Preproceedings of the 20th Conference on Formal Grammar

Annie Foret, Glyn Morrill, Reinhard Muskens and Rainer Osswald (eds.)

August 8–9th 2015, Barcelona

Preface

FG provides a forum for the presentation of new and original research on formal grammar, mathematical linguistics and the application of formal and mathematical methods to the study of natural language. Themes of interest include, but are not limited to:

- Formal and computational phonology, morphology, syntax, semantics and pragmatics
- Model-theoretic and proof-theoretic methods in linguistics
- Logical aspects of linguistic structure
- Constraint-based and resource-sensitive approaches to grammar
- Learnability of formal grammar
- Integration of stochastic and symbolic models of grammar
- Foundational, methodological and architectural issues in grammar and linguistics
- Mathematical foundations of statistical approaches to linguistic analysis

Previous Formal Grammar meetings were held in Barcelona (1995), Prague (1996), Aix-en-Provence (1997), Saarbrücken (1998), Utrecht (1999), Helsinki (2001), Trento (2002), Vienna (2003), Nancy (2004), Edinburgh (2005), Malaga (2006), Dublin (2007), Hamburg (2008), Bordeaux (2009), Copenhagen (2010), Ljubljana (2011), Opole (2012), Düsseldorf (2013) and Tübingen (2014).

The present volume collects the papers from the 20th Conference on Formal Grammar celebrated in Barcelona in 2015. This preproceedings comprises two invited contributions, by Robin Cooper and Tim Fernando, and 9 contributed papers selected from 14 submissions.

We thank for support the local organisers of ESSLLI 2015, with which the conference was colocated.

July 2015

Annie Foret
Glyn Morrill
Reinhard Muskens
Rainer Osswald

Program Committee

Alexander Clark	King's College London, UK
Berthold Crysmann	CNRS - LLF, France
Denys Duchier	Université d'Orleans, France
Nissim Francez	Technion, Israel
Philippe de Groote	Inria Nancy, France
Laura Kallmeyer	Heinrich-Heine-Universität Düsseldorf, Germany
Makoto Kanazawa	National Institute of Informatics, Japan
Greg Kobele	University of Chicago, USA
Robert Levine	Ohio State University, USA
Wolfgang Maier	Heinrich-Heine-Universität Düsseldorf, Germany
Stefan Müller	Freie Universität Berlin, Germany
Mark-Jan Nederhof	University of St Andrews, UK
Christian Retoré	LIRMM - Université Montpellier 2, France
Manfred Sailer	Goethe University Frankfurt, Germany
Ed Stabler	UCLA, USA
Jesse Tseng	CNRS - CLLE-ERSS, France
Oriol Valentín	Universitat Politècnica de Catalunya, Spain

Program Chairs and Standing Committee

Annie Foret	IRISA - IFSIC, France
Glyn Morrill	Universitat Politècnica de Catalunya, Spain
Reinhard Muskens	Tilburg University, The Netherlands
Rainer Osswald	Heinrich-Heine-Universität Düsseldorf, Germany

Table of Contents[1]

Invited contributions

Contributed papers

Frames as Records

Robin Cooper

University of Gothenburg
Department of Philosophy, Linguistics and Theory of Science
Box 200, 405 30 Göteborg, Sweden
cooper@ling.gu.se
http://www.ling.gu.se/~cooper

Abstract. We suggest a way of formalizing frames using records in type theory. We propose an analysis of frames as records which model situations (including events) and we suggest that frame types (record types) are important in both the analysis of the Partee puzzle concerning rising temperatures and prices and in the analysis of quantification which involves counting events rather than individuals likes passengers or ships passing through a lock.
Our original inspiration for frames comes from the work of [13, 14] and work on FrameNet (https://framenet.icsi.berkeley.edu). An important aspect of our approach to frames, which differs from the Fillmorean approach, is that we treat them as first class objects. That is, they can be arguments to predicates and can be quantified over. The proposal that we have made for solving the Partee puzzle is closely related to the work of [22, 23] whose inspiration is from the work of [2, 1, 3] rather than Fillmore.

Keywords: frames, type theory, record types, events, situations

1 Introduction

In this paper[1] we will suggest a way of formalizing frames using records in type theory and apply this to two phenomena: the "Partee puzzle" concerning rising temperatures and prices for which Montague [24] used individual concepts (functions from possible worlds and times to individuals) and the problem of apparent quantification over events rather than individuals in sentences like (1) discussed by Krifka [18] among others.

(1) Four thousand ships passed through the lock

Our leading idea is to model frames as records and the *roles* in frames (or *frame elements* in the terminology of FrameNet) as fields in records. Records

[1] An expanded version of this paper with a more detailed formal development is available as Chapter 5 of a book draft [7]. This book draft also gives a general introduction to TTR (the type theory with records that we are using here). For a published introduction see [6].

are in turn what we use to model situations so frames and situations in our view turn out to be the same. Given that we are working in a type theory which makes a clear distinction between types and the objects which belong to those types it is a little unclear whether what we call frame should be a record or a record type. We need both and we will talk of frames (records) and frame types (record types). For example, when we look up the frame Ambient_temperature (`https://framenet2.icsi.berkeley.edu/fnReports/data/frameIndex.xml?frame=Ambient_temperature`) in FrameNet we will take that to be an informal description of a frame type which can be instantiated by the kinds of situations which are described in the examples there. In our terms we can characterize a type corresponding to a very stripped down version of FrameNet's Ambient_temperature which is sufficient for us to make the argument we wish to make. This is the type *AmbTempFrame* defined in (2).[2]

$$(2) \quad \begin{bmatrix} \mathrm{x} & : \mathit{Real} \\ \mathrm{loc} & : \mathit{Loc} \\ \mathrm{e} & : \mathrm{temp}(\mathrm{loc}, \mathrm{x}) \end{bmatrix}$$

A record, r, will be of this type just in case in contains three fields with the labels 'x','loc' and 'e' and the objects in these fields are of the types *Real*, *Loc* and temp(r.loc,r.x), that is, a type whose witness is a proof-object (a situation) which shows that the real number in the 'x'-field of r (r.x) is the temperature at the location r.loc. r may contain more fields than those required by the type. A record may not contain more than one field with a given label.

In order to characterize temperature changes we will introduce a notion of scale relating to frames. A scale is a function which maps frames (situations) to a real number. Thus a scale for ambient temperature will be of the type (3a) and the obvious function to choose of that type is the function in (3b) which maps any ambient temperature frame to the real number in its 'x'-field.

(3) a. $(\mathit{AmbTempFrame} \rightarrow \mathit{Real})$
b. $\lambda r{:}\mathit{AmbTempFrame}\ .\ r.\mathrm{x}$

Let us call (3b) ζ_{temp}. As a first approximation we can take an event of a temperature rise to be a string[3] of two temperature frames, $r_1^\frown r_2$, where $\zeta_{\mathrm{temp}}(r_1) < \zeta_{\mathrm{temp}}(r_2)$. Using a notation where T^n is the type of strings of length n each of whose members are of type T and where for a given string, s, $s[0]$ is the first member of s, $s[1]$ the second and so on, a first approximation to the type of temperature rises could be (4).

$$(4) \quad \begin{bmatrix} \mathrm{e} & : \mathit{AmbTempFrame}^2 \\ \mathrm{c_{rise}} & : \zeta_{\mathrm{temp}}(\mathrm{e}[0]) < \zeta_{\mathrm{temp}}(\mathrm{e}[1]) \end{bmatrix}$$

[2] Our treatment of the Partee puzzle here represents an improvement over the proposal presented in Cooper [6] in that it allows a more general treatment of the type of rising events

[3] The idea of events as strings is taken from an important series of papers by Fernando, [8–12] among others

In the c_{rise}-field of (4) we are using $<$ as an infix notation for a predicate 'less-than' with arity $\langle Real, Real\rangle$ which obeys the constraint in (5).

(5) less-than(n, m) is non-empty ("true") iff $n < m$

A more general type for temperature rises is given by (6) where we abstract away from the particular temperature scale used by introducing a field for the scale into the record type. This, for example, allows for an event to be a temperature rise independent of whether it is measured on the Fahrenheit or Celsius scales.

$$(6) \quad \begin{bmatrix} \text{scale} : (AmbTempFrame \rightarrow Real) \\ \text{e} \quad\quad : AmbTempFrame^2 \\ c_{rise} \ : \text{scale(e[0])} < \text{scale(e[1])} \end{bmatrix}$$

This type, though, is now too general to count as the type of temperature rising events. To be of this type, it is enough for there to be some scale on which the rise condition holds and the scale is allowed to be any arbitrary function from temperature frames to real numbers. Of course, it is possible to find some arbitrary function which will meet the rise condition even if the temperature is actually going down. For example, consider a function which returns the number on the Celsius scale but with the sign (plus or minus) reversed making temperatures above 0 to be below 0 and *vice versa*. There are two ways we can approach this problem. One is to make the type in the scale-field a subtype of ($AmbTempFrame \rightarrow Real$) which limits the scale to be one of a number of standardly accepted scales. This may be an obvious solution in the case of temperature where it is straightforward to identify the commonly used scales. However, scales are much more generally used in linguistic meaning and people create new scales depending on the situation at hand. This makes it difficult to specify the nature of the relevant scales in advance and we therefore prefer our second way of approaching this problem.

The second way is to parametrize the type of temperature rising events. By this we mean using a dependent type which maps a record providing a scale to a record type modelling the type of temperature rising events according to that scale. The function in (7) is a dependent type which is related in an obvious way to the record type in (6).

$$(7) \quad \lambda r{:}\left[\text{scale}{:}(AmbTempFrame \rightarrow Real)\right] . \begin{bmatrix} \text{e} \quad\quad : AmbTempFrame^2 \\ c_{rise} \ : r.\text{scale(e[0])} < r.\text{scale(e[1])} \end{bmatrix}$$

According to (6) an event will be a temperature rise if there is some scale according to which the appropriate relation holds between the temperatures of the two stages of the event which we are comparing. According to (7) on the other hand, there is no absolute type of a temperature rise. We can only say whether an event is a temperature rise with respect to some scale or other. If we choose some

non-standard scale like the one that reverses plus and minus temperatures as we suggested above then what we normally call a fall in temperature will in fact be a rise in temperature *according to that scale*. You are in principle allowed to choose whatever scale you like, though if you are using the type in a communicative situation you had better make clear to your interlocutor what scale you are using and perhaps also why you are using this scale as opposed to one of the standardly accepted ones. The dependent types introduce a presupposition-like component to communicative situations. We are assuming the existence of some scale in the context.

Why do we characterize the domain of the function in (7) in terms of records containing a scale rather than just scales as in (8)?

(8) $\lambda\sigma{:}(\mathit{AmbTempFrame} \rightarrow \mathit{Real})\ .\ \begin{bmatrix} \mathrm{e} & : \mathit{AmbTempFrame}^2 \\ \mathrm{c_{rise}} & : \sigma(\mathrm{e}[0]) < \sigma(\mathrm{e}[1]) \end{bmatrix}$

The intuitive reason is that we want to think of the arguments to such functions as being contexts, that is situations (frames) modelled as records. The scale will normally be only one of many informational components which can be provided by the context and the use of a record type allows for there to be more components present. In practical terms of developing an analysis it is useful to use a record type to characterize the domain even if we have only isolated one parameter since if further analysis should show that additional parameters are relevant this will mean that we can add fields to the domain type thereby restricting the domain of the function rather than giving it a radically different type.

And indeed in this case we will now show that there is at least one more relevant parameter that needs to be taken account of before we have anything like a reasonable account of the type of temperature rise events. In (2) we specified that an ambient temperature frame relates a real number ("the temperature") to a spatial location. And now we are saying that a temperature rise is a string of two such frames where the temperature is higher in the second frame. But we have not said anything about how the locations in the two frames should be related. For example, suppose I have a string of two temperature frames where the location in the first is London and the location in the second is Marrakesh. Does that constitute a rise in temperature (assuming that the temperature in the second frame is higher than the one in the first)? Certainly not a temperature rise in London, nor in Marrakesh. If you want to talk about a temperature rise in a particular location then both frames have to have that location and we need a way of expressing that restriction. Of course, you can talk about temperature rises which take place as you move from one place to another and which therefore seem to involve distinct locations. However, it seems that even in these cases something has to be kept constant between the two frames. One might analyse it in terms of a constant path to which both locations have to belong or as a constant relative location such as the place where a particular person (or car, or airplane) is. You cannot just pick two arbitrary temperature frames without holding something constant which ties them together. We will

deal here with the simple case where the location is kept constant.[4] We will say that the background information for judging an event as a temperature rise has to include not only a scale but also a location which is held constant in the two frames. This is expressed in (9).

$$(9)\quad \lambda r{:}\begin{bmatrix}\text{fix}{:}\big[\text{loc}{:}\mathit{Loc}\big]\\ \text{scale}{:}(\mathit{AmbTempFrame}\rightarrow \mathit{Real})\end{bmatrix}\cdot \begin{bmatrix}\text{e} & : (\mathit{AmbTempFrame}\mathbin{\dot\wedge}\big[\text{loc}{=}r.\text{fix.loc}{:}\mathit{Loc}\big])^2\\ \text{c}_{\text{rise}} & : r.\text{scale}(\text{e}[0]) < r.\text{scale}(\text{e}[1])\end{bmatrix}$$

Here we have introduced two new pieces of notation. The symbol '$\dot\wedge$' represents that two record types are to be *merged*, an operation which corresponds to unification of feature structures. Essentially, $T_1 \dot\wedge T_2$ is a record type T_3 such that for any record r, $r : T_3$ iff $r : T_1$ and $r : T_2$. We also introduce a *manifest field* $\big[\text{loc}{=}r.\text{fix.loc}{:}\mathit{Loc}\big]$. A manifest field $\big[\ell{=}a{:}T\big]$ is a convenient notation for $\big[\ell{:}T_a\big]$ where T_a is a *singleton type* such that $b : T_a$ iff $a : T$ and $b = a$. Thus a manifest field in a type specifies what the value in the corresponding record must be. Precise definitions of these concepts can be found in [6, 7]. Here we give an example of the merge of *AmbTempFrame* (spelled out in (10a)) and $\big[\text{loc}{=}r.\text{fix.loc}{:}\mathit{Loc}\big]$ which is identical to (10b).

$$(10)\quad \text{a.}\ \begin{bmatrix}\text{x} & : \mathit{Real}\\ \text{loc} & : \mathit{Loc}\\ \text{e} & : \text{temp(loc, x)}\end{bmatrix} \mathbin{\dot\wedge} \big[\text{loc}{=}r.\text{fix.loc}{:}\mathit{Loc}\big]$$

$$\text{b.}\ \begin{bmatrix}\text{x} & : \mathit{Real}\\ \text{loc}{=}r.\text{fix.loc} & : \mathit{Loc}\\ \text{e} & : \text{temp(loc, x)}\end{bmatrix}$$

The 'fix'-field in the domain type of (9) is required to be a record which provides a location. One reason for making the 'fix'-field a record rather than simply a location is that we will soon see an example where more than one parameter needs to be fixed. It will also help us ultimately in characterizing a general type for a rising event (not just a rise in temperature) if we can refer to the type in the 'fix'-field as *Rec* ("record") rather than to list a disjunction of all the various types of the parameters that can be held constant in different cases.

The temperature rise event itself is now required to be a string of two frames which belong to a subtype of *AmbTempFrame*, namely where the 'loc'-field has been made manifest and is specified to have the value specified for 'loc' in the 'fix'-field. Here we are using the record in the 'fix'-field of the argument to the function to partially specify the type *AmbTempFrame* by fixing values for some of its fields. One can think of the 'fix'-record as playing the role of a partial

[4] Although in astronomical terms, of course, even a location like London is a relative location, that is, where London is according to the rotation of the earth and its orbit around the sun. Thus the simple cases are not really different from the cases apparently involving paths.

assignment of values to fields in the type. To emphasize this important role and to facilitate making general statements without having to name the particular fields involved, we shall introduce an operation which maps a record type, T, and a record, r to the result of specifying T with r, which we will notate as $T \parallel r$. (11) provides an abstract example of how it works.

$$(11)\quad \begin{bmatrix} \ell_1{:}T_1 \\ \ell_2{:}T_2 \\ \ell_3{:}T_3 \end{bmatrix} \parallel \begin{bmatrix} \ell_2{=}a \\ \ell_3{=}b \\ \ell_4{=}c \end{bmatrix} = \begin{bmatrix} \ell_1{:}T_1 \\ \ell_2{=}a{:}T_2 \\ \ell_3{=}b{:}T_3 \end{bmatrix}$$
provided that $a : T_2$ and $b : T_3$

In a case where for example $a : T_2$ but not $b : T_3$ we would have (12).

$$(12)\quad \begin{bmatrix} \ell_1{:}T_1 \\ \ell_2{:}T_2 \\ \ell_3{:}T_3 \end{bmatrix} \parallel \begin{bmatrix} \ell_2{=}a \\ \ell_3{=}b \\ \ell_4{=}c \end{bmatrix} = \begin{bmatrix} \ell_1{:}T_1 \\ \ell_2{=}a{:}T_2 \\ \ell_3{:}T_3 \end{bmatrix}$$

Using this notation we can now rewrite (9) as (13).

$$(13)\quad \lambda r{:}\begin{bmatrix} \text{fix}{:}\big[\text{loc}{:}\mathit{Loc}\big] \\ \text{scale}{:}(\mathit{AmbTempFrame} \rightarrow \mathit{Real}) \end{bmatrix} . \begin{bmatrix} \text{e} & : (\mathit{AmbTempFrame} \| r.\text{fix})^2 \\ \text{c}_{\text{rise}} & : r.\text{scale(e[0])} < r.\text{scale(e[1])} \end{bmatrix}$$

This is still a very simple theory of what a temperature rise event may be but it will be sufficient for our current purposes. We move on now to price rise events. We will take (14) to be the type of price frames, *PriceFrame*.

$$(14)\quad \begin{bmatrix} \text{x} & : \mathit{Real} \\ \text{loc} & : \mathit{Loc} \\ \text{commodity} & : \mathit{Ind} \\ \text{e} & : \text{price(commodity, loc, x)} \end{bmatrix}$$

The fields represented here are based on a much stripped down version of the FrameNet frame `Commerce_scenario` where our 'commdodity'-field corresponds to the frame element called 'goods' and the 'x'-field corresponds to the frame element 'money'. A price rise is a string of two price frames where the value in the 'x'-field is higher in the second. Here, as in the case of a temperature rise, we need to keep the location constant. It does not make sense to say that a price rise has taken place if we compare a price in Marrakesh with a price in London, even though the price in London may be higher. In the case of price we also need to keep the commodity constant, something that does not figure at all in ambient temperature. We cannot say that a price rise has taken place if we have the price of tomatoes in the first frame and the price of oranges in the second frame. Thus, following the model of (13), we can characterize the dependent type of price rises as (15).

$$(15)\quad \lambda r{:}\begin{bmatrix}\text{fix}{:}\begin{bmatrix}\text{loc}{:}\mathit{Loc}\\ \text{commodity}{:}\mathit{Ind}\end{bmatrix}\\ \text{scale}{:}(\mathit{PriceFrame}\rightarrow \mathit{Real})\end{bmatrix}.\ \begin{bmatrix}\text{e} & : (\mathit{PriceFrame}\|r.\text{fix})^2\\ \text{c}_{\text{rise}} & : r.\text{scale}(\text{e}[0]) < r.\text{scale}(\text{e}[1])\end{bmatrix}$$

Finally we consider a third kind of rising event discussed in [6] based on the example in (16).

(16) As they get to deck, they see the Inquisitor, calling out to a Titan in the seas. The giant Titan rises through the waves, shrieking at the Inquisitor.

`http://en.wikipedia.org/wiki/Risen_(video_game)`
accessed 4th February, 2010

Here what needs to be kept constant in the rising event is the Titan. What needs to change between the two frames in the event is the height of the location of the Titan. Thus in this example the location is *not* kept constant. In order to analyze this we can use location frames of the type *LocFrame* as given in (17).

$$(17)\quad \begin{bmatrix}\text{x} & : \mathit{Ind}\\ \text{loc} & : \mathit{Loc}\\ \text{e} & : \text{at}(\text{x},\ \text{loc})\end{bmatrix}$$

The dependent type for a rise in location event is (18).

$$(18)\quad \lambda r{:}\begin{bmatrix}\text{fix}{:}\begin{bmatrix}\text{x}{:}\mathit{Ind}\end{bmatrix}\\ \text{scale}{:}(\mathit{LocFrame}\rightarrow \mathit{Real})\end{bmatrix}.\ \begin{bmatrix}\text{e} & : (\mathit{LocFrame}\|r.\text{fix})^2\\ \text{c}_{\text{rise}} & : r.\text{scale}(\text{e}[0]) < r.\text{scale}(\text{e}[1])\end{bmatrix}$$

Here the obvious scale function does not simply return the value of a field in the location frame. What is needed is a scale based on the height of the location. One way to do this would be to characterize the type of locations, *Loc*, as the type of points in three-dimensional Euclidean space. That is, we consider *Loc* to be an abbreviation for (19).

$$(19)\quad \begin{bmatrix}\text{x-coord} & : \mathit{Real}\\ \text{y-coord} & : \mathit{Real}\\ \text{z-coord} & : \mathit{Real}\end{bmatrix}$$

Each of the fields in (19) corresponds to a coordinate in Euclidean space. A more adequate treatment would be to consider locations as regions in Euclidean space but we will not pursue that here. Treating *Loc* as (19) means that we can characterize the scale function as returning the height of the location in the location frame, as in (20).

(20) $\lambda r{:}\mathit{LocFrame}\ .\ r.\text{loc.z-coord}$

If we wish to restrict the dependent type of rising events to vertical rises we can fix the x and y-coordinates of the location as in (21).

$$(21)\quad \lambda r{:}\begin{bmatrix}\text{fix:}\begin{bmatrix}\text{x:}\mathit{Ind}\\ \text{loc:}\begin{bmatrix}\text{x-coord:}\mathit{Real}\\ \text{y-coord:}\mathit{Real}\end{bmatrix}\end{bmatrix}\\ \text{scale:}(\mathit{LocFrame}\rightarrow\mathit{Real})\end{bmatrix}\ .\ \begin{bmatrix}\text{e} & :(\mathit{LocFrame}\|r.\text{fix})^2\\ \text{c}_{\text{rise}} & : r.\text{scale(e[0])} < r.\text{scale(e[1])}\end{bmatrix}$$

We have now characterized three kinds of rising events. In [5, 6] we argued that there is in principle no limit to the different kinds of rising events which can be referred to in natural language and that new types are created on the fly as the need arises. The formulation in those works did not allow us to express what all these particular meanings have in common. We were only able to say that the various meanings seem to have some kind of family resemblance. Now that we have abstracted out scales and parameters to be fixed we have an opportunity to formulate something more general. There are two things that vary across the different dependent types that we have characterized for risings. One is the frame type being considered and the other is the type of the record which contains the parameters held constant in the rising event. If we abstract over both of these we have a characterization of rising events in general. This is given in (22).

$$(22)\quad \lambda r{:}\begin{bmatrix}\text{frame\_type:}\mathit{RecType}\\ \text{fix\_type:}\mathit{RecType}\\ \text{fix:fix\_type}\\ \text{scale:}(\text{frame\_type}\rightarrow\mathit{Real})\end{bmatrix}\ .\ \begin{bmatrix}\text{e} & :(r.\text{frame\_type}\|r.\text{fix})^2\\ \text{c}_{\text{rise}} & : r.\text{scale(e[0])} < r.\text{scale(e[1])}\end{bmatrix}$$

(22) is so general (virtually everything of content has been parametrized) that it may be hard to see it as being used in the characterization of the meaning of *rise*. What seems important for characterizing the meanings of *rise* that a speaker has acquired is precisely the collection of frame types, and associated fix types and scales which an agent has developed through experience. (22) seems to be relevant to a kind of meta-meaning which specifies what kind of contents can be associated with the word *rise*. In this sense it seems related to the notion of *meaning potential*, a term which has its origins in the work of Halliday [16] where meanings are spoken of informally as being "created by the social system" and charaterized as "integrated systems of meaning potential" (p. 199). The notion is much discussed in more recent literature, for example, Linell [19], where meaning potential is discussed in the following terms: "Lexical meaning potentials are (partly) open meaning resources, where actual meanings can only emerge in specific, situated interactions" (p. 330).

2 Individual vs. frame level nouns

Perhaps the most recent discussion of the Partee puzzle is that of Löbner [23]. As we will see, his proposal is closely related to our own. The puzzle is one that Barbara Partee raised while sitting in on an early presentation of the material that led to [24]. In its simplest form it is that (23c) should follow from (23a,b) given some otherwise apparently harmless assumptions.

(23) a. The temperature is rising
b. The temperature is ninety
c. Ninety is rising

Clearly, our intuitions are that (23c) does not follow from (23a,b).

A central aspect of our analysis of the Partee puzzle is that the contents of common nouns are functions that take frames, that is records, as arguments. We make a distinction between individual level predicates like 'dog' whose arity is $\langle Ind \rangle$ and frame level predicates like 'temperature' whose arity is $\langle Rec \rangle$. The content associated with an utterance event of type "dog" would be (24a). This is contrasted with the content for an utterance of type "temperature" given in (24b).

(24) a. $\lambda r{:}\left[\mathrm{x}{:}Ind\right] \,.\, \left[\,\mathrm{e} : \mathrm{dog}(r.\mathrm{x})\,\right]$
b. $\lambda r{:}\left[\mathrm{x}{:}Rec\right] \,.\, \left[\,\mathrm{e} : \mathrm{temperature}(r.\mathrm{x})\,\right]$

We make an exactly similar distinction between individual level and frame level verb phrases. In (25) we present contents which can be associated with utterances of type "run" and "rise" respectively.

(25) a. $\lambda r{:}\left[\mathrm{x}{:}Ind\right] \,.\, \left[\,\mathrm{e} : \mathrm{run}(r.\mathrm{x})\,\right]$
b. $\lambda r{:}\left[\mathrm{x}{:}Rec\right] \,.\, \left[\,\mathrm{e} : \mathrm{rise}(r.\mathrm{x})\,\right]$

When we predicate the content of *rise*, that is, (25b), of a temperature frame that has no consequence that the real number in the 'x´-field of the temperature frame also rises.

We have made a distinction between individual level nouns like *dog* and frame level nouns like *temperature*, differentiating their contents as in (24) and motivating the distinction with the Partee puzzle. Now consider (26).

(26) a. The dog is nine
b. The dog is getting older/aging
c. Nine is getting older/aging

We have the same intuitions about (26) as we do about the original temperature puzzle. We cannot conclude (26c) from (26a,b). Does this mean that *dog* is a frame level noun after all? Certainly, if we think of frames as being like entries in relational databases it would be natural to think of age (or information

allowing us to compute age such as date of birth) as being a natural field in a dog-frame.

Our strategy to deal with this will be to say that contents of individual level nouns can be coerced to frame level contents, whereas the contents of frame level nouns cannot be coerced "down" to individual level contents. Thus in addition to (24a), repeated as (27a) we have (27b).

(27) a. $\lambda r{:}\left[\text{x}{:}\mathit{Ind}\right] \,.\, \left[\,\text{e} : \text{dog}(r.\text{x})\,\right]$
b. $\lambda r{:}\left[\text{x}{:}\mathit{Rec}\right] \,.\, \left[\,\text{e} : \text{dog\_frame}(r.\text{x})\,\right]$

The predicate 'dog_frame' is related to the predicate 'dog' by the constraint in (28).

(28) dog_frame(r) is non-empty implies $r : \begin{bmatrix}\text{x}{:}\mathit{Ind}\\ \text{e}{:}\text{dog(x)}\end{bmatrix}$

There are several different kinds of dog frames with additional information about a dog which an agent may acquire or focus on. Here we will consider just frames which contain a field labelled 'age' as specified in (29).

(29) If $r : \begin{bmatrix}\text{x}{:}\mathit{Ind}\\ \text{e}{:}\text{dog(x)}\\ \text{age}{:}\mathit{Real}\\ \text{c}_{\text{age}}{:}\text{age\_of(x,age)}\end{bmatrix}$ then dog_frame(r) is non-empty

An age scale, ζ_{age}, for individuals can then be defined as the function in (30).

(30) $\zeta_{\text{age}} = \lambda r{:}\begin{bmatrix}\text{x}{:}\mathit{Ind}\\ \text{age}{:}\mathit{Real}\\ \text{c}_{\text{age}}{:}\text{age\_of(x,age)}\end{bmatrix} \,.\, r.\text{age}$

We can think of the sentence *the dog is nine* as involving two coercions: one coercing the content of *dog* to a frame level property and the other coercing the content of *be* to a function which when applied to a number will return a frame level property depending on an available scale. Such coercions do not appear to be universally available in languages. For example, in German it is preferable to say *die Temperatur ist 35 Grad* "the temperature is 35 degrees" rather than *#die Temperatur ist 35* "the temperature is 35". Similarly *der Hund ist neun Jahre alt* "the dog is nine years old" is preferred over *#der Hund ist neun* "the dog is nine".

3 Passengers and ships

Gupta [15] points out examples such as (31).

(31) a. National Airlines served at least two million passengers in 1975
b. Every passenger is a person
c. National Airlines served at least two million persons in 1975

His claim is that we cannot conclude (31c) from (31a,b). There is a reading of (31a) where what is being counted is not passengers as individual people but passenger events, events of people taking flights, where possibly the same people are involved in several flights. Gupta claims that it is the only reading that this sentence has. While it is certainly the preferred reading for this sentence (say, in the context of National Airlines' annual report or advertizing campaign), I think the sentence also has a reading where individuals are being counted. Consider (32).

(32) National Airlines served at least two million passengers in 1975. Each one of them signed the petition.

While (32) could mean that a number of passengers signed the petition several times our knowledge that people normally only sign a given petition once makes a reading where there are two million distinct individuals involved more likely. Similarly, while (31c) seems to prefer the individual reading where there are two million distinct individuals it is not impossible to get an event reading here. [18] makes a similar point. Gupta's analysis of such examples involves individual concepts and is therefore reminiscent of the functional concepts used by [20, 21] to analyze the Partee puzzle.

Carlson [4] makes a similar point about Gupta's examples in that nouns which appear to normally point to individual related readings can in the right context get the event related readings. One of his examples is a traffic engineer's report as in (33).

(33) Of the 1,000 cars using Elm St. over the past 49 hours, only 12 cars made noise in excess of EPA recommended limits.

It is easy to interpret this in terms of 1,000 and 12 car events rather than individual cars. Carlson's suggestion is to use his notion of *individual stage*, what he describes intuitively as "things-at-a-time". Krifka [18] remarks that "Carlson's notion of a stage serves basically to reconstruct events". While this is not literally correct, the intuition is nevertheless right. Carlson was writing at a time when times and time intervals were used to attempt to capture phenomena that in more modern semantics would be analyzed in terms of events or situations. Thus Carlson's notion of stage is related to a frame-theoretic approach which associates an individual with an event.

Consider the noun *passenger*. It would be natural to assume that passengers are associated with journey events. FrameNet[5] does not have an entry for *passenger*. The closest relevant frame appears to be TRAVEL which has frame elements for traveller, source, goal, path, direction, mode of transport, among

[5] As of 13th May 2015.

others. The FrameNet lexical entry for *journey* is associated with this frame. Let us take the type *TravelFrame* to be the stripped down version of the travel frame type in (34a). Then we could take the type *PassengerFrame* to be (34b).

(34) a. $\begin{bmatrix} \text{traveller} & : \textit{Ind} \\ \text{source} & : \textit{Loc} \\ \text{goal} & : \textit{Loc} \end{bmatrix}$

b. $\begin{bmatrix} \text{x} & : \textit{Ind} \\ \text{e} & : \text{passenger(x)} \\ \text{journey} & : \textit{TravelFrame} \\ \text{c}_{\text{travel}} & : \text{take\_journey(x, journey)} \end{bmatrix}$

A natural constraint to place on the predicate 'take_journey' is that in (35).

(35) If a:*Ind* and e:*TravelFrame*, then the type take_journey(a, e) is non-empty just in case e.traveller = a.

Let us suppose that the basic lexical entry for *passenger* provides the content (36).

(36) $\lambda r{:}[\text{x}{:}\textit{Ind}]\ .\ [\text{e:passenger}(r.\text{x})]$

We can coerce this lexical item to (37).

(37) $\lambda r{:}[\text{x}{:}\textit{Rec}]\ .\ [\text{e:passenger\_frame}(r.\text{x})]$

This means that the non-parametric content is a property of frames. An agent who has the frame type *PassengerFrame* available as a resource can use it to restrict the domain of the property. This produces (38).

(38) $\lambda r{:}[\text{x}{:}\textit{PassengerFrame}]\ .\ [\text{e:passenger\_frame}(r.\text{x})]$

This means that the non-parametric content will now be a property of passenger frames of type *PassengerFrame*. This introduces not only a passenger but also a journey, an event in which in which the passenger is the traveller.

It seems that we have now done something which Krifka [18] explicitly warned us against. At the end of his discussion of Carlson's analysis he comes to the conclusion that it is wrong to look for an explanation of event-related readings of these sentences in terms of a noun ambiguity. One of Krifka's examples is (39) (which gives the title to his paper).

(39) Four thousand ships passed through the lock

This can either mean that four thousand distinct ships passed through the lock or that the there were four thousand ship-passing-through-the-lock events a number of which might have involved the same ships. The problem he sees is that if we

treat *ship* as being ambiguous between denoting individual ships or ship stages in Carlson's sense then there will be too many stages which pass through the lock. For example, suppose that a particular ship passes through the lock twice. This gives us two stages of the ship which pass through the lock. But then, Krifka claims, there will be a third stage, the sum of the first two, which also passes through the lock. It is not clear to me that this is an insuperable problem for the stage analysis. We need to count stages that pass through the lock exactly once. Let us see how the frame analysis fares.

We will start with a singular example in order to avoid the additional problems offered by the plural. Consider (40).

(40) Every passenger gets a hot meal

Suppose that an airline has this as part of its advertizing campaign. Smith, a frequent traveller, takes a flight with the airline and as expected gets a hot meal. A few weeks later she takes another flight with the same airline and does not get a hot meal. She sues the airline for false advertizing. At the hearing, her lawyer argues, citing Gupta [15], that the advertizing campaign claims that every passenger gets a hot meal on every flight they take. The lawyer for the airline company argues, citing Krifka [18], that the sentence in question is ambiguous between an individual and an event reading, that the airline had intended the individual reading and thus the requirements of the advertizing campaign had been met by the meal that Smith was served on the first flight. Smith's lawyer then calls an expert witness, a linguist who quickly crowdsources a survey of native speakers' interpretations of the sentence in the context of the campaign and discovers that there is an overwhelming preference for the meal-on-every-flight reading. (The small percentage of respondents who preferred the individual reading over the event reading gave their occupation as professional logician.) Smith wins the case and receives an additional hot meal.

What is important for us at the moment is the fact that there is an event reading of this sentence. We use the coerced content associated with *passenger* in (38).

In order to simplify matters let us treat *gets a hot meal* as if it were an intransitive verb corresponding to a single predicate 'get_a_hot_meal'. This is a predicate whose arity is $\langle Ind \rangle$. It is individuals, not frames (situations), that get hot meals. Thus the content of *gets a hot meal* will be (41).

(41) $\lambda r{:}\big[\mathrm{x}{:}Ind\big]\ .\ \big[\mathrm{e}{:}\mathrm{get\_a\_hot\_meal}(r.\mathrm{x})\big]$

We need a coercion which will obtain a frame level intransitive verb to match the frame level noun. The new content for *get_a_hot_meal* will be (42).

(42) $\lambda r{:}\big[\mathrm{x}{:}Rec\big]\ .\ \big[\mathrm{e}{:}\mathrm{get\_a\_hot\_meal\_frame}(r.\mathrm{x})\big]$

Recall that if p is a predicate of individuals then p_frame is a predicate of frames that contain an individual of which p holds. This means that an argument, r,

to 'get_a_hot_meal_frame' which makes the type 'get_a_hot_meal_frame(r)' non-empty will be of type (43).

(43) $\begin{bmatrix} \text{x} : \textit{Ind} \\ \text{e} : \text{get\_a\_hot\_meal(x)} \end{bmatrix}$

Thus intuitively the 'every' relation holding between the two frame-level coerced individual properties corresponding to *passenger* and *get_a_hot_meal* will mean "every frame (situation) containing an individual in the 'x'-field who is a passenger taking a journey will be a frame where the individual in the 'x'-field gets a hot meal". Or, more formally, (44).

(44) every r of type $\begin{bmatrix} \text{x} & : \textit{Ind} \\ \text{e} & : \text{passenger(x)} \\ \text{journey} & : \textit{TravelFrame} \\ \text{c}_{\text{travel}} & : \text{take\_journey(x, journey)} \end{bmatrix}$ is of type $\begin{bmatrix} \text{x} : \textit{Ind} \\ \text{e} : \text{get\_a\_hot\_meal(x)} \end{bmatrix}$

This means that every frame of type *PassengerFrame* will be of type (45).

(45) $\begin{bmatrix} \text{x} & : \textit{Ind} \\ \text{e} & : \text{passenger(x)} \wedge \text{get\_a\_hot\_meal(x)} \\ \text{journey} & : \textit{TravelFrame} \\ \text{c}_{\text{travel}} & : \text{take\_journey(x, journey)} \end{bmatrix}$

Thus even though we have coerced to a frame-level reading it is still the passengers (i.e. individuals) in the frames who are getting the hot meal not the situation which is the frame.

4 Conclusion

In this paper we have proposed an analysis of frames as records which model situations (including events) and we have suggested that frame types (record types) are important in both the analysis of the Partee puzzle concerning rising temperatures and prices and in the analysis of quantification which involves counting events rather than individuals likes passengers or ships passing through a lock.

Our original inspiration for frames comes from the work of [13, 14] and work on FrameNet (`https://framenet.icsi.berkeley.edu`). An important aspect of our approach to frames is that we treat them as first class objects. That is, they can be arguments to predicates and can be quantified over. While this is important, it is not surprising once we decide that frames are in fact situations (here modelled by records) or situation types (here modelled by record types). The distinction between frames and frame types is not made in the literature

deriving from Fillmore's work but it seems to be an important distinction to draw if we wish to apply the notion of frame to the kind of examples we have discussed in this chapter.

The proposal that we have made for solving the Partee puzzle is closely related to the work of Löbner [22, 23] whose inspiration is from the work of Barsalou [2, 1, 3] rather than Fillmore. Barsalou's approach embedded in a theory of cognition based on perception and a conception of cognition as dynamic, that is, a system in a constant state of flux [25], seems much in agreement with what we are proposing. Barsalou's [3] characterization of basic frame properties constituting a frame as: "(1) predicates, (2) attribute-value bindings, (3) constraints, and (4) recursion" seem to have a strong family resemblance with our record types. Our proposal for incorporating frames into natural language semantics is, however, different from Löbner's in that he sees the introduction of a psychological approach based on frames as a reason to abandon a formal semantic approach whereas we see type theory as a way of combining the insights we have gained from model theoretic semantics with a psychologically oriented approach.

Our approach to frames has much in common with that of Kallmeyer and Osswald [17] who use feature structures to characterize their semantic domain. We have purposely used record types in a way that makes them correspond both to feature structures and discourse representation structures which allows us to relate our approach to more traditional model theoretic semantics at the same time as being able to merge record types corresponding to unification in feature-based systems. However, our record types are included in a richer system of types including function types facilitates a treatment of quantification and binding which is not available in a system which treats feature structures as a semantic domain.[6]

References

1. Barsalou, L.W.: Cognitive psychology. An overview for cognitive scientists. Lawrence Erlbaum Associates, Hillsdale, NJ (1992)
2. Barsalou, L.W.: Frames, concepts, and conceptual fields. In: Lehrer, A., Kittay, E.F. (eds.) Frames, fields, and contrasts: New essays in semantic and lexical organization, pp. 21–74. Lawrence Erlbaum Associates, Hillsdale, NJ (1992)
3. Barsalou, L.W.: Perceptual symbol systems. Behavioral and Brain Sciences 22, 577–660 (1999)
4. Carlson, G.N.: Generic Terms and Generic Sentences. Journal of Philosophical Logic 11, 145–81 (1982)
5. Cooper, R.: Frames in formal semantics. In: Loftsson, H., Rögnvaldsson, E., Helgadóttir, S. (eds.) IceTAL 2010. Springer Verlag (2010)
6. Cooper, R.: Type theory and semantics in flux. In: Kempson, R., Asher, N., Fernando, T. (eds.) Handbook of the Philosophy of Science, vol. 14: Philosophy of Linguistics, pp. 271–323. Elsevier BV (2012), general editors: Dov M. Gabbay, Paul Thagard and John Woods

[6] It is possible to code up a notation for quantification in feature structures but that is not the same as giving a semantics for it.

7. Cooper, R.: Type theory and language: from perception to linguistic communication (in prep), https://sites.google.com/site/typetheorywithrecords/drafts, draft of book chapters available from https://sites.google.com/site/typetheorywithrecords/drafts
8. Fernando, T.: A finite-state approach to events in natural language semantics. Journal of Logic and Computation 14(1), 79–92 (2004)
9. Fernando, T.: Situations as strings. Electronic Notes in Theoretical Computer Science 165, 23–36 (2006)
10. Fernando, T.: Finite-state descriptions for temporal semantics. In: Bunt, H., Muskens, R. (eds.) Computing Meaning, Volume 3, Studies in Linguistics and Philosophy, vol. 83, pp. 347–368. Springer (2008)
11. Fernando, T.: Situations in LTL as strings. Information and Computation 207(10), 980–999 (2009)
12. Fernando, T.: Constructing Situations and Time. Journal of Philosophical Logic 40, 371–396 (2011)
13. Fillmore, C.J.: Frame semantics. In: Linguistics in the Morning Calm, pp. 111–137. Hanshin Publishing Co., Seoul (1982)
14. Fillmore, C.J.: Frames and the semantics of understanding. Quaderni di Semantica 6(2), 222–254 (1985)
15. Gupta, A.: The Logic of Common Nouns: An Investigation in Quantified Model Logic. Yale University Press, New Haven (1980)
16. Halliday, M.A.K.: Text as semantic choice in social contexts. In: van Dijk, T., Petöfi, J. (eds.) Grammars and descriptions, pp. 176–225. Walter de Gruyter, Berlin (1977)
17. Kallmeyer, L., Osswald, R.: Syntax-driven semantic frame composition in Lexicalized Tree Adjoining Grammars. Journal of Language Modelling 1(2), 267–330 (2013)
18. Krifka, M.: Four Thousand Ships Passed through the Lock: Object-induced Measure Functions on Events. Linguistics and Philosophy 13, 487–520 (1990)
19. Linell, P.: Rethinking Language, Mind, and World Dialogically: Interactional and contextual theories of human sense-making. Advances in Cultural Psychology: Constructing Human Development, Information Age Publishing, Inc., Charlotte, N.C. (2009)
20. Löbner, S.: Intensionale Verben und Funktionalbegriffe. Untersuchung zur Syntax und Semantik von *wechseln* und den vergleichbaren Verben des Deutschen. Narr, Tübingen (1979)
21. Löbner, S.: Intensional Verbs and Functional Concepts: More on the "Rising Temperature" Problem. Linguistic Inquiry 12(3), 471–477 (1981)
22. Löbner, S.: Evidence for frames from human language. In: Gamerschlag, T., Gerland, D., Petersen, W., Osswald, R. (eds.) Frames and Concept Types, Studies in Linguistics and Philosophy, vol. 94, pp. 23–68. Springer, Heidelberg, New York (2014)
23. Löbner, S.: Functional Concepts and Frames (in prep), http://semanticsarchive.net/Archive/jI1NGEwO/Loebner_Functional_Concepts_and_Frames.pdf, available from http://semanticsarchive.net/Archive/jI1NGEwO/Loebner_Functional_Concepts_and_Frames.pdf.
24. 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, pp. 247–270. D. Reidel Publishing Company, Dordrecht (1973)

25. Prinz, J.J., Barsalou, L.W.: Steering a course for embodied representation. In: Dietrich, E., Markman, A.B. (eds.) Cognitive Dynamics: Conceptual and Representational Change in Humans and Machines, pp. 51–77. Psychology Press (2014), previously published in 2000 by Lawrence Erlbaum

Types from frames as finite automata

Tim Fernando

Trinity College Dublin, Ireland

Abstract. An approach to frame semantics is built on a conception of frames as finite automata, observed through the strings they accept. An institution (in the sense of Goguen and Burstall) is formed where these strings can be refined or coarsened to picture processes at various bounded granularities, with transitions given by Brzozowski derivatives.

Keywords: frame, finite automaton, trace, derivative, institution

1 Introduction

A proposal for frame semantics recently put forward in Muskens 2013 analyzes a frame of the sort studied by Barsalou 1999, Löbner 2014, and Petersen & Osswald 2014 (not to forget Fillmore 1982) as a *fact* in a *data lattice* $\langle \mathfrak{F}, \circ, 0 \rangle$ with zero $0 \in \mathfrak{F}$ and meet $\circ : (\mathfrak{F} \times \mathfrak{F}) \to \mathfrak{F}$ (Veltman 1985). A frame such as

is analyzed, relative to any three entities e, x and y, as the $\circ$-combination

$$smash\, e \circ \text{AGENT}\, ex \circ John\, x \circ \text{THEME}\, ey \circ window\, y$$

of five facts, $smash\, e$, $\text{AGENT}\, ex$, $John\, x$, $\text{THEME}\, ey$, and $window\, y$. In general, any fact $g \in \mathfrak{F}$ induces a function $[g]$ from facts to one of three truth values, $\mathbf{t}$, $\mathbf{f}$ and $\mathbf{n}$, such that for all $f \in \mathfrak{F} - \{0\}$,

$$[g](f) \quad = \quad \begin{cases} \mathbf{t} & \text{if } f \circ g = f \ \ \text{(i.e., } f \text{ incorporates } g\text{)} \\ \mathbf{f} & \text{if } f \circ g = 0 \ \ \text{(i.e., } f \text{ and } g \text{ are incompatible)} \\ \mathbf{n} & \text{otherwise.} \end{cases}$$

Functions such as $[g]$ from $\mathfrak{F}$ to $\{\mathbf{t}, \mathbf{f}, \mathbf{n}\}$ are what Muskens calls *propositions*, breaking from possible worlds semantics in replacing possible worlds with facts, and adding a third truth value, $\mathbf{n}$, for a gap between truth and falsity. Sentences are interpreted as propositions, assembled compositionally from an interpretation of words as λ-abstracts of propositions, as in

$$\mathsf{smash} \quad = \quad \lambda y x \lambda f \exists e.[smash\, e \circ \text{AGENT}\, ex \circ \text{THEME}\, ey] f. \tag{1}$$

Muskens attaches significance to the separation of facts from propositions. Identifying frames with facts, he declares

> I reject the idea (defended in Barsalou 1999, who explicitly discusses frame representations of negation, disjunction, and universal quantification) that all natural language meaning can profitably be represented with the help of frames

(page 176).

One way to evaluate Muskens' proposal is by comparing it with others. An alternative to existentially quantifying the event e in (1) is λ-abstraction, as in the analysis of sortal frames in Petersen & Osswald 2014, with

$$\lambda e.\ \mathit{smash}'(e) \wedge \mathit{animate}'(\textsc{agent}'(e)) \wedge \mathit{concrete}'(\textsc{theme}'(e)) \tag{2}$$

for the typed feature structure (a), or, to bring the example closer to (1),

$$\lambda e.\ \mathit{smash}'(e) \wedge \mathit{all}'(\textsc{agent}'(e)) \wedge \mathit{all}'(\textsc{theme}'(e)) \tag{3}$$

for the typed feature structure (b) over a vacuous all-encompassing type *all*.[1]

(a) $\begin{bmatrix} \mathit{smash} & \\ \textsc{agent} & \mathit{animate} \\ \textsc{theme} & \mathit{concrete} \end{bmatrix}$ (b) $\begin{bmatrix} \mathit{smash} & \\ \textsc{agent} & \mathit{all} \\ \textsc{theme} & \mathit{all} \end{bmatrix}$ (c) $\begin{bmatrix} \mathit{smash} \\ \textsc{agent} \\ \textsc{theme} \end{bmatrix}$

It is understood in both (2) and (3) that e is in the domain of the partial functions $\textsc{agent}'$ and $\textsc{theme}'$, making the terms $\textsc{agent}'(e)$ and $\textsc{theme}'(e)$ well-defined. We will simplify (b) shortly to (c), but before dropping *all*, let us use it to illustrate how to express the definedness presuppositions in (2) and (3) under the approach of Cooper 2012. To model a context, Cooper uses a record such as (d), which is of type (e) assuming *all* encompasses all.

(d) $\begin{bmatrix} \textsc{agent} = x \\ \textsc{theme} = y \end{bmatrix}$ (e) $\begin{bmatrix} \textsc{agent} : \mathit{all} \\ \textsc{theme} : \mathit{all} \end{bmatrix}$ (f) $\begin{bmatrix} p_1 : \mathit{smash}(r) \\ p_2 : \mathit{animate}(r.\textsc{agent}) \\ p_3 : \mathit{concrete}(r.\textsc{theme}) \end{bmatrix}$

Now, if *bg* is the record type (e), and φ is the type (f) dependent on a record r of type *bg*, we can form the function

$$(\lambda r : \mathit{bg})\ \varphi \tag{4}$$

mapping a record r of type *bg* to the record type φ. (4) serves as Cooper's meaning function with

- domain *bg* (for background) encoding the definedness presuppositions, and
- record type φ replacing what a Montagovian would have as a truth value.

Compared to the prefix λyx in Muskens' (1), the prefix $(\lambda r : \mathit{bg})$ in (4) provides not just the parameters x and y but the information that they are the agent and theme components of a record r, around which abstraction is centralized in accordance with the methodological assumption

[1] This introductory section presupposes some familiarity with the literature, but is followed by sections that proceed in a more careful manner, without relying on a full understanding of the Introduction.

(†) components are extracted from a single node associated with the frame.

The number of components may be open-ended, as argued famously for events in Davidson 1967

Jones did it slowly, deliberately, in the bathroom with a knife, at midnight

(page 81). Under pressure from multiple components, the assumption (†) is relaxed for non-sortal frames in Löbner 2014 and Petersen & Osswald 2014, with the latter resorting to ϵ-terms (page 249) and ι-terms (page 252). It is, however, possible to maintain (†) by adding attributes that extend the central node to incorporate the required components (making ϵ- and ι-terms unnecessary). At stake in upholding (†) is a record type approach, a finite-state fragment of which is the subject of the present paper.

In Cooper 2012, record types are part of a rich system TTR of types with function spaces going well beyond finite-state methods. Viewing frames as finite automata — bottom-dwellers in the Chomsky hierarchy — certainly leads us back to Muskens' contention that frames cannot capture all natural language meaning. But while any *single* finite automaton is bound to fall short, much can be done with many automata. Or so the present paper argues. Very briefly, the idea is to reduce the matrices (a)–(c) to the sets (a)′–(c)′ of strings over the alphabet $\{smash, \textsc{agent}, \textsc{theme}, animate, concrete, all\}$, and to represent the typing in (g) by the matrix (h) that is reduced to the language (h)′ over an expansion of the alphabet with symbols a_x and a_y for x and y respectively.

(a)′ $\{smash,\ \textsc{agent}\ animate,\ \textsc{theme}\ concrete\}$
(b)′ $\{smash,\ \textsc{agent}\ all,\ \textsc{theme}\ all\}$
(c)′ $\{smash,\ \textsc{agent},\ \textsc{theme}\}$

(g) $\begin{bmatrix} smash \\ \textsc{agent} = x \\ \textsc{theme} = y \end{bmatrix} : \begin{bmatrix} smash \\ \textsc{agent} : animate \\ \textsc{theme} : concrete \end{bmatrix}$

(h) $\begin{bmatrix} smash & \\ \textsc{agent} & \begin{bmatrix} animate \\ a_x \end{bmatrix} \\ \textsc{theme} & \begin{bmatrix} concrete \\ a_y \end{bmatrix} \end{bmatrix}$

(h)′ $\{smash,\ \textsc{agent}\ a_x,\ \textsc{theme}\ a_y\} \cup \{agent\ animate,\ \textsc{theme}\ concrete\}$

To interpret the strings in the sets (a)′, (b)′, (c)′ and (h)′, we assume every symbol a in the string alphabet Σ is interpreted as a (partial) function $[\![a]\!]$, which we extend to strings $s \in \Sigma^*$, setting $[\![\epsilon]\!]$ to the identity, and $[\![sa]\!]$ to the sequential composition $\lambda x.[\![a]\!]([\![s]\!](x))$. Then in place of the λ-expressions (1) to (4), a language L is interpreted as the intersection

$$\bigcap_{s \in L} domain([\![s]\!]) \tag{5}$$

of the domains of the interpretations of strings in L. It is customary to present the interpretations $[\![a]\!]$ model-theoretically (as in the case of the interpretation $\textsc{agent}'$ of $\textsc{agent}$ in (2)), making the interpretations $[\![s]\!]$ and (5) model-theoretic.

But as will become clear below, the functions $[\![\alpha]\!]$ can also be construed as the α-labeled transitions in a finite automaton. The ultimate objective of the present work is to link frames to the finite-state perspective on events in Fernando 2015 (the slogan behind the bias for finite-state methods being *less is more*), as well as to more wide ranging themes of "semantics in flux" in Cooper 2012, and "natural languages as collections of resources" in Cooper & Ranta 2008.

The intersection (5) approximates the image $\{r : bg \mid \varphi\}$ of Cooper's meaning function $(\lambda r : bg)\varphi$ but falls short of maintaining the careful separation that $(\lambda r : bg)\varphi$ makes between the presuppositions bg and the dependent record type φ. That separation is recreated below within what is called an *institution* in Goguen and Burstall 1992, with bg formulated as a signature and φ as a sentence of that signature. Clarifying this formulation is largely what the remainder of the present paper is about, which consists of three sections, plus a conclusion. The point of departure of section 2 is the determinism of a frame — the property that for every state (or node) q and every label a on an arc (or edge), there is at most one arc from q labeled a. Based on determinism and building on Hennessy & Milner 1985, section 2 reduces a state q to a set of strings of labels (i.e., a language). This reduction is tested against states as types and states as particulars in section 3. To ensure the languages serving as states are accepted by finite automata (i.e., regular languages), section 4 works with various finite sets Σ of labels. The sets Σ are paired with record types for signatures, around which approximations are structured following Goguen & Burstall 1992.

One further word about the scope of the present work before proceeding. Functions and λ's are commonly taken for granted in a compositional syntax/semantics interface, yet another significant proposal for which is detailed in Kallmeyer and Osswald 2013 using Lexicalized Tree Adjoining Grammar (distinct from frames with a first-order formulation in §§3.3.3−3.3.4 there compatible with (5) above[2]). The present paper steers clear of any choice of a syntactic formalism, making no claim of completeness in focusing (modestly) on types from frames as finite automata.

2 Deterministic systems and languages

Fix a (possibly infinite) set A of labels. An A-*deterministic system* is a partial function $\delta : Q \times \mathsf{A} \rightharpoonup Q$ from pairs $(q, a) \in Q \times \mathsf{A}$ to elements of Q, called states (of which there may or may not be infinitely many). Let ϵ be the null string (of length 0) and for any state $q \in Q$, let $\delta_q : \mathsf{A}^* \rightharpoonup Q$ be the partial Q-valued function from strings over the alphabet A that repeatedly applies δ starting at q; more precisely, δ_q is the $\subseteq$-least set P of pairs such that

(i) $(\epsilon, q) \in P$, and
(ii) $(sa, \delta(q', a)) \in P$ whenever $(s, q') \in P$ and $(q', a) \in \mathit{domain}(\delta)$.

[2] The compatibility here becomes obvious if the moves described in footnote 5 of page 281 in Kallmeyer and Osswald 2013 are made, and a root added with attributes to the multiple base nodes.

For example,

$$aa' \in \mathit{domain}(\delta_q) \iff (q,a) \in \mathit{domain}(\delta) \text{ and } a' \in \mathit{domain}(\delta_{\delta(q,a)}).$$

The partial functions δ_q determine

$$\mathit{transitions}\ q \xrightarrow{s} \delta_q(s) \ \text{ whenever } \ s \in \mathit{domain}(\delta_q)$$

which we can also read as

$$s\text{-}\mathit{components}\ \delta_q(s) \text{ of } q, \ \text{ for all } \ s \in \mathit{domain}(\delta_q).$$

The labels in A may correspondingly be regarded as acts or as attributes. In either case, there is, we will see, a useful sense in which the language $\mathit{domain}(\delta_q)$ over A holds just about all the A-deterministic system δ has to say about q. An element of $\mathit{domain}(\delta_q)$ is called a *trace of* q (*from* δ), and henceforth, we write $\mathit{trace}_\delta(q)$ interchangably with $\mathit{domain}(\delta_q)$.

2.1 Satisfaction and traces

Given a set A of labels, the set Φ_A of (A-modal) formulas φ is generated

$$\varphi \ ::= \ \top \mid \neg\varphi \mid \varphi \wedge \varphi' \mid \langle a\rangle\varphi$$

from a tautology $\top$, negation $\neg$, conjunction $\wedge$, and modal operators $\langle a\rangle$ with labels $a \in \mathsf{A}$ (Hennessy & Milner 1985). We interpret a formula $\varphi \in \Phi_\mathsf{A}$ relative to an A-deterministic system $\delta : Q \times \mathsf{A} \rightharpoonup Q$ and state $q \in Q$ via a satisfaction relation $\models$ in the usual way, with (keeping δ implicit in the background)

$$q \models \top,$$

'not' $\neg$

$$q \models \neg\varphi \iff \text{not } q \models \varphi,$$

'and' $\wedge$

$$q \models \varphi \wedge \varphi' \iff q \models \varphi \text{ and } q \models \varphi'$$

and the accessibility relation $\{(q, \delta_q(a)) \mid q \in Q \text{ and } a \in \mathit{domain}(\delta_q)\}$ for $\langle a\rangle$

$$q \models \langle a\rangle\varphi \iff a \in \mathit{domain}(\delta_q) \text{ and } \delta_q(a) \models \varphi$$

It is not difficult to see that the set

$$\Phi_\mathsf{A}(q) \ := \ \{\varphi \in \Phi_\mathsf{A} \mid q \models \varphi\}$$

of formulas $\models$-satisfied by q depends precisely on $\mathit{domain}(\delta_q)$. That is, recalling that $\mathit{trace}_\delta(q)$ is $\mathit{domain}(\delta_q)$, the following conditions, (a) and (b), are equivalent for all states $q, q' \in Q$.

(a) $trace_\delta(q) = trace_\delta(q')$
(b) $\Phi_{\mathsf{A}}(q) = \Phi_{\mathsf{A}}(q')$

Let us write $q \sim q'$ if (a), or equivalently (b), holds,[3] and pronounce $\sim$ *trace equivalence*.

2.2 Identity of indiscernibles and states as languages

Identity of indiscernibles (also known as Leibniz's law, and mentioned in Osswald 1999, invoking Quine) can be understood against the set A of attributes as the requirement on δ that distinct pairs q, q' of states (in Q) *not* be trace equivalent

$$q \neq q' \implies q \not\sim q'.$$

Basing discernibility on formulas $\varphi \in \Phi_{\mathsf{A}}$, we say φ *differentiates* q *from* q' if $q \models \varphi$ but not $q' \models \varphi$. It follows that

$$\varphi \text{ differentiates } q \text{ from } q' \iff \neg\varphi \text{ differentiates } q' \text{ from } q$$

and

$$q \sim q' \iff \textit{no} \text{ formula in } \Phi_{\mathsf{A}} \text{ differentiates } q \text{ from } q'.$$

We can replace formulas by attributes and make differentiation symmetric, by agreeing that a label a *differentiates* q *from* q' if (exactly) one of the following holds

(i) $a \in trace_\delta(q) - trace_\delta(q')$
(ii) $a \in trace_\delta(q') - trace_\delta(q)$
(iii) $a \in trace_\delta(q) \cap trace_\delta(q')$ and $\Phi_{\mathsf{A}}(\delta_q(a)) \neq \Phi_{\mathsf{A}}(\delta_{q'}(a))$.

In the case of (i) and (ii), we can see $q \not\sim q'$ already at a, whereas (iii) digs deeper. Two other equivalent ways to say a differentiates q from q' are (a) and (b) below.

(a) a is a prefix of a string in the symmetric difference of trace sets

$$(trace_\delta(q) \cup trace_\delta(q')) - (trace_\delta(q) \cap trace_\delta(q'))$$

(b) there exists $\varphi \in \Phi_{\mathsf{A}}$ such that the formula $\langle a \rangle \varphi$ differentiates either q from q' or q' from q

The notion of an attribute $a \in \mathsf{A}$ differentiating q from q' generalizes straightforwardly to a string $a_1 \cdots a_n \in \mathsf{A}^+$ differentiating q from q'.

In fact, if we reduce a state q to the language $trace_\delta(q)$, the notions of differentiation above link up smoothly with derivatives of languages (Brzozowski

[3] Readers familiar with bisimulations will note that $\sim$ is the largest bisimulation (determinism being an extreme form of image-finiteness; Hennessy & Milner 1985).

1964, Conway 1971, Rutten 1998, among others). Given a language L and a string s, the *s-derivative of L* is the set

$$L_s \;:=\; \{s' \mid ss' \in L\}$$

obtained from strings in L that begin with s, by stripping s off. Observe that for all $q \in Q$ and $s \in \mathit{trace}_\delta(q)$, if $L = \mathit{trace}_\delta(q)$ then the s-derivative of L corresponds to the s-component $\delta_q(s)$ of q

$$L_s \;=\; \mathit{trace}_\delta(\delta_q(s))$$

and L decomposes into its components

$$L \;=\; \epsilon + \sum_{a \in \mathsf{A}} aL_a. \tag{6}$$

The fact that ϵ belongs to $\mathit{trace}_\delta(q)$ reflects prefix-closure. More precisely, a language L is said to be *prefix-closed* if $s \in L$ whenever $sa \in L$. That is, L is prefix-closed iff $\mathit{prefix}(L) \subseteq L$, where the set $\mathit{prefix}(L)$ of prefixes in L

$$\mathit{prefix}(L) \;:=\; \{s \mid L_s \neq \emptyset\}$$

consists of all strings that induce non-empty derivatives. For any non-empty prefix-closed language L, we can form a deterministic system δ over the set

$$\{L_s \mid s \in L\}$$

of s-derivatives of L, for $s \in L$, including ϵ for $L_\epsilon = L = \mathit{trace}_\delta(L)$, where $\mathit{domain}(\delta)$ is defined to be $\{(L_s, a) \mid sa \in L\}$ with

$$\delta(L_s, a) \;:=\; L_{sa} \;\text{ whenever } sa \in L.$$

But what about languages that are not prefix-closed? Without the assumption that L is prefix-closed, we must adjust equation (6) to

$$L \;=\; o(L) + \sum_{a \in \mathsf{A}} aL_a$$

with ϵ replaced by $\emptyset$ in case $\epsilon \notin L$, using

$$o(L) \;:=\; \begin{cases} \epsilon & \text{if } \epsilon \in L \\ \emptyset & \text{otherwise} \end{cases}$$

(called the *constant part* or *output* of L in Conway 1971, page 41). Now, the chain of equivalences

$$a_1 \cdots a_n \in L \;\iff\; a_2 \cdots a_n \in L_{a_1} \;\iff\; \cdots \;\iff\; \epsilon \in L_{a_1 \cdots a_n}$$

means that L is accepted by the automaton with

(i) all s-derivatives of L as states (whether or not $s \in \mathit{prefix}(L)$)

$$Q \;:=\; \{L_s \mid s \in \mathsf{A}^*\}$$

(ii) s-derivatives L_s for $s \in L$ as *final* (accepting) states
(iii) transitions forming a total function $Q \times \mathsf{A} \to Q$ mapping (L_s, a) to L_{sa}

and initial state $L_\epsilon = L$ (e.g., Rutten 1998).

An alternative approach out of prefix closure (from deterministic systems) is to define for any label $a \in \mathsf{A}$ and language $L \subseteq \mathsf{A}^*$, the *a-coderivative of L* to be the set

$$_aL \;:=\; \{s \mid sa \in L\}$$

of strings that, with a attached at the end, belong to L. Observe that the a-coderivative of a prefix-closed language is not necessarily prefix-closed (in contrast to s-derivatives). Furthermore,

Fact 1. *Every language is the coderivative of a prefix-closed language.*

Fact 1 is easy to establish: given a language L, attach a symbol a not occurring in L to the end of L, and form $\mathit{prefix}(La)$ before taking the a-coderivative

$$_a\mathit{prefix}(La) \;=\; L.$$

An a-coderivative effectively builds in a notion of final state (lacking in a deterministic system δ) around not $o(L)$ but $o(L_a)$, checking if ϵ is in L_a, rather than L (i.e., $a \in L$, rather than $\epsilon \in L$). The idea of capturing a type of state through a label (such as a for a-coderivatives) is developed further next.

3 From attribute values to types and particulars

An A-deterministic system $\delta : Q \times \mathsf{A} \rightharpoonup Q$ assigns each state $q \in Q$ a set

$$\hat{\delta}(q) \;:=\; \{(a, \delta(q,a)) \mid a \in \mathsf{A} \cap \mathit{trace}_\delta(q)\}$$

of attribute value pairs (a, q') with values q' that can themselves be thought as sets $\hat{\delta}(q')$ of attribute values pairs. Much the same points in section 2 could be made appealing to the modal logic(s) of attribute value structures (Blackburn 1993) instead of Hennessy & Milner 1985. But is the reduction of q to its trace set, $\mathit{trace}_\delta(q)$, compatible with intuitions about attribute value structures? Let us call a state q *δ-null* if $\hat{\delta}(q) = \emptyset$; i.e., $\mathit{trace}_\delta(q) = \{\epsilon\}$. Reducing a δ-null state q to $\mathit{trace}_\delta(q) = \{\epsilon\}$ lumps all δ-null states into one — which is problematic if distinct atomic values are treated as δ-null (Blackburn 1993). But what if equality with a fixed value were to count as a discerning attribute? Let us define a state q to be *δ-marked* if there is a label $a_q \in \mathsf{A}$ such that for all $q' \in Q$,

$$a_q \in \mathit{trace}_\delta(q') \iff q = q'.$$

Clearly, a δ-marked state q is trace equivalent only to itself. If a state q is *not* δ-marked, we can introduce a fresh label a_q (not in A) and form the $(\mathsf{A} \cup \{a_q\})$-deterministic system

$$\delta[q] \;:=\; \delta \cup \{(q, a_q, q)\}$$

with q $\delta[q]$-marked. To avoid infinite trace sets $\mathit{trace}_{\delta[q]}(q) \supseteq {a_q}^*$ (from loops (q, a_q, q)), we can instead fix a δ-null (or fresh) state $\surd$ and mark q in

$$\delta[q, \surd] \;:=\; \delta \cup \{(q, a_q, \surd)\}.$$

Marking a state is an extreme way to impose Leibniz's law. A more moderate alternative described below introduces types over states, adding constraints to pick out particulars.

3.1 Type-attribute specifications and containment

To differentiate states in a set Q through subsets $Q_t \subseteq Q$ given by types $t \in T$ is to define a binary relation $v \subseteq Q \times T$ (known in Kripke semantics as a T-*valuation*) such that for all $q \in Q$ and $t \in T$

$$v(q, t) \iff q \in Q_t.$$

We can incorporate v into $\delta : Q \times \mathsf{A} \rightharpoonup Q$ by adding a fresh attribute a_t, for each $t \in T$, to A for the expanded attribute set

$$\mathsf{A}_T \;:=\; \mathsf{A} \cup \{a_t \mid t \in T\}$$

and forming either the A_T-deterministic system

$$\delta[v] \;:=\; \delta \cup \{(q, a_t, q) \mid t \in T \text{ and } v(q, t)\}$$

or, given a δ-null state $\surd$ outside $\bigcup_{t \in T} Q_t$, the A_T-deterministic system

$$\delta[v, \surd] \;:=\; \delta \cup \{(q, a_t, \surd) \mid t \in T \text{ and } v(q, t)\}.$$

In practice, a type t may be defined from other types, as in (a) below.

$$\text{(a)}\quad t = \begin{bmatrix} \mathit{smash} & : \top \\ \textsc{agent} & : \mathit{animate} \\ \textsc{theme} & : \mathit{concrete} \end{bmatrix} \qquad\qquad \text{(b)}\quad e = \begin{bmatrix} \mathit{smash} & = x \\ \textsc{agent} & = y \\ \textsc{theme} & = z \end{bmatrix}$$

The record e given by (b) is an instance of t just in case x, y and z are instances of the types $\top$, *animate* and *concrete* respectively

$$e : t \iff x : \top \;\text{ and }\; y : \mathit{animate} \;\text{ and }\; z : \mathit{concrete}.$$

It is natural to analyze t in (a) as the modal formula φ_t with three conjuncts

$$\varphi_t \;=\; \langle \mathit{smash} \rangle \top \wedge \langle \textsc{agent} \rangle \langle \mathit{animate} \rangle \top \wedge \langle \textsc{theme} \rangle \langle \mathit{concrete} \rangle \top$$

(implicitly analyzing $\top$, *animate* and *concrete* as $\top$, $\langle animate\rangle\top$ and $\langle animate\rangle\top$ respectively) and to associate e with the trace set

$$q(e) \;=\; \epsilon \;+\; smash\ q(x) \;+\; \textsc{agent}\ q(y) \;+\; \textsc{theme}\ q(z)$$

(given trace sets $q(x), q(y)$ and $q(z)$ for x, y and z respectively) for the reduction

$$\begin{aligned} e : t \;&\Longleftrightarrow\; q(e) \models \varphi_t \\ &\Longleftrightarrow\; animate \in q(y) \text{ and } concrete \in q(z). \end{aligned}$$

We can rewrite (a) as the A'-deterministic system

$$\tau' \;:=\; \{(t, smash, \top),\ (t, \textsc{agent}, animate),\ (t, \textsc{theme}, concrete)\}$$

over the attribute set

$$\mathsf{A}' \;:=\; \{smash, \textsc{agent}, \textsc{theme}\}$$

and state set

$$T' \;:=\; \{t, \top, animate, concrete\}$$

given by types. To apply τ' to a deterministic system δ with states given by tokens of types, the following notion will prove useful. A $(T, \mathsf{A}, \surd)$-*specification* is an A_T-deterministic system $\tau : T \times \mathsf{A}_T \rightharpoonup T$ where $\surd \in T$ is τ-null and each $t \in T - \{\surd\}$ is τ-marked, with for all $t' \in T$,

$$(t', a_t) \in domain(\tau) \;\Longleftrightarrow\; t = t'$$

(the intuition being to express a type t as the modal formula $\langle a_t\rangle\top$). For τ' given above, we can form the $(T' \cup \{\surd\}, \mathsf{A}', \surd)$-specification

$$\tau' \;\cup\; \{(x, a_x, \surd) \mid x \in T'\}.$$

Let us agree that a set Ψ of modal formulas is *true in* δ if every fornula in Ψ is satisfied, relative to δ, by every δ-state. The content of a $(T, \mathsf{A}, \surd)$-specification τ is given by the set

$$\begin{aligned} spec(\tau) \;:=\; &\{\langle a_t\rangle\top \supset \langle a\rangle\top \mid (t, a, \surd) \in \tau\} \cup \\ &\{\langle a_t\rangle\top \supset \langle a\rangle\langle a_{t'}\rangle\top \mid (t, a, t') \in \tau \text{ and } t' \neq \surd\} \end{aligned}$$

of formulas true in an A_T-deterministic system δ precisely if for every $(t, a, t') \in \tau$ and $q \in Q_t$,

$$a \in trace_\delta(q) \;\text{ and }\; \delta(q, a) \in Q_{t'}$$

where for every $t \in T - \{\surd\}$, Q_t is $\{q \in Q \mid a_t \in trace_\delta(q)\}$ and $Q_\surd = Q$. We can express membership in a trace set

$$s \in trace_\delta(q) \;\Longleftrightarrow\; q \models_\delta \langle s\rangle\top$$

through formulas $\langle s\rangle\varphi$ defined by induction on s:

$$\langle\epsilon\rangle\varphi \;:=\; \varphi \quad\text{and}\quad \langle as\rangle\varphi \;:=\; \langle a\rangle\langle s\rangle\varphi$$

so that for $a_1 a_2 \cdots a_n \in \mathsf{A}^n$,

$$\langle a_1 a_2 \cdots a_n\rangle\varphi \;=\; \langle a_1\rangle\langle a_2\rangle \cdots \langle a_n\rangle\varphi.$$

Given a language L, let us say q δ-*contains* L if $L \subseteq trace_\delta(q)$.

Fact 2. *For any $(T, \mathsf{A}, \surd)$-specification τ, $spec(\tau)$ is true in an A_T-deterministic system $\delta : Q \times \mathsf{A}_T \rightharpoonup Q$ iff for every $t \in T - \{\surd\}$ and $q \in Q$, q δ-contains $trace_\tau(t)$ whenever q δ-contains $\{a_t\}$.*

Under certain assumptions, $trace_\tau(t)$ is finite. More specifically, let $T_0 = \emptyset$ and for any integer $n \geq 0$,

$$T_{n+1} \ := \ \{t \in T \mid \tau \cap \ (\{t\} \times \mathsf{A} \times T) \subseteq T \times \mathsf{A} \times T_n\}$$

(making $T_1 = \{\surd\}$). For each $t \in T_n$, $trace_\tau(t)$ is finite provided

($\star$) for all $t \in T$, $\{a \in \mathsf{A} \mid (t, a) \in domain(\tau)\}$ is finite.

Although ($\star$) and $T \subseteq \bigcup_n T_n$ can be expected of record types in Cooper 2012, notice that if $(t, a, t) \in \tau$ then $t \notin \bigcup_n T_n$ and $a^* \subseteq trace_\tau(t)$.

3.2 Terminal attributes, $\Diamond$ and subtypes

Fact 2 reduces a $(T, \mathsf{A}, \surd)$-specification τ to its trace sets, $trace_\tau(t)$ for $t \in T - \{\surd\}$, unwinding $spec(\tau)$ to

$$\langle a_t \rangle \top \ \supset \bigwedge_{s \in trace_\tau(t)} \langle s \rangle \top \tag{7}$$

for $t \in T - \{\surd\}$ (where the conjunction might be infinite). The converse of (7) follows from $a_t \in trace_\tau(t)$. (7) has the form

$$\varphi_t \ \supset \ \varphi_{t[\tau]}$$

with antecedent $\langle a_t \rangle \top$ construed as a formula φ_t representing t, and consequent $\bigwedge_{s \in trace_\tau(t)} \langle s \rangle \top$ as a formula $\varphi_{t[\tau]}$ representing τ's conception of t. The attribute a_t often has the following property. Given an A-deterministic system δ, let us say an attribute $a \in \mathsf{A}$ is *δ-terminal* if the set

$$Tml_{\mathsf{A}}(a) \ := \ \{\neg \langle sab \rangle \top \mid s \in \mathsf{A}^* \text{ and } b \in \mathsf{A}\}$$

of formulas is true in δ — i.e., for every δ-state q, string $s \in \mathsf{A}^*$ and attribute $b \in \mathsf{A}$, $sab \notin trace_\delta(q)$. It is natural to associate a frame such as

with an A-deterministic system δ in which labels on nodes (i.e., *John*, *smash* and *window*) are δ-terminal, while labels on arcs (AGENT, THEME) are not.

Next, we quantify away the strings s mentioned in $Tml_{\mathsf{A}}(a)$ for a useful modal operator $\Diamond$. Given an A-deterministic system $\delta : Q \times \mathsf{A} \rightharpoonup Q$, and states $q, q' \in Q$, we say q' is a *δ-component of* q and write $q \rightsquigarrow_\delta q'$, if q' is $\delta_q(s)$ for

some string $s \in \mathsf{A}^*$. With the relation $\rightsquigarrow_\delta$, we extend satisfaction $\models$ to formulas $\Diamond\varphi$

$$q \models \Diamond\varphi \iff (\exists q')\ q \rightsquigarrow_\delta q' \text{ and } q' \models \varphi$$

making $\Diamond\varphi$ essentially the infinitary disjunction $\bigvee_{s\in\mathsf{A}^*}\langle s\rangle\varphi$ and its dual $\Box\varphi := \neg\Diamond\neg\varphi$ the infinitary conjunction $\bigwedge_{s\in\mathsf{A}^*}[s]\varphi$ (where $[s]\varphi := \neg\langle s\rangle\neg\varphi$). The extension preserves invariance under trace equivalence $\sim$

$$\text{whenever } q \sim q' \text{ and } q \models \Diamond\varphi,\ q' \models \Diamond\varphi.$$

That is, for the purpose of $\models$, we can reduce a state q to its trace set $\mathit{trace}_\delta(q)$. Accordingly, we collect all non-empty prefix-closed subsets of A^* in

$$\mathit{Mod}(\mathsf{A}) := \{\mathit{prefix}(L \cup \{\epsilon\}) \mid L \subseteq \mathsf{A}^*\}$$

(where "Mod" is for models) with the understanding that the transitions δ between $q, q' \in \mathit{Mod}(\mathsf{A})$ are given by derivatives

$$\delta(q,a) = q' \iff q_a = q'.$$

Given a set Ψ of modal formulas over A, we form the subset of $\mathit{Mod}(\mathsf{A})$ satisfying every formula of Ψ

$$\mathit{Mod}_\mathsf{A}(\Psi) := \{q \in \mathit{Mod}(\mathsf{A}) \mid (\forall\varphi \in \Psi)\ q \models \varphi\}$$

and say Ψ is A-*equivalent to* a set Ψ' of modal formulas over A if they have the same models

$$\Psi \equiv_\mathsf{A} \Psi' \iff \mathit{Mod}_\mathsf{A}(\Psi) = \mathit{Mod}_\mathsf{A}(\Psi').$$

Ψ can be strengthened to

$$\begin{aligned}\Box_\mathsf{A}(\Psi) &:= \{\neg\langle s\rangle\neg\varphi \mid s \in \mathsf{A}^* \text{ and } \varphi \in \Psi\}\\ &\equiv_\mathsf{A} \{\Box\varphi \mid \varphi \in \Psi\}\end{aligned}$$

requiring that every formula in Ψ holds in all components. Clearly,

$$\mathit{Tml}_\mathsf{A}(a) \equiv_\mathsf{A} \Box_\mathsf{A}(\{\neg\langle ab\rangle\top \mid b \in \mathsf{A}\}).$$

Given representations φ_t and φ_u of types t and u, we can express the subtyping $t \sqsubseteq u$ as the set

$$\text{`}t \sqsubseteq u\text{'} := \{\neg\langle s\rangle(\varphi_t \wedge \neg\varphi_u) \mid s \in \mathsf{A}^*\}$$

of modal formulas denying the existence of components of type t but not u. Then '$t \sqsubseteq u$' requires $\varphi_t \supset \varphi_u$ of all components

$$\text{`}t \sqsubseteq u\text{'} \equiv_\mathsf{A} \Box(\varphi_t \supset \varphi_u)$$

bringing us back to the implication (7) above of the form $\varphi_t \supset \varphi_{\tau[t]}$. Inasmuch as an attribute a is represented by the formula $\langle a\rangle\top$, we can speak of a being contained in an attribute b, $a \sqsubseteq b$, when asserting

$$\Box(\langle a\rangle\top \supset \langle b\rangle\top).$$

3.3 Typings and particulars

We can reduce a typing $p : t$ to a subtyping through the equivalence

$$p : t \iff \{p\} \sqsubseteq t$$

assuming we can make sense of the singleton type $\{p\}$. Given an A-deterministic system δ and a δ-state q, let us call a formula φ an (A, q)-*nominal* if q satisfies the set

$$\begin{aligned} \mathit{Nom}_{\mathsf{A}}(\varphi) \;&:=\; \{\neg(\langle s'\rangle(\varphi \wedge \langle s\rangle\top) \;\wedge\; \langle s''\rangle(\varphi \wedge \neg\langle s\rangle\top)) \mid s, s', s'' \in \mathsf{A}^*\} \\ &\equiv_{\mathsf{A}} \; \{\neg(\Diamond(\varphi \wedge \langle s\rangle\top) \;\wedge\; \Diamond(\varphi \wedge \neg\langle s\rangle\top)) \mid s \in \mathsf{A}^*\} \end{aligned}$$

of formulas that together say any two δ-components that δ-satisfy φ δ-contain the same languages over A. We can rephrase $\mathit{Nom}_{\mathsf{A}}(\varphi)$ as implications

$$\begin{aligned} \mathit{Nom}_{\mathsf{A}}(\varphi) \;&\equiv_{\mathsf{A}} \{\Diamond(\varphi \wedge \langle s\rangle\top) \supset \Box(\varphi \supset \langle s\rangle\top) \mid s \in \mathsf{A}^*\} \\ &\equiv_{\mathsf{A}} \; \{\Diamond(\varphi \wedge \psi) \supset \Box(\varphi \supset \psi) \mid \psi \in \Phi_{\mathsf{A}}\} \end{aligned}$$

making δ-components that δ-satisfy φ Φ_{A}-indistinguishable.[4]

Fact 3. *Let δ be an A-deterministic system, q be a δ-state and φ be an (A, q)-nominal.*

(i) For all δ-components q' and q'' of q,

$$q' \models \varphi \text{ and } q'' \models \varphi \;\text{ implies }\; q' \sim q''.$$

(ii) There is a δ-component of q that δ-satisfies φ and ψ

$$q \models \Diamond(\varphi \wedge \psi)$$

iff every δ-component of q satisfies $\varphi \supset \psi$ and some δ-component of q δ-satisfies φ

$$q \models \Box(\varphi \supset \psi) \;\wedge\; \Diamond\varphi.$$

Now, an (A, q)-*particular* is an (A, q)-nominal φ such that q has a δ-component that δ-satisfies φ — i.e., in addition to all formulas in $\mathit{Nom}_{\mathsf{A}}(\varphi)$, q satisfies $\Diamond\varphi$. We say (A, q) *verifies* $p : t$ if

$$\varphi_{\{p\}} \text{ is an } (\mathsf{A}, q)\text{-particular and } q \models \Box(\varphi_{\{p\}} \supset \varphi_t).$$

Part (ii) of Fact 3 says this is equivalent to $\varphi_{\{p\}}$ being an (A, q)-nominal and q having a δ-component that δ-satisfies $\varphi_{\{p\}} \wedge \varphi_t$. Note that if q is δ-marked by a_q then $\langle a_q\rangle\top$ is an (A, q')-nominal for all δ-states q'. The weaker notion of an (A, q)-nominal φ has the advantage over $\langle a_q\rangle\top$ that the set $\mathit{Nom}_{\mathsf{A}}(\varphi)$ of

[4] $\mathit{Nom}_{\mathsf{A}}(\varphi)$ does the work of the scheme (Nom_N) for nominals given in Blackburn 1993, just as $\mathit{Tml}_{\mathsf{A}}(a)$ is analogous to the scheme (*Term*) there for instantiating atomic information at terminal nodes.

modal formulas allows the modal formulas satisfied by a state δ-satisfying $\varphi_{\{p\}}$ to vary with δ.[5] We can use the set $Nom_{\mathsf{A}}(\varphi) \cup \{\Diamond\varphi\}$ to lift the notion of an (A, q)-particular to that of a (A, Ψ)-*particular*, given a set Ψ of modal formulas, requiring that $Nom_{\mathsf{A}}(\varphi) \cup \{\Diamond\varphi\}$ follow from Ψ in that

$$Mod_{\mathsf{A}}(\Psi) \ \subseteq \ Mod_{\mathsf{A}}(Nom_{\mathsf{A}}(\varphi) \cup \{\Diamond\varphi\}).$$

Having described how a particular can be understood as a formula (or attribute a via $\langle a\rangle\top$), it is perhaps worth emphasizing that we need *not* understand particulars as formulas (or attributes). In the present context, particulars are first and foremost δ-states in an A-deterministic system δ. For any δ-state q, we can speak of types to which q belongs with*out* introducing an attribute that (relative to an extension of δ) represents q. What is important is that we build into A all the structure in particulars we wish to analyze.

4 Finite approximations and signatures

Throughout this section, we shall work with a set A of labels that is large enough so that we may assume trace equivalence $\sim$ is equality. Identity of indiscernibles in an A-deterministic system δ reduces a state q to $trace_\delta(q)$, allowing us to identify a state with a prefix-closed, non-empty subset of A^*. As huge as A can be, we can form the set $Fin(\mathsf{A})$ of finite subsets of A and use the equality

$$\mathsf{A}^* \ = \ \bigcup_{\Sigma \in Fin(\mathsf{A})} \Sigma^*$$

to approach a language $L \subseteq \mathsf{A}^*$ via its intersections $L \cap \Sigma^*$ (for $\Sigma \in Fin(\mathsf{A})$)

$$L \ = \ \bigcup_{\Sigma \in Fin(\mathsf{A})} (L \cap \Sigma^*)$$

at the cost of bounding discernibility to Σ. For $X \in Fin(\mathsf{A}) \cup \{\mathsf{A}\}$, we define an *$X$-state* to be a non-empty, prefix-closed subset q of X^* on which transitions are given by derivatives

$$\delta(q, a) = q_a \ \text{ for } a \in \mathsf{A} \cap q$$

making $\delta_q(s) = q_s$ for $s \in q$. Henceforth, we put the notation δ aside, but track the parameter X, which we set to a finite subset Σ of A to pick out what is of particular interest.

Two subsections make up the present section. The first illustrates how the set A of labels can explode; the second how, after this explosion, to keep a grip on

[5] By contrast, for a language q over A, all formulas in the set

$$\{\Box(\langle a_q\rangle\top \supset (\langle s\rangle\top \wedge \neg\langle s'\rangle\top)) \mid s \in q \text{ and } s' \in \mathsf{A}^* - q\}$$

must be satisfied for a_q to mark q.

satisfaction $\models$ of formulas. We associate the formulas with sets $X \in \mathit{Fin}(\mathsf{A}) \cup \{\mathsf{A}\}$, defining $\mathit{sen}(X)$ to be the set of formulas generated from $a \in X$ and $Y \subseteq X$ according to

$$\varphi \quad ::= \quad \top \mid \neg\varphi \mid \varphi \wedge \varphi' \mid \langle a \rangle \varphi \mid \Box_Y \varphi.$$

We interpret these formulas over A-states q, treating $\top$, $\neg$ and $\wedge$ as usual, and setting

$$q \models \langle a \rangle \varphi \quad \Longleftrightarrow \quad a \in q \text{ and } q_a \models \varphi$$

which generalizes to strings $s \in X^*$

$$q \models \langle s \rangle \varphi \quad \Longleftrightarrow \quad s \in q \text{ and } q_s \models \varphi$$

(recalling that $\langle a_1 \rangle \cdots \langle a_n \rangle \varphi$ is $\langle a_1 \rangle \cdots \langle a_n \rangle \varphi$). As for $\Box_Y$, we put

$$q \models \Box_Y \varphi \quad \Longleftrightarrow \quad (\forall s \in q \cap Y^*)\ q_s \models \varphi$$

relativizing $\Box$ to Y for reasons that will become clear below. When $\Box$ appears with no subscript, it is understood to carry the subscript A; i.e., $\Box$ is $\Box_{\mathsf{A}}$. Also, for $s \in \mathsf{A}^*$, we will often shorten the formula $\langle s \rangle \top$ to s, writing, for example, $\neg a$ for $\neg \langle a \rangle \top$.

4.1 Labels refining partitions

For any $\Sigma \in \mathit{Fin}(\mathsf{A})$ and Σ-state q, we can formulate the Myhill-Nerode equivalence $\approx_q$ between strings $s, s' \in \Sigma^*$ with the same extensions in q

$$s \approx_q s' \quad \Longleftrightarrow \quad (\forall w \in \Sigma^*)\ (sw \in q \iff s'w \in q)$$

(e.g., Hopcroft & Ullman 1979) in terms of derivatives

$$s \approx_q s' \quad \Longleftrightarrow \quad q_s = q_{s'}$$

from which it follows that q is regular iff its set

$$\{q_s \mid s \in \Sigma^*\}$$

of derivatives is finite. There are strict bounds on what we can discern with Σ and Σ-states. For example, the regular language

$$q' \ = \ \textsc{father}^* + \textsc{father}^* \mathit{man}$$

over $\Sigma' = \{\textsc{father}, \mathit{man}\}$ is a model of

$$\mathit{man} \ \wedge \ \Box(\mathit{man} \ \supset \ \langle \textsc{father} \rangle \mathit{man})$$

(i.e., q' is a token of the record type *man* given by eq(*man*)= $\{(\text{FATHER}, man)\}$), with derivatives

$$q'_s \;=\; \begin{cases} q' & \text{if } s \in \text{FATHER}^* \\ \{\epsilon\} & \text{if } s \in \text{FATHER}^* man \\ \emptyset & \text{otherwise} \end{cases}$$

so that according to the definitions from the previous section (with δ given by derivatives of q'), the Σ'-state $\{\epsilon\}$ is null, and the label *man* is terminal. The Σ'-state q' does not differentiate between distinct tokens of *man*, although it is easy enough to introduce additional labels and formulas

$$\Box\,(man \;\supset\; (John \;\vee Peter \vee Other_{man:John,Peter})) \tag{8}$$

with

$$\Box\,(John \;\supset\; \neg\langle \text{FATHER}\rangle John)$$

unless say, *John* could apply to more than one particular, as suggested by

$$\Box\,(John \;\supset\; (John1 \;\vee John2 \vee Other_{John:1,2})).$$

Generalizing line (8) above, we can refine a label a to a finite partition from a set $\Sigma \in Fin(\mathsf{A})$, asserting

$$Partition_a(\Sigma) \;\;:=\;\; \Box\,(a \supset Uniq(\Sigma))$$

where $Uniq(\Sigma)$ picks out exactly one label in Σ

$$Uniq(\Sigma) \;\;:=\;\; \bigvee_{a\in\Sigma} (a \wedge \bigwedge_{a'\in\Sigma-\{a\}} \neg a')).$$

Co-occurrence restrictions on a set Σ of alternatives

$$Alt(\Sigma) \;\;:=\;\; \Box \bigwedge_{a\in\Sigma} \bigwedge_{a'\in\Sigma-\{a\}} \neg(a \wedge a')$$

(declaring any pair to be incompatible) is equivalent to the conjunction

$$\bigwedge_{a\in\Sigma} Partition_a(\Sigma).$$

And if the labels in Σ are understood to specify components, we might say a label marks a Σ-atom if it rules out a'-components for $a' \in \Sigma$

$$\begin{aligned} Atom_\Sigma(a) \;\;&:=\;\; \Box(a \supset \bigwedge_{a'\in\Sigma} \neg a') \\ &\iff\;\; Partition_a(\Sigma \cup \{a\}). \end{aligned}$$

That said, we arrived at $Partition_a(\Sigma)$ from *man* above through the example $\Sigma = \{John,\ Peter,\ Other_{man:John,Peter}\}$ of labels that differentiate between

tokens of *man* rather than (as in the case of the record label FATHER or AGENT or THEME) specifying components. We can extend the example

$$Partition_{man}(\{John, Peter,\ Other_{man:John,Peter}\})$$

through a function $f : T \to Fin(\mathsf{A})$ from some finite set T of labels (representing types) such that for $a \in T$, $f(a)$ outlines a partition of a (just as $\{John, Peter, Other_{man:John,Peter}\}$ does for man). An *f-token* is then an A-state q such that

$$q \models \bigwedge_{a \in T} Partition_a(f(a))$$

making (as it were) a choice from $f(a)$, for each $a \in T$.

4.2 Reducts for satisfaction

Given a string $s \in \mathsf{A}^*$ and a set $\Sigma \in Fin(\mathsf{A})$, the longest prefix of s that belongs to Σ^* is computed by the function $\pi_\Sigma : \mathsf{A}^* \to \Sigma^*$ defined by $\pi_\Sigma(\epsilon) := \epsilon$ and

$$\pi_\Sigma(as) \ := \ \begin{cases} a\,\pi_\Sigma(s) & \text{if } a \in \Sigma \\ \epsilon & \text{otherwise.} \end{cases}$$

The *Σ-reduct of* a language $q \subseteq \mathsf{A}^*$ is the image of q under π_Σ

$$q \restriction \Sigma \ := \ \{\pi_\Sigma(s) \mid s \in q\}\ .$$

If q is an A-state, then its Σ-reduct, $q \restriction \Sigma$, is a Σ-state and is just the intersection $q \cap \Sigma^*$ with Σ^*. Σ-reducts are interesting because satisfaction $\models$ of formulas in $sen(\Sigma)$ can be reduced to them.

Fact 4. *For every $\Sigma \in Fin(\mathsf{A})$, $\varphi \in sen(\Sigma)$ and A-state q,*

$$q \models \varphi \iff q \restriction \Sigma \models \varphi$$

and if, moreover, $s \in q \restriction \Sigma$, then

$$q \models \langle s \rangle \varphi \iff (q \restriction \Sigma)_s \models \varphi. \tag{9}$$

Fact 4 is proved by a routine induction on $\varphi \in sen(\Sigma)$. Were $sen(\Sigma)$ closed under $\Box = \Box_{\mathsf{A}}$, Fact 4 would fail for infinite A; hence, the relativized operators $\Box_Y$ for $Y \subseteq \Sigma$ in $sen(\Sigma)$.

There is structure lurking around Fact 4 that is most conveniently described in category-theoretic terms. For $\Sigma \in Fin(\mathsf{A})$, let $Q(\Sigma)$ be the category with

- Σ-states q as objects
- pairs (q, s) such that $s \in q$ as morphisms from q to q_s, composing by concatenating strings

  $$(q, s)\ ;\ (q_s, s')\ :=\ (q, ss')$$

  with identities (q, ϵ).

To turn Q into a functor from $Fin(\mathsf{A})^{op}$ (with morphisms (Σ', Σ) such that $\Sigma \subseteq \Sigma' \in Fin(\mathsf{A})$) to the category **Cat** of small categories, we map a $Fin(\mathsf{A})^{op}$-morphism (Σ', Σ) to the functor $Q(\Sigma', \Sigma) : Q(\Sigma') \to Q(\Sigma)$ sending a Σ'-state q' to the Σ-state $q' \restriction \Sigma$, and the $Q(\Sigma')$-morphism (q', s') to the $Q(\Sigma)$-morphism $(q' \restriction \Sigma, \pi_\Sigma(s'))$. The *Grothendieck construction for* Q is the category $\int Q$ where

- objects are pairs (Σ, q) such that $\Sigma \in Fin(\mathsf{A})$ and q is a Σ-state
- morphisms from (Σ', q') to (Σ, q) are pairs $((\Sigma', \Sigma), (q'', s))$ of $Fin(\mathsf{A})^{op}$-morphisms (Σ', Σ) and $Q(\Sigma)$-morphisms (q'', s) such that $q'' = q' \restriction \Sigma$ and $q = q''_s$

(e.g., Tarlecki, Burstall & Goguen 1991). $\int Q$ integrates the different categories $Q(\Sigma)$ (for $\Sigma \in Fin(\mathsf{A})$), lifting a $Q(\Sigma)$-morphism (q, s) to a $(\int Q)$-morphism from (Σ', q') to (Σ, q_s) whenever $\Sigma \subseteq \Sigma'$ and $q' \restriction \Sigma = q$.

Given a small category C, let us write $|C|$ for the set of objects of C. Thus, for $\Sigma \in Fin(\mathsf{A})$, $|Q(\Sigma)|$ is the set

$$|Q(\Sigma)| \;=\; \{q \subseteq \Sigma^* \mid q \neq \emptyset \text{ and } q \text{ is prefix-closed}\}$$

of Σ-states. Next, for $(\Sigma, q) \in |\int Q|$, let $Mod(\Sigma, q)$ be the full subcategory of $Q(\Sigma)$ with objects required to have q as a subset

$$|Mod(\Sigma, q)| \;:=\; \{q' \in |Q(\Sigma)| \mid q \subseteq q'\}.$$

That is, $|Mod(\Sigma, q)|$ is the set of Σ-states q' such that for all $s \in q$, $q' \models \langle s \rangle \top$. The intuition is that q is a form of record typing over Σ that allows us to simplify clauses such as

$$q' \models \langle s \rangle \varphi \quad \Longleftrightarrow \quad s \in q' \text{ and } {q'}_s \models \varphi \tag{10}$$

when $s \in q \subseteq q'$. The second conjunct in the righthand side of (10), $q'_s \models \varphi$, presupposes the first conjunct, $s \in q'$. We can lift that presupposition out of (10) by asserting that whenever $s \in q$ and $q \subseteq q'$,

$$q' \models \langle s \rangle \varphi \quad \Longleftrightarrow \quad q'_s \models \varphi.$$

This comes close to the equivalence (9) in Fact 4, except that Σ-reducts are missing. These reducts appear once we vary Σ, and step from $Q(\Sigma)$ to $\int Q$. Taking this step, we turn the categories $Mod(\Sigma, q)$ to a functor Mod from $\int Q$ to **Cat**, mapping a $\int Q$-morphism $\sigma = ((\Sigma', \Sigma), (q' \restriction \Sigma, s))$ from (Σ', q') to (Σ, q) to the functor

$$Mod(\sigma) : Mod(\Sigma', q') \to Mod(\Sigma, q)$$

sending $q'' \in |Mod(\Sigma', q')|$ to the s-derivative of its Σ-reduct, $(q'' \restriction \Sigma)_s$, and a $Mod(\Sigma', q')$-morphism (q'', s') to the $Mod(\Sigma, q)$-morphism $(q'' \restriction \Sigma, \pi_\Sigma(s'))$.

The syntactic counterpart of $Q(\Sigma)$ is $sen(\Sigma)$, which we turn into a functor sen matching Mod. A basic insight from Goguen & Burstall 1992 informing the present approach is the importance of a category **Sign** of *signatures* which

the functor *sen* maps to the category **Set** of sets (and functions) and which *Mod* maps contravariantly to **Cat**. The definition of *Mod* above suggests that $\mathbf{Sign}^{op}$ is $\int Q$.[6] A $\int Q$-morphism from $\int Q$-objects (Σ', q') to (Σ, q) is determined uniquely by a string $s \in q' \restriction \Sigma$ such that

$$q = (q' \restriction \Sigma)_s \text{ and } \Sigma \subseteq \Sigma'. \tag{11}$$

Let $(\Sigma, q) \xrightarrow{s} (\Sigma', q')$ abbreviate the conjunction (11), which holds precisely if $((\Sigma', \Sigma), (q' \restriction \Sigma, s))$ is a $\int Q$-morphism from (Σ', q') to (Σ, q). Now for $(\Sigma, q) \in |\int Q|$, let

$$sen(\Sigma, q) \;=\; sen(\Sigma)$$

(ignoring q), and when $(\Sigma, q) \xrightarrow{s} (\Sigma', q')$, let

$$sen(\sigma) : sen(\Sigma) \to sen(\Sigma')$$

send $\varphi \in sen(\Sigma)$ to $\langle s \rangle \varphi \in sen(\Sigma')$. To see that an *institution* arises from restricting $\models$ to $|Mod(\Sigma, q)| \times sen(\Sigma)$, for $(\Sigma, q) \in |\int Q|$, it remains to check the *satisfaction condition*:

whenever $(\Sigma, q) \xrightarrow{s} (\Sigma', q')$ and $q'' \in |Mod(\Sigma', q')|$ and $\varphi \in sen(\Sigma)$,

$$q'' \models \langle s \rangle \varphi \iff (q'' \restriction \Sigma)_s \models \varphi.$$

This follows from Fact 4 above, as s must be in $q' \restriction \Sigma$ and thus also in $q'' \restriction \Sigma$.

5 Conclusion

A process perspective is presented in section 2 that positions frames to the left of a satisfaction predicate $\models$ for Hennessy-Milner logic over a set A of labels (or, from the perspective of Blackburn 1993, attributes). A is allowed to become arbitrarily large so that under identity of indiscernibles relative to A, a frame can be identified with a non-empty prefix-closed language over A. This identification is tried out in section 3 on frames as types and particulars. A handle on A is provided by its finite subsets Σ, which are paired with languages $q \subseteq \Sigma^*$ for signatures (Σ, q), along which to reduce satisfaction to Σ-reducts and/or s-derivatives, for $s \in q$ (Fact 4). This plays out themes (mentioned in the introduction) of "semantics in flux" and "natural languages as collections of resources" (from Cooper & Ranta) in that, oversimplifying somewhat, s-derivatives specify transitions, while Σ-reducts pick out resources to use. The prominence of transitions (labeled by A) here contrasts strikingly with a finite-state approach

[6] That said, we might refine **Sign**, requiring of a signature (Σ, q) that q be a regular language. For this, it suffices to replace $\int Q$ by $\int R$ where $R : Fin(\mathsf{A})^{op} \to \mathbf{Cat}$ is the subfunctor of Q such that $R(\Sigma)$ is the full subcategory of $Q(\Sigma)$ with objects regular languages. We can make this refinement with*out* requiring that Σ-states in $Mod(\Sigma, q)$ be regular, forming $Mod(\Sigma, q)$ from Q (not R).

to events (most recently described in Fernando 2015), where a single string (representing a timeline) appears to the left of a satisfaction predicate.[7] A Σ-state q to the left of $\models$ above offers a choice of strings, any number of which might be combined with other strings from other Σ'-states over different alphabets Σ'. A combination can be encoded as a string describing a timeline of resources used. This type/token distinction between languages and strings to the left of satisfaction has a twist; the languages are conceptually prior to the strings representing timelines, as nonexistent computer programs cannot run. Indeed, a profusion of alphabets Σ and Σ-states compete to make, in some form or other, a timeline that has itself a bounded signature (of a different kind). The processes through which a temporal realm is pieced together from bits of various frames call out for investigation.

While much remains to be done, let us be clear about what is offered above. A frame is structured according to strings of labels, allowing the set Σ of labels to vary over finite sets. That variation is tracked by a signature (Σ, q) picking out non-empty prefix-closed languages over Σ that contain the set q of strings over Σ. For example, Cooper's meaning function

$$\left(\lambda r : \begin{bmatrix} \text{AGENT} : all \\ \text{THEME} : all \end{bmatrix}\right) \quad \begin{bmatrix} p_1 : smash(r) \\ p_2 : animate(r.\text{AGENT}) \\ p_3 : concrete(r.\text{THEME}) \end{bmatrix}$$

is approximated by the signature (Σ, q) as the language L, where

$$\begin{aligned} \Sigma &= \{\text{AGENT},\ \text{THEME},\ smash,\ animate,\ concrete\} \\ q &= \{\text{AGENT},\ \text{THEME},\ \epsilon\} \\ L &= q \cup \{smash,\ \text{AGENT}\ animate,\ \text{THEME}\ concrete\}. \end{aligned}$$

Further constraints can be imposed through formulas built with Boolean connectives and modal operators dependent on Σ — for instance, $Nom(\langle a \rangle \top)$. The possibility of expanding Σ to a larger set makes the notion of identity as Σ-indiscernibility open-ended, and Σ a bounded but refinable level of granularity. A measure of satisfaction is taken in a finite-state calculus with, as Conway 1971 puts it, Taylor series

$$L = o(L) + \sum_{a \in \Sigma} aL_a$$

(from derivatives L_a), and a Grothendieck signature

$$\mathbf{Sign} = \int Q.$$

Acknowledgements. My thanks to Robin Cooper for discussions, to Glyn Morrill for help with presenting this paper at Formal Grammar 2015, and to two referees for comments and questions.

[7] This is formulated as an institution in Fernando 2014.

References

1. L Barsalou. Perceptual symbol systems. *Behavioral and Brain Sciences*, 22:577–660, 1999.
2. P. Blackburn. Modal logic and attribute value structures. In M. de Rijke, editor, *Diamonds and Defaults*, pages 19–65. Kluwer, 1993.
3. J. Brzozowski. Derivatives of regular expressions. *J. ACM*, 11(4):481–494, 1964.
4. J.H. Conway. *Regular Algebra and Finite Machines.* Chapman and Hall, 1971.
5. R. Cooper. Type theory and semantics in flux. In *Philosophy of Linguistics*, pages 271–323. North-Holland, 2012.
6. R. Cooper and A. Ranta. Natural languages as collections of resources. In *Language in Flux: Dialogue Coordination, Language Variation, Change and Evolution*, pages 109–120. College Publications, 2008.
7. D. Davidson. The logical form of action sentences. In *The Logic of Decision and Action*, pages 81–95. University of Pittsburgh Press, 1967.
8. T. Fernando. Incremental semantic scales by strings. In *Proc EACL 2014 Workshop on Type Theory and Natural Language Semantics*, pages 63–71. ACL, 2014.
9. T. Fernando. The semantics of tense and aspect: a finite-state perspective. In S. Lappin and C. Fox, editors, *The Handbook of Contemporary Semantic Theory.* John Wiley and Sons, second edition, 2015.
10. C. Fillmore. Frame semantics. In *Linguistics in the Morning Calm*, pages 111–137. Hanshin Publishing Co., Seoul, 1982.
11. J. Goguen and R. Burstall. Institutions: abstract model theory for specification and programming. *J. ACM*, 39(1):95–146, 1992.
12. M. Hennessy and R. Milner. Algebraic laws for non-determinism and concurrency. *J. ACM*, 32(1):137–161, 1985.
13. J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, 1979.
14. L. Kallmeyer and R. Osswald. Syntax-driven semantic frame composition in lexicalized tree adjoining grammars. *J. Language Modelling*, 1(2):267–330, 2013.
15. S. Löbner. Evidence for frames from human language. In *Frames and Concept Types: Applications in Language and Philosophy*, pages 23–67. Springer, 2014.
16. R. Muskens. Data semantics and linguistic semantics. In *The dynamic, inquisitive, and visionary life of $\phi, ?\phi$, and $\Diamond\phi$: A festschrift for Jeroen Groenendijk, Martin Stokhof, and Frank Veltman*, pages 175–183. Amsterdam, 2013.
17. R. Osswald. Semantics for attribute-value theories. In *Proc Twelfth Amsterdam Colloquium*, pages 199–204. Amsterdam, 1999.
18. W. Petersen and T. Osswald. Concept composition in frames: Focusing on genitive constructions. In *Frames and Concept Types: Applications in Language and Philosophy*, pages 243–266. Springer, 2014.
19. J. Rutten. Automata and coinduction (an exercise in coalgebra). In *Proc CONCUR '98*, pages 194–218. Springer LNCS 1266, 1998.
20. A. Tarlecki, R. Burstall, and J. Goguen. Some fundamental algebraic tools for the semantics of computation: Part 3. indexed categories. *Theoretical Computer Science*, 91:239–264, 1991.
21. F. Veltman. Logics for conditionals. PhD dissertation, University of Amsterdam, 1985.

Cyclic Multiplicative-Additive Proof Nets of Linear Logic with an Application to Language Parsing

Vito Michele Abrusci and Roberto Maieli

Department of Mathematics and Physics, Roma Tre University
Largo San Leonardo Murialdo, 1 – 00146 Rome, Italy
{abrusci,maieli}@uniroma3.it

Abstract. This paper concerns a logical approach to natural language parsing based on proof nets (PNs), i.e. de-sequentialized proofs, of linear logic (LL). It first provides a syntax for proof structures (PSs) of the cyclic multiplicative and additive fragment of linear logic (CyMALL). A PS is an oriented graph, weighted by boolean monomial weights, whose conclusions Γ are endowed with a cyclic order σ. Roughly, a PS π with conclusions $\sigma(\Gamma)$ is correct (so, it is a proof net), if any slice $\varphi(\pi)$, obtained by a boolean valuation φ of π, is a multiplicative (CyMLL) PN with conclusions $\sigma(\Gamma_r)$, where Γ_r is an additive resolution of Γ, i.e. a choice of an additive subformula for each formula of Γ. The correctness criterion for CyMLL PNs can be considered as the non-commutative counterpart of the famous Danos-Regnier (DR) criterion for PNs of the pure multiplicative fragment (MLL) of LL. The main intuition relies on the fact that any DR-switching (i.e. any correction or test graph for a given PN) can be naturally viewed as a seaweed, that is, a rootless planar tree inducing a cyclic order on the conclusions of the given PN. Unlike the most part of current syntaxes for non-commutative PNs, our syntax allows a sequentialization for the full class of CyMALL PNs, without requiring these latter to be cut-free.
One of the main contributions of this paper is to provide a characterization of CyMALL PNs for the additive Lambek Calculus and thus a geometrical (non inductive) way to parse sentences containing words with syntactical ambiguity (i.e., with polymorphic type).

1 Introduction

Proof nets (PNs) are one of the most innovative inventions of linear logic (LL, [7]): they are used to represent demonstrations in a geometric (i.e., non inductive) way, abstracting away from the technical bureaucracy of sequential proofs. Proof nets quotient classes of derivations that are equivalent up to some irrelevant permutations of inference rules instances.

In this spirit, we present a syntax for PNs of the cyclic multiplicative and additive fragment of linear logic (CyMALL, Sections 1.1). This syntax, like the original one of Girard [8], is based on weighted (by boolean monomials) proof

structures with explicit binary contraction links (Section 2). The conclusions Γ (i.e., a sequence of formula occurrences) of any PS are endowed with a cyclic order σ on Γ. Naively, a CyMALL PS π with conclusions $\sigma(\Gamma)$ is correct if, for any slice $\varphi(\pi)$, obtained by a boolean valuation φ of π, there exists an additive resolution (i.e., a multiplicative restriction of $\varphi(\pi)$) that is a CyMLL PN with conclusion $\sigma(\Gamma_r)$, where Γ_r is an additive resolution of Γ (i.e. a choice of an additive sub-formula for each formula of Γ). In turn, the correctness criterion for CyMLL PNs can be considered as the non-commutative counterpart of the famous Danos-Regnier (DR) criterion for proof nets of linear logic (see [5] and [6]). The main intuition relies on the fact that any *DR-switching for a PS* (i.e. any correction or test graph, obtained by mutilating one premise of each disjunction ∇-link) can be naturally viewed as a rootless planar tree, called a *seaweed*, inducing a *cyclic ternary relation* on the conclusions of the given proof structure.

Unlike some previous syntaxes for non-commutative logic, like e.g., [3] and [13], this new syntax admits a *sequentialization* (i.e., a correspondence with sequential proofs) for the full class of CyMLL PNs including those ones with cuts. Actually, the presence of cut links is "rather tricky" in the non-commutative case, since cut links are not equivalent, from a topological point of view, to tensor links (like in the commutative MLL case): indeed, tensor links make new conclusions appear that may disrupt the original (i.e., in presence of cut links) order of conclusions.

CyMALL PNs satisfy a simple (lazy) convergent cut-elimination procedure (Section 2.2) in Laurent-Maieli's style ([12]): our strategy relies on the notion of *dependency graph of an eigen weight p* (Definition 9), that is, the smallest $\&_p$-box that must be duplicated in a commutative $\&_p/C$-cut reduction step ([14]). Moreover, cut-reduction preserves PNs sequentialization (Section 2.3).

CyMALL can be considered as a conservative classical extension of Lambek Calculus (LC, see [11], [1] and [17]) one of the ancestors of LL. The LC represents the first attempt of the so called *parsing as deduction*, i.e., parsing of natural language by means of a logical system. Following [4], in LC, parsing is interpreted as type checking in the form of theorem proving of Gentzen sequents. Types (i.e. propositional formulas) are associated to words in the lexicon; when a string $w_1...w_n$ is tested for grammaticality, the types $t_1, ..., t_n$ associated with the words are retrieved from the lexicon and then parsing reduces to proving the derivability of a one-sided sequent of the form $\vdash t_n^{\perp}, ..., t_1^{\perp}, s$, where s is the type associated with sentences. Moreover, forcing constraints on the Exchange rule, by e.g. allowing only *cyclic permutations* over sequents of formulas, gives the required computational control needed to view theorem proving as parsing in Lambek Categorial Grammar style. Anyway, LC parsing presents some syntactical ambiguity problems; actually, there may be:

1. (*non canonical proofs*) more than one cut-free proof for the same sequent;
2. (*lexical polymorphism*) more than one type associated with a single word.

Now, proof nets are commonly considered an elegant solution to the first problem of representing canonical proofs; in this sense, in Section 3, we give an *embedding* of extended MALL Lambek Calculus into Cyclic MALL PNs. Concerning the second problem, in Section 4, we propose a parsing approach based on CyMALL

PNs that could be considered a step towards a proof-theoretical solution to the problem of lexical polymorphism. Technically, CyMALL proof nets allow to manage formulas superposition (types polymorphism) by means of the additive &-links, or dually, $\oplus$-links. By means of Lambek CyMALL PNs, we propose the parsing of some sentences, suggested by [18], which make use of polymorphic words; naively, when a word has two possible formulas A and B assigned, then we can combine (or super-pose) these into a single additive formula $A\&B$.

1.1 The Cyclic MALL Fragment of Linear Logic

We briefly recall the necessary background of the *Cyclic MALL fragment* of LL, denoted *CyMALL*, without units (see [1]). We arbitrarily assume literals $a, a^\perp, b, b^\perp, ...$ with a polarity: *positive* $(+)$ for atoms, $a, b, ...$ and *negative* $(-)$ for their duals $a^\perp, b^\perp ...$. A *formula* is built from literals by means of the two groups of connectives: *negative*, ∇ ("par") and & ("with") and *positive*, $\otimes$ ("tensor") and $\oplus$ ("plus"). For these connectives we have the following De Morgan laws: $(A \otimes B)^\perp = B^\perp \nabla A^\perp$, $(A \nabla B)^\perp = B^\perp \otimes A^\perp$, $(A\&B)^\perp = B^\perp \oplus A^\perp$, $(A \oplus B)^\perp = B^\perp \& A^\perp$. A CyMALL (resp., CyMLL) proof is any derivation tree built by the following (resp., by only *identities and multiplicative*) inference rules where sequents Γ, Δ are sequences of formula occurrences endowed with a *total cyclic order* (or *cyclic permutation*).

$$\textit{identities}: \quad \frac{}{\vdash A, A^\perp}\ axiom \qquad \frac{\vdash \Gamma, A \qquad A^\perp \Delta}{\vdash \Gamma, \Delta}\ cut$$

$$\textit{multiplicatives}: \quad \frac{\vdash \Gamma, A \qquad \vdash B, \Delta}{\vdash \Gamma, A \otimes B, \Delta}\otimes \qquad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \nabla B}\nabla$$

$$\textit{additives}: \quad \frac{\vdash \Gamma, A \qquad \vdash \Gamma, B}{\vdash \Gamma, A\&B}\& \qquad \frac{\vdash \Gamma, A_i}{\vdash \Gamma, A_1 \oplus_i A_2}\oplus_{i=1,2}$$

A *total cyclic order* can be thought as follows; consider a set of points of an oriented circle; the orientation induces a total order on these points as follows: if a, b and c are three distinct points, then b is either between a and c $(a < b < c)$ or between c and a $(c < b < a)$. Moreover, $a < b < c$ is equivalent to $b < c < a$ or $c < a < b$.

Definition 1 (total cyclic order). *A* total cyclic order *is a pair* (X, σ) *where* X *is a set and* σ *is a ternary relation over* X *satisfying the following properties:*

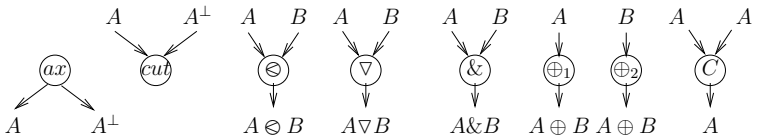
1. $\forall a, b, c \in X, \sigma(a, b, c) \rightarrow \sigma(b, c, a)$ (cyclic),
2. $\forall a, b \in X, \neg\sigma(a, a, b)$ (anti-reflexive),
3. $\forall a, b, c, d \in X, \sigma(a, b, c) \wedge \sigma(c, d, a) \rightarrow \sigma(b, c, d)$ (transitive),
4. $\forall a, b, c \in X, \sigma(a, b, c) \vee \sigma(c, b, a)$ (total).

Negative (or *asynchronous*) connectives correspond to *true determinism* in the way we apply bottom-up their corresponding inference rules. In particular, observe that Γ must appear as the same context (with the same order) in both premises

of the &-rule. Positive (or *synchronous*) connectives correspond to *true non-determinism* in the way we apply, bottom-up, their corresponding rules; there is no deterministic way to split, bottom up, the context (Γ, Δ) in the $\otimes$-rule; similarly, there not exist a deterministic way to select, bottom up, $\oplus_1$ or $\oplus_2$-rule.

2 Cyclic MALL Proof Structures

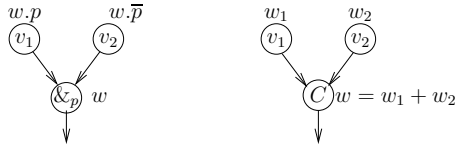
Definition 2 (CyMALL proof structure). *A* CyMALL proof structure (PS) *is an oriented graph π whose edges (resp., nodes) are labeled by formulas (resp., by connectives) of CyMALL and built by juxtaposing the following special graphs, called* links*, in which incident (resp., emergent) edges are called* premises *(resp.,* conclusions*):*



In a PS each premise (resp., conclusion) of a link must be conclusion (resp., premise) of exactly (resp., at most) one link. We call conclusion of a PS *any emergent edge that is not premises of any link. We call* CyMLL proof structure*, any PS built by only means of axioms, cut and multiplicative links ($\otimes, \nabla$).*

Definition 3 (Girard CyMALL proof structure). *A* Girard proof structure (GPS) *is a PS with* weights *associated as follows (a* weights assignment*):*

1. *first we associate a* boolean variable*, called* eigen weight *p, to each &-node (eigen weights are supposed to be different);*
2. *then we associate a* weight*, a product of (negation of) boolean variables $(p, \overline{p}, q, \overline{q}...)$ to each node, with the constraint that two nodes have the same weight if they have a common edge, except when the edge is the premise of a & or C-node; in these cases we do like below:*



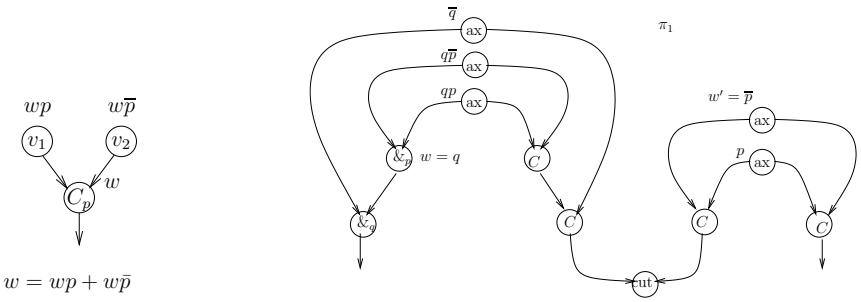
if p does not occur in w with $w_1.w_2 = 0$

3. *a* conclusion node *has weight* 1*;*
4. *if w is the weight of a &-node, with eigen weight p, and w' is a weight depending on p and appearing in the proof structure then $w' \leq w$.*

A weight w *depends on* the eigen weight p if p or $\bar{p}$ occurs in w. A node L with weight w *depends on* the eigen weight p if w depends on p or L is a C-node and one of the weights just above it depends on p.

Remark 1. Observe that:

1. since weights associated to a PS are products (*monomials*) of the Boolean algebra generated by the eigen weights associated to a proof structure, then, for each weight w associated to a binary contraction node, there exists a unique eigen weight p that *splits* w into $w_1 = wp$ *and* $w_2 = w\overline{p}$. We sometimes index a C-link with its *toggling variable* p, see the next left hand side picture;
2. the graph π_1 (the next r.h.s. picture) is not a GPS since it violates condition 4 of Definition 3; indeed, if $w = q$ is the weight of the $\&_p$-link and $w' = \overline{p}$ is a weight depending on p and appearing in the proof-structure then $\overline{p} \not\leq q$.



2.1 Correctness

Definition 4 (slices, switchings, resolutions). *Let π be a CyMALL GPS.*

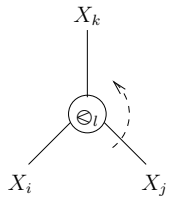
- *A* valuation *φ of π is a function from the set of all weights of π into $\{0, 1\}$.*
- *Fixed a valuation φ for π, the* slice *$\varphi(\pi)$ is the graph obtained from π by keeping only those nodes with weight* 1 *together with their incident edges.*
- *Fixed a slice $\varphi(\pi)$ a* multiplicative switching *S for π is the non-oriented graph $S^m_\varphi(\pi)$ built on the nodes and edges of $\varphi(\pi)$ with the modification that for each ∇-node we take only one premise (*left/right switch*).*
- *Fixed a slice $\varphi(\pi)$ an* additive switching*, denoted $S^a_\varphi(\pi)$ is a multiplicative switching $S^m_\varphi(\pi)$ for π, in which for each $\&_p$-node we erase the (unique) premise in $\varphi(\pi)$ and we add an oriented edge, called* jump*, from the $\&_p$-node to an link L whose weight depends on the eigen weight p.*
- *An* additive resolution *$\varphi_r(\pi)$ for a slice $\varphi(\pi)$ is the graph obtained by replacing in $\varphi(\pi)$ each unary link L (a link that, possibly, after the valuation has a single premise) by a single edge that is the (unique) premise of L. In particular, each conclusion of $\varphi_r(\pi)$ will be labeled by a multiplicative (CyMLL) formula.*

We call *additive resolution of a CyMALL sequent Γ* what remains of Γ after deleting one of the two sub-formulas in each additive (sub)formula of Γ.

In the following we characterize, by a *correctness criterion*, those CyMALL GPSs corresponding to proofs. This correctness criterion (Definition 7) is defined in terms of the correctness of CyMLL PSs (Definition 6). There exist several syntaxes for CyMLL proof nets; here we adopt the syntax of [2] inspired to [13].

Definition 5 (seaweeds). *Assume π is a CyMLL PS with conclusions Γ and assume $S(\pi)$ is an acyclic and connected multiplicative switching for π; $S(\pi)$ is the rootless planar tree whose nodes are labeled by $\invamp$-nodes, and whose leaves $X_1, ..., X_n$ (with $\Gamma \subseteq X_1, ..., X_n$) are the terminal (pending) edges of $S(\pi)$; $S(\pi)$ is a ternary relation, called a* seaweed*, with* support *$X_1, ..., X_n$; an ordered triple (X_i, X_j, X_k) belongs to the seaweed $S(\pi)$ iff:*

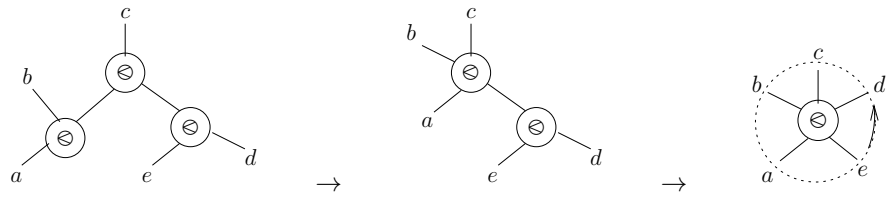
- *the intersection of the three paths X_iX_j, X_jX_k and X_kX_i is the node $\invamp_l$;*
- *the three paths $X_i\invamp_l, X_j\invamp_l$ and $X_k\invamp_l$ are in this cyclic order while moving anti-clockwise around the $\invamp_l$-node as below.*



If A is an edge of the seaweed $S(\pi)$, then $S_i(\pi)\downarrow^A$ is the restriction of the seaweed *$S(\pi)$, that is, the sub-graph of $S(\pi)$ obtained as follows:*

1. *disconnect the graph below (w.r.t. the orientation of π) the edge A.*
2. *delete the graph not containing A.*

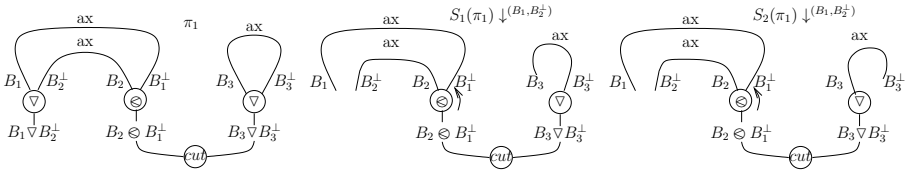
Fact 1 (seaweeds as cyclic orders) *Any seaweed $S(\pi)$ can be viewed as a cyclic total order (Definition 1) on its support $X_1, ..., X_n$; in other words, if a triple $(X_i, X_j, X_k) \in S(\pi)$, then $X_i < X_j < X_k$ are in cyclic order. Intuitively, we may contract a seaweed (by associating the $\invamp$-nodes) until it collapses into single n-ary $\invamp$-node with n pending edges (its support), like in the example below.*



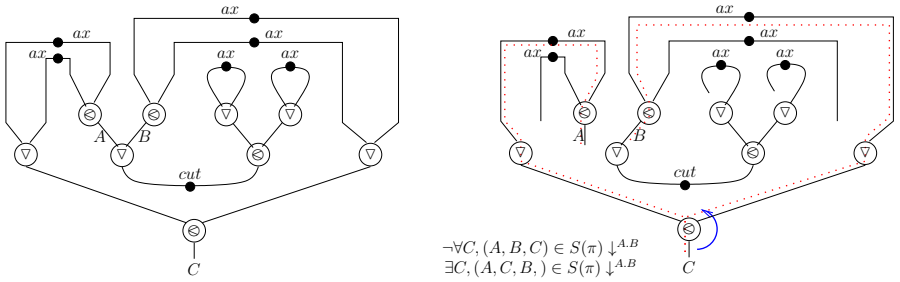
Definition 6 (CyMLL proof net). *A CyMLL PS π is* correct*, i.e. it is a CyMLL proof net (PN), iff:*

1. *π is a standard MLL PN, that is, any switching $S(\pi)$ is a connected and acyclic graph (therefore, $S(\pi)$ is a seaweed);*
2. *for any $\parr$-link $\frac{A \quad B}{A \parr B}$ the triple (A, B, C) must occur in this cyclic order in any seaweed $S(\pi)$ restricted to A, B, i.e., $(A, B, C) \in S(\pi)\downarrow^{(A,B)}$, for all pending leaves C (if any) in the support of the restricted seaweed.*

Example 1 (CyMLL PSs). We give below an instance of CyMLL PN π_1 with its two restricted seaweeds, $S_1(\pi_1)\downarrow^{(B_1,B_2^\perp)}$ and $S_2(\pi_1)\downarrow^{(B_1,B_2^\perp)}$, both satisfying condition 2 of Def. 6.



Mellies's counter-example. Observe that, unlike what happens in the commutative MLL case, the presence of cut links is "quite tricky" in the non-commutative case, since cut links are not equivalent, from a topological point of view, to tensor links: these latter make appear new conclusions that may disrupt the original (i.e., in presence of cut links) order of conclusions. In particular, the *Mellies's proof structure*[1] below (see page 224 of [17]) is not correct according to our correctness criterion since there exists a $\frac{A \quad B}{A \nabla B}$ link and a switching $S(\pi)$ s.t. $\neg\forall C, (A, B, C) \in S(\pi)\downarrow^{(A,B)}$, contradicting condition 2 of Definition 6: following the crossing dotted lines in the next r.h.s. figure, you can easily verify $\exists C$ pending s.t. $(A, C, B) \in S(\pi)\downarrow^{(A,B)}$.

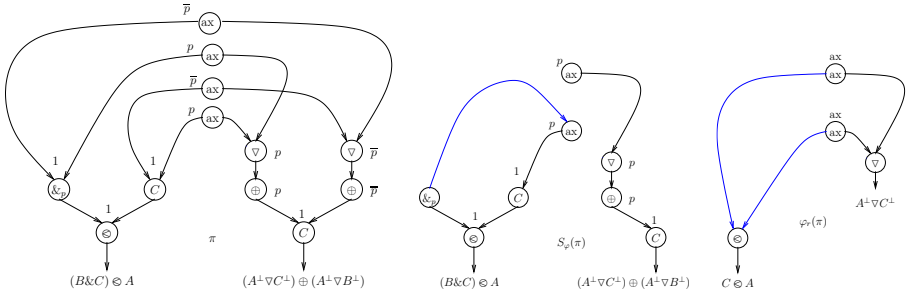


Definition 7 (CyMALL proof net). *We call* correct *(or* proof net, GPN*) any CyMALL GPS π s.t., its conclusions Γ are endowed with a cyclic order $\sigma(\Gamma)$ and for any valuation φ of π:*

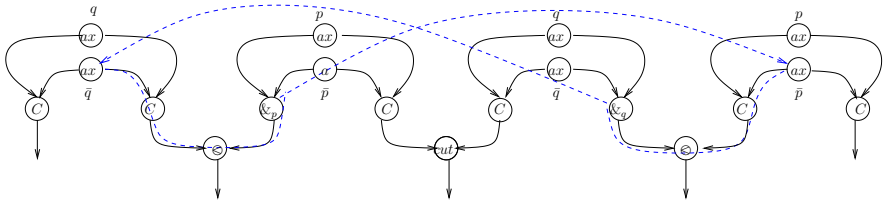
1. *each additive switching $S_\varphi(\pi)$ is an acyclic and connected graph (ACC);*
2. *there exists an additive resolution $\varphi_r(\pi)$ for $\varphi(\pi)$ that is a CyMLL PN with cyclic order conclusions $\sigma(\Gamma_r)$, where Γ_r is an additive resolution of Γ.*

Example 2 (CyMALL GPSs). Observe that the following proof structure π, on the left hand side, is not correct: actually, fixed a valuation φ s.t. $\varphi(p) = 1$, there exists an additive switching $S_\varphi(\pi)$ (with a jump) that is not ACC (see the center side figure). Nevertheless, any slice $\varphi(\pi)$ is ACC; for each slice $\varphi(\pi)$ there exists indeed an additive resolution $\varphi_r(\pi)$ that is a CyMLL PN like that one, on the rightmost hand side, with conclusions $C \otimes A, A^\perp \nabla C^\perp$. Observe, $\Gamma_r = (C \otimes A, A^\perp \nabla C^\perp)$ is an additive resolution of the conclusion of π, $\Gamma = (B \& C) \otimes A, (A^\perp \nabla C^\perp) \oplus (A^\perp \nabla B^\perp)$.

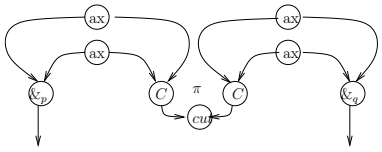
[1] This PS is considered as "a measure of the satisfiability degree" of correctness criteria of non-commutative logic: any "good" criterion should recognize this PS as uncorrect.



Similarly, the proof structure below is not correct: you can easily get an additive switching with a cycle like that one in (blue) dashed line.



Finally, the proof structure below is correct.

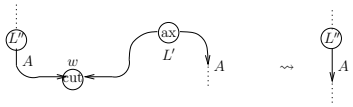


There currently exist other syntaxes for MALL PNs like the recent one by Hughes-van Glabbeek ([10]). Unlike the Girard's one, this new syntax only works with "uniform proof structures", i.e., proof structures with only η-expanded axioms and with contraction links only immediately below the axiom links.

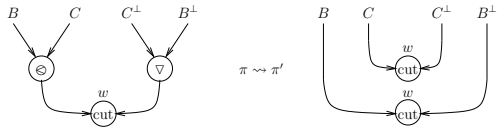
2.2 Cut Reduction

Definition 8 (ready cut reduction). *Let L be a cut link in a proof net π whose premises A and $A^\perp$ are conclusions of, resp., links L' and L'' with both of these different from contraction C. Then we define the result π' (called,* reductum*) of reducing a* ready cut *in π (called,* redex*), as follows:*

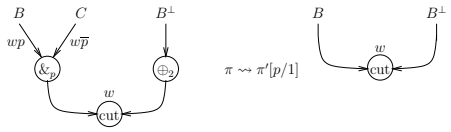
***Ax*-cut:** *if L' (resp., L'') is an axiom link then π' is obtained by removing in π both formulas $A, A^\perp$ (as well as L) and giving to L'' (resp., to L') the other conclusion of L' (resp., L'') as new conclusion:*



($\parr/\nabla$)**-cut:** *if L' is a $\parr$-link with premises B and C and L'' is a ∇-link with premises $C^\perp$ and $B^\perp$, then π' is obtained by removing in π both formulas A and $A^\perp$ as well as the cut link L together with L' and L'' and by adding two new cut links with, resp., premises B, $B^\perp$ and $C, C^\perp$, as follows:*



($\&/\oplus$)**-cut:** *if L' is a $\&_p$-link with premises B and C and L'' is a $\oplus_2$-link (resp., a $\oplus_1$-link) with premise $B^\perp$ (resp., $C^\perp$), then π' is obtained in three steps: first remove in π both formulas A, $A^\perp$ as well as the cut link L with L' and L'', then replace the eigen weight p by 1 (resp., p by 0) and keep only those links (vertexes and edges) that still have non-zero weight; finally we add a cut between B and $B^\perp$ (resp., between C and $C^\perp$) as below.*



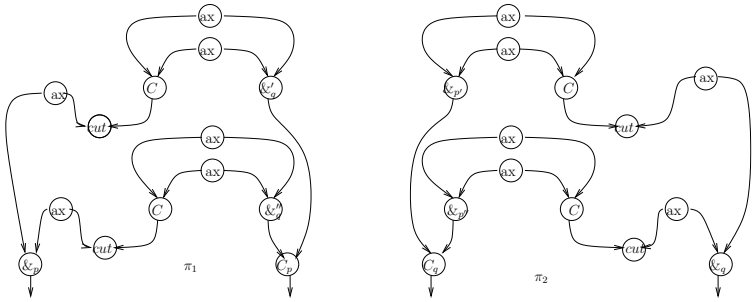
Theorem 1 (stability of GPN under ready cut reduction). *Assume π is a GPN that reduces to π' in one step of ready cut reduction, then π' is a GPN.*

Proof. Stability of condition 1 of Definition 6 and condition 1 of Definition 7, under ready cut reduction, follows as a consequence of the next graph theoretical property (see pages 250-251 of [9]):

Property 1 (Euler-Poicaré invariance). Given a graph $\mathcal{G}$, then $(\sharp CC - \sharp Cy) = (\sharp V - \sharp E)$, where $\sharp CC$, $\sharp Cy$, $\sharp V$ and $\sharp E$ denotes the number of, respectively, connected components, cycles, vertexes and edges of $\mathcal{G}$.

Meanwhile, stability of condition 2 of Definition 6 (resp., condition 2 of Definition 7) follows simply by calculation.

The confluence problem - Reducing a cut involving a contraction link as (at least) one of its premises may lead to different reductum, depending on which sub-graph of the redex we decide to duplicate. For instance, as illustrated below, reducing the commutative cut of the last proof net of Example 2 leads either to π_1 or to π_2 (in the next picture), depending on which additive box, $\&_q$ or $\&_p$, we decide to duplicate. There is no a-priori way to make π_1 and π_2 "equal". Girard, in [8], did not give a solution for this problem which has been later provided by Laurent and Maieli in [12]. Here we present an original lazy commutative cut reduction that simplifies the latter: technically, our reduction relies on the notion of *dependency graph* (Definition 9), i.e. the smallest &-box needed for duplication (see [14]). This cut reduction procedure preserves the notion of GPN (Theorem 1 and Theorem 2) and it is strong normalizing (Theorem 3 and Theorem 4).

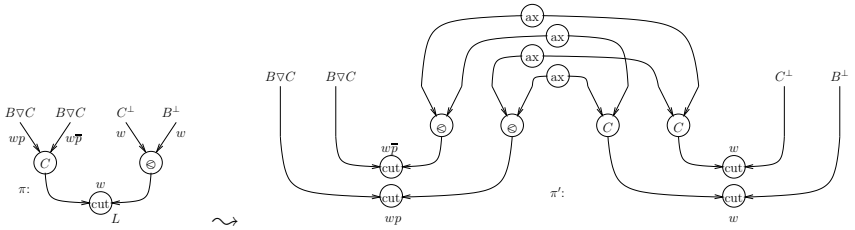


Definition 9 (empire and spreading). *Assume a proof structure π, an eigen weight p and a weight w, then:*

- *the* dependency graph *of p (w.r.t. π), denoted $\mathcal{E}_p$, is the (possibly disconnected) subgraph of π made by all links depending on p;*
- *the* spreading *of w over π, denoted by $w.[\pi]$, is the product of w for π, i.e., π where we replaced each weight v with the product of weights vw.*

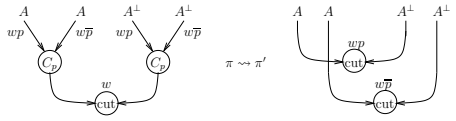
Definition 10 (commutative cut reduction). *Let L be cut link in a proof net π whose premises A and $A^\perp$ are the respective conclusions of links L' and L'' s. t. at least one of them is a contraction link C. Then we define the result π' (*reductum*) of reducing this* commutative *cut L in π (*redex*), as follows:*

$(C/\otimes)$-cut: *if L' is a C-link and L'' is a $\otimes$-link, then π reduces in one $(C/\otimes)$ step to π' (the (C/∇) step is analogous) as follows:*

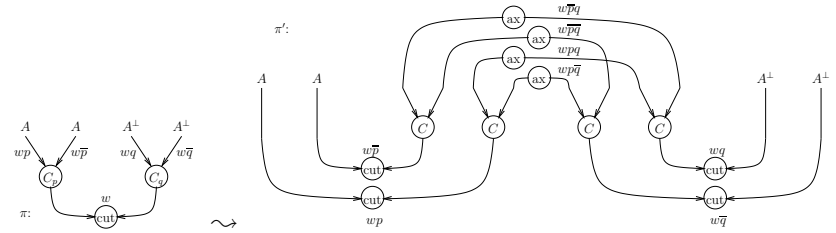


(C/C)-cut: *if both L' and L'' are C-links, then there are two cases:*

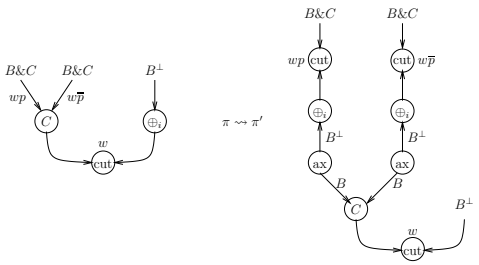
1. *either the weight w of both L' and L'' splits on the same p variable, then π reduces in one (C_p/C_p) step to π' as follows*



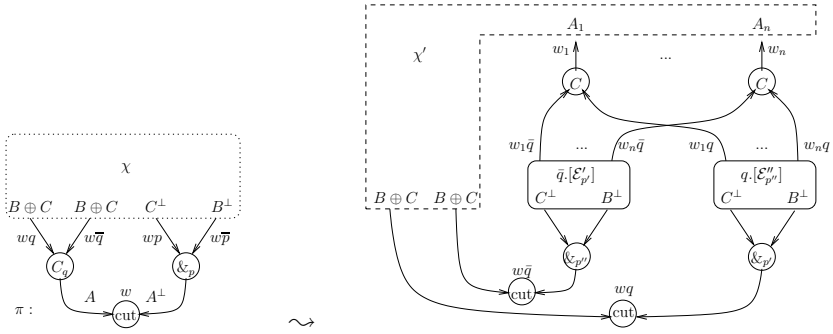
2. *or the weight w of L' (resp., w of L'') splits on p (resp., on q) then π reduces in one (C_p/C_q) step to π' as follows*



$(C/\oplus_i)$**-cut:** *if L' is a C-link and L'' a $\oplus_{i=1,2}$-link, then π reduces in one $(C/\oplus)$ step to π' as follows*



$(C/\&)$**-cut:** *if L' is a C-link and L'' a $\&_p$-link, then π reduces in one $(C/\&)$ step to π' as follows*



with the assumptions that graphs $\bar{q}.[\mathcal{E}'_{p'}]$ and $q.[\mathcal{E}''_{p''}]$ are obtained as follows:

1. *we take two copies, $\mathcal{E}'_p$ and $\mathcal{E}''_p$, of the dependency graph $\mathcal{E}_p$ of p;*
2. *we replace in $\mathcal{E}'_p$ (resp., in $\mathcal{E}''_p$) p with a new variable p' (resp., p'');*
3. *we spread $\bar{q}$ (resp., q) over $\mathcal{E}'_{p'}$ (resp., over $\mathcal{E}''_{p''}$).*

Technical details of proofs of Theorems 2, 3 and 4 can be found in [14].

Theorem 2 (stability). *If π is a GPN that reduces to π' in one step of commutative cut reduction then π' is a GPN too.*

Theorem 3 (termination). *We can always reduces a proof net π into a proof net π' that is cut-free, by iterating the reduction steps of Definitions 8 and 10.*

Theorem 4 (confluence). *Assume π is a proof net s.t. it reduces in one step α to π' ($\pi \rightsquigarrow_\alpha \pi'$) and it reduces in an other step β to π'' ($\pi \rightsquigarrow_\beta \pi''$); then, there exists a proof net σ such that both π' reduces, in a certain number of steps, to σ ($\pi' \rightsquigarrow^* \sigma$) and π'' reduces, in a certain number of steps, to σ ($\pi'' \rightsquigarrow^* \sigma$).*

2.3 Sequentialization

There exists a correspondence, called *sequentialization* (Theorem 5), between PNs and sequential proofs.

Lemma 1 (splitting). *Let π be a CyMLL PN with at least a $\otimes$-link or cut-link and with conclusions Γ not containing any terminal ∇-link (so, we say π is in* splitting condition*); then, there must exist a $\otimes$-link $\frac{A \quad B}{A \otimes B}$ (resp., a cut-link $\frac{A \quad A^{\perp}}{}$) that splits π in two CyMLL PNs, π_A and π_B (resp., π_A and $\pi_{A^{\perp}}$).*

Proof. Consequence of the Splitting Lemma for commutative MLL PNs ([7]).

Lemma 2 (PN cyclic order conclusions). *Let π be a CyMLL PN with conclusions Γ, then all seaweeds $S_i(\pi) \downarrow^{\Gamma}$, restricted to Γ, induce the same cyclic order σ on Γ, denoted $\sigma(\Gamma)$ and called* the (cyclic) order of the conclusions of π.

Proof. By induction on the size $\langle \sharp V, \sharp E \rangle$[2] of π.

Corollary 1 (stability of PN order conclusions under cut reduction). *If π, with conclusions $\sigma(\Gamma)$, reduces in one step of cut reduction to π', then also π' has conclusions $\sigma(\Gamma)$.*

Theorem 5 (sequentialization of CyMLL PNs). *Any CyMLL PN with conclusions $\sigma(\Gamma)$ can be* sequentialized *into a CyMLL sequent proof with the same cyclic order conclusions $\sigma(\Gamma)$.*

Proof. by induction on the size of the given proof net π via Lemmas 1 and 2.

Theorem 6 (sequentialization of CyMALL PNs). *A CyMALL GPN with conclusions $\sigma(\Gamma)$ can be* sequentialized *into a CyMALL sequent proof with the same cyclic conclusions $\sigma(\Gamma)$ and vice-versa (*de-sequentialization*).*

Proof. There are two parts.

Sequentialization-part. Any CyMALL proof net π can be sequentialized into a proof Π, by induction on the number of &-links. The base of induction corresponds to the sequentialization of the CyMLL proof nets (Theorem 5). The induction step follows by the sequentialization of standard MALL PNs (see [8]) where the only novelty is to show that: if a PN π contains a terminal &-link L, then π can be *toggled*[3] at L in two PNs preserving conditions 1 and 2 of Definition 7.

De-sequentialization-part. Any CyMALL proof Π of $\sigma(\Gamma)$ can be de-sequentialized into a PN π of $\sigma(\Gamma)$, by induction of the height of Π derivation.

Unlike most part of correctness criteria for non-commutative proof nets, like [3] and [13], our syntax enjoys a sequentialization for the full class of CyMLL PNs (with possible cuts). Observe that, *Mellies's counter-example* (Example 1) represents a non-sequentializable proof structure that becomes correct (therefore, sequentializable) after cut reduction.

[2] Number of vertexes and number of edges.

[3] We say that a terminal $\&_p$-link of a GPN π is *toggling* when the restriction of π w.r.t. p and the restriction of π w.r.t. $\bar{p}$ are both correct GPSs. We call the *restriction of π w.r.t. p* (resp., *w..r.t. $\bar{p}$*) what remains of π when we replace p with 1 (resp., $\bar{p}$ with 1) and keep only those vertexes and edges whose weights are still non-zero (see [8]).

3 Embedding Lambek Calculus into CyMALL PNs

Definition 11 (Lambek formulas and sequents of CyMALL). *Assume A and S are, respectively, a formula and a sequent of CyMALL.*

1. *A is a* pure Lambek formula (pLF) *if it is a CyMLL formula recursively built according to this grammar: $A := \text{positive atoms} \mid A \otimes A \mid A^{\perp} \triangledown A \mid A \triangledown A^{\perp}$.*
2. *A is an* additive Lambek formula (aLF or simply LF) *if it is a CyMALL formula recursively built according this grammar: $A := pLF \mid A \& A \mid A \oplus A$.*
3. *S is a* Lambek sequent of CyMALL *iff $S = (\Gamma^{\perp}, A)$, where A is a non-void LF and $\Gamma^{\perp}$ is a possibly empty finite sequence of negations of LFs (i.e., Γ is a possibly empty sequence of LFs and $\Gamma^{\perp}$ is obtained by taking the negation of each formula in Γ).*
4. *A* Lambek proof *is any derivation built by means of the CyMALL inference rules whose premise(s) and conclusions are CyMALL Lambek sequents.*

Definition 12 (Lambek proof net). *We call* Lambek CyMALL proof net *(resp.,* pure Lambek CyMLL proof net*) any CyMALL PN (resp., CyMLL PN) whose edges are labeled by LFs (resp. pLFs) or negation of LFs (resp., pLFs) and whose conclusions form a Lambek sequent.*

Corollary 2. *Any Lambek CyMALL proof net π is stable under cut reduction, i.e., if π reduces in one step to π', then π' is a Lambek CyMALL proof net too.*

Proof. Consequence of Theorems 1 and 2. Trivially, each reduction step preserves the property that each edge of the reductum is labeled by a Lambek formula or the negation of a Lambek formula.

Theorem 7 (de-sequentialization of Lambek CyMALL proofs). *Any proof of a CyMALL Lambek sequent $\vdash \sigma(\Gamma^{\perp}, A)$ can be de-sequentialized into a Lambek CyMALL PN with conclusions $\sigma(\Gamma^{\perp}, A)$.*

Proof. By induction on the height of the given sequent proof (similarly to the de-sequentialization part of Theorem 6).

Theorem 8 (sequentialization of Lambek CyMALL PNs). *Any Lambek CyMALL PN of $\sigma(\Gamma^{\perp}, A)$ can be sequentialized into a Lambek CyMALL proof of the sequent $\vdash \sigma(\Gamma^{\perp}, A)$.*

Proof. Sequentialization follows by induction on the number &-links of the given PN. The base of induction is given by next Theorem 9. The induction step, simply follows by Theorem 6.

Theorem 9 (sequentialization of pure Lambek CyMLL PNs). *Any Lambek CyMLL PN of $\sigma(\Gamma^{\perp}, A)$ can be sequentialized into a Lambek CyMLL proof of $\vdash \sigma(\Gamma^{\perp}, A)$.*

Proof. See details in [2].

4 Language Parsing with Lambek CyMALL PNs

In order to show how powerful PNs are, in this section we adapt to our syntax, some linguistics (typing) examples suggested by Richard Moot in his PhD thesis [18]. We use s, np and n as the types expressing, respectively, a *sentence*, a *noun phrase* and a *common noun*. According to the "parsing as deduction style", when a string $w_1...w_n$ is tested for grammaticality, the types $t_1, ..., t_n$ associated with the words are retrieved from the lexicon and then parsing reduces to proving the derivability of a two-sided sequent of the form $t_1, ..., t_n \vdash s$. Recall that proving a two sided Lambek derivation $t_1, ..., t_n \vdash s$ is equivalent to prove the one-sided sequent $\vdash t_n^\perp, ...t_1^\perp, s$ where $t_i^\perp$ is the dual (i.e., the linear negation) of each type t_i. Therefore, any phrase or sentence should be written like in a mirror (observing the opposite natural direction).

Assume the following lexicon, where *linear implication* $\multimap$ (resp., $\circ\!\!-$) is traditionally used for expressing types in two-sided sequent parsing style:

1. Vito $= np$;
2. Sollozzo $= np$;
3. him $= (s \circ\!\!- np) \multimap s \;=\; (s \nabla np^\perp)^\perp \nabla s \;=\; (np \parr s^\perp) \nabla s$;
4. trusts $= (np \multimap s) \circ\!\!- np \;=\; (np^\perp \nabla s) \nabla np^\perp$.

Cases of lexical ambiguity follow to words with several possible formulas A and B assigned it. For example, a verb like "*to believe*" can express a relation between two persons, np's in our interpretation, or between a person and a statement, interpreted as s, like in these examples:

(1) *Sollozzo believes Vito*; (2) *Sollozzo believes Vito trusts him.*

We can express this polymorphism by two lexical assignments as follows:

5. believes $= (np \multimap s) \circ\!\!- np \;=\; (np^\perp \nabla s) \nabla np^\perp$;
6. believes $= (np \multimap s) \circ\!\!- s \;=\; (np^\perp \nabla s) \nabla s^\perp$.

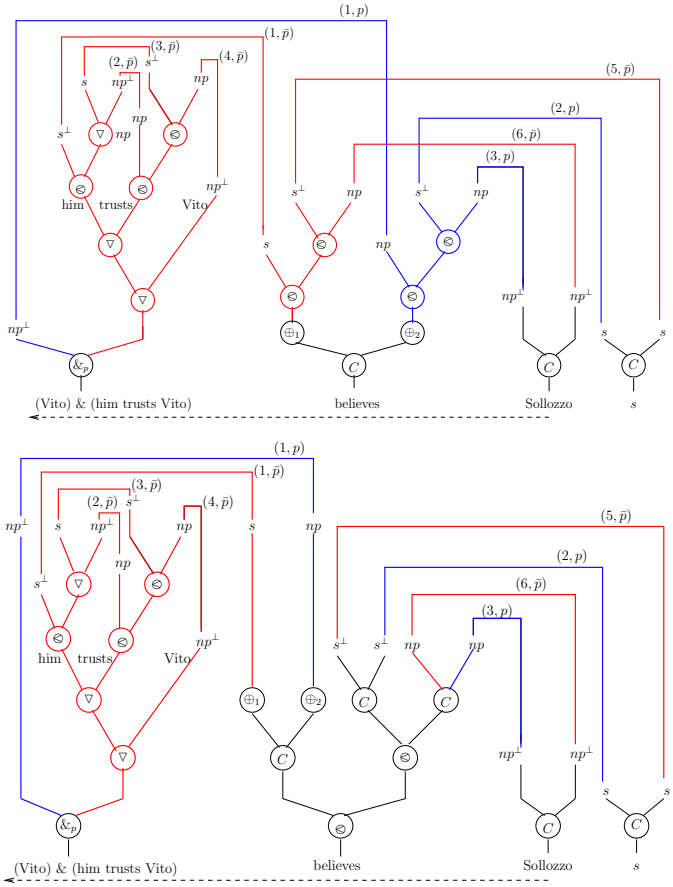
Typically, additives are used to capture cases of lexical ambiguity. When a word has two possible formulas A and B assigned it, we can combine these into a single additive formula $A \& B$ (resp., $A \oplus B$). Thus, we can collapse assignments 5 and 6 into the following single additive assignment:

7. believes $= ((np \multimap s) \circ\!\!- np) \& ((np \multimap s) \circ\!\!- s) = ((np^\perp \nabla s) \nabla np^\perp) \& ((np^\perp \nabla s) \nabla s^\perp)$.

Equivalently, via distributivity of negative connectives, we could also move the additive "inside" and generate a more compact lexical entry, in which the two assignments share their identical initial parts (see also [19] on type polymorphism):

8. believes $= ((np \multimap s) \circ\!\!- (s \oplus nps)) = ((np^\perp \nabla s) \nabla (np^\perp \& s^\perp))$.

Using that, we can then move lexical ambiguity into proof nets. In the following we give two equivalent Lambek PNs as parsing of the additive superposition of sentences (1) and (2); the first (resp., the second) PN makes use of the lexical entry 7 (resp., the lexical entry 8).



Observe that, for each slice of each proof net above, $\varphi_p(\pi)$ and $\varphi_{\bar{p}}(\pi)$, there exists an additive resolution that is a CyMLL PN with the same sequentialization:

$$\varphi_p(\pi) \;\Rightarrow\; \Pi_1 : \quad \dfrac{\dfrac{}{np^\perp, np}\, ax_1^p \qquad \dfrac{\dfrac{}{s^\perp, s}\, ax_2^p \qquad \dfrac{}{np, np^\perp}\, ax_3^p}{s^\perp \parr np, np^\perp, s}\,\parr}{np^\perp, np \otimes (s^\perp \parr np), np^\perp, s}\,\parr$$

$$\varphi_{\bar{p}}(\pi) \;\Rightarrow\; \Pi_2 : \quad \dfrac{\dfrac{\dfrac{}{s^\perp, s}\, ax_1^{\bar{p}} \qquad \dfrac{s, np^\perp, np \otimes (s^\perp \parr np), np^\perp}{s \nabla np^\perp, np \otimes (s^\perp \parr np), np^\perp}\,\nabla}{s^\perp \parr (s \nabla np^\perp), np \otimes (s^\perp \parr np), np^\perp, s}\,\parr \qquad \dfrac{\dfrac{\dfrac{}{np^\perp, np}\, ax_2^{\bar{p}} \qquad \dfrac{\dfrac{}{s^\perp, s}\, ax_3^{\bar{p}} \qquad \dfrac{}{np, np^\perp}\, ax_4^{\bar{p}}}{s^\perp \parr np, np^\perp, s}\,\parr}{np^\perp, np \otimes (s^\perp \parr np), np^\perp, s}\,\otimes \qquad \dfrac{\dfrac{}{s^\perp, s}\, ax_5^{\bar{p}} \qquad \dfrac{}{np, np^\perp}\, ax_6^{\bar{p}}}{s^\perp \parr np, np^\perp, s}\,\parr}{s^\perp \parr (s \nabla np^\perp), np \otimes (s^\perp \parr np), np^\perp, s \otimes (s^\perp \parr np), np^\perp, s}\,\otimes}{((s^\perp \parr (s \nabla np^\perp)) \nabla (np \otimes (s^\perp \parr np))) \nabla np^\perp, s \otimes (s^\perp \parr np), np^\perp, s}\,\nabla^2$$

5 Conclusions

As future work we aim at investigating the how topological correctness criteria based on graph rewriting (or retraction) of MALL proof structures ([6], [15]) may

be used for (linguistic) parsing. Naively, retraction criteria allow to switch from the paradigm of "parsing as deduction" to the paradigm of "parsing as rewriting" (see, e.g., [16]). Moreover, retraction could also be a useful computational tool for studying the complexity class of the CyMALL correctness criterion.

Acknowledgements. We thank the anonymous reviewers, Michael Moortgat and Richard Moot for their useful comments and suggestions. This work was partially supported by the PRIN Project *Logical Methods of Information Management*.

References

1. Abrusci, V. M.. Classical conservative extensions of Lambek calculus Studia Logica 71 (3):277 - 314 (2002).
2. Abrusci, V. M. and Maieli, R.. Cyclic multiplicative proof nets of linear logic with an application to language parsing. In: Proc. of WoLLIC 2015 Conference, July 20-23, 2015, Bloomington, USA. LNCS 9160, pp. 1-16, Springer-Verlag 2015.
3. Abrusci, V. M. and Ruet, P.. Non-commutative logic I: the multiplicative fragment. Annals of Pure and Applied Logic 101(1): 29-64, 2000.
4. Andreoli J.-M. and Pareschi, R.. From Lambek Calculus to word-based parsing. In *Proc. of Substructural Logic and Categorial Grammar Workshop*, Munchen, 1991.
5. Danos, V. and Regnier, L. The structure of multiplicatives. *Archive for Mathematical Logic*, 28:181-203, 1989,
6. Danos, V.. La Logique Linéaire appliquée à l'étude de divers processus de normalisation (principalment du λ-calcul). *PhD Thesis*, Paris, 1990.
7. Girard, J-Y.. Linear Logic. *Theoretical Computer Science*, 50:1-102, 1987
8. Girard, J-Y.. *Proof-nets: the parallel syntax for proof theory*. Logic and Algebra. Marcel Dekker, 1996.
9. Girard, J-Y.. Le point aveugle. *Cours de Logique. Volume I, Vers la Perfection*. Ed. Hermann, 2006, Paris.
10. Hughes, D. and van Glabbeek, R.. Proof Nets for unit-free multiplicative-additive linear logic. In *Proc. of IEEE LICS 2003*.
11. Lambek, J.. The mathematics of sentence structure. American Mathematical Montly, 65, 1958.
12. Laurent, L. and Maieli, R.. Cut elimination for monomial MALL proof nets. In Proc. of *IEEE LICS 2008*, pp 486-497, 2008. Pittsburgh, USA.
13. Maieli, R.. A new correctness criterion for multiplicative non-commutative proof-nets. Archive for Mathematical Logic, vol.42, 205-220, Springer-Verlag, 2003.
14. Maieli, R.. Cut elimination for monomial proof nets of the purely multiplicative and additive fragment of linear logic. IAC-CNR Report, n. 140 (2/2008). HAL Id: hal-01153910; available at `https://hal.archives-ouvertes.fr/hal-01153910`.
15. Maieli, R.. Retractile proof nets of the purely multiplicative and additive fragment of linear logic. In Proc. of *the 14th LPAR 2007*. Springer, LNAI 4790, pp. 363-377, 2007.
16. Maieli, R.. Construction of retractile proof structures. In: Proc. of Joint Conf. RTA-TLCA, July 14-17, 2014, Vienna. LNCS 8560, pp. 319-333, 2014. Springer.
17. Moot, R. and Retoré Ch.. The logic of categorial grammars: a deductive account of natural language syntax and semantics. Springer LNCS 6850, 2012.
18. Moot, R.. *Proof nets for linguistic analysis. PhD thesis*. Utrecht University, 2002.
19. Morrill, G.. Additive operators for polymorphism. In: *Categorial Grammar: logical syntax, semantics and processing*. Oxford University Press, 2011.

Algebraic Governance and Symmetry in Dependency Grammars

Carles Cardó

Universitat Politècnica de Catalunya
cardocarles@gmail.com

Abstract. We propose a purely algebraic approach to governance structure in dependency grammars aiming to capture all linguistic dependencies (such as morphological, lexico-semantic, etc.) in monoidal patterns. This provides a clear perspective and allows us to define grammars declaratively through classical projective structures. Using algebraic concepts the model is going to suggest some symmetries among languages.

Keywords: algebraic governance, dependency grammars, dependency structure, governance, symmetry, projective structure.

1 Preliminaries: Dependency Structures

Our model is based on an algebraic characterization of dependency trees (§2) and on an intuitive working hypothesis (§3). It consists in two objects: a *manifold* and a *linearization*. For reasons of space we can only address in this paper the construction of manifolds (§4). We show linearizations for specific cases but we do not develop the general mechanism. Fortunately manifolds and linearization are independent modules in our formalism. In §5 we implement some classical formal languages which encode certain cross-serial phenomena. In §6 we sketch how to build a manifold for a natural language. Using basic algebraic concepts, the model is going to suggest some symmetries of languages (§7) which will help us to understand our analysis. In the last section (§8) we show another equivalent interpretation of the presented results.

To connect our approach with the literature we now define some basic concepts in dependency grammars.

A *dependency structure* is a triplet $(D, \trianglelefteq, \leq)$, where $\trianglelefteq$ and $\leq$ are partial orders. The first relationship forms a tree (i.e. the Hasse diagram is a tree), the second relation forms a chain (i.e. the relationship is a total order and its Hasse diagram is a chain). The tree relation $\trianglelefteq$ is called *governance*; the total order $\leq$ is called *precedence*. To visualize a dependency structure we could draw both Hasse diagrams over the same set D, but it is more convenient to picture the Hasse diagram of the structures $(D, \trianglelefteq)$, $(D, \leq)$ separately and connect them with dashed lines, as in Figures 1a) and 1b). These lines are usually called *projection lines*.

The set D can be understood as a set of words, but since we frequently find sentences with repeated words we remember that we are really dealing with word

tokens. A dependency structure with words is a structure $(D, \trianglelefteq, \leq, f)$ such that $(D, \trianglelefteq, \leq)$ is a dependency structure and f is a mapping $f : D \longrightarrow \Sigma$, where Σ is a finite vocabulary.[1]

A dependency grammar is a finite description of a set $\mathbf{G}$ of dependency structures with words. The language of a dependency grammar is the same set of dependency structures without the governance relation, i.e. only the chains. If we want the language in Σ^* we can define the set:

$$\{f(x_1) \cdots f(x_n) \in \Sigma^* \mid \{x_1, \ldots, x_n\} = D,\ x_1 \leq \cdots \leq x_n \text{ and } (D, \trianglelefteq, \leq, f) \in \mathbf{G}\}.$$

There are several frameworks providing such description, for example *Lexicalized Grammars*, *Tree Adjoining Grammar*, *Regular Dependency Grammars* or *eXtensible Dependency Grammar*. See Debusmann and Kuhlmann, 2010[4] and Debusmann, Duchier and Kruijff, 2004[3].

Literature in dependency grammars focuses its efforts on explaining the relationship between governance and precedence. Since from the point of view of natural languages there are a lot of inadequate dependency structures some constraints must be imposed. The main one is called *projectivity*. A dependency structure is called projective iff each subtree of the governance structure is an interval on the precedence structure, as in Figure 1a) and 1c). Graphically classical projective structures are planar graphs (trees of governance) that can be embedded in the plane and be geometrically projected on a line (chain of precedence) without crossing lines of projection, see Figure 1a). Figure 1b) shows a non-projective structure. This is a geometrical characterization of projectivity, although there are others.

It is thought that projective structures cover over 75-80% of sentences of natural language (Debusmann and Kuhlman, 2010[4]). To explain the rest there are two options. Either we must assume a less intuitive analysis of the sentences, or we must loosen the sense of projectivity. For this reason other projections have been proposed, such as *weak non-projective structures* or *well-nested structures* (Kuhlman, 2013[6]).

2 Syntagmata and Manifolds

We make a few comments on notation. Given a finite set, A, we denote A^* the *free monoid generated by* A. The *length* of an $x \in A^*$ is denoted by $|x|$. We are going to use free monoids ζ^* and Σ^*. e is the identity of ζ^* and 0, the identity of Σ^*. We will use the product notation for both. Given two monoids Γ, Γ', the Cartesian product $\Gamma \times \Gamma'$ is newly a monoid, called the direct sum, with its operation defined componentwise. We abbreviate $\mathbb{N}^+ = \mathbb{N} \cup \{0\}$ and $\Sigma^+ = \Sigma \cup \{0\}$. We omit the brackets and write $\Sigma^{+n} = (\Sigma^+)^n = \Sigma^+ \times \cdots \times \Sigma^+$ n times. Equally $\zeta^{*n} = (\zeta^*)^n$. Given a set $A \subseteq \zeta^{*n}$, we denote $\langle A \rangle$ the minimum

[1] There are other definitions of dependency structures, such as those encoding structure as a graph or through *Robinson's axioms* (Debusmann 2000[2]), but the above is more concise.

Fig. 1a)

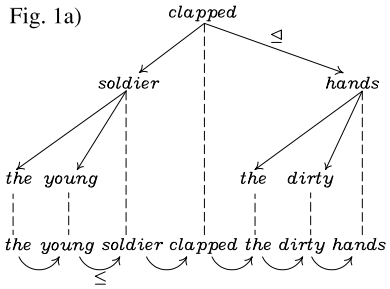


Fig. 1b)

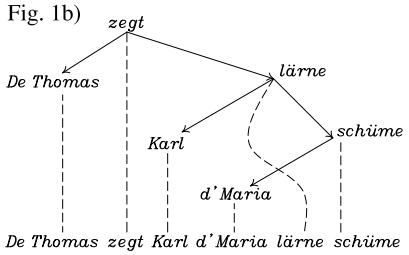


Fig. 1c)

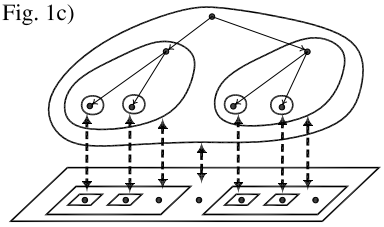
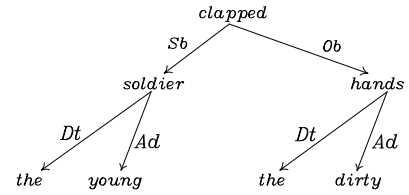


Fig. 1d)



submonoid in ζ^{*n} containing A. We say that a submonoid Γ is *finitely generated* iff there exists a finite subset $G \subseteq \Gamma$, called a *generator* set, such that $\langle G \rangle = \Gamma$.

Definition 1. *Let ζ be a finite set of syntactic functions, and Σ a finite set of words. We call a* syntagma *a mapping $S : \zeta^* \longrightarrow \Sigma^+$ such that*

i) S has finite order, i.e. $|\{x \in \zeta^* \,|\, S(x) \neq 0\}| < \infty$.
ii) S is non-elliptic, i.e. $S(x) = 0 \Longrightarrow S(yx) = 0, \ \forall x, y \in \zeta^*$.[2]

We call the elements of ζ^* loci. *We call the* support *of S the non-null loci, $Spt(S) = \{x \in \zeta^* \,|\, S(x) \neq 0\}$. We call the* order *the cardinality of the support, $|S| = |\{x \,|\, S(x) \neq 0\}|$. A locus x is a* leaf *in S iff $S(x) \neq 0$, but $S(yx) = 0$ for all $y \neq e, y \in \zeta^*$. We call the* depth *of S the number $d(S) = \max\{|x| \mid x \in \zeta^*$, such that x is a leaf in $S\}$.*

Some of the syntactic functions that we are going to use are: *Sb*, Subject function; *Ob*, Object; *In*, indirect object; *At*, Attribute; *Dt*, Determiner; *Ad*, Adjectival; *Av*, Adverbial; *Pr*, Prepositional; *Cc*, Circumstantial Complement (traditional name in Romance grammars to indicate time, place, ...); *Cr*, verbal regime complement (traditional name in Romance grammars to indicate a verb which needs a second verb). Nevertheless the names are not important: syntactic functions play roles only in relation to each other.

The locus $Dt \cdot Sb$, say, can be read as *the determiner of the subject*. The set of loci ζ^* can be interpreted as a set of addresses. Words can be repeated in a sentence but not loci, so invoking a word can be made unequivocally through

[2] An ellipsis occurs when a locus is null but there is underneath a non-null locus; otherwise the locus is definitively null. A class of *elliptic* syntagmata is also useful, but in order to simplify things we are not using them except in the last example.

the loci. The main benefit of syntagmata is in translating linguistic descriptions into algebraic descriptions.

Example 1. Let the syntactic functions be $\zeta = \{Sb, Ob, Dt, Ad\}$ and let the vocabulary be $\Sigma = \{\text{the, soldier, hands, clapped, dirty}\}$. Let S be the mapping $S : \zeta^* \longrightarrow \Sigma^+$ given by: $S(e) =$ clapped, $S(Sb) =$ soldier, $S(Ob) =$ hands, $S(Dt \cdot Sb) =$ the, $S(Ad \cdot Sb) =$ young, $S(Dt \cdot Ob) =$ the, $S(Ad \cdot Ob) =$ dirty, and $S(x) = 0$, otherwise. Then S is a syntagma with order $|S| = 8$ and depth $d(S) = 2$. The leaves are the loci $Dt \cdot Sb$, $Ad \cdot Sb$, $Dt \cdot Ob$ and $Ad \cdot Ob$.

The next proposition connects syntagmata with governance structures.

Proposition 1. *Given a syntagma S we call* algebraic governance *the relationship $(Spt(S), \trianglelefteq)$ such that $x \trianglelefteq y \iff \exists \varphi \in \zeta^*$ such that $x = \varphi \cdot y$. Algebraic governance is a governance relationship. So $(Spt(S), \trianglelefteq, S)$ is governance with words.*

Proof. On the one hand we see that $\trianglelefteq$ is a partial order as follows. Reflexivity: we have $x \trianglelefteq x, \forall x \in Spt(S)$, because $x = e \cdot x$. Antisymmetry: let x, y be in $Spt(S)$ such that $x \trianglelefteq y$ and $y \trianglelefteq x$, iff $x = x' \cdot y$ and $y = y' \cdot x$ for some $x', y' \in \zeta^*$. So $x = x' \cdot (y' \cdot x) = (x' \cdot y') \cdot x$. But free monoids enjoy the cancellation laws, therefore $e = x' \cdot y'$, and then $x' = y' = e$. So $x = y$. Transitivity: let x, y, z be in $Spt(S)$ such that $x \trianglelefteq y$ and $y \trianglelefteq z$, iff $x = x' \cdot y$ and $y = y' \cdot z$, then we compose both: $x = x' \cdot (y' \cdot z) \iff x = (x' \cdot y') \cdot z \iff x \trianglelefteq z$.

On the other hand we must prove that its Hasse diagram is a tree. This can be done by induction on the depth of adding new leaves to the syntagma. Finally we see that the mapping $S : \zeta^* \longrightarrow \Sigma^+$ can be restricted to the mapping $S : Spt(S) \longrightarrow \Sigma$, because the elements in $Spt(S)$ cannot be null, so we have the governance structure with words: $(Spt(S), \trianglelefteq, S)$. □

To represent graphically a syntagma we can use the Cayley digraph of a binary operation. The Cayley digraph of a free monoid ζ^* is the directed digraph (V, E) with vertexes $V = Spt(S)$ and edges $E = \{(x, \lambda x) \,|\, x \in \zeta^*, \lambda \in \zeta\}$; since the monoid is free the digraph is always tree-shaped. Then we label every edge $(x, \lambda x)$ with the function λ. Finally we just have to label the vertexes according the map S. See Figure 1d). Note that the Hasse diagram of the algebraic governance coincides with the Cayley digraph without labels of edges, so a locus is the labeling of a path from root to node.

Some frameworks in dependency grammars require another kind of underlying non-tree-shaped structure, such as tree acyclic graphs or trees with repeated syntactic functions in a same level like XDG (Debusmann, Duchier and Kruijff, 2004[3]), or Topological Grammar, (Duchier and Debusmann, 2001[5]). To accommodate this there is a natural generalization of our syntagmata based on the algebraic concept of action of a monoid on a set.[3]

[3] We cannot show here all the details, but a very short sketch follows. A *multiple action* is relationship $a \subseteq \zeta^* \times X \times X$ whose elements we notate $x \overset{\varphi}{\to} y$, $\varphi \in \zeta^*, x, y \in X$,

We notate $\mathbf{Synt}_{\zeta,\Sigma}$ the set of all syntagmata with ζ and Σ fixed. A manifold is a subset $W \subseteq \mathbf{Synt}_{\zeta,\Sigma}$. If we have a manifold and a procedure to assign a chain to each syntagma, called *linearization*, then we will have a set of dependency structures with words, i.e. a grammar and consequently a language. Thus our grammars consist in a pair $\mathbf{G} = (W, \Pi)$, where W is a manifold and $\Pi \subseteq W \times \Sigma^*$ is a relationship which relates syntagmata to strings. The language of the grammar will be $L(\mathbf{G}) = \Pi(W) = \{x \in \Sigma \,|\, (S, x) \in \Pi, S \in W\}$. Now there are two crucial questions. First, how can the manifolds be established? Second, how can Π be established? This paper tries to answer the first question. Regarding the second question we are going to define the chains through the figures; but this should not create problems. In other words, we do not show a general mechanism to establish Π, but we will do it for each specific case.

3 Grouping Concordances: Linguistic Basis

In order to understand the central proposal we are going to present some linguistic cases and extract a heuristic conclusion.

3.1 Verb Inflection in English

We are interested in grouping the several agreement instances of the words in a sentence. The most visible match is when two words are morphologically matched. Some languages are more profuse in this respect, for example Romance languages, however even English manifests some examples. In English when the subject of a sentence is the third person singular and the main verb is present tense and not modal, the verb goes with a -s at the end. For example:

(1) $\underline{\text{John}}_{Sb}$ often $\underline{\text{eats}}_{e}$ meat.

We have underlined the morphologically matched words and subscripted the loci. The analysis is given in Figure 2a); the agreeing words are linked by a curve dashed line in this and the following examples.[4] Since the main verb is always allocated the locus e and the subject is in Sb we can represent this match as an ordered pair (e, Sb). At the moment we are interested in the loci involved, not in the rule or condition. Now we can take the new sentence:

---

and satisfying $x \xrightarrow{e} x$ and $x \xrightarrow{\varphi} y$, $y \xrightarrow{\psi} z \Rightarrow x \xrightarrow{\psi\varphi} z$; also we ask that there be a unique $\theta \in X$ such that $\forall y \exists \varphi\ \theta \xrightarrow{\varphi} y$, which ensures a unique root. A *generalized syntagma* is a multiple action a with a mapping $S : X \longrightarrow \Sigma^+$. An *action syntagma* is a generalized syntagma such that if $x \xrightarrow{\varphi} y, x \xrightarrow{\varphi} z \Rightarrow y = z$. This class does not contain syntagmata with repeated functions in a same level. A *free syntagma* is a generalized syntagma such that $x \xrightarrow{\varphi} y, x \xrightarrow{\psi} y \Rightarrow \varphi = \psi$. It can be proved that free and action syntagmata are in fact ordinary syntagmata, up to isomorphism. There are interesting algebraic relationships among these classes. All the definitions in this paper can be easily adapted for generalized syntagmata.

[4] Agreements (dashed lines) are not dependencies (arrows), but loci can be used to describe them.

(2) Mary says John$_{Sb\cdot Ob}$ often eats$_{Ob}$ meat.

There are two matched pairs: a new pair (e, Sb) and the old which is now in a deeper position $(Ob, Sb \cdot Ob)$. The sentence has an analysis as in Figure 2b). The agreement phenomenon iterates in deeper sentences: *Paul says Mary says John often eats meat*, *Peter says Paul says Mary says John often eats meat*, ... In general we can describe the set of the loci of all these agreements as: $Agr = \{(Ob^n, Sb \cdot Ob^n) \mid n \in \mathbb{N}^+\}$.

Fig. 2a)

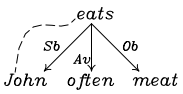


Fig. 2b)

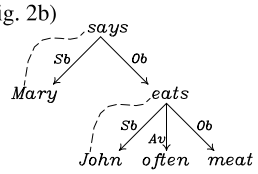


Fig. 2c)

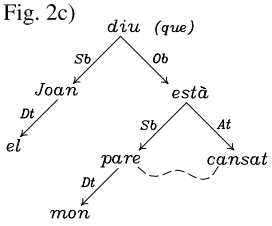


Fig. 2d)

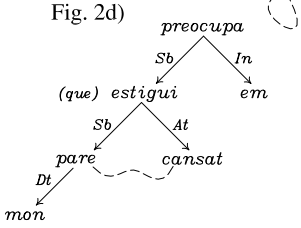


Fig. 2e)

3.2 Match of Subject and Attribute

We can consider other languages and other types of matching. In some romance languages the number and gender of the attribute of the copula must be equal to the subject.

(3) Mon pare$_{Sb}$ està cansat$_{At}$. / Les meves germanes$_{Sb}$ estan cansades$_{At}$.
My father is exhausted. / the my sisters are exhausted.
'My father is exhausted. / My sisters are exhausted.'

These examples are in Catalan, however the same occurs in Spanish, French or Italian. The matched pair is (Sb, At). We also consider other examples in a deeper position:

(4) a. El Joan diu que mon pare$_{Sb\cdot Ob}$ està cansat$_{At\cdot Ob}$.
The John says that my father is exhausted.
'John says my father is exhausted'.

b. Que mon pare$_{Sb\cdot Sb}$ estigui cansat$_{At\cdot Sb}$ em preocupa.
that my father is exhausted me worries.
'The fact of my father being exhausted is worrying'.

The analyses are given in Figures 2c) and 2d). The first agreement is $(Sb\cdot Ob, At\cdot Ob)$, but the second is $(Sb\cdot Sb, At\cdot Ob)$. This can be interpreted as that the set of agreements is $Agr = \{(Sb\cdot x, At\cdot x)\,|\,x\in\zeta^*\}$, in other words, this match would hold everywhere.

3.3 Pied-piping in Some Romance Languages

We consider another phenomenon called *pied-piping* consisting in the embedding of a filler such as a relative pronoun within accompanying material from the extraction site. Recall that we are only interested in the morphological matches, not in the semantical operation of the anaphora. In Catalan we must say:

(5) Aquesta reina$_{Sb}$, el pare del pare ... del pare de
This queen the father of-the father ... of-the father of
la$_{Dt\cdot Pr^n\cdot Ad\cdot Sb}$ qual fou destronat, regnà al segle XII.
the who was dethroned, reigned in century XII
'This queen the father of the father ... of the father of whom was dethroned reigned in XII century'.

The underlined words must agree in number and gender; see Figure 2e). In English there is not any match and we cannot underline any word because the pronoun is not preceded by any article. The proof of this agreement is that in Catalan we cannot say: **Aquesta reina, el pare del pare ... del pare de el qual fou destronat, regnà al segle XII.* Now the agreement set is $Agr = \{(Sb, Dt\cdot Pr^n\cdot Ad\cdot Sb)\,|\,n\in\mathbb{N}^+\}$. This is structurally a distinct type of construction because the monoid is growing right in the middle of certain loci. But this phenomenon can occur anywhere, for example in the object:

(6) El poble no acceptà mai aquesta reina$_{Ob}$ el pare del
The populace not accepted never this queen the father of-the
pare ... del pare de la$_{Dt\cdot Pr^n\cdot Ad\cdot Ob}$ qual fou destronat.
father ... of-the father of the who was dethroned.
'The populace never accepted this queen the father of the father ... of the father of whom was dethroned'.

So the agreements are more general: $Agr = (x, Dt\cdot Pr^n\cdot Ad\cdot x)\,|\,n\in\mathbb{N}^+, x\in\zeta^*\}$.

3.4 A Working Hypothesis

We can summarize all these agreements using submonoids of the monoid $\zeta^*\times\zeta^*$. In the first case we have: $Agr = \{(Ob^n, Sb\cdot Ob^n)\,|\,n\in\mathbb{N}^+\} = \varphi\cdot\Gamma$, where $\varphi = (e, Sb)\in\zeta^*\times\zeta^*$, $\Gamma = \langle(Ob, Ob)\rangle\subset\zeta^*\times\zeta^*$. In the second case we also have: $Agr = \{(Sb\cdot x, At\cdot x)\,|\,x\in\zeta^*\} = \varphi\cdot\Gamma$, where $\varphi = (Sb, At)\in\zeta^*\times\zeta^*$,

$\Gamma = \{(x, x) \,|\, x \in \zeta^*\} \subset \zeta^* \times \zeta^*$. Finally in the third case: $Agr = \{(x, Dt \cdot Pr^n \cdot Ad \cdot x) \,|\, n \in \mathbb{N}^+, x \in \zeta^*\} = \varphi \cdot \Gamma \cdot \psi \cdot \Gamma'$, where $\varphi = (e, Dt)$, $\Gamma = \langle (e, Pr) \rangle$, $\psi = (e, Ad)$ and $\Gamma' = \{(x, x) \,|\, x \in \zeta^*\}$. From the examples seen, and many others that could be given, we can extract a hypothesis. Let $\varphi_i \in \zeta^*$ and let Γ_i be submonoids of the monoid ζ^{*n}:

MONOIDAL HYPOTHESIS. *For any natural language the set of the places where agreements occur can be described as:*

$$Agr = \prod_{i=1}^{n} \varphi_i \cdot \Gamma_i \subseteq \zeta^{*n}.$$

Note that some submonoids Γ_i and constants φ_i can be trivial, thus we have a lot of patterns like: $\varphi\Gamma$, $\Gamma\varphi$, $\Gamma\Gamma'\varphi$, $\varphi\Gamma\psi\Gamma'$, ...

Even though the agreements shown were binary the monoidal hypothesis supports several arities, but the most common are 1, 2 and 3. For example: that *the subject Sb is always a noun* is itself an agreement of arity 1 which must hold generally in all loci of a syntagma, therefore the pattern must be $Sb \cdot \zeta^*$.

We have been talking about morphological agreement but there is no reason not to talk about other kinds. When a verb like *to drink* selects a "drinkable" object it is producing a semantic "agreement" (e, Ob), and so forth.

4 Rules, Patterns and Syntactic Manifolds

4.1 Valuations, Rules and Satisfiability

We are going to formalize the concept of *pattern*, which goes preceded by the concept of *rule*. A rule is a linking condition while a pattern tells us where the rule must be complied with.

A *valuation* of arity n is a mapping $B : \Sigma^{+n} \longrightarrow \{0, 1\}$, where $0, 1$ are the truth values. Some valuations that we will frequently use are the following: the *characteristic function* defined as $(\in \sigma) : \Sigma^{+n} \longrightarrow \{0, 1\}$, $(\in \sigma)(x) = 1 \iff x \in \sigma \subseteq \Sigma^{+n}$; the *equality* defined as $(\approx) : \Sigma^{+2} \longrightarrow \{0, 1\}$, $(\approx)(x, y) = 1 \iff x = y$; or its curried restriction, $(\approx a)(x) = (\approx)(x, a)$. We will use the infix notations: $x \in \sigma$, $x \approx y$ and $x \approx a$.

Definition 2. *Given $(\varphi_1, \ldots, \varphi_n) \in \zeta^{*n}$ and given a valuation B of arity n, we call the ordered pair $(B, (\varphi_1, \ldots, \varphi_n))$ a* rule. *Given a syntagma $S : \zeta^* \longrightarrow \Sigma^+$, and a rule, $R = (B, (\varphi_1, \ldots, \varphi_n))$, we say that S* satisfies *R, and we write, S sat R iff*

$$B(S(\varphi_1), \ldots, S(\varphi_n)) = 1.$$

Example 2. Let $\mathbf{Det} \subset \Sigma$ be the set of all determiners in English and let $Dt \in \zeta$ be a syntactic function. We want Dt to always take an element from **Det**. The rule $(x \in \mathbf{Det}, (Dt))$, or informally $Dt \in \mathbf{Det}$, says exactly that.

Due to the boolean nature of valuations we can define new valuations from others. Let a pair of valuations be $B : \Sigma^{+n} \longrightarrow \{0,1\}$, $B' : \Sigma^{+m} \longrightarrow \{0,1\}$; then we can define:

$$B \wedge B' : \Sigma^{+(n+m)} \longrightarrow \{0,1\}$$

$$(x_1, \ldots, x_n, y_1, \ldots, y_m) \longmapsto B(x_1, \ldots, x_n) \wedge B(y_1, \ldots, y_n).$$

In the same way we can define: $B \vee B'$, $B \to B'$, $\neg B$, and so forth. If we have two rules: $R = (B, \varphi)$ and $R' = (B', \varphi')$ we can define: $R \wedge R' = (B \wedge B', (\varphi, \varphi'))$, $R \vee R' = (B \vee B', (\varphi, \varphi'))$, $R \to R' = (B \to B', (\varphi, \varphi'))$, $\neg R = (\neg B, \varphi)$. Composing new valuations and rules does not increase our descriptive power but it makes reading easier.

Example 3. Let **Trans** $\subset \Sigma$ be the set of transitive verbs and let Ob be a syntactic function. The following rule asserts that the object cannot be null if a verb is transitive in infix notation: $\big(x \in \mathbf{Trans} \to y \not\approx 0, (e, Ob)\big)$; or by a mild abuse of notation: $e \in \mathbf{Trans} \to Ob \not\approx 0$.

4.2 Patterns and Manifolds

First of all we fix some easy set notation. As is usual in algebra, if $x \in X$ and $Y, Y' \subseteq X$, where X is a set with a binary operation $\cdot : X \times X \longrightarrow X$, then $x \cdot Y = \{x\} \cdot Y = \{x \cdot y \,|\, y \in Y\}$ and $Y \cdot Y' = \{y \cdot y' \,|\, y \in Y, y' \in Y'\}$.

Definition 3. *We say $\Gamma \subseteq \zeta^{*n}$ is a* (monoidal) pattern *of arity n iff it can be written as:*

$$\Gamma = \prod_{i=1}^{k} \varphi_i \cdot \Gamma_i,$$

where $\varphi_i \in \zeta^{*n}$ and each Γ_i is a finitely generated submonoid of ζ^{*n}.[5] *We call the number k the* length of the pattern.

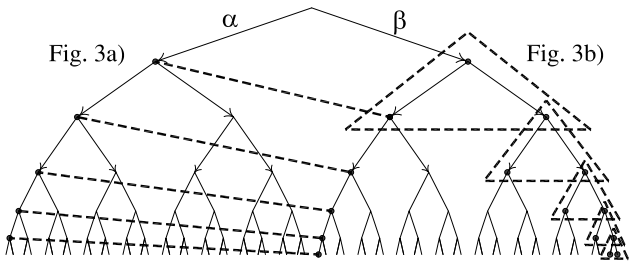
From the definition we see that every submonoid of ζ^{*n} is itself a monoidal pattern. Some simple examples are:

Example 4. The set $\{(e, e, e)\}$ is a monoidal pattern of arity 3. We call such a pattern trivial. The set $\{(\varphi_1, \varphi_2)\}$ is a monoidal pattern with arity 2 and of only one element. We call such patterns constants. The set $\{(\alpha\beta^i\gamma\alpha^j, \gamma^2\alpha^i) \;|\, i, j \in \mathbb{N}^+\}$ is a monoidal pattern of arity 2 because it can be written as $(\alpha, \gamma^2) \cdot \langle(\beta, \alpha)\rangle \cdot (\gamma, e) \cdot \langle(\alpha, e)\rangle$. However the set $\{(\alpha^i\beta^j, \beta^j\alpha^i) \,|\, i, j \in \mathbb{N}^+\}$ is not a monoidal pattern because the set in itself is not a monoid nor a constant and it cannot be decomposed into a succession of products of submonoids and constants. On the other hand, the set $\{(\alpha^i\beta^j, \alpha^i\beta^j) \,|\, i, j \in \mathbb{N}^+\}$ can be written as: $\langle(\alpha, \alpha)\rangle \cdot \langle(\beta, \beta)\rangle$, and this shows that it is a pattern.

[5] Remember that some of the φ_i and Γ_i can be trivial.

We notate $H_{\alpha_1,\ldots,\alpha_t} = \langle \alpha_1, \ldots, \alpha_t \rangle$, where $\alpha_1, \ldots, \alpha_t \in \zeta$. We also define, given a pattern Γ, the nth homogeneous power as the set $\Gamma^{n)} = \{(x, \ldots, x)_n \,|\, x \in \Gamma\}$, where the subscript means there are n copies of x in the vector. We call monoids such as $H^{n)}_{\alpha_1,\ldots,\alpha_t}$ homogeneous submonoids. So the last pattern from the example 4 can be written more comfortably as $\langle(\alpha,\alpha)\rangle \cdot \langle(\beta,\beta)\rangle = H^{2)}_{\alpha} H^{2)}_{\beta}$. We say that a pattern is homogeneous iff all its submonoids are homogeneous.

Figure 3 depicts the Cayley graph of the free monoid $\zeta^* = \{\alpha, \beta\}^*$. After the constant patterns, the simplest patterns are those of the form $\varphi\Gamma$, and they can be understood as follows: we fix some loci defined by the components of the constant φ, then we geometrically translate them around the syntagma according to Γ as in Figure 3b) which is depicting a pattern of arity 3, $(e,\alpha,\beta)H^{3)}_{\beta} = (e,\alpha,\beta)\langle(\beta,\beta,\beta)\rangle$. If $\Gamma = \zeta^{*n)}$ then for each subsyntagma under φ we have a copy of φ. However when we multiply on the left side, $\Gamma\varphi$, the constant is reproduced in parallel as in Figure 3a) which is depicting a pattern of arity 2, $H^{2)}_{\alpha}(e,\beta) = \langle(\alpha,\alpha)\rangle(e,\beta)$. More complex patterns composing these actions can also be visualized.



Definition 4. *Let Γ be a monoidal pattern and B a valuation, both with the same arity. We call the pair (B,Γ) a* pattern rule *which defines a set of rules denoted by $\binom{B}{\Gamma} = \{(B,\varphi) \,|\, \varphi \in \Gamma\}$.*

Definition 5. *Given a pattern rule (B,Γ) we define the* simple syntactic manifold $\mathbf{Synt}_{\Sigma,\zeta}\binom{B}{\Gamma} = \{S \in \mathbf{Synt}_{\Sigma,\zeta} \mid S \,sat\, R \;\forall R \in \binom{B}{\Gamma}\}$. *Given a number of pattern rules we define the* syntactical manifold*:*

$$\mathbf{Synt}_{\Sigma,\zeta}\begin{pmatrix} B_1 \cdots B_n \\ \Gamma_1 \cdots \Gamma_n \end{pmatrix} = \mathbf{Synt}_{\Sigma,\zeta}\begin{pmatrix} B_1 \\ \Gamma_1 \end{pmatrix} \cap \cdots \cap \mathbf{Synt}_{\Sigma,\zeta}\begin{pmatrix} B_n \\ \Gamma_n \end{pmatrix}.$$

When the sets ζ, Σ are understood we just write: **Synt**. From the definitions it immediately follows that if V, W are syntactic manifolds, then $V \cap W$ is a syntactic manifold, so syntactic manifolds form a semilattice.

5 Five Classical Examples

We show five classical formal language examples, presented in two groups. The monoidal patterns will highlight a symmetry between these languages.

5.1 Squares Language vs. Copy Language and Mirror Language

We fix a vocabulary, for instance $\Sigma^* = \{a, b, c\}$. Let there be the languages $L_{\text{squa}} = \{x_1^2 \cdots x_n^2 \,|\, x_1, \ldots, x_n \in \Sigma, n \in \mathbb{N}\}$, and $L_{\text{copy}} = \{xx \,|\, x \in \Sigma^*\}$. The first is a context free, indeed regular, language. It is a mathematical idealization of a chain of subordinate clauses in many languages such as English. E.g.:

(7) ... that John^a saw^a Peter^b help^b Mary^c read^c.

The second is not context free and it represents the typical chain of subordinate clauses of Dutch:[6]

(8) ... dat Jan^a Piet^b Marie^c zag^a helpen^b lezen^c.
... that Jan Piet Marie saw help read.
'... that J. saw P. help M. read'.

Consider the squares language L_{squa}. We take the syntactic manifold with $\zeta = \{\alpha, \beta\}$ and $\Sigma = \{a, b, c\}$ as follows. Let there be the valuation $x \approx 0$ (we abbreviate ≈ 0) and $x \approx y$ (we write simply $\approx$). Then we have the manifold:

$$W_{\text{squa}} = \mathbf{Synt}\begin{pmatrix} \approx 0 & \approx 0 \\ \alpha^2 H_\beta & \beta\alpha H_\beta \end{pmatrix} \cap \mathbf{Synt}\begin{pmatrix} \approx \\ (e, \alpha) H_\beta^{2)} \end{pmatrix}.$$

We separate this manifold in two parts because the second is giving important information about the syntagmata. The first part can be considered superfluous; it is just saying that all loci not invoked by the second part will be null.

If we take the projective linearization Π_{squa} as in Figure 4a) for the manifold W_{squa}, we will obtain the language $\Pi_{\text{squa}}(W_{\text{squa}}) = L_{\text{squa}}$.

For the copy language we define the manifold:[7]

$$W_{\text{copy}} = \mathbf{Synt}\begin{pmatrix} \approx 0 & \approx 0 \\ \alpha H_\beta \alpha & \beta H_\beta \alpha \end{pmatrix} \cap \mathbf{Synt}\begin{pmatrix} \approx \\ H_\beta^{2)}(e, \alpha) \end{pmatrix}.$$

If we take the projective the linearization Π_{copy} as in Figure 4b) for the manifold W_{copy}, we will obtain the language $\Pi_{\text{copy}}(W_{\text{copy}}) = L_{\text{copy}}$.

Note that both nonsuperfluous parts are very similar; the valuations are equal and the only difference lies on the laterality of patterns: $(e, \alpha)H_\beta^{2)}$ in the first language and $H_\beta^{2)}(e, \alpha)$ in the second.

Closely related to both the above languages we consider the mirror language defined as $L_{\text{mirr}} = \{xx^R \,|\, x \in \Sigma^*\}$, where x^R is the reversed word. This language captures the nested dependencies in German, with sentences like:

(9) ... dass Jan^a Piet^b Marie^c lesen^c helfen^b sah^a.
... that Jan Piet Marie read help saw.
'... that J. saw P. help M. read'.

[6] This also occurs in Swiss-German, (Shieber, 1987[7]).

[7] Our analysis of Dutch dependencies uses a parallel arrangement of the functions β which shares distant similarities with Bresnan, 1987[1], but there the framework was constituency grammars.

Fig. 4a)

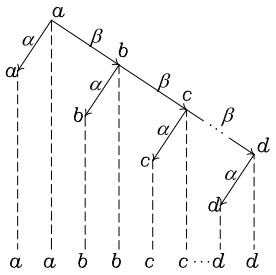


Fig. 4b)

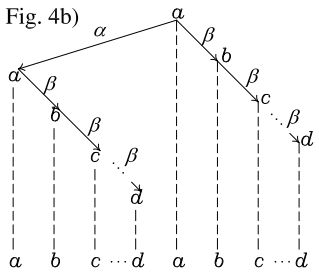


Fig. 4c)

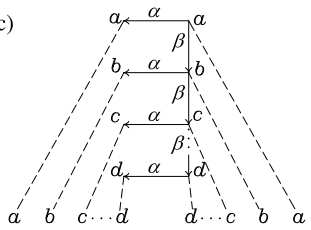
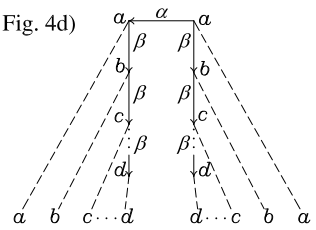


Fig. 4d)



Curiously these sentences can be projectively obtained from both manifolds W_{squa} and W_{copy} as can be checked in Fig. 4c) and 4d): if we define Π_1 as in Figure 4c) and Π_2 as in Figure 4d) we will have $\Pi_1(W_{\text{squa}}) = L_{\text{mirror}} = \Pi_2(W_{\text{copy}})$.

5.2 Language of Multiple *abc* vs. Respectively *abc*

Now we are going to consider the languages $L_{\text{mult}} = \{(abc)^n \,|\, n \in \mathbb{N}^+\}$ and $L_{\text{resp}} = \{a^n b^n c^n \,|\, n \in \mathbb{N}^+\}$. The first is a context free, indeed regular language, but not the second. The first corresponds to simple coordination, (10a), while the second is a mathematical idealization of the *respectively* construction (10b):

(10) a. Jean^a seems German^b but he is French^c, Pietro^a seems Russian^b but he is Italian^c and Peter^a seems Belgian^b, but he is English^c.

b. Jean^a, Pietro^a and Peter^a seem respectively German^b, Russian^b and Belgian^b, but they are French^c, Italian^c and English^c.

Let there be the sets $\zeta = \{\alpha, \beta, \gamma\}$, $\Sigma = \{a, b, c\}$, and let there be the valuations:

$$B(x, y, z) = (x \in \{0, a\}) \wedge (y \in \{0, b\}) \wedge (z \in \{0, c\}),$$

$$B'(x, y, z) = \big(x \approx 0 \wedge y \approx 0 \wedge z \approx 0\big) \vee \big(x \not\approx 0 \wedge y \not\approx 0 \wedge z \not\approx 0\big).$$

We take the manifold

$$W_{\text{mult}} = \mathbf{Synt}\begin{pmatrix} \approx 0 & \approx 0 & \approx 0 & \approx 0 & \approx 0 & \approx 0 \\ \alpha\alpha H_\beta & \beta\alpha H_\beta & \gamma\alpha H_\beta & \alpha\gamma H_\beta & \beta\gamma H_\beta & \gamma\gamma H_\beta \end{pmatrix}$$

$$\cap \;\; \mathbf{Synt}\begin{pmatrix} B & B' \\ (\alpha, e, \gamma)H_\beta^{3)} & (\alpha, e, \gamma)H_\beta^{3)} \end{pmatrix}.$$

The projective linearization Π_{mult} as in Figure 5a) for the manifold W_{mult} yields the language $\Pi_{\text{mult}}(W_{\text{mult}}) = L_{\text{mult}}$.

Finally we take the manifold with exactly the same valuations B, B' but with some specific changes in the patterns:

$$W_{\text{resp}} = \mathbf{Synt}\begin{pmatrix} \approx 0 & \approx 0 & \approx 0 & \approx 0 & \approx 0 & \approx 0 \\ \alpha H_\beta \alpha & \alpha H_\beta \gamma & \alpha H_\beta & \gamma H_\beta & \alpha H_\beta \gamma & \gamma H_\beta \gamma \end{pmatrix}$$

$$\cap \ \mathbf{Synt}\begin{pmatrix} B & B' \\ H_\beta^{3)}(\alpha, e, \gamma) & H_\beta^{3)}(\alpha, e, \gamma) \end{pmatrix}.$$

The projective linearization Π_{resp} as in Figure 5a) for the manifold W_{resp} yields the language $\Pi_{\text{resp}}(W_{\text{resp}}) = L_{\text{resp}}$. Again note that both nonsuperfluous parts are very similar; the valuations are equal and the only difference lies in the laterality: $(\alpha, e, \gamma)H_\beta^{3)}$ and $H_\beta^{3)}(\alpha, e, \gamma)$.

Fig. 5a)

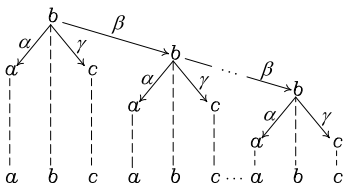


Fig. 5b)

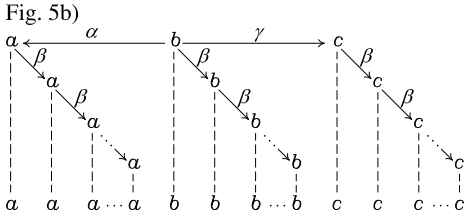


Fig. 5c)

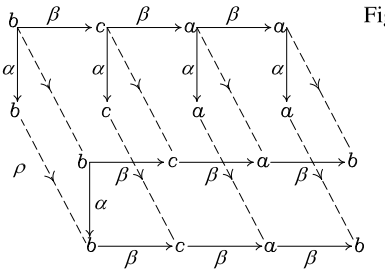
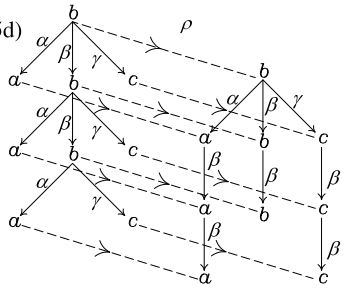


Fig. 5d)



6 How to Build Natural Languages

Due to the boolean nature of valuations we are able to implement a wide variety of rules. If we want to explain a natural language we can build a number of manifolds capturing different phenomena. We can begin with a manifold to describe for each word what kind of words or categories it can govern. For example, a noun can govern a determiner and an adjective, but not an adverb or verb. With another manifold we can define for each function what functions can follow, such

as the rule for transitive verbs in example 3.[8] We can define still another manifold to describe agreements. Then we have to think where these rules must hold, i.e. we must think of the patterns. For all rules of this kind the pattern will have the shape $\varphi\Gamma$. In fact these kind of patterns seems to be related to context free languages. Fortunately in order to add new phenomena, like *pied-piping*, *cross-serial dependencies* or others, we can build other manifolds and intersect with the existing ones. Possibly new phenomena demand new kinds of patterns. For example we already saw that pied-piping in Catalan had a pattern like $\varphi\Gamma\varphi'\Gamma'$.

7 Symmetric Syntagmata, Manifolds and Languages

To analyze sentences with cross-serial dependencies from (8) and (10) we can use the underlying logic of simplified languages L_{copy} and L_{resp}. See Figures 6b) and 6d). Nevertheless, this solution could be considered inadequate by the general reader for at least two reasons. First, it assumes different languages to have the same analysis (equal syntagmata up to words). Second, intuitively it is thought that governance must be coherent with semantical roles. We have violated both principles. On the one hand, we would have supposed different analyses for English and Dutch, or for respectively constructions and ordinary coordination; see again all the analyses in 6a), 6b), 6c) and 6d). On the other hand, in the Dutch analysis the function Cr should follow a verb, not a Noun. We need to address these two points. Even though our analysis seems *ex professo* chosen to be comfortably projective,[9] there exists an easy and closer relationship between both analysis: an algebraic permutation of syntactic functions. We now look at this.

As is usual in algebra a morphism is a mapping preserving structures. In our case a morphism between syntagmata $S : \zeta^* \longrightarrow \Sigma^+$ and $S : \zeta'^* \longrightarrow \Sigma'^+$ is a pair of mappings $f : \zeta^* \longrightarrow \zeta'^*$, $g : \Sigma \longrightarrow \Sigma'$ (which may be partial) such that $g(0) = 0$ and they make commutative the diagram, $S' \circ f = g \circ S$:

$$\begin{array}{ccc} Spt(S) & \xrightarrow{f} & Spt(S') \\ {\scriptstyle S}\downarrow & & \downarrow{\scriptstyle S'} \\ \Sigma & \xrightarrow{g} & \Sigma' \end{array}$$

Definition 6. *Let ρ be a permutation of subscripts $\{1, \ldots, m\}$, and consider a disjunctive decomposition $\zeta = \zeta_1 \cup \cdots \cup \zeta_m$. We call a pair (f, g) a* symmetry *when f is a permutation of submonoids, $f : \zeta_1^* \cdots \zeta_m^* \longrightarrow \zeta_{\rho(1)}^* \cdots \zeta_{\rho(m)}^*$, $f(x_1 \cdots x_m) = x_{\rho(1)} \cdots x_{\rho(m)}$. By a mild abuse of notation we simply write $\rho = (f, g)$.*

Proposition 2. *Using the same notation, given two manifolds W and W' we have that for each syntagma $S \in W$, $Spt(S) \subseteq \zeta_1^* \cdots \zeta_m^* \subseteq \zeta^*$, there exists a unique syntagma $S' \in W'$, $Spt(S') \subseteq \zeta_{\rho(1)}^* \cdots \zeta_{\rho(m)}^* \subseteq \zeta^*$, such that the symmetry becomes a morphism of syntagmata.*

[8] Using the Tesnèrian denomination, this constitutes defining the *valence* of each word.
[9] A similar strategy can be found in Topological Dependency Grammar, [5].

Proof. For (f, g) to be a morphism, we require that $S' \circ f = g \circ S$. Note that f is bijective and well defined since the decomposition of ζ^* is disjunctive, so given S we put $S' = g \circ S \circ f^{-1}$, and in this way S' is uniquely defined. Clearly if $Spt(S) \subseteq \zeta_1^* \cdots \zeta_m^*$ then $Spt(S') \subseteq \zeta_{\rho(1)}^* \cdots \zeta_{\rho(m)}^*$. □

Automatically a symmetry induces a mapping of manifolds $\tilde{\rho} : W \longrightarrow W'$, $S \mapsto S'$. When this occurs we say that the languages $L = \Pi(W)$ and $L' = \Pi'(W')$ are symmetric, and we write $L \bot L'$.

Example 5. We consider the symmetry, $f : \{\alpha\}^*\{\beta\}^* \longrightarrow \{\beta\}^*\{\alpha\}^*$; see Figure 5c). The pair $\rho = (f, Id)$ defines a bijection between manifolds $W_{\text{squa}} \longrightarrow W_{\text{copy}}$. So we have two symmetric languages $L_{\text{squa}} \bot L_{\text{copy}}$. Similary if we consider the symmetry, $f' : \{\alpha, \gamma\}^*\{\beta\}^* \longrightarrow \{\beta\}^*\{\alpha, \gamma\}^*$, see Figure 5d), we will have that $L_{\text{mult}} \bot L_{\text{resp}}$. Any language is symmetric with itself, because the pair (Id, Id) is trivially a symmetry. However the mirror language enjoys nontrivial selfsymmetry with the first pair (f, Id).

Symmetries seem to establish a close correspondence between context free languages and some non-context free languages. Regarding the question of the suitability of analysis, the fact is that in our model we do not have equal syntagmata, but isomorphic or symmetric syntagmata. Let us now see this effect in natural languages:

Example 6. Following the Figures 6a) and 6b), we consider the symmetry from English to Dutch: $f : \{Sb\}^* \cdot \{Cr\}^* \cdot \{Ob\}^* \longrightarrow \{Cr\}^* \cdot \{Sb\}^* \cdot \{Ob\}^*$. Let g be the vocabulary mapping defined as: $g : \Sigma_{\text{English}} \longrightarrow \Sigma_{\text{Dutch}}$, $g(\text{says}) = \text{zegt}$, $g(\text{saw}) = \text{zag}$, $g(\text{help}) = \text{helpen}$, ... This suggests that English and Dutch are symmetric languages.

Fig. 6a)

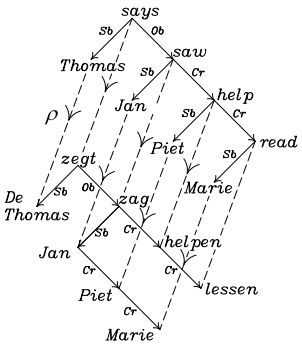


Fig. 6b)

Fig. 6c)

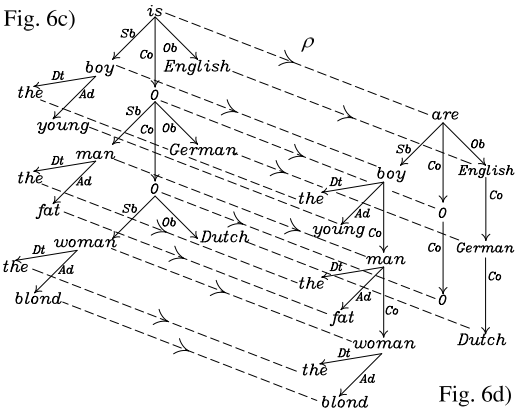


Fig. 6d)

Example 7. Finally let us show a slightly more complex situation where we need elliptic syntagmata. Consider the ordinary coordination and the respectively construction:

(11) a. The young boy is English, the fat man German, and the blond woman Dutch.

b. The young boy, the fat man and the blond woman are respectively English, German and Dutch.

See the dashed lines linking Figures 6c) and 6d). Now the symmetry follows by commutation of the coordination function Co to the subject and object: $f : \{Dt, Ad\}^* \cdot \{Sb, Ob\}^* \cdot \{Co\}^* \longrightarrow \{Dt, Ad\}^* \cdot \{Co\}^* \cdot \{Sb, Ob\}^*$, while the vocabulary does not change anything except $g(\text{is}) = \text{are}$.

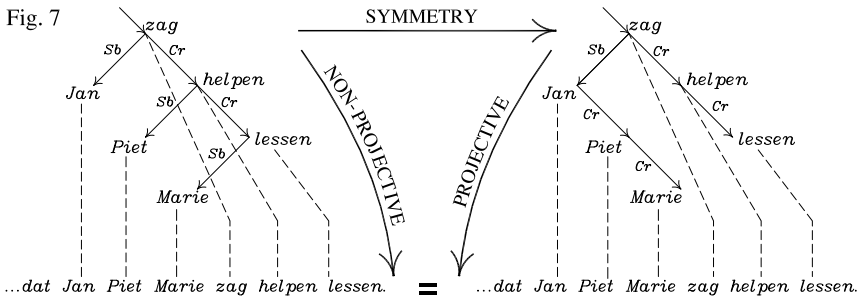
8 Conclusions

Initially in §1 we commented that there are some situations that seem not to accept projective structures. In such a circumstance we have two options: either we change the governance structure or we loosen the sense of projectivity. It is usually thought that changing the governance structure is more difficult than trying other projections. However patterns and symmetries allow us to explore the former solution.

Notwithstanding, there is another interpretation of the above results. If we want to insist on the usual analyses of dependencies, then we will have to linearize non-projectively some trees, say $\Pi_{\text{No-Pr}}$. However then the symmetries tell us that the relationship $\Pi_{\text{No-Pr}}$ in some cases can be factorized as $\Pi_{\text{No-Pr}} = \Pi_{\text{Pr}} \circ \tilde{\rho}$, where Π_{Pr} is a projective linearization, as in Fig 7. That is:

$$\text{Non-projectivity} = \text{Symmetry} + \text{Projectivity}.$$

Fig. 7



References

1. Joan Bresnan, Ronald M Kaplan, Stanley Peters, and Annie Zaenen. Cross-serial dependencies in dutch. In *The formal complexity of natural language*, pages 286–319. Springer, 1987.

2. Ralph Debusmann. An introduction to dependency grammar. *Hausarbeit fur das Hauptseminar Dependenzgrammatik SoSe*, 99:1–16, 2000.
3. Ralph Debusmann, Denys Duchier, and Geert-Jan M Kruijff. Extensible dependency grammar: A new methodology. In *Proceedings of the COLING 2004 Workshop on Recent Advances in Dependency Grammar*, pages 70–76, 2004.
4. Ralph Debusmann and Marco Kuhlmann. Dependency grammar: Classification and exploration. In *Resource-adaptive cognitive processes*, pages 365–388. Springer, 2010.
5. Denys Duchier and Ralph Debusmann. Topological dependency trees: A constraint-based account of linear precedence. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 180–187. Association for Computational Linguistics, 2001.
6. Marco Kuhlmann. Mildly non-projective dependency grammar. *Computational Linguistics*, 39(2):355–387, 2013.
7. Stuart M Shieber. *Evidence against the context-freeness of natural language*. Springer, 1987.

On the mild context-sensitivity of k-Tree Wrapping Grammar

Laura Kallmeyer

Heinrich-Heine-Universität Düsseldorf, Germany
kallmeyer@phil.hhu.de

Abstract. Tree Wrapping Grammar (TWG) has been proposed in the context of formalizing the syntactic inventory of Role and Reference Grammar (RRG). It is close to Tree Adjoining Grammar (TAG) while capturing predicate-argument dependencies in a more appropriate way and being able to deal with discontinuous constituents in a more general way. This paper is concerned with the formal properties of TWG. More particularly, it considers k-TWG, a constrained form of TWG. We show that for every k-TWG, a simple Context-Free Tree Grammar (CFTG) of rank k can be constructed, which is in turn equivalent to a well-nested Linear Context-Free Rewriting System (LCFRS) of fan-out $k+1$. This shows that, when formalizing a grammar theory such as RRG, which is based on thorough and broad empirical research, we obtain a grammar formalism that is mildly context-sensitive.

Keywords: tree rewriting grammars, Role and Reference Grammar, simple context-free tree grammar, mild context-sensitivity

1 Introduction

Tree Wrapping Grammar (TWG) [6] has been introduced in the context of formalizing the syntactic inventory of Role and Reference Grammar (RRG) [15, 14]. A TWG consists of elementary trees, very much in the spirit of Tree Adjoining Grammar (TAG) [5], and from these elementary trees, larger trees are obtained by the operations of *(wrapping) substitution* and *sister adjunction*. (Wrapping) substitutions are supposed to add syntactic arguments while sister adjunction is used to add modifiers and functional operators. A discontinuous argument can be split via the wrapping substitution operation: the argument tree has a specific split node v. When adding such an argument to a predicate tree, the lower part (rooted in v) fills an argument slot via substitution while the upper ends up above the root of the target predicate tree.

[6] adopts the rather flat syntactic structure from RRG with categories CORE, CLAUSE and SENTENCE. A sample RRG-inspired TWG derivation is shown in Fig. 1. This example involves substitutions of the arguments *John* and *Mary* into the *hates* tree and of *Bill* into *claims*, sister adjunction of *definitely* into *hates* (sister adjunction simply adds a new daughter to a node where

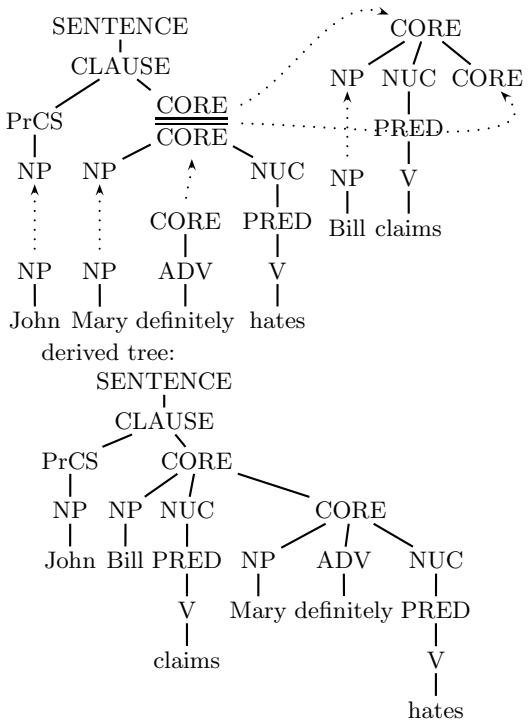


Fig. 1. RRG-style TWG derivation for *John Bill claims Mary definitely hates*

the root of the adjoining tree has to match the target node), and wrapping substitution of the *hates* tree around the *claims* tree.

It was shown in [6] that, in contrast to TAG, the TWG operations enable us to add even sentential arguments with long-distance extractions by a substitution operation. In this, TWG is close to other formalisms in the context of TAG-related grammar research that have been proposed in order to obtain derivation structures that reflect dependencies in a more appropriate way than it is done by TAG [10, 11, 1].

The focus of this paper is on the formal properties of TWG. After an introduction to TWG and in particular to k-TWG, a restricted form of TWG, we show how to construct an equivalent simple context-free tree grammar (CFTG) for a given k-TWG.

2 Tree Wrapping Grammar

The following introduction to TWG is largely taken from [6], except for the declarative definition of the notion of k-TWG, based on properties of the derivation trees decorated with wrapping substitution markings.

Borrowing from the TAG terminology, trees that can be added by substitution are called *initial* trees. In addition, we need *adjunct trees* for modeling

modifiers and functional elements in RRG. These trees are added by sister adjunction. We distinguish **l**eft-adjoining, **r**ight-adjoining and unrestricted adjunct trees, resp. called *l-adjunct*, *r-adjunct* and *d-adjunct* trees. (The latter can add a **d**aughter at any position.)

Definition 1 (Tree Wrapping Grammar). *A* Tree Wrapping Grammar *(TWG) is a tuple* $G = \langle N, T, I, A_D, A_L, A_R, C\rangle$ *where*

a) N, T *are disjoint alphabets of non-terminal and terminal symbols.*

b) I, A_D, A_L *and* A_R *are disjoint finite sets of ordered labeled trees such that*
- *each non-leaf in a tree is labeled by some element from* $N \cup N^2$*,*
- *there is at most one node with a label from* N^2*,*
- *leaves have labels from* $N \cup T$*, and*
- *the root of each tree in* $A_D \cup A_L \cup A_R$ *has exactly one daughter.*

c) $C \subseteq N$*.*

A non-terminal leaf is called a substitution node*, and the labels from* N^2 *are called* split categories*.*

Every tree in I *is called an* initial *tree, every tree in* $A_D \cup A_L \cup A_R$ *an* adjunct *tree and every tree in* $I \cup A_D \cup A_L \cup A_R$ *an* elementary tree*.*

As we will see later, C is the set of non-terminals that can occur on a *wrapping spine* (i.e., between root and substitution site of the target tree of a wrapping substitution).

There are two TWG composition operations (see Fig. 2):

1. *Standard/Wrapping substitution*: a substitution node v in a tree γ gets replaced with a subtree α' of an initial tree α. If $\alpha' \neq \alpha$, then the root node v' of α' must be labeled with a split category $\langle X, Y\rangle$ such that the root of γ is labeled X and v is labeled Y. α is then split at v' and wraps around γ, i.e., the upper part of α ends up above the root of γ while α' fills the substitution slot. In this case, we call the operation a *wrapping substitution*. Otherwise ($\alpha = \alpha'$), we have a *standard substitution* and the root of α (i.e., v') must have the same label as v.
2. *Sister adjunction*: an adjunct tree β with root category X is added to a node v of γ with label X. The root r_β of β is identified with v and the (unique) daughter of r_β is added as a new daughter to v. Furthermore, if $\beta \in A_L$ (resp. $\beta \in A_R$), then the new daughter must be a leftmost (resp. rightmost) daughter.

A slightly different form of tree wrapping is proposed in [9] for RRG, leading to a flatter structure. One can consider a split node as a very special dominance edge (with specific constraints on how to fill it). In our definition, we would then have for a split node with categories X, Y a dominance edge between a node labeled X and a node labeled Y such that the X-node does not have any other daughters. In the flatter version of wrapping ([9]), the X-node can have other daughters that end up being sisters of the target tree of the wrapping that fills this dominance edge (see Fig. 3).

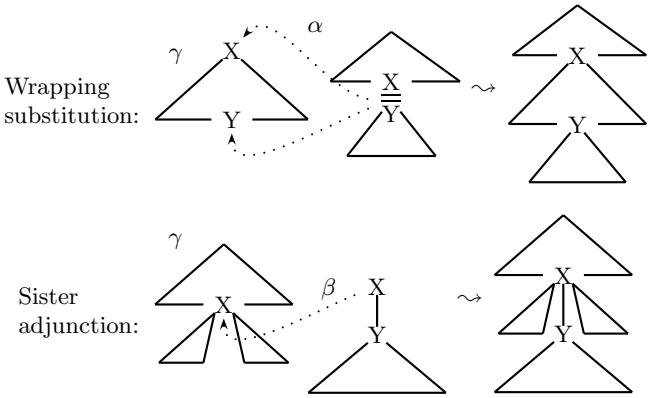


Fig. 2. Operations in TWG

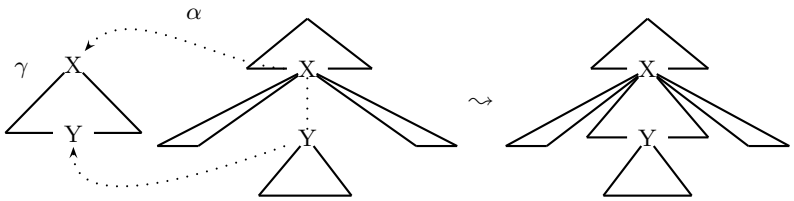


Fig. 3. Wrapping from [9]

It is easy to see that this form of wrapping can be transformed into the one used in this paper, simply by replacing the dominance edge with an immediate dominance edge and splitting the lower node with top and bottom categories X and Y respecitvely. As a result, we obtain trees that are slightly less flat than the ones from [9] and that, if we keep track of which edges have been added in this transformation, can be easily transformed back to the original flatter form. Therefore, without losing anything concerning the desired linguistic structures, for formal properties and parsing considerations we can work with the tree wrapping definition presented here.

Every elementary tree in a TWG G is a derived tree wrt. G, and every tree we can obtain with our composition operations from derived trees in G is again a derived tree. Wrapping substitutions require that, in the target tree, all categories on the path from the root to the substitution node (the *wrapping spine*) are in C. A further constraint is that wrapping substitution can target only initial trees, i.e., we cannot wrap a tree around an adjunct tree. Note that, in contrast to [6], we do not impose a limit on the number of wrapping substitutions stretching across a node in our definition of a TWG derived trees. This constraint comes later with the definition of k-TWG.

So far, wrapping can occur several times at the same elementary tree. Or, to put it differently, a node can be on several wrapping spines that are not nested.

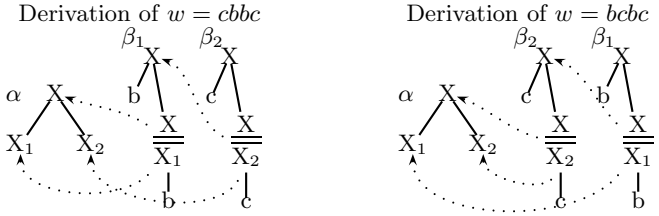


Fig. 4. Sample derivations: wrapping substitutions at sister nodes

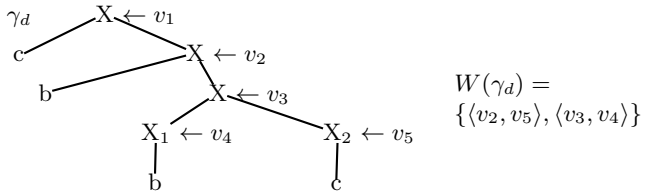


Fig. 5. Decorated derived tree arising from the first derivation in Fig. 4

See Fig. 4 for an example. In the two derivations, the root node of α is part of the two wrapping spines. In other words, both wrappings stretch across the root node of α. This has implications for the generative capacity and also for the parsing complexity.

We now want to restrict the wrapping substitutions in a derivation concerning the number of times a node can be part of a wrapping spine. To this end, we first introduce some decoration for the derived trees in the TWG given a specific derivation: Let γ_d be a tree derived in a TWG with a fixed derivation (there can be several derivations per derived tree and, consequently, several decorations). We define the wrapping decoration of γ_d as the following set of node pairs $W(\gamma_d)$: In every wrapping substitution step of the derivation in question with r and v being the root node r and the substitution node v of the target of the wrapping substitution, $\langle r, v\rangle \in W(\gamma_d)$. Nothing else is in $W(\gamma_d)$. We call every $v_\perp$ such that there exists a $v_\top$ with $\langle v_\top, v_\perp\rangle \in W(\gamma_d)$ a $\perp$ *node* in γ_d. We call a derived tree with such a decoration a *decorated* derived tree. An example is given in Fig. 5.

Once we have that, we can identify for a node v in such a decorated derived tree γ all wrapping substitution sites ($\perp$ nodes) where the wrapping substitution stretches across v.

Definition 2 (Gap set, wrapping degree). *Let $\gamma = \langle V, E, \prec, r, l\rangle$ be a decorated TWG derived tree with decoration $W(\gamma)$, and let $v \in V$.*

1. *A set $V_\perp \subset V$ of $\perp$ nodes is a* gap set *with respect to v if*
   a) *for every pair $\langle v_\top, v_\perp\rangle \in W(\gamma)$ with $v_\perp \in V_\perp$, it holds that $v_\top$ dominates v and v strictly dominates $v_\perp$, and*
   b) *for every pair $\langle v_\top, v_\perp\rangle \in W(\gamma)$ with $v_\perp \in V_\perp$, there is no pair $\langle v'_\top, v'_\perp\rangle \in W(\gamma)$, $\langle v'_\top, v'_\perp\rangle \neq \langle v_\top, v_\perp\rangle$ with $v_\top$ dominating $v'_\top$, $v'_\top$ dominating v, v strictly dominating $v'_\perp$, and $v'_\perp$ dominating $v_\perp$.*

2. *We then define the* wrapping degree *of v as the cardinality of its gap set.*
3. *The* wrapping degree *of a decorated derived tree is the maximal wrapping degree of its nodes, and the* wrapping degree *of a derived tree γ_d in the TWG G is the minimal wrapping degree of any decorated derived tree with respect to G that yields γ_d.*

In the example in Fig. 5, we have for instance a gap set $\{v_4, v_5\}$ for the node v_3. The wrapping degree of this derived tree is 2.

Now we can define the tree language for a TWG in general and in the case where wrapping degrees are limited by a $k \geq 0$:

Definition 3 (Language of a TWG). *Let G be a TWG.*

- *A* saturated *derived tree is a derived tree without substitution nodes and without split categories.*
- *The* tree language *of G is $L_T(G) = \{\gamma \mid \gamma$ is a saturated derived initial tree in $G\}$.*
- *The* string language *of G is the set of yields of trees in $L_T(G)$.*
- *The k-*tree language *of G is $L_T^k(G) = \{\gamma \mid \gamma$ is a saturated derived initial tree in G with a wrapping degree $\leq k\}$.*
- *The k-*string language *of G is the set of yields of trees in $L_T^k(G)$.*

Some TWGs are such that the maximal wrapping degree is limited, given the form of the elementary trees. But this is not always the case. In the following, we call a TWG a k-TWG if we impose k as a limit for the wrapping degree of derived trees, i.e., for a k-TWG, we consider the k-tree language of the grammar as its tree language.

As an example, Fig. 6 gives a TWG for the copy language. Here, all substitution nodes must be filled by wrapping substitutions since there are no trees with root label A, and the nodes with the split categories are always the middle nodes. The grammar is such that only derived trees with a wrapping degree 1 are possible.

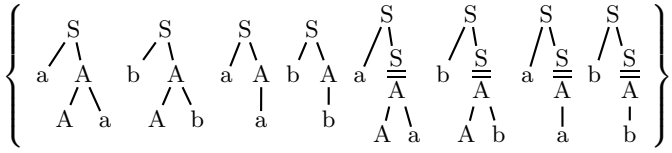
In order to facilitate the construction of an equivalent simple CFTG for a given k-TWG, we show the following normal form lemma:

Lemma 1. *For every k-TWG $G = \langle N, T, I, A_D, A_L, A_R, C\rangle$, there is exists a weakly equivalent k-TWG $G' = \langle N', T, I', \emptyset, \emptyset, \emptyset, C\rangle$, i.e., a k-TWG without adjunct trees.*

The construction idea is again rather simple. For every daughter position, we precompile the possibility to add something by sister adjunction in the following way: We start with $I_{temp} = I$ and $I' = \emptyset$ and we set $A_l = A_L \cup A_D$ and $A_r = A_R \cup A_D$.

1. For every adjunct tree β, we add a subscript l (r or d respectively) to the root label of β if β is in A_l (resp. in A_r or in A_D). The resulting tree is added to I_{temp}.

$G = \langle \{\text{S, A}\}, \{a, b\}, I, \emptyset, \emptyset, \{\text{S, A}\} \rangle$ with $I =$



Sample derivation of *baabaa*:

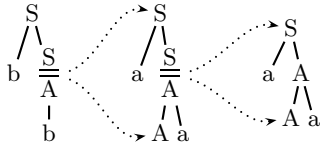


Fig. 6. TWG for the copy language $\{ww \mid w \in \{a, b\}^+\}$

2. For every $\gamma \in I_{temp}$: For every node v in γ that is not the root of a former adjunct tree: If v has i daughters, then we pick one combination $i_1, \ldots, i_k$ ($k \geq 0$) with $0 \leq i_1 < \ldots < i_k \leq i$ of positions between daughters. We then add new daughters to v at all these positions, labeled with the non-terminal of v and a subscript l for position 0, r for position i and d otherwise. The result is added to I'. This is repeated until I' does not change any more, i.e., all possible combinations of daughter positions for the nodes in γ have been taken into account.
3. For every $\gamma \in I'$ that is a former adjunct tree: Add
   - a tree γ_l to I' that consists of γ with an additional leftmost daughter of the root having the same label as the root and a subscript l in case the subscript of the root is l, d otherwise.
   - a tree γ_r to I' that consists of γ with an additional rightmost daughter of the root having the same label as the root and a subscript r in case the subscript of the root is r, d otherwise.
   - a tree γ_{lr} to I' that consists of γ with two additional daughters of the root, one leftmost and one rightmost daughter such that these daughters have the same label as the root and the following subscripts: the leftmost has a subscript l in case the subscript of the root is l, d otherwise. The rightmost has a subscript r in case the subscript of the root is r, d otherwise.

An example of this construction can be found in Fig. 7.

The equivalence of the original grammar and the constructed one is obvious. The latter requires the same number of wrapping substitutions as the original one and has the same string language for the same k. But it generates different derived trees since in cases of multiple adjunctions between two nodes we obtain a binary structure.

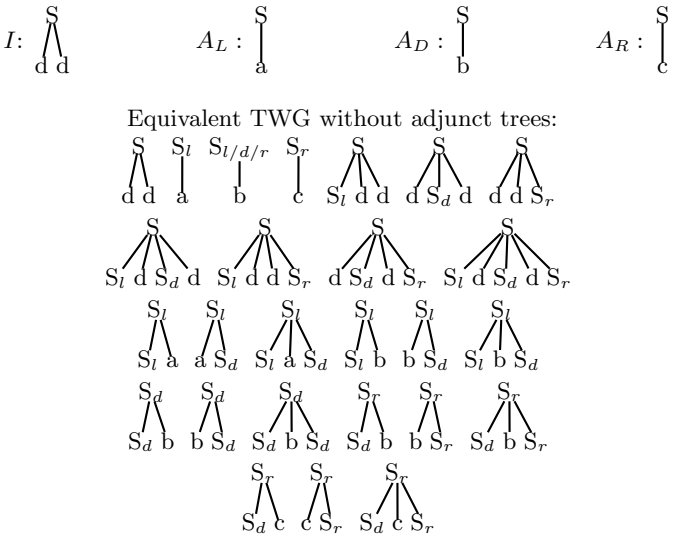


Fig. 7. Sample elimination of adjunct trees

3 Relation to context-free tree gramamrs

We now show that for every k-TWG one can construct an equivalent simple context-free tree grammar of rank k. This, in turn, is weakly equivalent to well-nested $(k+1)$-LCFRS (see [16, 13] for the definition of LCFRS and [7, 3] for well-nested LCFRS).

Without loss of generality, we assume the k-TWG to be without adjunct trees.

3.1 Context-free tree grammars

The following introduction to context-free tree grammars is taken from [8].

A ranked alphabet is a union $\Delta = \bigcup_{r\in\mathbb{N}} \Delta^{(r)}$ of disjoint sets of symbols. If $f \in \Delta^{(r)}$, r is the *rank* of f.

A tree over a ranked alphabet Δ is a labeled ordered tree where each node with n daughters is labeled by some $f \in \Delta^{(n)}$. We use the term representation of trees. The set $\mathcal{T}_\Delta$ of trees over Δ is defined as follows: 1. If $f \in \Delta^{(0)}$, then $f \in \mathcal{T}_\Delta$. 2. If $f \in \Delta^{(n)}$ and $t_1, \ldots, t_n \in \mathcal{T}_\Delta (n \geq 1)$, then $(ft_1 \ldots t_n) \in \mathcal{T}_\Delta$.

If Σ is an (unranked) alphabet and Δ a ranked alphabet ($\Sigma \cap \Delta = \emptyset$), let $\mathcal{T}_{\Sigma,\Delta}$ be the set of trees such that whenever a node is labeled by some $f \in \Delta$, then the number of its children is equal to the rank of f.

For a set $X = \{x_1, \ldots, x_n\}$ of variables, $\mathcal{T}_\Delta(X)$ denotes the set of trees over $\Delta \cup X$ where members of X all have rank 0. Such a tree t containing the variables X is often written $t[x_1, \ldots, x_n]$. If $t[x_1, \ldots, x_n] \in \mathcal{T}_\Delta(X)$ and $t_1, \ldots, t_n \in \mathcal{T}_\Delta$,

then $t[t_1, \ldots, t_n]$ denotes the result of substituting $t_1, \ldots, t_n$ for $x_1, \ldots, x_n$, respectively, in $t[x_1, \ldots, x_n]$. An element $t[x_1, \ldots, x_n] \in \mathcal{T}_\Delta(X)$ is an n-context over Δ if for each $i = 1, \ldots, n$, x_i occurs exactly once in $t[x_1, \ldots, x_n]$.

Definition 4 (Context-free tree grammar). *A* context-free tree grammar *(CFTG) [12, 2] is a quadruple $G = \langle N, \Sigma, P, S \rangle$, where*

1. *N is a ranked alphabet of non-terminals,*
2. *Σ an unranked alphabet of terminals,*
3. *$S \in N$ is of rank 0, and*
4. *P is a finite set of productions of the form*

$$Ax_1 \ldots x_n \to t[x_1, \ldots, x_n]$$

where $A \in N^{(n)}$ and $t[x_1, \ldots, x_n] \in \mathcal{T}_{\Sigma,N}(\{x_1, \ldots, x_n\})$.
The rank *of G is* $\max\{r \mid N^{(r)} \neq \emptyset\}$.

For every $s, s' \in \mathcal{T}_{\Sigma,N}$, $s \Rightarrow_G s'$ is defined to hold if and only if there is a 1-context $c[x_1] \in \mathcal{T}_{\Sigma,N}(\{x_1\})$, a production $Bx_1 ... x_n \to t[x_1, ..., x_n]$ in P, and trees $t_1, ..., t_n \in \mathcal{T}_{\Sigma,N}$ such that $s = c[Bt_1 \ldots t_n]$, $s' = c[t[t_1, \ldots, t_n]]$.

The relation $\Rightarrow_G^*$ is defined as the reflexive transitive closure of$\Rightarrow_G$. The tree language $L(G)$ generated by a CFTG G is defined as $\{t \in \mathcal{T}_\Sigma \mid S \Rightarrow_G^* t\}$. The string language is the set of yields of the trees in $L(G)$.

A CFTG is said to be simple if all right-hand sides of productions in the grammar are n-contexts, in other words, they contain exactly one occurrence of each of their n variables.

CFTG for $\{w^3 \mid w \in \{a, b\}^+\}$:
$N^0 = \{\overline{S}\}, N^{(3)} = \{\overline{X}\}$, $\Sigma = \{a, b, A\}$, S the start symbol.
P contains the following productions:
$\overline{S} \to \overline{X}aaa \mid \overline{X}bbb$
$\overline{X}x_1x_2x_2 \to \overline{X}(Aax_1)(Aa_2)(Aax_3) \mid \overline{X}(Abx_1)(Abx_2)(Abx_3) \mid Ax_1x_2x_3$
Sample derivation for the string *abaabaaba*:
$\overline{S} \Rightarrow \overline{X}aaa \Rightarrow \overline{X}(Aba)(Aba)(Aba)$
$\Rightarrow \overline{X}(Aa(Aba))(Aa(Aba))(Aa(Aba))$
$\Rightarrow A(Aa(Aba))(Aa(Aba))(Aa(Aba))$

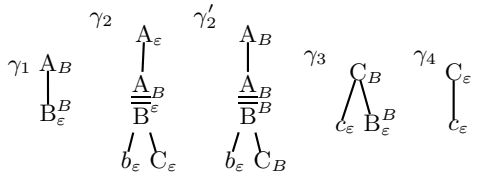
Fig. 8. Simple CFTG for the double copy language

3.2 *k*-TWG and simple CFTG

In the following, we will show that for each k-TWG, an equivalent simple context-free tree grammar of rank k can be constructed.

Let us explain the construction while going through the simple example in Fig. 9. The CFTG non-terminals have the form $[A, A_1 A_2 \ldots A_n]$ with $n \leq k$,

TWG for $\{(bc)^n \mid n \geq 1\} \cup \{c\}$:



Equivalent simple CFTG:
$N^{(0)} = \{S, [A], [C]\}, N^{(1)} = \{[A, B], [C, B]\}$, start symbol S, productions:
$S \to [A]$, $S \to [C]$
γ_1: $[A, B]x_1 \to Ax_1$
γ_2: $[A] \to A([A, B](Bb[C]))$
γ_2': $[A, B]x_1 \to A([A, B](Bb([C, B]x_1)))$
γ_3: $[C, B]x_1 \to Ccx_1$
γ_4: $[C] \to Cc$

Fig. 9. Sample TWG and equivalent simple CFTG

$A \in N$ and $A_i \in N$ for $1 \leq i \leq n$ where the intuition is the A is the root category of the tree this nonterminal expands to and $A_1 A_2 \ldots A_n$ are the categories of pending gaps from wrappings that stretch across this tree. In other words, in the final decorated derived tree, they are the categories of the gap set nodes of this root, in linear order. Note that, since we assume a k-TWG, there cannot be more than k such categories. The gap trees that are to be inserted in the gap nodes (which are substitution nodes) are the arguments of this non-terminal. In other words, such a non-terminal tells us that we have to find an A-tree and there are pending lower parts of split trees with categories $A_1, \ldots, A_n$, which have to be inserted into that A-tree. For instance, the category $[A, B]$ in our example expands to A-trees that need a B-substitution node at some point (maybe after some further substitution), in order to insert the B-gap tree to which this category applies. The γ_1-rule in the TWG for instance encodes that one way to find such a tree is to create an A-node with a single daughter, where this daughter is the pending B-tree.

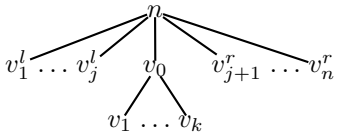
The construction does not go directly from an elementary TWG tree to a single production. Instead, it yields a single production for each possible decoration of the TWG tree with category sequences corresponding to possible gap set node labels in linear order that can arise within a derivation. An example where we have more than one possibility for a single TWG tree is the tree γ_2 in Fig. 9 where the γ_2 and γ_2' indicate the two cases. Accordingly, there are two productions. One $[A]$-production where $[A]$ is of rank 0. This is the case where nothing else is wrapped around γ_2 and, consequently, there is no pending gap at its root node. The second production (the γ_2' case) is the possibility to have something wrapped around the γ_2 tree. In this case, the gap category of the outer tree is B, and this gap must be placed somewhere below the C node, hence the non-terminal $[C, B]$ with the pending gap as argument for this node.

In order to keep track of these sequences of gap labels, we first define possible mappings f_1, f_2 for every elementary tree γ that assign to every node x in γ either a single sequence $f_1(x) = f_2(x)$ of non-terminals (= gap node labels) or, if the node is a split node or a substitution node that is used for a wrapping substitution, a pair of two possibly different such subsequences $\langle f_1(x), f_2(x)\rangle$. Intuitively, a split node starts a new gap, which is then filled by the lower part of the split node. Any gaps in the tree below the gap are accessible at the mother node of the split node.

Fig. 9 gives the assignments f_1 and f_2 for each node as a super- and a subscript. In cases where $f_1 = f_2$, there is just one subscript, while for $f_1 \neq f_2$ (split nodes and wrapping substitution nodes), we have both. For γ_2, we have two possible assignments. The mapping of γ_1 tells us that this tree is used in a wrapping configuration where a split tree with some lower category B is wrapped around it. Furthermore, according to the B-node annotation, this leaf is filled by the wrapping substitution ($f_2 = \varepsilon$). The first assignment for γ_2 tells us that this tree is used without wrapping anything around it. At its split node, we wrap it around something that has to contain a B-gap ($f_1 = B$), which will be filled by the lower part of the split tree, therefore at that point, no more gaps are pending ($f_2 = \varepsilon$). In contrast to this, the second γ_2 is used in a wrapping configuration where a split tree with some lower category B is wrapped around it ($f_1 = f_2 = B$ at the root). The B gap arising from the split node in the middle is filled by the lower B node. However, the overall B gap is still pending, therefore we have $f_2 = B$ at the split node. This pending gap is not inserted at the substitution node (category C), instead, the information about the B-gap is passed ($f_1 = f_2 = B$). It can be inserted in γ_3. There the substitution node has $f_1 = B$ (which means that we need a B-substitution node to be filled with some pending B-tree) and $f_2 = \varepsilon$ (signifying that this is the substitution node we were looking for, no more pending gaps below).

The definition of these assignments is such that we guess the pending gap categories for the leaves, we guess whether a substitution node is used for wrapping, and for split nodes, we guess the pending gap categories that arise out of the tree that this node is wrapped around. The rest is calculated in a bottom-up way as follows:

- For a substitution node v with category A: either v is not used for a wrapping substitution and we have $f_1(v) = f_2(v) = A_1 \dots A_i$ $(0 \leq i)$ or v is used for a wrapping substitution and we have $f_1(v) = A$ and $f_2(v) = \varepsilon$.
- $f_2(v_0) = f_1(v_1) \dots f_1(v_j)$ for every node v_0 with $v_1, \dots v_j$ being all daughters of v_0 in linear precedence order such that none of the daughters is a split node.
- For every split node v_0 with top label X and bottom label Y with $v_1, \dots v_k$ being all daughters of v_0 in linear precedence order, n being the mother of v_0 and $v_1^l, \dots, v_j^l$ and $v_{j+1}^r, \dots, v_n^r$ being the sisters of v_0 to the left and right in linear precedence order:



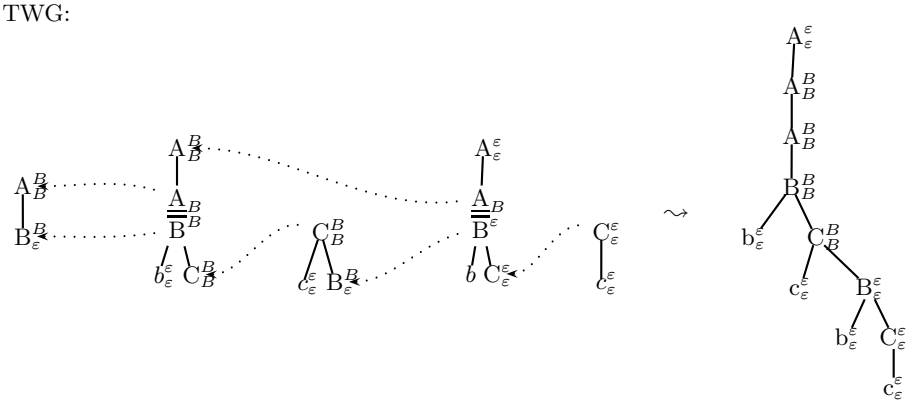
$f_2(v_0) = f_1(v_1) \dots f_1(v_k)$ and there are $B_1, \dots, B_j \in N$ such that
$f_1(v_0) = B_1 \dots B_i Y B_{i+1} \dots B_j$ and
$f_2(n) = f_1(v_1^l) \dots f_1(v_j^l) B_1 \dots B_i f_2(v_0) B_{i+1} \dots B_j f_1(v_{j+1}^r) \dots f_1(v_n^r)$.
We call the Y in this step the split category.

- For every node v that is neither a split node nor a non-terminal leaf, we have $f_1(v) = f_2(v)$.
- For every leaf v with a terminal label, we have $f_1(v) = f_2(v) = \varepsilon$.
- The length of the assigned sequences is limited to k.
- For every node v with a non-terminal label from $N \setminus C$ (C is the set of categories allowed on wrapping spines), it holds that $f_1 = f_2 = \varepsilon$.

Instead of using the original TWG, we can also use the trees with annotations f_1, f_2 in TWG derivations. For these derivations, let us make the following assumptions: The conditions for wrapping are that the f_1 value of the split node must be the f_1 value of the root of the target tree while the bottom category of the split node must be the f_1 of the target substitution node and the f_2 of this substitution node must be ε. The annotation of the root of the target tree remains while the annotation of the substitution node is the f_2 value of the split node. Furthermore, annotations of substitution nodes that are not used for wrapping have to be equal to the ones of the root of the tree that substitutes in.

An example of such a TWG derivation can be found in Fig. 10.

Sample derivations of $w = bcbc$:
TWG:



Corresponding CFTG derivation:
$S \Rightarrow [A] \Rightarrow A([A, B](Bb[C])) \Rightarrow A([A, B](Bb(Cc)))$
$\Rightarrow A(A([A, B](Bb[C, B](Bb(Cc))))) \Rightarrow A(A(A(Bb[C, B](Bb(Cc)))))$
$\Rightarrow A(A(A(Bb(Cc(Bb(Cc))))))$

Fig. 10. Sample derivations in the grammars from Fig. 9

For these TWG derivations, the following lemma holds:

Lemma 2. *With this annotated TWG we obtain exactly the set of derived trees of the original TWG including for each node in a derived tree, obtained with a specific derivation, a decoration with the labels of the nodes from its gap set in linear precedence order.*

This lemma holds since all possible combinations of pending gaps below substitution nodes and to the left and right of split nodes are considered in the f_1, f_2 annotations. Furthermore, gaps are passed upwards. The only way to get rid of a gap in the f_1, f_2 value of a root node is to wrap a tree filling this gap around it.

Given the gap assignment definition, we can now specify the set of productions in our CFTG that we obtain for each elementary tree.

1. For every non-terminal category X in our TWG, we add a production

$$S \to [X]$$

   which is used for derived trees with root category X.
2. For every tree γ in the k-TWG with root r and root category A and for every assignment $f = \langle f_1, f_2 \rangle$ for γ as defined above, we have productions

$$[A, f_1(r)]y_1 \dots y_{|f_1(r)|} \to \tau(\gamma, f)$$

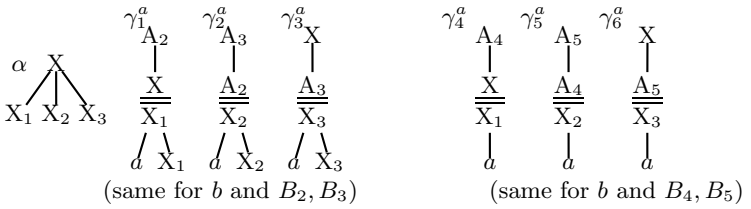
   where $\tau(\beta, f)$ for any subtree β of a TWG tree with gap assignment f is defined as follows:

   - If β has only a single node v with non-terminal category B and $f_1(v) = f_2(v)$, then $\tau(\beta, f) = [B, f_1(v)]x_1 \dots x_{|f_1(v)|}$.[1]
   - If β has only a single node v with non-terminal category B and $f_1(v) = A \in N$, $f_2 = \varepsilon$, then $\tau(\beta, f) = x_1$.
   - If the root v of β is not a split node, its root category is A, and if $\beta_1 \dots, \beta_n$ are the daughter trees of v, then $\tau(\beta, f) = (A\tau(\beta_1, f) \dots \tau(\beta_n, f))$.
   - If the root v of β is a split node with top category A and bottom category B, and if $\beta_1 \dots, \beta_n$ are the daughter trees of v, then $\tau(\beta, f) = ([A, f_1(v)]x_1 \dots x_i(B\tau(\beta_1, f) \dots \tau(\beta_n, f))x_{i+1} \dots x_j)$. where $f_1(v) = A_1 \dots A_i B A_{i+1} \dots A_j$ and B is the split category from the construction of f.

   The variables $y_1, \dots, y_{|f_1(r)|}$ in the lefthand side of the production are exactly the ones from the righthand side in linear precedence order.
3. These are all the productions in the CFTG.

[1] We assume that fresh variables are used each time a new variable is needed.

TWG for the double copy language $\{w^3 \mid w \in \{a,b\}^+\}$:



Equivalent simple CFTG:
Start symbol S, productions:

$S \to [X]$
γ_6^a: $[X] \to X([A_5, X_3](X_3 a))$
γ_5^a: $[A_5, X_3]x_1 \to A_5([A_4, X_2X_3](X_2 a)x_1)$
γ_4^a: $[A_4, X_2X_3]x_1x_2 \to A_4([X, X_1X_2X_3](X_1 a)x_1x_2)$
γ_3^a: $[X, X_1X_2X_3]x_1x_2x_3 \to X([A_3, X_1X_2X_3]x_1x_2(X_3 a x_3))$
γ_2^a: $[A_3, X_1X_2X_3]x_1x_2x_3 \to A_3([A_2, X_1X_2X_3]x_1(X_2 a x_2)x_3)$
γ_1^a: $[A_2, X_1X_2X_3]x_1x_2x_3 \to A_2([X, X_1X_2X_3](X_1 a x_1)x_2x_3)$
(same with b and B_2, B_3, B_4, B_5)
α: $[X, X_1X_2X_3]x_1x_2x_3 \to Xx_1x_2x_3$

Fig. 11. Sample 3-TWG and equivalent simple CFTG

An example of this construction can be found in Fig. 9, and a sample derivation in the TWG and the corresponding CFTG is given in Fig. 10. The TWG derivation involves two wrappings of γ_2 around γ_1, the first (inner one) with an additional substitution of γ_3 into the C substitution node, the second, outer one with a substitution of γ_4 into this slot. The corresponding CFTG derivation starts by expanding $[A]$ to the tree corresponding to the outer wrapping of γ_2, with a non-terminal $[C]$ for γ_4. Inside the resulting tree, we have a non-terminal $[A, B]$ of rank 1 whose argument is the tree $Bb(Cc)$) which has to fill a B-gap. This is then expanded to a γ_2 tree that assumes that there is a B-gap below its C-substitution node. This second use of γ_2 creates again the request for an A-tree with a B-gap (non-terminal $[A, B]$), which is now filled by γ_1, and, below its substitution node, it needs a C-tree with a B-substitution node (non-terminal $[C, B]$) which can then be filled by the pending B-tree $Bb(Cc)$) from the outer use of γ_2. Such a tree is provided by γ_3.

As a further example consider the TWG and corresponding CFTG in Fig.11.

The crucial part of the construction is actually the definition of the f_1, f_2 gap category annotations. Once we have this, the following holds:

Lemma 3. *There is a derived tree γ in the TWG with pending gap category sequence annotations $f = \langle f_1, f_2 \rangle$ (written $\langle \gamma, f \rangle$) as described above iff there is a corresponding derivation in the CFTG.*

Here, "corresponding derivation" means the following: if γ has gap sequence annotations f_1, f_2 and a root node v with node label A, then the correponding

derivation is of the form $[A, f_1(v)]y_1 \dots y_{|f_1(v)|} \Rightarrow_G^* \tau(\gamma, f)$ where $\tau(\gamma, f)$ is as defined above in the construction for the case of elementary trees.

We can show this by an induction over the derivation structure, proving that

- The claim holds for elementary trees. This follows immediately from the construction.
- We assume that the claim holds for $\langle \gamma, f \rangle$ and $\langle \alpha, f_\alpha \rangle$ with corresponding CFTG derivations $[A_\gamma, \mathit{gaps}_\gamma]\boldsymbol{x}_\gamma \Rightarrow_G^* \tau(\gamma, f)$ and $[A_\alpha, \mathit{gaps}_\alpha]\boldsymbol{x}_\alpha \Rightarrow_G^* \tau(\alpha, f_\alpha)$.
  Then:
  $\langle \gamma', f' \rangle$ can be derived from $\langle \gamma, f \rangle$ in the TWG via substitution of $\langle \alpha, f_\alpha \rangle$ into one of the non-termial leaves
  $\Leftrightarrow$ this non-terminal leaf in $\langle \gamma', f' \rangle$ has category A_α and gap sequence gaps_α
  $\Leftrightarrow$ there is a corresponding non-terminal $[A_\alpha, \mathit{gaps}_\alpha]$ in $\tau(\gamma, f)$ that can be expanded using the derivation $[A_\alpha, \mathit{gaps}_\alpha]\boldsymbol{x} \Rightarrow_G^* \tau(\alpha, f_\alpha)$ (induction assumption)
  $\Leftrightarrow$ $[A_\gamma, \mathit{gaps}_\gamma]\boldsymbol{x} \Rightarrow_G^* \tau(\gamma', f')$ where, in this derivation, we have one part $[A_\gamma, \mathit{gaps}_\gamma]\boldsymbol{x}_\gamma \Rightarrow_G^* \tau(\gamma, f)$ and a second part consisting of an application of $[A_\alpha, \mathit{gaps}_\alpha]\boldsymbol{x}_\alpha \Rightarrow_G^* \tau(\alpha, f_\alpha)$.
- We assume that the claim holds for $\langle \gamma, f \rangle$ and $\langle \beta, f_\beta \rangle$ with corresponding CFTG derivations $[A_\gamma, \mathit{gaps}_\gamma]\boldsymbol{x}_\gamma \Rightarrow_G^* \tau(\gamma, f)$ and $[A_\beta, \mathit{gaps}_\beta]\boldsymbol{x}_\beta \Rightarrow_G^* \tau(\beta, f_\beta)$.
  Then:
  $\langle \gamma', f' \rangle$ is derived from $\langle \gamma, f_\gamma \rangle$ in the TWG by wrapping $\langle \beta, f_\beta \rangle$ around $\langle \gamma, f_\gamma \rangle$
  $\Leftrightarrow$ there is a split node in $\tau(\beta, f_\beta)$ with category $\langle A_\gamma, Y \rangle$ and with an f_1 value gaps_γ and there is a non-terminal leaf in $\langle \gamma, f_\gamma \rangle$ with category Y and with $f_1 = Y$ and $f_2 = \varepsilon$
  $\Leftrightarrow$ there is a non-terminal $[A_\gamma, \mathit{gaps}_\gamma]$ corresponding to the split node in $\tau(\beta, f_\beta)$ that can be expanded by the derivation $[A_\gamma, \mathit{gaps}_\gamma]\boldsymbol{x}_\gamma \Rightarrow_G^* \tau(\gamma, f)$
  $\Leftrightarrow$ there is a derivation $[A_\beta, \mathit{gaps}_\beta]\boldsymbol{x}_\beta \Rightarrow_G^* \tau(\gamma', f')$ in the CFTG consisting of $[A_\beta, \mathit{gaps}_\beta]\boldsymbol{x}_\beta \Rightarrow_G^* \tau(\beta, f_\beta)$ and then an application of $[A_\gamma, \mathit{gaps}_\gamma]\boldsymbol{x}_\gamma \Rightarrow_G^* \tau(\gamma, f)$.

With this lemma, we obtain the following theorem:

Theorem 1. *For every k-TWG there is an equivalent simple CFTG of rank k.*

As a consequence, we obtain that the languages of k-TWGs are in the class of well-nested linear context-free rewriting languages[2] and therefore mildly context-sensitive [16]. This term, introduced by [4], characterizes formalisms beyond CFG that can describe cross-serial dependencies, that are polynomially parsable and that generate languages of constant growth. Joshi's conjecture is that mildly context-sensitive grammar formalisms describe the appropriate grammar class for dealing with natural languages.

[2] Note that the fact that we can construct an equivalent well-nested LCFRS for a k-TWG does not mean that k-TWG (for some fixed k) cannot deal with ill-nested dependencies. The structures described by the LCFRS do not correspond to the dependency structures obtained from TWG derivations. The latter are determined only by the fillings of substitution slots.

4 Conclusion

We have shown that k-TWG is a mildly context-sensitive grammar formalism, more particular, it falls into the class of simple context-free tree languages of rank k/well-nested $(k+1)$-LCFRS. This is an interesting result, considering that TWG arose out of an attempt to formalize the syntactic inventory of RRG, a grammar theory that emerged from broad empirical linguistic studies. Therefore the formal results in this paper support in a convincing way Joshi's conjecture about the mild context-sensitivity of natural languages.

References

1. Chiang, D., Scheffler, T.: Flexible composition and delayed tree-locality. In: TAG+9 Proceedings of the Ninth International Workshop on Tree-Adjoining Grammar and Related Formalisms (TAG+9). pp. 17–24. Tübingen (June 2008)
2. Engelfriet, J., Schmidt, E.M.: IO and OI. Journal of Computer and System Sciences (15), 328–353 (1977)
3. Gómez-Rodríguez, C., Kuhlmann, M., Satta, G.: Efficient parsing of well-nested linear context-free rewriting systems. In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics. pp. 276–284. Association for Computational Linguistics, Los Angeles, California (June 2010), `http://www.aclweb.org/anthology/N10-1035`
4. Joshi, A.K.: Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions? In: Dowty, D., Karttunen, L., Zwicky, A. (eds.) Natural Language Parsing, pp. 206–250. Cambridge University Press (1985)
5. Joshi, A.K., Schabes, Y.: Tree-Adjoning Grammars. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, pp. 69–123. Springer, Berlin (1997)
6. Kallmeyer, L., Osswald, R., Van Valin, Jr., R.D.: Tree wrapping for Role and Reference Grammar. In: Morrill, G., Nederhof, M.J. (eds.) Formal Grammar 2012/2013. Lecture Notes in Computer Science, vol. 8036, pp. 175–190. Springer, Berlin, Heidelberg (2013)
7. Kanazawa, M.: The pumping lemma for well-nested Multiple Context-Free Languages. In: Diekert, V., Nowotka, D. (eds.) DLT 2009. LNCS, vol. 5583, pp. 312–325. Springer, Berlin Heidelberg (2009)
8. Kanazawa, M.: Multidimensional trees and a Chomsky-Schützenberger-Weir representation theorem for simple context-free tree grammars. Journal of Logic and Computation (Published online June 30, 2014)
9. Osswald, R., Kallmeyer, L.: Towards a formalization of Role and Reference Grammar. In: Proceedings of the 2013 Conference on Role and Reference Grammar (to appear)
10. Rambow, O., Vijay-Shanker, K., Weir, D.: D-Tree Grammars. In: Proceedings of ACL (1995)
11. Rambow, O., Vijay-Shanker, K., Weir, D.: D-Tree Substitution Grammars. Computational Linguistics (2001)
12. Rounds, W.C.: Mappings and grammars on trees. Mathematical Systems Theory (4), 257–287 (1970)
13. Seki, H., Matsumura, T., Fujii, M., Kasami, T.: On multiple context-free grammars. Theoretical Computer Science 88(2), 191–229 (1991)

14. Van Valin, Jr., R.D.: Exploring the Syntax-Semantics Interface. Cambridge University Press (2005)
15. Van Valin, Jr., R.D., Foley, W.A.: Role and reference grammar. In: Moravcsik, E.A., Wirth, J.R. (eds.) Current approaches to syntax, Syntax and semantics, vol. 13, pp. 329–352. Academic Press, New York (1980)
16. Vijay-Shanker, K., Weir, D.J., Joshi, A.K.: Characterizing structural descriptions produced by various grammatical formalisms. In: Proceedings of ACL. Stanford (1987)

Distributional Learning and Context/Substructure Enumerability in Nonlinear Tree Grammars

Makoto Kanazawa[1] and Ryo Yoshinaka[2]

[1] National Institute of Informatics and SOKENDAI
[2] Graduate University of Informatics, Kyoto University

Abstract. We study tree-generating almost linear second-order ACGs that admit bounded nonlinearity either on the context side or on the substructure side, and give distributional learning algorithms for them.

1 Introduction

Originally developed for efficient learning of context-free languages [3, 13], the method of *distributional learning* under the paradigm of *identification in the limit from positive data and membership queries* has been successfully applied to a number of more complex grammatical formalisms that derive objects (strings, trees, λ-terms, etc.) through local sets of *derivation trees* [9, 12, 14]. In these formalisms, a subtree s of a complete derivation tree $t = c[s]$ contributes a certain "substructure" $S = \phi(s)$ which is contained in the whole derived object $T = \phi(t)$, and the remaining part $c[\,]$ of the derivation tree contributes a function $C = \phi(c[\,])$ that maps S to $T = C(S)$. We can think of C as a "context" that surrounds S in T. Fixing a class $\mathbb{G}$ of grammars fixes the set $\mathbb{S}$ of possible substructures and the set $\mathbb{C}$ of possible contexts that may be contributed by parts of possible derivation trees. Each language L generated by a grammar in $\mathbb{G}$ acts as an arbiter that decides which context $C \in \mathbb{C}$ should "accept" which substructure $S \in \mathbb{S}$ (i.e., whether $C(S) \in L$).

Distributional learning algorithms come in two broad varieties. In the *primal* approach, the learner first extracts all substructures and all contexts that are contained in the input data, which is a finite set of elements of the target language L_*. The learner then collects all subsets of the extracted substructures whose cardinality does not exceed a certain fixed bound m. These subsets are used as nonterminal symbols of the hypothesized grammar. Out of all possible grammar rules that can be written using these nonterminals, the learner lists those that use operations that may be involved in the generation of the objects in the input data. In the final step of the algorithm, the learner tries to validate each of these rules with the membership oracle, which answers a query "$C(S) \in L_*$?" in constant time. If a rule has a set $\mathbf{S}$ of substructures on the left-hand side and sets $\mathbf{S}_1, \ldots, \mathbf{S}_r$ on the right-hand side, and the grammatical operation associated with the rule is f, then the learner determines whether the following implication

holds for all contexts C extracted from the input data:

$$C(S) \in L_* \text{ for all } S \in \mathbf{S} \text{ implies}$$
$$C(f(S_1, \ldots, S_n)) \in L_* \text{ for all } S_1 \in \mathbf{S}_1, \ldots, S_n \in \mathbf{S}_n. \quad (1)$$

The grammar conjectured by the learner includes only those rules that pass this test.

The idea of the rule validation is the following: It is dictated that the elements of the nonterminal $\mathbf{S}$ together *characterize* the set of all substructures that can be derived from $\mathbf{S}$ by the hypothesized grammar in the sense that every context $C \in \mathbb{C}$ that accepts all elements of $\mathbf{S}$ must accept all substructures derived from $\mathbf{S}$. Thus, only those rules that are consistent with this requirement are allowed in the hypothesized grammar. A remarkable property of the algorithm is that it successfully learns the language of every grammar in the given class $\mathbb{G}$ that has the *m-finite kernel property* in the sense that each nonterminal is characterized by a set of substructures of cardinality up to m.

In the *dual* approach to distributional learning, the role of contexts and substructures is switched. The learner uses as nonterminals subsets of the contexts extracted from the input data with cardinality $\leq m$, and uses the extracted substructures to validate candidate rules. The algorithm learns those languages that have a grammar with the *m-finite context property* in the sense that each nonterminal is characterized by a set of contexts of cardinality $\leq m$.

Whether each of these algorithms runs in polynomial time in the size of the input data D depends on several factors that are all determined by the grammar class $\mathbb{G}$. The foremost among them is the enumeration of the two sets

$$\mathbb{S}|_D = \{\, S \in \mathbb{S} \mid C(S) \in D \text{ for some } C \in \mathbb{C} \,\},$$
$$\mathbb{C}|_D = \{\, C \in \mathbb{C} \mid C(S) \in D \text{ for some } S \in \mathbb{S} \,\}.$$

There are two possible difficulties in enumerating each of these sets in polynomial time. First, the sheer number of elements of the set may be super-polynomial, in which case explicit enumeration of the set is not possible in polynomial time. Second, recognizing which substructure/context belongs to the set may be computationally costly. The second problem, even when it arises, can often be dealt with by replacing the set in question by a more easily recognizable superset without disrupting the working of the algorithm. The first problem is the more pressing one.

With all *linear* grammar formalisms to which distributional learning has been applied, neither of these two difficulties arise. When these formalisms are extended to allow nonlinearity in grammatical operations, however, the problem of super-polynomial cardinality hits hard. Thus, with *parallel multiple context-free grammars*, the nonlinear extension of *multiple context-free grammars* (successfully dealt with in [12]), the set $\mathbb{C}$ becomes a much larger set, even though $\mathbb{S}$ stays exactly the same. As a result, the cardinality of $\mathbb{C}|_D$ is no longer bounded by a polynomial. The situation with *IO context-free grammars*, the nonlinear extension of the *simple context-free tree grammars* (treated in [9]), is even worse. Both of the sets $\mathbb{S}|_D$ and $\mathbb{C}|_D$ become super-polynomial in cardinality.

When only one of the two sets $\mathbb{S}|_D$ and $\mathbb{C}|_D$ is of super-polynomial cardinality, as is the case with PMCFGs, however, there is a way out of this plight [4]. The solution is to restrict the offending set by a certain property, parametrized by a natural number, so that its cardinality will be polynomial. The parametrized restriction leads to an increasing chain of subsets inside $\mathbb{S}$ or $\mathbb{C}$. In the case of PMCFGs, we get $\mathbb{C}_1 \subset \mathbb{C}_2 \subset \mathbb{C}_3 \subset \cdots \subset \mathbb{C} = \bigcup_k \mathbb{C}_k$, where $\mathbb{C}_k$ is the set of all possible contexts that satisfy the property with respect to the parameter k. The actual property used by [4] was a measure of nonlinearity of the context ("k-copying"), but this specific choice is not crucial for the correct working of the algorithm, as long as $\mathbb{C}_k|_D$ can be enumerated in polynomial time. The learning algorithm now has two parameters, m and k: the former is a bound on the cardinality of sets of contexts the learner uses as nonterminals as before, and the latter is a restriction on the kind of context allowed in these sets. The class of languages successfully learned by the algorithm includes the languages of all grammars in the target class that have the (k, m)-*finite context-property* in the sense that each nonterminal is characterized by a subset of $\mathbb{C}_k$ of cardinality $\leq m$.

This algorithm does not learn the class of all grammars with the m-finite context property, but a proper subset of it. Nevertheless, the parametrized restriction has a certain sense of naturalness, and the resulting learnable class properly extends the corresponding linear class, so the weaker result is interesting in its own right.

In this paper, we explore the connection between distributional learning and context/substructure enumerability in the general setting of *almost linear second-order abstract categorial grammars* generating trees [5–7] ("almost linear ACGs" for short). This class of grammars properly extends IO context-free tree grammars and is equivalent in tree generating power to *tree-valued attribute grammars* [1]. In fact, the expressive power of typed lambda calculus makes it possible to faithfully encode most known tree grammars within almost linear ACGs.

Like IO context-free tree grammars and unlike PMCFGs, almost linear ACGs in general do not allow polynomial-time enumerability either on the context side or on the substructure side. Only very special grammars do, and an interesting subclass of them consists of those grammars that allow only a bounded degree of nonlinearity in the contexts (or in the substructures). It is easily decidable whether a given ACG satisfies each of these properties. We show that both of the resulting classes of grammars indeed allow a kind of efficient distributional learning similar to that for PMCFGs.

2 Typed Lambda Terms and Almost Linear ACGs

2.1 Types and Typed Lambda Terms

We assume familiarity with the notion of a *simply typed λ-term* (à la Church) over a *higher-order signature* $\Sigma = (A_\Sigma, C_\Sigma, \tau_\Sigma)$, where A_Σ is the set of *atomic*

types, C_Σ is the set of constants, and τ_Σ is a function from C_Σ to types over A_Σ. We use standard abbreviations: $\alpha_1 \to \cdots \to \alpha_n \to p$ means $\alpha_1 \to (\cdots \to (\alpha_n \to p) \dots)$, and $\lambda x_1^{\alpha_1} \dots x_n^{\alpha_n}.MN_1 \dots N_m$ is short for $\lambda x_1^{\alpha_1}. \dots . \lambda x_n^{\alpha_n}.((\dots (MN_1) \dots) N_m)$. The *arity* of $\alpha = \alpha_1 \to \cdots \to \alpha_n \to p$ with $p \in A_\Sigma$ is $\mathrm{arity}(\alpha) = n$. We write $\beta^n \to p$ for the type $\beta \to \cdots \to \beta \to p$ of arity n.

We take for granted such notions as β- and η-*reduction*, β-*normal form*, and *linear* λ-*terms*. We write $\twoheadrightarrow_\beta$ and $\twoheadrightarrow_\eta$ for the relations of β- and η-reduction between λ-terms. Every typed λ-term has a β-normal form, unique up to renaming of bound variables, which we write as $|M|_\beta$.

The set $\mathrm{LNF}_X^\alpha(\Sigma)$ of λ-terms of type α in η-*long* β-*normal form* (with free variables from X) is defined inductively as follows:

- If $x^{\alpha_1 \to \cdots \to \alpha_n \to p} \in X$, $M_1 \in \mathrm{LNF}_X^{\alpha_1}(\Sigma), \dots, M_n \in \mathrm{LNF}_X^{\alpha_n}(\Sigma)$, and $p \in A_\Sigma$, then $x^{\alpha_1 \to \cdots \to \alpha_n \to p} M_1 \dots M_n \in \mathrm{LNF}_X^p(\Sigma)$.
- If $c \in C_\Sigma$, $\tau_\Sigma(c) = \alpha_1 \to \cdots \to \alpha_n \to p$, $p \in A_\Sigma$, and $M_1 \in \mathrm{LNF}_X^{\alpha_1}(\Sigma), \dots, M_n \in \mathrm{LNF}_X^{\alpha_n}(\Sigma)$, then $cM_1 \dots M_n \in \mathrm{LNF}_X^p(\Sigma)$.
- If $M \in \mathrm{LNF}^\beta_{X \cup \{x^\alpha\}}(\Sigma)$, then $\lambda x^\alpha.M \in \mathrm{LNF}_X^{\alpha \to \beta}(\Sigma)$.

We often suppress the superscript and/or subscript in $\mathrm{LNF}_X^\alpha(\Sigma)$. Note that $\mathrm{LNF}_\varnothing^\alpha(\Sigma)$ denotes the set of *closed* λ-terms of type α in η-long β-normal form. We note that if $M \in \mathrm{LNF}^{\alpha \to \beta}(\Sigma)$ and $N \in \mathrm{LNF}^\alpha(\Sigma)$, then $|MN|_\beta \in \mathrm{LNF}^\alpha(\Sigma)$.

Henceforth, we often suppress the type superscript on variables. This is just for brevity; each variable in a typed λ-term comes with a fixed type.

We use strings over $\{0, 1\}$ to refer to positions inside a λ-term or a type. We write ε for the empty string, and write $u \le v$ to mean u is a prefix of v. When $u = u'0^i$, we refer to u' as $u0^{-i}$.

The *shape* of a type α, written $[\alpha]$, is defined by

$$[p] = \{\varepsilon\} \text{ if } p \text{ is atomic}, \qquad [\alpha \to \beta] = \{\varepsilon\} \cup \{\, 1u \mid u \in [\alpha] \,\} \cup \{\, 0u \mid u \in [\beta] \,\}.$$

The elements of $[\alpha]$ are the *positions* of α. A position u is *positive* if its parity (i.e., the number of 1s in u modulo 2) is 0, and *negative* if its parity is 1. We write $[\alpha]^+$ and $[\alpha]^-$ for the set of positive and negative positions of α, respectively. A position u of α is a *subpremise* if $u = u'1$ for some u'. Such an occurrence is a *positive* (resp. *negative*) *subpremise* if it is positive (resp. negative). We write $[\alpha]^+_{\mathrm{sp}}$ (resp. $[\alpha]^-_{\mathrm{sp}}$) for the set of positive (resp. negative) subpremises of $[\alpha]$.

If $u \in [\alpha]$, the *subtype* of α occurring at u, written α/u, is defined by

$$\alpha/\varepsilon = \alpha, \quad (\alpha \to \beta)/0u = \beta/u, \quad (\alpha \to \beta)/1u = \alpha/u.$$

If $\alpha/u = \beta$, we say that β *occurs* at position u in α.

Given a λ-term M, the *shape* of M, written $[M]$, is defined by

$$\begin{aligned} [M] &= \{\varepsilon\} \quad \text{if } M \text{ is a variable or a constant,} \\ [MN] &= \{\varepsilon\} \cup \{\, 0u \mid u \in [M] \,\} \cup \{\, 1u \mid u \in [N] \,\}, \\ [\lambda x.M] &= \{\varepsilon\} \cup \{\, 0u \mid u \in [M] \,\}. \end{aligned}$$

The elements of $[M]$ are the *positions* of M.

If $u \in [M]$, the *subterm* of M occurring at u, written M/u, is defined by

$$M/\varepsilon = M, \quad (MN)/0u = M/u, \quad (MN)/1u = N/u, \quad (\lambda x.M)/0u = M/u.$$

When $N = M/u$, we sometimes call u an *occurrence of* N (in M).

When $v \in [M]$ but $v0 \notin [M]$, M/v is a variable or a constant. For each $u \in [M]$, we refer to the unique occurrence of a variable or constant in $[M]$ of the form $u0^k$ as the *head* of u (in M); we also call the variable or constant occurring at the head of u the *head* of M/u.

A position $v \in [M]$ *binds* a position $u \in [M]$ if M/u is a variable x and v is the longest prefix of u such that M/v is a λ-abstract of the form $\lambda x.N$. When v binds u in M, we write $v = b_M(u)$. When every occurrence in M of a λ-abstract is the binder of some position, M is called a *λI-term*.

Let $M \in \mathrm{LNF}^{\alpha}_{\varnothing}(\Sigma)$. Note that an occurrence $v \in [M]$ of a variable or a constant of type β with $\mathrm{arity}(\beta) = n$ is always accompanied by n arguments, so that $v0^{-i}$ is defined for all $i \leq n$. The set of *replaceable* occurrences [2] of bound variables in M and the negative subpremise $\mathrm{nsp}_M(u)$ of α associated with such an occurrence u, are defined as follows:[3]

(i) If $b_M(u) = 0^{j-1}$ for some $j \geq 1$ (i.e., $b_M(u)$ is the jth of the leading λs of M), then u is replaceable and $\mathrm{nsp}_M(u) = 0^{j-1}1$.
(ii) If $b_M(u) = v0^{-i}10^{j-1}$ for some replaceable v and $i, j \geq 1$ (i.e., $b_M(u)$ is the jth of the leading λs of the ith argument of v), then u is replaceable and $\mathrm{nsp}_M(u) = \mathrm{nsp}_M(v)0^{i-1}10^{j-1}1$.

It is easy to see that the following conditions always hold:

- If u is a replaceable occurrence of a bound variable x^{β}, then $\beta = \alpha/\mathrm{nsp}_M(u)$.
- If M is a λI-term (in addition to belonging to $\mathrm{LNF}^{\alpha}_{\varnothing}(\Sigma)$), then for every $v \in [\alpha]^{-}_{\mathrm{sp}}$, there exists a $u \in [M]$ such that $\mathrm{nsp}_M(u) = v$.

Example 1. Let

$$M = \lambda y_1^o y_2^{o\to(o\to(o\to o)\to o)\to o}.y_2(fy_1a)(\lambda y_3^o y_4^{o\to o}.f(y_4(fy_3y_1))(y_4(fy_3y_1))).$$

Then $M \in \mathrm{LNF}^{\alpha}_{\varnothing}(\Delta)$, where Δ contains constants f, a of type $o \to o \to o$ and o, respectively, and

$$\alpha = \overset{1}{o} \to (o \to (\overbrace{\overbrace{\overset{01011}{o} \to \underbrace{(o \to o)}_{010101}}^{0101} \to o) \to o}^{01}) \to o.$$

[3] A definition equivalent to $\mathrm{nsp}_M(u)$ for untyped λ-terms is in [2] (*access path*). The correspondence between these paths and negative subpremises for typed linear λ-terms is in [10].

– The bound variable y_1^o occurs in M at three positions, 000101, 001000111, 00100111, whose binder is ε. These positions are associated with the negative subpremise 1 in α.
– The bound variable $y_2^{o\to(o\to(o\to o)\to o)\to o}$ occurs in M at one position, 0000, whose binder is 0. This position is associated with the subpremise 01 in α.
– The bound variable y_3 occurs in M at two positions, 0010001101 and 001001101, whose binder is 001. These positions are associated with the negative subpremise 0101.
– The bound variable y_4 occurs in M at two positions, 00100010 and 0010010, whose binder is 0010. These positions are associated with the negative subpremise 010101.

2.2 Almost Linear Lambda Terms over a Tree Signature

Now we are going to assume that Δ is a tree signature; i.e., every constant of Δ is of type $o^r \to o$ for some $r \geq 0$, where o is the only atomic type of Δ. For a closed $M \in \mathrm{LNF}^{\alpha}_{\varnothing}(\Delta)$, every occurrence of a bound variable in M is replaceable.

A *tree* is an element of $\mathrm{LNF}^{o}_{\varnothing}(\Delta)$. A closed λ-term $M \in \mathrm{LNF}^{o^r\to o}_{\varnothing}(\Delta)$ is called a *tree context*. We say that a tree context $M = \lambda x_1 \dots x_r.N$ *matches* a tree T if there are trees $T_1, \dots, T_r$ such that $(\lambda x_1 \dots x_r.N)T_1 \dots T_r \twoheadrightarrow_\beta T$. We say that M is *contained* in T if it matches a subtree of T.

The notion of an *almost linear λ-term* was introduced by Kanazawa [5, 7]. Briefly, a closed typed λ-term is almost linear if every occurrence of a λ-abstract $\lambda x^\alpha.N$ in it binds a unique occurrence of x^α, unless α is atomic, in which case it may bind more than one occurrence of x^α. Almost linear λ-terms share many of the properties of linear λ-terms; see [5–8] for details.

Almost linear λ-terms are typically not β-normal. For instance, $\lambda y^{o\to o}.(\lambda x^o.fxx)(yc)$, where f and c are constants of type $o \to o \to o$ and o, respectively, is almost linear, but its β-normal form, $\lambda y^{o\to o}.f(yc)(yc)$, is not. In this paper, we choose to deal with the η-long β-normal forms of almost linear λ-terms directly, rather than through their almost linear β-expanded forms.

We write $\mathrm{AL}^{\alpha}(\Delta)$ for the set of *closed* λ-terms in $\mathrm{LNF}^{\alpha}_{\varnothing}(\Delta)$ that β-expand to an almost linear λ-term. (The superscript is often omitted.) The following lemma, which we do not prove here, may be taken as the definition of $\mathrm{AL}^{\alpha}(\Delta)$ (see [7, 8] for relevant properties of almost linear λ-terms):

Lemma 1. *Let M be a closed λI-term in $\mathrm{LNF}^{\alpha}_{\varnothing}(\Delta)$. Then $M \in \mathrm{AL}^{\alpha}(\Delta)$ if and only if the following conditions hold for all bound variable occurrences $u, v \in [M]$ such that $\mathrm{nsp}_M(u) = \mathrm{nsp}_M(v)$, where $n = \mathrm{arity}(\alpha/\mathrm{nsp}_M(u))$:*

(i) $\{\, w \mid u0^{-n}w \in [M] \,\} = \{\, w \mid v0^{-n}w \in [M] \,\}$.
(ii) *If $M/u0^{-n}w$ is a constant, then $M/u0^{-n}w = M/v0^{-n}w$.*
(iii) *If $M/u0^{-n}w$ is a variable, then $M/v0^{-n}w$ is also a variable and $\mathrm{nsp}_M(u0^{-n}w) = \mathrm{nsp}_M(v0^{-n}w)$.*

We call $M \in \mathrm{AL}^{\alpha}(\Delta)$ a *canonical writing* if for all bound variable occurrences u, v of M, $\mathrm{nsp}_M(u) = \mathrm{nsp}_M(v)$ implies $M/u = M/v$ and vice

versa. For example, $\lambda y_1^{(o\to o)\to o} y_2^{(o\to o)\to o}.f(y_1(\lambda z_1^o.z_1))(y_1(\lambda z_1.z_1))(y_2(\lambda z_2.z_2))$ is a canonical writing, whereas neither $\lambda y_1 y_2.f(y_1(\lambda z_1.z_1))(y_1(\lambda z_2.z_2))(y_2(\lambda z_3.z_3))$ nor $\lambda y_1 y_2.f(y_1(\lambda z_1.z_1))(y_1(\lambda z_1.z_1))(y_2(\lambda z_1.z_1))$ is.

Lemma 2. *For every $M \in \mathrm{AL}^\alpha(\Delta)$, there exists a canonical writing $M' \in \mathrm{AL}^\alpha(\Delta)$ such that $M' \equiv_\alpha M$.*

A *pure* λ-term is a λ-term that contains no constant. We write AL^α for the subset of $\mathrm{AL}^\alpha(\Delta)$ consisting of pure λ-terms. An important property of $\mathrm{AL}^\alpha(\Delta)$ that we heavily rely on in what follows is that every $M \in \mathrm{AL}^\alpha(\Delta)$ can be expressed in a unique way as an application $M^\circ M_1^\bullet \dots M_l^\bullet$ of a *pure* λ-term M° to a list of tree contexts $M_1^\bullet, \dots, M_l^\bullet$. We call the former the *container* of M and the latter its *stored tree contexts*. These λ-terms satisfy the following conditions:

1. $l \le |[\alpha]^+_{\mathrm{sp}}| + 1$,
2. $M_i^\bullet \in \mathrm{AL}^{o^{r_i}\to o}(\Delta)$ for some $r_i \le |[\alpha]^-_{\mathrm{sp}}|$ for each $i = 1, \dots, l$,
3. $M^\circ \in \mathrm{AL}^{(o^{r_1}\to o)\to\dots\to(o^{r_l}\to o)\to\alpha}$,
4. $M^\circ M_1^\bullet \dots M_l^\bullet \twoheadrightarrow_\beta M$.

The formal definition of this separation of $M \in \mathrm{AL}^\alpha(\Delta)$ into its container and stored tree contexts is rather complex, but the intuitive idea is quite simple. The stored tree contexts of M are the maximal tree contexts that can be discerned in the input λ-term.

Example 2. Consider the λ-term M of type $\alpha = o\to(o\to(o\to(o\to o)\to o)\to o)\to o$ in Example 1. This λ-term belongs to $\mathrm{AL}^\alpha(\Delta)$. Its container and stored tree contexts are:

$$M^\circ = \lambda z_1^{o\to o} z_2^{o\to o} z_3^{o\to o\to o} y_1^o y_2^{o\to(o\to(o\to o)\to o)\to o}.y_2(z_1 y_1)(\lambda y_3^o y_4^{o\to o}.z_2(y_4(z_3 y_3 y_1)),$$
$$M_1^\bullet = \lambda x_1.f x_1 a, \quad M_2^\bullet = \lambda x_1.f x_1 x_1, \quad M_3^\bullet = \lambda x_1 x_2.f x_1 x_2.$$

Here is the formal definition. Let $M \in \mathrm{AL}^\alpha(\Delta)$. We assume that M is canonical. Then $|[\alpha]^-_{\mathrm{sp}}|$ is exactly the number of distinct bound variables in M. Let $s_1, \dots, s_k$ list the elements of $[\alpha]^-_{\mathrm{sp}}$ in lexicographic order. Let $y_1, \dots, y_k$ be the corresponding list of bound variables in M, and let $n_i = \mathrm{arity}(\alpha/s_i)$ for each $i = 1, \dots, k$. Note that

$$\sum_{i=1}^k n_i \le |[\alpha]^+_{\mathrm{sp}}|.$$

The canonicity of M implies that every occurrence of y_i in M is accompanied by the exact same list of arguments $N_{i,1}, \dots, N_{i,n_i}$. The type of $N_{i,j}$ is $\alpha/s_i 0^{j-1}1$.

Let $x_1, \dots, x_k$ be fresh variables of type o. For each subterm N of M of type o, define $N^\blacktriangle$ by

$$(cT_1 \dots T_n)^\blacktriangle = cT_1^\blacktriangle \dots T_n^\blacktriangle, \quad (y_i N_{i,1} \dots N_{i,n_i})^\blacktriangle = x_i.$$

Let M' be the maximal subterm of M of atomic type; in other words, M' is the result of stripping M of its leading λs. Likewise, let $N'_{i,j}$ be the maximal subterm of $N_{i,j}$ of atomic type. Let $(M_1, \ldots, M_l)$ be the sublist of

$$(M', N'_{1,1}, \ldots, N'_{1,n_1}, \ldots, N'_{k,1}, \ldots, N'_{k,n_k})$$

consisting of the λ-terms whose head is a constant. (This list will contain duplicates if there exist i_1, j_1, i_2, j_2 such that $(i_1, j_1) \neq (i_2, j_2)$, $N'_{i_1,j_1} = N'_{i_2,j_2}$, and the head of this λ-term is a constant.) For each $i = 1, \ldots, l$, let $x_{m_{i,1}}, \ldots, x_{m_{i,r_i}}$ list the variables in $M_i^{\blacktriangle}$, in the order of their first appearances in $M_i^{\blacktriangle}$. Define

$$M_i^{\bullet} = \lambda x_{m_{i,1}} \ldots x_{m_{i,r_i}}.M_i^{\blacktriangle}, \quad \overrightarrow{M^{\bullet}} = (M_1^{\bullet}, \ldots, M_l^{\bullet}).$$

These are the stored tree contexts of M.

In order to define the container M°, we first define $N^{\triangle}$ by induction for each subterm N of M that is either (i) some M_i, (ii) a λ-term of atomic type whose head is a variable, or (iii) a λ-abstract. Let $z_1, \ldots, z_l$ be fresh variables of type $o^{r_1} \to o, \ldots, o^{r_l} \to o$, respectively.[4]

$$M_i^{\triangle} = z_i(y_{m_{i,1}} N_{m_{i,1},1} \ldots N_{m_{i,1},n_{m_{i,1}}})^{\triangle} \ldots (y_{m_{i,r_i}} N_{m_{i,r_i},1} \ldots N_{m_{i,r_i},n_{m_{i,r_i}}})^{\triangle},$$
$$(y_i N_{i,1} \ldots N_{i,n_i})^{\triangle} = y_i N_{i,1}^{\triangle} \ldots N_{i,n_i}^{\triangle},$$
$$(\lambda y_i.N)^{\triangle} = \lambda y_i.N^{\triangle}.$$

Finally, define

$$M^{\circ} = \lambda z_1 \ldots z_l.M^{\triangle}.$$

Lemma 3. *$M^{\circ}, \overrightarrow{M^{\bullet}}$ satisfy the required conditions.*

Lemma 4. *Let $N \in \mathrm{AL}^{\alpha_1 \to \cdots \to \alpha_n \to \beta}(\Delta), M_i \in \mathrm{AL}^{\alpha_i}(\Delta)$ $(i = 1, \ldots, n)$, and $P = |NM_1 \ldots M_n|_{\beta} \in \mathrm{AL}^{\beta}(\Delta)$. Suppose*

$$\overrightarrow{M_i^{\bullet}} = ((M_i)_1^{\bullet}, \ldots, (M_i)_{l_i}^{\bullet}), \quad (M_i)_j^{\bullet} \in \mathrm{AL}^{o^{r_{i,j}} \to o}(\Delta), \qquad \overrightarrow{P^{\bullet}} = (P_1^{\bullet}, \ldots, P_m^{\bullet}).$$

For $i = 1, \ldots, n$ and $j = 1, \ldots, l_i$, let $c_{i,j}$ be a fresh constant of type $o^{r_{i,j}} \to o$. Let Δ' be the tree signature that extends Δ with the $c_{i,j}$, and let

$$Q = |N((M_1)^{\circ} c_{1,1} \ldots c_{1,l_1}) \ldots ((M_n)^{\circ} c_{n,1} \ldots c_{n,l_n})|_{\beta}.$$

We can compute the container and stored tree contexts of $Q \in \mathrm{AL}^{\beta}(\Delta')$ with respect to Δ'. Then we have

$$P^{\circ} = Q^{\circ}, \quad P_i^{\bullet} = |(Q_i^{\bullet})[c_{i,j} := (M_i)_j^{\bullet}]|_{\beta},$$

where $[c_{i,j} := (M_i)_j^{\bullet}]$ denotes the substitution of $(M_i)_j^{\bullet}$ for each $c_{i,j}$.

[4] When $M_i = M_j$ for some distinct i, j, the definition of $M_i^{\triangle}$ in fact depends on the subscript i.

Definition 1. Let $M \in \mathrm{AL}^{\alpha}(\Delta)$.

(i) The *unlimited profile* of M is $\mathrm{prof}_{\infty}(M) = (M^{\circ}, w_1, \ldots, w_l)$, where l is the length of $\overrightarrow{M^{\bullet}} = (M_1^{\bullet}, \ldots, M_l^{\bullet})$ and for each i, w_i is the r_i-tuple of positive integers whose jth component is the number of occurrences of the jth bound variable in $M_i^{\bullet}$.

(ii) For $k \geq 1$, the *k-threshold profile* of M, written $\mathrm{prof}_k(M)$, is just like its unlimited profile except that any number greater than k is replaced by ∞.

The *type* of the (unlimited or k-threshold) profile of M is α.

Example 3. The unlimited profile of the λ-term M from Example 1 is $\mathrm{prof}(M) = (M^{\circ}, (1), (2), (1,1))$. Its 1-threshold profile is $\mathrm{prof}_1(M) = (M^{\circ}, (1), (\infty), (1,1))$, and its k-threshold profile for $k \geq 2$ is the same as its unlimited profile.

Lemma 5. *For each $k \geq 1$ and type α, there are only finitely many k-threshold profiles of type α.*

We say that a k-threshold profile $(M^{\circ}, w_1, \ldots, w_l)$ is *k-bounded* if $w_i \in \{1, \ldots, k\}^{r_i}$ for $i = 1, \ldots, l$. A λ-term $M \in \mathrm{AL}(\Delta)$ that has a k-bounded profile is called *k-bounded*. We write $\mathrm{AL}_k^{\alpha}(\Delta)$ for the set of all k-bounded λ-terms in $\mathrm{AL}^{\alpha}(\Delta)$.

Note that $M \in \mathrm{AL}(\Delta)$ is linear if and only if it is 1-bounded and has a linear container.

Lemma 6. *Let $N \in \mathrm{AL}^{\alpha_1 \to \cdots \to \alpha_n \to \beta}(\Delta)$, and $M_i, M_i' \in \mathrm{AL}^{\alpha_i}(\Delta)$ for each $i = 1, \ldots, n$. Suppose that for each $i = 1, \ldots, n$, $\mathrm{prof}_k(M_i) = \mathrm{prof}_k(M_i')$. Then $\mathrm{prof}_k(|NM_1 \ldots M_n|_{\beta}) = \mathrm{prof}_k(|NM_1' \ldots M_n'|_{\beta})$.*

The above lemma justifies the notation $N\pi_1 \ldots \pi_n$ for $\mathrm{prof}_k(|NM_1 \ldots M_n|_{\beta})$ with $\mathrm{prof}_k(M_i) = \pi_i$, when k is understood from context. When $N = \lambda x_1 \ldots x_n.Q$, we may also write $Q[x_1 := \pi_1, \ldots, x_n := \pi_n]$ for $N\pi_1 \ldots \pi_n$. In this way, we can freely write profiles in expressions that look like λ-terms, like $\lambda x.\pi_1(Mx\pi_2)$.

Lemma 7. *Given a λ-term $N \in \mathrm{AL}^{\alpha_1 \to \cdots \to \alpha_n \to \beta}(\Delta)$ and k-threshold profiles $\pi_1, \ldots, \pi_n$ of type $\alpha_1, \ldots, \alpha_n$, respectively, the k-threshold profile $N\pi_1 \ldots \pi_n$ can be computed in polynomial time.*

In what follows, we often speak of "profiles" to mean k-threshold profiles, letting the context determine the value of k.

2.3 Almost Linear Second-Order ACGs on Trees

A (tree-generating) *almost linear second-order ACG* $\mathscr{G} = (\Sigma, \Delta, \mathcal{H}, \mathscr{I})$ consists of a second-order signature Σ (*abstract vocabulary*), a tree signature Δ (*object vocabulary*), a set $\mathscr{I} \subseteq A_{\Sigma}$ of *distinguished types*, and a *higher-order homomorphism* $\mathcal{H}$ that maps each atomic type $p \in A_{\Sigma}$ to a type $\mathcal{H}(p)$ over A_{Δ} and each constant $c \in C_{\Sigma}$ to its *object realization* $\mathcal{H}(c) \in$

$\mathrm{AL}^{\mathcal{H}(\tau_\Delta(c))}(\Delta)$. It is required that the image of $\mathscr{I}$ under $\mathcal{H}$ is $\{o\}$. That Σ is second-order means that for every $c \in C_\Sigma$, its type $\tau_\Sigma(c)$ is of the form $p_1 \to \cdots \to p_n \to q$; thus, any λ-term in $\mathrm{LNF}^p_\varnothing(\Sigma)$ for $p \in A_\Sigma$ has the form of a tree. A closed *abstract term* $P \in \mathrm{LNF}^\alpha_\varnothing(\Sigma)$ is homomorphically mapped by $\mathcal{H}$ to its object realization $|\mathcal{H}(P)|_\beta \in \mathrm{AL}^{\mathcal{H}(\alpha)}(\Delta)$. For $p \in A_\Sigma$, we write $\mathcal{S}(\mathscr{G}, p)$ for $\{\, |\mathcal{H}(P)|_\beta \mid P \in \mathrm{LNF}^p_\varnothing(\Sigma) \,\}$ and $\mathcal{C}(\mathscr{G}, p)$ for $\{\, |\mathcal{H}(Q)|_\beta \mid Q$ is a closed linear λ-term in $\mathrm{LNF}^{p\to s}_\varnothing(\Sigma)$ for some $s \in \mathscr{I} \,\}$. The elements of these sets are *substructures* and *contexts* of $\mathscr{G}$, respectively. The tree language generated by $\mathscr{G}$ is $\mathcal{O}(\mathscr{G}) = \bigcup_{s\in\mathscr{I}} \mathcal{S}(\mathscr{G}, s)$.

An abstract constant $c \in C_\Sigma$ together with its type $\tau(c)$ and its object realization $\mathcal{H}(c)$ corresponds to a rule in more traditional grammar formalisms. An abstract atomic type $p \in A_\Sigma$ corresponds to a nonterminal. We say that $\mathscr{G}$ is *rule-k-bounded* if $\mathcal{H}(c)$ is k-bounded for every abstract constant $c \in C_\Sigma$.

Definition 2. Let $\mathscr{G} = (\Sigma, \Delta, \mathcal{H}, \mathscr{I})$ be a tree-generating almost linear second-order ACG.

(i) We say that $\mathscr{G}$ is *substructure-k-bounded* if $\mathcal{S}(\mathscr{G}, p) \subseteq \mathrm{AL}_k^{\mathcal{H}(p)}(\Delta)$ for all atomic types $p \in A_\Sigma$.

(ii) We say that $\mathscr{G}$ is *context-k-bounded* if $\mathcal{C}(\mathscr{G}, p) \subseteq \mathrm{AL}_k^{\mathcal{H}(p)\to o}(\Delta)$ for all atomic types $p \in A_\Sigma$.

The set of possible k-threshold profiles of elements of $\mathcal{S}(\mathscr{G}, p)$ or $\mathcal{C}(\mathscr{G}, p)$ can easily be computed thanks to Lemmas 5 and 6, so substructure-k-boundedness and context-k-boundedness are both decidable properties of almost linear second-order ACGs. Conversely, one can design a substructure-k-bounded almost linear ACG by first assigning to each $p \in A_\Sigma$ a possible profile set Π_p consisting of profiles of type $\mathcal{H}(p)$; then, as the realization $\mathcal{H}(c)$ of a constant c of type $p_1 \to \cdots \to p_n \to q$, we admit only λ-terms in $\mathrm{AL}_k^{\mathcal{H}(p_1\to\cdots\to p_n\to q)}(\Delta)$ that satisfy

$$\mathcal{H}(c)\Pi_{p_1} \dots \Pi_{p_n} \subseteq \Pi_q \,, \tag{2}$$

where $M\Pi_1 \dots \Pi_n = \{\, M\pi_1 \dots \pi_n \mid \pi_i \in \Pi_i \ (i = 1, \dots, n) \,\}$. To construct a context-k-bounded almost linear ACG, we need to assign a possible context profile set Ξ_p in addition to Π_p to each $p \in A_\Sigma$. The realization $\mathcal{H}(c)$ must satisfy

$$\lambda x.\Xi_q(\mathcal{H}(c)\Pi_{p_1} \dots \Pi_{p_{i-1}} x \Pi_{p_{i+1}} \dots \Pi_{p_n}) \subseteq \Xi_{p_i} \tag{3}$$

for all $i = 1, \dots, n$ in addition to (2). Note that (2) and (3) are "local" properties of rules of ACGs. Instead of Definition 2, one may take this local constraint as a definition of substrucure/context-k-bounded almost linear ACGs.

Example 4. Let $\mathscr{G} = (\Sigma, \Delta, \mathcal{H}, \mathscr{I})$, where $A_\Sigma = \{p_1, p_2, s\}$, $C_\Sigma = \{a, b, c_1, c_2, d_1, d_2\}$, $\tau_\Sigma(a) = p_1 \to s$, $\tau_\Sigma(b) = p_2 \to p_1$, $\tau_\Sigma(c_i) = p_i \to p_i$, $\tau_\Sigma(d_i) = p_i$, $A_\Delta = \{o\}$, $C_\Delta = \{e, f\}$, $\tau_\Delta(f) = o \to o \to o$, $\tau_\Delta(e) = o$, $\mathscr{I} = \{s\}$, $\mathcal{H}(p_i) = (o \to o) \to o \to o$,

$\mathcal{H}(s) = o$ and

$$\begin{aligned}
\mathcal{H}(a) &= \lambda x^{(o\to o)\to o\to o}.x(\lambda z^o.z)e,\\
\mathcal{H}(b) &= \lambda x^{(o\to o)\to o\to o} y^{o\to o} z^o.x(\lambda w^o.y(fww))z,\\
\mathcal{H}(c_i) &= \lambda x^{(o\to o)\to o\to o} y^{o\to o} z^o.x(\lambda w^o.yw)(fzz),\\
\mathcal{H}(d_i) &= \lambda y^{o\to o} z^o.y(fzz)\,.
\end{aligned}$$

This grammar is rule-2-bounded and generates the set of perfect binary trees of height ≥ 1. We have, for example, $\mathcal{H}(b(c_2 d_2)) \in \mathcal{S}(\mathscr{G}, p_1)$ and $\mathcal{H}(\lambda x^{p_2}.a(c_1(b(c_2 x)))) \in \mathcal{C}(\mathscr{G}, p_2)$, and

$$\begin{aligned}
|\mathcal{H}(b(c_2 d_2))|_\beta &= \lambda y^{o\to o} z^o.y(f(f(fzz)(fzz))(f(fzz)(fzz))),\\
|\mathcal{H}(\lambda x^p.a(c_1(b(c_2 x))))|_\beta &= \lambda x^{(o\to o)\to o\to o}.x(\lambda z.fzz)(f(fee)(fee)).
\end{aligned}$$

One can see

$$\mathrm{prof}_\infty(\mathcal{S}(\mathscr{G}, p_1)) = \mathrm{prof}_\infty(\mathcal{S}(\mathscr{G}, p_2)) = \{\, (\lambda z_1 yw.y(z_1 w), (2^n)) \mid n \geq 1 \,\}\,,$$

and

$$\begin{aligned}
&\mathrm{prof}_\infty(\mathcal{C}(\mathscr{G}, p_1)) = \{(\lambda z_1 x.x(\lambda w.w)z_1, ())\}\,,\\
&\mathrm{prof}_\infty(\mathcal{C}(\mathscr{G}, p_2)) = \{(\lambda z_1 x.x(\lambda w.w)z_1, ()),\ (\lambda z_1 z_2 x.x(\lambda w.z_1 w)z_2, (2), ())\}\,.
\end{aligned}$$

The grammar is context-2-bounded, but not substructure-k-bounded for any k. If a new constant a' of type $p_1 \to s$ with $\mathcal{H}(a') = \lambda x.x(\lambda z.fzz)e$ is added to $\mathscr{G}$, the grammar is not context-2-bounded any more, since $|\mathcal{H}(\lambda x.a'(bx))|_\beta = \lambda x^{(o\to o)\to o\to o}.x(\lambda z.f(fzz)(fzz))e \in \mathcal{C}(\mathscr{G}, p_2)$.

3 Extraction of Tree Contexts from Trees

We say that $M \in \mathrm{AL}^\alpha(\Delta)$ is *contained* in a tree T if there is an $N \in \mathrm{AL}^{\alpha\to o}(\Delta)$ such that $NM \twoheadrightarrow_\beta T$. The problem of extracting λ-terms in $\mathrm{AL}^\alpha(\Delta)$ contained in a given tree reduces to the problem of extracting tree contexts from trees.

Explicitly enumerating all tree contexts of type $o^r \to o$ is clearly intractable. A perfect binary tree with n leaves (labeled by the same constant) contains more than 2^n tree contexts of type $o \to o$.

It is easy to explicitly enumerate all tree contexts of type $o^r \to o$ that are *k-copying* in the sense that each bound variable occurs at most k times. (Just pick at most $rk + 1$ nodes to determine such a tree context.) Hence it is easy to explicitly enumerate all $M \in \mathrm{AL}^\alpha_k(\Delta)$ whose stored tree contexts (which are all k-copying) are contained in a given tree. (Recall that there is a fixed finite set of candidate containers for each α.) Not all these λ-terms are themselves contained in T, but it is harmless and simpler to list them all than to enumerate exactly those λ-terms $M \in \mathrm{AL}^\alpha_k(\Delta)$ for which there is an $N \in \mathrm{AL}^{\alpha\to o}(\Delta)$ (which may not be k-bounded) such that $MN \twoheadrightarrow_\beta T$.

We consider distributional learners for tree-generating almost linear second-order ACGs who are capable of extracting k-copying tree contexts from trees. Such a learner conjectures rule-k-bounded almost linear ACGs, and use only k-bounded substructures and k-bounded contexts in order to form hypotheses.

4 Distributional Learning of One-Side k-bounded ACGs

We present two distributional learning algorithms, a primal one for the context-k-bounded almost linear ACGs, and a dual one for the substructure-k-bounded almost linear ACGs.

In distributional learning, we often have to fix certain parameters that restrict the class $\mathbb{G}$ of grammars available to the learner as possible hypotheses, in order to make the universal membership problem solvable in polynomial time. This is necessary since the learner needs to check whether the previous conjecture generates all the positive examples received so far, including the current one. In the case of almost linear ACGs, the parameters are the maximal arity n of the type of abstract constants and the finite set Ω of the possible object images of abstract atomic types. When these parameters are fixed, the universal membership problem "$T \in \mathcal{O}(\mathscr{G})$?" is in P [7].

In addition to these two parameters, we also fix a positive integer k so that any hypothesized grammar is rule-k-bounded, for the reason explained in the previous section. The hypothesis space for our learners is thus determined by three parameters, Ω, n, k. We write $\mathbb{G}(\Omega, n, k)$ for the class of grammars determined by these parameters.

In what follows, we often use sets of profiles or λ-terms inside expressions that look like λ-terms, as we did in (2) and (3) in Section 2.3.

4.1 Learning Context-k-bounded ACGs with the Finite Kernel Property

For $\mathbf{T} \subseteq \mathrm{LNF}^o_{\varnothing}(\Delta)$ and $\mathbf{R} \subseteq \mathrm{AL}^{\alpha}(\Delta)$, we define the *$k$-bounded context set of* $\mathbf{R}$ *with respect to* $\mathbf{T}$ by

$$\mathrm{Con}_k(\mathbf{T}|\mathbf{R}) = \{\, Q \in \mathrm{AL}_k^{\alpha \to o}(\Delta) \mid |QR|_\beta \in \mathbf{T} \text{ for all } R \in \mathbf{R} \,\}.$$

Definition 3. A context-k-bounded ACG $\mathscr{G} = (\Sigma, \Delta, \mathcal{H}, \mathscr{I})$ is said to have the *profile-insensitive (k, m)-finite kernel property* if for every abstract atomic type $p \in A_\Sigma$, there is a nonempty set $\mathbf{S}_p \subseteq \mathcal{S}(\mathscr{G}, p) \cap \mathrm{AL}_k^{\mathcal{H}(p)}(\Delta)$ such that $|\mathbf{S}_p| \leq m$ and

$$\mathrm{Con}_k(\mathcal{O}(\mathscr{G})|\mathbf{S}_p) = \mathrm{Con}_k(\mathcal{O}(\mathscr{G})|\mathcal{S}(\mathscr{G}, p)).$$

This may be thought of as a primal analogue of the notion of (k, m)-FCP in [4] for the present case. It turns out, however, designing a distributional learning algorithm targeting grammars satisfying this definition is neither elegant nor quite as straightforward as existing distributional algorithms. One reason is that

simply validating hypothesized rules against k-bounded contexts (see (1) in Section 1) does not produce a context-k-bounded grammar. Recall that to construct a context-k-bounded grammar, we must fix an assignment of an admissible substructure profile set Π_p and an admissible context profile set Ξ_p to each atomic type p which restricts the object realizations of abstract constants of each type. We let our learning algorithm use such an assignment together with finite sets of k-bounded substructures in constructing grammar rules, and make the validation of rules sensitive to the context profile set assigned to the "left-hand side" nonterminal. This naturally leads to the following definition:

Definition 4. A context-k-bounded ACG $\mathscr{G} = (\Sigma, \Delta, \mathcal{H}, \mathscr{I})$ is said to have the *profile-sensitive* (k,m)*-finite kernel property* ((k,m)-$\mathrm{FKP}_{\mathrm{prof}}$) if for every abstract atomic type $p \in A_\Sigma$, there is a nonempty set $\mathbf{S}_p \subseteq \mathcal{S}(\mathscr{G}, p) \cap \mathrm{AL}_k^{\mathcal{H}(p)}(\Delta)$ such that $|\mathbf{S}_p| \leq m$ and

$$\mathrm{Con}_k(\mathcal{O}(\mathscr{G})|\mathbf{S}_p) \cap \mathrm{prof}_k^{-1}(\Xi) = \mathrm{Con}_k(\mathcal{O}(\mathscr{G})|\mathcal{S}(\mathscr{G}, p)) \cap \mathrm{prof}_k^{-1}(\Xi)\,, \qquad (4)$$

where $\Xi = \mathrm{prof}_k(\mathcal{C}(\mathscr{G}, p))$. Such a set $\mathbf{S}_p$ is called a *characterizing substructure set of* p.

Clearly, if a context-k-bounded grammar satisfies Definition 3, then it satisfies the (k,m)-$\mathrm{FKP}_{\mathrm{prof}}$, so the class of grammars with (k,m)-$\mathrm{FKP}_{\mathrm{prof}}$ is broader than the class given by Definition 3. The notion of (k,m)-$\mathrm{FKP}_{\mathrm{prof}}$ is also monotone in k in the sense that (4) implies

$$\mathrm{Con}_{k+1}(\mathcal{O}(\mathscr{G})|\mathbf{S}_p) \cap \mathrm{prof}_{k+1}^{-1}(\Xi') = \mathrm{Con}_{k+1}(\mathcal{O}(\mathscr{G})|\mathcal{S}(\mathscr{G}, p)) \cap \mathrm{prof}_{k+1}^{-1}(\Xi')\,,$$

where $\Xi' = \mathrm{prof}_{k+1}(\mathcal{C}(\mathscr{G}, p)) = \mathrm{prof}_k(\mathcal{C}(\mathscr{G}, p))$, as long as $\mathscr{G}$ is context-k-bounded. This means that as we increase the parameter k, the class of grammars satisfying (k,m)-$\mathrm{FKP}_{\mathrm{prof}}$ monotonically increases. This is another advantage of Definition 4 over Definition 3.

The polynomial enumerability of the k-bounded λ-terms makes an efficient primal distributional learner possible for the class of context-k-bounded grammars in $\mathbb{G}(\Omega, n, k)$ with the (k,m)-$\mathrm{FKP}_{\mathrm{prof}}$.

Algorithm Hereafter we fix a learning target $\mathbf{T}_* \subseteq \mathrm{LNF}_{\varnothing}^o(\Delta)$ which is generated by $\mathscr{G}_* = (\Sigma, \Delta, \mathcal{H}, \mathscr{I}) \in \mathbb{G}(\Omega, n, k)$ with the (k,m)-$\mathrm{FKP}_{\mathrm{prof}}$. We write $\mathbf{S}^{[\Xi]} = \mathrm{Con}_k(\mathbf{T}_*|\mathbf{S}) \cap \mathrm{prof}_k^{-1}(\Xi)$ for a k-bounded profile set Ξ.

For a tree $T \in \mathrm{LNF}_{\varnothing}^o(\Delta)$, let $\mathrm{Ext}_k^\alpha(T) = \{\, M \in \mathrm{AL}_k^\alpha(\Delta) \mid \overrightarrow{M^\bullet}$ are contained in $T \,\}$. Define

$$\begin{aligned}
\mathrm{Sub}_k^\Omega(\mathbf{D}) &= \bigcup\{\, \mathrm{Ext}_k^\alpha(T) \mid T \in \mathbf{D},\ \alpha \in \Omega \,\},\\
\mathrm{Glue}_k^{\Omega,n}(\mathbf{D}) &= \bigcup\{\, \mathrm{Ext}_k^{\alpha_1 \to \cdots \to \alpha_j \to \alpha_0}(T) \mid T \in \mathbf{D},\ \alpha_i \in \Omega \text{ for } i = 1, \ldots, j\\
&\qquad\qquad \text{and } j \leq n \,\},\\
\mathrm{Con}_k^\Omega(\mathbf{D}) &= \bigcup\{\, \mathrm{Ext}_k^{\alpha \to o}(T) \mid T \in \mathbf{D},\ \alpha \in \Omega \,\}.
\end{aligned}$$

Algorithm 1 Learning ACGs in $\mathbb{G}(\Omega, n, k)$ with the (k, m)-FKP$_{\text{prof}}$.

Data: A positive presentation $T_1, T_2, \ldots$ of $\mathbf{T}_*$; membership oracle on $\mathbf{T}_*$;
Result: A sequence of ACGs $\mathscr{G}_1, \mathscr{G}_2, \ldots$;
let $\mathbf{D} := \mathbf{K} := \mathbf{B} := \mathbf{F} := \varnothing$; $\hat{\mathscr{G}} := \mathcal{G}(\mathbf{K}, \mathbf{B}, \mathbf{F})$;
for $i = 1, 2, \ldots$ **do**
  let $\mathbf{D} := \mathbf{D} \cup \{T_i\}$; $\mathbf{F} := \mathrm{Con}_k^{\Omega}(\mathbf{D})$;
  if $\mathbf{D} \not\subseteq \mathcal{O}(\hat{\mathscr{G}})$ **then**
    let $\mathbf{B} := \mathrm{Glue}_k^{\Omega,n}(\mathbf{D})$;
    let $\mathbf{K} := \mathrm{Sub}_k^{\Omega}(\mathbf{D})$;
  end if
  output $\hat{\mathscr{G}} = \mathcal{G}(\mathbf{K}, \mathbf{B}, \mathbf{F})$ as $\mathscr{G}_i$;
end for

It is easy to see that $\mathcal{H}(c) \in \mathrm{Glue}_k^{\Omega,n}(\mathbf{T}_*)$ for all $c \in C_\Sigma$.

Our learner (Algorithm 1) constructs a context-k-bounded ACG $\hat{\mathscr{G}} = \mathcal{G}(\mathbf{K}, \mathbf{B}, \mathbf{F}) = (\Gamma, \Delta, \mathcal{J}, \mathscr{J})$ from three sets $\mathbf{K} \subseteq \mathrm{Sub}_k^{\Omega}(\mathbf{D})$, $\mathbf{B} \subseteq \mathrm{Glue}_k^{\Omega,n}(\mathbf{D})$ and $\mathbf{F} \subseteq \mathrm{Con}_k^{\Omega}(\mathbf{D})$, where $\mathbf{D}$ is a finite set of positive examples given to the learner. As with previous primal learning algorithms, whenever we get a positive example that is not generated by our current conjecture, we expand $\mathbf{K}$ and $\mathbf{B}$, while in order to suppress incorrect rules, we keep expanding $\mathbf{F}$.

Each abstract atomic type of our grammar is a triple of a subset of $\mathbf{K}$, a k-threshold profile set, and a k-bounded profile set:

$$A_\Gamma = \{\, [\![\mathbf{S}, \Pi, \Xi]\!] \mid \mathbf{S} \subseteq \mathbf{K} \cap \mathrm{prof}_k^{-1}(\Pi) \text{ with } 1 \leq |\mathbf{S}| \leq m, \text{ where for some } \alpha \in \Omega,$$
$$\Pi \text{ is a set of } k\text{-threshold profiles of type } \alpha \text{ and}$$
$$\Xi \text{ is a set of } k\text{-bounded profiles of type } \alpha \to o \,\}\,.$$

We have $|A_\Gamma| \leq 2^{2\ell}|\mathbf{K}|^m$, where ℓ is the total number of profiles of relevant types, which is a constant.

The set of distinguished types is defined as

$$\mathscr{J} = \{\, [\![\, \mathbf{S},\, \{(\lambda z^o.z)\},\, \{(\lambda y^o.y)\} \,]\!] \in A_\Gamma \mid \mathbf{S} \subseteq \mathbf{T}_* \,\}\,,$$

which is determined by membership queries. Define $\mathcal{J}([\![\mathbf{S}, \Pi, \Xi]\!])$ to be the type of the profiles in Π.

We have an abstract constant $d \in C_\Gamma$ such that

$$\tau_\Gamma(d) = [\![\mathbf{S}_1, \Pi_1, \Xi_1]\!] \to \cdots \to [\![\mathbf{S}_j, \Pi_j, \Xi_j]\!] \to [\![\mathbf{S}_0, \Pi_0, \Xi_0]\!] \text{ with } j \leq n\,,$$
$$\mathcal{J}(d) = R \in \mathbf{B}\,,$$

if

- $R\Pi_1 \ldots \Pi_j \subseteq \Pi_0$,
- $\lambda x.\Xi_0(R\Pi_1 \ldots \Pi_{i-1} x \Pi_{i+1} \ldots \Pi_j) \subseteq \Xi_i$ for $i = 1, \ldots, j$,
- $|Q(RS_1 \ldots S_j)|_\beta \in \mathbf{T}_*$ for all $Q \in \mathbf{S}_0^{[\Xi_0]} \cap \mathbf{F}$ and $S_i \in \mathbf{S}_i$ for $i = 1, \ldots, j$.

The last condition is checked with the aid of the membership oracle.

Lemma 8. *We have* $\mathrm{prof}_k(N) \in \Pi$ *for all* $N \in \mathcal{S}(\mathscr{G}, [\![\mathbf{S}, \Pi, \Xi]\!])$, *and* $\mathrm{prof}_k(M) \in \Xi$ *for all* $M \in \mathcal{C}(\mathscr{G}, [\![\mathbf{S}, \Pi, \Xi]\!])$. *The grammar* $\mathcal{G}(\mathbf{K}, \mathbf{B}, \mathbf{F})$ *is context-*k*-bounded.*

Lemma 9.
If $\mathbf{K} \subseteq \mathbf{K}'$, *then* $\mathcal{O}(\mathcal{G}(\mathbf{K}, \mathbf{B}, \mathbf{F})) \subseteq \mathcal{O}(\mathcal{G}(\mathbf{K}', \mathbf{B}, \mathbf{F}))$.
If $\mathbf{B} \subseteq \mathbf{B}'$, *then* $\mathcal{O}(\mathcal{G}(\mathbf{K}, \mathbf{B}, \mathbf{F})) \subseteq \mathcal{O}(\mathcal{G}(\mathbf{K}, \mathbf{B}', \mathbf{F}))$.
If $\mathbf{F} \subseteq \mathbf{F}'$, *then* $\mathcal{O}(\mathcal{G}(\mathbf{K}, \mathbf{B}, \mathbf{F})) \supseteq \mathcal{O}(\mathcal{G}(\mathbf{K}, \mathbf{B}, \mathbf{F}'))$.

Lemma 10. *Let* $\mathbf{S}_p$ *be a characterizing set of each atomic type* $p \in A_\Sigma$ *of the target grammar* $\mathscr{G}_*$. *Then* $\mathbf{S}_p \subseteq \mathrm{Sub}_k^\Omega(\mathbf{T}_*)$. *Moreover, if* $\mathbf{S}_p \subseteq \mathbf{K}$ *for all* $p \in A_\Sigma$ *and* $\mathcal{H}(c) \in \mathbf{B}$ *for all* $c \in C_\Sigma$, *then* $\mathbf{T}_* \subseteq \mathcal{O}(\mathcal{G}(\mathbf{K}, \mathbf{B}, \mathbf{F}))$ *for any* $\mathbf{F}$.

We say that an abstract constant d of type $[\![\mathbf{S}_1, \Pi_1, \Xi_1]\!] \to \cdots \to [\![\mathbf{S}_j, \Pi_j, \Xi_j]\!] \to [\![\mathbf{S}_0, \Pi_0, \Xi_0]\!]$ is *invalid* if $|Q(\mathcal{J}(c)S_1 \dots S_j)|_\beta \notin \mathbf{T}_*$ for some $Q \in \mathbf{S}_0^{[\Xi_0]}$ and $S_i \in \mathbf{S}_i$.

Lemma 11. *For every* $\mathbf{K}$ *and* $\mathbf{B}$, *there is a finite set* $\mathbf{F} \subseteq \mathrm{Con}_k^\Omega(\mathbf{T}_*)$ *of cardinality* $|\mathbf{B}||A_\Gamma|^{n+1}$ *such that* $\mathcal{G}(\mathbf{K}, \mathbf{B}, \mathbf{F})$ *has no invalid constant.*

Lemma 12. *If* $\mathcal{G}(\mathbf{K}, \mathbf{B}, \mathbf{F})$ *has no invalid constant, then* $\mathcal{O}(\mathcal{G}(\mathbf{K}, \mathbf{B}, \mathbf{F})) \subseteq \mathbf{T}_*$.

Theorem 1. *Algorithm 1 successfully learns all grammars in* $\mathbb{G}(\Omega, n, k)$ *with the* (k, m)-$\mathrm{FKP}_{\mathrm{prof}}$.

We remark on the efficiency of our algorithm. It is easy to see that the description sizes of $\mathbf{K}$ and $\mathbf{B}$ are polynomially bounded by that of $\mathbf{D}$, and so is that of Γ. We need at most a polynomial number of membership queries to construct a grammar. Thus Algorithm 1 updates its conjecture in polynomial time in $\|\mathbf{D}\|$. Moreover, we do not need too much data. To make $\mathbf{K}$ and $\mathbf{B}$ satisfy the condition of Lemma 10, $m|A_\Sigma|+|C_\Sigma|$ examples are enough. To remove invalid constants, polynomially many contexts are enough by Lemma 11.

4.2 Learning Substructure-k-bounded ACGs with the Finite Context Property

For sets $\mathbf{T} \subseteq \mathrm{LNF}_\varnothing^o(\Delta)$ and $\mathbf{Q} \subseteq \mathrm{AL}_k^{\alpha \to o}(\Delta)$, we define the *$k$-bounded substructure set of* $\mathbf{Q}$ *with respect to* $\mathbf{T}$ by

$$\mathrm{Sub}_k(\mathbf{T}|\mathbf{Q}) = \{\, R \in \mathrm{AL}_k^\alpha(\Delta) \mid |QR|_\beta \in \mathbf{T} \text{ for all } Q \in \mathbf{Q} \,\}.$$

Again, we target grammars that satisfy a property sensitive to profile sets assigned to nonterminals:

Definition 5. A substructure-k-bounded ACG $\mathscr{G} = (\Sigma, \Delta, \mathcal{H}, \mathscr{I})$ is said to have *the profile-sensitive* (k, m)*-finite context property* ((k, m)-$\mathrm{FCP}_{\mathrm{prof}}$) if for every abstract atomic type $p \in A_\Sigma$, there is a nonempty set $\mathbf{Q}_p \subseteq \mathcal{C}(\mathscr{G}, p) \cap \mathrm{AL}_k^{\mathcal{H}(p) \to o}(\Delta)$ of k-bounded λ-terms such that $|\mathbf{Q}_p| \leq m$ and

$$\mathrm{Sub}_k(\mathcal{O}(\mathscr{G})|\mathbf{Q}_p) \cap \mathrm{prof}_k^{-1}(\Pi) = \mathcal{S}(\mathscr{G}, p),$$

where $\Pi = \mathrm{prof}(\mathcal{S}(\mathscr{G}, p))$. We call $\mathbf{Q}_p$ a *characterizing context set* of p.

Algorithm Our dual learner turns out to be considerably simpler than its primal cousin. While the primal learner uses two profile sets, the dual learner assigns just a single profile to each nonterminal. This corresponds to the fact that the context-profiles play no role in constructing a structure-k-bounded grammar and that the (k,m)-$\mathrm{FCP}_{\mathrm{prof}}$ is preserved under the normalization which converts a grammar into an equivalent one $\mathscr{G}'$ where $\mathrm{prof}_k(\mathcal{S}(\mathscr{G}',p))$ is a singleton for all abstract atomic types p of $\mathscr{G}'$, where it is not necessarily the case for the (k,m)-$\mathrm{FKP}_{\mathrm{prof}}$.

Hereafter we fix a learning target $\mathbf{T}_* \subseteq \mathrm{LNF}^o_\varnothing(\Delta)$ which is generated by $\mathscr{G}_* = (\Sigma, \Delta, \mathcal{H}, \mathscr{I}) \in \mathbb{G}(\Omega, n, k)$ with the (k,m)-$\mathrm{FCP}_{\mathrm{prof}}$. We write $\mathbf{Q}^{[\pi]} = \mathrm{Sub}_k(\mathbf{T}_*|\mathbf{Q}) \cap \mathrm{prof}_k^{-1}(\pi)$ for a k-bounded profile π.

Our learner (Algorithm 2) constructs a context-k-bounded ACG $\hat{\mathscr{G}} = \mathcal{G}(\mathbf{F},\mathbf{B},\mathbf{K}) = (\Gamma, \Delta, \mathcal{J}, \mathscr{J})$ from three sets $\mathbf{F} \subseteq \mathrm{Con}^{\Omega}_k(\mathbf{D})$, $\mathbf{B} \subseteq \mathrm{Glue}^{\Omega,n}_k(\mathbf{D})$, and $\mathbf{K} \subseteq \mathrm{Sub}^{\Omega}_k(\mathbf{D})$, where $\mathbf{D}$ is a finite set of positive examples.

Algorithm 2 Learning ACGs in $\mathbb{G}(\Omega,n,k)$ with (k,m)-$\mathrm{FCP}_{\mathrm{prof}}$

Data: A positive presentation $T_1, T_2, \dots$ of $\mathbf{T}_*$; membership oracle on $\mathbf{T}_*$;
Result: A sequence of ACGs $\mathscr{G}_1, \mathscr{G}_2, \dots$;
let $\mathbf{D} := \mathbf{F} := \mathbf{B} := \mathbf{K} := \varnothing$; $\hat{\mathscr{G}} := \mathcal{G}(\mathbf{F},\mathbf{B},\mathbf{K})$;
for $i = 1, 2, \dots$ **do**
  let $\mathbf{D} := \mathbf{D} \cup \{T_i\}$; $\mathbf{K} := \mathrm{Sub}^{\Omega}_k(\mathbf{D})$;
  if $\mathbf{D} \not\subseteq \mathcal{O}(\hat{\mathscr{G}})$ **then**
    let $\mathbf{B} := \mathrm{Glue}^{\Omega,n}_k(\mathbf{D})$;
    let $\mathbf{F} := \mathrm{Con}^{\Omega}_k(\mathbf{D})$;
  end if
  output $\hat{\mathscr{G}} = \mathcal{G}(\mathbf{F},\mathbf{B},\mathbf{K})$ as $\mathscr{G}_i$;
end for

Each abstract atomic type of our grammar is a pair of a finite subset of $\mathbf{F} \cap \mathrm{AL}^{\alpha}_k(\Delta)$ of cardinality at most m and a profile π whose type is α:

$$A_\Gamma = \{\, [\![\mathbf{Q}, \pi]\!] \mid \pi \text{ is a } k\text{-bounded profile of type } \alpha \in \Omega, \\ \mathbf{Q} \subseteq \mathbf{F} \cap \mathrm{AL}^{\alpha\to o}_k(\Delta) \text{ and } 1 \le |\mathbf{Q}| \le m \,\}\,.$$

We have $|A_\Gamma| \le |\mathbf{F}|^m \ell$ for ℓ the number of possible profiles. We have only one distinguished type:

$$\mathscr{J} = \{\, [\![\{\lambda y.y\},\ (\lambda z^o.z)]\!] \,\}\,.$$

We define $\mathcal{J}([\![\mathbf{Q},\pi]\!])$ to be the type of π.

We have an abstract constant $c \in C_\Gamma$ such that

$$\tau_\Gamma(c) = [\![\mathbf{Q}_1,\pi_1]\!] \to \cdots \to [\![\mathbf{Q}_j,\pi_j]\!] \to [\![\mathbf{Q}_0,\pi_0]\!] \text{ with } j \le n\,, \quad \mathcal{J}(c) = P \in \mathbf{B}\,,$$

if

- $\pi_0 = P\pi_1 \dots \pi_j$,

– $|Q(PS_1 \dots S_j)|_\beta \in \mathbf{T}_*$ for all $Q \in \mathbf{Q}_0$ and $S_i \in \mathbf{Q}_i^{[\pi_i]} \cap \mathbf{K}$.

The second clause is checked with the aid of the membership oracle. By the construction, $\mathrm{prof}(|\mathcal{J}(M)|_\beta) \in \pi$ for every $M \in \mathrm{LNF}_\varnothing^{[\![\mathbf{Q},\pi]\!]}(\Gamma)$. Thus the grammar $\hat{\mathscr{G}}$ is substructure-k-bounded.

Lemma 13.
If $\mathbf{F} \subseteq \mathbf{F}'$*, then* $\mathcal{O}(\mathcal{G}(\mathbf{F},\mathbf{B},\mathbf{K})) \subseteq \mathcal{O}(\mathcal{G}(\mathbf{F}',\mathbf{B},\mathbf{K}))$.
If $\mathbf{B} \subseteq \mathbf{B}'$*, then* $\mathcal{O}(\mathcal{G}(\mathbf{F},\mathbf{B},\mathbf{K})) \subseteq \mathcal{O}(\mathcal{G}(\mathbf{F},\mathbf{B}',\mathbf{K}))$.
If $\mathbf{K} \subseteq \mathbf{K}'$*, then* $\mathcal{O}(\mathcal{G}(\mathbf{F},\mathbf{B},\mathbf{K})) \supseteq \mathcal{O}(\mathcal{G}(\mathbf{F},\mathbf{B},\mathbf{K}'))$.

Lemma 14. *Let* $\mathbf{Q}_p$ *be a characterizing set of each atomic type* $p \in A_\Sigma$ *of the target grammar* $\mathscr{G}_*$*. Then* $\mathbf{Q}_p \subseteq \mathrm{Con}_k^\Omega(\mathbf{T}_*)$*. Moreover, if* $\mathbf{Q}_p \subseteq \mathbf{F}$ *for all* $p \in A_\Sigma$ *and* $\mathcal{H}(c) \in \mathbf{B}$ *for all* $c \in C_\Sigma$*, then* $\mathbf{T}_* \subseteq \mathcal{O}(\mathcal{G}(\mathbf{F},\mathbf{B},\mathbf{K}))$ *for any* $\mathbf{K}$*.*

We say that an abstract constant c of type $[\![\mathbf{Q}_1,\pi_1]\!] \to \cdots \to [\![\mathbf{Q}_j,\pi_j]\!] \to [\![\mathbf{Q}_0,\pi_0]\!]$ is *invalid* if $|Q(\mathcal{J}(c)S_1 \dots S_j)|_\beta \notin \mathbf{T}_*$ for some $S_i \in \mathbf{Q}_i^{[\pi_i]}$ and $Q \in \mathbf{Q}_0$.

Lemma 15. *For every* $\mathbf{F}$ *and* $\mathbf{B}$*, there is a finite set* $\mathbf{K} \subseteq \mathrm{Sub}_k^\Omega(\mathbf{T}_*)$ *of cardinality* $n|\mathbf{B}||A_\Gamma|^{n+1}$ *such that* $\mathcal{G}(\mathbf{F},\mathbf{B},\mathbf{K})$ *has no invalid constant.*

Lemma 16. *If* $\mathcal{G}(\mathbf{F},\mathbf{B},\mathbf{K})$ *has no invalid constant, then* $\mathcal{O}(\mathcal{G}(\mathbf{F},\mathbf{B},\mathbf{K})) \subseteq \mathbf{T}_*$.

Theorem 2. *Algorithm 2 successfully learns all grammars in* $\mathbb{G}(\Omega,n,k)$ *with the* (k,m)-$\mathrm{FCP}_{\mathrm{prof}}$.

A remark similar to the one on the efficiency of Algorithm 1 applies to Algorithm 2.

Acknowledgement

This work was supported in part by MEXT/JSPS Kakenhi (24106010, 26330013) and NII joint research project "Algorithmic Learning of Nonlinear Formalisms Based on Distributional Learning".

References

1. Bloem, R., Engelfriet, J.: A comparison of tree transductions defined by monadic second order logic and by attribute grammars. Journal of Computer and System Sciences 61, 1–50 (2000)
2. Böhm, C., Coppo, M., Dezani-Ciancaglini, M.: Termination tests inside λ-calculus. In: Salomaa, A., Steinby, M. (eds.) Automata, Languages and Programming, Lecture Notes in Computer Science, vol. 52, pp. 95–110. Springer Berlin Heidelberg (1977)
3. Clark, A.: Learning context free grammars with the syntactic concept lattice. In: Sempere and García [11], pp. 38–51
4. Clark, A., Yoshinaka, R.: Distributional learning of parallel multiple context-free grammars. Machine Learning 96(1-2), 5–31 (2014), http://dx.doi.org/10.1007/s10994-013-5403-2

5. Kanazawa, M.: Parsing and generation as Datalog queries. In: Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics. pp. 176–183. Prague, Czech Republic (2007)
6. Kanazawa, M.: A lambda calculus characterization of MSO definable tree transductions (abstract). Bulletin of Symbolic Logic 15(2), 250–251 (2009)
7. Kanazawa, M.: Parsing and generation as Datalog query evaluation (2011), http://research.nii.ac.jp/%7Ekanazawa/publications/pagadqe.pdf
8. Kanazawa, M.: Almost affine lambda terms. In: Indrzejczak, A., Kaczmarek, J., Zawidzki, M. (eds.) Trends in Logic XIII. pp. 131–148. Łódź University Press, Łódź (2014)
9. Kasprzik, A., Yoshinaka, R.: Distributional learning of simple context-free tree grammars. In: Kivinen, J., Szepesvári, C., Ukkonen, E., Zeugmann, T. (eds.) Algorithmic Learning Theory. Lecture Notes in Computer Science, vol. 6925, pp. 398–412. Springer (2011)
10. Salvati, S.: Encoding second order string ACG with deterministic tree walking transducers. In: Wintner, S. (ed.) Proceedings of FG 2006: The 11th conference on Formal Grammar. pp. 143–156. FG Online Proceedings, CSLI Publications, Stanford, CA (2007)
11. Sempere, J.M., García, P. (eds.): Grammatical Inference: Theoretical Results and Applications, 10th International Colloquium, ICGI 2010, Valencia, Spain, September 13-16, 2010. Proceedings, Lecture Notes in Computer Science, vol. 6339. Springer (2010)
12. Yoshinaka, R.: Polynomial-time identification of multiple context-free languages from positive data and membership queries. In: Sempere and García [11], pp. 230–244
13. Yoshinaka, R.: Towards dual approaches for learning context-free grammars based on syntactic concept lattices. In: Mauri, G., Leporati, A. (eds.) Developments in Language Theory. Lecture Notes in Computer Science, vol. 6795, pp. 429–440. Springer (2011)
14. Yoshinaka, R., Kanazawa, M.: Distributional learning of abstract categorial grammars. In: Pogodalla, S., Prost, J.P. (eds.) LACL. Lecture Notes in Computer Science, vol. 6736, pp. 251–266. Springer (2011)

Between the Event Calculus and Finite State Temporality

Derek Kelleher, Tim Fernando, and Carl Vogel

Trinity College Dublin
Dublin 2, Ireland
kellehdt@tcd.ie
Tim.Fernando@cs.tcd.ie
vogel@cs.tcd.ie

Abstract. Event Calculus formulas dealing with instantaneous and continuous change are translated into regular languages interpreted relative to finite models. It is shown that a model over the real line for a restricted class of these Event Calculus formulas (relevant for natural language semantics) can be transformed into a finite partition of the real line, satisfying the regular languages. Van Lambalgen and Hamm's treatment of type coercion is reduced to changes in the alphabet from which the strings are formed.

Keywords: Event Calculus, Finite State Temporality, Type Coercion

1 Introduction

An important application of the Event Calculus (EC), originally developed by Kowalski and Sergot[14], has been Lambalgen and Hamm's treatment of Event Semantics[15](hereafter referred to as VLH). The EC concentrates on change in time-dependent properties, known as fluents, formalizing both instantaneous change (due to punctual events), and continuous change (under some force). The underlying model is taken to be a continuum, the real line, with a predicate (HoldsAt) interpreting fluents relative to points (real numbers). This is in contrast to the common practice since Bennett and Partee [3] of evaluating temporal propositions over intervals.

Another approach to event semantics, which uses finite-state methods, is Finite State Temporality (FST). In recent years FST has been shown to be applicable to a diverse range of linguistic phenomena, such as Dowty's Aktionsart[9]. Strings are taken from an alphabet made up of subsets of a finite set of fluents (Σ), shown visually as boxes containing the fluents. Intuitively, fluents in the same box hold at the same time, and fluents in subsequent boxes hold of subsequent times. For example, the event of John going from his home to the library could be represented as the string:

at(john,home)		at(john,library)	(1)

the empty box above symbolising that there is an interval of time between John being at home, and being at the library, with no commitment made to what occurred during this interval. A fuller representation could add to that box fluents representing John's journey to the library, and split it into various boxes representing various stages of this journey.

It should be noted that while both the EC and FST talk of "fluents", there are some crucial differences between the two formalisms in their interpretation of this notion. In the EC, a fluent is a "time-dependent property" in that a fluent holding of an instant indicates that property holds of that instant. For example, if the fluent "building" holds of time t, then there is some building going on at time t. Events have a different status, being the entities that initiate or terminate the fluents (cause them to become true or false), and temporal instants form another ontological category. In FST, "fluents" correspond to EC-fluents, events, temporal instants, and complex entities built from these. These entities are differentiated by their various

properties. For example, FST-fluents corresponding to events are not homogeneous, they hold only of particular intervals, and not any subintervals of that interval, whereas FST-fluents corresponding to EC-fluents are homogeneous.

Arising from the differences between these two formalisms is the question of whether the whole continuum is needed, or whether some finite representation will suffice for natural language semantics. Specifically, is anything lost by translating models for EC predicates that deal with instantaneous and continuous change,with the real line as the domain, into models for FST representations of these predicates, where the domain is a finite partition of the real line. The FST representations of EC predicates will be languages whose various strings represent different temporal granularities consistent with the EC predicates. A major advantage of this approach is that entailment can be expressed in terms of set-theoretic inclusions between languages[7], which is a decidable question for regular languages. Restricting our representations to regular languages ensures the computability of these entailments.

One application of the EC to event semantics is representing type coercion. Since Vendler[19], it is common to assign verb phrases or eventualities to certain categories such as "achievement" (punctual with consequent state), or "activity" (protracted with no associated end point). Under certain circumstances, however, eventualities typically associated with one category can behave as if in another category. For example, "sneeze" is usually considered to be a "point" (punctual with no consequent state). However, when it occurs with the progressive ("John was sneezing"), it can no longer be viewed as a punctual occurrence and is coerced into an activity (becoming a process of multiple sneezes by iteration).

Taking the concepts of instantaneous and continuous change from the EC, the possibility of representing type coercion in FST will be explored. The main idea is to associate coercion with a "change in perspective", implemented through a change in the alphabet from which strings are drawn.

Section 2 gives a brief introduction to Finite State Temporality. Section 3 will show how models for the EC-predicates can be translated into models for the FST-languages, and vice-versa. It also will show that if an EC-model satisfies an EC-predicate, the translated FST-model will satisfy the translated FST-language, and vice-versa. Section 4 will address type coercion, and will show how this can be implemented in FST as changes in Σ.

2 Finite State Temporality

Fix a finite set Φ of fluents. The alphabet from which strings are drawn is made up of subsets of Φ:

$$\Sigma = 2^{\Phi} \tag{2}$$

For example, if Φ = {f,g}, then $\Sigma = \{\{\}, \{\mathrm{f}\}, \{\mathrm{g}\}, \{\mathrm{f}, \mathrm{g}\}\}$. To emphasise that the elements of this alphabet are being used as symbols, and to improve readability, the traditional curly brackets are replaced with boxes enclosing the fluents, with the empty set becoming $\boxed{\ }$.

A typical string over this alphabet might be $\boxed{\mathrm{f,g}}\boxed{\ }\boxed{\mathrm{f}}$ (instead of {f,g}{}{f}), while a typical language might look something like $\boxed{\mathrm{f,g}}^*$, consisting of the strings ϵ, $\boxed{\mathrm{f,g}}$, $\boxed{\mathrm{f,g}}\boxed{\mathrm{f,g}}$, $\boxed{\mathrm{f,g}}\boxed{\mathrm{f,g}}\boxed{\mathrm{f,g}}$, etc.

Fernando[9] shows how a set of temporal points can be partitioned into a finite set of intervals, with a satisfaction relation holding between intervals and fluents. Since the EC takes the positive real line ($\mathbb{R}_+$) as its domain, for the purposes of this paper it is natural to take this as the set of temporal points to be partitioned. A model for FST consists of a segmentation of $\mathbb{R}_+$, and an interpretation function, $[\![]\!]_{FST}$, mapping a fluent to the set of intervals which satisfy it.

A segmentation is a sequence $\mathbb{I} = \mathrm{I}_1 \ldots \mathrm{I}_n$ of intervals such that:

$$\mathrm{I}_i \ \mathrm{m} \ \mathrm{I}_{i+1} \text{ for } 1 \leq i < n \tag{3}$$

Here, "m" is the relation "meets", defined by Allen[1] as: A meets B if and only if A precedes B, there is no point between A and B, and A and B do not overlap. $\mathbb{I}$ is a segmentation of I (in symbols: $\mathbb{I} \nearrow \mathrm{I}$) if and only if $\mathrm{I} = \bigcup_{i=1}^{n} \mathrm{I}_i$.

While there are various ways to partition the real line, one particular form of segmentation is especially useful where the EC is concerned:

$$[0, b_1](a_2, b_2] \ldots (a_{n-1}, b_{n-1}](a_n, \infty), \text{ where } \forall i \; a_{i+1} = b_i \tag{4}$$

So [0,b_1] includes every point between 0 and b_1, including both 0 and b_1 (closed at both ends), whereas (a_i,b_i] includes every point between a_i and b_i, including b_i, but not a_i (open at the beginning, closed at the end).

The reason for choosing this form of segmentation (where I_1 is closed at both ends, but $\mathrm{I}_2, \ldots, \mathrm{I}_n$ are open at the beginning and closed at the end) is as follows: In FST, one use of fluents is to represent states (though they can also represent events or "times", see section 2.1). The interpretation of these stative fluents will be a set of intervals from the segmentation. Intuitively, if an interval I is in the interpretation of a fluent f, then f holds across the interval I. Many temporal reasoning problems specify initial conditions: states or properties which hold at the beginning of time (EC's Initially predicate). These fluents hold from time-point 0 onwards, which requires an interval which includes 0 in their interpretation. This interval will have the form [0,s] for some $\mathrm{s} \in \mathbb{R}_+$.

Other states are "brought into being" by events. The assumption in EC (and common in the literature [2][11]) is that these states hold after the event, not during it. So, if an event happens at time point a_i and brings a state f into being, then an interval in the interpretation of f will not include a_i, but will include the points after it, giving an interval that is open at the beginning.

If an event at b_i causes state f to cease holding, the effect will be seen after b_i, meaning that f still holds at b_i but not at any point after. So b_i will be in an interval that is in the interpretation of f, and this interval will be closed at the end. Any fluent caused to hold by an event, and subsequently caused to stop holding by another event will therefore have an interval that is open at the beginning and closed at the end in its interpretation (with inertia causing it to hold at all points within the interval).

A satisfaction relation $\models_{FST}$ is defined between fluents and intervals:

$$\mathrm{I} \models_{FST} \mathrm{f} \iff \mathrm{I} \in [\![\mathrm{f}]\!]_{FST} \tag{5}$$

This can be extended to a relation holding between segmentations and strings:

$$\mathrm{I}_1 \ldots \mathrm{I}_n \models_{FST} \alpha_1 \ldots \alpha_n \iff (\forall \mathrm{f} \in \alpha_\mathrm{i}) \; \mathrm{I_i} \models_{\mathrm{FST}} \; \mathrm{f}, \text{ for } 1 \leq \mathrm{i} \leq \mathrm{n} \tag{6}$$

and further extended to a relation holding between segmentations and languages:

$$\mathrm{I}_1 \ldots \mathrm{I}_n \models_{FST} \; \mathrm{L} \iff (\exists \mathrm{s} \in \mathrm{L}) \; \mathrm{I}_1 \ldots \mathrm{I}_\mathrm{n} \models_{\mathrm{FST}} \; \mathrm{s} \tag{7}$$

2.1 States, Events, and Times

As noted in the introduction, the EC formalises two notions of change, instantaneous change due to punctual events, and continuous change due to some force. These distinctions may seem more at home in the field of physics than linguistics but, as will be seen later, linguistic "eventualities" can be represented as complex sequences of change, both instantaneous and continuous.

Before continuing, some comments must be made on differences between the EC's and FST's ontologies, and how they are related. In the EC, a distinction is drawn between "fluents", which are time-dependent properties that "hold" of time-points, and events, which are punctual entities that "happen" at

time-points, causing an instantaneous change by "initiating" (causing to hold), or "terminating" (causing to cease to hold) fluents. Temporal points make up a third category, represented by the positive real numbers (and 0).

In FST, the word "fluent" encompasses these three categories, though their fluents have different properties. Stative fluents represent states/properties. They are homogeneous. A fluent is homogeneous if for all intervals I and I′ such that I ∪ I′ is an interval:

$$\text{I} \models_{FST} \text{ f and I}' \models_{FST} \text{ f} \iff \text{I} \cup \text{I}' \models_{FST} \text{ f} \tag{8}$$

Essentially, if a stative fluent holds over an interval, and that interval is split into two intervals, the fluent will hold over both intervals. If John slept from 2pm to 4pm, then John slept from 2pm to 3pm, and John slept from 3pm to 4pm (and vice-versa).

Event fluents represent events. Depending on the event they may be homogeneous or "whole". Fernando[9] defines a fluent as being whole if for all intervals I and I′ such that I ∪ I′ is an interval:

$$\text{I} \models_{FST} \text{ f and I}' \models_{FST} \text{ f implies I} = \text{I}' \tag{9}$$

The EC makes use of an explicit timeline, necessitating the use of "temporal fluents" in FST to represent this timeline. For every possible interval (including points), there is a "temporal fluent" which marks this interval, i.e. every interval has a fluent whose interpretation is that interval. For simplicity of presentation, we use the same symbol for the fluent as we use for the interval or the point that it marks. It should be noted that the interval (a,b] is a complex symbol built from brackets, a comma, and numbers/variables, while the fluent "(a,b]" is an atomic symbol. This allows statements of the form $\{t\} \models t$ to be used.

3 Translating language and models

VLH (p.41) sketch an "intuitively appealing class of models of EC" with domain $\mathbb{R}_+$. The interpretation of fluents is given by the model, and is taken to be a set of intervals of the form (a,b] (a,b $\in \mathbb{R}_+$), or [0,a] (a $\in \mathbb{R}_+$). The intervals are, in a sense, "maximal". They are the longest contiguous stretches for which f holds. If an interval is in the interpretation of a fluent f, f will hold for time-point in that interval, but the time-point itself will not be in the interpretation of f.

It cannot be assumed that all models of the EC give rise to finite segmentations of the real line. VLH give the example of an initiating event for a fluent f happening a a time t iff t $\in \mathbb{Q}$, with terminating events happening at a time t′ iff t′ $\in \mathbb{R} - \mathbb{Q}$. This will lead to a fluent varying infinitely between holding and not holding over any interval, a situation which rules out a finite segmentation.

One necessary condition for there to be a finite segmentation is that each fluent is "alternation bounded" [8]. Intuitively this rules out the above case of a fluent varying infinitely between holding and not holding over some interval. We say a fluent is alternation bounded if the set of points or intervals in its extension (U) is alternation bounded. To define when a set of points U taken from a linear order $(T, <)$ is alternation bounded, the following notions are useful.

Given a subset U of T, a *segmentation of* U is a sequence $\mathbb{I} = I_1 \cdots I_n$ of intervals such that $U = \bigcup_{i=1}^{n} I_i$ and $I_i < I_{i+1}$ for $1 \leq i < n$.

A subset I of T is U*-homogeneous* if

$$(\exists t \in I)\, t \in U \iff (\forall t \in I)\, t \in U$$

— i.e.,

$$I \subseteq U \text{ or } I \cap U = \emptyset.$$

A U-*segmentation* is a segmentation $I_1 \cdots I_n$ of T such that for all i between 1 and n, I_i is U-homogeneous and when $i < n$,

$$I_i \subseteq U \iff I_{i+1} \cap U = \emptyset.$$

Given a subset U of T and a positive integer $n > 0$, an (n, U)-*alternation* is a string $t_1 t_2 \cdots t_n \in T$ of length n such that for $1 \leq i < n$, $t_i < t_{i+1}$ and

$$t_i \in U \iff t_{i+1} \notin U.$$

U is *alternation bounded* (a.b.) if for some positive integer n, there is no (n, U)-alternation. It will also be useful to define the equivalence relation

$$\begin{aligned} t \sim_U t' &\iff \text{there is an } (n, U)\text{-alternation with both } t \text{ and } t' \text{ only if } t = t' \\ &\iff [t, t'] \cup [t', t] \text{ is } U\text{-homogeneous} \end{aligned}$$

Lemma For any subset U of T,

$$\text{there is a } U\text{-segmentation} \iff U \text{ is a.b.}$$

For $\Rightarrow$, a U-segmentation of length n implies there can be no $(n+1, U)$-alternation. Conversely, let n be the largest positive integer for which there is an (n, U)-alternation. Let $t_1 \cdots t_n$ be an (n, U)-alternation, and define $I_1 \cdots I_n$ by

$$I_i := \{t \in T \mid t_i \sim_U t\}$$

Then $I_1 \cdots I_n$ is a U-segmentation.

As well as the traditional axioms of the EC, VLH include formulas in the EC, together known as a scenario, which further constrain the possible models. A formula in the scenario with the following form:

$$S(t) \Longrightarrow Happens(e, t) \tag{10}$$

where S(t) is a first-order formula built from:

1. literals of the form(¬)HoldsAt(f,t)
2. equalities between fluent terms, and between event terms
3. formulas in the language of the structure $(\mathbb{R}, <; +, \times, 0, 1)$

can cause a fluent to vary infinitely, as can be seen in the below example:

$$\begin{aligned} &HoldsAt(f, t) \Longrightarrow Happens(e_1, t + 1) \\ &\neg HoldsAt(f, t) \Longrightarrow Happens(e_2, t + 1) \\ &Initiates(e_2, f, t) \\ &Terminates(e_1, f, t) \\ &HoldsAt(f, 0) \end{aligned} \tag{11}$$

Of course, in this particular example, the fluent can only vary infinitely over an interval stretching to infinity, and while arguments can be made that this is an unrealistic condition for natural language semantics, the formulation of the EC does allow it.

Due to the inertial axioms, when a fluent is initiated, it holds until it is terminated. An initiating event has no effect on a fluent that already holds, and similarly, a terminating event has no effect on a fluent that

does not hold. It follows that an infinite variation of the holding of a fluent over an interval can only arise if there is an infinite sequence of alternating initiating and terminating events.

A further necessary condition for there to exist a finite segmentation of an EC-model is that we are dealing with only a finite number of these alternation bounded fluents. Each of these fluents defines a finite segmentation, and taking a finite number of these fluents together and intersecting the intervals in their segmentations gives a finite segmentation. It will be seen in section 4 on type coercion that eventualities are defined using a finite number of fluents, and type coercion involves adding or removing a finite number of fluents, allowing us to only deal with finite segmentations. Hereafter, when we refer to an "EC-model" it is understood we are discussing models consistent with these conditions:

1. The inferences of interest involve only a finite number of fluents.
2. Each fluent is alternation bounded.

An FST-model can be formed from an EC-model as follows:

1. For each fluent f in EC, form $[\![f]\!]_{EC\to FST} : I \in [\![f]\!]_{EC}$ implies $(\forall I' \subseteq I) I' \in [\![f]\!]_{EC\to FST}$
2. Choose a segmentation $\mathbb{I}$ of $\mathbb{R}_+$
3. Form $[\![f]\!]_{FST}$ by relativizing $[\![f]\!]_{EC\to FST}$ to $\mathbb{I} = I_1 \ldots I_n : I_i \in [\![f]\!]_{FST}$ iff $I_i \in [\![f]\!]_{EC\to FST}$. Only those intervals that are part of the segmentation, $\mathbb{I}$, can be in $[\![f]\!]_{FST}$.

An EC-model can be formed from an FST-model as follows:

1. Suppose $\mathbb{I} \nearrow \mathbb{R}_+$. Find a sequence $I_i \ldots I_j$ in $\mathbb{I}$ such that for all $(i \leq q \leq j)$ $I_q \in [\![f]\!]_{FST}, I_{i-1} \notin [\![f]\!]_{FST}$ and $I_{j+1} \notin [\![f]\!]_{FST}$. This sequence will be the longest unbroken stretch for which f holds.
2. Put $I = \bigcup_{r=i}^{j} I_r$ in $[\![f]\!]_{EC}$.

3.1 Initially

In the EC, Initially(f) signifies that the fluent f "was true at the beginning of time" (VLH p.38). It can be translated as follows:

$$\mathrm{L}_{\text{Initially(f)}} = \langle \trianglerighteq \rangle \boxed{\mathrm{f}\,\square}^{*} \tag{12}$$

Every string in this language includes the fluent f in its first box/symbol. Understanding the above equation requires two definitions:

$$\mathrm{s} \in \langle \mathrm{R} \rangle L \iff (\exists \mathrm{s}' \in L)\ \mathrm{s}\ \mathrm{R}\ \mathrm{s}' \tag{13}$$

where R is some relation between strings. Every string in ⟨R⟩L bears the relation R to some string in L.

$\trianglerighteq$ is the relation "subsumes". If s and s′ are strings, where $\mathrm{s} = \alpha_1 \ldots \alpha_n$ and $\mathrm{s}' = \beta_1 \ldots \beta_k$ then:

$$s \trianglerighteq s' \text{iff n = k, and for every i, } \alpha_i \supseteq \beta_i \tag{14}$$

So every symbol of s contains all the fluents (information) of the corresponding symbol of s′, and possibly more. Fernando[6] shows that this relation can be computed using finite-state methods.

According to VLH (p.42), a model for Initially(f) will have, in its interpretation of f, an interval that begins at 0:

$$Initially := \{f | (\exists s > 0)[0, s] \in [\![f]\!]_{EC}\} \tag{15}$$

In FST, a model for $\mathrm{L}_{\text{Initially(f)}}$ is a segmentation of $\mathbb{R}_+$, where the first interval is in the interpretation of f:

$$I_1 \ldots I_n \nearrow \mathbb{R}_+ \text{ and } I_1 \in [\![f]\!]_{FST} \tag{16}$$

EC$\rightarrow$ FST: If there is an EC-model for Initially(f) then, by (15), there is some s$\in \mathbb{R}$ where [0,s]$\in [\![f]\!]_{EC}$. By construction of an FST-model, [0,s] and all its subintervals are in $[\![f]\!]_{EC \rightarrow FST}$. Any segmentation where I_1 = [0,r], with r$\leq$s, will have $I_1 \in [\![f]\!]_{FST}$, satisfying (16).

FST$\rightarrow$ EC: For an FST-model of $L_{\text{Initially(f)}}$ where $I_1 \ldots I_n \nearrow \mathbb{R}_+$ and $I_1 \in [\![f]\!]_{FST}$ there exists some j with 1<j$\leq$n such that $I_j \notin [\![f]\!]_{FST}$. By construction of an EC-model, $I = \bigcup_{r=1}^{j-1} I_r \in [\![f]\!]_{EC}$. Since I_1 is included in I, I begins at 0 and therefore is of the form [0,s] (s $\in \mathbb{R}$), which fulfills condition (15).

3.2 Instantaneous Change: Happens and Initiates

When dealing with events and their effects on fluents, some new fluents will prove helpful:

$$I \models_{FST})_e \iff (\forall J \text{ such that } I \text{ m } J) I \not\models_{FST} f \text{ and } J \models_{FST} f \tag{17}$$

$$(a, b] \models_{FST} \langle end \rangle t \iff \{b\} \models_{FST} t \tag{18}$$

$$(a, b] \models_{FST} \langle start \rangle t \iff \{a\} \models_{FST} t \tag{19}$$

The fluent $)_e$ is used to mark the end of events. In the EC, events are punctual: they occur at points (interpreted as real numbers). In FST, events are allowed to occur over intervals. The reason for this will become clear when type coercion is dealt with. If $\langle end \rangle t$ holds of an interval, then that interval ends at the time point t. If $\langle start \rangle t$ holds of an interval, then that interval starts after the time point t.

The EC treats change due to an event as instantaneous. A fluent initiated by a punctual event holds AFTER, but not AT, the time at which the event occurs. When moving from punctual events to events that happen over intervals, a choice must be made as to when the change occurs. In keeping with the EC, FST formalizes the change as happening after the event ends. Only initiating events are dealt with below, the definitions for terminating events are broadly similar. Note that the fluents f and $)_e$ are linked in that this particular event initiates the fluent f, other events initiate other fluents.

Happens(e,t), which signifies that event e occurs at time t, can be translated as follows:

$$L_{\text{Happens(e,t)}} = \langle \sqsupseteq \rangle \boxed{(_e \boxed{}^* \boxed{\langle end \rangle t,)_e}} \tag{20}$$

If s = $\alpha_1 \ldots \alpha_n$, s $\sqsupseteq$ s$'$ iff there is some substring r of s, $\alpha_i \ldots \alpha_j$, and r $\trianglerighteq$ s$'$. ($_e$ represents the start of the event. It cannot be translated from the EC as the EC treats events as punctual. Its purpose in FST is to allow punctual events to be "stretched" and given an internal structure. It causes no problem in translations as $)_e$ is the necessary fluent to represent instantaneous change.

VLH (p.42) define the extension of Happens as follows:

$$Happens := \{(e, t) | (\exists f)(f, t) \in e\} \tag{21}$$

In an EC-model, the event e will either initiate or terminate some fluent f. If it is an initiating event then there must be some s$\in \mathbb{R}$ for which (t,s]$\in [\![f]\!]_{EC}$.

In FST, a model for $L_{\text{Happens(e,t)}}$ will be a segmentation $I_1 \ldots I_n \nearrow \mathbb{R}_+$ with the following conditions:

$$(\exists I_i, I_j) I_i < I_j \text{ and } I_i \models_{FST} (_e \text{ and } I_j \models_{FST})_e \text{ and } I_j \models_{FST} \langle end \rangle t \tag{22}$$

Note that $<$ is the relation "precedes". For intervals I and J, I precedes J if its end point is before J's start point.

Using the convention $\{t\} \models_{FST} t$, and the definition of $\langle end\rangle t$, then $I_j = (a_j$,t] for some $a_j \in \mathbb{R}$. From the definition of $)_e$, $(a_j$,t]$\not\models_{FST}$ f and (t,b_{j+1}]$\models_{FST}$ f for some $b_{j+1} \in \mathbb{R}$.

EC$\rightarrow$ FST: From above ($\exists$ s) (t,s]$\in [\![f]\!]_{EC}$. Because the intervals in the interpretation of fluents are maximal, there must be an interval (at least a point) that meets (t,s] which is not in the interpretation of f. Therefore, (r,t] $\notin [\![f]\!]_{EC}$ for some r $\in \mathbb{R}$. Constructing an FST-model from this, (t,s] and all its subintervals are in $[\![f]\!]_{EC\rightarrow FST}$, and there is some q $\in \mathbb{R}$ for which (q,t] and all its subintervals are not in $[\![f]\!]_{EC\rightarrow FST}$. An example segmentation would be [0,b_1]. . .(q,t](t,s]. . .(a_n,∞). $[\![f]\!]_{EC\rightarrow FST}$ relativized to this will have (q,t] $\notin [\![f]\!]_{FST}$, $(t, s] \in [\![f]\!]_{FST}$ satisfying the conditions above with a_j = q, and b_{j+1} = s. (Again there will be many segmentations of $\mathbb{R}_+$ that satisfy this language).

FST$\rightarrow$ EC: For an FST-model of $L_{\text{Happens(e,t)}}$ where $I_1 \ldots I_n \nearrow \mathbb{R}_+$ and $I_j \notin [\![f]\!]_{FST}$ and $I_{j+1} \in [\![f]\!]_{FST}$, there exists some r with j+1<r$\leq$n such that $I_r \notin [\![f]\!]_{FST}$. By construction of an EC-model $I = \bigcup_{d=j+1}^{r-1} I_d \in [\![f]\!]_{EC}$. Since I_{j+1} is the first interval in I, I is of the form (t,s], satisfying the condition above for an EC-model.

Initiates(e,f,t) can be translated as follows:

$$L_{\text{Initiates(e,f,t)}} = \boxed{\langle end\rangle t,)_e \boxed{\ }} \Longrightarrow \boxed{\ }\boxed{\text{f}} \tag{23}$$

This is encoded as a constraint on strings. The constraint L $\Longrightarrow$ L′ is the set of strings s, such that every stretch of s that subsumes L also subsumes L′. The notion of "stretched" is formalized as follows: s′ is a factor of s if s = us′v for some (possibly empty) strings u and v.

$$s \in L \Longrightarrow L' \iff \text{ for every factor } s' \text{ of } s, s' \trianglerighteq L \text{ implies } s' \trianglerighteq L' \tag{24}$$

The constraint for $L_{\text{Initiates(e,f,t)}}$ says that every string in this language that has a box containing $\langle end\rangle t$ and $)_e$, must contain f in the following box.

An EC-model for Initiates(e,f,t) must have the following condition: $(\exists s \in \mathbb{R})(t, s] \in [\![f]\!]_{EC}$.

In FST, a model for $L_{\text{Initiates(e,f,t)}}$ will be a segmentation $I_1 \ldots I_n \nearrow \mathbb{R}_+$ with the following conditions:

$$(\forall I_i, I_j)[I_i \text{ m } I_j \text{ and } I_i \models_{FST})_e \text{ and } I_i \models_{FST} \langle end\rangle t] \longrightarrow I_j \models_{FST} f \tag{25}$$

EC$\rightarrow$ FST: From above ($\exists$ s) (t,s]$\in [\![f]\!]_{EC}$. For the same reason as above for the Happens predicate, (t,s] and all its subintervals are in $[\![f]\!]_{EC\rightarrow FST}$, and there is some q $\in \mathbb{R}$ for which (q,t] and all its subintervals are not in $[\![f]\!]_{EC\rightarrow FST}$. Taking as an example segmentation, [0,b_1]. . .(q,t](t,s]. . .(a_n,∞), the interval (q,t] meets (t,s], it satisfies $\langle end\rangle t$, it also satisfies $)_e$ (by definition of $)_e$). Therefore it meets the conditions of the antecedent of (25), and since it is given that (t,s] $\in [\![f]\!]_{EC}$, and therefore in $[\![f]\!]_{FST}$, the consequent is also true, fulfilling the conditions for an FST-model.

FST$\rightarrow$ EC: Take an FST-model of $L_{\text{Initiates(e,f,t)}}$ where $I_1 \ldots I_n \nearrow \mathbb{R}_+$ and (25) holds. Given that there is only one interval I_i in this segmentation which satisfies $\langle end\rangle t$, and only one interval I_j that it meets, we can reduce (25) to:

$$(a_i, t] \models_{FST})_e \longrightarrow (t, b_j] \models_{FST} f \tag{26}$$

By definition of $)_e$, (a_i,t]$\not\models_{FST}$ f. (26) then becomes:

$$(a_i, t] \not\models_{FST} f \longrightarrow (t, b_j] \models_{FST} f \tag{27}$$

there exists some r with j+1<r≤n such that $I_r \notin [\![f]\!]_{FST}$. By construction of an EC-model, $I = \bigcup_{d=j}^{r-1} I_d \in [\![f]\!]_{EC}$. Since I_{j+1} is the first interval in I, I is of the form (t,s], satisfying the condition above for an EC-model.

3.3 Continuous change: Trajectory

Trajectory($f_1, t, f_2, t+d$) signifies that if fluent f_1 holds between t and t+d, then fluent f_2 will hold at t+d. In VLH, f_2 is generally a real-valued function describing continuous change, so the Trajectory predicate describes continuous change as long as fluent f_1 holds. It can be translated as follows:

$$L_{\text{Trajectory}(f_1,t,f_2,t+d)} = \boxed{\langle start\rangle t, f_1}\boxed{f_1}^*\boxed{\langle end\rangle t+d, f_1} \Longrightarrow \boxed{\ }^*\boxed{\langle end\rangle f_2} \tag{28}$$

While VLH does not give a model for Trajectory, it is not hard to construct an intuitive model consistent with the other predicates:

$$Trajectory := \{(f_1, t, f_2, t{+}d) | (\exists a, b \in \mathbb{R}_+) a \leq t \leq t{+}d \leq b \text{ and } (a, b] \in [\![f_1]\!]_{EC} \longrightarrow (\exists I \in [\![f_2]\!]_{EC}) t{+}d \in I\} \tag{29}$$

A FST-model for $L_{\text{Trajectory}(f_1,t,f_2,t+d)}$ will be a segmentation of $\mathbb{R}_+$ with the following conditions:

$$(\exists I_i = (t, b_i], I_j = (a_j, t+d])(\forall i \leq p \leq j) I_p \models_{FST} f_1 \longrightarrow \{t+d\} \models_{FST} f_2 \tag{30}$$

EC→ FST: If the antecedent of (30) is true, then the sequence $I_1 \ldots I_j$ is part of an unbroken stretch over which f holds. Let the start and end points of this stretch be a and b from (29). Therefore, (a,b] $\in [\![f]\!]_{EC}$, fulfilling the antecedent of the conditional contained in (29). Now from the consequent of (29), there is an interval I $\in [\![f_2]\!]_{EC}$, so all its subintervals, notably {t+d}, are in $[\![f_2]\!]_{EC \to FST}$. By definition of $\langle end\rangle t + d$, (a_j,t+d] $\in [\![\langle end\rangle f_2]\!]_{EC \to FST}$. Relativized to the segmentation given, (a_j,t+d] $\in [\![\langle end\rangle f_2]\!]_{FST}$. Therefore {t+d}$\models_{FST}$ f_2.

FST→ EC: For an FST-model of $L_{\text{Trajectory}(f_1,t,f_2,t+d)}$ where $I_1 \ldots I_n \nearrow \mathbb{R}_+$ and, supposing the antecedent contained in (29) is true, (t,b_i],(a_j,t+d] and all the intervals between them are in $[\![f_1]\!]_{FST}$ (by construction of an FST-model. Therefore, the consequent of (30) is true, so {t+d}$\models$ f_2, so {t+d} $\in [\![f_2]\!]_{FST}$. {t+d} is either a maximal interval for which f_2 holds or is a subinterval of that maximal interval. Either way, there is some I $\in [\![f_2]\!]_{EC}$ with t+d ∈ I.

While VLH (p.45) require a real-valued function of time in place of f_2, and while this can be implemented in FST as long as only a finite number of values from this function are used as fluents, it is not clear that natural language semantics needs the precision of this real-valued function to address continuous change. As will be seen in the next section, when dealing with type coercions that involve adding elements to an event structure, it is not always clear in advance what form these elements will have. An alternative in FST to the above representation of continuous change is as follows:

$$\boxed{f_1} \Longrightarrow \boxed{f_{2\uparrow}} \tag{31}$$

The above essentially says that if f_1 is in a box, then $f_{2\uparrow}$ is in the same box, or as long as f_1 holds, f_2 is increasing, or moving along some trajectory. For type coercion, it will be useful to have a fluent that holds when the end of the trajectory is reached, $f_{2,MAX}$.

These fluents can be interpreted model-theoretically, relative to an underlying function, representing trajectory or path taken, in the model. This is in line with the approach of Kennedy[13], who propose

that the semantics of degree achievements (such as "the soup cooled") rely on a "difference function", a function that measures the amount an object changes along a scalar dimension as a result of participating in an event. Certain functions describing change will have an end-point or maximum. This may be contextualy given, as Fernando[10] assumes, or may be a natural feature of the scale against which change is measured. Closed scales (such as "smooth") have a natural maximum, complete smoothness in this case[18].

3.4 Scenarios

The EC relates instantaneous and continuous change to natural language semantics through scenarios, which "state the specific causal relationships holding in a given situation" (VLH, p.43). Only the elements of the a scenario dealing with change will be dealt with here.

A scenario is a conjunct of statements of the form:

1. Initially(f)
2. S(t) $\rightarrow$ Initiates(e,f,t)
3. S(t) $\rightarrow$ Terminates(e,f,t)
4. S(t) $\rightarrow$ Happens(e,t)
5. S(f_1,f_2,t,d) $\rightarrow$ Trajectory(f_1,t,f_2,d)

where S(t) is a conjunction of statements of the form HoldsAt(f,t). Element 5 is referred to as the "dynamics", relating fluent f_1 (viewed as a force), to the change in f_2 (viewed as a changing, partial object).

As with EC-predicates, each element of an EC-scenario can be translated into FST. If S(t) is not a part of an element (for instance if one element of the scenario is Initiates(e,f,t), as opposed to S(t) $\rightarrow$ Initiates(e,f,t)), then the equivalent element of the FST-scenario is the language given above as equivalent. If S(t) is part of an element then the equivalent will be a constraint relating fluents holding at a certain time to the language given above as equivalent. For example, HoldsAt(f,t) $\rightarrow$ Happens(e,t) can be translated as follows:

$$\boxed{\ }^* \boxed{\langle end\rangle \text{ t},\langle end\rangle \text{ f}} \Longrightarrow \boxed{\ }^* \boxed{(_e} \boxed{\ }^* \boxed{)_e} \tag{32}$$

For the temporal fluents occurring in a string to have the correct ordering, one further set of constraints is needed. In an FST-model, every interval is ordered relative to every other interval by the "precedes" relation ($<$). A scenario will contain the set of constraints:

$$\forall I, I' \text{ in an FST-model such that } I < I' : \boxed{\text{I'}} \boxed{\ }^* \boxed{\text{I}} \Longrightarrow \emptyset \tag{33}$$

The above constraint says that if, in an FST-model, an interval I precedes an interval I$'$, then any string with the fluent I$'$ preceding I will not be in the language.

4 Type Coercion

Having developed the tools necessary to represent instantaneous and continuous change in FST, these are now applied to the natural language semantics problem of type coercion.

4.1 Eventualities vs. Events

The EC treats "events" as punctual occurrences that cause an instantaneous change (a fluent becomes initiated or terminated). However, most occurrences described as events have a more complex structure. Moens and Steedman[16] (hereafter referred to as M&S) have proposed a three-part event structure, called a "nucleus", which consists of a preparatory process, a culmination, and a consequent state. They note that all of these elements can be compound. The culmination "reaching the top of Mt. Everest" may have a number of processes such as climbing, eating lunch, etc. as part of its preparatory process, and there may be many consequent states.

Due to this, it is important to note that the discussion of coercion must have a quite general character. If adding a consequent state to an event structure, the question arises: what or which consequent state? For this reason, non-specific fluents such as f_3, g_1 etc. will often be used to describe "sailent' states or activities that have been added to an event structure.

The EC implements a similar event structure to that of M&S which they call "eventualities", having the following form: (f_1,f_2,e,f_3) where f_1 is an activity, f_2 is a fluent representing a partial object which changes under the influence of the activity, e is a culminating event, and f_3 is a consequent state.

Different approaches have used different terminologies for what are, essentially, the same categories. What Vendler[19] called activities, achievements, and accomplishments, M&S call processes, culminations, and culminated processes respectively.

4.2 Activity $\leadsto$ Accomplishments

VLH (p.171) discuss the case of "additive coercion", where a scenario is elaborated or added to. Alternative views of this type of coercion are given by Pulman[17] and M&S. Pulman sees this as supplementing a process with a consequent state (Pulman does not include culminating events in his ontology). The activity of "swimming" would have a salient consequent state added, perhaps as general as "has swum". M&S propose that the process is bundled into a point, with a new preparatory process and consequent state added. They cite the example "has John worked in the garden?" as only making sense if John working in the garden was part of some plan, with the preparatory process being whatever preparation was involved to work in the garden, the culmination being the working in the garden viewed as a point, and the consequent state perhaps being "has worked in the garden".

Both these views can be accounted for in FST. In the first, the activity f_1 is augmented with a fluent representing the change in the partial object (swimming augmented with the trajectory of the swim), a culminating event (finishing the swim), and a consequent state (has swum). In alphabet terms:

$$\Sigma \leadsto \Sigma' = \Sigma \cup \{f_{2\uparrow}, f_{2,MAX}, (_e,)_e, f_3\} \tag{34}$$

The scenario will also have to be augmented with general constraints of the form:

$$\boxed{\mathrm{f}_1} \Longrightarrow \boxed{f_{2\uparrow}} \tag{35}$$

$$\boxed{\mathrm{f}_{2,MAX}} \Longrightarrow \boxed{)_e} \tag{36}$$

$$\boxed{)_e}\boxed{\ } \Longrightarrow \boxed{\ }\boxed{f_3} \tag{37}$$

For M&S's coercion to be implemented, f_1, the original activity and internal structure of some event, is deleted, and fluents representing its start and end points are added, turning it into a point with no internal structure. A new preparatory process, g_1 is added, and a new consequent state, g_3. In alphabet terms:

$$\Sigma \leadsto \Sigma' = \Sigma \cup \{g_1, g_{2\uparrow}, g_{2,MAX}, (_e,)_e, g_3\} - \{f_1\} \tag{38}$$

The scenario will be elaborated as follows:

$$\boxed{\mathrm{g}_1} \Longrightarrow \boxed{g_{2\uparrow}} \tag{39}$$

$$\boxed{\mathrm{g}_{2,MAX}} \Longrightarrow \boxed{)_e} \tag{40}$$

$$\boxed{)_e}\boxed{\ } \Longrightarrow \boxed{\ }\boxed{g_3} \tag{41}$$

4.3 Achievements $\rightsquigarrow$ Accomplishments

As pointed out in M&S, the progressive needs a process (activity), or culminated process (accomplishment) as "input", so what to make of the following:

John was reaching the top (42)

"Reach the top" is usually seen as a culmination (achievement), punctual with an associated consequent state. The progressive coerces this culmination into a culminated process by adding a preparatory process, and focussing on this.

For VLH (p.172), the achievement (-,-,e_1,f_3), where e_1 would be the culmination (reaching the top), and f_3 would be the consequent state of this (perhaps being at the top), would be coerced into an accomplishment (g_1,g_2,e_2,g_3), where g_1 and g_2 are "unknown parameters", depending on what preparatory process is being associated with the culmination. Presumably the culminations and consequent states of "reach the top" viewed as an achievement and as an accomplishment are the same. The addition of the fluents representing the preparatory process and changing partial object means a dynamics must be added to the scenario to relate these fluents.

To account for the addition of a preparatory process, and changing object in FST, the fluents marking the beginning and ending of events $(_e$ and $)_e$ are associated with a fluent prog_e which signifies that the event in question is ongoing:

$$I \models prog_e \Longleftrightarrow I \models (_e \text{ or } I \models)_e \text{ or } [(\exists I_i, I_j) I_i < I < I_j \text{ and} I_i \models (_e \text{ and } I_j \models)_e]$$

This fluent corresponds to g_1 from the EC. Its addition to the alphabet of FST will lead to a finer-grained segmentation, effectively giving the event, previously viewed as punctual, an internal structure. In alphabetic terms:

$$\Sigma \rightsquigarrow \Sigma' = \Sigma \cup \{prog_e, g_{2\uparrow}, g_{2,MAX}, (_d,)_d, g_3\} - \{f_3\} \tag{43}$$

4.4 Accomplishments $\rightsquigarrow$ Activities

M&S discuss the case of a culminated process being coerced into a process in the presence of the progressive:

Roger was running a mile (44)

For this to happen, the culmination and consequent state must be "stripped off", leaving the preparatory process.

In VLH (p.172), this process is described as subtractive coercion, essentially removing the last two elements from the event-nucleus, and removing statements relating to the culmination and consequent

state from the scenario. The same can be achieved in FST by removing those constraints from the scenario that "cause" the culminating event to happen at the maximum point in the trajectory of the run, and relate the consequent state to the occurrence of the culminating event. In alphabetic terms:

$$\Sigma \rightsquigarrow \Sigma' = \Sigma - \{(_e,)_e, f_3\} \tag{45}$$

Another way for this coercion to occur is if the direct object is a mass noun. This would lead to a lack of end point in the trajectory, perhaps even a lack of trajectory at all.

4.5 Points $\rightsquigarrow$ Activities

As noted in M&S, the progressive requires a process as "input", yet interpretations can be given for:

Harry was sneezing (46)

M&S see this as being coerced by iteration, referring to a number of sneezes, rather than one sneeze.

Both Comrie[4] and Pulman[17] provide another interpretation for this coercion, where what was viewed as a point is stretched into a process, having internal structure.

VLH (p.175) deal with the first type, viewing it as a change from (-,-,e,-) to (f_1,-,-,-) or (f_1,f_2,-,-). While the EC needs to coerce an event into an activity fluent (a method for this is provided in [12]), FST does not face this ontological problem.

Events in FST can be represented by a multitude of fluents, provideing different "perspectives" on events. Up to now $(_e)_e$ have been used to mark the start and end points of events, where we were not concerned with their internal or overall structure. Not representing the internal structure leads to the events being viewed as points. A fluent representing internal structure can be introduced, and as a fluent with a model-theoretic definition, there are no ontological obstacles to doing this.

To represent a point-like event being "stretched' the fluent prog_e, defined above, is added. This represents the event in progress as a homogeneous, stative fluent.

To represent iteration, a fluent iter_e is defined, which holds of any interval if two or more events happen within that interval. It can be thought of as a homogeneous process. While a particular subinterval may not contain a sneeze, it is considered part of the process of iteratively sneezing.

$$I \models_{FST} iter_e \iff (\exists I_i < I_j < I_k < I_l)\ I_i \models_{FST} (_e \text{ and } I_j \models_{FST})_e \text{ and } I_k \models_{FST} (_e \text{ and } I_l \models_{FST})_e \tag{47}$$

This coercion can be achieved by adding either prog_e or iter_e to the alphabet:

$$\Sigma \rightsquigarrow \Sigma' = \Sigma \cup \{prog_e\} \tag{48}$$

$$\Sigma \rightsquigarrow \Sigma' = \Sigma \cup \{iter_e\} \tag{49}$$

4.6 States $\rightsquigarrow$ Activities

Croft[5] gives the following example of a state being coerced into a process:

She is resembling her mother more and more every day (50)

Though the acceptability of this type of coercion varies, VLH (p.173) give an account where a fluent representing the state "resembles her mother" is coerced into an activity fluent. They argue against coercing the state into the changing partial object, as might be expected given that increasing resemblance over time could be viewed as a trajectory.

Since a state is homogeneous, there are no problems treating it as an activity in FST. Activities and states are both represented by homogeneous fluents.

The activity fluent, which previously represented a state, must be related to a changing, partial object (represented by f_2) by some constraint set, leading to the crucial difference between "resembles" and "is resembling more and more".

5 Conclusion

The Event Calculus differs from other temporal representations by directly formalizing change, both instantaneous change as a result of events, and continuous change as a result of some constant force. Section 3 described how EC-predicates can be implemented as regular languages in FST. Furthermore, it was shown how a model, with a finite segmentation of the real number line as domain, for these FST-languages, could be formed from a model (with certain conditions excluding those models for which no finite segmentation exists), with the real number line as domain, for the equivalent EC-predicates. This shows that if a natural language semantics problem can be described in terms of change (or at least the kind of change that the EC formalizes), then the full real number line is not needed to model this problem.

Section 4 discusses type coercion, which the EC has been applied to. It is shown how various types of coercion, implemented in the EC as changes in scenario, or transformations of type (from fluents to events), can be implemented in FST. Type coercions in FST can be viewed as changes in "focus" or "perspective", formalized as changes in the alphabet from which strings are drawn.

Bibliography

[1] Allen, J.F.: An interval-based representation of temporal knowledge. In: IJCAI. vol. 1, pp. 221–226 (1981)

[2] Allen, J.F., Ferguson, G.: Actions and events in interval temporal logic. In: Spatial and temporal Reasoning, pp. 205–245. Springer (1997)

[3] Bennett, M., Partee, B.H.: Toward the logic of tense and aspect in English. Wiley Online Library (1978)

[4] Comrie, B.: Aspect: An introduction to the study of verbal aspect and related problems. Cambridge University Press (1976)

[5] Croft, W.: The structure of events. In: Tomasello, M. (ed.) The new psychology of language, chap. 3. Lawrence Erlbaum Associates (1998)

[6] Fernando, T.: Finite-state temporal projection. In: Implementation and Application of Automata, pp. 230–241. Springer (2006)

[7] Fernando, T.: Temporal propositions as regular languages. In: Finite-State Methods and Natural Language Processing, 6th International Workshop. pp. 132–48 (2008)

[8] Fernando, T.: Partitions representing chnage homogeneously. In: Aloni, M., Franke, M., Roelofsen, F. (eds.) A festschrift for Jeroen Groenendijk, Martin Stokhof, and Frank Veltman, pp. 91–95 (2013)

[9] Fernando, T.: Segmenting temporal intervals for tense and aspect. In: The 13th Meeting on the Mathematics of Language. p. 30 (2013)

[10] Fernando, T.: Incremental semantic scales by strings. EACL 2014 p. 63 (2014)

[11] Halpern, J.Y., Shoham, Y.: A propositional modal logic of time intervals. Journal of the ACM (JACM) 38(4), 935–962 (1991)

[12] Hamm, F., van Lambalgen, M.: Nominalization, the progressive and event calculus. Linguistics and Philosophy 26, 381–458 (2003)

[13] Kennedy, C., Levin, B.: Measure of change: The adjectival core of degree achievements. Adjectives and adverbs: Syntax, semantics and discourse pp. 156–182 (2008)

[14] Kowalski, R., Sergot, M.: A logic-based calculus of events. In: Foundations of knowledge base management, pp. 23–55. Springer (1989)

[15] van Lambalgen, M., Hamm, F.: The proper treatment of events. John Wiley & Sons (2005)

[16] Moens, M., Steedman, M.: Temporal ontology and temporal reference. Computational linguistics 14(2), 15–28 (1988)

[17] Pulman, S.G.: Aspectual shift as type coercion. Transactions of the Philological Society 95(2), 279–317 (1997)

[18] Solt, S.: Scales in natural language (2013)

[19] Vendler, Z.: Verbs and times. The philosophical review 66, 143–160 (1957)

A Modal Representation of Graded Medical Statements

Hans-Ulrich Krieger[1] and Stefan Schulz[2]

[1] German Research Center for Artificial Intelligence (DFKI) krieger@dfki.de
[2] Institute of Medical Informatics, Medical University of Graz
stefan.schulz@medunigraz.at

Abstract. Medical natural language statements uttered by physicians are usually *graded*, i.e., are associated with a degree of uncertainty about the validity of a medical assessment. This uncertainty is often expressed through specific *verbs*, *adverbs*, or *adjectives* in natural language. In this paper, we look into a *representation* of such graded statements by presenting a simple non-normal *modal logic* which comes with a set of modal operators, directly associated with the words indicating the uncertainty and interpreted through *confidence intervals* in the model theory. We complement the model theory by a set of RDFS-/OWL 2 RL-like entailment (*if-then*) rules, acting on the syntactic representation of modalized statements. Our interest in such a formalization is related to the use of OWL as the *de facto* standard in (medical) ontologies today and its weakness to represent and reason about assertional knowledge that is uncertain or that changes over time. The approach is not restricted to medical statements, but is applicable to other graded statements as well.

1 Introduction & Background

Medical natural language statements uttered by physicians or other health professionals and found in medical examination letters are usually *graded*, i.e., are associated with a degree of uncertainty about the validity of a medical assessment. This uncertainty is often expressed through specific verbs, adverbs, or adjectives in natural language (which we will call *gradation words*). E.g., *Dr. X suspects that Y suffers from Hepatitis* or *The patient probably has Hepatitis* or *(The diagnosis of) Hepatitis is confirmed*.

In this paper, we look into a representation of such graded statements by presenting a simple non-standard modal logic which comes with a small set of partially-ordered modal operators, directly associated with the words indicating the uncertainty and interpreted through confidence intervals in the model theory. The approach currently only addresses modalized propositional formulae in negation normal form which can be seen as a canonical representation of natural language sentences of the above form (a kind of a *controlled natural language*).

Our interest in such a formalization is related to the use of OWL in our projects as the *de facto* standard for (medical) ontologies today and its weakness to represent and reason about assertional knowledge that is uncertain [15] or that

changes over time [12]. There are two principled ways to address such a restriction: *either* by sticking with the existing formalism (viz., OWL) and trying to find an encoding that still enables some useful forms of reasoning [15]; *or* by deviating from a defined standard in order to arrive at an easier, intuitive, and less error-prone representation [12].
Here, we follow the latter avenue, but employ and extend the standard entailment rules from [6] and [18] for positive binary relation instances in RDFS and OWL towards modalized n-ary relation instances, including negation. These entailment rules talk about, e.g., subsumption, class membership, or transitivity, and have been found useful in many applications. The proposed solution has been implemented in *HFC* [13], a forward chaining engine that builds Herbrand models which are compatible with the open-world view underlying OWL. The approach presented in this paper is clearly not restricted to medical statements, but is applicable to other graded statements as well (including *trust*), e.g., technical diagnosis (*The engine is probably overheated*) or more general in everyday conversation (*I'm pretty sure that X has signed a contract with Y*) which can be seen as the common case (contrary to true *universal* statements).

2 Graded Medical Statements: OWL vs. Modalized Representation

We note here that our initial modal operators were inspired by the *qualitative information parts* of diagnostic statements from [15] shown in Figure 1, but we might have chosen other operators, capturing the meaning of the gradation words used in the examples at the beginning of Section 1 (e.g., *probably*).

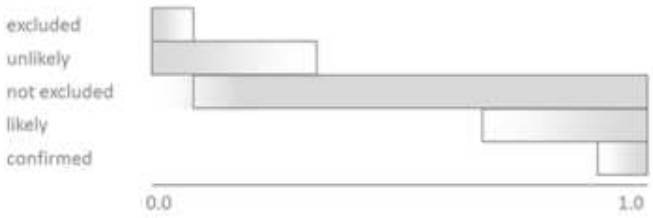


Fig. 1. Vague schematic mappings of the qualitative information parts *excluded* (E), *unlikely* (U), *not excluded* (N), *likely* (L), and *confirmed* (C) to *confidence intervals*, as used in this paper. Figure taken from [15].

These qualitative parts were used in statements about, e.g., liver inflammation with varying levels of detail. From this, we want to infer that, e.g., **if** *Hepatitis is confirmed* **then** *Hepatitis is likely* but **not** *Hepatitis is unlikely*. And **if** *Viral Hepatitis B is confirmed*, **then** both *Viral Hepatitis is confirmed* **and** *Hepatitis is confirmed* (generalization). Things "turn around" when we look at the adjectival modifiers *excluded* and *unlikely*: **if** *Hepatitis is excluded* **then** *Hepatitis is unlikely*, but **not** *Hepatitis is not excluded*. Furthermore, **if** *Hepatitis is excluded*,

then both *Viral Hepatitis is excluded* **and** *Viral Hepatitis B is excluded* (specialization). The set of *plausible* entailments for this kind of graded reasoning is depicted in Figure 2.

		Being said to have hepatitis (H) / viral hepatitis (vH) / viral hepatitis B (vHB) is…														
Precondition:		confirmed			likely			not excluded			unlikely			excluded		
Entailment:		H	vH	vHB	H	vH	vHB	H	vH	vHB	H	vH	vHB	H	vH	vHB
confirmed	H	x	x	x												
	vH		x	x												
	vHB			x												
likely	H	x	x	x	x	x	x									
	vH		x	x		x	x									
	vHB			x			x									
not excluded	H	x	x	x	x	x	x	x	x	x						
	vH		x	x		x	x		x	x						
	vHB			x			x			x						
unlikely	H										x			x		
	vH										x	x		x	x	
	vHB										x	x	x	x	x	x
excluded	H													x		
	vH													x	x	
	vHB													x	x	x

Fig. 2. Statements about liver inflammation with varying levels of detail: *Viral Hepatitis B* (vHB) implies *Viral Hepatitis* (vH) which implies *Hepatitis* (H). The matrix depicts entailments considered plausible, based on the inferences that follow from Figure 1. Hepatitis and its subclasses can be easily replaced by other medical situations/diseases. Figure taken from [15].

[15] consider five encodings (one outside the expressivity of OWL), from which only two were able to fully reproduce the inferences from Figure 2. Let us quickly look on approach 1, called *existential restriction*, before we informally present its modal counterpart (we will use abstract description logic syntax here [2]):

$$\mathsf{HepatitisSituation} \equiv \mathsf{ClinicalSituation} \sqcap \exists \mathsf{hasCondition.Hepatitis}$$

% *Hepatitis subclass hierarchy*

$$\mathsf{ViralHepatitisB} \sqsubseteq \mathsf{ViralHepatitis} \sqsubseteq \mathsf{Hepatitis}$$

% *vagueness via two subclass hierarchies*

$$\mathsf{IsConfirmed} \sqsubseteq \mathsf{IsLikely} \sqsubseteq \mathsf{IsNotExcluded} \qquad \mathsf{IsExcluded} \sqsubseteq \mathsf{IsUnlikely}$$

% *a diagnostic statement about Hepatitis*

$$\mathsf{BeingSaidToHaveHepatitisIsConfirmed} \equiv \mathsf{DiagnosticStatement} \sqcap \forall \mathsf{hasCertainty.IsConfirmed} \sqcap \exists \mathsf{isAboutSituation.HepatitisSituation}$$

Standard OWL reasoning under this representation then ensures that, for instance,

$$\mathsf{BeingSaidToHaveHepatitsIsConfirmed} \sqsubseteq \mathsf{BeingSaidToHaveHepatitisIsLikely}$$

is the case, exactly one of the plausible inferences from Figure 2.
The encodings in [15] were quite cumbersome as the primary interest was to stay within the limits of the underlying calculus (OWL). Besides coming up with complex encodings, only minor forms of reasoning were possible, viz., subsumption reasoning. These disadvantages are a result of two conscious decisions:

OWL only provides unary and binary relations (concepts and roles) and comes up with a (mostly) fixed set of entailment/tableaux rules.
In our approach, however, the *qualitative information parts* from Figure 1 are first class citizens of the object language (the modal operators) and *diagnostic statements* from the Hepatitis use case are expressed through the binary property $\mathsf{suffersForm}$ between p (patients, people) and d (diseases, diagnoses). The plausible inferences are then simply a *byproduct* of the *instantiation* of the entailment rule schemas (G) from Section 5.1, and (S1) and (S0) from Section 5.2 for property $\mathsf{suffersForm}$ (the rule variables are universally quantified; $\top$ = *universal truth*; C = *confirmed*; L = *likely*), e.g.,

(S1) $\top\mathsf{ViralHepatitisB}(d) \wedge \mathsf{ViralHepatitisB} \sqsubseteq \mathsf{ViralHepatitis} \rightarrow \top\mathsf{ViralHepatitis}(d)$
(G) $C\mathsf{suffersFrom}(p, d) \rightarrow L\mathsf{suffersFrom}(p, d)$

Two things are worth to be mentioned here. *Firstly*, not only OWL-like properties (binary relations) can be graded, such as $C\mathsf{suffersFrom}(p, d)$ (= *it is confirmed that p suffers from d*), but also class membership (unary relations), e.g., $C\mathsf{ViralHepatitisB}(d)$ (= *it is confirmed that d is Viral Hepatitis B*). However, as the original OWL example above is unable to make use of any modals, we employ a special modal $\top$ here:$\top\mathsf{ViralHepatitisB}(d)$. *Secondly*, modal operators are only applied to assertional knowledge, involving individuals (the ABox in OWL)—neither axioms about classes (TBox) nor properties (RBox) are being affected by modals, as they are supposed to express universal truth.

3 Confidence of Statements and Confidence Intervals

We address the *confidence* of an asserted medical statement [15] through *graded* modalities applied to propositional formulae: E (*excluded*), U (*unlikely*), N (*not excluded*), L (*likely*), and C (*confirmed*). For various (technical) reasons, we add a *wildcard* modality ? (*unknown*), a complementary *failure* modality ! (*error*), plus two further modalities to syntactically state definite truth and falsity: $\top$ (*true*) and $\bot$ (*false*). Let $\triangle$ now denotes the set of all modalities:

$$\triangle = \{?, !, \top, \bot, E, U, N, L, C\}$$

A *measure function*

$$\mu : \triangle \mapsto [0, 1] \times [0, 1]$$

is a mapping which returns the associated *confidence interval* $[l, h]$ for a modality from $\triangle$ ($l \leq h$). We presuppose that

- $\mu(?) = [0, 1]$
- $\mu(!) = \emptyset$[3]
- $\mu(\top) = [1, 1]$
- $\mu(\bot) = [0, 0]$

In addition, we define two disjoint subsets of $\triangle$, called

- $\mathsf{1} = \{\top, C, L, N\}$
- $\mathsf{0} = \{\bot, E, U\}$

[3] Recall that an interval is a set of real numbers, together with a total ordering relation (e.g., $\leq$) over the elements, thus $\emptyset$ is a perfect, although degraded interval.

and again make a presupposition: the confidence intervals for modals from 1 *end* in 1, whereas the confidence intervals for 0 modals always *start* with 0. It is worth noting that we do *not* make use of μ in the syntax of the modal language (for which we employ the modalities from $\triangle$), but in the semantics when dealing with the satisfaction relation of the model theory (see Section 4).

We have talked about *confidence intervals* now several times without saying what we actually mean by this. Suppose that a physician says that it is *confirmed* ($= C$) that patient p suffers from disease d, given a set of recognized symptoms $S = \{s_1, \ldots, s_k\}$: $C\mathit{suffersFrom}(p, d)$.

Assuming that a different patient p' shows the same symptoms S (and only S, and perhaps further symptoms which are, however, *independent* from S), we would assume that the same doctor would diagnose $C\mathit{suffersFrom}(p', d)$.

Even an other, but similar trained physician is supposed to grade the two patients *similarly*. This similarity which originates from patients showing the same symptoms and from physicians being taught at the same medical school is addressed by confidence *intervals* and not through a *single* (posterior) probability, as there are still variations in diagnostic capacity and daily mental state of the physician. By using intervals (instead of single values), we can usually reach a consensus among people upon the *meaning* of gradation words, even though the low/high values of the confidence interval for, e.g., *confirmed* might depend on the context.

Being a bit more theoretic, we define a *confidence interval* as follows. Assume a *Bernoulli experiment* [11] that involves a large set of n patients P sharing the same symptoms S. W.r.t. our example, we would like to know whether $\mathit{suffersFrom}(p, d)$ or $\neg\mathit{suffersFrom}(p, d)$ is the case for every patient $p \in P$, sharing S. Given a Bernoulli trials sequence $\boldsymbol{X} = \langle x_1, \ldots, x_n \rangle$ with indicator random variables $x_i \in \{0, 1\}$ for a patient sequence $\langle p_1, \ldots, p_n \rangle$, we can *approximate* the *expected value* E for *suffersFrom* being *true*, given disease d and background symptoms S by the *arithmetic mean* A:

$$\mathrm{E}[\boldsymbol{X}] \approx \mathrm{A}[\boldsymbol{X}] = \frac{\sum_{i=1}^{n} x_i}{n}$$

Due to the *law of large numbers*, we expect that if the number of elements in a trials sequence goes to infinity, the arithmetic mean will coincide with the expected value:

$$\mathrm{E}[\boldsymbol{X}] = \lim_{n \to \infty} \frac{\sum_{i=1}^{n} x_i}{n}$$

Clearly, the arithmetic mean for each new *finite* trials sequence is different, but we can try to *locate* the expected value within an interval around the arithmetic mean:

$$\mathrm{E}[\boldsymbol{X}] \in [\mathrm{A}[\boldsymbol{X}] - \epsilon_1, \mathrm{A}[\boldsymbol{X}] + \epsilon_2]$$

For the moment, we assume $\epsilon_1 = \epsilon_2$, so that $\mathrm{A}[\boldsymbol{X}]$ is in the center of this interval which we will call from now on *confidence interval*.

Coming back to our example and assuming $\mu(C) = [0.9, 1]$, $C\mathit{suffersFrom}(p, d)$ can be read as being true in 95% of all cases *known* to the physician, involving

patients p potentially having disease d and sharing the same prior symptoms (evidence) $s_1, \ldots, s_k$:

$$\frac{\sum_{p \in P} \text{Prob}(\mathit{suffersFrom}(p, d) | s_1, \ldots, s_k)}{n} \approx 0.95$$

The variance of $\pm 5\%$ is related to varying diagnostic capabilities between (comparative) physicians, daily mental form, undiscovered important symptoms or examinations which have not been carried out (e.g., lab values), or perhaps even the physical stature of the patient which unconsciously affects the final diagnosis, etc, as elaborated above. Thus the individual modals from $\triangle$ express (via μ) different forms of the physician's confidence, depending on the set of already acquired symptoms as (potential) explanations for a specific disease.

4 Model Theory and Negation Normal Form

Let $\mathcal{C}$ denote the set of constants that serve as the arguments of a relation instance. In order to define basic n-ary propositional formulae (ground atoms, propositional letters), let $p(\boldsymbol{c})$ abbreviates $p(c_1, \ldots, c_n)$, for some $c_1, \ldots, c_n \in \mathcal{C}$, given $\mathit{length}(\boldsymbol{c}) = n$. In case the number of arguments do not matter, we sometimes simply write p, instead of, e.g., $p(c, d)$ or $p(\boldsymbol{c})$. As before, we assume $\triangle = \{?, !, \top, \bot, E, U, N, L, C\}$. We inductively define the set of *well-formed formulae* ϕ of our modal language as follows:

$$\phi ::= p(\boldsymbol{c}) \mid \neg\phi \mid \phi \wedge \phi' \mid \phi \vee \phi' \mid \triangle\phi$$

4.1 Simplification and Normal Form

We now syntactically *simplify* the set of well-formed formulae ϕ by restricting the uses of *negation* and *modalities* to the level of propositional letters p and call the resulting language Λ:

$$\pi ::= p(\boldsymbol{c}) \mid \neg p(\boldsymbol{c})$$

$$\phi ::= \pi \mid \triangle\pi \mid \phi \wedge \phi' \mid \phi \vee \phi' \mid$$

To do so, we need the notion of a *complement* modal δ^{C} for every $\delta \in \triangle$, where

$$\mu(\delta^{\mathsf{C}}) := \mu(\delta)^{\mathsf{C}} = \mu(?) \setminus \mu(\delta) = [0, 1] \setminus \mu(\delta)$$

I.e., $\mu(\delta^{\mathsf{C}})$ is defined as the complementary interval of $\mu(\delta)$ (within the bounds of $[0, 1]$, of course). For example, E and N (*excluded, not excluded*) or ? and ! (*unknown, error*) are already existing complementary modals. We also require *mirror* modals δ^{M} for every $\delta \in \triangle$ whose confidence interval $\mu(\delta^{\mathsf{M}})$ is derived by "mirroring" $\mu(\delta)$ to the opposite site of the confidence interval, either to the left or to the right:

$$\textbf{if } \mu(\delta) = [l, h] \textbf{ then } \mu(\delta^{\mathsf{M}}) := [1 - h, 1 - l]$$

For example, E and C (*excluded, confirmed*) or $\top$ and $\bot$ (*top, bottom*) are mirror modals. In order to transform ϕ into its *negation normal form*, we need to apply simplification rules a finite number of times (until rules are no longer applicable). We depict those rules by using the $\vdash$ relation, read as *formula* $\vdash$ *simplified formula*:

1. $?\phi \vdash \epsilon$ % *$?\phi$ is not informative at all, but its existence should alarm us*
2. $\neg\neg\phi \vdash \phi$
3. $\neg(\phi \wedge \phi') \vdash \neg\phi \vee \neg\phi'$
4. $\neg(\phi \vee \phi') \vdash \neg\phi \wedge \neg\phi'$
5. $\neg\triangle\phi \vdash \triangle^{\mathsf{C}}\phi$ (example: $\neg E\phi = N\phi$)
6. $\triangle\neg\phi \vdash \triangle^{\mathsf{M}}\phi$ (example: $E\neg\phi = C\phi$)

Clearly, the mirror modals δ^{M} are not necessary as long as we explicitly allow for negated statements, and thus case 6 can, in principle, be dropped.
What is the result of simplifying $\triangle(\phi \wedge \phi')$ and $\triangle(\phi \vee \phi')$? Let us start with the former case and consider as an example the statement about an engine that *a mechanical failure m and an electrical failure e is confirmed*: $C(m \wedge e)$. It seems plausible to simplify this expression to $Cm \wedge Ce$. Commonsense tells us furthermore that neither Em nor Ee is compatible with this description.
Now consider the "opposite" statement $E(m \wedge e)$ which must *not* be rewritten to $Em \wedge Ee$, as *either Cm or Ce* is well *compatible* with $E(m \wedge e)$. Instead, we rewrite this kind of "negated" statement as $Em \vee Ee$, and this works fine with either Cm or Ce.
In order to address the other modal operators, we generalize these plausible inferences by making a distinction between $\mathsf{0}$ and $\mathsf{1}$ modals (see Section 3):

7a. $\mathsf{0}(\phi \wedge \phi') \vdash \mathsf{0}\phi \vee \mathsf{0}\phi'$
7b. $\mathsf{1}(\phi \wedge \phi') \vdash \mathsf{1}\phi \wedge \mathsf{1}\phi'$

Now let us consider disjunction inside the scope of a modal operator. As we do allow for the full set of Boolean operators, we are allowed to deduce

8. $\triangle(\phi \vee \phi') \vdash \triangle(\neg(\neg(\phi \vee \phi'))) \vdash \triangle(\neg(\neg\phi \wedge \neg\phi')) \vdash \triangle^{\mathsf{M}}(\neg\phi \wedge \neg\phi')$

This is, again, a conjunction, so we apply schemas 7a and 7b, giving us

8a. $\mathsf{0}(\phi \vee \phi') \vdash \mathsf{0}^{\mathsf{M}}(\neg\phi \wedge \neg\phi') \vdash \mathsf{1}(\neg\phi \wedge \neg\phi') \vdash \mathsf{1}\neg\phi \wedge \mathsf{1}\neg\phi' \vdash \mathsf{1}^{\mathsf{M}}\phi \wedge \mathsf{1}^{\mathsf{M}}\phi' \vdash \mathsf{0}\phi \wedge \mathsf{0}\phi'$
8b. $\mathsf{1}(\phi \vee \phi') \vdash \mathsf{1}^{\mathsf{M}}(\neg\phi \wedge \neg\phi') \vdash \mathsf{0}(\neg\phi \wedge \neg\phi') \vdash \mathsf{0}\neg\phi \vee \mathsf{0}\neg\phi' \vdash \mathsf{0}^{\mathsf{M}}\phi \vee \mathsf{0}^{\mathsf{M}}\phi' \vdash \mathsf{1}\phi \vee \mathsf{1}\phi'$

Note how the modals from $\mathsf{0}$ in 7a and 8a act as a kind of *negation* to turn the logical operators into their counterparts, similar to de Morgan's law.

4.2 Model Theory

In the following, we extend the standard definition of modal (Kripke) frames and models [3] for the *graded* modal operators from $\triangle$ by employing the measure function μ and focussing on the minimal definition for ϕ in Λ. A *frame* $\mathcal{F}$ for the probabilistic modal language Λ is a pair

$$\mathcal{F} = \langle \mathcal{W}, \triangle \rangle$$

where $\mathcal{W}$ is a non-empty set of *worlds* (or *situations, states, points, vertices*) and $\triangle$ a family of binary relations over $\mathcal{W} \times \mathcal{W}$, called *accessibility relations*. Note that we have overloaded $\triangle$ (and each $\delta \in \triangle$) in that it refers to the modals used in the syntax of Λ, but also to depict the binary relations, connecting worlds.

A *model* $\mathcal{M}$ for the probabilistic modal language Λ is a triple

$$\mathcal{M} = \langle \mathcal{F}, \mathcal{V}, \mu \rangle$$

such that $\mathcal{F}$ is a *frame*, $\mathcal{V}$ a *valuation*, assigning each proposition ϕ a subset of $\mathcal{W}$, viz., the set of worlds in which ϕ holds, and μ a mapping, returning the confidence interval for a given modality from $\triangle$. Note that we only require a definition for μ in $\mathcal{M}$ (the model, but *not* in the frame), as $\mathcal{F}$ represent the relational structure without interpreting the edge labeling (the modal names) of the graph.

The *satisfaction relation* $\models$, given a model $\mathcal{M}$ and a specific world w is inductively defined over the set of well-formed formulae of Λ in *negation normal form* (remember $\pi ::= p(\boldsymbol{c}) \mid \neg p(\boldsymbol{c})$):

1. $\mathcal{M}, w \models p(\boldsymbol{c})$ **iff** $w \in \mathcal{V}(p(\boldsymbol{c}))$ **and** $w \notin \mathcal{V}(\neg p(\boldsymbol{c}))$
2. $\mathcal{M}, w \models \neg p(\boldsymbol{c})$ **iff** $w \in \mathcal{V}(\neg p(\boldsymbol{c}))$ **and** $w \notin \mathcal{V}(p(\boldsymbol{c}))$
3. $\mathcal{M}, w \models \phi \wedge \phi'$ **iff** $\mathcal{M}, w \models \phi$ **and** $\mathcal{M}, w \models \phi'$
4. $\mathcal{M}, w \models \phi \vee \phi'$ **iff** $\mathcal{M}, w \models \phi$ **or** $\mathcal{M}, w \models \phi'$
5. **for all** $\delta \in \triangle$: $\mathcal{M}, w \models \delta\pi$ **iff** $\frac{\#\{u \mid (w,u) \in \delta \textbf{ and } \mathcal{M}, u \models \pi\}}{\#\{u \mid (w,u) \in \delta' \textbf{ and } \delta' \in \triangle\}} \in \mu(\delta)$

The last case of the satisfaction relation addresses the modals: for a world w, we look for the successor states u that are directly reachable via δ and in which π holds, and divide the number of such states by the number of all worlds that are directly reachable from w. This number between 0 and 1 must lie in the confidence interval $\mu(\delta)$ of δ in order to satisfy $\delta\pi$, given $\mathcal{M}, w$.

It is worth noting that the satisfaction relation above differs in its handling of $\mathcal{M}, w \models \neg p(\boldsymbol{c})$, as negation is *not* interpreted through the *absence* of $p(\boldsymbol{c})$ ($\mathcal{M}, w \not\models p(\boldsymbol{c})$), but through the *existence* of $\neg p(\boldsymbol{c})$. This treatment addresses the *open-world* nature in OWL and the evolvement of a (medical) domain over time.

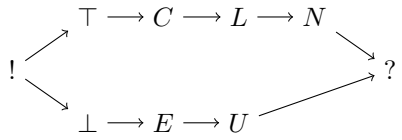
We also note that the definition of the satisfaction relation for modalities (last clause) is related to the *possibility operators* $M_k \cdot$ ($= \Diamond^{\geq k} \cdot$; $k \in \mathbb{N}$) [4] and *counting modalities* $\cdot \geq n$ [1], used in modal logic characterizations of *description logics* with *cardinality* restrictions.

4.3 Well-Behaved Frames

As we will see later, it is handy to assume that the graded modals are arranged in a kind of hierarchy—the more we move "upwards" in the hierarchy, the more a statement in the scope of a modal becomes *uncertain*. In order to address this, we slightly extend the notion of a *frame* by a third component $\preceq \subseteq \triangle \times \triangle$, a partial order between modalities:

$$\mathcal{F} = \langle \mathcal{W}, \triangle, \preceq \rangle$$

Let us consider the following modal hierarchy that we build from the set $\triangle$ of already introduced modals:



This graphical representation is just a compact way to specify a set of 33 binary relation instances over $\triangle$, such as, e.g., $\top \preceq \top$, $\top \preceq N$, $C \preceq N$, $\bot \preceq ?$, or $! \preceq ?$. The above mentioned form of uncertainty is expressed by the measure function μ in that the associated confidence intervals become larger:

$$\textbf{if } \delta \preceq \delta' \textbf{ then } \mu(\delta) \subseteq \mu(\delta')$$

In order to arrive at a proper and intuitive model-theoretic semantics which mirrors intuitions such as **if** ϕ *is confirmed* $(C\phi)$ **then** ϕ *is likely* $(L\phi)$, we will focus here on *well-behaved* frames $\mathcal{F}$ which enforce the existence of edges in $\mathcal{W}$, given $\preceq$ and $\delta, \delta^{\uparrow} \in \triangle$:

$$\textbf{if } (w,u) \in \delta \textbf{ and } \delta \preceq \delta^{\uparrow} \textbf{ then } (w,u) \in \delta^{\uparrow}$$

However, by imposing this constraint, we also need to adapt the last case of the satisfiability relation:

5. **for all** $\delta \in \triangle$: $\mathcal{M}, w \models \delta\pi$ **iff** $\frac{\#\{u|(w,u)\in\delta^{\uparrow}, \delta\preceq\delta^{\uparrow}, \textbf{ and } \mathcal{M},u\models\pi\}}{\#\{u|(w,u)\in\delta' \textbf{ and } \delta'\in\triangle\}} \in \mu(\delta)$

Not only are we scanning for edges (w,u) labeled with δ and for successor states u of w in which π holds in the denominator (original definition), but also take into account edges marked with more general modals $\delta^{\uparrow}$, s.t. $\delta^{\uparrow} \succeq \delta$. This mechanism implements a kind of *built-in model completion* that is not necessary in ordinary modal logics as they deal with only a *single* relation (viz., unlabeled arcs) that connects elements from $\mathcal{W}$ and the two modals $\Diamond$ and $\Box$ are defined in the usual dual way: $\Box\phi \equiv \neg\Diamond\neg\phi$.

5 Entailment Rules

This section addresses a restricted subset of entailment rules which will unveil new (or implicit) knowledge from graded medical statements. Recall that these kind of statements (in negation normal form) are a consequence of the application of simplification rules as depicted in Section 4.1. Thus, we assume a *preprocessing step* here that "massages" more complex statements that arise from a representation of graded (medical) statements in *natural language*. The entailments which we will present in a moment can either be *directly* implemented in a tuple-based reasoner, such as *HFC*, or in triple-based engines (e.g., Jena, OWLIM) which need to *reify* the medical statements in order to be compliant with the RDF triple model.

5.1 Modal Entailments

The entailments presented in this section deal with *plausible* inference centered around modals $\delta, \delta' \in \triangle$, some of them partly addressed in [15] in a pure OWL setting. We use the implication sign $\rightarrow$ to depict the entailment rules

$$lhs \rightarrow rhs$$

which act as *completion* (or *materialization*) rules the way as described in, e.g., [6] and [18], and used in today's semantic repositories. We sometimes even use the bi-conditional $\leftrightarrow$ to address that the LHS and the RHS are semantically equivalent, but will indicate the direction that should be used in a practical setting. As before, we define $\pi ::= p(\boldsymbol{c}) \mid \neg p(\boldsymbol{c})$.
We furthermore assume that for every modal $\delta \in \triangle$, a *complement* modal δ^{C} and a *mirror* modal δ^{M} exist (see Section 4.1).

Lift

$$(\mathsf{L}) \quad \pi \leftrightarrow \top\pi$$

This rule interprets propositional statements as special modal formulae. It might be dropped and can be seen as a pre-processing step. We have used it in the Hepatitis example above. Usage: left-to-right direction.

Generalize

$$(\mathsf{G}) \quad \delta\pi \wedge \delta \preceq \delta' \rightarrow \delta'\pi$$

This rule schema can be instantiated in various ways, using the modal hierarchy from Section 4.3; e.g., $\top\pi \rightarrow C\pi$, $C\pi \rightarrow L\pi$, or $E\pi \rightarrow U\pi$. It has been used in the Hepatitis example.

Complement

$$(\mathsf{C}) \quad \neg\delta\pi \leftrightarrow \delta^{\mathsf{C}}\pi$$

In principle, (C) is not needed in case the statement is already in negation normal form. This schema might be useful for natural language paraphrasing (explanation). Given $\triangle$, there are two possible instantiations, viz., $E\pi \leftrightarrow \neg N\pi$ and $N\pi \leftrightarrow \neg E\pi$ (note: $\mu(E) \cup \mu(N) = [0, 1]$).

Mirror

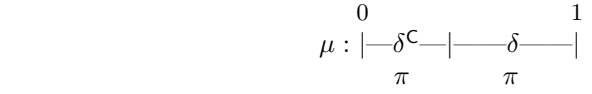
$$(\mathsf{M}) \quad \delta\neg\pi \leftrightarrow \delta^{\mathsf{M}}\pi$$

Again, (M) is in principle not needed as long as the modal proposition is in negation normal form, since we do allow for negated propositional statements $\neg p(\boldsymbol{c})$. This schema might be useful for natural language paraphrasing (explanation). For $\triangle$, there are six possible instantiations, viz., $E\pi \leftrightarrow C\neg\pi$, $C\pi \leftrightarrow E\neg\pi$, $L\pi \leftrightarrow U\neg\pi$, $U\pi \leftrightarrow L\neg\pi$, $\top\pi \leftrightarrow \bot\neg\pi$, and $\bot\pi \leftrightarrow \top\neg\pi$.

Uncertainty

$$(\mathsf{U}) \quad \delta\pi \wedge \neg\delta\pi \leftrightarrow \delta\pi \wedge \delta^{\mathsf{C}}\pi \leftrightarrow ?\pi$$

The *co-occurrence* of $\delta\pi$ and $\neg\delta\pi$ does *not* imply logical *inconsistency* (propositional case: $\pi \wedge \neg\pi$), but leads to complete *uncertainty* about the validity of π. Remember that $\mu(?) = \mu(\delta) \cup \mu(\delta^{\mathsf{C}}) = [0, 1]$ (usage: left-to-right direction):



Negation

$$\mathsf{(N)} \quad \delta(\pi \wedge \neg\pi) \leftrightarrow \delta\pi \wedge \delta\neg\pi \leftrightarrow \delta\pi \wedge \delta^{\mathsf{M}}\pi \leftrightarrow \delta^{\mathsf{M}}\neg\pi \wedge \delta^{\mathsf{M}}\pi \leftrightarrow \delta^{\mathsf{M}}(\pi \wedge \neg\pi)$$

(N) shows that $\delta(\pi \wedge \neg\pi)$ can be formulated equivalently using the mirror modal:

$$\begin{array}{cc} 0 & 1 \\ \mu : |\text{—}\delta^{\mathsf{M}}\text{—}|\text{——}|\text{—}\,\delta\,\text{—}| \\ \pi \wedge \neg\pi & \pi \wedge \neg\pi \end{array}$$

In general, (N) is *not* the modal counterpart of the *law of non-contradiction*, as $\pi \wedge \neg\pi$ is usually afflicted by vagueness, meaning that from $\delta(\pi \wedge \neg\pi)$, we can *not* infer that $\pi \wedge \neg\pi$ is the case for the concrete example in question (recall the intention behind the confidence intervals; see Section 3). There is one notable exception, involving the $\top$ and $\bot$ modals. This is formulated by the next entailment rule.

Error

$$\mathsf{(E)} \quad \top(\pi \wedge \neg\pi) \leftrightarrow \bot(\pi \wedge \neg\pi) \rightarrow !(\pi \wedge \neg\pi)$$

(E) *is* the modal counterpart of the *law of non-contradiction* (recall: $\top = \bot^{\mathsf{M}}$ and $\bot = \top^{\mathsf{M}}$). For this reason and by definition, the *error* (or *failure*) modal ! from Section 3 comes into play here. The modal ! can serve as a hint to either stop a computation the first time it occurs or to continue reasoning, but to syntactically memorize the ground atoms (viz., π and $\neg\pi$) which have led to an inconsistency. Usage: left-to-right direction.

5.2 Subsumption Entailments

As before, we define two subsets of $\triangle$, called $\mathsf{1} = \{\top, C, L, N\}$ and $\mathsf{0} = \{\bot, E, U\}$, thus 1 and 0 effectively become

$$\mathsf{1} = \{\top, C, L, N, U^{\mathsf{C}}\} \qquad \mathsf{0} = \{\bot, U, E, C^{\mathsf{C}}, L^{\mathsf{C}}, N^{\mathsf{M}}\}$$

due to the use of complement modals δ^{C} and mirror modals δ^{M} for every base modal $\delta \in \triangle$ and by assuming that $E = N^{\mathsf{C}}$, $E = C^{\mathsf{M}}$, $U = L^{\mathsf{M}}$, and $\bot = \top^{\mathsf{M}}$, together with the four "opposite" cases.

Now let $\sqsubseteq$ abbreviate relation subsumption as known from description logics and realized in OWL through `rdfs:subClassOf` (class subsumption) and `rdfs:subPropertyOf` (property subsumption). Given these remarks, we define two further very practical and plausible modal entailments which can be seen as the modal extension of the entailment rules (rdfs9) (for classes) and (rdfs7) (for properties) in RDFS; see [6].

$$\mathsf{(S1)} \quad \mathsf{1}p(\boldsymbol{c}) \wedge p \sqsubseteq q \rightarrow \mathsf{1}q(\boldsymbol{c}) \qquad \mathsf{(S0)} \quad \mathsf{0}q(\boldsymbol{c}) \wedge p \sqsubseteq q \rightarrow \mathsf{0}p(\boldsymbol{c})$$

Note how the use of p and q switches in the antecedent and the consequent, even though $p \sqsubseteq q$ holds in both cases. Note further that propositional statements π are restricted to the positive case $p(\boldsymbol{c})$ and $q(\boldsymbol{c})$, as their negation in the antecedent will not lead to any valid entailments. Here are four *instantiations* of (S0) and (S1) (remember, $C \in \mathsf{1}$ and $E \in \mathsf{0}$):

$C\mathsf{ViralHepatitisB}(x) \wedge \mathsf{ViralHepatitisB} \sqsubseteq \mathsf{ViralHepatitis} \rightarrow C\mathsf{ViralHepatitis}(x)$
$E\mathsf{Hepatitis}(x) \wedge \mathsf{ViralHepatitis} \sqsubseteq \mathsf{Hepatitis} \rightarrow E\mathsf{ViralHepatitis}(x)$

$C\mathsf{deeplyEnclosedIn}(x, y) \wedge \mathsf{deeplyEnclosedIn} \sqsubseteq \mathsf{containedIn} \rightarrow C\mathsf{containedIn}(x, y)$
$E\mathsf{containedIn}(x, y) \wedge \mathsf{superficiallyLocatedIn} \sqsubseteq \mathsf{containedIn}$
$\rightarrow E\mathsf{superficiallyLocatedIn}(x, y)$

5.3 Extended RDFS & OWL Entailments

In this section, we will consider some of the entailment rules for RDFS [6] and a restricted subset of OWL [18]. Remember that modals only head literals π, neither TBox nor RBox axioms. Concerning the original entailment rules, we will distinguish *four principal cases* to which the extended rules belong (we will only consider the unary and binary case here as used in description logics/OWL):

1. TBox and RBox axiom schemas will not undergo a modal extension;
2. rules get extended in the antecedent;
3. rules take over the modal from the antecedent to the consequent;
4. rules aggregate several modals from the antecedent in the consequent.

We will illustrate the individual cases in the following subsections with examples by using a kind of description logic syntax. Clearly, the set of extended entailments depicted here is *not complete*.

Case-1 Rules: No Modals Entailment rule rdfs11 from [6] deals with class subsumption: $\mathsf{C} \sqsubseteq \mathsf{D} \wedge \mathsf{D} \sqsubseteq \mathsf{E} \rightarrow \mathsf{C} \sqsubseteq \mathsf{E}$. As this is a terminological axiom schema, the rule stays *constant* in the modal domain. Example:

$\mathsf{ViralHepatitisB} \sqsubseteq \mathsf{ViralHepatitis} \wedge \mathsf{ViralHepatitis} \sqsubseteq \mathsf{Hepatitis}$
$\rightarrow \mathsf{ViralHepatitisB} \sqsubseteq \mathsf{Hepatitis}$

Case-2 Rules: Modals on LHS, No or $\top$ Modals on RHS The following original rule rdfs3 from [6] imposes a range restriction on objects of binary ABox relation instances: $\forall \mathsf{P.C} \wedge \mathsf{P}(\mathsf{x}, \mathsf{y}) \rightarrow \mathsf{C}(\mathsf{y})$.
The extended version (which we call Mrdfs3) needs to address the proposition in the antecedent, but must not change the consequent (even though we always use the $\top$ modality here for typing; see Section 2):

$$(\mathsf{Mrdfs3}) \quad \forall \mathsf{P.C} \wedge \delta\mathsf{P}(\mathsf{x}, \mathsf{y}) \rightarrow \top\mathsf{C}(\mathsf{y})$$

Example: $\forall \mathsf{suffersFrom.Disease} \wedge L\mathsf{suffersFrom}(x, y) \rightarrow \top\mathsf{Disease}(y)$

Case-3 Rules: Keeping LHS Modals on RHS Inverse properties switch their arguments [18]: $\mathsf{P} \equiv \mathsf{Q}^- \wedge \mathsf{P}(\mathsf{x}, \mathsf{y}) \rightarrow \mathsf{Q}(\mathsf{y}, \mathsf{x})$.
The extended version of rdfp8 simply keeps the modal operator:

$$(\mathsf{Mrdfp8}) \quad \mathsf{P} \equiv \mathsf{Q}^- \wedge \delta\mathsf{P}(\mathsf{x}, \mathsf{y}) \rightarrow \delta\mathsf{Q}(\mathsf{y}, \mathsf{x})$$

Example: $\mathsf{containedIn} \equiv \mathsf{contains}^- \wedge C\mathsf{containedIn}(x, y) \rightarrow C\mathsf{contains}(y, x)$

Case-4 Rules: Aggregating LHS Modals on RHS Now comes the most interesting case of modalized RDFS/OWL entailment rules that offers several possibilities on a varying scale between *skeptical* and *credulous* entailments, depending on the degree of uncertainty, as expressed by the measuring function μ of the modal operator. Consider the original rule rdfp4 from [18] for transitive properties P: $\mathsf{P}^+ \sqsubseteq \mathsf{P} \wedge \mathsf{P}(\mathsf{x},\mathsf{y}) \wedge \mathsf{P}(\mathsf{y},\mathsf{z}) \rightarrow \mathsf{P}(\mathsf{x},\mathsf{z})$.
How does the modal on the RHS of the extended rule look like, depending on the two LHS modals? There are several possibilities. By operating directly on the *modal hierarchy*, we are allowed to talk about, e.g., the *least upper bound* or the *greatest lower bound* of δ and δ'. When taking the associated *confidence intervals* into account, we might even play with the low and high number of the intervals, say, by applying the *arithmetic mean* or simply by *multiplying* the corresponding numbers.
Let us first consider the general rule from which more specialized versions can be derived, simply by instantiating the combination operator $\odot$:

$$\text{(Mrdfp4)} \quad \mathsf{P}^+ \sqsubseteq \mathsf{P} \wedge \delta\mathsf{P}(\mathsf{x},\mathsf{y}) \wedge \delta'\mathsf{P}(\mathsf{y},\mathsf{z}) \rightarrow (\delta \odot \delta')\mathsf{P}(\mathsf{x},\mathsf{z})$$

Here is an instantiation of Mrdfp4 dealing with the transitive relation contains from above: $C\mathsf{contains}(x,y) \wedge L\mathsf{contains}(y,z) \rightarrow (C \odot L)\mathsf{contains}(x,z)$

What is the result of $C \odot L$ here? It depends. Probably both on the application domain and the epistemic commitment one is willing to accept about the "meaning" of gradation words/modal operators. To enforce that $\odot$ is at least both *commutative* and *associative* is probably a good idea, making the sequence of modal clauses order-independent.

5.4 Custom Entailments

Custom entailments are inference rules that are not derived from universal non-modalized RDFS and OWL entailment rules (Section 5.3), but have been formulated to capture the domain knowledge of experts (e.g., physicians). Here is an example. Consider that Hepatitis B is an infectious disease

$$\mathsf{ViralHepatitisB} \sqsubseteq \mathsf{InfectiousDisease} \sqsubseteq \mathsf{Disease}$$

and note that there exist vaccines against it. Assume that the liver l of patient p quite hurts (modal C), but p has been definitely vaccinated (modal $\top$) against Hepatitis B before:

$$C\mathsf{hasPain}(p,l) \wedge \top\mathsf{vaccinatedAgainst}(p,\mathsf{ViralHepatitisB})$$

Given that p received a vaccination, the following custom rule will *not* fire (x and y below are now universally-quantified variables; z an existentially-quantified RHS-only variable):

$$\top\mathsf{Patient}(x) \wedge \top\mathsf{Liver}(y) \wedge C\mathsf{hasPain}(x,y) \wedge U\mathsf{vaccinatedAgainst}(x,\mathsf{ViralHepatitisB}) \rightarrow N\mathsf{ViralHepatitisB}(z) \wedge N\mathsf{suffersFrom}(x,z)$$

Now assume another person p' that is pretty sure (s)he was never vaccinated:

$$E\mathsf{vaccinatedAgainst}(p',\mathsf{ViralHepatitisB})$$

Given the above custom rule, we are allowed to infer that (h instantiation of z)

$N\mathsf{ViralHepatitisB}(h) \wedge N\mathsf{suffersFrom}(p', h)$

The subclass axiom from above thus assigns

$N\mathsf{InfectiousDisease}(h)$

so that we can query for patients for whom an infectious disease is *not unlikely*, in order to initiate appropriate methods (e.g., further medical investigations).

6 Related Approaches and Remarks

It is worth noting to state that this paper is interested in the representation of and reasoning with *uncertain assertional* knowledge, and neither in dealing with *vagueness* found in natural language (*very small*), nor in handling *defaults* and *exceptions* in *terminological* knowledge (*penguins can't fly*).

To the best of our knowledge, the modal logic presented in this paper uses for the first time modal operators for expressing the degree of (un)certainty of propositions. These modal operators are interpreted in the model theory through confidence intervals, by using a measure function μ. From a model point of view, our modal operators are related to *counting modalities* $\Diamond^{\geq k}$ [4, 1]—however, we do *not* require a *fixed* number $k \in \mathbb{N}$ of reachable successor states (*absolute* frequency), but instead *divide* the number of worlds v reached through label $\delta \in \triangle$ by the number of all reachable worlds, given current state w, yielding $0 \leq p \leq 1$. This fraction then is further constrained by requiring $p \in \mu(\delta)$ (*relative* frequency), as defined in case 5. of the satisfaction relation in Sections 4.2 and 4.3.

As [20] precisely put it: "*... what axioms and rules must be added to the propositional calculus to create a usable system of modal logic is a matter of philosophical opinion, often driven by the theorems one wishes to prove ...*". Clearly, the logic Λ is *no* exception and its design is driven by commonsense knowledge and plausible inferences, we try to capture.

Our modal logic can be regarded as an instance of the *normal* modal logic $\mathbf{K} := (N) + (K)$ when identifying the basic modal operator $\Box$ with the modal $\top$ (and *only* with $\top$) and by enforcing the *well-behaved* frame condition from Section 4.3. Given $\Box \equiv \top$, Λ then includes the *necessitation rule* (N) $p \rightarrow \top p$ and the *distribution axiom* (K) $\top(p \rightarrow q) \rightarrow (\top p \rightarrow \top q)$ where p, q being special theorems in Λ, viz., positive and negative propositional letters.

(N) can be seen as a special case of (L), the *Lift* modal entailment (left-to-right direction) from Section 5.1. (K) can be proven in Λ by choosing $\top \in \underline{1}$ in simplification rule *8b* (Section 4.1) and by instantiating (G), the *Generalize* modal entailment (Section 5.1), together with the application of the tautology $(p \rightarrow q) \Leftrightarrow (\neg p \vee q)$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\top(p \rightarrow q) \rightarrow (\top p \rightarrow \top q)}{\top(\neg p \vee q) \rightarrow (\neg \top p \vee \top q)}}{(\top \neg p \vee \top q) \rightarrow (\neg \top p \vee \top q)}}{\top \neg p \rightarrow \neg \top p}}{\bot p \rightarrow \top^{\mathsf{C}} p}$$

The final simplification at which we arrive is valid, since $\bot \preceq \top^{\mathsf{C}}$:

$$\mu(\bot) = [0,0] \subseteq [0,1) = \mu(\top^{\mathsf{C}})$$

Again, through (L) (right-to-left direction), Λ also incorporates the *reflexivity axiom* (T) $\top p \rightarrow p$ making Λ (at least) an instance of the system **T**. However, this investigation is in a certain sense *useless* as it does *not* address the other modals: almost always, neither (N), (K), nor (T) hold for modals from $\triangle$. Thus, we can *not* view Λ as an instance of a *poly-modal* logic.

Several approaches to representing and reasoning with uncertainty have been investigated in *Artificial Intelligence* (see [14, 5] for two comprehensive overviews). Very less so has been researched in the *Description Logic* community, and little or nothing of this research has find its way into implemented systems. [7] and [8] consider *uncertainty* in $\mathcal{ALC}$ concept hierarchies, plus concept typing of individuals (unary relations) in different ways (probability values vs. intervals; conditional probabilities in TBox vs. ABox). They do not address uncertain binary (or even n-ary) relations. [19] investigates *vagueness* in $\mathcal{ALC}$ concept descriptions to address statements, such as *the patient's temperature is high*, but also for determining membership degree (*38.5 °C*). This is achieved through *membership manipulators* which are functions, returning a truth value between 0 and 1, thus deviating from a two-valued logic. [17] defines a *fuzzy* extension of $\mathcal{ALC}$, based on Zadeh's *fuzzy logic*. As in [19], the truth value of an assertion is replaced by a membership value from $[0, 1]$. $\mathcal{ALC}$ assertions α in [17] are made fuzzy by writing, e.g., $\langle \alpha \geq n \rangle$, thus taking a single truth value from $[0, 1]$. An even more expressive description logic, Fuzzy OWL, based on OWL DL, is investigated in [16].

Our work might be viewed as a modalized version of a restricted fragment of *Subjective Logic* [9, 10], a probabilistic logic that can be seen as an extension of Dempster-Shafer belief theory. Subjective Logic addresses subjective believes by requiring numerical values for *believe* b, *disbelieve* d, and *uncertainty* u, called (subjective) opinions. For each proposition, it is required that $b + d + u = 1$. The translation from modals δ to $\langle b, d, u \rangle$ is determined by the length of the confidence interval $\mu(\delta) = [l, h]$ and its starting/ending numbers, viz., $u := h - l$, $b := l$, and $d := 1 - h$.

Acknowledgements

The research described in this paper has been co-funded by the Horizon 2020 Framework Programme of the European Union within the project PAL (Personal Assistant for healthy Lifestyle) under Grant agreement no. 643783. The authors have profited from discussions with our colleagues Miroslav Janíček and Bernd Kiefer and would like to thank the three reviewers for their suggestions.

References

1. Areces, C., Hoffmann, G., Denis, A.: Modal logics with counting. In: Proceedings of the 17th Workshop on Logic, Language, Information and Computation (WoLLIC). pp. 98–109 (2010)

2. Baader, F.: Description logic terminology. In: Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.) The Description Logic Handbook, pp. 495–505. Cambridge University Press, Cambridge (2003)
3. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, Cambridge (2001)
4. Fine, K.: In so many possible worlds. Notre Dame Journal of Formal Logic 13(4), 516–520 (1972)
5. Halpern, J.Y.: Reasoning About Uncertainty. MIT Press, Cambridge, MA (2003)
6. Hayes, P.: RDF semantics. Tech. rep., W3C (2004)
7. Heinsohn, J.: ALCP – Ein hybrider Ansatz zur Modellierung von Unsicherheit in terminologischen Logiken. Ph.D. thesis, Universität des Saarlandes (Jun 1993), in German
8. Jaeger, M.: Probabilistic reasoning in terminological logics. In: Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning (KR). pp. 305–316 (1994)
9. Jøsang, A.: Artificial reasoning with subjective logic. In: Proceedings of the 2nd Australian Workshop on Commonsense Reasoning (1997)
10. Jøsang, A.: A logic for uncertain probabilities. International Journal of Uncertainty, Fuzzyness and Knowledge-Based Systems 9(3), 279–311 (2001)
11. Krengel, U.: Einführung in die Wahrscheinlichkeitstheorie und Statistik. Vieweg, 7th edn. (2003), in German
12. Krieger, H.U.: A temporal extension of the Hayes/ter Horst entailment rules and an alternative to W3C's n-ary relations. In: Proceedings of the 7th International Conference on Formal Ontology in Information Systems (FOIS). pp. 323–336 (2012)
13. Krieger, H.U.: An efficient implementation of equivalence relations in OWL via rule and query rewriting. In: Proceedings of the 7th IEEE International Conference on Semantic Computing (ICSC). pp. 260–263 (2013)
14. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Francisco, CA, MA (1988)
15. Schulz, S., Martínez-Costa, C., Karlsson, D., Cornet, R., Brochhausen, M., Rector, A.: An ontological analysis of reference in health record statements. In: Proceedings of the 8th International Conference on Formal Ontology in Information Systems (FOIS 2014) (2014)
16. Stoilos, G., Stamou, G.B., Tzouvaras, V., Pan, J.Z., Horrocks, I.: Fuzzy OWL: uncertainty and the semantic web. In: Proceedings of the OWLED '05 Workshop on OWL: Experiences and Directions (2005)
17. Straccia, U.: Reasoning within fuzzy description logics. Journal of Artificial Intelligence Research 14, 147–176 (2001)
18. ter Horst, H.J.: Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. Journal of Web Semantics 3, 79–115 (2005)
19. Tresp, C.B., Molitor, R.: A description logic for vague knowledge. In: Proceedings of the 13th European Conference on Artificial Intelligence (ECAI). pp. 361–365 (1998)
20. Wikipedia: Modal logic — Wikipedia, The Free Encyclopedia (2015), https://en.wikipedia.org/wiki/Modal_logic, [Online; accessed 19-June-2015]

Bias in Japanese Polar Questions from Constraints on Commitment Spaces

Lukas Rieser

Kyoto University

Abstract. Since Ladd's [9] observations on inner and outer readings of negation in polar questions, two kinds of bias arising from (negative) polar questions have been distinguished: first, a polar question can convey a previous assumption of the speaker; second, the felicity of polar questions depends on the presence or absence of evidence for or against its proposition. To account for bias, outer negation has been analyzed as scoping over VERUM-focus [12] and as speech-act negation triggering a REQUEST-reading [8]. In this paper, we focus on Japanese polar questions and argue for a finer-grained classification than in previous research, taking their intonation [7] and structure into account. We argue that *no*-attachment (similar to English rising declaratives [3]) as well as speech-act negation give rise to a REQUEST-reading from which we attempt to derive observed bias patterns.

The paper is organized as follows: section one gives an overview of the negation and bias patterns of English polar questions. Section two gives an overview of Japanese polar questions including the role of accented and unaccented negation, proposing a finer-grained classification than in previous research, and arguing for two distinct positions at which outer negation enters the derivation. In section three, we lay out a proposal for analyzing some Japanese polar questions as REQUEST-speech acts, similar to rising declaratives and English polar questions with preposed negation. From this analysis, we attempt to derive their bias conditions in section four.

1 English polar questions

1.1 Negation patterns

In the literature on English polar questions, inner and outer or negation have been distinguished in at least two ways. First, negation can be interpreted inside or outside the predicate, which can be tested for by addition of polarity items. Second, negation can be realized on the main verb or on the auxiliary (preposed negation).

Three types of English polar questions have been distinguished: positive polar questions (PPQs), inner negation polar questions (INPQs), and outer negation polar questions (ONPQs). The latter two are distinguished by whether or not

negation is interpreted inside or outside the predicate, the first criterion. INPQs can be further divided into such with and without preposed negation, the second criterion which is not considered by all authors.

1.2 Bias patterns

C-bias The contextual bias (C-bias) pattern of an utterance of a proposition φ is determined by its felicity in utterance contexts which are positively biased, negatively biased, and neutral relative to φ. A context is biased relative to a proposition when there is evidence for φ or $\neg\varphi$. Büring and Gunlogson [1], from where the following examples and judgments have been adapted, define whether a context is biased or neutral based on "compelling contextual evidence", *i.e.* information which has just become available in the utterance situation and from which, considered in isolation, one can be conclude that φ or $\neg\varphi$ holds. It is important to note that this is different from one of the alternatives being more salient, as there are cases where there is no compelling contextual evidence, but where the speaker is *e.g.* for some reason more interested in the negated proposition then the non-negated proposition. We will illustrate such a case with examples (8) and (9) below.

For the bias patterns of English polar questions, consider A's utterances (1) to (3), which set up neutral, negative, and positive contexts relative to the proposition "there is a vegetarian restaurant around here". The results of judging the acceptability of the PPQ in (4), the ONPQ in (5), and the INPQ[1] in (6) against the three contexts are summarized in (7).

(1) A: Where do you want to go for dinner? (*Neutral context*)

(2) A: I bet we can find any type of restaurant in this city! (*Positive context*)

(3) A: Since you are vegetarians, we can't go out in this town.(*Negative context*)

(4) S: Is there a vegetarian restaurant around here? (*PPQ*)

(5) S: Isn't there some vegetarian restaurant around here? (*ONPQ*)

(6) S: Isn't there any vegetarian restaurant around here? (*INPQ*)

(7)

	Context: *neutral*	*positive*	*negative*	S-bias
PPQ	✓	✓	#	no
ONPQ	✓	#	✓	yes
INPQ	#	#	✓	yes

According to these results, English PPQs are good in all but negative contexts, ONPQs in all but positive contexts, and INPQs in negative contexts only. However, as example (8) below demonstrates, INPQs can be good in neutral contexts if the negative alternative is more salient[2], which differs from a negative context

[1] Changed to preposed negation from the original example, where negation is *in situ*, in order to focus on high vs. low negation as disambiguated by the polarity items.

[2] In a footnote, Büring and Gunlogson [1] note that the more marked morphological form of negation could be the cause of a preference for PPQs in neutral contexts.

in the narrow sense, as there is no direct evidence for the truth of the $\neg\varphi$. In any case, there is a clear divide when looking at biased contexts: INPQs and ONPQs are good in negative contexts, PPQs in positive ones.

S-bias An utterance of a proposition φ has S-bias when it conveys a previous belief of the speaker that φ holds. This can be determined by testing whether or not the utterance is felicitous in a scenario which supposes that the speaker is neutral,*i.e.* has no previous assumptions regarding the truth of φ. In English polar questions, the former arises from negation preposing, as this example from Romero and Han [12] shows:

[S hates Pat and Jane. As far as S knows, either of them may or may not be coming.]

A: Pat is not coming.

(8) S: Great! Is Jane not coming either?

(9) S: #Great! Isn't Jane coming either?

(8) and (9) are both polar questions with inner negation as evidenced by the NPI *either*. The difference is that negation in (8) is *in situ*, while negation in (9) is preposed, *i.e.*, realized on the auxiliary instead of the main verb, thus syntactically outside the predicate. In the given scenario, which has it that S is neutral in the sense of having no previous assumptions relative to the proposition "Jane is coming", (9) is bad, but (8) is good. From this, Romero and Han conclude that only preposed negation gives rise to obligatory S-bias.

However, in a scenario without A's preceding utterance or the information that S hates Jane, (8) can also give rise to S-bias, as the negative alternative is not made salient. We call this "circumstantial", as opposed to "obligatory" S-bias, as it is a pragmatic effect of using a negative question when the positive alternative is more salient (which is the default case).

Next, ONPQs always have obligatory S-bias due to preposed negation,[3] while PPQs do not as they have no negation in the first place. Summarizing, INPQs with *in situ*-negation have only circumstantial, INPQs with preposed negation and ONPQs obligatory, and PPQs no S-bias.

2 Japanese polar questions

2.1 Negation patterns

Negation patterns of Japanese polar questions differ slightly from English ones; in addition to the distinction of inner *vs.* outer negation testable by the addition of polarity items to the predicate, a distinction of inner vs. outer negation is possible by prosody. In this section, we summarize Sudo's [13] observations on bias

[3] This predicts that examples with syntactically inner, high negation like ?"Is Jane not coming, too?", if they are possible at all, do not have obligatory S-bias.

patterns of Japanese polar question and their interaction with *no*-attachment, followed by Ito and Oshima [7]'s observations on the connection between intonation and bias patterns. Combining both of them will allow for a finer-grained classification of Japanese polar questions.

2.2 Bias patterns and *no*-attachment

C-bias The C-bias patterns of the Japanese polar questions according to Sudo [13] are summarized in (12). Examples of an ONPQ and an INPQ are shown in (10) and (11), respectively. Note that in both examples, negation is *in situ*. The PPI *dareka* ("someone") disambiguates towards outer, the NPI *daremo* ("anyone") towards inner negation.

(10) Dareka hokani konai?
someone else come.NEG
"Isn't someone else coming?"

(11) Daremo hokani konai?
anyone else come.NEG
"Isn't anyone else coming?"

(12)

	Context: *neutral*	*positive*	*negative*	S-bias
PPQ	✓	#	#	no
ONPQ	✓	✓	#	yes
INPQ	#	#	✓	no
PPQ-*no*	#	✓	#	no
ONPQ-*no*	✓	✓	✓	yes
INPQ-*no*	#	#	✓	yes

(12) shows the bias pattern of polar questions without and with sentence-final attachment of a particle *no*[4] According to these results, *no*-attachment influences acceptability as follows:

- *No* is required for PPQs to be acceptable in positive contexts.
- *No* is required for ONPQs to be acceptable in negative contexts.
- *No* has no apparent effect on the C-bias pattern of INPQs.

Regarding the last point, we will argue, similar to the observations by Romero and Han [12] about English INPQs, that Japanese INPQs without *no*-attachment can also be good in neutral contexts given that the negative alternative is made sufficiently salient, based on example (15) due to Ito and Oshima [7].

[4] This particle *no* is structurally distinct form the previously mentioned morpheme *no* separating negation and the predicate in that it serves no such purpose. It has been analyzed as an evidential marker, which we will briefly discuss later.

S-bias Sudo reports S-bias of Japanese polar questions to be similar to that of English ones, *i.e.* PPQs do not have S-bias and ONPQs have obligatory S-bias. The only difference is that Japanese INPQs only show S-bias when *no* is attached. We will argue that outer negation gives rise to S-bias in Japanese polar questions, and that, again similar to English INPQs, S-bias in Japanese NO-INPQs is likely circumstantial.

2.3 Bias patterns and prosody

In this section, we discuss two distinct prosodic realizations of negation in polar questions which we will show are associated with high and low negation. The negation morpheme *nai* has lexical accent, that is an H*+L pitch accent (*cf.* Venditti [14]) with an *f0* peak during the first mora *na* and a fall in *f0* on the second mora *i*. When negation is realized on the main verb, the lexical accent of negation is incorporated into the main verb. We will conclude that negation can only be accented when interpreted inside of the predicate, while outside of the predicate, both accented and unaccented negation occur.

Prosody of inner and outer negation Ito and Oshima [7] observe that in negative polar questions from adjectival predicates, lexical accent of negation can be suppressed, resulting in two prosodic patterns: accented and unaccented negation. They label these the NN (neutral/negative) and the P (positive) pattern, respectively, in terms of "epistemic bias", while not explicitly differentiating between bias due to contextual evidence (C-bias) and previous assumptions by the speaker (S-bias).[5]

(13) and (14) show a minimal pair of a negative polar question from a predicate *amai* ("be sweet") with and without lexical accent on negation, adapted from Ito and Oshima [7]. We indicate the pitch accent on *nai* as "H*+L" for the accented, as "0" for the unaccented version.

[Scenario: S has heard oranges are sweet this year, A is eating one.]

(13) S: amaku- [$_0$ nai]?
sweet- NEG
"Isn't it sweet?"

[Scenario: S sees A, who is eating an orange, make a grimace.]

(14) S: amaku- [$_{\text{H*+L}}$ nai]?
sweet- NEG
"Is it not sweet?"

[5] In Hwang and Ito [6], experimental results are shown where positive S-bias, with or without positive evidence, is associated with unaccented negation, and neutral and negative contexts with accented negation.

The bias patterns of the negative polar questions (13) and (14) correspond to INPQs and ONPQs, respectively. In (14), the context is negatively biased as evidence in form of A's grimace has just come to the speaker's attention. Thus, considering Sudo's observations summarized above, (14) has the bias pattern of an INPQ which are good in negative contexts, in contrast to ONPQs (without *no*-attachment). Conversely, the context for (13) is neutral and the speaker has a previous expectation that oranges are sweet. As ONPQs, but not INPQs, are good in neutral contexts and ONPQs, but not INPQs have S-bias without *no*-attachment, (13) has the bias pattern of an ONPQ.

Unaccented negation in neutral contexts Parallel to the observations on English INPQs by Romero and Han [12], accented negation can occur in certain neutral contexts, given a scenario which makes the negated alternative more salient without providing contextual evidence for it, as in this example, again from Ito and Oshima [7]:

[S is looking for non-sweet Sake, A is helping.]

A: Kore to kore-wa amakunai.

(15) S: Kore-wa? Amaku- [$_{\text{H*+L}}$ nai]?
this-TOP sweet- NEG
"And this one? Is it not sweet [either]?"

This example illustrates a similar point for the C-bias pattern of Japanese polar questions as (8) does for English ones: given the right utterance situation, Japanese INPQs can be good in neutral contexts.

2.4 Prosody and outer negation

So far, we have observed that inner negation is associated with accented negation, outer negation with unaccented negation. However, as we will discuss in the next section, outer negation can occur with both accented and unaccented negation. In order to keep the prosodic distinction distinct from the structural one, we will also refer to unaccented negation as 11high", and to accented negation as "low" negation, while referring to negation interpreted inside the predicate as "inner", outside the predicate as "outer" negation.

Syntactically outer negation Negation in Japanese polar questions can occur on a copula separated from the predicate by a morpheme $n(o)$, akin to English "...isn't it?" tag-questions, but also similar to preposed negation in that there is no prosodic break between the main verb and the copula on which negation is realized. We will refer to this as "syntactically outer negation". Ito and Oshima [7] report that in such constructions, accented (thus low) negation is degraded, *i.e.*, a high negation reading is strongly preferred. (16) shows a version with syntactically outer negation of Sudo's example for an ONPQ in (10).

(16) Dareka hokani kuru n ja- [{0/#H*+L} nai?]
someone else come *no* COP NEG
"Isn't someone else coming?"

There are significant differences in usage between polar questions with syntactically outer negation and OPNQs with main verb negation. For instance, (16) can be uttered with final falling intonation, in which case it does not have interrogative force,[6] which is not possible with (10). However, we assume that negation is interpreted in the same structural position in both kinds of questions. This is supported by the facts that there is a PPI *dareka* within the predicate and that accented negation is degraded in both, which points to high, outer negation. Furthermore, they have the same C-bias conditions, both being degraded in negatively biased contexts.

***No*-attachment and accented negation** Ito and Oshima [7] further observe that when *no* attaches to an ONPQ with syntactically outer negation, accented negation becomes possible as in (17), a version of (16) with *no*-attachment.

(17) Dareka hokani kuru n ja[{0/H*+L}nai] no?
someone else come *no* COP.NEG *no*
"Isn't someone else coming?"

The availability of accented negation in (17) poses a problem for the dual split into high and outer vs. low and inner negation, as the PPI and negation being realized outside the predicate point towards outer negation, but negation can be accented, which is associated with low negation. We will claim that there are two structual positions for outer negation, of which accented negation in (17) is the lower, unaccented negation the higher one.

2.5 Additional types of polar questions

The accent pattern observed in (17) carries over to ONPQs with main verb negation and *no*-attachment:

(18) S: Dareka hokani [{0/H*+L} konai] no?
someone else come.NEG *no*
"Isn't someone else coming?"

With accented negation, the question in (18) is good in negative contexts, such as the one set up by the A's utterance in (19), given that A knows who will be coming to the meeting.

(19) A: We are all here now. Shall we begin the meeting?

[6] (16) with a final fall is an assertion with an additional expressive meaning that is outside the scope of this paper to discuss.

Conversely, in positive and neutral contexts, unaccented negation is preferred. We conclude that negative contexts require accented negation in Japanese polar questions, and propose the following, more fine-grained categorization of Japanese polar questions given alongside their C-bias patterns:

(20)

	NEG	Context:		
	accented	*neutral*	*positive*	*negative*
PPQ	n/a	✓	#	#
PPQ-*no*	n/a	#	✓	#
ONPQ(-*no*)	no	✓	✓	#
ONPQ-*no*	yes	#	#	✓
INPQ(-*no*)	yes	∼	#	✓

Table (20) reflects the observation that negative polar questions can be good in neutral contexts as well, but only in a limited number of scenarios (symbolized by "∼") in which the negative alternative is made more salient. It also shows the two types of ONPQs with unnacented (high) and accented (low) negation, respectively.

2.6 Structural positions for negation

In the previous section, we have shown that when *no*-attaches to an ONPQ, negation can be accented, but outside of the predicate. In order to explain this data point, we will argue that:

- Accented (low) negation is propositional negation,
  unaccentend (high) negation is non-propositional negation.
- There are two structural positions for outer negation in Japanese:
  Accented negation above FocP and unaccented negation above ForceP.

Low outer negation Based on a split CP in the spirit of Rizzi [11], we assume a simplified structure of the Japanese phrase as in (21)[7], where $(\neg)\varphi$ indicates the position of the (negated) proposition.

(21) $[_{\text{ForceP}}\ [_{\text{FocP}}\ [_{\text{FinP}}\ [_{\text{TP}}\ (\neg)\varphi\ ]\ \text{FIN}]\ \text{FOC}]\ \text{FORCE}]$

Building on the analysis focus-constructions in Hiraiwa and Ishihara [5], we assume that *no* separating negation from the main verb in syntactically outer negation is a FIN-head followed by a copula *Which they claim functions as a focus-particle.* in FOC, thus outer negation on a copula following N(O) is interpreted outside of FocP. In fact, when syntactically outer negation in a polar question is accented, as it can be in (17), both structure and intonation are very similar to focus constructions like the one shown in (22).

(22) Kyoo-wa [$_{\text{F}}$ Takashi-ga] kuru n ja[$_{\text{H*+L}}$ nai].
today-TOP PN-NOM come *no* COP.NEG
"Not Takashi is coming today."

[7] Similar to what Krifka [8] proposes for English polar questions.

The focus construction in (22) is an assertion with syntactically outer negation, and narrow focus on the constituent "Takashi". Comparing this construction to an OPNQ with syntactically outer, accented negation as in (17), the only difference except for final falling vs. rising intonation is whether or not a constituent is singled out for negation. Given this parallel, we assume that the structural position in which accented negation is realized in ONPQs with syntactically outer oas well as with main verb negation is the same as that of negation in focus constructions, *i.e.* above FocP.

High outer negation As for the other structural position for outer negation in Japanese polar questions, that for unaccented outer negation, we assume that it is higher than that of accented negation based on the observation that it does not share any properties with low, propositional negation, neither in intonation (recall that accented negation is obligatory in INPQs), nor in bias patterns (as it is bad in negative contexts). Thus, it must select a phrase higher than FocP, which we will argue to be ForceP in the following section.

3 Request speech-acts and spech-act negation

3.1 Constraining the commitment space

In order to explain the interpretation of unaccented high negation, we adopt Krifka's [8] idea that negation can be interpreted in a position higher than ForceP, thus "denegating" a speech-act. In order to illustrate this for Japanese polar questions, we make use of these three concepts, given here in a highly simplified form:

- *Commitment:*
  When a speaker asserts a proposition φ, she is committed to the truth of φ.
- *Commitment states* are the sets of all commitments that have been made in the course of the discourse.
- The *Commitment space* is the set of commitment states which are connected by speech-acts.

When a speech act is performed, this updates the commitment space. For example, when the speaker asserts φ, it will be added to the speakers commitments and thus update the current commitment state. We make use of the following three speech acts:

- $\textsc{request}_{S,A}$, a speech act performed by the speaker, which takes another speech act as its argument and restricts the future development of the commitment space to such in which the addresse performs, or, in the case of a denegation request, does *not* perform this speech act
- $\textsc{assert}(\varphi)$ updates the commitment space by adding a commitment by a discourse participant that φ or $\neg\varphi$ holds.
- $\sim\textsc{assert}(\varphi)$ (denegation, written as "$\sim$", of assertion) excludes the assertion of φ from future developments of the commitment space.

The idea we want to implement is that each type of polar question restricts the addressee's discourse moves in a specific way, which can be represented using the three speech-act operators above. Facing a constrained commitment space, the addressee can either comply or reject the speaker's move.

3.2 *No*-attachment, request-readings, and rising declaratives

Before moving on to speech-act negation, we will argue that *no*-attachment, together with a final rise, makes a REQUEST-speech act. In the case of PPQs and INPQs, which we will discuss first, this is effectively the same as considering them rising declaratives. [3] proposes that declaratives are context change potentials which publicly commit one of the discourse participants to their proposition, and that falling intonation resolves the discourse participant to the speaker, rising intonation to the addressee.[8] Publicly committing the addressee to a proposition and requesting that they commit to a proposition are effectively the same. However, denegation requests, which we will discuss below, can not be explained by rising declaratives. In this section, we make the following claims:

- *No* in polar questions attaches to the FORCE-head.
- *No*-attachment and a final rise result in a REQUEST speech-act.
- *No*-attachment in PPQs and in NPQs with accented negation are equivalent to rising declaratives.

PPQs and INPQs Our claims about *no*-attachment can explain that Japanese PPQs are degraded without *no* in positively biased contexts. Consider for example the PPQ in (23), modified from Sudo [13], which is good in the positive context in (24), but degraded in the neutral context in (25).

(23) S: Ima ame futteru no?
now rain fall.PROG *no*
"Is it raining now?"

(24) [S to A, who has entered the windowless room wearing a wet raincoat.]

(25) [S to A, over the phone.]

With a very similar example, Gunlogson [3] reports the same felicity conditions for rising declaratives as for *no*-PPQs.

As for INPQs, we have proposed that they are essentially the mirror image of PPQs, and are good in neutral contexts if they are constructed to make the negated proposition more salient for reasons beyond contextual evidence. This claim predicts that in neutral contexts, *no*-INPQs should be degraded, which we can confirm as in (15), this is the case[9], and assume the structures in (26) and (27) for Japanese PPQs/INPQs with and without *no*-attachment, respectively.

[8] The existence of such an intonational morpheme in Japanese has also been suggested by Davis [2] in his analysis of the particle *yo*.

[9] With the caveat that this may be because of the speakers neutrality and a lack of S-bias in bare INPQs, but see the comments on the circumstantial nature of bias in *no*-INPQs further below.

(26) $[_{\text{ForceP}}\ [_{\text{FinP}}\ [_{\text{TP}}\ (\neg)\varphi\ ]\ \text{FIN}]\ \text{INT}]$

(27) $[_{\text{ForceP}}\ [_{\text{ForceP}}\ [_{\text{FinP}}\ [_{\text{TP}}\ (\neg)\varphi\ ]\ \text{FIN}]\ \text{ASS}]\ \text{REQ-}no]$

While (26) is an unbiased question, which limits the addressee's admissible moves to committing to either φ or $\neg\varphi$, (27) is interpreted as in (28), leaving the addressee the option to commit to $(\neg)\varphi$, or to reject the speakers's move.

(28) $\text{REQUEST}_{\text{S,A}}(\text{ASSERT}((\neg)\varphi))$

As argued above, this is effectively the same as publicly committing the speaker to the truth of $(\neg)\varphi$.

Accented ONPQs Next, recall that accented outer negation is degraded without *no*-attachment. We propose that the structural similarities between focus negation and accented ONPQs can be explained by *no* attaching to sentential force just as in PPQs and INPQs, resulting a REQUEST- or rising declarative-reading. Following our assumptions made about the position of accented outer negation, the structure we propose for low outer negation such as the ONPQ in (18) with accented negation, repeated here as (29), is given in (30).

(29) S: Dareka hokani $[_{\text{H*+L}}$ konai] no?
someone else come.NEG *no*
"Isn't someone else coming?"

(30) $[_{\text{ForceP}}\ [_{\text{ForceP}}\ [_{\text{NegP}}\ [_{\text{FocP}}\ [_{\text{FinP}}\ [_{\text{TP}}\ \varphi]\ \text{FIN}]\ \text{FOC}]\ \text{NEG}]\ \text{ASS}]\ \text{REQ-}no]$

While negation is not interpreted within TP allowing PPIs, it is not higher than Force, resulting in the same interpretation as for INPQs, *i.e.*,(28) with a negated proposition, which is compatible with the C-bias patterns we have observed. We will discuss the possible differences between INPQs and low ONPQs in terms of S-bias at the end of the next section.

***No* as an evidential marker** An alternative account of the contribution of *no* which does not appeal to structural positions is taking it to be an evidential particle. For example, Davis [2] proposes that *no* adds an evidence presupposition to a question without making any other contribution. This is possibly a simpler alternative for PPQs and INPQs, and possibly for accented ONPQs. However, if we assume that *no* marks that there is evidence for the proposition it attaches to, the question arises what it does in ONPQs with unaccented negation, which are good in both neutral and positive contexts. Also, *no* should be obligatory in INPQs when there is evidence for $\neg\varphi$, which is not the case, as example (14) shows. Furthermore, while *no* has uses which can be interpreted as evidential in assertions,[10] it has a large number of other uses which are not related to evidentiality (*cf.* Noda [10]).

[10] For example, *daroo* utterances expressing speaker inferences in which evidence the inference is based on has to be obligatorily marked with *no* (*cf.* Hara [4])

3.3 Unaccented outer negation in denegation requests

We have proposed structures and interpretations for positive polar questions and such with accented negation. What is left are ONPQs with unaccented negation, as the one in (31) (repeated from (10) and (18) for versions without and with *no*-attachment).

(31) S: Dareka hokani [$_0$ konai] (no)?
someone else come.NEG *no*

Out of the two positions for negation outside of the TP, unaccented negation takes the higher one. In parallel to Krifka's proposal that outer negation in English polar questions takes scope over ASSERTION we propose the structure in (32). Our claim is that, similar to what Krifka proposes for English, negation scoping over assertive force triggers a REQUEST-reading.

(32) $[_{\text{ForceP}}\ [_{\text{NegP}}\ [_{\text{ForceP}}\ [_{\text{FinP}}\ [_{\text{TP}}\ \varphi]\ \text{FIN}]\ \text{ASS}]\ \text{NEG}]\ \text{REQ}]$

This is interpreted as the denegation request in (33).

(33) $\text{REQUEST}_{\text{S,A}}(\sim\text{ASSERT}(\varphi))$

This constrains the commitment space in a way that the possible moves for the addressee are: to reject the speaker's request (and, for example, assert φ or claim ignorance), or to accept the denegation of this speech act,*i.e.*, to exclude all future moves which depend on the addressee asserting φ, thus committing to it.

Furthermore, *no* can attach to ONPQs with high negation. If the proposal that unaccented negation triggers a REQUEST-reading is on the right track, *no*-attachment is not necessary to license it in high ONPQs. It seems an attractive possibility to assume that *no*-attachment is the preferred strategy in positive contexts, in parallel to PPQs, but we could not get clear judgments on this (which is actually expected if unaccented *no*-ONPQs are "double-marked" as REQUESTs).

4 Deriving the bias patterns

In this section, we will claim that: C-bias arises from assertion-request readings due to *no*-marking of (negated) propositions, S-bias arises from outer negation (both propositional and speech-act negation), discussing the different types of polar questions as alternatives to each other in a given context or utterance situation.

4.1 C-bias from *no*-marking and negation

Our claims regarding C-bias are:

- Negative C-bias arises from REQUEST-utterances with accented negation.
- Positive C-bias arises from REQUEST-utterances without negation.
- Linguistic marking of biased contexts is mandatory in Japanese.

Accented negation We have observed that in negatively biased context, accented negation is mandatory. Also, *no*-attachment to both PPQs and INPQs is degraded in neutral contexts, *cf.* table (20), while INPQs without *no*-attachment can be good in neutral contexts as well. We claim that biased contexts need to be marked, which can be achieved by *no*-attachment, which forces a REQUEST-reading. Here, a commitment space update $\text{REQUEST}_{S,A}(\text{ASSERT}(\neg\varphi))$ and a rising declarative are equivalent, as both require a reaction to a commitment to $\neg\varphi$ from the addressee. As for INPQs, without *no*-attachment, we hypothesize that chosing an INPQ over a PPQ is already signaling that the negative alternative is more salient, which, if the context is not set up in such a way, can conceivably substitue a REQUEST-utterance in terms of marking the biased context.[11]

In summary, we claim that accented negation in conjunction with *no* gives rise to negative C-bias. ONPQs with accented negation result in the same speech-act as REQUEST-INPQs, so they are the same in terms of C-bias (see below for differences in terms of S-bias).

PPQs The requirement that positively biased contexts be marked is satisfied by PPQs, in which *no*-attachment gives rise to a REQUEST-reading and thus C-bias when occurring in positive contexts.

Unaccented negation Finally, the case of ONPQs with unaccented negation and C-bias is more intricate. We can exclude them from our considerations on C-bias if we assume that they, as denegation requests rather than requests for assertions, do not give rise to C-bias at all. Their badness in negative contexts can be explained by the condition that biased contexts be marked and the availability of the alternative ONPQ with accented negation. A remaining puzzle is *no*-attachment: if unaccented negation is sufficient to give rise to a (denegation) REQUEST-reading *no* appears superfluous. The possibility that *no*-attachment somehow can serve to somehow pragmatically strengthen the denegation request is left for further research.

4.2 S-bias from outer and from accented negation

ONPQs have obligatory S-bias in Japanese, *no*-INPQs have been reported to have S-bias as well. Furthermore, there appears to be a difference in the strength of S-bias in different types of ONPQs. We claim that:

- S-bias arises from outer negation, more strongly from stressed negation.
- S-bias can arise from negative REQUEST-utterances.

From the observed bias patterns, we conclude that outer negation, *i.e.* negation outside of TP, is a source of obligatory S-bias.

[11] Another conceivable possiblity is that there are two versions of plain INPQs, one REQUEST-reading and one plain reading, as Krifka [8] proposes for English INPQs. However, since there is no such thing as preposed negation in Japanese INPQs, this seems less likely.

Unaccented ONPQs As for the communicative effect of OPNQs with high negation which we have argued to be denegation requests, Krifka [8] proposes that denegation requests can be used to check whether φ is "an option to be considered". This effect presumably arises in contrast with a request for assertion of $(\neg)\varphi$ on one hand, which would be stronger and thus preferred when there are clear reasons for the speaker to to assume that $(\neg)\varphi$ holds, and in contrast with a plain polar question on the other. Plain questions do not convey that the speaker has a specific interest in φ, which denegation requests do — this is a possible source for the conveyed speaker assumption, in the form of S-bias, that φ holds. In other words, OPNQs are oriented towards the positive option but do not force the addressee to react to a commitment to its truth.

Accented ONPQs Accented ONPQs have a similar communicative effect, but it appears to be stronger: in (13), for example, a rather general expectation of the speaker suffices to license the unaccented ONPQ, which would arguably not be sufficient for an accented ONPQ. Furthermore, if the positions we have proposed for outer negation and our considerations on S-bias in unaccented ONPQs are on the right track, the source of S-bias in accented ONPQs is different from unaccented ones, as they are not denegation requests. Here, a connection can be made with Romero and Han's [12] proposal of preposed negation scoping over VERUM-focus giving rise to S-bias in English polar questions: it is conceivable that in accented ONPQs, negation taking scope over FocP gives rise to the conveyed speaker assumption that φ holds.

***No*-INPQs** Finally, INPQs with *no*-attachment are also reported to show S-bias, which is not compatible with the claim that S-bias arises from outer negation. If the proposal that INPQs are the mirror image of PPQs, but marked when the context is not manipulated to make the negative alternative more salient, the apparent S-bias from *no*-INPQs could be an effect similar to the apparent badness of plain INPQs in neutral contexts. If this is the case, *no*-INPQs should be good in negative contexts even when the speaker has no previous assumptions. This is similar to negative rising declaratives in English, which require a negative context, but do not necessarily give rise to S-bias. If this is on the right track, *no*-INPQs have but circumstantial S-bias.

5 Conclusion

We have proposed that prosody has to be considered alongside the interpretation of negation inside or outside the predicate in order to derive the bias patterns of Japanese polar questions, and to split the category of Japanese ONPQs into such with high and unaccented, and such with low and accented negation. We also argued that, given a context in which the negative alternative is more salient, it is plausible that in Japanese (as it is possibly the case in English), INPQs are essentially the mirror image of PPQs, their differences in terms of bias patterns not arising in all contexts.

Three possible positions for negation in Japanese polar questions were identified: within TP (main verb negation), taking scope over FocP (accented negation) and taking scope over ForceP / assertion (unaccented negation). We proposed that the particle *no* attaches to force resulting in REQUEST speech-acts, and that denegation-requests arise from high outer negation.

We have attempted to derive bias in Japanese polar questions from their properties as summarized above, arguing that C-bias arises from REQUEST speech-acts selecting accented negation or non-negated propositions, while (obligatory) S-bias arises from outer negation in the form of speech-act negation and and focus negation, while (circumstantial) S-bias can arise from propositional negation in REQUEST speech-acts. The proposed properties of Japanese polar questions are summarized in (34).

(34)

	NEG	Context:				
	accented	*neutral*	*positive*	*negative*	S-bias	REQ
PPQ	–	✓	#	#	no	no
PPQ-*no*	–	#	✓	#	no	yes
ONPQ(-*no*)	no	✓	✓	#	yes	yes
ONPQ-*no*	yes	#	#	✓	yes	yes
INPQ	yes	∼	#	✓	no	no
INPQ-*no*	yes	#	#	✓	yes*	yes

*circumstantial S-bias

Open questions remain, among others, the contribution of *no* to ONPQs with unaccented negation, and why English, but not Japanese ONPQs, are good in positive contexts. Also, a more detailed investigation of the claim that INPQs are the mirror image of PPQs, which predicts a preference for *no*-attachment to INPQs in negatively biased contexts and circumstantial S-bias from INPQs, and of the claim that there are two different sources of C-bias in Japanse ONPQs, possibly in comparison with English polar questions, seems necessary in order to confirm whether they are tenable. In addition to this, an investigation of the interaction of polar question and their bias pattern with the question and disjunctive particle *ka* (as well as other sentence-final elements) and its implications for analyzing structure and interpretation of Japanese polar questions has potential to deepen our understanding of them.

Bibliography

[1] Büring, D., Gunlogson, C.: Aren't Positive and Negative Polar Questions the Same? (2000), ms. UCSC/UCLA, semanticsarchive.com

[2] Davis, C.: Constraining Interpretation: Sentence Final Particles in Japanese. Ph.D. thesis, University of Massachusets - Amherst (2011)

[3] Gunlogson, C.: True to Form: Rising and Falling Declaratives as Questions in English. Ph.D. thesis, UCSC (2003)

[4] Hara, Y.: Grammar of Knowledge Representation: Japanese Discourse Items at Interfaces. Ph.D. thesis, University of Delaware (2006)

[5] Hiraiwa, K., Ishihara, S.: Missing Links: Cleft, Sluicing, and "No da" Construction in Japanese. MIT working papers in linguistics 43, 35–54 (2002)

[6] Hwang, H.K., Ito, S.: Correlations between prosody and epistemic bias in negative polar interrogatives in japanese. pp. 925–928 (2014)

[7] Ito, S., Oshima, D.Y.: On two Varieties of Negative Polar Interrogatives in Japanese. Japanese/Korean Linguistics 23, forthcoming (2014)

[8] Krifka, M.: Negated polarity questions as denegations of assertions. In: Lee, C., Kiefer, F., Krifka, M. (eds.) Contrastiveness in Information Structure, Alternatives and Scalar Implicatures. Springer, Berlin (2015)

[9] Ladd, D.R.: A First Look at the Semantics and Pragmatics of Negative Questions and Tag Questions. In: Proceedings of Chicago Linguistic Society. vol. 17, pp. 164–171 (1981)

[10] Noda, H.: "No(da)" no kinoo [The functions of *no*(*da*). Kuroshio Shuppan, Tokyo (1997)

[11] Rizzi, L.: The fine structure of the left periphery. Elements of Grammar: Handbooks in Generative Syntax 1, 281–337 (1997)

[12] Romero, M., Han, C.H.: On negative *Yes/No*-questions. Linguistics and Philosophy 27, 609–658 (2004)

[13] Sudo, Y.: Biased Polar Questions in English and Japanese. In: Gutzmann, D., Gärtner, H.M. (eds.) Beyond Expressives: Explorations in Use-Conditional Meaning, pp. 275–296. Brill, Leiden (2013)

[14] Venditti, J.J.: The j_tobi model of japanese intonation. In: Jun, S.A. (ed.) Prosodic typology: The phonology of intonation and phrasing, pp. 172–200. Oxford University Press, Oxford (2005)

Models for the Displacement Calculus

Oriol Valentín*

Universitat Politècnica de Catalunya,

Abstract. The displacement calculus **D** is a conservative extension of the Lambek calculus $\mathbf{L}_*$ (with empty antecedent allowed in sequents). $\mathbf{L}_*$ can be said to be the logic of concatenation, while **D** can be said to be the logic of concatenation and intercalation. In many senses, it can be claimed that **D** mimics $\mathbf{L}_*$, namely that the proof theory, generative capacity and complexity of the former calculus are natural extensions of the latter calculus. In this paper, we strengthen this claim. We present the appropriate classes of models for **D** and prove its completeness results, and strikingly, we see that these results and proofs are natural extensions of the corresponding ones for $\mathbf{L}_*$.

1 Introduction

The displacement calculus **D** is a quite well-studied extension of the Lambek calculus (with empty antecedent allowed in sequents) $\mathbf{L}_*$. In many papers (see [6], [9] and [8]), **D** has proved to provide elegant accounts of a variety of linguistic phenomena of English, and of Dutch, namely a processing interpretation of the so-called Dutch cross-serial dependencies.

The hypersequent calculus $\mathbf{hD}$[1] is a pure sequent calculus free of structural rules, which subsumes the sequent calculus for $\mathbf{L}_*$. The Cut elimination algorithm for $\mathbf{hD}$ provided in [9] mimics the one of Lambek's syntactic calculus (with some minor differences concerning the possibility of empty antecedents). Like $\mathbf{L}_*$, **D** enjoys some nice properties such as the subformula property, decidablity, the finite reading property and the focalisation property ([4]).

Like $\mathbf{L}_*$, **D** is known to be NP-complete [3]. Concerning the (weak) generative capacity, **D** recognises the class of well-nested multiple context-free languages ([10]). In this sense, this result on generative capacity generalises the result that states that $\mathbf{L}_*$ recognises the class of context-free languages. One point of divergence in terms of generative capacity is that **D** recognises the class of the permutation closure of context-free languages ([7]). Finally, it is important to note that a Pentus-like upper bound theorem for **D** is not known.

In this paper we present natural classes of models for **D**. Several strong completeness results are proved, in particular strong completeness w.r.t. the class of residuated displacement algebras (a natural extension of residuated monoids).

* Research partially supported by SGR2014-890 (MACDA) of the Generalitat de Catalunya, and MINECO project APCOM (TIN2014-57226-P).

[1] Not to be confused with the hypersequents of Avron ([1]).

Powerset frames for $\mathbf{L}_*$ are considered of interest from the linguistic point of view. Here, again powerset residuated displacement algebras over displacement algebras are given, which generalise the powerset residuated monoids over monoids as well as free monoids. Strong completeness result for the so-called implicative fragment of **D**, which is very relevant linguistically, is proved in the style of Buszkowski ([2]). Moreover, full completeness with respect powerset residuated displacement algebras over displacement algebras is given following again Buszkowski's methods.

The structure of the paper is as follows. In Section 2 we present the basic proof-theoretic tools (useful for the construction of canonical models) for the study of **D** from a semantic point of view. In Section 3 we provide the proof of strong completeness of the implicative fragment w.r.t. L-models. In Section 4, the proof of strong completeness of full **D** w.r.t. powerset residuated displacement algebras over displacement algebras is outlined. Finally, we conclude in the last section.

2 The Hypersequent Calculus hD and the Categorical calculus cD

D is model-theoretically motivated, and the key to its conception is the class of standard displacement algebras. Some definitions are needed. Let $\mathcal{M} = (|\mathcal{M}|, +, 0, 1)$ be a free monoid where 1 is a distinguished element of the set of generators X of $\mathcal{M}$. We call such an algebra a *separated monoid*. Given an element $a \in |\mathcal{M}|$, we can associate to it a number, called its *sort* as follows:

$$(1)\quad \begin{array}{ll} s(1) & = 1 \\ s(a) & = 0 \text{ if } a \in X \text{ and } a \neq 1 \\ s(w_1 + w_2) & = s(w_1) + s(w_2) \end{array}$$

This induction is well-defined, for $\mathcal{M}$ is free, and 1 is a (distinguished) generator. The sort function $s(\cdot)$ in a separated monoid simply counts the number of separators an element contains.

Definition 1. *(***Sort Domains**)
Where $\mathcal{M} = (|\mathcal{M}|, +, 0, 1)$ is a separated monoid, the *sort domains* $|\mathcal{M}|_i$ of sort i are defined as follows:

$$|\mathcal{M}|_i = \{a \in |\mathcal{M}| : s(a) = i\}, i \geq 0$$

It is readily seen that for every $i, j \geq 0$, $|\mathcal{M}|_i \cap |\mathcal{M}|_j = \emptyset$ iff $i \neq j$.

Definition 2. *(***Standard Displacement Algebra**)
The *standard displacement algebra* (or standard DA) defined by a separated monoid $(|\mathcal{M}|, +, 0, 1)$ is the $\mathbb{N}$-sorted algebra with the $\mathbb{N}$-sorted signature $\Sigma_D = (+, \{\times_i\}_{i>0}, 0, 1\})$ with sort functionality $((i, j \to i + j)_{i,j\geq 0}, (i, j \to i + j - 1)_{i>0, j\geq 0}, 0, 1)$:

$$(\{|\mathcal{M}|_i\}_{i\geq 0}, +, \{\times_i\}_{i>0}, 0, 1)$$

where:

operation	is such that
$+ : \|\mathcal{M}\|_i \times \|\mathcal{M}\|_j \to \|\mathcal{M}\|_{i+j}$	as in the separated monoid
$\times_k : \|\mathcal{M}\|_i \times \|\mathcal{M}\|_j \to \|\mathcal{M}\|_{i+j-1}$	$\times_k(s,t)$ is the result of replacing the k-th separator in s by t

We will usually write standard DA instead of standard displacement algebra. The sorted types of $\mathbf{D}$, which we will define residuating w.r.t the sorted operations in definition 2, are defined by mutual recursion in Figure 1. We let $\mathbf{Tp} = \bigcup_{i \geq 0} \mathbf{Tp}_i$. A subset B of $|\mathcal{M}|$ is called a *same-sort* subset iff there exists an $i \in \mathbb{N}$ such that for every $a \in B$, $s(a) = i$. $\mathbf{D}$ types are to be interpreted as same-sort subsets of $|\mathcal{M}|$. I.e. every inhabitant of $[\![A]\!]$ has the same sort. The semantic

$\mathbf{Tp}_i ::= \mathbf{Pr}_i$	where $\mathbf{Pr}_i$ is the set of atomic types of sort i
$\mathbf{Tp}_0 ::= I$	Continuous unit
$\mathbf{Tp}_1 ::= J$	Discontinuous unit
$\mathbf{Tp}_{i+j} ::= \mathbf{Tp}_i \bullet \mathbf{Tp}_j$	continuous product
$\mathbf{Tp}_j ::= \mathbf{Tp}_i \backslash \mathbf{Tp}_{i+j}$	under
$\mathbf{Tp}_i ::= \mathbf{Tp}_{i+j} / \mathbf{Tp}_j$	over
$\mathbf{Tp}_{i+j} ::= \mathbf{Tp}_{i+1} \odot_k \mathbf{Tp}_j$	discontinuous product
$\mathbf{Tp}_j ::= \mathbf{Tp}_{i+1} \downarrow_k \mathbf{Tp}_{i+j}$	extract
$\mathbf{Tp}_{i+1} ::= \mathbf{Tp}_{i+j} \uparrow_k \mathbf{Tp}_j$	infix

Fig. 1. The sorted types of $\mathbf{D}$

interpretations of the connectives are shown in Figure 2. This interpretation is called the *standard interpretation*. We observe that for any type $A \in \mathbf{Tp}$, the interpretation of A, i.e. $[\![A]\!]$, is contained in $M_{s(A)}$, where the sort map $s(\cdot)$ for the set $\mathbf{Tp}$, is such that ():

$$
\begin{array}{lll}
 & s(p) & = i \qquad \text{for p} \in \mathbf{Pr}_i \\
 & s(I) & = 0 \\
 & s(J) & = 1 \\
(2) & s(A \bullet B) & = s(A) + s(B) \\
 & s(B/A) = s(A \backslash B) & = s(B) - s(A) \\
 & s(A \odot_k B) & = s(A) + s(B) - 1 \\
 & s(A \downarrow_k B) = s(B \uparrow_k A) & = s(B) - s(A) + 1
\end{array}
$$

$$
\begin{array}{rll}
[\![p]\!] \subseteq |\mathcal{M}|_i \text{ for } i \geq 0 & & p \in \mathbf{Pr}_i \\
\\
[\![I]\!] = \{0\} & & \text{continuous unit} \\
[\![J]\!] = \{1\} & & \text{discontinuous unit} \\
\\
[\![A \bullet B]\!] = \{s_1 + s_2 |\ s_1 \in [\![A]\!] \ \&\ s_2 \in [\![B]\!]\} & & \text{product} \\
[\![A \backslash C]\!] = \{s_2 |\ \forall s_1 \in [\![A]\!], s_1 + s_2 \in [\![C]\!]\} & & \text{under} \\
[\![C / B]\!] = \{s_1 |\ \forall s_2 \in [\![B]\!], s_1 + s_2 \in [\![C]\!]\} & & \text{over} \\
\\
[\![A \odot_k B]\!] = \{\times_k(s_1, s_2) |\ s_1 \in [\![A]\!] \ \&\ s_2 \in [\![B]\!]\} & & k > 0 \text{ discontinuous product} \\
[\![A \downarrow_k C]\!] = \{s_2 |\ \forall s_1 \in [\![A]\!], \times_k(s_1, s_2) \in [\![C]\!]\} & & k > 0 \text{ infix} \\
[\![C \uparrow_k B]\!] = \{s_1 |\ \forall s_2 \in [\![B]\!], \times_k(s_1, s_2) \in [\![C]\!]\} & & k > 0 \text{ extract}
\end{array}
$$

Fig. 2. Standard semantic interpretation of **D** types

2.1 The Hypersequent Calculus hD

We will now consider the *string-based* hypersequent syntax from [5].[2] The reason for using the prefix *hyper* in the term *sequent* is that the data-structure used in hypersequent antecedents is quite nonstandard. *Type segments* are defined as follows

$$
(3)\quad \begin{array}{l} \mathbf{TypSeg}_0 ::= A \text{ for } a \in \mathbf{Tp}_0 \\ \mathbf{TypSeg}_n ::= \lceil A \rceil_i \text{ for } 0 \leq i \leq n := s(A) \end{array}
$$

Type segments of sort 0 are types. But, type segments of sort greater than 0 are no longer types. Strings of type segments can form meaningful logical material like the set of hyperconfigurations, which we now define. The *hyperconfigurations* **HConfig** are defined unambiguously by mutual recursion as follows, where Λ is the empty string and 1 is the metalinguistic separator::

$$
(4)\quad \begin{array}{l} \mathbf{HConfig} ::= \Lambda \\ \mathbf{HConfig} ::= A, \mathbf{HConfig} \text{ for } s(A) = 0 \\ \mathbf{HConfig} ::= 1, \mathbf{HConfig} \\ \mathbf{HConfig} ::= \lceil A \rceil_0, \mathbf{HConfig}, \lceil A \rceil_1, \cdots, \lceil A \rceil_{s(A)-1}, \mathbf{HConfig}, \lceil A \rceil_{s(A)}, \mathbf{HConfig} \\ \qquad\qquad \text{for } s(A) > 0 \end{array}
$$

The semantic interpretation of the last clause from (4) consists of elements $\alpha_0 + \beta_1 + \alpha_1 + \cdots + \alpha_{n-1} + \beta_n + \alpha_n$ where $\alpha_0 + 1 + \alpha_1 + \cdots + \alpha_{n-1} + 1 + \alpha_n \in [\![A]\!]$ and $\beta_1, \cdots, \beta_n$ are the interpretations of the intercalated hyperconfigurations. The syntax in which **HConfig** has been defined is called *string-based hypersequent syntax*. An equivalent syntax for **HConfig** is called *tree-based hypersequent syntax* which was defined in [6], [9]. For proof-search and human readability, the tree-based notation is more convenient than the string-based notation, but for semantic purposes, the string-based notation turns out to be very useful, for

[2] Term which must not be confused with Avron's hypersequents ([1]).

the canonical model construction considered in Section 3 which relies on the set $\mathbf{TypSeg} = \bigcup_{n\geq 0} \mathbf{TypSeg}_n$.

In string-based notation the *figure* $\overrightarrow{A}$ of a type A is defined as follows:

$$(5)\quad \overrightarrow{A} = \begin{cases} A & \text{if } s(A) = 0 \\ \lceil A\rceil_0, 1, \lceil A\rceil_1, \cdots, \lceil A\rceil_{s(A)-1}, 1, \lceil A\rceil_{s(A)} & \text{if } s(A) > 0 \end{cases}$$

The sort of a hyperconfiguration is the number of metalinguistic separators it contains. We have $\mathbf{HConfig} = \bigcup_{i\geq 0} \mathbf{HConfig}_i$, where $\mathbf{HConfig}_i$ is the set of hyperconfigurations of sort i. Where Γ and Φ are hyperconfigurations and the sort of Γ is at least 1, $\Gamma|_k\Phi$ $(k > 0)$ signifies the hyperconfiguration which is the result of replacing the k-th separator in Γ by Φ. Where Γ is a hyperconfiguration of sort i and $\Phi_1, \cdots, \Phi_i$ are hyperconfigurations, the *fold* $\Gamma \otimes \langle \Phi_1, \cdots, \Phi_i\rangle$ is the result of simultaneously replacing the successive separators in Γ by $\Phi_1, \cdots, \Phi_i$ respectively. $\Delta\langle\Gamma\rangle$ abbreviates $\Delta_0(\Gamma \otimes \langle\Delta_1, \cdots, \Delta_i\rangle)$. When a type-occurrence A in a hyperconfiguration is written without vectorial notation, that means that the sort of A is 0. However, when one writes the metanotation for hyperconfigurations $\Delta\langle\overrightarrow{A}\rangle$, this does not mean that the sort of A is necessarily greater than 0.

A *hypersequent* $\Gamma \Rightarrow A$ comprises an antecedent hyperconfiguration in string-based notation of sort i and a succedent type A of sort i. The hypersequent calculus for **D** is as shown in Figure 3. The following lemma is useful for the strong completeness results of section 3:

Lemma 1. *The set of* $\mathbf{HConfig}$ *is a subset of* $(\mathbf{TypSeg} \cup \{1\})^*$. *We have that:*

i) $\mathbf{HConfig}$ *is closed by concatenation and intercalation.*

ii) If $\Delta \in (\mathbf{TypSeg} \cup \{1\})^*$, $\Gamma \in \mathbf{HConfig}$, *and* $\Delta, \Gamma \in \mathbf{HConfig}$, *then* $\Delta \in \mathbf{HConfig}$. *Similarly, if we have* $\Gamma, \Delta \in \mathbf{HConfig}$ *instead of* $\Delta, \Gamma \in \mathbf{HConfig}$. *Finally, If* $\Delta \in (\mathbf{TypSeg} \cup \{1\})^*$, $\Gamma \in \mathbf{HConfig}$, *and* $\Delta|_i\Gamma \in \mathbf{HConfig}$, *then* $\Delta \in \mathbf{HConfig}$.

2.2 D and its Categorical Presentation cD

In [11] it is proved that the identities (or equations) true of standard DAs has as equational theory the so-called class of (general) displacement algebras (DA) (see Figure 4).

The categorical calculus **cD** for **D** is as follows:

$$(6)\quad \begin{array}{l} A\bullet B \rightarrow \text{ iff } A \rightarrow C/B \text{ iff } B \rightarrow A\backslash C \\ A \odot_i B \rightarrow \text{ iff } A \rightarrow C\uparrow_i B \text{ iff } B \rightarrow A\downarrow_i C \\ \text{We add as postulates the ones corresponding} \\ \text{to the set of equations defining } \mathcal{DA} \end{array}$$

The set of postulates of **cD** would be in the case of the categorical calculus with unit for $\mathbf{L}_*$, the postulates corresponding to the equations defining the class of monoids. Let $\mathcal{RD}$ be the class of residuated DAs. Again, in [11], it is proved the following embedding translation:

$\overrightarrow{A} \Rightarrow A$ if A is primitive

$$\frac{\Delta\langle\Lambda\rangle \Rightarrow A}{\Delta\langle I\rangle \Rightarrow A}\, IL \qquad \frac{}{\Lambda \Rightarrow I}\, IR$$

$$\frac{\Delta\langle 1\rangle \Rightarrow A}{\Delta\langle \overrightarrow{J}\rangle \Rightarrow A}\, JL \qquad \frac{}{1 \Rightarrow J}\, JR$$

$$\frac{\Gamma \Rightarrow A \qquad \Delta\langle \overrightarrow{B}\rangle \Rightarrow C}{\Delta\langle \overrightarrow{B/A}, \Gamma\rangle \Rightarrow C}\, /L \qquad \frac{\Delta, \overrightarrow{A} \Rightarrow B}{\Delta \Rightarrow B/A}\, /R$$

$$\frac{\Gamma \Rightarrow A \qquad \Delta\langle \overrightarrow{B}\rangle \Rightarrow C}{\Delta\langle \Gamma, \overrightarrow{A\backslash B}\rangle \Rightarrow C}\, \backslash L \qquad \frac{\overrightarrow{A}, \Delta \Rightarrow B}{\Delta \Rightarrow A\backslash B}\, \backslash R$$

$$\frac{\Delta\langle \overrightarrow{A}, \overrightarrow{B}\rangle \Rightarrow C}{\Delta\langle \overrightarrow{A \bullet B}\rangle \Rightarrow C}\, \bullet L \qquad \frac{\Delta \Rightarrow A \qquad \Gamma \Rightarrow B}{\Delta, \Gamma \Rightarrow A \bullet B}\, \bullet R$$

$$\frac{\Gamma \Rightarrow A \qquad \Delta\langle \overrightarrow{B}\rangle \Rightarrow C}{\Delta\langle \overrightarrow{B\uparrow_i A}|_i \Gamma\rangle \Rightarrow C}\, \uparrow_i L \qquad \frac{\Delta|_i \overrightarrow{A} \Rightarrow B}{\Delta \Rightarrow B\uparrow_i A}\, \uparrow_i R$$

$$\frac{\Gamma \Rightarrow A \qquad \Delta\langle \overrightarrow{B}\rangle \Rightarrow C}{\Delta\langle \Gamma|_i \overrightarrow{A\downarrow_i B}\rangle \Rightarrow C}\, \downarrow_i L \qquad \frac{\overrightarrow{A}|_i \Delta \Rightarrow B}{\Delta \Rightarrow A\downarrow_i B}\, \downarrow_i R$$

$$\frac{\Delta\langle \overrightarrow{A}|_i \overrightarrow{B}\rangle \Rightarrow C}{\Delta\langle \overrightarrow{A \odot_i B}\rangle \Rightarrow C}\, \odot_i L \qquad \frac{\Delta \Rightarrow A \qquad \Gamma \Rightarrow B}{\Delta|_i \Gamma \Rightarrow A \odot_i B}\, \odot_i R$$

Fig. 3. Hypersequent Calculus for D

(7) For any type $A, B \in$, $\mathbf{cD} \vdash A \to B$ iff $\overrightarrow{A} \Rightarrow B$

We can define the well-known Lindenbaum-Tarski construction to see that **cD** is strongly complete w.r.t. $\mathcal{RD}$. The canonical model $(\mathcal{L}, v)$ where $\mathcal{L}$ is $(\mathbf{Tp}/\theta, \circ, (\circ_i)_{i>0}, \backslash\backslash, //, (\downarrow\downarrow_i)_{i>0}(\uparrow\uparrow_i)_{i>0}; \leq)$. θ is the equivalence relation on **Tp** defined as follows: $A\,\theta B$ iff $R \vdash_{\mathbf{cD}} A \to B$ and $R \vdash_{\mathbf{cD}} B \to A$, where R is a set of non-logical axioms. Using the classical tonicity properties for the connectives of **Tp**, one proves that θ is a congruence. Where A is a type, $\overline{A}$ is an element of $\mathbf{Tp}/\theta$ modulo θ. We define $\overline{A} \leq \overline{B}$ iff $R \vdash_{\mathbf{cD}} A \to B$. We define the valuation v as $v(p) = \overline{p}$ (p is a primitive type). We have that for every type A, $[\![A]\!]_v^{\mathcal{L}} = \overline{A}$. Finally, one has that $(\mathcal{L}, v) \models A \to B$ iff $R \vdash_{\mathbf{cD}} A \to B$. From this, we infer the following theorem:

Theorem 1. cD *is strongly complete w.r.t.* $\mathcal{RD}$.

Using the embedding translation (7), we see that **hD** is strongly complete w.r.t. $\mathcal{RD}$. Since $\mathcal{DA}$ is an equational class (see Figure 4), it is closed by subalgebras,

Continuous associativity

$x + (y + z) \approx (x + y) + z$

Discontinuous associativity

$x \times_i (y \times_j z) \approx (x \times_i y) \times_{i+j-1} z$
$(x \times_i y) \times_j z \approx x \times_i (y \times_{j-i+1} z)$ if $i \leq j \leq 1 + s(y) - 1$

Mixed permutation

$(x \times_i y) \times_j z \approx (x \times_{j-\ S(y)\ +1} z) \times_i y$ if $j > i + s(y) - 1$
$(x \times_i z) \times_j y \approx (x \times_j y) \times_{i+S(y)-1} z$ if $j < i$

Mixed associativity

$(x + y) \times_i z \approx (x \times_i y) + z$ if $1 \leq i \leq s(x)$
$(x + y) \times_i z \approx x + (y \times_{i-s(x)} z)$ if $x + 1 \leq i \leq s(x) + s(y)$

Continuous unit and discontinuous unit

$0 + x \approx x \approx x + 0$ and $1 \times_1 x \approx x \approx x \times_i 1$

Fig. 4. Equational theory for $\mathcal{DA}$

direct products and homomorphic images. We have other interesting examples of DAs, namely the *powerset DA over* $\mathcal{A} = (|\mathcal{A}|, +, \{\times_i\}_{i>0}, 0, 1)$, which we denote $\mathcal{P}(A)$. We have:

(8) $\mathcal{P}(A) = (|\mathcal{P}(A)|, \circ, \{\circ_i\}_{i>0}, \mathbb{I}, \mathbb{J})$

A subset B of $|\mathcal{A}|$ is called a *same-sort* subset iff:

(9) There exists an $i \in \mathbb{N}$ such that for every $a \in B$, $s(a) = i$

The notation of the carrier set of $\mathcal{P}(A)$ presupposes that its members are same-sort subsets. Where A, B and C denote same-sort subsets of $|\mathcal{A}|$, the operations $\mathbb{I}$, $\mathbb{J}$, $\circ$ and $\circ_i$ are defined as follows:

$$(10)\quad \begin{array}{ll} \mathbb{I} & \triangleq \{0\} \\ \mathbb{J} & \triangleq \{1\} \\ A \circ B & \triangleq \{a + b : a \in A \text{ and } b \in B\} \\ A \circ_i B & \triangleq \{a \times_i b : a \in A \text{ and } b \in B\} \end{array}$$

It is readily seen that for every $\mathcal{A}$, $\mathcal{P}(A)$ is in fact a DA. A residuated powerset displacement algebra over a displacement algebra $\mathcal{P}(A)$ is the following:

(11) $\mathcal{P}(A) = (|\mathcal{P}(A)|, \circ, \backslash\backslash, //, \{\circ_i\}_{i>0}, , \{\uparrow\uparrow_i\}_{i>0}, \{\downarrow\downarrow_i\}_{i>0}, \mathbb{I}, \mathbb{J}; \subseteq)$

$\backslash\backslash$, $//$, $\uparrow\uparrow_i$ and $\downarrow\downarrow_i$ are defined as follows:

$$
\begin{array}{l}
A\backslash\backslash B \triangleq \{d: \text{ for every } a \in A,\ a + d \in B\} \\
A//B \triangleq \{d: \text{ for every } a \in A,\ a + d \in B\} \\
B\uparrow\uparrow_i B \triangleq \{d: \text{ for every } a \in A,\ d \times_i b \in B\} \\
A\downarrow\downarrow_i B \triangleq \{d: \text{ for every } a \in A,\ a \times_i d \in B\}
\end{array} \tag{12}
$$

The class of powerset residuated DAs over a DA is denoted $\mathcal{PRDD}$. The class of powerset residuated DAs over a standard DA is denoted $\mathcal{PRSD}$. Finally, the subclass of powerset residuated algebras over finitely-generated standard DA, is denoted $\mathcal{PRSD}_{fg}$. Models over residuated DAs of this subclass, are called L-models. Every standard DA $\mathcal{A}$ has two remarkable properties, namely the property that sort domains $|\mathcal{A}|_i$ (for $i > 0$) can be defined in terms of $|\mathcal{A}|_0$, and the property that every element a of a sort domain $|\mathcal{A}|_i$ is decomposed uniquely around the separator 1:

$$
\begin{array}{ll}
\text{a) For } i > 0,\ |\mathcal{A}|_i = \underbrace{|\mathcal{A}|_0 \circ \{1\} \cdots \{1\} \circ |\mathcal{A}|_0}_{(i-1)\ 1's} & (1) \\
\text{b) For } i > 0, \text{ If } a_0 + 1 + \cdots + 1 + a_i = b_0 + 1 + \cdots + 1 + b_i \text{ then} & \\
a_k = b_k \text{ for } 0 \leq k \leq i &
\end{array} \tag{13}
$$

We say that the sort domains of $|\mathcal{A}|$ are *separated* by 1. The single Property (13 a) is called *weakly separtion*, and both properties of (13) constitute what we call *strong separation*.

Standard DAs, as suggests its denomination, are effectively general DAs:

Lemma 2. *$\mathcal{SD} \subseteq \mathcal{DA}$.*[3]

Proof. We define a useful notation which will help us to prove the lemma. Where $\mathcal{A} = (|\mathcal{A}|, +, (\times_i)_{i>0}, 0, 1)$ is a standard DA, let a be an arbitrary element of sort $s(a)$. We associate to every $a \in |\mathcal{A}|$ a sequence of elements $a_0, \cdots, a_{s(A)}$. We have the following vectorial notation:

$$
\overrightarrow{a}_i^j = \begin{cases} a_i, \text{ if } i = j \\ \overrightarrow{a}_i^{j-1} + 1 + a_j, \text{ if } j - i > 0 \end{cases} \tag{14}
$$

Since $\mathcal{A}$ is standard DA, the a_i associated to a given $\overrightarrow{a}$ are unique (by freeness of the underlying monoid). We have that $a = \overrightarrow{a}_0^{s(A)}$, and we write $\overrightarrow{a}$ in place of $\overrightarrow{a}_0^{s(A)}$. Consider arbitrary elements of $|\mathcal{A}|$, $\overrightarrow{a}$, $\overrightarrow{b}$ and $\overrightarrow{c}$:

- Continuous associativity is obvious.

- Discontinuous associativity. Let i, j be such that $i \leq j \leq i + s(\overrightarrow{a}) - 1$:

$$
\begin{array}{l}
\overrightarrow{b} \times_j \overrightarrow{c} = \overrightarrow{b}_0^{i-1} + \overrightarrow{c} + \overrightarrow{b}_i^{s(b)}, \text{ therefore:} \\
\overrightarrow{a} \times_i (\overrightarrow{b} \times_j \overrightarrow{c}) = \boxed{\overrightarrow{a}_0^{i-1} + \overrightarrow{b}_0^{j-1} + \overrightarrow{c} + \overrightarrow{b}_j^{s(b)} + \overrightarrow{a}_i^{s(a)}}
\end{array} \tag{15}
$$

[3] Later we see that the inclusion is proper, i.e. $\mathcal{SD} \subsetneq \mathcal{DA}$

On the other hand, we have that:

$$\overrightarrow{a} \times_i \overrightarrow{b} = \overrightarrow{a}_0^{i-1} + \overrightarrow{b} + \overrightarrow{a}_i^{s(\overrightarrow{a})} = \overrightarrow{a}_0^{i-1} + \overrightarrow{b}_0^{j-1} + \underbrace{1}_{(i+j-1)\text{-th } separator} + \overrightarrow{b}_j^{s(\overrightarrow{b})} + \overrightarrow{a}_i^{s(\overrightarrow{a})}$$

It follows that:

(16) $(\overrightarrow{a} \times_i \overrightarrow{b}) \times_{i+j-1} \overrightarrow{c} = \boxed{\overrightarrow{a}_0^{i-1} + \overrightarrow{b}_0^{j-1} + \overrightarrow{c} + \overrightarrow{b}_j^{s(\overrightarrow{b})} + \overrightarrow{a}_i^{s(\overrightarrow{a})}}$

By comparing the rhs of (15) and (16), we have therefore:

$$\overrightarrow{a} \times_i (\overrightarrow{b} \times_j \overrightarrow{c}) = (\overrightarrow{a} \times_i \overrightarrow{b}) \times_{i+j-1} \overrightarrow{c}$$

- Mixed Permutation. Consider $(\overrightarrow{a} \times_i \overrightarrow{b}) \times_j \overrightarrow{c}$ and suppose that $i + s(\overrightarrow{b}) - 1 < j$:

$$\overrightarrow{a} \times_i \overrightarrow{b} = \underbrace{\overrightarrow{a}_0^{i-1} + \overrightarrow{b} + \overrightarrow{a}_i^{j-s(\overrightarrow{b})}}_{j-s(\overrightarrow{b})+s(\overrightarrow{b})-1=j-1 \text{ separators}} + 1 + \overrightarrow{a}_{j-s(\overrightarrow{b}+1)}^{s(\overrightarrow{a})}$$

It follows that:

(17) $(\overrightarrow{a} \times_i \overrightarrow{b}) \times_j \overrightarrow{c} = \boxed{\overrightarrow{a}_0^{i-1} + \overrightarrow{b} + \overrightarrow{a}_i^{j-s(\overrightarrow{b})} + \overrightarrow{c} + \overrightarrow{a}_{j-s(\overrightarrow{b})+1}^{s(\overrightarrow{a})}}$

Since $i + s(\overrightarrow{b}) - 1 < j$, then $i < j - s(\overrightarrow{b}) + 1$. Then we have that:

$$\overrightarrow{a} \times_{j-s(\overrightarrow{b})+1} \overrightarrow{c} = \overrightarrow{a}_0^{i-1} + 1 + \overrightarrow{a}_i^{j-s(\overrightarrow{b})} + \overrightarrow{c} + \overrightarrow{a}_{j-s(\overrightarrow{b})+1}^{s(\overrightarrow{a})}$$

It follows that:

(18) $(\overrightarrow{a} \times_{j-s(\overrightarrow{b})+1} \overrightarrow{c}) \times_i \overrightarrow{b} = \boxed{\overrightarrow{a}_0^{i-1} + \overrightarrow{b} + \overrightarrow{a}_i^{j-s(\overrightarrow{b})} + \overrightarrow{c} + \overrightarrow{a}_{j-s(\overrightarrow{b})+1}^{s(\overrightarrow{a})}}$

By comparing the rhs of (17) and (18), we have therefore:

$$(\overrightarrow{a} \times_i \overrightarrow{b}) \times_j \overrightarrow{c} = (\overrightarrow{a} \times_{j-s(\overrightarrow{b})+1} \overrightarrow{c}) \times_i \overrightarrow{b}$$

- Mixed associativity. There are two cases: $i \leq s(\overrightarrow{a})$ or $i > s(\overrightarrow{a})$. Considering the first one, this is true for:

$$(\overrightarrow{a} + \overrightarrow{b}) \times \overrightarrow{c} = (\overrightarrow{a}_0^{i-1} + 1 + \overrightarrow{a}_i^{s(\overrightarrow{a})}) \times_i \overrightarrow{c} = \overrightarrow{a}_0^{i-1} + \overrightarrow{c} + \overrightarrow{a}_i^{s(\overrightarrow{a})} + \overrightarrow{b} = (\overrightarrow{a} \times_i \overrightarrow{c}) + \overrightarrow{b}$$

The other case corresponding to $i > s(\overrightarrow{a})$ is completely similar.

- The case corresponding to the units is completely trivial.

□

2.3 Some special DAs

The standard DA $\mathcal{S}$, induced by the separated monoid with generator set $\mathbf{TypSeg}_* \cup \{1\}$, plays an inportant role. The interpretation of the signature Σ_D in $|\mathcal{S}|$ is:

(19) $\mathcal{S} = ((\mathbf{TypSeg}_* \cup \{1\})^*, (,), \{|_i\}_{i>0}, \Lambda, 1)$

We have seen in section 2 that **HConfig** is closed by concatenation $(,)$ and intercalation $|_i$, $i > 0$, i.e. $\mathcal{C} = (\mathbf{HConfig}, (,), (|_i)_{i>0}, \Lambda, 1)$ is a Σ_D-algebra.[4] Since $\mathcal{DA}$ is an equational class, it is closed by (Σ_D)-subalgebras. Since $\mathcal{C}$ is a subalgebra of $\mathcal{S}$, hence $\mathcal{C}$ is a (general) DA, concretely a nonstandard DA. To see that $\mathcal{C}$ cannot be standard we notice that the sort domains of $\mathcal{C}$ are not separated by $\{1\}$. Recall that $|\mathcal{C}| = \bigcup_{i\geq 0} \mathbf{HConfig}_i$ ($|\mathcal{C}|_i = \mathbf{HConfig}_i$, for every $i \geq 0$). We have that:

(20) For $i > 0$, $|\mathcal{C}|_i \neq \underbrace{\mathbf{HConfig}_0 \circ \cdots \circ \mathbf{HConfig}_0}_{i \text{ times}}$

For, for example let us take $\overrightarrow{p\uparrow_1 p} = \lceil p\uparrow_1 p\rceil_0, 1, \lceil p\uparrow_1 p\rceil_1$, where $p \in \mathbf{Pr}$. Type $p\uparrow_1 p$ has sort 1, but clearly neither $\lceil p\uparrow_1 p\rceil_0$ nor $\lceil p\uparrow_1 p\rceil_1$ are members of $\mathbf{HConfig}_0$. In fact, we have the proper inclusion:

(21) For $i > 0$, $\underbrace{\mathbf{HConfig}_0 \circ \cdots \circ \mathbf{HConfig}_0}_{i \text{ times}} \subsetneq |\mathcal{C}|_i$

It follows that the class of standard DAs is a proper subclass of the class of general DAs:

(22) $\mathcal{SD} \subsetneq \mathcal{DA}$

The Lindenbaum algebra $\mathcal{L}$ defined in the previous Subsection is a nonstandard DA, because again, the sort domains are not separated by $\{1\}$. The initial Σ_D-algebra $\mathcal{N}$ in $\mathcal{DA}$ is the standard displacement algebra induced by the singleton generator set $\{1\}$, where $\mathcal{N} = (\mathbb{N}, +^{\mathcal{N}}, \{\times_i^{\mathcal{N}}\}_{i>0}, 0^{\mathcal{N}}, 1^{\mathcal{N}})$. The elements of the signature are interpreted as follows:

(23)
$$\begin{array}{ll} 0^{\mathcal{N}} & \triangleq 0 \\ 1^{\mathcal{N}} & \triangleq 1 \\ +^{\mathcal{N}}(n,m) & \triangleq n+m, \text{ where } n,m \in \mathbb{N} \\ \text{Where } i>0,\ \times_i^{\mathcal{N}}(n,m) & \triangleq n+m-1, \text{ where } n>0 \text{ and } m \geq 0 \end{array}$$

[4] Observe that the sort functionalities of $(,)$ and $|_i$ $(i > 0)$ are respectively $(i, j \to i + j)_{i,j\geq 0}$, and $(i, j \to i + j - 1)_{i>0, j\geq 0}$, where $s(0) = 0,$ and $, s(1) = 1$.

Synthetic Connectives and the Implicative fragment

From a logical point of view, synthetic connectives abbreviate derivations mainly in sequent systems. They form new connectives with left and right sequent rules. Using a linear logic slogan, synthetic connectives help to eliminate some *bureaucracy* in cut-free proofs and in the (syntactic) Cut elimination algorithms. We consider a set of synthetic connectives which are of linguistic interest (Figure 5 corresponds to their semantic interpretation, Figure 6 and Figure 7 correspond to their hypersequent rules). By these definition it is readily seen that the unary synthetic connectives form residuated pairs, and the binary synthetic connectives form residuated triples. We define what we call *the implicative fragment* of **D** which contains all the continuous and discontinous implicative rules, as well as the synthetic connectives defined here, which are considered implicative in the sense that their semantic interpretations in Figure 5 are defined in terms of implicational subset connectives such as $\backslash\backslash, //, \downarrow\downarrow_i$ and $\uparrow\uparrow_i$. We have the following notations: where $\mathbf{Tp}_*$ is the set of unit-free types, $\mathbf{Tp}_*[\rightarrow]$ is the set of $\mathbf{Tp}_*$-types restricted to the implicative fragment. Similarly, we have $\mathbf{hD}_*[\rightarrow]$, $\mathbf{TypSeg}_*[\rightarrow]$, and $\mathbf{HSeq}_*[\rightarrow]$ (where $\mathbf{HSeq}_*$ is the set of hypersequents over unit-free types).[5]

$$
\begin{array}{lll}
[\![\triangleleft^{-1}A]\!] \triangleq [\![A]\!]//\mathbb{J} & & \text{left projection} \\
[\![\triangleright^{-1}A]\!] \triangleq \mathbb{J}\backslash\backslash[\![A]\!] & & \text{right projection} \\
[\![\breve{}^{i}A]\!] \triangleq [\![A]\!]\uparrow_i\mathbb{I} & & i\text{-th split} \\
B\Uparrow A \triangleq B\uparrow_1 A \,\&\, \cdots \,\&\, B\uparrow_{s(B)-s(A)+1} A & & \text{nondeterministic extract} \\
A\Downarrow B \triangleq A\downarrow_1 B \,\&\, \cdots \,\&\, A\downarrow_{s(B)-s(A)+1} B & & \text{nondeterministic infix}
\end{array}
$$

Fig. 5. Semantic interpretation in standard DAs for the set of synthetic connectives

3 Strong Completeness of the implicative fragment w.r.t. L-models

In this section we prove theorem 3, which states that $\mathbf{D}_*[\rightarrow]$ is strongly complete w.r.t. language models. More concretely, the theorem establishes the strong completeness w.r.t. the hypersequent calculus $\mathbf{HSeq}_*[\rightarrow]$. In order to prove it, we demonstrate first strong completeness of $\mathbf{HSeq}_*[\rightarrow]$ w.r.t. powerset residuated DAs over standard DAs with a countable set of generators. Let $V = \mathbf{TypSeg}_*[\rightarrow] \cup \{1\}$. Clearly, V is countably infinite since $\mathbf{TypSeg}_*[\rightarrow]$ is the countable union $\bigcup_i \mathbf{TypSeg}_*[\rightarrow]_i$ (see (3)), where each $\mathbf{TypSeg}_*[\rightarrow]_i$ is also countably infinite.

[5] The unary connectives are definable in terms of (full) **Tp** (using the units, e.g. $\breve{}^{k}A \triangleq A\uparrow_k I$), but the nondeterministic $\Downarrow$ and $\Uparrow$ are no longer definable only in terms of **Tp**, (see [11]).

$$\frac{\Gamma\langle\overrightarrow{A}\rangle \Rightarrow B}{\Gamma\langle\overrightarrow{\triangleleft^{-1}A},1\rangle \Rightarrow B}\triangleleft^{-1}L \qquad \frac{\Gamma,1 \Rightarrow A}{\Gamma \Rightarrow \triangleleft^{-1}A}\triangleleft^{-1}R$$

$$\frac{\Gamma\langle\overrightarrow{A}\rangle \Rightarrow B}{\Gamma\langle 1,\overrightarrow{\triangleright^{-1}A}\rangle \Rightarrow B}\triangleright^{-1}L \qquad \frac{1,\Gamma \Rightarrow A}{\Gamma \Rightarrow \triangleright^{-1}A}\triangleright^{-1}R$$

$$\frac{\Delta\langle\overrightarrow{B}\rangle \Rightarrow C}{\Delta\langle\overrightarrow{\backsim_i B}|_i\Lambda\rangle \Rightarrow C}\backsim_i L \qquad \frac{\Delta|_i\Lambda \Rightarrow BB}{\Delta \Rightarrow \backsim_i B}\backsim R$$

Fig. 6. Hypersequent rules for synthetic unary implicative connectives

$$\frac{\Delta \Rightarrow A \qquad \Gamma\langle\overrightarrow{B}\rangle \Rightarrow C}{\Gamma\langle\overrightarrow{B\Uparrow A}|_i\Gamma\rangle \Rightarrow C}\Uparrow L \qquad \frac{\Delta|_1\overrightarrow{A} \Rightarrow B \qquad \cdots \qquad \Delta|_d\overrightarrow{A} \Rightarrow B}{\Delta \Rightarrow B\Uparrow A}\Uparrow R$$

$$\frac{\Delta \Rightarrow A \qquad \Gamma\langle\overrightarrow{B}\rangle \Rightarrow C}{\Gamma\langle\Gamma|_i\overrightarrow{A\Downarrow B}\rangle \Rightarrow C}\Downarrow L \qquad \frac{\overrightarrow{A}|_1\Delta \Rightarrow B \qquad \cdots \qquad \overrightarrow{A}|_a\Delta \Rightarrow B}{\Delta \Rightarrow A\Downarrow B}\Downarrow R$$

Fig. 7. Hypersequent calculus rules for nondeterministic synthetic connectives

Let us consider the standard DA $\mathcal{S}$ (from (19)), induced by the (countably) infinite set of generators V:

$$\mathcal{S} = (V^*, (,), \{|_i\}_{i>0}, \Lambda, 1)$$

In order to prove completeness of $\mathbf{HSeq}^*[\rightarrow]$ w.r.t. the class of powerset DAs over (countable) standard DAs ($\mathcal{PRSD}$), we define some useful notation:

Definition 3. *For any type $C \in \mathbf{Tp}_*[\rightarrow]$, and a set of sequents R of* **HSeq**:

$$[C]_R \triangleq \{\Delta : \Delta \in \mathbf{HConfig} \text{ and } R \vdash \Delta \Rightarrow C\}$$

In practice, when the set R of hypersequents is clear by the context, we simply write $[C]$ instead of $[C]_R$. Let us fix some set of hypersequents R:

Lemma 3. (**Truth Lemma**)
Let $\mathcal{P}(S)$ be the powerset residuated DA over the standard DA $\mathcal{S}$ from (19) . Let v be the following valuation on the powerset $\mathcal{P}(S)$:

$$\text{For every } p \in \mathbf{Pr},\ v(p) \triangleq [p]_R$$

Let $\mathcal{M} = (\mathcal{P}(S), v)$ be called as usual the canonical model. The following equality holds:

$$\text{For every } C \in \mathbf{Tp}_*[\rightarrow],\ [\![C]\!]_v^{\mathcal{P}(\mathcal{A})} = [C]$$

Proof. We proceed by induction on the structure of type C. Let us write $[\![\cdot]\!]$ instead of $[\![C]\!]_v^{\mathcal{P}(\mathcal{A})}$, and $[\cdot]$ instead of $[\cdot]_R$. We will say that an element $\Delta \in [\![A]\!]$ is *correct* iff $\Delta \in \mathbf{HConfig}$.

- C is primitive. True by definition.

- $C = B{\uparrow_i}A$. Let us see:
$$[B{\uparrow_i}A] \subseteq [\![B{\uparrow_i}A]\!]$$
Let Δ be such that $R \vdash \Delta \Rightarrow B{\uparrow_i}A$. Let $\Gamma_A \in [\![A]\!]$. By induction hypothesis (i.h.), $[\![A]\!] = [A]$. Hence, $R \vdash \Gamma_A \Rightarrow A$ We have:
$$\frac{\Delta \Rightarrow B{\uparrow_i}A \qquad \overrightarrow{B{\uparrow_i}A}|_i\Gamma_A \Rightarrow B}{\Delta|_i\Gamma_A \Rightarrow B}\,Cut$$
By (i.h.), $[\![B]\!] = [B]$. It follows that $\Delta|_i\Gamma_A \in [\![B]\!]$, hence $\Delta \in [\![B{\uparrow_i}A]\!]$. Whence, $[B{\uparrow_i}A] \subseteq [\![B{\uparrow_i}A]\!]$.
Conversely, let us see:
$$[\![B{\uparrow_i}A]\!] \subseteq [B{\uparrow_i}A]$$
Let $\Delta \in [\![B{\uparrow_i}A]\!]$. By i.h. $[\![A]\!] = [A]$. For any type A, we have eta-expansion, i.e. $\overrightarrow{A} \Rightarrow A$. Hence, $\overrightarrow{A} \in [\![A]\!]$. We have that $\Delta|_i\overrightarrow{A} \in [\![B]\!]$. By i.h., $\Delta|_i\overrightarrow{A} \Rightarrow B$. Since $\overrightarrow{A}$ is correct, and by i.h. $\Delta|_i\overrightarrow{A}$ is correct, by lemma 1, Δ is correct. By applying the $\uparrow_i$ right rule to the provable hypersequent $\Delta|_i\overrightarrow{A} \Rightarrow B$ we get:
$$\Delta \Rightarrow B{\uparrow_i}A$$
This ends the case of $B{\uparrow_i}A$.

- $C = A{\downarrow_i}B$. Completely similar to case $B{\uparrow_i}A$.

- $C = B/A$. Let us see:
$$[B/A] \subseteq [\![B/A]\!]$$
Let Δ be such that $R \vdash \Delta \Rightarrow B/A$. Let $\Gamma_A \in [\![A]\!]$. By induction hypothesis (i.h.), $[\![A]\!] = [A]$. Hence, $R \vdash \Gamma_A \Rightarrow A$ We have:
$$\frac{\Delta \Rightarrow B{\uparrow_i}A \qquad \overrightarrow{B/A}, \Gamma_A \Rightarrow B}{\Delta, \Gamma_A \Rightarrow B}\,Cut$$
By (i.h.), $[\![B]\!] = [B]$. It follows that $\Delta, \Gamma_A \in [\![B]\!]$. Whence, $[B/A] \subseteq [\![B/A]\!]$.
Conversely, let us see:
$$[\![B/A]\!] \subseteq [B/A]$$
Let $\Delta \in [\![B/A]\!]$. By i.h. $[\![A]\!] = [A]$. For any type A, we have eta-expansion, i.e. $\overrightarrow{A} \Rightarrow A$. Hence, $\overrightarrow{A} \in [\![A]\!]$. We have that $\Delta, \overrightarrow{A} \in [\![B]\!]$. By i.h., $\Delta, \overrightarrow{A} \Rightarrow B$.

Since $\overrightarrow{A}$ is correct, and by i.h. $\Delta, \overrightarrow{A}$ is correct, by lemma 1 Δ is correct. By applying the / right rule to the provable hypersequent $\Delta, \overrightarrow{A} \Rightarrow B$ we get:

$$\Delta \Rightarrow B/A$$

This ends the case of B/A.

- $C = A\backslash B$. Completely similar to the case $C = B/A$.

- Nondeterministic connectives. Consider the case $C = B \Uparrow_i A$.

$$[B \Uparrow A] \subseteq [\![B \Uparrow A]\!]$$

Let $\Gamma_A \in [\![A]\!]$. By i.h, $\Gamma_A \Rightarrow A$. Let $\Delta \Rightarrow B \Uparrow A$. By $s(B) - s(A) + 1$ applications of $\Uparrow$ left rule, we have

$$\frac{\Gamma_A \Rightarrow A \qquad \overrightarrow{B} \Rightarrow B, \text{ by eta-expansion}}{\overrightarrow{B \Uparrow A}|_i\Gamma_A \Rightarrow B, \text{ for } i = 1, \cdots, s(B) - s(A) + 1} \Uparrow L$$

By $s(B) - s(A) + 1$ Cut applications with $\Delta \Rightarrow B \Uparrow A$, we get:

$$\Delta|_i\Gamma_A \Rightarrow B$$

Hence, for $i = 1, \cdots s(B)-s(A)+1$, by i.h. $\Delta, \Gamma_A \in [\![B]\!]$. Hence, $\Delta \in [\![B \Uparrow A]\!]$. Conversely, let us see:

$$[\![B \Uparrow A]\!] \subseteq [B \Uparrow A]$$

By i.h, we see that $\overrightarrow{A} \in [\![A]\!]$. Let $\Delta \in [\![B \Uparrow A]\!]$. This means that for every $i = 1, \cdots, s(B) - s(A) + 1$ $\Delta|_i \overrightarrow{A} \in [\![B]\!]$. By i.h., $\Delta|_i \overrightarrow{A} \Rightarrow B$. By a similar reasoning to the deterministic case $C = B\uparrow_i A$, we see that Δ is correct. We have that:

$$\frac{\Delta|_1 \overrightarrow{A} \Rightarrow B \qquad \cdots \qquad \Delta|_{s(B)-s(A)+1} \overrightarrow{A} \Rightarrow B}{\Delta \Rightarrow B \Uparrow A} \Uparrow R$$

- The case $C = A \Downarrow B$ is completely similar to the previous one.
  Let us see the cases corresponding to the unary (implicative) connectives.
- Left projection case: $C = \triangleleft^{-1}A$. Let us see:

$$[\triangleleft^{-1}A] \subseteq [\![\triangleleft^{-1}A]\!]$$

Let $\Delta \in [\triangleleft^{-1}A]$. Hence, $\Delta \Rightarrow \triangleleft^{-1}A$. We have that:

$$\frac{\Delta \Rightarrow \triangleleft^{-1}A \qquad \dfrac{\overrightarrow{A} \Rightarrow A}{\overrightarrow{\triangleleft^{-1}A}, 1 \Rightarrow A}\triangleleft^{-1}L}{\Delta, 1 \Rightarrow A} Cut$$

By i.h., $\Delta, 1 \in [\![A]\!]$. Hence, $\Delta \in [\![\triangleleft^{-1}A]\!]$.
Conversely, let us see:

$$[\![\triangleleft^{-1}A]\!] \subseteq [\triangleleft^{-1}A]$$

Let $\Delta \in [\![\triangleleft^{-1}A]\!]$. By definition, $\Delta, 1 \in [\![A]\!]$. By i.h., $\Delta, 1 \Rightarrow A$, and by lemma 1, Δ is correct. By application of $\triangleleft^{-1}$ right rule, we get:

$$\Delta \Rightarrow \triangleleft^{-1}A$$

This proves the converse.

- Case $C = \triangleright^{-1}A$ is completely similar to the previous one.

- Case $C = {\scriptstyle\check{}}^{k}A$. Let us see:

$$[{\scriptstyle\check{}}^{k}A] \subseteq [\![{\scriptstyle\check{}}^{k}A]\!]$$

Let $\Delta \Rightarrow {\scriptstyle\check{}}^{k}A$. We have that:

$$\dfrac{\Delta \Rightarrow {\scriptstyle\check{}}^{i}A \qquad \dfrac{\overrightarrow{A} \Rightarrow A}{\Longrightarrow {\scriptstyle\check{}}^{i}A|_{k}\Lambda \Rightarrow A}\,{\scriptstyle\check{}}^{k}L}{\Delta|_{k}\Lambda \Rightarrow A}\,Cut$$

By i.h., $\Delta \in [\![{\scriptstyle\check{}}^{k}A]\!]$.
Conversely, let us see that:

$$[\![{\scriptstyle\check{}}^{k}A]\!] \subseteq [{\scriptstyle\check{}}^{k}A]$$

Let $\Delta \in [\![{\scriptstyle\check{}}^{k}A]\!]$. By definition, $\Delta|_{k}\Lambda \in [\![A]\!]$. By i.h. and lemma 1, Δ is correct and $\Delta|_{k}\Lambda \Rightarrow A$. By application of the ${\scriptstyle\check{}}^{k}$ right rule:

$$\Delta \Rightarrow {\scriptstyle\check{}}^{k}A$$

Hence, $\Delta \in [{\scriptstyle\check{}}^{k}A]$.

We have seen all the cases of the so-called implicative fragment. We are done. □

By induction on the structure of **HConfig** (see (4)) one proves the following lemma:

Lemma 4. (**Identity lemma**)
For any $\Delta \in \mathbf{HConfig}$, $\Delta \in [\![\Delta]\!]^{\mathcal{M}}$.

Let $(A_i)_{i=1,\cdots,n}$ be the sequence of type-occurrences in a hyperconfiguration Δ. Let $\Delta\begin{pmatrix}\Gamma_1 \cdots \Gamma_n\\ A_1 \cdots A_n\end{pmatrix}$ be the result of replacing every type-occurrence A_i with Γ_i. Recall that we have fixed a set of hypersequents R. We have the lemma:

Lemma 5. $\mathcal{M} = (\mathcal{P}(S), v) \models R$

Proof. Let $(\Delta \Rightarrow A) \in R$. For every type-occurrence A_i in Δ (we suppose that the sequence of type occurrences in Δ is $(A_i)_{i=1,\cdots,n}$), we have that $[\![A_i]\!]_v^{\mathcal{M}} = [A_i]_R$. For any $\Gamma_i \in [\![A_i]\!]_v^{\mathcal{M}}$, we have by the truth lemma 3 that $R \vdash \Gamma_i \Rightarrow A_i$. Since $(\Delta \Rightarrow A) \in R$, we have then that $R \vdash \Delta \Rightarrow A$. By n applications of the Cut rule with the premises Γ_i we get from $R \vdash \Delta \Rightarrow A$ that $R \vdash \Delta\begin{pmatrix} \Gamma_1 \cdots \Gamma_n \\ A_1 \cdots A_n \end{pmatrix} \Rightarrow A$. We have that $[\![\Delta]\!]_v^{\mathcal{M}} = \{\Delta\begin{pmatrix} \Gamma_1 \cdots \Gamma_n \\ A_1 \cdots A_n \end{pmatrix} : \Gamma_i \in [\![A_i]\!]_v^{\mathcal{M}}\}$. Since, we have $R \vdash \Delta\begin{pmatrix} \Gamma_1 \cdots \Gamma_n \\ A_1 \cdots A_n \end{pmatrix} \Rightarrow A$, again, by the truth lemma, $\Delta\begin{pmatrix} \Gamma_1 \cdots \Gamma_n \\ A_1 \cdots A_n \end{pmatrix} \in [\![A]\!]_v^{\mathcal{M}}$. We have then that $[\![\Delta]\!]_v^{\mathcal{M}} \subseteq [\![A]\!]_v^{\mathcal{M}}$. We are done. □

Theorem 2. $\mathbf{D}_*[\rightarrow]$ *is strongly complete w.r.t. the class* $\mathcal{PRSD}$.

Proof. Suppose $\mathcal{PRSD}(R) \models \Delta \Rightarrow A$. Hence, in particular this is true of the canonical model $\mathcal{M}$. Since $\Delta \in [\![\Delta]\!]^{\mathcal{M}}$, it follows that $\Delta \in [\![A]\!]^{\mathcal{M}}$. By the truth lemma, $[\![A]\!]^{\mathcal{M}} = [A]_R$. Hence, $R \vdash \Delta \Rightarrow A$. We are done. □

Let $\mathcal{A}$ and $\mathcal{B}$ be respectively standard DAs over separated monoids with a countable set of generators $V_1 = (a_i)_{i>0} \cup \{1\}$, and a finitely generated standard DA, concretely a standard DA with a set of three generators $V_2 = \{a, b, 1\}$. We have that $|\mathcal{A}| = V_1^*$, and $|\mathcal{B}| = V_2^*$. Let $\bar{\rho}$ be the following injective mapping from V_1 into V_2^*:

$$(24)\quad \begin{array}{l} \bar{\rho}(1) = 1 \\ \bar{\rho}(a_i) = a + b^i + a \end{array}$$

ρ extends recursively to the morphism of standard DAs $\bar{\rho}$:

(25)

$$\begin{array}{lll} \bar{\rho} : \mathcal{A} & \longrightarrow & \mathcal{B} \\ 0 & \mapsto & 0 \\ 1 & \mapsto & 1 \\ \bar{\rho}(w_1 + w_2) & \mapsto & \bar{\rho}(w_1) + \bar{\rho}(w_2) \\ \bar{\rho}(w_1 \times_i w_2) & \mapsto & \bar{\rho}(w_1) \times_i \bar{\rho}(w_2) \end{array}$$

Since for every $i > 0$, $\bar{\rho}(a_i)$ is prefix-free (and hence $\bar{\rho}$),[6] $\bar{\rho}$ is a monomorphism. $\bar{\rho}$ induces then a monomorphism of standard DAs. Let A, B and C range over subsets of $|\mathcal{A}|$. Since $\bar{\rho}$ is a monomorphism of DAs and the underlying monoids of $\mathcal{A}$ and $\mathcal{B}$ are free, one proves:

$$(26)\quad \begin{array}{llll} \bar{\rho}(A \circ B) = \bar{\rho}(A) \circ \bar{\rho}(B) & & \bar{\rho}(A \circ_i B) = \bar{\rho}(A) \circ_i \bar{\rho}(B) \\ \bar{\rho}(A \backslash\backslash B) = \bar{\rho}(A) \backslash\backslash \bar{\rho}(B) & & \bar{\rho}(B // A) = \bar{\rho}(B) // \bar{\rho}(A) \\ \bar{\rho}(A \downarrow\downarrow_i B) = \bar{\rho}(A) \downarrow\downarrow_i \bar{\rho}(B) & & \bar{\rho}(B \uparrow\uparrow_i A) = \bar{\rho}(B) \uparrow\uparrow_i \bar{\rho}(A) \\ \bar{\rho}(B \uparrow\uparrow_i \mathbb{I}) = \bar{\rho}(B) \uparrow\uparrow_i \mathbb{I} & & \bar{\rho}(A // \mathbb{J}) = \bar{\rho}(A) // \bar{\rho}(\mathbb{J}) \\ \bar{\rho}(\mathbb{J} \backslash\backslash A) = \mathbb{J} \backslash\backslash \bar{\rho}(A) & & \end{array}$$

[6] If w is a non-empty proper prefix of $\bar{\rho}(a_i)$, then $w \notin Im(\bar{\rho})$.

Moreover, one proves also

$$(27)\quad \begin{array}{l} \bar{\rho}(\bigcap_{i=1}^{s(B)-s(A)+1} B \Uparrow_i A) = \bigcap_{i=1}^{s(B)-s(A)+1} \bar{\rho}(B) \Uparrow_i \bar{\rho}(A) \\ \bar{\rho}(\bigcap_{i=1}^{s(B)-s(A)+1} A \Downarrow_i B) = \bigcap_{i=1}^{s(B)-s(A)+1} \bar{\rho}(A) \Downarrow_i \bar{\rho}(A) \end{array}$$

Now, let $(\mathcal{P}(\mathcal{A}), v)$ be the powerset residuated displacement model over the standard DA $\mathcal{A}$. For every $A \in \mathbf{Tp}_*[\rightarrow]$, and $\Delta \in \mathbf{HConfig}$, one has the following facts:

$$(28)\quad [\![A]\!]_v^{\mathcal{P}(\mathcal{A})} = [\![A]\!]_{\bar{\rho}\circ v}^{\mathcal{P}(\mathcal{B})} \text{ and } [\![A]\!]_v^{\mathcal{P}(\mathcal{A})} = [\![A]\!]_{\bar{\rho}\circ v}^{\mathcal{P}(\mathcal{B})}$$

By properties (28), (26), and (27) and the fact that $\bar{\rho}$ is a monomorphism, we have therefore the following equivalence:

$$(29)\quad [\![\Delta]\!]_v^{\mathcal{P}(\mathcal{A})} \subseteq [\![A]\!]_v^{\mathcal{P}(\mathcal{A})} \text{ iff } [\![\Delta]\!]_{\bar{\rho}\circ v}^{\mathcal{P}(\mathcal{B})} \subseteq [\![A]\!]_{\bar{\rho}\circ v}^{\mathcal{P}(\mathcal{B})}$$

Where $\mathcal{K}$ is a subclass of $\mathcal{RD}$, the notation $\mathcal{K}(R) \models \Delta \Rightarrow A$ (where R is a set of hypersequents) means that $\mathcal{K} \models R$ and $\mathcal{K} \models \Delta \Rightarrow A$.

Theorem 3. $\mathbf{D}_*[\rightarrow]$ *is strongly complete w.r.t. L-models.*

Proof. Let R be a set of sequents. By way of contradiction, consider a hypersequent $\Delta \Rightarrow A$ such that $\mathbf{D}_*[\rightarrow] + R \nvdash \Delta \Rightarrow A$ but $\mathcal{PRSD}_{fg}(R) \models \Delta \Rightarrow A$. Since $\mathbf{D}_*[\rightarrow]$ is strongly complete w.r.t. $\mathcal{PRSD}$ (theorem 2), there exists a model $(\mathcal{A}, v) \models R$ but $(\mathcal{A}, v) \not\models \Delta \Rightarrow A$. Let $\bar{\rho}$ the coding morphism from (25). Let $\mathcal{B}$ be the finitely generated standard displacement algebra with 3 generators a, b and 1. Since $\mathcal{PRSD}_{fg}(R) \models \Delta \Rightarrow A$, we have that for every valuation v', $[\![\Delta]\!]_{v'}^{\mathcal{B}} \subseteq [\![A]\!]_{v'}^{\mathcal{B}}$, in particular for the valuation $\bar{\rho} \circ v$. By (29) we have that:

$$[\![\Delta]\!]_v^{\mathcal{A}} \subseteq [\![A]\!]_v^{\mathcal{A}} \text{ iff } [\![\Delta]\!]_{\bar{\rho}\circ v}^{\mathcal{B}} \subseteq [\![A]\!]_{\bar{\rho}\circ v}^{\mathcal{B}}$$

But, by assumption, $[\![\Delta]\!]_v^{\mathcal{A}} \not\subseteq [\![A]\!]_v^{\mathcal{A}}$. Contradiction. □

Corollary 1. $\mathbf{HSeq}^*[\rightarrow]$ *is strongly complete w.r.t. powerset residuated DAs overs standard DAs with* 3 *generators.*

4 Towards Strong Completeness of full D w.r.t. $\mathcal{PRDD}$

We sketch[7] in this section strong completeness of full $\mathbf{D}$ w.r.t. $\mathcal{PRDD}$. We get this result by proving a representation theorem between $\mathcal{RD}$ and $\mathcal{PRDD}$. In order to get this representation theorem we need to consider $\mathbf{D}_*$ (unit-free $\mathbf{D}$), and consequently, $\mathbf{Tp}_*$ (unit-free $\mathbf{Tp}$), and $\mathbf{HConfig}_*$ (unit-free $\mathbf{HConfig}$). This is a step to prove strong completeness for full $\mathbf{D}$ (without restrictions on the units). We give the mutually recursive definition of the set $\mathcal{T}$ of hypertrees, and the set of atomic terms:

[7] We do not have enough space to justify the main claims. But, we believe that this sketch is quite illuminating.

$$
\begin{array}{l}
\Lambda, 1 \in \mathcal{T} \\
\text{If } A \in \mathbf{Tp}_*, \text{ then } A \text{ is an atomic term} \\
\text{If } T \in \mathcal{T}, A, B \in \mathbf{Tp}_*, \text{ then } (T; A\bullet B; \mathbf{f}) \text{ is an atomic term} \\
\text{If } T \in \mathcal{T}, A, B \in \mathbf{Tp}_*, \text{ then } (T; A\bullet B; \mathbf{s}) \text{ is an atomic term} \\
\text{If } T \in \mathcal{T}, A, B \in \mathbf{Tp}_*, \text{ then } (T; A\odot_i B; \mathbf{f}) \text{ is an atomic term} \\
\text{If } T \in \mathcal{T}, A, B \in \mathbf{Tp}_*, \text{ then } (T; A\odot_i B; \mathbf{s}) \text{ is an atomic term} \\
s((T; A\bullet B; \mathbf{f})) = s(A) \text{ and } s((T; A\bullet B; \mathbf{s})) = s(B) \\
s((T; A\odot_i B; \mathbf{f})) = s(A) \text{ and } s((T; A\odot_i B; \mathbf{s})) = s(B) \\
\Lambda \in \mathcal{T}, \text{ and } 1 \in \mathcal{T} \\
\text{If } L \text{ atomic, then } \overrightarrow{L} \triangleq L\{\underbrace{1 : \cdots : 1}_{s(L)\ 1\text{'s}}\} \in \mathcal{T} \\
\text{If } T, S \in \mathcal{T}, \text{ then } T, S \in \mathcal{T} \\
\text{If } T, S \in \mathcal{T}, \text{ then } T|_i S \in \mathcal{T}
\end{array} \tag{30}
$$

Like **HConfig**, $\mathcal{T}$ is sorted, and for every $T \in \mathcal{T}$, $s(T)$ is simply the number of separators T contains. We put $\mathcal{T}_i = \{T : T \in \mathcal{T} \text{ and } s(T) = i\}$ for $i \geq 0$. We have then that $\mathcal{T} = \bigcup_{i\geq 0} \mathcal{T}_i$. Notice that $\mathcal{T}$ includes the set **HConfig**. We define now a notion of reduction $\rhd$ in $\mathcal{T}$. Where $A, B \in \mathbf{Tp}_*$, and T_i, S_j, $(i = 1, \cdots, s(A), \text{ and }, j = 1, \cdots, s(B))$ are hypertreees, we have:

$$
\left\{
\begin{array}{l}
\overrightarrow{(T; A\bullet B; \mathbf{f})}\{T_1 : \cdots : T_{s(A)}\}, \overrightarrow{(T; A\bullet B; \mathbf{s})}\{R_1 : \cdots : R_{s(B)}\} \\
\rhd T \otimes \langle T_1 : \cdots : T_{s(A)} : R_1 : \cdots : R_{s(B)}\rangle
\end{array}
\right. \tag{31}
$$

$$
\left\{
\begin{array}{l}
\overrightarrow{(T; A\odot_i B; \mathbf{f})}\{T_1 : \cdots : \overrightarrow{(T; A\odot_i B; \mathbf{f})}\{R_1 : \cdots : R_{s(B)}\} : \cdots : T_{s(A)}\} \\
\rhd T \otimes \langle T_1 : \cdots : R_1 : \cdots : R_{s(B)} : \cdots : T_{s(A)}\rangle
\end{array}
\right.
$$

By a simple primitive type counting argument, one sees that the transitive closure of $\rhd$ $\rhd^*$, is always terminating, i.e. $\rhd^*$ is strongly normalising. Again, by primitive type counting arguments, one sees that $\rhd^*$ is weakly Church-Rosser, and hence, by Newman's lemma, $\rhd^*$ is Church-Rosser. This allows for every element of $T \in \mathcal{T}$ to define its normal form $irr(T)$. We put $\mathbf{Irr} = irr(\mathcal{T})$. Since $\mathcal{T}$ is sorted, **Irr** is also sorted. We have that $\mathbf{Irr} = \bigcup_{i\geq 0} \mathbf{Irr}_i$, where $\mathbf{Irr}_i = \{T : T \in \mathbf{Irr} \text{ and } s(T) = i\}$. Let us consider the Σ_D-algebra:

$$
\mathbf{Irr} = (\mathbf{Irr}, \tilde{+}, (\tilde{\times}_i)_{i\geq 0}, \Lambda, 1) \tag{32}
$$

Where $\tilde{+}$, and $(\tilde{\times}_i)_{i\geq 0}$, are defined as follows::

$$
\begin{array}{l}
T\tilde{+}S \triangleq irr(T, S) \\
T\tilde{\times}_i S \triangleq irr(T, S)
\end{array} \tag{33}
$$

By Church-Rosser, **Irr** is easily seen to be a (nonstandard) DA. For, given the arbitrary hypertrees T_1, T_2 and T_3, for example discontinuous associativity is proved as follows:

$$
\begin{array}{rl}
T_1\tilde{\times}_i(T_2\tilde{\times}_j T_3) & = irr(T_1|_i irr(T_2|_j T_3)) \\
& = Irr(T_1|_i(T_2|_j T_3)) = Irr((T_1|_i T_2)|_{i+j-1} T_3) \\
& = Irr(Irr(T_1|_i T_2)|_{i+j-1} T_3) \\
& = (T_1\tilde{\times}_i T_2)\tilde{\times}_{i+j-1} T_3
\end{array} \tag{34}
$$

Irr induces the powerset residuated DA over the DA **Irr**, which we denote $\mathcal{P}(\mathbf{Irr})$.

Following Buszkowski's technics on labelled deductive systems ([2]), we can now introduce in Figure 8 a natural deduction system $\mathbf{nD}_*$ for a conservative extension of $\mathbf{D}_*$. R is a given set of $\mathbf{D}_*$-hypersequents. The axiom rule has as

$$\overrightarrow{A}\twoheadrightarrow A \text{ for every } A \in \mathbf{Tp}_*$$

$$\frac{T\twoheadrightarrow B/A \qquad S\twoheadrightarrow A}{T,S\twoheadrightarrow B}\,/E \qquad \frac{T,\overrightarrow{A}\twoheadrightarrow B}{T\twoheadrightarrow B/A}\,/I$$

$$\frac{S\twoheadrightarrow A \qquad T\twoheadrightarrow A\backslash B}{T,S\twoheadrightarrow B}\,\backslash E \qquad \frac{\overrightarrow{A},T\twoheadrightarrow B}{T\twoheadrightarrow A\backslash B}\,\backslash I$$

$$\frac{T\twoheadrightarrow A\bullet B}{\overrightarrow{(T;A\bullet B;\mathbf{f})}\twoheadrightarrow A}\,\bullet E1 \qquad \frac{T\twoheadrightarrow A\bullet B}{\overrightarrow{(T;A\bullet B;\mathbf{s})}\twoheadrightarrow B}\,\bullet E2$$

$$\frac{T\twoheadrightarrow A \qquad S\twoheadrightarrow B}{T,S\twoheadrightarrow A\bullet B}\,\bullet I$$

$$\frac{T\twoheadrightarrow B\uparrow_i A \qquad S\twoheadrightarrow A}{T|_i S\twoheadrightarrow B}\,\uparrow_i E \qquad \frac{T|_i\overrightarrow{A}\twoheadrightarrow B}{T\twoheadrightarrow B\uparrow_i A}\,\uparrow_i I$$

$$\frac{S\twoheadrightarrow A\downarrow_i T\twoheadrightarrow A\backslash B}{S|_i T\twoheadrightarrow B}\,\downarrow_i E \qquad \frac{\overrightarrow{A}|_i T\twoheadrightarrow B}{T\twoheadrightarrow A\downarrow_i B}\,\backslash I$$

$$\frac{T\twoheadrightarrow A\odot iB}{\overrightarrow{(T;A\odot iB;\mathbf{f})}\twoheadrightarrow A}\,\odot iE1 \qquad \frac{T\twoheadrightarrow A\odot iB}{\overrightarrow{(T;A\odot iB;\mathbf{s})}\twoheadrightarrow B}\,\odot iE2$$

$$\frac{T\twoheadrightarrow A \qquad S\twoheadrightarrow B}{T|_i S\twoheadrightarrow A\odot iB}\,\odot iI$$

$$\frac{T\twoheadrightarrow A}{S\twoheadrightarrow A}\,Red \text{ If } \mathrm{T}\rhd^*\mathrm{S}$$

$$\frac{T_i\twoheadrightarrow A_i \text{ where } i=1,\cdots,n}{\Delta\begin{pmatrix}T_1\cdots T_n\\ A_1\cdots A_n\end{pmatrix}\twoheadrightarrow A}\,Axiom_R,\ (\Delta\Rightarrow A)\in R$$

Fig. 8. $\mathbf{nD}_*$ rules

premises $T_i\twoheadrightarrow A_i$ where $(A_i)_{i=1,\cdots,n}$ is the sequence of type-occurrences of Δ. We can prove that for $R\vdash\Delta\Rightarrow A$ iff $R\vdash\Delta\twoheadrightarrow A$.

One considers the following canonical model $\mathcal{M} = (\mathcal{P}(\mathbf{Irr}), \alpha_R)$, where $\alpha_R(p) = [p]_R \triangleq \{T : R \vdash T \twoheadrightarrow p\}$. Writing $[\![\cdot]\!]$ instead of $[\![\cdot]\!]^{\mathcal{M}}_{\alpha_R}$, one proves that for every type $A \in \mathbf{Tp}_*$, $[\![A]\!] = [A]_R$. By rule $Axiom_R$ of $\mathbf{nD}_*$, it is readily seen that $\mathcal{M} \models R$. The product rules of elimination help to straightforwardly prove that $[\![A \star B]\!] = [A \star B]_R$, where $\star \in \{\bullet, \odot_i : i > 0\}$. To prove that for every residuated DA algebra $\mathcal{A}$ is isomorphically embeddable into a powerset rediduated DA over a DA, one defines a bijection between the carrier set of $\mathcal{A}$ and the set of primitive types $Pr = (p_a)_{a \in |\mathcal{A}|}$. We define the valuation $\mu(p_a) = a$, and the set of hypersequents which hold in $(\mathcal{A}, \mu)$, i.e. $R = \{\Delta \Rightarrow A : \mu(\Delta) \leq \mu(A)\}$. We consider the canonical model $\mathcal{M} = (\mathcal{P}(\mathbf{Irr}), \alpha_R)$, and we define the faithful monomorphism $h : |\mathcal{A}| \to |\mathcal{P}(\mathbf{Irr})|$ such that $h(a) := \alpha_R(p_a)$, and $h(0) = \Lambda$, and $h(1) = 1$. Finally, in order to obtain full strong completeness w.r.t. $\mathcal{PRDD}$, one uses the representation theorem and the fact that $\mathbf{hD}$ is strongly complete with respect residuated DAs (see Subsection 2.2).

5 Conclusions

The strong completeness theorems we have proved are quite analogous to the ones of $\mathbf{L}_*$. The semantics are quite natural as in the case of $\mathbf{L}_*$. We think that these results constitute a big step towards the study of the model theory of $\mathbf{hD}$.

It is known that $\mathbf{L}_*$ is not weakly complete w.r.t. free monoids (see [11]). Hence, weak completeness of full $\mathbf{D}$ (with units) w.r.t. L-models is not possible. It remains open whether $\mathbf{D}_*$ is weakly complete w.r.t. L-models.

References

1. A. Avron. Hypersequents, Logical Consequence and Intermediate Logic form Concurrency. *Annals of Mathematics and Artificial Intelligence*, 4:225–248, 1991.
2. W. Buszkowski. Completeness results for Lambek syntactic calculus. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 32:13–28, 1986.
3. Richard Moot. Extended Lambek calculi and first-order linear logic. In Claudia Casadio, Bob Coecke, Michael Moortgat, and Philip Scott, editors, *Categories and Types in Logic, Language, and Physics*, volume 8222 of *Lecture Notes in Computer Science*, pages 297–330. Springer Berlin Heidelberg, 2014.
4. G. Morrill and O. Valentín. Spurious ambiguity and focalisation. Manuscript, Submitted.
5. Glyn Morrill, Mario Fadda, and Oriol Valentín. Nondeterministic Discontinuous Lambek Calculus. In Jeroen Geertzen, Elias Thijsse, Harry Bunt, and Amanda Schiffrin, editors, *Proceedings of the Seventh International Workshop on Computational Semantics, IWCS-7*, pages 129–141. Tilburg University, 2007.
6. Glyn Morrill and Oriol Valentín. Displacement Calculus. *Linguistic Analysis*, 36(1–4):167–192, 2010. Special issue Festschrift for Joachim Lambek, http://arxiv.org/abs/1004.4181.
7. Glyn Morrill and Oriol Valentín. On Calculus of Displacement. In Srinivas Bangalore, Robert Frank, and Maribel Romero, editors, *TAG+10: Proceedings of the 10th International Workshop on Tree Adjoining Grammars and Related Formalisms*, pages 45–52, New Haven, 2010. Linguistics Department, Yale University.

8. Glyn Morrill, Oriol Valentín, and Mario Fadda. Dutch Grammar and Processing: A Case Study in TLG. In Peter Bosch, David Gabelaia, and Jérôme Lang, editors, *Logic, Language, and Computation: 7th International Tbilisi Symposium, Revised Selected Papers*, number 5422 in Lecture Notes in Artificial Intelligence, pages 272–286, Berlin, 2009. Springer.
9. Glyn Morrill, Oriol Valentín, and Mario Fadda. The Displacement Calculus. *Journal of Logic, Language and Information*, 20(1):1–48, 2011. Doi 10.1007/s10849-010-9129-2.
10. Alexey Sorokin. Normal forms for multiple context-free languages and displacement lambek grammars. In Sergei Artemov and Anil Nerode, editors, *Logical Foundations of Computer Science*, volume 7734 of *Lecture Notes in Computer Science*, pages 319–334. Springer Berlin Heidelberg, 2013.
11. Oriol Valentín. *Theory of Discontinuous Lambek Calculus*. PhD thesis, Universitat Autònoma de Barcelona, Barcelona, 2012.

On some Extensions of Syntactic Concept Lattices: Completeness and Finiteness Results

Christian Wurm
cwurm@phil.hhu.de

Universität Düsseldorf

Abstract. We provide some additional completeness results for the full Lambek calculus and syntactic concept lattices, where the underlying structure is extended to tuples of arbitrary finite and infinite size. Whereas this answers an open question for finite tuples, infinite tuples have not been considered yet. Nonetheless, they have a number of interesting properties which we establish in this paper, such as a particular class of languages which results in a finite lattice.

1 Introduction

Syntactic concept lattices arise from the distributional structure of languages. Their main advantage is that they can be constructed on distributional relations which are weaker than strict equivalence. [3] has shown how these lattices can be enriched with a monoid structure to form residuated lattices. [19] has shown that the resulting class of syntactic concept lattices for arbitrary languages forms a complete class of models for the logics $\mathbf{FL}_\bot$, i.e. the full Lambek calculus, and its reducts $\mathbf{FL}$, $L1$, for which it is a conservative extension.

In this paper, we will consider syntactic concept lattices over extended monoids: these will no longer consist of (sets of) strings, but rather of (sets of) tuples of strings, first of arbitrary finite, then of infinite size. The monoid operation has to be modified accordingly, of course. We show that the completeness results can be extended to this case for $\mathbf{FL}_\bot$ and its reducts; our proof will be constructed on top of the completeness results in [19] by means of isomorphic embeddings.

Finite tuples have been considered in formal language theory in a huge number of different contexts; the most relevant for us are [5],[15]. The use of infinite tuples has not been considered yet (to our knowledge). We show that it comes with some interesting gain in expressive power, while still being well-behaved; we also establish the largest class of language which results in a finite lattice over infinite tuples.

2 Residuated Syntactic Concept Lattices and Extensions

2.1 Equivalences on Strings and Tuples

Syntactic concept lattices originally arose in the structuralist approach to syntax, back when syntacticians tried to capture syntactic structures purely in terms of

distributions of strings[1] (see, e.g. [10]). An obvious way to do so is by partitioning strings/substrings into *equivalence classes*: we say that two strings w, v are equivalent in a language $L \subseteq \Sigma^*$, in symbols

(1) $w \sim_L^1 v$, iff for all $x, y \in \Sigma^*$, $xwy \in L \Leftrightarrow xvy \in L$.

This can be extended to tuples of strings of arbitrary size:

(2) $(w_1, v_1) \sim_L^2 (w_2, v_2)$, iff for all $x, y, z \in \Sigma^*$, $xw_1yv_1z \in L \Leftrightarrow xw_2yv_2z$, etc.

The problem with equivalence classes is that they are too restrictive for many purposes: assume we want to induce our grammar on the basis of a given dataset; then it is quite improbable that we get the equivalence classes we would usually desire. And even if we have an unlimited supply of examples, it seems unrealistic to describe our grammar on the basis of equivalence classes only: there might be constructions, collocations, idioms which ruin equivalences which we would intuitively consider to be adequate. Another drawback of equivalence classes is that for context-free languages, there is no general way to relate them to the non-terminals of some grammar generating the language, whereas for syntactic concepts, there are some interesting connections (see [6]).

Syntactic concepts provide a somewhat less rigid notion of equivalence, which can be conceived of as equivalence restricted to a given set of contexts. This at least partly overcomes the difficulties we have mentioned here.

2.2 Syntactic Concepts and Polar Maps

For a general introduction to lattices, see [7]; for background on residuated lattices, see [9]. Syntactic concept lattices form a particular case of what is well-known as formal concept lattice (or formal concept analysis) in computer science. In linguistics, they have been introduced in [18]. They were brought back to attention and enriched with residuation in [3], [4], as they turn out to be useful representations for language learning.

Given a language $L \subseteq \Sigma^*$, we define two maps: a map $\triangleright : \wp((\Sigma^*)^n) \to \wp((\Sigma^*)^{n+1})$, and $\triangleleft : \wp((\Sigma^*)^{n+1}) \to \wp((\Sigma^*)^n)$, which are defined as follows:

(3) for $M \subseteq (\Sigma^*)^n$, $M^{\triangleright} := \{(x_1, ..., x_{n+1} : \forall (w_1, ..., w_n) \in M, x_1w_1...w_nx_{n+1} \in L\}$;

and dually

(4) for $C \subseteq (\Sigma^*)^{n+1}$, $C^{\triangleleft} := \{(w_1, ..., w_n) : \forall (x_1, ..., x_{n+1}) \in C, x_1w_1...w_nx_{n+1} \in L\}$.

That is, a set of tuples of strings is mapped to the set of tuples of contexts in which all of its elements can occur. The dual function maps a set of contexts to the set of strings, which can occur in all of them. Usually, the case where $n = 1$

[1] Or words, respectively, depending on whether we think of our language as a set of words or a set of strings of words; we will choose the former option.

has been in the focus, as in [3],[19]. The more general cases have been considered in one form or other by [5],[15]). Obviously, $\triangleleft$ and $\triangleright$ are only defined with respect to a given language L and a given tuple size, otherwise they are meaningless. As long as it is clear of which language (if any particular language) and tuple size (if any particular) we are speaking, we will omit however any reference to it, to keep notation perspicuous. For a set of contexts C, $C^{\triangleleft}$ can be thought of as an equivalence class with respect to the contexts in C; but there might be elements in $C^{\triangleleft}$ which can occur in a context $(v, w) \notin C$ (and conversely). There is one more extension we will consider which is not entirely trivial, namely the one from tuples of *arbitrary* size to tuples of *infinite* size.

(5) for $M \subseteq (\Sigma^*)^\omega$, $M^{\triangleright} := \{(x_1, x_2, ...) : \forall (w_1, w_2, ...) \in M, x_1 w_1 w_2 x_2 ... \in L\}$.

One might consider this meaningless, as L consists of finite words, M of infinite tuples. But this is unjustified: it only entails that for any infinite tuple $\overline{w} \in M$ or $\overline{w} \in M^{\triangleright}$, in order to be "meaningful", all but finitely many components must be ϵ. So for each "meaningful" $(w_1, w_2, ...)$, there is a $k \in \mathbb{N}$ such that for all $j \geq k$, $w_j = \epsilon$. We gladly accept this restriction and remain with tuples where *almost all* components are empty. This is still a proper generalization of any tuple size n, because there is no finite upper bound for non-empty components in sets of tuples.

We define $\cdot$ on (finite or infinite) tuples by componentwise concatenation, that is, $(w_1, w_2, ...) \cdot (v_1, v_2, ...) = (w_1 v_1, w_2 v_2, ...)$. This choice is not unquestionable: some authors seem to prefer concatenation of the type $\oplus$, where $(w, v) \oplus (x, y) = (wx, yv)$. In the context of multiple context-free grammars this is referred to as *well-nestedness* and has attracted great interest (see e.g. [12]). The problem with this type of concatenation is that it is not easily extended beyond tuples of size two. What is interesting in this context is that we can use the ω-tuples to *simulate* $\oplus$-style concatenation with $\cdot$-style concatenation. To briefly sketch what this means, we define the forgetful map fo by $\mathsf{fo}(w_1, w_2, ...) = w_1 w_2 ...$, for arbitrary finite/infinite tuple size. We can now for all sequences of tuples $(x_1, y_1), ..., (x_i, y_i)$ devise ω-tuples $\overline{w}_1, ..., \overline{w}_i$ such that for all $1 \leq j, j' \leq i$, we have

$$\mathsf{fo}((x_j, y_j) \oplus ... \oplus (x_{j'}, y_{j'})) = \mathsf{fo}(\overline{w}_j \cdot ... \cdot \overline{w}_{j'})$$

This is not generally possible with any finite tuple size, and this is what makes infinite tuples interesting for us. Note that we can now also simplify things for ω-tuples, as we have $\overline{v} \in \{\overline{w}\}^{\triangleright}$ iff $\mathsf{fo}(\overline{v} \cdot \overline{w}) \in L$.

Regardless of the underlying objects, the two compositions of the maps, $\triangleleft\triangleright$ and $\triangleright\triangleleft$, are closure operators, that is:

1. $M \subseteq M^{\triangleright\triangleleft}$,
2. $M^{\triangleright\triangleleft} = M^{\triangleright\triangleleft\triangleright\triangleleft}$,
3. $M \subseteq N \Rightarrow M^{\triangleright\triangleleft} \subseteq N^{\triangleright\triangleleft}$,

for $M, N \subseteq \Sigma^*$. The same holds for contexts and $\triangleleft\triangleright$. A set M is **closed**, if $M^{\triangleright\triangleleft} = M$ etc. The closure operator $\triangleright\triangleleft$ gives rise to a lattice $\langle \mathcal{B}_L^n, \leq \rangle$, where the

elements of $\mathcal{B}_L^n$ are the sets $M \subseteq (\Sigma^*)^n$ such that $M = M^{\triangleright\triangleleft}$, and $\leq$ is interpreted as $\subseteq$. The same can be done with the set of closed contexts. Given these two lattices, $\triangleright$ and $\triangleleft$ establish a Galois connection between the two:

1. $M \leq N \Leftrightarrow M^{\triangleleft} \geq N^{\triangleleft}$, and
2. $C \leq D \Leftrightarrow C^{\triangleright} \geq D^{\triangleright}$.

A syntactic concept A is usually defined to be an ordered pair, consisting of a closed set of strings, and a closed set of contexts, so it has the form $\langle S, C\rangle$, such that $S^{\triangleright} = C$ and $C^{\triangleleft} = S$. For our purposes, we only need to consider the left component, so we suppress the contexts and only consider the stringsets of the form $M^{\triangleright\triangleleft}$. For all operations we define below, it can be easily seen that the resulting structures are isomorphic. So when we refer to a concept, we only mean a $[-]^{\triangleright\triangleleft}$ closed set of strings, the concept in the classical sense being easily reconstructible.

Definition 1 *The set of concepts of a language L forms a lattice denoted by $SCL_n(L) := \langle \mathcal{B}_L^n, \wedge, \vee, \top, \bot\rangle$, where $\top = (\Sigma^*)^n$, $\bot = \emptyset^{\triangleright\triangleleft}$, and for $M, N \in \mathcal{B}_L^n$, $M \wedge N = M \cap N$, $M \vee N = (M \cup N)^{\triangleright\triangleleft}$.*

It is easy to see that this defines an order in the usual fashion which coincides with $\subseteq$ on closed sets of strings. It is easy to verify that this forms a complete lattice, as infinite joins are defined by (closure of) infinite unions, infinite meets by infinite intersections. Note also that for any set of (tuples of) strings S and contexts C, $S^{\triangleright} = S^{\triangleright\triangleleft\triangleright}$ and $C^{\triangleleft} = C^{\triangleleft\triangleright\triangleleft}$. $SCL_\omega(L)$ denotes the according structure with infinite tuple size. To see that things are properly different in the infinite case, we present the following result:

Lemma 2 *1. For any $n \in \mathbb{N}$, $SCL_n(L)$ is finite iff $L \in Reg$.*
2. There are $L \in Reg$ such that $SCL_\omega(L)$ is infinite.

Let $[\Sigma]^*_{\sim_L^1}$ denote the set of $\sim_L^1$-congruence classes over Σ^*.

Proof. 1. $SCL_1(L)$ is finite iff $[\Sigma]^*_{\sim_L^1}$ is finite iff L is regular (both are well-known). Moreover, $|[\Sigma^*]_{\sim_L^{n+1}}| \leq |[\Sigma^*]_{\sim_L^n}| \cdot |[\Sigma^*]_{\sim_L^1}|$, as $\sim_L^1$-equivalent strings are equivalent in all contexts. From this the claim easily follows.

2. Take the language $L = a^*b^*$, and all tuples of the form $(a, \overbrace{\epsilon, ..., \epsilon}^{n \text{ times}}, a, \epsilon, \epsilon, ...)$ for $n \in \mathbb{N}$. For every $m, m' \in \mathbb{N}$, $m < m'$, we take the tuple $(\overbrace{a, ..., a}^{m+1 \text{ times}}, \overbrace{b, ..., b}^{m+3 \text{ times}}, \epsilon, \epsilon, ...)$; it is easy to see that $\mathsf{fo}((\overbrace{a, ..., a}^{m+1 \text{ times}}, \overbrace{b, ..., b}^{m+3 \text{ times}}, \epsilon, \epsilon, ...) \cdot (a, \overbrace{\epsilon, ..., \epsilon}^{m \text{ times}}, a, \epsilon, \epsilon, ...)) \in L$, whereas if we substitute m with m', the result is not in L.

Consequently, there are infinitely concepts, namely for each $n \in \mathbb{N}$ at least one which contains $(a, \overbrace{\epsilon, ..., \epsilon}^{n \text{ times}}, a, \epsilon, \epsilon, ...)$ but no $(a, \overbrace{\epsilon, ..., \epsilon}^{n' \text{ times}}, a, \epsilon, \epsilon, ...)$ for $n' > n$. $\dashv$

This raises the question: what is the class C of languages such that $L \in C$ if and only if $SCL_\omega(L)$ is finite? We will give a concise characterization of this class later on.

2.3 Monoid Structure and Residuation

As we have seen, the set of concepts of a language forms a lattice. In addition, we can also give it the structure of a monoid: for concepts M, N, we define $M \circ N := (M \cdot N)^{\triangleright\triangleleft}$, where $M \cdot N = \{\overline{w} \cdot \overline{v} : \overline{w} \in M, \overline{v} \in N\}$. We often write MN for $M \cdot N$. '$\circ$' is associative on concepts: For $M, N, O \in \mathcal{B}_n^L$, $M \circ (N \circ O) = (M \circ N) \circ O$. This follows from the fact that $[-]^{\triangleright\triangleleft}$ is a *nucleus*, that is, it is a closure operator and in addition it satisfies $S^{\triangleright\triangleleft} T^{\triangleright\triangleleft} \subseteq (ST)^{\triangleright\triangleleft}$, and the associativity of $\cdot$-concatenation (no matter on which tuple size).

Furthermore, it is easy to see that the neutral element of '$\circ$' is $\{\epsilon\}^{\triangleright\triangleleft}$. The monoid operation respects the partial order of the lattice, that is, for $X, Y, Z, W \in \mathcal{B}_n^L$, if $X \leq Y$, then $W \circ X \circ Z \leq W \circ Y \circ Z$. We enrich this with residuals, using the following definition:

Definition 3 *Let X, Y be concepts. We define the right residual $X/Y := \bigvee\{Z : Z \circ Y \leq X\}$, the left residual $Y\backslash X := \bigvee\{Z : Y \circ Z \leq X\}$.*

Note that this is an entirely abstract definition which does not make reference to any underlying structure. That it works is ensured by the following lemma.

Lemma 4 *Let L be a complete lattice with a monoid operation respecting the order. Then for $X, Y, Z \in L$, residuals defined as above, we have $Y \leq X\backslash Z$ iff $X \circ Y \leq Z$ iff $X \leq Z/Y$.*

Proof. We only prove the first bi-implication.

If: assume $X \circ Y \leq Z$. Then $Y \in \{W : X \circ W \leq Z\}$; so $Y \leq \bigvee\{W : X \circ W \leq Z\} = X\backslash Z$.

Only if: This uses the fact that infinite joins distribute over $\circ$, that is, $M \circ \bigvee \mathbf{N} = \bigvee\{M \circ N : N \in \mathbf{N}\}$ (see [9] for this and many similar results). Consequently, we have $X \circ X\backslash Z = X \circ \bigvee\{W : X \circ W \leq Z\} = \bigvee\{X \circ W : X \circ W \leq Z\} \leq Z$; as $Y \leq X\backslash Z$, we have $X \circ Y \leq X \circ X\backslash Z \leq Z$. ⊣

So every complete lattice with a monoid operation respecting the order can be extended to a residuated lattice.

Definition 5 *The **syntactic concept lattice** of a language L is defined as $SCL_n(L) := \langle \mathcal{B}_L^n, \wedge, \vee, \top, \bot, 1, \circ, /, \backslash \rangle$, where $\mathcal{B}_L^n, \wedge, \vee, \top, \bot$ are defined as in definition 1, $1 = \{\epsilon\}^{\triangleright\triangleleft}$, and $\circ, /, \backslash$ are as defined above.*

Note that we somewhat overloaded the notation of $SCL_n(L)$; in the sequel we will however thereby always refer to definition 5. Moreover, we will denote by SCL the class of all lattices of the form $SCL_1(L)$ for some language L, without any further requirement regarding L; same for SCL_n for $n \in \mathbb{N} \cup \{\omega\}$.

3 Lambek Calculus and Extensions

3.1 The Logics L, $L1$, FL and $\mathbf{FL}_\bot$

The Lambek calculus L was introduced in [13]. $L1$ is a proper extension of L, and $\mathbf{FL}, \mathbf{FL}_\bot$ are each conservative extensions of $L1$ and the preceding one. Let

Pr be a set, the set of **primitive types**, and C be a set of **constructors**, which is, depending on the logics we use, $C_L := \{/, \backslash, \bullet\}$, or $C_{\mathbf{FL}} := \{/, \backslash, \bullet, \vee, \wedge\}$. By $Tp_C(Pr)$ we denote the set of types over Pr, which is defined as the smallest set, such that:

1. $Pr \subseteq Tp_C(Pr)$.
2. if $\alpha, \beta \in Tp_C(Pr)$, $\star \in C$, then $\alpha \star \beta \in Tp_C(Pr)$.

If there is no danger of confusion regarding the primitive types and constructors, we also simply write Tp for $Tp_C(Pr)$. We now present the inference rules corresponding to these constructors. We call an inference of the form $\Gamma \vdash \alpha$ a **sequent**, for $\Gamma \in Tp^*$, $\alpha \in Tp$, where by Tp^* we denote the set of all (possibly empty) *sequences* over Tp, which are concatenated by ',' (keep in mind the difference between *sequents*, which have the form $\Gamma \vdash \alpha$, and *sequences* like Γ, which are in Tp^*).

With few exceptions, rules of inference in our logics are not given in the form of sequents $\Gamma \vdash \alpha$, but rather as rules to derive new sequents from given ones. In general, uppercase Greek letters range as variables over sequences of types. In the inference rules for L, premises of $' \vdash '$ (that is, left hand sides of sequents) must be non-empty; in $L1$ they can be empty as well; everything else is equal. In **FL** and $\mathbf{FL}_\bot$ we also allow for empty sequents. Lowercase Greek letters range over single types. Below, we present the standard rules of the Lambek calculus L / $L1$.

$$(ax) \qquad \alpha \vdash \alpha \qquad\qquad (cut) \qquad \frac{\Delta, \beta, \Theta \vdash \alpha \quad \Gamma \vdash \beta}{\Delta, \Gamma, \Theta \vdash \alpha}$$

$$(\mathbf{I}-/) \qquad \frac{\Gamma, \alpha \vdash \beta}{\Gamma \vdash \beta/\alpha} \qquad\qquad (\mathbf{I}-\backslash) \qquad \frac{\alpha, \Gamma \vdash \beta}{\Gamma \vdash \alpha\backslash\beta}$$

$$(/-\mathbf{I}) \qquad \frac{\Delta, \beta, \Theta \vdash \gamma \quad \Gamma \vdash \alpha}{\Delta, \beta/\alpha, \Gamma, \Theta \vdash \gamma} \qquad\qquad (\backslash-\mathbf{I}) \qquad \frac{\Delta, \beta, \Theta \vdash \gamma \quad \Gamma \vdash \alpha}{\Delta, \Gamma, \alpha\backslash\beta, \Theta \vdash \gamma}$$

$$(\bullet-\mathbf{I}) \qquad \frac{\Delta, \alpha, \beta, \Gamma \vdash \gamma}{\Delta, \alpha \bullet \beta, \Gamma \vdash \gamma} \qquad\qquad (\mathbf{I}-\bullet) \qquad \frac{\Delta \vdash \alpha \quad \Gamma \vdash \beta}{\Delta, \Gamma \vdash \alpha \bullet \beta}$$

These are the standard rules of L / $L1$ (roughly as in [13]). We have rules to introduce either slash and '$\bullet$' both on the right hand side of $\vdash$ and on the left hand side of $\vdash$. We will now add two additional connectives, which are well-known from structural logics, namely $\vee$ and $\wedge$. These are not present in $L/L1$, have however been considered as extensions as early as in [14], and have been subsequently studied by [11].

$$(\wedge-\mathbf{I}\ 1) \qquad \frac{\Gamma, \alpha, \Delta \vdash \gamma}{\Gamma, \alpha \wedge \beta, \Delta \vdash \gamma} \qquad\qquad (\wedge-\mathbf{I}\ 2) \qquad \frac{\Gamma, \beta, \Delta \vdash \gamma}{\Gamma, \alpha \wedge \beta, \Delta \vdash \gamma}$$

$$(\mathbf{I}-\wedge)\qquad \frac{\Gamma\vdash\alpha\quad\Gamma\vdash\beta}{\Gamma\vdash\alpha\wedge\beta}$$

$$(\vee-\mathbf{I})\qquad \frac{\Gamma,\alpha,\Delta\vdash\gamma\quad\Gamma,\beta,\Delta\vdash\gamma}{\Gamma,\alpha\vee\beta,\Delta\vdash\gamma}$$

$$(\mathbf{I}-\vee\,1)\qquad \frac{\Gamma\vdash\alpha}{\Gamma\vdash\alpha\vee\beta}\qquad(\mathbf{I}-\vee\,2)\qquad \frac{\Gamma\vdash\beta}{\Gamma\vdash\alpha\vee\beta}$$

$$(1-\mathbf{I})\qquad \frac{\Gamma,\Delta\vdash\alpha}{\Gamma,1,\Delta\vdash\alpha}\qquad(\mathbf{I}-1)\qquad\vdash 1$$

This gives us the logic **FL**. Note that this slightly deviates from standard terminology, because usually, **FL** has an additional constant 0 (not to be confused with $\bot$!). In our formulation, 0 and 1 coincide. In order to have logical counterparts for the bounded lattice elements $\top$ and $\bot$, we introduce two logical constants, which are denoted by the same symbol.[2]

$$(\bot-\mathbf{I})\qquad \Gamma,\bot\,\Delta\vdash\alpha\qquad(\mathbf{I}-\top)\qquad\Gamma\vdash\top$$

This gives us the calculus $\mathbf{FL}_\bot$. ¿From a logical point of view, all these extensions of L are quite well-behaved: they are conservative, and also allow us to preserve the important result of [13], namely admissibility of the cut-rule.

We say that a sequent $\Gamma\vdash\alpha$ is derivable in a calculus, if it can be derived by the axiom and the rules of inference; we then write $\Vdash_L\Gamma\vdash\alpha$, $\Vdash_{L1}\Gamma\vdash\alpha$, $\Vdash_{\mathbf{FL}}\Gamma\vdash\alpha$ etc., depending on which calculus we use.

3.2 Interpretations of $L1$, FL and $\mathbf{FL}_\bot$

The standard model for $L1$ is the class of residuated monoids. These are structures $(M,\cdot,1,\backslash,/,\leq)$, where $(M,\cdot,1)$ is a monoid, $(M,\leq)$ is a partial order, and $\cdot,/,\backslash$ satisfy the law of residuation: for $m,n,o\in M$,

$$(6)\qquad m\leq o/n\Leftrightarrow m\cdot n\leq o\Leftrightarrow n\leq m\backslash o.$$

Note that this implies that $\cdot$ respects the order $\leq$. The standard model for **FL** is the class of residuated lattices, and for $\mathbf{FL}_\bot$, the class of bounded residuated lattices. A residuated lattice is an algebraic structure $\langle M,\cdot,\vee,\wedge,\backslash,/,1\rangle$, where in addition to the previous requirements, $(M,\vee,\wedge)$ is a lattice; the lattice order

[2] Whereas L and $L1$ are equally powerful in the sense of languages which are recognizable, [11] shows that **FL** is considerably more powerful than L: whereas L only recognizes context-free languages by the classical result of [17], **FL** can recognize any finite intersection of context-free languages. We only briefly mention this, because we have no space to make precise what it means for a calculus to recognize a class of languages.

$\leq$ need not be stated, as it can be induced by $\vee$ or $\wedge$: for $a, b \in M$, $a \leq b$ is a shorthand for $a \vee b = b$. A bounded residuated lattice is a structure $\langle M, \cdot, \vee, \wedge, \backslash, /, 1, \top, \bot \rangle$, where $\langle M, \cdot, \vee, \wedge, \backslash, /, 1 \rangle$ is a residuated lattice, $\top$ is the maximal element of the lattice order $\leq$ and $\bot$ is its minimal element.

For a general introduction see [9]. We will give definitions only once for each operator; we can do so because each definition for a given connector is valid for all classes in which it is present.

We call the class of residuated monoids RM, the class of residuated lattices RL; the class of bounded residuated lattices $RL_\bot$. We now give a semantics for the calculi above. We start with an interpretation $\sigma : Pr \to M$ which interprets elements in Pr in elements of the lattice, and extend σ to $\overline{\sigma}$ by defining it inductively over our type constructors, which is for now the set $C := \{/, \backslash, \bullet, \vee, \wedge\}$. For $\alpha, \beta \in Tp_C(Pr)$,

1. $\overline{\sigma}(\alpha) = \sigma(\alpha) \in M$, if $\alpha \in Pr$
2. $\overline{\sigma}(\top) = \top$
2' $\overline{\sigma}(\top)$ is an arbitrary $m \in M$ such for all $\alpha \in Tp_C(Pr)$, $\overline{\sigma}(\alpha) \leq m$.
3. $\overline{\sigma}(\bot) = \bot$
4. $\overline{\sigma}(1) = 1$
5. $\overline{\sigma}(\alpha \bullet \beta) := \overline{\sigma}(\alpha) \cdot \overline{\sigma}(\beta)$
6. $\overline{\sigma}(\alpha / \beta) := \overline{\sigma}(\alpha) / \overline{\sigma}(\beta)$
7. $\overline{\sigma}(\alpha \backslash \beta) := \overline{\sigma}(\alpha) \backslash \overline{\sigma}(\beta)$
8. $\overline{\sigma}(\alpha \vee \beta) := \overline{\sigma}(\alpha) \vee \overline{\sigma}(\beta)$
9. $\overline{\sigma}(\alpha \wedge \beta) := \overline{\sigma}(\alpha) \wedge \overline{\sigma}(\beta)$

Note that the constructors on the left-hand side and on the right-hand side of the definition look identical (with the exception of $\bullet$ and $\cdot$), but they are not: on the left-hand side, they are type constructors, on the right hand side, they are operators of a residuated lattice. The same holds for the constants $\top, \bot, 1$. Note that there are two alternative interpretations for $\top$: one which interprets it as the upper bound for the lattice, which is the standard interpretation, and one which just interprets it as an arbitrary element. The latter will be called the **non-standard** interpretation and play some role in the sequel. Non-standard interpretations form a generalization of standard interpretations, and as we will see below, this is a proper generalization. ¿From this it trivially follows that every completeness result which holds for standard interpretations also holds for non-standard interpretations, but we have to show that soundness is preserved. This however is also straightforward, as there is only one rule involving $\top$ and it can be easily seen to be sound under non-standard interpretations.

This is how we interpret the types of our logic. What we want to interpret next is the *sequents* of the form $\Gamma \vdash \alpha$. We say that a sequent $R = \gamma_1, ..., \gamma_i \vdash \alpha$ is true in a model $\mathcal{M}$ under assignment σ, in symbols: $(\mathcal{M}, \sigma) \models \gamma_1, ..., \gamma_i \vdash \alpha$, if and only if $\overline{\sigma}(\gamma_1 \bullet ... \bullet \gamma_i) \leq \overline{\sigma}(\alpha)$ holds in $\mathcal{M}$. That is, we interpret the ',', which denotes concatenation in sequents, as $\cdot$ in the model, and $\vdash$ as $\leq$. In the sequel, for Γ a sequence of types, we will often write $\overline{\sigma}(\Gamma)$ as an abbreviation, where we leave the former translation implicit. For the case of theorems, that is, derivable

sequents with no antecedent, we have the following convention: $(\mathcal{M}, \sigma) \models \vdash \alpha$, iff $1 \leq \overline{\sigma}(\alpha)$ in $\mathcal{M}$, where 1 is the unit element of $\mathcal{M}$ (note that this case does not arise in L).

More generally, for a given class of (bounded) residuated lattices (monoids, semigroups) $\mathfrak{C}$, we say that a sequent is *valid* in $\mathfrak{C}$, in symbols, $\mathfrak{C} \models \gamma_1, ..., \gamma_i \vdash \alpha$, if for all $\mathcal{M} \in \mathfrak{C}$ and all interpretations σ, $(\mathcal{M}, \sigma) \models \gamma_1, ...\gamma_i \vdash \alpha$ (here we have to distinguish between standard and non-standard interpretations).

4 Completeness: Previous Results

There are a number of completeness results for the logics we have considered here. We will consider the most general ones, which will be important in the sequel.

Theorem 6 *For the class RM of residuated monoids, the class RL of residuated lattices, the class $RL_\perp$ of bounded residuated lattices,*

1. *$RM \models \Gamma \vdash \alpha$ if and only if $\Vdash_{L1} \Gamma \vdash \alpha$,*
2. *$RL \models \Gamma \vdash \alpha$ if and only if $\Vdash_{\boldsymbol{FL}} \Gamma \vdash \alpha$,*
3. *$RL_\perp \models \Gamma \vdash \alpha$ if and only if $\Vdash_{\boldsymbol{FL}_\perp} \Gamma \vdash \alpha$.*

For reference on theorem 6, see [1], [2], [9]. The proofs for the above completeness theorems usually proceed via the Lindenbaum-Tarski construction: we interpret primitive types as atomic terms modulo mutual derivability, and define $\sigma(\alpha) \leq \sigma(\beta)$ iff $\alpha \vdash \beta$. Then we can perform an induction over constructors to get the same for arbitrary formulas/terms. So there are quite simple completeness proofs for the general case. These completeness results can actually be strengthened to the *finite model property*. A logic, equipped with a class of models and interpretations, is said to have finite model property if it is complete in the finite; that is, theorem 6 remains valid if we restrict ourself to finite models. These results are highly non-trivial; for example, classical first-order logic fails to have finite model property.

Theorem 7 *1. L1 has finite model property;*
*2. **FL** has finite model property;*
3. $\boldsymbol{FL}_\perp$ has finite model property.

For the first claim, consider [8]; the second and third has been established by [16]. We want to establish soundness and completeness of the calculi with respect to the class of syntactic concept lattices and their reducts. The latter results are crucial to show that completeness holds if we restrict ourselves to languages over finite alphabets.

Now we list some soundness and completeness results for syntactic concept lattices. First we see that our calculus is sound with respect to the model:

Theorem 8 *(Soundness) If $\Vdash_{\boldsymbol{FL}_\perp} \Gamma \vdash \alpha$, then for the class of syntactic concept lattices SCL, we have $SCL \models \Gamma \vdash \alpha$.*

This actually follows from soundness direction of theorem 6, because SCL is just a particular class of bounded residuated lattices. As $L, L1, \mathbf{FL}$ are fragments of $\mathbf{FL}_\bot$, we get the same result for $L, L1$ and $\mathbf{FL}$, considering the terms which contain only the operators which have a counterpart in the logic.

Let SCL_{L1} be the class of SCL reducts with $\{\circ, /, \backslash\}$, which specify a unit, and SCL_{FL} be the class of SCL reducts with operators $\{\circ, /, \backslash, \vee, \wedge\}$, that is, without the constants $\top$ and $\bot$.

Theorem 9 *(Completeness)*

1. *If* $SCL_{L1} \models \Gamma \vdash \alpha$, *then* $\Vdash_{L1} \Gamma \vdash \alpha$;
2. *if* $SCL_{FL} \models \Gamma \vdash \alpha$, *then* $\Vdash_{\mathbf{FL}} \Gamma \vdash \alpha$;
3. *if* $SCL \models \Gamma \vdash \alpha$, *then* $\Vdash_{\mathbf{FL}_\bot} \Gamma \vdash \alpha$;

The completeness proofs can be found in [19]. The proof shows that for any (bounded) residuated lattice (reduct) S, there is a language $L(S)$ such that S can be isomorphically embedded in $SCL(L(S))$. This embedding thus preserves validity in both directions, and thus completeness follows. The language $L(S)$ is constructed with the elements of S as underlying alphabet. By the finite model property, we can conclude that the result remains valid if we restrict ourselves to languages over finite alphabets: if S is finite, $L(S)$ is a language over a finite alphabet (though still infinite!). So theorem 9 also holds for languages over finite alphabets only.

5 SCL_n – Completeness via Embeddings

We now extend theorem 9 to the structures $SCL_n : n \in \mathbb{N} \cup \{\omega\}$ (henceforth: $\mathbb{N}_\omega$). Again, we proceed by showing that there is an isomorphic embedding from $SCL(L) \to SCL_n(L)$. To increase readability and avoid misunderstandings, in the following we let $\triangleright, \triangleleft, \circ$ denote operations in SCL, $\blacktriangleleft, \blacktriangleright, \odot$ denote the corresponding operations in SCL_n. This convention however only concerns this section! We will exemplify the embedding for SCL_2, but it is easy to see that this can be extended to any $n \in \mathbb{N}$.

Assume $M \in SCL(L)$. We take a map $\alpha : \wp(\Sigma^*) \to \wp(\Sigma^* \times \Sigma^*)$, which is defined as a lifting of $\alpha' : w \mapsto (w, \epsilon)$ to sets composed with closure, so we define $\alpha(M) = (\alpha'[M])^{\blacktriangleright\blacktriangleleft}$. The following are more or less immediate:

1. $\alpha(M) \in SCL_2(L)$;
2. if $w \in M$, then $(w, \epsilon) \in \alpha(M)$;
3. $\alpha'[M] \star \alpha'[N] = \alpha'[M \star N]$, for $\star \in \{\cdot, \cup, \cap\}$.

The third point ensures that α' is a homomorphism for sets and classical set-theoretic operations of languages. Moreover, it is easy to see that α' is a bijection. So α' is an isomorphic embedding from $(\wp(\Sigma^*), \cdot, \cup, \cap)$ to $(\wp(\Sigma^* \times \Sigma^*), \cdot, \cup, \cap)$ Note that all these points remain valid if we suitably extend α' to α'_n, with $\alpha'_n(w) = (w, \overbrace{\epsilon, ..., \epsilon}^{n \text{ times}})$, with α_n defined accordingly.

This is quite obvious. It becomes much less obvious, if we switch our attention to α, that is, add the closure operation. The reason is as follows: $\alpha(M)$ might contain elements of the form (w, v) with $v \neq \epsilon$; and thus $\alpha(M) \cdot \alpha(N)$ might contain terms of the form $(w_1, w_2) \cdot (v_1, v_2) = (w_1 v_1, w_2 v_2)$. This is obviously problematic, as in $\alpha(M) \cdot \alpha(N)$ substrings occur in an order which differs from the one in $\mathsf{fo}(\alpha(M))\mathsf{fo}(\alpha(N))$. We show that the map α_n is nonetheless an isomorphic embedding $SCL(L) \to SCL_n(L)$. This requires some work, and we prove the claim step by step via the following lemmas (again, we exemplify this for $n = 1$, but results can be easily extended to the general case). We first make the following observation:

Lemma 10 *$\alpha'[M]^{\blacktriangleright} = \{(x, y, z) : (x, yz) \in M^{\triangleright}\}$;*

Both inclusions are obvious. This means that $\alpha(M)$ is the set of all (a, b) such that if $xMyz \subseteq L$, then $xaybz \in L$. This allows to show the following:

Lemma 11 *For $M = M^{\triangleright\triangleleft}$, $(w, \epsilon) \in \alpha(M)$ if and only if $w \in M$.*

Proof. *If*-direction is immediate. *Only if*: If $(w, \epsilon) \in \alpha(M)$, then whenever $(x, y, z) \in \alpha(M)^{\blacktriangleright}$, we have $xwyz \in L$. Now, $M^{\triangleright}$ is exactly the set of all (x, yz) such that $(x, y, z) \in \alpha(M)^{\blacktriangleright}$. Thus we have $w \in M^{\triangleright\triangleleft} = M$. $\dashv$

Put $M^a := \{wav : wv \in M\}$.

Lemma 12 *$\alpha(M) \odot \alpha(N) \subseteq \alpha(M \circ N)$.*

Proof. *Case 1*: Assume $(w, v) \in \alpha(M) \cdot \alpha(N)$. Then $(w, v) = (a_1 b_1, a_2 b_2)$, where $(a_1, a_2) \in \alpha(M)$, $(b_1, b_2) = \alpha(N)$. So for (a_1, a_2) it follows: if $xMyz \subseteq L$, then $xa_1ya_2z \in L$; the same holds for $(b_1, b_2) \in \alpha(N)$. So we have the following:

1. if $wMv \subseteq L$ $wa_1v^{a_2} \subseteq L$, and
2. if $wNv \subseteq L$, then $wb_1v^{b_2} \subseteq L$.

Case 1a: Assume there are x, y such that $xMNy \subseteq L$. Then it follows (by 2) that for every z_1, z_2 with $z_1 z_2 = y$, $xMb_1z_1b_2z_2 \subseteq L$, and consequently (by 1) that $xa_1b_1z_1a_2b_2z_2 \in L$, and so we have $(a_1b_1, a_2b_2) \in (\alpha'[M \circ N])^{\blacktriangleright\blacktriangleleft} = \alpha(M \circ N)$.

Case 1b: There are no x, y, z such that $xMNyz \subseteq L$. Then we have $M \circ N = \top$, and $MN^{\triangleright} = \emptyset$. By lemma 10, it follows that $\alpha'[MN]^{\blacktriangleright} = \emptyset$, and therefore, $\alpha'[MN]^{\blacktriangleright\blacktriangleleft} = \alpha(M \circ N) = \top$.

Case 2: $(w, v) \notin \alpha(M) \cdot \alpha(N)$. In this case, for all (x, y, z) such that for all $(a, b) \in \alpha(M) \cdot \alpha(N)$, $xaybz \in L$, we have $xwyvz \in L$. So it follows that if $(x, y, z) \in \alpha'[M \circ N]^{\blacktriangleright}$, then $xwyvz \in L$; thus $(w, v) \in \alpha'[M \circ N]^{\blacktriangleright\blacktriangleleft} = \alpha(M \circ N)$. $\dashv$

This is the first of a number of lemmas which establish the main theorem of this section. The second one establishes the inverse inclusion:

Lemma 13 *$\alpha(M \circ N) \subseteq \alpha(M) \odot \alpha(N)$.*

Proof. Assume $(w, v) \in \alpha(M \circ N)$.

Case 1: $(w, v) \in \alpha'[M \circ N]$. Then $v = \epsilon$, and $w \in M \circ N$. For all $w \in MN$, $(w, \epsilon) \in \alpha(M) \cdot \alpha(N)$. Furthermore, if $w \in MN^{\triangleright\triangleleft}$, then $(w, \epsilon) \in (\alpha(M) \cdot \alpha(N))^{\blacktriangleright\blacktriangleleft}$ by as simple argument using lemma 10. Consequently, $(w, v) = (w, \epsilon) \in \alpha(M) \odot \alpha(N)$.

Case 2: Assume $(w, v) \notin \alpha'[M \circ N]$. Consequently, it holds that if $(x, y, z) \in \alpha'[M \circ N]^{\blacktriangleright}$, then $xwyvz \in L$. As $\alpha'[M \circ N] \subseteq \alpha(M) \odot \alpha(N)$ (by case 1), we have $\alpha'[M \circ N]^{\blacktriangleright} \supseteq (\alpha(M) \odot \alpha(N))^{\blacktriangleright}$, and so if $(x, y, z) \in (\alpha(M) \odot \alpha(N))^{\blacktriangleright}$, then $xwyvz \in L$, and $(w, v) \in (\alpha(M) \odot \alpha(N))^{\blacktriangleright\blacktriangleleft} = \alpha(M) \odot \alpha(N)$. ⊣

So α is a homomorphism of $\circ$ (more generally, every α_n is a $\circ$-homomorphism). To show that it preserves meets and joins does not require a lot of work. In order to save one step, we directly prove the claim for infinite meets.

Lemma 14 $\bigwedge_{i \in I} \alpha(M_i) = \alpha(\bigwedge_{i \in I} M_i)$.

This proof is rather straightforward, as $\wedge$ equals $\cap$ in our case, a fact we will make use of.

Proof. $\subseteq$ Assume $(w, v) \in \alpha(M_i)$ for all $i \in I$. This means for all $i \in I$, if $x(M_i)yz \subseteq L$, then $xwyvz \in L$. We have $(x, yz) \in (\bigwedge_{i \in I} M_i)^{\triangleright}$ iff and only if $(x, y, z) \in \alpha'[\bigwedge_{i \in I} M_i]^{\blacktriangleright}$ (lemma 10). So if $(x, y, z) \in \alpha'[\bigwedge_{i \in I} M_i]^{\blacktriangleright}$, then $x(\bigcap_{i \in I} M_i)yz \subseteq L$, and so $xwyvz \in L$, and so $(w, v) \in \alpha'[\bigwedge_{i \in I} M_i]^{\blacktriangleright\blacktriangleleft} = \alpha(\bigwedge_{i \in I} M_i)$.

$\supseteq$ Assume $(w, v) \in \alpha(\bigwedge_{i \in I} M_i)$. Because $\alpha(X) = \alpha'[X]^{\blacktriangleright\blacktriangleleft}$, α' being a pointwise map on sets and $[-]^{\blacktriangleright\blacktriangleleft}$ being a closure operator, from $\bigwedge_{i \in I} M_i \subseteq M_j : j \in I$ it follows that $\alpha(\bigwedge_{i \in I} M_i) \subseteq \alpha(M_j)$; and so $\alpha(\bigwedge_{i \in I} M_i) \subseteq \bigcap_{i \in I} \alpha(M_i) = \bigwedge_{i \in I} \alpha(M_i)$. ⊣

Now we can use the fact that in a complete lattice, we can use meets to define joins (and vice versa). This allows us to derive the following:

Lemma 15 $\alpha(M) \vee \alpha(N) = \alpha(M \vee N)$.

Proof. We use the facts that 1. both $SCL(L), SCL_2(L)$ are complete, and 2. α preserves infinite meets. For these reasons, the following equality holds:

$$\alpha(M) \vee \alpha(N) = \bigwedge\{\alpha(X) : X \geq M, N\} = \alpha(\bigwedge\{X : X \geq M, N\}).$$

Moreover, we can easily extend this to the infinite case:

$$\bigvee_{i \in I} \alpha(M_i) = \bigwedge\{\alpha(X) : X \geq M_i : i \in I\} = \alpha(\bigwedge\{X : X \geq M_i : i \in I\}) = \alpha(\bigvee_{i \in I} M_i).$$

⊣

Again, this can be easily extended to any α_n, $n \in \mathbb{N}_\omega$.

Needless to say, every map $\alpha_n : SCL(L) \to SCL_n(L)$ is an injection. To see this, just assume we have $M, N \in SCL(L)$; and assume without loss of generality that $(w, v) \in M^{\triangleright}$, $(w, v) \notin N^{\triangleright}$. Then we have $(w, \epsilon, v) \in \alpha'[M]^{\blacktriangleright}$, but $(w, \epsilon, v) \notin \alpha'[N]^{\blacktriangleright}$, so $\alpha(M) = \alpha'[M]^{\blacktriangleright\blacktriangleleft} \neq \alpha'[N]^{\blacktriangleright\blacktriangleleft} = \alpha(N)$. This together with the with fact that we preserve joins and meets makes the following rather obvious:

Lemma 16 *$X \circ N \leq M$ iff $\alpha(X) \bullet \alpha(N) \leq \alpha(M)$.*

Proof. *If*: For contraposition, assume $X \circ N \not\leq M$. Then there is $w \in X \circ N$, $w \notin M$. Consequently, there is $(w, \epsilon) \in \alpha(X) \bullet \alpha(N)$, but $w \notin \alpha(M)$ (by lemma 11).

Only if: Assume $X \circ N \leq M$. Then obviously $\alpha(X \circ N) = \alpha(X) \bullet \alpha(N) \leq \alpha(M)$, as α preserves $\subseteq$. ⊣

Lemma 17 *$\alpha(M)/\alpha(N) = \alpha(M/N)$*

Proof. We have $M/N = \bigvee\{X : X \circ N \leq M\}$; moreover $\alpha(\bigvee\{X : X \circ N \leq M\}) = \bigvee\{\alpha(X) : X \circ N \leq M\}$. Since $X \circ N \leq M$ iff $\alpha(X) \bullet \alpha(N) \leq \alpha(M)$, we have $\bigvee\{\alpha(X) : X \circ N \leq M\} = \bigvee\{\alpha(X) : \alpha(X) \bullet \alpha(N) \leq \alpha(M)\} = \alpha(M)/\alpha(N)$. ⊣

Again, this proof works perfectly fine for any α_n. To not get confused with $\bot, \top$ in SCL, SCL_2, we denote the latter elements with $\bot_2, \top_2$ etc.

Lemma 18 *$\alpha(\bot) = \bot_2$, but there are languages L such that $\alpha(\top) \neq \top_2$.*

Proof. 1. $\bot$ We have defined $\bot = \emptyset^{\triangleright\triangleleft}$. Assume $\bot = \emptyset$; in this case, the result is obvious. Assume there is $w \in \bot$. Then for every $(x, y) \in \Sigma^* \times \Sigma^*$, $xwy \in L$. Consequently, for all $(x, y, z) \in (\Sigma^*)^3$, $xwyz \in L$. So $\alpha'[\bot]^{\blacktriangleright} = \emptyset^{\blacktriangleright}$, and $\alpha(\bot) = \bot_2$.

2. Take the language $L = a(a+b)^*a$. Then $(a, a) \in ((a+b)^*)^{\triangleright}$, where $(a+b)^* = \top$. Consequently, $(a, a, \epsilon) \in \alpha'[\top]^{\blacktriangleright}$. As $abab \notin L$, we have $(b, b) \notin \alpha'[\top]^{\blacktriangleright\blacktriangleleft} = \alpha(\top)$, hence $\alpha(\top) \neq \top_2 = \Sigma^* \times \Sigma^*$. ⊣

Again, this is easily extended to arbitrary $n \in \mathbb{N}_\omega$. So we have a serious problem, because our embedding does not preserve $\top$. We can dodge this however by considering non-standard interpretations (see section 3.2 and proof of theorem 20 below).

So this proves the first main theorem:

Theorem 19 *For every $n \in \mathbb{N}_\omega$, there is an isomorphic embedding $\alpha_n : SCL(L) \to SCL_n(L)$, such that $\alpha_n(\bot) = \bot$, and which in addition preserves infinite meets and joins.*

¿From theorem 19 it is rather easy to extend the completeness result to SCL_n:

Theorem 20 *For arbitrary $n \in \mathbb{N}_\omega$, $\Vdash_{\boldsymbol{FL}_\bot} \Gamma \vdash \alpha$ iff $SCL_n \models \Gamma \vdash \alpha$.*

Proof. Soundness is clear and follows from more general results. Regarding completeness: Assume we have $\nVdash_{\mathbf{FL}_\bot} \Gamma \vdash \gamma$. Then there is an $L, \sigma : Pr \to SCL(L)$ such that $\overline{\sigma}(\Gamma) \not\subseteq \overline{\sigma}(\gamma)$. It follows that $\alpha_n(\overline{\sigma}(\Gamma) \not\subseteq \alpha_n(\overline{\sigma}(\gamma))$, and so $SCL_n(L), \alpha_n \circ \sigma \not\models \Gamma \vdash \gamma$, which proves the claim.

But keep in mind that $\alpha_n \circ \sigma$ is a non-standard interpretation, as $\alpha_n \circ \sigma(\top)$ need not be $\top$ in the lattice! ⊣

The results obviously also hold for the logics $\mathbf{FL}, L1$ and the corresponding syntactic concept lattice reducts (for these, the notions of standard and non-standard interpretation coincide). Note also that this shows that the notion of a non-standard interpretation properly generalizes standard interpretations.

6 A Characterization for Finite SCL_ω-structures

Obviously, if $L \notin Reg$, then $SCL_\omega(L)$ is infinite; but the converse is wrong (see lemma 2). A permutation π is a map on words which preserves all cardinalities of all letters in this word. For any language L, define $\Pi(L) = \{w : \pi(v) = w$ for some permutation π and some $v \in L\}$. Let $PermReg$ be the class of languages, which is defined as follows:

Definition 21 *$L \in PermReg$, iff 1. $L \in Reg$, and 2. $\Pi(L) = L$.*

This concerns, for example, languages like $\{w : |w|_a$ is even for $a \in \Sigma\}$. We will show that $SCL_\omega(L)$ is finite iff $L \in PermReg$. For the *if*-direction, we first show the following lemma, which at the same time gives some understanding of the combinatorics of permutations:

Lemma 22 *Assume $L \neq \Pi(L)$, so there is a permutation π, $w \in L$, such that $\pi(w) \notin L$. Then there are $\overline{w}, \overline{v} \in (\Sigma^*)^\omega$ such that $\mathsf{fo}(\overline{v} \cdot \overline{w}) = w$, $\mathsf{fo}(\overline{w})\mathsf{fo}(v) = \pi(w)$.*

Note firstly that assumptions assure that $w \neq \pi(w)$. From this follows that $\mathsf{fo}(\overline{w}) \neq \epsilon \neq \mathsf{fo}(\overline{v})$, as this would entail $w = \pi(w)$.

Proof. We choose some arbitrary w, π such that $w \in L, \pi(w) \notin L$ (which exist by assumption). We let a_i denote the ith letter of $\pi(w)$. We construct $\overline{w}$ in the following fashion:

Step 1 Take a_1, the first letter of $\pi(w)$, and put $\overline{w} = (a_1, \epsilon, \epsilon, ...)$. Of course, there is $\overline{v} \in (\Sigma^*)^\omega$ such that $\mathsf{fo}(\overline{v} \cdot (a_1, \epsilon, ...)) = w$, because a_1 occurs in some place in w. Now there are two possible cases:

Case 1: $\mathsf{fo}(\overline{w})\mathsf{fo}(\overline{v}) \notin L$; then we change the "target permutation" π from the lemma to ξ, where $\xi(w) = \mathsf{fo}(\overline{w})\mathsf{fo}(\overline{v})$ (this is clearly a permutation). Then we are done, as ξ, w satisfy the claim!

Case 2: $\mathsf{fo}(\overline{w})\mathsf{fo}(\overline{v}) \in L$. In this case, we discard w, π and consider w^1, π^1 instead, where $w^1 = \mathsf{fo}(\overline{w})\mathsf{fo}(\overline{v}) \in L$, and π^1 is defined by $\pi^1(w^1) = \pi(w)$ (this works because w, w^1 are permutations of each other). Then continue with step 2.

Step 2 Having chosen a_i before, we now take a_{i+1} (as $\pi^i(w^i) = \pi(w)$, it does not matter which of the two we consider). Put $\overline{w} = (a_1...a_i, a_{i+1}, \epsilon, ...)$; there is obviously a $\overline{v}$ such that $\mathsf{fo}((a_1...a_i, a_{i+1}, \epsilon, ...) \cdot \overline{v}) = w^i$, because w^i is constructed as $a_1...a_i v$, and v necessarily contains the letter a_{i+1}. Now we can go back to the case distinction and repeat the procedure.

In the end, there are two possibilities: as w is a finite word, either at some point we hit case 1, and the claim follows. Assume we do *not* hit case 1. Then at some point we have $i = |w| = |\pi(w)|$, so we construct $w^{|w|}$ as $a_1...a_{|w|}$. Then by definition and assumption, we have $a_1...a_{|w|} = \pi(w) \notin L$. But we also have, as we do not hit case 1 by assumption, $a_1...a_{|w|} = \mathsf{fo}(\overline{w})\mathsf{fo}(\overline{v}) = w^{|w|} \in L$ – contradiction. ⊣

As we can see, we can even make sure that $\overline{w}$ has the form $(w, a, \epsilon, \epsilon, ...)$, and $\overline{v} = (v_1, v_2, v_3, \epsilon, \epsilon, ...)$.

Lemma 23 *$L \in PermReg$ if and only if $SCL_\omega(L)$ is finite.*

Proof. *Only if*: There are only finitely many non-equivalent concepts of the form $(w, \epsilon, \epsilon, ...)$. Moreover, by permutation closure, we know that if $\mathsf{fo}(\overline{w}) = \mathsf{fo}(\overline{v})$, then $\{\overline{w}\}^{\triangleright} = \{\overline{v}\}^{\triangleright}$ and the claim follows easily.

If: For this we need the previous lemma. We prove the contraposition, so assume $L \notin PermReg$. Then either $L \notin Reg$, and the claim follows easily. Or $\Pi(L) \neq L$. In this case, we have w, π such that $w \in L$, $\pi(w) \notin L$, and there are $\overline{w}, \overline{v} \in (\Sigma^*)^\omega$ such that $\mathsf{fo}(\overline{v} \cdot \overline{w}) = w$, $\mathsf{fo}(\overline{w})\mathsf{fo}(v) = \pi(w)$. Moreover, $\overline{w} = (w, a, \epsilon, \epsilon, ...)$, and $\overline{v} = (v_1, v_2, v_3, \epsilon, \epsilon, ...)$.

Now for every $n \in \mathbb{N}$, we simply take a tuple $(\overbrace{\epsilon, ..., \epsilon}^{2n \text{ times}}, w, \epsilon, \epsilon, ...)$. It is clear that for every n, we get non-equivalent tuples: We have

$$(\#)\ \mathsf{fo}((\epsilon_1, ..., \epsilon_{2n}, v_1, v_2, v_3, \epsilon, \epsilon, ...) \cdot (\epsilon_1, ..., \epsilon_{2(n-1)}, w, a, \epsilon, \epsilon, ...)) = \pi(w) \notin L,$$

whereas

$$\mathsf{fo}((\epsilon_1, ..., \epsilon_{2n}, v_1, v_2, v_3, \epsilon, \epsilon, ...) \cdot (\epsilon_1, ..., \epsilon_{2n}, w, a, \epsilon, \epsilon, ...)) = w \in L.$$

Moreover, $(\#)$ holds if in the term $2n$ is replaced by any number $m \geq 2n$. Put $\overline{w}_m = (\epsilon_1, ..., \epsilon_{2m}, w, a, \epsilon, \epsilon, ...)$. So for any $\overline{w}_m, \overline{w}_n$, if $m \neq n$, then $\{\overline{w}_m\}^{\triangleright} \neq \{\overline{w}_n\}^{\triangleright}$, and as these sets are closed and there are infinitely many of them, $SCL_\omega(L)$ is infinite. ⊣

7 Conclusion

We have shown completeness results for extensions of syntactic concepts to finite and infinite tuples; moreover, we have given a precise characterization of the class of languages which result in finite lattices in all cases. Interpreting substructural logics in sets of tuples rather than sets of strings is interesting for a number of reasons: from the perspective of categorial grammar and/or Lambek calculus as language-recognizing devices, the interpretation in tuples allows us to recognize languages which are not context-free (by letting grammars recognize tuples modulo fo). This relates more "classical" categorial approaches to new approaches such as the displacement calculus **D**, which also recognizes languages which are not context-free. In this context, infinite tuples are particularly interesting, as they allow to simulate both the "wrapping"-style extended concatenation in **D** and the "crossing"-style extended concatenation we have looked at in this paper. The usage of formal concept analysis is particularly interesting in connection with learning theory; so the results here might also be of some interest for learning beyond context-free languages.

References

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

2. Wojciech Buszkowski. Algebraic structures in categorial grammar. *Theor. Comput. Sci.*, 1998(1-2):5–24, 1998.
3. Alexander Clark. A learnable representation for syntax using residuated lattices. In Philippe de Groote, Markus Egg, and Laura Kallmeyer, editors, *Proceedings of the 14th Conference on Formal Grammar*, volume 5591 of *Lecture Notes in Computer Science*, pages 183–198. Springer, 2009.
4. Alexander Clark. Learning context free grammars with the syntactic concept lattice. In José M. Sempere and Pedro García, editors, *10th International Colloquium on Grammatical Inference*, volume 6339 of *Lecture Notes in Computer Science*, pages 38–51. Springer, 2010.
5. Alexander Clark. Logical grammars, logical theories. In Denis Béchet and Alexander Ja. Dikovsky, editors, *LACL*, volume 7351 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2012.
6. Alexander Clark. The syntactic concept lattice: Another algebraic theory of the context-free languages? *Journal of Logic and Computation*, 2013.
7. B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, Cambridge, 2 edition, 1991.
8. Maciej Farulewski. On finite models of the lambek calculus. *Studia Logica*, 80(1):63–74, 2005.
9. Nikolaos Galatos, Peter Jipsen, Tomasz Kowalski, and Hiroakira Ono. *Residuated Lattices: An Algebraic Glimpse at Substructural Logics*. Elsevier, 2007.
10. Zellig S. Harris. *Structural Linguistics*. The University of Chicago Press, 1963.
11. Makoto Kanazawa. The Lambek calculus enriched with additional connectives. *Journal of Logic, Language, and Information*, 1:141–171, 1992.
12. Makoto Kanazawa, Jens Michaelis, Sylvain Salvati, and Ryo Yoshinaka. Well-nestedness properly subsumes strict derivational minimalism. In *Logical Aspects of Computational Linguistics - 6th International Conference, LACL 2011, Montpellier, France, June 29 - July 1, 2011. Proceedings*, pages 112–128, 2011.
13. Joachim Lambek. The Mathematics of Sentence Structure. *The American Mathematical Monthly*, 65:154–169, 1958.
14. Joachim Lambek. On the calculus of syntactic types. In Roman Jakobson, editor, *Structure of Language and its Mathematical Aspects*, pages 166–178. Providence, 1961.
15. Glyn Morrill, Oriol Valentín, and Mario Fadda. The displacement calculus. *Journal of Logic, Language and Information*, 20(1):1–48, 2011.
16. Mitsuhiro Okada and Kazushige Terui. The finite model property for various fragments of intuitionistic linear logic. *J. Symb. Log.*, 64(2):790–802, 1999.
17. M. Pentus. Lambek grammars are context free. In *Proceedings of the 8th Annual IEEE Symposium on Logic in Computer Science*, pages 429–433, Los Alamitos, California, 1993. IEEE Computer Society Press.
18. A. Sestier. Contributions à une théorie ensembliste des classifications linguistiques. (Contributions to a set–theoretical theory of classifications). In *Actes du Ier Congrès de l'AFCAL*, pages 293–305, Grenoble, 1960.
19. Christian Wurm. Completeness of full lambek calculus for syntactic concept lattices. In *Formal Grammar - 17th and 18th International Conferences, FG 2012, Opole, Poland, August 2012, Revised Selected Papers, FG 2013, Düsseldorf, Germany, August 2013. Proceedings*, pages 126–141, 2012.