Polarities in logic and semantics

Arno Bastenhof

Utrecht University

1 Introduction

Categorial type logics (CTL, [10]) attempt the reduction of natural language grammar to proof theory. Thus, syntactic categories are replaced with formulas, and a sentence's derivability is identified with the provability of its grammaticality. CTL may be considered a realization of Montague's Universal Grammar program ([9]), in the sense of offering a mathematical framework for studying both the syntax and semantics of natural and formal languages, in accordance with the principle of compositionality. In particular, CTL benefits of the close correspondence between the logician's formulas and the computer scientist's types, as cemented in the Curry-Howard isomorphism.

In practice, CTL serves as an umbrella term covering a plethora of logical calculi, each more or less agreeing on a fixed context-free core, but diverging in their explorations into the expressivity realms beyond. Still, one typically observes a shared asymmetry inherited from Lambek's founding work [7]. Roughly, derivability is considered a relation between a possible multitude of hypotheses (the categories, or formulas, assigned to the individual words) and a unique conclusion (the formula categorizing the phrase made up from the hypotheses).

Proposals for restoring symmetry are of more recent times, inspired by Girard's classical linear logic. While gaining in mathematical elegance, little improvement has been claimed towards strengthening linguistic coverage.[1] The current paper challenges this situation, treating scopal ambiguities. Typically, their categorial analyses involve non-context-free mechanisms, realized by controlled use of structural rules, or relax compositionality to a mapping of derivations into non-singleton sets of readings. In contrast, we show that once symmetry is embraced, already within the most structure-sensitive of logics (rejecting both commutativity and associativity) all combinatorially possible scopal readings may be accounted for, even when keeping to a unique reading for each derivation. We use a variation on *Classical Non-associative Lambek calculus* (**CNL**, [3]), dubbed 'polarized **CNL**', or simply $\mathbf{CNL}^{pol}$. While ensured equivalent provability-wise with **CNL**, $\mathbf{CNL}^{pol}$ makes a finer distinction between proofs, mapped in turn to separate readings. The concept of polarity, our main ingredient, originates in Girard's constructivization of classical logic ([5]), recognizable by computer scientists as a continuation-passing style (CPS) translation.

We proceed as follows. After introducing $\mathbf{CNL}^{pol}$ in §2, we motivate its applicability to scopal ambiguities in §3 by example. To gain a more precise

[1] See [11] for an exception, and §5 for a comparison with the approach discussed here.

understanding of the flexibility of our approach, we consider in §4 Hendriks' type-shifting rules ([6]), showing their derivability within $\mathbf{CNL}^{pol}$. §5 concludes with an evaluation, including discussion detailing the limitations of our work.

2 Polarized classical non-associative Lambek calculus

Classical non-associative Lambek calculus was introduced by de Groote and Lamarche ([3]) as a classical conservative extension of Lambek's non-associative syntactic calculus (**NL**, [8]). 'Classicality' here refers to the existence of a linear negation $A^{\perp}$ on formulas A satisfying involutivity ($A^{\perp\perp} = A$). Besides the usual multiplicative conjunctions, or *tensors* ($A \otimes B$), their dual disjunctions, or *pars* ($A \oplus B$) now arise through the De Morgan laws as $(B^{\perp} \otimes A^{\perp})^{\perp}$. Consequently, direction-sensitive implications (A/B) and ($B\backslash A$) need no longer be considered primitive, but may rather be defined in terms of the par and negation by ($A \oplus B^{\perp}$) and ($B^{\perp} \oplus A$) respectively. It was shown in [3] that **CNL** faithfully embeds **NL**.

We adapt **CNL** according to Girard's constructive interpretation of (traditional) classical logic ([5]), dubbing the result *polarized* **CNL**. Roughly, formulas are partitioned into those considered *positive(ly polar)* and *negative(ly polar)*, with two additional connectives, the *shifts* $\downarrow$ and $\uparrow$ mediating between the two. Polarity is interchanged through $\cdot^{\perp}$, lending proof-theoretic support. Intuitively, the shifts may be considered 'semantic annotations', recording the locations for inserting negations in the image of semantic interpretation. The following definition details the formula language, explicating the concept of polarity.

Definition 1. Positive formulas ($P, Q, \ldots$) and their negative duals ($M, N, \ldots$) are defined by mutual induction, starting from a countable set of atoms $p, q, \ldots$.

$$\begin{array}{rcll} P, Q & ::= & p \mid (P \otimes Q) \mid (\downarrow N) & \text{(Positive(ly polar) formulas)} \\ M, N & ::= & \bar{p} \mid (M \oplus N) \mid (\uparrow P) & \text{(Negative(ly polar) formulas)} \end{array}$$

Linear negation $\cdot^{\perp}$ acts as primitive on atoms only (written $\bar{\cdot}$), while extending to complex formulas through De Morgan's laws: $p^{\perp} = \bar{p}$, $\bar{p}^{\perp} = p$, $(P \otimes Q)^{\perp} = Q^{\perp} \oplus P^{\perp}$, $(M \oplus N)^{\perp} = N^{\perp} \otimes M^{\perp}$, $(\downarrow N)^{\perp} = \uparrow N^{\perp}$ and $(\uparrow P)^{\perp} = \downarrow P^{\perp}$.

In practice, we stick to atomic formulas s (categorizing sentences), np (noun phrases) and n (common nouns). For the target language of semantic interpretation, we take simply-typed λ-calculus with base types e (for individuals) and t (truth values). Besides the familiar *function types* ($\sigma \to \tau$) (written $\langle \sigma, \tau \rangle$ in [4]), semantic types σ, τ may include *products* ($\sigma \times \tau$). In mapping formulas to types, function types are restricted to ($\sigma \to t$), abbreviated $\neg\sigma$.

Definition 2. We associate, by mutual induction, formulas P, N with types $[\![P]\!]^+, [\![N]\!]^-$. At the level of atoms, we set $[\![s]\!]^+ = [\![\bar{s}]\!]^- = t$ (sentences interpret by truth values), $[\![np]\!]^+ = [\![\bar{np}]\!]^- = e$ (noun phrases by entities) and $[\![n]\!]^+ = [\![\bar{n}]\!]^- = \neg e$ (common nouns by first-order properties), while for complex types,

$$\begin{array}{cc} [\![P \otimes Q]\!]^+ = [\![P]\!]^+ \times [\![Q]\!]^+ & [\![M \oplus N]\!]^- = [\![N]\!]^- \times [\![M]\!]^- \\ [\![\downarrow N]\!]^+ = \neg[\![N]\!]^- & [\![\uparrow P]\!]^- = \neg[\![P]\!]^+ \end{array}$$

$$\frac{\Delta \vdash s' : P^{\perp} \quad \Gamma, x : P \vdash s}{\Gamma, \Delta \vdash s[s'/x]}\ Cut \qquad \frac{}{x : A \vdash x : A^{\perp}}\ Ax$$

$$\frac{\Gamma, \Delta \vdash s}{\Delta, \Gamma \vdash s}\ dp^1 \qquad \frac{\Gamma \bullet \Delta, \Theta \vdash s}{\Gamma, \Delta \bullet \Theta \vdash s}\ dp^2$$

$$\frac{\Gamma \vdash s : N}{\Gamma, \downarrow N^x \vdash (x\ s)}\ \downarrow \qquad \frac{\Gamma, P^x \vdash s}{\Gamma \vdash \lambda x^{[\![P]\!]^+} s : \uparrow P}\ \uparrow$$

$$\frac{\Gamma, P^y \bullet Q^z \vdash s}{\Gamma, P \otimes Q^x \vdash s[\pi^1(x)/y, \pi^2(x)/z]}\ \otimes \qquad \frac{\Gamma \vdash s : M \quad \Delta \vdash s' : N}{\Delta \bullet \Gamma \vdash \langle s', s \rangle : M \oplus N}\ \oplus$$

Fig. 1. Polarized classical non-associative Lambek calculus.

Derivability establishes pairings of binary-branching trees Γ, Δ of formulas, referred to by *structures*. Their leaves carry positive formulas, negative N appearing only when prefixed by $\downarrow$. Furthermore, in anticipation of the term assignment to derivations, formulas are labeled using (λ-)variables $x, y, \ldots$

Definition 3. Structures Γ, Δ collect pairings P^x into binary-branching trees:

$$\Gamma, \Delta ::= P^x \mid (\Gamma \bullet \Delta)$$

Proofs involve derivability judgements $\Gamma, \Delta \vdash s$ and $\Gamma \vdash s : N$, with s a λ-term of type t in the first case, and of type $[\![N]\!]^-$ in the latter, while containing free variables x of type $[\![P]\!]^+$ for each P^x in $\Gamma(, \Delta)$. Compared to [3], we have chosen an unconventional left-sided sequent notation, better reflecting the intuitionistic nature of the semantic target language.

Definition 4. Figure 1 details the mutually defined derivability judgements $\Gamma, \Delta \vdash s$ and $\Gamma \vdash s : N$. Double inference lines are used in (dp^2) to indicate that derivability goes in both directions.

We make the following observations regarding the above definition.

1. The rules (dp^1) and (dp^2) bear resemblance to Belnap's display postulates ([2]), in allowing one to isolate the main formula of a logical inference as the whole of one of the sequent's components by turning its context 'inside out'. They are not to be mistaken for associativity and commutativity postulates.
2. Compared to **CNL**, **CNL**^{pol} exploits both sides of the turnstile symbol $\vdash$, though treating them asymmetrically in order to restore the tight connection with λ-calculus. The newly added shifts serve solely to control traffic across $\vdash$. Provability-wise, these are harmless extensions over **CNL**, as argued for below in Theorems 6 and 7. When considering the actual proofs, however, shifts, together with the display postulates, offer a means of creating abstractions over any of the variables found in a sequent without compromising structure-sensitivity. In contrast, **NL** severely restricts the abstraction mechanism, accounting for its limited semantic expressivity.

Theorem 5. Cut is admissible, as are non-atomic instantiations of axioms.

Theorem 6. Under the forgetful translation, removing shifts and writing all sequents one-sided, derivations in $\mathbf{CNL}^{pol}$ map into those of **CNL**.

Theorem 7. There exists a decoration of **CNL**'s formulas by shifts relative to which derivations can be translated into $\mathbf{CNL}^{pol}$.

The previous theorems, proven in [1], ensure equivalence, provability-wise, of **CNL** and $\mathbf{CNL}^{pol}$. The proof of Theorem 7 in particular bears resemblance to a continuation-passing style (CPS) translation, as further elaborated upon in [1].

3 Case analyses

We illustrate the versatility of our approach through a survey of several classical examples. As further motivated in §4, the underlying generalization is that any combinatorially possible scopal reading may be accounted for through generous use of shifts inside the lexicon. Conversely, parsimonious use allows for certain readings to be blocked (§3.4), although this approach has its limitations (cf. §5). Below, we first provide a sample lexicon (§3.1), through which we can explain object-wide scope readings (§3.2, as a warm-up), embedded scope (§3.3), scope sieves (§3.4), and, finally, intensionality and coordination (§3.5).

3.1 Lexicon

A lexicon associates words w with formulas P and semantic interpretation a term s of type $[\![P]\!]^+$, and is usually written informally as a collection of sequents $w \vdash s : P^\perp$ (noting $[\![P]\!]^+ = [\![P^\perp]\!]^-$, by a routine induction). If this notation is taken seriously, allowing in particular words at the leaves of structures, lexical insertion into a derivation proceeds through Cut. In defining the lexical denotations s, we allow recourse to the usual predicate-logical connectives, considered constants $\wedge$, $\vee$, $\supset$ (of types $t \to (t \to t)$), $\forall$ and $\exists$ (of types $(e \to t) \to t$), with the binary operations written in the usual infix style. Here, $\supset$ refers to implication, its notation meant to prevent confusion with the type-forming operator $\to$.

F.2 presents a sample lexicon, using constants ALICE (of type e), PERSON, UNICORN, YAWN $(e \to t)$, FIND $(e \to (e \to t))$, SEEK $(((e \to t) \to t) \to (e \to t))$, THINK and HEAR $(t \to (e \to t))$. We have written (N/P) and $(P\backslash N)$ for $(N \oplus P^\perp)$ and $(P^\perp \oplus N)$ respectively, to facilitate comparison with traditional CTL. Finally, we use paired abstractions $\lambda\langle x, y\rangle s$ as abbreviations for $\lambda z s[\pi^1(z)/x, \pi^2(z)/y]$.

3.2 Object-wide scope

As a warm-up, consider the familiar linear and inverse scope readings (1a), (1b)

1. Everyone found a unicorn.
1a. For every person x, there exists some unicorn y s.t. x found y.
1b. There exists some unicorn y s.t. for every person x, x found y.

$$
\begin{array}{rl}
\text{Alice} \vdash & \textsc{alice} : np^{\perp} \\
\text{everyone} \vdash & \lambda Q \forall x((\textsc{person}\ x) \supset (Q\ x)) : (\downarrow\uparrow np)^{\perp} \\
\text{a} \vdash & \lambda\langle P, Q\rangle \exists x((P\ x) \wedge (Q\ x)) : (\downarrow(\uparrow np/n))^{\perp} \\
\text{unicorn} \vdash & \textsc{unicorn} : n^{\perp} \\
\text{yawn(ed)} \vdash & \lambda\langle q, x\rangle(q\ (\textsc{yawn}\ x)) : (\downarrow(np\backslash\uparrow s))^{\perp} \\
\text{found} \vdash & \lambda\langle y, \langle q, x\rangle\rangle(q\ ((\textsc{find}\ y)\ x)) : (\downarrow((np\backslash\uparrow s)/np))^{\perp} \\
\text{sought} \vdash & \lambda\langle Y, \langle q, x\rangle\rangle(q\ ((\textsc{seek}\ Y)\ x)) : (\downarrow((np\backslash\uparrow s)/\downarrow\uparrow np))^{\perp} \\
\text{thinks} \vdash & \lambda\langle Q, \langle q, x\rangle\rangle(q\ ((\textsc{think}\ (Q\ \lambda p p))\ x)) : (\downarrow((np\backslash\uparrow s)/\downarrow\uparrow s))^{\perp} \\
\text{heard} \vdash & \lambda\langle p, \langle q, x\rangle\rangle(q\ ((\textsc{hear}\ p)\ x)) : ((np\backslash\uparrow s)/s)^{\perp} \\
\text{and} \vdash & \lambda\langle R, \langle\langle Y, \langle q, x\rangle\rangle, S\rangle\rangle((S\ \langle Y, \langle q, x\rangle\rangle) \wedge (R\ \langle Y, \langle q, x\rangle\rangle)) \\
& : (\downarrow((\downarrow tv\backslash tv)/\downarrow tv))^{\perp}
\end{array}
$$

Fig. 2. Sample lexicon, where tv abbreviates $(np\backslash(\uparrow s/\downarrow\uparrow np))$.

The formulas assigned to the quantified noun phrases involved reflect their interpretation as second-order properties through the use of double shifts, inserting double negations at the level of types (i.e., implications with result type t). We first construct the following derivations, using the formulas found in F.2 (cf. F.3)

$$
\begin{array}{r}
\downarrow\uparrow np^{x} \bullet (\downarrow((np\backslash\uparrow s)/np)^{y} \bullet (\downarrow(\uparrow np/n)^{u} \bullet n^{v})), \downarrow \bar{s}^{z} \\
\vdash \begin{cases} (x\ \lambda a(u\ \langle v, \lambda b(y\ \langle b, \langle z, a\rangle\rangle)\rangle)) \\ (u\ \langle v, \lambda b(x\ \lambda a(y\ \langle b, \langle z, a\rangle\rangle))\rangle) \end{cases}
\end{array}
$$

with lexical insertion (through Cut) deriving, after β-conversion, the desired

$$
\begin{array}{r}
\text{everyone} \bullet (\text{found} \bullet (\text{a} \bullet \text{unicorn})), \downarrow \bar{s}^{z} \\
\vdash \begin{cases} \forall x((\textsc{person}\ x) \supset \exists y((\textsc{unicorn}\ y) \wedge (z\ ((\textsc{find}\ y)\ x)))) & (1a) \\ \exists y((\textsc{unicorn}\ y) \wedge \forall x((\textsc{person}\ x) \supset (z\ ((\textsc{find}\ y)\ x)))) & (1b) \end{cases}
\end{array}
$$

noting the presence of a free variable z for the resulting category. If desired, one may apply $(\uparrow)$ to create an abstraction, deriving $\uparrow\downarrow\bar{s}$.

3.3 Embedded scope

We illustrate non-local scope construal from inside a complement clause:

2. Alice thinks a unicorn yawned.

2a. Alice thinks there exists a unicorn y s.t. y yawned.

2b. For some unicorn y, Alice thinks y yawned.

By having the verb select for a clausal complement of category $\downarrow\uparrow s$ instead of s, we provide the deductive machinery with enough freedom to construct both of the desired readings. Using the formulas found in F.2, we derive three terms,

$$
\begin{array}{r}
np^{a} \bullet (\downarrow((np\backslash\uparrow s)/\downarrow\uparrow s)^{t} \bullet ((\downarrow(\uparrow np/n)^{u} \bullet n^{v}) \bullet \downarrow(np\backslash\uparrow s)^{w})), \downarrow \bar{s}^{z} \\
\vdash \begin{cases} (t\ \langle\lambda q(u\ \langle v, \lambda y(w\ \langle q, y\rangle)\rangle), \langle z, a\rangle\rangle) \\ (u\ \langle v, \lambda y(t\ \langle\lambda q(w\ \langle q, y\rangle), \langle z, a\rangle\rangle)\rangle) \\ (u\ \langle v, \lambda y(w\ \langle\lambda p(t\ \langle\lambda q(q\ p), \langle z, a\rangle\rangle), y\rangle)\rangle) \end{cases}
\end{array}
$$

collapsed into two readings after lexical insertion:

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{Ax}
\UnaryInfC{$np^a \vdash a : \bar{np}$}
\AxiomC{}
\RightLabel{Ax}
\UnaryInfC{$\downarrow \bar{s}^z \vdash z : \uparrow s$}
\RightLabel{$\oplus$}
\BinaryInfC{$\downarrow \bar{s}^z \bullet np^a \vdash \langle z, a\rangle : np\backslash \uparrow s$}
\AxiomC{}
\RightLabel{Ax}
\UnaryInfC{$np^b \vdash b : \bar{np}$}
\RightLabel{$\oplus$}
\BinaryInfC{$np^b \bullet (\downarrow \bar{s}^z \bullet np^a) \vdash \langle b, \langle z, a\rangle\rangle : (np\backslash \uparrow s)/np$}
\RightLabel{$\downarrow$}
\UnaryInfC{$np^b \bullet (\downarrow \bar{s}^z \bullet np^a), \downarrow((np\backslash \uparrow s)/np)^y \vdash (y\ \langle b, \langle z, a\rangle\rangle)$}
\doubleLine
\RightLabel{Dp}
\UnaryInfC{$(\downarrow \bar{s}^z \bullet np^a) \bullet \downarrow((np\backslash \uparrow s)/np)^y, np^b \vdash (y\ \langle b, \langle z, a\rangle\rangle)$}
\RightLabel{$\uparrow$}
\UnaryInfC{$(\downarrow \bar{s}^z \bullet np^a) \bullet \downarrow((np\backslash \uparrow s)/np)^y \vdash \lambda b(y\ \langle b, \langle z, a\rangle\rangle) : \uparrow np$}
\AxiomC{}
\RightLabel{Ax}
\UnaryInfC{$n^v \vdash v : \bar{n}$}
\RightLabel{$\oplus$}
\BinaryInfC{$n^v \bullet ((\downarrow \bar{s}^z \bullet np^a) \bullet \downarrow((np\backslash \uparrow s)/np)^y) \vdash \langle v, \lambda b(y\ \langle b, \langle z, a\rangle\rangle)\rangle : \uparrow np/n$}
\RightLabel{$\downarrow$}
\UnaryInfC{$n^v \bullet ((\downarrow \bar{s}^z \bullet np^a) \bullet \downarrow((np\backslash \uparrow s)/np)^y), \downarrow(\uparrow np/n)^u \vdash (u\ \langle v, \lambda b(y\ \langle b, \langle z, a\rangle\rangle)\rangle)$}
\doubleLine
\RightLabel{Dp}
\UnaryInfC{$(\downarrow((np\backslash \uparrow s)/np)^y \bullet (\downarrow(\uparrow np/n)^u \bullet n^v)) \bullet \downarrow \bar{s}^z, np^a \vdash (u\ \langle v, \lambda b(y\ \langle b, \langle z, a\rangle\rangle)\rangle)$}
\RightLabel{$\uparrow$}
\UnaryInfC{$(\downarrow((np\backslash \uparrow s)/np)^y \bullet (\downarrow(\uparrow np/n)^u \bullet n^v)) \bullet \downarrow \bar{s}^z \vdash \lambda a(u\ \langle v, \lambda b(y\ \langle b, \langle z, a\rangle\rangle)\rangle) : \uparrow np$}
\RightLabel{$\downarrow$}
\UnaryInfC{$(\downarrow((np\backslash \uparrow s)/np)^y \bullet (\downarrow(\uparrow np/n)^u \bullet n^v)) \bullet \downarrow \bar{s}^z, \downarrow \uparrow np^x \vdash (x\ \lambda a(u\ \langle v, \lambda b(y\ \langle b, \langle z, a\rangle\rangle)\rangle))$}
\doubleLine
\RightLabel{Dp}
\UnaryInfC{$\downarrow \uparrow np^x \bullet (\downarrow((np\backslash \uparrow s)/np)^y \bullet (\downarrow(\uparrow np/n)^u \bullet n^v)), \downarrow \bar{s}^z \vdash (x\ \lambda a(u\ \langle v, \lambda b(y\ \langle b, \langle z, a\rangle\rangle)\rangle))$}
\end{prooftree}
$$

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{Ax}
\UnaryInfC{$np^a \vdash a : \bar{np}$}
\AxiomC{}
\RightLabel{Ax}
\UnaryInfC{$\downarrow \bar{s}^z \vdash z : \uparrow s$}
\RightLabel{$\oplus$}
\BinaryInfC{$\downarrow \bar{s}^z \bullet np^a \vdash \langle z, a\rangle : np\backslash \uparrow s$}
\AxiomC{}
\RightLabel{Ax}
\UnaryInfC{$np^b \vdash b : \bar{np}$}
\RightLabel{$\oplus$}
\BinaryInfC{$np^b \bullet (\downarrow \bar{s}^z \bullet np^a) \vdash \langle b, \langle z, a\rangle\rangle : (np\backslash \uparrow s)/np$}
\RightLabel{$\downarrow$}
\UnaryInfC{$np^b \bullet (\downarrow \bar{s}^z \bullet np^a), \downarrow((np\backslash \uparrow s)/np)^y \vdash (y\ \langle b, \langle z, a\rangle\rangle)$}
\doubleLine
\RightLabel{Dp}
\UnaryInfC{$(\downarrow((np\backslash \uparrow s)/np)^y \bullet np^b) \bullet \downarrow \bar{s}^z, np^a \vdash (y\ \langle b, \langle z, a\rangle\rangle)$}
\RightLabel{$\uparrow$}
\UnaryInfC{$(\downarrow((np\backslash \uparrow s)/np)^y \bullet np^b) \bullet \downarrow \bar{s}^z \vdash \lambda a(y\ \langle b, \langle z, a\rangle\rangle) : \uparrow np$}
\RightLabel{$\downarrow$}
\UnaryInfC{$(\downarrow((np\backslash \uparrow s)/np)^y \bullet np^b) \bullet \downarrow \bar{s}^z, \downarrow \uparrow np^x \vdash (x\ \lambda a(y\ \langle b, \langle z, a\rangle\rangle))$}
\doubleLine
\RightLabel{Dp}
\UnaryInfC{$(\downarrow \bar{s}^z \bullet \downarrow \uparrow np^x) \bullet \downarrow((np\backslash \uparrow s)/np)^y, np^b \vdash (x\ \lambda a(y\ \langle b, \langle z, a\rangle\rangle))$}
\RightLabel{$\uparrow$}
\UnaryInfC{$(\downarrow \bar{s}^z \bullet \downarrow \uparrow np^x) \bullet \downarrow((np\backslash \uparrow s)/np)^y \vdash \lambda b(x\ \lambda a(y\ \langle b, \langle z, a\rangle\rangle)) : \uparrow np$}
\AxiomC{}
\RightLabel{Ax}
\UnaryInfC{$n^v \vdash v : \bar{n}$}
\RightLabel{$\oplus$}
\BinaryInfC{$n^v \bullet ((\downarrow \bar{s}^z \bullet \downarrow \uparrow np^x) \bullet \downarrow((np\backslash \uparrow s)/np)^y) \vdash \langle v, \lambda b(x\ \lambda a(y\ \langle b, \langle z, a\rangle\rangle))\rangle : \uparrow np/n$}
\RightLabel{$\downarrow$}
\UnaryInfC{$n^v \bullet ((\downarrow \bar{s}^z \bullet \downarrow \uparrow np^x) \bullet \downarrow((np\backslash \uparrow s)/np)^y), \downarrow(\uparrow np/n)^u \vdash (u\ \langle v, \lambda b(x\ \lambda a(y\ \langle b, \langle z, a\rangle\rangle))\rangle)$}
\doubleLine
\RightLabel{Dp}
\UnaryInfC{$\downarrow \uparrow np^x \bullet (\downarrow((np\backslash \uparrow s)/np)^y \bullet (\downarrow(\uparrow np/n)^u \bullet n^v)), \downarrow \bar{s}^z \vdash (u\ \langle v, \lambda b(x\ \lambda a(y\ \langle b, \langle z, a\rangle\rangle))\rangle)$}
\end{prooftree}
$$

Fig. 3. Deriving *everyone found a unicorn.* (Dp) abbreviates multiple (dp^2, dp^1)-steps.

$$
\begin{array}{l}
\qquad\qquad\text{Alice} \bullet (\text{thinks} \bullet ((\text{a} \bullet \text{unicorn}) \bullet \text{yawned})), \downarrow \bar{s}^z \\
\vdash \begin{cases} (z\ ((\textsc{think}\ \exists y((\textsc{unicorn}\ y) \wedge (\textsc{yawn}\ y)))\ \textsc{alice}))\ (2a) \\ \exists y((\textsc{unicorn}\ y) \wedge (z\ ((\textsc{think}\ (\textsc{yawn}\ y))\ \textsc{alice})))\ (2b) \end{cases}
\end{array}
$$

3.4 Scope sieves

Further insight into the categorization of verbs selecting for clausal complements is provided by the following example, illustrating scope sieves. Here, non-local scope is enforced, as typically observed with perception verbs (cf. [6], p.108):

3. Alice heard a unicorn yawn.

The desired result obtains by having the matrix verb select for s instead of $\downarrow\uparrow s$. For reasons of space, we only show the end result obtained after lexical insertion.

$$\begin{array}{l} \text{Alice} \bullet (\text{heard} \bullet ((\text{a} \bullet \text{unicorn}) \bullet \text{yawn})), \downarrow \bar{s}^z \\ \vdash \exists y((\textsc{unicorn}\ y) \wedge (z\ ((\textsc{hear}\ y)\ \textsc{alice}))) \end{array}$$

On the other hand, it does not seem clear how to enforce the local reading (e.g., as needed for scope islands). See §5 for further discussion.

3.5 Coordination and intensionality

We conclude with intensionality and coordination, treated together to show their interplay. Our treatment of intensionality is simplified w.r.t. our choice of semantic type for interpreting sentences, although this can be easily remedied if the reader so desires. Furthermore, *found* is treated extensionally, in line with [4].

4. Alice sought and found a unicorn.
4a. Alice sought a unicorn, and there exists a unicorn y s.t. Alice found y.
4b. There exists a unicorn y s.t. Alice sought y and Alice found y.

Again, showing only the end results after lexical insertion for reasons of space:

$$\text{Alice} \bullet ((\text{sought} \bullet (\text{and} \bullet \text{found})) \bullet (\text{a} \bullet \text{unicorn})), \downarrow \bar{s}^z$$

$$\vdash \begin{cases} (z\ ((\textsc{seek}\ \lambda P \exists y((\textsc{unicorn}\ y) \wedge (P\ y)))\ \textsc{alice})) \wedge \exists y((\textsc{unicorn}\ y) \wedge (z\ ((\textsc{find}\ y)\ \textsc{alice}))) & (4a) \\ \exists y((\textsc{unicorn}\ y) \wedge (z\ ((\textsc{seek}\ \lambda P(P\ y))\ \textsc{alice})) \wedge (z\ ((\textsc{find}\ y)\ \textsc{alice}))) & (4b) \end{cases}$$

4 Type-shifting

The previous section attempted the description of $\mathbf{CNL}^{pol}$'s semantic expressivity by empirical illustration. To arrive at a more precise characterization, we now compare our treatment of scopal ambiguities with a more well-understood solution. Hendriks ([6]), generalizing [12], proposed a flexible correspondence between syntax and semantics by closing the readings associated with a given syntactic derivation under the following derived semantic inference rules, referred to collectively by *type shifting*:

1. **Value Raising.** Assuming right-associative bracketing, terms of type $\sigma_1 \to \ldots \to \sigma_n \to \sigma$ may be lifted to the type $\sigma_1 \to \ldots \to (\sigma \to \tau') \to \tau'$, for any τ'. E.g., a proper name 'Alice' with interpretation of type e undergoes value raising to derive a term of the type $(e \to t) \to t$ for generalized quantifiers.
2. **Argument Raising.** Inhabitants of $\sigma_1 \to \ldots \to \sigma_n \to \tau$ may transition into $\sigma_1 \to \ldots \to ((\sigma_i \to \tau') \to \tau') \to \tau$. Typically applies to the subject and object positions of transitive verbs. For example, $e \to (e \to t)$ undergoes argument raising twice in order to derive the type of a binary relation on generalized quantifiers. Depending on which of the arguments is raised first, we obtain different scopal readings.
3. **Argument Lowering.** Inhabitants of $\sigma_1 \to \ldots \to \sigma_i \to \tau$ may be derived from terms of type $\sigma_1 \to \ldots \to ((\sigma_i \to \tau') \to \tau') \to \tau$. For example, lowering and subsequent raising (for the choice $\tau' = t$) of the object position of an intensional verb allows for the derivation of de re readings.

By applying the right combinations of these rules, one could derive all combinatorially available scopal readings for a given sentence, as proved by Hendriks. What is important to note is that all rules involved are derivable within λ-calculus. In contrast, analogous rules for the traditional incarnations of CTL necessitate full associativity and commutativity, leading to serious overgeneration. Consequently, type-shifting seemed exclusive to the realm of semantics, necessitating the relaxation of compositionality if it was to be made any use of.

We claim the existence of derivable rules of inference within $\mathbf{CNL}^{pol}$ mapping to Hendriks' type-shifting schemas under semantic interpretation. Consequently, the strict correspondence between syntax and semantics can be restored, while preserving semantic expressivity. One restriction, however, applies: the types τ' in the above explanation of type-shifting always are to be t. In practice, this still leaves us with sufficient generality to account for scoping within clausal domains.

Theorem 8. The following rules of *Value Raising* (VR), *Argument Raising* (AR) and *Argument Lowering* (AL) are derivable within $\mathbf{CNL}^{pol}$:

$$\begin{array}{rl}
\Gamma \vdash s : N \Rightarrow \Gamma \vdash \lambda x(x\ s) : \uparrow\downarrow N & (VR) \\
\Gamma \vdash s : (\downarrow(M/Q))^{\perp} \Rightarrow \Gamma \vdash \lambda\langle x, v\rangle(x\ \lambda u(s\ \langle u, v\rangle)) : (\downarrow(M/\downarrow\uparrow Q))^{\perp} & (AR^{r}) \\
\Gamma \vdash s : (\downarrow(P\backslash N))^{\perp} \Rightarrow \Gamma \vdash \lambda\langle v, x\rangle(x\ \lambda u(s\ \langle v, u\rangle)) : (\downarrow(\downarrow\uparrow P\backslash N))^{\perp} & (AR^{l}) \\
\Gamma \vdash s : (\downarrow(M/\downarrow\uparrow Q))^{\perp} \Rightarrow \Gamma \vdash \lambda\langle u, v\rangle(s\ \langle\lambda x(x\ u), v\rangle) : (\downarrow(M/Q))^{\perp} & (AL^{r}) \\
\Gamma \vdash s : (\downarrow(\downarrow\uparrow P\backslash N))^{\perp} \Rightarrow \Gamma \vdash \lambda\langle v, u\rangle(s\ \langle v, \lambda x(x\ u)\rangle) : (\downarrow(P\backslash N))^{\perp} & (AL^{l})
\end{array}$$

Example 9. We illustrate the above result with a lexical entry from Figure 2:

$$\text{found} \vdash \lambda\langle y, \langle q, x\rangle\rangle(q\ ((\textsc{find}\ y)\ x)) : (\downarrow((np\backslash\uparrow s)/np))^{\perp}$$

Note the term involved is of type $\neg(e \times (\neg t \times e))$, which by uncurrying is isomorphic to $e \rightarrow (e \rightarrow \neg\neg t))$. Applying Argument Raising (i.e., (AR^{r})), we get

$$\text{found} \vdash \lambda\langle Y, \langle q, x\rangle\rangle(Y\ \lambda y(q\ ((\textsc{find}\ y)\ x))) : (\downarrow((\downarrow\uparrow np\backslash\uparrow s)/np))^{\perp}$$

which is of type $\neg(\neg\neg e \times (\neg t \times e))$, i.e., $\neg\neg e \rightarrow (e \rightarrow \neg\neg t)$.

Still lacking is the means to target other positions besides the direct object for Argument Raising (or -Lowering), a possibility rendered available in Hendriks' presentation by allowing for any number of arguments preceding the one to which the operation was applied. In our case, the following result allows any intervening arguments to be stripped off one by one, and to be added back after the desired type-shifting rule has been applied.

Lemma 10. The following are derivable rules of inference:

$$\frac{\Gamma \bullet P^{x} \vdash s : (\downarrow N)^{\perp}}{\Gamma \vdash \lambda\langle x, y\rangle(s\ y) : (\downarrow(N/P))^{\perp}} > \qquad \frac{P^{x} \bullet \Gamma \vdash s : (\downarrow N)^{\perp}}{\Gamma \vdash \lambda\langle y, x\rangle(s\ y) : (\downarrow(P\backslash N))^{\perp}} <$$

$$\frac{\Gamma \vdash s : (\downarrow(N/P))^{\perp}}{\Gamma \bullet P^{x} \vdash \lambda y(s\ \langle x, y\rangle) : (\downarrow N)^{\perp}} >' \qquad \frac{\Gamma \vdash s : (\downarrow(P\backslash N))^{\perp}}{P^{x} \bullet \Gamma \vdash \lambda y(s\ \langle y, x\rangle) : (\downarrow N)^{\perp}} <'$$

Example 11. Continuing where we left off in E.9, (AR^l) combines with $(>)$ and $(>')$ to allow for Argument Raising of the subject position, resulting in

$$\text{found} \vdash \lambda\langle Y, \langle q, X\rangle\rangle(X\ \lambda x(Y\ \lambda y(q\ ((\textsc{find}\ y)\ x)))) : (\downarrow((\downarrow\uparrow np\backslash\uparrow s)/\downarrow\uparrow np))^{\perp}$$

being of type $\neg\neg e \to (\neg\neg e \to \neg\neg t)$ after uncurrying. Note we could also have applied (AR^l) first, followed by (AR^r) to derive

$$\text{found} \vdash \lambda\langle Y, \langle q, X\rangle\rangle(Y\ \lambda y(X\ \lambda x(q\ ((\textsc{find}\ y)\ x)))) : (\downarrow((\downarrow\uparrow np\backslash\uparrow s)/\downarrow\uparrow np))^{\perp}$$

The latter allows for the derivation of object-wide scope readings, whereas raising the subject after the object favors subject-wide scope.

5 Evaluation

We have presented polarized classical non-associative Lambek calculus, arguing for its applicability to the analysis of scopal ambiguities despite its resource sensitivity, and despite keeping to a strict correspondence between syntax and semantics. In contrast, earlier categorial accounts typically involved non-context-free mechanisms by allowing for controlled associativity and commutativity, or relaxed compositionality to a relation. We have compared our approach to one such traditional account, to wit, Hendriks' use of type-shifting in associating syntactic derivations with sets of readings. Below, we briefly discuss, first, a point of critique concerning our capacity to block scopal readings that, while combinatorially possible, are unrealized by linguistic reality. Second, we draw attention to a curious discrepancy with Hendriks' account of verbs taking clausal complements, and, finally, make a brief comparison with a proposal for the categorial analysis of scope closely related to ours.

5.1 Blocking scopal readings

While we could ensure derivability of all combinatorially available scopal readings for the various examples we considered, this same property also constitutes the main limitation of our approach: situations where linguistic reality excludes certain readings are not so easily accounted for. While we had a small success with the analysis of scope sieves, scope islands fall outside our coverage. The reasons for this go deep: through the display postulates, any variable-labeled formula may be isolated as the whole of one of the sequents' components, which may subsequently be abstracted over through $(\uparrow)$. In other words, accounting for scope islands means to restrict the display postulates, which constitute an essential ingredient for the notion of structure adopted by **CNL**, as exemplified graphically by De Groote and Lamarche's proof nets ([3]).

5.2 On verbs taking clausal complements

Our analysis of verbs taking clausal complements is host to an interesting curiosity, setting it apart from Hendriks' account. One might first expect the category

$\downarrow((np\backslash\uparrow s)/s)$ assigned to *heard* in §3.4 to foresee at least in the local reading, with type-shifting required for non-local scope construal. Such a situation would parallel Hendriks' [6], which starts from a minimal semantic type assignment. Instead, we find that only the non-local reading is derived, whereas a local reading necessitates the prefixing of the s argument by a double shift, cf. the category $\downarrow((np\backslash\uparrow s)/\downarrow\uparrow s)$ assigned to *thinks* in §3.3.

5.3 Comparison

Another account of scopal ambiguities similar to ours was recently put forward by Moortgat ([11]), working within the closely related *Lambek-Grishin calculus* (**LG**). There, scopal ambiguities were accounted for by adding minimal (co)negations to the logical vocabulary (referred to by (co-)Galois connections), used similarly to our shifts. We claim polarization allows for a better understanding of this proposal. Like **CNL**, **LG** turns out amendable to a polarized reformulation, as demonstrated in [1]. Moortgat's compositional semantics may then be recast as a translation of **LG** into $\mathbf{LG}^{pol}$, revealing that his minimal negations simply serve to enforce the appearance of shifts in the target.

References

1. A. Bastenhof. Polarized classical non-associative lambek calculus and formal semantics. In S. Pogodalla and J-P. Prost, editors, *Logical Aspects of Computational Linguistics - 6th International Conference, LACL 2011. Proceedings*, volume 6736 of *Lecture Notes in Computer Science*, pages 33–48. Springer, 2011.
2. N.D. Belnap. Display Logic. *Journal of Philosophical Logic*, 11(4):375–417, 1982.
3. P. De Groote and F. Lamarche. Classical non associative Lambek calculus. *Studia Logica*, 71:355–388, 2002.
4. L.T.F. Gamut. *Logic, Language and Meaning, Volume 2: Intensional Logic and Logical Grammar.* University of Chicago Press, 1991.
5. J-Y. Girard. A new constructive logic: classical logic. *Mathematical Structures in Computer Science*, 1(3):255–296, 1991.
6. H. Hendriks. *Studied flexibility. Categories and types in syntax and semantics.* PhD thesis, ILLC Amsterdam, 1993.
7. J. Lambek. The mathematics of sentence structure. *American Mathematical Monthly*, 65:154–169, 1958.
8. J. Lambek. On the calculus of syntactic types. In R. Jakobson, editor, *Structure of Language and its Mathematical Aspects, Proceedings of the Twelfth Symposium in Applied Mathematics*, 1961.
9. R. Montague. Universal grammar. *Theoria*, 36(3):373–398, 1970.
10. M. Moortgat. Categorial type logics. In *Handbook of Logic and Language*, pages 93–177. Elsevier, 1997.
11. M. Moortgat. Symmetric categorial grammar: residuation and Galois connections. *CoRR*, abs/1008.0170, 2010.
12. B. Partee and M. Rooth. Generalized conjunction and type ambiguity. In R. Bäuerle, C. Schwarze, and A. von Stechow, editors, *Meaning, use, and interpretation of language*, pages 361–383. Gruyter, Berlin, 1983.