CatLog: A Categorial Parser/Theorem-Prover*

Glyn Morrill

Universitat Politècnica de Catalunya
morrill@lsi.upc.edu.edu
http://www.lsi.upc.edu/~morrill

Abstract. We present CatLog, a parser/theorem-prover for logical categorial grammar. The logical fragment implemented is a displacement logic the multiplicative basis of which is the displacement calculus of Morrill, Valentín & Fadda (2011)[8].

(Logical) categorial grammar (Morrill 1994[9], 2011[10]; Moortgat 1997[6]; Carpenter 1998[1]; Jäger 2005[4]) originated with Lambek's (1958[5]) insight that a calculus of grammatical types (constituting a residuated monoid) can be formulated using Gentzen's method. The result is an algebraic rendering of grammar as logic and parsing as deduction. Although the design is, really, architecturally perfect and, by now, well-understood, linguistically it is strictly limited to continuity by the fact that it deals with a residuated family with parent (the canonical extension of) concatenation: after all, the whole challenge of modern linguistics for 50 years has been the ubiquity in natural grammar of *dis*continuity. In this relation Morrill, Valentín & Fadda (2011)[8] provides for discontinuity the displacement calculus **D**, deductively a conservative extension of the Lambek calculus **L** with residuated families with respect to both concatenation and intercalation. Like **L**, **D** is free of structural rules and enjoys Cut-elimination and its corollaries the subformula property, decidability, and the finite reading property.

CatLog is a categorial parser/theorem prover implementing a categorial logic extending **D**. It employs Cut-free backward chaining sequent theorem-proving. For **L** deductive spurious ambiguity can be removed by normalization (Hendriks 1993[3]). Because **D** is based on the same design principles, the same techniques can be adopted (Morrill 2011[7]) and CatLog depends on this. In addition to normalization CatLog uses sequent search space pruning by the count invariance of van Benthem (1991[11]). The type-constructors of the displacement logic of CatLog are shown in Fig. 1.

Version f1.2 of CatLog is provisional in a number of respects. In particular, not all spurious ambiguity is eliminated for the categorial logic fragment, and non-duplication of results is achieved by filtering according to a brute force duplication check. Furthermore, bracketing structure must be specified in the input, rather than be induced. And the count-invariance check for multiplicatives

* This research was partially supported by BASMATI MICINN project (TIN2011-27479-C04-03) and by SGR2009-1428 (LARCA).

$I, \backslash, \cdot, /$	Lambek connectives
$J, \{\downarrow_k, \odot_k, \uparrow_k\}_{k\in\{>,<\}}$	displacement connectives
$\otimes, \bar{}$	nondeterministic continuous connectives
$\Downarrow, \circledcirc, \Uparrow$	nondeterministic discontinuous connectives
$\{\hat{}^k, \check{}^k\}_{k\in\{>,<\}}$	bridge and split
$\hat{}, \check{}$	nondeterministic bridge and split
$\triangleleft^{-1}, \triangleleft, \triangleright, \triangleright^{-1}$	left and right projection and injection
$\&, +$	semantically active additives
$\sqcap, \sqcup$	semantically inactive additives
$\forall, \exists$	first-order quantifiers
$¡, ^+$	structural modalities
$[\,]^{-1}, \langle\rangle$	bracket modalities
$\Box, \blacksquare$	normal modalities
$\mid$	limited contraction for anaphora

Fig. 1. Type-constructors of CatLog

is not adapted to additives and structural modalities. These issues remain topics for future improvement. Nevertheless CatLog f1.2 already provides fast and wide-coverage Montague-like parsing.

The program comprises 3000 lines of Prolog implementing some 80 inference rules for the categorial logic fragment, LaTeX outputting, lexicon, and sample sentences. Among the examples four blocks are distininguished: Dutch examples (cross-serial dependencies), relativization including islands and parasitic gaps, the Montague example sentences of Dowty, Wall and Peters (1981)[2] Chapter 7, and the example sentences of Morrill, Valentín and Fadda (2011)[8].

The functionality is as follows. Once CatLog has been loaded into Prolog, the query `?- pplex.` will cause the lexicon to be pretty printed in the console window, the Dutch part of which is as shown in Fig. 2. The query `?- pplexlatex.` has no visible effect but will cause the lexicon to be output in LaTeX to a file named "s.tex". Querying $t(N)$ will test the examples unifying with term N. For example `?- t(rel(6)).` tests the relativization example 6, `?- t(rel(_)).` tests all the relativization examples, and `?- t(_).` tests all the examples. The analyses — the examples, the derivational proofs, and the semantic readings — appear in the Prolog window, and this information but without duplicate equivalent analyses is written in LaTeX to a file named "t.tex". LaTeXing the file "out.tex" will include s.tex and t.tex and format the lexicon and last analyses made. For example, `?- t(d(2)).` produces the contents in Fig. 3 in Prolog. The LaTeX output for the Dutch part of the lexicon and the same example is as shown in Figs. 4 and 5.

References

1. Bob Carpenter. *Type-Logical Semantics.* MIT Press, Cambridge, MA, 1997.
2. David R. Dowty, Robert E. Wall, and Stanley Peters. *Introduction to Montague Semantics*, volume 11 of *Synthese Language Library.* D. Reidel, Dordrecht, 1981.

```
wil: (NA\Si)in(NA\Sf): LBLC((want (B C)) C)
wil: Q/^(Sfex((NA\Si)in(NA\Sf))): LB(B LCLD((want (C D)) D))
alles: (SAexNt(s(n)))inSA: LBAC[(thing C) -> (B C)]
boeken: Np(n): books
cecilia: Nt(s(f)): c
de: Nt(s(A))/CNA: the
helpen: |>-1((NA\Si)in(NB\(NA\Si))): LCLD((help (C D)) D)
henk: Nt(s(m)): h
jan: Nt(s(m)): j
kan: (NA\Si)in(NA\Sf): LBLC((isable (B C)) C)
kunnen: |>-1((NA\Si)in(NA\Si)): LBLC((isable (B C)) C)
las: NA\(Nt(s(B))\Sf): read
lezen: |>-1(NA\(NB\Si)): read
nijlpaarden: CNp(n): hippos
voeren: |>-1(NA\(NB\Si)): feed
zag: (Nt(s(A))\Si)in(NB\(Nt(s(A))\Sf)): LCLD((saw (C D)) D)
```

Fig. 2. Dutch part of lexicon

```
(d(2)) jan+boeken+kan+lezen S_647


Nt(s(m)): j, Np(n): books, (NA\Si)in(NA\Sf): LBLC((isable (B C)) C), |>-1(ND\(NE\Si)): read => SF


Nt(s(m)), Np(n), (Nt(s(m))\Si)in(Nt(s(m))\Sf), |>-1(Np(n)\(Nt(s(m))\Si)) => Sf [inL]
    Np(n), 1, |>-1(Np(n)\(Nt(s(m))\Si)) => Nt(s(m))\Si [\R]
        Nt(s(m)), Np(n), 1, |>-1(Np(n)\(Nt(s(m))\Si)) => Si [|>-1L]
            Nt(s(m)), Np(n), Np(n)\(Nt(s(m))\Si){1} => Si [\L]
                Np(n) => Np(n)
                Nt(s(m)), Nt(s(m))\Si{1} => Si [\L]
                    Nt(s(m)) => Nt(s(m))
                    Si{1} => Si
    Nt(s(m)), Nt(s(m))\Sf => Sf [\L]
        Nt(s(m)) => Nt(s(m))
        Sf => Sf

((isable ((read books) j)) j)
```

Fig. 3. Dutch verb raising

wil : $(NA\backslash Si){\downarrow}(NA\backslash Sf) : \lambda B\lambda C((want\ (B\ C))\ C)$
wil : $Q/\hat{}(Sf{\uparrow}((NA\backslash Si){\downarrow}(NA\backslash Sf))) : \lambda B(B\ \lambda C\lambda D((want\ (C\ D))\ D))$
alles : $(SA{\uparrow}Nt(s(n))){\downarrow}SA : \lambda B\forall C[(thing\ C) \rightarrow (B\ C)]$
boeken : $Np(n) : books$
cecilia : $Nt(s(f)) : c$
de : $Nt(s(A))/CNA : the$
helpen : $\triangleright^{-1}((NA\backslash Si){\downarrow}(NB\backslash(NA\backslash Si))) : \lambda C\lambda D((help\ (C\ D))\ D)$
henk : $Nt(s(m)) : h$
jan : $Nt(s(m)) : j$
kan : $(NA\backslash Si){\downarrow}(NA\backslash Sf) : \lambda B\lambda C((isable\ (B\ C))\ C)$
kunnen : $\triangleright^{-1}((NA\backslash Si){\downarrow}(NA\backslash Si)) : \lambda B\lambda C((isable\ (B\ C))\ C)$
las : $NA\backslash(Nt(s(B))\backslash Sf) : read$
lezen : $\triangleright^{-1}(NA\backslash(NB\backslash Si)) : read$
nijlpaarden : $CNp(n) : hippos$
voeren : $\triangleright^{-1}(NA\backslash(NB\backslash Si)) : feed$
zag : $(Nt(s(A))\backslash Si){\downarrow}(NB\backslash(Nt(s(A))\backslash Sf)) : \lambda C\lambda D((saw\ (C\ D))\ D)$

Fig. 4. Dutch part of lexicon

(d(2)) **jan+boeken+kan+lezen** : $S_6 47$

$Nt(s(m)) : j, Np(n) : books, (NA\backslash Si){\downarrow}(NA\backslash Sf) : \lambda B\lambda C((isable\ (B\ C))\ C), \rhd^{-1}(ND\backslash(NE\backslash Si)) : read\ \Rightarrow\ SF$

$$\frac{\overline{Np(n)\ \Rightarrow\ Np(n)} \qquad \dfrac{\overline{Nt(s(m))\ \Rightarrow\ Nt(s(m))} \quad \overline{Si\{1\}\ \Rightarrow\ Si}}{Nt(s(m)), \boxed{Nt(s(m))\backslash Si\{1\}}\ \Rightarrow\ Si}\backslash L}{Nt(s(m)), Np(n), \boxed{Np(n)\backslash(Nt(s(m))\backslash Si)\{1\}}\ \Rightarrow\ Si}\backslash L$$

$$\frac{Nt(s(m)), Np(n), \boxed{Np(n)\backslash(Nt(s(m))\backslash Si)\{1\}}\ \Rightarrow\ Si}{Nt(s(m)), Np(n), 1, \boxed{\rhd^{-1}(Np(n)\backslash(Nt(s(m))\backslash Si))}\ \Rightarrow\ Si}\rhd^{-1}L$$

$$\frac{Nt(s(m)), Np(n), 1, \boxed{\rhd^{-1}(Np(n)\backslash(Nt(s(m))\backslash Si))}\ \Rightarrow\ Si}{Np(n), 1, \rhd^{-1}(Np(n)\backslash(Nt(s(m))\backslash Si))\ \Rightarrow\ Nt(s(m))\backslash Si}\backslash R$$

$$\frac{\overline{Nt(s(m))\ \Rightarrow\ Nt(s(m))} \quad \overline{Sf\ \Rightarrow\ Sf}}{Nt(s(m)), \boxed{Nt(s(m))\backslash Sf}\ \Rightarrow\ Sf}\backslash L$$

$$\frac{Np(n), 1, \rhd^{-1}(Np(n)\backslash(Nt(s(m))\backslash Si))\ \Rightarrow\ Nt(s(m))\backslash Si \qquad Nt(s(m)), \boxed{Nt(s(m))\backslash Sf}\ \Rightarrow\ Sf}{Nt(s(m)), Np(n), \boxed{(Nt(s(m))\backslash Si){\downarrow}(Nt(s(m))\backslash Sf)}, \rhd^{-1}(Np(n)\backslash(Nt(s(m))\backslash Si))\ \Rightarrow\ Sf}{\downarrow}L$$

$((isable\ ((read\ books)\ j))\ j)$

Fig. 5. Dutch verb raising

3. H. Hendriks. *Studied flexibility. Categories and types in syntax and semantics.* PhD thesis, Universiteit van Amsterdam, ILLC, Amsterdam, 1993.
4. Gerhard Jäger. *Anaphora and Type Logical Grammar*, volume 24 of *Trends in Logic – Studia Logica Library*. Springer, Dordrecht, 2005.
5. Joachim Lambek. The mathematics of sentence structure. *American Mathematical Monthly*, 65:154–170, 1958. Reprinted in Buszkowski, Wojciech, Wojciech Marciszewski, and Johan van Benthem, editors, 1988, *Categorial Grammar*, Linguistic & Literary Studies in Eastern Europe volume 25, John Benjamins, Amsterdam, 153–172.
6. Michael Moortgat. Categorial Type Logics. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*, pages 93–177. Elsevier Science B.V. and the MIT Press, Amsterdam and Cambridge, Massachusetts, 1997.
7. Glyn Morrill. Logic Programming of the Displacement Calculus. In Sylvain Pogodalla and Jean-Philippe Prost, editors, *Proceedings of Logical Aspects of Computational Linguistics 2011, LACL'11, Montpellier*, number LNAI 6736 in Springer Lecture Notes in AI, pages 175–189, Berlin, 2011. Springer.
8. Glyn Morrill, Oriol Valentín, and Mario Fadda. The Displacement Calculus. *Journal of Logic, Language and Information*, 20(1):1–48, 2011. Doi 10.1007/s10849-010-9129-2.
9. Glyn V. Morrill. *Type Logical Grammar: Categorial Logic of Signs.* Kluwer Academic Publishers, Dordrecht, 1994.
10. Glyn V. Morrill. *Categorial Grammar: Logical Syntax, Semantics, and Processing.* Oxford University Press, 2011.
11. J. van Benthem. *Language in Action: Categories, Lambdas, and Dynamic Logic.* Number 130 in Studies in Logic and the Foundations of Mathematics. North-Holland, Amsterdam, 1991. Revised student edition printed in 1995 by the MIT Press.