gefördert vom
Bundesministerium
für Forschung
und Technologie

Verbmobil
Verbundvorhaben

Incremental Natural–Language Processing with Schema–Tree Adjoining Grammars

Karin Harbusch

DFKI GmbH

Vm

Report 41
Dezember 1994

Dezember 1994

Karin Harbusch

DFKI GmbH — Deutsches Forschungszentrum
für Künstliche Intelligenz
Stuhlsatzenhausweg 3
D—66123 Saarbrücken

Tel.: (+49 681) 302 5271 - (+49 681) 302 5252
Fax: (+49 681) 302 5271 - (+49 681) 302 5341
e-mail: {harbusch}@dfki.uni-sb.de

Gehört zum Antragsabschnitt: 9. Spontansprachliche und inkrementelle Generierung

Die vorliegende Arbeit wurde im Rahmen des Verbundvorhabens Verbmobil vom Bundesministerium für Forschung und Technologie (BMFT) unter dem Förderkennzeichen 01 IV 101 G gefördert. Die Verantwortung für den Inhalt dieser Arbeit liegt bei der Autorin.

Abstract[1]

Schema–Tree Adjoining Grammars *are proposed for compressing natural–language grammars. Their rules allow for regular expressions associated with inner nodes of elementary trees referring to individual sons of inner nodes in order to describe their occurrences and repetitions in the set of elementary trees specified.*

In the following, the adequacy of representing a possibly infinite set of trees by one scheme is discussed for **incremental natural–language processing** *where the input is consumed and processed piecemeal. The basic idea is to avoid decisions for which not enough information is available at the moment by the condensed representation of all currently valid alternatives in terms of semi–instantiated schemata. We present an extended Earley processing which allows for mixing partially instantiated trees and uninstantiated schemata.*

1 Motivation

The formalism of *Schema–Tree Adjoining Grammars (Schema–TAGs)* was introduced by David Weir [Weir 87] in order to compress syntactic descriptions. For that purpose, a TAG (see, e.g., [Joshi 85]) is extended with respect to the facility to specify a regular expression at each inner node of an elementary tree. The resulting tree is called *schematic elementary tree.* Such a tree denotes an elementary tree set just as a regular expression denotes some regular set.

In the following, the advantages of representing a possibly infinite set of different trees by one scheme are demonstrated for the ongoing processing in natural–language processing especially if incrementality is presupposed. *Incrementality* means that the input is provided piecemeal to the component. Its processing must be able to work on such preliminary information. Obviously, one main problem of non–incremental natural–language processing is worsened namely that decisions must be realized with incomplete knowledge or must be postponed. For instance, in a translation system the input of the generator can be incomplete with respect to *translation mismatches* (see, e.g., [Kameyama et al. 91]). However, in an incremental system the missing knowledge can arrive later so that it is advantegous[2] to represent all valid alternatives in a condensed way for the meantime. If further knowledge arrives later on, the representation should allow for

[1]Also appeared in '3e Colloque International sur les grammaires d'Arbres Adjoints (TAG+3)'. Technical Report TALANA-RT-94-01, TALANA, Universite' Paris 7, 1994.

[2]With respect to the difficult problem of realizing *repair facilities* which become necessary in a system only evaluating the currently best alternative.

an efficient integration of this new information in order to represent the new set of currently valid alternatives. In a non–incremental system, the two strategies of best–first and breadth–first search are equally entitled because here no repair will take place.

For the strategy of breadth–first search, we propose an extended Earley processing. Thereby, a mixture of uninstantiated schemata and instantiated tree structures is represented in the individual item lists. The border line between the two modes is marked in an Earley typical way by additional dot positions in the regular expressions. Our claim is that this processing allows for an adequate representation of not yet solved ambiguities. Furthermore, we demonstrate the simplicity of the reshaping of the currently valid set of alternatives in order to form a new item.

In the next section, the formalism of Schema–TAGs is introduced and illustrated by a linguistic example. Then we present the processing modes for parsing and generating with Schema–TAGs. Finally, we mention future work.

2 Schema–Tree Adjoining Grammars

Basically, David Weir adopted the idea of schematic rules from *Generalized Phrase Structure Grammar (GPSG)* [Gazdar et al. 85] and *Categorial Grammar (CG)* (see, e.g., [Steedman 85]). He proposes to allow regular expressions at internal nodes of elementary trees. Each schematic elementary tree denotes an elementary tree set just as a regular expression denotes some regular set. Each schematic initial or auxiliary tree will denote a (perhaps infinite) set of initial or auxiliary trees.

A regular expression — characterizing the possible branchings for a node in the corresponding elementary trees — refers to subschemata by unique natural numbers, i.e. the individual sons of a node are associated with numbers enumerated from left to right starting with 1. A subdescription can be cut off by specifying $|a-b|$. The number a refers to the according son and b $(\in (I\!N \times (\{.\} \times I\!N)^{+}))$ denotes a path of numbers which leads to the suppressed structure.

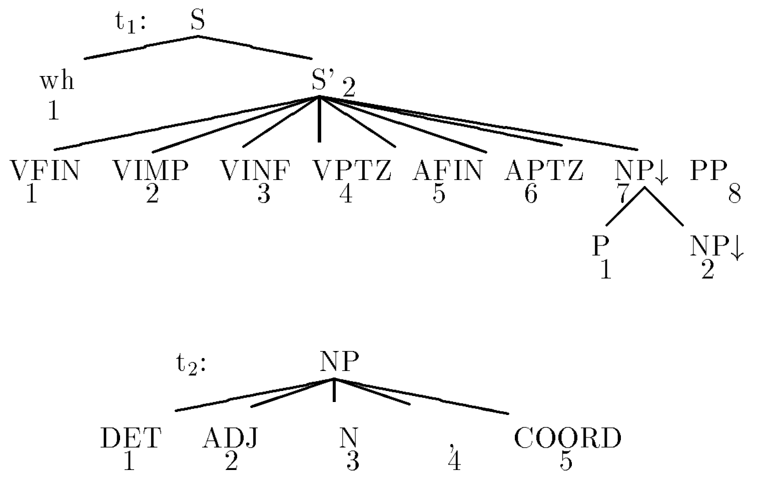
Basically, $|a|$ and $|b-c|$ are regular expressions, where a, b and c are numbers referring to sons of the node the regular expression is associated with. For a and b arbitrarily complex regular expressions, further regular expressions are defined inductively as $|a| \cdot |b|$, $|a| + |b|$, $|a|^{*}$, $|a|^{+}$, and $|a|^{(0|n)}$ with the following meanings. The concatenation represents neighborhood of the two addressed subschemata — after their own evaluation — in the resulting elementary tree. The "+" relates alternatives for realizing elementary trees. The Kleene Star allows for an infinite repetition (inclusive zero) of the addressed subscheme. The exponent "+" represents the Kleene Star exclusive zero repetition. The exponent (0|n) (n $\in$ $I\!N$) enumerates the n+1 alternatives where the corresponding subtree can occur up to n times.

This kind of representation is illustrated by Figure 1 which shows the compressing property of Schema–TAGs by a schematic initial tree for all sentence types in all tenses in active and passive voice. Such a tree represents the first hypothesis for analysis or generation at the beginning of processing. Newly arriving knowledge, e.g., a first scanned terminal in the analysis, rules out alternatives. For that purpose, parts of the regular expressions are marked to be evaluated in order to express the structures determined. In the following, we argue for the advantages of this kind of processing in natural–language parsing and generation.

3 Incremental Processing with Schema–TAGs

Incremental natural–language processing is a phenomenon well studied by psycholinguists (see, e.g., [Levelt 89]). For instance, during generation selfcorrections can be observed indicating that humans start uttering before the output is completely planned (*"The 8th and the 9th and ... oops ... only the 8th of August works with me."*). However, here we do not want to argue how human incremental processing actually takes place. Instead, we propose Schema–TAGs as an adequate technical representation for a computer simulation where all currently valid alternatives can be described in an efficient compressed way. First, the algorithm is formulated for analysis as an extension of an Earley–based TAG parser. For generation, similarly a partial instantiation process is formulated but under the restriction that here no ordering of the input from left to right can be presupposed.

Basically, it must be noted that it does not work to precompile the corresponding TAG for a given Schema–TAG because the resulting grammar can be infinite. Therefore — as it was proposed for direct ID/LP parsing in [Shieber 83] — the evaluation of concrete elementary trees is done on demand. Actually, we do not explicitly produce elementary trees but mark parts of the regular expression to be evaluated. This means, the scheme is the only structural represenation for the set of elementary trees. Therefore the dot between branches of a scheme only represents the current state of processing. However, the branches in the actual trees are annotated in the relevant regular expression by an additional dot position. This representation requires an adaptation of the predict, the scanning and the completion step. The overall processing is successful iff there exists an alternative in the final regular expression which is completely evaluated. 'Final' means that the regular expression is associated with the root node of an initial



Regular expressions in these trees:
at S in t_1:
$|1| \cdot |2 - 2.7| + |2|$
(i.e. wh–question or not)
at S' in t_1:
$(|2| + |1|) \cdot |7|^{(0|3)} \cdot |8|^*$
(i.e. imperative clause or not wh–question in present or past tense)
$+ |5| \cdot |7|^{(0|3)} \cdot |8|^* \cdot |4| \cdot |6|^{(0|1)}$
(i.e. active voice questions for future, past perfect and past participle; furthermore, passive voice questions for all tenses)
$+ (|7| + |8|) \cdot ((|1| \cdot |7|^{(0|3)} \cdot |8|^*) + (|5| \cdot |7|^{(0|3)} \cdot |8|^* \cdot |4| \cdot |6|^{(0|1)}))$
(i.e. declarative sentences in active and passive voice).
at NP in t_2 (substitution tree):
$|1|^{(0|1)} \cdot (|2| \cdot (|4| \cdot |2|)^{(0|1)} \cdot (|5| \cdot |2|)^{(0|1)})^{(0|1)} \cdot |3|$
$\cdot ((|4| \cdot X)^{(0|1)} \cdot (|5| \cdot X)^{(0|1)})^{(0|1)}$
where X abbreviates the whole term in the line before, i.e. coordination of adjectives; the whole expression represents NP coordination.

Figure 1: Example of initial schematic trees.

tree which is labelled with the start symbol S. In this tree all adjoinings and substitutions have taken place according to the input.

In terms of an Earley–based parser (see, e.g., [Schabes 90]) instead of an elementary tree a complete scheme is predicted. The nonterminal in question to be replaced by such a rule is computed by a function NEXT. As stated before, it visits all alternatives of the regular expression and enumerates all brothers which can follow the dot position specified there. In other words, another branching of a concrete tree is determined. For instance, NEXT at the beginning of the regular expression of the root node of t_2 in Figure 1 comes up with $|1|$ (DET), $|2|$ (ADJ) and $|3|$ (N). In a scanner and a completer step additionally an element in the regular expression is evaluated by shifting the additionally defined $\odot$–dot in the regular expression. For instance, $|1|^*$ becomes $|1| \odot |1|^*$ and $|1|^{(0|1)}$ becomes $|1| \odot \epsilon$. Completion functions on the structural level as normally. Since in the regular expression the concrete structures of elementary trees are determined, the process to rule out a complete scheme tests the following condition. An alternative fails iff the "normal" upper–right dot — in the sense of Schabes — is at the end of the rule and there occurs no alternative in the regular expression that is completely evaluated, i.e. the $\odot$–marker is at the end of the alternative.

The advantage for parsing with such a condensed grammar can be illustrated by applying the schemata in Figure 1. The scheme t_1 is predicted in I_0. During the ongoing processing, all object positions are processed once (e.g., producing a structure as $|5| \cdot |7| \cdot |8| \cdot |4| \odot |6|^{(0|1)}$). The next scanning step determines one alternative with respect to active or passive voice.

Similarly, for generation these partially instantiated schemata are helpful. However, the ordering in the input specification is completely free in an incremental mode. Therefore, explicit $\odot$–beginning ($\bigoplus$) and $\odot$–end markers ($\bigotimes$) are introduced. For instance, if all arguments are given without determining the associated verb, in all alternatives of t_1 in Figure 1 a subexpression of the form, e.g., $\bigoplus |7| \cdot |7| \cdot |8| \bigotimes$ is instantiated. For generation, NEXT must be able to predict in both directions, i.e. additionally to the left of $\bigoplus$ and as before to the right of $\bigotimes$.

We have now started to implement such a module which should be able to work *bidirectionally* for parsing and generation. For this purpose, we are writing a larger grammar fragment for German on the basis of this grammar formalism. This leads to the following open problems.

4 Future Work

It turned out that a natural–language S–TAG consists of a small number of trees but the large number of regular expressions. This brings up the question whether this organization influences the execution time of the algorithm.

A theoretical problem is the question if other annotations than regular expressions are meaningful. One idea is to go " one step upwards" and combine TAGs with contextfree grammars. This leads to a specific type of *Synchronous Rewriting Systems* as defined in the paper of [Harbusch & Poller 94] in this volume. The contextfree annotation serves as a stack where a characterization of the derivation can be stored. For instance, two additional exponents can be obtained by this method. One line of argumentation aims at the adequate power of the corresponding formalism for natural language description. Another open point is the question whether this representation allows for the efficient handling of alternatives. At the moment we try to represent linguistic descriptions in this formalism in order to answer these questions.

References

[Gazdar et al. 85] G. **Gazdar**, E. **Klein**, G.K. **Pullum** and I.A. **Sag**. *Generalized Phrase Structure Grammars.* Blackwell Publishing, Oxford, 1985.

[Harbusch & Poller 94] K. **Harbusch** and P. **Poller**. *Structural Translation with Synchronous Rewriting Systems.* See this volume.

[Joshi 85] A. **Joshi**. *An Introduction to TAGs.* Technical Report MS-CIS-86-64, University of Pennsylvania, Philadelphia, PA, 1985.

[Kameyama et al. 91] M. **Kameyama**, R. **Ochitani**, and S. **Peters**. *Resolving Translation Mismatches With Information Flow.* In Procs. of the 29th Annual Meeting of the Association for Computational Linguistics, Berkeley, CA, pp. 193–200, 1991.

[Levelt 89] W.J.M. **Levelt**. *Speaking: From Intention to Articulation.* The MIT Press, Cambridge, MA, 1989.

[Schabes 90] Y. **Schabes**. *Mathematical and Computational Aspects of Lexicalized Grammars.* Ph.D. Thesis, University of Pennsylvania, Philadelphia, PA, 1990.

[Shieber 83] S.M. **Shieber**. *Direct Parsing of ID/LP Grammars.* Technical Note 291R, SRI International, Menlo Park, CA, 1983.

[Steedman 85] M.J. **Steedman**. *Dependency and Coordination in the Grammar of Dutch and English.* Language, Vol. 61, pp. 523–568, 1985.

[Weir 87] D. **Weir**. *Characterizing Mildly Context–Sensitive Grammar Formalisms.* Ph.D. Proposal, Univ. of Pennsylvania, Philadelphia, PA, 1987.