# Yet Another Chart-Based Technique for Parsing Ill-Formed Input

**Tsuneaki Kato**
NTT Information and Communication Systems Laboratories
1-2356 Take, Yokosuka-shi, Kanagawa, 238-03 JAPAN
kato@nttnly.ntt.jp

## Abstract

A new chart-based technique for parsing ill-formed input is proposed. This can process sentences with unknown/misspelled words, omitted words or extraneous words. This generalized parsing strategy is, similar to Mellish's, based on an active chart parser, and shares the many advantages of Mellish's technique. It is based on pure syntactic knowledge, it is independent of all grammars, and it does not slow down the original parsing operation if there is no ill-formedness. However, unlike Mellish's technique, it doesn't employ any complicated heuristic parameters. There are two key points. First, instead of using a unified or interleaved process for finding errors and correcting them, we separate the initial error detection stage from the other stages and adopt a version of bi-directional parsing. This effectively prunes the search space. Second, it employs normal top-down parsing, in which each parsing state reflects the global context, instead of top-down chart parsing. This enables the technique to determine the global plausibility of candidates easily, based on an admissible A* search. The proposed strategy could enumerate all possible minimal-penalty solutions in just 4 times the time taken to parse the correct sentences.

## 1 Introduction

It is important that natural language interface systems have the capability of composing the globally most plausible explanation if a given input can not be syntactically parsed. This would be useful for handling erroneous inputs from the user and for offsetting grammar and lexicon insufficiency. Also, such a capability could be applied to deal with the ungrammatical sentences and sentence fragments that frequently appear in spoken dialogs (Bear, Dowding and Shriberg, 1992). Several efforts have been conducted to achieve this objective ((Lang, 1988; Saito and Tomita, 1988), for example.) One major decision to be made in designing this capability is whether knowledge other than purely syntactic knowledge is to be used. Other-than syntactic knowledge includes grammar specific recovery rules such as meta-rules (Weishedel and Sondheimer, 1983), semantic or pragmatic knowledge which may depend on a particular domain (Carbonell and Hayes, 1983) or the characteristics of the ill-formed utterances observed in human discourse (Hindle, 1983). Although it is obvious that the utilizing such knowledge allows us to devise more powerful strategies, we should first determine the effectiveness of using only syntactic knowledge. Moreover, the result can be applied widely, as using syntactic knowledge is a base of the most of strategies.

One significant advance in the usage of syntactic knowledge was contained in the technique proposed by Mellish (1989). It can handle not only unknown/misspelled words, but also omitted words and extraneous words in sentences. It can deal with such problems, and develop plausible explanations quickly since it utilizes the full syntactic context by using an active chart parser (Kay, 1980; Gazdar and Mellish, 1989). One problem with his technique is that its performance heavily depends on how the search heuristics, which is implemented as a score calculated from six parameters, is set. The heuristics complicates the algorithm significantly. This must be one of reasons why the performance of the method, as Mellish himself noted, dropped dramatically when the input contains multiple errors.

This paper proposes a new technique for parsing inputs that contain simple kinds of ill-formedness. This generalized parsing strategy is, similar to Mellish's, based on an active chart parser, and so shares the many advantages of Mellish's technique. It is based on pure syntactics, it is independent of all grammars, and it does not slow down the original parsing operation if there is no ill-formedness. However, unlike Mellish's technique, it doesn't employ any complicated heuristic parameters. There are two key points. First, instead of using a unified or interleaved process for finding errors and correcting them, we separate the initial error detection stage from the other stages and adopt a version of bi-directional parsing, which has been pointed out to be a useful strategy for fragment parsing by itself (Satta and Stock, 1989). This effectively prunes the search space and allows the new technique to take full account of the right-side context. Second, it employs normal top-down parsing, in which each parsing state reflects the global context, instead of top-down chart parsing. This enables the technique to determine the global plausibility of candidates easily. The results of preliminary experiments are encouraging. The proposed strategy could enumerate

all possible minimal-penalty solutions in just 4 times the time taken to parse the correct sentences. That is, it is almost twice as fast as Mellish's strategy.

## 2 Mellish's Technique And Its Problems

The basic strategy of Mellish's technique is to run a bottom-up parser over the input and then, if this fails to find a complete parse, to run a generalized top-down parser over the resulting chart to hypothesize complete parse candidates. When the input is well-formed, the bottom-up parser, precisely speaking, a left corner parser without top-down filtering, would generate the parse without any overhead. Even if it failed, it is guaranteed to find all complete constituents of all possible parses. Reference to these constituents, enables us to avoid repeating existing works and to exploit the full syntactic context rather just the left-side context of error candidates. The generalized top-down parser attempts to find out minimal errors by refining the set of "needs" that originates with bottom-up parsing. Each need indicates the absence of an expected constituent. The generalized parser hypothesizes, and so remedies an error, when it was sufficiently focused on. Next, the parser tries to construct a complete parse by taking account of the hypothesis. In the case of multiple errors, the location and recovery phases are repeated until a complete parse is obtained.

The data structure introduced for representing information about local needs is called the generalized edge. It is an extension of active and inactive edges, and is described as

$$< C \text{ from } S \text{ to } E \text{ needs } cs_1 \text{ from } s_1 \text{ to } e_1, \quad cs_2 \text{ from } s_2 \text{ to } e_2, \ldots, cs_n \text{ from } s_n \text{ to } e_n >$$

where C is category, $cs_i$ are sequences of categories (which will be shown inside square brackets), S, E, $s_i$, and $e_i$ are positions in the input. The special symbol "*" denotes the position that remains to be determined. The presence of an edge of this kind in the chart indicates that the parser is attempting to find a phrase of category C that covers the input from position S to E but that in order to succeed it must still satisfy all the needs listed. Each need satisfies a sequence of categories $cs_i$ that must be found contiguously to occupy the portion from $s_i$ to $e_i$. An edge with an empty need, which corresponds to an inactive edge is represented as

$$< C \text{ from } S \text{ to } E \text{ needs nothing}>.$$

The generalized top-down parser that uses the generalized edge as the data structure is governed by six rules: three for finding out errors and the other three for recovering from the three kinds of error. The three error locating rules are the top-down rule, the fundamental rule and the simplification rule. The first one is also used in the ordinary top-down chart parser, and the third one is just for house keeping. The second rule, the fundamental rule, directs to combine an active edge with a inactive edge. It was extended from the ordinary rule so that found constituents could be incorporated from either direction. However, the constituents that can be absorbed are limited to those in the first category sequence; that is, one of the categories belonging to $cs_1$. The application of the six rules is mainly controlled by the scores given to edges, that is, agenda control is employed. The score of a particular edge reflects its global plausibility and is calculated from six parameters, one of which, for example, says that edges that arise from the fundamental rule are preferable to those that arise from the top-down rule.

Although Mellish's technique has a lot of advantages such as the ability to utilize the right-side context of errors and independence of a specific grammar, it can create a huge number of edges, as it mainly uses the top-down rule for finding errors. That is, refining a set of error candidates toward a pre-terminal category by applying only the top-down rule may create too many alternatives. In addition, since the generalized edges represent just local needs and don't reflect the global needs that created them, it is hard to decide if they should be expanded. In particular, these problems become critical when parsing ill-formed inputs, since the top-down rule may be applied without any anchoring; pre-terminals can not be considered as anchors, as pre-terminals may be freely created by error recovery rules. This argument also applies to the start symbol, as that symbol may be created depending the constituent hypothesized by error recovery rules and the fundamental rule. Mellish uses agenda control to prevent the generation of potentially useless edges. For this purpose, the agenda control needs complicated heuristic scoring, which complicates the whole algorithm. Moreover, so that the scoring reflects global plausibility, it must employs a sort of dependency analysis, a mechanism for the propagation of changes and an easily reordered agenda, which clearly contradicts his original idea in which edges must be reflected only local needs.

## 3 Proposed Algorithm

The technique proposed here resolves the above problems as follows. First, some portion of the error location process is separated from and precedes the processes that are governed by agenda control, and is archived by using a version of bi-directional parsing. Second, so that the search process can be anchored by the start symbol, a data structure is created that can represent global plausibility. Third, in order to reduce the dependency on the top-down rule, a rule is developed that uses two active edges to locate errors. This process is closer to ordinary top-down parsing than chart parsing and global plausibility scoring is accurate and easily calculated. For simplicity of explanation, simple CF-PSG grammar formalism is assumed throughout this paper, although there are obvious generalizations to other formalism such as DCG (Pereira and Warren, 1980) or unification based grammars (Shieber, 1986).

**Bottom-up rule:**

<C from S to E needs nothing>

$C1 \rightarrow ...Cs_1\ C\ Cs_2...$ where $Cs_1$ is not empty (in the grammar)

---

<$C_1$ from * to $E_2$ needs $Cs_1$ from * to S , $Cs_2$ from E to $E_2$>

where if $Cs_2$ is empty then E2 = E else $E_2$=*.

**Fundamental rule:**

<C from S to E needs ... , [...Cs11, C1, Cs12...] from $s_1$ to $e_1$ , ...>

<$C_1$ from $S_1$ to $E_1$ needs nothing>

---

<C from S to E needs ... , $Cs_{11}$ from $s_1$ to $S_1$ ,$Cs_{12}$ from $E_1$ to $e_1$,...>

where $s_1 \leq S_1$ or $s_1$=*, $E_1 \leq e_1$ or $e_1$ =*.

**Simplification rule:**

<C from S to E needs

..., $Cs_{i-1}$ from $s_{i-1}$ to s, [] from s to s, $Cs_{i+1}$ from s to $e_{i+1}$, ...>

---

<C from S to E needs ..., $Cs_{i-1}$ from $s_{i-1}$ to s, $Cs_{i+1}$ from s to $e_{i+1}$, ...>

**Figure 1. The Bi-Directional Parsing Rules**

The first phase of the process is invoked after the failure of left corner parsing. The bottom-up parsing leaves behind all complete constituents of every possible parse and unsatisfied active edges for all error points that are to the immediate right of sequences of constituents corresponding to the RHS. Since parsing proceeds left to right, an active edge is generated only when an error point exists to the right of the found constituents. In the first phase, bi-directional bottom-up parsing generates all generalized edges that represent unsatisfied expectations to the right and left of constituents. From some perspectives, the role this phase plays is similar to that of the covered bi-directional phase of the Picky parser (Magerman and Weir, 1992), though the method proposed herein does not employ stochastic information at all. This process can be described in three rules as shown in Figure 1. As can be seen, this is bi-directional bottom-up parsing that uses generalized edges as the data structure. For simplicity, the details for avoiding duplicated edge generation have been omitted. It is worth noting that after this process, the needs listed in each generalized edge indicate that the expected constituents did not exist, while, before this process, a need may exist just because an expectation has not been checked.

The second phase finds out errors and corrects them. The location operation proceeds by refining a need into more precise one, and it starts from the global need that refers to the start symbol, S, from 0 to *n*, where *n* is the length of the given input. In the notion of generalized edges, that need can be represented as,

<GOAL from 0 to n needs [S] from 0 to *n*> .

The data structure reflecting global needs directly is used in this phase, so the left part of each generalized edge is redundant and can be omitted. In addition, two values, *g* and *h*, are introduced. *g* denotes how much cost has been expended for the recovery so far, and *h* is the estimation of how much cost will be needed to reach a solution. Cost involves solution plausibility; solutions with low plausibility have high costs. Thus, the data structure used in this phase is,

<needs $cs_1$ from $s_1$ to $e_1$, $cs_2$ from $s_2$ to $e_2$, ... ,

$cs_n$ from $s_n$ to $e_n$, *g*, *h*> .

Here, the number of errors corrected so far is taken as *g*, and the total number of categories in the needs is used as *h*. As mentioned above, since the needs listed indicate only the existence of errors as detected by the preceding process and to be refined, the value of *h* is always less than or equal to the number of the errors that must be corrected to get a solution. That is, the best first search using *g+h* as the cost functions is an admissible A* search (Rich and Knight, 1991). Needless to say, more sophisticated cost functions can also be used, in which, for example, the cost depends on the kind of error.

The rules governing the second phase, which correspond to the search state transition operators in the context of search problems, are shown in Figure 2. The top-down rule and the refining rule locate errors and the other three rules are for correcting them. Most important is the refining rule, which tries to find out errors by using generalized edges in a top-down manner toward pre-terminals. This reduces the frequency of using the top-down rule and prevents an explosion in the number of alternatives.

This process starts from

<needs [S] from 0 to n, g: 0, h: 1> .

To reach the following need means to get one solution.

<needs nothing, g: _, h: 0>.

**Top-down rule:**

<needs [$C_1$...$Cs_1$] from $s_1$ to $E_1$, ..., g: G, h: H>

$C_1 \rightarrow$ ...RHS (in grammar)

---

<needs [...RHS ...$Cs_1$] form $s_1$ to $E_1$ , ..., g: G, h: H+(length of RHS)-1>

**Refining rule:**

<needs [...$Cs_{11}$, $C_1$, $Cs_{12}$...] from $s_1$ to $e_1$ , ... , g: G, h: H>

<$C_1$ from S to E needs $Cs_1$ from $S_1$ to $E_1$ , ... , $Cs_n$ from $S_n$ to $E_n$ >

---

<needs $Cs_{11}$ from $s_1$ to S , $Cs_1$ from $S_1$ to $E_1$ , ... , $Cs_n$ from $S_n$ to $E_n$ ,
$Cs_{12}$ from E to e1 , ..., g: G, h: H+$\Sigma$(length of $Cs_n$)-1>

The result must be well-formed, that is s1 $\leq$ S1 or s1=* or S1=* and so on.

**Garbage rule:**

<needs [$C_1$...$Cs_1$] from $s_1$ to $e_1$ , ..., g: G, h: H> where $C_1$ is a pre-terminal

<$C_1$ from $S_1$ to $E_1$ needs nothing> where $s_1 \leq S_1$

---

<needs $Cs_1$ from $E_1$ to $e_1$ , ..., g: G+($S_1$-$s_1$), h: H-1>

**Unknown word rule:**

<needs [$C_1$...$Cs_1$] from $s_1$ to $e_1$ , ..., g: G, h: H> where $C_1$ is a pre-terminal

---

<needs $Cs_1$ from $s_1$+1 to $e_1$, ..., g: G+1, h: H-1>

where the edge, <$C_1$ from $s_1$ to $s_1$+1 needs nothing> does not exist in the chart

**Empty category rule:**

<needs $Cs_1$ from s to s , $Cs_2$ from $s_2$ to $e_2$ , ..., g: G, h: H>

---

<needs $Cs_2$ from $s_2$ to $e_2$ , ..., g: G+(length of $Cs_1$), h: H-(length of $Cs_1$)>

**Figure 2. The Error Locating and Recovery Rules**

The need with the smallest value of $g+h$ is processed first. If two needs have the same value of $g+h$, the one with the smaller $h$ values dominates. This control strategy guarantees to find the solution with minimal cost first; that is, the solution with the minimum number of recoveries.

Figure 3 shows an example of this technique in operation. (a) shows the sample grammar adopted, (b) shows the input to be processed, and (c) shows some of the edges left behind after the failure of the original bottom-up parsing. As shown in (d), the first phase generates several edges that indicate unsatisfied expectations to the left of found constituents. The second phase begins with need (e-1). Among the others, (e-2) and (e-3) are realized by applying the refining rule and the top-down rule, respectively. Since (e-2) has the smallest value of $g+h$, it takes precedence to be expanded. The refining rule processes (e-2) and generates (e-4) and (e-7), among others. The solution indicated by (e-6), which says that the fifth word of the input must be a preposition, is generated from (e-4). Another solution indicated by (e-9), which says that the fifth word of the input must be a conjunctive is derived from (e-7). That the top-down rule played no role in this example was not incidental. In reality, application of the top-down rule may be meaningful only when all the constituents listed in the RHS of a grammar rule contain errors. In every other case, generalized edges derived from that rule must have been generated already by the first phase. The application of the top-down rule can be restricted to cases involving unary rules, if one assumes at most one error may exist.

## 4 Preliminary Experiments

In order to evaluate the technique described above, some preliminary experiments were conducted. The experiments employed the same framework as used by Mellish, and used a similar sized grammar, the small e-free CF-PSG for a fragment of English with 141 rules and 72 categories. Random sentences (10 for each length considered) were generated from the grammar, and then random occurrences of specific types of errors were introduced into these sentences. The errors considered were none, deletion of one word, adding one known or unknown word, and substituting one unknown or known word for one word of the sentence. The amount of work done by the parser was calculated using the concept of "cycle". The parser consumes one cycle for processing each edge. The results are shown in Table 1. The

**(a) The grammar:**

| | | |
|---|---|---|
| S → NP VP ...(a-1) | NP → N | NP → Det N |
| NP → NP C NP | VP → Vt NP | VP → Vi |
| VP → VP PP | PP → P NP ...(a-2) | |

**(b) The input:**

| The | lady | bought | cakes | an | the | shop |
|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

**(c) Examples of the edges left behind:**

<NP from 0 to 2 needs nothing>
<Vi from 2 to 3 needs nothing>
<NP from 3 to 4 needs nothing>
<NP from 5 to 7 needs nothing> ...(c-1)
<S from 0 to * needs [VP] from 2 to *> ...(c-2)
<VP from 2 to * needs [PP] from 4 to *> ...(c-3)
<VP from 2 to * needs [NP] from 3 to *> ...(c-4)
<NP from 3 to * needs [C NP] from 4 to *> ...(c-5)

**(d) Examples edges generated in the bi-directional parsing:**

<PP from * to 7 needs [P] from * to 5> ...(d-1) Bottom up rule, (c-1), (a-2)
<NP from 3 to 7 needs [C] from 4 to 5> ...(d-2) Fundamental rule, (c-1), (c-5)

**(e) Focusing on and recovering from errors:**

<needs [S] from 0 to 7, g:0, h:1> ...(e-1) Initial needs
<needs [VP] from 2 to 7, g:0, h:1> ...(e-2) Refining rule, (e-1), (c-2)
<needs [NP VP] from 0 to 7, g:0, h:2> ...(e-3) Top-down rule, (e-1), (a-1)

<needs [PP] from 4 to 7, g:0, h:1> ...(e-4) Refining rule, (e-2), (c-3)
<needs [P] from 4 to 5, g:0, h:1> ...(e-5) Refining rule, (e-4), (d-1)
<needs nothing, g:1, h:0> ...(e-6) Unknown word rule, (e-5)
The fifth word, "an", is hypothesized to be an unknown preposition (P)

<needs [NP] from 3 to 7, g:0, h:1> ...(e-7) Refining rule, (e-2), (c-4)
<needs [C] from 4 to 5, g:0, h:1> ...(e-8) Refining rule, (e-7), (d-2)
<needs nothing, g:1, h:0> ...(e-9) Unknown word rule, (e-8)
The fifth word, "an", hypothesized to be an unknown conjunctive (C)

**Figure 3. An Example of the Error Recovery Process**

statistics in the table are described as follows. *BU cycles* is the number of cycles taken to exhaust the chart in the initial bottom-up parsing. *BD cycles* is the number of cycles required for bi-directional bottom-up parsing in the first phase. *#solns* is the number of different solutions and represents descriptions of possible errors. *First/Last* is the number of cycles required for error location and recovery to find the first / last solution. *LR cycles* is the number of cycles in the error locating and recovery phase required to exhaust all possibilities of sets of errors with the same penalty as the first solution.

The preliminary results show that, for short sentences with one error, enumerating all possible minimum-penalty errors takes about 4 times as long as parsing the correct sentences. This is almost twice the speed of Mellish's strategy. As 75% of the process are occupied by the first bi-directional parsing operation, more cycles are needed to get the first solution with the proposed technique than with Mellish's strategy.

## 5 Discussion

The second phase of the proposed technique is based on ordinary top-down parsing or tree search rather than chart parsing. As a consequence, some error location operations may be redundant, as Mellish pointed out. For example, suppose a new grammar rule, N → N PP is added to the grammar given in Figure 3. In that case, the following edge, located in the first phase, may cause a redundant error locating process, as the same search is triggered by (e-4).

<N from 3 to * needs [PP] from 4 to *>.

One way for avoiding such redundancies is to use a data structure that reflects just local needs. However, it is true that an effective error location process must take into account global needs. There is a tradeoff between simplicity and the avoidance of duplicated efforts. The technique proposed here employs a data structure that

**Table 1. Preliminary Experimental Results**

| Error | Length of original | BU cycles | BD cycles | #solns | First | Last | LR cycles |
|---|---|---|---|---|---|---|---|
| None | 6 | 70 | | 1.3 | | | |
| | 9 | 114 | | 1.4 | | | |
| | 12 | 170 | | 2.0 | | | |
| Delete one word | 6 | 42 | 132 | 6.0 | 8 | 24 | 31 |
| | 9 | 79 | 255 | 4.5 | 19 | 32 | 43 |
| | 12 | 111 | 378 | 6.2 | 25 | 43 | 57 |
| Add unknown word | 6 | 60 | 191 | 3.0 | 14 | 20 | 28 |
| | 9 | 99 | 322 | 3.7 | 25 | 37 | 46 |
| | 12 | 147 | 534 | 2.6 | 46 | 61 | 72 |
| Add known word | 6 | 69 | 221 | 5.7 | 14 | 25 | 33 |
| | 9 | 94 | 292 | 4.7 | 24 | 38 | 55 |
| | 12 | 159 | 578 | 6.9 | 51 | 70 | 80 |
| Substitute unknown word | 6 | 50 | 153 | 2.7 | 9 | 14 | 20 |
| | 9 | 76 | 239 | 4.2 | 21 | 34 | 43 |
| | 12 | 109 | 363 | 3.2 | 34 | 46 | 58 |
| Substitute known word | 6 | 51 | 151 | 3.8 | 9 | 18 | 27 |
| | 9 | 82 | 256 | 3.2 | 15 | 25 | 33 |
| | 12 | 116 | 384 | 3.8 | 33 | 58 | 71 |

directly reflects the global needs. Mellish, on the other hand, utilized a structure that reflected just local needs and tried to put global needs into the heuristic function. The result, at least so far as confirmed by tests, was that pruning allowed the simple method to overcome the drawback of duplicated effort. Moreover, Mellish's dependency control mechanism, introduced to maintain the plausibility scores, means that edges are no longer local. In addition, it can be expected that a standard graph search strategy for avoiding duplicated search is applicable to the technique proposed.

Theoretical investigation is needed to confirm how the number of grammar rules and the length of input will affect the amount of computation needed. Furthermore, the algorithm has to be extended in order to incorporate the high level knowledge that comes from semantics and pragmatics. Stochastic information such as statistics on category trigrams must be useful for effective control.

## References


Bear, John, Dowding, John and Shriberg, Elizabeth (1992). Integrating Multiple Knowledge Sources for Detection and Correction of Repairs in Human-Computer Dialog. *Proceedings of 30th ACL*, 56 - 63.

Carbonell, Jaime G. and Hayes, Philip J. (1983). Recovery Strategies for Parsing Extra grammatical Language. *JACL*, 9 (3-4), 123 - 146.

Gazdar, Gerald and Mellish, Chris (1989). *Natural Language Processing in LISP*. Workingham: Addison-Wesley.

Hindle, Donald (1983). Deterministic Parsing of Syntactic Non-fluencies. *Proceedings of 21st ACL*, 123 - 128.

Kay, Martin (1980). Algorithm Schemata and Data Structures in Syntactic Processing. *Research Report* CSL-80-12 Xerox PARC.

Lang, Bernard (1988). Parsing Incomplete Sentences. *Proceedings of COLING 88*, 365 - 371.

Magerman, David M. and Weir, Carl (1992). Efficiency, Robustness and Accuracy in Picky Chart Parsing. *Proceedings of 30th ACL*, 40 - 47.

Mellish, Chris S. (1989). Some Chart-Based Techniques for Parsing Ill-Formed Input. *Proceedings of 27th ACL*, 102 - 109.

Pereira, Fernando C.N. and Warren, David, H.D. (1980). Definite Clause Grammars for Language Analysis - A Survey of the Formalism and a Comparison with Augmented Transition Networks. *Artificial Intelligence* , 13 (3), 231 - 278.

Rich, Elaine and Knight, Kevin (1991). *Artificial Intelligence* (2nd ed.). New York: McGraw-Hill.

Saito, Hiroaki and Tomita, Masaru (1988). Parsing Noisy Sentences. *Proceedings of COLING 88*, 561 - 566.

Satta, Giorgio and Stock, Oliviero (1989). Formal Properties and Implementation of Bidirectional Charts. *Proceedings of IJCAI 89*, 1480 - 1485.

Shieber, Stuart M. (1986). *An Introduction to Unification-Based Approaches to Grammar*. Stanford: CSLI Lecture Notes 4.

Weishedel, Ralph M. and Sondheimer, Norman K. (1983). Meta-Rules as a Basis for Processing Ill-Formed Input. *JACL*, 9 (3-4), 161 - 177.