CHART PARSER FOR ILL-FORMED INPUT SENTENCES

Kyongho Min

School of Computer Science & Engineering
The University of New South Wales

My research is based on the parser for ill-formed input by Mellish in a paper in ACL 27th meeting Proceedings, 1989. My system is composed of two parsers: WFCP and IFCP. When WFCP fails to give the parse tree for the input sentence, the sentence is identified as ill-formed and is parsed by IFCP for error detection and recovery at the syntactic level. My system is independent of grammatical rules. It does not take into account semantic ill-formedness. My system uses a grammar composed of 25 context-free rules.

My system consists of two major parsing strategies: top-down expectation and bottom-up satisfaction. With top-down expectation, rules are retrieved under the inference condition and expanded by inactive arcs. When doing bottom-up parsing, my parser used two modes: Left-to-Right parsing and Right-to-Left parsing.
My system repairs errors successfully when the input contains an omitted word or an unknown word substituted for a valid word. Left-corner and right-corner errors are more easily detected and repaired than ill-formed sentences where the error is in the middle. The deviance note, with repair details, is kept in new inactive arcs which are generated by the error correction procedure.

The implementation of my system is quite different from Mellish's. When rules are invoked, my system invokes all rules with minimal inference. My bottom up parsing strategy uses Left-to-Right mode and Right-to-Left mode. My system is bottom-up-parsing-oriented like the chart parser. Errors are repaired in two ways: using top-down hypothesis, and using Need-Chart which keeps the information of expectation and completion of expanded goals by rules. To reduce the number of top-down cycles, all rules are invoked simultaneously and this invocation information is kept in Need-Chart. This idea will be extended for the implementation of multiple error recovery system.

1. INTRODUCTION

Computer Natural Language Processing (NLP) is divided into Two major areas, Natural Language Generation (NLG) and Natural Language Understanding (NLU). Then what is Natural Language? Language is a medium of communication between a speaker (a producer) and a hearer (a receiver) and is a system composed of three components including notion (meaning or semantics), function (syntactic structure description), and form (spoken and written form) (Jespersen, 1965). In terms of these concepts, NLG is a process of converting 'notion' to 'form' by a computer.
'Function' is a bridge between 'notion' and 'form' and describes the syntactic structure of sentences which are generated by grammatical rules. These rules in language are a filter to identify which sentences are syntactically grammatical. NLU systems treat sentences as input to be parsed by a computer program called a parser, form a structural description of sentences, and grasp their semantic meaning. To do that, the parser needs to filter ungrammatical sentences at both the syntatic and semantic levels.

We, as native speakers of a particular language, can produce both well-formed and ill-formed sentences. In terms of an experimental result (Eastman & McLean, 1981), errors including cooccurrence violations, ellipsis and extraneous terms, and conjunction, were found in 33% of NL database queries. In the case of sentences which are ill-formed because of unknown words, violations of meaningful relation, ellipsis, or metaphor etc., we can often identify the type of error, repair the ill-formed sentences, and finally understand those kinds of ill-formed sentences even though they are made at the lexical level, at the syntatic level, at the semantic level, and at the pragmatic level. In the case of a computer parser, ill-formed sentences can be handled in two ways. One is to reject them. The other is to accept them after minor changes to make them acceptable. The second system is better than the first one, because it is more human-like and more flexible in parsing sentences.
My project is based on an article by Mellish (1989). This article focused on the recovery of ill-formed input sentences at the syntactic level using a chart-based parser. His system is composed of two parsers: a well-formed input parser and an ill-formed input parser. My project follows his ideas. This report will cover:

- the concept of chart parsing;
- parsing strategies for ill-formed input;
- design and implementation of my system;
- comparison of Mellish's system with my system;
- results and their analysis;
- problems;
- conclusion.

2. CHART PARSER

2.1 General Review of Chart Parser

When a parser runs, there are two alternative parsing strategies like other problem solving systems: top-down parsing and bottom-up parsing. The top-down parsing hypothesises the final goal as a constituent S (sentence). Grammar rules are repeatedly invoked to expand the goal of a S-structure until RHS (right-hand side) of invoked rules
matches the lexical categories of the word in an input sentence. Bottom-up parsing starts by identifying the lexical categories of word in an input sentence. These categories lead to the invocation of rules and a parser generates appropriate active arcs until all well-formed substrings are produced including a S (complete sentence parse tree) node. In rule invocation strategies for context-free grammar, a bottom-up left-corner parsing strategy is suggested as the most efficient method (Wiren 1987).

Natural language syntax can largely be described using context-free grammars. With Earley’s algorithm (Earley 1970), a chart parser was firstly implemented by Kay and Kaplan (Kay 1973 & Kaplan 1973). The basic idea of chart parser is different from an ATN parser, which was an efficient and powerful natural language parser. In ATN parsers (Bate 1978), parts of input sentences can be re-parsed many times before a correct description of the syntactic (tree) structure of the input sentence is formed. If a chosen ATN arc does not lead to a complete parse, the ATN parser backtracks to a choice-point and resumes the parsing process. After backtracking, it is likely to be necessary to re-parse some parts of the sentence in a fashion identical to the previous parsing track.

On the other hand, chart parsers find all WFS (well-formed substrings) exhaustively whether their syntactic structures fit in to a complete parse or not. In chart parsers, the WFS which
are found are kept as *inactive arcs*. Therefore, there is no need to parse words and phrases in sentences repeatedly to produce a final syntactic parse tree. This is the key idea of chart parser: to record the constituents found by a parser in a table—the well-formed substring table (WFST). Many chart parser systems have been implemented with different grammatical descriptions.

MCHART is based on an agenda mechanism and carefully structured interfaces for the specification of scheduling and grammatical formalism (Thompson, 1983). MCHART had also been implemented using Phrase Structure Grammar (Thompson, 1981). In parsing linear sentences, a breadth-first parsing model was suggested as the best choice for disambiguation and attachment (Bear, 1983). A chart parser is usually a left-to-right parser. But the left-to-right mode is not effective enough to recognise spoken languages for many reasons, including noise words, unidentified words, etc. So a bidirectional chart parsing technique was suggested by Stock, Falcone, & Insinnamo (1988). In their system active arcs are begun at a distinctive input word (e.g. a noun) and expanded both left and right. To speed up parsing time, a parallel chart parser for a shared memory multi processor has been implemented (Grishman & Chitrao, 1988).

Other approaches include bottom-up left-to-right parser with combinatory categorial grammar to cope with spurious syntactic ambiguity (Pareschi & Steedman, 1987). GLP
(General Linguistic Processor) for understanding and generation of natural language in the speech understanding system (Goerz, 1981), and UCP(Uppsala Chart Parser) which was composed of a chart, process primitives, a network grammar and a scanning algorithm (Hein, 1987).

2.2 Chart Parsing Algorithm

The following algorithm expresses an efficient bottom-up parsing process. It is guaranteed to parse a sentence of length $N$ within the time $O(N^3)$ (the constant of proportionality depends on grammar size, inter alias) and it will perform better than this ($O(N^2)$ or $O(N)$) with well-behaved grammars.

The algorithm constructs (phrasal or lexical) constituents of a sentence. We shall use the sentence *the good boy likes an apple* as an example in describing a NL-oriented bottom-up parser. The sentence is annotated with positions: our sentence become $0^\text{the} 1^\text{good} 2^\text{boy} 3^\text{likes} 4^{an} 5^\text{apple} 6$. In terms of this notation, the parsing process succeeds if an S(sentence) constituent is found covering positions 0 to 6. Points (0) to (8) below do not completely specify the order in which parsing steps are carried out: one reasonable order is to scan a word (as in (2)) and then perform all possible parsing steps as specified in (3)-(6) before scanning another word. Parsing is complete when the
last word has been read and all possible subsequent parsing steps have been performed. **Parser inputs**: a sentence, a lexicon, and a grammar.

(0) The algorithm creates and operates on two data structures: the **active chart** – a collection of active arcs (see (3) below) and the **constituents** (see (2) and (5)). Both are initially empty.

(1) The grammar is considered to include lexical insertion rules: for example, if *likes* is a word in the lexicon being used, and if its lexical entry includes the fact that *likes* may be a NOUN or a VERB, then rules of the form NOUN→*likes* and VERB→*likes* are considered part of the grammar.

(2) As a word (e.g. *likes*) is scanned, constituents corresponding to its lexical categories are created: NOUNI::NOUN→*likes* From 3 To 4, and VERBI→*likes* From 3 To 4.

(3) If the grammar contains a rule like NP→DET ADJ NOUN, and there is a constituent like DET1::DET→the From i To j, then an active arc ARCI::DET1·ADJ NOUN From i To j is added to the active chart. (In our example sentence, i would be 0 and j would be 1.) The "·" in an active arc marks the boundary between constituents (in this case just DET1) which have been found and constituents which have not (yet) been found.

(4) **Advancing** the "·": If the active chart contains an active arc like ARCI→DET1·ADJ NOUN From i To j and
there is a constituent in the chart of type NOUN (i.e. the first item after the \cdot\), say $\text{ADJ1:ADJ} \rightarrow \text{good ADJ From j To k}$ (note the From position in the constituent matches the To position in the active arc) then the \cdot can be advanced, creating a new active arc: $\text{ARC2:NP} \rightarrow \text{DET1 ADJ1 · NOUN From i To k}$.

\begin{itemize}
  \item [5] If the process of advancing the \cdot creates an active arc whose \cdot is at the far right hand side of the rule: e.g. $\text{ARC3:NP} \rightarrow \text{DET1 ADJ1 NOUN1 · From 0 To 3}$ then this arc is converted to a constituent: $\text{NP1:NP} \rightarrow \text{DET1 ADJ1 NOUN1 From 0 To 3}$. Not all active arcs are ever completed in this sense: for example, if the second word of the sentence had not been an adjective, then the \cdot in ARC1 could never have been advanced.

  \item [6] Both lexical and phrasal constituents can be used in steps 3 and 4: thus for example if the grammar contains a rule $\text{S} \rightarrow \text{NP VP}$, then as soon as the constituent NP1 discussed in step 5 is created, it will be possible to make a new active arc $\text{ARC4:S} \rightarrow \text{NP1 · VP From to 3}$.

  \item [7] Names: When subsequent constituents are created, they would have names like NP2, NP3, ADJ2, ADJ3... and ultimately, perhaps, S1, S2, ...

  \item [8] As stated before, the object of parsing is to create phrasal
constituents (normally of type S) whose \textbf{From} is 0 and whose \textbf{To} is the length of sentence." There may be several such constituents. It is possible to have type-S constituents which do not cover the whole sentence.

3. STRATEGIES FOR PARSING ILL-FORMED LANGUAGE

We can easily produce ungrammatical sentences at various levels of grammar: the lexical level (phonological/graphemic level, morphological level), the syntactical level, the semantic level, and the pragmatic level. Some sentences are well-formed at the syntactic level, but ill-formed at the semantic level, and finally well-formed at the pragmatic level. In practical terms, such sentences are acceptable.\textsuperscript{a}

It is difficult to parse a sentence at all levels simultaneously. An incremental parsing strategy from syntax to pragmaics was suggested by Gazdar & Mellish(1989). In the case of ill-formed sentences, it is not easy to combine the output of the recovering process at each level to get the final meaning of sentence at the pragmatic level. Each level has some advantages in repairing errors. There have been many attempts to recover ill-formed sentences at various levels by Carbonell & Hayes(1983), Fass & Wilks(1983), Mellish(1989), Fink & Biermann(1986), Granger(1983), Grishman &

3.1 Chart-Based Parser (Mellish, 1989)

Mellish's parser is based on a chart parser. He set out to implement a parser, which did not take into account semantics and pragmatics, and was independent of any particular grammar, for recovering errors. His parser focused on the repair of errors at the syntactic level. Therefore, the meaning of a sentence did not affect the repair of errors. His implementation will be described in the next section when both systems (Mellish's and MINSYS—my parser) are compared.

Some of the other systems described below handled ill-formed sentences both at syntactic and semantic level. The semantic level is emphasised in the domain of Natural Language Interfaces (Carbonell & Hayes, 1983, Grishman & Peng, 1988, Hayes & Mouradian, 1981).
3.2 The Others

Hayes and Mouradian (1981) implemented and designed a system to provide flexibility for restricted natural language input to a limited-domain computer system. Their parser (FlexP) is a bottom-up pattern-matching parser with three features: pattern matching, bottom-up parsing, and parse suspension and continuation. The parser operated in a breadth-first mode. The final goal of FlexP was to deal with most of the grammatical deviations in the interface of a restricted-domain system, and to obtain an appropriate meaning for ill-formed inputs.

Kwasny & Sondheimer (1981) implemented an ATN parser for recovering errors using relaxation and pattern-matching algorithms. They focused on recovery from syntactic errors. In recovering errors, they used two methods: one is feature relaxation and the other method is pattern-matching algorithms with patterns and pattern arcs (e.g., Mary drove the car. PAT(Person Trans-Act OBJ)). These pattern arcs can be modified to permit optional constituents as a relaxation.

EPISTLE (Jensen, Heidron, Miller, & Ravin, 1983) focused on the processing of syntactic ill-formedness. Its processing strategy is fitted parsing. After applying conventional syntactic rules to input sentences to produce a structural description, if there is no parse tree, then the fitted parsing technique can be used to produce a reasonable approximate
parse tree. This system is implemented by an augmented phrase structure language and explains the relations between stylistic sentences such as fragmentation error, vocative, etc., and handling of ill-formed input.

EPISTLE consisted of Core Grammar and Peripheral Procedures for handling ambiguous parsing and parsing failure. The fitting procedure deals with selection of head constituents and fitting remaining constituents so as to select the largest sentence-like string within a text string and deem it to be central. Selecting and fitting constituents are preferentially ordered.

Weischedel & Sondheimer (1983) approached ill-formed input using rule-based system in an ATN parser which could recover errors at the syntactic and semantic levels. The left-hand side of a meta-rule diagnoses a problem as a violated rule of normal processing. The right-hand side relaxes the violated rule and states how processing may be resumed (e.g. (Failed-Test? (Subject-Verb Agree? ?X ?Y)) → (New-Configuration (Failed-Constraint (Subject-Verb-Agree? ?X ?Y)) (Substitute-In-Arc (Subject-Verb-Agree? ?X ?Y). T))). They divided ill-formedness into two types, ill-formedness in the absolute sense, such as misspellings, meaningless sentences, extraneous forms, etc. and in the relative sense, the input sentences beyond the limits of either the computer system or the natural language interface. They also explained four alternative approaches to ill-formedness in terms of the
combination of two systems, the well-formed system and the ill-formed system.

Carbonell & Hayes (1983) classified different types of grammatical deviations at the lexical, sentential, and dialogue levels and presented recovery strategies. They implemented two parsers called DYPAR and CASPAR. CASPAR is implemented by case frame instantiation and has many advantages as a framework for recovering errors of sentential level ungrammaticality. DYPAR explored the notion of combining several different parsing strategies including pattern-matching parsing, semantic grammar interpretation, and syntactic transformations, in 2 single parser. DYPAR can recover errors, misspelled words, spurious phrases, out-of-order constituents, and ellipsis in a restricted domain.

Fass & Wilks (1983) described relationships between PS (Preference Semantics) and Ill-formededness and between Preference Semantics and Metaphor. Preference Semantics focused on the understanding of ill-formed input sentences at the semantic level.\(^{10}\) Their system consists of a semantic formula,\(^{11}\) which is a representation of a word-sense, and a template,\(^ {12}\) which is a structure, with three slots: the action, agent, and object slots. Semantic information in dictionary entries was analysed as composed of Inherent Information (Data, e.g. drink (MOVE CAUSE)), Label information (Labels, e.g. drink (*ANI SUBJ), ((FLOW STUFF) OBJ)), and Contextual information (Expectations,
e.g. *drink ANI, (FLOW STUFF)). With their semantic information, semantic preferences are applied to the input sentences.

Fass & Wilks use four strategies for the representation of metaphor with the semantic formula of ill-formed input:

(1) Passive strategy: Relax the preference of the predicate.
(2) CTD(Change The Data) strategy: Change the inherent data (e.g. *the car drank gasoline). Change the semantic information of a word *car from (**INANI SUBJ) into (**ANI SUBJ)).
(3) CTE(Change The Expectation) strategy: Change the expectations (e.g. *the car drank gasoline). Change the semantic information about a word like *drank from (**ANI SUBJ) (((FLOW STUFF) OBJ) (MOVE CAUSE))) into (**INANI SUBJ) (((FLOW STUFF) OBJ) (MOVE CAUSE)))
(4) Active strategy: Produce a completely new semantic formula.

According to Fass & Wilks's experimental results, each strategy is good at recovering errors of certain phenomena but bad at others. Therefore, Preference Semantics needs delicate ways to control the four strategies to produce the most appropriate reading.

Other approaches to performing error recovery with ill-
formed input sentences include:

Semantic and syntactic corrections with user feedback systems
implemented by finding the closest pattern (Grishman & Peng, 1988).
NOMAD system (Expectation-based system by conceptual analysis) (Granger, 1983).
History-based Expectation system applying to speech understanding (Fink & Biermann, 1986).

4. DESIGN IMPLEMENTATION OF MINSYS

Mellish set up to recover from errors in ill-formed input sentences at the syntactic level and with semantics-free strategies. According to Weischedel & Sondheimer (1983), four alternative approaches to ill-formedness can be described. One of them is to build a dual system: a well-formed system and a backup ill-formed system. Mellish's system falls into this class, as does MINSYS.

I have implemented a system based on chart parsing called MINSYS. MINSYS comprises a well-formed chart parser (WFCP) and an ill-formed chart parser (IFCP). The
well-formed chart parser follows a bottom-up parsing strategy and parses left-to-right. The ill-formed parser uses two strategies: top-down hypothesis and bottom-up satisfaction. The bottom-up parsing method uses both left-to-right and right-to-left parsing techniques.

MINSYS is programmed in Allegro Common Lisp Version 1.3 in a Macintosh computer environment. The lexicon contains 370 words and has a word-based structure. The lexicon is implemented as an array structure. The grammar, which is implemented as a list structure, is composed of 25 context free rules. With these rules it is only possible to parse declarative sentences. I shall now explain MINSYS and the difference between MINSYS and Mellish's system. And I will focus on the comparison of the ill-formed input parsers in MINSYS and Mellish's system.

4.1 Implementation of WFCP(Well-Formed Input Chart Parser) in MYNSYS

WFCP is a bottom-up, left-to-right parser which produces a structural description of correct sentences. There are two data structures, the active chart and inactive(completed) arcs. The active chart includes information on what fragments of phrasal constituents are to be found between pairs of positions in an input sentence. The inactive arcs
contain the information about the well-formed substrings. And the *stack* is an agenda which determines the order for invoking new rules and transforming arcs in the active chart to produce, ultimately, inactive arcs. WFCP is composed of the *Data* (Grammar, Lexicon, and Sentence), the *Parser*, and the *Active chart*. Each active arc has six components: Ruletype, Category, Predot, Postdot, From, and To.

Inactive arcs are the data structure for completed constituents (lexical or phrasal). An inactive arc has seven components: Nodename, Ruletype, LHS, RHS, From, To, and Capinfo.¹⁰

### 4.2 Implementation of IFCP
( *Ill-Formed Input Chart Parser* )

If the WFCP fails to parse an input sentence, then the IFCP will be invoked. The goal of the IFCP is to recover the syntactic structure of an ill-formed sentence by performing minor error correction. The IFCP begins by hypothesising an S (sentence) node as its final goal. Unlike WFCP, IFCP uses two general strategies in parsing ill-formed sentences. The expected goal is expanded by invoking grammar rules (a top-down parsing strategy). The RHS (lexical or phrasal constituents) of the expanded goal may be satisfied with bottom-up parsing using inactive arcs generated by WFCP, or
further expanded (if phrasal) by top-down parsing. From this point of view, a chart-based parser is different from other systems recovering ill-formed input sentences. Without backtracking, as in an ATN parser, and so parsing the same input frequently, known well-formed substrings can be used for the satisfaction of expanded goals in bottom-up parsing. During the bottom-up parsing process, left-to-right parsing and right-to-left parsing are both used.

IFCP has four components:

- data (Grammar and Inactive arcs),
- ill-formed parser,
- need-chart,
- a fundamental rule, as in WFCP.

4.2.1 Goals

The S node is always the first goal of IFCP. Then this goal is expanded by the top-down parsing using grammar rules. An expanded goal generated by top-down parsing is repeatedly processed by the bottom-up parsing method until there is no further satisfaction.

The IFCP manipulates goals in four ways:
(1) make goals, which satisfies the repairing condition of errors, for the input of the inference engine.
(2) make goals for further expansion by the top-down parsing.
(3) make goals for further satisfaction by the bottom-up parsing.
(4) discard the expanded goals because they are beyond error recovery strategy.

4.2.2 Bottom-up Satisfaction

After the goal node is expanded by means of top-down parsing with grammar rules, bottom-up parsing processes these goals using inactive arcs found by WFCP. There are two bottom-up parsing applications: left-to-right and right-to-left methods. If a goal node is a phrasal constituent, there are four ways to apply bottom-up parsing:

(1) The left and right constituents of an expanded goal can be satisfied with inactive arcs. Then left-to-right parsing and right-to-left parsing are applied to this goal.
(2) Only the left constituent of an expanded goal can be satisfied with inactive arcs. Then left-to-right parsing is applied to this goal.
(3) Only the right constituent of an expanded goal can be
satisfied with inactive arcs. Then right-to-left parsing is applied to this goal.

(4) If there is no satisfaction with inactive arcs, then the goal, which is expanded by top-down parsing, having label (LHS of grammar rules) and ruletype, makes a *narc* (my term, which means an are generated by IFCP) without inactive arcs.

There are two outputs. One becomes the input to a *Need-Chart*, and the other becomes the input to bottom-up parsing again.

### 4.2.3 Need-Chart

This idea is similar to the active chart. This is a chart for needed constituents, used to store information about which constituents in expanded goals are satisfied and which are not and is the collection of an narc. With this Need-Chart, we can see both satisfied and expecting constituents of expanded goals by top-down parsing. A Need-Chart entry consists of a Label, a Ruletype, Inactive nodes, Neednode, From and To positions, and Penalty Score.

If a newly-made narc satisfies the error correction condition, it is inactivated as a new inactive are and becomes a new agenda item for recovering new syntactic structure with
deviance note.\textsuperscript{15)}

4.2.4 Inference Engine

This algorithm attempts to identify error and recover from them. There are two modes of inference used in error recovery.

(i) Inference from Goal: In the example sentences used, there are errors caused by deleting a word or substituting a word. Such cases are recovered when making a goal node. When a word is deleted, the recovery is to add a lexical category. In case of substitution, the recovery is to substitute a word for an expected lexical category.

(ii) Inference from Need-Chart: The Need-Chart contains information about known constituents. When the error is an added word, recovery is performed by deleting the word using Need-Chart.

4.3 Comparison between MINSYS and Mellish System

Both systems are based on syntactic ideas but the two systems have major differences. Mellish's system is implemented using Generalised Top-down parsing rules including a Top-down rule, a Fundamental rule, a Simplication
rule, a Garbage rule, an Empty category rule, and an Unknown word rule. MINSYS is implemented using the Need-Chart, Top-down Hypothesis (Goals), Bottom-up parsing, and an Inference composed of two different types of error recovery process. Both systems use penalty scores for retrieving rules by top-down hypothesis but there are many difference in system implementation (Figure 1.)

5. EXPERIMENTAL RESULTS

When parsing ill-formed inputs by the bottom-up parsing strategy, left-to-right or right-to-left mode is selected by bottom-up parsing conditions. Left-to-right parsing mode is the system default. Because of this use of left-to-right and right-to-left parsing modes, left-corner and right-corner errors are detected and recovered easily and rapidly but errors occurring in the middle of sentence are not so easily done. The testing of MINSYS was done with 4 sentences of different lengths.

(1) "I love you."
(2) "The boy likes a good girl."
(3) "The good boy likes a sad girl with anger."
(4) "The boy believes the fact that he likes the sad girl with anger."
These four sentences are chosen because: (i) the length of sentences is equal to that of sentences tested by Mellish's system; (ii) all four sentences contain some ambiguous words; (iii) the fourth sentence contains an embedded sentence.

In terms of types of solutions (Figure 2.), MINSYS is robust in repairing errors of omission and unknown words, but generates a strange result in the case of adding a known word. In such case, instead of deleting a word, which might be either the added word or not, my system tended to add another lexical category. This means that the error was identified as a missing word. When input sentence has an unknown substitute word, the error is recovered in 85% of 31 cases. MINSYS is better for recovering the unknown substitute word. The other results are delete one word (62% recovery), add unknown word (56% recovery), and add known word 24% recovery).

<table>
<thead>
<tr>
<th>COMPARISON</th>
<th>MELLISH</th>
<th>MINSYS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule Invocation</td>
<td>One by One</td>
<td>All Rules at once</td>
</tr>
<tr>
<td>Error Recovery</td>
<td>In Top-down rule</td>
<td>In Goal &amp; Need-Chart</td>
</tr>
<tr>
<td>BUP Strategy</td>
<td>Left-to-right parsing</td>
<td>LR &amp; RL parsing</td>
</tr>
<tr>
<td>Preferred Solution</td>
<td>Preferred solution</td>
<td>Not preferred solution</td>
</tr>
<tr>
<td>Focused Parsing Cycle</td>
<td>Top-down-Oriented</td>
<td>Bottom-up-Oriented</td>
</tr>
<tr>
<td>Other</td>
<td>Penalty score for the best-in-rule invocation</td>
<td>Uses Need-Chart &amp; Fundamental rules in IFCP</td>
</tr>
</tbody>
</table>
In the case of *delete one word*, there are 9 Well-formed sentences parsed by the WFCP. This means that it is impossible to identify them as ill-formed sentences and hence to repair them using the IFCP.

5.1 Conclusion on Results

The important evaluation factors for computational results are *running time* and *used space*. My results are measured in terms of time. I shall now explain the relationship between running time in MINSYS and the other factors such as TDC (Top-down cycle used in the top-down parsing), BUC (Bottom-up cycle used in the bottom-up parsing), number of ambiguous words, location of errors, and the depth of locality.

Number of top-down cycles is the major factor in determining running time. As it were, the expansion of goals influences the efficiency of error recovery. The efficiency of top-down parsing depends on how to localise the error by invoking best-fit rules. The efficient error recovery algorithm for the context free grammar is mentioned by (Anderson Blackhouse, 1981).

The number of top-down cycles is related to the length of sentence and the size of grammar. When the length of sentence increased, the number of top-down cycles is also
increased. The syntactic complexity of a sentence and the position of the error in the syntactic tree also affects the number of top-down cycles. My system runs two modes of bottom-up parsing: left-to-right and right-to-left parsing. Therefore, it is harder to detect errors in the middle of sentence than in the left-cornered or right-cornered errors. My system prodused solutions without preference factors in rule invocation.

According to some experimental results using spoken text. (Cooper, Tye-Murray, Nelson, 1987), listeners could detect errors better when they paid attention to detecting errors than when they didn’t pay attention to this. Their result indicated that listeners were highly accurate in reporting the presence of missing word (96% of 190 cases) when they were forwarned about the specific types of errors that might be encountered while listening. But the detection of missing words (34% of 80 cases) was quite poor when such words are readily predictable from context and when listeners were not forwarned about the specific types of errors. Thus when the types of error are notified before error recovery begins, the performance of error recovery is increased.

If the types of error are identified before parsing ill-formed sentences by the IFCP and the IFCP uses different strategies to parse different types
1. Delete One Word

<table>
<thead>
<tr>
<th>Length of Sentence</th>
<th>Number of Solutions</th>
<th>Delete a Word</th>
<th>Substitute a Word</th>
<th>Add a Word</th>
<th>WPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>6(100%)</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>14</td>
<td>2(14%)</td>
<td>2(14%)</td>
<td>10(72%)</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>29</td>
<td>7(24%)</td>
<td>8(28%)</td>
<td>14(48%)</td>
<td>3</td>
</tr>
<tr>
<td>13</td>
<td>79</td>
<td>8(10%)</td>
<td>22(28%)</td>
<td>49(62%)</td>
<td>3</td>
</tr>
<tr>
<td>Total(%)</td>
<td>128</td>
<td>17(13%)</td>
<td>32(25%)</td>
<td>79(62%)</td>
<td>9</td>
</tr>
</tbody>
</table>

2. Add Unknown Word.

<table>
<thead>
<tr>
<th>Length of Sentence</th>
<th>Number of Solutions</th>
<th>Delete a Word</th>
<th>Substitute a Word</th>
<th>Add a Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>8</td>
<td>4(50%)</td>
<td>4(50%)</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>14</td>
<td>7(50%)</td>
<td>7(50%)</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>16</td>
<td>10(63%)</td>
<td>6(37%)</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>50</td>
<td>28(56%)</td>
<td>22(44%)</td>
<td>0</td>
</tr>
<tr>
<td>Total(%)</td>
<td>88</td>
<td>49(56%)</td>
<td>39(44%)</td>
<td>0(%)</td>
</tr>
</tbody>
</table>

3. Add Unknown Word.

<table>
<thead>
<tr>
<th>Length of Sentence</th>
<th>Number of Solutions</th>
<th>Delete a Word</th>
<th>Substitute a Word</th>
<th>Add a Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>19</td>
<td>6(31%)</td>
<td>7(38%)</td>
<td>6(31%)</td>
</tr>
<tr>
<td>6</td>
<td>48</td>
<td>17(35%)</td>
<td>14(30%)</td>
<td>17(35%)</td>
</tr>
<tr>
<td>9</td>
<td>69</td>
<td>20(29%)</td>
<td>22(32%)</td>
<td>27(39%)</td>
</tr>
<tr>
<td>13</td>
<td>283</td>
<td>55(20%)</td>
<td>80(28%)</td>
<td>148(52%)</td>
</tr>
<tr>
<td>Total(%)</td>
<td>419</td>
<td>98(24%)</td>
<td>123(29%)</td>
<td>198(47%)</td>
</tr>
</tbody>
</table>

4. Substitute Unknown Word.

<table>
<thead>
<tr>
<th>Length of Sentence</th>
<th>Number of Solutions</th>
<th>Delete a Word</th>
<th>Substitute a Word</th>
<th>Add a Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>7</td>
<td>1(14%)</td>
<td>6(86%)</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>3(25%)</td>
<td>9(75%)</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>17</td>
<td>3(18%)</td>
<td>14(82%)</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>48</td>
<td>6(13%)</td>
<td>42(87%)</td>
<td>0</td>
</tr>
<tr>
<td>Total(%)</td>
<td>84</td>
<td>13(15%)</td>
<td>71(85%)</td>
<td>0(%)</td>
</tr>
</tbody>
</table>

Figure 2. Type of Solutions
of errors, the performance of parsing ill-formed sentences would be increased.

6. PROBLEMS

MINSYS is based on two bottom-up strategies, left-to-right and right-to-left modes. In the case of bottom-up parsing of phrasal constituents, left-to-right and right-to-left parsing have both been done. Therefore some redundant tasks are done by the ill-formed chart parser, WFCP and IFCP. If WFCP failed to produce syntactic description of inputs, then IFCP is applied to what WFCP leaves behind. Because of this mode of control between WFCP and IFCP, there are some cases of inability to detect and recover errors (in the case of Deleting one word).\textsuperscript{17}

A few problems are also noted below:

1. Some inactive arcs for the bottom-up sparsing are retrieved redundantly.

2. When retrieving rules by the top-down parsing, MEL (Minimal Extending Lenght of LHS in grammar rules) is the only source of information in deciding which rules to retrieve. The retrieving condition is $\text{MEL} \leq \text{RMWS} (\text{Remaining Maximal Word Score}) + 1)$. It is hard to retrieve the best rules for error recovery. Further, with this technique, some rules are not retrieves for bottom-up parsing. This, however, is not quite
enough for the best expectation of error recovery because some rules which are not related to the error are retrieved.

3. If there are constituents meeting left-right and right-left parsing conditions, they are parsed by both left-to-right and right-to-left bottom-up parsing. Such constituents increased the running cycle of the bottom-up parsing.

4. Because of left-to-right and right-to-left parsing modes and problem 1. every constituent expanded by top-down parsing and bottom-up parsing is kept in Need-Chart redundantly.

6.1 Further Development

For further development of my system, there are a few things which could increase the performance of top-down expectation.

1. When retrieving the rules to expand goals, the system should infer the best-fit rules by top-down parsing to minimize the cost of recovering errors.

2. When doing bottom-up parsing, the system should not redundantly retrieve inactive arcs required by both left-to-right and right-to-left parsing modes.

3. This is based on the recovery of primitive errors assuming that there are exactly one error in a sentence. When multiple errors happen, they can be also detected and recovered using the Need-Chart. One should consider how to extend my system to allow recovery from multiple errors.
7. Conclusion

The goals of my project were the optimization of the number of bottom-up cycles and top-down cycles and to implement basic ideas for recovery of ill-formedness with a chart parser. MINSYS produced all possible solutions in terms of the context-free grammar rules. Compared with Mellish's system, the number of top-down cycles is reduced, but the number of bottom-up cycles is increased. That is because my system is bottom-up-parsing-oriented, using both left-to-right and right-to-left parsing modes. Unlike Mellish's system, the diagnosis of rightward errors is easily done, like that of leftward errors. However, errors in the middle of sentences are not easily detected and recovered.

When retrieving rules at the top-down hypothesis, I used a MEL (Minimal Extending Length) score which allowed a decision on what rules to retrieve for bottom-up parsing. If MEL is used more intelligently, the performance of my parser could be increased. With the Need-Chart information about processing using the bottom-up parsing strategy is kept. Deviance notes on ungrammatically and error repair are kept in SNODEs.

My system is semantic-free and focused on syntactic ill-formedness. If some other features (syntactic features, number, case, gender, subcategorization of verb, etc.) are merged into my system, more accurate solutions would be produced. Finally Mellish tried to recover errors independent
of grammatical rules. My results indicated that the parsing of ill-formed sentences didn’t depend on grammar rules bit the repair of ill-formedness is greatly dependent on the rule descriptions.

For further improvement of the ill-formed chart parser at the syntactic level is:

• How to infer the best rules to use with the top-down hypothesis.

내용 주

1) These concepts are not similar to the concept of grammaticality. The syntactic rules can assess the ill-formedness or well-formedness of a sentence at the syntactic level. Sentences can be ill-formed or well-formed at three levels, syntactic, semantic, and pragmatic level.

2) With his algorithm, the sentence composed of n words would be parsed at least k(n^3) time.

3) Woods implemented a system using well-formed string tables for the further use in order that ATN parser could avoid to re-parse constituents parsed already.

4) When the ATN parser parses a word in a sentence, it stores lexical ambiguities of the word and the choice of an arc by processing forward the word. Then the arcs which are not chosen are stored for completion of all available tree structures. These arcs are used for 1) backtracking when the chosen are fails to creat complete tree structure and 2) generating all the available tree structures.

5) The inactive arcs (called as an agenda) invoke rules to be active. Then the rule invocation strategy (rule invocation order) has to be considered to get a best fit parse tree for a sentence.
6) Earley's algorithm is considered as the most efficient parsing algorithm for context-free grammar without any choice condition in processing arcs and backtrakings.

7) To identify the end of input sentence, my system uses a special word "full-stop" instead of a period ".” This is for distinguishing the end of sentence and the other use of period(e.g. 0.799, etc.)

8) The senses of "acceptable" is a little different from that of Chomskian term "acceptibility"(see "Aspect of the Theory of Syntax"(Chomsky, 1965)). "Acceptable" means that a hearer can understand a sentence, which has some degree of ill-formedness, after minor change of its ill-formedness.

9) (1) test relaxation: with opposite predicate values (e.g. Subject-Verb Agreement: Subject (Number Singular), Verb(Number Plural→Number Singular)) or with substitute predicate values(e.g.(Intrans V) → (Trans V)) and (2) category relaxation(e.g. Personal Pronoun → Demonstrative Pronoun).

10) Wilks aregeed against the Chomskians view, grammaticality. He stated the meaningfulness of a sentence to understand natural language. He objected the syntactic component of Chomskian grammar. Instead of syntactic parsing, he parsed sentences using direct semantic parsing method, preference semantics.

11) (e.g. drink (** ANImate SUBj ect) (((FLOW STUFF) OBJ ect) (MOVECAUSE))))

12) (e.g. the police man interrogated the crook. ( [(policeman) [interrogated] [crook (man)]]. ( [(policeman) [interrogated] [crook (thing)]]))

13) Times given in this paper relates to a Macintosh I si with 5MB.

14) Capinfo stands for control agreement principle information. Control Agreement principle is one of rules in "phrasal structure grammar" by Gazdar et al. CAP controls number, case, and gender agreement in a sentence.

15) Considerer part of a sentence " o the 1 gud 2 boy 3 ..." The grammar rule from my system NP→ (DET NOUN) is retrieved by top-down parsing. By bottom-up parsing DET1 ("the")and NOUN1 ("boy") satisfies the expanded goal NP. A narc NARC1 = (DET1 NOUN1) without neednode (which means the constituents of RHS of this NP are all satisfied by bottom-up
CHART PARSER FOR ILL-FORMED INPUT SENTENCES

parsing without any expecting constituents) from 0 to 3 is generated with
devience note (SNODE1: word = "gud", category= nil. from = 1, to =
2.type = delete. This NARC1 is under consideration of becoming a new
inactive arc.

16) An original sentence is "the good boy likes a dog." When deleting
the word "good", the sentence "the ( ) boy likes a dog." is parsed by the WFCP as
well-formed (See also chapter 6. Problems).

17) For example, The original sentence is "the good boy has a beautiful dog".
When the word "good" or "beautiful" is deleted respectively, the parse tree
of sentences (e.g. "the boy has a beutiful dog" or "the good boy has a
beautiful dog") are successfully made by the WFCP.

◆ REFERENCES


Network Grammars. In Lecture Notes in Computer Science. Goos,


Extragrammatical Language. AJCL. Vol.9, No.3-4. 123-46.

No.3. 233-40.


formed Input. AJCL, Vol.7, No.4. 257.

Metaphor. AJCL, Vol.9, No.3-4. 178-87.


 잘못 형성된 입력문장에 대한
CHART PARSER

민 경호

본 연구는 잘못 형성된 입력에 대한 멜리쉬의 연구(1989)에 기반하고 있다. 이 글은 chart-based parser를 이용하여 구문론적 차원에서 잘못 형성된 입력 문장의 복구에 초점을 둔다. 멜리쉬의 체계는 두가지 분석기, 즉 잘못 형성된 입력 분석기와 잘못 형성된 입력 분석기로 구성되어는데, 필자의 연구는 그의 생각을 따르고 있다. 이글에서는 주로 chart parsing의 개념, 잘못 형성된 입력에 대한 분석전략이 논의된다. 또한 필자가 제시하는 체계의 디자인과 구현, 필자의 체계를 멜리쉬의 체계와의 비교와 같은 사항들이 다루어질 것이다.

AI Lab
School of Computer Science & Engineering
The University of New South Wales
NSW 2033, Australia