On learning typed-unification grammars

Liviu Ciortuz

CS Department, University of York
Heslington, York, YO10 5DD, UK, E-mail: ciortuz@cs.york.ac.uk

Interest in typed unification grammars in Natural Language Processing can be traced back to the seminal work on the PATR-II system [13]. Since then, the *logics* of feature constraints were studied and became well-understood [15] [3], and different types of unification-based *grammar formalisms* were developed by the Computational Linguistics community — most notably Lexical Function Grammars [6], Head-driven Phrase Structure Grammars (HPSG, [11]) and Categorial (unification) Grammars [17]. During the last years *large-scale implementations* for such grammars were devised.[1] More recently, important advances in the elaboration of techniques for efficient unification and parsing with such grammars [16]. Up to our knowledge, the possibility of automate learning/development of such grammars is still (largely) unexplored.

We elaborated two procedures to be used in automatic learning of typed-unification grammars, namely *specialisation* and respectively *generalisation* of typed feature structure (FS) definitions. Their *aim* is to make an input set of sentences get a better coverage w.r.t. the given grammar.

In specialising a grammar, new atomic constraints are added to a certain feature path value (i.e. substructure in a typed FS), in such a way that sentences (or parses) which were wrongly accepted in the form of the grammar be finally rejected (by the learnt form of the grammar). Reversely, in generalising a grammar, parses (or more generally, sentences) which are (wrongly) rejected by the input the grammar are forwarded to the learning system, which makes their (partial) analysis provide clues about which constraints in the type FS definitions have to be eliminated or softened in order to make that parse become acceptable by the (new form of the) grammar. The two — specialisation and generalisation — procedures combine naturally into a unified learning strategy.

The *approach* we followed in developing the two procedures for learning typed-unification grammars is that proposed by Inductive Logic Programming (ILP) [9]. However, instead of applying it to a first-order translation of the grammar to be learnt, we adapted the ILP schema to the feature constraint (subset of the first-order) logic specific to such grammars [1] [3] [2].

As *implementation*, our prototype system for learning typed-unification grammars was developed by extending the LIGHT compiler system [4] with an ILP-like module. Experiments were successfully conducted on a grammar adapted from [12], which embodies the main principles of the HPSG theory.

Currently, in order to *scale-up* the range of application for the learning procedures we developed, we study a restricted form of generalisation of typed FSs (we call it F-generalisation) on LinGO, large scale HPSG grammar for English developed at CSLI, University of Stanford. The *aim* now is (not to provide a better coverage but) to maximally reduce the size of the grammar, while maintaining its coverage. To off-line F-generalisation of the typed FSs in the grammar will correspond on-line specialisation of the parsing (possibly partial) results.

[1] For instance the HPSG for English developed at Stanford [5] called LinGO, two HPSGs for Japanese developed at Tokyo University [8], and respectively at DFKI–Saarbruecken, Germany [14], and the HPSG for German, developed also at DFKI [10]. Large-scale LFG grammars were developed by Xerox Corp., but they are not publicly available.

The *motivation* behind this work on F-generalisation: measurements done with LinGO on the CSLI test suite revealed that more than 50% of the feature constraints in the (expanded) grammar can be eliminated without affecting the coverage.[2] As a consequence, the unification time — the most consuming task during parsing with such grammars — will be significantly reduced, provided that the number of parse items (i.e. partial parses) does not inflate.

As a consequence, the *future tasks* will be to design more efficient algorithms for F-generalisation of LinGO-like grammars, and to examine/find techniques for keeping the number of partial parses close to that in the original grammar. The link with stochastic learning and (shallow) parsing techniques will be explored.

References

1. H. Aït-Kaci and A. Podelski. Towards a meaning of LIFE. *Journal of Logic Programming*, 16:195–234, 1993.
2. H. Aït-Kaci, A. Podelski, and S.C. Goldstein. Order-sorted feature theory unification. *Journal of Logic, Language and Information*, 30:99–124, 1997.
3. B. Carpenter. *The Logic of Typed Feature Structures – with applications to unification grammars, logic programs and constraint resolution.* Cambridge University Press, 1992.
4. L.-V. Ciortuz. Compiling HPSG into C. Unpublished technical report, The German Research Center for Artificial Intelligence (DFKI), Saarbruecken, Germany, 2001.
5. A. Copestake, D. Flickinger, and I. Sag. *A Grammar of English in HPSG: Design and Implementations.* Stanford: CSLI Publications, 1999.
6. R. M. Kaplan and J. Bresnan. Lexical-functional grammar: A formal system for grammatical representation. In J. Bresnan, editor, *The Mental Representation of Grammatical Relations*, pages 173–381. The MIT Press, 1982.
7. R. Malouf, J. Carroll, and A. Copestake. Efficient feature structure operations without compilation. *Journal of Natural Language Engineering*, 6 (1) (Special Issue on Efficient Processing with HPSG):29–46, 2000.
8. Y. Mitsuishi, K. Torisawa, and J. Tsujii. HPSG-Style Underspecified Japanese Grammar with Wide Coverage. In *Proceedings of the 17th International Conference on Computational Linguistics: COLING-98*, pages 867–880, 1998.
9. S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19,20:629–679, 1994.
10. Stefan Müller. *Deutsche Syntax deklarativ. Head-Driven Phrase Structure Grammar für das Deutsche.* Number 394 in Linguistische Arbeiten. Max Niemeyer Verlag, Tübingen, 1999.
11. C. Pollard and I. Sag. *Head-driven Phrase Structure Grammar.* Center for the Study of Language and Information, Stanford, 1994.
12. S. Shieber. *An introduction to unification-based approaches to grammars.* CSLI Publications, Stanford, 1986.
13. S. M. Shieber, H. Uszkoreit, F. C. Pereira, J. Robinson, and M. Tyson. The formalism and implementation of PATR-II. In J. Bresnan, editor, *Research on Interactive Acquisition and Use of Knowledge.* SRI International, Menlo Park, Calif., 1983.
14. M. Siegel. HPSG analysis of japanese. In *Verbmobil: Foundations of Speech-to-Speech Translation.* Springer Verlag, 2000.
15. G. Smolka. Feature-constraint logics for unification grammars. *Journal of Logic Programming*, 12:51–87, 1992.
16. K. Torisawa, K. Nishida, Y. Miyao, and J. Tsujii. An HPSG parser with CFG filtering. *Journal of Natural Language Engineering*, 6 (1) (Special Issue on Efficient Processing with HPSG), 2000.
17. H. Uszkoreit. Categorial Unification Grammar. In *International Conference on Computational Linguistics (COLING'92)*, pages 498–504, Nancy, France, 1986.

This article was processed using the LaTeX macro package with LLNCS style

[2] This is due to the applicability of the so-called Quick-Check pre-unification filter [7].