TECHNICAL PAPER

1986

MS86-963

Natural Language Processing Applied to Engineering Notes

author

JAMES J. TYHURST
Technical Staff
Hughes Aircraft Company
El Segundo, California

abstract

Engineering notes are placed on design drawings in order to convey requirements or instructions to the manufacturing organization. Therefore, process planning expert systems must interpret these notes which are written in a technical form of English. This paper discusses a software package which converts standardized notes into a frame/slot representation. This representation is used by the HICLASS™ Software System, an automated process planning expert system. Standardized notes are restricted to the fixed subset of English commonly used on design drawings. Standardization provides a mechanism for conveying manufacturing producibility feedback early in the design cycle.

conference

Ultratech-Artificial Intelligence
September 22-25, 1986
Long Beach, California

index terms

Artificial Intelligence
Process Planning
Engineering Drawings
Standardization
Design

Society of Manufacturing Engineers • One SME Drive • P.O. Box 930
Dearborn, Michigan 48121 • Phone (313) 271-1500

Natural Language Processing Applied to Engineering Notes

JAMES J. TYHURST
Hughes Aircraft Company
El Segundo, California

Engineering notes are placed on design drawings in order to convey requirements or instructions to the manufacturing organization. Therefore, process planning expert systems must interpret these notes which are written in a technical form of English. This paper discusses a software package which converts standardized notes into a frame/slot representation. This representation is used by the HICLASS™ Software System, an automated process planning expert system. Standardized notes are restricted to the fixed subset of English commonly used on design drawings. Standardization provides a mechanism for conveying manufacturing producibility feedback early in the design cycle.

INTRODUCTION

Expert systems which generate manufacturing planning from engineering designs must be able to make use of the information contained in the engineering notes. These notes frequently call out detailed information about how a part is to be fabricated or assembled. For example, particular process specifications may be designated or tolerances may be specified. In most cases, this information is not contained anywhere else on the design drawing. Therefore, an assembly planning expert system must have the ability to interpret the notes and extract the information required to produce a correct planning package.

Notes sometimes consist solely of tables of information. However, it is much more common for them to be written in a very abbreviated technical form of English. Thus, the task of interpreting notes is a good candidate for the application of "natural language processing". It is important to have a software package which can read engineering notes, parse them into their separate constituents, and organize the constituents into a format which can be readily accessed by an expert system.

In this paper, I will discuss the development of a parser for engineering notes which was developed as part of the HICLASS (TM) Software System (HICLASS is a trademark of the Hughes Aircraft Company). First, we will look at the kinds of notes which can be handled by the system. A number of factors pointed towards the establishment of "standard" notes. The following section gives an

account of the motivation for using standard notes and their benefits to both the engineering and manufacturing organizations. We will see why standardization is useful even if no automation systems will refer to the notes.

Then we will examine the format of generic note templates which describe the structure of notes through the specification of variables and optional phrases. A brief overview of the grammatical forms which are acceptable in standard notes is also provided. Two different parsing strategies have been implemented and these are briefly presented and compared. Finally, some of the unresolved problems are discussed.

ADVANTAGES OF STANDARDIZATION

Standard notes were jointly designed by members of the engineering and manufacturing groups. Both organizations stand to benefit by their implementation. For engineering, design work is occurring more and more on computer aided design (CAD) workstations. The use of a standardized note format allows notes to be stored in libraries which can be called up as necessary (in the same way that component libraries are made available). Manually written notes are subject to a variety of errors. These range from innocuous spelling errors to more serious errors like non-existent specification numbers. On the other hand, by choosing a note from a menu displayed on a CAD workstation the note is more likely to be error free.

Another important aspect of standard notes is that they provide a vehicle for communicating producibility information to the design engineer. Designations for process specifications or choice of materials to fill in the note template are ranked according to the manufacturing organization's criteria of cost, availability, storage difficulties, etc. Such a ranking system forms the basis for another software package which provides producibility information to the design engineer early in the design/manufacturing cycle.

A third reason for standardizing notes is to be sure that they are understood by the manufacturing organization. In some cases, the exact intent of a note is not clear and manufacturing must consult with engineering to determine the meaning of the note. This problem should be alleviated to the extent that an engineer can draw from a common pool of notes which have been established by both organizations.

A final consideration has been particularly influential in the establishment of standard notes at the Electro-Optical and Data Systems Group of Hughes Aircraft. An expert system which automatically generates process planning for printed circuit boards and mechanical assemblies must be able to access the information contained in engineering notes. This means that the notes must be machine interpretable. In order to achieve this, we have restricted the form of standard notes to be a constrained subset of English. As the abilities of natural language processing expand, these restraints can be relaxed to encompass a larger and larger subset of free form English.

THE FORMAT OF STANDARD NOTES

The result of the standardization process is a set of "standard notes" which are templates to be used in writing notes for design drawings. Each note contains a constant part and a set of slots which are filled in by the designer. The possible values which can fill those slots are assigned a producibility priority by the manufacturing organization in order to inform the designer of manufacturing's preferences for particular processes and materials.

Consider the following example of a particularly simple standard note. Values to be filled in are enclosed in parentheses. Optional parts of the note are contained in square brackets. This note has a single slot (Dimension) which must be filled in by the note writer. Possible values for the slot are listed in order of manufacturing's producibility priorities. For the Dimension slot in this note, .070 has highest priority and .060 is second highest. In addition, the note writer can specify some other value if necessary. The phrase "WITHIN AREA INDICATED" is an optional part of the note.

TRIM LEADS TO (Dimension) MAXIMUM[WITHIN AREA INDICATED]

			Priority
(Dimension)	=	.070	1
		.060	2
		Specify	

Note Templates

As demonstrated by the example above, there are three basic parts in a standard note template:

(1) the constant portion;
(2) the variable values; and
(3) the optional phrases.

The reason for distinguishing between a constant part and a variable value is that a particular process will be noted repeatedly on a large number of drawings. The form of the note on each drawing will be nearly identical, except that different specification numbers or tolerances may be given. Thus, the part of the note which is always the same is standardized, while the values which must change from drawing to drawing are isolated in the variable portions of a note.

The constant portion of a standard note remains the same for any particular instance of that note. Usually, the type of process (trim, bond, stencil, etc.) is a constant. Other key words which add to the structure of the note may be constants also. As a convention, we have chosen to represent the constant portion of a note in all capital letters.

The variable values of a note are designated in parentheses with only the first letter of the name of the variable being capitalized. This part of a note template is sometimes referred to as a "slot" and the values which can fill this slot are called "fillers". The name of the variable explains the type of value which can be inserted at this point in the note. An important aspect of the variable slots is that possible filler values should be listed

along with their producibility priorities. The possible filler values are not part of the actual note template, but they are listed along with the template as part of the entire standard note specification.

There is a difference between a variable slot and an optional phrase. The variable slot must always be filled in with a value for a particular instance of a note. As its name indicates the value of a variable slot will vary from note to note. On the other hand, an optional phrase is always the same. However, some instances of the standard note require the phrase, while others do not. Optional phrases are indicated by enclosing them in square brackets. The designer has the choice of including this phrase in a particular use of a standard note or he/she may choose to leave it out.

Optional phrases may include variable values. For example, the note below includes two optional phrases each of which contains two variables. In the first optional phrase, "[(When) (Process)]", the (When) variable can be filled with "AFTER" and the (Process) variable can be filled with "ELECTRICAL TEST" to yield the phrase "AFTER ELECTRICAL TEST" which describes one possible option for when the board should be conformal coated.

CONFORMAL COAT (Item) PER (Process Specification)
[(When) (Process)]
[. MAXIMUM CURE TEMPERATURE (Temp) DEGREES (Scale)]

As indicated by the square brackets, the above note has 2 optional phrases, each of which contains variable slots. Whenever the designer chooses one of the optional phrases, the variable slots must also be filled with an appropriate value. The optional phrases are independent of one another. The designer may choose none of the options, any one of them, or both of them. The relative ordering of phrases must remain the same as in the note template.

Having examined the various parts of a note template, the value of the standard notes concept should now be clear. The approach explained above allows us to succinctly describe what is basically an infinite number of notes. The single "CONFORMAL COAT" template above already describes 4 different note configurations (i.e. different combinations of the base note plus the optional phrases) without even considering the different values which can occur in the variable slots. If we tried to represent each instance of this note with the different filler values, we would end up with a huge collection of "standard" notes. Such an approach fails to capture the similarity between all these notes.

Rather than just listing all possible notes, our approach has been to abstract out the portions which are always the same from those portions which must necessarily be specified by the design engineer. In addition, notes which are very similar in format have been joined together by specifying optional phrases. This allows us to specify a whole family of notes with a single standard note template.

Producibility Criteria

An important aspect of standard notes is the inclusion of producibility criteria. This means that possible slot fillers should be listed along with a producibility priority. For example, any note that contains a "material specification" variable should also give a list of preferred material

specifications. These specifications will be rated 1, 2, 3, ... according to producibility criteria.

Grammar

The current notes grammar covers a very limited subset of English, yet it is sufficient to handle the vast majority of notes which occur on engineering drawings in the area of printed wiring assemblies and mechanical assemblies. The grammar has been designed specifically to be machine interpretable. However, since it corresponds to common English usage, it is not limited to machine interpretation, but is completely intelligible to anyone who routinely works with engineering notes.

Commonly Observed Note Structures

Before designing the standard notes grammar, we collected a sample of notes from printed wiring assembly and mechanical assembly designs. Notes are almost always in the imperative form dictating a particular process to be followed during manufacturing. This type of note contains a verb in the imperative form and the verb usually begins the sentence. For example:

INSTALL COMPONENTS PER <process specification>.

PRIME AND SEAL THREAD USING <material specification>.

A sequence of steps is frequently described in the notes by a prepositional phrase which occurs either before or after the main clause:

AFTER ELECTRICAL TEST OF CIRCUIT BOARD, MASK CONNECTOR PINS AND CONTACTS.

CONFORMAL COAT PWB PER <process specification> AFTER ELECTRICAL TEST USING MATERIAL PER <material specification>.

Other notes express the function of a particular item or they express a relationship between two items. These notes are of the form Subject - Verb - Object:

```
D-SHAPED PAD   INDICATES   POSITIVE END OF CAPACITOR.
  Subject    -    Verb   -      Object

ALTERNATE COMPONENTS FOR <part number>   ARE    <part numbers>.
                Subject                  - Verb -   Object
```

The sentence structure of engineering notes tends to be relatively simple, but the structure of noun phrases internal to the sentence (e.g. the subject or the object of the sentence) can be complex. A few examples of complex noun phrases are given below:

compound noun
    ACCEPTANCE TEST REQUIREMENTS

noun + prepositional phrase
    BONDING MATERIAL INSIDE SPACER

noun modified by a participial clause
THE VALUE DETERMINED BY TP-315
ALL HCI IDENTIFIED PARTS
THE DARK-COLORED CONDUCTIVE COATING COVERING THE DIELECTRIC

Note_Structure_from_a_Functional_Perspective

The motivation for choosing a particular structure for a parsed note is based partly on grammatical considerations and partly on the needs of the expert system which accesses parsed notes. We have chosen to represent the structure of a standard note using the following 8 functional constituents. Any standard note will consist of some combination of these constituents:

Subject
Verb
Object
Indirect Object
Process Specification
Verb Modifier
Material Specification
Comment

In the sample notes which follow, actual specification numbers and part numbers have been replaced by XXX.

Subject

(a) Something which performs a function, e.g. "D-SHAPED_PAD INDICATES POSITIVE END OF CAPACITOR".
(b) Something which is assigned or equated with certain attributes, e.g. "REF_DES ARE FOR REFERENCE ONLY AND DO NOT APPEAR ON COMPONENT PARTS".

Verb

(a) Denotes a process, e.g. "TRIM LEADS TO .070 MAXIMUM WITHIN AREA INDICATED".
(b) Expresses equivalence between two items, e.g. "DOT INDICATES TAB OF COMPONENTS".

Object

is something on which a process is performed, e.g. "SEAL ITEM_1 PER MIL-XXX USING MATERIAL PER MIL-S-XXX, GRADE IV".

Indirect_Object

denotes a second object for processes which inherently involve two objects. It is always introduced by the preposition "to", e.g. "SOLDER ITEM 1 TO_ITEM_13 PER MIL-STD-XXX, REQUIREMENT 5". Note: Not all prepositional phrases starting with "to" are indirect objects. In the note "TRIM LEADS TO .070 MAXIMUM WITHIN AREA INDICATED", the phrase "TO .070 MAXIMUM" tells how the trimming is to be done and therefore is parsed as a Verb Modifier.

Process_Specification

identifies the specification relevant to the process denoted by the Verb, e.g. "TEST PER_TP-XXX".

Verb Modifier
gives additional information about how, when, where, or to what extent the process denoted by the Verb is to be carried out. The Verb Modifier may be a single word (an "adverb" in traditional terminology) or it may be an entire phrase. For example, the Prepositional Phrase "AFTER ELECTRICAL TEST" is a verb modifier in the following note: "CONFORMAL COAT PWB PER QQ-N-XXX, CLASS 1, GRADE F AFTER ELECTRICAL TEST USING MATERIAL PER MIL-I-XXX, TYPE UR"

Material Specification
identifies the material to be used for the indicated process, e.g. "FILLET BOND ITEMS 1 AND 2 PER QQ-N-XXX, CLASS XXX, GRADE XXX USING MATERIAL PER MIL-S-XXX, TYPE OPTIONAL, CLASS XXX".

Comment
contains additional information which falls outside the domain of specifications. It is usually passed directly to the assembly operator with little or no processing by the automated planning system.

Each of these major constituents has some internal structure which may also be represented. For example, the Process Specification constituent is a prepositional phrase which consists of the preposition "PER" followed by a specification number. It is this number which is of interest to the process planner and it can be accessed by referring to the "Object" of the prepositional phrase:

Process Specification: PER MIL-XXX
Head: PER
Object: MIL-XXX

Thus, a note is represented as a hierarchical structure and the planner can refer to any depth of the structure. Frequently, it is sufficient to refer only to the highest level (e.g. the Verb can be accessed directly to determine the indicated manufacturing process). At other times, embedded structures will be accessed (e.g. the specification example above).

The relative order of these constituents is important. It is rare that any particular standard note will contain all 8 of the constituents. However, whichever constituents are included must be in the order shown. For example, in the following correctly formatted note, the Verb Modifier, "AFTER ELECTRICAL TEST", follows the Process Specification and precedes the Material Specification:

CONFORMAL COAT PWB PER (Process Specification) [AFTER ELECTRICAL TEST] [USING MATERIAL PER (Material Specification)]

Several incorrect versions of this same note are listed below. These are perfectly acceptable English sentences, but they do not conform to the established standard note format. Note (1) begins with a Verb Modifier, whereas the Verb Modifier is supposed to follow the Process Specification. The Verb Modifier is also misplaced in note (2), since it must follow the Process Specification, not precede it. In note (3), the sentence begins with a Material Specification, but this phrase should occur at the end of the sentence.

(1) [AFTER ELECTRICAL TEST,]CONFORMAL COAT PWB PER (Process Specification)[USING MATERIAL PER (Material Specification)]

(2) CONFORMAL COAT PWB [AFTER ELECTRICAL TEST]PER (Process Specification)[USING MATERIAL PER (Material Specification)]

(3) [USING (Material Specification),]CONFORMAL COAT PWB PER (Process Specification)[AFTER ELECTRICAL TEST]

Syntactic coverage

While the grammar is relatively simple compared to a complete grammar of English, it is sufficient for our problem domain and robust enough to handle notes as they are commonly written. This follows from the fact that engineering notes do not contain the grammatical complexities (extraposition, embedded clauses, heavy use of auxiliary verbs, pronominal reference, etc.) which must be handled in a more general natural language understanding system.

The term sublanguage is frequently used to describe a subset of language which is constrained by special terminology or restricted syntactic structures (Lehrberger 1982). Our grammar makes use of the fact that engineering notes form a sublanguage within English. Notes use English words, but often in a very specialized sense. The syntax of notes also corresponds to (a very limited subset of) common English usage. Thus, our grammar uses traditional terms of English grammar (Noun Phrase, Transitive Verb, etc.) as a basis for the parser. Some extensions are then provided for expressions that are outside the domain of standard English. For example, Process Specification Phrases are defined in the grammar, because their highly restricted format does not correspond to the normal structure of a Noun Phrase in English.

PARSING STRATEGIES

Two parsers have been implemented which transform standard notes into a representation that can be accessed by the assembly planning expert system. Both parsers accept a standard note as input and produce a tree structure with traditional grammatical labels as output. The first version implemented a parser for a context sensitive grammar using the UNIX (TM) tools LEX and YACC (UNIX is a trademark of Bell Telephone Laboratories, Incorporated; see the UNIX Programmer's Manual (1979) for more information about LEX and YACC). This parser is currently used in a production environment. A second version has used the HICLASS expert system shell to implement an Augmented Transition Network grammar (Woods 1970, Bates 1978). The HICLASS Software System is a general purpose logic analyzer which is used for developing expert systems at Hughes Aircraft (Liu 1984, 1985a). This second parser is still being tested, although we expect it to replace the first parser and be used in production in the near future.

The overall design of the parsers is shown in Figure 1 below. The same lexicon and lexical analyzer are used for both parser versions. Only the block labelled "parser" in the figure differs between the two implementations. The following sub-sections present the structure of the lexicon, the function of the lexical analyzer, and the parsing strategies implemented in the two parsers.

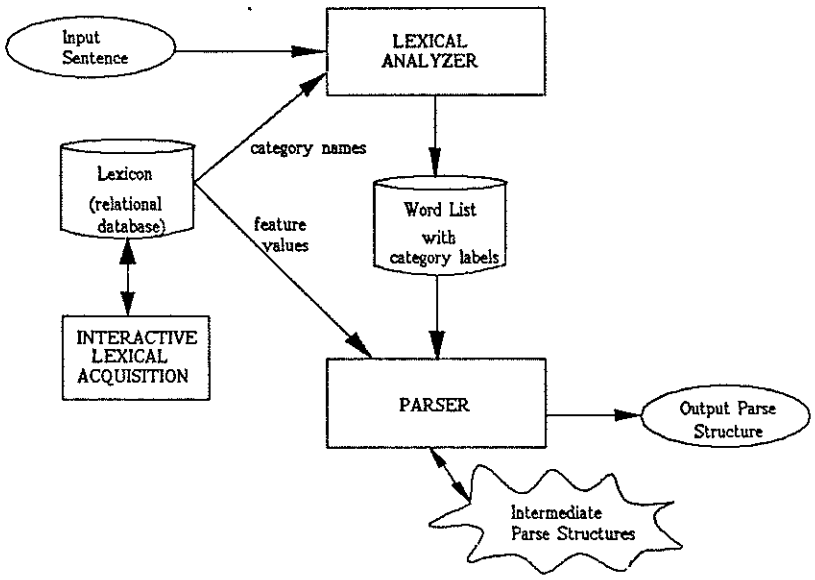


Figure 1. Logical structure of parser implementations.

Lexicon

All of the words which can occur in notes are stored in the lexicon. Each word is specified for major categories (e.g. Noun, Verb, Preposition, etc.) and for subcategories (e.g. verbs are further specified according to their form, type, and transitivity)

A word is frequently associated with several different sets of features. This ambiguity can occur either with respect to major category labels or subcategorization features. For example, the word "bond" is ambiguous with respect to the category it belongs to, since it can function either as a Noun or as a Verb. Similarly, the word "bonded" is ambiguous with respect to its subcategorization features as being either the Past tense or the Past Participle form of the verb "bond". All of these ambiguities are stored in the lexicon.

Lexical Analyzer

The UNIX tool LEX can generate a lexical analyzer given a regular expression specification of the desired categories. This type of lexical analyzer is used to extract the lexical items from a note and to assign them to a word class (e.g. Word, Integer Number, Real Number). Items of type 'Word' are then looked up in a lexicon to determine their syntactic category (e.g. Adjective, Noun, Verb). As mentioned above, a particular lexical item may belong to more than one grammatical category. The lexical analyzer finds all sets of features associated with a particular word and passes all possible interpretations to the parser. Thus, the problem of ambiguity at the lexical level is passed from the lexicon to the parser.

YACC Generated Parser

The first parser implemented for engineering notes was based on a set of augmented context free grammar rules. The UNIX tool YACC generated a parser from these rules. The interaction between the lexical analyzer and the action portion of the grammar rules allows one to generate a parser for a context sensitive language.

For example, modifier attachment is a notoriously difficult problem in English parsing. It is sometimes difficult to know if a prepositional phrase is part of a noun phrase or if it should be attached at the sentence level as a verb modifier. The notes parser makes this decision based on the syntactic context. For example, it checks the context to ask the following questions: Is the current phrase pre- or post-verbal? Is the main verb bitransitive? Can the head preposition refer to a timing sequence? etc.

When the right side of a grammar rule is matched, the action associated with that rule is performed. The actions build up the parse structure and they set various flags which can be used by later rules to identify a particular context. It was mentioned previously that the grammatical category of lexical items may be ambiguous. However, YACC generated parsers cannot handle this ambiguity, so there is a "front end" which feeds the alternatives to the parser one at a time. The order for checking the different possibilities is predetermined (Verb > Determiner > Noun) and the first successful parse is accepted as the correct one.

Parser Implemented with the HICLASS Software System

The HICLASS Software System is a general purpose inference engine which is used for developing expert systems at Hughes Aircraft. The initial application, an assembly planning expert system, has been used in production mode to generate planning for a number of printed wiring assemblies (Liu 1985b). The knowledge base is represented as a network of nodes, where each node contains a set of rules. Control passes from one node to another by use of "Visit" or "Goto" rules.

The notes grammar was specified as an Augmented Transition Network (ATN) grammar, since this formalism is similar in form to a HICLASS network. Each ATN node corresponds to a HICLASS node and each ATN arc is also represented as a HICLASS node. For example, the ATN subnetwork which parses a Prepositional Phrase is represented as shown in Figure 2. This shows that a Prepositional Phrase (PP) consists of a Preposition (Prep) followed by a Noun Phrase (NP).

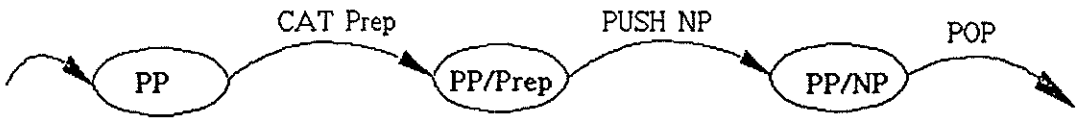


Figure 2. Augmented Transition Network for a Prepositional Phrase.

The 3 states and 3 arcs are implemented as 6 HICLASS nodes. The state nodes are simply lists of arcs. In this example, the state nodes are trivial, since each state has only one arc emanating from it. In general, there are several arcs which can be taken from a given state.

The arc nodes contain two types of rules. There are condition rules which determine if conditions are satisfied for traversing the arc. These are followed by action rules which perform the required actions when the arc is traversed. In the PP example above, the HICLASS node translation of the 'CAT Prep' arc contains a single condition rule. It checks to see if the current input word is a Preposition (i.e. that it is of CATegory Preposition). The node also has an action rule which sets the head of the current prepositional phrase to be the current input word.

The parse structure is represented as a hierarchy of frames. Each phrase type (Sentence, Noun Phrase, Verb Phrase, etc.) is represented as a frame whose slots contain the constituents of that phrase. To continue the Prepositional Phrase example, the parse structure of "TO 15 +/- 2 INCH LBS" is represented as a hierarchy of Prepositional Phrase and Noun Phrase frames:

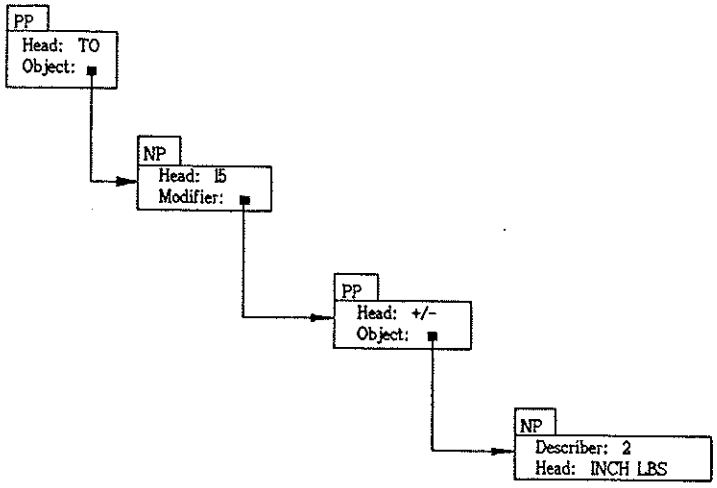


Figure 3. Hierarchy of PP and NP frames.

This diagram shows that the object of the Preposition "TO" is a Noun Phrase ("15 +/- 2 INCH LBS"), which consists of a head noun (the integer "15") followed by a Prepositional Phrase ("+/- 2 INCH LBS").

Comparison of Methods

The YACC generated parser and the one implemented with the HICLASS Software System are both able to handle the grammar outlined in this paper. However, there are two main reasons why the HICLASS version is to be preferred:

(1) ease of maintenance and expandability; and
(2) ability to parse languages generated by more powerful grammars.

The HICLASS code is a higher level description of the parser than the equivalent C code which makes up the actions in the YACC rules. This makes the HICLASS version easier to maintain and modify. The YACC rules for parsing a context free language are very simple and easy to maintain. However, I encountered problems in trying to stretch the parser coverage to a context sensitive language whose lexical items had ambiguous category membership. This led to the use of a number of flags being passed back and forth between the parser and the lexical analyzer. It seemed virtually impossible to maintain a modular approach to system development.

The problem of modularity was overcome by first designing an ATN grammar and then implementing it in HICLASS rules. A simple grammar and prototype implementation were quickly developed in order to test the frame/slot representation of parse structures. The parser was then expanded by simply adding states and arcs to the grammar specification.

The second advantage of the HICLASS implementation is in the inherent power of the ATN model. An ATN is computationally equivalent to a Turing machine. Thus, it provides a rich enough model for increasing the complexity of the notes grammar in the future.

It is not yet possible to offer an objective comparison of the speed of the two parsers. The HICLASS parser was originally developed on an interpreted version of the HICLASS inference engine. Obviously, this is significantly slower than running compiled C code generated by YACC. However, a compiled version of the HICLASS Software System is being developed. During initial testing, the HICLASS parser is performing at speeds which are adequate for our production environment.

REMAINING PROBLEMS

There are several problems which still require further study. The first problem involves the descriptive adequacy of the grammar of engineering notes. While the vast majority of notes are correctly described by our grammar, there are some notes which are not adequately handled. For the moment, we are willing to simply ignore those cases. The reason is that notes which are outside the bounds of the grammar are usually treated as exceptions by the expert system as well. Thus, even if these notes could be parsed, they could not be interpreted with the current system. We expect to expand both the syntactic and semantic treatment of notes in the future, but it should be noted that this is a minor weakness in the current system. These limitations have no adverse effects on the current expert system, but the semantic component will have to grow in order for the expert system to become more discerning.

A second problem must be handled by the engineering organization when note templates are displayed to the engineer on a computer aided design workstation. The presence of producibility rankings in the note descriptions is still quite controversial. I have assumed that a particular ranking of slot fillers will be valid in all contexts. This approach is undoubtedly overly simplistic. However, as long as it is possible to identify different contexts for which different rankings apply, the basic notion of presenting a ranked choice of slot fillers to the note writer is still valid.

SUMMARY

Machine interpretable standard engineering notes have been developed in response to two requirements for note preparation and interpretation. First, there was a need to reduce errors during note preparation. Secondly, an automated process planning expert system needs to access the information in those notes. Standard notes are a relatively constrained subset of English, but they have the same form as "free-form" engineering notes which are manually written by design engineers. A syntactic parser has been developed which accepts these notes as input and generates a parse structure as output. The first implementation of the parser was implemented in the C programming language using LEX and YACC as development tools. It generates a parse tree whose sentence level constituents are used by an assembly planning expert system.

A second version of the parser is being tested. It is implemented on the HICLASS inference engine using an Augmented Transition Network grammar. The resulting output parse structure is represented in a hierarchical frame structure and can be accessed directly by an assembly planning expert system.

BIBLIOGRAPHY

Bates, Madeleine. 1978. The theory and practice of augmented transition network grammars. In Leonard Bolc (ed.), Natual Language Communication with Computers. Berlin: Springer-Verlag. pp. 191-259.

Lehrberger, John. 1982. Automatic translation and the concept of sublanguage. In Richard Kittredge and John Lehrberger (eds.), Sublanguage: Studies of Language in Restricted Semantic Domains. Berlin: Walter de Gruyter. pp. 81-106.

Liu, David. 1984. Utilization of artificial intelligence in manufacturing. AUTOFACT 6 Conference Proceedings. pp. 2-60 to 2-78.

_____. 1985a. Intelligent manufacturing planning systems. AUTOFACT '85 Conference Proceedings Supplement, MS85-1070.

_____. 1985b. Expert systems for process planning. Computer Aided Engineering, Vol. 4, 5:65-72.

UNIX Programmer's Manual. Seventh Edition, 1979. Murray Hill, NJ: Bell Telephone Laboratories, Incorporated.

Woods, William. 1970. Transition network grammars for natural language analysis. Communications of the ACM, Vol. 13, 10:591-606.

04/30/85

Functional Specification:

Lexical Analyzer

for Standard Engineering Notes

Jim Tyhurst

0. Introduction

A lexical analyzer recognizes words in a sentence and determines which syntactic categories they belong to. It must refer to an established lexicon as well as to the input sentence. The output of the lexical analyzer must be in a format which can be used by a parser. This document provides a functional specification for a lexical analyzer by defining:

(1) strings to be recognized as tokens;
(2) delimiters used to mark word boundaries;
(3) the type of information to be output.

For further details about the lexicon and parser, see the documents "Functional Specification of a Lexicon for Standard Engineering Notes" and "An Augmented Transition Network Grammar for Standard Engineering Notes".

1. Input

Input to the lexical analyzer consists of a string which contains one or more sentences of a standard note. In addition, the lexical analyzer refers to a lexicon which gives a list of all possible words that can occur in sentences along with syntactic information about each of these words.

1.1 Sentence format and delimiters

The lexical analyzer does not have any syntactic knowledge about the structure of sentences. It simply accepts a string as input. Then using its knowledge of word boundaries, it separates the sentence into the

"tokens". I will use the term token to refer to either a word or a delimiter.

A word is any string of letters or digits which is bounded on each side by delimiters. This definition includes strings that are:

(1) "words": AFTER, OPTICAL, ITEM, etc.

(2) abbreviations: PWB, P/N (for PART NUMBER), SPEC (for SPECIFICATION), etc. Note that the abbreviation "P/N" contains a delimiter, but it is still recognized as a word, because it is listed in the lexicon.

(3) letter codes (used in specification identifiers): IPC, MIL, MIS, QQ, etc.

(4) alphanumeric codes (used for specification identifiers, part numbers, reference designators, etc.): 001D, 4A, U2, SD3589037, etc.

(5) integer values: 290, 137965, 22473, etc.

(6) real values: .070, .093, 1.060, etc.

A delimiter can be either the beginning of a string (not the first character of the string, but the boundary of the beginning of the string), the end of a string, or any of the following punctuation marks: comma (,), period (.), space ( ), semi-colon(;), slash (/), or hyphen(-).

1.2 Lexicon

Once the lexical analyzer has found a word, it then looks in the lexicon to find the syntactic features associated with that word. The category labels and feature values used in the lexicon are given in the document "Functional Specification of a Lexicon for Engineering Standard Notes".

"Words" which contain digits, i.e. words of type (4) - (6) in Section 1.1 above, are not contained in the lexicon. These words are simply assigned the category Noun without performing a lookup. In addition, they are assigned the appropriate value (alphanumeric, integer, real) for the subcategory feature "Type". Words found in the lexicon are assigned Type {word}.

2. Output

The output of the lexical analyzer is a list of tokens which make up the input sentence. Each token is associated with a set of properties which define the type of token and its features. Words and delimiters are both recognized by the lexical analyzer and output as tokens.

2.1 Format

Each token which is output contains:

(1) the substring of the sentence which is represented by the token ("ASSEMBLY", "/", "3", etc.);
(2) the category label of the token (Noun, Verb, Delimiter, etc.); and
(3) the set of subcategorization features (if any) which are associated with the token.

The subcategorization features are found in the lexicon and they vary according to the category (see the specification of the lexicon for details). Delimiters are marked with the category label "Delimiter" and have no subcategorization features.

In some cases, a particular substring will have more than one category specification. For example, many words can function either as a Noun or as a Verb. In this case, both category labels and both sets of subcategorization features must be returned by the lexical analyzer.

2.2 Words not in the lexicon

If a word is not found in the lexicon, then a warning message is printed. However, the program continues on by assuming that the unidentified word is a Noun with the feature values [+Singular], [+Plural], [+Count], [+Mass], [-Proper], [-Pronoun].

Filed as: //larry/ty.dir/doc.dir/fspec.lex.analyzer.doc

Functional Specification

STANDARD NOTES PARSER

May 7, 1985

Contents

An Augmented Transition Network Grammar

for Standard Engineering Notes

Jim Tyhurst

0. Introduction

The HICLASS system which produces computer generated planning for printed wiring assemblies has a module which interprets engineering notes on design drawings. This document is a proposal for an improved version of that module. The reason for writing a new parser is to extend the range of grammatical forms which can be recognized as well as to remove some of the arbitrary syntactic constraints which are imposed on notes by the current parser.

Sections 1 - 3 introduce the problem of interpreting standard notes and present background material for understanding the grammar. The proposed grammar is presented in Section 4 by showing each subnetwork with a detailed description of actions taken to build a parse structure at each level. The characteristics of this grammar are presented in Section 5 as they relate to implementation strategies. In Section 6, examples of sentences which can be parsed with this grammar are given. Also, the parse structures assigned to these sentences are shown. Finally, a complete set of test cases are listed in Section 7.

1. Current Standard Notes and Future Extensions

Engineering Standard Notes are a set of fixed format notes which have been defined as a part of the Producibility Based Automated Design and Manufacturing System (PADMS). These format templates specify the syntax to be followed for notes which occur on design drawings and it is up to the note writer to "fill in the blanks" with appropriate values. For example, the standard note in (1a) might be realized on a particular drawing as the note in (1b):

(1) a. SOLDER ITEM[S] (Number[s]) TO ITEM (Number) PER (Spec/Std).
b. SOLDER ITEMS 1, 2 TO ITEM 13 PER MIL-STD-454,REQUIREMENT 5.

1.1 Problems with the definition of standard notes

One important reason for standardizing the form of notes is to make them machine interpretable. It is not currently possible to interpret notes written in free form English due to the complexities of natural language structures. The current parsing system was proposed as an intermediate solution. The problem is that it places severe constraints on the form of notes. Three of these constraints are exemplified in (1a) above. First, within a particular constituent, single spaces are used to separate words, e.g. "TO ITEM (Number)" for the Object Modifier. This conforms to standard English usage. However, two blank spaces are used as delimiters between sentence constituents. Thus, double blanks come after the following sentence constituents in (1a):

- Verb: SOLDER
- Object: ITEM[S] (Number[s])
- Object Modifier: TO ITEM (Number)

A second constraint is that delimiters such as commas (,) are not treated in the same way in different constituents. This can be a source of confusion when people are writing the notes. For example, in the note of (1b), item numbers are separated by a comma and a blank, e.g. "ITEMS 1, 2". However, subparts of specifications are separated by a comma with no blank, e.g. "PER MIL-STD-454,REQUIREMENT 5".

The third constraint is that the note must begin with a verb. Therefore, notes such as (2) which do not begin with a verb cannot be parsed other than to extract the value of the Orientation variable. Notes of this type are simply treated as a whole and it is not possible to refer individually to the verb or the object.

(2) D-SHAPED PAD INDICATES (Orientation).

The grammar proposed in this document removes all three of these constraints. It does this by using (1) a more complex grammatical framework (an Augmented Transition Network as opposed to a finite state automaton); and (2) a lexical analyzer which determines the grammatical category of individual words. By complicating the grammar in this way, we

can extend the parser's capability to interpret an enhanced form of standard notes whose syntax and punctuation is more like that of standard English.

1.2 Problems with the current parser

With the current parser, each note gets divided into 7 groups or "constituents" (a group is a meaningful part of a sentence which functions as a unit):

Verb
Object
Object Modifier
Specification
Verb Modifier
Material
Comment

Any particular note will contain one or more of these groups. For example, the note: "SEAL ITEM 1 PER MIL-112 USING MATERIAL PER MIL-S-22473, GRADE IV" is parsed as:

Verb:	SEAL
Object:	1
Specification:	MIL-112
Material:	MIL-S-22473, GRADE IV

In this way, we can refer to the process involved by looking at the Verb. The items affected by the process are listed in the Object. The specification which gives details of how the process is to be done is given in the Specification. Finally, the material to be used is specified in the Material group. The parser automatically peels away the extraneous function words like "ITEM" and "PER" which only serve to identify a particular group in the sentence.

This is a reasonable approach to take, however, it needs some improvements. The current parser is not always consistent as to which group a particular phrase of the sentence is assigned to. For example, the note "TRIM LEADS TO .070 MAXIMUM WITHIN AREA INDICATED" is parsed as:

Verb:	TRIM
Object:	LEADS
Object Modifier:	TO .070 MAXIMUM
Specification:	WITHIN AREA INDICATED

The problem with this parsing, is that the phrase "WITHIN AREA INDICATED" is parsed as the Specification. It should be parsed as the Verb Modifier, since it tells where the trimming is to occur. In any case, it certainly is not a specification.

1.3 Functional sentence constituents

The new grammar parses standard notes into 9 functional sentence constituents. It is these constituents which provide a "handle" on the contents (or meaning) of the note. Sections 6 and 7 provide examples of how notes get parsed into these constituents. Each one can be characterized as follows:

Subject
(a) Something which performs a function, e.g. "D-SHAPED PAD INDICATES POSITIVE END OF CAPACITOR".
(b) Something which is assigned or equated with certain attributes, e.g. "REF DES ARE FOR REFERENCE ONLY AND DO NOT APPEAR ON COMPONENT PARTS".

Verb
(a) Denotes a process, e.g. "TRIM LEADS TO .070 MAXIMUM WITHIN AREA INDICATED".
(b) Expresses equivalence between two items, e.g. "DOT INDICATES TAB OF COMPONENTS".

Object
is something on which a process is performed, e.g. "SEAL ITEM 1 PER MIL-112 USING MATERIAL PER MIL-S-22473, GRADE IV".

Indirect Object
denotes a second object for processes which inherently involve two objects. It is always introduced by the preposition "to", e.g. "SOLDER ITEM 1 TO ITEM 13 PER MIL-STD-454, REQUIREMENT 5". Note: Not all prepositional phrases starting with "to" are indirect objects. In the note "TRIM LEADS TO .070 MAXIMUM WITHIN AREA INDICATED", the phrase "TO .070 MAXIMUM" tells how the trimming is to be done and therefore is parsed as a Verb Modifier.

Specification
identifies the specification relevant to the process denoted by the Verb, e.g. "TEST PER TP-1589".

Verb Modifier
gives additional information about how, when, where, or to what extent the process denoted by the Verb is to be carried out, e.g. "CONFORMAL COAT PWB PER QQ-N-290, CLASS 1, GRADE F AFTER ELECTRICAL TEST USING MATERIAL PER MIL-I-46058, TYPE UR" or "TORQUE ITEM 1 TO 15 +/- 2 INCH LBS".

Material
identifies a material specification, e.g. "FILLET BOND ITEMS 1, 2 PER QQ-N-290, CLASS 1, GRADE F USING MATERIAL PER MIL-S-8802, TYPE OPT, CLASS B".

Purpose
expresses the general purpose of the note or draws attention to the purpose of a particular process, e.g. "SEE DRAWING 12293955 FOR

SCHEMATIC DIAGRAM.

Comment
includes the part of a note which cannot be parsed.

These "functional categories" are only the top level of the parse structures. If necessary, HICLASS rules can refer to the internal structure of any of these phrases as described in Section 3. For example, it may be necessary to further break down the Object into its Head and Modifier.

2. Lexical Analysis

The structure of the lexicon is treated more completely in the documents: "Functional Specification of a Lexicon for Standard Engineering Notes" and "Functional Specification of a Lexical Analyzer for Standard Engineering Notes". In this section, I just mention the data structures which are expected to be returned by a lexical analyzer.

Sentences are first processed by a lexical analyzer and its output is then taken by the parser to determine the syntactic form of the input sentence. As a minimum, the lexical analyzer must determine the syntactic category of each lexical item. That is, it must determine if a word is a verb, noun, preposition, etc. In addition, some other features of a word should be returned as explained below. These features are taken from Winograd's (1983) description of ATN grammars. It seems unlikely that all of these features will be required for the limited syntax of standard notes. However, I include them here for completeness and with a view towards later expanding the syntactic coverage of notes.

Verbs are the most complicated structures. The lexical analyzer should return the root form of the word (i.e. the Infinitive form), as well as its syntactic category, type, form, and transitivity. The Type of the verb distinguishes between the different auxiliary verbs. The Form specifies the inflectional form of the word in the sentence. The Transitivity of the verb specifies the number of objects that it can occur with. The possible values for each of these features is listed below with several examples:

Verb features:
Type: Modal, Be, Do, Have, Non-Aux
Form: Infinitive, Present, 3rd-Singular-Present, Past, Present-Participle, Past-Participle
Transitivity: Intransitive, Transitive, Bitransitive

The following examples show different verbs in standard notes and how they are specified using these features.

D-SHAPED PAD INDICATES PIN ONE OF MICROCIRCUITS.
Verb: indicates
Root: indicate
Form: 3rd-Singular-Present
Type: Non-Aux
Trans: Transitive

TEST PER TP-1589.
Verb: test
Root: test
Form: Present
Type: Non-Aux
Trans: Transitive

DO NOT CONFORMAL COAT SURFACES INDICATED.

Verb: do	Verb: conformal coat
Root: do	Root: conformal coat
Form: Present	Form: Infinitive
Type: Do	Type: Non-Aux
Trans: Transitive	Trans: Transitive

The subcategorization features used for other categories are listed in the document "Functional Specification: A Lexicon for Standard Engineering Notes".

3. The ATN Framework

An Augmented Transition Network (ATN) consists of a set of states and arcs (Woods 1970, Bates 1978). Each network has a unique state called the start state and a set of states called final states. Arcs can have associated conditions which must be satisfied in order for the arc to be traversed. Arcs also have associated actions which are performed when the arc is traversed. It is these actions which build up the parse structure as constituents of the input sentence are recognized.

Each subnetwork is associated with a set of registers. These registers are used to store temporary data, e.g. flags, or the parse structure of the constituent corresponding to this subnetwork. Since the networks can be called recursively, the entire set of registers is placed on a stack when control passes from one network to another. When a constituent is correctly parsed and a POP arc is taken, then control returns to the calling network and its registers are reinstated. Each network has its own set of registers as defined under "Feature Dimensions" and "Roles" in Section 4.

As an example, consider the completed parse of the Noun Phrase, "THE VALUE DETERMINED BY TP-315". The feature "Function" and the roles "Determiner", "Describers", "Head", and "Modifier" exist as registers for the Noun Phrase network. In processing the noun phrase, "the value determined by TP-315", these registers will be put on the stack when the Adjective Phrase network is called to process "determined by TP-315". That network

has registers for the roles "Head" and "Modifier" which will in turn be placed on the stack when the Prepositional Phrase network is called to process the phrase "by TP-315". Finally, the Prepositional Phrase roles "Head" and "Object" will be placed on the stack when the Noun Phrase network is re-entered for the noun phrase "TP-315".

```
Noun Phrase: "the value determined by TP-315"
Function: Object
Determiner: ---> Determiner: "the"
Describers: nil
Head: ---> Noun: "value"
Modifier: ---> Adjective Phrase: "determined by TP-315"
               Head: ---> Adjective: "determined"
               Modifier: ---> Prepositional Phrase: "by TP-315"
                              Head: ---> Preposition: "by"
                              Object: ---> Noun Phrase: "TP-315"
                                           Function: Oblique
                                           Determiner: nil
                                           Describers: nil
                                           Head: ---> Noun: "TP-315"
                                           Modifier: nil
```

This structure is clearly a hierarchical representation of the noun phrase. That is, each constituent may itself be made up of other constituents. This allows programs which access parsed notes to refer to any level which interests them. For some procedures, it will be sufficient to know the sentence level constituents (Subject, Verb, Object, etc.). In other cases, it may be important to know the internal structure of one of these constituents. For example, it may be necessary to separate a Verb Modifier which is a Prepositional Phrase into its subparts of Preposition and Noun Phrase. Given the hierarchical structure exemplified above, both levels of analysis (sentence level vs. phrase level) are available to HICLASS rules which access parsed notes.

Six types of arcs are used in the grammar in Section 4 (Bates, 1978:197):

Category: A Category arc (CAT) may be taken if the current input word is of the syntactic category specified by the arc, e.g. "CATEGORY Verb" specifies that the arc is to be taken if the current input word belongs to the Verb category.

Word: A Word arc may be traversed if the current input word exactly matches the word specified by the arc, e.g. an arc specified as "WORD 'AND'" can only be traversed if the current input word is AND.

Member: A Member arc (MEM) may be taken if the current input word matches one of the members of the set specified by the arc, e.g. "MEMBER {',', 'AND', 'OR'}" matches a comma or the conjunctions AND or OR.

Jump: A Jump arc specifies the state to which a transition is to be made without "consuming" anything from the input string, e.g. the arc"JUMP (S/VP)" will allow a transition to the S/VP state without causing the read head to advance on the input string.

Push: A Push arc initiates a new, perhaps recursive, call to the network which begins in the indicated state and which looks for a constituent, e.g. "PUSH NP" calls the Noun Phrase network to try to find a noun phrase starting with the current input word.

Pop: A terminal state is marked by having a Pop arc which leaves it. The Pop causes control to return to the Push arc which invoked the current level of processing.

4. An ATN Grammar for Standard Notes

Each of the networks which make up the standard notes grammar is presented separately in the following subsections. First, a diagram is given which shows each state of the network and the arcs which emanate from each state. Then the conditions and actions associated with each arc are presented. Arcs are identified by listing the state from which they emanate, the type of arc, and the destination state. For example, the arc "(S/V) CATEGORY V (S/VP)" specifies a Category arc which can be traversed if the current input word is a Verb. The arc starts in state (S/V) and leads to state (S/VP).

Arcs are listed in the order in which they should be attempted. This ordering is represented in the diagram by listing the arcs clockwise in the order in which they should be tried. For example, PUSH NP is the first arc to be attempted from the Sentence (S) node. The JUMP arc is always the last arc to be tried.

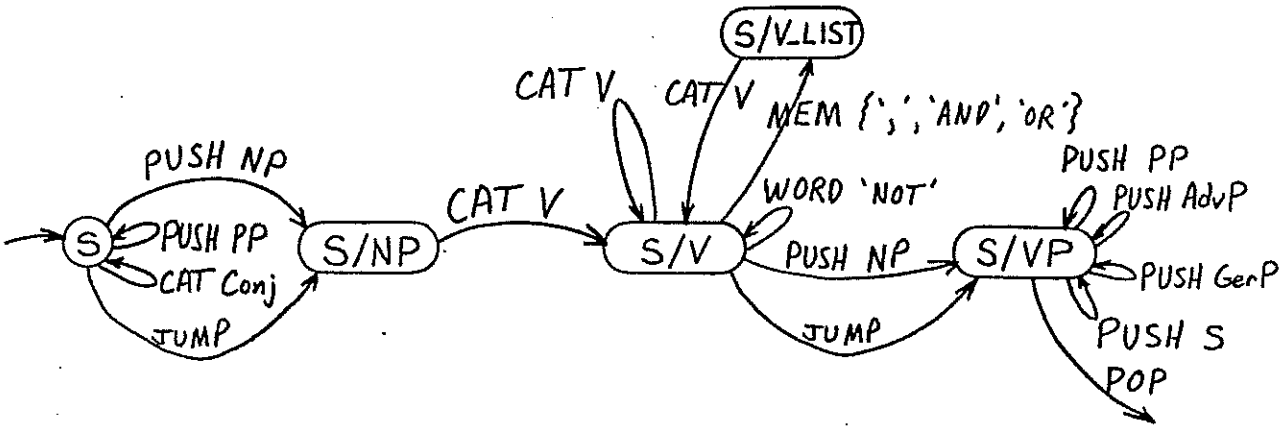
Each state in the ATN must have a unique name. The convention for state names is to list the subnetwork name, a slash, and then the name of the constituent just recognized. For example, the state (S/NP) identifies a state in the Sentence network at which a Noun Phrase has been recognized. The state (PP/NP) identifies a state in the Prepositional Phrase network at which a Noun Phrase has been recognized.

4.1 SENTENCE NETWORK

Feature Dimensions:
- none

Roles:
- Coordination
- Purpose
- Subject
- Auxiliaries
- Verb List (the last element of the list is referred to as "Verb")
- Object
- Indirect Object
- Specification
- Verb Modifier
- Material
- Sentence List

Registers:
- Verb Conjunction



Conditions and Actions on Arcs:

(S) PUSH NP (S/NP)
- Pre-Actions: Send 'Subject' to Function feature.
- Actions: Set Subject to *.

(S) PUSH PP (S)
- Actions: If Head of * is 'FOR' and Purpose is nil, then set Purpose to *; else append * to Verb Modifier.

(S) CATEGORY Conj (S)
- Actions: Append * to Coordination.

```
(S) JUMP (S/NP)
     (no conditions or actions).

(S/NP) CATEGORY V (S/V)
     Actions:      Set Verb to *.

(S/V) CATEGORY V (S/V)
     Conditions:   The Type of Verb is Be, Do, Have, or Modal.
                   Verb List has only one member.
                   {Auxiliaries are not acceptable in a verb list.}
     Actions:      Append Verb to Auxiliaries.
                   Set Verb to *.

(S/V) MEMBER {',', 'AND', 'OR'} (S/V_LIST)
     Conditions:   Verb is not nil.
     Actions:      If * is not a comma,
                   then (1) insert * before each member of Verb List (except
                            the first) which has no conjunction,
                        (2) set Verb Conjunction to *.

(S/V_LIST) CATEGORY V (S/V)
     Actions:      Append the * to Verb List.
                   If Conjunction is not nil,
                   then insert Conjunction before last Verb and
                        set Conjunction to nil.

(S/V) WORD 'NOT' (S/V)
     Conditions:   The Type of Verb is Do or Modal.
     Actions:      Append Verb to Auxiliaries.
                   Append * to Auxiliaries.
                   Set Verb to nil.

(S/V) PUSH NP (S/VP)
     Pre-Actions:  Send 'Object' to Function feature.
     Actions:      Set Object to *.

(S/V) JUMP (S/VP)
     (no conditions or actions).

(S/VP) PUSH PP (S/VP)
     Actions:      If Head of * is 'PER' and Specification is nil,
                   then set Specification to Object of *;
                   else
                     if Head of * is 'FOR' and Purpose is not nil and
                        Type of Verb is not {BE},
                     then set Purpose to *;
                     else
                       if Head of * is 'TO' and Indirect Object is nil and
                          Type of Verb is {Bitransitive},
                       then set Indirect Object to Object of *;
                       else insert * in Verb Modifier.
```

Transitivity

(S/VP) PUSH AdvP (S/VP)
Actions: Append * to Modifiers.

(S/VP) PUSH GerP (S/VP)
Actions: If Material is nil and *'s Object is 'MATERIAL',
then set Material to the Object of *'s Modifier;
else insert * in Verb Modifier.

(S/VP) PUSH S (S/VP)
Conditions: The Category of * is Conjunction.
Actions: Append * to Sentence List.

(S/VP) POP
Conditions: If Subject is not nil,
then Number of Subject must agree with Number of Verb.

The order of Auxiliaries must be: Modal < Have < Be.

If the first Auxiliary is Type Modal,
then the Form of the verb following it must be Infinitive.

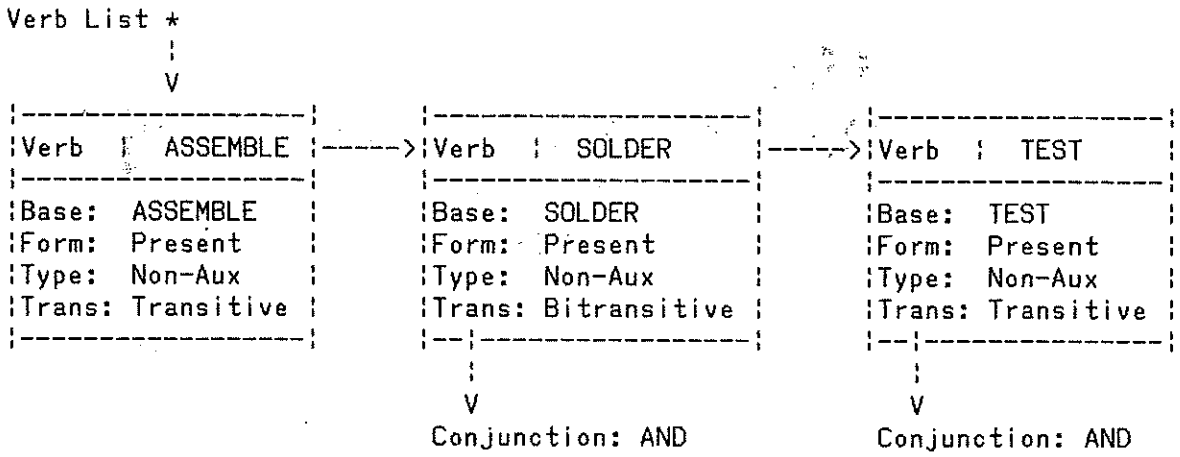
If one of the Auxiliaries is Type Have,
then the Form of the verb following it must be
Past Participle.

If the last Auxiliary is Type Be,
then the Form of Verb must be Present Participle.

Actions: Create a Verb Phrase frame with the Auxiliaries and
Verb List roles filling its slots.
Construct a Sentence frame and return Sentence structure.

Comments on the Sentence Network:

(1) The Verb List is used to represent conjunctions of main verbs. For example, "ASSEMBLE, SOLDER, AND TEST" is represented as a list where the conjunction is made explicit for each conjunct:



(2) The Sentence List can be thought of as a pointer to the following conjoined sentence. The conjoined sentence gets parsed in the arc: (S/VP) PUSH S (S/VP). For the PWA application, it would be better to restrict notes to be collections of simple sentences. A higher level module (something above the parser) can decide how to represent multiple sentences in a note. Then the parser would just be confronted with one sentence at a time.

(3) Passive sentences like "PARTIAL REFERENCE DESIGNATIONS ARE SHOWN" are parsed with BE as the main verb and the past participle is parsed in the Adjective Phrase network and it becomes the Verb Modifier:

PARTIAL REFERENCE DESIGNATIONS ARE SHOWN.
Subject: PARTIAL REFERENCE DESIGNATIONS
Verb: BE
Verb Modifier: SHOWN

(4) Adverbial phrases such as "AFTER P1 IS INSTALLED ON PWB" or "PRIOR TO INSTALLING P1 ON PWB" are parsed in the AdvP network. These are treated separately from conjoined sentences, although as shown by the first example, an adverbial phrase may contain a full sentence. In the Sentence network there is no check on the POP arc for the Form of the main verb. So gerundives like "INSERTING" are acceptable as a main verb and adverbial phrases such as "PRIOR TO INSERTING COMPONENT P1" or "PRIOR TO P1 BEING INSERTED" are partially parsed in a recursive call to the sentence network from the adverbial phrase network.

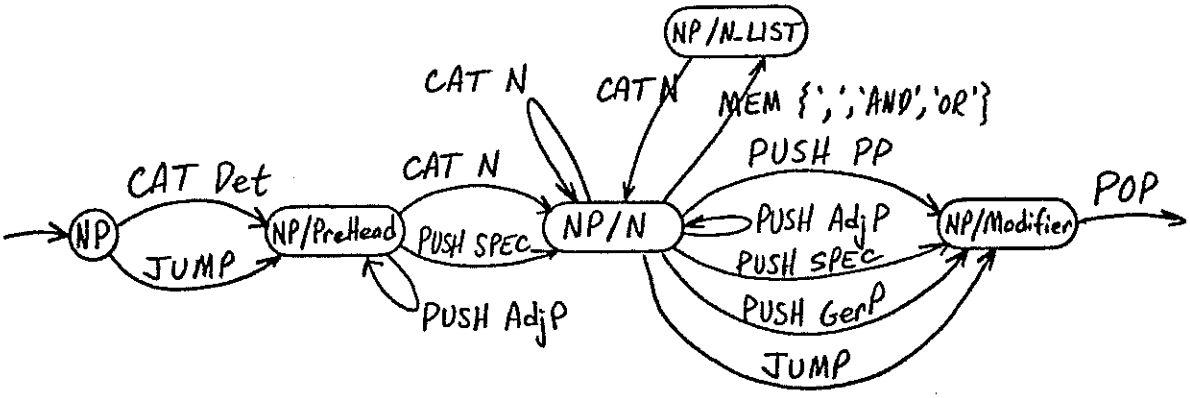
(5) The negative word NOT is placed in with the Auxiliary verbs, e.g. "DO NOT CONFORMAL COAT ..." or "... SHALL NOT EXCEED ...".

4.2 Noun Phrase Network

Feature Dimensions:
Function: {Subject, Object, Oblique} default = Object

Roles:
Determiner
Describers
Head List (last element of list is referred to as Head)
Modifier

Registers:
Noun Conjunction



Conditions and Actions on Arcs:

(NP) CATEGORY Det (NP/PreHead)
Actions: Set Determiner to *.

(NP) JUMP (NP/PreHead)
(no conditions or actions).

(NP/PreHead) CATEGORY N (NP/N)
Actions: Set Head to *.

(NP/PreHead) PUSH SPEC (NP/N)
Actions: Set Head to *.

(NP/PreHead) PUSH AdjP (NP/PreHead)
Actions: Append * to Describers;

```
(NP/N) CATEGORY N (NP/N)
     Actions:      If Modifier is not nil
                      and Category of Modifier is AdjP,
                   then append Head to Describers,
                        append Modifier to Describers,
                        set Modifier to nil.
                   Set Head to *;

(NP/N) MEMBER {',', 'AND', 'OR'} (NP/N_LIST)
     Conditions:   Head is not nil.
     Actions:      If * is not a comma,
                   then (1) insert * before each member of Head List
                            which has no conjunction,
                        (2) set Noun Conjunction to *.

(NP/N_LIST) CATEGORY N (NP/N)
     Actions:      If Modifier is nil
                   then append * to Head List;
                   else !ERROR!.
                   If Noun Conjunction is not nil,
                   then insert Noun Conjunction before last Noun and
                        set Noun Conjunction to nil.

(NP/N) PUSH PP (NP/Modifier)
     Conditions:   {Need to develop features to distinguish Object
                    Modifiers from Verb Modifiers.}
                   * is not 'BY'.
                   * is not 'PER'.
                   * is not 'TO'.
     Actions:      Set Modifier to *.

(NP/N) PUSH AdjP (NP/N)
     Conditions:   Length of Head List is 1.
     Actions:      Append * to Modifier.

(NP/N) PUSH SPEC (NP/Modifier)
     Actions:      Append Head to Describers.
                   Set Head to *.

(NP/N) PUSH GerP (NP/Modifier)
     Conditions:   * is not 'USING'.
                   {USING is a key word for the material specification.}
     Actions:      Set Modifier to *.

(NP/N) JUMP (NP/Modifier)
     (no conditions or actions).
```

(NP/Modifier) POP

Conditions: If Determiner is not nil,
then Number of Determiner must agree with Number of Head
and either (1) Count of Determiner must agree with Count of Head
or (2) Mass of Determiner must agree with Mass of Head.

Actions: Return NP structure.

Comments on Noun Phrase Network:

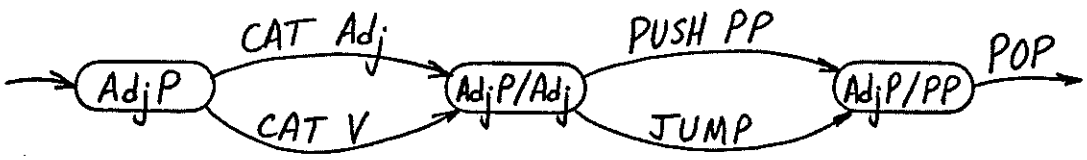
(1) The "Function" feature specifies the role this NP is filling in a sentence. It will be used to make sure that pronouns have the right case marking.

(2) The arc (NP/N) CATEGORY N (NP/N) refers to an Adjective Phrase in the Modifier. This is necessary to move a past participle from the Modifier to the Describer when it occurs between two nouns (e.g. ALL HCI IDENTIFIED PARTS). These types of Adjective Phrases are not allowed in a list of nouns as shown by arcs: (NP/N_LIST) CATEGORY N (NP/N) and (NP/N) PUSH AdjP (NP/N).

4.3 Adjective Phrase Network

Feature Dimensions:
none.

Roles:
Head
Modifier



Conditions and Actions on Arcs:

(AdjP) CATEGORY Adj (AdjP/Adj)
Actions: Set Head to *.

(AdjP) CATEGORY V (AdjP/Adj)
Conditions: Form of * is Past-Participle.
Transitivity of * is not Intransitive.
Actions: Set Head to *.

(AdjP/Adj) PUSH PP (AdjP/PP)
Conditions: The Category of Head is Verb.
Actions: Set Modifier to *.

(AdjP/Adj) JUMP (AdjP/PP)
(no conditions or actions).

(AdjP/PP) POP
Actions: Return AdjP structure.

4.4 Prepositional Phrase Network

Feature Dimensions:
none

Roles:
Head
Object

Conditions and Actions on Arcs:

(PP) CATEGORY Prep (PP/Prep)
Actions: Set Head to *.

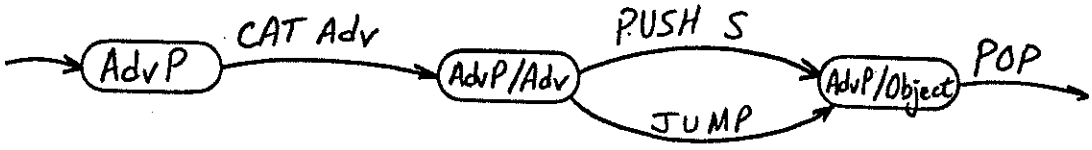
(PP/Prep) PUSH NP (PP/NP)
Pre-Actions: Send 'Oblique' to Function feature.
Actions: Set Object to *.

(PP/NP) POP
Actions: Return PP structure.

4.5 Adverbial Phrase Network

Feature Dimensions:
- none

Roles:
- Head
- Object



Conditions and Actions on Arcs:

(AdvP) CATEGORY Adv (AdvP/Adv)
Actions: Set Head to *.

(AdvP/Adv) PUSH S (AdvP/Object)
Actions: Set Object to *.

(AdvP/Adv) JUMP (AdvP/Object)
(no conditions or actions)

(AdvP/Object) POP
Actions: Return AdvP structure.

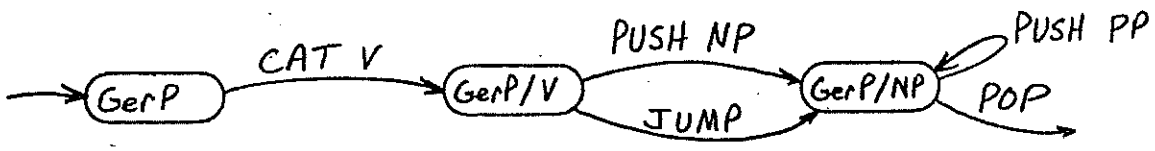
Comments on the Adverbial Phrase Network:

(1) In the Sentence network there is no check on the POP arc for the Form of the main verb. So gerundives like "INSERTING" are acceptable as a main verb and adverbial phrases such as "PRIOR TO INSERTING COMPONENT P1" or "PRIOR TO P1 BEING INSERTED" are partially parsed in the sentence network.

4.6 Gerundive Phrase Network

Feature Dimensions:
    none

Roles:
    Verb
    Object
    Modifier



Conditions and Actions on Arcs:

(GerP) CATEGORY V (GerP/V)
    Conditions: The Form of * is Present-Participle.
    Actions: Set Verb to *.

(GerP/V) PUSH NP (GerP/NP)
    Conditions: The Transitivity of * is not Intransitive.
    Pre-Actions: Send 'Object' to Function feature.
    Actions: Set Object to *.

(GerP/V) JUMP (GerP/NP)
    Conditions: The Transitivity of Verb is Intransitive.

(GerP/NP) PUSH PP (GerP/NP)
    Actions: If *'s Head is 'PER',
    then append *'s Object to Modifier;
    else append * to Modifier.

(GerP/NP) POP
    Actions: Return GerP structure.

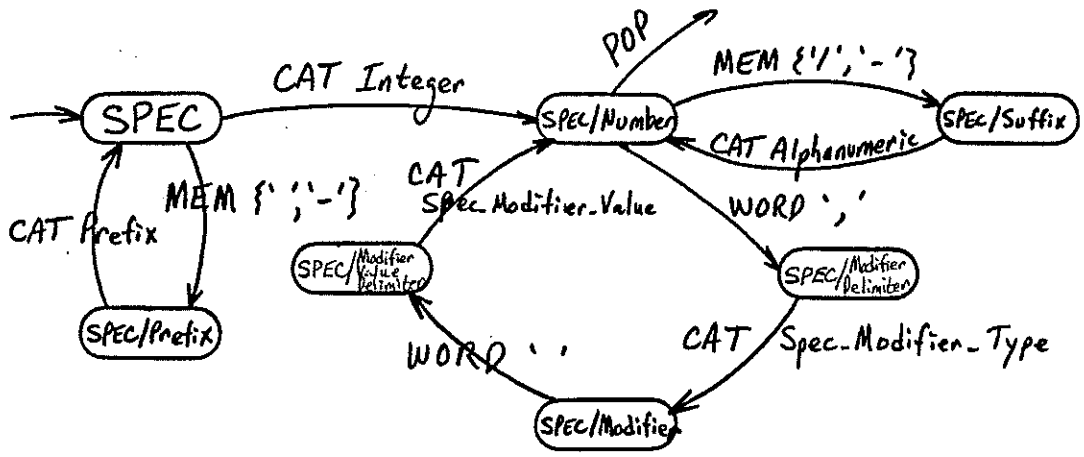
Comments on the AdjP Network:

(1) Perhaps this network can be merged with the Sentence Network. See note (4) under that network (Section 4.1) as to the processing of certain adverbial phrases containing a gerundive.

4.7 Specification Network

Feature Dimensions:
none

Roles:
Head



Conditions and Actions on Arcs:

(SPEC) CATEGORY Prefix (SPEC/Prefix)
Actions: Append * to Head.

(SPEC) CATEGORY Integer (SPEC/Number)
Actions: Append * to Head.

(SPEC/Prefix) MEMBER {' ', '-'} (SPEC)
Actions: Append * to Head.

(SPEC/Number) MEMBER {'/', '-'} (SPEC/Suffix_Delimiter)
Actions: Append * to Head.

(SPEC/Suffix_Delimiter) CATEGORY Alphanumeric (SPEC/Number)
Actions: Append * to Head.

(SPEC/Number) WORD ',' (SPEC/Modifier_Delimiter)
Actions: Append * to Head.

(SPEC/Modifier_Delimiter) CATEGORY Spec_Modifier_Type (SPEC/Modifier_Type)
Actions: Append * to Head.

(SPEC/Modifier_Type) WORD ' ' (SPEC/Modifier_Value_Delimiter)
Actions: Append * to Head.

(SPEC/Modifier_Value_Delimiter) CATEGORY Spec_Modifier_Value (SPEC/Number)
    Actions: Append * to Head.

(SPEC/Number) POP
    Actions: Return SPEC structure.

5. Constraints on the Grammar

In general, Augmented Transition Networks are non-deterministic. Therefore, when they are implemented, there must be some mechanism for back-tracking or parallel processing in order to handle the different possibilities which must be followed at any particular branching point. The ATN grammar presented here as been designed specifically to be deterministic. Thus, once an arc has been taken to a new state, the parser can never return to the previous state to try a different solution. Obviously, this significantly reduces its computational power. However, I believe this simplification can be made, because of the simple structures that we are dealing with in standard notes.

6. Sample Parse Structures

Examples of the types of sentences generated by this grammar are presented in this section. A more complete set of examples is given in Section 7. For the most part, examples will only show the sentence level constituents. Section 6.1 gives some examples of this type. Section 6.2 gives more detailed parse structures by showing examples of how the sentence level constituents are further broken down.

6.1 Sentence level constituents

The examples are divided into three categories of enhancements:

- Current notes which will be parsed just as they are now, except that delimiter constraints will be removed.

- Current notes which are not verb initial and therefore are not currently parsed in a satisfactory manner. These are completely parsed in the proposed grammar.

- New syntactic formats which do not currently exist as standard notes, but which will be handled by the proposed grammar.

6.1.1 Simple verb initial notes

The following two notes demonstrate how the new grammar handles existing standard notes. Each of these is verb initial. The first one shows that a bitransitive verb like SOLDER takes two objects:

```
SOLDER ITEM 1 TO ITEM 13 PER MIL-STD-454, REQUIREMENT 5
   Verb:              SOLDER
   Object:            ITEM 1
   Indirect Object:   ITEM 13
   Specification:     MIL-STD-454, REQUIREMENT 5
```

In the next note, the prepositional phrase "AFTER ELECTRICAL TEST" defines the sequence of operations. Since it describes when the process is to take place, it is parsed as a Verb Modifier.

```
CONFORMAL COAT PWB PER QQ-N-290, CLASS 1, GRADE F
  AFTER ELECTRICAL TEST
  USING MATERIAL PER MIL-I-46058, TYPE UR
   Verb:              CONFORMAL COAT
   Object:            PWB
   Specification:     QQ-N-290, CLASS 1, GRADE F
   Verb Modifier:     AFTER ELECTRICAL TEST
   Material:          MIL-I-46058, TYPE UR
```

6.1.2 Subject - verb notes

Some of the standard notes are not parsed correctly by the current HICLASS software, because they do not begin with a verb. The new parser will be able to handle these constructions as demonstrated by the following note:

```
PART NUMBER WILL FALL WITHIN RANGE OF M-15 THROUGH M-35
   Subject:           PART NUMBER
   Verb:              WILL FALL
   Verb Modifier:     WITHIN RANGE OF M-15 THROUGH M-35
```

6.1.3 New note forms

One of the constructions which is not handled well by the current parser is the use of a prepositional phrase to indicate sequencing information. The following example shows that "PRIOR TO INSTALLATION IN PWB" is correctly parsed as a Verb Modifier:

```
CUT OFF LEAD NUMBER 4 OF U10 APPROXIMATELY FLUSH
  WITH THE BOTTOM OF THE COMPONENT BODY PRIOR TO INSTALLATION IN PWB.
   Verb:              CUT OFF
   Object:            LEAD NUMBER 4 OF U10
   Verb Modifiers:    APPROXIMATELY FLUSH WITH THE
                        BOTTOM OF THE COMPONENT BODY
                      PRIOR TO INSTALLATION IN PWB
```

As mentioned previously, Subject - Verb notes are a problem for the current parser. The next two notes show common examples of Subject - Verb - Object notes which must be correctly parsed in order to identify the equivalence between the Subject and Object.

```
ALTERNATE COMPONENTS FOR P/N M38510/11604BCC ARE P/N DG303AP,
  DG303AAP, AND DG303AAK
   Subject:          ALTERNATE COMPONENTS FOR P/N M38510/11604BCC
   Verb:             BE
   Objects:          P/N DG303AP
                     AND P/N DG303AAP
                     AND P/N DG303AAK

SERIAL NUMBERS SHALL BE CONSECUTIVE STARTING WITH 0001
   Subject:          SERIAL NUMBERS
   Verb:             SHALL BE
   Object:           CONSECUTIVE STARTING WITH 0001
```

6.2 Phrase level constituents

A sentence has internal structure denoted by the functional roles, Subject, Verb, etc. Phrases also have internal structure. The roles specified for the subnetworks in Section 4 are phrase level constituents. For example, the Subject position of a sentence is filled by a Noun Phrase which is itself made up of a Determiner, Describers, Head, and Modifier. Examples of the structure of a few Noun Phrases are given below:

```
ALL HCI IDENTIFIED PARTS
  Determiner:  ALL
  Describers:  HCI IDENTIFIED
  Head:        PARTS

INDICATED HIGH VOLTAGE AREAS
  Describers:  INDICATED HIGH VOLTAGE
  Head:        AREAS

ACCEPTANCE TEST REQUIREMENTS
  Describers:  ACCEPTANCE TEST
  Head:        REQUIREMENTS

BONDING MATERIAL INSIDE SPACER
  Describers:  BONDING
  Head:        MATERIAL
  Modifiers:   INSIDE SPACER

FARSIDE OF ITEM 1
  Head:        FARSIDE
  Modifiers:   OF ITEM 1

PIN 5 SOLDERTAIL
  Describers:  PIN 5
  Head:        SOLDERTAIL
```

THE DARK-COLORED CONDUCTIVE COATING COVERING THE DIELECTRIC
  Determiner: THE
  Describers: DARK-COLORED CONDUCTIVE
  Head: COATING
  Modifiers: COVERING THE DIELECTRIC

7. Test Cases

Test cases are presented in two ways. Section 7.1 lists examples according to their constituent structure. Section 7.2 then lists the current standard notes for the printed wiring assembly (PWA) domain. It also gives a number of examples which are not standard notes, but which have been taken from PWA drawings and which are correctly parsed.

7.1 Examples of different groups

Verb initial notes

With Specification:
ASSEMBLE AND SOLDER PER MIL-STD-454, REQUIREMENT 5
  Verb: ASSEMBLE AND SOLDER
  Specification: MIL-STD-454, REQUIREMENT 5

With Verb Modifier:
TORQUE ITEM 1 TO 4 +/- 1 INCH LBS
  Verb: TORQUE
  Object: ITEM 1
  Verb Modifier: TO 4 +/- 1 INCH LBS

With Material:
PRIME AND SEAL ITEM 3 USING MATERIAL PER MIL-S2247, GRADE C
  Verb: PRIME AND SEAL
  Object: ITEM 3
  Material: MIL-S2247, GRADE C

With Specification and Verb Modifier:
CONFORMAL COAT PER SPECIFICATION 12296050 AFTER ELECTRICAL TEST
  Verb: CONFORMAL COAT
  Specification: 12296050
  Verb Modifier: AFTER ELECTRICAL TEST

With Specification and Material:
SEAL ITEM 1 PER MIL-112 USING MATERIAL PER MIL-S-22473, GRADE IV
  Verb: SEAL
  Object: ITEM 1
  Specification: MIL-112
  Material: MIL-S-22473, GRADE IV

With Specification, Verb Modifier, and Material:
CONFORMAL COAT PER QQ-N-290, CLASS 1, GRADE F
AFTER ELECTRICAL TEST USING MATERIAL PER MIL-I-46058, TYPE UR
Verb: CONFORMAL COAT
Specification: QQ-N-290, CLASS 1, GRADE F
Verb Modifier: AFTER ELECTRICAL TEST
Material: MIL-I-46058, TYPE UR

With Indirect Object and Specification:
SOLDER ITEM 1 TO ITEM 13 PER MIL-STD-454, REQUIREMENT 5
Verb: SOLDER
Object: ITEM 1
Indirect Object: ITEM 13
Specification: MIL-STD-454, REQUIREMENT 5

With Indirect Object, Specification, and Material:
BOND ITEM 1 TO ITEM 2 PER QQ-N-290, CLASS 1, GRADE F
USING MATERIAL PER MIL-S-8802, TYPE O.
Verb: BOND
Object: ITEM 1
Indirect Object: ITEM 2
Specification: QQ-N-290, CLASS 1, GRADE F
Material: MIL-S-8802, TYPE O

With Purpose in sentence initial position:
FOR COMPLETE DESIGNATION, PREFIX WITH UNIT NUMBER OR SUBASSEMBLY DESIGNATION
Purpose: FOR COMPLETE DESIGNATION
Verb: PREFIX
Verb Modifier: WITH UNIT NUMBER OR SUBASSEMBLY DESIGNATION

With Purpose in sentence final position:
SEE DRAWING 12293955 FOR ASSEMBLY
Purpose: FOR ASSEMBLY
Verb: SEE
Object: DRAWING 12293955

With Modifier in sentence initial position:
PRIOR TO INSTALLING P1 ON PWB, SHORTEN PIN 5 SOLDERTAIL
TO .12 MINIMUM
Verb: SHORTEN
Object: PIN 5 SOLDERTAIL
Verb Modifier: TO .12 MINIMUM
PRIOR TO INSTALLING P1 ON PWB

With Modifier in sentence final position:
CUT OFF LEAD NUMBER 4 OF U10 APPROXIMATELY FLUSH
WITH THE BOTTOM OF THE COMPONENT BODY PRIOR TO INSTALLATION IN PWB.
Verb: CUT OFF
Object: LEAD NUMBER 4 OF U10
Verb Modifier: APPROXIMATELY FLUSH WITH THE
BOTTOM OF THE COMPONENT BODY
PRIOR TO INSTALLATION IN PWB

Subject - Verb notes

With no Object:
PART NUMBER WILL FALL WITHIN RANGE OF M-15 THROUGH M-35
Subject: PART NUMBER
Verb: WILL FALL
Verb Modifier: WITHIN RANGE OF M-15 THROUGH M-35

With Object:
D-SHAPED PAD INDICATES PIN ONE OF MICROCIRCUIT
Subject: D-SHAPED PAD
Verb: INDICATE
Object: PIN ONE OF MICROCIRCUIT

THIS DRAWING DEPICTS NUCLEAR HARDNESS
Subject: THIS DRAWING
Verb: DEPICT
Object: NUCLEAR HARDNESS

7.2 Standard notes and future extensions

The parse structures for a representative sample of current standard notes are given below. Notes which have more than one sentence are divided so that each sentence gets parsed individually. Some suggested parse structures are also given for notes which are not currently considered as standard notes.

Current standard notes

217 SEE DRAWING 12293955 FOR SCHEMATIC DIAGRAM
Purpose: FOR SCHEMATIC DIAGRAM
Verb: SEE
Object: DRAWING 12293955

228 BOND ITEMS 32 AND 33 TO ITEM 1 PER 12296065, TYPE 2, CLASS A
Verb: BOND
Objects: ITEMS 32
AND 33
Indirect Object: ITEM 1
Specification: 12296065, TYPE 2, CLASS A

235 VENDOR ITEM - SEE SPEC CONTROL DOCUMENT
?? SEE SPEC CONTROL DOCUMENT FOR VENDOR ITEM
Purpose: FOR VENDOR ITEM
Verb: SEE
Object: SPEC CONTROL DOCUMENT

239 REFERENCE DESIGNATIONS ARE FOR REFERENCE ONLY AND DO NOT APPEAR ON COMPONENT PARTS

Subject:	REFERENCE DESIGNATIONS
Verb:	BE
Verb Modifier:	FOR REFERENCE ONLY
Coordination:	AND
Subject:	REFERENCE DESIGNATIONS
Verb:	DO NOT APPEAR
Verb Modifier:	ON COMPONENT PARTS

240a PARTIAL REFERENCE DESIGNATIONS ARE SHOWN.

Subject:	PARTIAL REFERENCE DESIGNATIONS
Verb:	BE
Verb Modifier:	SHOWN

240b FOR COMPLETE DESIGNATION, PREFIX WITH UNIT NUMBER OR SUBASSEMBLY DESIGNATION

Purpose:	FOR COMPLETE DESIGNATION
Verb:	PREFIX WITH
Object:	UNIT NUMBER OR SUBASSEMBLY DESIGNATION

241 ASSEMBLE AND SOLDER PWB PER MIL-STD-454, REQUIREMENT 5

Verb:	ASSEMBLE AND SOLDER
Object:	PWB
Specification:	MIL-STD-454, REQUIREMENT 5

242 HANDLE STATIC SENSITIVE DEVICES REF DES U1 AND U2 PER HP 10-39

Verb:	HANDLE
Objects:	STATIC SENSITIVE DEVICES REF DES U1 AND REF DES U2
Specification:	HP 10-39

244a SELECT PART NUMBER FROM VALUE DETERMINED BY TP-315.

Verb:	SELECT
Object:	PART NUMBER
Verb Modifier:	FROM VALUE DETERMINED BY TP-315

244b PART NUMBER WILL FALL WITHIN RANGE OF M-15 THROUGH M-35

Subject:	PART NUMBER
Verb:	WILL FALL
Verb Modifier:	WITHIN RANGE OF M-15 THROUGH M-35

246 BAND ON DIODES INDICATE CATHODE END

Subject:	BAND ON DIODES
Verb:	INDICATE
Object:	CATHODE END

247 D-SHAPED PAD INDICATES PIN ONE OF MICROCIRCUIT
Subject: D-SHAPED PAD
Verb: INDICATE
Object: PIN ONE OF MICROCIRCUIT

248 DOT INDICATES TAB OF COMPONENT
Subject: DOT
Verb: INDICATE
Object: TAB OF COMPONENT

250 INSTALL ITEMS 1 AND 2 WITHIN .50 MAXIMUM ABOVE PWB SURFACE
Verb: INSTALL
Object: ITEMS 1
AND 2
Verb Modifiers: WITHIN .50 MAXIMUM
ABOVE PWB SURFACE

251 TRIM LEADS TO .070 MAXIMUM WITHIN AREA INDICATED
Verb: TRIM
Object: LEADS
Verb Modifiers: TO .070 MAXIMUM
WITHIN AREA INDICATED

252 TORQUE ITEM 1 TO 4 +/- 1 INCH LBS
Verb: TORQUE
Object: ITEM 1
Verb Modifier: TO 4 +/- 1 INCH LBS

253 SEAL ITEM 1 PER MIL-112 USING MATERIAL PER MIL-S-22473, GRADE IV
Verb: SEAL
Object: ITEM 1
Specification: MIL-112
Material: MIL-S-22473, GRADE IV

255 SOLDER ITEM 1 TO ITEM 13 PER MIL-STD-454, REQUIREMENT 5
Verb: SOLDER
Object: ITEM 1
Indirect Object: ITEM 13
Specification: MIL-STD-454, REQUIREMENT 5

256a SPOT BOND ITEM 1 PER QQ-N-290, CLASS 1, GRADE F
EXCEPT CLEAN PER QQ-N-290,CLASS 2, GRADE D
USING MATERIAL PER MIL-S-8802, TYPE O.
Verb: BOND
Object: ITEM 1
Specification: QQ-N-290, CLASS 1, GRADE F
EXCEPT CLEAN PER QQ-N-290,CLASS 2, GRADE D
Material: MIL-S-8802, TYPE O

256b MAXIMUM CURE TEMPERATURE IS 275 DEGREES F
Subject: MAXIMUM CURE TEMPERATURE
Verb: BE
Object: 275 DEGREES F

257 FILLET BOND REF DES C3 TO ITEM 1
PER SPECIFICATION 12293452, TYPE 1, CLASS OPTIONAL
Verb: FILLET BOND
Object: REF DES C3
Indirect Object: ITEM 1
Specification: 12293452, TYPE 1, CLASS OPTIONAL

259 TEST PER TP-1589
Verb: TEST
Specification: TP-1589

*** Correct form for this note??? ***
260 STENCIL SERIAL NUMBER PER MIL-STD-130
(WITH) CHARACTERS .06 HIGH USING MATERIAL PER MIL-I-43553
Verb: STENCIL
Object: SERIAL NUMBER
Specification: MIL-STD-130
Verb Modifier: (WITH) CHARACTERS .06 HIGH
Material: MIL-I-43553

262 CONFORMAL COAT PWB PER QQ-N-290, CLASS 1, GRADE F
AFTER ELECTRICAL TEST USING MATERIAL PER MIL-I-46058, TYPE UR
Verb: CONFORMAL COAT
Object: PWB
Specification: QQ-N-290, CLASS 1, GRADE F
Verb Modifier: AFTER ELECTRICAL TEST
Material: MIL-I-46058, TYPE UR

263 DO NOT CONFORMAL COAT SURFACES INDICATED
Verb: DO NOT CONFORMAL COAT
Object: SURFACES INDICATED

<u>New notes</u> (under consideration)

CUT OFF LEAD NUMBER 4 OF U10 APPROXIMATELY FLUSH
WITH THE BOTTOM OF THE COMPONENT BODY PRIOR TO INSTALLATION IN PWB.
Verb: CUT OFF
Object: LEAD NUMBER 4 OF U10
Verb Modifiers: APPROXIMATELY FLUSH WITH THE
BOTTOM OF THE COMPONENT BODY
PRIOR TO INSTALLATION IN PWB

ALTERNATE COMPONENT FOR M39014/05-2615 IS M39014/05-2613
Subject: ALTERNATE COMPONENT FOR M39014/05-2615
Verb: BE
Object: M39014/05-2613

ALTERNATE COMPONENTS FOR P/N M38510/11604BCC ARE P/N DG303AP,
DG303AAP, AND DG303AAK
Subject: ALTERNATE COMPONENTS FOR P/N M38510/11604BCC
Verb: BE
Objects: P/N DG303AP
AND P/N DG303AAP
AND P/N DG303AAK

SERIAL NUMBERS SHALL BE CONSECUTIVE STARTING WITH 0001
Subject: SERIAL NUMBERS
Verb: SHALL BE
Object: CONSECUTIVE STARTING WITH 0001

LOCATE APPROXIMATELY AS SHOWN
Verb: LOCATE
Verb Modifier: APPROXIMATELY AS SHOWN

REMOVE THE DARK-COLORED CONDUCTIVE COATING COVERING THE
DIELECTRIC PER SPECIFICATION 12296099, PARAGRAPH 3.4.16
Verb: REMOVE
Object: THE DARK-COLORED CONDUCTIVE COATING
COVERING THE DIELECTRIC
Specification: 12296099, PARAGRAPH 3.4.16

BONDING MATERIAL INSIDE SPACER AND HOLE IN BOARD MUST BE REMOVED
Subject: BONDING MATERIAL INSIDE SPACER AND HOLE IN BOARD
Verb: MUST BE
Object: REMOVED

*** The preceding note is better expressed as:
REMOVE BONDING MATERIAL INSIDE SPACER AND HOLE IN BOARD
Verb: REMOVE
Object: BONDING MATERIAL INSIDE SPACER AND HOLE IN BOARD

REMOVE TIP OF CONTACT AFTER CONFORMAL COATING
Verb: REMOVE
Object: TIP OF CONTACT
Verb Modifier: AFTER CONFORMAL COATING

TOUCH-UP ANY EXPOSED COPPER SURFACE ON CONTACT
Verb: TOUCH-UP
Object: ANY EXPOSED COPPER SURFACE ON CONTACT

ASSEMBLY SHALL MEET THE REQUIREMENTS OF SPECIFICATION 82577-SC12271922
Subject: ASSEMBLY
Verb: SHALL MEET
Object: THE REQUIREMENTS OF SPECIFICATION 82577-SC12271922

ALL HCI IDENTIFIED PARTS MUST MEET THE REQUIREMENTS OF
NUCLEAR HARDNESS ASSURANCE DOCUMENT 01417-KZ-79/8113-012

Subject:	ALL HCI IDENTIFIED PARTS
Verb:	MUST MEET
Object:	THE REQUIREMENTS OF NUCLEAR HARDNESS ASSURANCE DOCUMENT 01417-KZ-79/8113-012

IF C40 DOES NOT MEET THE PHYSICAL REQUIREMENTS
USE TILT TO MEET MAXIMUM DIMENSION AS SHOWN

Purpose:	IF C40 DOES NOT MEET THE PHYSICAL REQUIREMENTS
Verb:	USE
Object:	TILT
Verb Modifier:	TO MEET MAXIMUM DIMENSION AS SHOWN

PLUG PIN 40 MOUNTING HOLE IN PWB DURING WAVE SOLDERING

Verb:	PLUG
Object:	PIN 40 MOUNTING HOLE IN PWB
Verb Modifier:	DURING WAVE SOLDERING

INTERPRET DRAWING IN ACCORDANCE WITH STANDARDS PRESCRIBED
BY DOD-STD-100C

Verb:	INTERPRET
Object:	DRAWING
Verb Modifier:	IN ACCORDANCE WITH STANDARDS PRESCRIBED BY DOD-STD-100C

Abbreviations

Adj	Adjective (complete, partial, schematic, functional)
AdjP	Adjective Phrase ("determined by TP-315")
Adv	Adverb (evenly, continuously, after, later)
AdvP	Adverbial Phrase ("after P1 is inserted", "prior to wave soldering")
CAT	Category arc
Det	Determiner (the, a, some, every)
GerP	Gerundive Phrase ("requiring inspection", "using material per MIL-S-22473, GRADE IV")
MEM	Member arc
N	Noun (requirement, number, degree)
NP	Noun Phrase ("all identified parts", "electrical test", "tip of contact")
PP	Prepositional Phrase ("on diodes", "of microcircuits", "after electrical test")
Prep	Preposition (by, for, to, per, on, of, after, prior to)
V	Verb (solder, remove, bond, conformal coat)
VP	Verb Phrase ("shall meet", "do not conformal coat", "shall be")

References

Bates, Madeleine. 1978. The theory and practice of augmented transition network grammars. In Leonard Bolc (ed.), Natural Language Communication with Computers. Berlin: Springer-Verlag. pp. 191-259.

EDSG Drawing Notes: Producibility Based Automated Design and Manufacturing System. October 26, 1984.

Tyhurst, Jim. Functional Specification: A Lexical Analyzer for Standard Engineering Notes.

Tyhurst, Jim. Functional Specification: A Lexicon for Standard Engineering Notes.

Winograd, Terry. 1983. Language as a Cognitive Process, Volume I: Syntax. Reading, Mass: Addison-Wesley Publishing Company.

Woods, William A. 1970. Transition Network Grammars for natural language analysis. Communications of the ACM, Vol. 13, 10:591-606.

Filed as: //larry/ty.dir/doc.dir/fspec.atn.grammar.doc

Functional Specification:

Interactive Lexical Acquisition Program

for Standard Engineering Notes

Jim Tyhurst

0. Introduction

This document describes a program for building and maintaining the lexicon database which will be required by the new standard notes parser. It is an interactive program which prompts the user for the required features. The user does not have to be sophisticated in grammatical terminology nor in the structure of the lexicon to use this program. It prompts for information by giving examples and then asks the user to tell whether the form is correct or not. From these answers, the program can extract the required information for the lexicon. The structure of the lexicon which is assumed here is described in the document "Functional Specification of a Lexicon for Standard Engineering Notes".

Four functions provide access to the lexicon: ADD is used to place a new word in the lexicon. UPDATE allows a lexical entry to be modified. DELETE allows the user to remove a word from the lexicon. LIST prints the lexicon or portions of it.

1. Getting Started

The program is called by typing its name and the name of the lexicon database to be accessed. (e.g. "interlex std.notes.lexicon"). If no database is specified, then the program will query the user to supply its

name. The program begins by printing a heading with the version number, time, and date. It then offers a menu of the different functions. The following example shows the opening sequence of the program where the user does not specify the database name in the initial program call:

```
$interlex
INTERLEX 1.0          10:25  04/12/85
Lexicon file name ? STD.NOTE.LEXICON

Options:  Add, Delete, List, Update, Exit.
Option ? ADD
```

2. Add

The Add function provides a facility for entering new words into the lexicon. The program always prompts for a new word and its category. The category labels are conventional grammatical terms. If the user does not know the category for a particular word, then he can look up the word and category in any dictionary. Depending on the category, the program then prompts for further information. If the word belongs to more than one category (e.g. SOLDER is both a Noun and a Verb), then the user can type both categories separated by a comma.

2.1 Nouns

Nouns have two forms, one for singular and one for plural (e.g. PLAN/PLANS). In addition, the lexicon contains subcategorization features for the Count/Mass distinction. The values for these subcategorization features are determined by the user's response to several questions. Each of these features is discussed in the following sections.

2.1.1 Singular/Plural

The program assumes that the singular form is entered at the "New word?" prompt. It then repeats the word for verification by the user. The <Return> key is used for this purpose. The program then generates a guess of the plural form of the noun (see the Appendix for further discussion). If either the singular or plural form does not exist, then the user should enter an asterisk (*) to mark the form as unacceptable.

For the majority of cases, the singular form will be the same as the word entered at the "New word?" prompt and the program will correctly generate the plural form of the noun. Thus, the user will simply verify both forms as shown below. In the examples that follow, the words typed by the user are shown by the underline. The <Return> key is assumed at the end of each line, although it it explicitly shown in the cases where the user types no text other than the <Return>.

```
Option    ? ADD
New word  ? PLAN
Category  ? NOUN
Singular:   PLAN   ? <Return>
Plural:     PLANS  ? <Return>
```

In the following example, the word "electricity" only occurs in singular form. Therefore, ELECTRICITY will be specified as [+Singular], [-Plural]. The program prompts the user with a plural form, "electricities", but the user then types "*" to mark this as a form which does not exist:

```
New word  ? ELECTRICITY
Category  ? NOUN
Singular    ELECTRICITY   ? <Return>
Plural      ELECTRICITIES ? *
```

In the next example, the form looks like a plural, because it has an "s" at the end. However, there is no contrast between a singular and plural form. So PLIERS would be specified in the lexicon as [+Singular] and [+Plural].

```
New word  ? PLIERS
Category  ? NOUN
Singular    PLIERS   ? <Return>
Plural      PLIERSES ? PLIERS
```

2.1.2 Proper/Pronoun

If the user indicates that either the singular or plural form of a noun does not exist, then the program will prompt to see if this is a Proper Noun (e.g. a name such as "Hughes" or "HP 10-39") or a Pronoun (e.g. "it", "they"). If the answer to the first question, "Is this a Proper Noun", is "yes", then there is no need to ask if it is a Pronoun. Proper Nouns and Pronouns are always specified [-Count], [-Mass] (see next section for further discussion of these features).

```
New word  ? HUGHES
Category  ? NOUN
Singular    HUGHES   ? <Return>
Plural      HUGHESES ? *
Is this a Proper Noun (yes) ? <Return>
```

```
New word  ? THEY
Category  ? NOUN
Singular    THEY   ? *
Plural      THEYS  ? THEY
Is this a Proper Noun (yes) ? NO
Is this a Pronoun (yes) ? <Return>
```

2.1.3 Count/Mass

Nouns can be subcategorized by the two features [Count] and [Mass]. Nouns which are [+Count] can have plural forms and they cooccur with numerals and certain quantifiers in noun phrases. Nouns which are [+Mass] do not occur with numerals (e.g. "3 electricities" is ungrammatical) and they cooccur with a different set of quantifiers than do [+Count] nouns. We will test for these features by using two questions. The first question, "Is it grammatical to say 'HOW MANY (plural form) ARE REQUIRED'?", tests for the [Count] feature. A "yes" answer means the noun is [+Count], whereas a "no" answer means the noun is [-Count]. The second question, "Is it grammatical to say 'HOW MUCH (singular form) IS REQUIRED'?", tests for the [Mass] feature. A "yes" answer means the noun is [+Mass], whereas a "no" answer means the noun is [-Mass]. A "yes" answer is assumed as the default in the following examples. In order to indicate that a sentence is ungrammatical (i.e. it is not "ok"), the user types an asterisk (*).

```
New word  ? PART
Category  ? NOUN
Singular:   PART   ? <Return>
Plural:     PARTS ? <Return>
HOW MANY PARTS ARE REQUIRED (ok) ? <Return>
HOW MUCH PART IS REQUIRED (ok) ? *

New word  ? MATERIAL
Category  ? NOUN
Singular:   MATERIAL  ? <Return>
Plural:     MATERIALS ? <Return>
HOW MANY MATERIALS ARE REQUIRED?  (ok) ? <Return>
HOW MUCH MATERIAL IS REQUIRED?  (ok) ? <Return>
```

Note that if no plural form is entered for a noun, there is no need to ask the question about MANY (see the example for STEAM in Section 6). In the preceding examples, PART will be specified in the lexicon as [+Count] and [-Mass]. The noun MATERIAL will be specified as both [+Count] and [+Mass]. Other nouns, such as STEAM, will be [-Count] and [+Mass]. The only nouns which will be specified as "-" for both features are Proper Nouns and Pronouns:

Count	Mass	Examples
+	+	MATERIAL
+	-	PART, COMPONENT, TAB
-	+	SOLDER, ELECTRICITY, GLUE, STEAM
-	-	IT, THEY, HUGHES, BFVS

2.1.4 Synonyms

All nouns, except for Pronouns, may have synonyms. These are abbreviations for words which have the same meaning. If there is more than one synonym, they should be separated by a comma (,). The plural of the abbreviation is formed by adding 's to the end of the singular.

```
New word  ? PART NUMBER
Category  ? NOUN
Singular:   PART NUMBER  ? <Return>
Plural:     PART NUMBERS ? <Return>
HOW MANY PART NUMBERS ARE REQUIRED?  (ok) ? <Return>
HOW MUCH PART NUMBER IS REQUIRED?  (ok) ? *
Synonym:  ? P/N, PART NO.
   Singular: P/N ? <Return>
   Plural: P/N's ? <Return>
   Singular: PART NO. ? <Return>
   Plural: PART NO.'S ? PART NOS.
```

2.2 Determiners

Determiners agree with the head noun in a noun phrase for certain features. The program will prompt the user to make grammatical judgments in order to determine the values of these features. There are three questions to check if the determiner can cooccur with a singular, plural, or mass noun. All three questions must be asked, since the value of one feature does not determine the value of another:

```
(determiner) PART IS STATIC SENSITIVE.  (ok)  ?
(determiner) PARTS ARE STATIC SENSITIVE.  (ok) ?
(determiner) STEAM MUST REACH EVERY SURFACE OF THE PART.  (ok) ?
```

The following examples show that the determiner THESE is [-Singular], [+Plural], and [-Mass], while the determiner SOME is [+Singular], [+Plural], and [+Mass]:

```
New word  ? THESE
Category  ? DETERMINER
THESE PART IS STATIC SENSITIVE.  (ok)  ? *
THESE PARTS ARE STATIC SENSITIVE.  (ok) ? <Return>
THESE STEAM MUST REACH EVERY SURFACE OF THE PART.  (ok) ? *

New word  ? SOME
Category  ? DETERMINER
SOME PART IS STATIC SENSITIVE.  (ok)  ? <Return>
SOME PARTS ARE STATIC SENSITIVE.  (ok) ? <Return>
SOME STEAM MUST REACH EVERY SURFACE OF THE PART.  (ok) ? <Return>
```

2.3 Verbs

Verbs are more complicated in that they are specified for multi-valued features. However, these values can also be determined by prompting the user with yes/no questions.

2.3.1 Type

Each verb must be specified for one of the types Modal, Be, Do, Have, or Non-Auxiliary. Since the class of Auxiliary verbs is very limited, we need not prompt the user for this feature. The program can refer to an internal list to determine the Type of the verb:

Modal: {will, would, shall, should, can, could, may, might, must, ought}

Be: {be}

Do: {do}

Have: {have}

Non-Aux: All verbs other than the ones listed in the 4 preceding types.

2.3.2 Form

The form of the verb entered at the "New word?" prompt is assumed to be the infinitive form of the verb. As with nouns, the program automatically generates hypotheses about other inflectional forms of the verb (see Appendix) and then prompts the user for verification of these forms. For example, the program will correctly generate SHOWED as the Past form of SHOW. However, it incorrectly assumes that SHOWED is also the Past Participle. In the example below, the user verifies the correct forms and then types the appropriate Past Participle:

```
New word  ? SHOW
Category  ? VERB
Additional forms:
  Present:              THEY SHOW ...      ? <Return>
  3rd Sg Present:       IT SHOWS ...       ? <Return>
  Present Participle:   IT IS SHOWING ...  ? <Return>
  Past:                 IT SHOWED ...      ? <Return>
  Past Participle:      IT HAS SHOWED ...  ? SHOWN
```

2.3.3 Transitivity

A verb is specified for the highest level of transitivity that it can manifest. The three possible values are:

```
Intransitive: the verb cannot occur with an object
Transitive:   the verb can occur with a single object
Bitransitive: the verb can take two objects
```

The user is prompted by first placing the verb with two objects and asking if the result is grammatical. If it is, then the verb is specified as [Bitransitive]. Otherwise, a second prompt is constructed with the verb

and a single object. If this is verified as grammatical, then the verb is specified as [Transitive]. Otherwise, the verb is [Intransitive]. Each of these possible feature values for Transitivity is demonstrated in the examples below:

```
New word  ? BOND
Category  ? VERB
Additional forms:
  Present:              THEY BOND ...       ? <Return>
  3rd Sg Present:       IT BONDS ...        ? <Return>
  Present Participle:   IT IS BONDING ...   ? <Return>
  Past:                 IT BONDED ...       ? <Return>
  Past Participle:      IT HAS BONDED ...   ? <Return>
Transitivity:
  BOND ITEM 1 TO ITEM 3.  (ok) ? <Return>

New word  ? ASSEMBLE
Category  ? VERB
Additional forms:
  Present:              THEY ASSEMBLE ...       ? <Return>
  3rd Sg Present:       IT ASSEMBLES ...        ? <Return>
  Present Participle:   IT IS ASSEMBLING ...    ? <Return>
  Past:                 IT ASSEMBLED ...        ? <Return>
  Past Participle:      IT HAS ASSEMBLED ...    ? <Return>
Transitivity:
  ASSEMBLE ITEM 1 TO ITEM 3.  (ok) ? *
  ASSEMBLE ITEM 1.  (ok) ? <Return>

New word  ? ESCAPE
Category  ? VERB
Additional forms:
  Present:              THEY ESCAPE ...       ? <Return>
  3rd Sg Present:       IT ESCAPES ...        ? <Return>
  Present Participle:   IT IS ESCAPING ...    ? <Return>
  Past:                 IT ESCAPED ...        ? <Return>
  Past Participle:      IT HAS ESCAPED ...    ? <Return>
Transitivity:
  ESCAPE ITEM 1 TO ITEM 3.  (ok) ? *
  ESCAPE ITEM 1.  (ok) ? *
```

2.3.4 Synonyms

Verbs may be specified as having synonymous forms. These forms will be entered in the lexicon as having all of the features of the original entry. For example, FILLET might be entered as a synonym of FILLET BOND:

```
New word  ? FILLET BOND
Category  ? VERB
```

```
Additional forms:
  Present:              THEY FILLET BOND ...    ? <Return>
  3rd Sg Present:       IT FILLET BONDS ...     ? <Return>
  Present Participle:   IT IS FILLET BONDING ...? <Return>
  Past:                 IT FILLET BONDED ...    ? <Return>
  Past Participle:      IT HAS FILLET BONDED ...? <Return>
Transitivity:
  FILLET BOND ITEM 1 TO ITEM 3.  (ok) ? <Return>
Synonym  ? FILLET
```

2.4 Prepositions

The binary feature [Timing] is used to subcategorize prepositions. Most prepositions which are [+Timing] can also function as adverbs (e.g. AFTER, BEFORE, PRIOR TO), but some cannot (e.g. DURING). This feature will be used by the parser to help determine the correct attachment for prepositional phrases (which can either modify a preceding Noun Phrase or the Verb).

```
New word  ? BEFORE
Category  ? PREPOSITION
Does BEFORE refer to a timing sequence  (no) ? YES
```

2.5 Other Categories

None of the other syntactic categories have subcategorization features. These categories are: Adjective, Adverb, and Conjunction. Therefore, the user is only required to enter the new word and its category. The program will not prompt for any subcategorization features:

```
New word  ? BLACK
Category  ? ADJECTIVE

New word  ? EVENLY
Category  ? ADVERB

New word  ? AND
Category  ? CONJUNCTION
```

3. Update

The Update function is used to modify the specification of words which have already been entered in the lexicon. The program prompts with all of the original questions. It provides the current feature values as the default for the user to verify. Note that these defaults may be different from the defaults that were supplied with the original questions. As an example, consider the following case in which the user originally verified

the Past Participle form "DRAWED". After becoming aware of the mistake, the user can return in the Update mode to correct the problem:

```
Options:  Add, Delete, List, Update, Exit.
Option ? UPDATE

Update word ? DRAWED
Category:  VERB ? <Return>
Infinitive: DRAW ? <Return>
Past Participle:  IT HAS DRAWED ... ? DRAWN
Transitivity:
  DRAW ITEM 1 TO ITEM 3.  (*) ? <Return>
  DRAW ITEM 1.  (ok) ? <Return>
```

4. Delete

In order to remove a word from the lexicon, the Delete function must be used. It prompts the user for each category label associated with a particular word, so that one sense of a word can be deleted without deleting the entire entry. In the following example, the entry for BOND as a Noun is deleted, but it remains as a Verb:

```
Options:  Add, Delete, List, Update, Exit.
Option ? DELETE

Delete word ? BOND
Noun (no) ? YES
Verb (no) ? <Return>
```

5. List

The List function can be used to print all or part of the lexicon. The user must specify the beginning and ending of the range of words to be printed, where A to Z is the default. If the user specifies a word other than "A" as the starting point, then the ending default is set to the same word. This allows someone to print a single entry while only typing the word once. The user has the choice of printing just the words or the words and their feature specifications. If the starting and ending point are the same word, then the default for listing feature specifications should be "yes". Otherwise, the default should be "no". The default output location is "stdout" which in most cases is the terminal. The following session represents an attempt to print out the entire lexicon without feature specifications:

```
Options:  Add, Delete, List, Update, Exit.
Option ? LIST
```

```
Start listing at (A) ? <Return>
End listing at (Z) ? <Return>
List feature specifications (no) ? <Return>
Output file (stdout) ? <Return>
```

The following sample session would result in all the words from BOND to CATHODE (inclusive) being printed along with their feature specifications. The listing would go to the file "lex.list".

```
Options:  Add, Delete, List, Update, Exit.
Option ? LIST
Start listing at (A) ? BOND
End listing at (BOND) ? CATHODE
List feature specifications (no) ? YES
Output file (stdout) ? LEX.LIST
```

A sample listing with feature specifications is shown below. This listing corresponds to the sample request to print all entries between BOND and CATHODE.

BOND Verb, [Base: BOND], [Type: Non-Aux], [Form: Infinitive, Present], [Transitivity: Bitransitive].

BONDED Verb, [Base: BOND], [Type: Non-Aux], [Form: Past, Past Participle], [Transitivity: Bitransitive].

BONDING Verb, [Base: BOND], [Type: Non-Aux], [Form: Present Participle], [Transitivity: Bitransitive].

BOTH Determiner, [-Singular], [+Plural], [-Mass].

BRIEFLY Adverb

BROWN Adjective

BURR Noun, [Base: BURR], [+Singular], [-Plural], [+Count], [-Mass], [-Proper], [-Pronoun]

BURRS Noun, [Base: BURR], [-Singular], [+Plural], [+Count], [-Mass], [-Proper], [-Pronoun]

BY Preposition, [-Timing]

CATHODE Noun, [Base: CATHODE], [+Singular], [-Plural], [+Count], [-Mass], [-Proper], [-Pronoun]

6. Sample Session

This section provides a sample interactive session which includes examples from all of the grammatical categories.

```
$interlex std.note.lexicon
INTERLEX 1.0          10:25   04/12/85

Options:  Add, Delete, List, Update, Exit.
Option ? ADD

New word  ? SHIELD
Category  ? NOUN
Singular:   SHIELD   ? <Return>
Plural:     SHIELDS ? <Return>
HOW MANY SHIELDS ARE REQUIRED?  (ok) ? <Return>
HOW MUCH SHIELD IS REQUIRED?  (ok) ? *
Synonym  ? <Return>

New word  ? PAINT
Category  ? NOUN
Singular:   PAINT   ? <Return>
Plural:     PAINTS ? <Return>
HOW MANY PAINTS ARE REQUIRED?  (ok) ? <Return>
HOW MUCH PAINT IS REQUIRED?  (ok) ? <Return>
Synonym  ? <Return>

New word  ? PAINT
PAINT has already been entered as a Noun.
Category  ? VERB
Additional forms:
  Present:              THEY PAINT ...      ? <Return>
  3rd Sg Present:       IT PAINTS ...       ? <Return>
  Present Participle:   IT IS PAINTING ...  ? <Return>
  Past:                 IT PAINTED ...      ? <Return>
  Past Participle:      IT HAS PAINTED ...  ? <Return>
Transitivity:
  PAINT ITEM 1 TO ITEM 3.  (ok) ? *
  PAINT ITEM 1.  (ok) ? <Return>
Synonym  ? <Return>

New word  ? BFVS
Category  ? NOUN
Singular:   BFVS   ? <Return>
Plural:     BFVSES ? *
Is this a Proper Noun (yes) ? <Return>
Synonym  ? <Return>

New word  ? STEAM
Category  ? NOUN
Singular:   STEAM  ? <Return>
Plural:     STEAMS ? *
Is this a Proper Noun (yes) ? NO
Is this a Pronoun (yes) ? NO
HOW MUCH STEAM IS REQUIRED?  (ok) ? <Return>
Synonym  ? <Return>
```

```
New word  ? THE
Category  ? DETERMINER
THE PART IS STATIC SENSITIVE.  (ok) ? <Return>
THE PARTS ARE STATIC SENSITIVE.  (ok) ? <Return>
THE STEAM MUST REACH EVERY SURFACE OF THE PART.  (ok) ? <Return>

New word  ? THIS
Category  ? DETERMINER
THIS PART IS STATIC SENSITIVE.  (ok) ? <Return>
THIS PARTS ARE STATIC SENSITIVE.  (ok) ? *
THIS STEAM MUST REACH EVERY SURFACE OF THE PART.  (ok) ? <Return>

New word  ? THESE
Category  ? DETERMINER
THESE PART IS STATIC SENSITIVE.  (ok) ? *
THESE PARTS ARE STATIC SENSITIVE.  (ok) ? <Return>
THESE STEAM MUST REACH EVERY SURFACE OF THE PART.  (ok) ? *

New word  ? BLACK
Category  ? ADJECTIVE

New word  ? WIDE
Category  ? ADJECTIVE

New word  ? CONTINUOUSLY
Category  ? ADVERB

New word  ? EVENLY
Category  ? ADVERB

New word  ? AND
Category  ? CONJUNCTION

New word  ? OR
Category  ? CONJUNCTION

New word  ? TO
Category  ? PREPOSITION
Does TO refer to a timing sequence (no) ? <Return

New word  ? AFTER
Category  ? PREPOSITION, ADVERB
Does AFTER refer to a timing sequence (no) ? YES

New word  ? MARK
Category  ? VERB
Additional forms:
  Present:              THEY MARK ...      ? <Return>
  3rd Sg Present:       IT MARKS ...       ? <Return>
  Present Participle:   IT IS MARKING ...  ? <Return>
  Past:                 IT MARKED ...      ? <Return>
  Past Participle:      IT HAS MARKED ...  ? <Return>
```

```
Transitivity:
  MARK ITEM 1 TO ITEM 3.  (ok) ? *
  MARK ITEM 1.  (ok) ? <Return>
Synonym  ? <Return>

New word  ? DRAW
Category  ? VERB
Additional forms:
  Present:              THEY DRAW ...       ? <Return>
  3rd Sg Present:       IT DRAWS ...        ? <Return>
  Present Participle:   IT IS DRAWING ...   ? <Return>
  Past:                 IT DRAWED ...       ? DREW
  Past Participle:      IT HAS DRAWED ...   ? DRAWN
Transitivity:
  DRAW ITEM 1 TO ITEM 3.  (ok) ? *
  DRAW ITEM 1.  (ok) ? <Return>
Synonym  ? <Return>

New word  ? <Return>

Options:  Add, Delete, List, Update, Exit.
Option ? DELETE

Delete word ? AXIS
Noun (no) ? YES
AXIS has been deleted.

Delete word ? SOLDER
Noun (no) ? YES
Verb (no) ? <Return>
The Noun entry for SOLDER has been deleted.

Delete word ? <Return>

Options:  Add, Delete, List, Update, Exit.
Option ? UPDATE

Update word ? EVERY
Category:  DETERMINER ? <Return>
EVERY PART IS STATIC SENSITIVE.  (ok) ? <Return>
EVERY PARTS ARE STATIC SENSITIVE.  (ok) ? *
EVERY STEAM MUST REACH EVERY SURFACE OF THE PART.  (*) ? <Return>

Update word ? <Return>

Options:  Add, Delete, List, Update, Exit.
Option ? List

Start listing at (A) ? DRAIN
End listing at (DRAIN) ? <Return>
```

```
List feature specifications (yes) ? <Return>
Output file (stdout) ? <Return>

DRAIN  Verb, [Base: DRAIN], [Type: Non-Aux], [Form: Infinitive, Present],
             [Transitivity: Transitive].

Options:  Add, Delete, List, Update, Exit.
Option ? Exit

EXIT INTERLEX 1.0          11:05   04/12/85
$
```

Appendix: Inflectional Morphology for Nouns and Verbs

A.1 Nouns

Start with the singular form and derive the plural by the following rules:

(1) If the singular form ends in the letter(s) s, x, z, ch, or sh, then add es to form the plural. For example:

BOX ==> BOXES
BYPASS ==> BYPASSES
FINISH ==> FINISHES
PATCH ==> PATCHES

(2) If the singular form ends in a consonant followed by the letter y, then delete the y and add ies to form the plural.

ASSEMBLY ==> ASSEMBLIES
COPY ==> COPIES

(3) For all other singular forms, add s to obtain the plural form.

DIMENSION ==> DIMENSIONS
LINE ==> LINES
PART ==> PARTS
PATH ==> PATHS
TRAY ==> TRAYS

A.2 Verbs

All inflectional forms are built by starting with the Infinitive form and then adding a suffix.

Present Remains the same as the Infinitive form.

3rd Present Use the same rules as for forming plural nouns from singular ones.

(1) ATTACH ==> ATTACHES
PREFIX ==> PREFIXES
PUSH ==> PUSHES

(2) APPLY ==> APPLIES
COPY ==> COPIES
MODIFY ==> MODIFIES

(3) DENOTE ==> DENOTES
SEAL ==> SEALS
SHOW ==> SHOWS

Past and Past participle

(1) If the verb ends in a consonant followed by the letter y, then delete the y and add ied.

APPLY ==> APPLIED
COPY ==> COPIED
MODIFY ==> MODIFIED

(2) If the verb ends in e, then add d.

CURE ==> CURED
LOCATE ==> LOCATED
REMOVE ==> REMOVED

(3) If the verb ends in a Consonant-Vowel-Consonant (CVC) sequence, where:
the first consonant is any alphabetic character other than {a, e, i, o, u};
the vowel is one of the characters {a, e, i, o, u};
the second consonant is one of {b, d, f, g, k, l, m, n, p, r, s, t, v};
then double the second consonant and add ed.

STOP ==> STOPPED
TRIM ==> TRIMMED

(4) For all other Past and Past Participle forms, simply add ed to the Infinitive form.

BOND ==> BONDED
DRILL ==> DRILLED
EXCEED ==> EXCEEDED
SPRAY ==> SPRAYED

Present Participle

(1) If the verb ends in e, then delete the e and add ing.

CURE ==> CURING
REMOVE ==> REMOVING
USE ==> USING

(2) If the verb ends in a CVC sequence as described for Past and Past Participle forms above, then double the second consonant and add ing.

CUT ==> CUTTING
PAD ==> PADDING
SCRUB ==> SCRUBBING

(3) For all other Present Participle forms, simply add ing to the end of the verb.

BAND ==> BANDING
BREAK ==> BREAKING
TEST ==> TESTING

Abbreviations

The following abbreviations are acceptable within the program:

A Add

ADJ Adjective

ADV Adverb

CONJ Conjunction

D Delete

DET Determiner

E Exit ("X" is also acceptable)

L List

N Noun or No (depending on the context)

P Preposition ("PREP" is also acceptable)

PREP Preposition ("P" is also acceptable)

U Update

V Verb

X Exit ("E" is also acceptable)

Y Yes

Filed as: //larry/ty.dir/doc.dir/fspec.lex.interface.doc

Functional Specification

A Lexicon

for Standard Engineering Notes

Jim Tyhurst

0. Introduction

A lexicon is a list of words and corresponding grammatical information. This document specifies the contents and structure of a lexicon to be used for the task of parsing Standard Notes. It describes the forms of words to be represented and the type of information to be stored for each word.

In Section 1, some general criteria are established for choosing between various features and structures depending on the use of the lexicon. Then a structure is proposed in Section 2 which provides the information required by a standard notes parser (which is described in the document "An Augmented Transition Network Grammar for Standard Engineering Notes"). Given the developmental nature of the grammar, it will undoubtedly be necessary to change the format of certain entries in the lexicon. Issues related to this problem are discussed in Section 3. Finally, a sample lexicon is presented in Section 4 which demonstrates the use of the categories and features presented in previous sections.

1. Purpose of the Lexicon

The primary purpose of the lexicon is to provide a list of all the words of a language and to specify the grammatical category (or categories) of each word. A second function is to list the subcategorization features which are relevant to the grammar.

1.1 Categorial information for the parser

A parser deals with category labels, rather than with individual lexical items. Thus, in order to parse the noun phrase, "THE SCHEMATIC DIAGRAM", it is sufficient for the parser to know that THE is a determiner, SCHEMATIC is an adjective, and DIAGRAM is a noun. The parser can tell that this is a noun phrase, because it knows that "Determiner + Adjective + Noun" is one possible form of a noun phrase. Note that at times a parser will refer to individual words. For example, it may check to see if a preposition is "TO" or "FOR" in order to distinguish between different roles that a prepositional phrase might fill. However, in general, the parser's analysis of a sentence is based on the lexical categories assigned to each input word.

Therefore, a lexical analyzer is required to pre-process a sentence before the sentence is input to the parser. The lexical analyzer divides the sentence into words, looks up the words in the lexicon, and passes on the words and their associated grammatical information to the parser. The parser then uses the category information to parse groups of words into phrases.

1.2 Subcategorization features

Two types of features are necessary for parsing standard notes. One type is used to check for certain types of grammatical agreement between phrases in a sentence. A second type is used to specify the different distributional properties of lexical items belonging to the same category. Each of these types of subcategorization features is necessary in our lexicon.

English has a system of number agreement by which the number of a noun agrees with the number of its determiner. For example, the singular form THIS DOT versus the plural THESE DOTS. Also, the number of the verb agrees with the number of the subject. For example, THE DOT INDICATES... versus THE DOTS INDICATE_...

The second type of subcategorization feature is required to specify the environments in which particular lexical items may occur. For example, a bitransitive verb (solder, bond, fillet) can take two objects (e.g. SOLDER ITEM 1 TO ITEM 2). However, a transitive verb (seal, conformal coat, assemble) can only take one object (e.g. SEAL ITEM 3, but not SEAL ITEM 3 TO ITEM 5). These restrictions on the number of objects which can cooccur with a particular verb can be represented in the lexicon by a Transitivity feature.

2. Structure of the Lexicon

There are two basic issues involved in specifying the structure of the lexicon. First, we need to decide what words will appear as entries in the lexicon. Some lexicons only list base forms and then other derived forms are generated by rules. We will take the other approach of listing all possible forms in the lexicon. The second issue is to decide what grammatical information should be specified for each word. These two issues are discussed in detail in the following two sub-sections.

2.1 Types of entries

Every possible word of the language will have a separate listing in the lexicon. Different inflectional forms will be tied together in the sense that each will refer to the base form. For example, each of the following forms will occur in the lexicon:

SPECIFICATION: noun, singular, base: specification
SPECIFICATIONS: noun, plural, base: specification
SPECIFY: verb, {infinitive, present}, base: specify
SPECIFIES: verb, 3rd-singular, base: specify
SPECIFIED: verb, {past, past participle}, base: specify
SPECIFYING: verb, present participle, base: specify

In this example, all of the different verb forms are listed separately, although each lists the base form of the verb. The form "specified" is ambiguous as being both the past tense form and the past participle form.

2.2 Categories and features

The features to be used in the lexicon for standard notes are presented in this section. Not all of the features are required at this point in time, since a number of simplifying restrictions have been placed on the grammar for standard notes. However, a more complete set of features will be required when the grammar is generalized in the future. Therefore, we want the lexicon to be as general as possible right from the very beginning of the developmental cycle. There are seven major categories. The sub-categorization features for each category are listed in the following sub-sections. The choice of categories and features is based on Winograd's (1983) description of Augmented Transition Networks.

Binary features are marked as either + or - for a particular word in order to indicate presence or absence of the feature. Multi-valued features take their values from a restricted set of values which are listed in curly brackets, {}.

2.2.1 NOUN

Binary features:
- Singular
- Plural
- Count
- Mass
- Proper
- Pronoun
  - Pronouns are further subcategorized for Person and Case:
  - (we are ignoring possessive pronouns)
  - Person: {1st, 2nd, 3rd}
  - Case: {Subject, Object}

Multi-valued features:
- Synonyms

Some nouns will be specified as either Singular or Plural, while others can be both Singular and Plural. If a noun is specified for both features, that means it can be either one (e.g. THIS REF DES or THESE REF DES):

PART: Noun, [+Singular], [-Plural]
PARTS: Noun, [-Singular], [+Plural]
REF DES: Noun, [+Singular], [+Plural]

The features Count and Mass are used to characterize certain cooccurrence restrictions in the noun phrase. Count nouns can have a plural form and occur with numerals and certain quantifiers. For example, PART is [+Count], so forms like "3 PARTS" or "MANY PARTS" are grammatical. However, HEAT is [-Count]. Therefore, "3 HEATS" is ungrammatical. Nouns which are specified as [+Mass] can occur with the determiner MUCH, but nouns which are [-Mass] cannot (e.g. "MUCH HEAT" is grammatical, but "MUCH PART" is not). Some English nouns can be specified as both [+Count] and [+Mass] (Baker 1978:252-253), although I have not found many examples from the vocabulary to be used for the Printed Wiring Assembly problem domain. MATERIAL is at least one example, since the two phrases "MUCH MATERIAL" which has a [+Mass] reading and "MANY MATERIALS" which has a [+Count] reading are both grammatical.

Nouns which are [+Proper] (e.g. specification names like QQ-N-290) or [+Pronoun] (e.g. IT or THEY) cannot cooccur with determiners or adjectives. For example, "EVERY QQ-N-290" and "PARTIAL IT" are ungrammatical. Only nouns which are both [-Proper] and [-Pronoun] can have a determiner or an adjective, e.g. "THE PARTIAL REFERENCE DESIGNATION" (where THE is a determiner and PARTIAL is an adjective).

Abbreviations can be indicated by using the Synonym feature. For example:

PART NUMBER: Synonym: P/N
REFERENCE DESIGNATION: Synonym: REF DES
SPECIFICATION: Synonym: SPEC

2.2.2 DETERMINER

Binary features:
- Singular
- Plural
- Mass

Many determiners can occur with either a singular or plural noun (e.g. THE, EVERY, SOME). These determiners will be marked [+Singular], [+Plural]. However, some determiners agree in number with the noun that they cooccur with (e.g. THIS/THESE, THAT/THOSE). The singular member of these pairs, e.g. THIS, will be marked [+Singular], [-Plural]. Whereas, the plural member, e.g. THOSE, will be marked [-Singular], [+Plural].

Certain determiners can occur with nouns marked [+Mass], while others cannot. This cooccurrence restriction cannot be derived from the features [Singular] and [Plural] as demonstrated by the different contexts which A and THIS can occur in. For example, "a water" is ungrammatical, but it is okay to say "this water". The following tables list the features for a few common determiners. Note that it is logically impossible for a determiner to be specified negatively for all three features, since that would indicate that the determiner cannot occur with any noun. I have not been able to find any examples of determiners which occur with the singular and plural form of count nouns, but which cannot occur with a mass noun.

Singular	Plural	Mass	Examples
-	-	+	MUCH
-	+	-	THESE, THOSE, FEW, MANY
-	+	+	ALL
+	-	-	A, EVERY
+	-	+	THIS, THAT
+	+	-	
+	+	+	THE, SOME, NO

2.2.3 ADJECTIVE (No subcategorization features)

Examples of adjectives are: PARTIAL, SCHEMATIC, D-SHAPED, FLUSH, NUCLEAR.

2.2.4 PREPOSITION

Binary features:
- Timing

Examples of prepositions that are [-Timing]: IN, ON, BY, PER, TO, FOR, FROM, OF, WITH. Examples of prepositions which are [+Timing]: AFTER, BEFORE, DURING, PRIOR TO.

2.2.5 VERB

Multi-valued features:
Type: {Modal, Be, Do, Have, Non-Aux}
Form: {Infinitive, Present, 3rd Present, Past, Present Participle, Past Participle}
Transitivity: {Intransitive, Transitive, Bitransitive}
Synonyms

Verb features are required to check for grammaticality within the verb phrase. The feature Type is used to distinguish different types of Auxiliary verbs from each other and from Non-Auxiliary verbs. The modal verbs are: {will, would, shall, should, can, could, may, might, must, ought}. The auxiliary verbs "be", "do", and "have" must be individually identified, because of their special status in the verb phrase. All other verbs are simply listed as Non-Auxiliary.

Verbs take different inflectional forms depending on tense. Most verbs have the same form in the infinitive and the present except for the 3rd person singular present form. Some exceptional verbs, like "be", require more distinctions than the features presented here, however, we will ignore this additional complexity in this implementation. The different verb forms are chosen as follows:

Infinitive: The form of the verb when it occurs with "to" or following one of the modals. For example, "It needs to be..." or "It shall meet..."

3rd Present: The form of the verb when the subject of the sentence is a 3rd person singular noun phrase. For most verbs, this form is derived by simply adding "-s" to the end of the infinitive form. For example, "It meets..." or "Each part requires..."

Past: The Past tense form of the verb frequently has an "-ed" ending, but there are many exceptions. Examples of verbs in past tense form are: "Each part met..." or "Assemblies required..."

Present Participle: This form has an "-ing" ending. For example, "While each part is drying..." or "Any part requiring...". The present participle form of a verb can frequently function as a noun, e.g. twisting and chipping (but not using).

Past Participle: This is the form of the verb that occurs in passive constructions following the verb "be" or in perfect constructions following "have". It ends in "-ed" or "-n". For example, "Heat is required...", "Parts are shown...", or "Components have been..."

The third verb feature is Transitivity. Verbs which cannot be followed by a noun phrase are Intransitive. I think there will be very few of these in the Printed Wiring Assembly vocabulary (but maybe exist, sit, remain, etc.?). Transitive verbs can be followed by a a single noun phrase, e.g. "Insert components...", "Torque item 5...", or "Stencil serial number...". Bitransitive verbs denote processes which involve two objects, e.g. "Solder item 2 to item 3" or "Bond item 5 to ref des C3...".

2.2.6 ADVERB (No subcategorization features)

Examples of adverbs are: APPROXIMATELY, COMPLETELY, EVENLY.

2.2.7 CONJUNCTION (No subcategorization features)

Examples of conjunctions are: AND, OR.

2.3 Words belonging to multiple categories

Some words will belong to more than one category. Many words can function both as a noun and a verb:

```
SOLDER:    Noun
              [+Singular], [-Plural]
              [-Count], [+Mass]
              [-Proper], [-Pronoun].
           Verb
              Base: SOLDER
              Type: {Non-Aux}
              Form: {Infinitive, Present}
              Transitivity: {Bitransitive}.

PREFIXES:  Noun
              [-Singular], [+Plural]
              [+Count], [-Mass]
              [-Proper], [-Pronoun].
           Verb
              Base: PREFIX
              Type: {Non-Aux}
              Form: {3rd Present}
              Transitivity: {Bitransitive}.
```

In addition to some words belonging to more than one category, some words will have to be specified for more than one of the multi-valued feature values. For example, the Past and Past Participle forms of verbs are frequently the same:

```
INDICATED:    Verb: Base: {INDICATE}
                    Type: {Non-Aux}
                    Form: {Past, Past Participle}
                    Transitivity: {Transitive}
```

3. Modifications to the Lexicon

The implementation of the lexicon and representation of features should be very flexible. We do not want to be restricted to using only the features which are presented in this document. Routines which access the lexicon should make it easy to add or delete features which are used to subcategorize the different categories.

4. Sample Lexicon

The following sample includes words from all of the major categories and most of the subcategorization features are exemplified. Each of the inflected forms of a particular root must be included in the lexicon. The words are listed in alphabetical order.

A — Determiner: [+Singular], [-Plural], [-Mass]

ABOVE — Preposition: [-Timing]

AFTER — Preposition: [+Timing]

AND — Conjunction

APPROXIMATELY — Adverb

ARE — Verb: Base: {BE}
Type: {Be}
Form: {Present}
Transitivity: {Transitive}

ASSEMBLE — Verb: Base: {ASSEMBLE}
Type: {Non-Aux}
Form: {Infinitive}
Transitivity: {Transitive}

ASSEMBLED — Verb: Base: {ASSEMBLE}
Type: {Non-Aux}
Form: {Past, Past Participle}
Transitivity: {Transitive}

ASSEMBLES — Verb: Base: {ASSEMBLE}
Type: {Non-Aux}
Form: {3rd Present}
Transitivity: {Transitive}

ASSEMBLIES — Noun: Base: {ASSEMBLY}
[-Singular], [+Plural]
[+Count], [-Mass]
[-Proper], [-Pronoun]

ASSEMBLING Verb: Base: {ASSEMBLE}
Type: {Non-Aux}
Form: {Present Participle}
Transitivity: {Transitive}

ASSEMBLY Noun: Base: {ASSEMBLY}
[+Singular], [-Plural]
[+Count], [-Mass]
[-Proper], [-Pronoun]

BASIC Adjective

BE Verb: Base: {BE}
Type: {Be}
Form: {Infinitive}
Transitivity: {Transitive}

BEING Verb: Base: {BE}
Type: {Be}
Form: {Present Participle}
Transitivity: {Transitive}

BLACK Adjective

BY Preposition: [-Timing]

CONTINUOUS Adjective

CONTINUOUSLY Adverb

EXCEPT Conjunction

IS Verb: Base: {BE}
Type: {Be}
Form: {3rd Present}
Transitivity: {Transitive}

MAY Verb: Base: {MAY}
Type: {Modal}
Form: {Infinitive}
Transitivity: {Intransitive}

MUST Verb: Base: {MUST}
Type: {Modal}
Form: {Infinitive}
Transitivity: {Intransitive}

NON-FUNCTIONAL Adjective

NUMBER
Noun: Base: {NUMBER}
[+Singular], [-Plural]
[+Count], [-Mass]
[-Proper], [-Pronoun]

Verb: Base: {NUMBER}
Type: {Non-Aux}
Form: {Infinitive}
Transitivity: {Transitive}

NUMBERED
Verb: Base: {NUMBER}
Type: {Non-Aux}
Form: {Past, Past Participle}
Transitivity: {Transitive}

NUMBERING
Noun: Base: {NUMBERING}
[+Singular], [-Plural]
[+Count], [-Mass]
[-Proper], [-Pronoun]

Verb: Base: {NUMBER}
Type: {Non-Aux}
Form: {Present Participle}
Transitivity: {Transitive}

NUMBERINGS
Noun: Base: {NUMBERING}
[-Singular], [+Plural]
[+Count], [-Mass]
[-Proper], [-Pronoun]

NUMBERS
Noun: Base: {NUMBER}
[-Singular], [+Plural]
[+Count], [-Mass]
[-Proper], [-Pronoun]

Verb: Base: {NUMBER}
Type: {Non-Aux}
Form: {3rd Present}
Transitivity: {Transitive}

ONLY
Adjective, Adverb

PART
Noun: Base: {PART}
[+Singular], [-Plural]
[+Count], [-Mass]
[-Proper], [-Pronoun]

PARTS
Noun: Base: {PART}
[-Singular], [+Plural]
[+Count], [-Mass]
[-Proper], [-Pronoun]

PART NUMBER Noun: Base: {PART NUMBER}
[+Singular], [-Plural]
[+Count], [-Mass]
[-Proper], [-Pronoun]
Synonyms: {P/N}

PART NUMBERS Noun: Base: {PART NUMBER}
[-Singular], [+Plural]
[+Count], [-Mass]
[-Proper], [-Pronoun]
Synonyms: {P/N's}

PLAN Noun: Base: {PLAN}
[+Singular], [-Plural]
[+Count], [-Mass]
[-Proper], [-Pronoun]

Verb: Base: {PLAN}
Type: {Non-Aux}
Form: {Infinitive}
Transitivity: {Transitive}

PLANNED Verb: Base: {PLAN}
Type: {Non-Aux}
Form: {Past, Past Participle}
Transitivity: {Transitive}

PLANNING Noun: Base: {PLANNING}
[+Singular], [-Plural]
[-Count], [+Mass]
[-Proper], [-Pronoun]

Verb: Base: {PLAN}
Type: {Non-Aux}
Form: {Present Participle}
Transitivity: {Transitive}

PLANS Noun: Base: {PLAN}
[-Singular], [+Plural]
[+Count], [-Mass]
[-Proper], [-Pronoun]

Verb: Base: {PLAN}
Type: {Non-Aux}
Form: {3rd Present}
Transitivity: {Transitive}

PRIOR TO Preposition: [+Timing]

SHALL Verb: Base: {SHALL}
Type: {Modal}
Form: {Infinitive}
Transitivity: {Intransitive}

SOLDER Noun: Base: {SOLDER}
[+Singular], [-Plural]
[-Count], [+Mass]
[-Proper], [-Pronoun]

Verb: Base: {SOLDER}
Type: {Non-Aux}
Form: {Infinitive}
Transitivity: {Bitransitive}

SOLDERED Verb: Base: {SOLDER}
Type: {Non-Aux}
Form: {Past, Past Participle}
Transitivity: {Bitransitive}

SOLDERING Verb: Base: {SOLDER}
Type: {Non-Aux}
Form: {Present Participle}
Transitivity: {Bitransitive}

SOLDERS Verb: Base: {SOLDER}
Type: {Non-Aux}
Form: {3rd Present}
Transitivity: {Bitransitive}

SOME Determiner: [+Singular], [+Plural], [+Mass]

THE Determiner: [+Singular], [+Plural], [+Mass]

THESE Determiner: [-Singular], [+Plural], [-Mass]

THIS Determiner: [+Singular], [-Plural], [+Mass]

THROUGH Preposition: [-Timing]

WIDE Adjective

WIDTH Noun: Base: {WIDTH}
[+Singular], [-Plural]
[+Count], [-Mass]
[-Proper], [-Pronoun]

WIDTHS Noun: Base: {WIDTH}
[-Singular], [+Plural]
[+Count], [-Mass]
[-Proper], [-Pronoun]

References

Baker, C. L. 1978. Introduction to Generative-Transformational Syntax. Englewood Cliffs, N.J.: Prentice-Hall, Inc.

EDSG Drawing Notes: Producibility Based Automated Design and Manufacturing System. October 26, 1984.

Tyhurst, Jim. "An Augmented Transition Network Grammar for Standard Engineering Notes".

Winograd, Terry. 1983. Language as a Cognitive Process, Volume I: Syntax. Reading, Mass: Addison-Wesley Publishing Company.

Filed as: //larry/ty.dir/doc.dir/fspec.lexicon.doc