Meta-rules as a Basis for Processing Ill-Formed Input[1]

Ralph M. Weischedel[2]

Department of Computer & Information Sciences
University of Delaware
Newark, DE 19716

Norman K. Sondheimer

USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292

If natural language processing systems are ever to achieve natural, cooperative behavior, they must be able to process input that is ill-formed lexically, syntactically, semantically, or pragmatically. Systems must be able to partially understand, or at least give specific, appropriate error messages, when input does not correspond to their model of language and of context.

We propose meta-rules and a control structure under which they are invoked as a framework for processing ill-formed input. 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, if at all.

Examples discussed in the paper include violated grammatical tests, omitted articles, homonyms, spelling/typographical errors, unknown words, violated selection restrictions, personification, and metonymy. An implementation of a meta-rule processor within the framework of an augmented transition network parser is also described.

1. Introduction

Natural language understanding systems have improved markedly in recent years, and natural language interfaces have even begun to enter the commercial marketplace, for example, the INTELLECT system of Artificial Intelligence Corporation (Harris 1978). These systems promise to make major improvements in the ease of use of data base management and other computer systems. However, they have only begun to consider the problems of truly natural input. The emphasis has been, and continues to be, on the understanding of well-formed inputs. True natural language input is often ill-formed *in the absolute sense* of being filled with misspellings, mistypings, mispunctuations, tense and number errors, word order problems, run-on sentences, extraneous forms, meaningless sentences, and impossible requests. In addition, natural input is ill-formed *in the relative sense* of containing requests that are beyond the limits of either the computer system or the natural language interface.

Evidence indicates that absolutely ill-formed input regularly occurs in a data base query environment. For instance, in an extensive study (Thompson 1980) including 1615 inputs, only 1093 were parsable, and an overall total of 446 contained various kinds of errors: 161 with vocabulary problems, 72 with punctuation errors, 62 with ungrammaticality, and 61 with

[1] This material is based upon work supported in part by the National Science Foundation under Grant Nos. IST-8009673 and IST-8311400 and in part by the Defense Advanced Research Projects Agency under Contract No. MDA 903-81-C-0335, ARPA Order No. 2223. Views and conclusions contained in this paper are the authors' and should not be interpreted as representing the official opinion or policy of DARPA, the U.S. Government, or any person or agency connected with them.

[2] Currently visiting at the Department of Computer & Information Science, University of Pennsylvania, Philadelphia, PA 19104.

0362-613X/83/030161-17$03.00

spelling errors. Furthermore, 211 inputs were fragmentary, including 91 parsed terse replies and 67 parsed terse questions.

In another experiment (Eastman and McLean 1981), 693 English queries to a data base system were analyzed. Co-occurrence violations, including subject-verb disagreement, tense errors, apostrophe problems, and possessive/plural errors occurred in 12.3% of the queries. Omitted words, extraneous words/phrases, telegraphic ellipsis, and incomplete sentences arose in 14% of the queries.

Our conjecture is that wherever natural language interfaces are employed, ill-formed input will occur. Any natural language interface, when faced with ill-formed input, must either intelligently guess at a user's intent, request direct clarification, or at the very least accurately identify the ill-formedness. As Wilks (1976) states, "Understanding requires, at the very least, ... some attempt to interpret, rather than merely reject, what seems to be ill-formed utterances."

Though experience has shown that users can adapt to the limitations of the system's well-formed anticipated input (Harris 1977b, Hendrix et al. 1978), we feel that relying on such user adaptation ignores one of the most powerful motivations for English input: enabling infrequent users to access data without an intermediary and without extensive practice. Even the person who frequently uses such a system will be exasperated if it cannot explain why it misunderstands an input.

In some circumstances, ill-formed input may be less frequent. For instance, Fineman (1983) reports that in an experiment where users were constrained to *discrete speech,* ill-formedness occurred in as little as 2% of the input. Another unusual environment can be created by informing the users that the system cannot really understand natural language, thus biasing language use.

In addition to natural language interfaces, processing ill-formed input is critical to correcting language use. Prototype systems have already been investigated in the language-learning environment (Weischedel et al. 1978) and in the office automation environment of document preparation (Miller et al. 1981). Even in published text unintentional ungrammaticalities occur.

Most natural language understanding systems deal with a few types of ill-formedness. Out of our own work, and that of others, we have produced a framework for processing ill-formed input. This approach treats ill-formedness as *rule-based.* First, natural language interfaces should process all input as presumably well-formed until the *rules* of normal processing are violated. At that point, error handling procedures based on *meta-rules* relating ill-formed input to well-formed structures through the modification of the violated normal rules should be employed. These meta-rules correspond to types of errors.[3]

The rest of the paper argues for this rule-based approach. Section 2 characterizes both the types of ill-formed input, and the types of possible approaches to them, including our proposal. Section 3 gives examples of meta-rules for processing ill-formed input. Section 4 describes how some heuristics developed by others fit within our paradigm. An implementation is sketched in Section 5. Section 6 discusses limitations of the proposal. Sections 7 and 8 present directions for future work and conclusions.

2. Approaches to Ill-formedness

This section introduces the problem of interpreting ill-formed input. First, we discuss the types of ill-formed input briefly. Then we consider the range of approaches that have been tried for allowing for such input.

Ill-formedness phenomena can be divided into two sets. The first defines what we call *absolute ill-formedness.* An utterance is absolutely ill-formed if the typical listener considers it ill-formed. The definition unfortunately appeals to subjective evaluations; these are known to differ widely (Ross 1979). But it seems to include the majority of typical cases and exclude the majority of types of good English sentences.

The second set defines *relative ill-formedness.* This is ill-formedness with respect to the normal processing rules of the formal computing system including the natural language interface and the underlying application system. The set of ill-formed inputs for an interface can be defined as the union of these two sets for that interface.

The set of ill-formed input captured by these definitions can also be seen through the four typical phases of interpretation in natural language interfaces: lexical, syntactic, semantic, and pragmatic processing. In lexical processing, absolute ill-formedness can come from misspelling, mistyping, and mispronunciation; relative ill-formedness can arise from unknown words. In syntactic processing, absolute ill-formedness is seen in faulty subject-verb agreement, word order errors, omitted words, run-on sentences, etc; relative ill-formedness is seen in grammatical combinations of words that exceed the interface's grammar.

Semantic processing can be defined as the interpretation of the input in isolation. Knowledge of the task domain can be applied, but the context of input with respect to previous interactions and the state of the underlying computing system are only considered in pragmatic processing.

Absolute ill-formedness in semantics includes omitting needed information and violating of selectional restrictions. Absolute ill-formedness in pragmatics

[3] The purpose of these meta-rules is therefore quite distinct from that of Gawron (1982).

includes breaking the rules of conversation, as when answering a question with a question, having presuppositions of the speaker fail, and failing to make clear an anaphoric reference. Relative ill-formedness in both cases includes "overshoot", requesting capabilities or information not covered by the system in its current state, and parenthetical expressions incomprehensible to the system.

2.1. Four alternative approaches to ill-formedness

There are at least five approaches one can take to ill-formedness. This section outlines the four alternatives to the approach we have formulated; our approach is covered in Section 2.2. In describing the five approaches, we use the following informal notation. SYSTEM[s] refers to a system designed to process a set of sentences s. WELL-FORMED is a set of well-formed utterances; ILL-FORMED is a set of ill-formed utterances. Naturally, an approach that covers the broadest range of linguistic behaviour should be preferred.

One alternative is to treat the processing of ill-formed and well-formed inputs identically, by ignoring constraints. That is, one designs SYSTEM[WELL-FORMED U ILL-FORMED]. Schank et al. (1980) and Waltz (1978) have taken this approach toward grammatical constraints. CASPAR (Hayes and Carbonell 1981) exhibits this approach for grammatical constraints as well. Since there is much redundancy in language, the practice of not using certain constraints will often work. However, this will fail on many utterances, since it ignores rules that not only constrain search but also eliminate unintended interpretations. One can see this by considering subject-verb agreement, a grammatical constraint that people sometimes violate and that is often left out of natural language systems. Though other constraints, such as semantic (selection) restrictions between a verb and its subject, often indicate the intended interpretation, it is easy to think of examples where subject-verb agreement is crucial to understanding. Comparing examples (1) and (2) below, subject-verb agreement is crucial to determining whether the company or assets were purchased.

(1) List the assets of the company that was purchased by XYZ Corp.

(2) List the assets of the company that were purchased by XYZ Corp.

A second approach is to build systems for well-formed input and for ill-formed input together; that is, one designs SYSTEM[ILL-FORMED] merged with SYSTEM[WELL-FORMED]. Unlike the first approach, well-formedness constraints are employed on well-formed input. LUNAR (Woods et al. 1972), an early English interface to a question-answering system, and SOPHIE (Burton and Brown 1977), an intelligent tutoring system with an English interface, both used this approach. The problem with this approach is that it does not reflect the fact that constraints indicate preferences in interpretation. For instance, though example (3) below has two legitimate syntactic interpretations, the one that violates our model of the world is rejected, causing us to reject the "garden path" interpretation.

(3) I saw the Statue of Liberty flying to New York.

As another example, the two pronouns in "He shot him" are normally considered to refer to different people; the alternative that the speaker meant "He shot himself" does not arise unless there are strong expectations ahead of time that that is the correct proposition.

A third approach is to build two systems, but to use SYSTEM[ILL-FORMED] only if SYSTEM[WELL-FORMED] finds no interpretation. A commercially available English interface to data bases (Harris 1977) has taken this approach. The EPISTLE project (Jensen and Heidorn 1983, Miller et al. 1981) employs this alternative for grammatical violations. DYPAR (Carbonell et al. 1983) has taken this approach in an interface to an expert system. Kaplan (1978) developed a strategy to give more useful responses when a data base query yields a negative response, for example, when no entity satisfies the desired conditions. Chang (1978) created a heuristic for inferring missing joins in incomplete queries to relational data bases. The defect in this model is that there is no means of relating strategies for processing ill-formedness explicitly to the strategies for processing well-formedness. We argue here that one can explicitly relate the two classes of strategies.

A fourth approach is to build only one system, SYSTEM[WELL-FORMED], and to employ a metric to measure how far a postulated interpretation is from satisfying all well-formedness constraints. Charniak (1981) has advocated this for grammatical processing; Wilks (1975) has made this the basis of semantic processing during the interpretation phase. Of course, the notion of weighing alternatives and using metrics has been used for phenomena other than ill-formedness, such as parsing (Robinson 1982) and speech understanding (Walker 1978, Woods et al. 1976). Clearly, ranking alternative interpretations is necessary. However, if one relies *solely* on a metric and SYSTEM[WELL-FORMED], then an account of the fact that the ill-formedness often has specific implications is still needed. In example (4), the selection restriction that "like" requires animate agents is violated; a reasonable inference is that the speaker somewhat personifies the computer in question.

(4) My home computer doesn't like to run BASIC.

Nor does a metric reflect the fact that there are clear patterns of error, such as those that have been reported in linguistic studies (Fromkin 1973) and in application studies (Thompson 1980, Eastman and McLean 1981).

Table I summarizes these four approaches.

2.2. Our approach

Based on previous work, both our own and that of others, we propose a framework employing meta-rules to relate the processing of ill-formed input to well-formedness rules. This framework may be stated as follows:

1. Process the input using SYSTEM[WELL-FORMED].
2. If no interpretation is found by SYSTEM [WELL-FORMED], apply a meta-rule to the well-formedness rules, based on a ranking of the alternatives, in order to
   a) diagnose the problem, that is, the rule that is violated and how it is violated,
   b) relax the rule,
   c) add a "deviance note" to the interpretation recording the violation,
   d) resume processing via the well-formedness rules, if possible.
3. Repeat step 2 as necessary.

Each meta-rule should correspond to a pattern of ill-formedness and should account for utterances corresponding to only that pattern. SYSTEM[ILL-FORMED] is therefore implicit in the meta-rules.

This framework has advantages lacking in one or more of the other approaches. Well-formedness constraints, whether syntactic, semantic, or pragmatic, are employed to eliminate unintended interpretations. Well-formed interpretations are always preferred. Ill-formedness processing is explicitly related to the well-formedness rules. Only the constraint that seems to be violated is relaxed; all other well-formedness constraints are still effective. Furthermore, the deviance notes record the aspect that deviates from well-formedness, thus allowing pragmatic inferences by later processing.

In the next two sections, we propose a handful of primitives for syntactic and semantic problems and also propose a formalism for writing meta-rules. As supporting evidence, we state meta-rules for a number of problems and describe approaches for several others. These phenomena include the following:

failed grammatical tests,
word confusions,
spelling errors,
unknown words,
restarted sentences,
resumptive pronouns and noun phrases,
contextual ellipses,
selection restriction violation,
metonymy,
personification, and
presupposition failure.

3. Examples of Meta-rules

To further argue for meta-rules as a uniform framework for processing ill-formed input, we describe a wide variety of meta-rules in this section and the next. We adopt the following notation for meta-rules in this paper:

Approach 1	
Characterization:	Do not encode well-formedness constraints.
Flaw:	Well-formedness rules convey meanings by constraining interpretations.
Approach 2	
Characterization:	Design systems for well-formed input and ill-formed input together.
Flaw:	This gives no preference of well-formed interpretations over ill-formed ones.
Approach 3	
Characterization:	Search for well-formed interpretations prior to considering any ill-formed ones.
Flaw:	This does not explicitly relate handling ill-formedness to processing well-formedness.
Approach 4	
Characterization:	Use a metric to rank ill-formedness interpretations, and select the one that comes closest to satisfying all constraints.
Flaw:	This does not state what the deviation is so that one may draw inferences from the ill-formedness, nor does it capture actual patterns of error.

Table 1. Four Rejected Approaches.

C1 C2 ... Cn --> A1 A2 ... Am

The left-hand side (LHS) diagnoses what the problem might be; the right-hand side (RHS) states how to relax the failed constraint. The Ci are conditions on the computational state of the system; all must be true if the meta-rule is to apply. The Ai are actions stating how to rewrite the violated constraint and resume processing; all will be executed if the rule applies. Many of the actions we suggest here can be viewed as rewriting a rule of the normative system, for example, a grammar rule or case frame. However, some are more appropriately viewed as changing the computational state when blockage occurs; an example is correcting the spelling of a word. In Section 5 we will argue that it is best to implement all of the actions as modifications of a blocked alternative.

Naturally, in rewriting a rule, pattern-matching and substitution are fundamental. We adopt the following definition of patterns. A pattern is a LISP s-expression. Atoms preceded by a question mark are variables. Expressions preceded by a dollar sign are evaluated using the LISP rules; their values are treated as patterns.[4] If a period appears before a pattern variable that is the last item in a list, that pattern variable matches the tail of a list. All other items in patterns are literals. The scope of a pattern variable is the whole meta-rule. The first time a variable is bound in a meta-rule, it retains the binding throughout the rule.

Potentially, there may be many places where relaxation can occur. If a meta-rule applies to more than one configuration, it will be applied to each in turn, creating a list of possibilities for processing after recovery is complete. Consequently, the meta-rules will refer to only one failed configuration at a time.

3.1. Meta-rules related to syntax

First, let us consider meta-rules dealing with the grammar. Many of our examples here are reformulations of our earlier work (Weischedel et al. 1978, Kwasny and Sondheimer 1979, Weischedel and Black 1980) within the uniform framework of meta-rules. All meta-rules pertaining to syntax should have a first condition which is (SYNTAX-FAILED?); this is true iff the parser is blocked. Since all rules in this section would contain that predicate, we will not include it in the examples.

Many syntactic formalisms have a similar framework for expressing rules: these include context-free grammars, augmented phrase structure grammars (Heidorn 1975), Programmar (Winograd 1972), the linguistic string parser (Sager 1981), Lifer (Hendrix et al. 1978), and augmented transition networks (ATNs) (Woods 1970). In fact, all of these can be viewed as formally equivalent to ATNs, and we will describe our techniques in that framework.

Figure 1 gives several predicates that should be useful in the LHS of meta-rules. The LHS of the meta-rule is matched against the environment in which an ATN arc failed. The environment is called a configuration and includes the current ATN state, the arc, all ATN registers, and the remainder of the input string.

(IN-STATE? state)	Did the configuration block in state?
(CAT? category)	Is the current word in category?
(WRD? list)	Is the current word a member of list?
(NEXTCAT? category)	Is the word after the current one in category?
(NEXTWRD? list)	Is the word after the current one in list?
(FAILED-TEST? pattern)	Is the pattern a predicate expression in the test of the arc and did both the expression and the test evaluate to false?
(FAILED-ARC? pattern)	Does the failed arc match pattern?
(HOLDLIST-NOT-EMPTY?)	Is the hold list empty?
(CONFUSION-WORD? x)	Is x a word frequently confused with another? If so, the related word is returned.

Figure 1. Useful Conditions for Syntactic Meta-rules.

[4] The expression $expr could be implemented as (*EVAL* expr). The pattern variable ?atom could be implemented as (*VAR* atom).

(EMPTY-HOLD) defines the hold list to be empty.

(FAILED-CONSTRAINT pattern) adds the instantiation of the pattern to a list of violated constraints stored in the configuration. The position of the parser within the input string will automatically be recorded as well.

(SUBSTITUTE-IN-ARC pattern1
pattern2) changes the arc in the failed configuration by replacing all expressions matching pattern1 by pattern2.

(REPLACE-* x) makes x the current word in the blocked configuration.

Figure 2. Useful Actions for Syntactic Meta-rules.

```
State       Arcs                                                                  (Figure 3.)

S/          (PUSH NP/ T ...
                (SETR SURFACE-SUBJECT *)
                (* We think this is the subject)
                (TO S/NP))
            (VIR NP T (SETR SURFACE-SUBJECT *)
                (* In relative clauses, this identifies a noun phrase from the hold list as surface subject)
                (TO S/NP))

S/NP        (CAT VERB (SUBJECT-VERB-AGREE? (GETR SURFACE-SUBJECT) *)
                (* The predicate SUBJECT-VERB-AGREE? is true iff the number and person of the subject and verb are
                   compatible)
                ... (SETR VERB *)
                (TO S/V))

S/V         (JUMP S/POP (INTRANSITIVE-VERB? (GETR VERB)))
            (CAT ADV T ... (TO S/V))
            (PUSH NP T (SETR OBJECT *)... (TO S/POP))
            (VIR NP T (SETR OBJECT *)
                (* Identifies a noun phrase from the hold list as the direct object in relative clauses)
                (TO S/POP))

S/POP       (POP (BUILDQ ...)
                (AND (TRANSITIVE-VERB? (GETR VERB))
                    (NOT (REQUIRES-INDIRECT-OBJ? (GETR VERB)))))

NP/         (CAT PRO T ...
                (SETR PRO *)
                (TO NP/POP))
            (CAT DET T ...
                (SETR DET *)
                (SETR NUMBER (GETNUMBER DET))
                (TO NP/DET))

NP/DET      (CAT ADJ T ...
                (* collecting adjectives before head noun)
                (TO NP/DET))
            (CAT N T ...
                (SETR N *)
                (TO NP/N))
```

```
NP/N      (CAT N T ...
              (ADDR COMPOUND (GETR N)) (SETR N *)
              (* a possible nominal compound)
              (TO NP/N)
          (JUMP NP/POP
              (DET&NOUN-AGREE? (GETR NUMBER) (GETR N))
              (* The predicate DET&NOUN-AGREE? is true iff the determiner used is incompatible with the head noun)
                ... )
          (PUSH RELCL/ T
              (SENDR TRACE (TRACE))
              (* This sends a trace of a noun phrase to a relative clause)
              (SETR RELATIVE-CLAUSE *) (TO NP/POP))

NP/POP    (POP (BUILDQ ...) T)

RELCL/    (CAT RELPRO T (HOLD (GETR TRACE)) ...
              (TO S/))
          (PUSH NP T (SETR SURFACE-SUBJECT *)
              (HOLD (GETR TRACE))
              (* This allows for relative clauses where the subject is present)
              (TO S/NP))
          (JUMP S/NP (SETR SURFACE-SUBJECT (GETR TRACE))
              (* Allows for elided subjects in reduced
                   relative clauses))
```

Figure 3. A Simple ATN Graph.

An important action for the RHS is NEW-CONFIGURATION, which defines a new parser configuration, thus replacing the failed configuration that the meta-rule matched. It may take any number of arguments which set parts of the configuration. For example, SETR will define the new value of an ATN register. A list of useful arguments to NEW-CONFIGURATION is given in Figure 2. Failed constraints fill the role of the deviance notes of Kwasny and Sondheimer (1979). All parts of the failed configuration that are not explicitly changed in NEW-CONFIGURATION remain the same. Our implementation assumes that there is only one NEW-CONFIGURATION per meta-rule, though one could generalize this so that executing NEW-CONFIGURATION n times in a meta-rule gives n new configurations to replace the failed one. If a new configuration is generated, the parse can be resumed.

Figure 3 gives a trivial ATN which will be used for the sample meta-rules. The start state is S/. A list beginning with an asterisk in the actions of an arc is a comment.

3.1.1. Simple grammatical tests

Our earlier work showed how to relax tests that appear on ATN arcs. In one study (Eastman and McLean 1981), subject-verb disagreement occurred in 2.3% of the English queries. Meta-rule (i) relaxes that agreement test. The new configurations here are the result of replacing the agreement test in each failed arc by the predicate T. Since a new configuration is generated, parsing is resumed using it. Though the substitution was trivial in this case, SUBSTITUTE-IN-ARC is a general pattern-matching and substitution facility. As an example, consider "A curious problem showing unusual conditions appear ..." A top-down, left-to-right parse using a grammar such as the one in Figure 3 would block at the word "appear". One of the blocked configurations would correspond to the agreement test failure in the arc leaving state S/NP; meta-rule (i) would apply, allowing the sentence to be parsed.

```
(i)   (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))
```

3.1.2. Omitted articles

Another frequently occurring problem is omitting required articles from count nouns. In the study by Eastman and McLean (1981) this occurred in 3.3% of all queries. In the grammar of Figure 3, blockage would occur at NP/N because of the test DET&NOUN-AGREE? on an example such as "Print price of P27 over the last five years". Rule (ii) relaxes the test. When the parser starts on the new configuration, the modified test will be checked, verifying that no determiner is present. If none is, the message from FAILED-CONSTRAINT is available for error recovery.

The meta-rule approach allows for more sophisticated actions. Suppose that a linguistic study of utterances with missing determiners showed that a default assumption of definite reference is a good heuristic. In this case, one could simply add the action (SETR DET 'the) to the actions in NEW-CONFIGURATION.

One could argue that, in a data base environment, the grammar should simply treat omitted determiners as a normative construction. Even though determiners are frequently omitted in data base contexts, preferring well-formed interpretations can eliminate some ambiguities in complex noun phrases such as "a machine running programs". The determiner constraint suggests that "running programs" modifies the head noun "machine" rather than "machine" and "running" both modifying "programs".

3.1.3. Confusion words

A number of word pairs are frequently confused, such as homonyms and "good" for "well". Meta-rule (iii) allows for such errors, since REPLACE-* will modify the current word in the blocked configuration. MR-SETQ binds the value of its second argument to the pattern variable appearing as its first argument. Hence, "You performed good" would block at S/V, and the meta-rule would substitute "well" for "good".

3.1.4. Resumptive pronouns

Another kind of ill-formedness is resumptive pronouns and resumptive noun phrases. These occur in relative clauses where the entity referred to by the relative pronoun is improperly repeated in the relative clause as a pronoun or noun phrase. An example is "John's friend Mary married the man that she planned to marry him", since there is no syntactic slot in the relative clause for the relative pronoun "that" to fill. A typical ATN strategy for interpreting relative clauses is to put a place holder or trace on a "hold list"; the ATN processor prevents POPping from a level if the hold list is non-empty. That test prevents accepting clauses where traces are not used. Meta-rule (iv) provides for resumptive pronouns and resumptive noun phrases. One can imagine more complicated tests, since there are specific conditions (Kroch 1981) under which resumptive pronouns and resumptive noun phrases are more likely.

```
(ii)   (FAILED-TEST? (DET&NOUN-AGREE? ?X ?Y))
       --> (NEW-CONFIGURATION
              (SUBSTITUTE-IN-ARC (DET&NOUN-AGREE? ?X ?Y) (NULL ?X))
              (FAILED-CONSTRAINT (DETERMINER&NOUN ?Y -- MISSING DETERMINER)))

(iii)  (MR-SETQ ?X (CONFUSION-WRD *))
       ---> (NEW-CONFIGURATION
              (REPLACE-* ?X)
              (FAILED-CONSTRAINT (?X SUBSTITUTED FOR *)))

(iv)   (FAILED-ARC? (POP ?VALUE . ?Z))
       (IN-STATE? S/POP)
       (HOLDLIST-NOT-EMPTY?)
       ---> (NEW-CONFIGURATION
              (FAILED-CONSTRAINT (Resumptive Clause $?VALUE))
              (EMPTY-HOLD))

(v)    (IN-STATE? S/POP)
       --> (PRINT-RESPONSE-PATTERN
              (I DO NOT UNDERSTAND YOUR USE OF THE VERB $(GETR VERB) /.  WOULD YOU LIKE
              EXAMPLES OF WHAT I UNDERSTAND?))
           (SELECTQ (READ)
              ((YES Y) (PRINT-EXAMPLES (GETR VERB))) NIL)
```

(FAILED-SEMANTIC-TEST? pattern)	Is the pattern a predicate expression (on a constituent) and is the predicate false?
(MANDATORY-CASE-MISSING?)	Is a required case absent?
(TOO-MANY-FILLERS?)	In trying to fill case, does it already have the maximum number assigned?
(SEMANTIC-CLASS? class)	Is class a semantic class predicate, for example, human?
(MATRIX-TYPE? class)	Is the matrix to which we are trying to assign the current constituent in class?
(VIEWABLE? x y z)	Can entity x be viewed as a y in the context z? (This is a question for the pragmatic component.)
(CASE? case)	Is the constituent supposed to fill role case?

Figure 4. Predicates for Semantic Meta-rules.

(FAILED-CONSTRAINT pattern)	adds the instantiation of the pattern to a list of violated constraints stored in the configuration.
(SUBSTITUTE-IN-CASE pattern1 pattern2)	replaces pattern1 by pattern2 everywhere in the constraint.
(SUBSTITUTE-FOR-CASE case)	tries assigning the constituent as case.

Figure 5. Actions for Semantic Meta-rules.

3.1.5. Error messages

Of course, the parser may not be able to recover at all due to either absolute or relative ill-formedness. Weischedel and Black (1980) presented a technique for associating error messages with states where the parser blocked. The only way to block in S/POP is if the verb complement expected for the main verb is not present. Meta-rule (v) could handle this simple case. Notice that this is a different class of meta-rule, for it does not resume computation. Naturally, such rules should be tried only after no other meta-rules are available. One could define different classes of meta-rules by appropriate declarations; alternatively, this class can be recognized easily, since none of the actions resume processing.

This is not the only alternative in the face of failure to parse even with relaxation; Jensen and Heidorn (1983) present heuristics for what to pass to the semantic interpreter in this case, given bottom-up parsing.

3.2. Meta-rules related to semantics

In addition to these syntactic examples, semantic problems can also be addressed within the formalism. If some semantic tests are included in the parser, say a certain arc test contains calls on the semantic component, specific semantic tests can be relaxed by the general mechanism we described for relaxing tests on ATN arcs.

Instead, suppose that semantic constraints are encoded in a separate component. Semantic constraints may be expressed in several formalisms, such as semantic nets (Bobrow and Webber 1980a,b; Sondheimer et al. 1984), first-order logic, and production rules (for example, PROLOG, Warren et al. 1977). It is generally agreed that all are formally equivalent to first-order logic. For the purposes of this paper, we assume that the selection restrictions are encoded in first-order logic.

One of the most common designs for a semantic interpreter is based on selection restrictions and case frames (Bruce 1975). At least five kinds of constraints may be violated:

1) what may fill a given case,
2) which cases are required for a complete constituent,
3) which may have multiple fillers without conjunction,
4) which are allowed for a given case frame, and
5) what order cases may appear in.

Figure 4 lists tests useful for diagnosing failures in such a semantic interpreter. Assume that any predicate on the semantic class of a constituent is encoded simply in LISP notation, for example, (HUMAN x) is true iff x is of class human. All meta-rules in this section can be assumed to include an initial test (SEMANTICS-FAILED?.)

For convenience, we have used the same names for some of the actions as in the syntactic cases (for example, FAILED-CONSTRAINT, NEW-CONFIGURATION, etc.). When implemented in a particular system, different names may be used, since the concept of configuration, blockage, etc., is usually different for the types of processing (for example, lexical, syntactic, semantic, and pragmatic). Figure 5 lists several actions useful in semantic meta-rules.

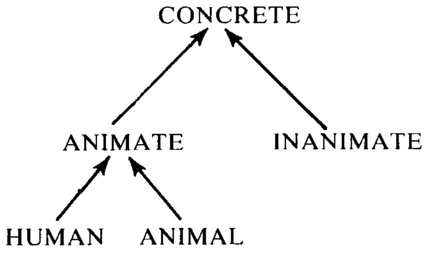

Figure 6. A Fragment of a Semantic Hierarchy.

3.2.1. Universal relaxation of semantic class

Meta-rule (vi) is a very general rule. Assuming that semantic class tests are organized in a hierarchy, it states that the failed test is to be replaced by its parent in the hierarchy, yielding the next most general test.

An example of the use of meta-rule (vi) is "My car drinks gasoline". The restriction on the AGENT case could be the predicate ANIMATE. A fragment of a semantic hierarchy appears in Figure 6. In that, ANIMATE has a parent predicate of CONCRETE that would include cars. The failure of the initial sentence and the subsequent processing using the meta-rule would accept a sentence with the special deviance note identifying the semantic oddity.

3.2.2. Personification

In a way similar to our arguments against approach 4 in Section 2.1, we feel that general meta-rules such as (vi) will prove less valuable than specific rules. A particular test that can be relaxed is the requirement for a human; for instance, the verbs of saying and those of propositional attitude, such as "believe" and "think", normally have a restriction that their agent be human. Nevertheless, such a constraint is regularly violated through personification of pets, higher animals, machines, etc.

Since personification is infrequent compared to the norm of descriptions designating humans, a case constraint of "human" can trim the search space. Since personification conveys particular inferences (Lakoff and Johnson 1980), a relaxation rule that records the detected personification can trigger appropriate inference processes. Figures of speech certainly are not absolutely ill-formed; we argue here that it is useful to treat them as relatively ill-formed.

Meta-rule (vii) is one simple relaxation for personifying animals. More specific ones may prove preferable, if classes of personification are taxonomized.

3.2.3. Metonymy

There are at least seven classes of metonymy (Lakoff and Johnson 1980), including a part for the whole, the producer for the product, the object for its user, the controller for the controlled entity, the institution for the people responsible, the place for the institution, and the place for the event. This analysis suggests two kinds of strategies. A particular class of descriptions may occur in exactly the same linguistic environments as their class of metonymous descriptions. For instance, institutions and people appear interchangeable as the logical subject of the verbs of saying and of propositional attitude. That can be encoded directly in the case frames of those verbs.

However, many types of metonymy are conditioned on a highly specialized relationship. For instance, places can be used metonymously for events only if the speaker believes an event is identifiably associated with the location. For instance, compare the following examples:

Pearl Harbor caused us to enter the war.

*Fifth and Lombard caused us to reconsider graduated income taxes.

Therefore, a meta-rule such as (viii) seems appropriate to prefer the normal, but accept metonymous descriptions of events by places. In meta-rule (viii), we have assumed that there is a variable FILLER of the semantic interpreter that holds the constituent to be assigned. Also, in the call to VIEWABLE?, CURRENT indicates that the pragmatic component should use its current context.

3.2.4. Phrase ordering

Failure in selectional restrictions can indicate other semantic errors. These include ordering problems, for example, "John killed with a gun Mary", and unexpected prepositions, for example, "John killed Mary by a gun". The LHS of the appropriate meta-rules would begin with identification of selectional restriction failures but would also include other tests. The RHS would change the assumed case. A rule for the first example is (ix). Here the assumption is that the ordering problem will be first noted when "Mary" is tried as a time modifier. Using SUBSTITUTE-FOR-CASE postulates the constituent "Mary" to fill the object case and attempts to do so.

```
(vi)   (FAILED-SEMANTIC-TEST? (?Y ?Z))
       (SEMANTIC-CLASS? ?Y)
       --> (NEW-CONFIGURATION
              (FAILED-CONSTRAINT (?Y TOO RESTRICTIVE -- USING PARENT $(PARENT-OF ?Y)))
              (SUBSTITUTE-IN-CASE (?Y ?Z) ($(PARENT-OF ?Y) ?Z)))

(vii)  (FAILED-SEMANTIC-TEST? (HUMAN ?Z))
       --> (NEW-CONFIGURATION
              (FAILED-CONSTRAINT
                 ((HUMAN ?Z) -- ASSUMING PERSONIFICATION OF ANIMAL))
                 (SUBSTITUTE-IN-CASE (HUMAN ?Z)

(viii) (FAILED-SEMANTIC-TEST? (EVENT ?X)) (LOCATION FILLER) (VIEWABLE? FILLER 'EVENT 'CURRENT)
       --> (NEW-CONFIGURATION
              (SUBSTITUTE-IN-CASE (EVENT ?X) T) (FAILED-CONFIGURATION
                 (METONYMY PLACE-FOR-EVENT FILLER)))

(ix)   (FAILED-SEMANTIC-TEST? (TIME ?Z)) (MATRIX-TYPE? 'CLAUSE) (CASE? 'TEMPORAL)
       --> (NEW-CONFIGURATION (FAILED-CONSTRAINT (ORDERING PROBLEM -- OBJECT CASE AS-
           SUMED))
```

3.3. Generality of the approach

Though we have experience in implementing our framework for ATN parsers only, we believe the framework to be applicable over a broad range of parsers. It assumes only that a "configuration" or "alternative" representing a blocked, partial interpretation can be stored, modified, and restarted. No assumption regarding the direction of processing (for example, left to right), the nature of search (for example, top-down vs. bottom-up), nor the class of problem (for example, lexical, syntactic, or semantic) is made. For instance, the design of an implementation for semantic meta-rules as in Section 3.2 is complete. The underlying semantic component is based on searching case frames breadth-first with both top-down and bottom-up characteristics. Except for the one meta-rule regarding incorrect phrase ordering in Section 3.2.4, the semantic meta-rules themselves are independent of whether proposing a phrase for a given case in a frame is based on syntactic considerations or other criteria (for example, Schank et al. 1980). Naturally, the primitive conditions and actions of a given set of rules will depend on a particular formalism. In the next section, we relate our framework to a variety of parsers and problems.

4. Additional Supporting Evidence

Many natural language interfaces have some heuristics for processing one or more classes of ill-formed input. Since an exhaustive analysis would be impossible here, we will review only a handful of techniques that have inspired us to develop the meta-rule framework. We describe each technique by showing how it could be phrased as a meta-rule within our paradigm.

The LADDER system (Hendrix et al. 1978) implements three major techniques for processing ill-formed input. All fit within the framework we suggest. One deals with recovery from lexical processing. In this system, the developer of a question-answering system prepares only a dictionary of well-formed words. If a sentence contains a word that is not in the dictionary, the parser will fail. The system localizes the area of failure to the ATN state associated with the partial interpretation that has proceeded rightmost in the input and that is shallowest (in terms of incompleted ATN PUSH arcs). Candidates for the correct spelling are limited to the words that would permit the parser to proceed and that are close to the spelling that appears. An equivalent meta-rule would check in the LHS that the parser failed. The RHS would compute a list of words expected next for each type of arc leaving that state, for example, the category members and literal words expected next. The next action would apply the Interlisp spelling correction algorithm to postulate a known word that was expected next. This word would replace the unrecognized one in the input and parsing would resume. A similar heuristic is running in our current implementation, with the addition that, if the unrecognized word appears to have an inflected ending, spelling correction is performed on the possible root.

A second technique in LADDER deals with understanding contextual ellipsis, if no parse for the input is found. This heuristic interprets "the fastest submarine" as "To what country does the fastest submarine belong", if it occurs after a query such as "To what country does each merchant ship in the North Atlantic belong". In Weischedel and Sondheimer (1982), we extended that heuristic to allow for turn-taking in dialogues and to allow expansions as well as substitutions, such as the elliptical form "Last month" following "Did you go to Chicago?"

A third technique in LADDER is the printing of error messages, in the same sense that meta-rule (v) above prints a message when all attempts have failed. We could phrase this heuristic as a meta-rule whose LHS would check that the parser has blocked. This meta-rule would be ordered strictly after the ones for spelling correction and contextual ellipsis. A state would be postulated as the locale of the problem by the same heuristic as for spelling correction. The RHS would print for each arc that leaves that state the category, constituent, or word that was expected by that arc.

Hayes and Mouradian (1980) emphasize recovery techniques for blocking during left-corner parsing (Aho and Ullman 1972). Their strategies are invoked only if the parser blocks. Two of them can be reformulated as meta-rules as follows. One meta-rule would check in its LHS that the parser was blocked and that a special parser variable (call it BLOCKED-PARSE) was empty. The RHS would save the blocked configuration in BLOCKED-PARSE, and start parsing as if the current word were the first word of the input. This would enable the system to ignore initial strings that could not be understood. A useful example of this is restarted inputs, such as "Copy all print all headers of messages". A second meta-rule is related. The LHS would check whether the parser was blocked and BLOCKED-PARSE had a configuration in it. Furthermore, the LHS would check to see that another parser variable (call it DONE-ONCE) was NIL. If so, the RHS would set DONE-ONCE to T. The RHS would then swap the current configuration with BLOCKED-PARSE and would try resuming the parse from the current word with that configuration. This heuristic is designed to ignore incomprehensible material in the middle of an input. For instance, it would enable skipping the parenthetical material in "List all messages, assuming there are any, from Brown".

In the area of pragmatics, solutions that could fit within our paradigm have been suggested for two classes of problems. One problem is the failure of presuppositions of an input. In the environment of an intelligent tutor for computer-assisted language instruction, a technique suggested in Weischedel et al. (1978) could be formulated as a meta-rule as follows. The LHS would check whether processing was blocked due to a presupposition being false. Since that system would have a more complete knowledge of language than a beginning student of a foreign language, the system could treat the input as absolutely ill-formed. A sophisticated RHS could paraphrase the false presupposition for the student and indicate which word or syntactic construction was used inappropriately. Thus, the tutor could point out mistakes such as "Das Fraeulein ist Student", indicating that the student should look up the meaning of "Student" (which applies only to males).

Kaplan (1978) suggests an alternative heuristic for false extensional presuppositions in a data base environment. One can reformulate it as a meta-rule whose LHS would check that the query had requested a set as a response and that the set was empty. The RHS would compute queries corresponding to subsets that the original query presupposed would have a non-empty extension. The RHS would paraphrase the most general such query with an empty response set, reporting to the user that the system knew of no such entities.

5. Implementation

We implemented a grammatical meta-rule processor first for an ATN interpreter and more recently for an ATN compiler (Burton and Brown 1977). Our experiments have used RUS (Bobrow 1978), a broad-coverage grammar of English with calls to a semantic component to block anomalous interpretations proposed by the grammar.

Design and implementation of a meta-rule processor for violation of semantic constraints is currently underway in two different semantic interpreters. In one, case constraints are expressed as sets of logical formulas; in the other, KL-ONE is used to encode case frames (Sondheimer et al. 1984).

Four design issues are considered in the following sections.

5.1. Applying meta-rules

The set of meta-rules dealing with the grammar or semantic system could be viewed formally as a function f from a component's rules S to a new component's rules S'.

$$f(S) = S'$$

S' is the transitive closure of applying every meta-rule pertaining to the system rules in every possible way. (Since it is the transitive closure, S is contained in S'). There are three alternatives. One is to compute S' and use it, rather than S, as the basis of processing, assuming that the transitive closure S' is a finite closure. The second is to apply meta-rules only as needed, thus making S' a virtual system. The third alternative is a combination of applying some meta-rules as needed and applying others in advance.

The first alternative is superficially similar to approach 2 of Section 2.1, where ill-formedness processing is embedded in the normative system; however, S' will maintain the preference for normal interpretations over ill-formed ones. We have rejected this alternative because of the combinatorial growth of rules needed for S'. For instance, one can write meta-rules for handling relaxation of word categories and relaxation of predicates on ATN arcs. Since both can occur throughout the grammar, they should not be expanded ahead of time. A similar argument is used to justify treating conjunction processing as a separate process rather than building it directly into the grammar (Woods 1973). Since the classes of ill-formedness can occur in combination, the number of relaxed rules in S' can be very large. Furthermore, since utterances where many, many combinations of errors occur should be rare, computing the transitive closure seems uncalled for.

The second alternative, generating a relaxed rule each time it is needed, is the one we implemented first in the context of an ATN interpreter. This alternative provides a kind of virtual system and avoids the increased memory necessary to hold S'.

The third alternative, applying some rules ahead of time and using others only as needed, offers the greatest flexibility and a variety of alternatives. We have implemented a version in which the underlying parser is the output of an ATN compiler. When the meta-rule processor applies a meta-rule at a given arc, the relaxed version of the arc is compiled and saved.[5] If the meta-rule is to be tried by the meta-rule processor at that arc again, the form of the relaxed arc need not be re-computed; it can simply be executed.

This third alternative also offers the potential of adapting the system to the idiosyncrasies of an individual's language and also the potential of extending its own model of language. Obviously, this is an area for future research.

Alternatives two and three assume only that the processor applying well-formedness rules is able to store a "configuration" in a queue or agenda. No assumption about the type of processing (for example, bottom-up or top-down), nor the class of violated rule (for example, lexical, syntactic, semantic, or pragmatic) is necessary.

5.2. What to store

When a configuration blocks because of the well-formedness rules, should the blocked configuration be stored or the results of applying each relevant meta-rule? Both of the implementations in the ATN environment save only the blocked configuration, namely, a blocked arc at the end of a path. The number of blocked configurations can be large. At present, there is insufficient evidence to determine whether a well-tuned set of meta-rules will yield a substantially larger (or smaller) number of relaxed configurations compared to the set of blocked configurations.

Some types of problems, for example, subject-verb agreement, may be so common, and some types of relaxation, for example, an unrecognized word, may be so diagnostically clear that the corresponding meta-rules should be applied immediately. In the case of subject-verb agreement, hand-compiling the meta-rule into the grammar may be appropriate (that is, writing an arc whose test is that subject-verb agreement failed and whose action places the new configuration on a queue that is tried only after all normal configurations have failed).

5.3. Localizing the problem

When processing ill-formed inputs, some means of ranking alternatives is appropriate, since the system must determine what is intended in the face of violated constraints and possible error. Also, the number of relaxed configurations may be large, even with a set of well-tuned meta-rules designed to open the search space minimally.[6] The ideal solution is that the ranking of alternatives should be based on syntactic, semantic, and pragmatic evidence, in addition to the diagnosis and recovery strategy.

The current implementation uses only some of those bases and employs a rather simple ranking. Since both grammatical constraints and selection restrictions are employed while parsing with RUS, both syntactic and semantic evidence is used. Blocked configurations are ordered on the amount of input processed; there is also a partial order on the meta-rules.

One of our students, Amir Razi, is designing an experiment to collect data on the performance of the system. The current system can be run in one of two modes: saving all blocked configurations or using only ones that proceeded rightmost in the input. One aspect of the experiment is to determine the frequency with which the interpretations covering the most input in a left-to-right parse block at the true source of the problem. Some preliminary evidence (Weischedel and Black 1980) indicates that this heuristic frequently does indicate where the problem is, if the normative system is nearly deterministic, for example, because the grammar is a fairly constrained subset of English or because semantic criteria filter out parses that have no meaning in the application domain.

Our long-term goal is accurate determination of both the problem in an ill-formed utterance and what was intended. The current implementation represents the first step toward that by employing both syntactic and semantic evidence. We are investigating the use of pragmatic evidence for that purpose as well. In addition, we wish to explore techniques for examining both the left and right contexts of a blocked interpretation, for instance, by employing bottom-up processing.

[5] The current implementation is limited somewhat; it saves the relaxed arc only if the RHS of the meta-rule modifies only the arc itself. Our misspelling meta-rule, for example, does not modify the arc at all, but rather the input string.

[6] However, it is not clear whether the combinatorics alone for typical inputs will be a problem, given the rapid increase in processor power/cost and the prospect of multi-processing.

5.4. A meta-rule index

Hand-compilation of meta-rules as mentioned in Section 5.1 is just one way to pinpoint the configurations to which a meta-rule applies; another is providing an index from blocked configurations to the meta-rules that could apply. We have implemented a preprocessor that builds an index from an ATN arc to the meta-rules that can apply to it. When loading the ATN grammar, our preprocessor localizes the syntactic meta-rules having IN-STATE?, FAILED-TEST?, and FAILED-ARC? in their LHS to the few arcs to which they could possibly apply. Clearly, if IN-STATE? is in the LHS, that meta-rule can apply to only the handful of arcs leaving one state. Since FAILED-ARC? and FAILED-TEST? require the arc to match a given pattern, meta-rules using these tests can be identified with the arcs satisfying those patterns.[7] Such preprocessing provides an index into the possible rules that apply to a blocked configuration, since the state and the arc will be part of the configuration. Furthermore, the pattern-matching operations in the LHS need not be repeated at run-time, since the preprocessor stores for each arc an altered form of the meta-rule (without the calls to the pattern matcher) and the bindings that pattern matching created.

Some meta-rules will not have any tests that localize their applicability; an example is the one for confusion words, which can appear almost anywhere. These are stored separately, and must be checked for any arc to which relaxation is to be tried.

6. Limitations

There are a number of points of caution. It should be clear that relaxation does not necessarily guarantee understanding. After all, relaxing any arc to (TST X T ...) will accept any word syntactically; yet that is no guarantee that the word will be understood. Relaxing constraints introduces additional potential for confusion.

What one classifies as "absolutely ill-formed" is clearly open to dispute, as Ross (1979) points out. Therefore, the system may classify something as ill-formed, ranking it behind other interpretations, even though the user views it as well-formed. We suspect that categorizing almost any particular constraint as normative could be the basis for argument. The criteria for deciding whether a constraint should be included in the normative system should include at least the following:

a) whether a native speaker would edit inputs that violate it,
b) whether violating the constraint can yield useful inferences,
c) whether examples exist in which the constraint carries meaning,
d) whether the constraint, if classified as normative, trims the search space, and
e) whether a processing strategy for the constraint can be stated more easily as a modification of normative processing, as in the case of conjunction (Woods 1973) or the case of contextual ellipsis in the data base environment (Weischedel and Sondheimer, 1982).

Thus far we have considered only constraints that are associated with a single point in the processing, such as relaxing a single case frame or relaxing a single ATN arc. Obviously, this need not be the case if, for instance, word or phrase order is permuted. At present, we have no general way of dealing with such problems.

7. Future Work

The problems of processing ill-formed input require several substantial research efforts. One is collecting additional corpora to determine patterns of errors and their frequency of occurrence. This is particularly important for two reasons. First, the more detail uncovered on patterns of error, the tighter the meta-rules for relaxing constraints. In our experience, the effort to make relaxation procedures as constrained and accurate as warranted by the patterns of occurrence is highly worthwhile, not only in trimming the search space, but also in eliminating senseless interpretations. Second, the patterns of ill-formedness will depend on the user community and the modality of input. For instance, non-native speakers of a language make different errors than native speakers. Typed input has a predominance of typographical/spelling errors; spoken input may have more restarted utterances.

As a correlate to the need for more corpora of ill-formed natural language, there is an obvious need to define highly specific heuristics (as meta-rules) to diagnose and recover from each type of ill-formedness. Some of the heuristics should involve clarification dialogue, another area for research.

There are many possible responses given a diagnosed problem. Consider a simple problem: violation of selection restrictions. In German, the verb "fressen" presupposes that the one eating is an animal. To an input such as "Dieser Mann frisst oft",[8] several recovery strategies could apply:

a) The selection restriction could be ignored.
b) The selection restriction could be generalized for future use.
c) The system could conclude that an error has occurred, as in the aforementioned language learning environment.

[7] Of course, this preprocessing assumes that no patterns in the LHS contain a form $expr.

[8] "This man eats often."

d) The system could engage in clarification dialog to determine whether the user intended to use that word.
e) The system could assume the user believes that the man referred to eats like an animal.

The conditions for selecting a strategy need to be studied. An explicit model of the user is needed for deciding the intent of the user and for appropriate recovery from ill-formedness.

Learning the idiosyncrasies of particular users and automatic extension of the system (based on detecting relatively ill-formed input) is very challenging. Some initial steps in this direction have been taken in Carbonell (1979), but there is much to be done. A significant aspect of the learning problem in this environment is the substantial uncertainty about whether the system has the intended interpretation, and the effect on both the functional and time performance of the system as the abnormal is viewed as more normal (and the search space correspondingly grows).

For syntactic ill-formedness, pure bottom-up parsing is intuitively very appealing, since one has descriptions of what is present both to the left and the right of the problem(s). The EPISTLE project (Jensen and Heidorn 1983) is employing bottom-up parsing. The advantage of employing top-down strategies, including left-corner parsing strategies, is the strong expectations available when a configuration blocks. Consequently, many relaxation strategies and systems in the literature (for example, Hendrix, et al. 1978; Kwasny and Sondheimer 1981; Weischedel and Sondheimer 1982) have been proposed and implemented in that framework. Use of bottom-up strategies offers interesting new classes of relaxation, such as rearranging constituents for ordering problems. It is not obvious how the combinatorics of bottom-up strategies will compare to those of top-down strategies. However, developing relaxation techniques for bottom-up processing and extensive empirical studies comparing them to top-down are certainly needed.

One of the most critical problems is control. The need to relax the very rules that constrain the search for an interpretation is like opening Pandora's box. This affects not only the time required to understand an ill-formed input, but also ambiguity through the additional alternatives the system is prepared to accept. There are several aspects to controlling this search. First, the well-formedness constraints should reflect strictly what is normative. Second, the relaxation rules should be made as tight as warranted by patterns of ill-formedness in language use. Third, a partial order on the relaxations should be established. Fourth, not only syntactic constraints and selection restrictions should be used (as in our system) but also pragmatic information to suggest the most promising alternatives. We have begun research on how to use pragmatic knowledge in an information-seeking environment for this purpose; see Carberry (1983, 1984) and Ramshaw and Weischedel (1984). In the environment of messages reporting events, Granger (1983) reports on using expectations based on stereotypical events for this purpose. Extensive empirical studies regarding effective control of the search space are needed.

The acid test for a framework, relaxation heuristics, and control strategies is not relaxing simple tests like subject-verb agreement or diagnosing obvious problems like a word not in the dictionary. Rather the acid test is a wide spectrum of problems, including examples like misspellings/typographical errors that result in a known word, because in this type of example, all of the local evidence can indicate that the incorrect word is perfectly correct. Trawick (1983) has initiated work on such misspelling problems.

8. Conclusions

Ill-formed input cannot be ignored by natural language processing systems. This paper has suggested a uniform framework for processing ill-formed input in the hope of providing a basis for standardizing work on ill-formedness.

Our framework has several advantages:

a) Well-formed interpretations are always preferred.
b) Ill-formedness processing is explicitly related to the well-formedness rules.
c) Only the constraint that seems to be violated is relaxed; all other well-formedness constraints are still effective for eliminating senseless interpretations and trimming search.
d) Deviance notes record the aspect that deviates from well-formedness, thus allowing pragmatic inferences by later processing.
e) Though our approach is uniform, it permits encoding as much specific knowledge into the diagnosis and recovery procedure as one desires for highly specialized cases.
f) Though this paper has drawn most of its examples from ATN grammars and from case frame processing, as argued in Section 3.3., the framework is not dependent on a particular model of language processing.
g) The framework should be applicable to lexical, syntactic, semantic, and pragmatic constraints.

Acknowledgments

The authors appreciate the helpful suggestions of Sudhir Advani, Robert J. Bobrow, Madeleine Bates, Sandra Carberry, Sheila Coyazo, Giorgio Ingargiola, Stan C. Kwasny, William Mann, William Mark, Geoff Pullum, Lance Ramshaw, Amir Razi, Richard L. Wexelblat, and William A. Woods in discussions about the research.

Much of the coding for the implementation described was done by Amir Razi.

References

Aho, A.V. and Ullman, J. 1972 *The Theory of Parsing, Transiation, and Compiling, Vol. 1.* Prentice-Hall, Englewood Cliffs, New Jersey.

Bates, M. 1976 Syntax in Automatic Speech Understanding. *American Journal of Computational Linguistics* Microfiche 45.

Bobrow, R.J. 1978 The RUS System. In Webber, B.L. and Bobrow, R., eds., Research in Natural Language. BBN Report No. 3378. Bolt Beranek and Newman Inc., Cambridge, Massachusetts.

Bobrow, R.J. and Webber, B. 1980a PSI-KLONE: Parsing and Semantic Interpretation in the BBN Natural Language Understanding System, *Proceedings of the 1980 Conference of the Canadian Society for Computational Studies of Intelligence.* CSCSI/SCEIO.

Bobrow, R.J. and Webber, B. 1980b Knowledge Representation for Syntactic/Semantic Processing. *Proceedings of the National Conference on Artificial Intelligence.* AAAI.

Bruce, B. 1975 Case Systems for Natural Language. *Artificial Intelligence* 6(4): 327-360.

Burton, R.R. and Brown, J.S. 1977 Semantic Grammar: A Technique for Constructing Natural Language Interfaces to Instructional Systems. BBN Report No. 3587. Bolt Beranek and Newman Inc., Cambridge, Massachusetts.

Carberry, S. 1983 Tracking User Goals in an Information-Seeking Environment. *Proceedings of the National Conference on Artificial Intelligence.* Washington, D.C.: 59-63.

Carberry, S. 1984 Understanding Pragmatically Ill-formed Input. *Proceedings of COLING 84.*

Carbonell, J.G. 1979 Towards a Self-Extending Parser. *Proceedings of the 17th Annual Meeting of the Association for Computational Linguistics.* La Jolla, California: 3-8.

Carbonell, J.G.; Boggs, W.M.; Mauldin, M.L.; and Anick, P.G. 1983 The XCALIBUR Project: A Natural Language Interface to Expert Systems. *Proceedings of the Eighth International Joint Conference on Artificial Intelligence.* Karlsruhe, West Germany: 653-656.

Chang, C.L. 1978 Finding Missing Joins for Incomplete Queries in Relational Data Bases. Research Report RJ2145. IBM Research Laboratory, San Jose, California.

Charniak, E. 1983 A Parser with Something for Everyone. In King, Margaret, ed., *Parsing Natural Languages.* Academic Press, New York, New York.

Eastman, C.M. and McLean, D.S. 1981 On the Need for Parsing Ill-formed Input. *American Journal of Computational Linguistics* 7(4): 257.

Fineman, L. 1983 Questioning the Need for Parsing Ill-formed Inputs. *American Journal of Computational Linguistics* 9(1): 22.

Fromkin, V.A., ed. 1973 *Speech Errors as Linguistic Evidence.* Janua Linguarum, Series maior 77. Mouton, The Hague.

Gawron, J.M.; King, J.; Lamping, J.; Loebner, E.; Paulson, E.A.; Pullum, G.K.; Sag, I.A.; and Wasow, T. 1982 The GPSG Linguistic System. *Proceedings of the 20th Annual Meeting of the Associaton for Computational Linguistics.* Cambridge, Massachusetts: 74-81.

Granger, R.H.; Staros, C.J.; Taylor, G.B.; and Yoshii, R. 1983 Scruffy Text Understanding: Design and Implementation of the NOMAD System. *Proceedings of the Conference on Applied Natural Language Processing.* Santa Monica, California: 104-106.

Harris, L.R. 1977a ROBOT: A High Performance Natural Language Interface for Data Base Query. Technical Report TR 77-1. Department of Mathematics, Dartmouth College, Hanover, New Hampshire.

Harris, L.R. 1977b User Oriented Data Base Query with the ROBOT Natural Language Query System. *International Journal for Man-Machine Studies* 9: 697-713.

Harris, L.R. 1978 The ROBOT System: Natural Language Processing Applied to Data Base Query. *Proceedings 1978 Annual Conference, Association for Computing Machinery.* Washington, D.C.: 165-172.

Hayes, P.J. and Carbonell, J.G. 1981 Multi-strategy Construction-Specific Parsing for Flexible Data Base Query and Update. *Proceedings of the Seventh International Joint Conference on Artificial Intelligence.* Vancouver, BC: 432-439.

Hayes, P. and Mouradian, G. 1980 Flexible Parsing. *Proceedings of the 18th Annual Meeting of the Association for Computational Linguistics and Parasession on Topics on Interactive Discourse.* Philadelphia, Pennsylvania: 97-103.

Heidorn, G.E. 1975 Augmented Phrase Structure Grammars. *Proceedings of the Workshop: Theoretical Issues in Natural Language Processing.* Cambridge, Massachusetts: 1-5.

Hendrix, G.G.; Sacerdoti, E.E.; Sagalowicz, D.; and Slocum, J. 1978 Developing a Natural Language Interface to Complex Data. *ACM Transactions on Database Systems* 3(2): 105-147.

Jensen, K.E. and Heidorn, G.E. 1983 The Fitted Parse: 100% Parsing Capability in a Syntactic Grammar of English. *Proceedings of the Conference on Applied Natural Language Processing.* Santa Monica, California: 93-98.

Kaplan, S.J. 1978 Indirect Responses to Loaded Questions. *Theoretical Issues in Natural Language Processing 2.* University of Illinois at Urbana-Champaign.

Kroch, A.S. 1981 On the Role of Resumptive Pronouns in Amnestying Island Constraint Violations. *The Proceedings of the 17th Regional Meeting of the Chicago Linguistic Society.*

Kwasny, S.C. and Sondheimer, N.K. 1981 Relaxation Techniques for Parsing Ill-formed Input. *American Journal of Computational Linguistics* 7(2): 99-108.

Linde, C. and Labov, W. 1975 Spatial Network as a Site for the Study of Language and Thought. *Language* 51(4): 924-938.

Malhotra, A. 1975 Design Criteria for a Knowledge-Based English Language System for Management: An Experimental Analysis. MAC TR 146. Project MAC, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Miller, L.A.; Heidorn, G.E.; and Jensen, K. 1981 Text-critiquing with the EPISTLE System: An Author's Aid to Better Syntax. *AFIPS Conference Proceedings, 1981 NCC.* AFIPS Press, Montvale, New Jersey: 649-655.

Ramshaw, L.A. and Weischedel, R.M. 1984 Problem Localization Strategies for Pragmatics in Natural Language Front Ends. *Proceedings of COLING 84.*

Robinson, J.J. 1982 DIAGRAM: A Grammar for Dialogues. *Communications of the ACM* 25(1): 27-46.

Ross, J.R. 1979 Where's English. In Fillmore, C.J.; Kempler, D.; and Wang, W.S-Y., eds., *Individual Differences in Language Ability and Language Behavior.* Academic Press, New York, New York: 127-163.

Sager, N. 1981 *Natural Language Information Processing: A Computer Grammar of English and its Applications.* Addison-Wesley, Reading, Massachusetts.

Schank, R.C.; Lebowitz, M.; and Birnbaum, L. 1980 An Integrated Understander. *American Journal of Computational Linguistics* 6(1): 13-30.

Sondheimer, N.K.; Weischedel, R.M.; and Bobrow, R.J. 1984 A Knowledge Representation for Semantic Interpretation. *Proceedings of COLING 84.*

Thompson, B.H. 1980 Linguistic Analysis of Natural Language Communication with Computers. *Proceedings of the Eighth International Conference on Computational Linguistics.* Tokyo, Japan: 190-201.

Trawick, D.J. 1983 Robust Sentence Analysis and Habitability. Technical Report 5074:TR:83. Computer Science Department, California Institute of Technology, Pasadena, California.

Walker, D.E. 1978 *Understanding Spoken Language.* North-Holland, New York, New York.

Waltz, D.L. 1978 An English Language Question Answering System for a Large Relational Database. *Communications of the ACM* 21(7): 526-539.

Warren, D.H.D.; Pereira, L.M.; and Pereira, F. 1977 PROLOG – The Language and its Implementation Compared to LISP. *SIGPLAN Notices* 12(8): 109-115.

Weischedel, R.M. and Black, J. 1980 Responding Intelligently to Unparsable Inputs. *American Journal of Computational Lingustics* 6(2): 97-109.

Weischedel, R.M. and Sondheimer, N.K. 1982 An Improved Heuristic for Ellipsis Processing. *Proceedings of the 20th Annual Meeting of the Association for Computational Linguistics.* Cambridge, Massachusetts: 85-88.

Weischedel, R.M.; Voge, W.M.; and James, M. 1978 An Artificial Intelligence Approach to Language Instruction. *Artificial Intelligence* 10: 225-240.

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

Wilks, Y. 1976 Natural Language Understanding Systems Within the AI Paradigm – A Survey and Some Comparisons. *American Journal of Computational Linguistics* Microfiche 40.

Winograd, T. 1972 *Understanding Natural Language.* Academic Press, New York, New York.

Woods, W.A. 1970 Transition Network Grammars for Natural Language Analysis. *Communications of the ACM* 13(10): 591-606.

Woods, W.A. 1973 Progress in Natural Language Understanding – An Application to Lunar Geology. *AFIPS Conference Proceedings, 42.* AFIPS Press, Montvale, New Jersey: 441-450.

Woods, W.A.; Kaplan, R.M.; and Nash-Webber, B. 1972 The Lunar Sciences Natural Language Information System: Final Report. BBN Report No. 2378. Bolt Beranek and Newman Inc., Cambridge, Massachusetts.

Woods, W.A.; Bates, M.; Brown, G.; Cook, C.; Klovstad, J.; Makhoul, J.; Nash-Webber, B.; Schwartz, R.; Wolf, J.; and Zue, V. 1976 Speech Understanding Systems: Final Report. Volumes 1-5. Bolt Beranek and Newman Inc., Cambridge, Massachusetts.