Relaxation Techniques for Parsing Grammatically Ill-Formed Input in Natural Language Understanding Systems[1]

Stan C. Kwasny

Computer Science Department
Indiana University
Bloomington, Indiana 47401

Norman K. Sondheimer

Sperry Univac
Blue Bell, Pennsylvania 19424

This paper investigates several language phenomena either considered deviant by linguistic standards or insufficiently addressed by existing approaches. These include co-occurrence violations, some forms of ellipsis and extraneous forms, and conjunction. Relaxation techniques for their treatment in Natural Language Understanding Systems are discussed. These techniques, developed within the Augmented Transition Network (ATN) model, are shown to be adequate to handle many of these cases.

1. Introduction

Among the components included in a Natural Language Understanding (NLU) system is a grammar which specifies much of the linguistic structure of the utterances that can be expected. However, it is certain that inputs that are ill-formed with respect to the grammar will be received, both because people regularly form ungrammatical utterances and because there are a variety of forms that cannot be readily included in current grammatical models and are hence "extra-grammatical". These might be rejected, but as Wilks (1976) stresses, "... understanding requires, at the very least, ... some attempt to interpret, rather than merely reject, what seem to be ill-formed utterances."

This paper investigates several language phenomena commonly considered ungrammatical or extra-grammatical and discusses techniques directed at integrating them as much as possible into the conventional framework of grammatical processing performed by NLU systems. A "normative" grammar is assumed which specifies the structure of well-formed inputs. Rules that are both manually added to the original grammar and, by relaxing the constraints of the grammar, automatically constructed during parsing analyze the ill-formed input. The ill-formedness is shown at the completion of a parse by an indication of its deviance from a fully grammatical structure. We have been able to accomplish this while preserving the structural characteristics of the original grammar and its inherent efficiency.

The Augmented Transition Network (ATN) model was chosen as the tool in which to express our ideas, and a basic understanding of this model is assumed throughout the paper.[2] However, we believe our ideas are more general in scope than the ATN implementation may suggest. Similar ideas have recently been integrated into the Augmented Phrase Structure Grammar model by Miller, Heidorn, and Jensen (1981).

Some of the phenomena discussed have been considered previously in particular NLU systems, as, for example, the ellipsis handling in LIFER described in Hendrix, Sacerdoti, Sagalowicz, and Slocum (1978). Similar methods for processing conjunction have been

[1] This paper is a revised and extended version of a paper "Ungrammaticality and Extra-Grammaticality in Natural Language Understanding Systems" presented at the 17th Annual Meeting of the Association for Computational Linguistics, San Diego, California, August, 1979.

[2] For a thorough introduction to the ATN formalism, see Bates (1978).

0362-613X/81/020099-10$01.00

used in the LUNAR system as discussed in Woods (1973). In linguistic studies, Chomsky (1964) and Katz (1964), among others, have considered the treatment of ungrammaticality within the Transformational Generative model. Contemporary studies closest to ours are those of Weischedel and Black (1980) and Hayes and Mouradian (1980). Similarities will be pointed out as our techniques are presented. The present study is distinguished by the range of phenomena considered, structural and efficiency goals, and the inclusion of techniques in one uniform framework.

This paper surveys these problems, discusses mechanisms aimed at solving them, and discusses how these mechanisms are used. At the end, some limitations are discussed and extensions suggested. Unless otherwise noted, all ideas have been tested through implementation. A more detailed discussion of all points may be found in Kwasny (1980).

2. Language Phenomena

Throughout this paper we will argue that success in handling ungrammatical and extra-grammatical input depends on two factors. The first is the identification of types of ill-formedness and the patterns that they follow. The second is the relating of ill-formed input to the parsing path of a grammatical input the user intends. This section introduces the types of ill-formedness we have studied and discusses their relationship to grammatical structures in terms of ATN grammars.

2.1 Co-Occurrence Violations

Our first class of errors is connected to co-occurrence restrictions within a sentence. There are many occasions in a sentence where two or more parts must agree, as in the following examples (an asterisk "*" indicates an ill-formed or ungrammatical sentence):

*Draw *a circles.*

**I*, along with many other Germans, *are* concerned about the Russian threat.

*I will *stay from* now *under* midnight.

The errors in the above involve coordination between the italicized words. The first and second examples illustrate simple agreement problems. The third involves a complicated relation among at least the three italicized terms.

Such phenomena do occur naturally. For example, Shores (1977) analyzes fifty-six freshman English papers written by black college students and reveals patterns of nonstandard usage ranging from uninflected plurals, possessives, and third person singulars to overinflection (use of inappropriate endings).

In ATN terms, the significant blocks that keep inputs with co-occurrence violations from being parsed as the user intended arise from a failure of a test on an arc or the failure to satisfy an arc type restriction, e.g., the failure of a word to be in the correct category. The essential block in the first example above would likely occur on an agreement test on an arc accepting a noun. The essential blockages in the second and third examples are likely to come from failure of the arcs testing the verb and final preposition, respectively.

2.2 Ellipsis and Extraneous Terms

In handling ellipsis, the most relevant distinction to make is between *contextual* and *telegraphic* ellipsis. Contextual ellipsis occurs when a form makes sense only in the context of other sentences. For example, the form

*President Reagan has.

seems ungrammatical without the preceding question

Who has a wife named Nancy?

Telegraphic ellipsis, on the other hand, occurs when a form only makes proper sense in a particular situation. For example, the forms

*3 chairs no waiting (sign in barber shop)

*Yanks split (headline in sports section)

*Profit margins for each product
(query submitted to a NLU system)

are cases of telegraphic ellipsis with appropriate situations for their use noted in parentheses. The third one is from an experimental study of NLU for management information by Malhotra (1975) which indicated that such forms must be considered.

Further evidence to justify a study of ellipsis is given by Thompson (1980) who analyzes several modes of communication and concludes that an important subset of utterances used in communication are fragments, not sentences. Of particular interest in the analysis is the notion of a terse question, which is a type of telegraphic ellipsis, and a terse reply, which is a type of contextual ellipsis. In her human-to-computer experiments, she found 7.6% terse questions and 10.3% terse replies out of 882 parsed messages.

Another type of ungrammaticality complementary to ellipsis occurs when the user puts unnecessary words or phrases in an utterance. The reason for an extra word may be a change of intention in the middle of an utterance, an oversight, or simply for emphasis. For example,

*Did you ... did I erase any files?

*List prices of single unit prices for 72 and 73.

The second example again comes from Malhotra (1975).

The best way to view ill-formedness in terms of the ATN is to think of the user as trying to complete a path through the grammar, but having produced an input that has too many or too few forms necessary to traverse all arcs. For contextual ellipsis, grammatical processing can be guided, in part, by expectations of what the form of the input will be.

For telegraphic ellipsis, it might be argued that the forms could be placed independently in the grammar, but since they allow so much variation compared to grammatical forms, including them with existing techniques would dramatically increase the size of the grammar. Furthermore, there is a real distinction in terms of completeness and clarity of intent between grammatical and ungrammatical forms. Hence we feel justified in suggesting special techniques for their treatment which will relate their structure to the normative grammar.

2.3 Conjunction

Conjunction is an extremely common phenomenon, but it is seldom directly treated in a grammar. We have considered several types of conjunction. Simple forms of conjunction occur most frequently, as in

John loves Mary and hates Sue.

Gapping occurs when internal segments of the second conjunct are missing, as in

John loves Mary and Mary John.

The list form of conjunction occurs when more than two elements are joined in a single phrase, as in

John loves Mary, Sue, Nancy, and Bill.

Correlative conjunction occurs in sentences to coordinate the joining of constituents, as in

John both loves and hates Sue.

Conjunctions are generally not included in grammars because, similarly to ungrammatical forms, they can appear in so many places that their inclusion would dramatically increase the size of the grammar.

We see conjunction as similar to ellipsis. For example, we see some of the above as resulting from sentence-joining with elided forms:

John loves Mary and *John* hates Sue.

John loves Mary and Mary *loves* John.

John loves Mary, *John loves* Sue,
John loves Nancy, and *John loves* Bill.

This view of conjunction is consistent with views expressed by Halliday and Hasan (1976) and by Quirk, Greenbaum, Leech, and Svartik (1972).

3. The Mechanisms and How They Apply

In this section, techniques are described for addressing the phenomena introduced in the previous section. All of the techniques follow a general paradigm of relaxation wherein a normative grammar is assumed which describes the set of acceptable inputs to the system. During parsing, whenever a relaxable arc cannot be traversed, a backtrack point is created which includes the relaxed version of the arc. These alternatives are not considered until after all possible grammatical paths have been attempted, thereby insuring that grammatical inputs will be handled correctly. Relaxation of previously relaxed arcs is also possible. When processing is completed, a "deviance note" which describes how the input deviated from the expected grammatical form is passed along to other processing components of the system.

3.1 Feature Relaxation Techniques

The most straightforward methods of relaxation are those that operate at the lexical level. The principle involved is to permit a word which is slightly inappropriate to stand in place of the correct word. Slight feature variations are present in the lexical entry for the word found in the input when compared to the expected word. Two methods of feature relaxation have been investigated.

Our first method involves relaxing a test on an arc. *Test relaxation* occurs when the test portion of an arc contains a relaxable predicate and the test fails. Two methods of test relaxation have been devised and implemented based on predicate type. Predicates can be designated by the grammar writer as either *absolutely violable* in which case the opposite value of the predicate is substituted for the predicate during relaxation or *conditionally violable* in which case a substitute predicate is provided. For example, consider the following to be a test that fails

```
(AND
     (NUMBER-AGREE SUBJ V)
     (INTRANS V))
```

If the predicate NUMBER-AGREE was declared absolutely violable and its use in this test returned the value NIL, then the negation of (NUMBER-AGREE SUBJ V) would replace it in the test creating a new arc with the test:

```
(AND
     T
     (INTRANS V))
```

If INTRANS were conditionally violable with the substitute predicate TRANS, then the following test would appear on the new arc:

```
(AND
     (NUMBER-AGREE SUBJ V)
     (TRANS V))
```

Whenever more than one test in a failing arc is violable, all possible single relaxations are attempted independently. This is similar to a method of Weischedel and Black (1980).

Chomsky (1964), in developing his ideas on degrees of grammaticalness, discusses the notion of organizing word categories hierarchically. We have applied and extended these ideas in our second method of relaxation called *category relaxation*. In this method, the grammar writer produces, along with the grammar, a hierarchy describing the relationship among words, categories, and phrase types which is used by the relaxation mechanism to construct relaxed versions of arcs that have failed. When an arc fails because of an arc type failure, i.e., because a particular word, category, or phrase was not found, a new arc (or arcs) may be created according to the description of the word, category, or phrase in the hierarchy. Typically, PUSH arcs will relax to PUSH arcs, CAT arcs to CAT or PUSH arcs, and WRD or MEM arcs to CAT arcs. Consider, for example, the syntactic category hierarchy for pronouns shown in Figure 1. For this example, the category relaxation mechanism would allow the relaxation of a CAT arc identifying PERSONAL pronouns to include the entire PRONOUN category, including the subcategories REFLEXIVE and DEMONSTRATIVE. As with test relaxation, successive relaxations could occur.

For both methods of relaxation, "deviance notes" are generated which describe the nature of the relaxation in each case. For example, in the first relaxation example given above, The deviance note produced by our implementation would look like:

```
(RELAXED-TEST
     (FROM
          (AND
               (NUMBER-AGREE SUBJ V)
               (INTRANS V)))
     (TO
          (AND
               T
               (INTRANS V))))
```

Where multiple types or multiple levels of relaxation occur, a note is generated for each. The entire list of deviance notes accompanies the final structure produced by the parser. In this way, the final structure is marked as deviant and the nature of the deviance is available for use by other components of the understanding system.

In our implementation, test relaxation has been fully implemented, while category relaxation has been implemented for all cases except those involving PUSH arcs. In general, the CAT to PUSH category relaxation should involve no special problems in implementation, while the PUSH to PUSH relaxation involves identifying the failure of a PUSH arc, which involves keeping track of well-formed substrings.

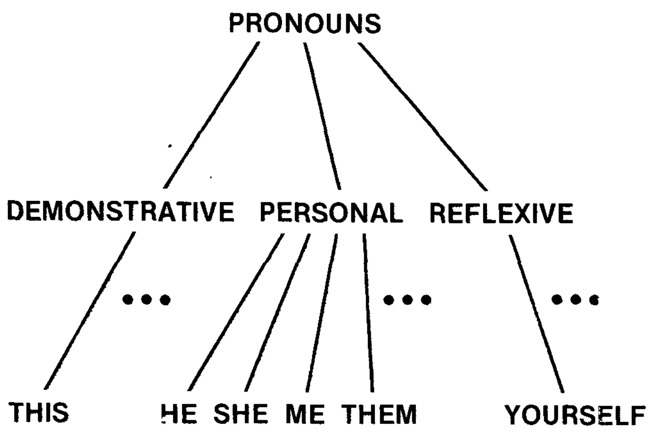

Figure 1. Category hierarchy.

3.2 Co-Occurrence and Feature Relaxation

Handling forms that are deviant because of co-occurrence violations involves using the above feature relaxation methods. Where simple tests exist within a grammar to filter out unacceptable forms, these tests may be relaxed to allow the acceptance of these forms.

For co-occurrence violations, the points in the grammar where parsing becomes blocked are useful in determining exactly where the test or category violations occur. Arcs at those points are being attempted and fail due to the failure of a co-occurrence test or categorization requirement. Relaxation is then applied and alternatives generated which may be explored at a later point via backtracking. For example, the sentence:

*John love Mary

shows a number disagreement between the subject and the verb. Most probably this would show up during parsing in a test on an arc when the verb of the sentence is being considered. That test could in fact be the one considered in our example. The test would fail and the traversal would not be allowed. At that point, an ungrammatical alternative similar to the one we described would be created for later backtracking consideration, and processing would proceed as discussed.

Absolutely violable predicates can be permitted in cases where the test describes some superficial consistency checking or where the test's failure or success does not have a direct effect on meaning, while conditionally violable predicates apply to predicates which must be relaxed cautiously to avoid loss of meaning.

An example of the application of category relaxation can be seen utilizing the category hierarchy in Figure 1:

Draw me a circle.

The normative grammar may be written so that a WRD arc is used to specify the processing of the word "me". Subsequent relaxation of that arc, following Figure 1, would permit any personal pronoun. This would specify a method of processing sentences such as the following:

*Draw he a circle.

A second level of relaxation would permit forms like:

*Draw this a circle.

The technique of category relaxation is certainly not limited merely to syntactic categories. The Burton (1976) semantic grammar approach to language processing makes exclusive use of categories which distinguish semantic entities rather than the traditional syntactic ones. A category hierarchy can easily be constructed which describes how these categories relate in the system without extending the mechanism described here.

3.3 Patterns and the Pattern Arc

In this section, relaxation techniques are applied to the grammar itself through the use of patterns and pattern-matching algorithms. Other researchers have used patterns for parsing, such as Parkison, Colby, and Faught (1977), Wilks (1978), and Hayes and Mouradian (1980), but we have devised a method of integrating patterns within the ATN formalism for the purpose of improving the grammar's responsiveness to ungrammatical and extra-grammatical inputs.

In our current formulation, a *pattern* is a linear sequence of ATN arcs which is matched against the input string. A *pattern arc* (PAT) has been added to the ATN formalism with a form similar to that of other arcs:

(PAT <pat spec> <test> <act>* <term>)

The pattern specification (<pat spec>) is described in Figure 2 through a modified BNF grammar, where an asterisk indicates zero or more occurrences of the preceding non-terminal. The pattern (<patt>) is either a user-assigned pattern name, a ">", or a list of ATN arcs, each of which may be preceded by the symbol ">" to indicate a potentially optional arc, while the pattern mode (<mode>) can be any of the keywords, UNANCHOR, OPTIONAL, or SKIP. These are discussed below.

An index of patterns is supported in our implementation to permit naming and referencing of patterns. Also supported is an index of arcs, allowing the naming and referencing of arcs as well. Further, named patterns and named arcs are defined as LISP macros, which expand into arcs according to how they are referenced. This allows the two indexes and the grammar to be substantially reduced in size.

Consider the following example of a pattern arc. Suppose the sentence:

Mary drove the car.

was processed by an ATN path through a sentence network of three arcs, namely PERSON, TRANS-ACT, and OBJ which processed "Mary", "drove", and "the car" respectively. Then a pattern of the sentence could be specified as:

```
(PERSON  TRANS-ACT  OBJ)
```

This, then, could be included in a pattern arc, where the ". . ." is realized by the appropriate tests and actions, as:

```
(PAT  ((PERSON  TRANS-ACT  OBJ))  T ... )
```

It could further be modified to permit optional constituents, as in

```
(PAT  ((>PERSON >TRANS-ACT OBJ )
              OPTIONAL) T ... )
```

Pattern matching proceeds by matching each arc in the pattern against the input string, but is affected by the chosen "mode" of matching. Since the individual component arcs are, in a sense, complex patterns, the ATN interpreter can be considered part of the matching algorithm as well. In arcs within patterns, explicit transfer to a new state is ignored and the next arc attempted on success is the one following in the pattern. As in the example above, an arc in a pattern prefaced by ">" can be considered optional, if the OPTIONAL mode has been selected to activate this feature. When this is done, the matching algorithm still attempts to match optional arcs, but may ignore them. In the example of a pattern arc using the op-

<pat spec>	::=	(<patt> <mode>*)
<patt>	::=	(<p arc>*)
		<pat name>
<mode>	::=	UNANCHOR
		OPTIONAL
		SKIP
<p arc>	::=	<arc>
		> <arc>
<pat name>	::=	user-assigned pattern name
		>

Figure 2. Description of pattern specification.

tional feature above, the pattern component would be able to match any of the fragments:

*Mary the car

*Drove the car

*The car

as well as the original sentence. We will return to this example in our discussion on conjunction.

The manipulation of registers within optional arcs poses some interesting problems. With the optionality feature, not every arc will be executed in a pattern. To complicate matters, registers previously set may be used within a test or they may hold elements of the final structure. Tests involving undefined registers are defaulted to allow traversal of these arcs, while structure-building actions involving undefined registers mark the missing components as undefined.

In addition, there are two other modes of matching. A pattern unanchoring capability is activated by specifying the mode UNANCHOR. In this mode, patterns are permitted to skip words prior to matching. Specifying the SKIP mode results in words being ignored between matches of the arcs within a pattern. This is a generalization of the UNANCHOR mode.

As with all forms of relaxation, pattern matching results in deviance notes. For patterns, these notes contain information necessary to determine how matching succeeded. Sufficient information is contained in the deviance notes to recover skipped words and skipped arcs, if desired.

3.4 Patterns, Ellipsis, and Extraneous Forms

The pattern arc is the primary mechanism for handling ellipsis and extraneous forms. A pattern arc can be seen as capturing a single path through a network. The matcher gives some freedom in how that path relates to a string. The appropriate parsing path through a network relates to an elliptical sentence or one with extra words in the same way. With contextual ellipsis, the relationship will be in having some of the arcs on the correct path not satisfied. In pattern arcs, these will be represented by arcs marked as optional. Dialogue context will provide the defaults for the missing components. With pattern arcs, the deviance notes will show what was left out and the other components in the NLU system will be responsible for supplying the values.

As an example, consider the following question and its possible elliptical responses:

Who drove a car?

*Mary.

*Mary did.

*Mary her Ford.

The form of the basic expected response could be shown by the following pattern, where AUX refers to an arc that identifies auxiliary verbs:

```
(PAT ((PERSON >AUX >TRANS-ACT >OBJ)
                 OPTIONAL) T ... )
```

This would serve to identify the three elliptical responses as well as others. It would also accept some unlikely inputs such as:

*Mary could a car.

*Mary the Empire State Building.

These would have to be filtered by later processing by other components through the use of the deviance notes. A pattern for the similar question:

Mary drove what?

could be as follows:

```
(PAT ((>PERSON >AUX >TRANS-ACT OBJ)
                 OPTIONAL) T ... )
```

The source of patterns for contextual ellipsis is important. In the LIFER system discussed in Hendrix, Sacerdoti, Sagalowicz, and Slocum (1978), the previous user input can be seen as a pattern for elliptical processing of the current input. The automatic pattern generator introduced in the next section, along with an expectation mechanism, will capture this level of processing. But, with the ability to construct arbitrary patterns and to add them to the grammar from other components of the NLU system, our approach can accomplish much more. For example, a question generation routine could add an expectation of a yes/no answer in front of a transformed rephrasing of a question, as in

Did Mary drive a car?

*Yes, she did.

*No, she did not.

The appropriate patterns might be the following, where YES, NO, and NOT refer to arcs that accept these words:

```
(PAT ((YES >PERSON >AUX >TRANS-ACT
        >OBJ) OPTIONAL) T ... )
(PAT ((NO >PERSON >AUX >NOT >TRANS-ACT
        >OBJ) OPTIONAL) T ... )
```

Patterns for telegraphic ellipsis have to be added to the grammar manually. One typical usage would be in accepting terse questions where the actions default to a standard operation in the system. For example, in a database situation, a reference to a set of objects could be understood as a command to retrieve them, as with the "profit margin" query given earlier. A pattern here might be the following, where IMP-ACT is an arc that identifies the typical action, and SET identifies a set description:

```
        (PAT ((>IMP-ACT SET)
             OPTIONAL) T ... )
```

Generally, patterns of usage must be identified, say in a study like that of Malhotra (1975), so that appropriate patterns can be constructed. Patterns for extraneous forms must also be added in advance. These would use either the UNANCHOR option in order to skip false starts, or use dynamically-produced patterns to catch repetitions for emphasis. In general, only a limited number of these patterns should be required. The value of the pattern mechanism here, especially in the case of telegraphic ellipsis, is in connecting the ungrammatical forms to the grammatical ones.

The problem of extraneous words can be attacked by two of the <mode> settings in pattern arcs. The UNANCHOR mode applies to restarts, as in the earlier example, through the use of a simple pattern arc:

```
      (PAT (((PUSH S/ ... )) UNANCHOR) ... )
```

where the S/ network is used for parsing sentences. Extraneous words in the interior of sentences can be accommodated by the SKIP mode. These methods are similar to those described by Hayes and Mouradian (1980).

3.5 Dynamic Generation of Patterns

Patterns need to be generated dynamically in order to make use of the patterns of language occurring in the sentence. In this way, previous processing can be made to contribute more directly to later processing. An automatic pattern generation mechanism has been implemented using the trace of the current execution path to produce a pattern. This is invoked by using the symbol ">" in place of the pattern name. Patterns produced in this fashion contain only those arcs traversed at the current level of recursion in the network, although this could easily be generalized in a way which would permit PUSH arcs to be automatically replaced by their subnetwork paths. Each arc in an automatic pattern of this type is marked as optional; however, additional control of the optionality feature of the pattern matcher can be exercised through judicious use of the test component of the arc. Patterns can also be constructed dynamically in precisely the same way grammatical structures are built by using BUILDQ and other structure-building functions.

Pattern arcs enter the grammar in two ways. They are manually written into the grammar in those cases where the ungrammaticalities are common and they are added to the grammar automatically in those cases where the ungrammaticality is dependent on context. Those that are produced dynamically enter the grammar through one of two devices. They may be constructed as needed by special macro arcs or they may be constructed for future use through an expectation mechanism.

As the expectation-based parsing efforts clearly show, syntactic elements, especially words, contain important clues on processing (Riesbeck and Schank, 1976). Indeed, we also have found it useful to make the ATN mechanism more "active" by allowing it to produce new arcs based on such clues. To achieve this, the CAT, MEM, TST, and WRD arcs have been generalized and four new "macro" arcs, known as CAT*, MEM*, TST*, and WRD*, have been added to the ATN formalism. These are similar in every way to their counterparts, except that as a final action, instead of indicating the state to which the traversal leads, a new arc is constructed dynamically and immediately executed. The difference in the form that the new arc takes is seen in the following comparison:

```
(CAT   <cat> <test> <act>* <term>)
(CAT*  <cat> <test> <act>* <creat act>)
```

In this example, <creat act> is used to define the dynamic arc through the use of BUILDQ and similar structure-building functions, while <term> represents any terminal action (usually a TO action). Arcs computed by macro arcs can be of any type permitted by the ATN, but one of the most useful arcs to compute in this manner is the pattern arc discussed above. This allows a prescribed path of arcs to be attempted in processing the sentence.

An example of much of what was just described is the following macro arc:

```
(WRD*  AND  T
   (BUILDQ
      (PAT  ( >  OPTIONAL )
          T
          (SETR S (BUILDQ (AND + * ) S))
)  )  )
```

This WRD* macro arc attempts to find the word AND at the current position of processing the input. If successful, it computes a pattern arc with the pattern left to be automatically generated upon execution.

While the macro arc forces immediate execution of an arc, arcs may also be computed and temporarily added to the grammar for later execution through the *expectation* mechanism. Expectations are performed as actions within arcs (analogous to the HOLD action for parsing structures) or as actions elsewhere in the NLU system, e.g., during generation, when particular types of responses can be foreseen. Two forms are allowed:

(EXPECT <creat act> <state>)
(EXPECT <creat act>)

In the first case, the arc created is bound to a state as specified. When later processing leads to that state, the expected arc will be attempted as one alternative at that state. In the second case, where no state is specified, the effect is to attempt the arc at every state visited during the parse.

The range of an expectation produced during parsing is ordinarily limited to a single sentence, with the arc disappearing after it has been used; however, the start state, S/, is reserved for expectations intended to be active at the beginning of the *next* sentence. These will disappear in turn at the end of processing for that sentence.

3.6 Conjunction and Macro Arcs

Besides their utility in handling cases of ellipsis, pattern arcs are also the primary mechanism for handling conjunction. As mentioned earlier, the rationale for this is the connection between conjunction and ellipsis. Whenever a conjunction is seen, a pattern is developed from the already identified elements and matched against the remaining segments of input.

All of the forms of conjunction described above are treated through a globally defined set of "conjunction arcs", which are active at every state of the grammar. Some restricted cases, such as "and" following "between", may have the conjunction built into the grammar. These forms are not subject to the conjunction arcs. In general, the conjunction arcs are made up of macro arcs which compute pattern arcs. The automatic pattern mechanism is heavily used.

Returning to the earlier example of pattern arcs, it can be shown how these conjunction arcs work. Consider what takes place when the macro arc previously described is activated during the processing of the sentence:

Mary drove the car and John drove the truck.

Processing proceeds until the conjunction "and" is encountered. For the purpose of this example, it is assumed that the top level path of arcs followed during processing is simply the pattern discussed earlier:

```
(PERSON  TRANS-ACT  OBJ)
```

The action of the macro arc is, therefore, to (essentially) produce the arc:

```
(PAT ((>PERSON >TRANS-ACT >OBJ) OPTIONAL)
     T
     (SETR S (BUILDQ (AND + * ) S))
)    )
```

Processing continues from the state that triggered the execution of the conjunction arc, and a structure for the conjoined sentence is produced. In fact, any of the following variations on the sentence can be handled in precisely the same way:

Mary drove the car and the truck.
Mary drove the car and John the truck.
Mary drove the car and drove the truck.

Of course, more actions could be specified in the arc, as well as more complicated tests. In particular, a test should be included to force matching of the OBJ arc.

With simple conjunctions, the rightmost elements in the patterns are matched. Internal elements in patterns are skipped when gapping occurs. The list form of conjunction which uses punctuation in place of some of the conjunctions, can also be handled through the careful construction of dynamic patterns which form deferred expectations. Correlative conjunction is treated similarly, with expectations based on deferring the dynamic building of patterns as well.

The fact that conjunction arcs occur implicitly at every state of the grammar permits the same conjunction to trigger the same set of conjunction arcs from several different states at potentially several different levels in the grammar. With pattern arcs being computed dynamically, different patterns are attempted, depending largely on the subnetwork in which the state occurs. In this way the proper joining of constituents is realized.

3.7 Interaction of Techniques

As grammatical processing proceeds, ungrammatical possibilities are continually being suggested from the various mechanisms we have implemented. To coordinate all of these activities, the backtracking mechanism has been improved to keep track of these alternatives. All paths in the original grammar are attempted first. Only when all of these fail are the conjunction alternatives and the manually-added and dynamically-produced ungrammatical alternatives tried. All of the alternatives associated with a single state can be thought of as a single possibility. A selection mechanism is used to determine which backtrack point among the many potential alternatives should be explored next. Currently, we use a method also used by Weischedel and Black (1980) of selecting the alternative with the longest path, i.e., the one with the deepest penetration in the network.

4. Limitations and Extensions

Work has only begun on the problems associated with handling ill-formed input. The techniques discussed in this paper, we believe, are a step toward solving these problems, but work is continuing along several dimensions.

As they are presented and implemented, the test and category relaxation methods can be too non-specific in permitting relaxation of these types, since all arcs with the named tests and categories become potentially relaxable. But, the same function can be used for different purposes. For example, the function GETR can be used just to retrieve a value for a later test or as a predicate when used as a register containing a true/false flag. Weischedel and Black (1980) allow for such cases by having each instance of a test individually predicated. A compromise that would allow more generality to be captured would be to specify some or all of the arguments to the relaxable function when describing the form of relaxable function calls. The identification of arcs for category relaxation can be treated similarly.

The SKIP mode on patterns is limited in its applicability to sentences with extraneous forms. First, the mode setting can only be used on patterns, not on the general grammar. Second, it is probably too non-selective. The interpreter does prefer to skip as few words as possible; however no preference is given as to where to skip them. Thus, even if patterns of extraneous words are identified precisely, the word skipping mechanism would simply not be able to allow for them.

One weakness in the pattern arc is the treatment of register settings in optional arcs that are skipped. Rather than ignoring the use of registers that would have been set in the arc, it would be better to devise a default strategy to force their setting.

It is instructive to compare our conjunction technique to the SYSCONJ facility of Woods (1973). The latter treats conjunction as showing alternative ways of continuing a sentence. This allows for sentences such as

> He drove his car through and broke
> a plate glass window.

which at best we will accept with a misleading deviance note. This is in part an example of what Quirk (1972) calls "cataphoric" ellipsis:

> John can *pass the examination*, and Bob
> certainly will, pass the examination
>
> He drove his car through *a plate glass window*
> and *he* broke a plate glass window.

Here, the ellipsis occurs before, not after, the conjunction. To utilize techniques in our style on the above examples, it would be necessary to extract the proper pattern from the entire path of traversed arcs. However, unlike SYSCONJ, our technique can handle the obvious elliptical cases such as gapping, or the tightly constrained cases such as correlatives.

One of our original hopes for this work was that each of the relaxations would be applicable to a broad set of individual grammars. However, it quickly became obvious that the relaxations and grammar work best if they are developed together. The test relaxations that depend on the use of the predicates in the grammar are an obvious case in point. Another example is the case of conjunction arcs, which work best when constituent boundaries in the grammar reflect the structures that can be conjoined.

Another area for improvement is in the amount of time spent parsing before any ungrammatical choices are considered. Currently, all grammatical paths are considered first. This organization may be tempered by adjusting operational parameters in our implementation so that when a block occurs some or all of the ungrammatical alternatives may be investigated. However, guidelines for adjusting these parameters are lacking at the moment.

Another area for more research is the production of patterns for interpreting contextual ellipsis. At the moment, we inspect question forms and add predefined patterns to accept their elliptical answers. In order to produce these patterns automatically, a theory of question-and-answer dialogue is necessary.

Finally, developing heuristics which choose alternatives for relaxation among many blockages is a major problem. Our heuristic which picks the alternative with the longest path can overshoot the actual failure and possibly lead to an incorrectly built structure. We believe, as do Weischedel and Black (1980), that the answer may lie in being able to examine the semantic plausibility of partial structures.

Several other types of ill-formedness have not been considered in this study, for example, idioms, metaphors, incorrect word order, run-together sentences, incorrect punctuation, misspelling, and presuppositional failure. For each of these phenomena, either little is known about it or it has been studied elsewhere independently. In either case, work remains to be done to develop processing for them within our framework.

5. Conclusions

Despite the limitations and needed extensions discussed above, the results described here are significant, we believe, because they extend the state of the art in several ways. Most important are the following:

1. The use of the category hierarchy to handle arc type failures.

2. The use of the pattern mechanism to allow for contextual ellipsis and gapping.
3. More generally, the use of patterns to allow for many sorts of ellipsis and conjunctions.
4. Finally, the combination of all of the techniques in one coherent system, where because all grammatical alternatives are tried first and no modifications are made to the original grammar, its inherent efficiency and structure are preserved.

The theme of the final point is further developed in Sondheimer and Weischedel (1980).

Acknowledgments

We wish to acknowledge the comments of Ralph Weischedel and Marc Fogel on previous drafts of this paper. The cooperation and comments of George Heidorn, Richard Wexelblat, and the reviewers of this paper also contributed greatly.

References

Bates, M., "The Theory and Practice of Augmented Transition Network Grammars," in *Natural Language Communication with Computers*, L. Bolc (Ed.), 191-259, Springer-Verlag, Berlin, 191-259, 1978.

Burton, Richard R., "Semantic Grammar: An Engineering Technique for Constructing Natural Language Understanding Systems," Bolt Beranek and Newman, Report No. 3453, December, 1976.

Chomsky, N., "Degrees of Grammaticalness," in *The Structure of Language: Readings in the Philosophy of Language*, Fodor, J.A. and J.J. Katz (Eds.), 384-389, Prentice-Hall, Englewood Cliffs, New Jersey, 1964.

Halliday, M.A.K. and R. Hasan, *Cohesion in English*, Longman, London, 1976.

Hayes, P., and G. Mouradian, "Flexible Parsing," *Proceedings of the 18th Annual Meeting of the Association for Computational Linguistics*, 97-103, Philadelphia, June, 1980.

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

Katz, J.J., "Semi-Sentences," in *The Structure of Language: Readings in the Philosophy of Language*, Fodor, J.A. and J.J. Katz (Eds.), 400-416, Prentice-Hall, Englewood Cliffs, New Jersey, 1964.

Kwasny, S.C., "Treatment of Ungrammatical and Extragrammatical Phenomena in Natural Language Understanding Systems," PhD dissertation, Ohio State University, 1980, (available through the Indiana University Linguistics Club, Bloomington, Indiana).

Malhotra, A., "Design Criteria for a Knowledge-Based English Language System for Management: An Experimental Analysis," MAC TR-146, M.I.T., Cambridge, MA, February, 1975.

Miller, L.A., G.E. Heidorn, and K. Jensen, "Text-Critiquing with the EPISTLE System: An Author's Aid to Better Syntax," *Proceedings of the National Computer Conference*, 50, 649-655, AFIPS Press, Arlington, VA, 1981.

Parkison, R.C., K.M. Colby, and W.S. Faught, "Conversational Language Comprehension Using Integrated Pattern-Matching and Parsing," *Artificial Intelligence* 9, 2, 111-134, October, 1977.

Quirk, R., S. Greenbaum, G. Leech, and J. Svartik, *A Grammar of Contemporary English*, Seminar Press, New York, 1972.

Riesbeck, C.K., and R.C. Schank, "Comprehension by Computer: Expectation-Based Analysis of Sentences in Context," Yale University, Research Report 78, October, 1976.

Shores, D.L., "Black English and Black Attitudes," in *Papers in Language Variation*, D.L. Shores and C.P. Hines (Ed.), The University of Alabama Press, University, Alabama, 1977.

Sondheimer, N.K., and R.M. Weischedel, "A Rule-Based Approach to Ill-Formed Input," *COLING 80: Proceedings of the Eighth International Conference on Computational Linguistics*, 46-53, Tokyo, October, 1980.

Thompson, B.H., "Linguistic Analysis of Natural Language Communication with Computers," *COLING 80: Proceedings of the Eighth International Conference on Computational Linguistics*, 190-201, Tokyo, October, 1980.

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

Wilks, Y., "Natural Language Understanding Systems Within the A.I. Paradigm: A Survey," *American Journal of Computational Linguistics*, microfiche 40, 1, 1976.

Woods, W.A., "An Experimental Parsing System for Transition Network Grammars," in *Natural Language Processing*, R. Rustin (Ed.), 111-154, Algorithmics Press, New York, 1973.

Stan C. Kwasny is an Assistant Professor in the Computer Science Department at Indiana University. He received the Ph.D degree in Computer and Information Science from Ohio State University in 1980.

Norman K. Sondheimer is a Senior Computer Scientist in the Software Research Department of Sperry Univac. He received the Ph.D degree in Computer Science from the University of Wisconsin-Madison in 1975.