Structural Correspondence between Finite-State Intersection Grammar and Constraint Satisfaction Problem

Anssi Yli-Jyrä
Department of General Linguistics, University of Helsinki
Anssi.Yli-Jyra@Helsinki.Fi

Extended abstract

The Problem

The aim of this paper is to exactly identify a new tractable subset of Finite-State Intersection Grammar (FSIG) [Koskenniemi, 1990]. To accomplish this goal, I introduce a new approach to FSIG parsing. The effectiveness of my approach is illustrated with an example.

The parsing framework of FSIG consists of two components, an acyclic sentence automaton S and a set of constraint automata $C_1, \ldots, C_n$. The sentence automaton recognizes a set of feature strings that are potential parses for the input sentence. The parsing is carried out by computing the intersection $S \cap C_1 \cap \cdots \cap C_n$.

The (FSIG) Non-Emptiness Problem (NEP) is, given a linguistically motivated but non-fixed set of constraint automata and a sentence automaton, to determine if their intersection accepts a feature string i.e. if there is a parse for the input sentence.

No one has completely explicated how the linguistically motivated NEP differs from an intersection of arbitrary acyclic automata. An arbitrary intersection of acyclic automata, a special case of the intersection of cyclic automata [Garey and Johnson, 1979], is an NP-complete problem. Note that I do not assume that the set of constraints in FSIG is fixed. That restriction would make NEP solvable in polynomial time [Tapanainen, 1997].

An important tractable subset of finite-state intersection problem has been characterized with locally testable sets [McNaughton and Papert, 1971]. Local testability is the ground for feasibility of some consistency enforcing methods [Barton, 1987, Piitulainen, 1995], but it benefits also many other reported FSIG parsing algorithms [Tapanainen, 1997] including the backtracking algorithm and even some algorithms for constructing a deterministic intersection automaton. The backtracking FSIG parsing algorithm can take advantage of the local pruning of search space even when the constraint automata does not meet the criteria

for local testability [Karttunen, 1994, p. 410].

However, there are instances of NEP for which a more general condition of tractability is highly desirable. For example, constraints with unbounded (non-local) contexts are practically necessary in a high-quality syntactic description. As the most FSIG parsing methods rely on the assumption of a locally testable intersection, a wider tractable class of the NEP instances has not been identified with an algorithmic characterization applied to FSIG.

The Approach

The current paper introduces a novel approach for FSIG parsing. This is based on the polynomial equivalence between the acyclic intersection problem and the Constraint Satisfaction Problem (CSP). The approach includes a relational representation for FSIG constraints and an auxiliary data structure called a constraint graph. The function of the new relational representation is to allow piecewise (k-wise) testability [Pin, 1995] of the constraint languages. The constraint graph is used as a tool to select an order in which the piecewise tests can be combined most efficiently.

A Constraint Satisfaction Problem [Mackworth, 1977] can be formulated with a constraint network that consists of a finite set of variables $X = \{X_1, \ldots, X_n\}$, a domain D_i of possible values for each variable $x_i, 1 \leq i \leq n$, and a set of t constraint relations $R_{S_1}, \ldots, R_{S_t}$. The problem is to find an assignment to the variables so that it satisfies all the constraint relations. A constraint relation R_S over a set of variables $S = \{x_1, \ldots, x_r\} \subseteq X$ is a subset of the Cartesian product of their domains i.e. $R_S \subseteq D_1 \times \cdots \times D_r$. A binary constraint network is a special case where all the constraint relations are associated with no more than two variables.

Barton's tableau representation [Barton, 1987] for TWOL constraints is, in fact, a constraint network with trinary relations. His constraint network contains, in addition to a variable for each string position, a set of auxiliary variables each of which indicates a state of a constraint automaton at a specific string position.

In my approach, the transformation of NEP to CSP does not involve auxiliary variables. Assume that the strings accepted by the sentence automaton have the length of n and that each string position corresponds to a variable in the constraint network. I replace each FSIG constraint automata with a huge constraint relation in the constraint network. To carry out this simple transformation, I separately intersect every cyclic constraint automaton with the sentence automaton. The language of each result automaton is a finite n-ary constraint relation over the positions in the potential feature strings.

The simple transformation hides the fact that most constraint automata ignore a big proportion of the features (and positions) in the strings. Such constraints are inefficiently coded as n-ary relations. I have managed, however, to decompose some constraint relations to a set of small-arity ones using the existing algorithms [Dechter, 1990, Yli-Jyrä, 1995]. Decompositions to so-

called join trees is highly desirable because this makes it more probable to find tractable orderings (see below) in the whole constraint network [Dechter, 1992]. Nevertheless, such decompositions are not generally possible.

When some FSIG constraints are informally studied in the non-compiled form [Voutilainen, 1997], one may be mislead to think that some simple constraints involve piecewise tests. For example, the constraint `nowhere(VFIN .. VFIN)`, which says that no clause contains two finite verbs, seems delusively to correspond to a 2-testable constraint language. An idea behind FSIG is, however, to disambiguate part-of-speech tags, syntactic functions and clause boundaries in parallel [Koskenniemi, 1990]. So, the expression *dotdot* '..' must match anything within (but only within) the same clause.

Unfortunately, every combination of potential clause boundaries between the two finite verbs must be taken into account, and, therefore, this constraint is hardly 2-wise testable. Tapanainen [Tapanainen, 1997] argues, however, that some carefully selected constraints should be processed before other constraints, so that the possible clause boundaries become less ambiguous. These complications make a successful decomposition of a constraint relation dependent of the other constraints in the constraint network.

Since the methods and tractability results concerning the binary constraint networks generalize to the non-binary ones [Dechter, 1992], assume now, for clarity, that all the constraint relations are binary.

A binary constraint network can be associated with a constraint graph in which there is a node for each variable of the constraint network, and a pair of such nodes is connected if the corresponding variables are constrained with a constraint relation in the constraint network. If we manage to order the nodes in the constraint graph in such a way that the so-called induced width of the graph [Dechter, 1992] is not greater than r, then the CSP instance can be solved in $O(exp(r))$ steps.

Although finding an ordering of a constraint graph with the optimal induced width is NP-complete, it is easy to decide if the induced width w of an ordered constraint graph is below an integer [Dechter, 1992]. So-called directional (strong) $(w+1)$-consistency can be established in time that is exponential to w. At such a consistency level, existence of a solution is already decided, because a consistent assignment to the first $w+1$ (ordered) variables can be extended to a complete solution [Dechter, 1992].

If the constraint graph forms a tree, its optimal induced width is 1. In such a case, enforcing directional strong 2-consistency is enough to decide whether a solution to the CSP instance exists [Dechter, 1992].

The Results

I give an example that shows that the new approach to FSIG parsing extends the tractable intersection from the locally testable sets to the non-projective linearizations of trees (constraint graphs). The result generalizes to the case where each node in the tree corresponds to a bounded set of string positions.

References

[Barton, 1987] Barton, G. E. J. (1987). The complexity of two-level morphology. In Barton, E., Berwick, R., and Ristad, E., editors, *Computational Complexity and Natural Languages*, pages 115–310. MIT Press, Cambridge, Mass.

[Dechter, 1990] Dechter, R. (1990). Decomposing a relation into a tree of binary relation. *Journal of Computer and System Sciences*, 41:2–24.

[Dechter, 1992] Dechter, R. (1992). Constraint networks. In *Encyclopedia of Artificial Intelligence*, pages 276–285. Wiley and Sons, 2nd edition.

[Garey and Johnson, 1979] Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, New York.

[Karttunen, 1994] Karttunen, L. (1994). Constructing lexical transducers. In *Proceedings of the 15th International Conference on Computational Linguistics*, volume 1, pages 406–411, Kioto, Japan.

[Koskenniemi, 1990] Koskenniemi, K. (1990). Finite-state parsing and disambiguation. In Karlgren, H., editor, *Papers presented to the 13th International Conference on Computational Linguistics*, volume 2, pages 229–232, Helsinki.

[Mackworth, 1977] Mackworth, A. K. (1977). Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–118.

[McNaughton and Papert, 1971] McNaughton, R. and Papert, S. (1971). *Counter-free Automata.* Research Monograph No. 65. MIT Press.

[Piitulainen, 1995] Piitulainen, J. (1995). Locally tree-shaped sentence automata and resolution of ambiguity. In *Proceedings of the 10th Nordic Conference of Computational Linguistics*, number 26 in Publ. of the Dept. of Gen. Linguistics, University of Helsinki, pages 50–58, Yliopistopaino, Helsinki.

[Pin, 1995] Pin, J.-E. (1995). Finite semigroups and recognizable languages: An introduction. In Fountain, J., editor, *NATO Advanced Study Institute Semigroups, Formal Language and Groups*, pages 1–32. Kluwer Academic Publishers.

[Tapanainen, 1997] Tapanainen, P. (1997). Applying a finite-state intersection grammar. In Roche, E. and Schabes, Y., editors, *Finite-state language processing*, pages 311–327. A Bradford Book, MIT Press.

[Voutilainen, 1997] Voutilainen, A. (1997). Designing a (finite-state) parsing grammar. In Roche, E. and Schabes, Y., editors, *Finite-state language processing*, pages 283–310. A Bradford Book, MIT Press.

[Yli-Jyrä, 1995] Yli-Jyrä, A. M. (1995). Schematic finite-state intersection parsing. In Koskenniemi, K., editor, *Short Papers Presented at the 10th Nordic Conf. of Computational Linguistics*, pages 95–103, Yliopistopaino, Helsinki.