preservation of proof identity; i.e., in stating the existence of some N for which $N : \Gamma \Rightarrow \Xi$ whenever $M : \Gamma \vdash \Xi$, we do not make any assertion as to the relation between N and M, whereas we would desire $N \equiv_{\beta\eta} M$ for a suitable definition of $\beta\eta$-equality $\equiv_{\beta\eta}$. Another way of looking at this is as follows. Our proof is constructive, implying it exhibits an algorithm for transforming arbitrary derivations into normal form. To additionally assert $M \equiv_{\beta\eta} N$ serves to ascertain the correctness of this algorithm. For now, we will instead suffice with reporting on the formalization of our reasoning inside Coq ([8]), which facilitates the automatic extraction of the normalization routine implicit in our proof. This in particular allows us to observe its behavior when applied to a concrete derivation in the presence of Don't Know nondeterminism (such as the example of S4.2 concerning scopal ambiguities), observing that it picks out the right normal form. We will report on these 'experiments' in S5.2, and while very far from actually demonstrating correctness, they serve as an indication that our choice of model may have relevance to proof identity as well. Future research in these directions we expect to take as a starting point the generalization of our model to the interpretation of proof terms, similar to the intuitionistic models explored in [31] and [14].

Before continuing, we mention some additional literature that bears similarity to the present work. First, phase spaces for **CNL** were studied in [4] and [10], the former proving a (weak) completeness result for *polarized* **CNL**, while the latter established strong completeness for **CNL** along with a host of other important results, including that **CNL** constitutes a strong conservative extension of **NL**

and that grammars based on **CNL** with assumptions are context-free. Next to that, we also find in [10] a definition of phase spaces for involutive non-associative Lambek calculus, differing from **CNL** in having two negations instead of one, the study of which was continued in [11]. Outside the realm of substructural logics, we find, for instance, the works of [23], [25] and [24], applying methods similar to those of Okada's in their study of intuitionistic and classical logic while specifically emphasizing the constructive nature of their reasoning. In particular, [25] even presents a hand-extracted algorithm from proofs of soundness and completeness; an exercise we shall also attempt in part at the end of S5.2.

5.1 Proof of Completeness

Definition 1. For any formula A, define $\bot_A := \{\Gamma \mid \Gamma \Rightarrow A\}$.

Definition 2. For X, Y sets of structures, define:[6]

$$\begin{aligned} Cl(X) &:= \{\Gamma \mid (\forall \Delta[], \Xi)((\forall \Theta \in X)(\Delta[\Theta] \Rightarrow \Xi) \supset \Delta[\Gamma] \Rightarrow \Xi)\} \\ \neg X &:= \{\Gamma \mid (\forall \Delta \in X)(\Gamma \Rightarrow {\sim}\Delta)\} \\ (X \cdot Y) &:= \{(\Gamma, \Delta) \mid \Gamma \in X \wedge \Delta \in Y\} \\ (X \otimes Y) &:= Cl(X \cdot Y) \end{aligned}$$

The next lemma establishes $Cl(X)$ is a closure operation, its usage in the preceding definition resembling that found in the theory of intuitionistic phase spaces.

Lemma 1. *$X \subseteq Cl(X)$, $Cl(Cl(X)) \subseteq Cl(X)$ and $Cl(X) \subseteq Cl(Y)$ if $X \subseteq Y$.*

Proof. First, let $\Gamma \in X$. Assuming $\Delta[\Theta] \Rightarrow \Xi$ for all $\Theta \in X$, it then follows $\Delta[\Gamma] \Rightarrow \Xi$, implying $\Gamma \in Cl(X)$. Next, suppose (a) $\Gamma \in Cl(Cl(X))$. To show $\Gamma \in Cl(X)$, we must prove $\Delta[\Gamma] \Rightarrow \Xi$ if (b) $\Delta[\Theta] \Rightarrow \Xi$ for all $\Theta \in X$. By (a), it suffices to show $\Delta[\Lambda] \Rightarrow \Xi$ if (c) $\Lambda \in Cl(X)$, as is immediate from definitional unfolding of (c) and application of (b). Finally, suppose (d) $X \subseteq Y$ and that (e) $\Gamma \in Cl(X)$. Then assuming $\Delta[\Theta] \Rightarrow \Xi$ for all $\Theta \in Y$, we infer $\Delta[\Theta] \Rightarrow \Xi$ whenever $\Theta \in X$ by (d), and so $\Delta[\Gamma] \Rightarrow \Xi$ by (e), proving $\Gamma \in Cl(Y)$.

The next lemma implies that $\bot_A$, $\neg X$ and $X \otimes Y$ are closed sets.

Lemma 2. *$Cl(\bot_A) \subseteq \bot_A$, $Cl(\neg X) \subseteq \neg X$ and $Cl(X \otimes Y) \subseteq X \otimes Y$.*

Proof. Note the third statements reduces to $Cl(Cl(X \cdot Y)) \subseteq Cl(X \cdot Y)$ which holds by Lemma 1. Next, suppose $\Gamma \in Cl(\bot_A)$. Then $\Gamma \Rightarrow A$, and hence $\Gamma \in \bot_A$, if $\Theta \Rightarrow A$ for all $\Theta \in \bot_A$, which holds by definition. Finally, let (a) $\Gamma \in Cl(\neg X)$. To see $\Gamma \in \neg X$, we must prove $\Gamma \Rightarrow {\sim}\Delta$ if (b) $\Delta \in X$. By (a), it suffices to show $\Theta \Rightarrow {\sim}\Delta$ if $\Theta \in \neg X$, as follows from (b).

The set of all closed sets of structures, ordered by set inclusion and together with the operations $\otimes$ and $\neg$ from Definition 2 constitutes an $\mathbf{NL}_{\otimes\neg}$-algebra. As explained below, however, we will not make use of this fact and so omit the proof. Nevertheless, we may continue to define valuations as in S2.

[6] We denote implication by '$\supset$' to avoid overloading '$\rightarrow$,' used previously in S2.

Definition 3. For any formula A, $[\![A]\!]$ is a set of structures defined by induction, as follows: $[\![p]\!] := \bot_p$, $[\![\neg A]\!] := \neg[\![A]\!]$ and $[\![A \otimes B]\!] := [\![A]\!] \otimes [\![B]\!]$. If $\Gamma \in [\![A]\!]$, we also write $\Gamma \Vdash A$ and say Γ *forces* A.

We seek to prove soundness to the extent that $\Gamma \vdash \Xi$ implies $[\![\Gamma]\!] \subseteq [\![\Xi]\!]$ for suitable definitions of $[\![\Gamma]\!]$ and $[\![\Xi]\!]$, yet to be provided. A common approach, such as found, e.g., in [10], is to define $[\![\Gamma]\!]$ by translating Γ to a formula through the systematic replacement of $(\cdot,\cdot)$ with $\otimes$ and resorting to Definition 3. Soundness then follows by proving that the sets of closed structures together form an $\mathbf{NL}_{\otimes\neg}$-algebra in the way described above. Here we will instead exploit the particularities of our syntactic model in pursuing a simplified definition of $[\![\Gamma]\!]$ using a set that is not necessarily closed, based on the observation that all we really need is for $[\![\Delta[A]]\!] \subseteq X$ to imply $\Delta[\Theta] \in X$ for all $\Theta \Vdash A$ when X is closed.

Definition 4. Define $[\![\Gamma]\!]$ by induction, as follows. The case $\Gamma = A$ reduces to the prior definition of $[\![A]\!]$, whereas $[\![(\Delta,\Theta)]\!] := ([\![\Delta]\!] \cdot [\![\Theta]\!])$. In addition, $[\![\cdot]\!]$ extends to right-hand sides Ξ by stipulating $[\![\Xi]\!] = [\![A]\!]$ if $\Xi = A$, while $[\![\Xi]\!] = \neg[\![\Gamma]\!]$ if $\Xi = {\sim}\Gamma$. Again, we write $\Gamma \Vdash \Delta$ and $\Gamma \Vdash \Xi$ if $\Gamma \in [\![\Delta]\!]$ resp. $\Gamma \in [\![\Xi]\!]$.

Lemma 2 implies $[\![A]\!]$ and $[\![\Xi]\!]$ are closed, although $[\![\Gamma]\!]$ in general need not be.

Lemma 3. *$Cl([\![A]\!]) \subseteq [\![A]\!]$ and $Cl([\![\Xi]\!]) \subseteq [\![\Xi]\!]$ for all A, Ξ.*

While our definition of $[\![\Gamma]\!]$ simplifies much of the soundness proof, one case that it actually complicates is that of ($\otimes$E). To help tackle it, we first state a few additional lemmata.

Lemma 4. *If $[\![\Gamma]\!] \subseteq [\![\Delta]\!]$, then $[\![\Theta[\Gamma]]\!] \subseteq [\![\Theta[\Delta]]\!]$ for all contexts $\Theta[]$.*

Proof. By a trivial induction on $\Theta[]$.

Lemma 5. *If $[\![\Gamma]\!] \subseteq Cl([\![\Delta]\!])$, then $[\![\Theta[\Gamma]]\!] \subseteq Cl([\![\Theta[\Delta]]\!])$ for all contexts $\Theta[]$.*

Proof. Assume (a) $[\![\Gamma]\!] \subseteq Cl([\![\Delta]\!])$ and proceed by induction on $\Theta[]$, the case $\Theta[] = []$ being immediate. Next, consider $\Theta[] = (\Theta_1, \Theta_2[])$, with $\Theta[] = (\Theta_1[], \Theta_2)$ being handled similarly. Thus, suppose (b) $[\![\Theta_2[\Gamma]]\!] \subseteq Cl([\![\Theta_2[\Delta]]\!])$ (by induction hypothesis) and (c) $\Lambda \in [\![(\Theta_1, \Theta_2[\Gamma])]\!]$, iff $\Lambda = (\Lambda_1, \Lambda_2)$ for (d) $\Lambda_1 \Vdash \Theta_1$ and (e) $\Lambda_2 \Vdash \Theta_2[\Gamma]$. We are then to prove $\Lambda \in Cl([\![\Theta[\Delta]]\!])$, for which it suffices to show $\Pi[\Lambda] \Rightarrow \Xi$ if (f) $\Pi[\Sigma] \Rightarrow \Xi$ for all $\Sigma \Vdash \Theta[\Delta]$. By (b) and (e), $\Lambda_2 \in Cl([\![\Theta_2[\Delta]]\!])$, and so we need but prove $\Pi[(\Lambda_1, \Upsilon)] \Rightarrow \Xi$ whenever (g) $\Upsilon \Vdash \Theta_2[\Delta]$. It follows by (d) and (g) that $(\Lambda_1, \Upsilon) \Vdash \Theta[\Delta]$, with the desired result obtained by (f).

Corollary 1. *$[\![\Delta[A \otimes B]]\!] \subseteq Cl([\![\Delta[(A,B)]]\!])$ for all contexts $\Delta[]$.*

Proof. By the previous lemma, it suffices to prove the above statement when $\Delta[] = []$. In fact, we have $[\![A \otimes B]\!] = [\![A]\!] \otimes [\![B]\!] = Cl([\![A]\!] \cdot [\![B]\!]) = Cl([\![(A,B)]\!])$.

Theorem 1. *(Soundness) If $\Gamma \vdash \Xi$, then $[\![\Gamma]\!] \subseteq [\![\Xi]\!]$.*

Proof. By induction on the derivation of $\Gamma \vdash \Xi$, noting (Ax) and ($\neg$I) are immediate. We next consider the cases (P2), ($\neg$E), ($\otimes$E) and ($\otimes$I), with (P1) being handled similarly to (P2).

- For (P2), suppose $\Lambda \Vdash (\Delta, \Theta)$, iff $\Lambda = (\Lambda_1, \Lambda_2)$ where $\Lambda_1 \Vdash \Delta$ and $\Lambda_2 \Vdash \Theta$. To show $\Lambda \Vdash {\sim}\Gamma$, we must prove $(\Lambda_1, \Lambda_2) \Rightarrow {\sim}\Gamma_1$ assuming $\Gamma_1 \Vdash \Gamma$. By induction hypothesis, $(\Gamma_1, \Lambda_1) \Rightarrow {\sim}\Lambda_2$ and we apply (P2).
- For ($\neg$E), assume $\Gamma_1 \Vdash \Gamma$. To prove $\Gamma_1 \Vdash {\sim}\Delta$, let $\Delta_1 \Vdash \Delta$. By induction hypothesis, $\Gamma_1 \Vdash \neg A$ and $\Delta_1 \Vdash A$, implying $\Gamma_1 \Rightarrow {\sim}\Delta_1$. As Δ_1 was arbitrary, it follows that $\Gamma_1 \Vdash {\sim}\Delta$, as desired.
- For ($\otimes$E), $[\![\Gamma]\!] \subseteq [\![A \otimes B]\!]$ by induction hypothesis, and hence, by Lemma 4 and Corollary 1, $[\![\Delta[\Gamma]]\!] \subseteq [\![\Delta[A \otimes B]]\!] \subseteq Cl([\![\Delta[(A, B)]]\!])$. Further, since, again by induction hypothesis, $[\![\Delta[(A, B)]]\!] \subseteq [\![\Xi]\!]$, it follows by Lemmas 1 and 3 that $Cl([\![\Delta[(A, B)]]\!]) \subseteq Cl([\![\Xi]\!]) \subseteq [\![\Xi]\!]$.
- For ($\otimes$I), let $\Theta \Vdash (\Gamma, \Delta)$, iff $\Theta = (\Theta_1, \Theta_2)$ for some $\Theta_1 \Vdash \Gamma$ and $\Theta_2 \Vdash \Delta$. To prove $\Theta \Vdash A \otimes B$, it suffices to show $\Pi[\Theta] \Rightarrow \Xi$ assuming $\Pi[(\Sigma_1, \Sigma_2)] \Rightarrow \Xi$ for all $\Sigma_1 \Vdash A$ and $\Sigma_2 \Vdash B$. Since $\Theta_1 \Vdash A$ and $\Theta_2 \Vdash B$ by induction hypothesis, the desired result readily follows.

We next set out to prove completeness for Cut-free derivations (Theorem 2).

Lemma 6. *For all A, $[\![A]\!] \subseteq \bot_A$ and $A \Vdash A$.*

Proof. By induction on A, noting the case where $A = p$ to be immediate.

- For $A = \neg B$, assume $\Gamma \Vdash \neg B$. As $B \Vdash B$ by induction hypothesis, it follows that $\Gamma \Rightarrow {\sim}B$ and we apply ($\neg$R). To show $\neg B \Vdash \neg B$, suppose $\Delta \Vdash B$. By induction hypothesis, $\Delta \Rightarrow B$, and so $\neg B \Rightarrow {\sim}\Delta$ by ($\neg$L), as desired.
- For $A = B \otimes C$, let $\Gamma \Vdash B \otimes C$. To prove $\Gamma \Rightarrow B \otimes C$, it suffices to show $(\Theta_1, \Theta_2) \Rightarrow B \otimes C$ for all $\Theta_1 \Vdash B$ and $\Theta_2 \Vdash C$. By induction hypothesis, $\Theta_1 \Rightarrow B$ and $\Theta_2 \Rightarrow C$ and we apply ($\otimes$R). To prove $B \otimes C \Vdash B \otimes C$, we must show $\Delta[B \otimes C] \Rightarrow \Xi$ whenever $\Delta[(\Theta_1, \Theta_2)] \Rightarrow \Xi$ for all $\Theta_1 \Vdash B$ and $\Theta_2 \Vdash C$. As, $B \Vdash B$ and $C \Vdash C$, we infer $\Delta[(B, C)] \Rightarrow \Xi$ and apply ($\otimes$L).

Corollary 2. *For all Γ, $\Gamma \Vdash \Gamma$.*

Proof. By a simple induction on Γ, invoking Lemma 6 for the base case.

Theorem 2. *(Completeness) For all Γ, Ξ, if $[\![\Gamma]\!] \subseteq [\![\Xi]\!]$, then $\Gamma \Rightarrow \Xi$.*

Proof. Since $\Gamma \Vdash \Gamma$ by Corollary 2, it follows $\Gamma \Vdash \Xi$. If $\Xi = A$, then $\Gamma \Rightarrow A$ by Lemma 6. If instead $\Xi = {\sim}\Delta$, then $\Gamma \Rightarrow {\sim}\Delta_1$ for every $\Delta_1 \Vdash \Delta$, and so in particular $\Gamma \Rightarrow {\sim}\Delta$ as $\Delta \Vdash \Delta$ (again, by Corollary 2).

The time has come to harvest. By composing Theorems 1 and 2 it follows that derivability in natural deduction implies Cut-free sequent calculus derivability.

Theorem 3. *For all Γ, Ξ, if $\Gamma \vdash \Xi$, then $\Gamma \Rightarrow \Xi$.*

Next, we may state the admissibility of Cut.

Theorem 4. *If $\Gamma \Rightarrow A$ and $\Delta[A] \Rightarrow \Xi$, then $\Delta[\Gamma] \Rightarrow \Xi$.*

Proof. $\Gamma \Rightarrow A$ implies $\Gamma \vdash A$, and so $[\![\Gamma]\!] \subseteq [\![A]\!]$ by Theorem 1. Similarly, from $\Delta[A] \Rightarrow \Xi$ we infer $[\![\Delta[A]]\!] \subseteq [\![\Xi]\!]$. By Lemma 4, $[\![\Delta[\Gamma]]\!] \subseteq [\![\Delta[A]]\!]$, hence $[\![\Delta[\Gamma]]\!] \subseteq [\![\Xi]\!]$. We conclude $\Delta[\Gamma] \Rightarrow \Xi$ by Theorem 2.

The completeness of the Cut-free sequent calculus implies that the question of provability for any given sequent may be settled by an exhaustive backward chaining proof search. We may thus state:

Theorem 5. *Provability in $\mathbf{NL}_{\otimes\neg}$ is decidable.*

In particular, our claims regarding the non-associativity and -commutativity of $\otimes$ can now be backed up by proof.

Corollary 3. *There exist no derivations for the sequents $A \otimes B \Rightarrow B \otimes A$, $(A \otimes B) \otimes C \Rightarrow A \otimes (B \otimes C)$ and $A \otimes (B \otimes C) \Rightarrow (A \otimes B) \otimes C$.*

5.2 Constructive Content

We recall that the results of S5.1 ignore proof identity, merely stating that if there exists M s.t. $M : \Gamma \vdash \Xi$, then there exists N s.t. $N : \Gamma \Rightarrow \Xi$. On the other hand, our proof is constructive, in the intuitive sense that it offers definite instructions on how to assemble derivations in normal form for provable sequents, as opposed to merely establishing the impossibility of their non-existence. In S5.2.1, we discuss the automatic extraction of the normalization routine implicit in our proof using Coq and its application to a concrete non-normal derivation in the presence of Don't Know nondeterminism, observing the right normal form is selected. Next, in S5.2.2, like Ilik [24], we present a *hand*-extracted algorithm from part of our proof (specifically Lemma 6), offering further insight into its actual operation. Note in this regard that we could not simply have reproduced the output of Coq's code generators for this latter purpose, given that their main goal of producing executable code does not lend itself readily to the writing of an algorithmic description meant for human reading.

Experiments in Scopal Ambiguities In S4.2 we considered the set of proof terms M s.t. $M : \neg\neg np^X, (\neg(np \otimes (\neg s \otimes np))^R, \neg\neg np^Y) \vdash \neg\neg s$, among which we now pick out the following:

$$\lambda\gamma(X\ \lambda x(\lambda z(Y\ \lambda y(R\ \langle y, z\rangle))\ \langle \gamma, x\rangle)) \tag{8}$$

Note the corresponding derivation is not in normal form, containing an application of (¬E) whose left premise instantiates (¬I):

$$\dfrac{\dfrac{\neg(np \otimes (\neg s \otimes np))^R, \neg\neg np^Y \vdash {\sim}(\neg s \otimes np)^z}{\neg(np \otimes (\neg s \otimes np))^R, \neg\neg np^Y \vdash \neg(\neg s \otimes np)}\ \neg I \qquad \neg s^\gamma, np^x \vdash \neg s \otimes np}{\neg(np \otimes (\neg s \otimes np))^R, \neg\neg np^Y \vdash {\sim}(\neg s^\gamma, np^x)}\ \neg E$$

As observed in S4.2, the sequent under consideration has two different derivations in normal form (that is to say, with distinct proof terms) and applying the algorithm implicit in Theorem 3 as obtained using Coq results in the subject-wide scope reading:

$$\lambda\gamma(X\ \lambda x(Y\ \lambda y(R\ \langle y, \langle \lambda k(\gamma\ k), x\rangle\rangle))) \tag{9}$$

While the above observation is consistent with our understanding of β-redexes in the simply-typed λ-calculus, we repeat that we have, of course, nót *proved* that in general terms of the form $(\lambda xM\ N)$ are always reduced to $M[x := N]$, or likewise for $(\mathbf{case}\ \langle M_1, M_2\rangle\ \mathbf{of}\ \langle x, y\rangle N)$ and $N[x := M_1][y := M_2]$.

We conclude with a few words on the representations we employed in Coq for the derivations of $\mathbf{NL}_{\otimes\neg}$. To this end we chose a first-order encoding, in the sense that, e.g., ($\neg$I) becomes a function mapping (derivations of) sequents $\Gamma \vdash {\sim}A$ into $\Gamma \vdash \neg A$. Such, however, ignores the variable-binding that occurs in our corresponding proof term λxM. Alternatively, we may conceive of ($\neg$I) as a *higher-order* function that takes as its argument a transformation of $\Delta \vdash A$ into $\Gamma \vdash {\sim}\Delta$ for arbitrary Δ, exploiting the abstraction mechanisms already available in the intuitionistic meta-language as a means to encode object-level bindings. This second approach is referred to by 'higher-order abstract syntax' (HOAS), although certain properties of the typing system of Coq necessitate a slight variation on this theme called '*parameterized* HOAS', or simply PHOAS ([13]). We intend to explore such encodings for $\mathbf{NL}_{\otimes\neg}$ in future work, together with a definition of intuitionistic models for its proof terms so as to achieve a strengthening of our results taking $\beta\eta$-equality into account, proceeding along the lines discussed in [31].

Normalization by Evaluation To further clarify the correspondence between the contents of S5.1 and a concrete normalization routine for derivations of $\mathbf{NL}_{\otimes\neg}$, we present a hand-extracted algorithm that we obtained from Lemma 6. First, note that we are now to read '$\Gamma \Vdash A$' as denoting a *set*, defined by induction on A as below, the elements of which are functions (of arity 0, in the base case) that evaluate to proof terms.

- $\Gamma \Vdash p$ is the set of proof terms M s.t. $M : \Gamma \Rightarrow p$.
- $f \in (\Gamma \Vdash \neg A)$ iff for all Δ and $g \in (\Delta \Vdash A)$, $f(\Delta, g) : \Gamma \Rightarrow {\sim}\Delta$.[7]
- $f \in (\Gamma \Vdash A \otimes B)$ iff for all $\Delta[]$, Ξ and g s.t. $g(\Theta_1, \Theta_2, h, k) : \Delta[(\Theta_1, \Theta_2)] \Rightarrow \Xi$ whenever $h \in (\Theta_1 \Vdash A)$ and $k \in (\Theta_2 \Vdash B)$, $f(\Delta[], \Xi, g) : \Delta[\Gamma] \Rightarrow \Xi$.

The proof of Lemma 6 may now be understood as defining, by mutual induction, a function $\uparrow_{\Gamma,A}$: $(\Gamma \Vdash A) \to \{M \mid M : \Gamma \Rightarrow A\}$ (*reflection*) and a constant $\downarrow_{x,A} \in (A^x \Vdash A)$ (*reification*), having adopted the notation and terminology from the literature on normalization by evaluation ([14]). First, reflection is defined as

[7] Note the 'type' of the second argument of f depends on the value of the first argument, requiring the use of dependent types in the formalization.

below, where, like in [24], we write '$\mapsto$' to denote abstraction at the level of the meta-language.

$$\begin{aligned}
\uparrow_{\Gamma,p} &:= M \mapsto M \\
\uparrow_{\Gamma,\neg A} &:= f \mapsto (\lambda x\ f(A, \downarrow_{x,A})) \quad (x \text{ fresh}) \\
\uparrow_{\Gamma,A\otimes B} &:= f \mapsto f([], A \otimes B, (\Theta_1, \Theta_2, g, h) \mapsto \langle \uparrow_{\Theta_1,A}(g), \uparrow_{\Theta_2,B}(h)\rangle)
\end{aligned}$$

Next, for reification, we have:

$$\begin{aligned}
\downarrow_{x,A} &:= x \\
\downarrow_{x,\neg A} &:= (\Delta, f) \mapsto (x\ \uparrow_{\Delta,A}) \\
\downarrow_{x,A\otimes B} &:= (\Delta[], \Xi, f) \mapsto (\textbf{case}\ z\ \textbf{of}\ \langle x, y\rangle f(A, B, \downarrow_{x,A}, \downarrow_{y,B})) \quad (y, z \text{ fresh})
\end{aligned}$$

Similarly, one may understand the proof of soundness (i.e., Theorem 1) as associating with the derivation of a sequent $M : \Gamma \Rightarrow \Xi$ a function f s.t. for every Δ and $g \in (\Delta \Vdash \Gamma)$, $f(\Delta, g) \in (\Delta \Vdash \Xi)$. Conversely, completeness maps any such function to a term M s.t. $M : \Gamma \vdash \Xi$.

6 Discussion

The above treatment of $\mathbf{NL}_{\otimes\neg}$ has still left many traditional areas of study in the proof theory of type-logical grammars unexplored. To partially remedy this situation, as well as to place our proposals in a larger context, we will now briefly touch on some of these topics. Specifically, we will discuss the complexity of proof search, the expressivity of grammars based on $\mathbf{NL}_{\otimes\neg}$, relational models, focused proof search, and, finally syntactic proofs of Cut admissibility. In a few cases we include sketches of proofs for important results, as for completeness w.r.t. relational models, while in others we will restrict to stating conjectures.

6.1 The Complexity of Proof Search

Although proof search in the associative Lambek calculus is NP-complete ([38]), polynomial time results were obtained for its non-associative restriction ([21]). Similarly, provability in **CNL** is decidable in polynomial time as well ([15, 10]). In light of these results, we state the following conjecture.

Conjecture 1. *Provability in* $\mathbf{NL}_{\otimes\neg}$ *is decidable in polynomial time.*

6.2 Expressivity

Grammars based on the Lambek calculus are context-free ([37]), with [12] proving the same for its non-associative restriction in the presence of a *finite* number of nonlogical axioms (or simply 'assumptions') as well as additives $\wedge$ and $\vee$ satisfying the distributive law. Context-freeness was similarly established in [10] for **CNL**, again with assumptions, and we conjecture that the same holds of $\mathbf{NL}_{\otimes\neg}$.

Conjecture 2. *Grammars based on* $\mathbf{NL}_{\otimes\neg}$ *enriched with a finite number of nonlogical axioms are context-free.*

6.3 Relational Semantics

We describe a possible relational semantics for $\mathbf{NL}_{\otimes\neg}$, similar to that found in [26] and [27] for various other type-logical grammars. To this end, we treat negation as impossibility, as explored in S2 of [17]. Define, then, a *frame* to be a tuple $F = (W, R_\otimes, R_\neg)$ where W is a nonempty set and $R_\otimes$, $R_\neg$ are ternary, resp. binary relations on W satisfying the following frame conditions:

$$\begin{array}{c}(\forall x, y)(R_\neg xy \supset R_\neg yx)\\ (\forall u, x, y, z)(R_\otimes xyz \wedge R_\neg xu \supset (\exists w)(R_\otimes wuy \wedge R_\neg zw))\end{array} \tag{10}$$

A *model* consists of a frame together with a valuation v mapping atomic formulas to subsets of W. The latter extends to arbitrary formulas as follows.

$$\begin{array}{rcl} x \Vdash p & \iff & x \in v(p)\\ x \Vdash \neg A & \iff & (\forall y)(R_\neg xy \supset y \nVdash A)\\ x \Vdash A \otimes B & \iff & (\exists y, z)(R_\otimes xyz \wedge y \Vdash A \wedge z \Vdash B)\end{array} \tag{11}$$

Soundness states that if $A \to B$ is derivable, then for any model, $x \Vdash A$ implies $x \Vdash B$. The first condition in (10), adopted from [17], is used to show the validity of (A2) $A \to \neg\neg A$, while the second one is needed for (A3) $B \otimes \neg(A \otimes B) \to \neg A$.

To prove completeness we follow the strategy found in [27], restricting here to a brief outline of their argument while providing details only when deviations occur. To construct a canonical model, we let W consist of all sets X of formulas s.t. if $A \in X$ and $A \to B$, also $B \in X$. Next, define, for all $X, Y \in W$,

$$\begin{array}{rcl} X\widehat{\otimes}Y & = & \{C \mid (\exists A, B)(A \in X \wedge B \in Y \wedge A \otimes B \to C)\}\\ \widehat{\neg}X & = & \{A \mid (\forall B)(A \to \neg B \supset B \notin X)\}\end{array} \tag{12}$$

If $X, Y \in W$, also $X\widehat{\otimes}Y, \widehat{\neg}X \in W$. Further, $(W, \widehat{\otimes}, \widehat{\neg}, \subseteq)$ is an $\mathbf{NL}_{\otimes\neg}$-algebra, as will be exploited below in establishing the frame conditions and proving the truth lemma. We now define $R_\otimes XYZ$ iff $Y\widehat{\otimes}Z \subseteq X$ (like in [27]) and $R_\neg XY$ iff $X \subseteq \widehat{\neg}Y$. The first frame condition in (10) reduces to $X \subseteq \widehat{\neg}Y$ implying $Y \subseteq \widehat{\neg}X$, while the verification of the second one makes use of $X\widehat{\otimes}Y \subseteq \widehat{\neg}Z$ only if $Y\widehat{\otimes}Z \subseteq \widehat{\neg}X$. It thus follows that we have defined a frame for $\mathbf{NL}_{\otimes\neg}$.

Next, let $v(p) = \{X \in W \mid p \in X\}$. It remains to prove the *truth lemma*, from which completeness readily follows; i.e., $A \in X$ iff $X \Vdash A$. Proceeding by induction on A, the base case is immediate while the case $A \otimes B$ is proved in [27]. We complete the proof for $\neg A$, concluding our discussion of relational models. First, assume $X \Vdash \neg A$, implying, by induction hypothesis, that, for all $Y \in W$, $A \notin Y$ whenever $X \subseteq \widehat{\neg}Y$. As $X \subseteq \widehat{\neg}\widehat{\neg}X$, it follows $A \notin \widehat{\neg}X$. Hence, there exists $B \in X$ s.t. $A \to \neg B$. Since $B \to \neg A$ by (R6), also $\neg A \in X$, as desired. Conversely, we must prove $X \Vdash \neg A$ assuming $\neg A \in X$. To this end, let $R_\neg XY$, it sufficing to show $Y \nVdash A$, or, by induction hypothesis, $A \notin Y$. On the contrary, assume $A \in Y$. As $R_\neg XY$ iff $Y \subseteq \widehat{\neg}X$, it follows $A \in \widehat{\neg}X$ and hence $\neg A \notin X$ as $A \to \neg\neg A$, contradicting our assumption. Ergo, $A \notin Y$, completing the proof.

6.4 Focused Proof Search

The Cut-free sequent calculus facilitates backward-chaining proof search due to its local subformula property. The practical utility of this method is tempered, however, by the presence of Don't Care nondeterminism, where distinct derivations of a given sequent may represent equivalent proof terms. Several remedies have been explored in the literature, among which we find proof nets ([18, 35]) and focused proof search ([2]). We here briefly discuss the latter, noting terms like 'first' and 'subsequent' as pertaining to the application of inference rules are to be interpreted from the perspective of top-down proof construction (i.e., starting from the to-be-proved sequent and proceeding towards the axioms).

Observe left introductions for $\otimes$ in $\mathbf{NL}_{\otimes\neg}$ are invertible (as may be shown using Cut), while right introductions aren't. In contrast, the opposite situation applies to $\neg$. In general, focused proof search prunes the number of derivations by requiring that invertible rules are to be applied as soon as possible, whereas the active formulas in the application of a non-invertible rule are to be principal in the rules instantiating the premises. The resulting proofs are thus seen to alternate between invertible and non-invertible 'phases,' sometimes referred to as negative, resp. positive. To make this more concrete, we observe the (partial) derivation below violates this scheme due to the intervening application of ($\otimes$L).

$$\dfrac{\dfrac{\dfrac{\Gamma[(C,D)] \Rightarrow A \quad \Delta \Rightarrow B}{\Gamma[(C,D)], \Delta \Rightarrow A \otimes B}\otimes R}{\Gamma[C \otimes D], \Delta \Rightarrow A \otimes B}\otimes L}{\neg(A \otimes B) \Rightarrow {\sim}(\Gamma[C \otimes D], \Delta)}\neg L$$

At least to the extent that it is shown, the above derivation does, however, have the redeeming quality that it can be turned into a focalized proof by permuting ($\otimes$L) downward (using (P1)). In the terminology of [30], we may call it *weakly* focalized, henceforth referring to our previous definition of focalization as *strong*. In fact, it's not hard to see that *all* Cut-free sequent derivations in $\mathbf{NL}_{\otimes\neg}$ are weakly focalized, in that they can always be made strongly focalized solely by permuting applications of invertible rules downward over non-invertible ones when possible (using (P1) if needed).

It is possible to provide a sequent calculus that admits only strongly focused proofs. Going a step further, one may consider a so-called 'polarized' version of $\mathbf{NL}_{\otimes\neg}$ (again, following [30]), where the formula language is split between a positive fragment (whose right introductions are non-invertible) and a negative one (vice versa), with two additional connectives $\uparrow$ and $\downarrow$ (the 'shifts') to explicitly mark transitions therebetween.

$$\begin{aligned} P, Q &::= \; p \mid (P \otimes Q) \mid {\downarrow}N \\ M, N &::= \; \neg P \mid {\uparrow}P \end{aligned} \tag{13}$$

The details for **CNL** were worked out in [4], while those for another closely related logic, the Lambek-Grishin calculus, may be found in [20].

6.5 Proving Cut Admissibility Syntactically

Besides our model-theoretic argument for Cut admissibility one may also provide a traditional syntactic proof, proceeding as usual. One type of circumstance warranting special attention, however, is when the Cut formula appears on opposite sides of the turnstile in the premise and conclusion of an application of (P1) or (P2). E.g.,

$$\dfrac{\Gamma \Rightarrow A \quad \dfrac{\Theta \Rightarrow {\sim}\Delta[A]}{\Delta[A] \Rightarrow {\sim}\Theta}\, P1}{\Delta[\Gamma] \Rightarrow {\sim}\Theta}\, Cut$$

To be able to carry the application of Cut over that of (P1), we shall want to consider generalizing the former's statement as compared to that found in Theorem 4 of S5.1. Specifically, let $S[]$ denote a context of the form $\Gamma[] \Rightarrow \Xi$ or $\Gamma \Rightarrow {\sim}\Delta[]$, with $S[\Theta]$ referring to the sequent obtained by substituting Θ for the unique occurrence of $[]$. Assuming a suitable definition of $\beta\eta$-equality $\equiv_{\beta\eta}$, the statement of the theorem now reads that if $M : \Gamma \Rightarrow A$ and $N : S[A^x]$ are derivable, then there exists $M' \equiv_{\beta\eta} N[x := M]$ s.t. $M' : S[\Gamma]$. Alternatively, we may consider the simpler statement, similar to that proved in S5, where we do not further qualify the existence of M' in terms of its relation to $N[x := M]$. Using the formulation of Cut involving contexts $S[]$, the above example can now be handled by a simple permutation.

Acknowledgements This work has benefited from the comments of three anonymous referees. All remaining errors are my own.

Bibliography

[1] Abrusci, V.M.: Sequent calculus for intuitionistic linear propositional logic. In: Mathematical Logic, pp. 223–242. Springer (1990)

[2] Andreoli, J.M.: Logic programming with focusing proofs in linear logic. Journal of logic and computation **2**(3), 297–347 (1992)

[3] Barker, C., Shan, C.: Continuations and Natural Language, Oxford Studies in Theoretical Linguistics, vol. 53. Oxford University Press (2014)

[4] Bastenhof, A.: Polarized classical non-associative Lambek calculus and formal semantics. In: International Conference on Logical Aspects of Computational Linguistics. pp. 33–48. Springer (2011)

[5] Belnap, N.D.: Display logic. Journal of philosophical logic **11**(4), 375–417 (1982)

[6] van Benthem, J.: Categorial grammar and lambda calculus. In: Mathematical logic and its applications, pp. 39–60. Springer (1987)

[7] Bernardi, R., Moortgat, M.: Continuation semantics for the Lambek–Grishin calculus. Information and Computation **208**(5), 397–416 (2010)

[8] Bertot, Y., Castéran, P.: Interactive theorem proving and program development: Coq'Art: the calculus of inductive constructions. Springer Science & Business Media (2013)

[9] Buszkowski, W.: Completeness results for Lambek syntactic calculus. Mathematical Logic Quarterly **32**(1-5), 13–28 (1986)

[10] Buszkowski, W.: On classical nonassociative Lambek calculus. In: International Conference on Logical Aspects of Computational Linguistics. pp. 68–84. Springer (2016)

[11] Buszkowski, W.: On involutive nonassociative Lambek calculus. Journal of Logic, Language and Information **28**(2), 157–181 (2019)

[12] Buszkowski, W., Farulewski, M.: Nonassociative Lambek calculus with additives and context-free languages. In: Languages: From formal to natural, pp. 45–58. Springer (2009)

[13] Chlipala, A.: Parametric higher-order abstract syntax for mechanized semantics. In: Proceedings of the 13th ACM SIGPLAN international conference on Functional programming. pp. 143–156 (2008)

[14] Coquand, T., Dybjer, P.: Intuitionistic model constructions and normalization proofs. Mathematical Structures in Computer Science **7**(1), 75–94 (1997)

[15] De Groote, P., Lamarche, F.: Classical non-associative Lambek calculus. Studia Logica **71**(3), 355–388 (2002)

[16] Došen, K.: Sequent-systems and groupoid models. I. Studia Logica **47**(4), 353–385 (1988)

[17] Dunn, J.M., Zhou, C.: Negation in the context of gaggle theory. Studia Logica **80**(2), 235–264 (2005)

[18] Girard, J.Y.: Linear logic. Theoretical computer science **50**(1), 1–101 (1987)

[19] Girard, J.: A new constructive logic: Classical logic. Mathematical Structures in Computer Science **1**(3), 255–296 (1991)
[20] Greco, G., Richard, V.D., Moortgat, M., Tzimoulis, A.: Lambek-Grishin calculus: focusing, display and full polarization. arXiv preprint arXiv:2011.02895 (2020)
[21] de Groote, P.: The non-associative Lambek calculus with product in polynomial time. In: International Conference on Automated Reasoning with Analytic Tableaux and Related Methods. pp. 128–139. Springer (1999)
[22] Hendriks, H.: Studied flexibility: Categories and types in syntax and semantics. Institute for Logic, Language and Computation (1993)
[23] Herbelin, H., Lee, G.: Forcing-based cut-elimination for Gentzen-style intuitionistic sequent calculus. In: International Workshop on Logic, Language, Information, and Computation. pp. 209–217. Springer (2009)
[24] Ilik, D.: Continuation-passing style models complete for intuitionistic logic. Annals of Pure and Applied Logic **164**(6), 651–662 (2013)
[25] Ilik, D., Lee, G., Herbelin, H.: Kripke models for classical logic. Annals of Pure and Applied Logic **161**(11), 1367–1378 (2010)
[26] Kurtonina, N.: Frames and labels. A modal analysis of categorial inference. Ph.D. thesis, OTS Utrecht University, ILLC Amsterdam University (1995)
[27] Kurtonina, N., Moortgat, M.: Relational semantics for the Lambek-Grishin calculus. In: The Mathematics of Language, pp. 210–222. Springer (2009)
[28] Lambek, J.: The mathematics of sentence structure. The American Mathematical Monthly **65**(3), 154–170 (1958)
[29] Lambek, J.: On the calculus of syntactic types. Structure of language and its mathematical aspects **12**, 166–178 (1961)
[30] Laurent, O.: A proof of the focalization property of linear logic. Unpublished note (2004)
[31] Martin-Löf, P.: About models for intuitionistic type theories and the notion of definitional equality. In: Studies in Logic and the Foundations of Mathematics, vol. 82, pp. 81–109. Elsevier (1975)
[32] Moortgat, M.: Multimodal linguistic inference. Journal of Logic, Language and Information **5**(3), 349–385 (1996)
[33] Moortgat, M.: Categorial type logics. In: Handbook of logic and language, pp. 93–177. Elsevier (1997)
[34] Moortgat, M.: Symmetric categorial grammar: residuation and Galois connections. arXiv preprint arXiv:1008.0170 (2010)
[35] Moot, R., Puite, Q.: Proof nets for the multimodal Lambek calculus. Studia Logica **71**(3), 415–442 (2002)
[36] Okada, M.: Phase semantic cut-elimination and normalization proofs of first-and higher-order linear logic. Theoretical Computer Science **227**(1-2), 333–396 (1999)
[37] Pentus, M.: Product-free Lambek calculus and context-free grammars. The Journal of Symbolic Logic **62**(2), 648–660 (1997)
[38] Pentus, M.: Lambek calculus is NP-complete. Theoretical Computer Science **357**(1-3), 186–201 (2006)

[39] Van Benthem, J.: Language in Action: categories, lambdas and dynamic logic. MIT Press (1995)
[40] Yetter, D.N.: Quantales and (noncommutative) linear logic. The Journal of Symbolic Logic **55**(1), 41–64 (1990)

Computable Functionals of Finite Types in Montague Semantics*

Artem Burnistov[1] and
Alexey Stukachev[1,2]

[1] Novosibirsk State University, Pirogova str. 1, 630090 Novosibirsk, Russia
[2] Sobolev Institute of Mathematics, Acad. Koptyug av. 4, 630090 Novosibirsk, Russia
a.burnistov@g.nsu.ru, aistu@math.nsc.ru
http://www.math.nsc.ru/\~{}stukachev

Abstract. We consider a computable model of functionals of finite types used in Montague semantics to represent grammar categories in natural language sentences. The model is based on the notion of Σ-predicates of finite types in admissible sets introduced by Yu.L.Ershov.

1 Introduction

Type theory and finite-order functionals are essentially and fruitfully used in Montague intensional logic for formalizing the basic grammar categories of natural languages. In Montague original works, as well as in works of other researchers in this area of mathematical linguistics, to the authors' knowledge, the complexity issues and algorithmic aspects of objects and constructions of this theory were not considered so far. The functionals of finite types (i.e., functions, functions on functions, etc.) are complex objects and hard to present constructively in a way which allows possible applications in (for example) ontological models from computational linguistics.

Our research, being purely theoretical and based on notions and methods from generalized computability and Ershov-Scott theory of domains and approximation spaces, is aimed to describe some algorithmic properties of these objects. This work continues publications [9, 10].

In this paper, we present two models of Montague intensional logic. Since they are constructed within the framework of Σ-definability in admissible sets proposed by Yu.L.Ershov, these models can be regarded as effective or computable in a generalized sense. The differences between the models is in the definition of denotation spaces for the basic types of entities and truth values. At first, the simplest possible variant of these spaces is presented. The second model with the ontological space for entities with partial information is briefly described at the end of the paper.

* This work was carried out within the framework of the state contract of the Sobolev Institute of Mathematics (project no. 0314-2019-0003).

2 Basic Notions

2.1 Montague Intensional Logic

Let e, t and s be the some fixed symbols used, correspondingly, as names for basic types of entities and truth values, and for marking an intensional shift, i.e. relativization to a state or situation.

Definition 1. *The set $Types_{IL}$ is defined as follows:*

- $t \in Types_{IL}$, $e \in Types_{IL}$;
- *if* $a \in Types_{IL}$ *and* $b \in Types_{IL}$ *then* $(a \to b) \in Types_{IL}$;
- *if* $a \in Types_{IL}$ *then* $(s \to a) \in Types_{IL}$.

The language of intensional logic IL (see [2, 6, 7]) contains countably many constants of any type $a \in Types_{IL}$ and countably many variables of each type $a \in Types_{IL}$.

A model of intensional logic IL is a quadruple $\langle A, W, T, \leq, F\rangle$ such that A, W, T are nonempty sets, $\leq$ is a linear order on T, F is a function defined on the set of constants of IL as described below. Sets W and T correspond to the set of possible worlds and time moments correspondingly.

Definition 2. *The set D_τ of possible denotations of type $\tau \in Types_{IL}$ is defined by induction on complexity of τ:*

- $D_e = A$, $D_t = \{0, 1\}$;
- $D_{(a\to b)} = D_b^{D_a}$ *(the set of functions from D_a to D_b);*
- $D_{(s\to a)} = D_a^{W\times T}$ *(the set of functions from $W \times T$ to D_a).*

We denote by S_a the set $D_{(s\to a)}$. Function F defines for each constant of type a some element from S_a which is called its *intension*. Elements from D_a are called *extensions* of type a.

Finite types from definition 1 are used to represent grammar categories (parts of speech) of natural languages. Some correspondences between categories and types are listed in Table 1.

For example, proper names correspond to the type $((s \to (e \to t)) \to t)$ – "property of being a property" (or the set of properties true for the individual with this name). Here we do not consider one of the most complex cases, intensional transitive verbs with the type $((s \to ((s \to (e \to t)) \to t)) \to (e \to t)))$.

Extension (the set of denotations) of type a is the set of possible values of the grammar category interpreted by type a in a model of intensional logic. Correspondingly, intension of type a is a function from $W \times T$ to the extension of type a.

2.2 Hereditarily Finite Superstructures

Hereditary finite superstructures are the "simplest" examples of models of theory KPU proposed by S.Kripke, R.Platek, J.Barwise and Yu.L.Ershov for studying generalized computability via Σ-definability in admissible sets (see [1, 5, 11]).

By ω we denote the set of natural numbers. For arbitrary set M, we construct the set $HF(M)$ of hereditarily finite sets over M as follows:

Table 1. Categories and types of some expressions

Category	Grammar equivalent	Corresponding type	Basic expressions
e	no	e	no
t	sentences	t	no
IV	intransitive verbs	(e → t)	walk, talk
CN	common nouns	(e → t)	man, woman
TV	extensional transitive verbs	(e → (e → t))	love, find
CN/CN	extensional adjectives	((e → t) → (e → t))	tall, young
CN/CN	extensional adverbs	((e → t) → (e → t))	rapidly, slowly
T	noun phrases and proper names	((s → (e → t)) → t)	John, ninety, he
t/t	sentence determinants	((s → t) → t)	necessarily, possibly
IV/t	connective verbs	((s → t) → (e → t))	believe, assert

$HF_0(M) = \varnothing$;
$HF_{n+1}(M) = \mathcal{P}_\omega(M \cup HF_n(M))$, $n < \omega$
(here $\mathcal{P}_\omega(X)$ is the set of all finite subsets of X);
$HF(M) = \bigcup\limits_{n<\omega} HF_n(M)$.

If $\mathfrak{M}$ is a structure of some relational signature σ then one can define on $M \cup HF(M)$ a structure $\mathbb{HF}(\mathfrak{M})$ of signature $\sigma' = \sigma \cup \{U, \varnothing, \in\}$ ($U, \varnothing, \in$ are some symbols not in σ) with the following interpretation of signature symbols:

$U^{\mathbb{HF}(\mathfrak{M})} = M$;
$P^{\mathbb{HF}(\mathfrak{M})} = P^{\mathfrak{M}}$, $P \in \sigma$;
$\varnothing^{\mathbb{HF}(\mathfrak{M})} = \varnothing \in HF_0(M)$;
$\in^{\mathbb{HF}(\mathfrak{M})} = \in \cap((M \cup HF(M)) \times HF(M))$.

A class of Δ_0-*formulas* of signature σ' is the least one containing atomic formulas which is closed under $\vee$, $\wedge$, $\to$, $\neg$ and bounded quantifiers $\forall x \in y$ and $\exists x \in y$ ($\forall x \in y\, \varphi$ and $\exists x \in y\, \varphi$ are abbreviations for $\forall x(x \in y \to \varphi)$ and $\exists x(x \in y \wedge \varphi)$ respectively).

A class of Σ-*formulas* of signature σ' is the least one containing Δ_0-formulas and closed under $\vee$, $\wedge$, bounded quantifiers $\forall x \in y$, $\exists x \in y$, and $\exists x$. As usual, a set is called Σ-definable if it is definable by some Σ-formula with parameters, and Δ-definable if it and its complement are Σ-definable.

2.3 Ershov-Scott Functional Spaces

To construct an effective model of Montague intensional logic we apply the domain theory proposed by D.S. Scott [8] and the theory of functional spaces of finite types proposed by Yu.L. Ershov [3, 4, 5]. The definitions below are from [5].

Let $\mathbb{A}$ be a model of KPU (see [5]). If $a \in A$ then $p_l^* a = \{b \,|\, \exists c(\langle b, c\rangle \in a)\}$, $p_r^* a = \{b \,|\, \exists c(\langle c, b\rangle \in a)\}$. If $B \subseteq A$ then $B^* = \{b \,|\, b \subseteq B \text{ and } b \in A\}$.

The notion of effectively presented functional space is based on the general

Definition 3. *Quadruple* $\mathfrak{B} = \langle B, \leq, Cons, \sqcup \rangle$ *is called an* f-base on $\mathbb{A}$ *(see [3, 4]) if the following holds:*

1) B *is a* Δ*-definable subset of* $\mathbb{A}$*;*
2) $\leq$ *is a* Δ*-definable preorder on* B*;*
let $[B]$ *be the quotient of set* B *by the equivalence relation* $\equiv$ *defined by the preorder* $\leq$ *(*$b_0 \equiv b_1 \Leftrightarrow b_0 \leq b_1$ *and* $b_1 \leq b_0$*); as usual,* $[b]$ *denotes the element of* $[B]$ *which is the equivalence class of* $b \in B$*; if* $C \subseteq B$ *then* $[C] = \{[b] \mid b \in C\}$*; we also use* $\leq$ *to denote the preorder induced on* $[B]$ *by the original preorder* $\leq$*;*
3) $Cons$ *is a* Δ*-definable subset of* $B^* \setminus \{\emptyset\}$*, and for any* $b_* \in B^*$ *holds*

$$b_* \in Cons \Leftrightarrow (\exists b \in B)(\forall b^{'} \in b_*)(b^{'} \leq b);$$

4) $\sqcup : Cons \to B$ *is a* Σ*-definable function such that* $[\sqcup b_*]$ *for any* $b_* \in Cons$ *is the least upper bound of* $[b_*] \subseteq [B]$ *in* $\langle [B], \leq \rangle$*.*

Definition 4. *Let* $\mathfrak{B}_0 = \langle B_0, \leq_0, Cons_0, \sqcup_0 \rangle$ *and* $\mathfrak{B}_1 = \langle B_1, \leq_1, Cons_1, \sqcup_1 \rangle$ *be some* f*-bases on* $\mathbb{A}$*. A direct product* $\mathfrak{B}_1 \times \mathfrak{B}_2$ *of* $\mathfrak{B}_0$ *and* $\mathfrak{B}_1$ *is the* f*-base* $\langle B_0 \times B_1, \leq, Cons, \sqcup \rangle$*, where* $\leq$*,* $Cons$ *and* $\sqcup$ *are defined as follows:*

1) $\langle b_0, b_1 \rangle \leq \langle b_0^{'}, b_1^{'} \rangle$ *iff* $b_0 \leq_0 b_0^{'}$ *and* $b_1 \leq_1 b_1^{'}$ *for every* $b_0, b_0^{'} \in B_0$ *and every* $b_1, b_1^{'} \in B_1$*;*
2) $b_* \in Cons$ *iff* $p_l^*(b_*) \in Cons_0$ *and* $p_r^*(b_*) \in Cons_1$ *for every* $b_* \in (B_0 \times B_1)^*$*;*
3) $\sqcup b_* \leftrightharpoons \langle \sqcup_0 p_l^*(b_*), \sqcup_1 p_r^*(b_*) \rangle$ *for every* $b_* \in Cons$*.*

In case the set $Cons$ of mutually consistent fragments (approximations) should be as large as possible, we need

Definition 5. *Quadruple* $\mathfrak{B} = \langle B, b_0, \leq, \sqcup \rangle$ *is called an* f^*-base on $\mathbb{A}$ *if* $\langle B, \leq, B^* \setminus \{\emptyset\}, \sqcup \rangle$ *is an* f*-base on* $\mathbb{A}$*,* $[b_0]$ *is the least element in* $\langle [B], \leq \rangle$ *and* $\sqcup \emptyset = b_0$*.*

In general, the range (the set of possible values) of a functional can be arbitrary, so the notion of f^*-base is used in the following

Definition 6. *Let* $\mathfrak{B}_0 = \langle B_0, \leq_0, Cons_0, \sqcup_0 \rangle$ *be an* f*-base,*
$\mathfrak{B}_1 = \langle B_1, b_1, \leq_1, \sqcup_1 \rangle$ *be an* f^**-base. A functional product* $F(\mathfrak{B}_0, \mathfrak{B}_1)$ *of* f*-base* $\mathfrak{B}_0$ *and* f^**-base* $\mathfrak{B}_1$ *is the* f^**-base* $\langle (B_0 \times B_1)^*, \emptyset, \leq, \sqcup \rangle$*, where* $\leq$ *and* $\sqcup$ *are defined as follows:*

1) $f_0 \leq f_1$ *iff* $\forall b_0 \in p_l^* f_0 (\sqcup_1 \{b_1 \mid \exists b_0^{'} \in p_l^* f_0 (b_0^{'} \leq_0 b_0$ *and* $\langle b_0^{'}, b_1 \rangle \in f_0)\} \leq_1$
$\leq_1 \sqcup_1 \{b_1 \mid \exists b_0^{'} \in p_l^{'} f_1 (b_0^{'} \leq_0 b_0$ *and* $\langle b_0^{'}, b_1 \rangle \in f_1)\})$ *for* $f_0, f_1 \in (B_0 \times B_1)^*$*;*
2) $\sqcup f_* \leftrightharpoons \cup f_*$ *for every* $f_* \in ((B_0 \times B_1)^*)^*$*.*

Definition 7. *For an* f*-base* $\mathfrak{B} = \langle B, \leq, Cons, \sqcup \rangle$*, the family* $I_\Sigma(\mathfrak{B})$ *of* Σ*-ideals in* $\mathfrak{B}$ *consists of nonempty* Σ*-definable subsets* $C \subseteq B$ *such that*

1) from $c \in C, b \in B, b \leq c$ *it follows that* $b \in C$*;*

2) from $c \in C^*$ it follows that $c \in Cons$ and $\sqcup c \in C$.

We define a topology on the set $I_\Sigma(\mathfrak{B})$ by fixing the basis

$$V_b \rightleftharpoons \{C \mid C \in I_\Sigma(\mathfrak{B}), b \in C\}, b \in B.$$

The set $I_\Sigma(\mathfrak{B})$ together with the topology specified above is called the *space of Σ-ideals* of f-base $\mathfrak{B}$. The space $I_\Sigma(\mathfrak{B})$ is a topological T_0-space.

Let $\mathfrak{B}_0$ be an f-base and let $\mathfrak{B}_1$ be an f^*-base. For any ideal I of f-base $F(\mathfrak{B}_0, \mathfrak{B}_1)$ we can define the continuous function $f_I : I_\Sigma(\mathfrak{B}_0) \to I_\Sigma(\mathfrak{B}_1)$ as follows. Let $I_0 \in I_\Sigma(\mathfrak{B}_0)$. We define

$$f_I(I_0) \rightleftharpoons \{b_1 \mid b_1 \in B_1, (\exists c^* \in I)(\exists b_0 \in I_0)\exists b_1^{'}(b_1 \leq_1 b_1^{'} \text{ and } \langle b_0, b_1^{'}\rangle \in c^*)\}.$$

If $\{\langle b_0, b_1\rangle\} \in I, b_0 \in I_0$, then $b_1 \in f_I(I_0)$.

The mapping $I \to f_I$ from $I_\Sigma(F(\mathfrak{B}_0, \mathfrak{B}_1))$ to $C(I_\Sigma(\mathfrak{B}_0), I_\Sigma(\mathfrak{B}_1))$ (the set of all continuous functions from the space $I_\Sigma(\mathfrak{B}_0)$ to the space $I_\Sigma(\mathfrak{B}_1)$) is injective.

To introduce the simplest example of spaces for entities and truth values, let $\mathfrak{A} = \langle A, =, P_1(A), \cup \restriction P_1(A)\rangle$, where $P_1(A) \rightleftharpoons \{\{a\} \mid a \in A\}$. This quadruple is an f-base with $I_\Sigma(\mathfrak{A}) = P_1(A)$. Also, let α be an arbitrary ordinal in $\mathbb{A}$ and let $\mathfrak{B}_\alpha = \langle \alpha, \emptyset, \subseteq, \cup \restriction \alpha\rangle$. This quadruple is an f^*-base with $I_\Sigma(\mathfrak{B}_\alpha) = (\alpha + 1) \setminus \emptyset$. Further on we consider the case $\alpha = 2$.

Definition 8. *The set of functional types $Types_f$ together with its proper subset $PTypes_f$ are defined as follows:*

1) $\boldsymbol{o} \in Types_f \setminus PTypes_t, B \in PTypes_f \subseteq Types_f$;
2) if $\tau_0, \tau_1 \in Types_f(PTypes_f)$ then $(\tau_0 \times \tau_1) \in Types_f(PTypes_f)$;
3) if $\tau_0 \in Types_f, \tau_1 \in PTypes_f$ then $(\tau_0 \to \tau_1) \in PTypes_f$.

Definition 9. *For every type $\tau \in Types_f$, the f-base $\mathcal{F}_\tau$ is defined by induction on the complexity of τ:*

1) $\mathcal{F}_{\boldsymbol{o}} \rightleftharpoons \mathfrak{A}$, $\mathcal{F}_B \rightleftharpoons \mathfrak{B}_2$;
2) $\mathcal{F}_{(\tau_0 \times \tau_1)} \rightleftharpoons \mathcal{F}_{\tau_0} \times \mathcal{F}_{\tau_1}$;
3) $\mathcal{F}_{(\tau_0 \to \tau_1)} \rightleftharpoons F(\mathcal{F}_{\tau_0}, \mathcal{F}_{\tau_1})$.

If $\tau \in PTypes_f$ then $\mathcal{F}_\tau$ is an f^*-base.

Definition 10. *By a Σ-predicate of type $\tau \in Types_f$ on A we mean an arbitrary element of $I_\Sigma(\mathcal{F}_\tau)$.*

The propositions below easily follow from the definitions. Here $\Sigma(A)$ denotes the set of all Σ-definable subsets of $\mathbb{A}$.

Lemma 1. *For any $n > 0$ there is a natural bijective correspondence between Σ-predicates of type $\boldsymbol{o}^n \to B$ and n-ary Σ-predicates on A.*

Proposition 1. *A mapping $F : \Sigma(A) \to \Sigma(A)$ is a restriction of a Σ-operator if and only if F is continuous with respect to the strong topology and there is a Σ-function $f : A \to A$ such that $F(Q_{u,a}) = Q_{u,f(a)}$ for all $a \in A$.*

Proposition 2. *For a family $S \subseteq \Sigma(A)$ the following are equivalent:*

1. *S is represented by a Σ-predicate of type $((\boldsymbol{o} \to B) \to B)$;*
2. *there is a Σ-formula $\Phi(P^+)$ of signature $\sigma \cup \langle P^1 \rangle$ such that*

$$S = \{Q \mid Q \in \Sigma(A), \langle \mathbb{A}, Q \rangle \vDash \Phi(P)\}.$$

Proposition 3. *There is a natural bijective correspondence between Σ-predicates of type $((\boldsymbol{o} \to B) \to (\boldsymbol{o} \to B))$ and unary Σ-operators.*

3 Σ-predicates of Finite Types in $\mathbb{HF}(\mathbb{R})$ and Intensional Logic

For arbitrary model $\mathbb{A}$ of KPU, the simplest example of spaces for entities and truth values are the f-base

$$\mathfrak{A} = \langle A, =, P_1(A), \cup \restriction P_1(A) \rangle$$

and the f^*-base

$$\mathfrak{B}_2 = \langle 2, \emptyset, \subseteq, \cup \restriction 2 \rangle.$$

Further on, we restrict ourselves to the case $\mathbb{A} = \mathbb{HF}(\mathbb{R})$, where $\mathbb{R}$ is the ordered field of real numbers. This choice is motivated by the fact that the reals are very natural to use for representing the scale of time and for coding. Let $W = P_1(\mathbb{R})$. We also fix the f-base

$$\mathfrak{W} = \langle W, =, P_1(W), \cup \restriction P_1(W) \rangle.$$

The set of ideals is defined to be the set of singletons from W, i.e. $P_1(P_1(\mathbb{R}))$.

Singletons from $\mathbb{HF}(\mathbb{R})$ (Σ-ideals of f-base $\mathfrak{A}$) correspond to basic entities, objects of type e. Real numbers (urelements) represent possible worlds. In particular, each possible world can be considered as a substructure of the whole structure with some partial information about the universe.

We give analogues of definition 8 and 9.

Definition 11. *The set $Types$ together with its proper subset $PTypes$ are defined as follows:*

1) $e \in Types \setminus PTypes$, $t \in PTypes$;
2) if $a \in Types$ and $b \in PTypes$, then $(a \to b) \in PTypes$;
3) if $a \in PTypes$ then $(s \to a) \in PTypes$.

Definition 12. *For every $a \in Types$, the f-base $\mathcal{F}_a$ is defined as follows:*

1) $\mathcal{F}_e \leftrightharpoons \mathfrak{A}$, $\mathcal{F}_t \leftrightharpoons \mathfrak{B}_2$;
2) $\mathcal{F}_{(a \to b)} \leftrightharpoons F(\mathcal{F}_a, \mathcal{F}_b)$;
3) $\mathcal{F}_{(s \to a)} \leftrightharpoons F(\mathfrak{W}, \mathcal{F}_a)$.

For every type $a \in Types$, Σ-predicates of this type are defined to Definition 10.

Definition 13. *The set of possible denotations D_τ of type $\tau \in Types$ is the set of all Σ-predicates of type τ.*

Note that $\mathfrak{A}$ is not precisely an f^*-base (there is no minimal element in $\mathfrak{A}$, but A^* can be considered as $Cons$). Nevertheless, we define a functional product

$$\mathcal{F}_{(s \to e)} \leftrightharpoons F(\mathfrak{W}, \mathcal{F}_e).$$

Since in $\mathbb{HF}(\mathbb{R})$ holds the uniformization property, for every Σ-ideal I of type $(s \to e)$ there is a corresponding (partial) Σ-function $f_I : P_1(\mathbb{R}) \to A$. Hence the following is true:

Lemma 2. *There is a natural bijective correspondence between Σ-predicates of type $(s \to e)$ and Σ-functions from $P_1(\mathbb{R})$ to A.*

As a corollary of Lemma 1 and Lemma 2 we get the following

Lemma 3. *There is a natural bijective correspondence between Σ-predicates of type $(s \to t)$ and unary Σ-predicates on $P_1(\mathbb{R})$.*

Proposition 4. *For a family $S \subseteq \Sigma(P_1(\mathbb{R}))$ the following are equivalent:*

1. *S is represented by a Σ-predicate of type $((s \to t) \to t)$;*
2. *there exists Σ-formula $\Phi(P^+)$ of signature $\sigma \cup \langle P^1 \rangle$ such that*

$$S = \{Q \,|\, Q \in \Sigma(P_1(\mathbb{R})), \langle \mathbb{A}, Q \rangle \models \Phi(P^+)\}.$$

We obtain the correspondence for type $((s \to t) \to (e \to t))$ analogous to Proposition 3.

Lemma 4. *There is a natural bijective correspondence between Σ-predicates of type $((s \to t) \to (e \to t))$ and unary Σ-operators on A.*

Proof. If $I \in I_\Sigma(F(\mathfrak{W}, \mathfrak{B}_2))$, by Lemma 3 we get that I can be uniquely constructed from some $Q \in \Sigma(P_1(\mathbb{R}))$. We denote $I = I_Q$ and assume that elements from $I_\Sigma(F(\mathfrak{W}, \mathfrak{B}_2))$ are of the form I_Q for a suitable $Q \in \Sigma(P_1(\mathbb{R}))$. In the same way (by Lemma 1), elements from $I_\Sigma(F(\mathfrak{A}, \mathfrak{B}_2))$ are of the form I_Q for a suitable $Q \in \Sigma(A)$.

With any Σ-predicate I of type $((s \to t) \to (e \to t))$, a continuous mapping $f_I : I_\Sigma(F(\mathfrak{W}, \mathfrak{B}_2)) \to I_\Sigma(F(\mathfrak{A}, \mathfrak{B}_2))$ is naturally associated. Hence for f_I there is a unique mapping $F_I : \Sigma(P_1(\mathbb{R})) \to \Sigma(A)$ such that for any $Q \in \Sigma(P_1(\mathbb{R}))$ holds $f_I(I'_Q) = I'_{F_I(Q)}$ for $I'_Q \in I_\Sigma(F(\mathfrak{W}, \mathfrak{B}_2))$ and for $I'_{F_I(Q)} \in I_\Sigma(F(\mathfrak{A}, \mathfrak{B}_2))$.

Since F_I is continuous with respect to weak topology because of the continuity of f_I, it is enough to show that the set

$$\Gamma^*_{F_I} = \{\langle a, b \rangle \,|\, a \in A^*, b \in F_I(a)\}$$

is a Σ-subset.

For arbitrary $a \in A^*$, we have

$$b \in F_I(a) \Leftrightarrow \exists c(c \in I'_{F_I(a)} \ \& \ \langle b, 1\rangle \in c),$$

where

$$c \in I'_{F_I(a)} \Leftrightarrow \exists c^* \exists b_0 \exists b_1 (\forall x \in b_0 (\exists a_0 (x = \langle a_0, 0\rangle) \ \vee \ \exists a_1 (a_1 \in a \text{ and}$$

$$\text{and } x = \langle a_1, 1\rangle)) \text{ and } c \leq b_1 \text{ and } \langle b_0, b_1\rangle \in c^*).$$

Hence $I^*_{F_I}$ is a Σ-subset and F_I is a restriction of some Σ-operator.

On the other hand, if $F : P(A) \to P(A)$ is a Σ-operator then from the bijective correspondence between the ideals of f^*-base $F(F(\mathfrak{W}, \mathfrak{B}_2), F(\mathfrak{A}, \mathfrak{B}_2))$ and continuous mappings from $I_\Sigma(F(\mathfrak{W}, \mathfrak{B}_2))$ to $I_\Sigma(F(\mathfrak{A}, \mathfrak{B}_2))$, an operator F (more exactly, its restriction on $P_1(\mathbb{R})$) uniquely determines the ideal I_F such that for all $Q \in \Sigma(P_1(\mathbb{R}))$ holds $f_{I_F}(I'_Q) = I'_{F(Q)}$.

Now we describe Σ-predicates of type $(s \to (e \to t))$. Let $\Phi(x, a)$ be a Σ-formula with parameters, and let $Q \subseteq A$. We connect with Φ and with set Q the family of subsets $S^Q_\Phi = \{\Phi^{\mathbb{A}}(x, a) \,|\, a \in Q\}$. The set Q is called the set of indices of S and denoted by $Ind(S)$. If possible, we will omit indices Q and Φ.

Lemma 5. *There is a natural bijective correspondence between Σ-predicates of type $(s \to (e \to t))$ and families $S^{Ind(S)}_\Phi$ of Σ-subsets defined by Σ-formulas with parameters, such that $Ind(S) \subseteq P_1(\mathbb{R})$ is a Σ-subset.*

Proof. Let I be a Σ-predicate of type $(s \to (e \to t))$, $I \subseteq (P_1(\mathbb{R}) \times (A \times 2)^*)^*$. Let

$$\Phi(x, a) \leftrightharpoons (\exists f \in I)(\exists f' \in f)(p_l f' = a \text{ and } \langle x, 1\rangle \in p_r f'),$$

$$\Psi(x) \leftrightharpoons (\exists f \in I)(\exists f' \in f)(p_l f' = x \text{ and } \exists y(\langle y, 1\rangle \in p_r f')).$$

We define $Q = \Psi^{\mathbb{A}}(x)$, $S_I = \{\Phi^{\mathbb{A}}(x, w) \,|\, w \in Q\}$. Then $S_I = S^Q_\Phi$ is a family of Σ-subsets which is defined by a Σ-formula $\Phi(x, a)$ with a parameter, which set of indices $Ind(S)(= Q) \subseteq P_1(R)$ is a Σ-subset.

Let $S^{Ind(S)}_\Phi$ be a family of Σ-subsets which is defined by a Σ-formula $\Phi(x, a)$ with a parameter, which set of indices $Ind(S) \subseteq P_1(\mathbb{R})$ is a Σ-subset. Let

$$I_S = (\{\langle w, \{\langle a, 0\rangle \,|\, a \in A\}^*\rangle \,|\, w \in P_1(\mathbb{R})\} \bigcup$$

$$\bigcup \{\langle w, (\{\langle a, 0\rangle \,|\, a \in A\} \cup \{\langle a, 1\rangle \,|\, a \in \Phi^{\mathbb{A}}(x, w)\})^*\rangle \,|\, w \in Ind(S)\})^*.$$

We prove that this is a Σ-ideal. It is clear that $I_S \subseteq (P_1(\mathbb{R}) \times (A \times 2)^*)^*$ and that condition 2 for ideals is satisfied. Check condition 1: let $f \in I_S$, $g \in (P_1(\mathbb{R}) \times (A \times 2)^*)^*$ and $g \leq f$. We need to show that in that case $g \in I_S$. For this, it is enough to show that

$$g \subseteq \{\langle w, \{\langle a, 0\rangle \,|\, a \in A\}^*\rangle \,|\, w \in P_1(\mathbb{R})\} \bigcup$$

$$\bigcup\{\langle w, (\{\langle a,0\rangle \mid a \in A\} \cup \{\langle a,1\rangle \mid a \in \Phi^{\mathbb{A}}(x,w)\})^*\rangle \mid w \in Ind(S)\}.$$

Let $w \notin Ind(S)$ and there is an a_0 such that $\langle w, a_0\rangle \in g$. Since $g \leq f$ (and by the definition of I_S), from $\langle a_1, \epsilon\rangle \in a_0$ it follows that $\epsilon = 0$. Hence

$$\langle w, a_0\rangle \in \{\langle w, \{\langle a,0\rangle \mid a \in A\}^*\rangle \mid w \in P_1(\mathbb{R})\}.$$

Let $w \in Ind(S)$ and there is an a_0 such that $\langle w, a_0\rangle \in g$. Let $\langle a_1, \epsilon\rangle \in a_0$. If $\epsilon = 0$ then $\langle a_1, \epsilon\rangle \in \{\langle a,0\rangle \mid a \in A\}$. On the other hand, if $\epsilon = 1$, then since $g \leq f$ there exists b such that $\langle w, b\rangle \in f$ and $\langle a_1, 1\rangle \in b$. By the definition of I_S we get that $\langle a_1, \epsilon\rangle \in \{\langle a,1\rangle \mid a \in \Phi^{\mathbb{A}}(x,w)\}$. Hence

$$g \subseteq \{\langle w, \{\langle a,0\rangle \mid a \in A\}^*\rangle \mid w \in P_1(\mathbb{R})\} \bigcup$$

$$\bigcup\{\langle w, (\{\langle a,0\rangle \mid a \in A\} \cup \{\langle a,1\rangle \mid a \in \Phi^{\mathbb{A}}(x,w)\})^*\rangle \mid w \in Ind(S)\}$$

and $g \in I_S$.

Using the correspondences described above we can get for any family $S_\Phi^{Ind(S)}$ of Σ-subsets that $S_\Phi^{Ind(S)} = S_{I_S}$. On reverse, for any Σ-predicate I of type $(s \to (e \to t))$ we get that $I = I_{S_I}$.

Corollary 1. *There is a natural bijective correspondence between Σ-predicates of type $(e \to (e \to t))$ and families $S_\Phi^{Ind(S)}$ of Σ-subsets defined by Σ-formulas with parameters such that $Ind(S) \subseteq A$ is a Σ-subset.*

Remark: objects of type $(s \to (e \to t))$ and $(e \to (e \to t))$ can also be considered as binary Σ-predicates (in case of type $(s \to (e \to t))$ they are subsets of $P_1(\mathbb{R}) \times A$). This is equivalent to the correspondence described above and will be also convenient to use.

The set of all families described in Proposition 5 is denoted by $\Sigma'(A)$.

Using the proof of Proposition 2 we describe Σ-predicates of type $((s \to (e \to t)) \to t)$: they correspond to definable families of families of Σ-subsets obtained above.

We use the following notation: if $S_\Phi^{Ind(S)} \in \Sigma'(A)$ then

$$C_S = \{\langle w, a\rangle : w \in Ind(S), a \in (\Phi^{\mathbb{A}}(x,w))^*\}.$$

Lemma 6. *For a family $K \subseteq \Sigma'(A)$ the following are equivalent:*

1. *K corresponds to a Σ-predicate of type $((s \to (e \to t)) \to t)$;*
2. *there is a Σ-formula $\Phi(P^+)$ of signature $\sigma \cup \langle P^1\rangle$ such that*

$$K = \{S \mid S \in \Sigma'(A), \langle \mathbb{A}, C_S\rangle \models \Phi(P^+)\}.$$

Proof. Let I be a Σ-predicate of type $((s \to (e \to t)) \to t)$, $I \subseteq ((P_1(\mathbb{R}) \times (A \times \times 2)^*)^* \times 2)^*$. We connect with I the family

$$K_I = \{S \mid S \subseteq \Sigma'(A), f_I(I_S) = 2\}$$

(here I_S is the Σ-predicate from Proposition 5).

$(1 \Rightarrow 2)$. Let

$$K = \{S \mid S \subseteq \Sigma^{'}(A), \langle \mathbb{A}, C_S \rangle \vDash \Phi(P^+)\},$$

where

$$\Phi(P^+) \leftrightharpoons \exists x \exists y \exists z [x \in I \text{ and } \langle y, 1\rangle \in x \text{ and } (\forall w \in p_l^* y)(\forall a \in p_r^* y)(\langle w, a\rangle \in y \rightarrow$$

$$\rightarrow \exists z^{'} \subseteq z (a = z^{'} \times \{1\} \text{ and } P(\langle w, z^{'}\rangle)))].$$

We prove that $K = K_I$.

$(\subseteq)$. If $S \in K$ then by the definition there are $x \in I$, $y \in (P_1(\mathbb{R}) \times (A \times 2)^*)^*$ and $z \in A$ such that $\langle y, 1\rangle \in x$ and from $\langle w, a\rangle \in y$ follows that $a = z^{'} \times 1$ for some $z^{'} \subseteq z$. Besides, for all such pairs we have $P(\langle w, z^{'}\rangle)$, hence $y \in I_S$. By the definition of mapping f_I we get $1 \in f_I(I_S)$, and hence $S \in K_I$.

$(\supseteq)$ Let $S \in K_I$. Then $f_I(I_S) = 2$ and there are $c^* \in I$, $b_0 \in I_S$ such that $\langle b_0, 1\rangle \in I$. Define the sets b^w and b as follows:

$$b^w = \{a \mid \exists a_0 (\langle w, a_0\rangle \in b_0 \text{ and } \langle a, 1\rangle \in a_0)\}$$

for $w \in p_l^* b_0$,

$$b = \{\{w\} \times (b^w \times \{1\}) \mid w \in p_l^* b_0 \text{ and } b^w \neq \emptyset\}.$$

Let us show that $\{\langle b, 1\rangle\} \in I$. Since $\{\langle b_2, 1\rangle\} \leq \{\langle b_2, 1\rangle\} \cup c*$, it is enough to show that $\{\langle b_2, 1\rangle\} \cup c^* \in I$. We show that $\{\langle b_2, 1\rangle\} \cup c^* \leq c^*$. Since $\langle b_0, 1\rangle \in c^*$, it is enough to find a pair $\langle b_1, 1\rangle \in c^*$ such that $b_1 \leq b$ (the orders belong to corresponding f-bases). We can take $b_1 = b_0$: if for some a, a_0 it is true that $\langle w, a\rangle \in b_0$ and $\langle a_0, 1\rangle \in a$, by the definition of $a_0 \in b^w$ there exists c such that $\langle w, c\rangle \in b$ and $\langle a_0, 1\rangle \in c$. Hence $b_0 \leq b$ (and also $b \leq b_0$).

So, $\{\langle b, 1\rangle\} \in I$. Now take

$$x = \{\langle b, 1\rangle\},\ y = b,\ z = \bigcup_{w \in p_l^* b_0} b^w.$$

Then in $\langle \mathbb{A}, C_S\rangle$ it is true that

$$x \in I \text{ and } \langle y, 1\rangle \in x \text{ and } (\forall w \in p_l^* y)(\forall a \in p_r^* y)(\langle w, a\rangle \in y \rightarrow$$

$$\rightarrow \exists z^{'} \subseteq z (a = z^{'} \times \{1\} \text{ and } P(\langle w, z^{'}\rangle))).$$

Hence $S \in K$.

$(2 \Rightarrow 1)$. Let $K = \{S \mid S \subseteq \Sigma^{'}(A), \langle \mathbb{A}, C_S\rangle \vDash \Phi(P^+)\}$ for some Σ-formula $\Phi(P^+)$. Let $\Phi^*(x) = [\Phi(P^+)]_x^P$ (this means that all atomic subformulas of kind $P(y)$ are substituted by $y \in x$). Prove first that

$$K = \{S \mid S \subseteq \Sigma^{'}(A), \exists a (\Phi^*(a) \text{ and } a \subseteq C_S)\}.$$

Let $\Psi(a, P^+) = (\Phi(P^+)\ \&\ a = \emptyset)$. Consider the operator

$$\Gamma_\Psi(Q) = \{a \in A \mid \langle \mathbb{A}, S\rangle \vDash \Psi(a, P^+)\}.$$

For the corresponding Σ-operator F_Ψ such that for every $Q \subseteq \Sigma(A)$ holds $\Gamma_\Psi(Q) = F_\Psi(Q)$, we have

$$F_\Psi(Q) = \{a \in A \mid \exists y(y \subseteq Q \text{ and } \Psi(a, y)\}.$$

Hence for F_Ψ we have $S \in K \Leftrightarrow F_\Psi(C_S) = \{\emptyset\}$, and so

$$K = \{S \mid S \subseteq \Sigma^{'}(A), \exists a(\Phi^*(a) \text{ and } a \subseteq C_S\}.$$

Now define the ideal I as follows:

$$I = \{c \in ((P_1(R) \times (A \times 2)^*)^* \times 2)^* \mid \exists a(\Phi^*(a) \wedge$$

$$\wedge c \leq \{\langle\{\langle w, b \times \{1\}\rangle \mid \langle w, b\rangle \in a\}, 1\rangle\})\}$$

(it is easy to check it is an ideal). We show that $K = K_I$.

($\subseteq$). Let $S \in K$. Then there is an a such that $a \subseteq C_S$ and $\Phi^*(a)$. Let

$$c^* = \{\langle\{\langle w, b \times \{1\}\rangle \mid \langle w, b\rangle \in a\}, 1\rangle\}.$$

Then $c^* \in I$, but

$$b_0 = \{\langle w, b \times \{1\}\rangle \mid \langle w, b\rangle \in a\} \in I_S,$$

hence there are $c^* \in I$ and $b_0 \in I_S$ such that $\langle b_0, 1\rangle \in c^*$. Henceforth, $f_I(I_S) = 2$ and $S \in K_I$.

($\supseteq$). Let $S \in K_I$. Then $f_I(I_S) = 2$ and there are $c^* \in I$ and $b_0 \in I_S$ such that $\langle b_0, 1\rangle \in c^*$. Since $c^* \in I$, there is an a such that $\Phi^*(a)$ and

$$c^* \leq \{\langle\{\langle w, b \times \{1\}\rangle \mid \langle w, b\rangle \in a\}, 1\rangle\} (= B).$$

Since B is a singleton and $\langle b_0, 1\rangle \in c^*$, we have

$$\{\langle w, b \times \{1\}\rangle \mid \langle w, b\rangle \in a\} \leq b_0.$$

Hence

$$\{\langle w, b \times \{1\}\rangle \mid \langle w, b\rangle \in a\} \in I_S$$

and $a \subseteq C_S$, that means that $S \in K$.

All these correspondences are summarized in Table 2.

Table 2. Intensional Logic Types and $\mathbb{HF}(\mathbb{R})$

Category	Grammar equivalent	Type	Object in $\mathbb{HF}(\mathbb{R})$
e	no	e	sets $\{a\}$ for $a \in \mathbb{HF}(\mathbb{R})$
t	sentenses	t	no
IV	intransitive verbs	$(e \to t)$	unary Σ-predicates
CN	common nouns	$(e \to t)$	unary Σ-predicates
TV	existential transitive verbs	$(e \to (e \to t))$	binary Σ-operators
CN/CN	existential adjectives	$((e \to t) \to (e \to t))$	Σ-operators
CN/CN	existential adverbs	$((e \to t) \to (e \to t))$	Σ-predicates
T	noun phrases and proper names	$((s \to (e \to t)) \to t)$	Σ-definable families of binary Σ-predicates
t/t	sentence determiners	$((s \to t) \to t)$	Σ-definable families of Σ-predicates on $P_1(\mathbb{R})$
IV/t	connective verbs	$((s \to t) \to (e \to t))$	Σ-operators

3.1 Analysis of Natural Language Sentences

Consider some simple examples of English sentences by means of Σ-predicates of finite types. Truth value of any sentence is regarded relative to a possible world $w \in P_1(\mathbb{R})$. However, we consider the simplest absolute case first.

According to Table 1, the proper name **John** has the type $((s \to (e \to t)) \to t)$. But the semantics of the sentence

(1) **John walks**.

is the same in case we consider **John** as an object of type e and in case we consider it as an object of type $((s \to (e \to t)) \to t)$. From the grammar point of view the latter case is more appropriate: in sentences of kind "subject + predicate" the subject is regarded as a functor which receives a predicate as an argument. This is not so when **John** is considered as an object of type e.

Yet consider **John** as an object of type e (a set $\{j\}$ for some $j \in \mathbb{HF}(\mathbb{R})$), and (intransitive) verb **walk** as an object of type $(e \to t)$ (unary Σ-predicate **walk'**). The truth value of this sentence is equivalent to the truth value of Σ-formula

$$\{j\} \in \textbf{walk'}$$

in $\mathbb{HF}(\mathbb{R})$.

Consider now the sentence

(2) **John loves Mary**.

As in the previous case, names **John** and **Mary** are considered as objets of type e ($\{j\}$ and $\{m\}$ respectively). Transitive verb **love** is considered as an object of type $(e \to (e \to t))$, hence it is interpreted by some binary Σ-predicate **love'**. Hence the truth value of this sentence is equivalent to the truth value of Σ-formula

$$\langle \{j\}, \{m\} \rangle \in \textbf{love'}$$

in $\mathbb{HF}(\mathbb{R})$.

In addition, consider the quantified sentence

(3) **Every fish walks**.

The common noun **fish** is an object of type $(e \to t)$, that is, the set of all fishes, ad is denoted by **fish'**. The quantifier can be regarded from the usual point of view in first-order logic: the truth value of sentence **Every fish walks** is equivalent to the truth value of formula

$$\forall x(x \in \textbf{fish'} \to x \in \textbf{walk'})$$

in $\mathbb{HF}(\mathbb{R})$. This, however, is not a Σ-formula. But from the point of the complexity, checking the truth this formula is a finite search through the domain used by the model.

Now we relativize our model via the concept of possible worlds and consider the truth values of sentences (1) and (2) in a given possible world $w \in P_1(\mathbb{R})$. First we look at sentence (1). Previous interpretations for **John** and **walk** remain the same, but now for checking the truth value it becomes necessary to connect with objects j and **walk'** their intensions, i.e., objects of types $(s \to e)$ and$(s \to (e \to t))$. These are, correspondingly, a Σ-function $\hat{}j : P_1(\mathbb{R}) \to HF(\mathbb{R})$ and a binary Σ-predicate $\hat{}\textbf{walk'} \subseteq P_1(\mathbb{R}) \times (HF(\mathbb{R}) \cup \mathbb{R})$. The truth value of this sentence in world w is equivalent to the truth value of Σ-formula (with a parameter)

$$\langle w, \hat{}j(w)\rangle \in \hat{}\textbf{walk'}$$

in $\mathbb{HF}(\mathbb{R})$.

In the same way, the truth value of sentence (2) in possible world w is equivalent to the truth value of Σ-formula (with a parameter)

$$\langle w, \hat{}j(w), \hat{}m(w)\rangle \in \hat{}\textbf{love'}$$

in $\mathbb{HF}(\mathbb{R})$.

4 Ontological Models

In models described above, the space D_t of truth values consists of elements 0 and 1 with natural ordering $(0 < 1)$. The intuition is that 0 means "the property is not true now but probably will became true in the future", while "1" means "the property true now and forever". One can consider a different type of models of intensional logic. The main difference is that these models use other spaces of entities and truth values. Let

$$D_t = \{0, 1, \bot, \top\}$$

and the ordering is defined as follows: $\bot < 0$, $\bot < 1$, $0 < \top$, $1 < \top$ while 0 and 1 are uncomparable. Intuitively, 0, 1 and $\bot$ correspond to *no, yes*, and

unknown. The element $\top$ corresponds to inconsistency of data and is necessary for constructing f^*-spaces. We set

$$D_e = (\mathbb{R} \cup \{\bot\})^{<\omega}.$$

In particular, $(D_t)^{<\omega} \subseteq D_e$. Intuitively, every element from D_e is interpreted as a tuple (sequence) of properties of this element. Properties can be discrete (in our case, binary) and continuous, i.e. described by some measure. We assume that even positions of elements from D_e correspond to binary properties (the values in these positions belong to D_t and compared via the corresponding ordering), while odd positions correspond to continuous properties (the values in these positions belong to $\mathbb{R} \cup \{\bot\}$ and compared as follows: $\bot < a$ for all $a \in \mathbb{R}$, a and b are uncomparable for $a, b \in \mathbb{R}$, $a \neq b$).

The ordering on D_e is defined as follows: for $\alpha, \beta \in D_e$, $\alpha \leqslant \beta$ if and only if the length of the tuple α is less or equal to the length of the tuple β and $\alpha(i) \leqslant \beta(i)$ for all $i \leqslant lh(\alpha)$.

As in models described previously, it is possible to interprete by Σ-definable objects in $\mathbb{HF}(\mathbb{R})$ the corresponding objects (types) of intensional logic. For example, a common noun "man" (of type $(e \to t)$) can be interpreted by Σ-predicate $(\alpha(human) = 1) \wedge (\alpha(gender) = "M'')$ for some fixed in this model positions $human, gender \in \omega$. In the same way, an adjective "tall" (type $(e \to t) \to (e \to t)$) can be interpreted by Σ-operator H such that, for example, $\alpha \in H(\text{man}) \iff \alpha(height) \geqslant 180$, $\alpha \in H(\text{woman}) \iff \alpha(height) \geqslant 175$, $\alpha \in H(\text{chair}) \iff \alpha(height) \geqslant 120$ Again, $height \in \omega$ is some fixed position.

In our version of ontological models the scale of time is identified with the ordered set of real numbers: $T = \mathbb{R}$. The set of possible worlds is identified with the set of natural numbers: $W = \mathbb{N}$. Entities of world n are identified with the tuples of length $n + 1$. The moment of time in which a given entity is described by the corresponding tuples is identified with the value of the leftmost position of these tuples.

The detailed description of this type of models can be given in the same way as before and will be presented in a subsequent publication.

Bibliography

[1] Barwise, J.: Admissible Sets and Structures. Springer Verlag, Heidelberg (1975)

[2] Dowty, D.: Introduction to Montague Semantics. D. Reidel Publishing Company, Dodrecht (1989)

[3] Ershov, Y.: The theory of A-spaces. Algebra and Logic **12**(4), 209–232 (1973). https://doi.org/http://dx.doi.org/10.1007/BF02218570, 10.1007/BF02218570

[4] Ershov, Y.: Theory of domains and nearby. Formal Methods in Programming and Their Applications. Lecture Notes in Computer Science **735**, 1–7 (1993). https://doi.org/10.1007/BFb0039696, http://dx.doi.org/10.1007/BFb0039696

[5] Ershov, Y.: Definability and Computability. Plenum, New York (1996)

[6] Montague, R.: The proper treatment of quantification in ordinary english. In: Hintikka, J., Moravcsik, J., Suppes, P. (eds.) Approaches to Natural Language, pp. 221–242. D. Reidel Publishing Company, Dodrecht (1973). https://doi.org/10.1007/978-94-010-2506-5-10, http://dx.doi.org/10.1007/978-94-010-2506-5-10

[7] Montague, R.: English as a formal language. In: et al., B.V. (ed.) Linguaggi nella Societa a nella Tecnica, pp. 189–224. Edizioni di Comunita, Milan (1974), reprinted in: Formal Philosophy: selected papers of Richard Montegue, pp. 108–121

[8] Scott, D.: Outline of a mathematical theory of computation. Proc. 4th Annual Princeton Conf. on Information Sciences and Systems pp. 169–176 (1970)

[9] Stukachev, A.I.: Approximation spaces of temporal processes and effectivenes of interval semantics. Advances in Intelligent Sysstems and Computing **1242**, 53–61 (2021)

[10] Stukachev, A.I.: Interval extensions of orders and temporal approximation spaces. Siberian Math. Journal **62**(4), 730–741 (2021)

[11] Stukachev, A.: Effective model theory: an approach via Σ-definability. Lecture Notes in Logic **41**, 164–197 (2013)

The logic of the English auxiliary system

Yusuke Kubota[1] and Robert Levine[2]

[1] National Institute for Japanese Language and Linguistics
[2] Ohio State University

Abstract. This paper proposes an analysis of the English auxiliary system in Hybrid TLG. Our proposal differs from related approaches in lexicalist syntactic theories (such as HPSG and earlier variants of categorial grammar) in taking auxiliaries to be higher-order operators that syntactically (and not just semantically) scope over the local clauses in which they appear. We formulate an analysis of the familiar NI(C)E properties of auxiliaries in this approach. An advantage for the higher-order analysis comes from the fact that it offers a straightforward solution for a long-standing puzzle for earlier lexicalist approaches pertaining to the distribution of the unstressed *do*.

Keywords: auxiliary verb, Type-Logical Grammar, *do* insertion, higher-order auxiliary, categorial grammar, NICE properties

1 The NICE properties and unstressed *do*

Auxiliaries are commonly introduced in introductory syntax courses as members of a natural class whose distributional characteristics are captured by their occurrence in three supposedly quite independent constructions—inversion, sentential negation and VP ellipsis and one morphological form—NEG contraction.

(1) a. John {will/should} buy the book.
b. John {will/should} not buy the book. (cf. *John buys not the book.)
c. {Will/Should} John buy the book? (cf. *Buys John the book?)
d. Who will buy the book? – John {will/should}. (cf. *John buys.)
e. John {won't/shouldn't} buy the book. (cf. *John buysn't the book.)

Any syntactic theory should provide an explicit (and coherent) analysis of these so-called 'NICE' properties (Negation, Inversion, Contraction and Ellipsis). The distribution of the unstressed form of *do* is especially important in this connection as it has played a non-negligible role in the history of generative grammar. As is well-known, unstressed *do* (notated as *dŏ* in what follows) appears in all the NICE environments but not in simple declarative sentences:

(2) a. *John {dĭd/dŏes} buy the book.
b. John {dĭd/dŏes} not buy the book.
c. {Dĭd/Dŏes} John buy the book?

d. Who {bought/buys} the book? – John {dĭd/dŏes}.
e. John {dĭdn't/dŏesn't} buy the book.

On the one hand, in the early history of transformational generative grammar (starting with Chomsky [5]), the analysis of the otherwise puzzling patterns in (2) via the so-called *do* insertion transformation was regarded as one of the most successful applications of transformational analysis to the grammar of English. On the other hand, the somewhat peculiar distributional restriction on *do* exemplified in (2a), where, unlike other auxiliaries, it is banned from non-negative declarative environments, has long remained problematic in nontransformational treatments of English auxiliaries, a point emphasized in Sag et al. [38]. In fact, Sag et al. [38] take the '*do* insertion' paradigm in (2) to be one of the major pieces of evidence supporting their construction-based analysis of English auxiliaries (involving a 'slight' reorganization of the role that the AUX feature plays in the overall system), which departs from the strictly lexical analysis pioneered in Gazdar et al. [8] that has since been widely assumed as the standard analysis in the lexicalist tradition.

Given the prominent role that facts about English auxiliaries have played in the history of generative grammar, the scarcity of literature on this issue in categorial grammar research is rather surprising. In particular, to our knowledge, there is as yet no single explicit account of the well-known *do* insertion facts in the categorial grammar literature. There are of course sporadic accounts of some specific aspects of auxiliary syntax (and semantics), such as the important pioneering work by Bach [1, 2], which provides the basis for an explicit semantic account of subject position quantifiers in sentences with modal auxiliaries in lexicalist syntactic theories (more on this point in section 2), the analysis of VP ellipsis by Morrill and Merenciano [28] and Jäger [13] in the 90s, and the analysis of the 'anomalous scope' patterns that modal auxiliaries display in Gapping by Kubota and Levine [17, 18], to mention just a few. However, oddly enough, the NICE properties and *do* insertion facts—which have been considered to be one of the key touchstones for contemporary syntactic theory in the generative tradition—seem to have completely escaped the attention of categorial grammarians to date. This is perhaps due to the implicit assumption that at least the core of the PSG analyses of auxiliaries will more or less straightforwardly carry over to categorial grammar, given the many common theoretical assumptions that the two approaches share at the fundamental level (see for example Kubota [16] in this connection).

But the premise of this implicit assumption is threatened when it comes to the treatment of 'challenging' facts, for which different approaches tend to resort to idiosyncratic properties of their own. This is exactly what we see in the treatment of *do* insertion in the most recent incarnation of the analysis of English auxiliaries in HPSG by Sag et al. [38]. So far as we can tell, the elaborate constructional analysis they offer is by no means straightforwardly translatable to any variant of categorial grammar. It is for this reason that we take up the old issue of NICE properties and *do* insertion facts in the present paper. In particular, we aim to shed a new light on this problem by formulating an analysis

in Type-Logical Grammar. Our starting point is the 'higher-order' analysis of modal auxiliaries whose key idea is due to Siegel [39] and which was explicitly formalized in a type-logical setup in Kubota and Levine [17]. This represents a departure from the traditional VP/VP analysis in the lexicalist syntax tradition, by entertaining a movement-like operation in the analysis of modal auxiliaries. We formulate an explicit account of the core syntactic properties of auxiliary verbs in English (including the NIE of the NICE properties). While formulated at a more abstract level, our approach directly builds on the lexicalist approach in identifying the commonality of the NIE constructions as phenomena that target the VP/VP lexical signs of auxiliaries. The key claim of the present paper is that there is a direct empirical payoff for entertaining this more abstract perspective on the syntax of auxiliaries, and that the evidence comes from *do* insertion. Unlike the phrase structure-based or constructional setup, in an inference-based (or deductive) system like ours, operations that target VP/VP signs can themselves be the target of still higher-order operations. This enables us to entertain a more abstract view on *do* support than a construction-based encoding of the sort proposed by Sag et al. [38]: by seeing *do* insertion as a 'last resort' inference strategy, as it were, we can capture the key insight of the classical transformational account in a way that completely does away with the ad-hoc structure manipulation operations inherent to the latter.

2 Modals as scope-taking operators

Our approach to modal auxiliaries is heavily influenced by Oehrle's (1994) foundational work on quantifier scope. Oehrle's key insight involves utilizing the lambda calculus for characterizing the prosodic component of linguistic expressions, which enables him to model Montague's quantifying-in via lambda abstraction in the prosodic component. We implement this analysis with the non-directional implicational connective $\uparrow$ in Hybrid Type-Logical Grammar (Hybrid TLG).[3] In this approach, (3) is analyzed as in (4).

(3) John read every book.

(4)
$$\dfrac{\dfrac{\dfrac{\dfrac{\mathsf{read};\ \mathbf{read};\ (\mathrm{NP}\backslash\mathrm{S})/\mathrm{NP} \quad [\varphi_1; x; \mathrm{NP}]^1}{\mathsf{read}\bullet\varphi_1;\ \mathbf{read}(x);\ \mathrm{NP}\backslash\mathrm{S}} \qquad \mathsf{john};\ \mathbf{j};\ \mathrm{NP}}{\mathsf{john}\bullet\mathsf{read}\bullet\varphi_1;\ \mathbf{read}(x)(\mathbf{j});\ \mathrm{S}}}{\lambda\varphi_1.\mathsf{john}\bullet\mathsf{read}\bullet\varphi_1;\ \lambda x.\mathbf{read}(x)(\mathbf{j});\ \mathrm{S}\uparrow\mathrm{NP}}\uparrow\mathrm{I}^1 \qquad \begin{array}{l}\lambda\sigma_1.\sigma_1(\mathsf{every}\bullet\mathsf{book});\\ \mathbf{V}_{\mathbf{book}};\ \mathrm{S}\uparrow(\mathrm{S}\uparrow\mathrm{NP})\end{array}}{\mathsf{john}\bullet\mathsf{read}\bullet\mathsf{every}\bullet\mathsf{book};\ \mathbf{V}_{\mathbf{book}}(\lambda x.\mathbf{read}(x)(\mathbf{j}));\ \mathrm{S}}$$

The crucial innovation in Oehrle's approach is that abstraction on a prosodic variable makes it possible to separate the surface position in which a quantifier appears and its semantic scope: at the last step in (4), the quantifier sign applies to the S$\uparrow$NP constituent that is its (semantic) scope, but its prosodic contribution, or,

[3] Appendix A below contains a brief overview of Hybrid TLG. See Kubota and Levine [20, chapter 2 and Appendix A] for a more detailed exposition.

'string support' for the higher order prosodic specification $\lambda\sigma_1.\sigma_1(\mathsf{every} \bullet \mathsf{book})$, ends up in a position corresponding to the prosodic variable φ_1 in the scope constituent.

This analysis, as Oehrle demonstrates, captures scope ambiguity effortlessly, without resort to any special mechanisms. (5b) shows how the inverse scope reading for (5a) is obtained in this approach.

(5) a. Some student read every book.

b.
$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\varphi_1 \bullet \mathsf{read} \bullet \varphi_2;\ \mathbf{read}(u)(v);\ \mathrm{S}} \qquad \vdots}{\lambda\varphi_1.\varphi_1 \bullet \mathsf{read} \bullet \varphi_2;\ \lambda v.\mathbf{read}(u)(v);\ \mathrm{S}\uparrow\mathrm{NP} \qquad \lambda\sigma_1.\sigma_1(\mathsf{some} \bullet \mathsf{student});\ \exists_{\mathbf{student}};\ \mathrm{S}\uparrow(\mathrm{S}\uparrow\mathrm{NP})}}{\mathsf{some} \bullet \mathsf{student} \bullet \mathsf{read} \bullet \varphi_2;\ \exists_{\mathbf{student}}(\lambda v.\mathbf{read}(u)(v));\ \mathrm{S} \qquad \vdots}}{\lambda\varphi_2.\mathsf{some} \bullet \mathsf{student} \bullet \mathsf{read} \bullet \varphi_2;\ \lambda u.\exists_{\mathbf{student}}(\lambda v.\mathbf{read}(u)(v));\ \mathrm{S}\uparrow\mathrm{NP} \qquad \lambda\sigma_2.\sigma_2(\mathsf{every} \bullet \mathsf{book});\ \forall_{\mathbf{book}};\ \mathrm{S}\uparrow(\mathrm{S}\uparrow\mathrm{NP})}$$
$$\mathsf{some} \bullet \mathsf{student} \bullet \mathsf{read} \bullet \mathsf{every} \bullet \mathsf{book};\ \forall_{\mathbf{book}}(\lambda u.\exists_{\mathbf{student}}(\lambda v.\mathbf{read}(u)(v)));\ \mathrm{S}$$

The order in which the two GQs compose into the proof determines the scopal ordering; an alternative derivation whose only difference from (5b) is the introduction of the universal before the existential will yield a second reading for (5a) corresponding to surface scope ($\exists > \forall$).

It is not only multiple tokens of GQs that create scope ambiguities, however. Modals interact with GQs in much the same way:

(6) Every student can vote.

(6) has two subtly—but critically—different readings. On one reading, where the universal scopes widely, (6) says that every individual who happens to be a student (in the actual world) has the right or ability to vote. On the other reading, the sentence does not refer to students in the actual world, but instead merely makes a statement about a possible situation: whoever happens to be a student in that situation has the right or ability to vote. The following formulas disambiguate these two readings:

(7) a. $\forall_{\mathbf{student}}(\lambda y.\Diamond\mathbf{vote}(y))$
b. $\Diamond\forall_{\mathbf{student}}(\lambda y.\mathbf{vote}(y))$

In the face of ambiguous data such as (6), it seems natural to extend Oehrle's treatment of GQs to modals as well.[4] This is in fact straightforward, by assuming

[4] The wide scope reading for the modal in sentences like (6) has long been known to pose a challenge for the VP/VP analysis in lexicalist approaches. For example, Gazdar et al. [7], noting this difficulty, wind up positing a version of the modal *may* with the semantics $(\lambda Q\lambda\mathscr{P}.\Diamond\,\mathscr{P}(Q)$ in an extensionalized fragment) that directly subcategorizes for a GQ-type expression as the subject following Bach [1, 2]. It is unclear how the modal narrow scope reading is obtained in their approach.

that modals are GQ-like expressions, except that they scope over S with a VP/VP functor (instead of an NP) withdrawn (here, $\mathsf{id}_{et} = \lambda P_{et}.P$):[5]

(8) $\lambda\sigma.\sigma(\mathsf{can});\ \lambda\mathscr{F}.\Diamond\mathscr{F}(\mathsf{id}_{et});\ \mathrm{S}_{\mathit{fin}}\uparrow(\mathrm{S}_{\mathit{fin}}\uparrow(\mathrm{VP}_{\mathit{fin}}/\mathrm{VP}_{\mathit{bse}}))$

With this specification, we straightforwardly obtain (9) for the modal wide-scope reading for (6) (for how the narrow scope reading (7a) is obtained, see below):

(9)
$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathsf{vote};\ \mathbf{vote};\ \mathrm{VP}_b \qquad \left[\begin{array}{l}\varphi_1;\\ f;\ \mathrm{VP}_f/\mathrm{VP}_b\end{array}\right]^1}{\varphi_1 \bullet \mathsf{vote};\ f(\mathbf{vote});\ \mathrm{VP}_f} \qquad \left[\begin{array}{l}\varphi_2;\\ y;\ \mathrm{NP}\end{array}\right]^2}{\varphi_2 \bullet \varphi_1 \bullet \mathsf{vote};\ f(\mathbf{vote})(y);\ \mathrm{S}_f}\ \uparrow\mathrm{I}^2 \qquad \begin{array}{l}\lambda\sigma_1.\sigma_1(\mathsf{every} \bullet \mathsf{student});\\ \mathbf{\forall}_{\mathbf{student}};\\ \mathrm{S}_f\uparrow(\mathrm{S}_f\uparrow\mathrm{NP})\end{array}}{\mathsf{every} \bullet \mathsf{student} \bullet \varphi_1 \bullet \mathsf{vote};\ \mathbf{\forall}_{\mathbf{student}}(\lambda y.f(\mathbf{vote})(y));\ \mathrm{S}_f}\ \uparrow\mathrm{I}^1 \qquad \begin{array}{l}\lambda\sigma_2.\sigma_2(\mathsf{can});\\ \lambda\mathscr{F}.\Diamond\mathscr{F}(\mathsf{id}_{et});\\ \mathrm{S}_f\uparrow(\mathrm{S}_f\uparrow(\mathrm{VP}_f/\mathrm{VP}_b))\end{array}}{\mathsf{every} \bullet \mathsf{student} \bullet \mathsf{can} \bullet \mathsf{vote};\ \Diamond\mathbf{\forall}_{\mathbf{student}}(\lambda y.\mathbf{vote}(y));\ \mathrm{S}_f}$$

The key point here is that the quantifier is introduced in the derivation before the modal auxiliary, entailing its narrow scope.

Before proceeding further, it is important to recognize a potential overgeneration problem that arises with the scope-operator analysis of modals along the lines of (8). Unless appropriate constraints are imposed, the present analysis has the danger of predicting readings for modals (and related expressions such as VP negation) in which they scope out of their local clauses. This is clearly impossible in English. For example, the following sentence does not have a reading paraphrasable as something like 'it should be the case that John thought Ann is to buy/is buying the car':

(10) John thought Ann should buy the car.

In order to prevent this type of overgeneration and restrict the scope of modal auxiliaries to the local clause in which they occur, we can employ a clause-level indexing mechanism of the sort proposed by Pogodalla and Pompigne [34] for a slightly different purpose.[6] A full description of the indexing convention is given

[5] The features *fin* and *bse* here (abbreviated as *f* and *b* below) should be thought of as the (analogues of) 'VFORM' features (in G/HPSG terms) that mark finite and base forms of verbs respectively. This ensures that modals can only combine with base forms of verbs and after the modal is combined with the verb, the result is finite, and no other modal can stack on top of the resultant VP.

[6] Another, and perhaps more standard, approach for dealing with this type of issue in the TLG literature is to employ certain types of 'modality' operators in type logic. See, e.g., Moortgat [25] and Morrill [27] and references cited therein. A reviewer notes that it may be possible to simplify certain aspects of our index-based approach (which is essentially nothing more than a bookkeeping device) by recasting the relevant aspects of the analysis by using modality operators. We leave it for future work to investigate this issue in detail.

in section 9.2.2 of Kubota and Levine [20]. We illustrate its key points briefly in what follows. With the indexing restrictions made explicit, we have the following lexicon:

(11) a. $\lambda\sigma.\sigma(\mathsf{should})$; $\lambda\mathscr{G}.\Box\mathscr{G}(\mathsf{id}_{et})$; $\mathrm{S}^n_f\uparrow(\mathrm{S}^n_f\uparrow(\mathrm{VP}^n_f/\mathrm{VP}^n_b)$

b. $\mathsf{thought}$; $\mathbf{think}$; $\mathrm{VP}^{n+1}_f|\mathrm{S}^n_f$

The explicit indexing on the S and VP/VP categories in (11a) ensures that modals take scope directly over the clauses that are 'projections' of the VP/VP gaps that they bind, guaranteeing the clause-boundedness of the scope of these operators, as we now show.

A failed derivation for (10) is given in (12).

(12)
$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\mathsf{buy}\bullet\mathsf{the}\bullet\mathsf{car};\ \mathbf{buy}(\iota(\mathbf{car}));\ \mathrm{VP}^1_b}\quad \begin{bmatrix}\varphi_1;\\ f;\\ \mathrm{VP}^1_f/\mathrm{VP}^1_b\end{bmatrix}^1}{\varphi_1\bullet\mathsf{buy}\bullet\mathsf{the}\bullet\mathsf{car};\ f(\mathbf{buy}(\iota(\mathbf{car})));\ \mathrm{VP}^1_b}\quad \begin{array}{l}\mathsf{ann};\\ \mathbf{a};\\ \mathrm{NP}\end{array}}{\mathsf{ann}\bullet\varphi_1\bullet\mathsf{buy}\bullet\mathsf{the}\bullet\mathsf{car};\ f(\mathbf{buy}(\iota(\mathbf{car})))(\mathbf{a});\ \mathrm{S}^1_f}\quad \begin{array}{l}\mathsf{thought};\\ \mathbf{think};\\ \mathrm{VP}^{n+1}_f|\mathrm{S}^n_f\end{array}}{\mathsf{thought}\bullet\mathsf{ann}\bullet\varphi_1\bullet\mathsf{buy}\bullet\mathsf{the}\bullet\mathsf{car};\ \mathbf{think}(f(\mathbf{buy}(\iota(\mathbf{car})))(\mathbf{a}));\ \mathrm{VP}^2_f}\quad \begin{array}{l}\mathsf{john};\\ \mathbf{j};\\ \mathrm{NP}\end{array}}{\mathsf{john}\bullet\mathsf{thought}\bullet\mathsf{ann}\bullet\varphi_1\bullet\mathsf{buy}\bullet\mathsf{the}\bullet\mathsf{car};\ \mathbf{think}(f(\mathbf{buy}(\iota(\mathbf{car})))(\mathbf{a}))(\mathbf{j});\ \mathrm{S}^2_f}$$
$$\dfrac{\lambda\varphi_1.\mathsf{john}\bullet\mathsf{thought}\bullet\mathsf{ann}\bullet\varphi_1\bullet\mathsf{buy}\bullet\mathsf{the}\bullet\mathsf{car};\ \lambda f.\mathbf{think}(f(\mathbf{buy}(\iota(\mathbf{car})))(\mathbf{a}))(\mathbf{j});\ \mathrm{S}^2_f|(\mathrm{VP}^1_f/\mathrm{VP}^1_b)}{\mathbf{FAIL}}\ \uparrow^1\quad \begin{array}{l}\lambda\sigma.\sigma(\mathsf{should});\\ \lambda\mathscr{G}.\Box\mathscr{G}(\mathsf{id}_{et});\\ \mathrm{S}^n_f\uparrow(\mathrm{S}^n_f\uparrow(\mathrm{VP}^n_f/\mathrm{VP}^n_b))\end{array}$$

Here, the withdrawn VP/VP (from the embedded clause) carries the index 1 but this doesn't match the index value 2 on the S. Since the modal operator explicitly requires these values to match with each other, the derivation fails at the step at which the modal is introduced. We assume this clause-level indexing mechanism throughout, but omit the indices in the interest of minimizing notational clutter.

A strong indication that the higher-order treatment of modals presented above is on the right track comes from the fact that just such an analysis seems to be required independently in order to account for the seemingly anomalous scope of modals in examples such as (13):

(13) Mrs J can't live in Boston and Mr J in LA! $(\neg\Diamond > \wedge)$

See Kubota and Levine [18, 20] for detailed discussion of how the kind of higher order description in (8) provides a natural account of such examples, which display (on one of their readings) an unusual scoping pattern in which the negative modal scopes over the conjunction.

Importantly, in such examples, as well as simpler examples such as (6), modals can also scope narrowly with respect to the other scopal operator (in (6), the quantifier; in (13), the conjunction). In the present setup, this falls out as a

straightforward consequence of the higher-order lexical entry in (8). That is, a VP/VP entry for a modal auxiliary which essentially corresponds to (the simpler version of) the lexical entries for modal auxiliaries in lexicalist theories of syntax falls out as a theorem from (8), as in the following proof:

(14)

$$\dfrac{\dfrac{\dfrac{\lambda\sigma.\sigma(\mathsf{can't});\ \lambda\mathscr{F}.\neg\Diamond\mathscr{F}(\mathsf{id}_{et});\ \mathrm{S}_f\uparrow(\mathrm{S}_f\uparrow(\mathrm{VP}_f/\mathrm{VP}_b))\quad \dfrac{[\varphi_1; x; \mathrm{NP}]^1 \qquad \dfrac{[\varphi_2; g; \mathrm{VP}_f/\mathrm{VP}_b]^2 \quad [\varphi_3; f; \mathrm{VP}_b]^3}{\varphi_2 \bullet \varphi_3;\ g(f);\ \mathrm{VP}_f}\,/\mathrm{E}}{\varphi_1 \bullet \varphi_2 \bullet \varphi_3;\ g(f)(x);\ \mathrm{S}_f}\,\backslash\mathrm{E}}{\lambda\varphi_2.\varphi_1 \bullet \varphi_2 \bullet \varphi_3;\ \lambda g.g(f)(x);\ \mathrm{S}_f\uparrow(\mathrm{VP}_f/\mathrm{VP}_b)}\,\uparrow\mathrm{I}^2}{\varphi_1 \bullet \mathsf{can't} \bullet \varphi_3;\ \neg\Diamond f(x);\ \mathrm{S}_f}\,\uparrow\mathrm{E}}{\mathsf{can't} \bullet \varphi_3;\ \lambda x.\neg\Diamond f(x);\ \mathrm{VP}_f}\,\backslash\mathrm{I}^1}{\mathsf{can't};\ \lambda f\lambda x.\neg\Diamond f(x);\ \mathrm{VP}_f/\mathrm{VP}_b}\,/\mathrm{I}^3$$

Using the VP/VP entry for the modal derived in (14), the narrow scope readings for the modal for both (8) and (13) follow straightforwardly.

Beyond its wide empirical reach illustrated above, our more abstract treatment of the syntax-semantics interface of modals can be seen as a unification of what have been viewed as two very distinct, competing analyses. A common treatment of modals in the Principles and Parameters approach proposed in the period following Pollock [35] is to take them as originating under their own functional head and moving to Spec of TP (see, for example, Iatridou and Zeijlstra [12], Harwood [10] and Radford [37, 241] for discussion of this general line of analysis). This treatment of modals as operators raised into higher positions to scope over propositional content offers—at least in principle—a strategy for solving the puzzles posed by (6) and (13), in conjunction with certain ancillary assumptions (e.g., Johnson [14] on Gapping, but see Kubota and Levine [18] for a critique). On the other hand, the lexicalist treatment of modals as verbs combining with VP complements has been empirically successful in capturing morphosyntactic dependencies involving auxiliaries since its introduction in Gazdar [6] and Gazdar et al. [8],[7] but encounters the serious hurdles noted above. On our analysis, it is exactly these two types of seemingly rival analyses which fall out of the single lexical entry in (8).

3 Higher-order operator analysis of NIE properties

From the discussion above, it should be clear that our approach differs from the traditional lexicalist analyses of auxiliaries in that it takes modal auxiliaries to be higher-order operators that scope over clausal constituents. Aside from some advantages it offers in the analysis of scopal interactions with other operators (such as generalized quantifiers and conjunction in Gapping, as noted above), one may rightfully wonder whether there is any payoff to this more abstract analysis

[7] Note also that the distributive interepretation of modals in Gapping is straightforward in this type of approach, whereas the high-modal analysis such as Johnson [14] in the transformational literature struggles to obtain this reading. See Kubota and Levine [18] Park [33] and Potter et al. [36] for some relevant discussion.

of English auxiliaries. We argue in this section that additional advantage does in fact come from the analysis of the familiar syntactic properties of English auxiliaries, in particular, the somewhat puzzling distribution of the unstressed *do*, which has—as noted in section 1—proven problematic in lexicalist analyses of English auxiliaries.

In order to formulate an analysis of *do* insertion, we need an explicit analysis of (at least a subset of) NICE properties. In the rest of this paper, we set aside contraction since this phenomenon involves morphological idiosyncrasy that justifies a lexical treatment. For the other three phenomena, our analysis builds on the key idea that auxiliary verbs are syntactically operators that fill in the preverbal gap position of type VP/VP. Unlike the more traditional VP/VP analysis, the higher-order analysis of auxiliaries introduced in the previous section opens up an analytic possibility in which we can define operators that manipulate the type VP/VP gap before it gets filled in by the lexical auxiliary. Our analysis of the NIE operators crucially exploits this possibility.

3.1 Basic actions of the NIE operators

We start with inversion. In sentences with inverted auxiliaries such as polar questions, the auxiliary verb appears at the beginning of the clause rather than in the preverbal position. This word order change can simply be handled by positing the following higher-order operator that maps a $\mathrm{S}\uparrow(\mathrm{VP}/\mathrm{VP})$ to another $\mathrm{S}\uparrow(\mathrm{VP}/\mathrm{VP})$, which differs only in the prosodic specification. The fact that inversion has taken place is recorded in the syntactic feature *inv*, a standard technique for distinguishing inverted from non-inverted clauses in lexicalist approaches.[8,9]

(15) $\lambda\sigma\lambda\varphi.\varphi \bullet \sigma(\boldsymbol{\epsilon});\ \lambda\mathscr{F}.\mathscr{F};\ (\mathrm{S}_{inv}\uparrow(\mathrm{VP}_f/\mathrm{VP}_b))\uparrow(\mathrm{S}_f\uparrow(\mathrm{VP}_f/\mathrm{VP}_b))$

After the inversion operator applies to $\mathrm{S}\uparrow(\mathrm{VP}/\mathrm{VP})$, the result is passed on to the higher-order auxiliary. The latter fills in the auxiliary string in the gap position—which has been moved to the clause-initial position by the inversion operator—to complete the derivation (here, we have slightly generalized the lexical entry for the auxiliary, replacing S_f with S_α, where $\alpha \in \{\mathit{fin}, \mathit{inv}\}$).

(16)

$$\dfrac{\lambda\sigma.\sigma(\mathsf{should});\ \lambda\mathscr{F}.\Box\mathscr{F}(\mathsf{id}_{et});\ \mathrm{S}_\alpha\uparrow(\mathrm{S}_\alpha\uparrow(\mathrm{VP}_f/\mathrm{VP}_b)) \qquad \dfrac{\lambda\sigma\lambda\varphi.\varphi \bullet \sigma(\boldsymbol{\epsilon});\ \lambda\mathscr{F}.\mathscr{F};\ (\mathrm{S}_{inv}\uparrow(\mathrm{VP}_f/\mathrm{VP}_b))\uparrow(\mathrm{S}_f\uparrow(\mathrm{VP}_f/\mathrm{VP}_b)) \qquad \dfrac{\vdots}{\lambda\varphi.\mathsf{john} \bullet \varphi \bullet \mathsf{come};\ \lambda f.f(\mathbf{come})(\mathbf{j});\ \mathrm{S}_f\uparrow(\mathrm{VP}_f/\mathrm{VP}_b)}}{\lambda\varphi.\varphi \bullet \mathsf{john} \bullet \mathsf{come};\ \lambda f.f(\mathbf{come})(\mathbf{j});\ \mathrm{S}_{inv}\uparrow(\mathrm{VP}_f/\mathrm{VP}_b)}}{\mathsf{should} \bullet \mathsf{john} \bullet \mathsf{come};\ \Box\mathbf{come}(\mathbf{j});\ \mathrm{S}_{inv}}$$

[8] The semantics of the inversion operator is the identity function. We assume that a separate operator (of the sort assumed in Kubota and Levine [21]) is responsible for introducing the semantics of polar questions. The separation of the syntactic operation of inversion and question semantics is motivated by the fact that inversion is found in contexts other than polar questions.

[9] $\boldsymbol{\epsilon}$ designates the empty string.

An important fact about auxiliary inversion is that it is a clause-bound phenomenon.

(17) *Will anybody who __ be vaccinated after arrival may visit Japan?

We utilize the clause-level indexing mechanism from the previous section to capture this fact explicitly. Specifically, (15) is actually an abbreviated version of (18), which makes it explicit that the 'auxiliary gap' that the inversion operator targets is a local one.

(18) $\lambda\sigma\lambda\varphi.\varphi \bullet \sigma(\boldsymbol{\epsilon})$; $\lambda\mathscr{F}.\mathscr{F}$; $(\mathrm{S}^n_{inv}\uparrow(\mathrm{VP}^n_f/\mathrm{VP}^n_b))\uparrow(\mathrm{S}^n_f\uparrow(\mathrm{VP}^n_f/\mathrm{VP}^n_b))$

We make the same assumption about the ellipsis and negation operators we introduce below, but continue to suppress the clause-level indexing for the sake of readability. It should be kept in mind that these indices are present in the official version of the analysis.

Moving on to ellipsis, descriptively, VP ellipsis is a phenomenon in which an auxiliary stands in for a full VP. We here assume a somewhat more elaborate analysis of VP ellipsis than the one we utilized in Kubota and Levine [19, 20], where the the VP ellipsis operator is defined as a higher-order operator that replaces a type VP gap by a type VP/VP gap.

(19) $\lambda\sigma\lambda\varphi.\sigma(\varphi)$; $\lambda\mathscr{G}\lambda f.\mathscr{G}(f(P))$; $(\mathrm{S}_f\uparrow(\mathrm{VP}_f/\mathrm{VP}_b))\uparrow(\mathrm{S}_b\uparrow\mathrm{VP}_b)$

The ellipsis operator supplies the contextual variable P as the meaning of the missing VP. We assume that technically P is just a free variable and that its value is contextually determined just like the referent of (free) pronouns. A sample derivation is given in (20). Just as in the case of inversion, the ellipsis operator applies first and the resultant $\mathrm{S}\uparrow(\mathrm{VP}/\mathrm{VP})$ expression is passed on to the higher-order auxiliary.

(20)

$$\dfrac{\lambda\sigma.\sigma(\mathsf{should});\ \lambda\mathscr{F}.\Box\mathscr{F}(\mathsf{id}_{et});\ \mathrm{S}_\alpha\uparrow(\mathrm{S}_\alpha\uparrow(\mathrm{VP}_f/\mathrm{VP}_b)) \qquad \dfrac{\lambda\sigma\lambda\varphi.\sigma(\varphi);\ \lambda\mathscr{G}\lambda f.\mathscr{G}(f(P));\ (\mathrm{S}_f\uparrow(\mathrm{VP}_f/\mathrm{VP}_b))\uparrow(\mathrm{S}_b\uparrow\mathrm{VP}_b) \qquad \vdots \\ \lambda\varphi.\mathsf{john}\bullet\varphi;\ \lambda Q.Q(\mathbf{j});\ \mathrm{S}_b\uparrow\mathrm{VP}_b}{\lambda\varphi.\mathsf{john}\bullet\varphi;\ \lambda f.f(P)(\mathbf{j});\ \mathrm{S}_f\uparrow(\mathrm{VP}_f/\mathrm{VP}_b)}}{\mathsf{john}\bullet\mathsf{should};\ \Box P(\mathbf{j});\ \mathrm{S}_f}$$

Finally, negation is listed in the lexicon as a higher-order operator similar to the modal auxiliaries as in (21) in order to capture the polarity-sensitive scopal interactions with modals (see Kubota and Levine [21] for details), but an alternative sign that applies to a VP/VP-gapped sentence and inserts the negation morpheme right after the gap can be obtained as a theorem as in (22).

(21) $\lambda\sigma.\sigma(\mathsf{not})$; $\lambda\mathscr{F}.\neg\mathscr{F}(\mathsf{id}_{et})$; $\mathrm{S}_\alpha\uparrow(\mathrm{S}_\alpha\uparrow(\mathrm{VP}_b/\mathrm{VP}_b))$

(22)

$$\dfrac{\dfrac{\lambda\sigma.\sigma(\mathsf{not});\ \lambda\mathscr{F}.\neg\mathscr{F}(\mathsf{id}_{et});\ \mathrm{S}_\alpha\uparrow(\mathrm{S}_\alpha\uparrow(\mathrm{VP}_b/\mathrm{VP}_b)) \quad \dfrac{\dfrac{\begin{bmatrix}\varphi_4;\\ x;\\ \mathrm{NP}\end{bmatrix}^4 \quad \dfrac{\begin{bmatrix}\varphi_1;\\ f;\\ \mathrm{VP}_f/\mathrm{VP}_b\end{bmatrix}^1 \quad \dfrac{\begin{bmatrix}\varphi_2;\\ g;\ \mathrm{VP}_b/\mathrm{VP}_b\end{bmatrix}^2 \quad \begin{bmatrix}\varphi_3;\\ P;\ \mathrm{VP}_b\end{bmatrix}^3}{\varphi_2 \bullet \varphi_3;\ g(P);\ \mathrm{VP}_b}}{\varphi_1 \bullet \varphi_2 \bullet \varphi_3;\ f(g(P));\ \mathrm{VP}_f}}{\varphi_4 \bullet \varphi_1 \bullet \varphi_2 \bullet \varphi_3;\ f(g(P))(x);\ \mathrm{S}_f}\uparrow^2}{\lambda\varphi_2.\varphi_4 \bullet \varphi_1 \bullet \varphi_2 \bullet \varphi_3;\ \lambda g.f(g(P))(x);\ \mathrm{S}_f\uparrow(\mathrm{VP}_b/\mathrm{VP}_b)}}{\varphi_4 \bullet \varphi_1 \bullet \mathsf{not} \bullet \varphi_3;\ \neg f(P)(x);\ \mathrm{S}_f}\backslash^4$$

$$\dfrac{\dfrac{\begin{bmatrix}\sigma;\\ \mathscr{F};\\ \mathrm{S}_f\uparrow(\mathrm{VP}_f/\mathrm{VP}_b)\end{bmatrix}^5 \quad \dfrac{\varphi_1 \bullet \mathsf{not} \bullet \varphi_3;\ \lambda x.\neg f(P)(x);\ \mathrm{VP}_f}{\varphi_1 \bullet \mathsf{not};\ \lambda P\lambda x.\neg f(P)(x);\ \mathrm{VP}_f/\mathrm{VP}_b}/^3}{\sigma(\varphi_1 \bullet \mathsf{not});\ \mathscr{F}(\lambda P\lambda x.\neg f(P)(x));\ \mathrm{S}_f}}{\lambda\varphi_1.\sigma(\varphi_1 \bullet \mathsf{not});\ \lambda f.\mathscr{F}(\lambda P\lambda x.\neg f(P)(x));\ \mathrm{S}_f\uparrow(\mathrm{VP}_f/\mathrm{VP}_b)}\uparrow^1}{\lambda\sigma\lambda\varphi_1.\sigma(\varphi_1 \bullet \mathsf{not});\ \lambda\mathscr{F}\lambda f.\mathbf{Cf}(\lambda P\lambda x.\neg f(P)(x));\ (\mathrm{S}_f\uparrow(\mathrm{VP}_f/\mathrm{VP}_b))\uparrow(\mathrm{S}_f\uparrow(\mathrm{VP}_f/\mathrm{VP}_b))}\uparrow^5$$

This is a theorem that falls out from (21) regardless of whether one wants it or not. This more complex sign in (22) turns out to play a crucial role in our analysis of the interactions of the other NIE operators in the next section.

We can use (22) for deriving a simple sentence containing negation goes as follows:

(23)

$$\dfrac{\lambda\sigma.\sigma(\mathsf{should});\ \lambda\mathscr{F}.\Box\mathscr{F}(\mathsf{id}_{et});\ \mathrm{S}_\alpha\uparrow(\mathrm{S}_\alpha\uparrow(\mathrm{VP}_f/\mathrm{VP}_b)) \quad \dfrac{\begin{array}{l}\vdots\\ \lambda\sigma\lambda\varphi_1.\sigma(\varphi_1 \bullet \mathsf{not});\\ \lambda\mathscr{F}\lambda f.\mathscr{F}(\lambda P\lambda x.\neg f(P)(x));\\ (\mathrm{S}_f\uparrow(\mathrm{VP}_f/\mathrm{VP}_b))\uparrow(\mathrm{S}_f\uparrow(\mathrm{VP}_f/\mathrm{VP}_b))\end{array} \quad \begin{array}{l}\vdots\\ \lambda\varphi.\mathsf{john} \bullet \varphi \bullet \mathsf{come};\\ \lambda f.f(\mathbf{come})(\mathbf{j});\\ \mathrm{S}_f\uparrow(\mathrm{VP}_f/\mathrm{VP}_b)\end{array}}{\lambda\varphi.\mathsf{john} \bullet \varphi \bullet \mathsf{not} \bullet \mathsf{come};\ \lambda f.\neg f(\mathbf{come})(\mathbf{j});\ \mathrm{S}_f\uparrow(\mathrm{VP}_f/\mathrm{VP}_b)}}{\mathsf{john} \bullet \mathsf{should} \bullet \mathsf{not} \bullet \mathsf{come};\ \Box\neg\mathbf{come}(\mathbf{j});\ \mathrm{S}_f}$$

3.2 NIE Interactions

An important property of the analysis of the NIE operators above is that these operators interact with one another systematically to yield the right results for cases in which the relevant phenomena interact with one another. In order to facilitate discussion on this point, we introduce some abbreviatory notation first. Specifically, we write **Inv**, **Ell** and **Neg** for the three operators introduced above and **Lex** for some auxiliary lexical entry (*should* is chosen just for an illustration; $\neg$ is 'generalized negation' such that $\neg_{et\to et} = \lambda\mathscr{F}\lambda P\lambda x.\neg\mathscr{F}(P)(x)$).[10,11]

[10] The entries in (24) need the index markings of the sort discussed in section 3.1 (in order to prevent overgeneration of examples such as **Should John say Mary ∅ get the job?*), but we keep omitting these for the sake of notational transparency.

[11] Computationally inclined readers may find it unfortunate that our entries for **Inv** and **Ell** are not lexicalized—a concern that we share. We agree with the reviewer who raised this issue that it may plausibly be argued that these operations (or at least **Inv**) are morphological processes rather than empty operators in syntax. See also footnote 12 in this connection.

(24) a. $\textsc{Inv} = \lambda\sigma\lambda\varphi.\varphi \bullet \sigma(\epsilon);\ \lambda\mathscr{F}.\mathscr{F};\ (\mathrm{S}_{inv}\uparrow(\mathrm{VP}_f/\mathrm{VP}_b))\uparrow(\mathrm{S}_f\uparrow(\mathrm{VP}_f/\mathrm{VP}_b))$

b. $\textsc{Ell} = \lambda\sigma\lambda\varphi.\sigma(\varphi);\ \lambda\mathscr{G}\lambda f.\mathscr{G}(f(P));\ (\mathrm{S}_f\uparrow(\mathrm{VP}_f/\mathrm{VP}_b))\uparrow(\mathrm{S}_b\uparrow\mathrm{VP}_b)$

c. $\textsc{Neg} = \lambda\sigma\lambda\varphi.\sigma(\varphi \bullet \mathsf{not});$
$\lambda\mathscr{F}\lambda g.\mathscr{F}(\neg_{et\to et}g);(\mathrm{S}_f\uparrow(\mathrm{VP}_f/\mathrm{VP}_b))\uparrow(\mathrm{S}_f\uparrow(\mathrm{VP}_f/\mathrm{VP}_b))$

d. $\textsc{Lex} = \lambda\sigma.\sigma(\mathsf{should});\ \lambda\mathscr{F}.\Box\mathscr{F}(\mathsf{id}_{et});\ \mathrm{S}_\alpha\uparrow(\mathrm{S}_\alpha\uparrow(\mathrm{VP}_f/\mathrm{VP}_b))$

Given these abbreviatory notations, we can derive the inverted, complement-elided and negated versions of the auxiliary lexical signs as follows, via function composition of **Lex** and the three operators (here, $\circ$ denotes function composition; proofs for the theorems in (25) are omitted due to space constraints but are all straightforward):

(25) a. $\textsc{Lex} \circ \textsc{Inv} = \lambda\sigma.\mathsf{should} \bullet \sigma(\epsilon);\ \lambda\mathscr{F}.\Box\mathscr{F}(\mathsf{id}_{et});\ \mathrm{S}_{inv}\uparrow(\mathrm{S}_f\uparrow(\mathrm{VP}_f/\mathrm{VP}_b))$

b. $\textsc{Lex} \circ \textsc{Ell} = \lambda\sigma.\sigma(\mathsf{should});\ \lambda\mathscr{G}.\Box\mathscr{G}(P);\ \mathrm{S}_f\uparrow(\mathrm{S}_b\uparrow\mathrm{VP}_b)$

c. $\textsc{Lex} \circ \textsc{Neg} = \lambda\sigma.\sigma(\mathsf{should} \bullet \mathsf{not});\ \lambda\mathscr{F}.\Box\mathscr{F}(\neg_{et\to et});\ \mathrm{S}_f\uparrow(\mathrm{S}_f\uparrow(\mathrm{VP}_f/\mathrm{VP}_b))$

One interesting consequence that immediately follows from the above analysis is that a 'slanted' version of the inverted auxiliary sign is obtained as a theorem, as in the following proof:

(26)
$$\dfrac{\dfrac{\dfrac{\textsc{Lex} \circ \textsc{Inv}}{\vdots}}{\lambda\sigma.\mathsf{should} \bullet \sigma(\epsilon);\ \lambda\mathscr{F}.\Box\mathscr{F}(\mathsf{id}_{et});\ \mathrm{S}_{inv}\uparrow(\mathrm{S}_f\uparrow(\mathrm{VP}_f/\mathrm{VP}_b))} \quad \dfrac{\dfrac{\dfrac{[\varphi_3;\ x;\ \mathrm{NP}]^3 \quad \dfrac{[\varphi_1;\ f;\ \mathrm{VP}_f/\mathrm{VP}_b]^1 \quad [\varphi_2;\ P;\ \mathrm{VP}_b]^2}{\varphi_1 \bullet \varphi_2;\ f(P);\ \mathrm{VP}_f}}{\varphi_3 \bullet \varphi_1 \bullet \varphi_2;\ f(P)(x);\ \mathrm{S}_f}}{\lambda\varphi_1.\varphi_3 \bullet \varphi_1 \bullet \varphi_2;\ \lambda f.F(P)(x);\ \mathrm{S}_f\uparrow(\mathrm{VP}_f/\mathrm{VP}_b)}\uparrow\mathrm{I}^1}{\dfrac{\mathsf{should} \bullet \varphi_3 \bullet \varphi_2;\ \Box P(x);\ \mathrm{S}_{inv}}{\mathsf{should} \bullet \varphi_3;\ \lambda P.\Box P(x);\ \mathrm{S}_{inv}/\mathrm{VP}_b}\uparrow\mathrm{I}^2}\uparrow\mathrm{I}^3$$
$$\mathsf{should};\ \lambda x\lambda P.\Box P(x);\ \mathrm{S}_{inv}/\mathrm{VP}_b/\mathrm{NP}$$

Thus, just as there is a close connection between the present approach and the more traditional lexicalist approach in the analysis of basic cases, (the analog of) the inverted auxiliary entry in lexicalist approaches also falls out as a theorem in the present setup.

On the present approach, the interactions of the NIE phenomena can be captured by the interactions of the three operators. Since the derivations are all straightforward, we omit them but just list the relevant composed operators that are involved in each of the NIE interactions in (27e–h).

(27) a. John will come. **Lex**

b. Will John come? **Lex ∘ Inv**

c. John will ∅. **Lex ∘ Ell**

d. John will not come. **Lex ∘ Neg**

e. Will John not come? **Lex ∘ Inv ∘ Neg**

f. John will not ∅. **Lex ∘ Neg ∘ Ell**

g. Will John? **Lex ∘ Inv ∘ Ell**

h. Will John not ∅? **Lex** ∘ **Inv** ∘ **Neg** ∘ **Ell**

In contemporary lexicalist approaches and their variants (such as Sag et al. [38]), the NIE phenomena are typically treated via lexical operations, or in terms of constructional schemata. In such approaches, the interactions of these phenomena are captured by letting the lexical rules or constructional schemata feed into one another. Here, the idea is the same, except that in our case the interactions of the relevant operators are governed by the same logic of syntactic combinatorics (in which function composition is a theorem) that governs other aspects of syntax.[12]

4 The distribution of the unstressed *do*

4.1 *Do* insertion as a higher-order operator

With the analyses of the NIE operators in place, we are now ready to account for the distribution of unstressed *do*. As noted in section 1, the curious property of this auxiliary is that its distribution is limited to environments in which an auxiliary *must* appear. The challenge here is that if we simply posit the following lexical entry for *dŏ/dĭd* that is essentially identical in form to modal auxiliaries, then, we predict that *dŏ/dĭd* can appear in environments in which an auxiliary *can* appear, but this immediately leads to overgeneration of the declarative case (here, $\mathbf{P}$ is the past tense operator).

(28) $\mathbb{Lex} = \lambda\sigma.\sigma(\mathsf{d\breve{i}d});\ \lambda\mathscr{F}.\mathbf{P}\,\mathscr{F}(\mathsf{id}_{et});\ \mathrm{S}_\alpha\uparrow(\mathrm{S}_\alpha\uparrow(\mathrm{VP}_f/\mathrm{VP}_b))$

The problem that earlier PSG-based approaches face essentially stems from the fact that if we view *dŏ/dĭd* as a distinct lexical entry with the same syntactic properties as other auxiliaries in order to capture the parallel behaviors in the NI(C)E environments, then there is no straightforward solution for the overgeneration issue represented by (2a). Sag et al. [38] resort to a complex solution within Construction-based HPSG that involves a substantial change to the way in which the AUX feature is used in earlier lexicalist approaches.

Our claim here is that in the logic-based setup of Type-Logical Grammar, there is a conceptually simpler solution for this problem that cannot be easily translated to a PSG setup. Specifically, we take '*do* insertion' to be mediated by an operator that takes NIE operators as arguments to produce the same effect as the 'phantom' (i.e. non-existent) lexical entry for *dŏ/dĭd* in (28) above. In other words, we posit **Do** which satisfies the following property:

(29) For $f \in \{\textbf{Neg}/\textbf{Ell}/\textbf{Inv}\}$, $\textbf{Do}(f) \equiv \mathbb{Lex} \circ f$

[12] This of course raises the concern that we are perhaps blurring the boundary between syntax and morphology, blatantly going against the tenet of the lexicalist thesis (in the narrower sense of adhering to the 'lexical integrity principle'). We acknowledge this to be an important question, but leave further investigation to a future occasion.

Given that we know what $\mathbb{LEX}$ is (see (28)), defining **Do** turns out to be straightforward:[13]

(30) $\mathbf{Do} = \lambda f.\ \mathbb{LEX} \circ f$
$= \lambda f \lambda g.\ \mathbb{LEX}(f(g))$
$= \lambda\rho\lambda\sigma.[\lambda\sigma_0.\sigma_0(\mathsf{did})](\rho(\sigma));\ \lambda\mathscr{G}\lambda h.[\lambda\mathscr{F}.\mathbf{P}\,\mathscr{F}(\mathsf{id}_{et})](\mathscr{G}(h));\ (\mathrm{S}_\alpha\uparrow\mathrm{X})\uparrow(\mathrm{S}_\alpha\uparrow(\mathrm{VP}_f/\mathrm{VP}_b)\uparrow\mathrm{X})$
$= \lambda\rho\lambda\sigma.\rho(\sigma)(\mathsf{d\check{\imath}d});\ \lambda\mathscr{G}\lambda h.\mathbf{P}\,\mathscr{G}(h)(\mathsf{id}_{et});\ (\mathrm{S}_\alpha\uparrow\mathrm{X})\uparrow(\mathrm{S}_\alpha\uparrow(\mathrm{VP}_f/\mathrm{VP}_b)\uparrow\mathrm{X})$
where $\mathrm{X} \in \{\mathrm{S}_f\uparrow(\mathrm{VP}_f/\mathrm{VP}_b),\ \mathrm{S}_b\uparrow\mathrm{VP}_b\}$

Given the definition in (30), it should be straightforward to see that we get the right result in the NIE sentences. For example, in the inversion case, we have:

(31) $\mathbf{Do}(\textsc{Inv}) = \lambda f.[\mathbb{LEX} \circ f](\textsc{Inv})$
$= \mathbb{LEX} \circ \textsc{Inv}$
$= \lambda\sigma.\mathsf{d\check{\imath}d} \bullet \sigma(\boldsymbol{\epsilon});\ \lambda\mathscr{F}.\mathbf{P}\,\mathscr{F}(\mathsf{id}_{et});\ \mathrm{S}_{inv}\uparrow(\mathrm{S}_f\uparrow(\mathrm{VP}_f/\mathrm{VP}_b))$

The key point here is that **Do** closes off the VP/VP gap by directly *applying to* the NIE operators. This means that it can't work alone, from which it immediately follows that (2a) is not licensed. In this respect, the present proposal is reminiscent of the idea of 'last resort' that constitutes the underlying intuition of various formulations of *do* insertion in derivational approaches in generative grammar since Chomsky [5]. While intuitively appealing, the exact status of the *do* insertion operation has been problematic in virtually all variants of derivational approaches throughout the history of generative grammar. In the present logic-based setup, there is a conceptually simple and mathematically precise way of formalizing the operation of *do* insertion as a higher-order operator that specifically targets operators that apply to modal auxiliaries. In this connection, we would like to emphasize that 'logic' in the title of this paper should be construed broadly, not just referring to the underlying deductive system itself, but also the way in which the whole system of grammar is set up. This enables us to make sense of the systematicity inherent to the grammar of natural language in terms of the notion of logical inference. Section 4.2 discusses some further consequences of the present approach where (at least in our view) this perspective becomes important.

At this point, a reader with sound skepticism may wonder about the cognitive plausibility of positing such an abstract operator—a concern that we share. This is a complex issue, but our own current view is that this is a cost that is worth paying, since it makes the competence grammar more streamlined, and moreover, it has the potential of shedding new light on other aspects such as diachronic

[13] Note that the **Do** operator exploits the fact that the prosodic calculus in Hybrid TLG is a lambda calculus in which any higher-order operator can be defined. So far as we can tell, this cannot be implemented straightforwardly as a lexical item in related TLG approaches such as the Displacement Calculus [30] and NL_λ [3], let alone CCG. One might wonder whether this operator really needs to be posited as a lexical item, as opposed to, e.g., being treated as some kind of meta-operation in the lexicon. The discussion in section 4.2 is relevant for this issue, but we refrain from investigating it in detail in this paper.

change. Specifically, the analysis we have presented above can potentially shed a new light on the *origin* of *do* insertion as well, but we need to review the history of English in order to address this point. Thus, in the next section we present a preliminary sketch of an account that addresses the status of *do* insertion, an issue that has received considerable attention in historical syntax. We hope to convince the reader that the logic-based approach we advocate here has something new to offer to this debate in the neighboring field of diachronic syntax as well.

4.2 The origin of *do*

In this section, we argue that the analysis of unstressed *do* presented above has a potential further advantage in offering a natural explanation for how this peculiar distribution arose in the history of English. Admittedly, our discussion is highly speculative, and it also builds heavily on work on historical syntax by other scholars, most importantly, by Anthony Warner [41]. The point we would like to make is modest: we believe that Warner's view, originally expressed in the theoretical vocabulary of HPSG, is essentially on the right track, but that its key insight can be expressed even more transparently by taking a logic-based perspective of the sort we have advocated above.

According to Warner [41], the development of *do* took place in two stages in the period of early Modern English. The first stage coincides with the establishment of modals as a distinct class during the second half of the 15th century and the first half of the 16th century. During this period, there was a set of systematic changes to a class of verbs including *can*, *may* and *will* in the direction that the notion 'modal auxiliary' became a coherent grammatical category (this included the loss of nonfinite forms of *can*, *may* and *will*, and the loss of nonmodal (i.e., main verb) meanings for *can*, *may* and *will*, among other changes). The auxiliary use of *do* in the Standard variety of English started to develop around the same period. Warner follows Traugott in viewing that periphrastic *do* at this stage was associated with a range of pragmatic functions pertaining to 'affirmation of speaker truthfulness' [40, 257].

In an influential study that statistically demonstrated the two-stage development of the auxiliary use of *do*, Kroch [15] observes that from the beginning of the 16th century to the period of 1550–1575, the occurrence of *do* increases in all syntactic environments, including the declarative. But after the period of 1575–1600, the development in different contexts starts to diverge. *Do* in polar questions continues to increase but the use of *do* in declarative sentences starts to decline. In negative environments, there is an initial sharp decline, followed by a steady increase (but at a lower rate than in positive polar questions). Kroch attributes this two-stage change to the loss of V2 in English within an account of parametric change in the Principles and Parameters (P&P) framework.

Warner offers a reinterpretation of Kroch's data that does not rely on the P&P assumptions (Warner's view is endorsed by Hudson [11] as well). According to Warner, the split in the development of *do* in different syntactic environments after the period of 1575–1600 can be attributed to a process of reanalysis of the following sort. The key factor is the fact that the pragmatic function associated

with *do* (which is essentially semantic focus on polarity contrast) is an inherent property of polar questions. Thus, with the steady increase of *do* in polar questions, the pragmatic function originally associated with *do* was reanalyzed as the constructional meaning of the polar interrogative itself, with *do* having only the function of contributing tense information. *Do* then becomes essentially an 'allomorph' of the tense affix whose distribution is limited to the inverted interrogative environment. The use of periphrastic *do* in the declarative then declines gradually with this reanalysis, via blocking by simple verb inflection for expressing tense. Note that this account is also consistent with the initial drop and the later recovery in negative environments; the reanalysis of *do* is triggered in the interrogative context, and then spreads to other syntactic contexts via analogy. Warner attributes the source of this reanalysis to child language learning. Due to the pragmatic function, *do* in the declarative environment was restricted to the literary style, but it was common in polar interrogatives in colloquial speech. Then, the child learning the language was most likely exposed to utterances containing the auxiliary use of *do* only in polar questions, from which s/he would infer that *do* is nothing more than a tense auxiliary.

While all this seems plausible, one point that remains unclear in Warner's account is the status of *do* as an 'allomorph' of the tense affix. This is of course a descriptively accurate characterization of *do* in present-day English, but the theoretical issue (which previous PSG proposals struggled to account for) is how exactly this descriptive generalization is to be implemented in a formally explicit theory. Warner [41, 250, n28] claims that a lexical specification involving an implicational constraint along the lines of (32) captures this generalization:

(32) SUBCAT $\langle$[−ELLIPSIS, −AUX]$\rangle$ & *do* $\supset$ semantic prominence

This says that an overt *do* carries the effect of 'semantic prominence' unless it appears in a NICE environment (SUBCAT $\langle$[−ELLIPSIS, −AUX]$\rangle$). However, this is just a restatement of the facts, and, perhaps more disturbingly, it is unclear how such a complex lexical restriction arose out of a simple process of reanalysis of the sort Warner himself advocates.

Thus, even though the general outline of Warner's reanalysis-based account is quite attractive, the exact nature of the reanalysis (in particular, the formal status of the resultant lexical entry for *do*) within the PSG-based setup he adopts is somewhat unclear. It is then interesting to see that a conceptually much simpler reformulation becomes available once we recast his account within a type-logical setup. From our perspective, Warner's account can be simply understood as a reanalysis of $\mathbb{LEX}$ by **Do**. That is, in the first phase of the two-stage development of *do*, $\mathbb{LEX}$ used to exist in the grammar of English, so its distribution was not restricted to NICE environments. But its usage was restricted to literary style in declaratives, due to the pragmatic function it was associated with. A child acquiring the language then gets exposed to occurrences of *do* only in polar questions. This much is the same as in Warner's account. But then, we essentially have a situation in which the child has to choose between two competing hypotheses, $\mathbb{LEX}$ or **Do**, but the available evidence gives

him/her confidence only for the weaker hypothesis of positing **Do**. Given the lack of positive evidence for the stronger hypothesis, the child opts for the more conservative hypothesis. Crucially, unlike in Warner's original account, the formal status of **Do** is perfectly explicit here: it is an operator that mimics the effect of an auxiliary just when, despite the lack of evidence for an independent lexical auxiliary, we find an operator (**Inv**) that is looking to combine with an auxiliary—without the help of **Do**, there is no way to complete the derivation. Moreover, our reconceptualization provides a clear motivation for why the reanalysis has happened: **Do** wins over $\mathbb{L}\text{EX}$ in the acquisition context since it is the less risky hypothesis that is consistent with all the data that the child encounters.

Thus, the higher-order analysis of *do* insertion not only characterizes the distribution of *do* in present-day English, but it also potentially illuminates the process by which it arose. Of course, there is no way of directly proving or disproving the hypothesis we have entertained above, but we take it that, other things being equal, an analysis that provides a natural motivation for known facts about historical change is more preferable than one that doesn't.

5 Conclusion

In this paper, we have proposed an analysis of the English auxiliary system in Hybrid TLG. Given the lack of a detailed and systematic study of English auxiliaries in the past literature of categorial grammar, our conclusions in this paper should be of interest to practitioners of categorial grammar of various stripes (and for practitioners of other grammatical theories more generally). First, we believe that the conceptually simple analysis of *do* insertion that crucially relies on the higher-order treatment of modals and the NIE operators generally argues for the advantage of contemporary variants of TLG, all of which are equipped with machinery for dealing with this type of abstract syntactic composition in one way or another (but there are some possible subtleties here; note the point we have made in footnote 13). This then raises an interesting question for CCG. On the one hand, the higher-order analysis we have argued for is not directly implementable in CCG—to see this point, the reader is invited to think about how to reformulate the auxiliary entry in (8) (which is crucial for obtaining the 'anomalous scope' interpretation in Gapping and related phenomena) or the **Do** operator in (30) in CCG. On the other hand, an elaborate construction-based analysis of the sort recently advocated by Sag et al. [38] is also unlikely to straightforwardly carry over to CCG. But then, the treatment of *do* insertion seems to remain one of the major open questions for CCG. We refrain from speculating about possible responses to this challenge, but instead just raise this issue explicitly here since it is (in our view) yet another variant of the long-term tension between 'surface-oriented' vs. 'abstract' syntax of the sort that various scholars have commented on since the very inception of nontransformational variants of syntax at the beginning of the 80s.

Bibliography

[1] Bach, E.: Tenses and aspects as functions on verb-phrases. In: Rohrer, C. (ed.) Time, Tense, and Quantifiers, pp. 19–37. Niemeyer, Tuebingen (1980)

[2] Bach, E.: Generalized categorial grammars and the English auxiliary. In: Heny, F., Richards, B. (eds.) Linguistic Categories: Auxiliaries and Related Puzzles, vol. 2, pp. 101–120. Reidel, Dordrecht (1983)

[3] Barker, C., Shan, C.: Continuations and Natural Language. Oxford University Press, Oxford (2015)

[4] Bernardi, R.: Reasoning with Polarity in Categorial Type Logic. Ph.D. thesis, University of Utrecht (2002)

[5] Chomsky, N.: Syntactic Structures. Mouton, The Hague (1957)

[6] Gazdar, G.: Phrase structure grammar. In: Jacobson, P., Pullum, G.K. (eds.) The Nature of Syntactic Representation, pp. 131–186. Kluwer (1982)

[7] Gazdar, G., Klein, E., Pullum, G.K., Sag, I.A.: Generalized Phrase Structure Grammar. Harvard University Press, Cambridge, MA (1985)

[8] Gazdar, G., Pullum, G., Sag, I.: Auxiliaries and related phenomena in a restrictive theory of grammar. Language 58, 591–638 (1982)

[9] de Groote, P.: Towards abstract categorial grammars. In: Association for Computational Linguistics, 39th Annual Meeting and 10th Conference of the European Chapter. pp. 148–155. Association for Computational Linguistics (2001)

[10] Harwood, W.: Rise of the auxiliaries: A case for auxiliary raising vs. . affix lowering. The Linguistic Review 31(2), 295–362 (2014)

[11] Hudson, R.: The rise of auxiliary *do*: Verb raising or category-strengthening? Transactions of the Philological Society pp. 41–72 (1997)

[12] Iatridou, S., Zeijlstra, H.: Negation, polarity and deontic modals. Linguistic Inquiry 44, 529–568 (2013)

[13] Jäger, G.: Anaphora and Type-Logical Grammar. Springer, Berlin (2005)

[14] Johnson, K.: Few dogs eat Whiskas or cats Alpo. In: Kusumoto, K., Villalta, E. (eds.) University of Massachusetts Occasional Papers, vol. 23, pp. 47–60. GLSA, University of Massachusetts Amherst (2000)

[15] Kroch, A.: Function and grammar in the history of English: Periphrastic *do*. In: Fasold, R.W., Schriffin, D. (eds.) Language Change and Variation, pp. 132–172. Benjamins, Amsterdam (1989)

[16] Kubota, Y.: HPSG and categorial grammar. In: Abeillé, A., Borsley, R.D., Koenig, J.P., Müller, S. (eds.) Head-Driven Phrase Structure Grammar: The Handbook. Language Science Press (2021)

[17] Kubota, Y., Levine, R.: Gapping as like-category coordination. In: Béchet, D., Dikovsky, A. (eds.) Logical Aspects of Computational Linguistics 2012. pp. 135–150. Springer, Heidelberg (2012)

[18] Kubota, Y., Levine, R.: Gapping as hypothetical reasoning. Natural Language and Linguistic Theory 34(1), 107–156 (2016)

[19] Kubota, Y., Levine, R.: Pseudogapping as pseudo-VP ellipsis. Linguistic Inquiry 48(2), 213–257 (2017)
[20] Kubota, Y., Levine, R.: Type-Logical Syntax. MIT Press, Cambridge, MA (2020), available Open Access at https://direct.mit.edu/books/book/4931/Type-Logical-Syntax
[21] Kubota, Y., Levine, R.: NPI licensing and the logic of the syntax-semantics interface. Linguistic Research 38(2), 151–204 (2021)
[22] Lambek, J.: The mathematics of sentence structure. American Mathematical Monthly 65(3), 154–170 (1958)
[23] Martin, S., Pollard, C.: A dynamic categorial grammar. In: Morrill, G., Muskens, R., Osswald, R., Richter, F. (eds.) Proceedings of Formal Grammar 2014. pp. 138–154. Springer, Heidelberg (2014)
[24] Montague, R.: The proper treatment of quantification in ordinary English. In: Hintikka, J., Moravcsik, J.M.E., Suppes, P. (eds.) Approaches to Natural Language: Proceedings of the 1970 Stanford Workshop on Grammar and Semantics, pp. 221–242. Reidel, Dordrecht (1973)
[25] Moortgat, M.: Categorial type logics. In: van Benthem, J., ter Meulen, A. (eds.) Handbook of Logic and Language, pp. 95–179. Elsevier, Amsterdam, 2 edn. (2011)
[26] Morrill, G.: Type Logical Grammar: Categorial Logic of Signs. Kluwer, Dordrecht (1994)
[27] Morrill, G.: Categorial Grammar: Logical Syntax, Semantics, and Processing. Oxford University Press, Oxford (2010)
[28] Morrill, G., Merenciano, J.M.: Generalizing discontinuity. Traitement Automatique des Langues 27(2), 119–143 (1996)
[29] Morrill, G., Solias, T.: Tuples, discontinuity, and gapping in categorial grammar. In: Proceedings of the Sixth Conference of the European Chapter of the Association for Computational Linguistics. pp. 287–297. Association for Computational Linguistics, Morristown, NJ (1993)
[30] Morrill, G., Valentín, O., Fadda, M.: The displacement calculus. Journal of Logic, Language and Information 20(1), 1–48 (2011)
[31] Muskens, R.: Language, lambdas, and logic. In: Kruijff, G.J., Oehrle, R. (eds.) Resource Sensitivity in Binding and Anaphora, pp. 23–54. Kluwer, Dordrecht (2003)
[32] Oehrle, R.T.: Term-labeled categorial type systems. Linguistics and Philosophy 17(6), 633–678 (1994)
[33] Park, S.H.: Gapping: A Constraint-Based Syntax-Semantics Interface. Ph.D. thesis, State University of New York at Buffalo (2019)
[34] Pogodalla, S., Pompigne, F.: Controlling extraction in Abstract Categorial Grammars. In: de Groote, P., Nederhof, M.J. (eds.) Formal Grammar 2010/2011. pp. 162–177. Springer, Heidelberg (2012)
[35] Pollock, J.Y.: Verb movement, Universal Grammar, and the structure of IP. Linguistic Inquiry 20, 365–424 (1989)
[36] Potter, D., Frazier, M., Yoshida, M.: A two-source hypothesis for Gapping. Natural Language and Linguistic Theory 35, 1123–1160 (2017)

[37] Radford, A.: An Introduction to English Sentence Structure. Cambridge University Press, Cambridge, 2 edn. (2018)
[38] Sag, I., Chaves, R.P., Abeillé, A., Estigarribia, B., Eynde, F.V., Flickinger, D., Kay, P., Michaelis, L., Müller, S., Pullum, G., Wasow, T.: Lessons from the English auxiliary system. Journal of Linguistics 56, 87–155 (2019)
[39] Siegel, M.E.A.: Gapping and interpretation. Linguistic Inquiry 15(3), 523–530 (1984)
[40] Traugott, E.C.: From propositional to textual and expressive meanings: Some semantic-pragmatic aspects of grammaticalization. In: Lehmann, W.P., Malkiel, Y. (eds.) Perspectives on Historical Linguistics, pp. 245–271. Benjamins, Amsterdam (1982)
[41] Warner, A.: English Auxiliaries: Structure and History. Cambridge University Press, Cambridge (1993)

A A brief introduction to Hybrid Type-Logical Grammar

The main text assumes familiarity with Hybrid TLG [20]. This appendix provides a brief, self-contained introduction to Hybrid TLG so that readers who are not familiar with it can understand the analysis in this paper. For a more detailed exposition, see Kubota and Levine [20].

Hybrid TLG is essentially an extension of the Lambek calculus [22] with one additional, non-directional mode of implication. The full set of inference rules in Hybrid TLG are given in (33).

(33)

Connective	Introduction	Elimination
$/$	$\dfrac{\begin{array}{c}\vdots\ [\varphi;\ x;\ A]^n\ \vdots \\ \vdots\qquad\vdots\qquad\vdots \\ b \bullet \varphi;\ \boldsymbol{\mathcal{F}};\ B\end{array}}{b;\ \lambda x.\boldsymbol{\mathcal{F}};\ B/A}\ /\mathrm{I}^n$	$\dfrac{a;\ \boldsymbol{\mathcal{F}};\ A/B \quad b;\ \boldsymbol{\mathcal{G}};\ B}{a \bullet b;\ \boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{G}});\ A}\ /\mathrm{E}$
$\backslash$	$\dfrac{\begin{array}{c}\vdots\ [\varphi;\ x;\ A]^n\ \vdots \\ \vdots\qquad\vdots\qquad\vdots \\ \varphi \bullet b;\ \boldsymbol{\mathcal{F}};\ B\end{array}}{b;\ \lambda x.\boldsymbol{\mathcal{F}};\ A\backslash B}\ \backslash\mathrm{I}^n$	$\dfrac{b;\ \boldsymbol{\mathcal{G}};\ B \quad a;\ \boldsymbol{\mathcal{F}};\ B\backslash A}{b \bullet a;\ \boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{G}});\ A}\ \backslash\mathrm{E}$
$\restriction$	$\dfrac{\begin{array}{c}\vdots\ [\varphi;\ x;\ A]^n\ \vdots \\ \vdots\qquad\vdots\qquad\vdots \\ b;\ \boldsymbol{\mathcal{F}};\ B\end{array}}{\lambda\varphi.b;\ \lambda x.\boldsymbol{\mathcal{F}};\ B\restriction A}\ \restriction\mathrm{I}^n$	$\dfrac{a;\ \boldsymbol{\mathcal{F}};\ A\restriction B \quad b;\ \boldsymbol{\mathcal{G}};\ B}{a(b);\ \boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{G}});\ A}\ \restriction\mathrm{E}$

The key difference between /,\ and ↑ is that while the Introduction and Elimination rules for /,\ refer to the phonological forms of the input and output strings (so that, for example, the applicability of the /I rule is conditioned on the presence of the phonology of the hypothesis φ on the right periphery of the phonology of the input $b \bullet \varphi$), the rules for ↑ is not constrained that way.[14] For reasoning involving ↑, the phonological terms themselves fully specify the ways in which the output phonology is constructed from the input phonologies. Specifically, for ↑, the phonological operations associated with the Introduction and Elimination rules mirror exactly the semantic operations for these rules: function application and λ-abstraction, respectively. We assume that the binary connective $\bullet$ in the phonological term calculus represents the string concatenation operation and that $\bullet$ is associative in both directions. For notational convenience, we implicitly assume the axiom $(\varphi_1 \bullet \varphi_2) \bullet \varphi_2 \equiv \varphi_1 \bullet (\varphi_2 \bullet \varphi_3)$ and leave out all the brackets indicating the internal constituency of complex phonological terms.

Thus, the present system without the rules for ↑ is equivalent to the Lambek calculus (Lambek 1958), while the system with only the rules for ↑ is essentially equivalent to the family of approaches known as Linear Categorial Grammar [9, 31, 23], all of which are essentially direct descendants of Oehrle [32].

The latter component, namely, the lambda abstraction in the prosodic calculus is what distinguishes Hybrid TLG from other variants of TLG that are based on the Lambek calculus (such as Moortgat [25] and Morrill et al. [30]). As demonstrated by Oehrle [32], this enables a straightforward and formally explicit implementation of Montague's (1973) quantifying-in:

(34)
$$\dfrac{\dfrac{\dfrac{\dfrac{\mathsf{read};\ \mathbf{read};\ (\mathrm{NP}\backslash\mathrm{S})/\mathrm{NP} \quad [\varphi_2; x; \mathrm{NP}]^1}{\mathsf{read} \bullet \varphi_2;\ \mathbf{read}(x);\ \mathrm{NP}\backslash\mathrm{S}} \qquad \mathsf{john};\ \mathbf{j};\ \mathrm{NP}}{\mathsf{john} \bullet \mathsf{read} \bullet \varphi_2;\ \mathbf{read}(x)(\mathbf{j});\ \mathrm{S}}}{\lambda\varphi_2.\mathsf{john} \bullet \mathsf{read} \bullet \varphi_2;\ \lambda x.\mathbf{read}(x)(\mathbf{j});\ \mathrm{S}\uparrow\mathrm{NP}}\ \uparrow\mathrm{I}^1 \qquad \begin{array}{l}\lambda\sigma.\sigma(\mathsf{every} \bullet \mathsf{book});\\ \mathbf{\forall_{book}};\ \mathrm{S}\uparrow(\mathrm{S}\uparrow\mathrm{NP})\end{array}}{\mathsf{john} \bullet \mathsf{read} \bullet \mathsf{every} \bullet \mathsf{book};\ \mathbf{\forall_{book}}(\lambda x.\mathbf{read}(x)(\mathbf{j}));\ \mathrm{S}}$$

As (34) illustrates, quantifiers are entered in the lexicon in type $\mathrm{S}\uparrow(\mathrm{S}\uparrow\mathrm{NP})$, with the standard generalized quantifier meanings for their semantics and a phonology that is a higher-order function typed $(\mathbf{st} \to \mathbf{st}) \to \mathbf{st}$ (with $\mathbf{st}$ the type of strings), which 'lowers' the quantifier string in the position in the sentence (bound by the λ-operator in the phonology) corresponding to the semantic variable bound. As in Montague's quantifying-in, the order in which the quantifier combines with the sentence that it lowers into determines its scope.

In the analysis in the main text, we extend Oehrle's approach to scopal operators beyond the familiar domain of generalized quantifiers. In particular, we show how the behavior of English auxiliaries, long known to display puzzling and refractory relationships between their syntactic and semantic behavior, can

[14] In this respect, the proof system of Hybrid TLG follows most closely Morrill and Solias [29] and Morrill [26]; see Moortgat [25] and Bernardi [4] for an alternative formulation where sensitivity to directionality is mediated through a presumed correspondence between surface string and the form of structured antecedents in the sequent-style notation of natural deduction.

be explained through a treatment which parallels the way in which the scopal interaction of generalized quantifiers is accounted for.

On Type-Theoretical Semantics of Donkey Anaphora

Zhaohui Luo*

Royal Holloway, University of London
zhaohui.luo@hotmail.co.uk

Abstract. Donkey sentences are among the challenging examples that present a difficult problem in compositional logical semantics. Their semantic treatment is one of the early applications of dependent type theory to linguistic semantics, where the type constructor Σ is used to play a double role, both as the existential quantifier and as the subset constructor. However, it is known that this method is inadequate because it fails to deal with counting properly – this is also called the cardinality problem. In this paper, we analyse this problem to explicate that it originates from the use of Σ to play the double role and propose to consider the semantics of donkey sentences in a type theory with both Σ and a traditional existential quantifier. It is shown that, with both operators, donkey sentences can be given adequate semantic interpretations which, in particular, take care of counting properly.

1 Introduction

Donkey sentences, as first studied by Geach [14] and exemplified in (1), where an anaphoric expression refers to an existentially quantified entity, are among the challenging examples that present a difficult problem in compositional logical semantics.

(1) Every farmer who owns a donkey beats it.

Their studies (and that of trans-sentential anaphora) have led to the development of dynamic semantics such as Discourse Representation Theory (DRT) [21, 22, 17] and Dynamic Predicate Logic (DPL) [16], which are widely accepted by linguistic semanticists as a framework with proper means to deal with anaphora and, however, require us to consider substantial changes of the underlying logical systems for formal semantics.[1] (See [3], among others, for a recent summary of the dynamic approach to donkey anaphora.)

In the mid-80s, as one of the early applications of dependent type theory in logical semantics, researchers such as Mönnich [31] and Sundholm [39] have

* This work is partially supported by the EU research network EUTypes (COST Action CA15123).

[1] For example, DPL is a rather non-standard logical system: among other things, it is non-monotonic and the notion of dynamic entailment fails to be reflexive or transitive [16, 15].

proposed to use Martin-Löf's type theory [29] to deal with donkey anaphora, where Σ-types are employed to play a double role, representing the existential quantifier as well as the subset constructor. Σ-types $\Sigma x{:}A.P(x)$ are also called *strong sums*, as opposed to the traditional existentially quantified formulas $\exists x{:}A.P(x)$ which are called *weak sums*. They are called so because, from an object of the strong sum, one can obtain its witness by means of a projection operation, while this is not possible for the weak sum. It is because of the availability of witness projection that an anaphoric reference can be obtained from an object of a Σ-type, while this is not possible for an existential quantification in the traditional case (and hence the problem in the first place). However, it is known that this method of using Σ-types to deal with donkey anaphora suffers from a problem of counting [40, 41], also called the cardinality problem, and fails to provide us an adequate solution (see S 2 for more details).

In this paper, we contend that the cardinality problem of the above type-theoretical approach has come from a double role played by Σ, as an existential quantifier, on the one hand, and as a structural mechanism to represent collections (subsets) of objects, on the other. These two roles should be separated and played by different type constructors. But in traditional logics (for example, first-order logic or simple type theory) or in Martin-Löf's type theory, only either $\exists$ or Σ exists, not both, and therefore there is no way to consider such a separation of labour. We show that, in a type theory with both strong and weak sums, donkey sentences can be given adequate semantics in which counting is taken into proper account.

Our proposal is also linked to the research on different readings of donkey sentences and, in particular, the strong and weak readings as studied by Chierchia and others [7, 8, 23]. Also, donkey anaphora are closely related to (and, for some researchers, they are examples of) the so-called E-type anaphora, as first studied by Evans [12, 13], which may be interpreted by means of descriptions (see, for example, [33] for a recent discussion). It is not surprising that Σ-types are essentially useful in semantic interpretations of donkey sentences since they have close links to descriptions [29, 5, 30] and we shall give some brief discussions about this.

Combining strong and weak sums in type theory is a subtle matter that needs us to tread carefully, for otherwise we may easily slip into inconsistency or other problems. Although there are already some results in this respect [24], they are not widely known, partly because of their technical nature. We shall discuss this issue briefly, explaining both possibilities and potential problems.

In the following section S 2, we shall explain the concepts of strong and weak sums, define the notion of cardinality for finite types, and illustrate the counting/cardinality problem of the Σ-type interpretation of donkey sentences. S 3 describes our proposed solution with both strong and weak sums: in S 3.1, we first briefly describe the type structure of type theory UTT which has both Σ and $\exists$, followed by S 3.2 to explain how donkey sentences may be interpreted in UTT, and then by S 3.3 which considers E-type anaphora and the use of Σ-types for descriptions. In S 4, we shall briefly explain the possibilities and potential

problems in having both strong and weak sums in a type theory, followed by some concluding remarks, including that about considering dynamics in type theory.

2 Strong and Weak Sums in Type Theory

In this section, we explicate the concepts of weak sums (for example, traditional existential quantifiers), to illustrate the original problem of using existentially quantified variables to interpret donkey anaphora, and strong sums (Σ-types), to explain the counting problem when using Σ-types to play a double role in interpreting donkey sentences, as proposed by Mönnich [31] and Sundholm [39].

Weak sums (existential quantifiers). Under the Curry-Howard propositions-as-types principle [11, 20], traditional existentially quantified formulas are examples of weak sum types of the form $\exists x.P(x)$. In first-order logic, depending on whether it is intuitionistic or classical, the existential quantifier can be introduced directly or defined by means of the universal quantifier together with negation, respectively. In higher-order logic (or simple type theory) as used in Montague's semantics, where there is an impredicative type $\mathbf{t}$ of all formulas, it can be either directly introduced or defined by means of the universal quantifier as in (2), where x ranges over entities and X over formulas of type $\mathbf{t}$.[2]

(2) $\exists x.P(x) = \forall X.\ (\forall x.(P(x) \Rightarrow X)) \Rightarrow X.$

It is known that, given a proof of $\exists x.P(x)$, although one knows that there is an entity such that P holds, in the logical calculus one cannot find out which entity it is. It is because of this that an anaphoric reference to an existentially quantified entity becomes problematic. For example, in a traditional compositional semantics, the donkey sentence (1) would obtain (3) as its interpretation, where the free variable y in $beat(x, y)$ is out of the scope of the existential quantifier $\exists$ and is different from the bound variable y bounded by $\exists$.

(3) $(\#)\ \forall x.\ [farmer(x)\ \&\ \exists y.(donkey(y)\ \&\ own(x, y))] \Rightarrow beat(x, y)$

This illustrates the original problem in interpreting donkey sentences, as mentioned at the beginning of Introduction.

Strong sums (Σ-types). Σ is a dependent type constructor. If A is a type and B is a family of types that depend on objects of type A, then $\Sigma x{:}A.B(x)$ is a type, consisting of pairs (a, b) such that a is of type A and b is of type $B(a)$. Σ-types are associated with the projection operators π_1 and π_2 so that, for (a, b) of type $\Sigma x{:}A.B(x)$, $\pi_1(a, b) = a$ and $\pi_2(a, b) = b$. Formally, Σ-types are governed by the inference rules in Appendix A.

[2] The fact that other logical operators can be defined in higher-order logical systems by means of universal quantifier was discovered in the 60s by Prawitz [34] (and several others, independently) and, this is the same in an impredicative type theory – see definitions in (8-9) and Appendix C.

Besides being useful mechanisms to organise structures in various applications, Σ-types may also play other roles. For example, in Martin-Löf's type theory [29], Σ also plays the role of existential quantifier in its logic.[3] This is the basis for Mönnich [31] and Sundholm [39] to propose using Σ-types to interpret donkey sentences.[4] For instance, the donkey sentence (1) can be interpreted as (4), in which F_Σ, as defined in (5), is the type intended to represent the collection of donkey-owning farmers, where F and D are the types that interpret farmer and donkey, respectively.

(4) $\forall z : F_\Sigma.\ beat(\pi_1(z), \pi_1(\pi_2(z)))$

(5) $F_\Sigma = \Sigma x{:}F\ \Sigma y{:}D.\ own(x, y)$

Σ-types are strong in the sense that from a proof of $\Sigma x{:}A.P(x)$ one can preform the first projection operation to obtain the witness of this 'existentially' quantified formula and it is because of this, if Σ is used as existential quantifier, one can project out its witness from a proof term of the Σ-type, even outside the Σ-type concerned (the terms $\pi_1(z)$ and $\pi_1(\pi_2(z))$ in (4) are such examples).

The type F_Σ in (5) contains two occurrences of Σ and they play two different roles: the first acts as a structural mechanism to represent the collection of the farmers who own donkeys and the second as the existential quantifier to say that there exists a donkey owned by the farmer concerned. As we shall see below, using Σ to play this double role is problematic. In particular, F_Σ is in fact representing a collection whose cardinality (the number of its objects) is different from that of the collection of donkey-owning farmers and, therefore, the semantic interpretation (4) of (1) is inadequate [40, 41].

Counting and cardinality of finite types. When a type A is finite in the sense that it has finitely many objects, it is possible to define its cardinality $|A|$ as the number of its objects. Formally, a type is finite if, for some n, it is isomorphic to $Fin(n)$, the type with exactly n objects – see Appendix B. For example, the cardinality of a finite Σ-type is the number of pairs in the type.

The problem of counting (or the cardinality problem) can be illustrated by considering the sentence in (6), where the quantifier Every in (1) is replaced by Most. Its formal semantics by means of Σ-types in Martin-Löf's type theory, as proposed by Mönnich [31] and Sundholm [39], is given in (7), which can be seen as obtained by replacing $\forall$ by the quantifier $Most_S$. Here, $Most_S$ is defined by Sundholm in [40] (S in $Most_S$ for Sundholm) so that, for a finite type A, $Most_S\ x{:}A.P(x)$ is true if, and only if, more than half of the objects in A satisfy P.

[3] This is concerned with intuitionistic philosophy – a strongly minded intuitionist may believe that the witness of a proven existentially quantified formula can be obtained internally in a logical calculus. We omit further discussions here.

[4] In formal semantics based on modern type theories, CNs such as 'farmer' and 'donkey' are interpreted as types (rather than predicates). This was first proposed by Mönnich [31] and Sundholm [39] and further elaborated in [36, 25].

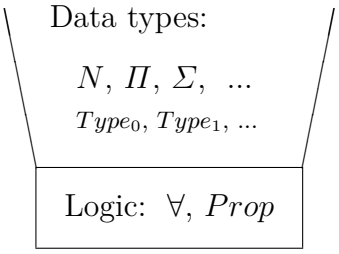


Fig. 1. The type structure in UTT.

(6) Most farmers who own a donkey beat it.

(7) $Most_S\ z : F_\Sigma.\ beat(\pi_1(z), \pi_1(\pi_2(z)))$, where F_Σ is defined in (5).

Let us now consider the cardinality of F_Σ, as defined in (5). Because of the second Σ in F_Σ, $|F_\Sigma|$ is not that of the collection of donkey-owning farmers; instead, to calculate $|F_\Sigma|$, we'd have to count every triple (x, y, p) of farmers x, donkeys y and proofs p that x owns y. For example, if there are ten farmers, one of whom owns twenty donkeys and beats all of them, and the other nine own one donkey each and do not beat their donkeys. Then, $|F_\Sigma| \geq 29$ (it is an inequality because, if farmer x owns donkey y, there may be more than one proof that x owns y), but the number of farmers who do not beat their donkeys is 9. Therefore, the above semantics (7) of (6) would be true in such a case, which is obviously incorrect.[5]

3 Donkey Anaphora: Type-Theoretical Semantics with Both Strong and Weak Sums

In this section, we shall first introduce the dependent type theory UTT (Unifying Theory of dependent Types) [24], which has both strong and weak sums, and then show how donkey sentences like (1) and (6) can be interpreted in UTT, giving adequate treatments for different readings and taking care of counting in a proper way as well.

3.1 UTT: an impredicative type theory

The type structure of UTT consists of two parts: the world of data types and that of logical propositions (see Fig. 1). It contains various types such as dependent product types (Π-types), strong sum types (Σ-types), the type N of natural

[5] This is similar to the 'proportion problem' when one uses DRT to interpret such donkey sentences, where one counts farmer-donkey pairs rather than the donkey-owning farmers. See [23] and [3], among others, for discussions.

numbers, the predicative universes $Type_i$ ($i \in \omega$), and many others. UTT also contains an impredicative type universe $Prop$ of logical propositions which provide means to describe logical properties of objects of any type (see Appendix C). Formally, UTT can be considered as the combination of Martin-Löf's (intensional) type theory [28, 32] with Coquand-Huet's Calculus of Constructions [10]. In computer science, type theories such as UTT have been implemented in theorem proving systems (called proof assistants) for formalisation of mathematics and verification of programs, and recently, they have been used for formal reasoning based on linguistic semantics (see, for example, [6]).[6]

Note that UTT contains both strong sums $\Sigma x{:}A.B(x)$ (Σ-types) as 'data types' and weak sums $\exists x{:}A.P(x)$ (existentially quantified types) as logical propositions, and this is essential when considering semantic interpretations of donkey sentences in S 3.2 below.

Logic and proof irrelevance. In UTT, a type is a logical proposition if it is of type $Prop$. The type universe $Prop$ is impredicative and, therefore, the other logical operators can be defined by means of the operator $\forall$ for universal quantification (cf., Footnote 2). For example, the conjunction operator and the existential quantifier $\exists$ can be defined as in (8) and (9), respectively, and the definitions of the other operators can be found in Appendix C.

(8) $P \wedge Q = \forall X : Prop.\ (P \Rightarrow Q \Rightarrow X) \Rightarrow X$

(9) $\exists x : A.P(x) = \forall X{:}Prop.(\forall x : A.(P(x) \Rightarrow X)) \Rightarrow X$

The principle of *proof irrelevance* says that any two proofs of the same logical proposition should be the same. For instance, it implies that, for farmer x and donkey y, any two proof terms of the proposition $own(x, y)$ should be the same. It has been shown that, when employing a type theory for natural language semantics, proof irrelevance should be enforced [25, 26]. Note that, because in UTT there is a clear distinction between logical propositions and other types (the former being those of type $Prop$), it is straightforward to introduce proof irrelevance by means of the following rule [43, 25]:

$$\frac{P : Prop \quad p : P \quad q : P}{p = q : P}$$

Intuitively, it says that, if P is a logical proposition and if p and q are proof terms of P, then p and q are the same. In particular, according to the above rule, every proposition of type $Prop$ is either an empty type or a singleton type. In terms of cardinality, we have $|P| \leq 1$ for every $P : Prop$ and, therefore, if A is finite and $Q : A \to Prop$ is a predicate over A, then we have

(10) $|\Sigma x{:}A.Q(x)| \leq |A|$.

[6] There are several proof assistants based on type theories including Agda [1] based on Martin-Löf's type theory, Coq [9] implementing the type theory pCIC, and Lego/Plastic [27, 4] implementing UTT. It may be worth remarking that Coq's type system pCIC is very similar to UTT – this is especially the case after Coq's universe Set became predicative in 2004 (it was impredicative in earlier versions).

3.2 Semantic interpretations of donkey anaphora in UTT

When a type theory has both strong and weak sums (Σ-types and $\exists$-propositions as in UTT), together with proof irrelevance, there is a new way to semantically interpret donkey sentences, which takes care of counting adequately. We'll use the example (6), which is repeated as (11) below, to explain.

(11) Most farmers who own a donkey beat it.

In S 2, we have shown that, because in Martin-Löf's type theory Σ is used to play a double role, the semantic interpretation (7) of (11) is inadequate because it gets counting wrong. In that definition, we have used quantifier $Most_S$ defined in Martin-Löf's type theory and, here, we can define a semantic interpretation of the quantifier 'most' in UTT in a similar fashion as in [40] but with a crucial difference: instead of Σ, we shall use $\exists$ as defined in (9) as the existential quantifier and, intuitively, for a finite A, $Most\ x{:}A.P(x)$ also means that more than half of the objects in A satisfy P. Note that, $Most_S\ x{:}A.P(x)$ is a non-propositional type, but $Most\ x{:}A.P(x)$ is a logical proposition of type $Prop$ (see Appendix D for the formal definition).

Having defined *Most* in UTT, we can now interpret the donkey sentence (11) as (12), in which $F_\exists$ is defined in (13):

(12) $Most\ z : F_\exists.\ \forall y' : \Sigma y{:}D.own(\pi_1(z), y).\ beat(\pi_1(z), \pi_1(y'))$

(13) $F_\exists = \Sigma x{:}F.\ \exists y{:}D.own(x, y)$

Note that $|\exists y{:}D.own(x, y)| \leq 1$, that is, if $\exists y{:}D.own(x, y)$ is true, the cardinality of the proposition is 1. Therefore, the type $F_\exists$ correctly represents the collection of donkey-owning farmers, as intended, and the above semantics (12) is adequate and, in particular, it deals with counting correctly.

Researchers have studied different readings (in particular, strong and weak readings) of donkey anaphora, as studied by Chierchia [7, 8] and others. For instance, the strong and weak readings of (11) are (14) and (15), respectively:

(14) Most farmers who own a donkey beat *the donkeys they own*.

(15) Most farmers who own a donkey beat *some donkeys they own*.

The above interpretation (12) of (11) is a strong one, interpreting (14) directly: most donkey-owning farmers beat *all* donkeys they own. A weaker interpretation of its weak reading (15) would be (16), obtained from (12) by changing $\forall$ into $\exists$:

(16) $Most\ z : F_\exists.\ \exists y' : \Sigma y{:}D.own(\pi_1(z), y).\ beat(\pi_1(z), \pi_1(y'))$

People have also considered more sophisticated examples with donkey anaphora. For example, (17) is one of them, taken from Brasoveanu's thesis [2, 3], in which the readings for the donkey anaphora are different ('a TV' having a strong reading and 'a credit card' a weak one). Its type-theoretical semantics with both strong and weak sums is given in (18).

(17) Every person who buys a TV and has a credit card uses it to pay for it.

(18) $\forall z : \Sigma x{:}Person.\ \exists y_1{:}TV.\ buy(x, y_1) \wedge \exists y_2{:}Card.\ own(x, y_2)$
$\forall y : \Sigma y_1{:}TV.\ buy(\pi_1(z), y_1)$
$\exists y' : \Sigma y_2{:}Card.\ own(\pi_1(z), y_2).$
$pay(\pi_1(z), \pi_1(y), \pi_1(y'))$

One may change the quantifier Every in (17) into Most (and make other minor changes in the sentence to make it grammatically correct by changing, for example, 'person/buys/has/uses' into 'persons/buy/have/use') and, in that case, we can use the quantifier *Most* defined in UTT (see Appendix D) to interpret the sentence and the resulting interpretations take care of counting correctly as well.

3.3 E-type anaphora

Here, we discuss, albeit rather briefly, the so-called E-type anaphora to which donkey anaphora are closely related (and, for some researchers, donkey anaphora are examples of E-type anaphora).[7] E-type anaphora are first studied by Evans [12, 13], and further discussed by many, including [18] among others. They can be interpreted by means of descriptions [37, 38] (see, for example, [33] for a recent discussion). An example, due to Evans, is (19). Note that the pronoun 'they' in (19) is not bound by 'Few' for otherwise the semantic interpretation of the sentence would be incorrect. A common conceptual answer, proposed by Evans, is that these pronouns are *descriptive* in that they can be paraphrased by means of descriptions as exemplified in (20) that paraphrases (19).

(19) Few congressmen admire Kennedy, and they are very junior.

(20) Few congressmen admire Kennedy, and *the congressmen that do admire Kennedy* are very junior.

As pointed out by Martin-Löf [29], strong sum types (Σ-types) are related to descriptions, because he regards them as logical propositions as well. If one considers $\Sigma x{:}A.B(x)$ as the existentially quantified formula and because it is strong, therefore its first projection operator π_1 gives us an internal means of obtaining the witness from a proof of the existentially quantified formula. As explained in S 2, this is stronger than the traditional existential operator $\exists$ for which such a projection operator does not exist, and it is exactly because of this that Σ offers a form of description, as pointed out by Martin-Löf and further studied by Carlström [5] and Mineshima [30]. For example, the E-type example (19) may be interpreted as (21), either in Martin-Löf's type theory or in UTT, where we assume that the quantifier Few has been defined and C is the type that interprets 'congressman':

(21) $Few\ x{:}C.admire(x, K) \wedge \forall z{:}[\Sigma x{:}C.admire(x, K)].junior(\pi_1(z))$

However, it should be made clear that, since Σ-types are not the same as traditional existentially quantified formulas, it is unclear how far one may go

[7] Here, I use the term 'E-type' for a kind of anaphora, rather than an approach to solving anaphora ('the E-type approach' as people often put it).

in analysing E-type anaphora by means of Σ-types. Actually, it would not go very far since, as analysed above, using Σ as existential quantifier does cause problems such as counting, which would show up in context of E-type anaphora as well.

4 Combining Strong and Weak Sums in Type Theories

It is worth mentioning that combining both strong and weak sums at the same time in a type theory is a subtle matter and, if not careful, it is easy to get into problems. We summarise some of the known results in this section so that the readers can become aware of them. However, because of the technical nature of the results (and their proofs), we will only be brief and sketch some of them briefly and informally.[8]

Adding strong sums to impredicative type theories. First, let's consider how to add strong sums (Σ-types) to an impredicative type theory, where one already has the weak sum ($\exists$-propositions). Note that, as briefly described in S 3.1, although it has both Σ-types (as data types) and $\exists$-propositions (as logical formulas), UTT does not have 'Σ-propositions' because the so-called 'large Σ-propositions' would lead to inconsistency and the so-called 'small Σ-propositions' would make the weak sum types become strong. Consider, for example, to add large Σ-propositions into the impredicative universe $Prop$ by means of the following rule (together with those for introduction and eliminations that we omit):

$$(*) \qquad \frac{A\ type \quad P : A \to Prop}{\Sigma x{:}A.P(x) : Prop}$$

It turns out that such Σ-propositions cannot be consistently added – if they were added using the above rule $(*)$ (and related ones), the resulting type theory would be inconsistent in the sense that even the false proposition would become provable [19, 24].

One may want to add Σ-propositions (so-called small Σ-propositions) by a rule like the following, this time restricting A to be a proposition of type $Prop$:

$$\frac{A : Prop \quad P : A \to Prop}{\Sigma x{:}A.P(x) : Prop}$$

Although the resulting type theory may be consistent[9], there is another problem: the addition of such small strong sum as propositions in $Prop$ would make the weak sum proposition $\exists x{:}A.P(x)$ become strong (rather unexpectedly!) in the sense that there is now an internal function in the type theory that, from a proof

[8] These results, except that about MLTT_h, are discussed by the author in [24] (S 2.3.2 of [24], in particular), from which the interested reader may obtain more information.

[9] This consistency is a folklore – most researchers, including the author, believe that it is the case, although the author has not seen a proof of it.

of $\exists x{:}A.P(x)$, returns an object $a : A$ such that $P(a)$ holds. That would mean that the traditional existential quantifier is not weak anymore – such a side effect is of course problematic and, in particular, would make the interpretation method we proposed above in S 3.2 fail to deal with counting correctly.

Therefore, neither of the above large or small Σ-propositions is a viable possibility and, put in another way, the approach taken in UTT seems to be the only viable approach to adding Σ to an impredicative type theory.

Adding weak sums to predicative type theories. Now, let's consider how to add weak sums to a predicative type theory such as Martin-Löf's type theory MLTT [28, 32],[10] where we already have strong sum types (Σ-types).

We first remark that defining a form of 'weak sum types' in a similar way as the definition in (9), but with the difference of replacing the impredicative universe *Prop* by a predicative universe U_i in MLTT, would not work. This would amount to the definition in (22), where $A : U_i$ and $B : A \to U_i$:

$$(22) \quad \exists_i x{:}A.B(x) = \Pi X{:}U_i.\ (\Pi x{:}A.(B(x) \to X)) \to X$$

The above definition (22) does not work because it fails to deliver a weak sum type – $\exists_i x{:}A.B(x)$ thus defined is actually strong (!) and, in fact, it is equivalent to the strong sum type $\Sigma x{:}A.B(x)$ (see S 2.3.2 of [24] for a proof).

Therefore, it seems that the only possible way to add a weak sum type to MLTT is to consider MLTT_h, as proposed by the author in [26], where MLTT is extended with the h-logic for the HoTT type theory [35]. In MLTT_h, in particular, the existentially quantified formula is defined to be the truncation of the corresponding Σ-type, as in (23). (We omit the details here and interested readers may consult [26] and the references about HoTT [35] there.)

$$(23) \quad \dot{\exists} x{:}A.B(x) = |\Sigma x{:}A.B(x)|$$

The existential quantifier $\dot{\exists}$ is a weak sum and, therefore, can be used to define the semantic interpretations of donkey sentences as proposed in S 3.2. For example, employing MLTT_h (instead of UTT) for our formal semantics, the donkey sentence (11) can be interpreted as (24), where $Most_h$ is defined as in Appendix D, but in MLTT_h, with $\dot{\exists}$ to replace $\exists$ in the definition.

$$(24) \quad Most_h\ z : F_{\dot{\exists}}.\ \forall y' : \Sigma y{:}D.own(\pi_1(z), y).\ beat(\pi_1(z), \pi_1(y'))$$

$$(25) \quad F_{\dot{\exists}} = \Sigma x{:}F.\ \dot{\exists} y{:}D.own(x, y)$$

5 Concluding Remarks

In this paper, we have studied how to consider donkey anaphora in type theory, pointing out that the cardinality problem in the proposed type-theoretic semantics in Martin-Löf's type theory is due to the fact that strong sums (Σ-types) are

[10] We use MLTT to denote Martin-Löf's *intensional* type theory as described in [28] or Part III of [32], not his extensional type theory [29].

used to play a double role in the representation. We then show that, in a type theory (like UTT) with both strong and weak sums, donkey sentences can be given adequate semantics which, in particular, does not suffer from the cardinality problem. The paper then briefly discussed the issue of how to incorporate both strong and weak sums in type theory: it explains that, for impredicative type theories, it seems that UTT provides the only viable approach and, for predicative type theories, a proposal of studying MLTT_h provides a promising way forward.

It is worth pointing out that our analysis and proposal are given in a completely proof-theoretic fashion. This is rather different from the model-theoretic approaches that have been considered in the literature (see, for example, [3]). This work may be taken as a part of the more general endeavour of studying a type-theoretic approach to dynamics in semantics. Two remarks are in order here. The first is about a possible suggestion to extend a type theory into a 'dynamic type theory', just like extending the first-order logic to become dynamic predicate logic [16]. We do not think that this is a right way forward partly because, even if such a 'dynamic type theory' is possible (a big if), for type theory to lose its standard properties to become a non-standard logical system is too much a price to pay (cf., Footnote 1). Secondly, existing work on proof theory of dynamic semantic systems like DPL (see, for example, [42]) has shown the difficulties in doing proof theory for such non-standard systems. In logical semantics, it would be preferable, if we can, to stay with more standard logical systems.

Acknowledgement. I am very grateful to Justyna Grudziñska for a discussion about (6), the example involving 'most', which has motivated me to start the research reported in this paper. My thanks also go to the anonymous reviewers for their useful comments that have helped to improve the paper.

Bibliography

[1] The Agda proof assistant (version 2). Available from the web page: `http://appserv.cs.chalmers.se/users/ulfn/wiki/agda.php` (2008)

[2] Brasoveanu, A.: Structured Nominal and Modal Reference. Ph.D. thesis, The State University of New Jersey (2007)

[3] Brasoveanu, A., Dotlacil, J.: Donkey anaphora: farmers and bishops. In D. Gutzmann et al. (eds), The Wiley Blackwell Companion to Semantics (2021)

[4] Callaghan, P., Luo, Z.: An implementation of LF with coercive subtyping and universes. Journal of Automated Reasoning 27(1), 3–27 (2001)

[5] Carlström, J.: Interpreting descriptions in intensional type theory. The Journal of Symbolic Logic 70(2) (2005)

[6] Chatzikyriakidis, S., Luo, Z.: Proof assistants for natural language semantics. In: Amblard, M., de Groote, P., Pogodalla, S., Retoré, C. (eds.) Proceedings of Logical Aspects of Computational Linguistics LACL 2016. pp. 85–98. No. 10054 in Lecture Notes in Computer Science, Springer, Nancy, France (2016)

[7] Chierchia, G.: Anaphora and dynamic logic. ITLI Publication Series for Logic, Semantics and Philosophy of Language, LP90-07 (1990)

[8] Chierchia, G.: Anaphora and dynamic binding. Linguistics and Philosophy 15 (1992)

[9] The Coq Development Team: The Coq Proof Assistant Reference Manual (Version 8.3), INRIA (2010)

[10] Coquand, T., Huet, G.: The calculus of constructions. Information and Computation 76(2-3), 95–120 (1988)

[11] Curry, H., Feys, R.: Combinatory Logic, vol. 1. North Holland Publishing Company (1958)

[12] Evans, G.: Pronouns, quantifiers and relative clauses. Canadian Journal of Philosophy 7(2) (1977)

[13] Evans, G.: Pronouns. Linguistic Inquiry 11(2) (1980)

[14] Geach, P.: Reference and Generality: An Examination of Some Medieval and Modern Theories. Cornell University Press (1962)

[15] Gillies, A.: (Re-)reading 'dynamic predicate logic' (electronic manuscript)

[16] Groenendijk, J., Stokhof, M.: Dynamic predicate logic. Linguistics and Philosophy pp. 39–100 (1991)

[17] Heim, I.: The Semantics of Definite and Indefinite Noun Phrases. Ph.D. thesis, University of Massachusetts (1982)

[18] Heim, I., Kratzer, A.: Semantics in Generative Grammar. Blackwell (1998)

[19] Hook, J., Howe, D.: Impredicative strong existential equivalent to Type:Type. Technical Report TR86-760, Cornell University (1986)

[20] Howard, W.A.: The formulae-as-types notion of construction. In: Hindley, J., Seldin, J. (eds.) To H. B. Curry: Essays on Combinatory Logic. Academic Press (1980), (Notes written and distributed in 1969.)

[21] Kamp, H.: A theory of truth and semantic representation. In J. Groenendijk et al (eds.) Formal Methods in the Study of Language pp. 189–222 (1981)

[22] Kamp, H., Reyle, U.: From Discourse to Logic. Kluwer (1993)
[23] Kanazawa, M.: Weak vs. strong readings of donkey sentences and monotonicity inference in a dynamic setting. Linguistics and Philosophy 17(2) (1994)
[24] Luo, Z.: Computation and Reasoning: A Type Theory for Computer Science. Oxford University Press (1994)
[25] Luo, Z.: Common nouns as types. In: Bechet, D., Dikovsky, A. (eds.) Logical Aspects of Computational Linguistics (LACL'2012). LNCS 7351. pp. 173–185 (2012)
[26] Luo, Z.: Proof irrelevance in type-theoretical semantics. Logic and Algorithms in Computational Linguistics 2018 (LACompLing2018), Studies in Computational Intelligence (SCI) 1–15, 1–15 (2019), springer.
[27] Luo, Z., Pollack, R.: LEGO Proof Development System: User's Manual. LFCS Report ECS-LFCS-92-211, Dept of Computer Science, Univ of Edinburgh (1992)
[28] Martin-Löf, P.: An intuitionistic theory of types: predicative part. In: H.Rose, J.C.Shepherdson (eds.) Logic Colloquium'73 (1975)
[29] Martin-Löf, P.: Intuitionistic Type Theory. Bibliopolis (1984)
[30] Mineshima, K.: Aspects of Inference in Natural Language. Ph.D. thesis, Keio University (2013)
[31] Mönnich, U.: Untersuchungen zu einer konstruktiven Semantik fur ein Fragment des Englischen. Habilitation. University of Tübingen (1985)
[32] Nordström, B., Petersson, K., Smith, J.: Programming in Martin-Löf's Type Theory: An Introduction. Oxford University Press (1990)
[33] Nouwen, R.: E-type pronouns: congressmen, sheep and paychecks. In D. Gutzmann et al. (eds), The Wiley Blackwell Companion to Semantics (2021)
[34] Prawitz, D.: Natural Deduction, a Proof-Theoretic Study. Lmqvist and Wiksell (1965)
[35] Program, T.U.F.: Homotopy type theory: Univalent foundations of mathematics. Tech. rep., Institute for Advanced Study (2013)
[36] Ranta, A.: Type-Theoretical Grammar. Oxford University Press (1994)
[37] Russell, B.: On denoting. Mind 14(56) (1905)
[38] Russell, B.: Introduction to Mathematical Philosophy. George Allen and Unwin (1919)
[39] Sundholm, G.: Proof theory and meaning. In: Handbook of philosophical logic, pp. 471–506. Springer (1986)
[40] Sundholm, G.: Constructive generalized quantifiers. Synthese 79(1), 1–12 (1989)
[41] Tanaka, R.: Generalized quantifiers in dependent type semantics (2015), talk given at Ohio State University
[42] Veltman, F.: Proof systems for Dynamic Predicate Logic (2000)
[43] Werner, B.: On the strength of proof-irrelevant type theories. Logical Methods in Computer Science 4(3) (2008)

A Rules for Σ-types

$$\frac{A\ type \quad B\ type\ [x{:}A]}{\Sigma x{:}A.B\ type}$$

$$\frac{a : A \quad b : [a/x]B \quad B\ type\ [x{:}A]}{(a,b) : \Sigma x{:}A.B}$$

$$\frac{p : \Sigma x{:}A.B}{\pi_1(p) : A} \qquad \frac{p : \Sigma x{:}A.B}{\pi_2(p) : [\pi_1(p)/x]B}$$

$$\frac{a : A \quad b : [a/x]B \quad B\ type\ [x{:}A]}{\pi_1(a,b) = a : A} \qquad \frac{a : A \quad b : [a/x]B \quad B\ type\ [x{:}A]}{\pi_2(a,b) = b : [a/x]B}$$

B Cardinality of Finite Types

We give the formal definition of finite types. It will use the auxiliary type $Fin(n)$ for which we define first.

The type $Fin(n)$, indexed by $n : N$ with N being the type of natural numbers, consists of exactly n objects and can be specified by means of the following introduction rules (we omit their elimination and computation rules):

$$\frac{n : N}{zero(n) : Fin(n+1)}$$

$$\frac{n : N \quad i : Fin(n)}{succ(n,i) : Fin(n+1)}$$

The cardinality of a finite type A, notation $|A|$, is defined to be n if, and only if, A is isomorphic to $Fin(n)$, that is, in the type theory concerned, there is a bijective function between A and $Fin(n)$. In particular, $|Fin(n)| = n$, since the identity function over $Fin(n)$ is bijective.

C Logic in UTT

The logic in UTT[11] consists of the impredicative universe $Prop$, specified by the following rules:

$$\frac{}{Prop\ type} \qquad \frac{P : Prop}{P\ type}$$

and the operator $\forall$ for universal quantification, specified by

$$\frac{A\ type \quad P : Prop\ [x{:}A]}{\forall x{:}A.P : Prop}$$

[11] One can find its definition in S 9.2.1 of [24], where it is specified in terms of the logical framework LF.

$$\frac{x{:}A \vdash b : P \quad P : Prop\ [x{:}A]}{\lambda x{:}A.b : \forall x{:}A.P}$$

$$\frac{f : \forall x{:}A.P \quad a : A}{f(a) : [a/x]P}$$

$$\frac{P : Prop\ [x{:}A] \quad b : P\ [x{:}A] \quad a : A}{(\lambda x{:}A.b)(a) = [a/x]b : [a/x]P}$$

In UTT, other logical operators can be defined by means of $\forall$ and here are some definitions (see, for example, S 5.1 of [24]):

$$\begin{aligned}
P \Rightarrow Q &= \forall x : P.\ Q \\
\mathbf{true} &= \forall X : Prop.\ X \Rightarrow X \\
\mathbf{false} &= \forall X : Prop.\ X \\
P\ \wedge\ Q &= \forall X : Prop.\ (P \Rightarrow Q \Rightarrow X) \Rightarrow X \\
P\ \vee\ Q &= \forall X : Prop.\ (P \Rightarrow X) \Rightarrow (Q \Rightarrow X) \Rightarrow X \\
\neg P &= P \Rightarrow \mathbf{false} \\
\exists x : A.P(x) &= \forall X : Prop.\ (\forall x : A.(P(x) \Rightarrow X)) \Rightarrow X \\
(a =_A b) &= \forall P : A \to Prop.\ P(a) \Rightarrow P(b)
\end{aligned}$$

D *Most* in UTT

Let A be a finite type with $|A| = n_A$, $P : A \to Prop$ a predicate over A, and $Fin(n)$ the types with n objects defined in Appendix B. Then, in UTT, the logical proposition $Most\ x{:}A.P(x)$ of type $Prop$ is defined as follows, where $inj(f)$ is a proposition expressing that f is an injective function:

$$\begin{aligned}
Most\ x{:}A.P(x) = \exists k : N.\ &(k \geq \lfloor n_A/2 \rfloor + 1) \\
&\wedge \exists f{:}Fin(k) \to A.\ inj(f) \wedge \forall x{:}Fin(k).P(f(x))
\end{aligned}$$

Vector Space Semantics for Lambek Calculus with Soft Subexponentials

Lachlan McPheat, Hadi Wazni, and Mehrnoosh Sadrzadeh

University College London, UK
{l.mcpheat, hadi.wazni.20, m.sadrzadeh}@ucl.ac.uk

Abstract. We develop a vector space semantics for Lambek Calculus with Soft Subexponentials, apply the calculus to construct compositional vector interpretations for parasitic gap noun phrases and discourse units with anaphora and ellipsis, and experiment with the constructions in a distributional sentence similarity task. As opposed to previous work, which used Lambek Calculus with a Relevant Modality the calculus used in this paper uses a bounded version of the modality and is decidable. The vector space semantics of this new modality allows us to meaningfully define contraction as projection and provide a linear theory behind what we could previously only achieve via nonlinear maps.

1 Introduction

The origin of logics that add copying and movement modalities to the Lambek calculus can be traced back to the work of Morrill et al [1], who used restrictions on linear logic's permutation and contraction rules in order to reason about larger fragments of natural language. Modalities were also added to Lambek calculus by Moortgat [22] by means of indexed families of connectives that would primarily control different forms of associativity. Later Moot explored the use of indexed modalities in linear logic and developed proof nets for them [23], a modality for controlled permutation was amongst this set of indexed modalities. In [6], Jäger presented a multimodal Lambek calculus that modelled anaphora and ellipsis. Later in his book [7], he put forward an alternative unimodal edition for the same phenomena. The preferred semantics for these logics has always been a traditional Montague-style semantics, until a vector space interpretation was provided in [28]. Here, the authors develop a vector space interpretation for a Lambek calculus with restricted permutation and contraction rules and test the efficacy of a Jäger style treatment of verb-phrase ellipsis on a disambiguation task of distributional semantics [3]. This work led to a unified type-logical distributional framework for co-reference resolution in natural language [27], but its underlying logic was undecidable and also could not distinguish between the strict and sloppy readings of anaphora with ellipsis examples.

Lambek calculus with a Relevant Modality $!\mathbf{L}^*$ [11] is a recent logic which follows suit from the work of Jäger and Morrill. In previous work, we developed sound categorical and vector space semantics for this logic [19]. We demonstrated

how it can model anaphora and ellipsis and can also distinguish between the sloppy vs strict readings [20]. Our categorical semantics $\mathcal{C}(!\mathbf{L}^*)$ is a monoidal biclosed category $(\mathcal{C}, \otimes, I, \Leftarrow, \Rightarrow)$, whose objects are the formulas of $!\mathbf{L}^*$ and whose morphisms are its derivable sequents. We require this category to be equipped with a coalgebra modality [2] i.e. a monoidal comonad $(!, \delta, \varepsilon)$ on $\mathcal{C}$ and with diagonal natural transformation $\Delta : ! \to ! \otimes !$. This is the modality responsible for interpreting contraction. We further require that the coalgebra modality makes the objects of the category permute, i.e. we require a natural isomorphism $1_{\mathcal{C}} \otimes ! \cong ! \otimes 1_{\mathcal{C}}$. The vector space semantics of $!\mathbf{L}^*$, again from [19], is a monoidal biclosed functor $[\![\ ]\!]$ from $\mathcal{C}(!\mathbf{L}^*)$ to the category of finite dimensional real vector spaces, **fdVect**, maping formulas A of $!\mathbf{L}^*$ (objects of $\mathcal{C}(!\mathbf{L}^*)$) to finite dimensional real vector spaces $[\![A]\!]$, and derivable sequents $\Gamma \longrightarrow A$ of $!\mathbf{L}^*$ (morphisms of $\mathcal{C}(!\mathbf{L}^*)$) to linear maps $[\![\Gamma]\!] \to [\![A]\!]$. We instantiated the categorical model in **fdVect** by interpreting ! in several different ways, the most classical being a Fermionic Fock Space functor, originally used in [25] to interpret the ! modality of linear logic.

The calculus we depended on for this work, $!\mathbf{L}^*$, however, was proven undecidable in the same paper it was introduced which has posed a challenge for all the work we have done on top of it. Thankfully, the creators of $!\mathbf{L}^*$ have since defined a new logic with a similar expressive power but with a decidable fragment, referred to by **SLLM** [10]. In **SLLM** permutation and contraction (now called 'multiplexing') are separated into two different modalities, whose logical rules are inspired by Light and Soft linear logic [4, 14]. It is this logic we soundly interpret in **fdVect** in this paper in the style of [19]. In doing so, we also managed to produce an interpretation for copying vectors, e.g. a vector $\overrightarrow{v}$ can now essentially be fully copied into two vectors $\overrightarrow{v} \otimes \overrightarrow{v}$ using a simple projection that was made possible due to the presence of accessible bounds. The map $\overrightarrow{v} \mapsto \overrightarrow{v} \otimes \overrightarrow{v}$ is non linear, and in previously we had to be content with linear approximations of it, e.g. maps that would only copy the bases. Achieving full copying has been a surprising bonus when working in **SLLM**. As with our previous work, we demonstrate the applicability of the logic to natural language by modelling anaphora and ellipsis and deriving distinct interpretations of strict and sloppy examples.

No vector space semantics is complete without being accompanied by large scale experimentation. So as in previous work, we also implement our model using neural word embeddings Word2vec, FastText, and BERT. In previous work [20] we experimented with a distributional disambiguation task and here we experiment with a distributional similarity task. We build type-driven vector representations for sentences with verb phrase elliptical phrases of the similarity dataset of [27] and compare the results of the linear copying map suggested in this paper with the different non-linear copying maps of previous work, with the non compositional verb-only, compositional grammar unaware additive, and BERT baselines.

$$A ::= A \in At \mid A \cdot A \mid A/A \mid A\backslash A \mid !A \mid \nabla A$$

$$\frac{}{A \longrightarrow A}\ I$$

$$\frac{\Gamma \longrightarrow A \quad \Sigma_1, B, \Sigma_2 \longrightarrow C}{\Sigma_1, \Gamma, A\backslash B, \Sigma_2 \longrightarrow C}\ \backslash_L \qquad \frac{A, \Gamma \longrightarrow B}{\Gamma \longrightarrow A\backslash B}\ \backslash_R$$

$$\frac{\Gamma \longrightarrow A \quad \Sigma_1, B, \Sigma_2 \longrightarrow C}{\Sigma_1, B/A, \Gamma, \Sigma_2 \longrightarrow C}\ /_L \qquad \frac{\Gamma, A \longrightarrow B}{\Gamma \longrightarrow B/A}\ /_R$$

$$\frac{\Gamma_1, A, B, \Gamma_2 \longrightarrow C}{\Gamma_1, A \cdot B, \Gamma_2 \longrightarrow C}\ \cdot_L \qquad \frac{\Gamma_1 \longrightarrow A \quad \Gamma_2 \longrightarrow B}{\Gamma_1, \Gamma_2 \longrightarrow A \cdot B}\ \cdot_R$$

$$\frac{\Gamma_1, \overbrace{A, A, \ldots, A}^{n \text{ times}}, \Gamma_2 \longrightarrow B}{\Gamma_1, !A, \Gamma_2 \longrightarrow B}\ !_L \qquad \frac{A \longrightarrow B}{!A \longrightarrow !B}\ !_R$$

$$\frac{\Gamma_1, A, \Gamma_2 \longrightarrow B}{\Gamma_1, \nabla A, \Gamma_2 \longrightarrow B}\ \nabla_L \qquad \frac{A \longrightarrow B}{\nabla A \longrightarrow \nabla B}\ \nabla_R$$

$$\frac{\Gamma_1, \Gamma_2, \nabla A, \Gamma_3 \longrightarrow B}{\Gamma_1, \nabla A, \Gamma_2, \Gamma_3 \longrightarrow B}\ perm \qquad \frac{\Gamma_1, \nabla A, \Gamma_2, \Gamma_3 \longrightarrow B}{\Gamma_1, \Gamma_2, \nabla A, \Gamma_3 \longrightarrow B}\ perm'$$

Table 1. Formulas and rules of **SLLM**. Where $1 \leq n \leq k_0$.

2 SLLM, Lambek Calculus with Soft Subexponentials

SLLM is a cut-free logic with a decidable fragment and a directed proof system. We recall its definition from [10] in table 1. The decidable fragment is found when one chooses a global bound (k_0) on the number of formulas you may contract using the multiplexing rule (denoted by $!_L$). Note that there are k_0 different instances of the multiplexing rule for a given bottom sequent $\Gamma_1, !A, \Gamma_2 \longrightarrow B$; one for each $1 \leq n \leq k_0$. This determines the number of instances (n) of the formula A that is being contracted (or 'multiplexed') into $!A$. Also notice the separation of multiplexing and permutation connectives and rules. A priori there is no reason to assume that the two properties of contractibility and permutation should coincide as they do in [11].

3 Vector Space Semantics of SLLM

In the following subsections we first inductively define a vector space semantics for **SLLM**, prove its soundness, and then in detail explain how to construct vectors which effectively let us copy using projection maps. The interpretation of the subexponential ! is an adaptation of the tensor algebras of [19], which in turn

originated from [25]. The global multiplexing bound of **SLLM** allows us to use a truncated form of the tensor algebras, we exploit this in the subsection following the proof of proposition 1, and show how it helps us achieve 'full copying' as a linear map.

A great deal of the definition of our semantics uses tensor products and dual spaces, which we recall in the following definitions.

Definition 1. *The **direct sum** (Cartesian Product, direct product) of two vector spaces V, W is denoted $V \oplus W$ and contains vectors of the form (v, w) where $v \in V$ and $w \in W$. It is a quick exercise to check that the interchange map $V \oplus W \to W \oplus V :: (v, w) \mapsto (w, v)$ is a linear isomorphism[1], showing that $V \oplus W \cong W \oplus V$.*

Definition 2. *A **tensor product** of two vector spaces V, W is a vector space $V \otimes W$ such that for any bilinear map $h : V \oplus W \to U$ for some vector space U, there is a unique linear map $\tilde{h} : V \otimes W \to U$ such that the original map h is equal to the composite of $\tilde{h}$ and the canonical map $V \oplus W \to V \otimes W :: (v, w) \mapsto v \otimes w$.*

Note also that since we have $V \oplus W \cong W \oplus V$, one can derive that $V \otimes W \cong W \otimes V$ from definition 2.

Definition 3. *Given a vector space V, its **dual vector space** is the set of linear functionals on V, that is the set $\{f : V \to \mathbb{R} \mid f \text{ linear}\}$, which we denote by V^*. We leave it to the reader to verify that V^* is indeed a vector space.*

The last thing we recall before defining the vector space semantics is that the set of linear maps from a space V to a space W is itself a vector space, and is isomorphic to the space $V^* \otimes W$. With definitions 2 and 3 and the note about spaces of maps under our belt, we may proceed to define vector space models of **SLLM**.

Definition 4. *Vector space models of* **SLLM** *consist of a pair of maps, one mapping formulas of* **SLLM** *to finite dimensional real vector spaces, the other mapping proofs of* **SLLM** *to linear maps. We define a vector space model, denoted $[\![\ ]\!] : \mathbf{SLLM} \to \mathbf{fdVect}$, and define it inductively on formulas and derivable sequents below.*

$$[\![A]\!] := V_A \in \mathbf{fdVect}_{\mathbb{R}}, \text{ for atomic formulas } A,$$

$$[\![A, B]\!] = [\![A \cdot B]\!] := [\![A]\!] \otimes [\![B]\!] \qquad [\![!A]\!] := T_{k_0}[\![A]\!] \qquad [\![\nabla A]\!] := [\![A]\!]$$

$$[\![A \backslash B]\!] := [\![A]\!]^* \otimes [\![B]\!] \qquad [\![B / A]\!] := [\![A]\!]^* \otimes [\![B]\!]$$

where $k_0 \in \mathbb{N}$ is the multiplexing bound and $T_{k_0}[\![A]\!]$ is the k_0-th truncated Fock space of $[\![A]\!]$, defined on a vector space V as:

$$T_{k_0} V := \bigoplus_{i=0}^{k_0} V^{\otimes i} = \mathbb{R} \oplus V \oplus (V \otimes V) \ \oplus \cdots \oplus V^{\otimes k_0}.$$

[1] That is, a linear map which is bijective.

Proofs of sequents $\Gamma \longrightarrow A$ in **SLLM** *are interpreted as linear maps $[\![\Gamma]\!] \longrightarrow [\![A]\!]$.*

Definition 5. *A rule of* **SLLM** *is sound in a vector space model iff whenever the interpretation of the top part is a linear map, so is the interpretation of the bottom part.* **SLLM** *is sound in a vector space model iff all its rules are.*

Proposition 1. SLLM *is sound in $[\![\ ]\!]$ of definition 4.*

Proof. We proceed by a case analysis on the soundness of the rules.

Axiom: The axiom of **SLLM** is interpreted as the existence of identity maps and is immediately sound.

$/_{\mathbf{L}}, \backslash_{\mathbf{L}}$ are interpreted as:

$$\frac{f : [\![\Gamma]\!] \longrightarrow [\![A]\!] \quad g : [\![\Delta_1]\!] \otimes [\![B]\!] \otimes [\![\Delta_2]\!] \longrightarrow [\![C]\!]}{g^f : [\![\Delta_1]\!] \otimes [\![B]\!] \otimes [\![A]\!]^* \otimes [\![\Gamma]\!] \otimes [\![\Delta_2]\!] \longrightarrow [\![C]\!]}$$

and

$$\frac{f : [\![\Gamma]\!] \longrightarrow [\![A]\!] \quad g : [\![\Delta_1]\!] \otimes [\![B]\!] \otimes [\![\Delta_2]\!] \longrightarrow [\![C]\!]}{{}^f g : [\![\Delta_1]\!] \otimes [\![\Gamma]\!] \otimes [\![A]\!]^* \otimes [\![B]\!] \otimes [\![\Delta_2]\!] \longrightarrow [\![C]\!]} \quad ,$$

where g^f and ${}^f g$ are the defined below. All of the maps in the definition are linear, thus making g^f and ${}^f g$ linear too, as required.[2]

$$g^f := g \circ (\mathrm{id}_{[\![\Delta_1]\!]} \otimes \mathrm{ev}_{[\![B]\!] \Leftarrow [\![A]\!]} \otimes \mathrm{id}_{[\![\Delta_2]\!]}) \circ (\mathrm{id}_{[\![\Delta_1]\!]} \otimes \mathrm{id}_{[\![B]\!]} \otimes \mathrm{id}_{[\![A]\!]^*} \otimes f \otimes \mathrm{id}_{[\![\Delta_2]\!]})$$

$${}^f g := g \circ (\mathrm{id}_{[\![\Delta_1]\!]} \otimes \mathrm{ev}_{[\![B]\!] \Rightarrow [\![A]\!]} \otimes \mathrm{id}_{[\![\Delta_2]\!]}) \circ (\mathrm{id}_{[\![\Delta_1]\!]} \otimes f \otimes \mathrm{id}_{[\![A]\!]^*} \otimes \mathrm{id}_{[\![B]\!]} \otimes \mathrm{id}_{[\![\Delta_2]\!]})$$

$/_{\mathbf{R}}, \backslash_{\mathbf{R}}$ are interpreted using the *tensor-hom adjunction*, also referred to as the *left and right currying*. The semantics of $/_R$ is as follows:

$$\frac{f : [\![\Gamma]\!] \otimes [\![A]\!] \longrightarrow [\![B]\!]}{\Lambda^r(f) : [\![\Gamma]\!] \longrightarrow [\![B]\!] \otimes [\![A]\!]^*}$$

where for vectors $\gamma \in [\![\Gamma]\!]$, we have a linear map $\Lambda^r(f)(\gamma) \in [\![B]\!] \otimes [\![A]\!]^*$ given by $\Lambda^r(f)(\gamma)(a) := f(\gamma \otimes a)$. Similarly for the $(\backslash R)$ rule we have:

$$\frac{f : [\![A]\!] \otimes [\![\Gamma]\!] \longrightarrow [\![B]\!]}{\Lambda^l(f) : [\![\Gamma]\!] \longrightarrow [\![A]\!]^* \otimes [\![B]\!]}$$

where $\Lambda^l(f)$ is defined analogously.

[2] The map ev refers to the evaluation map. For example, $\mathrm{ev}_{V \Rightarrow W}(f \otimes w)$ is defined to be equal to $f(w)$ and $\mathrm{ev}_{W \Leftarrow V}(w \otimes g)$ is defined to be equal to $g(w)$ given $f \in V \otimes W^*, g \in W^* \otimes V, w \in W$.

$\cdot_{\mathbf{L}}, \cdot_{\mathbf{R}}$ are immediately sound by the identification of the comma and the $\cdot$ in the semantics. Explicitly:

$$\frac{f : [\![\Gamma_1]\!] \otimes [\![A]\!] \otimes [\![B]\!] \otimes [\![\Gamma_2]\!] \to [\![C]\!]}{f : [\![\Gamma_1]\!] \otimes [\![A]\!] \otimes [\![B]\!] \otimes [\![\Gamma_2]\!] \to [\![C]\!]} \qquad \frac{f : [\![\Gamma_1]\!] \to [\![A]\!] \quad g : [\![\Gamma_2]\!] \to [\![B]\!]}{f \otimes g : [\![\Gamma_1]\!] \otimes [\![\Gamma_2]\!] \to [\![A]\!] \otimes [\![B]\!]} \ .$$

$!_{\mathbf{L}}$ is interpreted as:

$$\frac{f : [\![\Delta_1]\!] \otimes \overbrace{[\![A]\!] \otimes [\![A]\!] \cdots \otimes [\![A]\!]}^{n-\text{times}} \otimes [\![\Delta_2]\!] \longrightarrow [\![B]\!]}{C_n(f) : [\![\Delta_1]\!] \otimes T_{k_0}[\![A]\!] \otimes [\![\Delta_2]\!] \longrightarrow [\![B]\!]}$$

where $C_n(f)$ is a linear map defined using the n-th projection $\pi_n : T_{k_0}([\![A]\!]) \to [\![A]\!]^{\otimes n}$ and as $f \circ (\mathrm{id}_{[\![\Delta_1]\!]} \otimes \pi_n \otimes \mathrm{id}_{[\![\Delta_2]\!]})$.

$!_{\mathbf{R}}$ is interpreted as an application of T_{k_0}. That is,

$$\frac{f : [\![A]\!] \to [\![B]\!]}{T_{k_0} f : T_{k_0}[\![A]\!] \to T_{k_0}[\![B]\!]}$$

where $T_{k_0}(f)$ is a linear map, defined as:

$$T_{k_0}(f)\left(\sum_{i=0}^{k_0} \bigotimes_{j=1}^{n_i} v_j\right) := \sum_{i=0}^{k_0} \bigotimes_{j=1}^{n_i} f(v_j).$$

For an easier example of how $T_{k_0}(f)$ is applied, consider a vector $u+v\otimes w \in T_{k_0}[\![A]\!]$ which is mapped to $f(u) + f(v) \otimes f(w) \in T_{k_0}[\![B]\!]$.

$\nabla_{\mathbf{R}}, \nabla_{\mathbf{L}}$ are interpreted trivially, since ∇ is interpreted as identity on formulas. One sees this explicitly when writing out the interpretations of ∇_L, ∇_R in the semantics. Consider ∇_L, which corresponds to

$$\frac{f : [\![\Gamma_1]\!] \otimes [\![A]\!] \otimes [\![\Gamma_2]\!] \to [\![B]\!]}{f' : [\![\Gamma_1]\!] \otimes [\![A]\!] \otimes [\![\Gamma_2]\!] \to [\![B]\!]} \ .$$

Since we interpret ∇A as $[\![\nabla A]\!] = [\![A]\!]$ there is no distinction remaining between the top and bottom rules, meaning that the a priori different interpretation f' is in fact equal to f. The case for ∇_L is slightly simpler, and we encourage the reader to write it out for themselves.

$\mathbf{perm}, \mathbf{perm}'$ are interpreted using the symmetry of $\otimes$ in **fdVect**. □

Interpreting Projection as Copying. Syntactically speaking, the multiplexing rule does the job of contraction for **SLLM**. This new syntax, however, allows its vector interpretation to greatly differ from the vector space interpretation

of contraction, developed for $!\mathbf{L}^*$ in [19]. This new interpretation provides us with 'full' copies. In the framework of [19] it was only possible to achieve an approximation of (the nonlinear) full copying $v \mapsto v \otimes v$, a problem that is remedied here by exploiting the global multiplexing bound which in turn induced a bound on our interpretation of the !-modality.

Given any n in the range $1 \leq n \leq k_0$, multiplexing $T_{k_0}V \to V^{\otimes n}$ is interpreted as the projection map π_n. In general, this map will not appear to copy any of its inputs and contrarily it seems to 'forget' most of them. This 'forgetting' is exactly what we exploit when editing the shape of the input vectors using the simple $\sim$-construction: given any vector $v \in V$, we define $\tilde{v} \in T_{k_0}V$ as

$$\tilde{v} := \sum_{i=0}^{k_0} v^{\otimes i} = 1 + v + v \otimes v + \cdots + \overbrace{v \otimes \cdots \otimes v}^{k_0 \text{times}}$$

By applying the n-th projection map to $\tilde{v}$ we get $\pi_n(\tilde{v}) = v^{\otimes n}$. To achieve 'full' copying, take $n = 2$ and we obtain $\pi_2(\tilde{v}) = v \otimes v$. So we arrive at an interpretation of contraction which at the surface level looks like copying of $\sim$-ed vectors but which is in fact a simple, even canonical linear map. Clearly we are not making the map $v \mapsto v \otimes v$ linear, nor are we truly copying our input (i.e. we are not achieving $\tilde{v} \mapsto \tilde{v} \otimes \tilde{v}$), instead, we think of the map $\pi_2 : \tilde{v} \mapsto v \otimes v$ as copying v by going through the layers of its bounded Fock space.

4 Categorical Semantics

For readers not familiar with category theory, this section may be skipped entirely. We present the categorical semantics as it is a useful theoretical object for defining further kinds of semantics, and broadens the audience for this work to include applied category theorists. We provide a brief introduction to the category theory used in this paper, but also recommend reading a brief introduction to category theory such as [16], as well as descriptions of categories in linguistic analysis, such as [15]. We will try to introduce the necessary categorical machinery next, before defining our categorical semantics.

Definition 6. *By **monoidal biclosed category** we mean a set of **objects** and a set of **morphisms**. To define the objects, one first fixes a set of atomic symbols $\{A, B, C, \ldots\}$ and add a distinguished symbol I, and three binary operations $\otimes, \Rightarrow, \Leftarrow$. The full set of objects is then defined to be the free $(\otimes, \Rightarrow, \Leftarrow)$-algebra over the given set of atomic symbols. Thus our objects may look like I, A, $A \otimes A$, $A \Rightarrow B$ $B \Leftarrow (A \otimes B)$ and so on.*

*Once we are familiar with the objects, we may define morphisms as a set of arrows pointing from the **domain** to the **codomain** (usually denoted $A \to B$ or $B \to C \otimes D$ etc). We may label morphisms by writing, for example, $f : A \to B$. For every object A, there is a unique identity morphism $\mathrm{id}_A : A \to A$. We may compose pairs of arrows if the codomain of one agrees with the domain of the other. That is:*

$$(f : A \to B, \quad g : B \to C) \longmapsto g \circ f : A \to C.$$

Composing with identity morphisms does nothing. That is, for any $f : A \to B$ we have

$$\mathrm{id}_B \circ f = f = f \circ \mathrm{id}_A.$$

We also define monoidal products of morphisms as:

$$(f : A \to B, \quad g : B \to C) \longmapsto g \otimes f : B \otimes A \to C \otimes B.$$

Both $\circ$ and $\otimes$ are associative, and interact according to the following rule

$$(f \otimes g) \circ (f' \otimes g') = (f \circ f') \otimes (g \circ g'),$$

given that the domains and codomains agree in the necessary way. When it is clear, one often omits the composition symbol, thus $g \circ f$ would be written as gf.

Some typical examples of monoidal biclosed categories are the category of sets and functions, **Set**, where the objects are sets, the monoidal product is the cartesian product, and for any two sets A, B we define $A \Rightarrow B := \{f : A \to B\}$, i.e. $A \Rightarrow B$ is the set of functions from A to B. In **Set** it so happens that $A \Rightarrow B \cong B \Leftarrow A$, although this is not necessarily the case in an arbitrary monoidal biclosed category. An example of a monoidal biclosed category which we make use of is the category of finite dimensional (real) vector spaces **fdVect**, where the objects are finite dimensional vector spaces, and the morphisms are linear maps. In **fdVect** the monoidal product is the tensor product, and the object $V \Rightarrow W$ is the set of linear maps from V to W.

We also recall briefly what a functor is.

Definition 7. *Given two (monoidal biclosed) categories $\mathcal{C}, \mathcal{D}$ a **functor** $F : \mathcal{C} \to \mathcal{D}$ is a pair of functions, one mapping objects of $\mathcal{C}$ to objects of $\mathcal{D}$ the other mapping the morphisms of $\mathcal{C}$ to morphisms of $\mathcal{D}$, such that given a morphism $f : A \to B$ in $\mathcal{C}$, we have $F(f) : F(A) \to F(B)$ in $\mathcal{D}$. We also require composition to be preserved, i.e. $F(g \circ f) = F(g) \circ F(f)$.*

A typical example of a functor is the identity functor $1_{\mathcal{C}} : \mathcal{C} \to \mathcal{C}$, which is defined on any category $\mathcal{C}$ as $1_{\mathcal{C}}(A) := A$ on objects and $1_{\mathcal{C}}(f) := f$ on morphisms.

Definition 8. *Given two functors $F, G : \mathcal{C} \to \mathcal{D}$, A **natural transformation** α from F to G, denoted $\alpha : F \to G$, is a collection of morphisms indexed by objects of $\mathcal{C}$, $(\alpha_A : F(A) \to G(A))_{A \in \mathcal{C}}$ such that for any morphism $f : A \to B$ in $\mathcal{C}$, the following diagram commutes:*

$$\begin{array}{ccc} F(A) & \xrightarrow{F(f)} & F(B) \\ \downarrow{\scriptstyle \alpha_A} & & \downarrow{\scriptstyle \alpha_B} \\ G(A) & \xrightarrow{G(f)} & G(B) \end{array}$$

That is, $G(f) \circ \alpha_A = \alpha_B \ \circ F(f)$.

Having recalled what monoidal biclosed categories and functors and natural transformations are, we may define a categorical semantics for **SLLM** with relative ease.

Definition 9. *Our **categorical semantics of the decidable fragment of** **SLLM**, with global multiplexing bound k_0 is a monoidal biclosed category $\mathcal{C}(\mathbf{SLLM})$, equipped with two functors $M, P : \mathcal{C}(\mathbf{SLLM}) \to \mathcal{C}(\mathbf{SLLM})$, defined below.*

*The objects of $\mathcal{C}(\mathbf{SLLM})$ are the formulas of **SLLM**, and the morphisms $A \to B$ of $\mathcal{C}(\mathbf{SLLM})$ are derivations of the sequent $A \longrightarrow B$ in **SLLM**. The two functors are given names M (for **multiplexing**) and P (for **permutation**), which in turn have the following three structures.*

1. *For each $n = 1, \dots k_0$ and every object $A \in \mathcal{C}(\mathbf{SLLM})$ there is a map $\delta_n : MA \to A^{\otimes n}$.*
2. *There is a counit for P, that is, a natural transformation $\varepsilon : P \to 1$.*
3. *There is a natural isomorphism[3] $\sigma : 1 \otimes P \cong P \otimes 1$.*

*Note that the global multiplexing bound of **SLLM**, k_0, determines the number of morphisms $\delta_n : MA \to A^{\otimes n}$ in our multiplexing functor.*

*For every atomic formula A of **SLLM** we prescribe an object $(\!|A|\!) := C_A$. For the complex formulas of **SLLM** we interpret them as follows:*

$$(\!|A, B|\!) = (\!|A \cdot B|\!) := (\!|A|\!) \otimes (\!|B|\!) \qquad (\!|!A|\!) := M(\!|A|\!) \qquad (\!|\nabla A|\!) := P(\!|A|\!)$$

$$(\!|A \backslash B|\!) := (\!|A|\!) \Rightarrow (\!|B|\!) \qquad (\!|B / A|\!) := (\!|B|\!) \Leftarrow (\!|A|\!)$$

We can now prove a proposition similar to our proposition 1, that the rules of **SLLM** are sound in $\mathcal{C}(\mathbf{SLLM})$. We sketch the proof for the more complicated rules and leave the rest for the attentive reader to fill in.

Proof (Sketch). **axiom** is interpreted as the identity morphism for objects of $\mathcal{C}(\mathbf{SLLM})$. $\backslash_{\mathbf{L}}$ is interpreted as:

$$\frac{f : (\!|\Gamma|\!) \to (\!|A|\!) \quad g : (\!|\Delta_1|\!) \otimes (\!|B|\!) \otimes (\!|\Delta_2|\!) \to (\!|C|\!)}{{}^f g : (\!|\Delta_1|\!) \otimes (\!|\Gamma|\!) \otimes (\!|A|\!) \Rightarrow (\!|B|\!) \otimes (\!|\Delta_2|\!) \to (\!|C|\!)}$$

where ${}^f g := g \circ (\mathrm{id}_{(\!|\Delta_1|\!)} \otimes \mathrm{ev}_{(\!|A|\!) \Rightarrow (\!|B|\!)} \otimes \mathrm{id}_{(\!|\Delta_2|\!)}) \circ (\mathrm{id}_{(\!|\Delta_1|\!)} \otimes f \otimes \mathrm{id}_{(\!|\Delta_2|\!)})$, much like the definition in the vector space semantics. $\backslash_{\mathbf{R}}$ is interpreted immediately, via the same tensor-hom adjunction used for vector space semantics. In the category theory, this comes naturally from the requirement that $\mathcal{C}(\mathbf{SLLM})$ be monoidal closed. $!_{\mathbf{L}}$ is interpreted in k_0 different instances, one for each $n = 1, \dots, k_0$. Explicitly, given n, this is:

$$\frac{f : (\!|\Gamma_1|\!) \otimes (\!|A|\!)^{\otimes n} \otimes (\!|\Gamma_2|\!) \to (\!|B|\!)}{C_n(f) : f : (\!|\Gamma_1|\!) \otimes M(\!|A|\!) \otimes (\!|\Gamma_2|\!) \to (\!|B|\!)}$$

[3] A natural isomorphism is a natural transformation whose components are all isomorphisms.

where $C_n(f)$ is defined to be $f \circ (\mathrm{id}_{(\!|\Gamma_1|\!)} \otimes \delta_n \otimes \mathrm{id}_{(\!|\Gamma_2|\!)})$. $!_{\mathbf{R}}$: is interpreted immediately as the application of the functor M. $\nabla_{\mathbf{L}}$ is interpreted using the counit $\varepsilon : P \to 1$ as follows:

$$\frac{f : (\!|\Gamma_1|\!) \otimes (\!|A|\!) \otimes (\!|\Gamma_2|\!) \to (\!|B|\!)}{Q(f) : (\!|\Gamma_1|\!) \otimes P(\!|A|\!) \otimes (\!|\Gamma_2|\!) \to (\!|B|\!)}$$

where $Q(f)$ is defined to be $f \circ (\mathrm{id}_{(\!|\Gamma_1|\!)} \otimes \varepsilon \otimes \mathrm{id}_{(\!|\Gamma_2|\!)})$. $\nabla_{\mathbf{R}}$: is interpreted just like $!_R$ above, but as an application of P. $\mathbf{perm}, \mathbf{perm}'$ are interpreted using the natural isomorphism $\sigma : \mathrm{id} \otimes P \cong P \otimes \mathrm{id}$. □

Note that since we have fixed a bound k_0 in the syntax of the modality ! see item 1 of definition 9, the categorical semantics defined above is not only sound, but also a complete model of **SLLM**. This is an automatic construction akin to that of syntactic categories [9] or free categories over a monoidal signature as in [26] and it lets you define semantics of **SLLM** in other categories $\mathcal{D}$ as structure preserving functors $\mathcal{C}(\mathbf{SLLM}) \to \mathcal{D}$, rather than tediously specifying a semantics in $\mathcal{D}$ directly from **SLLM**. In fact, the vector space semantics $[\![\ ]\!] : \mathbf{SLLM} \to \mathbf{fdVect}$ of section 3 factors through $(\!|\ |\!) : \mathbf{SLLM} \to \mathcal{C}(\mathbf{SLLM})$.

5 ! and ∇ in Natural Language Syntax and Semantics

We demonstrate how to analyse the same phenomena as in previous work [19] and [20], namely parasitic gaps, anaphora, (verb phrase) ellipsis, and a combination of the latter two. Again, **SLLM** distinguishes between strict and sloppy readings in the ambiguous anaphora with ellipsis cases and this distinction is also carried into the semantics.

5.1 Parasitic Gaps

The noun phrase *Papers that John signed without reading*, with **SLLM** types[4]:

$$\{(\text{Papers} : n), (\text{that} : (n\backslash n)/(s/!\nabla n)), (\text{John} : n), (\text{signed} : n\backslash s/n),\\ (\text{without}, ((n\backslash s)\backslash(n\backslash s))/n), (\text{reading}, n/n)\}$$

[4] The choice of n/n for the type of "reading" is from original work of [11].

acquired the following **SLLM** derivation:

$$\dfrac{\dfrac{\dfrac{}{n\longrightarrow n}\quad\dfrac{}{n\longrightarrow n}}{n,n\backslash n\longrightarrow n}\backslash_L\qquad\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{n\longrightarrow n}\quad\dfrac{}{s\longrightarrow s}}{n,n\backslash s\longrightarrow s}\backslash_L}{n\backslash s\longrightarrow n\backslash s}\backslash_L\quad\dfrac{}{n\longrightarrow n}}{n,n\backslash s,(n\backslash s)\backslash(n\backslash s)\longrightarrow s}/_L\quad\dfrac{}{n\longrightarrow n}}{n,n\backslash s/n,n,(n\backslash s)\backslash(n\backslash s)\longrightarrow s}/_L\quad\dfrac{}{n\longrightarrow n}}{n,n\backslash s/n,n,((n\backslash s)\backslash(n\backslash s))/n,n\longrightarrow s}/_L}{n,n\backslash s/n,n,((n\backslash s)\backslash(n\backslash s))/n,n/n,n\longrightarrow s}\nabla_L}{n,n\backslash s/n,n,((n\backslash s)\backslash(n\backslash s))/n,n/n,\nabla n\longrightarrow s}\nabla_L}{n,n\backslash s/n,\nabla n,((n\backslash s)\backslash(n\backslash s))/n,n/n,\nabla n\longrightarrow s}perm'}{n,n\backslash s/n,((n\backslash s)\backslash(n\backslash s))/n,\nabla n,n/n,\nabla n\longrightarrow s}perm'}{n,n\backslash s/n,((n\backslash s)\backslash(n\backslash s))/n,n/n,\nabla n,\nabla n\longrightarrow s}!_L}{n,n\backslash s/n,((n\backslash s)\backslash(n\backslash s))/n,n/n,!\nabla n\longrightarrow s}/_R}{n,n\backslash s/n,((n\backslash s)\backslash(n\backslash s))/n,n/n\longrightarrow s/!\nabla n}\backslash_L$$

$$n,(n\backslash n)/(s/!\nabla n),n,n\backslash s/n,((n\backslash s)\backslash(n\backslash s))/n,n/n\longrightarrow n$$

The linear map representing this derivation is:

$$N\otimes(N^*\otimes N)\otimes(S\otimes(T_{k_0}N)^*)^*\otimes N\otimes N^*\otimes S\otimes N^*\otimes(N^*\otimes S)^*\otimes(N^*\otimes S)\otimes N^*\otimes N\otimes N^*\to N$$

$$\begin{aligned}&::\overrightarrow{Papers}\otimes\overline{that}\otimes\overrightarrow{John}\otimes\overline{signed}\otimes\overline{without}\otimes\overline{reading}\\&\mapsto\overline{that}(\overrightarrow{Papers},\overrightarrow{without}(\overrightarrow{John},\overline{signed}(-,-),\overline{reading}(-)))\end{aligned}$$

simplified to the following by interpreting *that* as evaluation:

$$\overrightarrow{without}(\overrightarrow{John},\overline{signed}(-,\overrightarrow{Papers}),\overline{reading}(\overrightarrow{Papers}))).$$

5.2 Pronominal Anaphora

Following our motivating example in [20], we consider the utterance *John sleeps. He snores*, with the following **SLLM** types:

$$\{(John:!(\nabla n)),(sleeps:n\backslash s),(He:\ \nabla n\backslash n),(snores:n\backslash s)\}.$$

Note the type of *John* in this example is $!(\nabla n)$, whereas earlier it was simply n. In fact, one can always retrieve the type n from $!(\nabla n)$ using the following type raising operation:

$$\dfrac{\dfrac{\Gamma_1,n,\Gamma_2\longrightarrow X}{\Gamma_1,\nabla n,\Gamma_2\longrightarrow X}\nabla_L}{\Gamma_1,!(\nabla n),\Gamma_2\longrightarrow X}!_L$$

The derivation corresponding to the syntax of this utterance in **SLLM** is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{n\longrightarrow n}\quad\dfrac{\dfrac{}{\nabla n\longrightarrow\ \nabla n}\quad\dfrac{\dfrac{}{n\longrightarrow n}\quad\dfrac{}{s,s\longrightarrow s,s}}{s,n,n\backslash s\longrightarrow s,s}\backslash_L}{s,\ \nabla n,\ \nabla n\backslash n,n\backslash s\longrightarrow s,s}\backslash_L}{n,n\backslash s,\ \nabla n,\ \nabla n\backslash n,n\backslash s\longrightarrow s,s}\backslash_L}{\nabla n,n\backslash s,\ \nabla n,\ \nabla n\backslash n,n\backslash s\longrightarrow s,s}\nabla_L}{\nabla n,\nabla n,n\backslash s,\ \nabla n\backslash n,n\backslash s\longrightarrow s,s}perm}{!(\nabla n),n\backslash s,\ \nabla n\backslash n,n\backslash s\longrightarrow s,s}contr_2$$

This derivation translates into the following linear map:

$$f : T_{k_0}N \otimes N^* \otimes S \otimes N^* \otimes N \otimes N^* \otimes S \longrightarrow S \otimes S$$
$$:: \tilde{n} \otimes \nu \otimes s \otimes \nu' \otimes n' \otimes \nu'' \otimes s' \longmapsto \nu(n)s \otimes \nu'(n)\nu''(n')s$$

Taking $\widetilde{John} \in T_{k_0}N$, $\overrightarrow{sleeps} \in N^* \otimes S$, $\overrightarrow{He} \in N^* \otimes N$ and $\overrightarrow{snores} \in N^* \otimes S$ we obtain the simplified version below:

$$f\left(\widetilde{John} \otimes \overrightarrow{sleeps} \otimes \overrightarrow{He} \otimes \overrightarrow{snores}\right) = \overrightarrow{sleeps}(\overrightarrow{John}) \otimes \overrightarrow{snores}(\overrightarrow{He}(\overrightarrow{John})).$$

5.3 Ellipsis

Consider the utterance *John plays guitar. Lisa does too.* with **SLLM** types:

$$\{(John : n), (plays\ guitar :!(\nabla(n\backslash s))), (Lisa : n), (does\text{-}too :\ \nabla(n\backslash s)\backslash n\backslash s)\}$$

The derivation of this utterance in **SLLM** is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\nabla(n\backslash s) \longrightarrow \nabla(n\backslash s)} \quad \dfrac{\dfrac{}{n \longrightarrow n} \quad \dfrac{}{s, s \longrightarrow s, s}\backslash_L}{s, n, n\backslash s \longrightarrow s, s}\backslash_L}{\dfrac{}{n \longrightarrow n} \quad s, n,\ \nabla(n\backslash s),\ \nabla(n\backslash s)\backslash n\backslash s \longrightarrow s, s}\backslash_L}{n, n\backslash s, n,\ \nabla(n\backslash s),\ \nabla(n\backslash s)\backslash n\backslash s \longrightarrow s, s}\backslash_L}{n,\ \nabla(n\backslash s), n,\ \nabla(n\backslash s),\ \nabla(n\backslash s)\backslash n\backslash s \longrightarrow s, s}\nabla_L}{n,\ \nabla(n\backslash s),\ \nabla(n\backslash s), n,\ \nabla(n\backslash s)\backslash n\backslash s \longrightarrow s, s}\ perm}{n, !(\nabla(n\backslash s)), n,\ \nabla(n\backslash s)\backslash n\backslash s \longrightarrow s, s}\ contr_2$$

which gives us the following linear map

$$f : N \otimes T_{k_0}(N^* \otimes S) \otimes N \otimes (N \otimes S)^* \otimes N^* \otimes S \longrightarrow S \otimes S$$

and again with the following notational convenience

$$\overrightarrow{John}, \overrightarrow{Lisa} \in N \qquad \widetilde{plays\ guitar} \in T_{k_0}(N^* \otimes S)$$
$$\overrightarrow{does\text{-}too} \in (N \otimes S)^* \otimes N^* \otimes S$$

we can see the action of f simplified to:

$$f(\overrightarrow{John} \otimes \widetilde{plays\ guitar} \otimes \overrightarrow{Lisa} \otimes \overrightarrow{does\text{-}too})$$
$$:= \overrightarrow{plays\ guitar}(\overrightarrow{John}) \otimes \overrightarrow{does\text{-}too}(\overrightarrow{plays\ guitar} \otimes \overrightarrow{Lisa})$$

5.4 Anaphora with Ellipsis

Consider the utterance *Kim likes their code. Sam does too*, with **SLLM** types:

$$\{(Kim :!(\nabla n)), (Sam :!(\nabla n)), (likes :!(\nabla(!(\nabla(n\backslash s))/n))), (their :!(\nabla(\nabla n\backslash n))/n),$$
$$(code : n), (does\ too :\ \nabla(n\backslash s)\backslash(n\backslash s))\}.$$

This utterance is ambiguous and its ambiguity comes from the fact that one can read it in a *sloppy* way as: *Kim likes Kim's code. Sam likes* ***Sam's*** *code*, or a *strict* way and as *Kim likes Kim's code. Sam likes* ***Kim's*** *code*. In [20] we showed how to distinguish between these two readings syntactically and semantically in the framework of [19]. One can do the same in **SLLM** by means of their two distinct derivations, provided in figures 1 and 2 of the Appendix.

6 Experimental Evaluation

To experiment with ellipsis, we use an existing elliptical sentence similarity dataset[5], referred to as ELLSIM and originally developed in [27]; This dataset extends the transitive sentence similarity dataset[6], referred to as KS2013 and developed in [12] from transitive sentences of the form *Subject Verb Object* to sentences with verb phrase ellipsis of the form *Subject Verb Object and Subject* does too*. We briefly review these datasets below.

6.1 Descriptions of the Datasets

The KS2013 dataset consists of 108 transitive sentence pairs of the form of *Subject Verb Object*, where the *Verb Object* pairs come from the dataset ML2010 of [8]. KS2013 extends these *Verb Object* pairs by adding subjects to them, according to the following procedure. Each pair in ML2010 consists of phrases with verbs and objects that are synonymous. To the first phrase of each pair, KS2013 adds a subject selected from the 5 most frequent subjects of the verb of the phrase. To the second phrase of the pair, it adds a subject that is synonymous to the –just added– subject of the first sentence. As a result of this procedure, KS2013 obtains pairs of sentences that have similar subjects, verbs, and objects to each other. These sentences are rendered as 'highly similar' to each other. As an example, consider the following pair from the ML2010 dataset:

$$\langle \textit{produce effect, achieve result} \rangle$$

This becomes as follows in KS2013:

$$\langle \textit{drug produce effect, medication achieve result} \rangle$$

Another example is the following pair of ML2010:

$$\langle \textit{pose problem, address question} \rangle$$

which became as follows in KS2013:

$$\langle \textit{study pose problem, paper address question} \rangle$$

ML2010 and KS2013 datasets are divided into three bands of pairs: pairs of sentences that have high, low or medium degrees of similarity to each other. The procedure described above, returns sentences pairs that are in the high band. The pairs of the medium band are obtained by pairing the sentences of the high band with sentences that only have one synonymous words with them. Similarly, the pairs in the low band are obtained by pairing the sentences in the high band

[5] https://github.com/gijswijnholds/compdisteval-ellipsis/blob/master/datasets/ELLSIM.txt

[6] http://www.cs.ox.ac.uk/activities/compdistmeaning/emnlp2013_turk.txt

with sentences that only have one or no synonymous words to them. Examples of pairs of the medium and low bands are provide below:

⟨ *drug produce effect, medication address problem*⟩ : MEDIUM
⟨ *drug produce effect, medication pose result*⟩ : MEDIUM
⟨ *drug produce effect, medication pose problem*⟩ : LOW
⟨ *drug produce effect, employee start work*⟩ : LOW

Similar to ML2010, the KS2013 dataset is uploaded to AmazonTurk and annotations are obtained for it by multiple participants in a scale from 1 to 7;1 for low similarity, 7 for highly similar. For example, ⟨*drug produce effect*⟩ and ⟨*employee start work*⟩ are judged by most annotators to have a low degree of similarity, while ⟨*drug produce effect*⟩ and ⟨*medication achieve result*⟩ are considered highly similar. A value in the median of this range represents a medium degree of similarity.

The ELLSIM dataset is built from KS2013 by choosing two new subjects, let's call denote them by *Subject**, for each sentence. *Subject** is chosen from a list of most frequent subjects of the verb of the sentence, extracted from ukWaC and Wackypedia corpora. The new subjects are appended to a "does too" ellipsis marker, thus forming the elliptical phrase "*Subject** does too". Joining this phrase to the original sentence with a conjunctive word such as "and" turns that sentence into the sentence *Subject Verb Object and Subject* does too*, which is a sentence with an elliptical phrase. Here is an example: consider the KS2013 sentence:

drug produce effect

Two new subjects *plant* and *combination* are chosen for this sentence. These new subjects are used to form the following two elliptical phrases:

plant does too
combination does too

which are then conjoined with the first sentence with an "and", resulting in the following sentences with elliptical phrases:

drug produce effect and plant does too
drug produce effect and combination does too

Another example is the following KS2013 sentence:

medication achieve result

which is turned into the following two sentences with elliptical phrases:

medication achieve result, and patient does too
medication achieve result and programme does too

At the end of this procedure all of the sentences of KS2013 are turned into sentences with elliptical phrases. Each of these newly generated sentences are paired with the same sentences as they were before, except that the sentences they were paired with before are now themselves sentences with elliptical phrases in them. A snapshot of the entries of the dataset is presented in Table 2. The final dataset contains 432 entries and new annotations are obtained for it following the same procedure as the one that was followed for KS2013, described above. 9800 annotations by 42 different participants are obtained for ELLSIM.

Sentence 1	**Sentence 2**
drug produce effect and combination does too	medication achieve result and patient does too
drug produce effect and plant does too	medication achieve result and patient does too
drug produce effect and combination does too	medication achieve result and programme does too
drug produce effect and plant does too	medication achieve result and programme does too

Table 2. An example of a pair of sentences from ELLSIM dataset.

6.2 Building the Representations

We build vectors for each sentence of ELLSIM and calculate the cosine similarities between pairs of sentences therein. For building these vectors, we first use 300 dimensional pre-trained **Word2vec** and **FastText** word embeddings for the $\overrightarrow{Subject}, \overrightarrow{Subject^*}$, and $\overrightarrow{Object}$ vectors of the dataset. Then, and for the verbs of ELLSIM, we form matrices $\overline{Verb}$ obtained according to a formula developed in [5], called the *Relational* method. This formula is given below:

$$\overline{Verb} := \sum_i \overrightarrow{s}_i \otimes \overrightarrow{o_i}$$

It takes the Kronecker products of all the subjects $\overrightarrow{s}_i$ and objects $\overrightarrow{o}_i$ that the verb related to each other in the sentences of the corpus. The $\overrightarrow{s}_i$ and $\overrightarrow{o}_i$ vectors of this formulae are built in the same way as the vectors for subjects and objects of the sentences of the dataset, described above.

Leaving the elliptical phrases aside for the moment, in the next step, we describe how we build vector representations for the *Subject Verb Object* parts of the sentences of the dataset. In this step, the relational verb matrix of the verb of each sentence is *composed* with the *Subject* and *Object* vectors of the subject and object of the sentence using the methods developed in [13], referred to as *Copy Obj* and *Copy Subj*. We also use the *Frob Add* and *Frob Mult* methods developed in [21]. These compositions are obtained according to the following formulae:

$$\textbf{Copy Obj: } (\overline{Verb} \times \overrightarrow{Object}) \odot \overrightarrow{Subject}$$
$$\textbf{Copy Subj: } (\overline{Verb} \times \overrightarrow{Subject}) \odot \overrightarrow{Object}$$
$$\textbf{Frob Add: } \quad \textbf{Copy Obj} + \textbf{Copy Sub}$$
$$\textbf{Frob Mult: } \quad \textbf{Copy Obj} \odot \textbf{Copy Sub}$$

The ellipsis is taken into account in the final step, where, we use the methodology described in the paper to resolve the ellipsis. Conceptually, this is obtained by treating the *does too* phrase as a copying map and applying it to the result of each of the compositional methods described above. In effect, this procedure will produce two copies of the verb phrase *Verb Object* of the first part of the sentence and then applies each one of them to the vector of each of the subjects: *Subject* and *Subject**. At the end, we obtain a vector representation for the whole sentence with elliptical phrase, that is for *Subject Verb Object and Subject* does too*. Each compositional method will indeed result in a different vector representation. For the *Copy Obj* method we have the following representation:

$$\textbf{Multiplex} : ((\overline{Verb} \times \overrightarrow{Object}) \odot \overrightarrow{Subject}) + ((\overline{Verb} \times \overrightarrow{Object}) \odot \overrightarrow{Subject^*})$$

The formula for the *Copy Subj* method is obtained from the above by exchanging $\overrightarrow{Object}$ and $\overrightarrow{Subject}$; *Frob Add* is the sum of *Copy Obj* and *Copy Subj*, *Frob Mult* is their product.

After a representation is built for each sentence of the dataset, we measure the degree of similarity between sentences of each pair. Traditionally, this is done by computing the cosine distance between the sentence vectors. The distances are then put against the degrees of similarity obtained from human annotations for each pair. In order to form a judgement about whether the representations were good or bad, we perform three statistical tests, described in the following three subsections.

For each of these tests, we compare different vector embeddings and compositional methods with each other and with a few baselines. We describe our baselines below. Firstly, we consider a non-compositional baseline that takes the representation of each sentence to be the same as the representation of its verb, thus ignoring all the other constituents of the sentence. For this, we use the *Verb* only vector and *Verb* only tensor representations of the *Verb*. For the *Verb* only tensor, we use the *Relational* matrix of the verb as described above. Another of our baselines is the Additive model; this model is compositional but does not take the grammatical structure into account. It is obtained by adding the vectors of *Subject*, *Verb*, *Object*, and *Subject** of the sentence. Our next baseline is the BERT base model; for this we used pre-trained embeddings of dimension 768 extracted from the second-to-last hidden layer of all tokens in the sentence with average pooling.

As an example, consider the pair of sentences:

S1: *drug produce effect and combination does too*
S2: *medication achieve result and patient does too*

Method	Embeddings	Results
Copy Subject	word2vec	0.644
	fasttext	0.591
Copy Object	word2vec	0.604
	fasttext	0.599
Frobenius Add.	word2vec	0.653
	fasttext	0.610
Frobenius Mult.	word2vec	0.587
	fasttext	0.579
Baselines		
Verb Only Vector	word2vec	0.583
	fasttext	0.651
Verb Only Tensor	word2vec	0.566
	fasttext	0.533
Additive	word2vec	0.768
	fasttext	0.783
BERT phrase		0.575

Table 3. Experiment 1: Spearman ρ scores

Following our methods, we obtain the following vectors for them:

- Our copying map applied to the *Copy Obj*:
  $\overrightarrow{S1} = ((\overline{produce} \times \overrightarrow{effect}) \odot \overrightarrow{drug}) + ((\overline{produce} \times \overrightarrow{effect}) \odot \overrightarrow{combination})$
  $\overrightarrow{S2} = ((\overline{achieve} \times \overrightarrow{result}) \odot \overrightarrow{medication}) + ((\overline{achieve} \times \overrightarrow{result}) \odot \overrightarrow{patient})$

  Our copying map applied to the *Copy Subj*:
  $\overrightarrow{S1} = ((\overline{produce} \times \overrightarrow{drug}) \odot \overrightarrow{effect}) + ((\overline{produce} \times \overrightarrow{combination}) \odot \overrightarrow{effect})$
  $\overrightarrow{S2} = ((\overline{achieve} \times \overrightarrow{medication}) \odot \overrightarrow{result}) + ((\overline{achieve} \times \overrightarrow{patient}) \odot \overrightarrow{result})$
- The Additive baseline:
  $\overrightarrow{S1} = \overrightarrow{drug} + \overrightarrow{produce} + \overrightarrow{effect} + \overrightarrow{combination}$
  $\overrightarrow{S2} = \overrightarrow{medication} + \overrightarrow{achieve} + \overrightarrow{result} + \overrightarrow{patient}$

6.3 Spearman Experiment

Following [27], we calculate Spearman's rank correlation coefficient between the cosine similarity scores of pairs of sentences of ELLSIM and the average human annotation judgments. This is a value between -1 and +1. Results are presented in Table 3. Our first observation is that the best performing model was the Additive baseline. It achieved the highest Spearman correlation score of 0.783 in the **FastText** space. Between all our compositional methods, the highest correlation score was 0.653 with the Frobenius Additive and in the **Word2vec** space. All of the compositional models outperformed the BERT phrasal model but not the Additive baseline model. In both cases, the results are

Method	Embeddings	Results
Copy Subject	word2vec	15.99
	fasttext	15.09
Copy Object	word2vec	14.82
	fasttext	14.40
Frobenius Add.	word2vec	14.86
	fasttext	14.24
Frobenius Mult.	word2vec	15.97
	fasttext	14.85
Baselines		
Verb Only Vector	word2vec	15.05
	fasttext	15.13
Verb Only Tensor	word2vec	15.01
	fasttext	14.56
Additive	word2vec	14.32
	fasttext	14.32
BERT phrase		13.76

Table 4. Experiment 2: Student's t-test results

impressive especially since BERT is considered to be the state-of-the-art context-based neural model, trained on a huge dataset, and consisting of millions of parameters. BERT created a breakthrough in the field of NLP by providing greater results in many NLP tasks, such as question answering, text generation, sentence classification, and many more besides. Getting better results than BERT in this sentence similarity task is considered a good achievement, but of course, we fell short of a very simple additive model.

6.4 Student's *t*-test Experiment

Following [24] we examine the difference between two group means: the average human annotation judgments and the cosine similarity scores via student's t-test. First, we match each sentence pair in the dataset with its corresponding group of sentences. Then, we calculate the t-score for each group and compute the overall average. The smaller the t-score, the more similarity exists between the two sets. Results are presented in Table 4. Here, BERT did the best with the lowest t-score, which was 13.76. Then, we had our Frobenius Add. model with a t-score of 14.24. In the third place, came the Additive baseline with a t-score of 14.32. Our best results were achieved best while using **FastText** embeddings.

6.5 Classification Experiment

Following [19, 20], we also perform a classification experiment. We compare the average human annotation judgments and the cosine similarity scores between triplets of

Method	Embeddings	Results
Copy Subject	word2vec	71.18%
	fasttext	73.44%
Copy Object	word2vec	76.33%
	fasttext	74.19%
Frobenius Add.	word2vec	76.04%
	fasttext	73.67%
Frobenius Mult.	word2vec	69.79%
	fasttext	71.24%
Baselines		
Verb Only Vector	word2vec	76.44%
	fasttext	78.53%
Verb Only Tensor	word2vec	73.30%
	fasttext	73.30%
Additive	word2vec	82.00%
	fasttext	81.77%
BERT phrase		75.93%

Table 5. Experiment 3: Classification Accuracy results

sentences. An example of these triplets is shown below. A new dataset[7] was generated from the original dataset and used in this experiment.

Sentence 1: *drug produce effect and combination does too*
Sentence 2: *employee start work and team does too*
Sentence 3: *committee consider matter and panel does too*

A correct classification is obtained when both the average human annotation score and cosine similarity between one pair of the triplet is higher than the other pair. In other words, the average human annotation score and cosine similarity agree. An overall accuracy is computed by counting the number of correct classifications out of these comparisons. For instance, the cosine similarity measure between Sentences 1 and 2 above using the Additive model and **Word2vec** embeddings is 0.104, while that between Sentences 1 and 3 above is 0.195. On the other hand, the average human annotation score between Sentences 1 and 2 is 2.217 while that between Sentences 1 and 3 is 2.695. As a result, the cosine similarity score between Sentences 1 and 3 is higher than that between 1 and 2 and similarly the human annotation scores. This match is counted as a correct classification.

Table 5 shows that similar to the Spearman's test, the Additive model achieved the highest accuracy of 82.00%, even higher than BERT, whose accuracy was 75.93%. Our best model was *Copy Obj* with an accuracy of 76.33%. This was in fact the second best accuracy after the Additive model with values even higher than BERT's.

[7] https://cutt.ly/QmacrwT

7 Conclusion and Outlook

We develop a categorical and a vector space semantics for a decidable version of an extension of Lambek Calculus used to model parasitic gaps and coreference. This enables us to rely on a decidable logic to produce the type-logical derivations of these phenomena and develop compositional type-driven vector representations for them. We also achieved what previously was not possible: working with full copies of types, rather than their non-linear approximations. We perform three statistical tests using data from large scale corpora and a distributional verb phrase elliptical similarity task. In two out of the three experiments, our model outperformed the state-of-the-art BERT. We think this is impressive since BERT has perform very well on downstream NLP tasks, which require complex modelling of structural relationships [18]. Our models, however, were outperformed by a very simple additive model. The difference between our best model and the Additive model was were however, very small and in the range 0.1 or around 10%.

What can we learn from these results? That we beat BERT in one test and came very close to it in another, is quite impressive; that in two tests, we were beaten by a very simple additive model that ignores all structure, is disappointing, however, we did come very close to this model. We can for sure say that it is worth taking the grammatical and discourse structures into account when building vector representations for sentences. The resulting representations might not perform better than other simple or state of the art models, but will nonetheless come very close to it. So if our aim is to represent structures of sentences as well as their distributional properties, our model is the one to go for. One might not obtain the best results, but the loss is very little. In our experiments, the differences between our best model and the BERT or Additive models were only about 0.1 in t-test and Spearman and 10% in accuracy. A qualitative analysis only made sense for our classification test, the other tests were made over the patterns of the whole dataset. Going through the instances of the classification test, we observed that in pairs of sentences such as $\langle$ *medication achieve result and patient does too, drug produce effect and combination does too*$\rangle$ where the disambiguation is more clear, the additive model does better. In this example, the sentence pair belonged to the HIGH similarity band, the cosine of the angle between its sentences in the additive model was 0.74, whereas the cosine of the angle in our model was 0.72. The human annotations where also on the high side with an average score of 4.26 (recall that these are between 1 and 7). In examples where the disambiguation was less clear, e.g. in the pair $\langle$*user send message and server does too, man hear word and listener does too*$\rangle$, from the MEDIUM similarity band, our model worked better by producing a cosine of 0.33 as opposed to the additive model which resulted in a cosine of 0.13. The average human score in this case was 3.33.

A few directions to pursue for future work are as follows. Firstly, the theoretical challenge of finding complete vector space models remains an open challenge. Investigating completeness possibly through centres of monoidal functors over finite dimensional vector spaces with a Hopf structure is our work in progress. Then, experimenting with the setting on the Winograd Schema Challenge [17] using the plausibility models of [24] is work in progress. Still on the experimental side, we have compared the current results to its approximations, as developed in previous work [19]. The basis-copying operations used before, the so called Frobenius copying maps, did not perform well. Their additive combination using a method we called k-extension, however, produced comparable results. Here we observed an anomaly that we could not resolve in this paper and which we leave to further experimentation.

Bibliography

[1] G. Barry, M. Hepple, N. Leslie, and G. Morrill. Proof figures and structural operators for categorial grammar. In *Fifth Conference of the European Chapter of the Association for Computational Linguistics*, 1991.

[2] R. F. Blute, J. R. B. Cocket, and R. A. G. Seely. Differential categories. *Mathematical Structures in Computer Science*, 16(6):1049–1083, 2006.

[3] J.R Firth. A synopsis of linguistic theory, 1930-1955, 1957.

[4] J.-Y. Girard. Light Linear Logic. *Information & Computation*, 143:175–204, 1998.

[5] E. Grefenstette and M. Sadrzadeh. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1394–1404, 2011.

[6] G. Jäger. A multi-modal analysis of anaphora and ellipsis. *University of Pennsylvania Working Papers in Linguistics*, 5(2):2, 1998.

[7] G. Jäger. *Anaphora and type logical grammar*, volume 24. Springer Science & Business Media, 2006.

[8] Mirella Lapata Jeff Mitchell. Composition in Distributional Models of Semantics. *Cognitive Science*, 34(8):1388–1429, 2010.

[9] P. T. Johnstone. *Sketches of an elephant: a Topos theory compendium.* Oxford logic guides. Oxford Univ. Press, New York, NY, 2002.

[10] M. Kanovich, S. Kuznetsov, V. Nigam, and A. Scedrov. Soft Subexponentials and Multiplexing. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2020.

[11] M. Kanovich, S. Kuznetsov, and A. Scedrov. Undecidability of the Lambek calculus with a relevant modality. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9804 LNCS:240–256, 2016.

[12] D. Kartsaklis and M. Sadrzadeh. Prior disambiguation of word tensors for constructing sentence vectors. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1590–1601, 2013.

[13] Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Stephen Pulman. A unified sentence space for categorical distributional-compositional semantics: Theory and experiments. In *Proceedings of COLING 2012: Posters*, pages 549–558, Mumbai, India, December 2012. The COLING 2012 Organizing Committee.

[14] Y. Lafont. Soft linear logic and polynomial time. *Theoretical Computer Science*, 2004.

[15] J. Lambek. Categorial and categorical grammars. In Richard T. Oehrle, Emmon Bach, and Deirdre Wheeler, editors, *Categorial Grammars and Natural Language Structures*, pages 297–317. Springer Netherlands, Dordrecht, 1988.

[16] Tom Leinster. Basic category theory, 2016.

[17] Hector J. Levesque, Ernest Davis, and Leora Morgenstern. The winograd schema challenge. In *Proceedings of the International Workshop on Temporal Representation and Reasoning*, 2012.

[18] Yongjie Lin, Yi Chern Tan, and Robert Frank. Open sesame: Getting inside bert's linguistic knowledge, 2019.

[19] L. McPheat, M. Sadrzadeh, H. Wazni, and G. Wijnholds. Categorical vector space semantics for lambek calculus with a relevant modality,https://arxiv.org/abs/2005.03074, 2020.

[20] L. McPheat, G. Wijnholds, M. Sadrzadeh, A. Correia, and A. Toumi. Anaphora and ellipsis in lambek calculus with a relevant modality: Syntax and semantics. *Journal of Cognitive Science*, 2021. to appear.
[21] D. Milajevs, D. Kartsaklis, M. Sadrzadeh, and M. Purver. Evaluating neural word representations in tensor-based compositional settings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 708–719, 2014.
[22] M. Moortgat. Multimodal linguistic inference. *Journal of Logic, Language and Information*, 5(3-4):349–385, 1996.
[23] R. Moot. *Proof Nets for Linguistic Analysis*. PhD thesis, Utrecht University, 2002.
[24] Tamara Polajnar, Luana Făgărăşan, and Stephen Clark. Reducing dimensions of tensors in type-driven distributional semantics. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1036–1046, Doha, Qatar, October 2014. Association for Computational Linguistics.
[25] P. Panangaden R. Blute and R. Seely. Fock space: a model of linear exponential types. *Mathematical Foundations of Programming Semantics*, pages 474–512, 1994.
[26] P. Selinger. A survey of graphical languages for monoidal categories. In Bob Coecke, editor, *New Structures for Physics*, volume 813 of *Lecture Notes in Physics book series*, pages 289–355. Springer, 2010.
[27] G. Wijnholds and M. Sadrzadeh. Evaluating composition models for verb phrase elliptical sentence embeddings. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 261–271, 2019.
[28] G. Wijnholds and M. Sadrzadeh. A type-driven vector semantics for ellipsis with anaphora using lambek calculus with limited contraction. *J of Log Lang and Inf*, 28:331–358, 2019.

8 Appendix

$$\overline{n \longrightarrow n} \qquad \overline{s, s \longrightarrow s, s}$$
$$n \longrightarrow n \qquad s, n, n\backslash s \longrightarrow s, s \quad \backslash_L$$
$$n, n\backslash s, n, n\backslash s \longrightarrow s, s \quad \backslash_L$$
$$n, n\backslash s,\ \nabla n, n\backslash s \longrightarrow s, s \quad \nabla_L$$
$$n,\ \nabla(n\backslash s),\ \nabla n, n\backslash s \longrightarrow s, s \quad \nabla_L$$
$$\nabla n,\ \nabla(n\backslash s),\ \nabla n, n\backslash s \longrightarrow s, s \quad \nabla_L$$
$$\overline{\nabla(n\backslash s) \longrightarrow\ \nabla(n\backslash s)} \qquad \nabla n,\ \nabla(n\backslash s), !(\nabla n), n\backslash s \longrightarrow s, s \quad contr_1$$
$$\nabla n,\ \nabla(n\backslash s), !(\nabla n),\ \nabla(n\backslash s),\ \nabla(n\backslash s)\backslash(n\backslash s) \longrightarrow s, s \quad \backslash_L$$
$$\nabla n,\ \nabla(n\backslash s),\ \nabla(n\backslash s), !(\nabla n),\ \nabla(n\backslash s)\backslash(n\backslash s) \longrightarrow s, s \quad perm$$
$$\overline{n \longrightarrow n} \qquad \nabla n, !(\nabla(n\backslash s)), !(\nabla n),\ \nabla(n\backslash s)\backslash(n\backslash s) \longrightarrow s, s \quad contr_2$$
$$\overline{\nabla n \longrightarrow\ \nabla n} \qquad \nabla n, !(\nabla(!(\nabla(n\backslash s))/n)), n, !(\nabla n),\ \nabla(n\backslash s)\backslash(n\backslash s) \longrightarrow s, s \quad /_L$$
$$\nabla n, !(\nabla(!(\nabla(n\backslash s))/n)),\ \nabla n,\ \nabla n\backslash n, !(\nabla n),\ \nabla(n\backslash s)\backslash(n\backslash s) \longrightarrow s, s \quad \backslash_L$$
$$\nabla n, !(\nabla(!(\nabla(n\backslash s))/n)),\ \nabla n,\ \nabla(\nabla n\backslash n), !(\nabla n),\ \nabla(n\backslash s)\backslash(n\backslash s) \longrightarrow s, s \quad \nabla_L$$
$$\nabla n, !(\nabla(!(\nabla(n\backslash s))/n)),\ \nabla n, !(\nabla(\nabla n\backslash n)), !(\nabla n),\ \nabla(n\backslash s)\backslash(n\backslash s) \longrightarrow s, s \quad contr_1$$
$$\nabla n,\ \nabla n, !(\nabla(!(\nabla(n\backslash s))/n)), !(\nabla(\nabla n\backslash n)), !(\nabla n),\ \nabla(n\backslash s)\backslash(n\backslash s) \longrightarrow s, s \quad perm$$
$$\overline{n \longrightarrow n} \qquad !(\nabla n), !(\nabla(!(\nabla(n\backslash s))/n)), !(\nabla(\nabla n\backslash n)), !(\nabla n),\ \nabla(n\backslash s)\backslash(n\backslash s) \longrightarrow s, s \quad contr_2$$
$$!(\nabla n), !(\nabla(!(\nabla(n\backslash s))/n)), !(\nabla(\nabla n\backslash n))/n, n, !(\nabla n),\ \nabla(n\backslash s)\backslash(n\backslash s) \longrightarrow s, s \quad /_L$$

Fig. 1. The strict reading of *Kim likes their code. Sam does too.*

$$\mathbf{D}$$
$$\overline{\nabla n \longrightarrow \nabla n} \qquad \nabla n, !(\nabla(!(\nabla(n\backslash s))/n)), n, n,\ \nabla n,\ \nabla(n\backslash s)\backslash(n\backslash s) \longrightarrow s, s \quad contr_2$$
$$\overline{\nabla n\ \longrightarrow\ \nabla n} \qquad \nabla n, !(\nabla(!(\nabla(n\backslash s))/n)), n,\ \nabla n,\ \nabla n\backslash n,\ \nabla n,\ \nabla(n\backslash s)\backslash(n\backslash s) \longrightarrow s, s \quad \backslash_L$$
$$\nabla n, !(\nabla(!(\nabla(n\backslash s))/n)),\ \nabla n, \nabla n\backslash n,\ \nabla n,\ \nabla n\backslash n,\ \nabla n,\ \nabla(n\backslash s)\backslash(n\backslash s) \longrightarrow s, s \quad \backslash_L$$
$$\nabla n, !(\nabla(!(\nabla(n\backslash s))/n)),\ \nabla n, \nabla n\backslash n,\ \nabla n,\ \nabla(\nabla n\backslash n),\ \nabla n,\ \nabla(n\backslash s)\backslash(n\backslash s) \longrightarrow s, s \quad \nabla_L$$
$$\nabla n, !(\nabla(!(\nabla(n\backslash s))/n)),\ \nabla n,\ \nabla(\nabla n\backslash n),\ \nabla n,\ \nabla(\nabla n\backslash n),\ \nabla n,\ \nabla(n\backslash s)\backslash(n\backslash s) \longrightarrow s, s \quad \nabla_L$$
$$\nabla n, !(\nabla(!(\nabla(n\backslash s))/n)),\ \nabla n,\ \nabla(\nabla n\backslash n),\ \nabla(\nabla n\backslash n),\ \nabla n,\ \nabla n,\ \nabla(n\backslash s)\backslash(n\backslash s) \longrightarrow s, s \quad perm'$$
$$\nabla n,\ \nabla n, !(\nabla(!(\nabla(n\backslash s))/n)),\ \nabla(\nabla n\backslash n),\ \nabla(\nabla n\backslash n),\ \nabla n,\ \nabla n,\ \nabla(n\backslash s)\backslash(n\backslash s) \longrightarrow s, s \quad perm$$
$$!(\nabla n), !(\nabla(!(\nabla(n\backslash s))/n)),\ \nabla(\nabla n\backslash n),\ \nabla(\nabla n\backslash n),\ \nabla n,\ \nabla n,\ \nabla(n\backslash s)\backslash(n\backslash s) \longrightarrow s, s \quad contr_2$$
$$!(\nabla n), !(\nabla(!(\nabla(n\backslash s))/n)), !(\nabla(\nabla n\backslash n)),\ \nabla n,\ \nabla n,\ \nabla(n\backslash s)\backslash(n\backslash s) \longrightarrow s, s \quad contr_2$$
$$\overline{n \longrightarrow n} \qquad !(\nabla n), !(\nabla(!(\nabla(n\backslash s))/n)), !(\nabla(\nabla n\backslash n)), !(\nabla n),\ \nabla(n\backslash s)\backslash(n\backslash s) \longrightarrow s, s \quad contr_2$$
$$!(\nabla n), !(\nabla(!(\nabla(n\backslash s))/n)), !(\nabla(\nabla n\backslash n))/n, n, !(\nabla n),\ \nabla(n\backslash s)\backslash(n\backslash s) \longrightarrow s, s \quad /_L$$

$$\overline{n \longrightarrow n} \qquad \overline{s, s \longrightarrow s, s}$$
$$\overline{n \longrightarrow n} \qquad s, n, n\backslash s \longrightarrow s, s \quad \backslash_L$$
$$n, n\backslash s, n, n\backslash s \longrightarrow s, s \quad \backslash_L$$
$$n, n\backslash s,\ \nabla n, n\backslash s \longrightarrow s, s \quad \nabla_L$$
$$n,\ \nabla(n\backslash s),\ \nabla n, n\backslash s \longrightarrow s, s \quad \nabla_L$$
$$\nabla n,\ \nabla(n\backslash s),\ \nabla n, n\backslash s \longrightarrow s, s \quad \nabla_L$$
$$\overline{\nabla(n\backslash s)\ \longrightarrow\ \nabla(n\backslash s)} \qquad \nabla n, !(\nabla(n\backslash s)),\ \nabla n, n\backslash s \longrightarrow s, s \quad contr_1$$
$$\nabla n, !(\nabla(n\backslash s)),\ \nabla n,\ \nabla(n\backslash s),\ \nabla(n\backslash s)\backslash(n\backslash s) \longrightarrow s, s \quad \backslash_L$$
$$\nabla n, !(\nabla(n\backslash s)),\ \nabla(n\backslash s),\ \nabla n,\ \nabla(n\backslash s)\backslash(n\backslash s) \longrightarrow s, s \quad perm$$
$$\overline{n \longrightarrow n} \qquad \nabla n, !(\nabla(n\backslash s)), !(\nabla(n\backslash s)),\ \nabla n,\ \nabla(n\backslash s)\backslash(n\backslash s) \longrightarrow s, s \quad contr_1$$
$$\overline{n \longrightarrow n} \qquad \nabla n, !(\nabla(n\backslash s))/n, n, !(\nabla(n\backslash s)),\ \nabla n,\ \nabla(n\backslash s)\backslash(n\backslash s) \longrightarrow s, s \quad /_L$$
$$\nabla n,\ \nabla(!(\nabla(n\backslash s))/n), n, !(\nabla(n\backslash s)),\ \nabla n,\ \nabla(n\backslash s)\backslash(n\backslash s) \longrightarrow s, s \quad \nabla_L$$
$$\nabla n,\ \nabla(!(\nabla(n\backslash s))/n), n, !(\nabla(n\backslash s))/n, n,\ \nabla n,\ \nabla(n\backslash s)\backslash(n\backslash s) \longrightarrow s, s \quad /_L$$
$$\nabla n,\ \nabla(!(\nabla(n\backslash s))/n), n,\ \nabla(!(\nabla(n\backslash s))/n), n,\ \nabla n,\ \nabla(n\backslash s)\backslash(n\backslash s) \longrightarrow s, s \quad \nabla_L$$
$$\mathbf{D} = \quad \nabla n,\ \nabla(!(\nabla(n\backslash s))/n),\ \nabla(!(\nabla(n\backslash s))/n), n, n,\ \nabla n,\ \nabla(n\backslash s)\backslash(n\backslash s) \longrightarrow s, s \quad perm$$

Fig. 2. The sloppy reading of *Kim likes their code. Sam does too.*

Algebraic Semantics for Full Displacement Calculus with Linguistic Subexponentials and Bracket Modalities

Oriol Valentín

Departament of Computer Science, Universitat Politècnica de Catalunya
ovalentin@cs.upc.edu

Abstract. This paper states and proves some key mathematical properties of the full Displacement calculus with linguistic subexponentials and bracket modalities ($\mathbf{DAb}^*\mathbf{1}!_b?$).
The universal exponential modality $!_b$ of $\mathbf{DAb}^*\mathbf{1}!_b?$ licenses a special version of the structural rule of Contraction which interacts with bracket modalities. In this way, the complexity of so-called parasitic gaps are accounted for.
On the other hand, the existential exponential modality '?' of $\mathbf{DAb}^*\mathbf{1}!_b?$ licenses the rule of Expansion (or Mingle), a limited version of the rule of Weakening. Expansion is involved in situations in which linguistic expressions are iterated, as in multi-coordination (more than two conjuncts), generalized non-constituent coordination.
This study provides a semantic proof of Cut-Elimination for the whole system, and, as a by-product, completeness with respect to Phase Models.

1 Introduction

In [11], the logic for the displacement calculus with the existential exponential **DA**? is shown to be sound and complete w.r.t. phase semantics (see [1]). This phase semantics is specially adapted for the core logic **D**. Here we extend the work of [11] to bracket operators and universal bracketed exponential which account for parasitic gaps in an elegant way (see [10]). We provide strong semantics completeness à la Okada (see [14]) for the whole system. This means that besides completeness w.r.t. to phase spaces we get a semantic/algebraic proof of the Cut-elimination theorem.

We survey key proof-theoretic findings (in particular, some proof-theoretic and linguistic pathologies), and we explain which road we take in proof-theoretic and algebraic terms.

We introduce phase spaces well-suited for **D** and we show how we algebraically deal with additive connectives, bracket modalities, universal and existential subexponential modalities. Finally, enough algebraic material is covered in order to state and prove strong completeness à la Okada. In the proofs of this paper we will cover essentially the new connectives added to **DA**?.

Before concluding, we will give some insights regarding the so-called Lambek's empty antecedent restriction.

2 Full Displacement Calculus with brackets and linguistic exponentials: $\mathbf{DAb^*1!_b?}$

$\mathbf{DAb^*1!_b?}$ denotes the Displacement calculus with units, additives and two linguistic exponentials: the existential ? and the universal $!_b$. **Lb** refers to the Lambek Calculus with brackets. The set of types $\mathcal{F}$, their sort, and their semantic map (**SEM**) are defined in Figure 1. $(\mathtt{Pr}_i)_i$ is the collection of sets of sort i, where $i \in \omega$. We will simply write $\mathtt{Pr}$.

0.	$\mathcal{F}_i ::= \mathtt{Pr}_i$	$\mathbf{SEM}(p)$	primitive type p of sort i
1.	$\mathcal{F}_i ::= \mathcal{F}_{i+j}/\mathcal{F}_j$	$\mathbf{SEM}(C/B) = \mathbf{SEM}(B) \to \mathbf{SEM}(C)$	over [5]
2.	$\mathcal{F}_j ::= \mathcal{F}_i \backslash \mathcal{F}_{i+j}$	$\mathbf{SEM}(A\backslash C) = \mathbf{SEM}(A) \to \mathbf{SEM}(C)$	under [5]
3.	$\mathcal{F}_{i+j} ::= \mathcal{F}_i \bullet \mathcal{F}_j$	$\mathbf{SEM}(A \bullet B) = \mathbf{SEM}(A) \& \mathbf{SEM}(B)$	continuous product [5]
4.	$\mathcal{F}_0 ::= I$	$\mathbf{SEM}(I) = \top$	continuous unit [4]
5.	$\mathcal{F}_{i+1} ::= \mathcal{F}_{i+j} \uparrow_k \mathcal{F}_j, 1 \leq k \leq i+j$	$\mathbf{SEM}(C \uparrow_k B) = \mathbf{SEM}(B) \to \mathbf{SEM}(C)$	extract [12]
6.	$\mathcal{F}_j ::= \mathcal{F}_{i+1} \downarrow_k \mathcal{F}_{i+j}, 1 \leq k \leq i+1$	$\mathbf{SEM}(A \downarrow_k C) = \mathbf{SEM}(A) \to \mathbf{SEM}(C)$	infix [12]
7.	$\mathcal{F}_{i+j} ::= \mathcal{F}_{i+1} \odot_k \mathcal{F}_j, 1 \leq k \leq i+1$	$\mathbf{SEM}(A \odot_k B) = \mathbf{SEM}(A) \& \mathbf{SEM}(B)$	discontinuous product [12]
8.	$\mathcal{F}_1 ::= J$	$\mathbf{SEM}(J) = \top$	discontinuous unit [12]
9.	$\mathcal{F}_i ::= \mathcal{F}_i \& \mathcal{F}_i$	$\mathbf{SEM}(A \& B) = \mathbf{SEM}(A) \& \mathbf{SEM}(B)$	additive conjunction [3, 8]
10.	$\mathcal{F}_i ::= \mathcal{F}_i \oplus \mathcal{F}_i$	$\mathbf{SEM}(A \oplus B) = \mathbf{SEM}(A) + \mathbf{SEM}(B)$	additive disjunction [3, 8]
11.	$\mathcal{F}_i ::= \langle\rangle \mathcal{F}_i$	$\mathbf{SEM}(\langle\rangle A) = \mathbf{SEM}(A)$	bracket modality [13]
12.	$\mathcal{F}_i ::= [\,]^{-1} \mathcal{F}_i$	$\mathbf{SEM}([\,]^{-1} A) = \mathbf{SEM}(A)$	antibracket modality [13]
13.	$\mathcal{F}_0 ::= ?\mathcal{F}_0$	$\mathbf{SEM}(?A) = \mathbf{SEM}(A)^+$	existential exponential [13]
14.	$\mathcal{F}_0 ::= !_b \mathcal{F}_0$	$\mathbf{SEM}(!_b A) = \mathbf{SEM}(A)$	universal (bracketed) exponential [13]

Fig. 1. Categorial logic types of $\mathbf{DAb^*1!_b?}$

The set of antecedents **Config** is defined by mutual recursion as follows:

(1) $\mathbf{Config} ::= \Lambda$
$\mathbf{Config} ::= 1\,\mathbf{Config}$
$\mathbf{Config} ::= \mathcal{F}_0\,\mathbf{Config}$
$\mathbf{Config} ::= \mathcal{F}_{i>0}\{\underbrace{\mathbf{Config}, \ldots, \mathbf{Config}}_{i\,\mathbf{Config}'\mathrm{s}}\}, \mathbf{Config}$
$\mathbf{Config} ::= [\mathbf{Config}]$

Notice a change on notation in the definition of configurations (1). Crucially, concatenation is done by juxtaposition. In other papers ([12],[9]) **D**-configurations use ',' as a metalinguistic concatenation. The sort $s(\Gamma)$ (for $\Gamma \in \mathbf{Config}$) is $|\Gamma|_1$, i.e. the number of metalinguistic separators 1 which it contains (including bracketed strings[1]).**Config** can be seen as an ω-indexed family of sets $(\mathbf{Config}_i)_{i\in\omega}$, where $\mathbf{Config}_i$ is the set of configurations of sort i.

The set of sequents **Seq** is defined as:

(2) $\mathbf{Seq} ::= \mathbf{Config} \Rightarrow \mathcal{F}$

[1] For example, the sort of $A\,1\,b(B\{1,1\}\,1)$ is equal to 4.

Sequents of **D** require that the sort of the antecedent equals the sort of the succendent. Sequents with an empty antecedent will be notated $\Lambda \Rightarrow A$ or $\Rightarrow A$. Distinguished subconfigurations of a configuration in **D** have been usually written as $\Delta\langle\Gamma\rangle$ (see [12]). In this paper, we simply denote these configurations as $\Delta(\Gamma)$. $\Delta()$ will refer to contexts.

For a configuration Δ we define the *type-equivalent* $\Delta^\bullet$, which is a type that has the same algebraic meaning as Δ. Via the BNF formulation of **Config** in (1) one can recursively define $\Delta^\bullet$:

(3) $\Lambda^\bullet =_{def} I$
$(1, \Gamma)^\bullet =_{def} J \bullet \Gamma^\bullet$
$(A, \Gamma)^\bullet =_{def} A \bullet \Gamma^\bullet$, if $s(A) = 0$
$(A\{\Delta_1, \ldots, \Delta_{s(A)}\}, \Gamma)^\bullet =_{def}$
$((\cdots(A \odot_1 \Delta_1^\bullet)\cdots) \odot_{1+s(\Delta_1)+\cdots+s(\Delta_{s(A)})} \Delta_{s(A)}^\bullet) \bullet \Gamma^\bullet$, if $s(A) > 0$

$$\frac{\Delta([\,A\,]) \Rightarrow B}{\Delta(\langle\rangle A) \Rightarrow B}\,\langle\rangle L \qquad \frac{\Delta \Rightarrow A}{[\Delta] \Rightarrow \langle\rangle A}\,\langle\rangle R$$

$$\frac{\Delta(A) \Rightarrow B}{\Delta([\,]^{-1}A]) \Rightarrow B}\,[\,]^{-1}L \qquad \frac{[\Delta] \Rightarrow A}{\Delta \Rightarrow [\,]^{-1}A}\,[\,]^{-1}R$$

$$\frac{\Delta(A) \Rightarrow B}{\Delta(!_b A) \Rightarrow B}\,!_b L \qquad \frac{!_b\Delta \Rightarrow A}{!_b\Delta \Rightarrow !_b A}\,!_b R^*,\ !_b\Delta \neq \Lambda$$

$$\frac{\Delta(!_b\Gamma\,[!_b\Gamma\,\Delta]) \Rightarrow A}{\Delta(!_b\Gamma\,\Delta) \Rightarrow A}\,!_b C^*,\ \text{where } \Delta \neq \Lambda \text{ and } !_b\Gamma \neq \Lambda$$

$$\frac{\Delta(!_b A\,\Gamma) \Rightarrow B}{\Delta(\Gamma\,!_b A) \Rightarrow B}\,!_b P1 \qquad \frac{\Delta(\Gamma\,!_b A) \Rightarrow B}{\Delta(!_b A\,\Gamma) \Rightarrow B}\,!_b P2$$

$$\frac{\Delta(A^i) \Rightarrow B \text{ for } i > 0}{\Delta(?A) \Rightarrow B}\,?L \qquad \frac{\Delta \Rightarrow A}{\Delta \Rightarrow ?A}\,?R$$

$$\frac{\Delta_1 \Rightarrow A \qquad \Delta_2 \Rightarrow ?A}{\Delta_1\,\Delta_2 \Rightarrow ?A}\,?M$$

Fig. 2. Modal connectives of $\mathbf{DAb}^*\mathbf{1}!_b?$

In order to simplify and focus on the fine interaction of the different modalities we only show the rules associated to these connectives, see Figure 3. Crucially, rules $!_b C^*$ and $!_b R$ have the following restrictions:

(4) – $!_b$-modalized configurations in rule $!C$ are non-empty. Δ must be also non-empty in rule $!_b C^*$.

– Rule $!_b\,R$ has as side-condition that antecedents must be non-empty.

2.1 On the Cut-Elimination Property and the $!_b$-licensed Rule of Contraction in DAb*1$!_b$?

In [2] some issues w.r.t. Cut-elimination are noticed. [2] also explores some potential linguistic pathologies related to the rule of $!_b$-contraction. Our goal in this subsection is to address these problems.

Consider the following derivation:

(5)
$$\dfrac{\dfrac{\Rightarrow I}{\Rightarrow !_bI}\,!_bR \qquad \dfrac{\dfrac{\dfrac{\dfrac{p\Rightarrow p}{[p]\Rightarrow\langle\rangle p}\,\langle\rangle R}{I\,[I\,p]\Rightarrow\langle\rangle p}\,IL \text{ applied twice}}{!_bI\,[!_bI\,p]\Rightarrow\langle\rangle p}\,!_bL \text{ applied twice}}{!_bI\,p\Rightarrow\langle\rangle p}\,!_bC}{p\Rightarrow\langle\rangle p}\,Cut$$

If $!_b$-modalized configurations cannot be empty, we lose the Cut-elimination property as (5) shows. But, if one allowed empty $!_b$-modalized configurations, clearly the sequent $p\Rightarrow\langle\rangle p$ could have a Cut-free derivation by contracting the empty $!_b$-modalized configuration:

(6)
$$\dfrac{\dfrac{\dfrac{p\Rightarrow p}{[p]\Rightarrow\langle\rangle p}\,\langle\rangle R}{!_bA\,[!_bA p]\Rightarrow\langle\rangle p}\,\langle\rangle R}{p\Rightarrow\langle\rangle p}\,!_b\,C,\ \text{where } !_bA \text{ is contracted}$$

Notice how in (6) we contract the empty configuration. In this case, we would get that **DAb*****1**$!_b$? is **not** a conservative extension of **Lb**, which is totally undesirable.

But, in fact the situation is more delicate than expected. If $\Rightarrow A$ is a **Lb** theorem (a provable sequent with empty antecedent), and p any atomic type (of sort 0) one easily derives $A\,[A\,p]\Rightarrow A\bullet\langle\rangle(A\bullet p)$. By applying $!_bL$ twice we get $!_bA\,[!_bA\,p]\Rightarrow A\bullet\langle\rangle(A\bullet p)$, and by $!_bC$ we obtain $!_bA\,p\Rightarrow A\bullet\langle\rangle(A\bullet p)$. By necessitation ($!_b$ R) over the empty antecedent $\Rightarrow !_bA$ is derived. Finally, by Cut between $\Rightarrow !_bA$ and $!_bA\,[!_bA\,p]\Rightarrow A\bullet\langle\rangle(A\bullet p)$ we obtain:

(7) $p\Rightarrow A\bullet\langle\rangle(A\bullet p)$

But this time the sequent from (7) clearly does not a Cut-free proof. The interaction of bracketed contraction and the rule of necessitation $!_b$R generates infinite sequents with no Cut-free proofs! Notice that the corresponding sequent without brackets from (7), i.e. $p\Rightarrow A^2\bullet p$ would have a Cut-free proof:

$$(8)\qquad \dfrac{\dfrac{\Rightarrow A \qquad \Rightarrow A}{\Rightarrow A^2}\bullet R \qquad\qquad p \Rightarrow p}{p \Rightarrow A^2 \bullet p}\bullet R$$

Some solutions to the pathologies originated by bracketed contraction
We have a priori two solutions:

a) Consider the full Displacement calculus with Lambek's restriction (no empty antecedents[2]). The rule of necessitation $!_bR$ would only apply with non-empty antecedents. This calculus notated $\mathbf{DAb}^+!_b?$, would not generate sequents like those in (7). However, $\mathbf{DAb}^+!_b?$ would have at least an important loss of linguistic descriptive adequacy. As it is now, **D** and related calculi of discontinuity are specially well-adapted to *gapping* phenomena (see [12]). Crucially, in the these calculi gapping categories have type[3] $(X\backslash X)/(X \odot I)$, where the continuous product unit closes off the point of discontinuity in gapping constructions. The use of I would be incompatible with Lambek's restriction.

b) We allow empty antecedents. It is the calculus $\mathbf{DAb}^*\mathbf{1}!_b?$ with the rules of necessitation ($!_b$R) and contractions with restrictions in such a way that issues with Cut-elimination property are avoided. See Figure 3 and the restrictions to rules with $!_b$ listed in (4). In this way, we can preserve the **D** descriptive adequacy of gapping. This is the road we take in this paper. Moreover $\mathbf{DAb}^*\mathbf{1}!_b?$ is a conservative extension of **DAb**!?.
In this calculus we rule out also a pathology related to parasitic gap (see [2]):
(10) *man that likes
If one types *that* with (cn\cn)/($!_b$n\s), a non restricted use of bracketed contraction would generate (10):

$$(11)\qquad \dfrac{\dfrac{}{cn\,cn\backslash cn \Rightarrow cn}\backslash L \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{n \Rightarrow n}{[n] \Rightarrow \langle\rangle n}\langle\rangle R \qquad s \Rightarrow s}{[n]\,(\langle\rangle n)\backslash s \Rightarrow s}\backslash L}{[n]\,(\langle\rangle n\backslash s)/n\,n \Rightarrow s}/L}{[!_bn](\langle\rangle n\backslash s)/n\,!_bn \Rightarrow s}!_bL \text{ applied twice}}{!_bn\,[!_bn](\langle\rangle n\backslash s)/n \Rightarrow s}!_bP}{!_bn\,(\langle\rangle n\backslash s)/n \Rightarrow s}!_bC}{(\langle\rangle n\backslash s)/n \Rightarrow\, !_bn\backslash s}\backslash R}{{}^*cn\,(cn\backslash cn)/(!_bn\backslash s)\,(\langle\rangle n\backslash s)/n \Rightarrow cn}/L$$

[2] Lambek's restriction would apply to empty (continuous) antecedents. But, "discontinuous empty antecedents" would be allowed, i.e., sequents like:

(9) $\underbrace{1\ldots1}_{n+1 \text{ separators}} \Rightarrow A$, with $s(A) = n+1$

[3] This is a typing in **D**.

In this derivation, the instance of $!_b$-contraction accepts configurations like $!\Gamma[!_b\Gamma\ \Delta]$ with Δ equal to the empty antecedent. If we adhere to restrictions listed in (4) we rule out the above derivation.

3 Phase Semantics for DAb!?

3.1 Phase Spaces

A bracketed separated monoid (MONb) $\mathbf{M} = (|\mathbf{M}|, \cdot, b, \epsilon, 1)$ is a free monoid with an extra-unary function symbol b and a distinguished generator called the *separator*, in notation 1.

The *sort* $i \in \omega$ of a (discontinuous) string $\alpha \in |\mathbf{A}|$ is the number of separators it contains and these punctuate it into i+1 maximal continuous substrings or segments. $s(\alpha)$ denotes the sort of α.

A subset B of the carrier set $|\mathbf{A}|$ is called a *same-sort* subset iff there exists an $i \in \omega$ such that for every $a \in B$, $s(a) = i$. This *same-sort* property gives us the way to interpret **D** (and all its extensions considered in this paper) in such a way that the sorting regime is consistently treated. Notice that $\emptyset$ vacuously satisfies the *same-sort* condition for every $i \in \omega$. We consider the ω-sorted set $(\mathcal{P}(|\mathbf{A}|)_i)_{i\in\omega}$, where for every i, $\mathcal{P}(|\mathbf{A}|)_i = \{X : X \text{ is a same-sort subset of sort } i\}$.

Definition 1 (Closure System). *An ω-sorted closure system $(\mathcal{C}_i)_{i\in\omega}$ on a separated monoid $\mathbf{A}$ is an ω-indexed collection such that $\mathcal{C}_i \subseteq \mathcal{P}\mathbf{A}_i$, and for every $\mathcal{B} \subseteq \mathcal{C}_i$, $\bigcap \mathcal{B} \in \mathcal{C}_i$.*

In particular, every $\mathcal{C}_i$ contains the intersection of an empty collection, i.e. $\bigcap \emptyset$, which is equal to $\mathcal{P}(|\mathbf{A}|)_i$. Elements of $\mathcal{C}$ are called *closed* sets. We define the following operators at the level of same-sort subsets:

- $b(F) = \{b(\alpha) : \alpha \in F\}$
- $b^{-1}(F) = \{\alpha : b(\alpha) \in F\}$
- $F \circ G =_{def} \{f \cdot g : f \in F \text{ and } g \in G\}$
- $F \circ_i G =_{def} \{f \times_i g : f \in F \text{ and } g \in G\}$
- $G//F =_{def} \{h : \forall f \in F, h \cdot f \in G\}$.
- $F\backslash\backslash G =_{def} \{h : \forall f \in F, f \cdot h \in G\}$
- $G\uparrow\uparrow_i F =_{def} \{h : \forall f \in F, h \times_i f \in G\}$
- $F\downarrow\downarrow_i G =_{def} \{h : \forall f \in F, f \times_i h \in G\}$
- $F \cup G$ is the standard union of sets, provided that both F, G are same-sort subsets and $s(F) = s(G)$.
- $F \cap G$ is the standard intersection of sets, provided that both F, G are same-sort subsets, and $s(F) = s(G)$.
- $\mathbb{I} =_{def} \{\epsilon\}$
- $\mathbb{J} =_{def} \{1\}$

We will write $G//f$ instead of $G//\{f\}$. Similarly, we use the same notation for $f\backslash\backslash G$, $G\uparrow\uparrow_i f$, and $f\downarrow\downarrow_i G$.

Remark: in order to simplify the exposition, we avoid the subtleties of a closure system over a sorted domain (in our case, the set of sorts is ω). Since non-linearity (iteration and bracketed contraction) applies to sort 0 domains, this simplification is justified.

Let $(\mathcal{C}_i)_{i\in\omega}$ be a closure system over a separated monoid $\mathbf{M}$. If A is a same-sort subset of $|\mathbf{M}|$, then $\overline{A}$ denotes the least closed (same-sort) subset containing A. For any same-sort subset G, it is readily seen that the $[G \mapsto \overline{G}]$ map satisfies the following properties:

i) Extensiveness, i.e.: $G \subseteq \overline{G}$.
ii) Monotony, i.e.: if $G_1 \subseteq G_2$ then $\overline{G_1} \subseteq \overline{G_2}$.
iii) Idempotence, i.e.: $\overline{\overline{G}} = \overline{G}$.

Definition 2 (Phases Spaces). *A* $\mathbf{DAb}^*\mathbf{1}!_b?$ *phase space* $\mathbf{P} = (\mathbf{M}, (\mathcal{C}_i)_i, J)$ *is a structure partially ordered by the relation of subset inclusion such that:*

1. $\mathbf{M}$ *is a bracketed separated monoid.*
2. $(\mathcal{C}_i)_i$ *is a* (ω*-sorted) closure system, and:*
   - *b) For all closed (same-sort) subset* F*, and for all* $x \in |\mathbf{M}|$*:*
     - *b.1)* $x\backslash\backslash F$*,* $F//x$*,* $F{\uparrow_i}{\uparrow_i}x$*, and* $x{\downarrow_i}{\downarrow_i}F$ *are closed subsets.*
     - *b.2)* $b^{-1}(F)$ *is a closed subset.*
   - *c)* J *is a subsemigroup of* $\mathbf{A}$ *such that:*
     - *c.1)* $\forall x_1, \cdots, x_n \in J - \{\epsilon\}$ *and* $\forall y \in |\mathbf{A}| - \{\epsilon\}$, $x_1 \cdot \ldots \cdot x_n \cdot y \in \overline{\{x_1 \cdot \ldots \cdot x_n \cdot b(x_1 \cdot \ldots \cdot x_n \cdot y)\}}$*, where* ϵ *denotes the empty string of* $\mathbf{M}$.
     - *c.2)* $\forall x \in J - \{\epsilon\}, \forall y \in |\mathbf{A}|$ $x \cdot y \in \overline{\{y \cdot x\}}$ *and* $y \cdot x \in \overline{\{x \cdot y\}}$

Notice the detailed algebraic properties of the subsemigroup J. Their purpose is to fit the proof-theoretic restrictions detailed in subsection 2.1.

Subsets of $|\mathbf{A}|$ obviously satisfy the residuation property w.r.t. bracket operators:

$$b(F) \subseteq G \text{ iff } F \subseteq b^{-1}(G) \tag{1}$$

The following basic properties are evident:

Lemma 1.

- $F \circ G \subseteq H$ *iff* $F \subseteq H//G$ *iff* $G \subseteq F\backslash\backslash H$.
- $F \circ_i G \subseteq H$ *iff* $F \subseteq H{\uparrow\uparrow_i}G$ *iff* $G \subseteq F{\downarrow\downarrow_i}H$.
- *By construction,* $\overline{F}$ *is the least closed subset such that* $F \subseteq \overline{F}$*. Hence:*
- *If* $A \subseteq F$ *and* $\overline{F} = F$ *then* $\overline{A} \subseteq \overline{F}$.

Lemma 2. *If A is closed, then:*

- $A//F, F\backslash\backslash A, A{\uparrow\uparrow}_i F$, *and* $F{\downarrow\downarrow}_i A$ *are closed.*
  Proof: $A{\uparrow\uparrow}_i F = \bigcap_{x\in F} A{\uparrow\uparrow}_i x$, *whence* $A{\uparrow\uparrow}_i F$ *is closed.* □
- *Similarly for the other implicative operations.*
- $\overline{F}\circ\overline{G} \subseteq \overline{F\circ G}$. *Similarly,* $\overline{F}\circ_i\overline{G} \subseteq \overline{F\circ_i G}$
- *Hence,* $\overline{\overline{F}\circ\overline{G}} \subseteq \overline{F\circ G}$, *and* $\overline{\overline{F}\circ_i\overline{G}} \subseteq \overline{F\circ_i G}$
- *It follows that* $\overline{F\circ\overline{G}} = \overline{F\circ G}$ *and* $\overline{F\circ_i\overline{G}} = \overline{F\circ_i G}$
  Proof: Let us see the case of $\circ_i$. $F\circ_i G \subseteq \overline{F\circ_i\overline{G}}$. *By residuation,* $F \subseteq \overline{F\circ_i\overline{G}}{\uparrow\uparrow}_i G$. $\overline{F\circ_i\overline{G}}{\uparrow\uparrow}_i G$ *is a closed subset (see previous proof). Hence,* $\overline{F} \subseteq \overline{F\circ_i\overline{G}}{\uparrow\uparrow}_i G$. *Applying again residuation, we have* $\overline{F}\circ_i G \subseteq \overline{F\circ G}$
  We repeat the process with G, *obtaining* $\overline{G} \subseteq \overline{F}{\downarrow\downarrow}_i\overline{F\circ_i G}$. *It follows that:* $\overline{F}\circ_i\overline{G} \subseteq \overline{F\circ_i G}$. *Hence,* $\overline{\overline{F}\circ_i\overline{G}} \subseteq \overline{F\circ_i G}$

Lemma 3. *For any subset G, the following equality holds:*

$$\overline{b(G)} = \overline{b(\overline{G})}$$

Proof. Since $G \subseteq \overline{G}$, we have $b(G) \subseteq b(\overline{G})$. Applying closure we get $\overline{b(G)} \subseteq \overline{b(\overline{G})}$. For the converse, by extensiveness (of the closure map), we have $b(G) \subseteq \overline{b(G)}$. By residuation of the pair (b, b^{-1}), it follows that $G \subseteq b^{-1}(\overline{b(G)})$. Hence, by definition of cl, and that $b^{-1}(\overline{b(G)})$ is closed ($\overline{b(G)}$ is closed and property 2d) in definition 2) $\overline{G} \subseteq b^{-1}(\overline{b(G)})$, whence by residuation $b(\overline{G}) \subseteq (\overline{b(G)})$. Applying closure, we get $\overline{b(\overline{G})} \subseteq \overline{b(G)}$. □

3.2 Algebra of Closed Sets

We see now operations on closed subsets which return values into the set of closed subsets. This paves the path to the definition of valuations from the set of types into phase spaces, concretely into the set of closed sets. Given F, G closed sets:

(12)
$$\begin{array}{ll}
F\overline{\circ}G & =_{def} \overline{F\circ G}\\
F\overline{\circ_i}G & =_{def} \overline{F\circ_i G}\\
F\overline{/}G & =_{def} F/G\\
F\overline{\&}G & =_{def} F\cap G \text{ (By definition of phase spaces, } F\cap G \text{ is closed)}\\
F\overline{\cup}G & =_{def} \overline{F\cup G}\\
 & \quad \text{Similarly for the other implications.}\\
\langle\rangle F & =_{def} \overline{b(F)}\\
[\,]^{-1}F & =_{def} b^{-1}(F) \text{ (By lemma 3, } b^{-1}(F) \text{ is closed)}\\
!_b F & =_{def} \overline{F\cap J}\text{, where J is the special subset of the phase space}\\
?F & =_{def} \bigcup_{i\geq 1}\overline{F^i}
\end{array}$$

3.3 Models for DAb*1!$_b$?

A phase model for $\mathbf{DAb^*1!}_b?$ is a pair $(\mathbf{P}, v)$ where $\mathbf{P} = (\mathbf{A}, (\mathcal{C}_i)_i, J)$ is a phase space and v is a mapping $\mathtt{Pr} \longrightarrow (\mathcal{C}_i)_i$ which recursively extends to the set of types $\mathcal{F}$ as follows (again, we focus on the modal connectives):

Definition 3.

$$v(\langle\rangle A) =_{def} \overline{b(v(A))} \quad \text{and } v\,([\,]^{-1}A) =_{def} b^{-1}(v(A))$$
$$v(?A) =_{def} \overline{\bigcup_{i\geq 1} v(A)^i} \quad \text{and } v\,(!_b A) =_{def} \overline{v(A) \cap J}$$

Lemma 4. *(***Going to the context)**
If $v(\Gamma_1) \subseteq v(\Gamma_2)$ *then* $v(\Delta(\Gamma_1)) \subseteq v(\Delta(\Gamma_2))$.

Proof. By the tonicity properties of continuous and discontinuous product and the associativity of product, we have that $v(\Delta(\Gamma_1)) = v(\Delta())(v(\Gamma_1)) \subseteq v(\Delta())(v(\Gamma_2)) = v(\Delta(\Gamma_2))$.

□

In the following, $\mathbf{P}$ denotes a phase space, and $v : \mathrm{Pr} \longrightarrow (\mathcal{C}_i)_i$ a valuation in $\mathbf{P}$.

Lemma 5. *For any types* A, B *we have that:*

$$a)\ v([A]) = v(\langle\rangle A)$$
$$b)\ v([[\,]^{-1}A]) \subseteq v(A)$$

Proof. a) is satisfied by definition. b) is proved as follows. We have the identity $v([\,]^{-1}A) = v([\,]^{-1}A)$, i.e. $b^{-1}(v(A)) = b^{-1}(v(A))$, from which, by residuation of (b, b^{-1}), we get $b(b^{-1}(v(A))) \subseteq v(A)$, which is equivalent to $b(v([\,]^{-1}A))) \subseteq v(A)$. Taking closure, we get $v([[\,]^{-1}A]) \subseteq v(A)$. □

Lemma 6 (Bracketed contraction). *For any* Δ *and* Γ*:*

$$v(!\Delta\,\Gamma) \subseteq v(!\Delta\,[!\Delta\,\Gamma])$$

Proof. Recall that $!\Delta$ is a list of !-modalized types, so that we have $!\Delta = !_bA_1 \cdots !_bA_n$, for $n \geq 1$. From $v(!_bA_i) = \overline{v(A) \cap J}$, $v(!_bA_i) \subseteq \overline{J}$. Since J is subsemigroup[4] of $|\mathbf{P}|$, it follows that $v(!_bA_1) \circ \cdots \circ v(!_bA_n) \subseteq \overline{J}$. Let $x_i \in v(!_bA_i)$ for any i, and $y \in v(\Delta)$. By the definition of a phase space, we have that $x_1 \cdot \ldots \cdot x_n \cdot y \in \overline{\{x_1 \cdot \ldots \cdot x_n \cdot b(x_1 \cdot \ldots \cdot x_n \cdot y)\}}$. Therefore, $v(!_bA_1) \circ \ldots \circ v(!_bA_n) \circ v(\Gamma)) \subseteq v(!_bA_1) \circ \ldots \circ v(!_bA_n) \circ \overline{b(v(!_bA_1) \circ \ldots \circ v(!_bA_n) \circ v(\Gamma))}$. Taking closure over the last set inclusion, we get $\overline{v(!_bA_1) \circ \ldots \circ v(!_bA_n) \circ v(\Gamma))} \subseteq \overline{v(!_bA_1) \circ \ldots \circ v(!_bA_n)} \overline{\circ b(v(!_bA_1) \circ \ldots \circ v(!_bA_n) \circ v(\Gamma))}$.p □

Lemma 7 (Controlled Permutation). *For any type* A *and configuration* Γ*:*

$$a)\ v(!_bA\,\Gamma)) \subseteq v(\Gamma\,!_bA)$$
$$b)\ v(\Gamma!_bA)) \subseteq v(!_bA\,\Gamma)$$

[4] But not a submonoid, for otherwise the universal subexponential would allow weakening!

Proof. We prove a). b) has a completely similar proof. Let $x \in v(!_b A)$ and $y \in v(\Gamma)$. By definition of a phase space for $\mathbf{DAb^*1}!_b?$, $x \in J$, and $\overline{x \cdot y \in \{y \cdot x\}}$. Mimicking the reasoning of the previous lemma 6, we have that $\overline{v(!_b A) \circ v(\Gamma)} \subseteq \overline{v(\Gamma) \circ v(!_b A)}$.

Theorem 1 (Soundness). *If* $\mathbf{DAb^*1}!_b? \vdash \Delta \Rightarrow A$ *then* $\models \Delta \Rightarrow A$.

Proof. By induction on the length of $\mathbf{DAb^*1}!_b?$ derivations.

- Cut: We prove it simply by going to the context (see lemma 4).

- $\langle\rangle$:

$$\frac{\Delta([A]) \Rightarrow B}{\Delta(\langle\rangle A) \Rightarrow B}\,\langle\rangle L \quad \frac{\Delta \Rightarrow A}{[\Delta] \Rightarrow \langle\rangle A}\,\langle\rangle R$$

Assume $v(\Delta([A])) \subseteq v(B)$. By lemma 5, $v([A]) = v(\langle\rangle A)$. By going to the context we get $v(\Delta(\langle\rangle A)) \subseteq v(B)$.

- $[\,]^{-1}$:

$$\frac{\Delta(A) \Rightarrow B}{\Delta([[\,]^{-1} A]) \Rightarrow B}\,[\,]^{-1} L \quad \frac{[\Delta] \Rightarrow A}{\Delta \Rightarrow [\,]^{-1} A}\,[\,]^{-1} R$$

By lemma 5, $v([[\,]^{-1} A]) \subseteq v(A)$. By going to the context, the conclusion of rule $[\,]^{-1}$L is sound. The premise of rule $[\,]^{-1}$ R reads semantically as $v([\Delta]) \subseteq v(A)$. $b(v(\Delta) \subseteq v([\Delta])$. It follows that $b(v(\Delta) \subseteq v(A)$. By residuation of the pair (b, b^{-1}), we have $v(\Delta) \subseteq b^-1(v(A))$. But $v([\,]^{-1} A) = b^-1(v(A))$, whence $v(\Delta) \subseteq v([\,]^{-1} A)$.

- !:

$$\frac{\Delta(A) \Rightarrow B}{\Delta(!_b A) \Rightarrow B}\,!_b L \quad \frac{!_b \Delta \Rightarrow A}{!_b \Delta \Rightarrow !_b A}\,!_b R$$

Recall that $v(!_b A) = \overline{v(A) \cap J}$. $v(A) \cap J \subseteq v(A)$. Taking closure we have that $v(!_b A) \subseteq v(A)$. Going to the context, we get the desired conclusion. This proves rule $!_b$ L. $v(!_b A_1) \circ \cdots \circ v(!_b A_n) \subseteq v(!_b A_1 \ldots !_b A_n)$. By the definition ! and the fact that the set J of the underlying phase space is a semigroup $v(!_b A_1) \circ \cdots \circ v(!_b A_n) \subseteq J$. Hence, $v(!_b A_1) \circ \cdots \circ v(!_b A_n) \subseteq v(A) \cap J$. Taking closure we get the desired conclusion. This proves rule $!_b$ R.

Let us see the structural rules licensed by !.

- !-contraction:

$$\frac{\Delta(!\Gamma\,[!_b \Gamma\,\Delta]) \Rightarrow A}{\Delta(!\Gamma\,\Delta) \Rightarrow A}\,!_b C$$

By lemma 6, $v(!_b \Gamma\,\Delta) \subseteq v(!_b \Gamma\,[!_b \Gamma\,\Delta])$. By going to the context, we conclude.

- !-permutation:

$$\frac{\Delta(!_b A\,\Gamma) \Rightarrow B}{\Delta(\Gamma\,!_b A) \Rightarrow B}\,!_b P1 \qquad \frac{\Delta(\Gamma\,!_b A) \Rightarrow B}{\Delta(!_b A\,\Gamma) \Rightarrow B}\,!_b P2$$

We prove the case of !P1 since !P2 is completely similar. By lemma 7, we have $v(!_b A\,\Gamma) \subseteq v(\Gamma\,!_b A)$. By going to the context we get the desired conclusion. □

4 Semantic Cut-Elimination and Completeness

4.1 The Syntactic Phase Space

Let $\mathbf{M}$ be the following algebra $(\mathbf{Config}, conc, b, \Lambda)$, where **Config** is the set of configurations, which by its inductive definition is closed by concatenation, and does not contain the empty configuration Λ. We denote the concatenation by *conc*. Hence, $\mathbf{M}$ is a monoid. Define $b : \mathbf{Config} \longrightarrow \mathbf{Config}$ as the mapping such that $b(\Delta) =_{def} [\Delta]$ for any configuration Δ. By the definition of **Config**, $\mathbf{M}$ is closed by b. It follows that $\mathbf{M}$ is a bracketed monoid. We define J as the set of !-modalized non-empty lists. Clearly, J is a subsemigroup of $\mathbf{M}$.

We define $\vdash^{-1}(A)$ as the set $\{\Delta : \Delta \Rightarrow {}^{-}A\}$. Let $(\mathcal{C}_i)_{i\vdash}$ be the smallest subset of $\mathcal{P}(|\mathbf{M}|)$ closed by arbitrary intersections containing the set $\{\vdash^{-1}(D) : D \in \mathcal{F}\}$. We will use the following crucial property:

$$\forall F \in (\mathcal{C}_i)_{i\vdash}\ \exists \mathcal{D}_F \subseteq (\mathcal{C}_i)_{i\vdash} \text{ such that } F = \textstyle\bigcap_{D\in\mathcal{D}_F} \vdash^{-1}(D) \quad (\star)$$

Lemma 8. *(***The syntactic phase space)**
Let $\mathbf{Synt} = (\mathbf{M}, (\mathcal{C}_i)_{i\vdash}, J)$. **Synt** *is a phase space.*

Proof. In the following F denotes an arbitrary closed set.
- Let $\Gamma \in \mathbf{Config}$. Consider $F//\Gamma$. Then, for any $\Delta \in \mathbf{Config}$ $\Delta \in F$ iff $\Delta, \Gamma \Rightarrow D$ for any $D \in \mathcal{D}$ for a certain $\mathcal{D}$ iff $\forall D \in \mathcal{D}$, $\Delta \Rightarrow D//\Gamma^{\bullet}$. It follows that $F//\Gamma$ is an arbitrary intersection of basic closed sets, whence it is a closed set. The other implicative connectives have a similar proof.
- $\Delta \in b^{-1}(F)$ iff $[\Delta] \in F$ iff $\forall D \in \mathcal{D}$ (for a given a family of closed subsets),$[\Delta] \Rightarrow D$ iff $\forall D \in \mathcal{D}$, $\Delta \in [\,]^{-1}D$, whence $b^{-1}(F)$ is the intersection of a family of (basic) closed sets. □

Let us prove condition *c*.1) from definition 2. We have to see that for any $!_b\Gamma_1, \cdots, !_b\Gamma_n, \Delta$:

$$!_b\Gamma_1 \ldots !_b\Gamma_n\, \Delta \in \overline{\{!_b\Gamma_1 \ldots !_b\Gamma_n\, [!_b\Gamma_1 \cdots !_b\Gamma_n\, \Delta]\}}$$

By $(\star)$ there exists $\mathcal{D} \subseteq (\mathcal{C}_i)_{i\vdash}$ such that $\overline{\{!_b\Gamma_1 \ldots !_b\Gamma_n\, [!_b\Gamma_1 \cdots !_b\Gamma_n\, \Delta]\}} = \bigcap_{D\in\mathcal{D}} \vdash^{-1}(D)$. We have that:

$$\begin{gathered}
!_b\Gamma_1 \ldots !_b\Gamma_n\, [!_b\Gamma_1 \cdots !_b\Gamma_n\, \Delta] \in \overline{\{!_b\Gamma_1 \ldots !_b\Gamma_n\, [!_b\Gamma_1 \cdots !_b\Gamma_n\, \Delta]\}} \\
\text{iff} \\
!_b\Gamma_1 \ldots !_b\Gamma_n\, [!_b\Gamma_1 \cdots !_b\Gamma_n\, \Delta] \in \vdash^{-1}(D) \forall D \in \mathcal{D} \\
\text{iff} \\
\forall D \in \mathcal{D},\ !_b\Gamma_1 \ldots !_b\Gamma_n\, [!_b\Gamma_1 \cdots !_b\Gamma_n\, \Delta] \Rightarrow D, \\
\text{iff} \\
\forall D \in \mathcal{D},\ !_b\Gamma_1 \ldots !_b\Gamma_n\, \Delta \Rightarrow D \text{ by } !_bC \\
\text{iff} \\
!_b\Gamma_1 \ldots !_b\Gamma_n\, \Delta \in \overline{\{!_b\Gamma_1 \ldots !_b\Gamma_n\, [!_b\Gamma_1 \cdots !_b\Gamma_n\, \Delta]\}}
\end{gathered}$$

This proves c.1). c.2) is proved similarly. There exists another family $\mathcal{D}$ of closed sets such that:

$$!_b\Gamma_2\, !_b\Gamma_1 \in \overline{\{!_b\Gamma_2\, !_b\Gamma_1\}}$$
iff
$$\forall D \in \mathcal{D},\ !_b\Gamma_2\, !_b\Gamma_1 \in \vdash^{-1}(D)$$
iff
$$\forall D \in \mathcal{D},\ !_b\Gamma_2\, !_b\Gamma_1 \Rightarrow D$$
iff
$$\forall D \in \mathcal{D}, !_b\Gamma_1\, !_b\Gamma_2 \Rightarrow D \text{ by } !_bP$$

Notice that in the last step above we have applied several times the rule of permutation $!_bP$. This proves c.2). □

4.2 The Truth Lemma and its Corollaries

Lemma 9. *Let v be the valuation $v : \mathtt{Pr} \to (\mathcal{C}_i)_{i\vdash}$ such that $v(p) =_{def} \vdash^{-1}(p)$ for any primitive type p. There holds:*

(13) $\overrightarrow{A} \in v(A) \subseteq \vdash^{-1}(A)$ *for any type A*

Proof. We consider the more relevant connectives for the main topic of this paper, i.e. bracketed controlled universal and existential exponential, and bracket modalities. For the other connectives, see [11].

- Let us prove $\overrightarrow{\langle\rangle A} \in v(\langle\rangle A) \subseteq \vdash^{-1}(\langle\rangle A)$. Let us see the second inclusion. By i.h. $v(A) \subseteq \vdash^{-1}(A)$. Let Δ any arbitrary configuration such that $\Delta \in v(A)$. Hence, $\Delta \in \vdash^{-1}(A)$, i.e. $\Delta \Rightarrow^{-} A$, from which we get by $\langle\rangle R$, $[\Delta] \Rightarrow^{-} \langle\rangle A$. Therefore $[v(A)] \subseteq \vdash^{-1}(\langle\rangle A)$. Taking closure to the preceding inclusion we have $v(\langle\rangle A) \subseteq \vdash^{-1}(\langle\rangle A)$.

Let us prove the first belonging. There exists $\mathcal{F}$ such that $v(\langle\rangle A) = \bigcap_{D\in\mathcal{F}} \vdash^{-1}(D)$. By i.h. $\overrightarrow{A} \in v(A)$. Therefore, $[\overrightarrow{A}] \in v(\langle\rangle A)$. It follows that $[\overrightarrow{A}] \in \vdash^{-1}(D)$ for any $D \in \mathcal{F}$, i.e. $[\overrightarrow{A}] \Rightarrow^{-} D$ for any $D \in \mathcal{F}$. By $\langle\rangle L$ rule we get $\overrightarrow{\langle\rangle A} \Rightarrow^{-} D$ for any $D \in \mathcal{F}$, i,e. $\overrightarrow{\langle\rangle A} \in \vdash^{-1}(D)$ for any $D \in \mathcal{F}$. Hence, $\overrightarrow{\langle\rangle A} \in v(\langle\rangle A)$.

- Let us prove $\overrightarrow{[\,]^{-1}A} \in v([\,]^{-1}A) \subseteq \vdash^{-1}([\,]^{-1}A)$.

Let us see the second inclusion. We have $v([\,]^{-1}A) = b^{-1}(v(A))$. For any configuration Δ, $\Delta \in b^{-1}(A)$ iff $[\Delta] \in v(A)$. Since by i.h $v(A) \subseteq \vdash^{-1}(A)$, we have that $[\Delta] \in \vdash^{-1}(A)$, which means $[\Delta] \Rightarrow^{-} A$, whence, by $[\,]^{-1}R$ rule, $\Delta \Rightarrow^{-} [\,]^{-1}A$, i.e. $\Delta \in \vdash^{-1}([\,]^{-1}A)$. Therefore, $v([\,]^{-1}A) \subseteq \vdash^{-1}([\,]^{-1}A)$.

Let us see the first belonging. There exists $\mathcal{F}$ such that $v(A) = \bigcap_{D\in\mathcal{F}} \vdash^{-1}(D)$. By i.h. $\overrightarrow{A} \in v(A)$. This means that,for any $D \in \mathcal{F}$:

$$\overrightarrow{A} \Rightarrow^{-} D$$
iff
$$\overrightarrow{[\,]^{-1}A}] \Rightarrow^{-} D$$

Therefore, $\overrightarrow{[\,]^{-1}A}] \in v(A)$, whence $\overrightarrow{\,]^{-1}A} \in b^{-1}(v(A) = v([\,]^{-1}A)$.

- $!_bA \in v(!_bA) \in \vdash^{-1}(!A)$

Let us see that $v(!_bA) \subseteq \vdash^{-1}(!_bA)$. We have $v(A) \cap J \subseteq \vdash^{-1}(A) \cap J$. Any $!_b\Gamma$ belongs to $v(A) \cap J$. By $!_bR$, $!\Gamma \Rightarrow^{-} !_bA$, i.e., $!\Gamma \in \vdash^{-1}(!_bA)$. It follows that $v(A) \cap J \subseteq \vdash^{-1}(!_bA)$. Taking closure to the previous subset inclusion, we get $v(!_bA) \subseteq \vdash^{-1}(!_bA)$. Let us prove now that $!_bA \in v(!_bA)$. By i.h., $A \in v(A)$, whence $!_bA \in v(A) \cap J$. It follows that (taking closure) $!_bA \in v(!_bA)$.

- Case $A = B \oplus C$. By i.h. $v(B) \subseteq [B]$ and $v(C) \subseteq [C]$. Hence, $v(B) \cup v(C) \subseteq cl([B] \cup [C]) \subseteq [B \oplus C]$. The first inclusion is due to the monotony property and properties of cl. In fact, we have $[B] \cup [C] \subseteq [B \oplus C]$. For, $[B] \subseteq [B \oplus C]$ and $[C] \subseteq [B \oplus C]$ by $\oplus iR$ $(i = 1, 2)$. It follows that $cl(v(B) \cup v(C)) \subseteq [B \oplus C]$.

On the other hand, $v(B \oplus C) = \bigcap_{D \in \mathcal{G}}[D]$ for a certain $\mathcal{G}$. By i.h $\overrightarrow{B} \in v(B)$. Hence, $\overrightarrow{B} \subseteq cl(v(B) \cup v(C))$. Similarly, $\overrightarrow{C} \subseteq cl(v(B) \cup v(C))$. Therefore, for any $D \in \mathcal{G}$, $\overrightarrow{B} \in [D]$ and $\overrightarrow{C} \in [D]$. By $\oplus L$ we get $\overrightarrow{B \oplus C} \in [D]$. It follows that $\overrightarrow{B \oplus C} \subseteq v(B \oplus C)$.

- Case $C = ?A$

$$(14)\qquad \dfrac{\Gamma_1 \Rightarrow A \qquad \dfrac{\dfrac{\Gamma_{i-1} \Rightarrow A \qquad \dfrac{\Gamma_i \Rightarrow A}{\Gamma_i \Rightarrow ?A}\,?R}{\vdots}\,?M}{\Gamma_2, \ldots, \Gamma_i \Rightarrow ?A}\,?M}{\Gamma_1, \ldots, \Gamma_i \Rightarrow ?A}\,?M$$

The proof above shows that for every $i > 0$, $v(A)^i \subseteq \vdash^{-1}(?A)$. We have then $\bigcup_{i>0} v(A)^i \subseteq \vdash^{-1}(?A)$. Applying the closure map we get $\overline{\bigcup_{i>0} v(A)^i} \subseteq \vdash^{-1}(?A)$, whence $v(?A) \subseteq \vdash^{-1}(?A)$.
We prove now $?A \in v(?A)$. We know that $v(?A) = \bigcap_{D \in \mathcal{G}} \vdash^{-1}(D)$, for a certain family of closed sets $\mathcal{G}$. By i.h. $A \in v(A)$. It follows that for every $i > 0$ $A^i \in v(A^i)$, whence $A^i \in \bigcup_{k>0} v(A^i) \subseteq v(?A)$. We have therefore:

For every $i > 0$ $A^i \in v(?A)$ iff For every $i > 0$, and for every D $\in \mathcal{G}$, $A^i \in \vdash^{-1}(D)$
iff For every D $\in \mathcal{G}$, $?A \in \vdash^{-1}(D)$, by application of $?R$
iff $?A \in v(?A)$

□

Theorem 2 (Strong Completeness à la Okada).
Let $\Delta \Rightarrow A$ be such that for every $(\mathbf{P}, v)$, $(\mathbf{P}, v) \models \Delta \Rightarrow B$. It follows that $\Delta \Rightarrow {}^{-}B$.

Proof. In particular, this sequent holds in the syntactic phase displacement model. By the previous lemma, for any A, $A \in v(A)$. Hence $\Delta \in v(\Delta)$. By soundness, for every $(\mathbf{P}, w)$ $w(\Delta) \subseteq w(B)$. Therefore we have that $v(\Delta) \subseteq v(B)$. Since $\Delta \in v(\Delta)$, $\Delta \in v(A)$, which entails (by the truth lemma) that $\Delta \in \vdash^{-1}(A)$, i.e. $\Delta \Rightarrow {}^{-}A$. □

By the previous theorem $\Delta \Rightarrow A$ is provable without Cut, whence:

Corollary 1 (Cut admissibility). *The Cut rule is admissible.*

□

5 On Lambek's Empty Antecedent Restriction

Since in this paper we have advocated for a system of categorial logic with the possibility of empty antecedents, it is worth considering what happens with Lambek's Empty Antecedent Restriction, and its source of ungrammatical provable sequents. Essentially, endocentric syntactic categories of the form X/X or $X\backslash X$ can cause problems (see [7],[2]).

If one types adjectives with $adj{=}_{def}cn/cn$, then intensifiers like 'very' with type adj/adj generate undesirable expressions like '*very man'. This is caused by the fact that $*adj/adj\, cn \Rightarrow cn$ is a provable sequent if we allow empty antecedents. By the same token, verbal intensifiers like 'very quickly' with endocentric types adv/adv and $adv{=}_{def}(n\backslash s)\backslash(n\backslash s)$ can generate 'john ran very', since it is provable the sequent: $*n\, n\backslash s\, adv/adv \Rightarrow S$

Finally, coordination categories defined as $\mathbf{coord}(\mathbf{X}){=}_{def}(\mathbf{X}\backslash\mathbf{X})/\mathbf{X}$ suffer from this source of ungrammaticality.

In this paper, we allow in our calculus empty antecedents since we need units for closing off points of discontinuity like those in gapping phenomena. We outline a possible solution to the problem of endocentric categories.The idea is to use finite sets of atomic postulates.[5] By closure of the entailment relation, we can maintain Cut-elimination.

- Avoiding the pathology *Very man

$$\begin{aligned} adv &= \overline{adj}/\overline{adj} \\ \overline{adj} &= (cn/\overline{cn}) \\ &\vdash cn \Rightarrow \overline{cn} \\ &\nvdash \overline{cn} \Rightarrow cn \end{aligned} \tag{2}$$

[5] A reviewer points out another interesting solution: the use of Moortgat/Oehrle ([6]) style inclusion relations, e.g. $cn \Rightarrow [\,]^{-1}\langle\rangle cn$.

We have the following facts:

$$\begin{aligned} &\vdash adv^{+}\, adj^{+}\, cn \Rightarrow cn \\ &\vdash adj^{+}\, cn \Rightarrow cn \\ &\nvdash adv^{+} cn \Rightarrow cn \end{aligned} \tag{3}$$

- Avoiding the pathology *John ran very

$$\begin{aligned} \overline{adv} &= (\overline{n}\backslash s)\backslash(n\backslash s) \\ &\vdash n \Rightarrow \overline{n} \\ &\nvdash \overline{n} \Rightarrow n \\ intens &= \overline{adv}/\overline{adv} \\ vp &= n\backslash s \end{aligned} \tag{4}$$

We have the following facts:

$$\begin{aligned} &\vdash adv^{+} adv \Rightarrow adv \\ &\nvdash n\, n\backslash s\, intens \Rightarrow s \\ &\vdash intens^{+}\, adv^{+} \Rightarrow adv \\ &\vdash intens^{+}\, intens^{*} \Rightarrow intens \end{aligned} \tag{5}$$

Playing with atomic postulates one can also deal with coordination categories like $\mathbf{coord}(\mathbf{X}) = (X\backslash X)/X$. This outline of a solution to the problem of empty antecedents should be more carefully studied in order to see how atomic postulates interact with the lexicon.

6 Conclusions

We have proved completeness of $\mathbf{DAb}^{*}!_{b}?$ w.r.t. phase spaces. Moreover, a novel semantic proof of Cut-elimination for the whole system is given. Readers may ask whether all the proof-theoretic machinery used in this paper is needed to account for extraction with parasitic gaps. An important research line would point to investigate the minimal proof-theoretic machinery needed to account for parasitic gaps and related phenomena. In [10] it is given a decidability proof of a $\mathbf{DAb}^{*}\mathbf{1}!_{b}?$ fragment without additives. It remains open to see whether we can extend the decidability result to additives.

Bibliography

[1] Girard, J.Y.: Linear logic. Theoretical Computer Science **50**, 1–102 (1987)

[2] Kanovich, M.I., Kuznetsov, S., Scedrov, A.: The multiplicative-additive lambek calculus with subexponential and bracket modalities. CoRR **abs/2008.00075** (2020), https://arxiv.org/abs/2008.00075

[3] Lambek, J.: On the Calculus of Syntactic Types. In: Jakobson, R. (ed.) Structure of Language and its Mathematical Aspects, Proceedings of the Symposia in Applied Mathematics XII, pp. 166–178. American Mathematical Society, Providence, Rhode Island (1961). https://doi.org/10.1090/psapm/012/9972

[4] Lambek, J.: Categorial and Categorical Grammars. In: Oehrle, R., Bach, E., Wheeler, D. (eds.) Categorial Grammars and Natural Language Structures, Studies in Linguistics and Philosophy, vol. 32, pp. 297–317. D. Reidel, Dordrecht (1988). https://doi.org/10.1007/978-94-015-6878-4 11

[5] Lambek, J.: The mathematics of sentence structure. American Mathematical Monthly **65**, 154–170 (1958). https://doi.org/10.2307/2310058

[6] Moortgat, M., Oehrle, R.: Adjacency, dependency and order. In: Dekker, P., Stokhof, M. (eds.) Proceedings of the Ninth Amsterdam Colloquium. pp. 447–466. ILLC, Amsterdam (1994)

[7] Moot, R., Retoré, C.: The Logic of Categorial Grammars: A Deductive Account of Natural Language Syntax and Semantics. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31555-8

[8] Morrill, G.: Grammar and Logical Types. In: Stockhof, M., Torenvliet, L. (eds.) Proceedings of the Seventh Amsterdam Colloquium. pp. 429–450. University of Amsterdam, Amsterdam (1990)

[9] Morrill, G.: CatLog: A Categorial Parser/Theorem-Prover. In: LACL 2012 System Demonstrations. pp. 13–16. Logical Aspects of Computational Linguistics 2012, Nantes (2012)

[10] Morrill, G., Valentín, O.: Computational Coverage of TLG: Nonlinearity. In: Kanazawa, M., Moss, L., de Paiva, V. (eds.) Proceedings of NLCS'15. Third Workshop on Natural Language and Computer Science. EPiC, vol. 32, pp. 51–63. Kyoto (2015), Workshop affiliated with Automata, Languages and Programming (ICALP) and Logic in Computer Science (LICS)

[11] Morrill, G., Valentín, O.: On the logic of expansion in natural language. In: Amblard, M., de Groote, P., Pogodalla, S., Retoré, C. (eds.) Logical Aspects of Computational Linguistics. Celebrating 20 Years of LACL (1996–2016): 9th International Conference, LACL 2016, Nancy, France, December 5-7, 2016, Proceedings, pp. 228–246. Springer Berlin, Heidelberg (2016), http://dx.doi.org/10.1007/978-3-662-53826-5_14

[12] Morrill, G., Valentín, O., Fadda, M.: The Displacement Calculus. Journal of Logic, Language and Information **20**(1), 1–48 (2011). https://doi.org/10.1007/s10849-010-9129-2

[13] Morrill, G.V.: Type Logical Grammar: Categorial Logic of Signs. Kluwer Academic Publishers, Dordrecht (1994)

[14] Okada, M.: Phase semantics cut-elimination and normalization proofs of first- and higher-order linear logic. Theoretical Computer Science **227**(1–2), 333–396 (September 1999)

The proviso problem from a proof-theoretic perspective

Yukiko Yana[1], Koji Mineshima[2], and Daisuke Bekki[3]

[1] Ochanomizu University yana.yukiko@is.ocha.ac.jp
[2] Keio University minesima@abelard.flet.keio.ac.jp
[3] Ochanomizu University bekki@is.ocha.ac.jp

Abstract. The proviso problem (Geurts 1999) is the problem of predicting the presupposition of a complex sentence while distinguishing between conditional and unconditional presuppositions arising from sentences with the same syntactic structures. This paper addresses the proviso problem from a viewpoint of proof-theoretic semantics, in comparison with previous approaches based on a model-theoretic conception of semantics, particularly satisfaction theory and discourse representation theory. More specifically, we build on Dependent Type Semantics (DTS), a proof-theoretic natural language semantics, and show that it provides proper predictions for a wide range of presuppositional phenomena concerning the proviso problem. We argue that the notion of context and judgment in DTS can provide a proper basis for formulating the presuppositions arising from the proviso problem.

Keywords: The proviso problem, Presupposition, Dependent types

1 Introduction

Theories of presuppositions, including satisfaction theory [10], make weaker predictions about what certain sentences presuppose than they do. For example, satisfaction theory predicts that the conditional sentence in (26a) presupposes (26b), whereas it actually presupposes (26c).

(1) a. If Theo hates sonnets, so does his wife.
b. If Theo hates sonnets, Theo has a wife.
c. Theo has a wife.

The proviso problem refers to the problem that the predictions made by a theory are weaker than the actual presupposition under specific syntactic constructions such as conditionals and coordination. This problem was pointed out by Geurts [7, 8] and has been widely discussed since then.

We first investigate approaches in satisfaction theory that try to solve this problem in the direction of strengthening the weak (conditional) presupposition as in (26b) and point out their problems. We also examine Discourse Representation Theory (DRT) [17, 8], arguing that in DRT the proviso problem behaves in

the opposite way to how it behaves in satisfaction theory; DRT fails to derive fully correct predictions regarding the so-called conditional presupposition and semi-conditional presupposition.

Given this background, we introduce the framework of Dependent Type Semantics (DTS) [3], a proof-theoretic natural language semantics, and argue that it provides a new perspective on the proviso problem that arises in the other approaches. By handling presuppositions from the proof-theoretic perspective in DTS, we show that the proviso problem can be regarded not as a problem specific to presuppositions but as an instance of more-general problems of constructing a proof in a context. We argue that the proviso problem does not arise in DTS in the way it does in satisfaction theory because of its proof-theoretic treatment of presuppositions. We also discuss how the proviso problem arises in satisfaction theory by focusing on the distinction between propositions and judgments.

The paper is structured as follows. In Section 2, we provide an overview of the proviso problem and how it is approached by two major theories of presupposition: satisfaction theory and DRT. In Section 3, we introduce the framework of DTS with a focus on the treatment of presuppositions in a proof-theoretical setting. Section 4 is devoted to the analysis of the proviso problem in DTS. We provide an analysis of various constructions concerning the proviso problem and discuss the differences from the other approaches.

2 The proviso problem: background

2.1 Satisfaction theory

We begin with a brief exposition of how the proviso problem arises under satisfaction theory. This provides a necessary background for our subsequent discussion. Satisfaction theory goes back to Heim [10], who developed a theory of presupposition building on the notion of *context change potential* (CCP). A context c is identified with a set of possible worlds, which represents the common ground of the conversation [21]. The CCP of a declarative sentence S is identified with a partial function that takes an input context c and returns an updated context c'. Then the presuppositions of S are taken as the definedness condition of the CCP. Let us illustrate these basic ideas with a simple propositional language consisting of $\neg$, $\wedge$, and $\rightarrow$. We use A_B for representing a sentence that asserts A and presupposes B. Given a model (W, I), where W is a (non-empty) set of worlds and I an evaluation function that assigns a set of worlds to each atomic formula p, that is, $I(p) \subseteq W$, we can define the CCP of any formula A along the following set of rules. Here, we write the CCP of a formula A as $[A]$ and the result of applying $[A]$ to a context c as $c[A]$. We say a context c *satisfies* A if $c \subseteq c[A]$ holds, that is, the updating of c with A does not add any new information.

1. If p is atomic, $c[p]$ is always defined and $c[p] = c \cap I(p)$
2. $c[\neg A] = c - c[A]$
3. $c[A \wedge B] = c[A][B]$
4. $c[A \rightarrow B] = c - (c[A] - c[A][B])$

5. $c[A_B] = c[A]$ if c satisfies B; undefined otherwise.

The effect of updating a context c with a simple sentence S is to take the intersection of c with the set of possible worlds in which S is true. The CCP of conditional $A \rightarrow B$ can also be derived by defining $A \rightarrow B$ as $\neg(A \wedge \neg B)$. Crucially, it follows from the definition of CCPs that conjunction and conditional have the same definedness condition; that is, $c[A \wedge B]$ and $c[A \rightarrow B]$ are defined if and only if both $c[A]$ and $c[A][B]$ are defined.

In satisfaction theory, we say A *presupposes* B if c satisfies B for any context c where $c[A]$ is defined. Thus, we can say that in satisfaction theory, what is presupposed is a *proposition*, taken as a set of worlds. It can be easily seen that both $p \wedge q_p$ and $p \rightarrow q_p$ are always defined and thus presuppose nothing; for, in the case of conjunction, $c[p \wedge q_p] = c[p][q_p]$ for any context c by definition and $c[p][q_p]$ is always defined since $c[p] \subseteq c[p]$. This correctly captures the fact that the presuppositions are filtered out in the following sentences. Here p represents "Theo has a wife" and q represents "Theo's wife hates sonnets".

(2) a. Theo has a wife and his wife hates sonnets. $p \wedge q_p$
b. If Theo has a wife, his wife hates sonnets. $p \rightarrow q_p$

However, in the case of the sentence (26a), repeated below as (28b), and the corresponding conjunctive sentence (28a), satisfaction theory wrongly predicts that these sentences have a conditional presupposition in (29a), rather than an unconditional one in (29b).[1] Here we use r to represent "Theo hates sonnets".

(3) a. Theo hates sonnets and so does his wife. $r \wedge q_p$
b. If Theo hates sonnets, so does his wife. $r \rightarrow q_p$

(4) a. If Theo hates sonnets, Theo has a wife. $r \rightarrow p$
b. Theo has a wife. p

Note that conditional presuppositions can arise in certain contexts [2, 8, 18]. For example, let us look at the example in (30a), taken from Schlenker [18]. This, like (26a), is a conditional sentence whose consequence has a presupposition triggered by the factive verb *know*. However, the presupposition of (30a) is the conditional one as in (30b), not the unconditional one as in (30c).

(5) a. If this applicant is 64 years old, he knows that we cannot hire him.
b. If this applicant is 64 years old, we cannot hire him.
c. We cannot hire this applicant.

Satisfaction theory can provide correct predictions for such cases. Thus the problem for satisfaction theory is to explain how some sentences have a conditional presupposition while others have an unconditional one.

[1] In the case of (28a), this can be seen as follows. Let c be an arbitrary context and suppose $c[r \wedge q_p]$ is defined. Then $c[r][q_p]$ is defined, so $c[r]$ satisfies p. So c satisfies $r \rightarrow p$. Hence by definition, $r \wedge q_p$ presupposes $r \rightarrow p$.

Various theories that set out to improve satisfaction theory have been proposed in the literature. The basic idea, going back to Karttunen and Peters [11], is to keep the presupposition of (26a) to be the conditional proposition $r \to q$ and introduce some additional inference to strengthen $r \to q$ to q. As already pointed out by Geurts [7, 8], however, this approach has at least two problems. First, (31a) presupposes the conditional proposition in (31b), but in this case, the strengthening inference should not work to strengthen it to (31c).

(6) a. Susan knows that if Theo hates sonnets, he has a wife.
b. If Theo hates sonnets, he has a wife.
c. Theo has a wife.

Second, examples such as (32a) give rise to the "semi-conditional" presupposition as in (32b), rather than the "fully conditional" presuppositions as in (32c).

(7) a. If John is a scuba diver and he wants to impress his girlfriend, he'll bring his wetsuit. $p \wedge q \to r_s$
b. If John is a scuba diver, he has a wetsuit. $p \to s$
c. If John is a scuba diver and he wants to impress his girlfriend, he has a wetsuit. $p \wedge q \to s$

To address these issues, some additional mechanisms have been proposed to repair satisfaction theory, such as scalar alternatives [20], skeltal alternatives [19], and the Stalnakerian semantics of conditionals and implicatures [4]. These moves can make the resulting theory much more complicated. Furthermore, these theories tend to crucially rely on the notion of *relevance*; but there is a notable counter-example to a relevance-based satisfaction theory. Let us take a look at the example sentence in (3), pointed out by Mandelkern [13].

(8) **Context**: It is common ground that Smith has gone missing, and we don't know whether he is still alive. A detective enters and says:
If the butler's clothes contain traces of Smith's blood, then it was the butler who killed Smith.

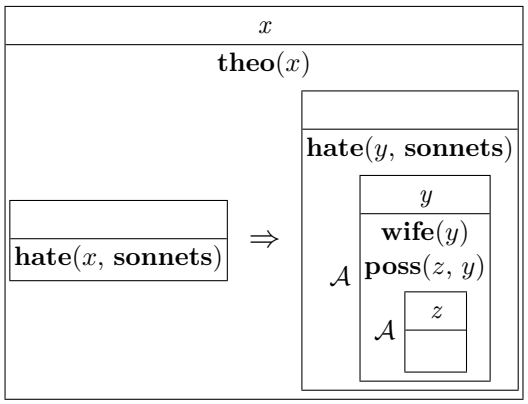
Let p be "the butler's clothes contain traces of Smith's blood", q "it was the butler who killed Smith", and r "someone killed Smith". Then the sentence in (33) has the form $p \to q_r$. According to the relevance-based satisfaction theory, a conditional presupposition is triggered if the antecedent is relevant to the presupposed material (in this case, r). In the above situation, there is clearly a strong relevance of p to r. Thus, it predicts that the presupposition of (33) is the conditional one, namely "If the butler's clothes contain traces of Smith's blood, someone killed Smith" ($p \to r$). However, the empirical presupposition of (33) is the unconditional one, "Someone killed Smith" (r). This suggests that the notion of relevance does not fully provide the proper solution to the proviso problem.

2.2 Discourse Representation Theory

Geurts [7, 8] proposes that a theory based on Discourse Representation Theory (DRT) can derive correct predictions for (26). DRT introduces a level of semantic

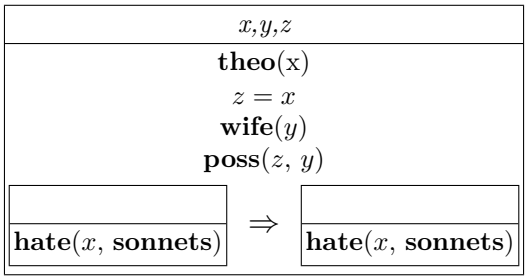
representation called Discourse Representation Structures (DRSs), which are assigned to sentences. For example, the sentence (26a) can be assigned the following DRS.

(9)



DRT defines an anaphoric-DRS as a DRS marked with $\mathcal{A}$ (referred to as $\mathcal{A}$-structure) as in (34) and resolves the presupposition by associating the discourse referent with the discourse referent already introduced. In the case of (34), however, there is no discourse referent that can be identified with y (while z can be identified with x). In such a case, the anaphoric-DRS is resolved at the top-level DRS via *global accommodation*. The resulting DRS is (35).

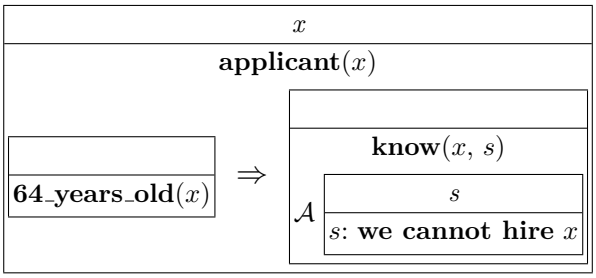
(10)



In this case, there are two conditions, **wife**(y) and **poss**(z, y), that are added to the top-level DRS via global accommodation. This means that DRT predicts the presupposition of (26) as a strong (unconditional) presupposition.

Although DRT can predict unconditional presupposition via global accommodation, it has a difficulty in predicting conditional presupposition. As an example, let us try to analyze (30a) with DRT.

(11)



In this case, there is no reason for it to be resolved by the antecedent DRS. Thus, the anaphoric-DRS can be resolved either (i) by the top-level DRS or (ii) by the local DRS. In the case of (i), then, the variable s in $\mathbf{know}(x, s)$ refers to the discourse referent associated with the condition added to the top-level DRS via global accommodation. Thus, DRT predicts an unconditional presupposition, not the correct unconditional one in (30c). On the other hand, option (ii) is also problematic (cf. [18]). If the anaphoric-DRS is accommodated locally, it is treated as a mere entailment. However, the inference in question cannot be an entailment because it survives in a question such as (37).

(12) If this applicant is 64 years old, does he know that we cannot hire him?

See [18] for other examples involving conditional presuppositions that are problematic for DRT.

It is also not clear how to derive semi-conditional presupposition as in (32). If the presupposed content (anaphoric-DRS) is copied in the consequence of a conditional via local accommodation, we can obtain a kind of fully conditional presupposition as in (32c) as an entailment, because all the antecedents stay in the same position after local accommodation. However, the desired prediction is the semi-conditional presupposition as in (32b).

Geurts [7, 8] criticized satisfaction theory for only being able to predict conditional presuppositions, but DRT can only predict unconditional presuppositions. The proviso problem was found to behave in the exact opposite way and still be a problem in DRT.

3 Presupposition from a proof-theoretic perspective

This section describes the framework of DTS, focusing on the mechanism by which DTS treats presuppositions. In contrast to the model-theoretic approaches we have seen so far, presuppositions are analyzed in DTS in a proof-theoretic way, treated as *judgments* that trigger the proof-search process to resolve presuppositions in contexts.

3.1 Dependent Type Semantics

DTS [3] is a proof-theoretic natural language semantics based on *dependent types* [14]. Dependent types extend simply-typed λ-calculus so that types can

depend on terms. The attempts to use dependent types in natural language semantics began with the work by Ranta [15]. Recent work includes Type Theory with Records [6], Modern Type Theory [12, 5], and applications to generalized quantifiers [9], among others. The main characteristic of DTS is that it provides a compositional analysis of anaphora and presupposition using dependent types (see also [22, 23]).

In DTS, the two essential types are Π-type and Σ-type.[2] Π-types generalize function types in simply-typed λ-calculus. We write $(x : A) \to B(x)$ for Π-types.[3] When a term f of type $(x : A) \to B(x)$ takes a term a of type A as an argument, the term $f(a)$ is of type $B(a)$. The basis of Σ-type is the product type in simple-typed λ-calculus. We write $\begin{bmatrix} x : A \\ B \end{bmatrix}$ for a Σ-type.[4] A term (t, u) of type $\begin{bmatrix} x : A \\ B(x) \end{bmatrix}$ is a pair. When t is of type A and u is of type $B(t)$, $\begin{bmatrix} x : A \\ B(x) \end{bmatrix}$ is a type of pair (t, u). The projection functions π_1 and π_2 extract a term from a pair having a Σ-type: we write $\pi_1(t, u) = t$ and $\pi_2(t, u) = u$. Fig. 1 shows the inference rules for Π-types and Σ-types.

According to the *Curry–Howard correspondence*, types and propositions can be used interchangeably. This means that types can be read as propositions, so that, say, the type $\mathbf{scuba\_diver}(x)$ represents the proposition that *x is a scuba diver.*[5] The judgment $w : \mathbf{scuba\_diver}(x)$ means that w is a *proof* for the proposition that *x is a scuba diver*, thus w is called a *proof term.* Again, under the Curry–Howard correspondence, Π-type corresponds to universal quantification and Σ-type corresponds to existential quantification. For instance, in DTS the semantic representation of (38a) is described as (38b) using a Σ-type.

(13) a. Theo has a wife.

b. $\begin{bmatrix} x : \mathbf{entity} \\ \mathbf{of\_wife}(x, theo) \end{bmatrix}$

[2] DTS employs other types as well: enumeration types (including the empty type $\bot$, via which the negation of A is defined as $A \to \bot$), disjoint union types, intensional equality types, natural number types, and universes. In this paper, we focus on the presentation of Π-types and Σ-types for the sake of space.

[3] When the variable x does not occur free in B, $(x : A) \to B(x)$ is equivalent to $A \to B$.

[4] When the variable x does not occur free in B, $\begin{bmatrix} x : A \\ B(x) \end{bmatrix}$ is equivalent to $A \times B$.

[5] There has been a controversy in type-theoretic semantics regarding whether the semantic representations for common nouns are *types* [1, 5, 15, 16] or *predicates* [6, 3]. The meaning of "A man walked" is represented as $\begin{bmatrix} x : \mathbf{man} \\ \mathbf{walk}(x) \end{bmatrix}$ under the former (=nouns-as-types) view, while it is represented as $\begin{bmatrix} u : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{man}(x) \end{bmatrix} \\ \mathbf{walk}(x) \end{bmatrix}$ under the latter (=nouns-as-predicates) view. In this paper, we assume the nouns-as-predicates view, following [3], which gives the detailed comparision between the two views.

$$\frac{\begin{array}{c}\overline{x : A}\ i \\ \vdots \\ A : \mathsf{type} \qquad B : \mathsf{type}\end{array}}{(x : A) \to B : \mathsf{type}}\ (\Pi\mathrm{F}), i \qquad \frac{\begin{array}{c}\overline{x : A}\ i \\ \vdots \\ A : \mathsf{type} \qquad M : B\end{array}}{\lambda x.M : (x : A) \to B}\ (\Pi\mathrm{I}), i$$

$$\frac{M : (x : A) \to B \qquad N : A}{MN : B[N/x]}\ (\Pi\mathrm{E})$$

$$\frac{\begin{array}{c}\overline{x : A}\ i \\ \vdots \\ A : \mathsf{type} \qquad B : \mathsf{type}\end{array}}{\begin{bmatrix} x : A \\ B \end{bmatrix} : \mathsf{type}}\ (\Sigma\mathrm{F}), i \qquad \frac{M : A \qquad N : B[M/x]}{(M, N) : \begin{bmatrix} x : A \\ B \end{bmatrix}}\ (\Sigma\mathrm{I})$$

$$\frac{M : \begin{bmatrix} x : A \\ B \end{bmatrix}}{\pi_1 M : A}\ (\Sigma\mathrm{E}) \qquad \frac{M : \begin{bmatrix} x : A \\ B \end{bmatrix}}{\pi_2 M : B[\pi_1 M/x]}\ (\Sigma\mathrm{E})$$

Fig. 1. Formation rules (ΠF, ΣF), Introduction rules (ΠI, ΣI), and Elimination rules (ΠE, ΣE) for Π-types and Σ-types.

3.2 Presuppositions in DTS

In DTS, terms with incomplete information such as those triggered by presupposition and anaphora are represented using *underspecified* terms, written as @. For example, *Theo's wife* in the sentence in (39a) triggers the presupposition that *Theo has a wife*. This can be captured by the semantic representation in (39b), where the underspecified term @ is annotated with a Σ-type for the proposition that *Theo has a wife*.

(14) a. Theo's wife loves Theo.

b. $\mathbf{love}\left(\pi_1\left(@ : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{of\_wife}(x, theo) \end{bmatrix}\right), theo\right)$

Given the semantic representation containing an underspecified term @, such as (39b), type checking is launched to construct a proof term with the type annotated to the underspecified term. The type checking is triggered by the following rule (henceforth (@)-rule).

$$\frac{A : \mathsf{type} \quad A\ true}{(@ : A) : A}\ (@)$$

Here the judgment $A\ true$ means that there exists a proof term of type A. If a term of type A is constructed, the underspecified term @ is replaced with the constructed term.

Type checking runs under the *global context*, consisting of a set of judgments, which represents the information that can be used unconditionally in the proof, such as signatures specifying the type of each expression (e.g., **love** : **entity** → **entity** → type and *theo* : **entity**), previously uttered information, and world knowledge.

It is crucial to see that some information may be added to the context during a proof by inference rules such as formation rules for Π-type and Σ-type in Fig. 1, repeated here.

$$\frac{\begin{array}{c}\overline{x : A}\ i \\ \vdots \\ A : \mathsf{type} \qquad B : \mathsf{type}\end{array}}{(x : A) \to B : \mathsf{type}}\ (\Pi\mathrm{F}), i \qquad \frac{\begin{array}{c}\overline{x : A}\ i \\ \vdots \\ A : \mathsf{type} \qquad B : \mathsf{type}\end{array}}{\begin{bmatrix} x : A \\ B \end{bmatrix} : \mathsf{type}}\ (\Sigma\mathrm{F}), i$$

Here, the judgment $x : A$ can be used locally in a sub-derivation to prove $B : \mathsf{type}$. The locally added information that can be used for type checking in addition to the global context is called a *local context*.

Let us illustrate how the type checking works by the example in (39). For the semantic representation with an underspecified term in (39b), the type checking runs in the following way.

(15)
$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\left[\begin{smallmatrix} x : \mathbf{entity} \\ \mathbf{of\_wife}(x, theo) \end{smallmatrix}\right] : \mathsf{type}} \quad \left[\begin{smallmatrix} x : \mathbf{entity} \\ \mathbf{of\_wife}(x, theo) \end{smallmatrix}\right] \mathit{true}}\ (@)}{\left(@_1 : \left[\begin{smallmatrix} x : \mathbf{entity} \\ \mathbf{of\_wife}(x, theo) \end{smallmatrix}\right]\right) : \left[\begin{smallmatrix} x : \mathbf{entity} \\ \mathbf{of\_wife}(x, theo) \end{smallmatrix}\right]}\ (\Sigma\mathrm{E})}{\pi_1\left(@_1 : \left[\begin{smallmatrix} x : \mathbf{entity} \\ \mathbf{of\_wife}(x, theo) \end{smallmatrix}\right]\right) : \mathbf{entity}}\ (\Sigma\mathrm{E})}{\mathbf{love}\left(\pi_1\left(@_1 : \left[\begin{smallmatrix} x : \mathbf{entity} \\ \mathbf{of\_wife}(x, theo) \end{smallmatrix}\right]\right)\right) : \mathbf{entity} \to \mathsf{type}} \qquad \dfrac{}{theo : \mathbf{entity}}\ (\mathrm{CON)}}{\mathbf{love}\left(\pi_1\left(@ : \left[\begin{smallmatrix} x : \mathbf{entity} \\ \mathbf{of\_wife}(x, theo) \end{smallmatrix}\right]\right), theo\right) : \mathsf{type}}\ (\Pi\mathrm{E})$$

The goal is to prove that the semantic representation in (39b) is a *type* (i.e., proposition) under the global context, which means that the sentence is semantically felicitous in a given context. Thus, the goal of the type checking is to prove the following judgment, where the global context is made explicit by $\mathcal{C}$.

(16) $\mathcal{C} \vdash \mathbf{love}\left(\pi_1\left(@ : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{of\_wife}(x, theo) \end{bmatrix}\right), theo\right) : \mathsf{type}$

In the type checking tree in (40), we use (CON) when the relevant judgment is in the global context. The crucial part is the judgment $\begin{bmatrix} x : \mathbf{entity} \\ \mathbf{of\_wife}(x, theo) \end{bmatrix}$ *true*, which appears in the type checking tree by (@)-rule. This judgment means that there exists a proof for the proposition that *Theo has a wife*. At this stage, a process of proof search is launched to find a proof term of the type annotated to the underspecified term under the global context. Thus, the judgment to be proved can be written as follows:

(17) $\mathcal{C} \vdash \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{of\_wife}(x, theo) \end{bmatrix} true$

Proof search can construct a proof term using the global context together with a local context introduced by formation rules for Π-type and Σ-type. In this case, let us assume that *Theo has a wife* is already in the global context $\mathcal{C}$. Thus, we have a judgment $u : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{of\_wife}(x, theo) \end{bmatrix}$ in $\mathcal{C}$. Then the underspecified term @ can be identified with the proof term u and successfully eliminated after the proof search. By replacing @ with u, we obtain the final semantic representation for (39a) as follows:

(18) $\mathbf{love}(\pi_1(u), theo)$

The process of obtaining (43) from (39b) can be summarized as follows.

1. Type checking tries to ensure that the semantic representation in (39b) is a type, that is, it is semantically felicitous under the global context.
2. In the type checking process, proof search runs to obtain an appropriate proof term from the global context and possibly the local contexts, as indicated in (42).[6]
3. If a proof term is constructed, it is replaced with the underspecified term @ and the final semantic representation in (43) is derived.

Through these steps, DTS resolves the underspecified term which encodes a presupposition trigger. Given this procedure, the presupposition of a sentence is regarded as a *judgment*, such as (42), in DTS. In other words, the presupposition in DTS is the requirement to find/construct a proof term for the proposition associated with an underspecified term under a given context. It is worth emphasizing that a presupposition in DTS is not a *proposition* as is the case in satisfaction theory. This difference would be crucial when considering the proviso problem from a proof-theoretic perspective of DTS.

4 The proviso problem in DTS

In this section, we will discuss how DTS analyzes the proviso problem from the proof-theoretic perspective, in comparison with satisfaction theory and DRT. We claim that the proviso problem arises from the perspective from which satisfaction theory views presuppositions. Following the proof-theoretic way of handling presupposition in DTS as described in the previous section, the problem can be seen as an instance of more-general problems of selecting premises in proof search, rather than a problem specific to presuppositions.

[6] In most cases, we obtain several proof terms for a given underspecified term. Those proof terms correspond to different readings, namely, different antecedents/resolutions for the anaphora/presupposition in question.

4.1 Presupposition: proposition or judgment

In satisfaction theory, as explained in Section 2.1, what a sentence presupposes is a *proposition*. For example, satisfaction theory predicts that the presupposition of (26a) is a conditional proposition of the form $r \to p$, which can be expressed as *If Theo hates sonnets, Theo has a wife*. This requires satisfaction theory to strengthen the conditional proposition $r \to p$ to the unconditional p.

In DTS, on the other hand, what a sentence presupposes is a *judgment*, as described in the previous section. DTS predicts the presupposition of the sentence (26a) in the following way. First, the semantic representation (44), which contains an underspecified term @ triggered by *Theo's wife*, is assigned to the sentence (26a).

$$(19)\quad \left(v : \begin{bmatrix} w : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{sonnet}(x) \end{bmatrix} \\ \mathbf{hate}(theo, \pi_1 w) \end{bmatrix}\right) \to \begin{bmatrix} w : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{sonnet}(x) \end{bmatrix} \\ \mathbf{hate}\left(\pi_1\left(@ : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{of\_wife}(x, theo) \end{bmatrix}\right), \pi_1 w\right) \end{bmatrix}$$

Then the type checking is launched to prove that the semantic representation in (44) is of type `type`, which yields the type checking diagram (45).

$$(20)\quad \dfrac{\left(v : \begin{bmatrix} w : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{sonnet}(x) \end{bmatrix} \\ \mathbf{hate}(theo, \pi_1 w) \end{bmatrix}\right) \to \begin{bmatrix} w : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{sonnet}(x) \end{bmatrix} \\ \mathbf{hate}\left(\pi_1\left(@ : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{of\_wife}(x, theo) \end{bmatrix}\right), \pi_1 w\right) \end{bmatrix} : \mathsf{type}}{(\Pi\mathrm{F}),\, 1} \qquad \dfrac{\begin{bmatrix} w : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{sonnet}(x) \end{bmatrix} \\ \mathbf{hate}(theo, \pi_1 w) \end{bmatrix} : \mathsf{type}}{\vdots} \qquad \dfrac{\begin{bmatrix} w : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{sonnet}(x) \end{bmatrix} \\ \mathbf{hate}\left(\pi_1\left(@ : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{of\_wife}(x, theo) \end{bmatrix}\right), \pi_1 w\right) \end{bmatrix} : \mathsf{type}}{\vdots}$$

$$\dfrac{\left(@ : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{of\_wife}(x, theo) \end{bmatrix}\right) : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{of\_wife}(x, theo) \end{bmatrix}}{\vdots} (@) \qquad \dfrac{\begin{bmatrix} x : \mathbf{entity} \\ \mathbf{of\_wife}(x, theo) \end{bmatrix} true}{\vdots}$$

$$\dfrac{v : \begin{bmatrix} w : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{sonnet}(x) \end{bmatrix} \\ \mathbf{hate}(theo, \pi_1 w) \end{bmatrix}}{\vdots\ \mathcal{D}_1}\ 1$$

This diagram leaves a gap indicated by $\mathcal{D}_1$, which needs to be filled by constructing a proof for $\begin{bmatrix} x : \mathbf{entity} \\ \mathbf{of\_wife}(x, theo) \end{bmatrix}$ under a given context. Crucially, the formation rule for Σ-type allows us to use the local context associated with the antecedent of the sentence to construct the proof $\mathcal{D}_1$. Thus, given a global context $\mathcal{C}$ (which plays a role of world knowledge in DTS), what needs to hold is the following judgment:

$$(21)\quad \mathcal{C}, v : \begin{bmatrix} w : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{sonnet}(x) \end{bmatrix} \\ \mathbf{hate}(theo, \pi_1 w) \end{bmatrix} \vdash \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{of\_wife}(x, theo) \end{bmatrix} true$$

This way, DTS predicts that the sentence (26a) presupposes that the judgment (46) holds. This prediction has a different look from the "unconditional presupposition" in satisfaction theory, namely *Theo has a wife*, in that the judgment (46) has a context. The key point in interpreting the judgment (46) as a presupposition is that the use of local context (=*Theo hates sonnets*) is *optional* for constructing a proof term for *Theo has a wife*. This is derived from a general interpretation of judgments in proof theory where $\Gamma \vdash \phi$ means that there exists a proof diagram from Γ to ϕ, but it does not mean that every proposition in Γ must be used in the diagram. In the case of (46), the proof of *Theo has a wife* is constructed without using the local context *Theo hates sonnets*, and this orientation of proofs and world knowledge represents an unconditional presupposition. In other words, the linguistic intuition that the sentence (26a) has an unconditional presupposition corresponds to the independence of the proofs of *Theo has a wife* and the local context.

Then, how does DTS determine whether a sentence of the form (26a) has a conditional or unconditional presupposition? It is done by two steps: firstly, the grammar derives a presupposed judgment (if any), and secondly, the proof search constructs a required proof by using the global and local contexts. The presupposition is conditional if the local contexts are used in the latter process, and unconditional otherwise. So, the distinction between conditional and unconditional presuppositions is not a matter of compositional semantics, but a matter of proof constructions, which are the acts belonging to more pragmatic and/or cognitive domains. Technically, the problem of which premise is necessary to prove a given conclusion in a proof search is known as the *premise selection*, which is a problem typically arising when we conduct inferences in the real world.

In fact, the conditional/unconditional distinction between presuppositions is highly dependent on the context, including previous utterances and world knowledge. For example, if the world is such that all people who hate sonnets are married (—†), then the sentence (26a) has a conditional presupposition. In DTS, this situation is also naturally explained: the global context should contain a proof term representing (†) in this case, which, together with the local context *Theo hates sonnets*, entails that *Theo has a wife*. Now there is a proof of *Theo has a wife* that *uses* the local context, which means that the presupposition is conditional. This explanation is available because of the optional nature of the premises in the context.

However, in satisfaction theory where presupposition is a proposition, the local context either appears as an antecedent of the presupposed implication or does not appear at all. Because of this dictonomy, satisfaction theory has a difficulty in capturing the context-dependent nature of conditional/unconditional distinctions between presuppositions.

Unlike satisfaction theory, DRT has a general procedure for anaphora resolution/presupposition binding via anaphoric-DRS (or $\mathcal{A}$-structure). However, as pointed out in Section (2.2), DRT cannot resolve anaphoric-DRS in a conditional manner, because DRT lacks the notion of the local context. Thus, the interpretation of the presupposition of a conditional sentence remains problematic in DRT.

On the other hand, DTS can resolve anaphora and presupposition in the unified manner, via the type-checking and proof-search, which refers to the notion of global and local contexts in a essential way. Therefore, unlike DRT, DTS can derive conditional presuppositions without problems. For a detailed comparison of anaphora resolutions in DTS and DRT, see Yana et al. [24].

4.2 Conditional and semi-conditional presuppositions in DTS

In this section, we discuss how DTS deals with constructions that give rise to conditional and semi-conditional presuppositions as discussed in Section 2.

Let us start with the sentence in (30a), an instance of conditional presupposition. For this sentence, the semantic representation, the type checking tree, and the judgment to be proved in proof search for resolving the presupposition are shown in (47a), (47b), and (47c), respectively.[7]

(22) a. $\left(u : \begin{bmatrix} y : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{applicant}(x) \end{bmatrix} \\ \mathbf{64\_years\_old}(\pi_1 y) \end{bmatrix}\right) \rightarrow \mathbf{know}(\pi_1\pi_1 u, @ : \mathbf{we\_cannot\_hire}(\pi_1\pi_1 u))$

b.
$$\dfrac{\dfrac{\vdots}{\begin{bmatrix} y : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{applicant}(x) \end{bmatrix} \\ \mathbf{64\_years\_old}(\pi_1 y) \end{bmatrix} : \mathsf{type}} \quad \dfrac{\dfrac{\dfrac{}{u : \begin{bmatrix} y : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{applicant}(x) \end{bmatrix} \\ \mathbf{64\_years\_old}(\pi_1 y) \end{bmatrix}}\,1 \quad \vdots}{\mathbf{we\_cannot\_hire}(\pi_1\pi_1 u)\ \mathit{true}}}{(@ : \mathbf{we\_cannot\_hire}(\pi_1\pi_1 u)) : \mathbf{we\_cannot\_hire}(\pi_1\pi_1 u)}\,(@) \quad \vdots}{\mathbf{know}(\pi_1\pi_1 u, @ : (\mathbf{we\_cannot\_hire}(\pi_1\pi_1 u))) : \mathsf{type}}}{\left(u : \begin{bmatrix} y : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{applicant}(x) \end{bmatrix} \\ \mathbf{64\_years\_old}(\pi_1 y) \end{bmatrix}\right) \rightarrow \mathbf{know}(\pi_1\pi_1 u, @ : (\mathbf{we\_cannot\_hire}(\pi_1\pi_1 u))) : \mathsf{type}}\,(\Pi\ \mathrm{F}),\,1$$

c. $\mathcal{C}, u : \begin{bmatrix} y : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{applicant}(x) \end{bmatrix} \\ \mathbf{64\_years\_old}(\pi_1 y) \end{bmatrix} \vdash \mathbf{we\_cannot\_hire}(\pi_1\pi_1 u)\ \mathit{true}$

Let us assume that the global context contains the axiom for the missing premise, *If an applicant is 60 years old and over, then we cannot hire the applicant*, whose proof term is f. Then, the proof term constructed for the judgment in (47c) is $f(u)$, which applies the term u in the local context to the axiom f in the global context. In contrast to the case of (26), (47c) uses the term u, which is incorporated as a local context by the formation rule for Π-type, as an argument in the proof search. This time, the proof search uses the antecedent of

[7] The semantic representation and derivations are simplified in the following way: we assume that the anaphora in the consequent, *he*, has been resolved. Also, *know* is a factive verb, not just a binary predicate as it should be, with @ attached to the second argument. This simplification does not pose a problem when focusing on the proviso problem in this setting. For a detailed analysis of *know* in DTS, see Tanaka et al. [22].

the conditional sentence in the local context to construct the proof term. This corresponds to the "conditional (weak) presupposition" in satisfaction theory.

Second, let us consider the sentence that give rise to "semi-conditional presuppositions" in (32a). The type checking runs as follows.

(23)
$$\dfrac{\dfrac{\vdots}{u : \begin{bmatrix}\mathbf{scuba\_diver}(j)\\ \mathbf{want\_to\_impress}(j)\end{bmatrix} : \mathsf{type}} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{u : \begin{bmatrix}\mathbf{scuba\_diver}(j)\\ \mathbf{want\_to\_impress}(j)\end{bmatrix}}\,1}{\vdots\ \mathcal{D}}}{\begin{bmatrix} w : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{wetsuit}(x)\end{bmatrix} \\ \mathbf{have}(j, \pi_1 w)\end{bmatrix}\ true}}{\left(@ : \begin{bmatrix} w : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{wetsuit}(x)\end{bmatrix} \\ \mathbf{have}(j, \pi_1 w)\end{bmatrix}\right) : \begin{bmatrix} w : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{wetsuit}(x)\end{bmatrix} \\ \mathbf{have}(j, \pi_1 w)\end{bmatrix}}\,(@)}{\vdots}}{\mathbf{bring}\left(j, @ : \begin{bmatrix} w : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{wetsuit}(x)\end{bmatrix} \\ \mathbf{have}(j, \pi_1 w)\end{bmatrix}\right) : \mathsf{type}}}{\left(u : \begin{bmatrix}\mathbf{scuba\_diver}(j)\\ \mathbf{want\_to\_impress}(j)\end{bmatrix}\right) \to \mathbf{bring}\left(j, @ : \begin{bmatrix} w : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{wetsuit}(x)\end{bmatrix} \\ \mathbf{have}(j, \pi_1 w)\end{bmatrix}\right) : \mathsf{type}}\,(\Pi\ \mathrm{F}),\ 1$$

As in the other examples, we first use the formation rule for Π-type to incorporate the antecedent of the sentence into the local context. By making explicit the context, the judgment to be proved for filling the gap in $\mathcal{D}$ in this derivation can be written as follows:

(24) $$\mathcal{C}, u : \begin{bmatrix}\mathbf{scuba\_diver}(j)\\ \mathbf{want\_to\_impress}(j)\end{bmatrix} \vdash \begin{bmatrix} w : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{wetsuit}(x)\end{bmatrix} \\ \mathbf{have}(j, \pi_1 w)\end{bmatrix}\ true$$

In this case, let us assume that axiom $g : \left(u : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{scuba\_diver}(x)\end{bmatrix}\right) \to \begin{bmatrix} w : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{wetsuit}(x)\end{bmatrix} \\ \mathbf{have}(\pi_1 u, \pi_1 w)\end{bmatrix}$ is in the global context $\mathcal{C}$. The proof term constructed in (48b) is the term $g(j, \pi_1(u))$ that applies the pair term $(j, \pi_1(u))$ to the axiom f in the global context. This means that the first element of the term u is assigned to **scuba_diver** that appears in the antecedent of the axiom f, and the second element of the term u is not used to construct the proof term. Only $\pi_1(u)$, that is, the proposition **scuba_diver**, is used in this proof term, which corresponds to the semi-conditional presupposition.

5 Conclusion

We showed how the notion of context and judgment in the proof-theoretic framework of DTS can provide a proper basis for handling presuppositions and the proviso problem. By handling presuppositions from this proof-theoretic

perspective, we showed that the proviso problem can be regarded as an instance of general problems in proof search, not as a problem specific to presupposition resolution. We argued that the proviso problem does not arise in DTS in the way it does in satisfaction theory because of its proof-theoretic treatment of presuppositions. We believe that handling presuppositions from a proof-theoretic perspective can contribute to future discussions of the proviso problem and presuppositions in general.

Bibliography

[1] Asher, N.: Lexical Meaning in Context: A Web of Words. Cambridge University Press (2011)

[2] Beaver, D.: Presupposition and Assertion in Dynamic Semantics. CSLI Publications, Stanford, CA (2001)

[3] Bekki, D., Mineshima, K.: Context-passing and underspecification in Dependent Type Semantics. In: Modern Perspectives in Type Theoretical Semantics, pp. 11–41. Springer (2017)

[4] Carballo, A.P.: Toward a dissolution of the proviso problem. In: Proceedings of MIT-France Workshop on Scalar Implicatures and Presupposition. MIT Working Papers in Linguistics (MITWPL). vol. 60, pp. 169–184 (2008)

[5] Chatzikyriakidis, S., Luo, Z.: On the interpretation of common nouns: Types versus predicates. In: Chatzikyriakidis, S., Luo, Z. (eds.) Modern Perspectives in Type-Theoretical Semantics, pp. 43–70. Springer (2017)

[6] Cooper, R.: Type theory and semantics in flux. Handbook of the Philosophy of Science **14**, 271–323 (2012)

[7] Geurts, B.: Local satisfaction guaranteed: a presupposition theory and its problems. Linguistics and Philosophy **19**, 259–294 (1996)

[8] Geurts, B.: Presuppositions and Pronouns. Elsevier, Amsterdam (1999)

[9] Grudzińska, J., Zawadowski, M.: Generalized quantifiers on dependent types: A system for anaphora. In: Chatzikyriakidis, S., Luo, Z. (eds.) Modern Perspectives in Type-Theoretical Semantics, pp. 95–131. Springer (2017)

[10] Heim, I.: On the projection problem for presuppositions. In: Barlow, M., Flickinger, D., Wescoat, M. (eds.) Proceedings of WCCFL 2, pp. 114–125. Stanford University, Stanford, CA (1983)

[11] Karttunen, L., Peters, S.: Conventional implicatures. In: Oh, C.K., Dineen, D.A. (eds.) Syntax and Semantics 11: Presupposition, pp. 1–56. Academic Press, New York (1979)

[12] Luo, Z.: Formal semantics in modern type theories with coercive subtyping. Linguistics and Philosophy **35**(6), 491–513 (2012)

[13] Mandelkern, M.: A note on the architecture of presupposition. Semantics and Pragmatics **9**(13), 1–24 (2016)

[14] Martin-Löf, P.: Intuitionistic type theory. Bibliopolis (1984)

[15] Ranta, A.: Type-theoretical grammar. Oxford University Press (1994)

[16] Retoré, C.: The Montagovian generative lexicon ΛTyn: a type theoretical framework for natural language semantics. In: Matthes, R., Schubert, A. (eds.) TYPES 2013. Leibniz International Proceedings in Informatics (LIPIcs), vol. 26, pp. 202–229. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany, Toulouse, France (Apr 2013), https://hal.archives-ouvertes.fr/hal-01009838

[17] van der Sandt, R.A.: Presupposition projection as anaphora resolution. Journal of Semantics **9**, 333–377 (1992)

[18] Schlenker, P.: DRT with local contexts. Natural Language Semantics **19**(4), 373–392 (2011)
[19] Schlenker, P.: The proviso problem: A note. Natural Language Semantics **19**(4), 395–422 (2011)
[20] Singh, R.: Formal alternatives as a solution to the proviso problem. In: Semantics and Linguistic Theory. vol. 17, pp. 264–281 (2007)
[21] Stalnaker, R.: Pragmatic presuppositions. In: Munitz, M., Unger, P. (eds.) Semantics and Philosophy, pp. 197–214. New York University Press, New York (1974)
[22] Tanaka, R., Mineshima, K., Bekki, D.: Factivity and presupposition in dependent type semantics. Journal of Language Modelling **5**(2), 385–420 (2017)
[23] Tanaka, R., Mineshima, K., Bekki, D.: Paychecks, presupposition, and dependent types. In: Proceedings of the Fifth Workshop on Natural Language and Computer Science (NLCS 2018), EasyChair Preprint no. 215 (2018)
[24] Yana, Y., Mineshima, K., Bekki, D.: Variable handling and compositionality: Comparing DRT and DTS. Journal of Logic, Language and Information **28**(2), 261–285 (2019)

Author Index