An HPSG-to-CFG Approximation of Japanese

Bernd Kiefer, Hans-Ulrich Krieger, Melanie Siegel
German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3, D-66123 Saarbr¨ucken
{kiefer,krieger,siegel}@dfki.de

Abstract

We present a simple approximation method for turning a Head-Driven Phrase Structure Grammar into a context-free grammar. The approximation method can be seen as the construction of the least fixpoint of a certain monotonic function. We discuss an experiment with a large HPSG for Japanese.

1 Introduction

This paper presents a simple approximation method for turning an HPSG (Pollard and Sag, 1994) into a context-free grammar. The theoretical underpinning is established through a least fixpoint construction over a certain monotonic function, similar to the instantiation of a rule in a bottom-up passive chart parser or to partial evaluation in logic programming; see (Kiefer and Krieger, 2000a).

1.1 Basic Idea

The intuitive idea underlying our approach is to generalize in a first step the set of all lexicon entries. The resulting structures form equivalence classes, since they abstract from word-specific information, such as `FORM` or `STEM`. The abstraction is specified by means of a restrictor (Shieber, 1985), the so-called *lexicon restrictor*. The grammar rules/schemata are then instantiated via unification, using the abstracted lexicon entries, yielding derivation trees of depth 1. We apply the *rule restrictor* to each resulting feature structure, which removes all information contained only in the daughters of the rule. Due to the *Locality Principle* of HPSG, this deletion does not alter the set of derivable feature structures. Since we are interested in a finite fixpoint from a practical point of view, the restriction also gets rid of information that will lead to infinite growth of feature structures during derivation. Additionally, we throw away information that will not restrict the search space (typically, parts of the semantics). The restricted feature structures (together with older ones) then serve as the basis for the next instantiation step. Again, this gives us feature structures encoding a derivation, and again we are applying the rule restrictor. We proceed with the iteration, until we reach a fixpoint, meaning that further iteration steps will not add (or remove) new (or old) feature structures.

Our goal, however, is to obtain a context-free grammar, but since we have reached a fixpoint, we can use the entire feature structures as (complex) context-free symbols (e.g., by mapping them to integers). By instantiating the HPSG rules a final time with feature structures from the fixpoint, applying the rule restrictor and finally classifying the resulting structure (i.e., find the right structure from the fixpoint), one can easily obtain the desired context-free grammar (CFG).

1.2 Why is it Worth?

Approximating an HPSG through a CFG $\mathcal{G}$ is interesting for the following practical reason: assuming that we have a CFG that comes close to an HPSG, we can use the CFG as a cheap filter (running time complexity is $O(|\mathcal{G}|^2 \times n^3)$ for an arbitrary sentence of length n). The main idea is to use the CFG first and then let the HPSG deterministically replay the derivations licensed by the CFG. The important point here is that one can find for every CF production exactly one and only one HPSG rule. (Kasper et al., 1996) describe such an approach for word graph parsing which employs only the relatively unspecific CF backbone of an HPSG-like grammar. (Diagne et al., 1995) replaces the CF backbone through a restriction of the original HPSG. This grammar, however, is still an unificationbased grammar, since it employs coreference constraints.

1.3 Content of Paper

In the next section, we describe the Japanese HPSG that is used in Verb*mobil*, a project that deals with the translation of spontaneously spoken dialogues between English, German, and Japanese speakers. After that, section 3 explains a simplified, albeit correct version of the implemented algorithm. Section 4 then discusses the outcome of the approximation process.

2 Japanese Grammar

The grammar was developed for machine translation of spoken dialogues. It is capable of dealing with spoken language phenomena and ungrammatical or corrupted input. This leads on the one hand to the necessity of robustness and on the other hand to ambiguities that must be dealt with. Being used in an MT system for spoken language, the grammar must firstly accept fragmentary input and be able to deliver partial analyses, where no spanning analysis is available. A complete fragmentary utterance could, e.g., be:

daijoubu
okay

This is an adjective without any noun or (copula) verb. There is still an analysis available.

If an utterance is corrupted by not being fully recognized, the grammar delivers analyses for those parts that could be understood. An example would be the following transliteration of input to the MT system:

sou	*desu*	*ne*	*watakushi*
so	COP	TAG	I
no	*hou*	*wa*	*daijoubu*
GEN	side	TOP	okay
desu	*da ga*	*kono*	*hi*
COP	but	this	day
wa	*kayoubi*	*desu*	*ne*
TOP	Tuesday	COP	TAG

(lit.: Well, it is okay for my side, but this day is Tuesday, isn't it?)

Here, analyses for the following fragments are delivered (where the parser found *opera wa* in the word lattice of the speech recognizer):

sou	*desu*	*ne*	*watakushi*
so	COP	TAG	I
no	*hou*	*wa*	*daijoubu*
GEN	side	TOP	okay
desu			
COP			

(Well, it is okay for my side.)

opera	*wa*
opera	TOP

(The opera)

kono	*hi*	*wa*	*kayoubi*
this	day	TOP	Tuesday
desu	*ne*		
COP	TAG		

(This day is Tuesday, isn't it?)

Another necessity for partial analysis comes from real-time restrictions imposed by the MT system. If the parser is not allowed to produce a spanning analysis, it delivers best partial fragments.

The grammar must also be applicable to phenomena of spoken language. A typical problem is the extensive use of topicalization and even omission of particles. Also serialization of particles occur more often than in written language, as described in (Siegel, 1999). A well-defined type hierarchy of Japanese particles is necessary here to describe their functions in the dialogues.

Extensive use of honorification is another significance of spoken Japanese. A detailed description is necessary for different purposes in an MT system: honorification is a syntactic restrictor in subject-verb agreement and complement sentences. Furthermore, it is a very useful source of information for the solution of zero pronominalization (Metzing and Siegel, 1994). It is finally necessary for Japanese generation in order to find the appropriate honorific forms. The sign-based information structure of HPSG (Pollard and Sag, 1994) is predestined to describe honorification on the different levels of linguistics: on the syntactic level for agreement phenomena, on the contextual level for anaphora resolution and connection to speaker and addressee reference, and via co-indexing on the semantic level. Connected to honorification is the extensive use of auxiliary and light verb constructions that require solutions in the areas of morphosyntax, semantics, and context (see (Siegel, 2000) for a more detailled description).

Finally, a severe problem of the Japanese grammar in the MT system is the high potential of ambiguity arising from the syntax of Japanese itself, and especially from the syntax of Japanese spoken language. For example, the Japanese particle *ga* marks verbal arguments in most cases. There are, however, occurrences of *ga* that are assigned to verbal adjuncts. Allowing *ga* in any case to mark arguments or adjuncts would lead to a high potential of (spurious) ambiguity. Thus, a restriction was set on the adjunctive *ga*, requiring the modified verb not to have any unsaturated *ga* arguments.

The Japanese language allows many verbal arguments to be optional. For example, pronouns are very often not uttered. This phenomenon is basic for spoken Japanese, such that a syntax urgently needs a clear distinction between optional and obligatory (and adjacent) arguments. We therefore used a description of subcategorization that differs from standard HPSG description in that it explicitly states the optionality of arguments.

3 Basic Algorithm

We start with the description of the top-level function *HPSG2CFG* which initiates the approximation process (cf. section 1.1 for the main idea). Let $\mathcal{R}$ be the set of all rules/rule schemata, $\mathcal{L}$ the set of all lexicon entries, R the rule restrictor, and L the lexicon restrictor. We begin the approximation by first abstracting from the lexicon entries $\mathcal{L}$ with the help of the lexicon restrictor L (line 5 of the algorithm). This constitutes our initial set T_0 (line 6). Finally, we start the fixpoint iteration calling *Iterate* with the necessary parameters.

```
1 $HPSG2CFG(\mathcal{R}, \mathcal{L}, R, L) :\Longleftrightarrow$
2   **local** $T_0$;
3   $T_0 := \emptyset$;
4   **for each** $l \in \mathcal{L}$
5     $l := L(l)$;
6     $T_0 := T_0 \cup \{l\}$;
7   $Iterate(\mathcal{R}, R, T_0)$.
```

After that, the instantiation of the rule schemata with rule/lexicon-restricted elements from the previous iteration T_i begins (line 11–14). Instantiation via unification is performed by *Fill-Daughters* which takes into account a single rule r and T_i, returning successful instantiations (line 12) to which we apply the rule restrictor (line 13). The outcome of this restriction is added to the actual set of rule-restricted feature structures T_{i+1} iff it is new (remember how set union works; line 14). In case that really new feature structures have not been added during the current iteration (line 15), meaning that we have reached a fixpoint, we immediately exit with T_i (line 16) from which we generate the context-free rules as indicated in section 1.1. Otherwise, we proceed with the iteration (line 17).

```
8  $Iterate(\mathcal{R}, R, T_i) :\Longleftrightarrow$
9    **local** $T_{i+1}$;
10   $T_{i+1} := T_i$;
11   **for each** $r \in \mathcal{R}$
12     **for each** $t \in Fill\text{-}Daughters(r, T_i)$ **do**
13       $t := R(t)$;
14       $T_{i+1} := T_{i+1} \cup \{t\}$;
15   **if** $T_i = T_{i+1}$
16     **then return** $Compute\text{-}CF\text{-}Rules(\mathcal{R}, T_i)$
17     **else** $Iterate(\mathcal{R}, R, T_{i+1})$.
```

We note here that the pseudo code above is only a na¨ıve version of the implemented algorithm. It is still correct, but not computationally tractable when dealing with large HPSG grammars. Technical details and optimizations of the actual algorithm, together with a description of the theoretical foundations are described in (Kiefer and Krieger, 2000a). Due to space limitations, we can only give a glimpse of the actual implementation.

Firstly, the most obvious optimization applies to the function *Fill-Daughters* (line 12), where the number of unifications is reduced by avoiding recomputation of combinations of daughters and rules that already have been checked. To do this in a simple way, we split the set T_i into $T_i \setminus T_{i-1}$ and T_{i-1} and fill a rule with only those permutations of daughters which contain at least one element from $T_i \setminus T_{i-1}$. This guarantees checking of only those configurations which were enabled by the last iteration.

Secondly, we use techniques developed in (Kiefer et al., 1999), namely the so-called *rule filter* and the *quick-check method*. The rule filter precomputes the applicability of rules into each other and thus is able to predict a failing unification using a simple and fast table lookup. The quick-check method exploits the fact that unification fails more often at certain points in feature structures than at others. In an off-line stage, we parse a test corpus, using a special unifier that records all failures instead of bailing out after the first one in order to determine the most prominent failure points/paths. These points constitute the so-called quick-check vector. When executing a unification during approximation, those points are efficiently accessed and checked using type unification prior to the rest of the structure. Exactly these quick-check points are used to build the lexicon and the rule restrictor as described earlier (see fig. 1). During our experiments, nearly 100% of all failing unifications in *Fill-Daughters* could be quickly detected using the above two techniques.

Thirdly, instead of using set union we use the more elaborate operation during the addition of new feature structures to T_{i+1}. In fact, we add a new structure only if it is not subsumed by some structure already in the set. To do this efficiently, the quick-check vectors described above are employed here: before performing full feature structure subsumption, we pairwise check the elements of the vectors using type subsumption and only if this succeeds do a full subsumption test. If we add a new structure, we also remove all those structures in T_{i+1} that are subsumed by the new structure in order to keep the set small. This does not change the language of the resulting CF grammar because a more general structure can be put into at least those daughter positions which can be filled by the more specific one. Consequently, for each production that employs the more specific structure, there will be a (possibly) more general production employing the more general structure in the same daughter positions. Extending feature structure subsumption by quick-check subsumption definitely pays off: more than 98% of all failing subsumptions could be detected early.

Further optimizations to make the algorithm works in practice are described in (Kiefer and Krieger, 2000b).

4 Evaluation

The Japanese HPSG grammar used in our experiment consists of 43 rule schemata (28 unary, 15 binary), 1,208 types and a test lexicon of 2,781 highly diverse entries. The lexicon restrictor, as introduced in section 1.1 and depicted in figure 1, maps these entries onto 849 lexical abstractions. This restrictor tells us which parts of a feature structure have to be deleted—it is the kind of restrictor which we are usually going to use. We call this a *negative* restrictor, contrary to the *positive* restrictors used in the PATR-II system that specify those parts of a feature structure which will survive after restricting it. Since a restrictor could have reentrance points, one can even define a *recursive* (or cyclic) restrictor to foresee recursive embeddings as is the case in HPSG.

The rule restrictor looks quite similar, cutting off additionally information contained only in the daughters. Since both restrictors remove the `CONTENT` feature (and hence the semantics which is a source of infinite growth), it happened that two very productive head-adjunct schemata could be collapsed into a single rule. This has helped to keep the number of feature structures in the fixpoint relatively small.

We reached the fixpoint after 5 iteration steps, obtaining 10,058 feature structures. The computation of the fixpoint took about 27.3 CPU hours on a 400MHz SUN Ultrasparc 2 with Franz Allegro Common Lisp under Solaris 2.5. Given the feature structures from the fixpoint, the 43 rules might lead to $28 \times 10,058 + 15 \times 10,058 \times 10,058 = 1,517,732,084$ CF productions in the worst case. Our method produces 19,198,592 productions, i.e., 1.26% of all possible ones. We guess that the enormous set of productions is due the fact that the grammar was developed for spoken Japanese (recall section 2 on the ambiguity of Japanese). Likewise, the choice of a 'wrong' restrictor often leads to a dramatic increase of structures in the fixpoint, and hence of CF rules—we are not sure at this point whether our restrictor is a good compromise between the specificity of the context-free language and the number of context-free rules. We are currently implementing a CF parser that can handle such an enormous set of CF rules. In (Kiefer and Krieger, 2000b), we report on a similar experiment that we carried out using the English Verb*mobil* grammar, developed at CSLI, Stanford. In this paper, we showed that the workload on the HPSG side can be drastically reduced by using a CFG filter, obtained

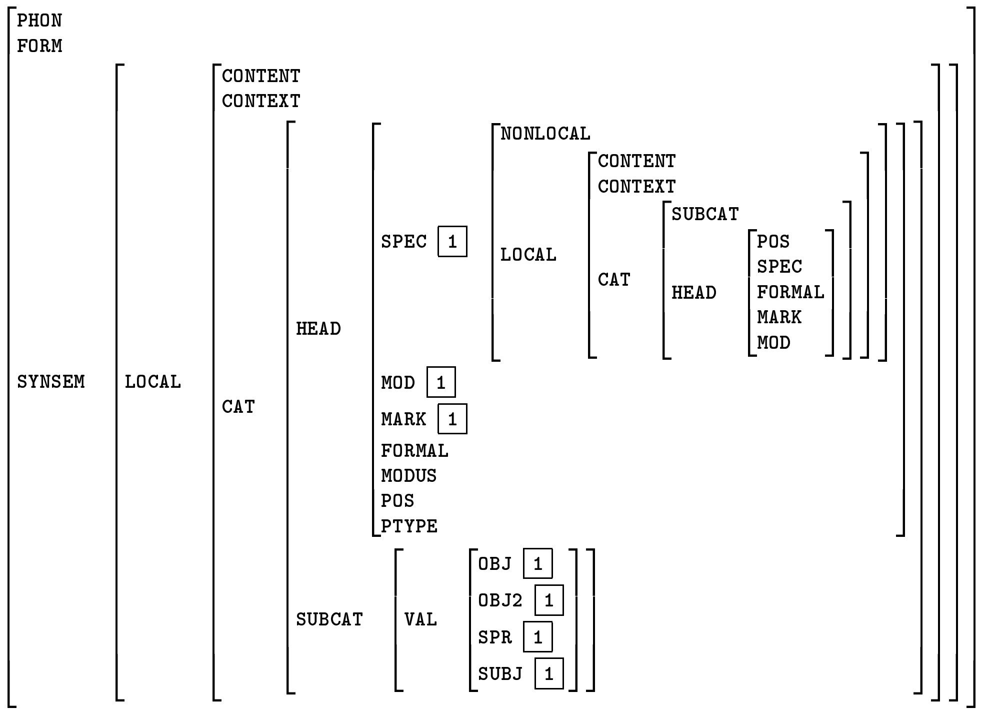

Figure 1: The lexicon restrictor used during the approximation of the Japanese grammar. In addition, the rule restrictor cuts off the `DAUGHTERS` feature.

from the HPSG. Our hope is that these results can be carried over to the Japanese grammar.

Acknowledgments

This research was supported by the German Ministry for Education, Science, Research, and Technology under grant no. 01 IV 701 V0.

References

Abdel Kader Diagne, Walter Kasper, and Hans-Ulrich Krieger. 1995. Distributed parsing with HPSG grammars. In *Proceedings of the 4th International Workshop on Parsing Technologies, IWPT'95*, pages 79–86.

Walter Kasper, Hans-Ulrich Krieger, Jörg Spilker, and Hans Weber. 1996. From word hypotheses to logical form: An efficient interleaved approach. In D. Gibbon, editor, *Natural Language Processing and Speech Technology*, pages 77–88. Mouton de Gruyter, Berlin.

Bernd Kiefer and Hans-Ulrich Krieger. 2000a. A context-free approximation of Head-Driven Phrase Structure Grammar. In *Proceedings of the 6th International Workshop on Parsing Technologies, IWPT2000*, pages 135–146.

Bernd Kiefer and Hans-Ulrich Krieger. 2000b. Experiments with an HPSG-to-CFG approximation. Research report.

Bernd Kiefer, Hans-Ulrich Krieger, John Carroll, and Rob Malouf. 1999. A bag of useful techniques for efficient and robust parsing. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 473–480.

Dieter Metzing and Melanie Siegel. 1994. Zero pronoun processing: Some requirements for a Verbmobil system. Verbmobil-Memo 46.

Carl Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar.* Studies in Contemporary Linguistics. University of Chicago Press, Chicago.

Stuart M. Shieber. 1985. Using restriction to extend parsing algorithms for complex-feature-based formalisms. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, pages 145–152.

Melanie Siegel. 1999. The syntactic processing of particles in Japanese spoken language. In *Proceedings of the 13th Pacific Asia Conference on Language, Information and Computation*, pages 313–320.

Melanie Siegel. 2000. Japanese honorification in an HPSG framework. In *Proceedings of the 14th Pacific Asia Conference on Language, Information and Computation*, pages 289–300.