CHART PARSING AS CONSTRAINT PROPAGATION

Frank Morawietz
Universität Tübingen,
Wilhelmstr. 113,
72074 Tübingen
frank@sfs.nphil.uni-tuebingen.de

The parsing-as-deduction approach proposed in Pereira and Warren [6] and extended in Shieber et al. [7] and the parsing schemata defined as special deduction systems in Sikkel [8] are well established parsing paradigms. They allow for a uniform presentation and comparison of a variety of parsing algorithms. Furthermore, they are extensible to more complex formalisms, e.g., augmented phrase structure rules or the ID/LP format. Constraint Programming has been used in computational linguistics in several areas, for example in (typed) feature-based systems based on Oz [9], or conditional constraints [5], or advanced compilation techniques [3] or specialized constraint solvers [4]. But to my knowledge, none of these approaches uses constraint programming techniques to implement standard chart parsing algorithms directly in a constraint system.

The core idea of the proposal is that the items of a conventional chart parser are constraints on labeled links between the words and positions of an input string. The inference rules allow for the deduction of new constraints, again labeled and spanning parts of the input string, via constraint propagation. The resulting constraint store represents the chart which can be accessed to determine whether the parse was successful or to reconstruct a parse tree. While this may seem a trivial observation it allows for a rapid and flexible method of implementation. The goal is not necessarily to build the fastest parser, but rather to build – for an arbitrary algorithm – a parser fast and perspicuously. The advantage of our approach compared to the one proposed in Shieber et al. [7] is that we do not have to design a special deduction engine, we do not have to handle chart and agenda explicitly. and that it can be seamlessly integrated into existing applications (e.g., within the Constraint Handling Rules (CHR) framework [2]).

Assuming familiarity with parsing-as-deduction, I will presuppose the definitions as given in Shieber et al. [7]. Their algorithms are implemented by specifying the inference rules as constraint propagation rules, the axioms and the goal items as constraints. As an example, Earley's algorithm is presented in Tab. 1. We cannot go into the details of the definitions, but it is easily observable that the given constraint propagation rules are nothing but literal realizations of the inference rules. And the only other ingredient we need for the specification of a parser is a predicate which traverses the input and posts the corresponding initial edge constraints. This predicate is universal to all algorithms, but can be varied with respect to the order the constraints are posted, thereby achieving for example a left-to-right or right-to-left traversal of the string.

So, the similarity between parsing-as-deduction and constraint propagation is used to propose a flexible and simple system which is easy to implement and offers itself as a testbed for different

Table 1: Parsing systems as CHR programs: Earley's algorithm

Items	**Axiom**	**Goal**
`edge(A,Alpha,Beta,I,J)`	`edge(sprime,[],[s],0,0)`	`edge(sprime,[s],[],0,Len)`

Scan
```
edge(A,Alpha,[Cat|Beta],I,J), word(J,Cat-_Word) ==>
                J1 is J+1,
                edge(A,[Cat|Alpha],Beta,I,J1).
```

Predict
```
edge(_A,_Alpha,[B|_Beta],_I,J) ==>
                findall(Gamma, rule(Gamma,B), List) |
                post_edges(List,B,J).
```

Complete
```
edge(A,Alpha,[B|Beta],I,K), edge(B,Gamma,[],K,J) ==>
                edge(A,[B|Alpha],Beta,I,J).
```

parsing strategies with varying traversals of the input and even for different types of grammars (such as for example minimalist grammars). Maybe the most interesting extension is that one can use each created edge to post other constraints, for example about the well-formedness of associated typed feature structures. The posted constraints automatically become available for other constraint solvers. In particular, systems implementing HPSG seem to suffer from the problem how to drive the constraint resolution process efficiently. Some systems, as for example ALE [1], use a phrase structure backbone to drive the process. The proposal here would allow to use the ID/LP schemata directly as constraints, but nevertheless as the driving force behind the other constraint satisfaction techniques on the well-formedness of feature structures.

References

[1] B. Carpenter and G. Penn. ALE: The attribute logic engine. Carnegie Mellon University, 1998.

[2] T. Frühwirth. Theory and practice of constraint handling rules. *JLP*, 37(1–3):95–138, 1998.

[3] T. Götz and D. Meurers. Interleaving universal principles and relational constraints over typed feature logic. In *ACL/EACL Conference '97*, Madrid, Spain, 1997.

[4] S. Manandhar. An attributive logic of set descriptions and set operations. In *ACL Conference '94*, 1994.

[5] J. Matiasek. *Principle-Based Processing of Natural Language Using CLP Techniques.* PhD thesis, TU Wien, 1994.

[6] F. C. N. Pereira and D. H. D. Warren. Parsing as deduction. In *ACL Conference '83*, 1983.

[7] S. M. Shieber, Y. Schabes, and F. C. N. Pereira. Principles and implementation of deductive parsing. *JLP*, 24(1–2):3–36, 1995.

[8] K. Sikkel. *Parsing Schemata.* ETACS series, Springer, 1997.

[9] G. Smolka. The Oz programming model. In J. van Leeuwen, editor, *Computer Science Today*, LNCS 1000, pages 324–343. Springer, 1995.