Displacement Logic for Grammar

Glyn Morrill & Oriol Valentín

Department of Computer Science
Universitat Politècnica de Catalunya
morrill@cs.upc.edu & oriol.valentin@gmail.com

ESSLLI 2016 Bozen-Bolzano

Lecture 4post

Some Metatheoretical Results

Some Metatheoretical Results

- The displacement hypersequent calculus **hD** has no structural rules.
- The absence of structural rules allows Morrill and Valentín (2010), Morrill et al (2011) to prove the Cut elimination theorem for **hD** by mimicking the Cut-elimination procedure provided by Lambek (1958) for the sequent calculus of the Lambek calculus (with some minor differences concerning the possibility of empty antecedents).
- **D** enjoys some nice properties such as the subformula property, decidablity, the finite reading property.
- Morrill and Valentn (2015) prove the *focalisation* property for **D** with additive connectives. As is known, focalisation (invented for linear logic by Andreoli (1992)) is a crucial property which holds for many Gentzen systems. Focalisation allows to reduce dramatically the spurious ambiguity of the proof search in sequent calculi.
- **D** is known to be NP-complete (Moot (2014)). Although S. Kuznetsov (p.c.) believes that a polynomial result can be proved in the style of the Lambek calculus, using the the measure of the order of a type.
- The unit-free fragment of **D** can be encoded in first-order linear logic (Morrill and Fadda (2008), Fadda(2010), and Moot (2014)). This allows to give a Girard style proof-net machinery for **D** (but not for additives!).

Some Metatheoretical Results

- Since we consider full displacement logic **DL**, proof-nets for multiplicative **D** are not satisfactory since **DL** heavily uses polymorphism, exponentials, and continuous and discontinuous units. NO satisfactory proof-net machinery is known for **DL**.

Displacement (Lambek) Grammars

- Given a finite vocabulary $V = \Sigma \cup \{1\}$, where $1 \notin \Sigma$, the set of prosodic strings is simply V^*.
- Define **AssignStrings** as $V^* - \{1^n | n \geq 0\}$. **AssignStrings** is the set of assignable V^* strings to a type. Intuitively, every string assigned to a type must have a *contribution* of at least one element of Σ.
- A lexicon **Lex** is a finite relation of **AssignStrings** $\times \mathcal{F}$, where each pair of *Lex* is called a *lexical assigment*, which is notated $\alpha\colon A$. In other words, a lexicon is a finite set of lexical assigments.
- Where $w\colon A \in$ **Lex**, we say that w is the *prosodic component* of $w\colon A$, and A is the *type component* of $w\colon A$.

Displacement (Lambek) Grammars

- Let Δ be a (hyper)configuration. Observe that Δ is in fact a *mixed hedge* where each internal node is either a type of sort strictly greater than 0 or a *concatenation* node. Nodes which are types have arity equal to the sort of their type, whereas concatenation nodes have unbounded arity. A labelling map is a function between the mixed hedge tree domain of Δ into **AssignStrings**.
- A *labelled hyperconfiguration* Δ^σ is pair comprising a hyperconfiguration Δ and a labelling σ of Δ. We define the *yield* of a labelled hyperconfiguration Δ^σ as follows:

(1)
$$\begin{array}{l}
\mathit{yield}(\Lambda^\sigma) = \Lambda \\
\mathit{yield}(1^\sigma) = 1 \\
\mathit{yield}((\Delta, \Gamma)^\sigma) = \mathit{yield}(\Delta^\sigma) + \mathit{yield}(\Gamma^\sigma) \\
\mathit{yield}(A^\sigma) = \sigma(A) \text{ for } A \text{ of sort } 0 \\
\mathit{yield}((A\{\Delta_1 : \cdots : \Delta_{sA}\})^\sigma) = \\
a_1 + \mathit{yield}(\Delta_1^\sigma) + a_2 + \cdots + a_{sA-1} + \mathit{yield}(\Delta_{sA}^\sigma) + a_{sA}
\end{array}$$

Where in the last line of the definition A is of sort greater than 0 and $\sigma(A)$ is $a_1 + 1 + a_2 + \cdots + a_{sA-1} + 1 + a_{sA}$.

Displacement (Lambek) Grammars

- A labelling σ of a hyperconfiguation Δ is *compatible* with a lexicon **Lex** if and only if $\sigma(A)$: $A \in$ **Lex** for every A in Δ.
- A grammar is a pair $G = (\mathbf{Lex}; S)$ where **Lex** is a lexicon and S a subtype of the type components of the lexicon. S is the target (type) symbol.
- The language of G $L(G)$ is defined as follows:

  (2) $L(\mathbf{Lex}, A) = \{yield(\Delta^{\sigma})|$ such that $\Delta \Rightarrow A$ is a theorem of **D** and σ is compatible with **Lex**$\}$

- The problem of recognition in the class of **D**-grammars is decidable.

  Proof. Since for every labelling σ compatible with a lexicon for every type A, $\sigma(A)$ contains at least one symbol of Σ ($\sigma(A) \in$ **AssignStrings**!), the set of labelled hyperconfigurations such that their yield equals a given α is finite. Now as theoremhood in the **D** is decidable we have then that the problem of recognition is decidable since it reduces to a finite number of tests of theoremhood. $\square$

On the Generative Capacity of the Core Logic **D**: Lower Bounds

The generative capacity of **D** has as lower bounds two axes of classes of languages:

- The class of *well-nested multiple context-free languages* (Wijnholds (2011) and Sorokin (2013))
- The class of the *permutation closure* of context-free languages (Morrill and Valentín 2010)

On the Generative Capacity of the Core Logic **D**: Well-Nested Multiple Context-Free Languages

- Wijnholds (2011) shows that *lexicalized well-nested range-concatenation languages* are generable by first-order displacement Lambek grammars. As a matter of fact, the class of well-nested range-concatenation languages equals the class of well-nested multiple context-free languages. In order to show this theorem this author proves a result of lexicalization of well-nested range-concatenation grammars.
- Sorokin (2013) generalises DMCFGs to an unbounded number of points of discontinuity (an infinite set of function modes of intercalation). In this way, he gives among other nomal forms a Greibach-like normal form for what he calls *displacement grammars* (not to be confused with our displacement Lambek grammars!). The Greibach normal for displacement grammars allows Sorokin to define a first-order displacement (Lambek) grammar which generates the language of displacement grammars. But, the class of (Sorokin) displacement languages equals the class of well nested multiple context-free languages.

On the Generative Capacity of the Core Logic **D**: The Class of the Permutation Closure of Context-Free Languages

- This result is obtained using a restricted fragment of the calculus. We define the set $T = \{A|\ A \text{ is an atomic type}\} \cup \{(A{\uparrow}I){\downarrow}B|\ A \text{ and } B \text{ are atomic types}\}$. A *T-hypersequent* is a hypersequent such that the types of the antecedent belong to T and the succedent is an atomic type. Note every type of T has sort 0.
- Interestingly, one can see that every provable T-hypersequent satisfies that every permutation of the antecedent preserves the provability of the hypersequent.
- To every right-linear grammar corresponds a lexicon constituted by types belonging to T.
- Invoking properties of semi-linear sets (Van Benthem (1991)), one proves that displacement (Lambek) grammars generate the permutation closure of context-free languages.

Some Examples of Formal Languages: the Copy Language

Let **Lex** contain the following lexical assignments:

$$
\begin{array}{lcl}
a & : & A, J\backslash(A\backslash S), J\backslash(S{\downarrow}(A\backslash S)) \\
b & : & B, J\backslash(B\backslash S), J\backslash(S{\downarrow}(B\backslash S))
\end{array}
$$

Where A and B are of sort 0, and S of sort 1. Let the **D** grammar $G = (\mathbf{Lex}; S \odot I)$. The target symbol is $S \odot I$. $L(G) = \{w + w | w \in \{a, b\}^+\}$. We have the following hypersequent derivation for $a + b + a + b : S \odot I$:

$$
\dfrac{1 \Rightarrow J \qquad \dfrac{\dfrac{B \Rightarrow B \qquad S\{1\} \Rightarrow S}{B, 1, \underline{J\backslash(B\backslash S)} \Rightarrow S}\,\backslash L \qquad \dfrac{A \Rightarrow A \qquad S\{1\} \Rightarrow S}{A, \underline{A\backslash S}\{1\} \Rightarrow S}\,\backslash L}{A, B, \underline{S{\downarrow}(A\backslash S)}\{1\}, J\backslash(B\backslash S) \Rightarrow S}\,{\downarrow}L}{a : A, b : B, 1, a : \underline{J\backslash(S{\downarrow}(A\backslash S))}, b : J\backslash(B\backslash S) \Rightarrow S \quad (\star)}\,\backslash L
$$

From $(\star)$ we have:

$$
\dfrac{A, B, 1, J\backslash(S{\downarrow}(A\backslash S)), J\backslash(B\backslash S) \Rightarrow S \qquad \Lambda \Rightarrow I}{a : A, b : B, \Lambda, a : J\backslash(S{\downarrow}(A\backslash S)), b : J\backslash(B\backslash S) \Rightarrow \underline{S \odot I}}\,\odot R
$$

Some Examples of Formal Languages: MIX

Recall that $MIX = \{w | w \in \{a, b, c\}^+ \text{ and } \#_a(w) = \#_b(w) = \#_c(w)\}$. Let $\mathbf{Lex} = \{a{:}\check{}S_1{\downarrow}S, b{:}\check{}S_2{\downarrow}S_1, c{:}\, S_2, c{:}\check{}S{\downarrow}S_2\}$. Let $G = (\mathbf{Lex}; S)$. We have $L(G) = MIX$. A sample of a derivation of $c + a + b + a + c + b$:

$$\dfrac{1, S_2 \Rightarrow \check{}S_2 \qquad S_1 \Rightarrow S_1}{\Lambda, \check{}S_2{\downarrow}S_1, S_2 \Rightarrow S_1}\,{\downarrow}L$$

$$\dfrac{\Lambda, \check{}S_2{\downarrow}S_1, S_2 \Rightarrow S_1}{1, \check{}S_2{\downarrow}S_1, S_2 \Rightarrow \check{}S_1}\,\check{}R$$

$$\dfrac{1, \check{}S_2{\downarrow}S_1, S_2 \Rightarrow \check{}S_1 \qquad S \Rightarrow S}{\Lambda, \check{}S_1{\downarrow}S, \check{}S_2{\downarrow}S_1, S_2 \Rightarrow S}\,{\downarrow}L$$

$$\dfrac{\Lambda, \check{}S_1{\downarrow}S, \check{}S_2{\downarrow}S_1, S_2 \Rightarrow S}{1, \check{}S_1{\downarrow}S, \check{}S_2{\downarrow}S_1, S_2 \Rightarrow \check{}S}\,\check{}R$$

$$\dfrac{1, \check{}S_1{\downarrow}S, \check{}S_2{\downarrow}S_1, S_2 \Rightarrow \check{}S \qquad S_2 \Rightarrow S_2}{\check{}S{\downarrow}S_2, \check{}S_1{\downarrow}S, \check{}S_2{\downarrow}S_1, S_2, \Lambda \Rightarrow S_2}\,{\downarrow}L$$

$$\dfrac{\check{}S{\downarrow}S_2, \check{}S_1{\downarrow}S, \check{}S_2{\downarrow}S_1, S_2, \Lambda \Rightarrow S_2}{\check{}S{\downarrow}S_2, \check{}S_1{\downarrow}S, \check{}S_2{\downarrow}S_1, S_2, 1 \Rightarrow \check{}S_2}\,\check{}R$$

$$\dfrac{\check{}S{\downarrow}S_2, \check{}S_1{\downarrow}S, \check{}S_2{\downarrow}S_1, S_2, 1 \Rightarrow \check{}S_2 \qquad S_1 \Rightarrow S_1}{\check{}S{\downarrow}S_2, \check{}S_1{\downarrow}S, \check{}S_2{\downarrow}S_1, \Lambda, S_2, \check{}S_2{\downarrow}S_1 \Rightarrow S_1}$$

$$\dfrac{\check{}S{\downarrow}S_2, \check{}S_1{\downarrow}S, \check{}S_2{\downarrow}S_1, \Lambda, S_2, \check{}S_2{\downarrow}S_1 \Rightarrow S_1}{\check{}S{\downarrow}S_2, \check{}S_1{\downarrow}S, \check{}S_2{\downarrow}S_1, 1, S_2, \check{}S_2{\downarrow}S_1 \Rightarrow \check{}S_1}\,\check{}R$$

$$\dfrac{\check{}S{\downarrow}S_2, \check{}S_1{\downarrow}S, \check{}S_2{\downarrow}S_1, 1, S_2, \check{}S_2{\downarrow}S_1 \Rightarrow \check{}S_1 \qquad S \Rightarrow S}{c{:}\check{}S{\downarrow}S_2, a{:}\check{}S_1{\downarrow}S, b{:}\check{}S_2{\downarrow}S_1, a{:}\check{}S_1{\downarrow}S, c{:}\, S_2, b{:}\check{}S_2{\downarrow}S_1 \Rightarrow S}\,{\downarrow}L$$

Towards Algebraic Semantics

- **D** is model-theoretically motivated, and the key to its conception is the use of many-sorted universal algebra (Goguen and Meseguer (1985)), namely ω-sorted universal algebra.
- Here, we assume a version of many-sorted algebra such that the sort domains of an ω-sorted algebra $\mathcal{A}$ are non-empty. With this condition we avoid some pathologies which arise in a naïve version of many-sorted universal algebra (Goguen and Meseguer (1985), Lalement (1991)).
- In the naïve version of many-sorted universal algebra the completeness theorem of many-sorted equational logic does not hold!

Towards Algebraic Semantics

- Consider the ω-sorted signature $\Sigma_D = (+, \{\times_i\}_{i>0}, 0, 1)$ with sort functionalities $((i, j \to i + j)_{i,j \geq 0}, (i, j \to i + j - 1)_{i>0, j \geq 0}, 0, 1)$. Displacement algebras (DAs) for **D** have this signature.
- The ω-sorted signature for *residuated* DAs is $\Sigma_D^{Res} = (+, \backslash\backslash, //, \{\times_i\}_{i>0}, \{\uparrow\uparrow_i\}_{i>0}, \{\downarrow\downarrow_i\}_{i>0}, 0, 1)$ with sort functionalities:

$$((i, j \to i + j)_{i,j \geq 0}, (i, i + j \to j), (j, i + j \to i), (i, j \to i + j - 1)_{i>0, j \geq 0}, (i + j, j \to i + 1), (i + 1, j \to i + j), 0, 1)$$

.

Residuated DAs

A residuated DA $\mathcal{A}$ is a Σ_D^{Res} algebra such that

- The Σ_D-reduct of $\mathcal{A}$ is a DA
- The $(+, //, \backslash\backslash)$ forms a residuated triple
- For every $i > 0$, $(\times_i, \uparrow\uparrow_i, \downarrow\downarrow_i)$ forms a residuated triple

Displacement Models

- Consider the Σ_D^{Res} $\mathcal{F}$ algebra of **D** types. Let $\mathcal{PR}$ be the set of ω-sorted primitive types.
- A model $\mathcal{M} = (\mathcal{A}, v)$ comprises a residuated DA and a ω-sorted mapping $v : \mathcal{PR} \to \mathcal{F}$ called a valuation. The mapping $\widehat{v}$ is the unique Σ_D^{Res}-morphism which extends v in such a way that:

$$
\text{(3)} \quad \begin{array}{lcll}
\widehat{v}(A * B) & = & \widehat{v}(A) * \widehat{v}(B) & \text{if } * \text{ is a binary connective} \\
\widehat{v}(I) & = & 0^{\mathcal{A}} & \\
\widehat{v}(J) & = & 1^{\mathcal{A}} &
\end{array}
$$

- Needless to say, the mappings v and $\widehat{v}$ preserve the sorting regime.

The (very) First Step towards Algebraic Semantics

- The Lindenbaum-Tarski construction in algebraic semantics (Font et al (2003))
- This classical construction leads to the strong completeness of **D** w.r.t. the class of residuated DAs

Some Special Residuated DAs

- Since the class of DAs form a variety, it is closed by subalgebras, direct products and homomorphic images, which give additional DAs, in which we can consider residuation.
- We have other interesting examples of DAs, for instance the *powerset DA over* a DA $\mathcal{A} = (A, +, \{\times_i\}_{i>0}, 0, 1)$, which we denote $\mathcal{P}(\mathcal{A})$. We have:

  (4) $\mathcal{P}(\mathcal{A}) = (\mathcal{P}(\mathcal{A}), \cdot, \{\circ_i\}_{i>0}, \mathbb{I}, \mathbb{J})$

  The notation of the carrier set of $\mathcal{P}(\mathcal{A})$ presupposes that its members are same-sort subsets; notice that $\emptyset$ vacuously satisfies the *same-sort* condition.

  It is readily seen that for every $\mathcal{A}$, $\mathcal{P}(\mathcal{A})$ is in fact a DA. Notice that every sort domain $\mathcal{P}(\mathcal{A})_i$ is a collection of same-sort subsets.
- The continuous and discontinuous residuals are naturally induced by the powerset operations

Some Special Completeness results

Consider the so-called *implicative fragment*, which we denote $\mathbf{D}[\rightarrow]$. This fragment comprises the continuous and discontinuous implications, the non-deterministic discontinuous connectives, and the (synthetic) unary connectives $\check{}_k$ and *projections* $(\triangleleft^{-1}, \triangleright^{-1})$.

- Projections can simplify the account of cross-serial dependencies in Dutch.
- The nondeterministic discontinous implications $(\Uparrow, \Downarrow)$ can be used to account for particle shift nondeterminism where the object can be intercalated between the verb and the particle, or after the particle. For a particle verb like *call* $+\ 1\ +$ *up* we can give the lexical assigment $\triangleleft^{-1}(\check{}_1(N\backslash S) \Uparrow N)$.
- The split connective can be used for parentheticals like *fortunately* with the type assignment $\check{}_1 S{\downarrow_1}S$.

Some Special Completeness results

- **D**[$\rightarrow$] is strongly complete w.r.t. the so-called *free separated monoids* (Valentín (2016)).
- **D**[$\rightarrow$] without the *split* connectives are strongly complete w.r.t. the so-called *language models* (ibid).
- In fact, the last result is true of language models with exacly three generators, one of them being of course the separator (ibid).
- **D** is strongly complete over residuated powerset residuated DAs over DAs, via a representation theorem à la Buszkowski (1997) (to be submitted).