Parsing Logical Grammar: CatLog3

Glyn Morrill
Department of Computer Science
Universitat Politècnica de Catalunya
Barcelona, Spain.
morrill@cs.upc.edu

Abstract
CatLog3 is a Prolog parser/theorem-prover for (type) logical (categorial) grammar. In such logical grammar, grammar is reduced to logic: a string of words is grammatical if and only if an associated logical statement is a theorem. CalLog3 implements a logic extending displacement calculus, a sublinear fragment including as primitive connectives the continuous (Lambek) and discontinuous wrapping connectives of the displacement calculus, additives, 1st order quantifiers, normal modalities, bracket modalities and subexponentials. In this paper we survey how CatLog3 is implemented on the principles of Andreoli’s focusing and a generalisation of van Benthem’s count-invariance.

1 Introduction
The linguistics that has descended from formal grammar as popularised by Chomsky (1957[6]) has renegaded on formalisation, and discrete computational grammar in the genre of the 1980s has given way to statistical NLP. However, there is a venerable older line of linguistics practicing grammar according to the standards of mathematical logic. The seminal paper in this line is Lambek (1958[19]) which defines a syntactic calculus and proves Cut-elimination for it, but the tradition dates back at least to Bar-Hillel (1953[3]) and Ajdukiewicz (1935[1]). The Lambek calculus is a calculus of concatenation which is free of structural rules. The displacement calculus of Morrill et al. (2011[46]) generalises Lambek calculus with intercalation, containing both continuous and discontinuous connective families, while remaining free of structural rules, and preserving Cut-elimination and its good corollaries: the subformula property, decidability, the finite reading property, and the focusing property. There are several monographs and reference articles on this type logical approach: Moortgat (1988[22]; 1997[24]), Morrill (1994[47];

*This work was partially supported by an ICREA Academia 2012, and MINECO project APCOM (TIN2014-57226-P). Thanks to Oriol Valentín for discussions in relation to this work.

- CatLog1 (Morrill 2012[33]) was based on the method of uniform proof (Miller et al. 1991[20]), and the method of count-invariance for multiplicatives (van Benthem 1991[51]).

- CatLog2 was based on Andreoli’s focusing (Andreoli 1992[2]), and count-invariance for multiplicatives, additives and bracket modalities (Valentín et al. 2013[50]).

- CatLog3 is based on focalisation and count-invariance for multiplicatives, additives, bracket modalities and exponentials (Kuznetov et al. 2017[16]).

In this paper we survey the methods on which the implementation of CatLog3 is based. In Section 2 we describe the primitive connectives of the logical fragment for which parsing/theorem-proving is implemented. In Sections 3 and 4 we discuss focusing and count-invariance respectively. In Section 5 we illustrate in relation to the Montague Test (Morrill and Valentín 2016[43]): the task of providing a computational grammar of Montague’s (1973[21]) fragment.

## 2 Displacement logic

The formalism used comprises the connectives of Table 1. The heart of the logic is the displacement calculus of Morrill and Valentín (2010[38]) and Morrill, Valentín and Fadda (2011[46]) made up of twin continuous and discontinuous residuated families of connectives having a pure Gentzen sequent calculus —without labels and free of structural rules— and enjoying Cut-elimination (Valentín 2012[49]). Other primary connectives include additives, 1st order quantifiers, normal (i.e. distributive) modalities, bracket (i.e. nondistributive) modalities, and exponentials.\(^1\)

We can draw a clear distinction between the primary connectives, the semantically inactive connectives, and the synthetic connectives; the latter two are abbreviatory and are there for convenience, and to simplify derivation. There are semantically inactive variants of the continuous and discontinuous multiplicatives, and semantically inactive variants of the additives, 1st order quantifiers, and normal modalities.\(^2\) Synthetic connectives (Girard 2011[11]) divide into the continuous

\(^1\) Once Cut-elimination is established, the only challenge to decidability comes from nonlinearity: the contraction rule of the universal exponential, and the infinitary left rule of the existential exponential. In this connection, linguistically the existential left rule is not required; and Morrill and Valentín (2015[41]) introduced displacement logics $\text{Db}!\phi$ and $\text{Db}!\phi$ with a relevant modality !, with and without bracket conditioning for contraction. Kanovich et al. (2016[15]) prove the undecidability of $\text{Db}!\phi$ and in unpublished work announce the undecidability of $\text{Db}!\phi$. But Morrill and Valentín (2015[41]) prove the decidability of a linguistically sufficient ‘bracket non-negative’ special case of $\text{Db}!\phi$.

\(^2\) For example, the semantically inactive additive conjunction $A\triangleq B$: $\phi$ abbreviates $A\&B$: ($\phi, \phi$).
<table>
<thead>
<tr>
<th>primary</th>
<th>cont. mult.</th>
<th>disc. mult.</th>
<th>add.</th>
<th>qu.</th>
<th>norm. mod.</th>
<th>brack. mod.</th>
<th>exp.</th>
<th>lim. contr. &amp; weak.</th>
</tr>
</thead>
<tbody>
<tr>
<td>/ · \</td>
<td>↑ ⊗ ↓ J</td>
<td>&amp; ^ □</td>
<td>[ ]^{-1}</td>
<td>!</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>J</td>
<td>V ◇ ⟨⟩</td>
<td>?</td>
<td></td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sem. inactive variants</td>
<td>← → ← → → → →</td>
<td>⊠ ⊠ ⊠ ⊠ ⊠ ⊠</td>
<td>⊡ ⊡ ⊡ ⊡</td>
<td>⊡ ⊡ ⊡ ⊡</td>
<td>⊡ ⊡ ⊡ ⊡</td>
<td>⊡ ⊡ ⊡ ⊡</td>
<td></td>
<td></td>
</tr>
<tr>
<td>det. synth.</td>
<td>←^{-1} ←^{-1}</td>
<td>~</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nondet. synth.</td>
<td>÷</td>
<td>⊤ ⊥</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Categorial connectives
1. \( F_i \ ::= \ F_{i+j} / F_j \quad T(C/B) = T(B) \rightarrow T(C) \) over
2. \( F_j \ ::= \ F_{i+j} / F_i \quad T(A \land C) = T(A) \land T(C) \) under
3. \( F_{i+j} \ ::= \ F_i \bullet F_j \quad T(A \land B) = T(A) \land T(B) \) continuous product
4. \( F_0 \ ::= \ I \quad T(I) = T \) continuous unit
5. \( F_{i+1} \ ::= \ F_{i+j} / F_{i+j} \quad T(C \land_1 F_{i+j}) = T(B) \rightarrow T(C) \) circumflex
6. \( F_{i+1} \ ::= \ F_{i+j} / F_{i+j} \quad T(A \land_1 C) = T(A) \land T(C) \) infix
7. \( F_{i+j} \ ::= \ F_{i+j} \circ F_j \quad T(A \land_2 B) = T(A) \land T(B) \) discontinuous product
8. \( F_1 \ ::= \ J \quad T(J) = T \) discontinuous unit
9. \( F_i \ ::= \ F_i \& F_i \quad T(A \& B) = T(A) \& T(B) \) additive conjunction
10. \( F_i \ ::= \ F_i \oplus F_i \quad T(A \oplus B) = T(A) \oplus T(B) \) additive disjunction
11. \( F_i \ ::= \ \land F_i \quad T(\land vA) = F \rightarrow T(A) \) 1st order universal qu.
12. \( F_i \ ::= \ \lor F_i \quad T(\lor vA) = F \& T(A) \) 1st order exist. qu.
13. \( F_i \ ::= \ \Box F_i \quad T(\Box A) = LT(A) \) universal modality
14. \( F_i \ ::= \ \Diamond F_i \quad T(\Diamond A) = MT(A) \) existential modality
15. \( F_i \ ::= \ [ ]^{-1} F_i \quad T([ ]^{-1} A) = T(A) \) univ. bracket modality
16. \( F_i \ ::= \ \langle F_i \quad T(\langle A) = T(A) \) exist. bracket modality
17. \( F_0 \ ::= \ ? F_0 \quad T(?A) = T(A) \) universal exponential
18. \( F_0 \ ::= \ ! F_0 \quad T(!A) = T(A)^* \) existential exponential

Table 2: Syntactic types of \( DA1S4b!_b \)

and discontinuous deterministic (unary) synthetic connectives, and the continuous and discontinuous nondeterministic (binary) synthetic connectives.\(^3\)

2.1 Syntactic types

The syntactic types of displacement logic are sorted \( F_0, F_1, F_2, \ldots \) 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 of \( DA1S4b!_b \) are formed as illustrated in Table 2, and the structure preserving semantic type map \( T \) associates these with semantic types.

2.2 Gentzen sequent calculus

We use a Gentzen sequent presentation standard from Gentzen (1934[9]) and Lambek (1958[19]). In Gentzen sequent antecedents for displacement logic with bracket modalities (structural inhibition) and exponentials (structural facilitation) there are also bracket constructors and ‘stoups’.

\( Stoups \) (cf. the linear logic of Girard 2011[11]) (ζ) 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). The stoup of linear logic is for resources which can be contracted (copied)

\(^3\)For example, the nondeterministic continuous division \( B \div A \) abbreviates \( (A \div B) \div (B/A) \).
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 here the stoup
is used for resources which are also commutative. A configuration together with a
stoup is a zone (Ξ). 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 (θ is the empty stoup; Λ is
the empty configuration; the separator 1 marks points of discontinuity):

\[
\begin{align*}
S & ::= \emptyset \mid T_0, S \\
O & ::= \Lambda \mid T, O \\
T & ::= 1 \mid T_0 \mid T_{>0}(O : \ldots : O) \mid [S; O]
\end{align*}
\]

For a type A, its sort s(A) is the i such that A ∈ F_i. For a configuration Γ, its sort
s(Γ) is |Γ|_j, i.e. the number of points of discontinuity 1 which it contains. Sequents
are of the form:

(2) S; O ⇒ T such that s(O) = s(T)

The figure A of a type A is defined by:

\[
(3) A = \begin{cases} 
A & \text{if } s(A) = 0 \\
A[1 : \ldots : 1] & \text{if } s(A) > 0 \\
\end{cases}
\]

Where Γ is a configuration of sort i and Δ_1, . . . , Δ_i are configurations, the fold
Γ ⊗ (Δ_1 : . . . : Δ_i) is the result of replacing the successive 1’s in Γ by Δ_1, . . . , Δ_i
respectively. Where Γ is of sort i, the hyperoccurrence notation Δ(Γ) abbreviates
Δ_0(Γ ⊗ (Δ_1 : . . . : Δ_i)), i.e. a context configuration Δ (which is externally Δ_0 and
internally Δ_1, . . . , Δ_i) with a potentially discontinuous distinguished subconfiguration
Γ (continuous if i = 0, discontinuous if i > 0). Where Δ is a configuration of
sort i > 0 and Γ is a configuration, the kth metalinguistic intercalation Δ |_k Γ, 
1 ≤ k ≤ i, is given by:

\[
(4) \Delta |_k \Gamma =_{df} \Delta \otimes (\underbrace{1 : \ldots : 1}_k : \underbrace{\Gamma : 1 : \ldots : 1}_{i-k})
\]

i.e. Δ |_k Γ is the configuration resulting from replacing by Γ the kth separator in Δ.

2.3 Rules and linguistic applications

A semantically labelled sequent is a sequent in which the antecedent type occur-
rences A_1, . . . , A_n are labelled by distinct variables x_1, . . . , x_n which are of types

\footnote{Note 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 or expansion: 0 + 0 = 0,
but i + i ≠ i for i > 0.}
$T(A_1), \ldots, T(A_n)$ 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 section we give the semantically labelled Gentzen sequent rules for the connectives of DAIS4b1b?, and indicate some linguistic applications.

\[
\begin{align*}
1. & \frac{\zeta_1; \Gamma \Rightarrow B; \psi}{\Xi(\zeta_2; \Delta(C, z)) \Rightarrow D; \omega} / L & \frac{\zeta_1; \Gamma \Rightarrow C; \chi}{\zeta; \Gamma \Rightarrow C/B; \chi} / R \\
2. & \frac{\zeta_2; \Delta(C, z) \Rightarrow D; \omega}{\Xi(\zeta_1 \cup \zeta_2; A \setminus C; y) \Rightarrow D; \omega(y \phi / z)} \setminus L & \frac{\zeta_2; \Gamma \Rightarrow C; \chi}{\zeta; \Gamma \Rightarrow A; A \setminus C; \chi} \setminus R \\
3. & \frac{\Xi(\zeta_2; \Delta(C, z)) \Rightarrow D; \omega}{\Xi(\zeta_1 \cup \zeta_2; A \setminus C; y) \Rightarrow D; \omega(y \phi / z)} \bullet L & \frac{\zeta_1 \cup \zeta_2; \Gamma \Rightarrow B; \psi}{\zeta_1; \Delta \Rightarrow A; \phi \quad \zeta_2; \Gamma \Rightarrow B; \psi} \bullet R \\
4. & \frac{\Xi(\Lambda)}{\Xi(\zeta_1; x)} \Rightarrow A; \phi} & \frac{\emptyset; \Lambda \Rightarrow \Gamma; 0}{IL \quad IR}
\end{align*}
\]

Figure 1: Lambek multiplicatives

The continuous multiplicatives, the Lambek connectives of Lambek (1958[19]; 1988[18]), Figure 1, defined in relation to concatenation/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, $\downarrow$, and under, $\downarrow$, are exemplified by assignments such as the: $N/\text{CN}$ for the man: $N$ and $\text{sings}: N \setminus S$ for John sings: $S$, and loves: $(N \setminus S)/N$ for John loves Mary: $S$.

The discontinuous multiplicatives of Figure 2, the displacement connectives, Morrill and Valentín (2010[38]), Morrill et al. (2011[46]), are defined in relation to intercalation/plugging. When the value of the $k$ subindex indicates the first (leftmost) point of discontinuity it may be omitted, i.e. it defaults to 1. Circumfixation, $\uparrow$, is exemplified by a discontinuous particle verb assignment such as calls+1+up: $(N \setminus S) \uparrow N$ for Mary calls John up: $S$, and infixation, $\downarrow$, and circumfixation together are exemplified by a quantifier phrase assignment of the form everyone: $(S \uparrow N) \downarrow S$ simulating Montague’s S14 treatment of quantifying in; see Section 5.

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 additives of Figure 3, Lambek (1961[17]), Morrill (1990[28]), Kanazawa (1992[14]), have application to polymorphism. For example the additive conjunc-
Tully is Cicero

cally the latter does not capture the generalisation that in both cases the verb eats operators were allowed they could occur any number of times in any positions.

at best a promissory solution, unless there is true ambiguity.

Note the advantage of this over simply listing intransitive and transitive lexical entries: empirically the latter does not capture the generalisation that in both cases the verb eats combines with a subject to the left, and computationally every lexical ambiguity doubles the lexical insertion search space. Appeal to lexical ambiguity constitutes resignation from the capture of generalisations and is at best a promissory solution, unless there is true ambiguity.

Figure 2: Displacement multiplicatives

Figure 3: Additives

Note the computational advantage of this approach over assuming an empty determiner: if empty operators were allowed they could occur any number of times in any positions.

Note the advantage of this over simply listing intransitive and transitive lexical entries: empirically the latter does not capture the generalisation that in both cases the verb eats combines with a subject to the left, and computationally every lexical ambiguity doubles the lexical insertion search space. Appeal to lexical ambiguity constitutes resignation from the capture of generalisations and is at best a promissory solution, unless there is true ambiguity.
The quantifiers of Figure 4, Morrill (1994[47]), have application to features. For example, singular and plural number in sheep: $\wedge nCNn$ for the sheep grazes: $S$ and the sheep graze: $S$. And for a past, present or future tense finite sentence complement we can have said: $(N,S)/\vee tS$ $f(t)$ in John said Mary walked: $S$, John said Mary walks: $S$ and John said Mary will walk: $S$.

Notice how the stoup is identical in conclusion and premise in each quantifier rule.

With respect to the $(S4)$ normal modalities of Figure 5, the universal (Morrill 1990[29]) has application to intensionality. For example, for a propositional attitude verb such as believes we can assign type $\Box((N,S)/\Box S)$ with a modality outermost since the word has a sense, and a modality on the first argument but not the second, since the sentential complement is an intensional domain, but not the subject. The modalities are in the categorial type, distinctly from, but in relation to, the logical interpretation of the propositional attitude verb. The $\Box$ Right rule is semantically interpreted by intensionalisation $\wedge$ and the $\Box$ Left rule is semantically interpreted by extensionalisation $\vee$ in such a way that the Curry-Howard correspondence for the modality yields the law of down-up cancellation (Dowty et al. 1981[7]): $\vee \wedge \phi = \phi$.

Notice how the stoup is identical in conclusion and premise in each normal modality rule.

The bracket modalities of Figure 6, Morrill (1992[30]) and Moortgat 1995[23]), have application to nonassociativity and syntactical domains such as extraction.
islands and prosodic phrases. For example, single bracketing for weak islands: \textit{walks:} ⟨⟩N,S for the subject condition, and \textit{without:} [ ]−1(VP/VP)/VP for the adverbial island constraint; and double bracketing for strong islands of the kind \textit{and:} (S [ ]−1[S])/S for the coordinate structure constraint.

Notice how the stoup is identical in conclusions and premises of bracket modality rules.

Finally, there is nonlinearity. The universal exponential of Figure 7, Girard (1987[10]), Barry, Hepple, Leslie and Morrill (1991[4]), Morrill (1994[47]), Morrill and Valentín (2015[41]), and Morrill (2017[35]), has application to extraction including parasitic extraction. In the formulation here !L moves the operand of a universal exponential (e.g. the hypothetical subtype of relativisation) into the stoup, where it will percolate as commented for the above rules. From there it can be copied into the stoup of a newly-created bracketed domain by the contraction rule !C (producing a parasitic gap), and it can be moved into any position in the matrix configuration of its zone by !P (producing a nonparasitic or host gap).
Using the universal exponential, $!$, for which contraction induces island brackets, we can assign a relative pronoun type that: $(CN\backslash CN)/(S/\backslash N)$ allowing parasitic extraction such as paper that John filed without reading: $CN$, where parasitic gaps can appear only in (weak) islands, but can be iterated in subislands, for example, man who the fact that the friends of admire without praising surprises. Crucially, in the linguistic formulation $!$ does not have weakening, i.e. deletion, since, e.g., the body of a relative clause must contain a gap: *man who John loves Mary.

The existential exponential $?$ has application to iterated coordination (Morrill 1994[47]; Morrill and Valentín 2015[41]) and (unboundedly iterated) respectively (Morrill and Valentín 2016[44]). Using the existential exponential, $?$, we can assign a coordinator type and: $(?N)/N$ allowing iterated coordination as in John, Bill, Mary and Suzy: $N$, or and: $(?S/\backslash N)/(S/\backslash N)$ for John likes Mary dislikes, and Bill hates, London (iterated right node raising), and so on.

In relation to the rest of the primary connectives: the limited contraction $|$ of Jäger (2005[13]) has application to anaphora and the limited weakening $W$ of Morrill and Valentín (2014[40]) has application to words as types. The remaining, semantically inactive, connectives listed here were introduced as follows. Semantically inactive multiplicatives $\left\{\leftrightarrow, \rightarrow, \leftarrow, \oplus, \bigcirc, \uparrow, \downarrow, \triangleleft, \triangleright\right\}$: Morrill and Valentín (2014[40]). Semantically inactive additives $\left\{\sqcap, \sqcup\right\}$: Morrill (1994[47]). Semantically inactive first-order quantifiers $\left\{\forall, \exists\right\}$: Morrill (1994[47]). Semantically inactive normal modalities $\left\{\lozenge, \Box\right\}$: Hepple (1990[12]), Morrill (1994[47]). The rules for semantically inactive variants are the same as those for the semantically active versions syntactically, but have the same label on premises and conclusions semantically.\(^7\)

### 3 Focusing

Spurious ambiguity is the phenomenon whereby distinct derivations in grammar may assign the same structural reading, resulting in redundancy in the parse search space and inefficiency in parsing. Understanding the problem depends on identifying the essential mathematical structure of derivations. This is trivial in the case of context free grammar, where the parse structures are ordered trees; in the case of type logical categorial grammar, the parse structures are proof nets. However, with respect to multiplicatives intrinsic proof nets have not yet been given for displacement calculus (but see Morrill and Fadda (2008[36], Fadda 2010[8], and Moot 2014[25], 2016[26]) In this context CatLog3 approaches spurious ambiguity by means of Andreoli’s (1982[2]) proof-theoretic technique of focalisation, which engenders a substantial reduction of spurious ambiguity.

\(^7\)The synthetic connectives are: left and right projection and injection $\left\{\leftarrow^-, \rightarrow^-, \leftarrow, \rightarrow, \oplus, \bigcirc\right\}$, Morrill, Fadda and Valentín (2009[45]); split and bridge $\left\{\wedge, \vee\right\}$, Morrill and Merenciano (1996[37]); continuous and discontinuous nondeterministic multiplicatives $\left\{\div, \times, \uparrow, \downarrow, \bigcirc\right\}$, Morrill, Valentín and Fadda (2011[46]). The difference operator $-\Delta$ of Morrill and Valentín (2014[39]) has application to linguistic exceptions.
In focalisation, situated (in the antecedent of a sequent, input, *) in the succe-
dent of a sequent, output, *) non-atomic types are classified as of reversible/negative
or irreversible/positive polarity according as their associated rule is reversible or
not. There are alternating phases of don’t-care nondeterministic negative rule ap-
lication, and positive rule application locking on to focalised
formulas. Given a
sequent with no occurrences of negative formulas, one chooses a positive formula
as principal formula (which is boxed; we say it is focalised) and applies proof
search to its subformulas while these remain positive. When one finds a negative
formula or a literal, invertible rules are applied in a don’t care nondeterministic fash-
ion until no longer possible, when another positive formula is chosen, and so on.
CatLog3 can be set to focus all atoms in the input (as in the example at the end) or
in the output, i.e. it implements uniform bias.

A sequent is either unfocused and as before, or else focused and has exactly one
type boxed. This is the focused type. The focalised logical rules for displacement
calculus are given in Figures 8–12. Sequents are accompanied by judgements:
 focalised or not focalised and reversible or not reversible.\(^8\) The completeness of this
focalisation, together with additives, is proved in Morrill and Valentín (2015[42]).
The completeness of focalisation for other connectives of CatLog3 is a topic of
ongoing research.

\(^8\)This idea is due to Oriol Valentín.
Γ ⇒ [P: φ  foc ∧ ¬ rev] \[Δ(\overrightarrow{Q}: z) ⇒ Δ(Γ, Q: y) ⇒ D: ω((y φ)/z)] \ foc ∧ ¬ rev \]

Γ ⇒ [P_1: φ  foc ∧ ¬ rev] \[Δ(Δ(Γ, P_1/P_2: z) ⇒ D: ω((y φ)/z)] \ foc ∧ ¬ rev \]

Γ ⇒ [Q_1: φ  ¬ foc ∧ ?Q_1 rev] \[Δ(Δ(Γ, Q_1/Q_2: z) ⇒ D: ω((y φ)/z)] \ foc ∧ ¬ rev \]

Γ ⇒ [Q: φ  ¬ foc ∧ ?Q rev] \[Δ(Δ(Γ, Q: y) ⇒ D: ω((y φ)/z)] \ foc ∧ ¬ rev \]

Γ ⇒ [P: ψ  foc ∧ ¬ rev] \[Δ(Δ(Γ, Q: z) ⇒ D: ω((x ψ)/z)] \ foc ∧ ¬ rev \]

Γ ⇒ [Q_1: ψ  ¬ foc ∧ ?Q_1 rev] \[Δ(Δ(Γ, Q_2/Q_1: x, y) ⇒ D: ω((x ψ)/z)] \ foc ∧ ¬ rev \]

Γ ⇒ [P_1: ψ  foc ∧ ¬ rev] \[Δ(Δ(Γ, P_2/P_1: x, y) ⇒ D: ω((x ψ)/z)] \ foc ∧ ¬ rev \]

Γ ⇒ [Q: ψ  ¬ foc ∧ ?Q rev] \[Δ(Δ(Γ, Q: x) ⇒ D: ω((x ψ)/z)] \ foc ∧ ¬ rev \]

Figure 9: Left irreversible continuous multiplicative rules
Figure 10: Left irreversible discontinuous multiplicative rules
\[\Delta \Rightarrow \begin{array}{c} P_1 \phi \text{ foc} \wedge \neg \text{ rev} \\ \Gamma \Rightarrow P_2 \psi \text{ foc} \wedge \neg \text{ rev} \end{array} \quad \Rightarrow \quad \Gamma \Rightarrow Q: \psi \neg \text{ foc} \wedge ?Q \text{ rev} \]

\[\Delta, \Gamma \Rightarrow \begin{array}{c} P_1 \bullet P_2 \phi, \psi \text{ foc} \wedge \neg \text{ rev} \end{array} \quad \Rightarrow \quad \Delta, \Gamma \Rightarrow \begin{array}{c} P \bullet Q \phi, \psi \text{ foc} \wedge \neg \text{ rev} \end{array} \quad \Rightarrow \quad \Delta, \Gamma \Rightarrow \begin{array}{c} N \bullet P \phi, \psi \text{ foc} \wedge \neg \text{ rev} \end{array} \quad \Rightarrow \quad \Delta, \Gamma \Rightarrow \begin{array}{c} N \bullet Q \phi, \psi \text{ foc} \wedge \neg \text{ rev} \end{array} \quad \Rightarrow \quad \Delta, \Gamma \Rightarrow \begin{array}{c} N_1 \bullet N_2 \phi, \psi \text{ foc} \wedge \neg \text{ rev} \end{array} \quad \Rightarrow \quad \Delta, \Gamma \Rightarrow \begin{array}{c} N_1 \bullet Q_2 \phi, \psi \text{ foc} \wedge \neg \text{ rev} \end{array} \quad \Rightarrow \quad \Delta, \Gamma \Rightarrow \begin{array}{c} Q_1 \bullet Q_2 \phi, \psi \text{ foc} \wedge \neg \text{ rev} \end{array} \]

Figure 11: Right irreversible continuous multiplicative rules

\[\Delta \Rightarrow \begin{array}{c} P_1 \phi \text{ foc} \wedge \neg \text{ rev} \\ \Gamma \Rightarrow P_2 \psi \text{ foc} \wedge \neg \text{ rev} \end{array} \quad \Rightarrow \quad \Gamma \Rightarrow Q: \psi \neg \text{ foc} \wedge ?Q \text{ rev} \]

\[\Delta, \Gamma \Rightarrow \begin{array}{c} P_1 \bullet P_2 \phi, \psi \text{ foc} \wedge \neg \text{ rev} \end{array} \quad \Rightarrow \quad \Delta, \Gamma \Rightarrow \begin{array}{c} P \bullet Q \phi, \psi \text{ foc} \wedge \neg \text{ rev} \end{array} \quad \Rightarrow \quad \Delta, \Gamma \Rightarrow \begin{array}{c} N \bullet P \phi, \psi \text{ foc} \wedge \neg \text{ rev} \end{array} \quad \Rightarrow \quad \Delta, \Gamma \Rightarrow \begin{array}{c} N \bullet Q \phi, \psi \text{ foc} \wedge \neg \text{ rev} \end{array} \quad \Rightarrow \quad \Delta, \Gamma \Rightarrow \begin{array}{c} N_1 \bullet N_2 \phi, \psi \text{ foc} \wedge \neg \text{ rev} \end{array} \quad \Rightarrow \quad \Delta, \Gamma \Rightarrow \begin{array}{c} N_1 \bullet Q_2 \phi, \psi \text{ foc} \wedge \neg \text{ rev} \end{array} \quad \Rightarrow \quad \Delta, \Gamma \Rightarrow \begin{array}{c} Q_1 \bullet Q_2 \phi, \psi \text{ foc} \wedge \neg \text{ rev} \end{array} \]

Figure 12: Right irreversible discontinuous multiplicative rules
4 Count-invariance

We define infinitary count invariance for categorial logic extending count invariance for multiplicatives (van Benthem 1991[51]) and additives and bracket modalities (Valentín et al. 2013[50]) to include exponentials. This affords effective pruning of proof search in categorial parsing/theorem-proving.

Count invariance for multiplicatives in (sub)linear logic is introduced in van Benthem (1991[51]). This involves simply checking the number of positive and negative occurrences of each atom in a sequent. Thus where \( \#(\Sigma) \) is a count of the sequent \( \Sigma \) we have:

\[
(5) \quad \vdash \Sigma = \Rightarrow \#(\Sigma) = 0
\]

I.e. the numbers of positive and negative occurrences of each atom must exactly balance. This provides a necessary, but of course not sufficient, criterion for theoremhood, and it can be checked rapidly. It can be used as a filter in proof search: if backward chaining proof search generates a goal which does not satisfy the count invariant, the goal can be safely made to fail immediately. This notion of count for multiplicatives was included in the categorial parser/theorem-prover CatLog1 (Morrill 2012[33]).

In Valentín et al. (2013[50]) the idea is extended to additives (and bracket modalities). Instead of a single count for each atom of a sequent \( \Sigma \) we have a minimum count \( \#_{\min}(\Sigma) \) and a maximum count \( \#_{\max}(\Sigma) \) and for a sequent to be a theorem it must satisfy two inequations:

\[
(6) \quad \vdash \Sigma = \Rightarrow \#_{\min}(\Sigma) \leq 0 \leq \#_{\max}(\Sigma)
\]

I.e. the count functions \( \#_{\min} \) and \( \#_{\max} \) define an interval which must include the point of balance 0; for the multiplicatives, \( \#_{\min} = \#_{\max} = \# \) and (6) reduces to the special case (5). This count-invariance is included in the categorial parser/theorem-prover CatLog2. Here we describe the count-invariance of CatLog3 which includes further infinitary count functions for exponentials (Kuznetsov et al. 2017[16]).

We consider terms built over the constants 0, 1, ⊥ (minus infinity, −∞), and ⊤ (plus infinity, +∞) by operations plus (+), minus (−), minimum (min) and maximum (max), and infinitary step functions \( X \) and \( Y \) thus; \( i, j \in \mathbb{Z} \) and \( n \in \mathbb{Z}^+ \):

\[
\begin{array}{c|c|c|c}
+ & j & \perp & \top \\
\hline
i & i+j & \perp & \top \\
\perp & \perp & \perp & \ast \\
\top & \top & \top & \ast \\
\end{array}
\quad
\begin{array}{c|c|c|c}
- & j & \perp & \top \\
\hline
i & i-j & \top & \perp \\
\perp & \perp & \ast & \perp \\
\top & \top & \top & \ast \\
\end{array}
\]

\[
\begin{array}{c|c|c|c}
\min & j & \perp & \top \\
\hline
i & \frac{i+j+|i-j|}{2} & \perp & i \\
\perp & \perp & \perp & \perp \\
\top & j & \perp & \top \\
\end{array}
\quad
\begin{array}{c|c|c|c}
\max & j & \perp & \top \\
\hline
i & \frac{i+j+|i-j|}{2} & i & \top \\
\perp & \perp & \perp & \perp \\
\top & \top & \top & \top \\
\end{array}
\]
\[ X(i) = \begin{cases} \top & \text{if } i > 0 \\ i & \text{if } i \leq 0 \end{cases} \]

\[ Y(i) = \begin{cases} i & \text{if } i \geq 0 \\ \bot & \text{if } i < 0 \end{cases} \]

Where \( \mathcal{P} \) is the set of primitive types, \( P \in \mathcal{P}, Q \in \mathcal{P} \cup \{[]\} \), \( p \in \{\star, \circ\} \), and \( \overline{e} = \circ \) and \( \overline{\circ} = \star \) we define the count functions for \( DAISb_4 \) as shown in Figure 13.

For zones, stoup, tree terms and configurations, counts are as follows:

\[
\begin{align*}
\#_{m,Q}(S; O) &= \#_{m,Q}(S) + \#_{m,Q}(O) \\
\#_{m,Q}(\emptyset) &= 0 \\
\#_{m,Q}(\mathcal{F}, S) &= \#_{m,Q}(\mathcal{F}) + \#_{m,Q}(S) \\
\#_{m,Q}(\Lambda) &= 0 \\
\#_{m,Q}(\mathcal{T}, O) &= \#_{m,Q}(\mathcal{T}) + \#_{m,Q}(O) \\
\#_{m,Q}(1) &= 0 \\
\#_{m,Q}(\mathcal{F}) &= \#_{m,Q}(\mathcal{F}) \\
\#_{m,Q}(\mathcal{F}[O_1 : \ldots : O_i]) &= \#_{m,Q}(\mathcal{F}) + \Sigma_{n=1}^{i} \#_{m,Q}(O_n) \\
\#_{m,P}([Z]) &= \#_{m,P}([Z]) \\
\end{align*}
\]

The count-invariance theorem is:

(7) Theorem.

\[ \vdash \Xi \Rightarrow A \implies \forall Q \in \mathcal{P} \cup \{[]\}, \#_{\min,Q}(\Xi \Rightarrow A) \leq 0 \leq \#_{\max,Q}(\Xi \Rightarrow A) \]

where, \( \#_{m,Q}(\Xi \Rightarrow A) = \#_{m,Q}(A) - \#_{\overline{e},Q}(\Xi) \).

Relativisation including medial and parasitic extraction is obtained by assigning a relative pronoun a type \((CN,N)!(N)\) whereby the body of a relative clause is analysed as \(N!N\). By way of example of count-invariance, we show how it discards \(N,N\!\!N\Rightarrow N!N\) corresponding to the ungrammaticality of a relative clause without a gap: *paper that John walks*. We have the max \(N\)-count:

\[
\begin{align*}
\#_{\max,N}(N,N!N\Rightarrow N!N) &= \#_{\max,N}((N)!N\!\!N\!\!N) - \#_{\min,N}(N,N) \!\!N\!\!N\!\!N = \#_{\max,N}(S) - \#_{\min,N}(N) - \#_{\min,N}(N\!\!N\!\!N\!\!N) = 0 - Y(\#_{\min,N}(N)) - 1 - \#_{\min,N}(S) + \#_{\min,N}(N) = -Y(1) - 1 - 0 + 1 = -1 - 1 + 1 = -1 \not\geq 0 \text{ which means that the count-invariance is not satisfied.}
\end{align*}
\]

Iterated sentential coordination is obtained by assigning a coordinator the type \((?S\!\!S)/S\). By way of a second example we show how count-invariance discards \(N,N,N\!\!S \Rightarrow ?S\) corresponding to the ungrammaticality of unequilibrated coordination: *John Mary walks and Suzy talks*. For this example maximum \(N\)-count is:

\[
\begin{align*}
\#_{\max,N}(N,N,N!S \Rightarrow ?S) &= \#_{\max,N}(S) - \#_{\min,N}(N,N,N!S) = X(\#_{\max,N}(S)) - \#_{\min,N}(N) - \#_{\min,N}(N!S) = X(0) - 1 - 1 = 0 - 2 - 0 + 1 = -1 \not\geq 0 \text{ which means that the count-invariance is not satisfied.}
\end{align*}
\]
\[
\begin{align*}
\#_{m,Q}(P) &= 1 \text{ if } Q = P \\
0 &\text{ if } Q \neq P \\
\#_{m,Q}(A \setminus C) &= \#_{m,Q}(A \downarrow_c C) = \#_{m,Q}(C) - \#_{m,Q}(A) \\
\#_{m,Q}(C \uparrow B) &= \#_{m,Q}(C \uparrow B) = \#_{m,Q}(C) - \#_{m,Q}(B) \\
\#_{m,Q}(A \cdot B) &= \#_{m,Q}(A \cap B) = \#_{m,Q}(A) + \#_{m,Q}(B) \\
\#_{m,Q}(J) &= \#_{m,Q}(J) = 0 \\
\#_{m,Q}(A \& B) &= \text{ mim}(\#_{m,Q}(A), \#_{m,Q}(B)) \\
\#_{m,Q}(A \& B) &= \text{ mim}(\#_{m,Q}(A), \#_{m,Q}(B)) \\
\#_{m,Q}(A \oplus B) &= \text{ mim}(\#_{m,Q}(A), \#_{m,Q}(B)) \\
\#_{m,Q}(A \ominus B) &= \text{ mim}(\#_{m,Q}(A), \#_{m,Q}(B)) \\
\#_{m,Q}(\land xA) &= \#_{m,Q}(\lor xA) = \#_{m,Q}(A) \\
\#_{m,Q}(\boxdot A) &= \#_{m,Q}(\bowtie A) = \#_{m,Q}(A) \\
\#_{m,p}(\top^- A) &= \#_{m,p}(A) \\
\#_{m,p}(\bot^- A) &= \#_{m,p}(\bot^- A) \\
\#_{m,p}(\top^- A) &= \#_{m,p}(\bot^- A) = \#_{m,p}(A) \\
\#_{m,p}(\bot^- A) &= \#_{m,p}(\bot^- A) = \#_{m,p}(A) + 1 \\
\#_{\min,Q}(\top\uparrow A) &= Y(\#_{\min,Q}(A)) \\
\#_{\max,p}(\top\uparrow A) &= X(\#_{\max,p}(A)) \\
\#_{\max,Q}(\top\uparrow A) &= \top \\
\#_{\min,Q}(\top\uparrow A) &= \#_{m,Q}(A) \\
\#_{\max,Q}(\top\uparrow A) &= \#_{m,Q}(A) \\
\#_{\min,Q}(\top\uparrow A) &= \#_{m,Q}(A) \\
\#_{m,Q}(\top\uparrow A) &= \#_{m,Q}(A)
\end{align*}
\]

Figure 13: Count function
Figure 14: Montague sentences

5 Illustration

Morrill and Valentín (2016[43]) defines as the Montague Test the task of providing a computational grammar of the PTQ fragment of Montague (1973[21]), and shows how CatLog fulfils this task. We are not aware of any other system which has passed the Montague Test. The example sentences of Chapter 7 of Dowty et al. (1981[7]) are given in Figure 14; the lexicon is given in Figure 15.

The CatLog3 LTEX output for the (ambiguous) last sentence is as follows:

\((dwp((7-116, 118))) \text{[every+man]+doesn't+walk : } S f\)

\[\forall g(Vf(S f\\top Nt(s(g))\downarrow S f)/CNs(g)) : \lambda AAB\forall\langle\lambda C(A C) \rightarrow (B C)\rangle,\]
\[\square CNs(m) : \text{man}, \forall g Va(S g(1/(\langle\langle Na\rangle S f\rangle/(\langle\langle Na\rangle S b\rangle))))/S g) : \lambda D\rightarrow(\lambda E\lambda F(E F),\]
\[\square(\langle\langle Na\rangle \rightarrow \exists g Nt(s(g)))\rangle/\neg\lambda G(\text{Pres (walk } G)) \Rightarrow S f\]

\[\forall g(Vf(S f\top Nt(s(g))\downarrow S f)/CNs(g)) : \lambda AAB\forall C(A C) \rightarrow (B C),\]
\[\square CNs(m) : \text{man}, \forall g Va(S g((\langle\langle Na\rangle S f\rangle/(\langle\langle Na\rangle S b\rangle))))/S g) : \lambda D\rightarrow(D \lambda E\lambda F(E F),\]
\[\square(\langle\langle Na\rangle S b\rangle) : \lambda G(\text{walk } G) \Rightarrow S f\]
a: \[ Vg(\lambda f(S^f \cdot \lambda Nt(s(g)))\cdot S_f)\cdot CNs(g) \] : AAB3C[(\lambda C) \wedge (B C)]

and: \[ Vg(\lambda f(S^f \cdot \lambda Nt(s(g)))\cdot S_f) \] : (\Phi^+ 0 or)

and: \[ Vg(\lambda f(\lambda Nt(s(g)))\cdot S_f) \] : (\Phi^+ (s 0) or)

believes: \[ \lambda AAB(\lambda \lambda IB(\lambda Pres ((\lambda believe A) B)) \)

bill: \[ \lambda Nt(s(m)) : b \]
catch: \[ \lambda AAB(\lambda \lambda IB(\lambda Pres((\lambda catch A) B)) \)
doesnt: \[ Vg(\lambda f(\lambda Nt(s(g)))\cdot S_f)\cdot CNs(g) \] : AAB(C[(\lambda A) \wedge (B C)]

eat: \[ \lambda AAB(\lambda IB(\lambda Pres((\lambda eat A) B)) \)

every: \[ Vg(\lambda f(\lambda Nt(s(g)))\cdot S_f)\cdot CNs(g) \] : AAB[C[(\lambda A) \wedge (B C)]

finds: \[ \lambda AAB(\lambda IB(\lambda Pres((\lambda find A) B)) \)

fish: \[ \lambda CNs(n) : \lambda fish \]

he: \[ \lambda Pres((\lambda Nt(s(m)))\cdot S_f) \] : AAA

her: \[ Vg(\lambda f(\lambda Nt(s(g)))\cdot S_f)\cdot \lambda Pres((\lambda Nt(s(m)))\cdot S_f) \] : AAA

in: \[ \lambda AAB(\lambda IB(\lambda Pres((\lambda in A) B)) \)

is: \[ Vg(\lambda f(\lambda Nt(s(g)))\cdot S_f)\cdot \lambda Pres((\lambda in A) B)) \]

\[ \lambda AAB(\lambda IB(\lambda Pres((\lambda in A) B)) \]

it: \[ \lambda Pres((\lambda Nt(s(m)))\cdot S_f)\cdot \lambda Pres((\lambda in A) B)) \]

\[ \lambda AAB(\lambda IB(\lambda Pres((\lambda in A) B)) \]

john: \[ \lambda Nt(s(m)) : j \]

loses: \[ \lambda AAB(\lambda IB(\lambda Pres((\lambda lose A) B)) \]

loves: \[ \lambda AAB(\lambda IB(\lambda Pres((\lambda love A) B)) \]

man: \[ \lambda CNs(n) : \lambda man \]

necessarily: \[ \lambda (S\lambda A)\cdot (\lambda S\lambda A) : \lambda Nec \]

or: \[ \lambda Vg(\lambda f(\lambda Nt(s(g)))\cdot S_f) \] : (\Phi^+ 0 or)

or: \[ \lambda Vg(\lambda f(\lambda Nt(s(g)))\cdot S_f)\cdot (\lambda Nt(s(m))) \] : (\Phi^+ (s 0) or)

or: \[ \lambda Vg(\lambda f(\lambda Nt(s(g)))\cdot S_f)\cdot (\lambda Nt(s(m))) \] : (\Phi^+ (s 0) or)

park: \[ \lambda CNs(n) : \lambda park \]

seeks: \[ \lambda AAB(\lambda IB(\lambda Pres((\lambda find A) B)) \]

she: \[ \lambda Vg(\lambda f(\lambda Nt(s(g)))\cdot S_f)\cdot (\lambda Nt(s(m))) : \lambda AAA \]

slowly: \[ \lambda AAB(\lambda IB(\lambda Pres((\lambda find A) B)) \]

such that: \[ \lambda \lambda AAB(\lambda IB((\lambda B) C) \cdot (\lambda C)) \]

talks: \[ \lambda AAB(\lambda Pres((\lambda talk A)) \]

that: \[ \lambda (\lambda CPthat(\lambda S\lambda f) : \lambda AAA \]

the: \[ \lambda Vg(\lambda f(\lambda Nt(s(g)))\cdot S_f)\cdot \lambda Pres((\lambda talk A)) \]

to: \[ \lambda Vn(\lambda Nt(s(g)))\cdot \lambda Nt(s(m))) : \lambda AAA \]

tries: \[ \lambda Vg(\lambda f(\lambda Nt(s(g)))\cdot S_f)\cdot (\lambda Nt(s(m))) : \lambda AAA \]

unicorn: \[ \lambda CNs(n) : \lambda unicorn \]

walk: \[ \lambda AAB(\lambda IB(\lambda Pres((\lambda walk A)) \]

walks: \[ \lambda AAB(\lambda Pres((\lambda walk A)) \]

woman: \[ \lambda CNs(f) : \lambda woman \]

Figure 15: Montague lexicon
∀C[‘man C] → ¬∀G[‘walk G]


