Handling Syntactic Extra-Grammaticality

Fuliang Weng

Computer Science Department and Computing Research Lab
New Mexico State University, Las Cruces, NM 88003
email: fweng@nmsu.edu

Abstract

This paper reviews and summarizes six different types of extra-grammatical phenomena and their corresponding recovery principles at the syntactic level, and describes some techniques used to deal with four of them completely within an Extended GLR parser (EGLR). Partial solutions to the remaining two by the EGLR parser are also discussed. The EGLR has been implemented.

1 Introduction

Extragrammatical phenomena in natural languages are very common and there has been much effort devoted to dealing with them (Carbonell - Hayes, 1983; DARPA 1991, 1992). Although (Generalized)LR parsers have many merits when applied to NL, most progress with extra-grammatical phenomena has been through rule-based systems, in contrast to the applications of LR parsers in programming languages. In this paper some techniques are developed to extend the ability of a (G)LR parser in dealing with extra-grammatical phenomena, though similar techniques can also be applied in other parsers. The extended GLR (EGLR) parser is implemented.

In section 2, six types of extra-grammatical phenomena at the syntax level are classified, and then five recovery principles are introduced. Section 3 then describes the techniques used in the EGLR parser. A correctness theorem is given, and a property is also presented, showing that any sentence can be accepted if the relaxation parameter is set large enough. Section 4 gives some examples to show how the EGLR parser works. Section 5 briefly discusses the sixth type of extra-grammatical phenomenon and relevant issues, and other people's work is also compared.

2 Categorizing Extra-grammaticality at the Syntax Level

Among discussions of extra-grammaticality in natural language processing, (Carbonell — Hayes, 1983) gave a comprehensive and complete overview in this area . Here we try to rephrase these extra-grammatical phenomena from the viewpoint of purely structural possibilities at the syntactic level.

Following (Carbonell — Hayes, 1983), we use extra-grammaticality to refer to phenomena in which sequences of words are not covered by current grammar rules. In doing so, we try, at this moment, to avoid unnecessary debate about the possibility of drawing a clear line between grammatical and ungrammatical phenomena. At the end of the paper, we shall show an interesting result with a deviation degree parameter, produced by our parser that may provide some hint for understanding (un)grammaticality. In what follows, extra-grammatical phenomena contain both ungrammatical and uncovered grammatical phenomena.

The six types of syntactic extra-grammaticalities are:[1]

Phenomenon 1: the absence of a word's cate-

[1]Phenomenon 1 was discussed in (Tomita, 1985) and might be also considered to be at the lexical level. Phenomena 2, 3 and 4 and the similar recovery principles were also discussed in (Aho — Peterson, 1973; Saito — Tomita, 1991).

gories.
For example, *xyz leads arbitrariness.*, when *xyz* is not present in the English dictionary and therefore the sentence is not covered by a grammar.

Phenomenon 2: category switching.
For example, *The man on the left was talking non-sense.*, when in the dictionary entry, *left* only has a category *ADJ* but not *N*, and therefore the sentence can't be covered by a grammar with only *N* as its head in *NP* rules.

Phenomenon 3: ellipses at different levels; a category or a phrasal category is missing with respect to a rule while no other rules can cover such phenomena.
For example, *He give an apple to* _, given the only rule $PP \leftarrow P\ NP$ for *PP* formation.

Phenomenon 4: redundancy; an extra category or a phrasal category occurs with respect to a rule while no other rules can cover such phenomena.
For example, *The man lives in in a house.*, there is an extra *in*, given the only rule $PP \leftarrow P\ NP$ for PP formation.

Phenomenon 5: constituent swapping; two constituents swapped their positions with respect to a rule while no other rules can cover such phenomena.
For example, *He is happy?*, given the only rule $Q \leftarrow AUX\ NP\ VP?$ for question formation.

Phenomenon 6: non-extragrammaticality failure: a constituent is covered by rules but not intended by the speaker (or writer).
For example, *The man sit on the river bank,* given the only rule $NP \leftarrow Det\ N$ for NP formation. In this case, *the river* will form a NP, instead of *the river bank.*

We put aside the last phenomenon for a moment since it needs the coordination of multiple levels besides the syntactic one.

Corresponding to these phenomena, we propose five principles which try to remedy the failures caused by them. When a parser fails, at a point, we hypothesize several alternatives according to the following five principles, which correspond to the first five types of phenomena:

Principle 1: hypothesizing all the categories as the categories of the absent of the failed word.

Principle 2: hypothesizing the complement of all the categories of the failed word.

Principle 3: hypothesizing a pseudo word which has all the categories.

Principle 4: hypothesizing that the failed word is improperly added.

Principle 5: let xy be two consecutive words (or constituents) in the input sequence that is not parsable with respect to the current grammar, hypothesizing another word (or constituent) order, i.e., replacing xy by yx in the input sequence.

It can be easily noticed that all the principles try to *assimilate* abnormal phenomena by using known rules.

But the five principles themselves alone do not solve the problem and at least another two issues have to be dealt with:

1. to determine where the failure occurs, since most principles presuppose knowing the exact position of the failed word.

2. to determine what type of failure it is, since failure itself does not inform the type of failures the parser is encountering.

In the next section, we will present some techniques to incorporate the first four principles into an extended GLR parser, and a limited version of the fifth principle can be achieved by setting the relaxation parameter to 2, a number which will be explained later.

3 The Extended GLR Parser

The reason we choose GLR parsers as our starting point for the extensions is not arbitrary: it is closely related to the purpose of settling the two issues raised at the end of the last section. Like LR parsers, GLR parsers have the ability to detect errors in an early phase. We feel comfortable with the first issue if we can tolerate the locality principle, i.e. the true error position is somewhere close to the place where the parser reports an error. As for the second issue, instead of deciding which type of error it is, we hypothesize all the possible error types and let all the corresponding hypotheses compete so that we can avoid answering *which type* of error beforehand. Again, GLR parsers provide a good platform for competing alternatives at different levels, although they require that comparisons be over the same input lengths.

Like GLR parsers, the extended GLR parser also has a generalized action table, a goto table and a parsing algorithm. The two tables are generated by a compiler generator, given a context-free grammar. Every entry $(s,\ t)$ in the action table may either contain a set of actions (i.e., A(s, t) $= \{a_i\}$), where s stands for a particular state, t stands for a particular terminal, and a_i stands for actions which could be shifting, reducing or accepting, and A(s, t) can also be empty. The last case (i.e., A(s,t) is empty) indicates that, in state s, it is impossible to meet terminal t if the input is grammatical: in other words, the parser will report an error if, for any current state s and any category c associated with the current word, A(s, c) is empty. Like the GLR parsers when an input sentence appears, the EGLR algorithm reads one word after another from left to right, goes from one state to another, and does what the two tables specify: reduce, shift, etc. And the actions of the EGLR parser, i.e., reduce, shift and accept, are very similar to the ones in GLR parsers, except in places that will be specified later. The main differences between the GLR parsers and the EGLR parser are caused by the additional requirement of EGLR, i.e., allowing different hypotheses to compete when an error is detected (or a mismatch occurs).

The realization of different hypotheses is not so direct if we still want to stick to the following idea: allow different principles to compete at a same time, whenever it is possible, since different principles create different new sentence lengths, i.e. principles 1 and 2 do not change the length of the input sentence, principle 3 increases the length by one unit, and principle 4 reduces the length by one unit. The alignment of these length discrepancies and the compensation of its effect are realized as follows (see the next section for examples):

1. Creating a special terminal called ∗span∗;
2. In the action table, adding an additional column which has ∗span∗ as its terminal and $S\ i$ as the value of the row of state i;
3. In the GLR parsing algorithm, the special terminals ∗span∗ along any reduce path in the graph-structured stack are ignored when a reduce action is taken.
4. Reconstructing the input sentence in the assumed error place cat_i as follows:
$\ldots\ \{*\text{span}*\} \sqcup \overline{cat_i},\ \{*\text{span}*\} \sqcup cat_i,\ \ldots$ [2]

Among the four steps, the first three are for the parser itself and the fourth is for the input sentence. Since there might be more than one error in an input sentence, the fourth step can be repeated. We use a relaxation parameter to characterize the maximum number of times allowed in performing such relaxation.

Notice that in the fourth step, the four combinations of the two successive category sets, i.e. $\{*\text{span}*\}\ \{*\text{span}*\}$, $\overline{cat_i}\ \{*\text{span}*\}$, $\{*\text{span}*\}\ cat_i$ and $\overline{cat_i}\ cat_i$, are exactly the first four principles [3] plus the original sentence. Since the original sentence is blocked at that point the reconstruction realizes the first four principles at the same time.

The first four principles are all in favor of local mismatches. In order to accommodate the fifth principle better, which is non-local, simple alignment is not sufficient and a more complicated mechanism needs to be used. One method for this is to use the notion of parameterization in a universal grammar. Notice that swapping two linguistic units requires the same underlying mechanism as resetting a parameter in X-bar theory (Chomsky, 1980; Gibson, 1989; Nyberg III, 1989).

Although the parser only encoded the first four principles directly, it is not difficult to see that a limited version of the fifth principle, i.e. swapping two adjacent words, is implied by the

[2] The curly parentheses indicate sets, $\sqcup$ set union, and the overline on a set refers to the set complement operation w.r.t. the set of all the terminals, *terms*.

[3] The first two principles share much in common and are dealt with in a same way.

first four principles with the relaxation parameter being 2. The details will be illustrated by an example in section 4.

One may not agree with the locality principle, i.e. that the position of real error(s) is not necessarily the place where there is no entry in the generalized action table (GAT), in other words, there are cases that can be characterized by phenomenon 6, which we will call *extra-grammatical garden path*, abbreviated as *xgp*. Obviously, *xgp* can also occur in combination with the first 5 types of phenomena. One solution to the *xgp* problem is to make some changes to cat_{i-1}. Here, we can reconstruct the input sentence at cat_{i-1} in step 4 of the alignment procedure, instead of cat_i.

It is not necessary that cat_{i-1} should be modified. In general, as a *k*-step *xgp*, cat_{i-k} is modified, where $k \geq 1$ and the choice of k can be based on knowledge sources at other levels, such as semantics and discourse, when $word_{i-k}$ seem to be a particularly odd fit in the local context.

Note too that step 4 in the alignment procedure can be replaced by any reasonable hypothetical sequence of categories, including the pseudo-terminal *span*. We consider the one presented here as a good candidate.

One may notice that in the phenomena and the principles enumerated, we not only allow words but also constituents to be manipulated, while up till this moment in the EGLR only words are taken as the basic units. One way to deal with constituents is as follows:

We introduce a notion called *virtual terminals*, abbreviated as *vtm*, that contain information about the corresponding non-terminals; a normal grammar is appended with a set of rules with *vtms*: $A \leftarrow *vtm_A$, where $A \in NTM$ and NTM is the set of non-terminals[4]; since $*vtm_A$ will never occur as a category of any word, the parser will not take any path containing $*vtm_A$ unless we reconstruct the input category sequence when necessary, which is very much like the case when we deal with word as unit. An example with the extended rules and their GPTs is shown in Table 3. for the set of grammar rules and its GPTs given in Tables 1 and 2.

To conclude this section, a correctness theorem and a property concerning the EGLR parser are presented as follows:[5]

Theorem 1. Let $input_0 = C_1, C_2, \ldots, C_n$ be a sequence of category sets without *span*, $input_1 = C_1, C_2, \ldots, C_{i-1}, C_{i+1}, \ldots, C_n$, and $input_2 = C_1, C_2, \ldots, C_i \sqcup \{*span*\}, \ldots, C_n$. Then a GLR parser accepts $input_0$ or $input_1$ with grammar G iff the EGLR parser accepts $input_2$ with the same grammar and relaxation parameter being 0.

Property: Given any sentence and context free grammar, we can always find a value for the relaxation parameter such that the EGLR parser accepts the sentence based on the four recovery principles.

4 Some Results Produced by the EGLR

Like (G)LR parsers, the EGLR parser has two main components: a parsing table generator and a parsing algorithm. The generator takes a context-free grammar as its input and produces an action table and a goto table; while the parsing algorithm takes a sequence of words and a value of the relaxation parameter as its input and produces results, usually with four parts:

1. a sequence of triples, whose first element is the index of the triple in the sequence, whose second element is the set of the categories attached to the word in the input sentence, and whose third element is the word itself;

2. the history of the sequences of the triples, whose first element is the index of the triple in the sequence like in part 1, whose second

[4]Some non-terminals may not be able to derive a string containing only terminals, we can use a standard algorithm to detect them in (Hopcroft — Ullman, 1979) Ch 4.4, and there is no need to have the corresponding rules.

[5]The proof of the two properties are given in (Weng, 1993).

element is either the same set of the categories as in part 1 when it is parsed, or the derived one based on the alignment procedure when a mismatch happens, and whose third element is also the same as in part 1 except in the case when a mismatch occurs and then NIL is supplied;

3. (partially) successful hypothesized sequences of categories if there are mismatches, and the number of times the relaxation process is actually performed (i.e., the value of the deviation parameter);

4. a shared-packed forest representation for the sequences in item 3.

Items 2 and 3 will not be present if the original category sequence is covered by the grammar.

We now present some simple examples in order to explain better to the reader the underlying ideas, using the grammar and its corresponding GPTs given in Table 1 and Table 2.

The grammar in Table 1 does not contain any rules for pronouns. Assuming that a pronoun occurs in an input sentence and the lexicon does provide a category (*PRON) for the pronoun (i.e., a case when phenomenon 2 happens), how will the parser behave? Example sentence 1 (*They read the book.*) shows this case, where part 1 contains a sequence of sets of categories attached to the words in the sentence together with their indices in the sentence. The indices start at 0. *1-st hypothesized sentence* line in part 2 gives a guess when the parser found that the grammar does not cover *PRON, according to the principles in section 2. After feeding the new sentence to the parser internally, the parser returns the successful category sequence in part 3 and its parsing forest in part 4. A few more words about the shared-packed forest representation are given here: each row in the list is a description of a node in the forest, and it consists of two parts: the first part is the index of a node, and the second part is a tuple; the first argument of the tuple is either a terminal or a non-terminal; if it is a terminal, a T and the word[6] associated with the terminal are followed; if it is a non-terminal the follow-up lists include all its child node sets, e.g., 10 (NP (8 9)) means that node 10 is a NP having nodes 8 and 9 as its children.

[6]NIL is filled if the terminal is from a hypothesized category set

```
Example sentence 1.

* (understander '(they read the book))
Part 1: the indexed sentence =
               ((0 (*PRON) (THEY 0 0))
                (1 (*N *V) (READ 1 1))
                (2 (*DET) (THE 2 2))
                (3 (*N) (BOOK 3 3)))
Part 2: the history of hypothesized
sentences:
  1-st hypothesized sentence:
         ((0 (*SPAN* *V *PREP *N *DET)
             (NIL -1 0))
          (1 (*SPAN* *PRON) (THEY -1 1))
          (2 (*N *V) (READ 1 2))
          (3 (*DET) (THE 2 3))
          (4 (*N) (BOOK 3 4))
          (5 ($) (NIL NIL 5)))
Part 3:
 The following hypothesized sentences
get parsed:
(((*N (NIL -1 0)) (*V (READ 1 2))
  (*DET (THE 2 3)) (*N (BOOK 3 4))))
 and the value of the deviation
parameter is 1
Part 4:
 it is accepted !
 the root of its parse forest is (12),
and the forest is:
0   0
1   (*DET T (NIL -1 0))
2   (*N T (NIL -1 0))
3   (*SPAN* T (NIL -1 0))
4   (*SPAN* T (THEY -1 1))
5   (NP (2))
6   (*V T (READ 1 2))
7   (*N T (READ 1 2))
8   (*DET T (THE 2 3))
9   (*N T (BOOK 3 4))
10   (NP (8 9))
11   (VP (6 10))
12   (S (5 11))
the end of the forest.
NIL
*
```

Example sentence 2 shows the phenomenon 1 and the result produced by the parser. It is similar to example sentence 1, and hypotheses a set of categories for the unknown word *researchers* and produces a parsing forest for that guess.

```
Example sentence 2.

* (understander '(researchers
                     understand the book))
Part 1: the indexed sentence =
              ((0 NIL (RESEARCHERS 0 0))
               (1 (*V) (UNDERSTAND 1 1))
               (2 (*DET) (THE 2 2))
               (3 (*N) (BOOK 3 3)))
Part 2: the history of hypothesized
sentences:
  1-st hypothesized sentence:
       ((0 (*SPAN* *V *PREP *N *DET)
           (NIL -1 0))
        (1 (*SPAN*) (RESEARCHERS -1 1))
        (2 (*V) (UNDERSTAND 1 2))
        (3 (*DET) (THE 2 3))
        (4 (*N) (BOOK 3 4))
        (5 ($) (NIL NIL 5)))
Part 3:
 The following hypothesized sentences
get parsed:
(((*N (NIL -1 0)) (*V (UNDERSTAND 1 2))
  (*DET (THE 2 3)) (*N (BOOK 3 4))))
 and the value of the deviation
parameter is 1
Part 4:
 it is accepted !
 the root of its parse forest is (11),
and the forest is:
0   0
1   (*DET T (NIL -1 0))
2   (*N T (NIL -1 0))
3   (*SPAN* T (NIL -1 0))
4   (*SPAN* T (RESEARCHERS -1 1))
5   (NP (2))
6   (*V T (UNDERSTAND 1 2))
7   (*DET T (THE 2 3))
8   (*N T (BOOK 3 4))
9   (NP (7 8))
10   (VP (6 9))
11   (S (5 10))
the end of the forest.
NIL
*
```

Example sentence 3 shows a combination of several possible phenomena. The category set sequence derived from sentence *the man lives in in the house* can not get parsed without relaxation, and the parser detects an error at the second *in* indexed as 4-th in the input sequence. Then its first hypothesized sequence is proposed and shown in part 2. Because the default value of the relaxation parameter is 1, this relaxation process is permitted and the indexed sequence is internally resubmitted to the parser. This time, it gets parsed. The parsing forest is shown in part 4 and a set of successfully hypothesized sequences are given in part 3.

```
Example sentence 3.

* (understander '(the man lives
                 in in the house))
Part 1: the indexed sentence =
           ((0 (*DET) (THE 0 0))
            (1 (*N) (MAN 1 1))
            (2 (*N *V) (LIVES 2 2))
            (3 (*PREP) (IN 3 3))
            (4 (*PREP) (IN 4 4))
            (5 (*DET) (THE 5 5))
            (6 (*N) (HOUSE 6 6)))
Part 2: the history of hypothesized
sentences:
  1-th hypothesized sentence:
           ((0 (*DET) (THE 0 0))
            (1 (*N) (MAN 1 1))
            (2 (*N *V) (LIVES 2 2))
            (3 (*PREP) (IN 3 3))
            (4 (*SPAN* *V *N *DET)
               (NIL -1 4))
            (5 (*SPAN* *PREP)
               (IN -1 5))
            (6 (*DET) (THE 5 6))
            (7 (*N) (HOUSE 6 7))
            (8 ($) (NIL NIL 8)))
Part 3:
 The following hypothesized sentences
get parsed:
(((*DET (THE 0 0)) (*N (MAN 1 1))
  (*V (LIVES 2 2)) (*PREP (IN 3 3))
  (*N (NIL -1 4)) (*PREP (IN -1 5))
  (*DET (THE 5 6)) (*N (HOUSE 6 7)))
 ((*DET (THE 0 0)) (*N (MAN 1 1))
  (*V (LIVES 2 2)) (*PREP (IN 3 3))
  (*DET (THE 5 6)) (*N (HOUSE 6 7))))
 and the value of the deviation
parameter is 1
Part 4:
 it is accepted !
 the root of its parse forest is (20),
and the forest is:
0   0
1   (*DET T (THE 0 0))
2   (*N T (MAN 1 1))
3   (NP (1 2))
4   (*V T (LIVES 2 2))
5   (*PREP T (IN 3 3))
6   (*DET T (NIL -1 4))
7   (*N T (NIL -1 4))
8   (*SPAN* T (NIL -1 4))
9   (NP (7))
10   (PP (5 9))
11   (VP (4 10))
12   (S (3 11))
13   (*PREP T (IN -1 5))
14   (*SPAN* T (IN -1 5))
15   (*DET T (THE 5 6))
16   (*N T (HOUSE 6 7))
17   (NP (15 16))
18   (PP (13 17))
19   (PP (5 17) (5 21))
20   (S (12 18) (3 22))
21   (NP (9 18))
22   (VP (4 19))
the end of the forest.
NIL
*
```

The above examples only show that if there is one extra-grammatical place in a single sentence the parser can deal with it. Actually, the ability of the parser is not limited to this. An optional relaxation parameter is offered (default is 1) by the parser. So, if there are multiple extra-grammatical places in one sentence, by setting this relaxation parameter properly the parser can still proceed with various guesses. Example 4 shows this situation. Sentence *the man home likes* is not covered by the grammar given in Table 1.

When the parser tries to parse the category set sequence derived from sentence *the man home likes* with the relaxation parameter being 1, it detects an error at the word *home* indexed as 2-nd in the input sequence. Then its first hypothesized sequence is proposed and shown in part 2. Because the value of the relaxation parameter is 1, this relaxation process is permitted and the indexed sequence is submitted internally to the parser. With the reconstruction of the input sequence, the parser proceeds and goes through word *likes*, and then it detects another error and can not go further because the relaxation parameter is 1 and no further relaxation is allowed.

A next example is about the same sentence being presented to the parser with the relaxation parameter being 2. The same process as the one in the previous example happens until the parser detects the second error at position 5. This time, another hypothesized sequence gets proposed and allowed to be internally resubmitted to the parser because the relaxation parameter is 2. It gets parsed. The parsing forest is shown in part 4.

The two successfully hypothesized sequences in part 3 need a little bit more explanation. The first sequence ((*DET THE) (*N MAN) (*V LIKES) (*N NIL)) is created as follows: after the parser parsed ((*DET) THE) and ((*N) MAN), it meets ((*ADJ *ADV *N) HOME) and detects an error as described earlier. The parser then uses the four principles to create hypotheses, and after the parser processes ((*V) LIKES), the deleting principle survives. Since the next symbol in the input sequence is the end of the sequence and the parser detects another error. Further hypotheses are made and the one with *N finally survives. Back linking the *N with the one deleted (i.e., HOME), we actually could realize a limited version of the fifth principle, i.e., swapping two adjacent words, as mentioned before. A finer degree of relaxation related to the fifth principle can also be classified as below, although its usefulness may likely be finally decided through multiple knowledge interaction:

1. When the deleted category set (e.g., cat(HOME) in the current example) has no intersection with the inserted category set (e.g., (*N) in the current example), rank it as bad; the relaxation may not be allowed in this case;[7]

2. When the deleted category set has intersection with the inserted category set, but not inclusion, rank it as acceptable;

3. When the deleted category set is contained the inserted category set, rank it as good;

4. When the deleted category set contains the inserted category set, rank it as good;

The second hypothesized sequence is similar to the first one. The difference between the first hypothesized sequence and the second one is that in the first relaxation, instead of deleting principle, the inserting principle survives. All the rest of the two hypothesized sequences are more or less the same and we are not going to explain further.

```
Example sentence 4.

* (understander '(the man
                  home likes))
Part 1: the indexed sentence =
      ((0 (*DET) (THE 0 0))
       (1 (*N) (MAN 1 1))
       (2 (*ADJ *ADV *N) (HOME 2 2))
       (3 (*V) (LIKES 3 3)))
Part 2: the history of hypothesized
sentences:
  1-th hypothesized sentence:
  ((0 (*DET) (THE 0 0))
   (1 (*N) (MAN 1 1))
   (2 (*SPAN* *V *PREP *DET) (NIL -1 2))
   (3 (*SPAN* *ADJ *ADV *N) (HOME -1 3))
   (4 (*V) (LIKES 3 4))
   (5 ($) (NIL NIL 5)))
Part 3:
 and the value of the deviation
parameter is 1
Part 4:
 the grammar does not accept this sentence!
 here is a partial result
```

[7]If the deleted category set is empty or contains only unrecognized categories, the ranking may be different.

```
0   0
1   (*DET T (THE 0 0))
2   (*N T (MAN 1 1))
3   (NP (1 2))
4   (*PREP T (NIL -1 2))
5   (*V T (NIL -1 2))
6   (*SPAN* T (NIL -1 2))
7   (*N T (HOME -1 3))
8   (*SPAN* T (HOME -1 3))
9   (NP (7))
10   (PP (4 9))
11   (NP (3 10) (1 2))
12   (*V T (LIKES 3 4))
the end of the forest.
NIL
* (understander '(the man
                  home likes) 2)
Part 1: the indexed sentence =
     ((0 (*DET) (THE 0 0))
      (1 (*N) (MAN 1 1))
      (2 (*ADJ *ADV *N) (HOME 2 2))
      (3 (*V) (LIKES 3 3)))
Part 2: the history of hypothesized
sentences:
  1-th hypothesized sentence:
   ((0 (*DET) (THE 0 0))
    (1 (*N) (MAN 1 1))
    (2 (*SPAN* *V *PREP *DET) (NIL -1 2))
    (3 (*SPAN* *ADJ *ADV *N) (HOME -1 3))
    (4 (*V) (LIKES 3 4))
    (5 ($) (NIL NIL 5)))
  2-th hypothesized sentence:
   ((0 (*DET) (THE 0 0))
    (1 (*N) (MAN 1 1))
    (2 (*SPAN* *V *PREP *DET) (NIL -1 2))
    (3 (*SPAN* *ADJ *ADV *N) (HOME -1 3))
    (4 (*V) (LIKES 3 4))
    (5 (*SPAN* *V *PREP *N *DET)
       (NIL -1 5))
    (6 ($) (NIL NIL 6)))
Part 3:
 The following hypothesized sentences
get parsed:
(((*DET (THE 0 0)) (*N (MAN 1 1))
  (*V (LIKES 3 4)) (*N (NIL -1 5)))
 ((*DET (THE 0 0)) (*N (MAN 1 1))
  (*PREP (NIL -1 2)) (*N (HOME -1 3))
  (*V (LIKES 3 4)) (*N (NIL -1 5))))
 and the value of the deviation
parameter is 2
Part 4:
 it is accepted !
 the root of its parse forest is (19),
and the forest is:
0   0
1   (*DET T (THE 0 0))
2   (*N T (MAN 1 1))
3   (NP (1 2))
4   (*PREP T (NIL -1 2))
5   (*V T (NIL -1 2))
6   (*SPAN* T (NIL -1 2))
7   (*N T (HOME -1 3))
8   (*SPAN* T (HOME -1 3))
9   (NP (7))
10   (PP (4 9))
11   (NP (3 10) (1 2))
12   (*V T (LIKES 3 4))
13   (*DET T (NIL -1 5))
14   (*N T (NIL -1 5))
15   (*PREP T (NIL -1 5))
16   (*SPAN* T (NIL -1 5))
17   (NP (14))
18   (VP (12 17))
19   (S (3 18) (11 18))
the end of the forest.
NIL
*
```

5 Future Work and Conclusions

As we mentioned above, it is not always sufficient by syntactic knowledge alone to determine which k we should take in *xgp* phenomena, i.e. knowledge sources at other levels are needed in identifying which word(s) contribute to the inconsistency of the local context and/or multi-sentential (global) context. It is further discussed in (Weng, 1993) how to identify a proper k and narrow down the number of alternative interpretations while building up the syntactic-semantic structures, based on preference Semantics (Wilks, 1975; Fass — Wilks, 1983; Slater — Wilks, 1990).

It is quite interesting to contrast one of the conclusions made in (Carbonell — Hayes, 83), i.e. error correction in compiled version parsers is not flexible, with what we have done here. We are not claiming that our work completely invalidates that conclusion, but at least we see some promise in that direction. (Malone — Felshin, 1989; Moore — Dowding, 1991) also express the feeling that it is not easy to change GLR parsers to perform relaxation, while our modification is quite moderate although somewhat tricky.

(Aho — Peterson, 1973) describes an algorithm that parses any input string to completion finding the fewest possible number of errors, by changing the grammar. The algorithm does not require the locality principle. A similar effect can be realized by trying one relaxation for every word parsed when an error message is signalled.

The last point we would like to make concerns the relationship between the criteria for grammaticality and the deviation/relaxation parameter. The hints provided by our parser seem to suggest that grammaticality is a graded and individual-related notion, i.e. it depends not only on the individual's knowledge about language but also the ease of absorption of certain phenomena into her or his knowledge.

Acknowledgments

The author would like to express his thanks to Dr. Y. Wilks for his helpful comments on the content and presentation of this paper. Thanks also goes to the anonymous referees for valuable comments.

References

Aho A. — T. Peterson (1972) *A Minimum Distance Error-Correcting Parser for Context-Free Languages*, SIAM J. Computing, Vol.1, No.4.

Aho A. — J. Ullman (1977) *Principles of Compiler Design*, Reading (MA): Addison-Wesley.

——— (1986) *Compiler: Principles, Techniques, and Tools*, Reading (MA): Addison-Wesley.

Berwick R. (1985) *The Acquisition of Syntactic Knowledge*, Cambridge (MA): MIT Press.

Carbonell J. — P. Hayes (1983) *Recovery Strategies for Parsing Extragrammatical Language*, American Journal of Computational Linguistics, Vol 9 (3-4).

Chomsky N. (1980) *Lectures on Government and Binding*, Dordrecht: Foris Publications.

DARPA (1991) Workshop on Speech and Natural Language Workshop.

DARPA (1992) Workshop on Speech and Natural Language Workshop.

Fass D. — Y. Wilks (1983) *Preference Semantics, Ill-Formedness, and Metaphor*, American Journal of Computational Linguistics, Vol 9 (3-4).

Fong S. — R. Berwick (1989) *The Computational Implementation of Principle-Based Parsers*, 1st International Workshop on Parsing Technologies.

Gibson E. (1989) *Parsing with Principles: Predicting a Phrasal Node Before Its Head Appears*, 1st International Workshop on Parsing Technologies.

Hopcroft J. — J. Ullman (1979) Introduction to Automata Theory, Languages, and Computation, Reading (MA): Addison-Wesley.

Leung H. — D. Wotschke (manuscript) *Tradeoffs in Economy of Description in Parsing.*

Malone S. — Sue Felshin (1989) *An Efficient Method for Parsing Erroneous Input*, 1st International Workshop on Parsing Technologies.

McRoy S. — G. Hirst (1990) *Race-Based Parsing and Syntactic Disambiguation*, in Cognitive Science 14.

Moore R. — J. Dowding (1991) *Efficient Bottom-Up Parsing*, DARPA Speech and Natural Language Workshop.

Nyberg E. 3rd (1989) *Weight Propagation and Parameter Setting*, Ph.D Thesis Proposal, Department of Philosophy, CMU.

Passonneau R. et al., (1990) *Integrating Natural Language Processing and Knowledge Based Processing*, AAAI-90.

Piaget J. — B. Inhelder (1941) *Le Développement des Quantités chez L'enfant*, Neuchatel: Delachaux it Niestle. (A Translated Chinese Version).

Saito H. — M. Tomita (1991) *GLR Parsing for Noisy Input*, in *Generalized LR Parsing*, M. Tomita (ed.), Boston: Kluwer Academic Publishers.

Slater B. — Y. Wilks (1990) *PREMO: Parsing by conspicuous lexical consumption.*

Stallard D. — R. Bobrow (1992) *Fragment Processing in the DELPHI System*, DARPA Workshop on Speech and Natural Language Workshop.

Tomita M. (1984) *Disambiguation by Asking*, COLING-84.

——— (1985) *Efficient Parsing for Natural Language*, Kluwer Academic Publishers.

——— (1987) *An Efficient Augmented-Context-Free Parsing Algorithm.*, Computational Linguistics, 13, 31-46.

Weng F. (1993) *Handling Extra-Grammaticality at the Syntactic Level in Natural Language Processing*, Master Thesis, Computer Science Department and Computing Research Lab, New Mexico State University, Las Cruces, New Mexico, U.S.A.

Wilks Y. (1975) *A Preferential Pattern-Seeking Semantics for Natural Language Inference.* Artificial Intelligence 6: 53-74.

Table 1: A Set of Grammar Rules.

Rule-id	Rule form used in EGLR	CFG Rules (sorted)
0-th	(NP → (*DET *N))	NP → *DET *N
1-th	(NP → (*N))	NP → *N
2-th	(NP → (NP PP))	NP → NP PP
3-th	(PP → (*PREP NP))	PP → *PREP NP
4-th	(S → (NP VP))	S → NP VP
5-th	(S → (S PP))	S → S PP
6-th	(START → (S))	START → S
7-th	(VP → (*V NP))	VP → *V NP
8-th	(VP → (*V PP))	VP → *V PP

Table 2: The GAT and goto Table

the generalized action table:

state-id	*DET	*N	*PREP	*V	$
0	s3	s4			
1			s7	s8	
2			s7		a
3		s10			
4			r1	r1	r1
5			r4		r4
6			r2	r2	r2
7	s3	s4			
8	s3	s4	s7		
9			r5		r5
10			r0	r0	r0
11			r3 s7	r3	r3
12			r8		r8
13			r7 s7		r7

the goto table:

state-id	NP	PP	S	VP
0	1		2	
1		6		5
2		9		
3				
4				
5				
6				
7	11			
8	13	12		
9				
10				
11		6		
12				
13		6		

Table 3: The Extended GAT.

the extended generalized action table:

state id	*DET	*N	*PREP	*V	$	*SPAN*
0	s3	s4				s0
1			s7	s8		s1
2			s7		a	s2
3		s10				s3
4			r1	r1	r1	s4
5			r4		r4	s5
6			r2	r2	r2	s6
7	s3	s4				s7
8	s3	s4	s7			s8
9			r5		r5	s9
10			r0	r0	r0	s10
11			r3 s7	r3	r3	s11
12			r8		r8	s12
13			r7 s7		r7	s13

the goto table:

state id	NP	PP	S	VP
0	1		2	
1		6		5
2		9		
3				
4				
5				
6				
7	11			
8	13	12		
9				
10				
11		6		
12				
13		6		

Table 4: The Extended Grammar.

the extended grammar:

Rule-id	Rule form used in EGLR	CFG Rules (sorted)
0-th	(NP → (*DET *N))	NP → *DET *N
1-th	(NP → (*N))	NP → *N
2-th	(NP → (*VTMNP))	NP → *VTMNP
3-th	(NP → (NP PP))	NP → NP PP
4-th	(PP → (*PREP NP))	PP → *PREP NP
5-th	(PP → (*VTMPP))	PP → *VTMPP
6-th	(S → (*VTMS))	S → *VTMS
7-th	(S → (NP VP))	S → NP VP
8-th	(S → (S PP))	S → S PP
9-th	(START → (S))	START → S
10-th	(VP → (*V NP))	VP → *V NP
11-th	(VP → (*V PP))	VP → *V PP
12-th	(VP → (*VTMVP))	VP → *VTMVP

Table 5: The Extended GPTs.

the extended generalized action table:

stt-id	*DET	*N	*PREP	*V	$	*SPAN*	*VTMNP	*VTMPP	*VTMS	*VTMVP
0	s3	s4				s0	s5		s6	
1			s9	s11		s1		s10		s12
2			s9		a	s2		s10		
3		s14				s3				
4			r1	r1	r1	s4		r1		r1
5			r2	r2	r2	s5		r2		r2
6			r6		r6	s6		r6		
7			r7		r7	s7		r7		
8			r3	r3	r3	s8		r3		r3
9	s3	s4				s9	s5			
10			r5	r5	r5	s10		r5		r5
11	s3	s4	s9			s11	s5	s10		
12			r12		r12	s12		r12		
13			r8		r8	s13		r13		
14			r0	r0	r0	s14		r0		r0
15			r4 s9	r4	r4	s15		r4 s10		r4
16			r11		r11	s16		r11		
17			r10 s9		r10	s17		r10 s10		

the goto table:

state id	NP	PP	S	VP
0	1		2	
1		8		7
2		13		
3				
4				
5				
6				
7				
8				
9	15			
10				
11	17	16		
12				
13				
14				
15		8		
16				
17		8		