Computational Coverage of Type Logical Grammar: The Montague Test

Glyn Morrill • Oriol Valentín

Abstract  It is nearly half a century since Montague made his contributions to the field of logical semantics. In this time, computational linguistics has taken an almost entirely statistical turn and mainstream linguistics has adopted an almost entirely non-formal methodology. But in a minority approach reaching back before the linguistic revolution, and to the origins of computing, type logical grammar (TLG) has continued championing the flags of symbolic computation and logical rigor in discrete grammar. In this paper, we aim to concretise a measure of progress for computational grammar in the form of the Montague Test. This is the challenge of providing a computational cover grammar of the Montague fragment. We formulate this Montague Test and show how the challenge is met by the type logical parser/theorem-prover CatLog2.

Keywords  Montague semantics · Montague grammar · categorial grammar · type logical grammar · computational grammar · semantic parsing · parsing as deduction · parsing/theorem-proving

1 Introduction
Perhaps nobody does Montague semantics anymore, or perhaps everybody does Montague semantics now and it has become a part of the scenery. Around 1970, Richard Montague wrote three papers, “Universal grammar” (Montague 1970b), “English as a formal language” (Montague 1970a), and “The proper treatment of quantification in ordinary English” (Montague 1973), which overturned the prevailing view that natural language semantics was too ephemeral to be formalised. The third paper, especially, introduced lambda calculus and higher-order intensional logic for semantic representation by presenting a formal fragment of English with a translation into logic.
Montague’s approach was first popularised in the textbook Dowty et al. 1981. Since then, linguistics has become infused with Montague semantics starting with journals such as Linguistics and Philosophy and conferences such as the Amsterdam Colloquium, and spreading out in such a way that today there is an extensive interdisciplinary field of formal semantics based on lambda calculus and type logic. It is not that nobody does Montague semantics anymore, it is that now Montague semantics is taken for granted by many.

If you don’t know where you have come from, you don’t know where you are going. How can we be sure we are making progress? Here, in relation to Montague semantics, we propose as an exercise of intermediate difficulty, as a health check on approaches, the Montague Test, which is to provide a computational cover grammar of the Montague fragment as represented by the example sentences of Dowty et al. 1981:chap. 7.

Our broad concern is whether linguistics, rather than building on the achievements of the past and consolidating them, is rather in danger of drifting from trend to trend or lurching from fashion to fashion, in an aleatory or even cyclic fashion. Linguistics has its scholarly roots in the arts and humanities and from such origins a certain tendency to fantasia and self-proclamation persists. Perhaps this headiness partially explains why linguistics has remained a novice science while, for example, biology and computational biology have gone from strength to strength. Our plea here is that before a linguistic approach is deemed the new revolution, it proves its credentials by providing a computational cover grammar of the 50 years old Montague fragment.

In providing a computational cover grammar, we semantically parse the sentences provided with analysis trees in Dowty et al. 1981:chap. 7, assigning them logical translations “corresponding” to those given there, and distinguishing the same readings with comparable truth conditions. This minicorpus, which includes quantification, intensionality and some coordination and anaphora, is as follows:¹

(7-7) John walks \( \text{walk}'(j) \)

¹The reference numbers are taken directly from Dowty et al. 1981:chap. 7. Observe that the minicorpus preserves Montague’s practice of assigning raised types to exten- sional verbs for uniformity with intensional verbs.
(7-16) every man talks \( \forall x[\text{man}'(x) \to \text{talk}'(x)] \)

(7-19) the fish walks \( \exists y[\forall x[\text{fish}'(x) \leftrightarrow x = y] \land \text{walk}'(y)] \)

(7-32) every man walks or talks \( \forall y[\text{man}'(y) \to [\text{walk}'(y) \lor \text{talk}'(y)]] \)

(7-34) every man walks or every man talks
\[ [\forall x[\text{man}'(x) \to \text{walk}'(x)] \lor \forall x[\text{man}'(x) \to \text{talk}'(x)]] \]

(7-39) a woman walks and she talks
\( \exists x[\text{woman}'(x) \land [\text{walk}'(x) \land \text{talk}'(x)]] \)

(7-43, 45) John believes that a fish walks
\begin{align*}
\text{believe}'(j, ^\exists x[\text{fish}'(x) \land \text{walk}'(x)]) \\
\exists x[\text{fish}'(x) \land \text{believe}'(j, ^\exists [\text{walk}'(x)])]
\end{align*}

(7-48, 49, 52) every man believes that a fish walks
\begin{align*}
\exists x[\text{fish}'(x) \land \forall y[\text{man}'(y) \to \text{believe}'(y, ^\exists [\text{walk}'(x)])]] \\
\forall y[\text{man}'(y) \to \exists x[\text{fish}'(x) \land \text{believe}'(y, ^\exists [\text{walk}'(x)])]] \\
\forall y[\text{man}'(y) \to \text{believe}'(y, ^\exists [\exists x[\text{fish}'(x) \land \text{walk}'(x)])]]
\end{align*}

(7-57) every fish such that it walks talks
\( \forall x[[\text{fish}'(x) \land \text{walk}'(x)]] \to \text{talk}'(x)] \)

(7-60, 62) John seeks a unicorn
\begin{align*}
\text{try}'(j, ^\exists x[\text{find}'(\forall P \exists x[\text{unicorn}'(x) \land [\forall P](x)])]) \\
\text{try}'(j, ^\lambda z[\exists x[\text{unicorn}'(x) \land [\text{find}'(\forall P[[\forall P](z)])(j)]]])
\end{align*}

(7-73) John is Bill \( j = b \)

(7-76) John is a man \( \text{man}'(j) \)

(7-83) necessarily John walks \( \Box[\text{walk}'(j)] \)

(7-86) John walks slowly \( \text{slowly}'(\forall \text{walk}') (j) \)

(7-91) John tries to walk \( \text{try}'(\forall \text{walk}') (j) \)

(7-94) John tries to catch a fish and eat it
\begin{align*}
\text{try}'(j, ^\lambda y \exists x[\text{fish}'(x) \land \\
\text{catch}'(\forall P[[\forall P](y)])'(x) \land \text{eat}'(\forall P[[\forall P](y)])'(x))]
\end{align*}
Table 1  Categorial connectives

(7-98) **John finds a unicorn**
\[ \exists x [\text{unicorn}'(x) \land \text{find}'(\wedge \lambda P[[\forall P](x))(j))] \]

(7-105) **every man such that he loves a woman loses her**
\[ \exists y [\text{woman}'(y) \land \forall x [[\text{man}'(x) \land \text{love}'(\wedge \lambda P([[\forall P](y))(x))] \rightarrow \text{lose}'(\wedge \lambda P([[\forall P](y)])(x))] \]

(7-110) **John walks in a park**
\[ \exists x [\text{park}'(x) \land \text{in}'(\wedge \lambda P[[\forall P](x))(\text{walk}'(j))] \]

(7-116, 118) **every man doesn’t walk**
\[ \neg \forall x [\text{man}'(x) \rightarrow \text{walk}'(x)] \]
\[ \forall x [\text{man}'(x) \rightarrow \neg \text{walk}'(x)] \]

### 2 Type Logical Grammar

Type logical grammar (TLG) is a categorial theory of syntax and semantics in which words and expressions are classified by logical types. TLG
Computational Coverage of Type Logical Grammar: The Montague Test

is expounded in Moortgat 1988, 1997, Morrill 1994, 2011, Carpenter 1997, Jäger 2005, Moot & Retoré 2012. The logical types form an intuitionistic sublinear logic and their rules are universal; a grammar comprises just a lexicon classifying basic expressions. TLG is thus a purely lexical formalism.

A sign $\alpha: A: \phi$ consists of a prosodic form $\alpha$, a syntactic type $A$, and a semantic form $\phi$. A prosodic sort map $s$ maps syntactic types to prosodic sorts which are the number of points of discontinuity of expressions of that type; a semantic type map $T$ maps syntactic types to semantic types which are essentially formulas of intuitionistic propositional logic/types of lambda calculus under the Curry-Howard correspondence. In a sign $\alpha: A: \phi$, $\alpha$ must be of prosodic sort $s(A)$ and $\phi$ must be of semantic type $T(A)$.

The categorial connectives of our type logical grammar are as shown in table 1. They comprise the primary connectives, in the first row, semantically inactive variants, in the second row, and deterministic (unary) and nondeterministic (binary) defined connectives in the third and fourth rows.

Regarding the primary connectives, the displacement connectives (Morrill et al. 2011) are made up of the continuous (Lambek) and discontinuous multiplicatives. Then there are additives (Morrill 1990a), quantifiers (Morrill 1994), normal modalities (Morrill 1990b, Moortgat 1997), bracket modalities (Morrill 1992, Moortgat 1996), exponentials (Morrill & Valentín 2015a), limited contraction (Jäger 2005) and limited weakening (Morrill & Valentín 2014b).

The semantically inactive secondary connectives are made up of semantically inactive multiplicatives (Morrill & Valentín 2014b), additives (Morrill 1994), quantifiers (Morrill 1994), and normal modalities (Hepple 1990, Moortgat 1997). The deterministic secondary connectives are made up of the unary connectives projection and injection (Morrill et al. 2009) and split and bridge (Morrill & Merenciano 1996), and the nondeterministic secondary connectives are made up of concatenative binary connectives of division and product and discontinuous binary connectives of extraction, infixation and product (Morrill et al. 2011). At the bottom right is a metalogical (“negation as failure”) connective of difference (Morrill & Valentín 2014a).
A lexicon consists of a set of (lexical) signs. Our lexicon for the Montague fragment is as follows; rules for connectives used in the fragment are given in the Appendix:

\[
\begin{align*}
\text{a: } & \forall g(\forall f((Sf \downarrow \exists nt(s(g)))\downarrow Sf)/\text{CNs}(g)) : \lambda A \lambda B \exists C[(A C) \land (B C)] \\
\text{and: } & \forall f((\forall s f/\downarrow \exists \downarrow Sf)/\exists Sf) : (\Phi^n \circ \text{and}) \\
\text{and: } & \forall a \forall f((\forall (\exists na/\downarrow \exists \downarrow f)/\exists Sf))/\exists ((\exists na/\downarrow \exists \downarrow Sf))/\exists ((\exists na/\downarrow \exists \downarrow Sf)) : (\Phi^n \circ \text{s o and}) \\
\text{believes: } & \Box((\exists g \exists nt(s(g))/\exists Sf)/(\exists CPthat/\exists Sf)) : ^\lambda A \lambda B((^\exists \text{believe} A) B) \\
\text{bill: } & \exists nt(s(m)) : b \\
\text{catch: } & \Box((\exists a \exists na/\exists Sb)/\exists a Na) : ^\lambda A \lambda B((^\exists \text{catch} A) B) \\
\text{doesn't: } & \forall g \forall a((Sg \uparrow (\exists na/\exists Sf))/((\exists na/\exists Sb))\downarrow Sg) : \lambda A \neg (A \lambda B \lambda C(B C)) \\
\text{eat: } & \Box((\exists a \exists na/\exists Sb)/\exists a Na) : ^\lambda A \lambda B((^\exists \text{eat} A) B) \\
\text{every: } & \forall g(\forall f((Sf \uparrow \exists nt(s(g)))\downarrow Sf)/\exists \text{CNs}(g)) : \lambda A \lambda B \forall C[(A C) \rightarrow (B C)] \\
\text{finds: } & \Box((\exists g \exists nt(s(g))/\exists Sf)/\exists a Na) : ^\lambda A \lambda B((^\exists \text{find} A) B) \\
\text{fish: } & \exists \text{CNs}(n) : \text{fish} \\
\text{he: } & ^\exists [(\exists Sg/\exists nt(s(m)))/(\exists nt(s(m))/\exists Sg)) : \lambda A A \\
\text{her: } & \forall g \forall a((((\exists na/\exists Sg) \uparrow \exists nt(s(f)))\downarrow ((\exists na/\exists Sg)\downarrow \exists nt(s(f)))) : \lambda A A \\
\text{in: } & \Box((\forall a \forall f(((\exists na/\exists Sf)/(\exists na/\exists Sf))/\exists a Na) : ^\lambda A \lambda B((^\exists \text{in} A) (B C)) \\
\text{is: } & \Box((\exists g \exists nt(s(g))/\exists Sf)/(\exists a Na/(\exists g((\exists CNs/\exists CNs))\downarrow ((\exists CNs)) \rightarrow I))) : ^\lambda A \lambda B(A \rightarrow C.[B = C]; D.((D \lambda E[E = B]) B)) \\
\text{it: } & \forall f \forall a(((\exists na/\exists Sf) \uparrow \exists nt(s(n)))/(\exists nt(s(n))/\exists Sf)) : \lambda A A \\
\text{it: } & ^\exists [(\exists Sf/\exists nt(s(n)))/(\exists nt(s(n))/\exists Sf)) : \lambda A A \\
\text{john: } & \exists nt(s(m)) : j \\
\text{loses: } & \Box((\exists g \exists nt(s(g))/\exists Sf)/\exists a Na) : ^\lambda A \lambda B((^\exists \text{lose} A) B) \\
\text{loves: } & \Box((\exists g \exists nt(s(g))/\exists Sf)/\exists a Na) : ^\lambda A \lambda B((^\exists \text{love} A) B) \\
\text{man: } & \Box \exists \text{CNs}(m) : \text{man} \\
\text{necessarily: } & \exists (\exists A/\Box \exists A) : \text{Nec} \\
\text{or: } & \forall f((\exists Sf/\exists \downarrow \exists \downarrow Sf))/\exists Sf) : (\Phi^n \circ \text{or}) \\
\text{or: } & \forall a \forall f((\exists (\exists na/\exists Sf) \downarrow \exists \downarrow (\exists na/\exists Sf))/\exists ((\exists na/\exists Sf)) : (\Phi^n \circ \text{or}) \\
\text{or: } & \forall f((\exists Sf/((\exists g \exists nt(s(g))/\exists Sf)))/(\exists nt(s(g))/\exists Sf)) : (\Phi^n \circ \text{or}) \\
\text{park: } & \Box \exists \text{CNs}(n) : \text{park} \\
\text{seeks: } & \Box((\exists g \exists nt(s(g))/\exists Sf)/\Box \exists a \forall f(((\exists Na/\exists Sf)/\exists a \exists Nb)/(\exists Na/\exists Sf))) : ^\lambda A \lambda B((^\Box \exists \text{seeks} (B)) B) \\
\text{she: } & ^\exists [(\exists Sg/\exists nt(s(f)))/(\exists nt(s(f))/\exists Sg)) : \lambda A A
slowly : □∀f(□⟨⟩Na\(Sf)\/(⟨⟩□Na\(Sf))) : ^λA^λB(^slowly ^^(^A ^B))
such+that : □∀((CNn\CNn)/(Sf\□Nt(n))) : λAλBλC[(B C) ∧ (A C)]
talks : □(⟨⟩∃gNt(s(g))\(Sf) : ^λA(^talk A)
that : □(CPthat\□Sf) : λAA
the : □∀n(Nt(n)/CNn) : ι
to : □((PPto/∃aNa)\∀n((⟨⟩Nn\Si)/(⟨⟩Nn\Sb))) : λAA
tries : □(⟨⟩∃gNt(s(g))\(Sf)/□(⟨⟩∃gNt(s(g))\(Si)) : ^λA^λB(□^try ^^(^A B) B)
unicorn : □CNs(n) : unicorn
walk : □(⟨⟩∃aNa\(Sb) : ^λA(^walk A)
walks : □(⟨⟩∃gNt(s(g))\(Sf) : ^λA(^walk A)
woman : □CNs(f) : woman

3 Performing the Montague Test
CatLog2 is a type logical parser/theorem prover with a web interface at http://www.cs.upc.edu/~morrill/CatLog/CatLog2/index.php. It:

• comprises 6000 lines of prolog
• has 20 primitive categorial connectives, 29 defined connectives, and 1 metalogical connective: a total of 50 connectives
• has typically 2 rules for each connective: a rule of use and a rule of proof: roughly 50 × 2 = 100 rules
• uses backward chaining sequent proof search and uses focusing (Andreoli 1992); for the focused rules—about half of them—for a binary connective there are 4 cases of “polarity”: +/-, +/-, -/+ , -/-: 50 + 50 × 4 = a total of about 250 rules

At CSSP in Paris on 9 October 2015, the Montague Test was performed by CatLog2 version “gmontague” with input in the following format; note that currently it is necessary to give syntactic domains in the input to CatLog2 (though these play no role in Montague’s grammar):

str(dwp(’(7-7)’), [b([john]), walks], s(f)).
str(dwp(’(7-16)’), [b([every, man]), talks], s(f)).
str(dwp(’(7-19)’), [b([the, fish]), walks], s(f)).
str(dwp(’(7-32)’), [b([every, man]), b([b([walks, or, talks])])], s(f)).
The \LaTeX{} output generated was as follows. Each item comes in the form of its identifier and the prosodic form of its input, followed by each semantically labelled sequent that results from lexical lookup. Where there is a derivation or derivations for a sequent, these appear in figures with the semantic forms delivered by the analysis in the main text. CatLog2 observes the proof search discipline of focusing (Andreoli 1992, Morrill & Valentín 2015b): in the derivations the focused types are boxed, which means that when a complex type in a conclusion is boxed, it is the active type of the inference. For reasons of space, some derivations are omitted.

\[(dwp(\langle7-7\rangle)) \text{[john]} + \text{walks} : Sf\]

\[(\Box Nt(s(m)) : j], \Box(\exists g Nt(s(g)) \\Box Sf) : \lambda A(\Box \text{walk } A) \Rightarrow Sf\]

For the derivation, see figure 1.
(walk $j$)

(dwp((7-16))) [every+$\text{man}$]+talks : $S_f$

$$[\square Nt(s(m))] \Rightarrow Nt(s(m)) \quad \square L$$

$$[\square Nt(s(m))] \Rightarrow Nt(s(m)) \quad \exists R$$

$$[\square Nt(s(m))] \Rightarrow \exists g Nt(s(g)) \quad \langle R \rangle$$

$$[\square Nt(s(m))], \langle \exists g Nt(s(g)) \rangle S_f \Rightarrow S_f \quad \square L$$

$$[\square Nt(s(m))], [\square] (\exists g Nt(s(g)) \langle \rangle S_f) \Rightarrow S_f$$

**Figure 1** Derivation of (dwp((7-7)))

For the derivation, see figure 2.

$\forall C[(\forall \text{man} C) \rightarrow (\forall \text{talk} C)]$

(dwp((7-19))) [the+$\text{fish}$]+walks : $S_f$

$$[\forall n(Nt(n)/CNn) : i, \square CNs(n) : \text{fish}], [\square] (\exists g Nt(s(g)) S_f) : \lambda A(\forall \text{walk} A) \Rightarrow S_f$$

(derivation omitted)

(\forall \text{walk} (i \forall \text{fish}))

(dwp((7-32))) [every+$\text{man}$]+[walks+or+talks] : $S_f$

$$[\forall g(\forall f((S_f \uparrow Nt(s(g))) \downarrow S_f)/CNs(g)) : \lambda A \lambda B \forall C[(A C) \rightarrow (B C)], \square CNs(m) : \text{man}], [\square] (\exists g Nt(s(g)) S_f) : \lambda D(\forall \text{walk} D) \Rightarrow S_f$$

$$[\forall f((\square S_f) \downarrow S_f) \uparrow S_f] / S_f : (\Phi^n \circ \text{or}), [\square] (\exists g Nt(s(g)) S_f) : \lambda E(\forall \text{talk} E)$$

$$[\forall g(\forall f((S_f \uparrow Nt(s(g))) \downarrow S_f)/CNs(g)) : \lambda A \lambda B \forall C[(A C) \rightarrow (B C)], \square CNs(m) : \text{man}], [\square] (\exists g Nt(s(g)) S_f) : \lambda D(\forall \text{walk} D), \square CNs(n) : \text{fish}, \square (\exists g Nt(s(g)) S_f) : \lambda E(\forall \text{talk} E)$$

$$[\square Nt(s(m))] \Rightarrow Nt(s(m)) \quad \exists R$$

$$[\square Nt(s(m))] \Rightarrow \exists g Nt(s(g)) \quad \langle R \rangle$$

$$[\square Nt(s(m))], \langle \exists g Nt(s(g)) \rangle S_f \Rightarrow S_f \quad \square L$$

$$[\square Nt(s(m))], [\square] (\exists g Nt(s(g)) \langle \rangle S_f) \Rightarrow S_f$$
\[
\begin{array}{c}
\frac{\text{CNs}(m) \Rightarrow \text{CNs}(m)}{\blacksquare} \\
\frac{\text{CNs}(m) \Rightarrow \exists g \text{Nt}(s(g))}{\blacksquare} \\
\frac{\text{Nt}(s(m)) \Rightarrow \exists g \text{Nt}(s(g))}{\exists R} \\
\frac{\text{Nt}(s(m)) \Rightarrow \langle \exists g \text{Nt}(s(g)) \rangle S_f \Rightarrow S_f}{\langle R \rangle} \\
\frac{[\text{Nt}(s(m))], \langle \exists g \text{Nt}(s(g)) \rangle S_f \Rightarrow S_f}{\exists R} \\
\frac{[\text{Nt}(s(m))], \Box (\langle \exists g \text{Nt}(s(g)) \rangle S_f) \Rightarrow S_f}{\Box} \\
\frac{[1], \Box (\langle \exists g \text{Nt}(s(g)) \rangle S_f) \Rightarrow S_f}{\Box} \\
\end{array}
\]

**Figure 2** Derivation of (dwp((7-16)))

\[\Box \text{CNs}(m) : \text{man}], [[[\Box (\langle \exists g \text{Nt}(s(g)) \rangle S_f)]] : ^\lambda D (\sim \text{walk D}), \]
\[\forall a \forall f ((\langle \exists ?(\langle N a \rangle S_f) \rangle [\Box]^{-1} [\Box]^{-1} (\langle N a \rangle S_f)) / (\langle N a \rangle S_f)) : (\Phi^n + (s o) or), \]
\[\Box (\langle \exists g \text{Nt}(s(g)) \rangle S_f) : ^\lambda E (\sim \text{talk E}) ] ] \Rightarrow S_f \]

(Derivation omitted)

\[\forall C [(\sim \text{man C}) \rightarrow [(\sim \text{walk C}) \lor (\sim \text{talk C})]]\]

\[\Box \forall g (\forall f ((S_f \uparrow \text{Nt}(s(g))) \downarrow S_f) / \text{CNs}(g)) : \lambda A \lambda B \forall C [(A C) \rightarrow (B C)], \]
\[\Box \text{CNs}(m) : \text{man}], [[\Box (\langle \exists g \text{Nt}(s(g)) \rangle S_f)]] : ^\lambda D (\sim \text{walk D}), \]
\[\forall f ((\langle ? (S_f / (\langle \exists g \text{Nt}(s(g)) \rangle S_f)) [\Box]^{-1} [\Box]^{-1} (S_f / (\langle \exists g \text{Nt}(s(g)) \rangle S_f))) / (\langle \exists g \text{Nt}(s(g)) \rangle S_f)) : (\Phi^n + (s o) or), \Box (\langle \exists g \text{Nt}(s(g)) \rangle S_f) : ^\lambda E (\sim \text{talk E})] ] \Rightarrow S_f \]

(dwp((7-34))) [[['\text{every+man}]+\text{walks+or}+['\text{every+man}]+\text{talks}']] : S_f

\[\Box \forall g (\forall f ((S_f \uparrow \text{Nt}(s(g))) \downarrow S_f) / \text{CNs}(g)) : \lambda A \lambda B \forall C [(A C) \rightarrow (B C)], \]
\[\Box \text{CNs}(m) : \text{man}], [\Box (\langle \exists g \text{Nt}(s(g)) \rangle S_f)]] : ^\lambda D (\sim \text{walk D}), \]
\[\forall f ((\langle ? (S_f / (\langle \exists g \text{Nt}(s(g)) \rangle S_f)) [\Box]^{-1} [\Box]^{-1} (S_f / (\langle \exists g \text{Nt}(s(g)) \rangle S_f))) / (\langle \exists g \text{Nt}(s(g)) \rangle S_f)) : (\Phi^n + o or), \]
\[\Box g (\forall f ((S_f \uparrow \text{Nt}(s(g))) \downarrow S_f) / \text{CNs}(g)) : \lambda E \lambda F \forall G [(E G) \rightarrow (F G)], \]
\[\Box \text{CNs}(m) : \text{man}], [\Box (\langle \exists g \text{Nt}(s(g)) \rangle S_f)]] : ^\lambda H (\sim \text{talk H}) ] ] \Rightarrow S_f \]

(Derivation omitted)
\[ \forall H[(\text{``man } H) \rightarrow (\text{``walk } H)] \lor \forall C[(\text{``man } C) \rightarrow (\text{``talk } C)]] \]

\[ [[[\exists \forall g(\forall f((Sf^\uparrow Nt(s(g))))^\downarrow Sf)/CNs(g)) : \lambda A \lambda B \forall C[(A C) \rightarrow (B C)],
enna CNs(m) : \text{man}], \Box(\langle \exists \forall g Nt(s(g)) \rangle Sf) : ^\forall D(\text{``walk } D),
\forall a \forall f((\Box ?(\langle Nt \rangle Sf) \downarrow 1 \downarrow 1(\langle Nt \rangle Sf))/\Box(\langle Nt \rangle Sf)) : (\Phi^n + (s o) or),
\Box \forall g((\forall f((Sf^\uparrow Nt(s(g))))^\downarrow Sf)/CNs(g)) : \lambda E \lambda F \forall G[(E G) \rightarrow (F G)],
enna CNs(m) : \text{man}], \Box(\langle \exists \forall g Nt(s(g)) \rangle Sf) : ^\forall H(\text{``talk } H)] \Rightarrow Sf \]

\[ [[[\exists \forall g(\forall f((Sf^\uparrow Nt(s(g))))^\downarrow Sf)/CNs(g)) : \lambda A \lambda B \exists C[(A C) \land (B C)],
enna CNs(f) : \text{woman}], \Box(\langle \exists \forall g Nt(s(g)) \rangle Sf) : ^\forall D(\text{``walk } D),
\forall f((\Box ?(Sf) \downarrow 1 \downarrow 1 Sf)/\Box Sf) : (\Phi^n + (s o) and),
\Box \forall g((\forall f((Sf^\uparrow Nt(s(g))))^\downarrow Sf)/\Box(\langle \exists \forall g Nt(s(g)) \rangle Sf)) : ^\forall E E,
\Box(\langle \exists \forall g Nt(s(g)) \rangle Sf) : ^\forall F(\text{``talk } F)] \Rightarrow Sf \]

(Derivation omitted)

\[ \forall C[(\text{``woman } C) \land [(\text{``walk } C) \land (\text{``talk } C)]] \]

\[ [[[\exists \forall g(\forall f((Sf^\uparrow Nt(s(g))))^\downarrow Sf)/CNs(g)) : \lambda A \lambda B \exists C[(A C) \land (B C)],
enna CNs(f) : \text{woman}], \Box(\langle \exists \forall g Nt(s(g)) \rangle Sf) : ^\forall D(\text{``walk } D),
\forall a \forall f((\Box ?(\langle Nt \rangle Sf) \downarrow 1 \downarrow 1(\langle Nt \rangle Sf))/\Box(\langle Nt \rangle Sf)) : 
(\Phi^n + (s o) and),
\Box \forall g((\forall f((Sf^\uparrow Nt(s(f))))/\Box(\langle \exists \forall g Nt(s(f)) \rangle Sg)) : ^\forall E E,
\Box(\langle \exists \forall g Nt(s(g)) \rangle Sf) : ^\forall F(\text{``talk } F)] \Rightarrow Sf \]

(dwp((7-43, 45))) [[john] + believes + that + [a + fish] + walks : Sf}

\[ [\exists \forall g(\forall f((Sf^\uparrow Nt(s(m))))^\downarrow Sf)/\Box(\langle \exists \forall g Nt(s(g)) \rangle Sf) : (CP\text{that} \Box Sf)) : ^\forall A \lambda B(\text{``believe } A) B, \Box(\text{CP\text{that}} Sf) : ^\forall C C,
\Box \forall g((\forall f((Sf^\uparrow Nt(s(g))))^\downarrow Sf)/CNs(g)) : ^\forall D \lambda E \exists F[(D F) \land (E F)],
\]
∀CNs(n) : fish], □((⟨⟩∃gNt(s(g))\ Sf) : ^ λG (\^walk G) ⇒ Sf

For the derivation, see figure 3.

∃C[(\^fish C) ∧ ((\^believe (\^walk C)) j)]

For the derivation, see figure 4.

((\^believe ^∃F[(\^fish F) ∧ (\^walk F)]) j)

(dwp((7-48, 49, 52))) [every+man]+believes+that+[a+fish]+walks : Sf

[■∀g(∀f ((Sf↑Nt(s(g)))↓Sf)\ CNs(g)) : λAλB∀C[(A C) → (B C)],

□CNs(m) : man], [□((⟨⟩∃gNt(s(g))\ Sf)/(CPthat□Sf)) : ^ λDλE((\^believe D) E), ■(CPthat/□Sf) : λFF,

[■∀g(∀f ((Sf↑Nt(s(g)))↓Sf)\ CNs(g)) : λGλH∃I[(G I) ∧ (H I)],

□CNs(n) : fish], [□((⟨⟩∃gNt(s(g))\ Sf) : ^ λJ(\^walk J) ⇒ Sf

(Derivation omitted)

∃C[(\^fish C) ∧ ∀G[(\^man G) → ((\^believe ^ (√walk C)) G)]]

(Derivation omitted)

∀C[(\^man C) → ∃G[(\^fish G) ∧ ((\^believe ^ (√walk G)) C)]]

(Derivation omitted)

∀C[(\^man C) → ((\^believe ^∃J[(\^fish J) ∧ (√walk J))] C)]

(dwp((7-57))) [every+fish+such+that+[it]+walks]+talks : Sf

[■∀g(∀f ((Sf↑Nt(s(g)))↓Sf)\ CNs(g)) : λAλB∀C[(A C) → (B C)],

□CNs(n) : fish, ■∀n((CNn↓CNn)/(Sf \■Nt(n))) : λDλEλF[(EF) ∧ (DF)],

[■∀v∀a(((⟨⟩Na\ Sf)↑■Nt(n)))↓(⟨⟩Na\ Sf)\■Nt(n))) : λGG,

□((⟨⟩∃gNt(s(g))\ Sf) : ^ λH(\^walk H), □((⟨⟩∃gNt(s(g))\ Sf) : ^ λI(\^talk I) ⇒ Sf

[■∀g(∀f ((Sf↑Nt(s(g)))↓Sf)\ CNs(g)) : λAλB∀C[(A C) → (B C)],

□CNs(n) : fish, ■∀n((CNn↓CNn)/(Sf \■Nt(n))) : λDλEλF[(EF) ∧ (DF)],

[■¬1∀f ((■Sf\■Nt(n)))\ ((□Nt(s(n)))\ Sf) : λGG,

□((⟨⟩∃gNt(s(g))\ Sf) : ^ λH(\^walk H), □((⟨⟩∃gNt(s(g))\ Sf) :
\[ \lambda I(\^{'talk} I) \Rightarrow Sf \]

(Derivation omitted)

\[ \forall C[[\^{'fish} C \land (\^{'walk} C) \rightarrow (\^{'talk} C)] \]

(dwp((7-60, 62))) \[ \text{[john]+seeks+a+unicorn : } Sf \]

\[ \sqcap Nt(s(m)) : j], \Box(((\langle g Nt(s(g)) \rangle Sf) / \exists a Na \oplus (\exists g ((CNg/\sqcup(CNg\backslash CGn)) - I)))) : \lambda A \lambda B(A \rightarrow C. [B = C]; D.((D \lambda E[E = B]) B)), \sqcap Nt(s(m)) : b \Rightarrow Sf \]

For the derivation, see figure 5.

\[ \exists C[\^{'unicorn} C \land ((\^{'try} ^{\langle g Nt(s(g)) \rangle Sf} \exists a Na \oplus (\exists g ((CNg/\sqcup(CNg\backslash CGn)) - I)))) : \lambda A \lambda B(A \rightarrow C. [B = C]; D.((D \lambda E[E = B]) B)), \Box C] \]

For the derivation, see figure 6.

\[ ((\^{'try} ^{\langle g Nt(s(g)) \rangle Sf} \exists a Na \oplus (\exists g ((CNg/\sqcup(CNg\backslash CGn)) - I)))) : \lambda A \lambda B(A \rightarrow C. [B = C]; D.((D \lambda E[E = B]) B)), \Box C] \]

For the derivation, see figure 7.

\[ j = b \]

(dwp((7-76))) \[ \text{[john]+is+a+man : } Sf \]

\[ \sqcap Nt(s(m)) : j], \Box(((\langle g Nt(s(g)) \rangle Sf) / \exists a Na \oplus (\exists g ((CNg/\sqcup(CNg\backslash CGn)) - I)))) : \lambda A \lambda B(A \rightarrow C. [B = C]; D.((D \lambda E[E = B]) B)), \Box C \]

For the derivation, see figure 8.

\[ \exists C[\^{'man} C \land [j = C]] \]
Figure 5

First derivation of (dmp(7-60, 62))
The image contains a diagram of a computational derivation process for a Type Logical Grammar, specifically the Montague Test. The derivation involves a set of rules and transformations that are typical in formal logic and type theory. The diagram illustrates how a given input is transformed step by step through various logical and syntactic operations.

The diagram is complex and includes symbols and expressions that are central to the Montague Test, reflecting the computational coverage of the Type Logical Grammar. The specific rules and transformations shown are essential for understanding how the test is conducted within this context.
Figure 7 Derivation of (dwp((7-73)))

(dwp((7-83))) necessarily+[john]+walks : Sf

■(SA/□SA) : Nec, [□Nt(s(m)) : j], □(⟨⟩∃gNt(s(g))\Sf) : ^λB(‘walk B) ⇒ Sf

(Derivation omitted)

(Nec ^((‘walk j))

(dwp((7-86))) [john]+walks+slowly : Sf

[□Nt(s(m)) : j], □(⟨⟩∃gNt(s(g))\Sf) : ^λA(‘walk A), □∀a∀f(□(⟨⟩Na\Sf)\Nn((⟨⟩□Na\Sf)) : ^λBλC(‘slowly ^(‘B ‘C)) ⇒ Sf

(Derivation omitted)

(‘slowly ^((‘walk j))

(dwp((7-91))) [john]+tries+to+walk : Sf

[□Nt(s(m)) : j], □(⟨⟩∃gNt(s(g))\Sf)/□(⟨⟩∃gNt(s(g))\Si)) : ^λAλB(‘try ^(‘A B)) B, (PPto/∃aNa)\∀n((⟨⟩Nn\Si)/(⟨⟩Nn\Sb)) : λCC, □(⟨⟩∃aNa\Sb) : ^λD(‘walk D) ⇒ Sf

(Derivation omitted)

((‘try ^((‘walk j)) j)

(dwp((7-94))) [john]+tries+to+[[catch+a+fish+and+eat+it]] : Sf

[□Nt(s(m)) : j], □(⟨⟩∃gNt(s(g))\Sf)/□(⟨⟩∃gNt(s(g))\Si)) : ^λAλB(‘try ^(‘A B)) B, (PPto/∃aNa)\∀n((⟨⟩Nn\Si)/(⟨⟩Nn\Sb)) :
Figure 8

Derivation of (dwp((/seven.oldstyle-/seven.oldstyle/six.oldstyle)))

\[ S \vdash CNs(m) \]
\[ \lambda C, [\square((\langle \exists a Na \backslash S f \rangle / \exists a Na) : ^\lambda D \lambda E(\langle \text{catch } D \rangle) E), \]
\[ \forall g(\forall f ((S f \uparrow \exists N t(s(g))) \downarrow S f) / \exists C N s(g)) : ^\lambda F \lambda G \forall H([F H] \land \langle G H \rangle)], \]
\[ \Box C N s(n) : \text{fish}, \forall f ((\exists ? S f \downarrow \exists S f) / \exists S f) : (\Phi^n ° \text{ and}), \]
\[ ((\langle \exists a Na \backslash S f \rangle / \exists a Na) : ^\lambda I \lambda J(\langle \text{eat } I \rangle J), \]
\[ \forall f \forall a (((\langle N a \backslash S f \rangle \uparrow \exists N t(s(n))) \downarrow \exists (\langle N a \backslash S f \rangle \exists N t(s(n)))) : \lambda K K] \Rightarrow S f \]

\[ [\exists N t(s(m)) : j], \Box((\langle \exists g N t(s(g)) \backslash S f \rangle / \exists g N t(s(g)) / \exists S i)) : \]
\[ ^\lambda A \lambda B(\langle \text{try } \langle \text{catch } D \rangle B), \Box((P P t o / \exists a N a) \exists \forall ((\langle N n \backslash S i \rangle / (\langle N n \backslash S b \rangle))), \]
\[ \lambda C C, [\square((\langle \exists a Na \backslash S f \rangle / \exists a Na) : ^\lambda D \lambda E(\langle \text{catch } D \rangle) E), \]
\[ \forall g(\forall f ((S f \uparrow \exists N t(s(g))) \downarrow S f) / \exists C N s(g)) : ^\lambda F \lambda G \forall H([F H] \land \langle G H \rangle)], \]
\[ \Box C N s(n) : \text{fish}, \forall f ((\exists ? S f \downarrow \exists S f) / \exists S f) : (\Phi^n ° \text{ and}), \]
\[ ((\langle \exists a Na \backslash S f \rangle / \exists a Na) : ^\lambda I \lambda J(\langle \text{eat } I \rangle J), \]
\[ [\exists ^\downarrow S f (\exists N t(s(n))) / (\exists N t(s(n)) / \exists S f)) : \lambda K K] \Rightarrow S f \]

(Derivation omitted)

\[ \exists C (\langle \text{fish } C \rangle \land (\langle \text{try } \langle \text{catch } C \rangle j \rangle \land \langle \text{eat } C \rangle j)))] \]

(Derivation omitted)

\[ (\langle \text{try } \exists F [\langle \text{fish } F \rangle \land (\langle \text{catch } F \rangle j \rangle \land (\langle \text{eat } F \rangle j)]] j) \]

\[ (\langle \text{try } \exists H [\langle \text{fish } H \rangle \land (\langle \text{catch } H \rangle j \rangle \land (\langle \text{eat } H \rangle j)]) j) \]

\[ [\exists N t(s(m)) : j], \Box((\langle \exists g N t(s(g)) \backslash S f \rangle / \exists g N t(s(g)) / \exists S i)) : \]
\[ ^\lambda A \lambda B(\langle \text{try } \langle \text{catch } D \rangle B), \Box((P P t o / \exists a N a) \exists \forall ((\langle N n \backslash S i \rangle / (\langle N n \backslash S b \rangle))), \]
\[ \lambda C C, [\square((\langle \exists a Na \backslash S f \rangle / \exists a Na) : ^\lambda D \lambda E(\langle \text{catch } D \rangle) E), \]
\[ \forall g(\forall f ((S f \uparrow \exists N t(s(g))) \downarrow S f) / \exists C N s(g)) : ^\lambda F \lambda G \forall H([F H] \land \langle G H \rangle)], \]
\[ \Box C N s(n) : \text{fish}, \forall f ((\exists ? S f \downarrow \exists S f) / \exists S f) : (\Phi^n ° \text{ and}), \]
\[ ((\langle \exists a Na \backslash S f \rangle / \exists a Na) : ^\lambda I \lambda J(\langle \text{eat } I \rangle J), \]
\[ [\exists ^\downarrow S f (\exists N t(s(n))) / (\exists N t(s(n)) / \exists S f)) : \lambda K K] \Rightarrow S f \]
(Φⁿ⁺ (s o) and), □(((∃aNa \ Sb)/∃aNa) : ^λIAJ(“eat I) J),
[[]⁻¹∀f([[Sf|∀Nt(s(n))/((⟨⟨Nt(s(n))\ Sf⟩): λKK]] → Sf

(dwp((7-98))) [john]+finds+a+unicorn : Sf

[∀g(∀f(((Sf↑Nt(s(g))\ Sf)/CNs(g)) : λAλB∀C([A C) → (B C)],
□CNs(m) : man, □∀n((CNn\ CNn)/(\ Sf|∀Nt(n)) : λDλEλF[(EF)\ (DF)],
[[]⁻¹∀g(([]Sg|∀Nt(s(m)))/⟨Nt(s(m))\ Sg⟩) : λGG],
[[]⁻¹∀g(∀gNt(s(g))\ Sf)/∃aNa) : ^λHλI(“love H) I),
[∀g(∀f(((Sf↑Nt(s(g))\ Sf)/CNs(g)) : λJλK∀L[(J L) \ (K L)],
□CNs(f) : woman],
[[(∃gNt(s(g))\ Sf)/∃aNa) : ^λMλN(“lose M) N),
[∀g∀a(((∃Nt(s(f))↑Nt(s(f)))/((⟨Nt(s(f))\ Sg)|Nt(s(f)))) : λOO → Sf

Figure 9: Derivation of (dwp((7-98)))

(Φⁿ⁺ (s o) and), □(((∃aNa \ Sb)/∃aNa) : ^λIAJ(“eat I) J),
[[]⁻¹∀f([[Sf|∀Nt(s(n))/((⟨⟨Nt(s(n))\ Sf⟩): λKK]] → Sf

(dwp((7-98))) [john]+finds+a+unicorn : Sf

[∀g(∀f(((Sf↑Nt(s(g))\ Sf)/CNs(g)) : λAλB∀C([A C) → (B C)],
□CNs(m) : man, □∀n((CNn\ CNn)/(\ Sf|∀Nt(n)) : λDλEλF[(EF)\ (DF)],
[[]⁻¹∀g(([]Sg|∀Nt(s(m)))/⟨Nt(s(m))\ Sg⟩) : λGG],
[[]⁻¹∀g(∀gNt(s(g))\ Sf)/∃aNa) : ^λHλI(“love H) I),
[∀g(∀f(((Sf↑Nt(s(g))\ Sf)/CNs(g)) : λJλK∀L[(J L) \ (K L)],
□CNs(f) : woman],
[[(∃gNt(s(g))\ Sf)/∃aNa) : ^λMλN(“lose M) N),
[∀g∀a(((∃Nt(s(f))↑Nt(s(f)))/((⟨Nt(s(f))\ Sg)|Nt(s(f)))) : λOO → Sf

For the derivation, see figure 9.

∃C(“unicorn C) \ (”find C) j]
\( \exists C[(\forall woman C) \land \forall G[(\forall man G) \land ((\forall love C) G]) \to ((\forall lose C) G)]] \)

\( \text{dwp((7-110))) } [\text{john}] + \text{walks} + \text{in} + \text{a} + \text{park} : Sf \)

\[ \begin{align*}
\square Nt(s(m)) : j, \& ((\forall g Nt(s(g)) \rightarrow Sf)) : \lambda A(\forall walk A), \\
\& (\forall a \forall f (((\forall Na \rightarrow Sf) \land ((\forall Na \rightarrow Sf)) \& ) \forall a Na)) : \lambda ABAC\lambda D(\forall in B) (C D), \\
\& \forall g(\forall f (((Sf \to \forall Nt(s(g))) \rightarrow Sf) \& CNs(g)) : \lambda E\lambda F\forall G[(E G) \land (F G)], \\
\& CNs(n) : park \Rightarrow Sf
\end{align*} \]

\( \exists C[(\forall park C) \land ((\forall in C) (\forall walk j))] \)

\( \text{dwp((7-116, 118))) } [\text{every} + \text{man}] + \text{doesnt} + \text{walk} : Sf \)

\[ \begin{align*}
\square \forall g(\forall f (((Sf \to \forall Nt(s(g))) \rightarrow Sf) \& CNs(g)) : \lambda A\lambda B\forall C[(A C) \to (B C)], \\
\& CNs(m) : man], \& \forall g \forall a (((Sg \to (((\forall Na \rightarrow Sf) \land ((\forall Na \rightarrow Sb)))) \rightarrow Sg) : \\
\lambda D \& (D \lambda E\lambda F(E F)), \& ((\forall a Na \& Sb) : \lambda G(\forall walk G) \Rightarrow Sf
\end{align*} \]

\( \forall C[(\forall man C) \to \neg (\forall walk C)] \)

\( \text{Appendix: Rules} \)

The syntactic types of displacement logic are sorted \( \mathcal{F}_0, \mathcal{F}_1, \mathcal{F}_2, \ldots \) according to the number of points of discontinuity 0, 1, 2, \ldots their expressions contain. Each type predicate letter has a sort and an arity which are naturals, and a corresponding semantic type. Assuming ordinary terms to be already given, where \( P \) is a type predicate letter of sort \( i \) and arity \( n \) and \( t_1, \ldots, t_n \) are terms, \( Pt_1 \ldots t_n \) is an (atomic) type of sort \( i \) of the corresponding semantic type. Compound types are formed by connectives as indicated in table 2,\(^2\) and the structure preserving semantic type map \( T \)

\(^2\)We list only connectives drawn from the first two rows of table 1, omitting some which are not central here.
associates these with semantic types.

In Gentzen sequent configurations \((\Gamma, \Delta)\) for displacement calculus a discontinuous type is a mother, rather than a leaf, and dominates its discontinuous components marked off by curly brackets and colons.

In Gentzen sequent antecedents for displacement logic with bracket modalities (structural inhibition) and exponentials (structural facilitation) there is also a bracket constructor for the former and ‘stoups’ for the latter.

**Stoups** (cf. the linear logic of Girard 2011 \((\zeta)\) are stores read as multisets for re-usable (nonlinear) resources which appear at the left of a configuration marked off by a semicolon (when the stoup is empty the semicolon may be omitted, as in the derivations of the previous section). The stoup of linear logic is for resources which can be contracted (copied) or weakened (deleted). By contrast, our stoup is for a linguistically motivated variant of contraction, and does not allow weakening. Furthermore, whereas linear logic is commutative, our logic is in general noncommutative and the stoup is used for resources which are also commutative.

A configuration together with a stoup is a *zone* \((\Xi)\). The bracket constructor applies not to a configuration alone but to a configuration with a
stoup, i.e a zone: reusable resources are specific to their domain.

Stoups $S$ and configurations $O$ are defined by ($\emptyset$ is the empty stoup; $\Lambda$ is the empty configuration; the separator 1 marks points of discontinuity:³

$$
S ::= \emptyset | F_0, S \\
O ::= \Lambda | T, O \\
T ::= 1 | F_0 | F_{i>0}(O: \ldots : O) | [S; O]
$$

For a type $A$, its sort $s(A)$ is the $i$ such that $A \in F_i$. For a configuration $\Gamma$, its sort $s(\Gamma)$ is $|\Gamma|_1$, that is, the number of points of discontinuity 1 which it contains. Sequents are of the form:

$$
S; O \Rightarrow T \text{ such that } s(O) = s(T)
$$

The figure $\overrightarrow{A}$ of a type $A$ is defined by:

$$
\overrightarrow{A} = \begin{cases} 
A & \text{if } s(A) = 0 \\
A\{1: \ldots : 1\} & \text{if } s(A) > 0
\end{cases}
$$

Where $\Gamma$ is a configuration of sort $i$ and $\Delta_1, \ldots, \Delta_i$ are configurations, the fold $\Gamma \otimes \langle \Delta_1 : \ldots : \Delta_i \rangle$ is the result of replacing the successive 1’s in $\Gamma$ by $\Delta_1, \ldots, \Delta_i$ respectively. Where $\Gamma$ is of sort $i$, the hyperoccurrence notation $\Delta(\Gamma)$ abbreviates $\Delta_0(\Gamma \otimes \langle \Delta_1 : \ldots : \Delta_i \rangle)$, that is, a context configuration $\Delta$ (which is externally $\Delta_0$ and internally $\Delta_1, \ldots, \Delta_i$) with a potentially discontinuous distinguished subconfiguration $\Gamma$. Where $\Delta$ is a configuration of sort $i > 0$ and $\Gamma$ is a configuration, the $k$th metalinguistic intercalation $\Delta|_k \Gamma$, $1 \leq k \leq i$, is given by:

$$
\Delta|_k \Gamma = \text{df } \Delta \otimes \langle 1: \ldots : 1: \Gamma: 1: \ldots : 1 \rangle
$$

that is, $\Delta|_k \Gamma$ is the configuration resulting from replacing by $\Gamma$ the $k$th separator in $\Delta$.

A semantically labelled sequent is a sequent in which the antecedent type occurrences $A_1, \ldots, A_n$ are labelled by distinct variables $x_1, \ldots, x_n$

³Note that only types of sort 0 can go into the stoup; reusable types of other sorts would not preserve the sequent antecedent-succedent sort equality under contraction.
1. \( \xi_1; \Gamma \Rightarrow B: \psi \quad \xi_2; \Delta(\overrightarrow{C}: z) \Rightarrow D: \omega /L \)
\( \xi_1 \cup \xi_2; \Delta(B: x, \Gamma) \Rightarrow D: \omega\{(x \psi)/z\} \)
\( \xi_1; \Gamma \Rightarrow C/B: \chi \quad \xi; \Gamma \Rightarrow \overrightarrow{C}/B: \lambda y \chi \quad /R \)

2. \( \xi_1; \Gamma \Rightarrow A: \phi \quad \xi_2; \Delta(\overrightarrow{C}: z) \Rightarrow D: \omega \)
\( \xi_1 \cup \xi_2; \Delta(\Gamma, A\overrightarrow{C}: y) \Rightarrow D: \omega\{(y \phi)/z\} \)
\( \xi_1; \Gamma \Rightarrow C/A: \chi \quad \xi; \Gamma \Rightarrow \overrightarrow{A}: x, \Gamma \Rightarrow C: \chi \quad /R \)

3. \( \xi; \Delta(\overrightarrow{A}: x, \overrightarrow{B}: y) \Rightarrow D: \omega L \)
\( \xi; \Delta(\overrightarrow{A}\bullet\overrightarrow{B}: z) \Rightarrow D: \omega\{\pi_1z/x, \pi_2z/y\} \)
\( \xi_1; \Gamma_1 \Rightarrow A: \phi \quad \xi_2; \Gamma_2 \Rightarrow B: \psi \quad \bullet R \)

4. \( \xi; \Delta(\overrightarrow{I}: x) \Rightarrow A: \phi \quad IL \quad \theta; \Lambda \Rightarrow I: 0 \quad IR \)

**Figure 10** Continuous multiplicatives

of types \( T(A_1), \ldots , T(A_n) \) respectively, and the succedent type \( A \) is labelled by a term of type \( T(A) \) with free variables drawn from \( x_1, \ldots , x_n \). In this appendix we give the semantically labelled Gentzen sequent rules for some primary connectives, and indicate some linguistic applications.

The continuous multiplicatives of figure 10, the Lambek connectives (Lambek 1958, 1988), defined in relation to appending, are the basic means of categorial categorization and subcategorization. Note that here and throughout the active types in antecedents are figures (vectorial) whereas those in succedents are not; intuitively this is because antecedents are structured but succedents are not. The directional divisions over, /, and under, \, are exemplified by assignments such as the: \( N/CN \) for the man: \( N \), sings: \( N/S \) for John sings: \( S \), and loves: \( (N/S)/N \) for John loves Mary: \( S \). The continuous product \( \bullet \) is exemplified by a ‘small clause’ assignment such as considers: \( (N/S)/(N\bullet(CN/CN)) \).

The discontinuous multiplicatives of figure 11, the displacement connectives (Morrill & Valentín 2010, Morrill et al. 2011), are defined in relation to plugging. When the value of the \( k \) subindex indicates the first (leftmost) point of discontinuity, it may be omitted. Extraction, \( \uparrow \), is exemplified by a discontinuous idiom assignment gives+1+the+cold+shoulder: \( (N/S)^{\uparrow}N \) for Mary gives John the cold shoulder: \( S \), and infixation, \( \downarrow \), and extraction together are exemplified by a quantifier phrase assignment everyone: \( (S^{\uparrow}N)^{\downarrow}S \), simulating Montague’s S14 treatment of quantifying in. Extrac-
allowing both peripheral and medial extraction, as in
ousunitinanassignmenttoarelativepronoun
	that

tion and discontinuous product, \odot, are shown together with the continuous unit in an assignment to a relative pronoun that: \((\text{CN}\backslash\text{CN})/((\text{S}\uparrow\text{N})\odot I))\), allowing both peripheral and medial extraction, as in that John likes: \(\text{CN}\backslash\text{CN}\) and that John saw today: \(\text{CN}\backslash\text{CN}\).

In relation to the multiplicative rules, notice how the stoup is distributed reading bottom-up from conclusions to premise: it is partitioned between the two premises in the case of binary rules, copied to the premise in the case of unary rules, and empty in the case of nullary rules (axioms).

The remaining figures give rules for additives, quantifiers, normal modalities, bracket modalities, exponentials, and limited contraction for anaphora.

**Acknowledgments** This research was partially financed by an ICREA Acadèmia 2012 to Morrill, MINECO project APCOM (TIN2014-57226-P), and Generalitat de Catalunya grant SGR2014-890 (MACDA). Many thanks to C. Piñón for valuable editorial guidance, and for anonymous EISS reviewing. All errors are our own.
11. \[ \Xi \langle A[t/v] : x \rangle \Rightarrow B: \psi \land L \]
\[ \Xi \langle \land vA : z \rangle \Rightarrow B: \psi \{z/x\} \]
\[ \Xi \Rightarrow A[a/v] : \phi \land R^\dagger \]
\[ \Xi \Rightarrow \land vA : \lambda v \phi \land R^\dagger \]

Figure 13 Quantifiers, where \(^\dagger\) indicates that there is no \(a\) in the conclusion

12. \[ \Xi \langle A[a/v] : x \rangle \Rightarrow B: \psi \lor L^\dagger \]
\[ \Xi \langle \lor vA : z \rangle \Rightarrow B: \psi \{\pi_2 z/x\} \]
\[ \Xi \Rightarrow A[t/v] : \phi \lor R \]
\[ \Xi \Rightarrow \lor vA : (t, \phi) \lor R \]

Figure 14 Normal modalities, where \(\boxslash/\boxdot\) marks a structure all the types of which have main connective a box/diamond

13. \[ \Xi \langle \boxslash A : x \rangle \Rightarrow B: \psi \boxslash L \]
\[ \Xi \langle \boxslash \land A : z \rangle \Rightarrow B: \psi \{\land z/x\} \]
\[ \Xi \boxslash \Rightarrow A: \phi \boxslash R \]
\[ \Xi \boxslash \Rightarrow \boxslash A: ^\phi \boxslash R \]

Figure 15 Bracket modalities

15. \[ \Xi \langle [\ ]^{-1} A : x \rangle \Rightarrow B: \psi \]
\[ \Xi \Rightarrow [\ ]^{-1} A: \phi \]
\[ \Xi \Rightarrow \{\ ]^{-1} R \]

16. \[ \Xi \langle \langle A : x \rangle \Rightarrow B: \psi \rangle L \]
\[ \Xi \langle \langle \rangle A : x \rangle \Rightarrow B: \psi \]
\[ \Xi \Rightarrow \langle \rangle A: \phi \rangle R \]

Figure 16 Exponentials


\[
\begin{align*}
\zeta; \Gamma \Rightarrow A: \phi & \quad \zeta^*; \Delta(A; x; \overline{B}; y) \Rightarrow D: \omega \\
\zeta \land \zeta^*; \Delta(\Gamma; \overline{B}; z) & \Rightarrow D: \omega(x \land y) \\
\zeta; \Gamma(\overline{B}_0; \ldots; \overline{B}_n; y_0) & \Rightarrow D: \omega \\
\zeta; \Gamma(\overline{B}_0; \ldots; \overline{B}_n; z_0) & \Rightarrow D: A: \lambda x \omega((z_0 x)/y_0, \ldots, (z_n x)/y_n)
\end{align*}
\]


Morrill, Glyn & Josep-Maria Merenciano. 1996. Generalising discontinuity. Traite-
m ent automatique des langues 37(2). 119–143.


Morrill, Glyn & Oriol Valentín. 2015a. Computational coverage of tlg: Nonlinear-


Morrill, Glyn, Oriol Valentín & Mario Fadda. 2009. Dutch grammar and process-

