Analysing Ill-formed Input with Parallel Chart-based Techniques

Thanaruk Theeramunkong   Hozumi Tanaka

Department of Computer Science,
Tokyo Institute of Technology
2-12-1, O-okayama, Meguro-ku, Tokyo 152, Japan
Tel. (03)3726-1111 (Ext. 4175)
E-mail {ping,tanaka}@cs.titech.ac.jp

Abstract

The fact that a natural language system cannot tackle with a sentence which is not suitable with its database; its current grammar and its current dictionary, inspired some researches on the ill-formed input analysis problem. However, there is a trend that it is time-consuming task to analyse ill-formed input in order to cover all possible error patterns and to find all possible interpretations. The ill-formedness framework defined in this paper is based on a set of defined rules to diagnose simple errors, such as unknown/misspelled words, omitted words, extra noise word. In this work, we consider the issue of parallelism of chart-based parser paradigm and its extension for dealing grammatically ill-formed sentence analysis. The techniques described here are semantic-free, grammar independent. Our system have been implemented in a practical concurrent logic programming language KL1 (based on GHC) on a parallel inference machine, PIM, which is a loosely-coupled system developed under the Fifth Generation Computer Project (ICOT). At the final part of this paper, we also show the preliminary result of our system testing with some grammatical inputs and some ill-formed inputs. Our system gets a fine performance when multi processors (to 16 processors) are used to analyse ill-formed inputs which have one and two errors detected.

1 Introduction

In recent year, researches on natural language processing, particularly syntactical analysis, have been developed rapidly to be used in satisfying real applications. As natural language is frequently used ungrammatically, an applicable system has to have a property of handling with grammatically ill-formed input.

There have been many studies for processing ill-formed input. With the criterion of the framework that any errors are found, these approaches are classified to two groups, the frame-based group and the syntactically-oriented group. The former attempts to deal with ungrammatical input through extensions to pattern matching [Hayes and Mouradian, 1981], though conceptual case frame instantiation [Schank *et al.*, 1980] and though approaches involving multiple cooperating parsing strategies [Carbonell and Hayes, 1983]. These researches work on dialogue phenomena in communication with the limited-domain systems, such as data-base systems, electronic mail systems, etc.

As the syntactically-oriented approach, the latter is the most popular among the early systematic researches on the robust paring systems. The representative types of this group are the robust parsers based on Augmented Transition Networks (ATN) which use the relaxation techniques [Kwasny and Sondheimer, 1981] or the meta-rule [Weischedel and Black, 1980], [Weischedel and Sondheimer, 1982] and the EPISTLE system which addresses the problems of the checking grammar and style of texts, such as letters, reports and manuals, written in ordinary English [Heidorn, 1982], [Jensen *et al.*, 1983]. In addition, the robust chart-based parsers are developed to solve the problem that the "longest path" heuristic, which is used in ATN-based approach, fails to indicate the minimal error owing to the left-right bias of an ATN parser [Mellish, 1989], [Kato, 1991], [Meknavin *et al.*, 1993].

Our system's robust paradigm is based on the chart-based parser in [Kato, 1991] that occupies a simple algorithm instead of using complicated searching control applied in [Mellish, 1989], to get all plausible analyses of ill-formed sentences. When the input sentence enters, the system will attempt to find the well-formed interpretation for the input in bottom up style but if it fails, that is the input has ill-formedness, the system starts the second process to find the possible interpretation in top down style.

Naturally, the more coverage the parser can give, the more time it consumes. This work is a parallel approach developed among the possible resolutions to this time-consuming problem. In the rest of this paper, we show the short description of the parallel bottom up process and the parallel top down process in order. Finally, we show the preliminary experiment on a parallel inference machine, PIM, which is a loosely-coupled system developed under the Fifth Generation Computer Project (ICOT).

2 Parallel Chart Parsers

Chart parsing is achieved by keeping two data records, a record of all parses undertaken (also called active edge) as an Active Edge Table (AET) and a record of all sub-strings (also called inactive edge) found in the Well Formed Substring Table (WFST) [Trehan and P.F., 1988]. The actions in chart parser are of two types, *creating edge* and *extending edge*. The edge creation is the action that an existing inactive edge builds a new (active or inactive) edge based on the current grammar rules. The edge extension is the action that an existing active edge extends by using an inactive edge to build a new (active or inactive) edge.

There are some existing attempts to construct parallel chart parser for context free grammar in both shared-memory environment and loosely-coupled environment, e.g. [Grishman and Chitrao, 1988], [Henry, 1989], [Trehan and P.F., 1988]. The shared-memory based method faces with synchronization problem. For the loosely-coupled environment method, the communication cost becomes the important factor. In [Grishman and Chitrao, 1988], The parser was developed for the NYU Ultra-computer, a shared-memory MIMD parallel processor with a special instruction, *fetch-and-add*, for processor synchronization. A set of processors all execute the main loop of the serial algorithm (get task from *agenda* / create edge / extend edge). The *agenda* is a task queue. This work is implemented in ZLISP, a parallel lisp. Using low-level synchronization operations, the system gets good performances as its result. The system ran 5 to 7 faster than the serial version.

In [Henry, 1989], the construction of a parallel chart parser on the Intel Hypercube, a loosely-coupled system is proposed. The chart is distributed among the processors on vertex by vertex basis to acquire the parallelism. However, when compared with the shared-memory, the system suffered with clearly dominate communication costs owing to the fast processors but slow network communication and no advantage is gained.

In addition, a parallel chart parser for the committed choice non-deterministic (CCND) logic language is proposed in [Trehan and P.F., 1988]. This chart parsing framework is developed to support an incremental bi-directional process, called active chart parsing [Winograd, 1983] and has been iŋplemented in Parlog. However, there is no evaluation on the practical computational time shown in this work.

3 Error Types Defined

As the first step, primitive errors defined in the system are of four types as following. An error may be a combination of these types. The definition is the same as ones defined in [Mellish, 1989].[Kato, 1991]. We illustrate these types of errors through a correct example sentence, *The movie is a world event.*

- Added word : during analysing an input sentence, if there are one or more words excessive in the input sentence, a perfect correct interpretation is not found. e.g., *The movie is a a world event*
- Omitted word : the contrary case of added-word error, if there are one or more words disappearing from the input sentence, a perfect correct interpretation is also not found. e.g., *The movie a world event*
- Unknown (Misspelled) word : generally, when a mistyped word or a undefined word is used in the sentence, it is rejected by the parser. e.g., *The movie is a wordl event*
- Substituted word : when a word in the correct sentence is substituted by a known/unknown word, the complete parse of the input sentence will not be generated and the error is detected. e.g., *The movie his a world event*

4 Overview of Parsing Process

Our basic strategy is to run a parallel bottom-up parser over the input and then, if this fails to find a complete parse for the input, to execute a parallel edge-completion process and a parallel augmented top-down parser. Figure 1 indicates the overview of our chart-based parsing.

Bottom up parsing (in the sense of left corner parsing without top-down filtering) is guaranteed to find all complete constituents of every parse which is a part in the current input sequences.

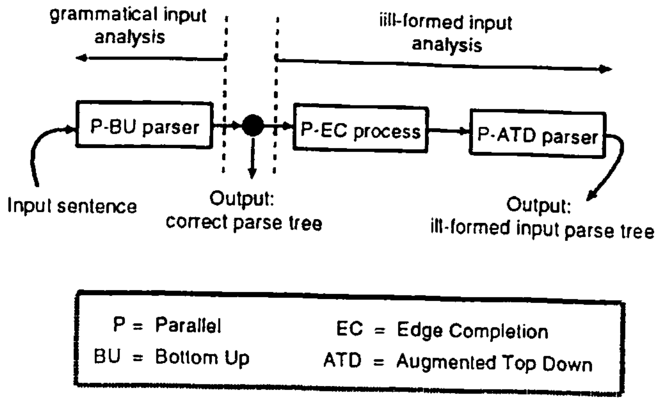


Figure 1: Three constructions in parsing

In addition, it will only create an active edge restricted to left-to-right order.

Edge-completion process produces some edges which were not produced by bottom-up parsing, to reduce task done in parallel augmented top-down parser. The restriction of parsing in left-to-right order as occurred in bottom up parsing is relaxed and some other edges are generated.

Using the set of edges generated from the two previous processes, parallel augmented top-down parser attempts to find in parallel the errors that enable a complete parse to be constructed. Top-down parser produces the tree structure of the ill-formed input where errors are detected.

5 Parallel Grammatical Input Analysis

An interpretation of a grammatical input is recognized by a simple and efficient chart-based bottom-up parsing algorithm called *Word Incorporation (WI)* proposed in [Simpkins, 1990]. The WI algorithm is a specialization of the Chart which is restricted to be solely bottom-up (left-to-right or right-to-left). WI is superior to the active chart algorithm[Winograd, 1983] in the way that it need not check whether an edge has previously been proposed or not, and in the way that it generates fewer edges because of generating in only bottom up style while active chart algorithm generates edges in both bottom-up and top-down direction.

This section described the parallelisation of WI. Similar to the parallel construction of Chart parser in [Henry, 1989] , the chart is distributed among the processors on vertex by *vertex basis* to acquire the parallelism. The term, *vertex basis*, refers to the position in the sentence. Figure 2 shows the way that the processors are distributed to the chart, using example sentence : *The movie is a world event.*

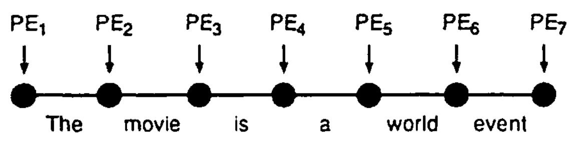


Figure 2: Processor distribution

Processor PE_n takes charge of creating and extending edges of which left position is n. Figure 3 shows the process that processor PE_n creates the active edge (2) using the inactive edge (1) of which left position is n.

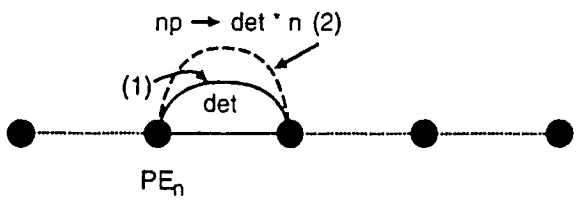


Figure 3: Edge Creation

Figure 4 represents the process that processor PE_i communicates with processor PE_j to get the inactive edge (4) and uses it to extend the active edge (3) and the inactive edge (5) is generated. .

The input is recognized to be grammatical when there is at least one edge covering between 0 and n (where n is the length of word sequences in the input). Such an edge appears at processor PE_0.

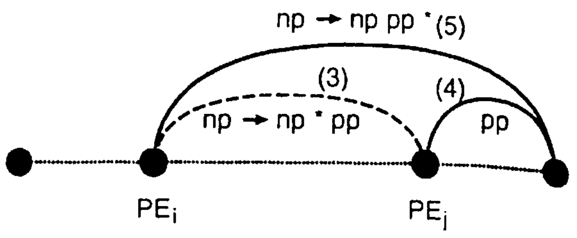


Figure 4: Edge Extension

6 Parallel grammatically Ill-formed Input Analysis

After the parallel bottom-up parser (P-BU parser) detects that the input is not grammatical, parallel edge-completion process (P-EC process) and parallel augmented top-down parser (P-ATD parser) get start to find all interpretations of the ill-formed input. P-EC process produces some other edges which are not generated in P-BU parser due to the restriction of left-to-right order. The distribution of processors in P-EC process is the same as used in P-BU parser. That is the processors are distributed among vertex by vertex basis (positions in the input sentence).

There are three actions in P-EC process. The first action is to build active edges from inactive edges acquired from BU parser using grammar rules. The generation occurs only when the inactive edge is not the leftmost element in the grammar rule. For instance, for a grammar rule (A → B C D), the active edge (A → B [C] D) will be generated when C is found during P-BU parser. Here, as our notation, [Cat] means Cat has already found. The second action is to extend active edges acquired from P-BU parser using inactive edges. The extension in the original chart parsing is to extend active edge from left-to-right in RHS but the extension in this process occurs to active edge to extend one which is not the leftmost in RHS. For instance, for an active edge (A → [B] C D) will be extended to get (A → [B] C [D]) when D is found during P-

BU parser but (A $\rightarrow$ [B] [C] D) is not generated because it has already generated during BU parser.

The last action is to extend the active edge acquired from two actions above. The action will occur to any element remaining in RHS with no restriction of left-to-right order. For instance, for an active edge, (A $\rightarrow$ B [C] D E), the action will extend D or E or both of them.

Top down parser can be viewed as a searching process using the information (generated edges) acquired from P-EC process. The searching process forms a tree of states whose root is initial state. The notation of each state is described in the form of < hole:N err:M need $CatList_1$ from S_1 to E_1, ..., $CatList_k$ from S_k to E_k >, where $CatList_i$ is a list of categories; $S_1, E_1, \ldots, S_k, E_k$ are positions in the input(sentence); ($CatList_i$ from S_i to E_i) means $CatList_i$ is needed between S_i and E_i; N is the total number of categories in $CatList_1 \ldots CatList_k$; M is the number of errors occurring before reaching this state.

The initial searching state is < *hole*:1 *err*:0 need [S] from 0 to n >, n is the final position in the chart (the length of the input). The parser uses active edges and inactive edges generated in EC process to make progress of searching.

Top down parser occupies five rules to find errors in the input sentence. These rules is similar to ones defined in [Kato, 1991] as shown in table 1 and 2. The *top-down rule* allows to refine a need into more precise one, using a rule in the grammar, that is it goes one level down in parse tree. The *active fundamental rule* allows a need to be refined with an active edge to be a more refined state. The *garbage rule*, *unknown word rule* and *empty category rule* are used to find respectively an added word, an unknown word and a deleted word in the current state.

A state corresponds to a task of searching. The searching process starts from initial state and after each application of a rule, a new state(task) may be generated. The five rules are the basis routines in each processor. When some new states(tasks) are produced in any processor, one of them is done in the current processor, while others will be distributed to processors which are idle. A processor manager is established to control the task distribution. Other processors, we called them *working processors*, send requests to the processor manager when they are idle and if there are some tasks in the queue of the processor manager, tasks in the top of the queue will be sent to those working processors. The task distribution occurs recursive until the queue is empty.

Top-down rule:

$$\frac{\begin{array}{l} < \text{hole: } N \text{ err: } M\ [C_1 \ldots Cs_1] \text{ from } S_1 \text{ to } E_1, \ldots, Cs_n \text{ from } S_n \text{ to } E_n > \\ C_1 \rightarrow RHS \end{array}}{< \text{hole: } N + (\text{length of RHS}) - 1 \text{ err: } M\ [\ldots \text{RHS} \ldots Cs_1] \text{ from } S_1 \text{ to } E_1, \ldots >}$$

Active Fundamental rule:

$$\frac{\begin{array}{l} < \text{hole: } N \text{ err: } M\ [\ldots Cs_{11}, C_1, Cs_{12} \ldots] \text{ from } s_1 \text{ to } e_1, \ldots > \\ < C_1 \text{ from } S \text{ to } E \text{ needs } Cs_1 \text{ from } S_1 \text{ to } E_1, \ldots, Cs_n \text{ from } S_n \text{ to } E_n > \end{array}}{\begin{array}{l} < \text{hole: } N + \Sigma(\text{length of } Cs_i) - 1 \text{ err: } M \\ Cs_{11} \text{ from } s_1 \text{ to } S, Cs_1 \text{ from } S_1 \text{ to } E_1, \ldots, Cs_n \text{ from } S_n \text{ to } E_n,\ Cs_{12} \text{ from } E \text{ to } e_1, \ldots > \end{array}}$$

Table 1: Augmented Top down rules(1)

7 Experimental Result

As a preliminary experiment, the parallel version of chart parsing is implemented on Parallel Inference Machine (PIM), which is a loosely-coupled system developed under the Fifth Generation Computer Project (ICOT) in Japan. The language used on the machine is a practical concurrent logic programming language named KL1.

Garbage rule:

$< \text{hole: } N \text{ err: } M\ [C_1 \ldots Cs_1] \text{ from } s_1 \text{ to } e_1, \ldots >$

$< C_1 \text{ from } S \text{ to } E_1 \text{ needs nothing} >$

$< \text{hole: } N-1 \text{ err: } M + (S_1 - s_1)\ Cs_1 \text{ from } E_1 \text{ to } e_1, \ldots >$

Unknown word rule:

$< \text{hole: } N \text{ err: } M\ [C_1 \ldots Cs_1] \text{ from } s_1 \text{ to } e_1, \ldots >$

$< C_1 \text{ from } s_1 \text{ to } s_1 + 1 \text{ needs nothing} >$

$< \text{hole: } N-1 \text{ err: } M+1\ Cs_1 \text{ from } s_1+1 \text{ to } e_1, \ldots >$

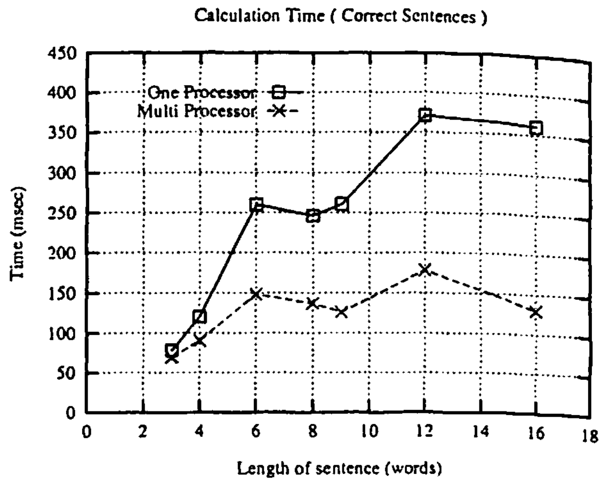
Empty category rule:

$< \text{hole: } N \text{ err: } M\ Cs_1 \text{ from } s \text{ to } s,\ Cs_2 \text{ from } s_2 \text{ to } e_2, \ldots >$

$< \text{hole: } N - (\text{length of } Cs_1) \text{ err: } M + (\text{length of } Cs_1)\ Cs_2 \text{ from } s_2 \text{ to } e_2, \ldots >$

Table 2: Augmented Top down rules(2)

The efficiency of the system is tested on 1-16 processors, using the set of 162 Thai grammar rules which are built based on Thai grammatical structure in [Panupong, 1984]. The length of the input ranges from three words to sixteen words for the grammatical input case and from nine to seventeen words for the ill-formed input case. Figure 5 shows the analytical time and speedup rate of the tested grammatical inputs. We can observe that there are consistent speed-ups in the range of 2-3 over one processor's time in grammatical case. The longer the input is, the larger the speedup rate is. Note that the number of processors used in each analysis corresponds to the length of the input. We have also considered and implemented another version of the parser with more fine-grained parallelism, that is the processors are not only distributed among vertex by vertex basis but there is no benefit obtained owing to the large cost of communication.



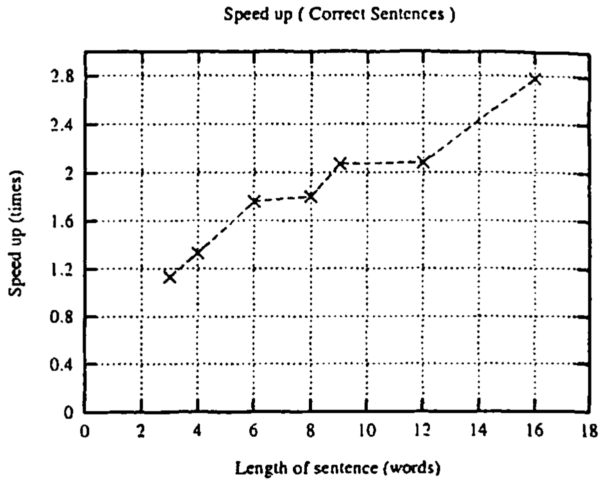


Figure 5: Analytical time and speedup rate gained when the input is grammatical and has a length ranging between 3 and 16 words

The experimental result of speedup rate in analysing ill-formed input is shown in figure 6 and 7. We tested the inputs with one or two added-known-word-typed, added-unknown-word-typed, deleted-word-typed, substituted-known-word-typed and substituted-unknown-word-typed errors. Figure 6 shows speedup rate during the analysis of a sentence of which length is nine while figure 7 shows speedup rate of a sentence consisting of seventeen words. We observed that the system get a higher speedup rate when the input is longer and has more errors. In the case of an input with seventeen words, all processors almost works without being idle and the speedup rate to be almost linear. However, in the case of an input with nine words, the analytical time is not dominant. So we think it is reasonably sufficient enough, in spite of this speedup rate.

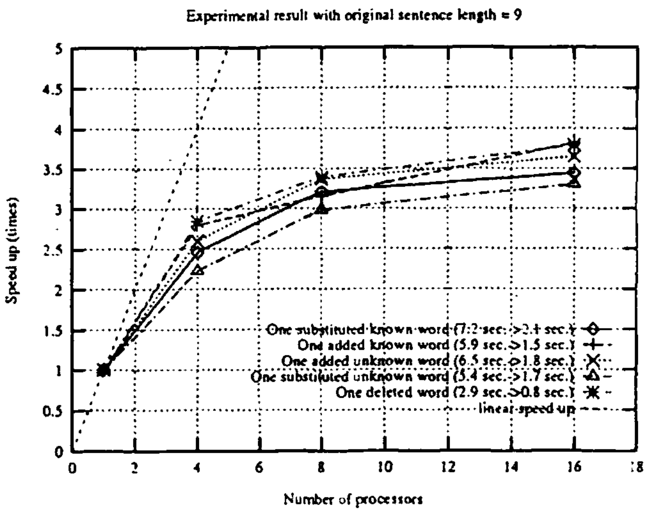


Figure 6: Speedup rate in analysing an ill-formed input with one error

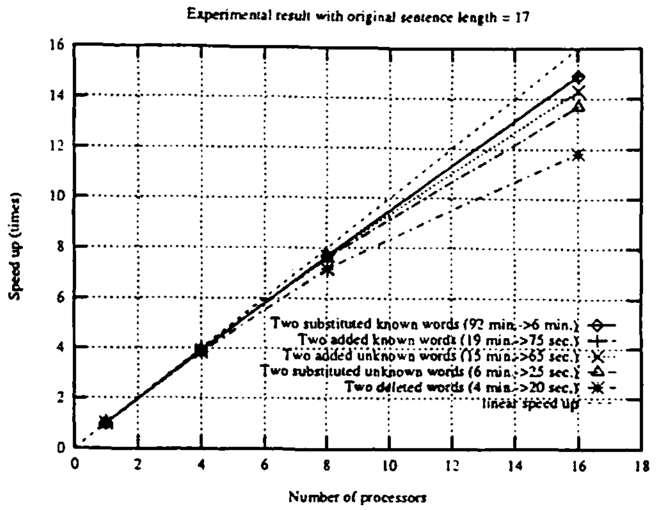


Figure 7: Speedup rate in analysing an ill-formed input with two error

8 Conclusion

Some Techniques in constructing parallel chart-based robust parser are proposed. The preliminary experiments are done to test the efficiency of the parser. The experimental result shows that the efficiency of the system is almost linear speedup rate when the number of processors used is from 1 to 16. Good load balance is acquired in this small configuration. We are on way to implement and test the idea on a large number of processors.

Finally, This work is a semantic-free, grammar independent analysis with context free grammar but we think it is important to understand the limits of recovery strategies that are based entirely on syntax and which are independent of any particular grammar, although the ultimate solution to the problem of processing ill-formed input must take into account semantic and pragmatic factors. The chart is the flexible and explicit representation and facilitates not only the extension in ill-formed paradigm but also the extension for semantic analysis in parallel paradigm as our further work.

Acknowledgement

We would like to thank ICOT for providing access to Parallel Inference Machine(PIM) and giving us the opportunity to conduct this research. We would like to thank K.Susaki who helps us in setting the machine and also gives us many useful comments. We also would like to thank M.Koshimura who helps me in some useful techniques during constructing the system. Finally, we would like to thank T.Tokunaga and our lab's members who give us many useful comments.

References

[Carbonell and Hayes, 1983] Carbonell, J.G. and Hayes, Philip J. 1983. Recovery strategies for parsing extragrammatical language. *American Journal of Computational Linguistics* 9:123–146.

[Grishman and Chitrao, 1988] Grishman, Ralph and Chitrao, Mahesh 1988. Evaluation of a parallel chart parser. In *Second Conference on Applied Natural Language Processing*, Austin, Texas. Association for Computer Linguistics. 71–76.

[Hayes and Mouradian, 1981] Hayes, Philip J. and Mouradian, George V. 1981. Flexible parsing. *American Journal of Computational Linguistics* 7(4):232–242.

[Heidorn, 1982] Heidorn, G.E. 1982. Experience with an easily computed metric for ranking alternative parses. In *Proceeding of 20th Annual Meeting of the ACL*, Totont, Canada. 82–84.

[Henry, 1989] Henry, S. Thompson 1989. Chart parsing for loosely coupled parallel systems. In *International Workshop on Parsing Technology*. Carnegie Mellon. 320–328.

[Jensen *et al.*, 1983] Jensen, K.; Heidorn, G.E.; Miller, L.A.; and Ravin, Y. 1983. Parse fitting and prose fixing: Getting a hold on ill-formedness. *American Journal of Computational Linguistics* 9(3-4):147–160.

[Kato, 1991] Kato, Tsuneaki 1991. Yet another chart-based technique for parsing ill-formed input. In *Natural Language Processing*. Information Processing Society of Japan. 83–100. in Japanese.

[Kwasny and Sondheimer, 1981] Kwasny, S.C and Sondheimer, N.K. 1981. Relaxation techniques for parsing grammatically ill-formed input in natural language understanding systems. *American Journal of Computational Linguistics* 7(2):99–108.

[Meknavin *et al.*, 1993] Meknavin, Surapant; Theeramunkong, Thanaruk; and Hozumi, Tanaka 1993. Parsing ill-formed input with id/ip rules. In *The Fourth International Workshop on Natural Language Understanding and Logic Programming(NLULP4)*.

[Mellish, 1989] Mellish, Chris S. 1989. Some chart-based techniques for parsing ill-formed input. In *Proceeding of 27th Annual Meeting of the ACL*. 102–109.

[Panupong, 1984] Panupong, Vichin 1984. *The Structure of Thai: Grammatical System*. Ramkamhaeng University. in Thai language.

[Schank *et al.*, 1980] Schank, R.C; Leboeitz, M.; and Birnbaum, L. 1980. An integrated understander. *American Journal of Computational Linguistics* 6(1):13–30.

[Simpkins, 1990] Simpkins, Neil K. 1990. Chart parsing in prolog. *New Generation Computering, An international Journal* 8(2):113–138.

[Trehan and P.F., 1988] Trehan, R and P.F., Wilk 1988. A parallel chart parser for the committed choice non-deterministic logic languages. In *Logic Programming 1, Proceedings of 5th International Conference ans Symposium*. MIT Press, Cambridge, Mass. 212–232.

[Weischedel and Black, 1980] Weischedel, R.M. and Black, J.E. 1980. Responding intelligently to unparsable inputs. *American Journal of Computational Linguistics* 6(2):97–109.

[Weischedel and Sondheimer, 1982] Weischedel, Ralph M. and Sondheimer, Norman K. 1982. An improved heuristics for ellipsis processing. In *Proceeding of 20th Annual Meeting of the ACL*, Totont, Canada. 85–88.

[Winograd, 1983] Winograd, T. 1983. *Language as a Cognitive Process*, volume 1:Syntax. Addison-Wesley. chapter 3, 116–128.