Compilation of quasi-destructive unification of type feature structures

Liviu Ciortuz

Computer Science Faculty, University of Iasi, Romania.
E-mail: ciortuz@infoiasi.ro

Recent work on large-scale unification grammars for natural language processing [7][11] has shown that Tomabechi's quasi-destructive unification algorithm with feature structure (FS) sharing [12] is the most appropriate for such kind of tasks. This conclusion was reached by careful engineering of (parsing as deduction) interpreters [2][6]. Compilation attempts for FS unification exist [5] [10] [13] [3] [1], but up to our knowledge none of them performs non-destructive unification, and only the first two mentioned systems have been scaled up so to cope with large grammars like LinGO [8], the large HPSG grammar for English.

Our current work on the LIGHT system concerns its extension so to achieve compilation of destructive unification of FSs à la Tomabechi. Prior to the extension we report here, the Light compiler system used a (destructive) FS sharing schema similar to [9], which maximally (re)uses a FS at the expense of (repeated) storing and restoring modifications done by unifications with different FSs. Following this (less elaborated) approach, the LIGHT system already has already achieved a speed performance comparable to PET [2], known as being the fastest system running the LinGO grammar. It is expected that in the quasi-destructive approach, the performance of the system will be significantly better.

Basically, Tomabechi's idea for feature structure sharing is: let FSs use common (maximal) substructures as long as the nodes (including all descendants) in these substructures are not modified. (In this way, the repeated storing and restoring following successful unification on a FS is eliminated.) The implementation of this idea has two stages. First, during unification, modified nodes in the two (rooted) FSs are marked so that in the second stage one could identify the substructures to be subsequently shared. During that second stage, the remaining part of the FSs, i.e. the ancestor nodes of the shared substructures are copied. With respect to approaches which fully copy FSs before unification, this approach substantially reduces the amount of copying needed while generating different new edges in the parsing process.

The LIGHT system's design extension so to perform quasi-destructive unification is not affecting the compiler's abstract machine (LIGHT AM), which is the core of the system [4], in charge with compiled unification of FSs. Instead, quasi-destructive unification is achieved in LIGHT by modifying the control level above unification, namely the feature structure sharing mechanism. This control level was implemented in our system as a virtual machine, called LIGHT VM. It constitutes the functional interface between the LIGHT AM and the parser module, which asks for certain unifications and uses their results.

The main gains/advantages of compilation of quasi-destructive unification of FSs are:

1. In the compilation approach only one of the two FSs to be unified is transformed during unification; the other is under compiled form, as a function.
2. Compilation reduces the time spent on computing the feature substructures to be copied, since *a.* the ancestors of the nodes to be copied during unification can be computed at compilation time, and *b.* only non-visited features and nodes explored by on-line unification (when calling the abstract instruction `unify_feature`) have to be explored as in the first stage of Tomabechi's algorithm. These nodes and features represent only a (reduced) fraction of the whole FS.

Moreover, the fact that the control of the unification mechanism in the LIGHT system is devised as a virtual machine will nicely allow one to dynamically opt for either destructive or quasi-destructive FS sharing, depending on the amount (and depth) of changes done during unification and the number of times the (affected) FS is reused.

We are currently in the process of testing this new important extension/modification of the LIGHT system on the whole LinGO grammar for English, and we will soon provide measurements on the parsing speed gain.

Acknowledgements: The LIGHT system was designed and implemented while the author was employed at the Language Technology Lab of the German Research Center for Artificial Intelligence (DFKI) in Saarbrücken, Germany. Later, the system was extended and documented while the author was supported by an EPSRC grant at the University of York, UK.

References

1. H. Aït-Kaci and R. Di Cosmo. Compiling order-sorted feature term unification. Technical report, Digital Paris Research Laboratory, 1993.
2. U. Callmeier. PET — a platform for experimentation with efficient HPSG processing techniques. *Journal of Natural Language Engineering*, 6 (1) (Special Issue on Efficient Processing with HPSG):99–108, 2000.
3. B. Carpenter and Y. Qu. An abstract machine for attribute-value logic. In *Proceedings of the Fourth International Workshop on Parsing Technologies*, pages 59–70, 1995.
4. L. Ciortuz. LIGHT — a constraint language and compiler system for typed-unification grammars. In *KI-2002: Advances in Artificial Intelligence*, volume 2479, pages 3–17, Berlin, Germany, 2002. Springer-Verlag. Proceedings of the 25th Annual German Conference on AI, Aachen, Germany.
5. L. Ciortuz. LIGHT – another abstract machine for feature structure unification. In S. Oepen, D. Flickinger, J. Tsujii, and H. Uszkoreit, editors, *Efficiency in Unification-based Processing*, pages 167–194. CSLI Publications, The Center for the Studiy of Language and Information, Stanford University, 2002.
6. A. Copestake. *The (new) LKB system.* CSLI, Stanford University, 1999.
7. D. Flickinger, S. Oepen, J. Tsujii, and H. Uszkoreit (eds.). Special issue on efficient processing with HPSG, 2000. Journal of Natural Language Engineering, 6 (1).
8. Daniel P. Flickinger, Ann Copestake, and Ivan A. Sag. HPSG analysis of English. In Wolfgang Wahlster, editor, *Verbmobil: Foundations of Speech-to-Speech Translation*, Artificial Intelligence, pages 254–263. Springer-Verlag, 2000.
9. L. Karttunen. D-PATR: a development environment for unification-based grammars. In *Proceedings of the International Conference on Computational Linguistics (COLING)*. 1986.
10. Y. Miyao, T. Makino, K. Torisawa, and J. Tsujii. The LiLFeS abstract machine and its evaluation with the LinGO grammar. *Journal of Natural Language Engineering*, 6 (1) (Special Issue on Efficient Processing with HPSG):47–61, 2000.
11. S. Oepen, D. Flickinger, H. Uszkoreit, and J. Tsujii (eds.), editors. *Collaborative Language Engineering*. CSLI Publications, Stanford, CA, 2002.
12. H. Tomabechi. Quasi-destructive graph unification with structure-sharing. In *Proceedings of COLING-92*, pages 440–446, Nantes, France, 1992.
13. S. Wintner and N. Francez. Efficient implementation of unification-based grammars. *Journal of Language and Computation*, 1(1):53–92, 1999.