NATURAL LANGUAGE ANALYSIS VIA AUGMENTED TRANSITION NETWORKS (ATN)

by

JOSE MA. U. LAZARO, JR.

B.S. in Physics, Ateneo de Manila University, Philippines, 1978

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

in

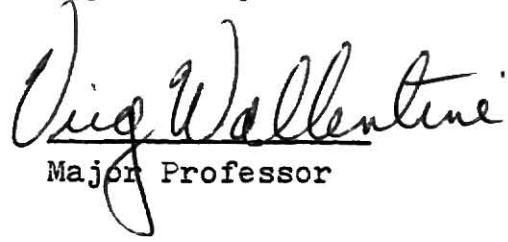
COMPUTER SCIENCE

Department of Computer Science

Kansas State University
Manhattan, Kansas

1982

Approved by:



Major Professor

TABLE OF CONTENTS

LIST OF FIGURES

PAGE

ACKNOWLEDGEMENTS

I would like to thank my major professor, Dr. Roger Hartley, for his guidance, attention, and encouragement, to thank Dr. David Schmidt for his lectures in Programming Language Semantics and his advice, to thank Dr. and Mrs. Carroll V. Hess for their personal encouragement and hospitality, and to especially thank my parents, Joe and Rosie, and my brothers and sisters, for their patience, understanding, support, and love.

CHAPTER 1

INTRODUCTION

These days, there is a growing armamentarium of programs that exhibit what most people consider intelligent behavior. Nearly all of these intelligent or seemingly intelligent programs are written in LISP. Many have the potential of great practical importance. Some examples are: Expert problem solvers, Common-sense reasoning, Learning, Natural Language interfaces, Education and intelligent support systems, Speech and vision, Word Processing, Symbolic Mathematics, and Systems Programming.

This report is a summary of a project that focused on gaining experience in the implementation in LISP of an Augmented Transition Network (ATN) Interpreter to parse certain aspects of English syntax and an ATN Compiler to translate ATN descriptions into a form that LISP understands directly. In addition, an ATN Syntax Analyzer is attached to the ATN Interpreter to investigate syntax errors in the ATN description. The ATN Interpreter and Compiler are based on Chapters 19 and 20 of the book entitled "LISP", authored by Patrick Henry Winston and Berthold Klaus Paul Horn. Chapter 19 is entitled "Interpreting Augmented Transition Networks" while Chapter 20 is entitled "Compiling Augmented Transition Networks".

1.1 Purpose of Report

This report is to serve as documentation of a project that involved the implementation in LISP of an Interpreter and a Compiler for Augmented Transition Networks and the design and implementation of an ATN Syntax Analyzer. One of the important aspects of this project is to help an ATN description recover from singular or multiple syntax errors.

1.2 Content of Report

Presented in this report is a discussion of the ATN Interpreter and Compiler. Also specified are the functional capabilities and design considerations of the ATN Syntax Analyzer together with the Error Recovery System. Formally, the ATN syntax is outlined. There is an appendix containing descriptions of the local data and the routines of the ATN Interpreter, Syntax Analyzer, and Compiler. A listing of the source code of each is provided.

1.3 Environment

The Augmented Transition Network (ATN) Interpreter, Syntax Analyzer, and Compiler are written in LISP and designed to run under IBM 370's Conversational Monitor System (CMS) interactive system. The input and output devices are the CRT and the DEC printer.

1.4 Performance Requirements

Although there are no actual execution time requirements, it is understood that to be of any use in the LISP development and maintenance environment, the ATN Interpreter, Syntax Analyzer, and Compiler should perform in a reasonable and timely fashion, i.e. not take days to analyze an ATN syntax, interpret the syntax of an English clause or even compile an ATN description.

CHAPTER 2

THE AUGMENTED TRANSITION NETWORK (ATN) INTERPRETER

2.1 Definition of an Augmented Transition Network

An ATN formalism is intended to facilitate sentence analysis by capturing word-order regularities. Figure 2.1 exhibits a simple ATN that captures some of the regularities involved in English noun groups.

Any given ATN consists of nodes linked by arcs labeled with names for word classes like noun, verb, adjective, preposition, determiner, and the like. The analysis of a word sequence is accomplished in the course of driving a path through the network, using the word classes as instructions for what arcs to take. Legitimate sequences lead to so-called terminal nodes.

Several smaller ATNs may be linked together into one larger system through a convention by which some arcs require successful traversal of subordinate ATNs rather than just the consumption of single words. For example, a noun group ATN may have a preposition-group arc and a preposition group may have a noun-group arc, thus giving a recursive flavor to the formalism.

There may be more action taking place on the arcs than just the recognition of word classes and movement down a

sentence. The word "augmented" in Augmented Transition Networks means that an ATN description is permitted to make notes as arcs are traversed and to refer to them later. One might, for example, make a note that a noun group is surely singular in moving over the determiner arc with the word "a". A later test could check that conclusion when the noun is encountered.

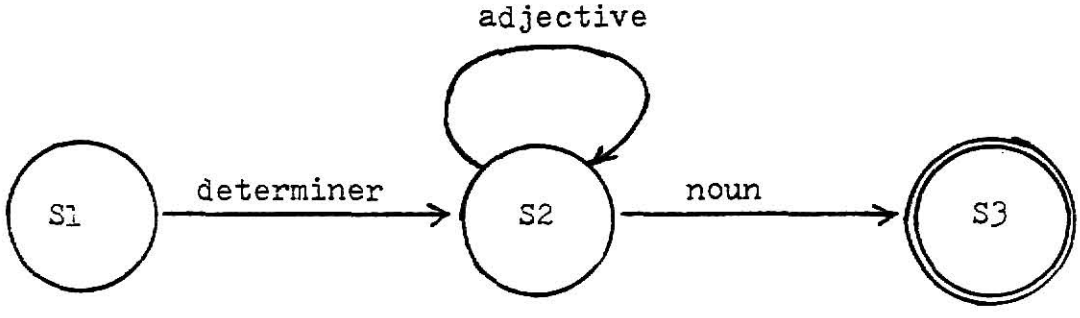


Figure 2.1: A simple Augmented Transition Network representing the structure of simple noun groups.

2.2 Functional Specification for the ATN Interpreter

2.2.1 Scope

This section defines the functional capabilities of the Augmented Transition Network Interpreter.

2.2.2 General Description

The ATN Interpreter is capable of accepting ATN descriptions and certain English sentences as input. The results of the ATN Interpreter's parsing process are represented by the parents, the children, the registers and their values. Figure 2.2 illustrates the ATN Interpreter's general program flow. Figure 2.3 illustrates the Parent-Child-Register-Value table while Figure 2.4 illustrates a sample parse tree based on the table.

The ATN Interpreter has an ATN Syntax Analyzer which is capable of accepting an ATN description as input, identifying, diagnosing and possibly correcting any syntax errors, eventually saving the entire ATN description. The output consists of the original ATN description, the repaired ATN description, if any, and the ATN syntax error codes with the corresponding error messages, if any. Figure 3.1 illustrates the ATN Syntax Analyzer's general program flow.

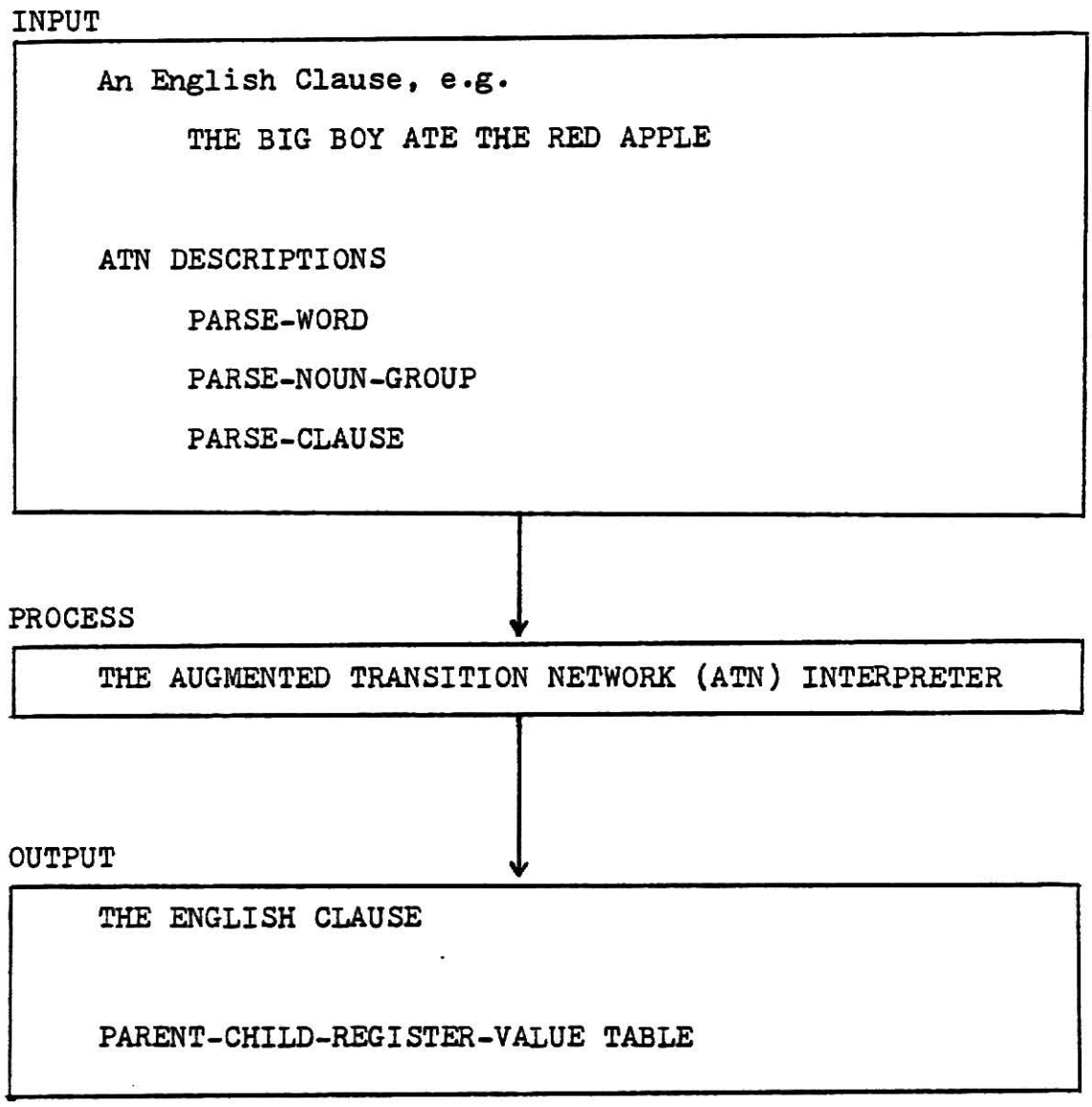


Figure 2.2: General Program Flow for the Augmented Transition Network (ATN) Interpreter.

PARENT	CHILD	REGISTER	VALUE
PNG-2	(THE)	NUMBER	SINGULAR
		DETERMINER	DEFINITE
PNG-2	(THE BIG)	ADJECTIVES	(BIG)
PNG-2	(THE BIG BOY)	NUMBER	SINGULAR
		NOUN	BOY
PC-1	(PNG-2)	SUBJECT	PNG-2
PC-1	(PNG-2 ATE)	VERB	EAT
PNG-8	(THE)	NUMBER	SINGULAR
		DETERMINER	DEFINITE
PNG-8	(THE RED)	ADJECTIVES	(RED)
PNG-8	(THE RED APPLE)	NUMBER	SINGULAR
		NOUN	APPLE
PC-1	(PNG-2 ATE PNG-8)	OBJECT	PNG-8
ROOT	(PC-1)		

NOTE: PNG means PARSE-NOUN-GROUP

PC means PARSE-CLAUSE

Figure 2.3: The Parent-Child-Register-Value Table based on the English clause, THE BIG BOY ATE THE RED APPLE.

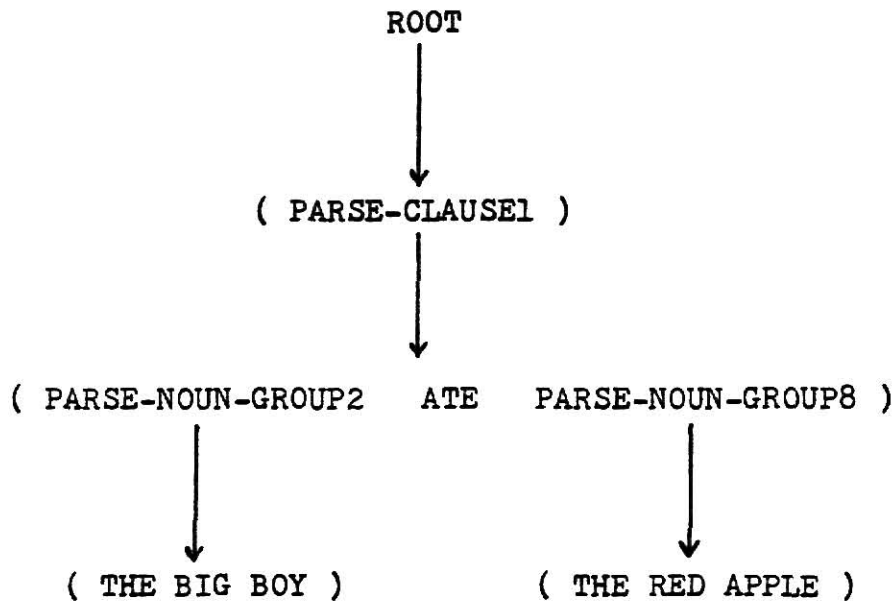


Figure 2.4: A Parse Tree based on the English clause, THE BIG BOY ATE THE RED APPLE.

2.2.3 Functional Description of the ATN Interpreter

The ATN Interpreter is designed with an ATN Syntax Analyzer to syntactically investigate an ATN description. Ultimately, if all of the available ATN descriptions are syntactically correct or made to be syntactically correct, then the ATN Interpreter attempts to parse the given English sentence. Parsing an English sentence is based upon three ATN descriptions, namely, PARSE-CLAUSE, PARSE-NOUN-GROUP, and PARSE-WORD which capture the ideas graphically presented in Figure 2.5. A listing is produced which consists of the English sentence to be parsed and the results of the ATN Interpreter's parsing process. The results are represented by the parents, the children, the registers and their values. Seven registers are available, namely, subject, verb, number, determiner, adjective, noun, and object. Figure 2.6 presents the ATN descriptions for PARSE-WORD and PARSE-NOUN-GROUP while Figure 2.7 presents the ATN description for PARSE-CLAUSE.

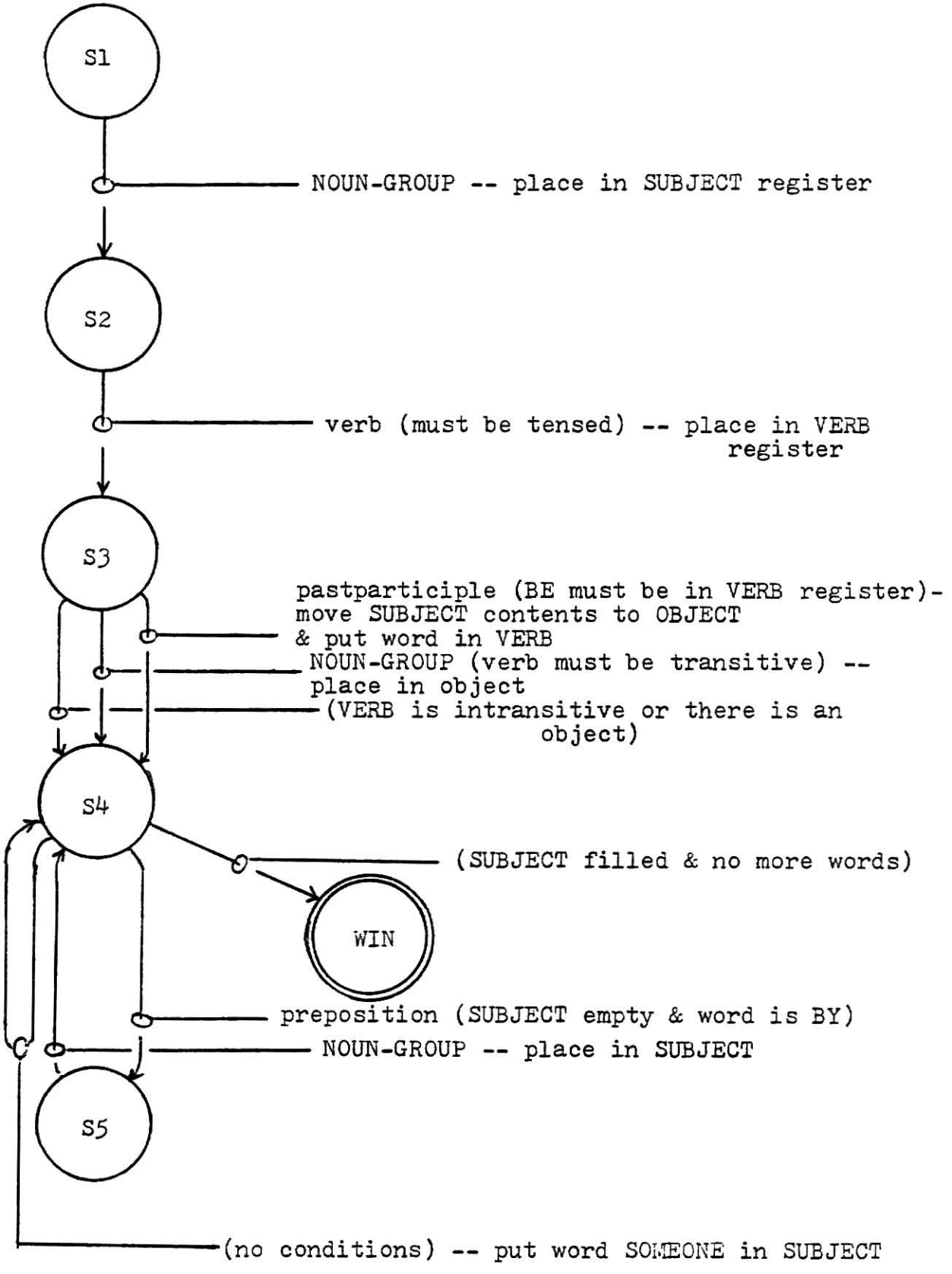


Figure 2.5: An Augmented Transition Network representing the structure of simple sentences. The arcs are tried in clockwise order, starting from the right.

```
(PARSE-WORD
   (S1 (IF T »» WIN
           AFTER
           (SETQ THIS-NODE CURRENT-WORD)
           (SETQ REMAINING-WORDS (CDR REMAINING-WORDS))
           (COND (REMAINING-WORDS
                  (SETQ CURRENT-WORD
                        (CAR REMAINING-WORDS)))
                 (T (SETQ CURRENT-WORD NIL))))))
```

```
(PARSE-NOUN-GROUP
   (S1 (IF (PARSE-WORD THIS-NODE 'DETERMINER)
           »» S2
           AFTER
           (SETR 'NUMBER (SELECT '(SINGULAR PLURAL)
                                 (GETF LAST-PARSED)))
           (SETR 'DETERMINER (SELECT '(DEFINITE INDEFINITE)
                                     (GETF LAST-PARSED)))))
   (S2 (IF (PARSE-WORD THIS-NODE 'ADJECTIVE)
           »» S2
           AFTER
           (ADDR 'ADJECTIVES LAST-PARSED))
       (IF (PARSE-WORD THIS-NODE 'NOUN)
           »» WIN
           AFTER
           (SETR 'NUMBER (SELECT '(SINGULAR PLURAL)
                                 (GETF LAST-PARSED)))
           (SETR 'NOUN LAST-PARSED))))
```

Figure 2.6: The ATN Descriptions for PARSE-WORD and PARSE-NOUN-GROUP.

```
(PARSE-CLAUSE
   (S1 (IF (PARSE-NOUN-GROUP THIS-NODE NIL)
           >>> S2
           AFTER
           (SETR 'SUBJECT LAST-PARSED)))
   (S2 (IF (PARSE-WORD THIS-NODE '(VERB TENSED))
           >>> S3
           AFTER (SETR 'VERB (GET LAST-PARSED 'ROOT))))
   (S3 (IF (AND (EQUAL (GETR 'VERB) 'BE)
                (PARSE-WORD THIS-NODE 'PASTPARTICIPLE))
           >>> S4
           AFTER
           (SETR 'OBJECT (GETR 'SUBJECT))
           (SETR 'SUBJECT NIL)
           (SETR 'VERB LAST-PARSED))
       (IF (AND (TESTF (GETR 'VERB) 'TRANSITIVE)
                (PARSE-NOUN-GROUP THIS-NODE NIL))
           >>> S4
           AFTER (SETR 'OBJECT LAST-PARSED))
       (IF (OR (TESTF (GETR 'VERB) 'INTRANSITIVE)
               (GETR 'OBJECT))
           >>> S4))
   (S4 (IF (AND (GETR 'SUBJECT)
                (NULL REMAINING-WORDS))
           >>> WIN)
       (IF (AND (NOT (GETR 'SUBJECT))
                (EQUAL CURRENT-WORD 'BY)
                (PARSE-WORD THIS-NODE NIL))
           >>> S5)
       (IF (NOT (GETR 'SUBJECT))
           >>> S4
           AFTER
           (SETR 'SUBJECT 'SOMEONE)))
   (S5 (IF (PARSE-NOUN-GROUP THIS-NODE NIL)
           >>> S4
           AFTER
           (SETR 'SUBJECT LAST-PARSED))))
```

Figure 2.7: The ATN description for PARSE-CLAUSE.

Description of Input

The input processed by the ATN Interpreter are ATN descriptions and an English sentence. Based on the ATN descriptions which the ATN Interpreter examines, only certain English sentences are capable of being successfully and completely parsed.

Three ATN descriptions are available. These are the following:

1. Parse-Word
2. Parse-Noun-Group
3. Parse-Clause

Description of Output

Listing of Output

The listing from the ATN Interpreter consists of the following:

1. The English clause to be parsed.
2. The results of the parsing process which are represented by the parents, the children, the registers and their values.

Messages

The following messages will be output to the output device if a detected situation warrants such a message to be displayed.

The English Clause to be parsed

The message

THE-ENGLISH-CLAUSE-TO-BE-PARSED-IS

<An English clause>

indicates the English clause to be parsed by the ATN Interpreter.

The ATN Interpreter Results

The message

THE-RESULTS-OF-THE-INTERPRETING-PROCESS-ARE

where the parents, the children, the registers, and the register values are indicated by

THE-PARENT-IS <parent>

THE-CHILD-IS <child>

THE-REGISTER-IS <register name>

THE-VALUE-IS <register value>

indicates the results of the ATN Interpreter.

2.3 Design Specification for the ATN Interpreter

2.3.1 Scope

This section highlights the significant functions that make up the ATN Interpreter.

2.3.2 Pertinent Functions

INTERPRET is a user-defined iterative routine that accepts a network, fetches a state-description, and tests a state-transition rule. The tests specify what courses to take in the process of driving a path through the network and what must be done when an ATN arc has been taken.

RECORD is a FEXPR routine that creates instances of function definitions such as PARSE-WORD, PARSE-NOUN-GROUP, and PARSE-CLAUSE.

IDENTIFY is the main routine called by the user to execute the ATN Interpreter and the ATN Syntax Analyzer. It contains a dictionary where properties of some English words are defined. IDENTIFY states the English clause to be parsed by the ATN Interpreter. Figure 2.8 presents a sample English dictionary.

GETF, TESTF, ATTACH, SELECT, GENNAME, SETR, GETR, and ADDR are significant auxiliary functions in analyzing certain aspects of English syntax.

```
(DEFPROP A (DETERMINER INDEFINITE SINGULAR) FEATURES)
(DEFPROP AN (DETERMINER INDEFINITE SINGULAR) FEATURES)
(DEFPROP APPLE (NOUN SINGULAR) FEATURES)
(DEFPROP ATE (VERB TENSED) FEATURES)
(DEFPROP ATE EAT ROOT)
(DEFPROP BIG (ADJECTIVE) FEATURES)
(DEFPROP BOY (NOUN SINGULAR) FEATURES)
(DEFPROP EAT (TRANSITIVE) FEATURES)
(DEFPROP EATEN (PASTPARTICIPLE) FEATURES)
(DEFPROP EATS (VERB TENSED) FEATURES)
(DEFPROP EATS EAT ROOT)
(DEFPROP RED (ADJECTIVE) FEATURES)
(DEFPROP SAT (VERB TENSED) FEATURES)
(DEFPROP SAT SIT ROOT)
(DEFPROP SIT (INTRANSITIVE) FEATURES)
(DEFPROP THE (DETERMINER DEFINITE SINGULAR) FEATURES)
(DEFPROP WAS (VERB TENSED) FEATURES)
(DEFPROP WAS BE ROOT)
```

Figure 2.8: A Sample Dictionary used by the ATN Interpreter.

CHAPTER 3

THE AUGMENTED TRANSITION NETWORK (ATN) SYNTAX ANALYZER

3.1 Functional Specification for the ATN Syntax Analyzer

3.1.1 Scope

This section defines the functional capabilities of the Augmented Transition Network Syntax Analyzer.

3.1.2 General Description

The Augmented Transition Network Syntax Analyzer is capable of accepting an ATN description as input, and identifying and diagnosing any syntax errors. It includes an Error Recovery feature so that subsequent syntax errors may be detected and possibly corrected, eventually saving the entire ATN description. The output consists of the ATN description, the repaired ATN description, if any, the ATN syntax error codes with the corresponding error messages, if any, and the results of the ATN Interpreter, considering that the original ATN description is syntactically correct or is made to be syntactically correct. Figure 3.1 illustrates the program flow of the ATN Syntax Analyzer. Figure 3.2 presents a sample ATN description with syntax errors and its repaired version.

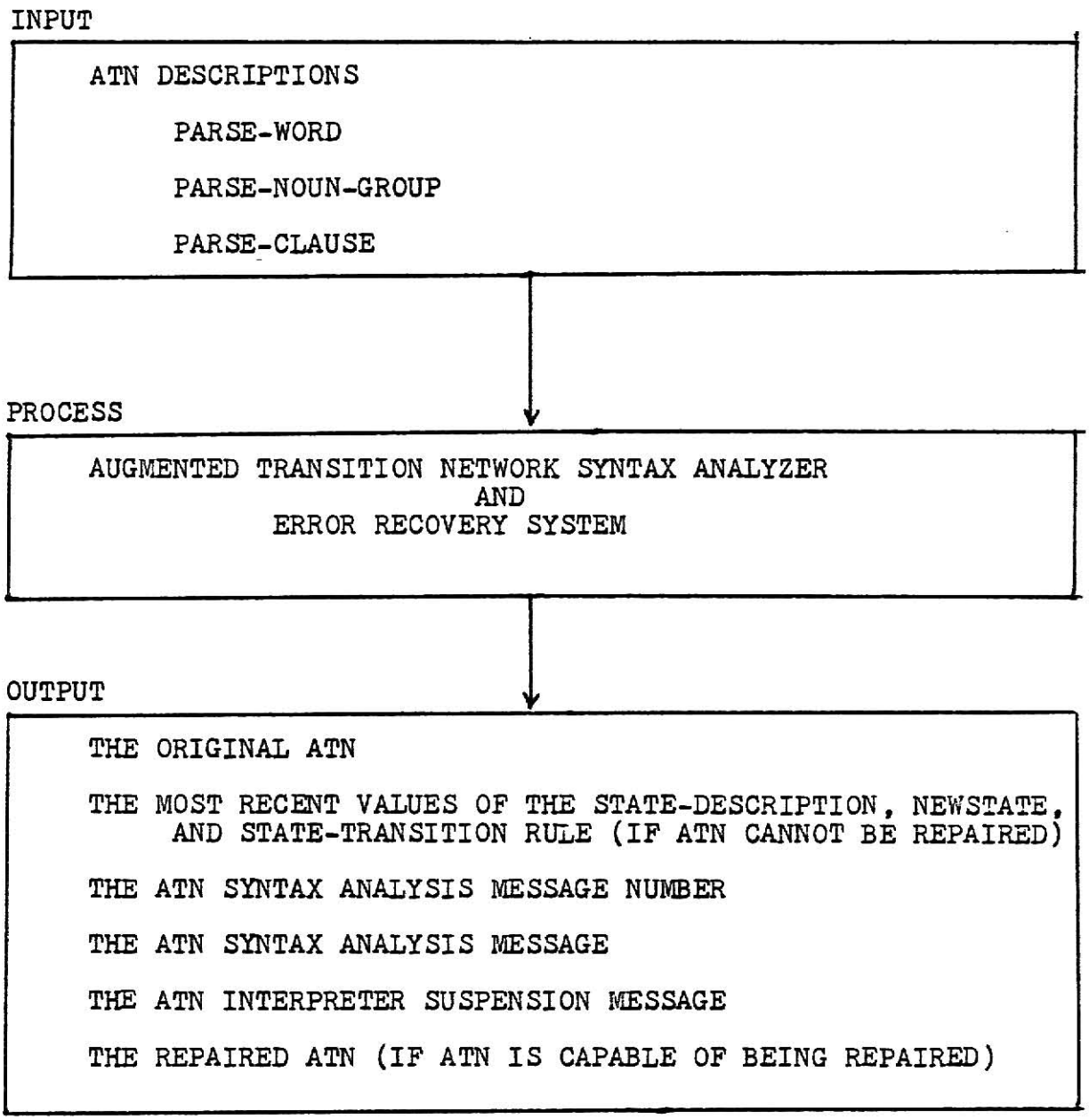


Figure 3.1: The Program Flow for the Augmented Transition Network (ATN) Syntax Analyzer.

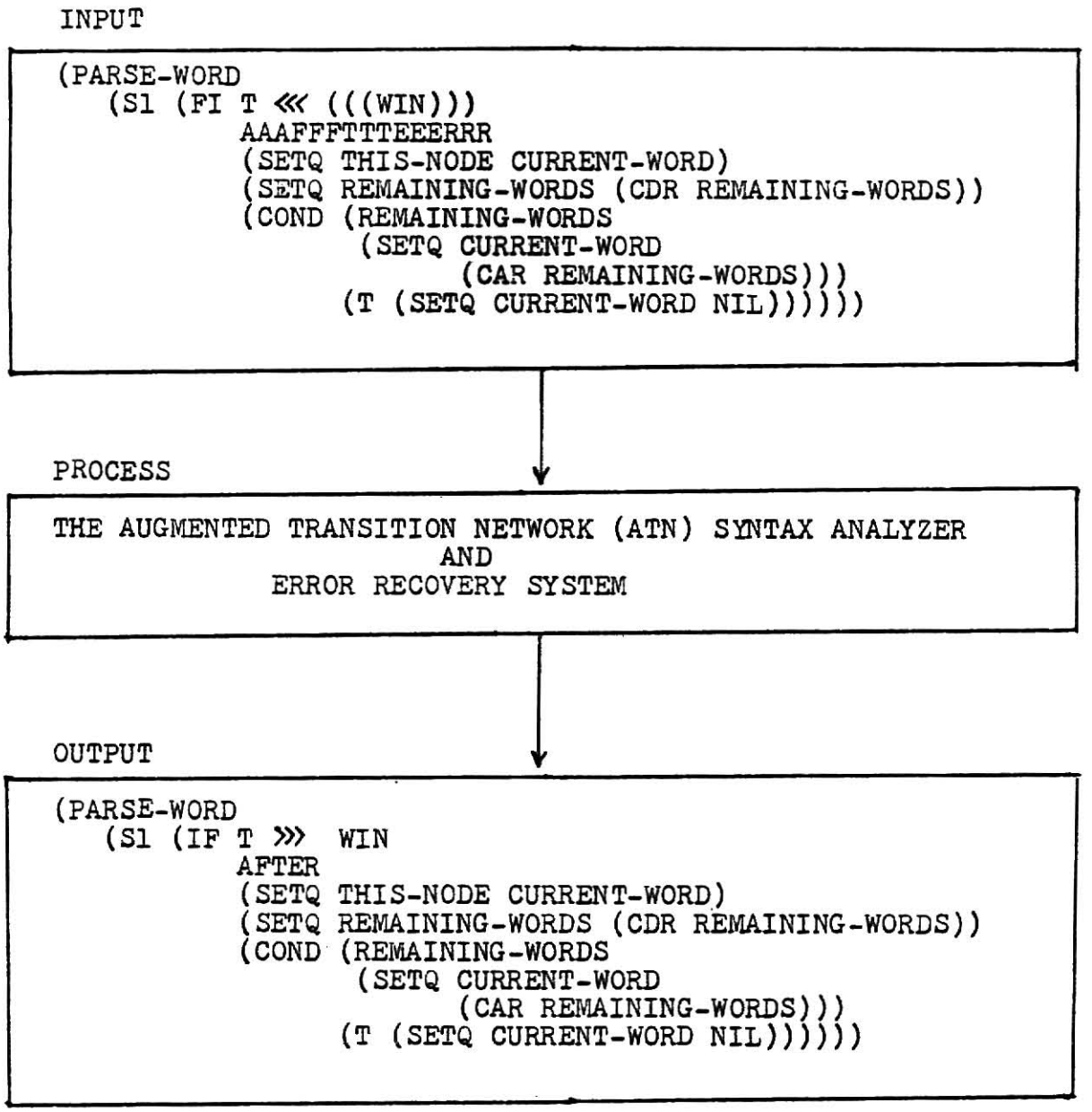


Figure 3.2: A sample transformation that involves the error recovery system of the ATN Syntax Analyzer.

3.1.3 The Augmented Transition Network (ATN) Syntax

The syntax of the Augmented Transition Network can be described by using the traditional Backus-Naur Form (BNF), where syntactic constructs are denoted by English words enclosed between the angular brackets < and > . Such words suggest the meaning of the construct.

The symbols used to help describe the ATN syntax are outlined below.

The following meta-symbols belong to the Regular BNF formalism:

::= denotes the production symbol of a syntactic construct.

| denotes the logical operator which means 'or'.

The following meta-symbols belong to the Extended BNF formalism:

[] denotes an optional occurrence of the enclosed symbol(s).

$[]^{+}$ denotes a possible repetition of the enclosed symbol(s) one or more times.

The production rules of the Augmented Transition Network Syntax are outlined in Figure 3.1 while the ATN Syntax diagrams are illustrated in Figure 3.2.

⟨ATN⟩ ::= ([⟨state-description⟩]$^{+}$)

⟨state-description⟩ ::= (⟨node name⟩ [⟨rule⟩]$^{+}$)

⟨rule⟩ ::= (IF ⟨test⟩ ⋙ ⟨new node⟩ [AFTER [⟨side effects⟩]$^{+}$])

⟨test⟩ ::= (⟨function-name⟩ ⟨arguments⟩) | T

⟨side effects⟩ ::= (⟨function-name⟩ ⟨arguments⟩)

⟨node name⟩ ::= ⟨atom⟩

⟨new node⟩ ::= ⟨atom⟩

Figure 3.3: The Production Rules of the Augmented Transition Network (ATN) Syntax.

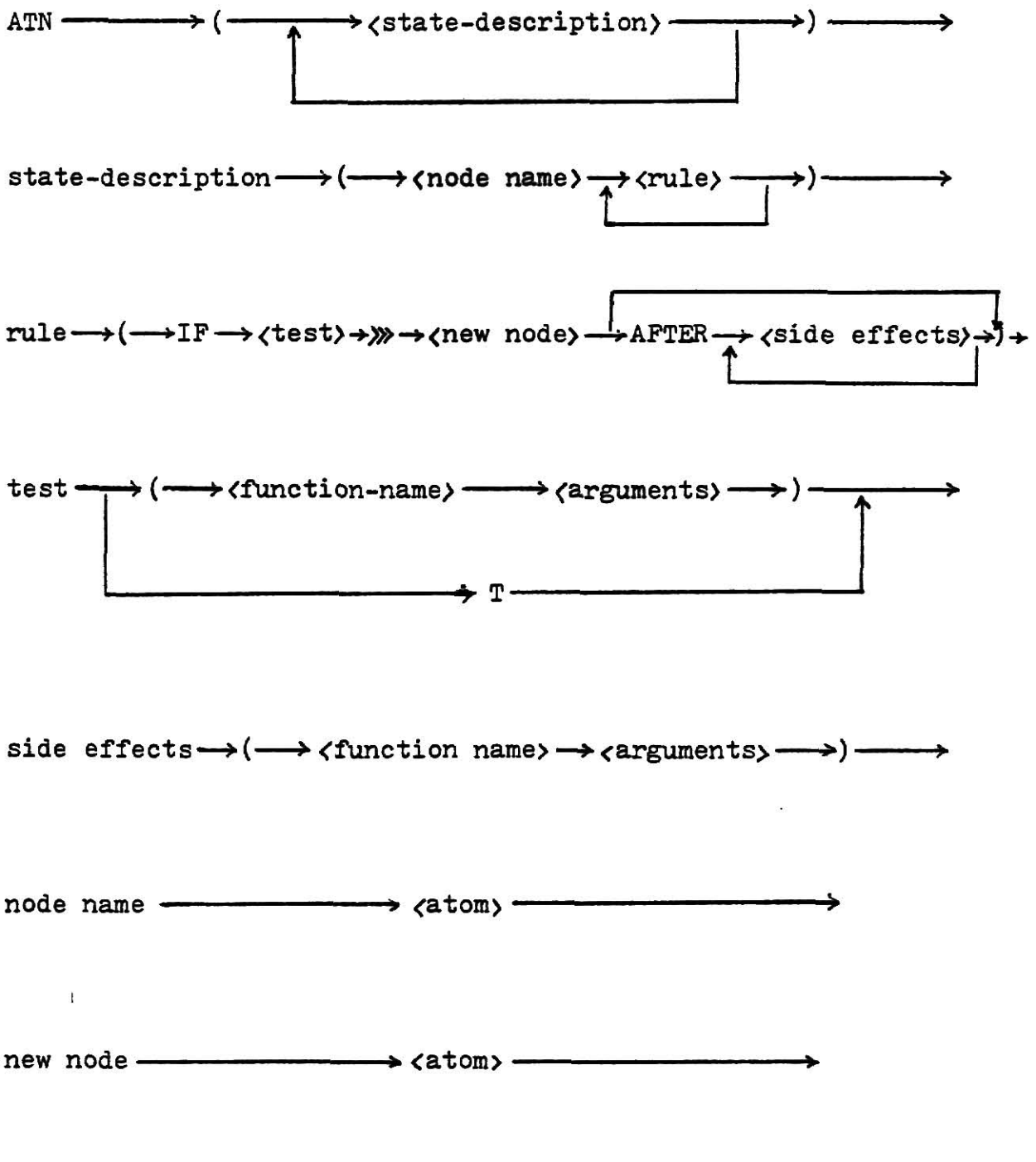


Figure 3.4: The Augmented Transition Network (ATN) Syntax Diagram. These are based on the ATN Production Rules.

3.1.4 Functional Description of the ATN Syntax Analyzer

The Augmented Transition Network Syntax Analyzer is designed as an aid to the ATN Interpreter. It provides a means of syntactically investigating an ATN description before the ATN Interpreter can even function. The ATN Syntax Analyzer includes an Error Recovery feature so that subsequent syntax errors may be detected and possibly corrected, eventually saving the entire ATN description. A listing is produced which consists of the ATN description, the repaired ATN description, if any, the ATN syntax error codes with the corresponding error messages, if any, and the results of the ATN Interpreter, considering that the original ATN description is syntactically correct or is made to be syntactically correct.

Description of Input

The input processed by an Augmented Transition Network Syntax Analyzer is an Augmented Transition Network description. Three ATN descriptions are syntactically investigated by the Syntax Analyzer. These are the following:

1. Parse-Word
2. Parse-Noun-Group
3. Parse-Clause

Description of Output

Listing of Output

The listing output from the Augmented Transition Network Syntax Analyzer consists of the following, depending upon the cases given below:

Case 1: If the original Augmented Transition Network is syntactically incorrect, then the following are output:

1. The original Augmented Transition Network description.
2. The most recent values of the ATN state-description, the newstate, and the state-transition rule.
3. The ATN syntax analysis message number which ranges from Network-Syntax-Error-Message1 to Network-Syntax-Error-Message9 or from Rule-Syntax-Error-Message1 to Rule-Syntax-Error-Message34.
4. The ATN syntax analysis message reflecting the reason for the error.
5. The ATN syntax analysis message reflecting the suspension of the interpreting process due to ATN syntax errors.

Case 2: If the original Augmented Transition Network is syntactically correct, then the following are output:

1. The original Augmented Transition Network (ATN) description.
2. The ATN syntax analysis message number which has a NIL value for this case.
3. The ATN syntax analysis message reflecting the fact that no syntax errors have been found.

Case 3: If the original Augmented Transition Network is syntactically incorrect but has been repaired, then the following are output:

1. The original Augmented Transition Network (ATN) description.
2. The repaired Augmented Transition Network (ATN) description.
3. The ATN syntax analysis message number which ranges from Network-Syntax-Error-Message1 to Network-Syntax-Error-Message9 or from Rule-Syntax-Error-Message1 to Rule-Syntax-Error-Message34. In the case of multiple repairs, then the most recent message number will be output.
4. The ATN syntax analysis message reflecting the reason for the error.

Messages

The following messages will be output to the output device if a detected situation warrants such a message to be displayed.

The Augmented Transition Network Title

The message

THE-ORIGINAL-AUGMENTED-TRANSITION-NETWORK-TITLE-IS <ATN Title>

indicates the Augmented Transition Network name whose body will be syntactically investigated in a thorough manner.

The Augmented Transition Network Body

The message

THE-ORIGINAL-AUGMENTED-TRANSITION-NETWORK-IS

<ATN Body>

indicates the Augmented Transition Network body.

The Augmented Transition Network State-Description

The message

THE-STATE-DESCRIPTION-IS

<ATN state-description>

indicates the most recent value of the ATN state-description.

The Augmented Transition Network Newstate

The message

THE-NEWSTATE-IS

<ATN newstate>

indicates the most recent value of the ATN newstate.

The Augmented Transition Network State-Transition Rule

The message

THE-RULE-IS

⟨ATN state-transition rule⟩

indicates the most recent value of the ATN state-transition rule.

The Augmented Transition Network Syntax Analysis Message Number, if syntax errors occur

The message

THE-NETWORK-SYNTAX-ANALYSIS-MSSG-CODE-IS

⟨ATN syntax analysis message number⟩

indicates the error message number of the ATN syntax analysis.

The Augmented Transition Network Syntax Message, if syntax errors occur

The message

THE-NETWORK-SYNTAX-ERROR-MESSAGE-IS

⟨ATN syntax analysis message⟩

indicates the reason for the ATN syntax error or subsequent repair.

The Augmented Transition Network Interpreter Suspension Message

The message

THE-FINAL-NETWORK-SYNTAX-ERROR-MESSAGE-IS

(FINAL ERROR MESSAGE *** THE ENGLISH CLAUSE
SYNTAX IDENTIFICATION PROCESS HAS BEEN SUSPENDED
DUE TO AUGMENTED TRANSITION NETWORK SYNTAX ERRORS)

indicates the suspension of the English clause Interpreting

Process due to ATN syntax errors.

The Augmented Transition Network Syntax Analysis Message Number, if no syntax errors have been detected or if a repair has been made.

The message

THE-FINAL-NETWORK-SYNTAX-ANALYSIS-MSSG-CODE-IS

⟨ATN syntax analysis message number⟩

indicates the message number of the ATN Syntax Analysis when no errors have been detected or a repair of the ATN has been made.

The Augmented Transition Network Syntax Analysis Message, if no syntax errors have been detected or if a repair has been made.

The message

THE-FINAL-SYNTAX-ANALYSIS-MESSAGE-FOR-THE-NETWORK-IS

⟨ATN syntax analysis message⟩

indicates the message of the ATN Syntax Analysis when no errors have been detected or a repair has been made.

3.2 Design Specification for the ATN Syntax Analyzer

3.2.1 Scope

This section discusses the Error Recovery feature and the design considerations of the Augmented Transition (ATN) Syntax Analyzer.

3.2.2 General Description

The guiding principle behind the design of the ATN Syntax Analyzer has been the observance of the qualities of good program workmanship, namely, reliability, modifiability, maintainability, modularity, robustness, generality, portability, and efficiency. The process of fetching an ATN description syntactic component and the methods of properly identifying, diagnosing, and possibly correcting any ATN syntax errors are accomplished by the availability of layers of routines to attain syntax investigation and error recovery. In that respect, the ATN Syntax Analyzer supports abstraction and hierarchical structuring of the routines very systematically.

3.2.3 The Error Recovery Feature of the ATN Syntax Analyzer

The ATN Syntax Analyzer includes a significant operation which is Error Recovery. The error recovery feature takes control when certain syntax errors in the ATN description are detected. Its purpose is to diagnose the syntax error and to attempt some kind of recovery, so that the ATN need not be discarded immediately. The ATN Syntax Analyzer Error Recovery System may, for example, decide to insert missing ATN syntactic components or to replace invalid ATN syntactic components with the proper ATN syntactic components.

The objectives of the ATN Syntax Analyzer Error Recovery System are as follows:

1. Identify and diagnose the ATN syntax error, as an aid to the ATN syntax correction.
2. Recover from the syntax error, such that subsequent syntax errors may be detected.

The routines for identifying, diagnosing, and possibly correcting any ATN syntax errors are organized in a hierarchical fashion. To be able to identify and to diagnose an ATN syntax error, an ATN syntactic component has to be fetched from an ATN description and then tested to determine its syntactic validity. Iteratively, the routine CHECK-NETWORK-SYNTAX fetches each syntactic component of the ATN description, namely, the state-description, the ATN node name or newstate, and the state-transition rule. Immediately, routines are invoked to identify any ATN syntax errors. The routines are

CHECK-STATE-DESCRIPTION-SYNTAX, CHECK-NODE-NAME-IN-STATE-DESCRIPTION, and CHECK-RULE-SYNTAX. Subsequent state-transition rule syntactic segments are fetched and tested to identify and diagnose any ATN syntax errors by invoking other syntax checking routines. The routines are CHECK-KEYWORD-IF-IN-TRANSITION-RULE, CHECK-TEST-IN-TRANSITION-RULE, CHECK-STATE-TRANSITION-SYMBOL-IN-TRAN-RULE, CHECK-NEW-NODE-IN-TRANSITION-RULE, CHECK-KEYWORD-AFTER-IN-TRANSITION-RULE, and lastly, CHECK-SIDE-EFFECTS-SYNTAX.

When the identification and diagnosis of the ATN syntax errors are accomplished and a possibility for the ATN to recover from any syntax errors arises, then the proper error recovery routines are invoked from a secondary layer of procedures. This layer, called the ATN Syntax Error Recovery System, is composed of Insertion and Replacement routines. Figures 3.5 to 3.7 illustrate the layers.

The decision to insert missing ATN syntactic components depends upon a detailed examination of the proceeding syntactic segment based upon the ATN production rules. This is called a look-ahead situation. For instance, inserting the ATN keyword IF depends upon what syntactic component should come after the keyword IF. Other cases should hold true for the ATN state-transition symbol and the ATN keyword AFTER.

The Insertion routines are REPAIR-VIA-KEYWORD-IF-INSERTION, REPAIR-VIA-STATE-TRAN-SYMBOL-INSERTION, INSERT-KEYWORD-IF-INTO-NETWORK-TRAN-RULE, and lastly, INSERT-TRAN-SYMBOL-INTO-NETWORK-TRAN-RULE.

The decision to replace invalid ATN syntactic components with the proper ATN syntactic components or segments depends upon a detailed examination of the present syntactic segment, again based upon the ATN production rules. For example, the first segment of a state-transition rule might be a list. However, embedded in the list might be the ATN keyword IF. In such a situation, replacement takes place.

The Replacement routines are REPAIR-STATE-DESCRIPTION-AND-NETWORK, REPAIR-VIA-KEYWORD-IF-REPLACEMENT, REPAIR-VIA-TEST-REPLACEMENT, REPAIR-VIA-STATE-TRAN-SYMBOL-REPLACEMENT, REPAIR-VIA-TRAN-RULE-NEW-NODE-REPLACEMENT, REPAIR-VIA-KEYWORD-AFTER-REPLACEMENT, REPLACE-RULE-SYNTAX-SEGMENT-WITH-KEYWORD-IF, REPLACE-RULE-SYNTAX-SEGMENT-WITH-STATE-TRAN-SYMBOL, REPLACE-RULE-SYNTAX-SEGMENT-WITH-TRAN-RULE-NEW-NODE, and lastly, REPLACE-RULE-SYNTAX-SEGMENT-WITH-KEYWORD-AFTER.

Repairing the ATN state-description means repairing the entire ATN description. Likewise, repairing the ATN state-transition rule means repairing the ATN state-description and the entire ATN. The routines which repair the ATN state-description and the entire ATN are LIST-DIFFERENCE and REPAIR-STATE-DESCRIPTION-AND-NETWORK. The LISP function SUBST plays a major role in the repair process.

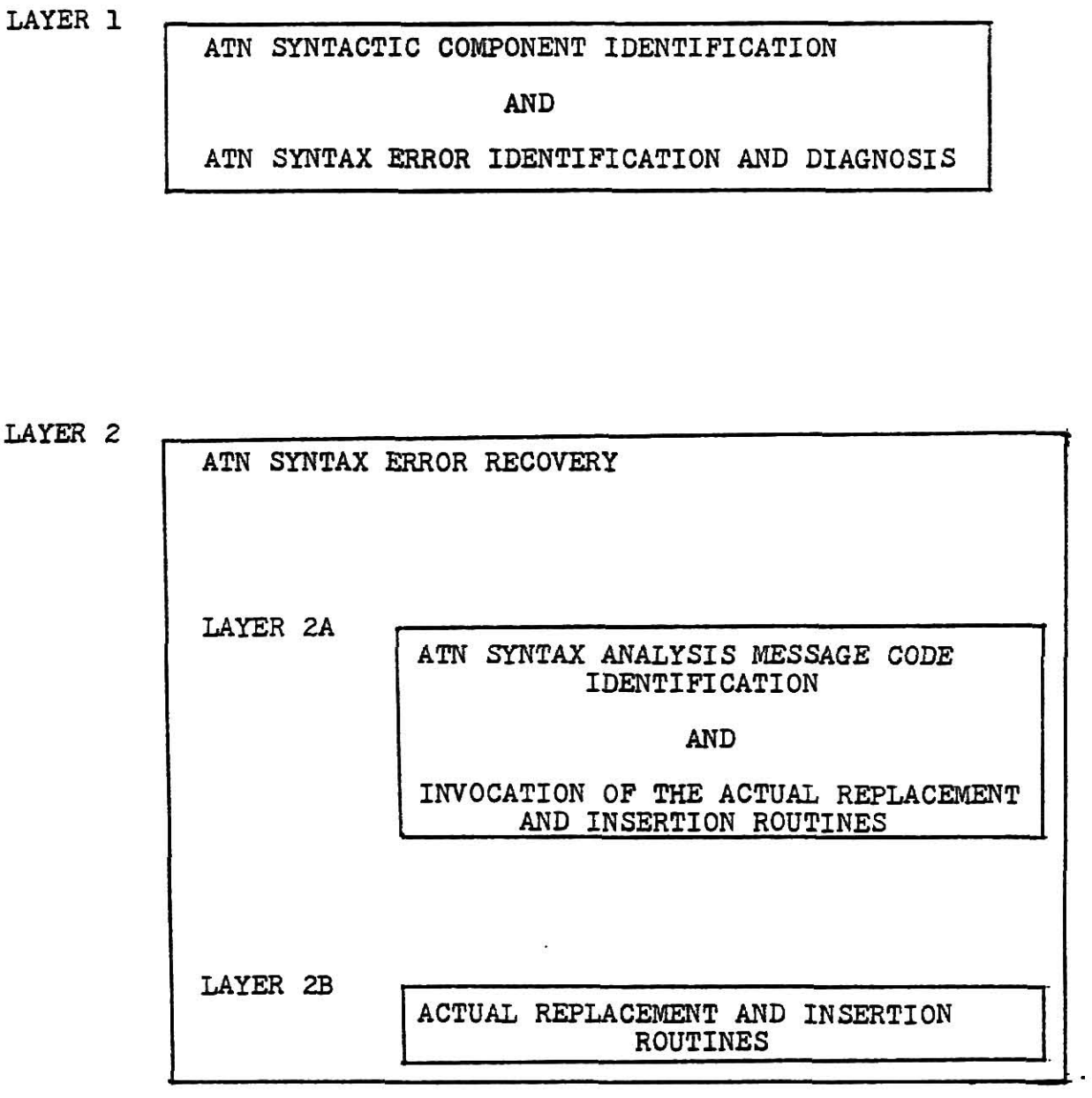


Figure 3.5: The Augmented Transition Network (ATN) Syntax Analyzer Hierarchy.

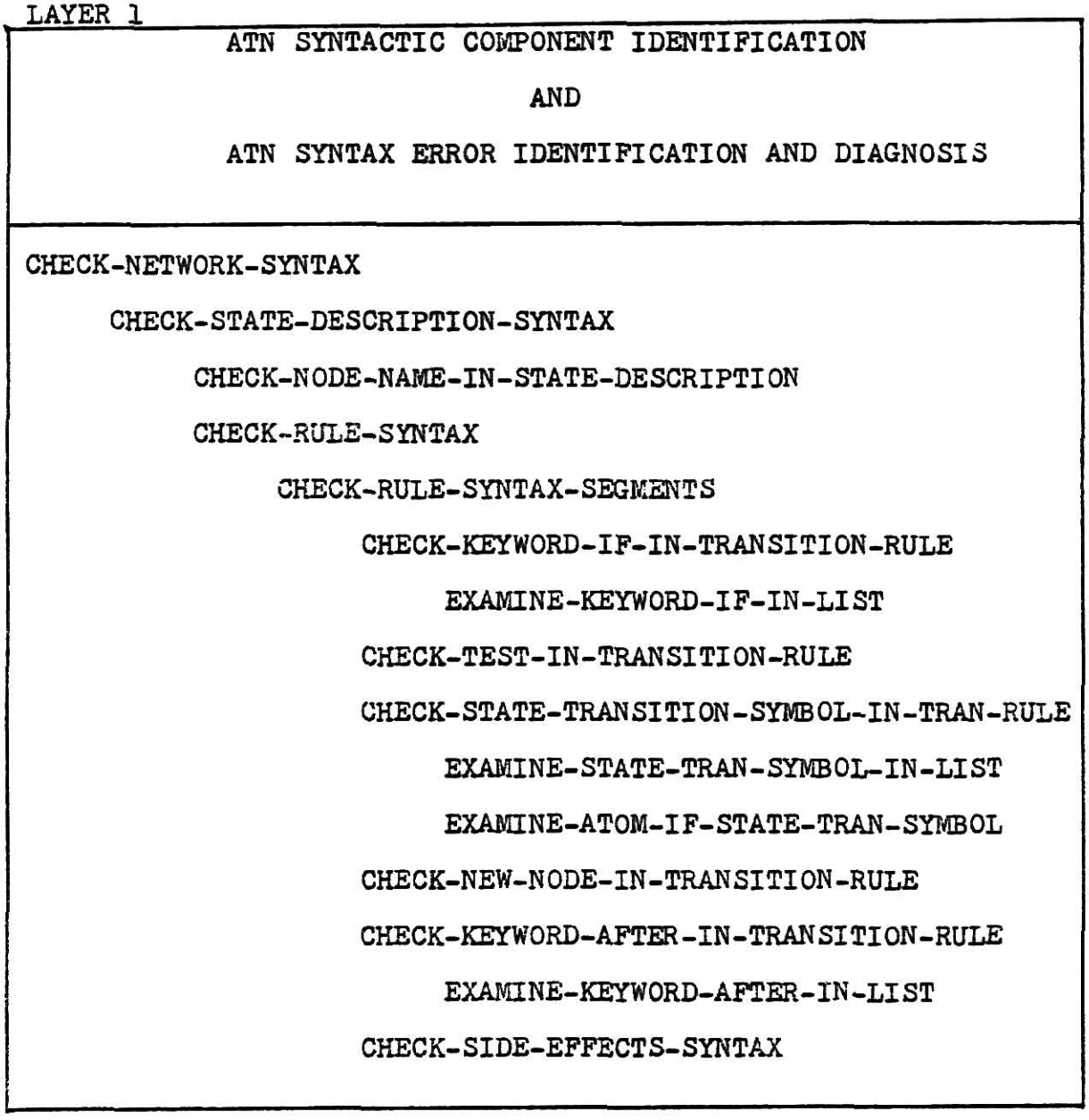


Figure 3.6: The First Layer of the Augmented Transition Network (ATN) Syntax Analyzer Hierarchy.

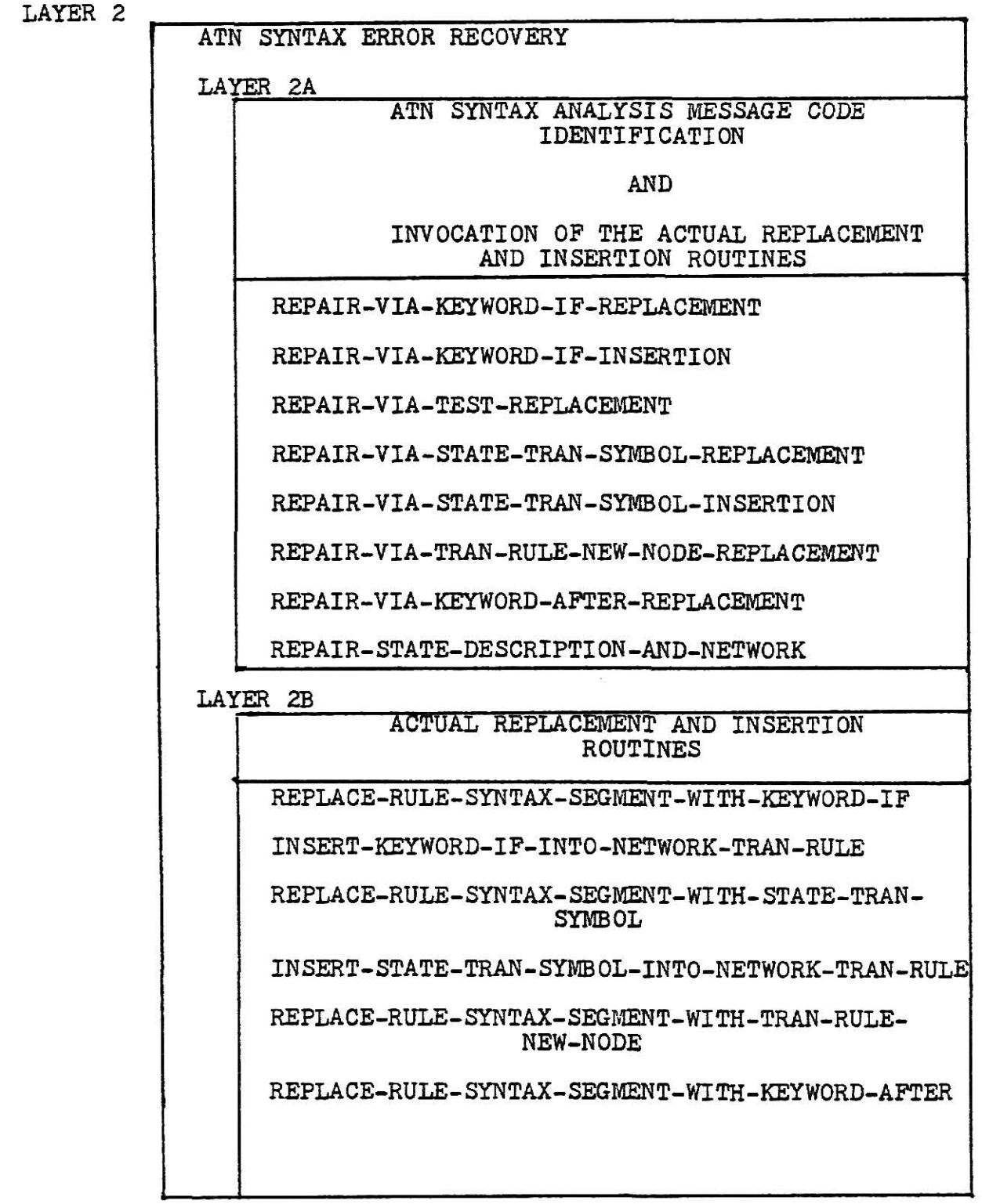


Figure 3.7: The Second Layer of the Augmented Transition Network (ATN) Syntax Analyzer Hierarchy.

3.2.4 Design Considerations

Reliability

The ATN Syntax Analyzer is capable of accomplishing its functional objectives. It is able to fetch an ATN syntactic component, properly identify, diagnose, and possibly correct any syntax errors. In addition, the ATN Syntax Analyzer can help the ATN description recover from single or multiple syntax errors.

Modifiability

The ATN Syntax Analyzer employs standard LISP functions so as to keep invariant the logical effects of the system. The system is modularized to be able to adapt to controlled changes in order to add new capabilities and thus improve the performance of the system.

Maintainability

In order to facilitate maintainability, the ATN Syntax Analyzer is designed to be modular, readable, and understandable. In this manner, it is easy to keep the system current and ultimately correct.

The dictionary and the English clause are located in the function IDENTIFY. The words in the dictionary are alphabetized for easy reference, insertion, and deletion. A new English sentence can be introduced simply by changing the value of English-Clause.

Modularity

In order to achieve modularity, the ATN Syntax Analyzer is made into a collection of layers of routines where the whole system can be read routine by routine as one would read a novel. In modularization, the system supports abstraction and hierarchical structuring very nicely. Consequently, modularization structures the system to make it readable and understandable. The physical length of each routine is limited to a page.

Robustness

The ATN Syntax Analyzer, when provided with erroneous ATN syntax, will provide meaningful data on the CRT. A total of forty-three possible combinations of syntax errors are listed in order to avoid an abnormal termination of the system. These are located in the routines INITIALIZE-NETWORK-SYNTAX-ERROR-MESSAGES and INITIALIZE-RULE-SYNTAX-ERROR-MESSAGES.

Generality

The ATN Syntax Analyzer, together with its Error Recovery System, is designed to cope with any arbitrary symbolic-expressions. The system can therefore recover from a wide variety of invalid syntactic components.

Portability

The ATN Syntax Analyzer utilizes standard LISP features to make it adaptable to other systems.

Efficiency

The ATN Syntax Analyzer and the output are user-efficient since both are readable and understadable. The users will not spend much time and effort to learn how to use the program, how to prepare the data, and how to interpret and possibly use the output.

CHAPTER 4

THE AUGMENTED TRANSITION NETWORK (ATN) COMPILER

4.1 Functional Specification for the ATN Compiler

4.1.1 Scope

This section defines the functional capabilities of the Augmented Transition Network (ATN) Compiler.

4.1.2 General Description

The Augmented Transition Network (ATN) Compiler is capable of accepting ATN descriptions as input to be translated into a form that LISP understands directly. The syntax of the ATN descriptions should conform syntactically with the ATN production rules outlined in Chapter 3 since the ATN Compiler does not have a facility to investigate the syntax of an ATN description. Figure 4.1 illustrates the program flow of the ATN Compiler. Figure 4.2 provides a sample transformation. Figures 4.3 and 4.4 illustrate the results of compiling the ATN descriptions PARSE-NOUN-GROUP and PARSE-CLAUSE, respectively.

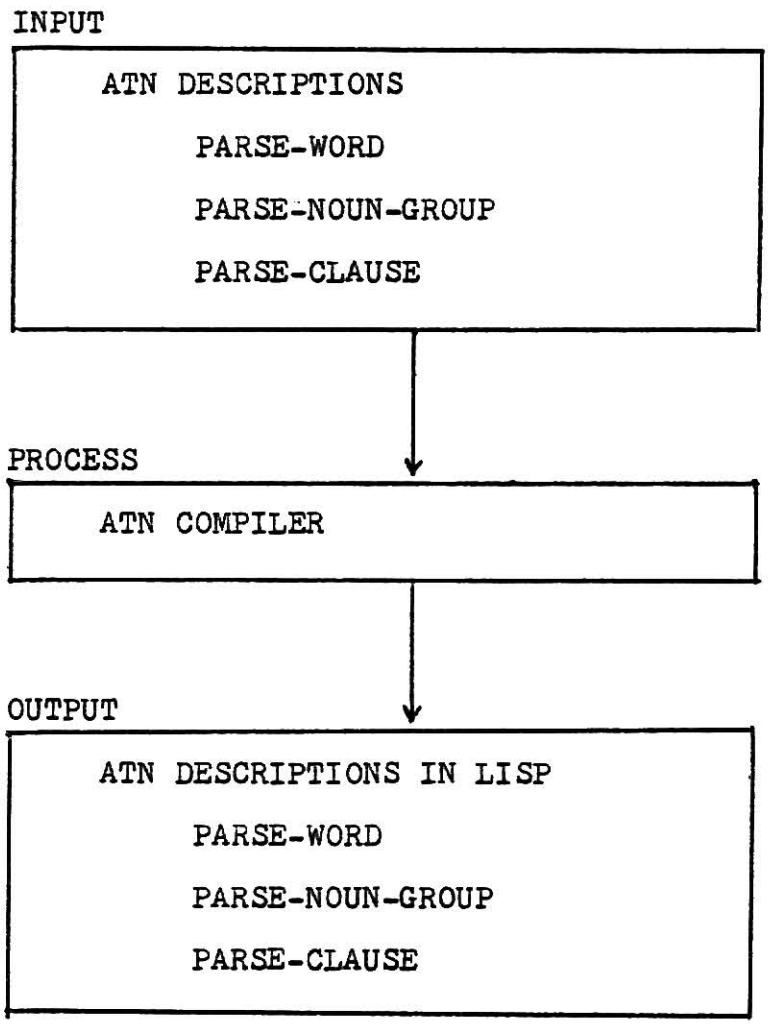


Figure 4.1: The Program Flow for the Augmented Transition Network (ATN) Compiler.

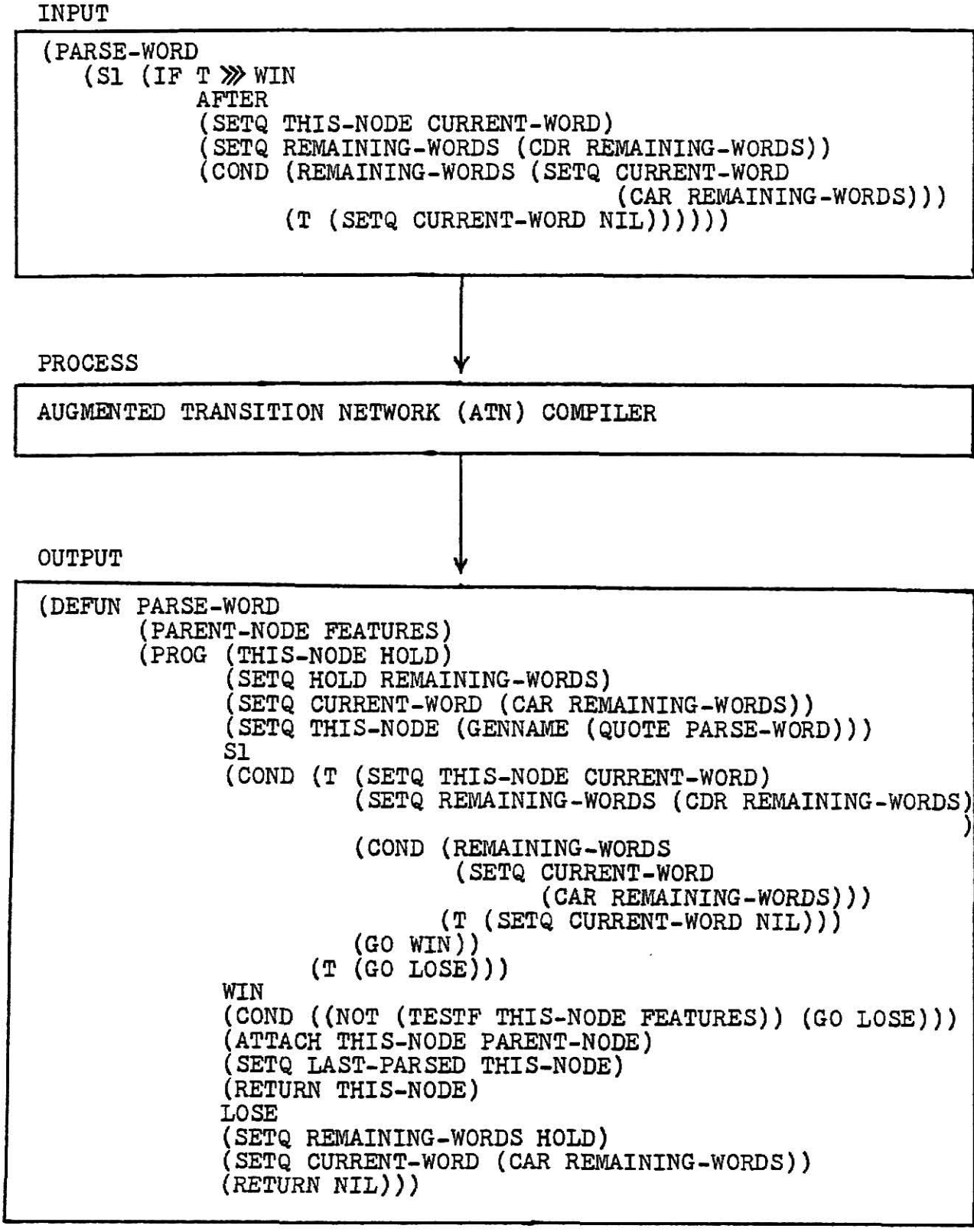


Figure 4.2: A Sample Transformation involving the ATN desription PARSE-WORD.

```
(DEFUN PARSE-NOUN-GROUP
       (PARENT-NODE FEATURES)
       (PROG (THIS-NODE HOLD)
             (SETQ HOLD REMAINING-WORDS)
             (SETQ CURRENT-WORD (CAR REMAINING-WORDS))
             (SETQ THIS-NODE (GENNAME (QUOTE PARSE-NOUN-GROUP)))
             S1
             (COND ((PARSE-WORD THIS-NODE (QUOTE DETERMINER))
                    (SETR (QUOTE NUMBER)
                          (SELECT (QUOTE (SINGULAR PLURAL))
                                  (GETF LAST-PARSED)))
                    (SETR (QUOTE DETERMINER)
                          (SELECT (QUOTE (DEFINITE INDEFINITE))
                                  (GETF LAST-PARSED)))
                    (GO S2))
                   (T (GO LOSE)))
             S2
             (COND ((PARSE-WORD THIS-NODE (QUOTE ADJECTIVE))
                    (ADDR (QUOTE ADJECTIVES) LAST-PARSED)
                    (GO S2))
                   ((PARSE-WORD THIS-NODE (QUOTE NOUN))
                    (SETR
                     (QUOTE NUMBER)
                     (SELECT (QUOTE (SINGULAR PLURAL))
                             (GETF LAST-PARSED)))
                    (SETR (QUOTE NOUN) LAST-PARSED)
                    (GO WIN))
                   (T (GO LOSE)))
             WIN
             (COND ((NOT (TESTF THIS-NODE FEATURES))
                    (GO LOSE)))
             (ATTACH THIS-NODE PARENT-NODE)
             (SETQ LAST-PARSED THIS-NODE)
             (RETURN THIS-NODE)
             LOSE
             (SETQ REMAINING-WORDS HOLD)
             (SETQ CURRENT-WORD (CAR REMAINING-WORDS))
             (RETURN NIL)))
```

Figure 4.3: The result of compiling the ATN description PARSE-NOUN-GROUP.

```
(DEFUN PARSE-CLAUSE
       (PARENT-NODE FEATURES)
       (PROG (THIS-NODE HOLD)
             (SETQ HOLD REMAINING-WORDS)
             (SETQ CURRENT-WORD (CAR REMAINING-WORDS))
             (SETQ THIS-NODE (GENNAME (QUOTE PARSE-CLAUSE)))
             S1
             (COND ((PARSE-NOUN-GROUP THIS-NODE NIL)
                    (SETR (QUOTE SUBJECT) LAST-PARSED) (GO S2))
                   (T (GO LOSE)))
             S2
             (COND ((PARSE-WORD THIS-NODE (QUOTE (VERB TENSED)))
                    (SETR (QUOTE VERB)
                          (GET LAST-PARSED (QUOTE ROOT)))
                    (GO S3))
                   (T (GO LOSE)))
             S3
             (COND ((AND (EQUAL (GETR (QUOTE VERB)) (QUOTE BE))
                         (PARSE-WORD THIS-NODE
                                     (QUOTE PASTPARTICIPLE)))
                    (SETR (QUOTE OBJECT) (GETR (QUOTE SUBJECT)))
                    (SETR (QUOTE SUBJECT) NIL)
                    (SETR (QUOTE VERB) LAST-PARSED)   (GO S4))
                   ((AND (TESTF (GETR (QUOTE VERB))
                                (QUOTE TRANSITIVE))
                         (PARSE-NOUN-GROUP THIS-NODE NIL))
                    (SETR (QUOTE OBJECT) LAST-PARSED)   (GO S4))
                   ((OR (TESTF (GETR (QUOTE VERB))
                               (QUOTE INTRANSITIVE))
                        (GETR (QUOTE OBJECT)))   (GO S4))
                   (T (GO LOSE)))
             S4
             (COND ((AND (GETR (QUOTE SUBJECT))
                         (NULL REMAINING-WORDS))   (GO WIN))
                   ((AND (NOT (GETR (QUOTE SUBJECT)))
                         (EQUAL CURRENT-WORD (QUOTE BY))
                         (PARSE-WORD THIS-NODE NIL))   (GO S5))
                   ((NOT (GETR (QUOTE SUBJECT)))
                    (SETR (QUOTE SUBJECT) (QUOTE SOMEONE))
                    (GO S4))
                   (T (GO LOSE)))
             S5
             (COND ((PARSE-NOUN-GROUP THIS-NODE NIL)
                    (SETR (QUOTE SUBJECT) LAST-PARSED)   (GO S4))
                   (T (GO LOSE)))
             WIN
             (COND ((NOT (TESTF THIS-NODE FEATURES))  (GO LOSE)))
             (ATTACH THIS-NODE PARENT-NODE)
             (SETQ LAST-PARSED THIS-NODE)
             (RETURN THIS-NODE)
             LOSE
             (SETQ REMAINING-WORDS HOLD)
             (SETQ CURRENT-WORD (CAR REMAINING-WORDS))
             (RETURN NIL)))
```

Figure 4.4 : The result of compiling the ATN description PARSE-CLAUSE.

4.1.3 Functional Description of the ATN Compiler

The Augmented Transition Network (ATN) Compiler translates an ATN description into a form that LISP understands directly. The ATN Compiler does not have an ATN description syntax checking facility. In that respect, the ATN descriptions which the ATN Compiler accepts must conform syntactically with the ATN production rules outlined in Chapter 3. A listing is produced which consists of the compiled ATN description and the results of the interpreting process, namely, the parents, the children, the registers, and their values. The function IDENTIFY contains the English clause to be parsed and a dictionary which specifies the properties of some English words. IDENTIFY invokes a routine created by the ATN Compiler to parse an English sentence.

Description of Input

The input processed by the ATN Compiler are ATN descriptions. Only ATN descriptions which conform syntactically to the ATN production rules are capable of being successfully and completely compiled. The ATN Interpreter can accept certain English sentences. Three ATN descriptions are available. These are the following:

1. PARSE-WORD
2. PARSE-NOUN-GROUP
3. PARSE-CLAUSE

Description of Output

Listing of Output

The listing output from the Augmented Transition Network (ATN) Compiler consists of the following:

1. The compiled ATN description
2. The results of the ATN Interpreter which are represented by the parents, the children, the registers and their values.

Messages

The following messages will be output to the output device if a detected situation warrants such a message to be displayed.

The Compiled Augmented Transition Network

The message

THE-COMPILED-AUGMENTED-TRANSITION-NETWORK-IS

⟨The Compiled ATN⟩

indicates that the ATN description is in a form that LISP understands directly.

The English Clause to be Parsed

The message

THE-ENGLISH-CLAUSE-TO-BE-PARSED-IS

⟨An English Clause⟩

indicates the English clause to be parsed.

The Results of the Interpreting Process

The message

THE-RESULTS-OF-THE-INTERPRETING-PROCESS-ARE

where the parents, the children, the registers, and the register values are indicated by

THE-PARENT-IS ⟨parent⟩

THE-CHILD-IS ⟨child⟩

THE-REGISTER-IS ⟨register name⟩

THE-VALUE-IS ⟨register value⟩

indicates the results of the interpreting process.

4.2 Design Specification for the ATN Compiler

4.2.1 Scope

This section highlights the significant functions that make up the Augmented Transition Network Compiler.

4.2.2 Pertinent Functions

COMPILE is a user-defined MACRO routine that converts perspicuous descriptions into descriptions that LISP understands directly. It translates source-language descriptions by ripping apart a transition specifying parcel and reassembling it into a COND clause. Several clauses for each ATN node are amalgamated into a single COND structure where CARs, CDRs, CONS, and APPENDs abound.

Thus, the following invocation

```
( COMPILE  <ATN name>
           <ATN body>  )
```

will have the resultant effect;

```
<node name>
( COND  ( <test>  <side effect>
                  .
                  .
                  .
          <side effect>
          ( GO <new node>  ))
        ( T  ( GO LOSE )))
```

IDENTIFY is the function that is called by the user to execute the ATN compiler function COMPILE and to parse an English sentence. The body of the function IDENTIFY consists of the English clause to be parsed and a dictionary which specifies the properties of some English words. IDENTIFY invokes a routine PARSE-CLAUSE created by the ATN Compiler to parse an English sentence.

GETF, TESTF, ATTACH, SELECT, GENNAME, SETR, GETR, and ADDR are significant auxiliary functions in analyzing certain aspects of English syntax.

CHAPTER 5

FUTURE DIRECTIONS AND CONCLUSION

5.1 Future Directions

In any undertaking, there is always room for changes and, hopefully, improvements. This endeavor is no exception.

The ATN descriptions that this project accepted had a definite syntactic structure. There may be other ways of syntactically defining those ATN descriptions but still maintain the underlying meaning. If such is the case, then the ATN production rules and the ATN syntax diagrams will need to be re-defined. More importantly, the ATN Syntax Analyzer will have to be completely overhauled but the design principles should still be preserved.

If the case develops in which the ATN Interpreter will have to accept a wider range of English syntax, other ATN descriptions based on varying production rules might be presented. It will then be necessary, therefore, to design and implement further ATN Syntax Analyzers and incorporate them in a single package.

5.2 Conclusion

This project has become a thorough exercise in learning how to appreciate designing and implementing programs that exhibit intelligent behavior. Such a project is just a tip of a large iceberg.

The ATN Interpreter has demonstrated that ATNs certainly capture aspects of English Syntax. LISP has made the understanding of ATNs an easy process. Registers have added power to ATN description note-taking.

The most important feature of the ATN Syntax Analyzer has been the Error Recovery System. The Syntax Analyzer, by way of routine layers, has systematically identified, diagnosed, and at times corrected ATN Syntax Errors. In that view, the ATN Syntax Analyzer supported abstraction and hierarchical structuring.

LISP has shown its elegance in the ATN Compiler. Since compiling is a symbol-manipulating task, LISP has proven to be eminently suited. The ATN Compiler does complement the ATN Interpreter since it has made the ATN description more readable to readers who are well-acquainted with LISP.

BIBLIOGRAPHY

Barrett, William A. and John D. Couch, Compiler Construction: Theory and Practice, Science Research Associates, Inc., 1979.

Maclin, Alice, Reference Guide to English, CBS College Publishing, 1981.

Winston, Patrick Henry and Berthold Klaus Paul Horn, LISP, Addison-Wesley Publishing Company, 1981.

APPENDIX A

THE AUGMENTED TRANSITION NETWORK (ATN) INTERPRETER AND SYNTAX ANALYZER COMPONENT DESCRIPTION

The following constitute the description of the routines and the local data for the ATN Interpreter and the ATN Syntax Analyzer.

ROUTINE: CAAR

OBJECTIVE: CAAR is a routine that takes a list as its argument and evaluates it according to the following sequence: CAR, CAR.

FUNCTION USED: CAR

BOUND VARIABLE: Listl represents the list to be evaluated.

FREE VARIABLES: None

ROUTINE: CADR

OBJECTIVE: CADR is a routine that takes a list as its argument and evaluates it according to the following sequence: CDR, CAR.

FUNCTIONS USED: CAR, CDR

BOUND VARIABLE: Listl represents the list to be evaluated.

FREE VARIABLES: None

ROUTINE: CADDR

OBJECTIVE: CADDR is a routine that takes a list as its argument and evaluates it according to the following sequence: CDR, CDR, CAR.

FUNCTIONS USED: CAR, CDR

BOUND VARIABLE: Listl represents the list to be evaluated.

FREE VARIABLES: None

ROUTINE: CADDDR

OBJECTIVE: CADDDR is a routine that takes a list as its argument and evaluates it according to the following sequence: CDR, CDR, CDR, CAR.

FUNCTIONS USED: CAR, CDR

BOUND VARIABLE: Listl represents the list to be evaluated.

FREE VARIABLES: None

ROUTINE: CADDDDR

OBJECTIVE: CADDDDR is a routine that takes a list as its argument and evaluates it according to the following sequence: CDR, CDR, CDR, CDR, CAR.

FUNCTIONS USED: CAR, CDR

BOUND VARIABLE: Listl represents the list to be evaluated.

FREE VARIABLES: None

ROUTINE: CADDDDDR

OBJECTIVE: CADDDDDR is a routine that takes a list as its argument and evaluates it according to the following sequence: CDR, CDR, CDR, CDR, CDR, CAR.

FUNCTIONS USED: CAR, CDR

BOUND VARIABLE: Listl represents the list to be evaluated.

FREE VARIABLES: None

ROUTINE: CDAR

OBJECTIVE: CDAR is a routine that takes a list as its argument and evaluates it according to the following sequence: CAR, CDR.

FUNCTIONS USED: CAR, CDR

BOUND VARIABLE: Listl represents the list to be evaluated.

FREE VARIABLES: None

ROUTINE: CDDR

OBJECTIVE: CDDR is a routine that takes a list as its argument and evaluates it according to the following sequence: CDR, CDR.

FUNCTION USED: CDR

BOUND VARIABLE: Listl represents the list to be evaluated.

FREE VARIABLES: None

ROUTINE: CDDDR

OBJECTIVE: CDDDR is routine that takes a list as its argument and evaluates it according to the following sequence: CDR, CDR, CDR.

FUNCTION USED: CDR

BOUND VARIABLE: Listl represents the list to be evaluated.

FREE VARIABLES: None

ROUTINE: CDDDDR

OBJECTIVE: CDDDDR is a routine that takes a list as its argument and evaluates it according to the following sequence: CDR, CDR, CDR, CDR.

FUNCTION USED: CDR

BOUND VARIABLE: Listl represents the list to be evaluated.

FREE VARIABLES: None

ROUTINE: CDDDDDR

OBJECTIVE: CDDDDDR is a routine that takes a list as its argument and evaluates it according to the following sequence: CDR, CDR, CDR, CDR, CDR.

FUNCTION USED: CDR

BOUND VARIABLE: Listl represents the list to be evaluated.

FREE VARIABLES: None

ROUTINE: REMOVE-NESTING-IN-LIST

OBJECTIVE: REMOVE-NESTING-IN-LIST is a recursive routine that takes an s-expression as its argument and returns a non-nested list of all atoms found in the s-expression.

FUNCTIONS USED: NULL, ATOM, LIST, APPEND, CAR, CDR, and REMOVE-NESTING-IN-LIST

BOUND VARIABLE: S-expr represents an s-expression that will be converted into a non-nested list.

FREE VARIABLES: None

ROUTINE: CHECK-STATE-DESCRIPTION-SYNTAX

OBJECTIVE: CHECK-STATE-DESCRIPTION-SYNTAX examines the syntax of an ATN state-description.

SYNTAX ANALYSIS: The COND function constitutes three clauses which test possible illegal occurrences of s-expressions used to represent an ATN state-description, namely, an empty-list, a non-empty list of length one, and an atom.

A non-empty list of length two or more is a valid syntactic representation of an ATN state-description. The ATN should consist of an atom and one or more lists to represent the ATN node name and the state-description rule(s), respectively.

A syntactically legal ATN state-description can be any of the following:

(⟨atom⟩ ⟨list⟩) or

(⟨atom⟩ ⟨$list_1$⟩ ... ⟨$list_n$⟩)

A syntactically illegal ATN state-description can be any of the following:

( ) or ((... ( ) ...))

(⟨atom⟩)

((... (<s-expr>) ...))

< atom >

Variables reflecting the results of the ATN Syntax Analysis are initialized appropriately.

FUNCTIONS USED: COND, NULL, SETQ, NOT, ATOM, EQUAL, and LENGTH

BOUND VARIABLES: None

FREE VARIABLES: State-Description, Network-Syntax-Error-Message, Network-Syntax-Analysis-Mssg-Code, Error-Exists-in-Network-Syntax, Network-Syntax-Error-Message2 to Network-Syntax-Error-Message4

ROUTINE: CHECK-NODE-NAME-IN-STATE-DESCRIPTION

OBJECTIVE: CHECK-NODE-NAME-IN-STATE-DESCRIPTION examines the syntax of an ATN node name.

SYNTAX ANALYSIS: The COND function constitutes three clauses which test possible illegal occurrences of s-expressions used to represent an ATN node name, namely, an empty and a non-empty list.

An atom is a valid syntactic representation of an ATN node name. If the ATN node name is a non-empty list, further testing is done to check possible occurrences of s-expressions within the list, namely, a nested or non-nested empty or non-empty list of length greater than or equal to one.

A syntactically legal ATN node name is:

< atom>

A syntactically illegal ATN node name

can be any of the following:

( ) or ((... ( ) ...))

(<atom>) or
((... (<atom>) ...))

(<s-expr> ... <s-expr>) or
((... (<s-expr> ... <s-expr>) ...))

Error Recovery occurs when the ATN node name is a nested or non-nested list containing an atom. The recovery process involves fetching the atom from within the list and substituting the old ATN node name with the new ATN node name. Likewise, the ATN state-description and the network are repaired.

Variables reflecting the results of the ATN Syntax Analysis are initialized appropriately.

FUNCTIONS USED: COND, NULL, SETQ, NOT, ATOM, EQUAL, LENGTH, SUBST, REMOVE-NESTING-IN-LIST

BOUND VARIABLES: Old-Newstate represents the ATN node name that will be replaced by the new ATN node name in the error recovery process.

Old-State-Description represents the ATN state-description that will be replaced by the new ATN state-description.

FREE VARIABLES: Newstate, Network-Syntax-Error-Message, Network-Syntax-Analysis-Mssg-Code, Error-Exists-in-Network-Syntax, Old-Newstate, Old-State-Description, Network-Syntax-Error-Message5 to Network-Syntax-Error-Message9

ROUTINE: LIST-DIFFERENCE

OBJECTIVE: LIST-DIFFERENCE takes two lists, namely, List1 and List2. The difference of the two lists, List1 and List2, is what remains of the list List1 after all elements that are also elements

of the list List2 are removed.

FUNCTIONS USED: NULL, MEMBER, CAR, CDR, LIST-DIFFERENCE, CONS

BOUND VARIABLES: List1 represents a list of s-expressions.

List2 represents a list of s-expressions.

FREE VARIABLES: None

ROUTINE: REPLACE-RULE-SYNTAX-SEGMENT-WITH-KEYWORD-IF

OBJECTIVE: REPLACE-RULE-SYNTAX-SEGMENT-WITH-KEYWORD-IF does the actual substitution of the ATN state-transition rule first syntactic segment with the atom IF.

FUNCTIONS USED: RETURN, SUBST

BOUND VARIABLES: None

FREE VARIABLES: If, Old-Rule-Syntax-Segment1, Old-Rule

ROUTINE: INSERT-KEYWORD-IF-INTO-NETWORK-TRAN-RULE

OBJECTIVE: INSERT-KEYWORD-IF-INTO-NETWORK-TRAN-RULE does the actual inclusion of the atom IF into the ATN state-transition rule.

FUNCTIONS USED: RETURN, CONS

BOUND VARIABLES: None

FREE VARIABLES: If, Old-Rule

ROUTINE: REPLACE-RULE-SYNTAX-SEGMENT-WITH-STATE-TRAN-SYMBOL

OBJECTIVE: REPLACE-RULE-SYNTAX-SEGMENT-WITH-STATE-TRAN-SYMBOL does the actual substitution of the ATN state-

transition rule third syntactic segment with the atom >>> .

FUNCTIONS USED: RETURN, SUBST

BOUND VARIABLES: None

FREE VARIABLES: State-Transition-Symbol, Rule-Syntax-Segment3, Old-Rule

ROUTINE: INSERT-STATE-TRAN-SYMBOL-INTO-NETWORK-TRAN-RULE

OBJECTIVE: INSERT-STATE-TRAN-SYMBOL-INTO-NETWORK-TRAN-RULE does the actual inclusion of the atom >>> into the ATN state-transition rule.

FUNCTIONS USED: SETQ, MEMBER, LIST-DIFFERENCE, CONS, RETURN, APPEND

BOUND VARIABLES: Sub-Rule represents a sub-part of the ATN state-transition rule.

Difference-of-Lists represents the difference of two lists.

FREE VARIABLES: Rule-Syntax-Segment3, Rule, List-Difference, State-Transition-Symbol

ROUTINE: REPLACE-RULE-SYNTAX-SEGMENT-WITH-TRAN-RULE-NEW-NODE

OBJECTIVE: REPLACE-RULE-SYNTAX-SEGMENT-WITH-TRAN-RULE-NEW-NODE does the actual substitution of the ATN state-transition rule fourth syntactic segment with the proper ATN state-transition rule new node.

FUNCTIONS USED: RETURN, SUBST

BOUND VARIABLES: None

FREE VARIABLES: Tran-Rule-New-Node, Rule-Syntax-Segment4, Old-Rule

ROUTINE: REPLACE-RULE-SYNTAX-SEGMENT-WITH-KEYWORD-AFTER

OBJECTIVE: REPLACE-RULE-SYNTAX-SEGMENT-WITH-KEYWORD-AFTER does the actual substitution of the ATN state-transition rule fifth syntactic segment with the atom AFTER.

FUNCTIONS USED: RETURN, SUBST

BOUND VARIABLES: None

FREE VARIABLES: After, Rule-Syntax-Segment5, Old-Rule

ROUTINE: REPAIR-STATE-DESCRIPTION-AND-NETWORK

OBJECTIVE: REPAIR-STATE-DESCRIPTION-AND-NETWORK repairs the ATN state-description and the network by substituting the old syntactic component with the new syntactic component.

FUNCTIONS USED: SETQ, SUBST

BOUND VARIABLES: None

FREE VARIABLES: State-Description, Rule, Old-Rule, Old-State-Description, Network

ROUTINE: REPAIR-VIA-KEYWORD-IF-REPLACEMENT

OBJECTIVE: REPAIR-VIA-KEYWORD-IF-REPLACEMENT repairs the ATN state-transition rule by replacing the invalid first syntactic segment with the legal atom IF. The actual repair work is performed by the routine REPLACE-RULE-SYNTAX-WITH-KEYWORD-IF. Since the ATN state-transition rule has been repaired, REPAIR-STATE-DESCRIPTION-AND-NETWORK modifies the ATN state-description and the network.

Variables reflecting the results of the modification are initialized appropriately.

FUNCTIONS USED: SETQ, REPLACE-RULE-SYNTAX-SEGMENT-WITH-KEYWORD-IF, REPAIR-STATE-DESCRIPTION-AND-NETWORK

BOUND VARIABLES: None

FREE VARIABLES: Rule, Network-Syntax-Error-Message, Network-Syntax-Analysis-Mssg-Code, Rule-Syntax-Error-Message2

ROUTINE: REPAIR-VIA-KEYWORD-IF-INSERTION

OBJECTIVE: REPAIR-VIA-KEYWORD-IF-INSERTION repairs the ATN state-transition rule by inserting the atom IF into the ATN state-transition rule. The actual repair work is performed by the routine INSERT-KEYWORD-IF-INTO-NETWORK-TRAN-RULE. Since the ATN state-transition rule has been repaired, REPAIR-STATE-DESCRIPTION-AND-NETWORK modifies the ATN state-description and the network.

Variables reflecting the results of the modification are initialized appropriately.

FUNCTIONS USED: SETQ, INSERT-KEYWORD-IF-INTO-NETWORK-TRAN-RULE, REPAIR-STATE-DESCRIPTION-AND-NETWORK

BOUND VARIABLES: None

FREE VARIABLES: Rule, Network-Syntax-Error-Message, Rule-Syntax-Error-Message3, Network-Syntax-Analysis-Mssg-Code

ROUTINE: REPAIR-VIA-TEST-REPLACEMENT

OBJECTIVE: REPAIR-VIA-TEST-REPLACEMENT repairs the ATN state-transition rule by replacing the invalid second syntactic segment with the legal atom T. The actual repair work utilizes the function SUBST. Since the ATN state-transition rule has been repaired, REPAIR-STATE-DESCRIPTION-AND-NETWORK modifies the ATN state-description and

the network.

Variables reflecting the results of the modification are initialized appropriately.

FUNCTIONS USED: SETQ, SUBST, REPAIR-STATE-DESCRIPTION-AND-NETWORK

BOUND VARIABLES: None

FREE VARIABLES: Rule, Rule-Syntax-Segment2, Old-Rule, Network-Syntax-Error-Message, Rule-Syntax-Error-Message20, Network-Syntax-Analysis-Mssg-Code

ROUTINE: REPAIR-VIA-STATE-TRAN-SYMBOL-REPLACEMENT

OBJECTIVE: REPAIR-VIA-STATE-TRAN-SYMBOL-REPLACEMENT repairs the ATN state-transition rule by replacing the invalid third syntactic segment with the legal atom >>> . The actual repair work is performed by the routine REPLACE-RULE-SYNTAX-SEGMENT-WITH-STATE-TRAN-SYMBOL. Since the ATN state-transition rule has been repaired, REPAIR-STATE-DESCRIPTION-AND-NETWORK modifies the ATN state-description and the network.

Variables reflecting the results of the modification are initialized appropriately.

FUNCTIONS USED: SETQ, REPLACE-RULE-SYNTAX-SEGMENT-WITH-STATE-TRAN-SYMBOL, REPAIR-STATE-DESCRIPTION-AND-NETWORK

BOUND VARIABLES: None

FREE VARIABLES: Rule, Network-Syntax-Error-Message, Rule-Syntax-Error-Message9, Network-Syntax-Analysis-Mssg-Code

ROUTINE: REPAIR-VIA-STATE-TRAN-SYMBOL-INSERTION

OBJECTIVE: REPAIR-VIA-STATE-TRAN-SYMBOL-INSERTION repairs the ATN state-transition rule by inserting the atom >>> into the ATN state-transition rule. The actual repair work is performed by the routine INSERT-STATE-TRAN-SYMBOL-INTO-NETWORK-TRAN-RULE. Since the ATN state-transition rule has been repaired, REPAIR-STATE-DESCRIPTION-AND-NETWORK modifies the ATN state-description and the network.

Variables reflecting the results of the modification are initialized appropriately.

FUNCTIONS USED: SETQ, INSERT-STATE-TRAN-SYMBOL-INTO-NETWORK-TRAN-RULE, REPAIR-STATE-DESCRIPTION-AND-NETWORK

BOUND VARIABLES: None

FREE VARIABLES: Rule, Network-Syntax-Error-Message, Rule-Syntax-Error-Message10, Network-Syntax-Analysis-Mssg-Code

ROUTINE: REPAIR-VIA-TRAN-RULE-NEW-NODE-REPLACEMENT

OBJECTIVE: REPAIR-VIA-TRAN-RULE-NEW-NODE-REPLACEMENT repairs the ATN state-transition rule by replacing the invalid fourth syntactic segment with the proper ATN state-transition rule new node. The actual repair work is performed by the routine REPLACE-RULE-SYNTAX-SEGMENT-WITH-TRAN-RULE-NEW-NODE. Since the ATN state-transition rule has been repaired, REPAIR-STATE-DESCRIPTION-AND-NETWORK modifies the ATN state-description and the network.

Variables reflecting the results of the modification are initialized appropriately.

FUNCTIONS USED: SETQ, REPLACE-RULE-SYNTAX-SEGMENT-WITH-TRAN-RULE-NEW-NODE, REPAIR-STATE-DESCRIPTION-AND-NETWORK

BOUND VARIABLES: None

FREE VARIABLES: Rule, Network-Syntax-Error-Message, Rule-Syntax-Error-Message14, Network-Syntax-Analysis-Mssg-Code

ROUTINE: REPAIR-VIA-KEYWORD-AFTER-REPLACEMENT

OBJECTIVE: REPAIR-VIA-KEYWORD-AFTER-REPLACEMENT repairs the ATN state-transition rule by replacing the invalid fifth syntactic segment with the legal atom AFTER. The actual repair work is performed by the routine REPLACE-RULE-SYNTAX-SEGMENT-WITH-KEYWORD-AFTER. Since the ATN state-transition rule has been repaired, REPAIR-STATE-DESCRIPTION-AND-NETWORK modifies the ATN state-description and the network.

Variables reflecting the results of the modification are initialized appropriately.

FUNCTIONS USED: SETQ, REPLACE-RULE-SYNTAX-SEGMENT-WITH-KEYWORD-AFTER, REPAIR-STATE-DESCRIPTION-AND-NETWORK

BOUND VARIABLES: None

FREE VARIABLES: Rule, Network-Syntax-Error-Message, Rule-Syntax-Error-Message19, Network-Syntax-Analysis-Mssg-Code

ROUTINE: PRINT-THE-REPAIRED-NETWORK

OBJECTIVE: PRINT-THE-REPAIRED-NETWORK outputs the repaired Augmented Transition Network (ATN).

FUNCTIONS USED: PRINT, SP

BOUND VARIABLES: None

FREE VARIABLES: Network

ROUTINE: EXAMINE-KEYWORD-IF-IN-LIST

OBJECTIVE: EXAMINE-KEYWORD-IF-IN-LIST examines the first syntactic segment or component of the ATN state-transition rule to determine if the atom IF is in the list.

SYNTAX ANALYSIS: The COND function constitutes three clauses which test other possible illegal occurrences of s-expressions used to represent the first syntactic segment of the ATN state-transition rule.

EXAMINE-KEYWORD-IF-IN-LIST identifies the following illegal s-expressions:

((... ( ) ...))

(<atom>) or
((... (<atom>) ...))

(<s-expr> ... <s-expr>) or
((... (<s-expr> ... <s-expr>) ...))

The only candidate for Error Recovery is:

(<atom>) or
((... (<atom>) ...))

Error Recovery involves testing < atom >. Insertion of the atom IF is made before (<atom>) or ((... (<atom>) ...)) if the first segment is the atom T. Otherwise, Replacement of (<atom>) or ((... (<atom>) ...)) by the atom IF takes place. Likewise, the ATN state-description and the network are repaired.

Variables reflecting the results of the ATN Syntax-Analysis are initialized appropriately.

FUNCTIONS USED: SETQ, REMOVE-NESTING-IN-LIST, COND, NULL, RETURN, EQUAL, LENGTH, CAR, REPAIR-VIA-KEYWORD-IF-INSERTION, PRINT-THE-REPAIRED-NETWORK

BOUND VARIABLE: Temp-Rule-Syntax-Segment1 represents the temporary first syntactic segment.

FREE VARIABLES: Rule-Syntax-Segmentl, Network-Syntax-Error-Message, Rule-Syntax-Error-Message27, Network-Syntax-Analysis-Mssg-Code, Rule-Syntax-Error-Message28

ROUTINE: CHECK-KEYWORD-IF-IN-TRANSITION-RULE

OBJECTIVE: CHECK-KEYWORD-IF-IN-TRANSITION-RULE examines the first syntactic segment or component of the ATN state-transition rule.

SYNTAX ANALYSIS: The COND function constitutes three clauses which test possible illegal occurrences of s-expressions used to represent the first syntactic segment of an ATN state-transition rule, namely, an empty list, a non-empty list, or an atom that is different from IF.

The atom IF is the only valid syntactic representation of the first syntactic component or segment of the ATN state-transition rule. If the first segment is a non-empty list, then further testing is done to check the possible occurrence of the atom IF which might be nested within the list. Such a check is made in the routine EXAMINE-KEYWORD-IF-IN-LIST which uses error recovery routines.

The routine CHECK-KEYWORD-IF-IN-TRANSITION-RULE directly identifies a single syntax error:

( )

The reason is because EXAMINE-KEYWORD-IF-IN-LIST notes the other syntax errors.

Error Recovery occurs when the first segment is any atom. Insertion of the atom IF is made before the atom in question only if the first segment is the atom T. Otherwise, Replacement of the atom by the atom IF takes place. Likewise, the ATN state-description and the network are repaired.

Variables reflecting the results of the ATN Syntax Analysis are initialized appropriately.

FUNCTIONS USED: SETQ, CAR, COND, NULL, RETURN, NOT, ATOM, EQUAL, REPAIR-VIA-KEYWORD-IF-INSERTION, PRINT-THE-REPAIRED-NETWORK, REPAIR-VIA-KEYWORD-IF-REPLACEMENT

BOUND VARIABLES: Rule-Syntax-Segment1 represents the first syntactic segment of the ATN state-transition rule.

Old-Rule-Syntax-Segment1 represents the temporary first syntactic segment.

If represents the only valid syntactic representation of the first syntactic segment.

Old-Rule represents the ATN state-transition rule to be repaired.

Old-State-Description represents the ATN state-description to be repaired.

FREE VARIABLES: Rule, State-Description, Network-Syntax-Error-Message, Rule-Syntax-Error-Message26, Network-Syntax-Analysis-Mssg-Code

ROUTINE: CHECK-TEST-IN-TRANSITION-RULE

OBJECTIVE: CHECK-TEST-IN-TRANSITION-RULE examines the second syntactic component or segment of the ATN state-transition rule.

SYNTAX ANALYSIS: The COND function constitutes two clauses to test the existence of a second syntactic component or segment and to check possible illegal occurrences of s-expressions used to represent the second syntactic segment of an ATN state-transition rule.

CHECK-TEST-IN-TRANSITION-RULE identifies the following illegal s-expressions:

( ) or ((... ( ) ...))

(<atom>) or ((... (<atom>) ...))

< atom> other than T

The only candidate for Error Recovery is (<atom>) or ((... (<atom>) ...)). Error Recovery involves testing < atom> . If < atom> is the boolean atom T, then < atom> replaces (<atom>) or ((... (<atom>) ...)). Otherwise, an error will be displayed. Such a replacement takes place in the routine REPAIR-VIA-TEST-REPLACEMENT, thus repairing the state-transition rule. Likewise, the ATN state-description and the network are repaired.

The only valid syntactic representations of the fourth syntactic segment are:

< atom> equal to T

(<s-expr> ... <s-expr>) or
((... (<s-expr> ... <s-expr>) ...))

Variables reflecting the results of the ATN Syntax Analysis are initialized appropriately.

FUNCTIONS USED: COND, NULL, CDR, SETQ, RETURN, NOT, REMOVE-NESTING-IN-LIST, AND, LENGTH, REPAIR-VIA-TEST-REPLACEMENT, PRINT-THE-REPAIRED-NETWORK

BOUND VARIABLES: Rule-Syntax-Segment2 represents the second syntactic segment of the ATN state-transition rule.

Temp-Rule-Syntax-Segment2 represents the temporary second syntactic segment.

Old-Rule represents the ATN state-transition rule to be repaired.

Old-State-Description represents the ATN state-description to be repaired.

FREE VARIABLES: Rule, Network-Syntax-Error-Message, Rule-Syntax-Error-Message4 to Rule-Syntax-Error-Message7, Network-Syntax-Analysis-Mssg-Code

ROUTINE: EXAMINE-STATE-TRAN-SYMBOL-IN-LIST

OBJECTIVE: EXAMINE-STATE-TRAN-SYMBOL-IN-LIST examines the third syntactic component or segment of the state-transition rule to determine if the atom is in the list.

SYNTAX ANALYSIS: The COND function constitutes three clauses which test other possible illegal occurrences of s-expressions used to represent the third syntactic segment of the ATN state-transition rule.

EXAMINE-STATE-TRAN-SYMBOL-IN-LIST identifies the following illegal s-expressions:

((... ( ) ...))

(<atom>) or
((... (<atom>) ...))

(<s-expr> ... <s-expr>) or
((... (<s-expr> ... <s-expr>) ...))

The only candidate for Error Recovery is (<atom>) or ((... (<atom>) ...)). Error Recovery involves testing < atom >. If < atom > is either an ATN node name, an atom WIN, or an atom LOSE, then the state-transition symbol >>> is inserted before (<atom>) or ((... (<atom>) ...)). Otherwise, replacement of (<atom>) or ((... (<atom>) ...)) by the state-transition symbol >>> takes place. Likewise, the ATN state-description and the network are repaired.

Variables reflecting the results of the ATN Syntax Analysis are initialized appropriately.

FUNCTIONS USED: SETQ, REMOVE-NESTING-IN-LIST, COND, NULL, RETURN, EQUAL, LENGTH, OR, ASSOC, REPAIR-VIA-STATE-TRAN-SYMBOL-INSERTION, PRINT-THE-REPAIRED-NETWORK, REPAIR-VIA-STATE-TRAN-SYMBOL-REPLACEMENT

BOUND VARIABLES: Temp-Rule-Syntax-Segment3 represents the temporary third syntactic segment.

FREE VARIABLES: Rule-Syntax-Segment3, Network-Syntax-Error-Message, Rule-Syntax-Error-Message30, Network-Syntax-Analysis-Mssg-Code, Network, Win, Lose

ROUTINE: EXAMINE-ATOM-IF-STATE-TRAN-SYMBOL

OBJECTIVE: EXAMINE-ATOM-IF-STATE-TRAN-SYMBOL examines the third syntactic segment or component of the state-transition rule to determine if the atom in question is the state-transition symbol >>> .

SYNTAX ANALYSIS: The COND function constitutes two clauses which test if the third syntactic segment is a state-transition symbol >>> and to check whether to make an insertion or replacement in the Error Recovery process.

If the third syntactic segment is represented by < atom> is not a state-transition symbol >>> , then further testing is made. If <atom> is either an ATN node name, an atom WIN, or an atom LOSE, then the state-transition symbol >>> is inserted before <atom> . Otherwise, replacement of <atom> by the state-transition symbol >>> takes place. Likewise, the ATN state-description and the network are repaired.

Variables reflecting the results of the ATN Syntax Analysis are initialized appropriately.

FUNCTIONS USED: COND, EQUAL, RETURN, OR, ASSOC, REPAIR-VIA-STATE-TRAN-SYMBOL-INSERTION, PRINT-THE-REPAIRED-NETWORK, REPAIR-VIA-STATE-TRAN-SYMBOL-REPLACEMENT

BOUND VARIABLES: None

FREE VARIABLES: Rule-Syntax-Segment3, State-Transition-Symbol, Network, Win, Lose

ROUTINE: CHECK-STATE-TRANSITION-SYMBOL-IN-TRAN-RULE

OBJECTIVE: CHECK-STATE-TRANSITION-SYMBOL-IN-TRAN-RULE examines the third syntactic component or segment of the ATN state-transition rule.

SYNTAX ANALYSIS: The COND function constitutes two clauses to test the existence of a third syntactic component and to check possible illegal occurrences of s-expressions used to represent the third syntactic segment of an ATN state-transition rule, namely, an empty list, a non-empty list, or an atom other than >>> .

The atom >>> is the only valid syntactic representation of the third syntactic component or segment of the ATN state-transition rule. If the third segment is a non-empty list, then further testing is done to check the possible occurrence of the atom >>> which might be nested within the list. Such a check is made in the routine EXAMINE-STATE-TRAN-SYMBOL-IN-LIST which uses Error Recovery routines.

Besides checking for the existence of a third syntactic segment, the routine CHECK-STATE-TRANSITION-SYMBOL-IN-TRAN-RULE directly identifies a single syntax error:

( )

The reason is because EXAMINE-STATE-TRAN-SYMBOL-IN-LIST and EXAMINE-ATOM-IF-STATE-TRAN-SYMBOL note the other syntax errors and do Error Recovery processes.

Variables reflecting the results of the ATN Syntax Analysis are initialized appropriately.

FUNCTIONS USED: COND, NULL, CDDR, SETQ, RETURN, NOT, EXAMINE-STATE-TRAN-SYMBOL-IN- LIST, EXAMINE-ATOM-IF-STATE-TRAN-SYMBOL.

BOUND VARIABLES: Rule-Syntax-Segment3 represents the third syntactic segment of the ATN state-transition rule.

Old-Rule-Syntax-Segment3 represents the temporary third syntactic segment.

State-Transition-Symbol represents the only valid syntactic representation of the third syntactic segment. The State-Transition-Symbol is signified by the atom >>> .

Win represents a state-transition rule new node which has a constant value WIN.

Lose represents a state-transition rule new node which has a constant value LOSE.

Old-Rule represents the ATN state-transition rule to be repaired.

Old-State-Description represents the ATN state-description to be repaired.

FREE VARIABLES: Rule, Network-Syntax-Error-Message, Rule-Syntax-Error-Message8, Network-Syntax-Analysis-Mssg-Code, Rule-Syntax-Error-Message29

ROUTINE: EXAMINE-NEW-NODE-IN-LIST

OBJECTIVE: EXAMINE-NEW-NODE-IN-LIST examines the fourth syntactic component or segment of the ATN state-transition rule to determine if the ATN state-transition rule new node is in the list.

SYNTAX ANALYSIS: The COND function constitutes two clauses to test other possible illegal occurrences of s-expressions used to represent the fourth syntactic segment of the ATN state-transition rule.

EXAMINE-NEW-NODE-IN-LIST identifies the following illegal s-expressions:

((... ( ) ...))

(<atom>) or ((... (<atom>) ...))

(<s-expr> ... <s-expr>) or
((... (<s-expr> ... <s-expr>) ...))

The only candidate for Error Recovery is:

(<atom>) or ((... (<atom>) ...))

Error Recovery involves testing < atom > . If <atom> is either an ATN node name, the atom WIN, or the atom LOSE, then <atom> replaces (<atom>) or ((...(<atom>)...)). Otherwise, an error will be displayed. Such a replacement takes place in the routine REPAIR-VIA-TRAN-RULE-NEW-NODE-REPLACEMENT, thus repairing the state-transition rule. Likewise, the ATN state-description and the network are repaired.

Variables reflecting the results of the ATN Syntax Analysis are initialized appropriately.

FUNCTIONS USED: COND, EQUAL, LENGTH, SETQ, REMOVE-NESTING-IN-LIST, NULL, RETURN, CAR, OR, ASSOC, REPAIR-VIA-TRAN-RULE-NEW-NODE-REPLACEMENT, PRINT-THE-REPAIRED-NETWORK

BOUND VARIABLES: Temp-Rule-Syntax-Segment4 represents the temporary fourth syntactic_segment.

Tran-Rule-New-Node represents the state-transition rule new node.

FREE VARIABLES: Rule-Syntax-Segment4, Network-Syntax-Error-Message, Rule-Syntax-Error-Message13, Network-Syntax-Analysis-Mssg-Code, Network, Win, Lose, Rule-Syntax-Error-Message15 to Rule-Syntax-Error-Message17

ROUTINE: CHECK-NEW-NODE-IN-TRANSITION-RULE

OBJECTIVE: CHECK-NEW-NODE-IN-TRANSITION-RULE examines the the fourth syntactic component or segment of the ATN state-transition rule.

SYNTAX ANALYSIS: The COND function constitutes two clauses to test the existence of a fourth syntactic component and to check possible illegal occurrences of s-expressions used to represent the fourth syntactic segment of an

ATN state-transition rule, namely, an empty list, a non-empty list, or an atom that is not an ATN node name. An atom that is an ATN node name is the only valid syntactic representation of the fourth syntactic component or segment of the ATN state-transition rule. If the fourth segment is a non-empty list, then the routine EXAMINE-NEW-NODE-IN-LIST is invoked to check the atom within the list. If the fourth segment is an atom, then a check is made to determine if the atom is an ATN node name.

Variables reflecting the results of the ATN Syntax Analysis are initialized appropriately.

FUNCTIONS USED: COND, NULL, CDDDR, SETQ, RETURN, CADDDR, NOT, ATOM, EXAMINE-NEW-NODE-IN-LIST, OR, ASSOC, EQUAL

BOUND VARIABLES: Rule-Syntax-Segment4 represents the fourth syntactic segment of the ATN state-transition rule.

Old-Rule-Syntax-Segment4 represents the temporary fourth syntactic segment.

Win represents a state-transition rule new node which has a constant value WIN.

Lose represents a state-transition rule new node which has a constant value LOSE.

Old-Rule represents the ATN state-transition rule to be repaired.

Old-State-Description represents the ATN state-description to be repaired.

FREE VARIABLES: Rule, Network-Syntax-Error-Message, Rule-Syntax-Error-Message1, Network-Syntax-Analysis-Mssg-Code, State-Description, Rule-Syntax-Error-Message12, Rule-Syntax-Segment4, Rule-Syntax-Error-Message18

ROUTINE: EXAMINE-KEYWORD-AFTER-IN-LIST

OBJECTIVE: EXAMINE-KEYWORD-AFTER-IN-LIST examines the fifth syntactic component or segment of the state-transition rule to determine if the atom AFTER is in the list.

SYNTAX ANALYSIS: The COND function constitutes three clauses which test other possible illegal occurrences of s-expressions used to represent the fifth syntactic segment of the ATN state-transition rule.

EXAMINE-KEYWORD-AFTER-IN-LIST identifies the following illegal s-expressions:

((... ( ) ...))

(<atom>) or ((... (<atom>) ...))

(<s-expr> ... <s-expr>) or
((... (<s-expr> ... <s-expr>) ...))

The only candidate for Error Recovery is:

(<atom>) or ((... (<atom>) ...)).

Error Recovery involves testing < atom >. If <atom> is AFTER, then the atom AFTER replaces (<atom>) or ((... (<atom>)...)). Otherwise, an error will be displayed. The ATN state-description and the network are repaired.

Variables reflecting the results of the ATN Syntax Analysis are initialized appropriately.

FUNCTIONS USED: SETQ, REMOVE-NESTING-IN-LIST, COND, NULL, RETURN, EQUAL, LENGTH, REPAIR-VIA-KEYWORD-AFTER-REPLACEMENT, PRINT-THE-REPAIRED-NETWORK

BOUND VARIABLES: Temp-Rule-Syntax-Segment5 represents the temporary fifth syntactic segment.

FREE VARIABLES: Rule-Syntax-Segment5, Network-Syntax-Error-Message, Network-Syntax-Analysis-Mssg-Code, Rule-Syntax-Error-Message33, Rule-Syntax-Error-Message34

ROUTINE: CHECK-KEYWORD-AFTER-IN-TRANSITION-RULE

OBJECTIVE: CHECK-KEYWORD-AFTER-IN-TRANSITION-RULE examines the fifth syntactic component or segment of the ATN state-transition rule.

SYNTAX ANALYSIS: The COND function constitutes three clauses to check possible illegal occurrences of s-expressions used to represent the fifth syntactic segment of an ATN state-transition rule, namely, an empty list, a non-empty list, or an atom other than AFTER.

The atom AFTER is the only valid syntactic representation of the fifth syntactic component or segment of the ATN state-transition rule. If the fifth segment is a non-empty list, then further testing is done to check the possible occurrence of the atom AFTER which might be nested within the list. Such a check is made in the routine EXAMINE-KEYWORD-AFTER-IN-LIST which also does Error Recovery. If the fifth segment is an atom other than the atom AFTER, then replacement of <atom> by the atom AFTER takes place. Such an Error Recovery process is accomplished in the routine REPAIR-VIA-KEYWORD-AFTER-REPLACEMENT.

Variables reflecting the results of the ATN Syntax Analysis are initialized appropriately.

FUNCTIONS USED: CADDDDR, COND, NULL, SETQ, RETURN, NOT, ATOM, EXAMINE-KEYWORD-AFTER-IN-LIST, REPAIR-VIA-KEYWORD-AFTER-REPLACEMENT, PRINT-THE-REPAIRED-NETWORK

BOUND VARIABLES: Rule-Syntax-Segment5 represents the fifth syntactic segment of the ATN state-transition rule.

Old-Rule-Syntax-Segment5 represents the temporary fifth syntactic segment.

After represents the only valid syntactic representation of the fifth syntactic segment.

Old-Rule represents the temporary ATN state-transition rule.

Old-State-Description represents the temporary ATN state-description.

FREE VARIABLES: Rule-Syntax-Segment5, Rule, State-Description, Network-Syntax-Error-Message, Rule-Syntax-Error-Message32, Network-Syntax-Analysis-Mssg-Code

ROUTINE: CHECK-SIDE-EFFECTS-SYNTAX

OBJECTIVE: CHECK-SIDE-EFFECTS-SYNTAX examines the sixth syntactic component or segment of the ATN state-transition rule.

SYNTAX ANALYSIS: CHECK-SIDE-EFFECTS-SYNTAX is an iterative routine that fetches a side effect from the ATN state-transition rule and checks its syntax. The COND function constitutes three clauses to check possible illegal occurrences of s-expressions used to represent the sixth syntactic segment of an ATN state-transition rule.

CHECK-SIDE-EFFECTS-SYNTAX identifies the following illegal s-expressions:

( ) or ((... ( ) ...))

< atom >

The only valid syntax representation of the sixth syntax segment are:

(<s-expr>) or
(<s-expr> <s-expr>) or
((... (<s-expr> ... <s-expr>) ...))

Variables reflecting the results of the ATN Syntax Analysis are initialized appropriately.

FUNCTIONS USED: SETQ, CDDDDDR, CAR, COND, NULL, RETURN, NOT, ATOM, REMOVE-NESTING-IN-LIST, GO

BOUND VARIABLES: Rule-Syntax-Segment6 represents the sixth syntactic segment of the ATN state-transition rule.

List-of-Side-Effects represents a collection of all the side effects of the ATN state-transition rule.

No-Errors-Exist-In-List-of-Side-Effects represents a boolean variable to signify that a side effect has an improper syntax.

FREE VARIABLES: Rule, Network-Syntax-Error-Message, Rule-Syntax-Error-Message22 to Rule-Syntax-Error-Message24, Network-Syntax-Analysis-Mssg-Code

ROUTINE: CHECK-RULE-SYNTAX-SEGMENTS

OBJECTIVE: CHECK-RULE-SYNTAX-SEGMENTS examines the syntax of an ATN state-transition rule.

SYNTAX ANALYSIS: Each syntactic component of the ATN state-transition rule is examined in sequence. Thus, the ATN state-transition rule keyword IF, the Test, the State-Transition Symbol, and the State-Transition Rule New Node are checked by calling the proper syntax checking routines. If side effects do occur, the ATN state-transition rule keyword AFTER and its accompanying side effects are syntactically examined. If the keyword AFTER exists without any side effect, then an appropriate error message will be displayed.

FUNCTIONS USED: COND, AND, NOT, NULL, CDDDDR, CDDDDDR, CHECK-KEYWORD-IF-IN-TRANSITION-RULE, CHECK-TEST-IN-TRANSITION-RULE, CHECK-STATE-TRANSITION-SYMBOL-IN-TRAN-RULE, CHECK-NEW-NODE-IN-TRANSITION-RULE, CHECK-KEYWORD-AFTER-IN-TRANSITION-RULE, CHECK-SIDE-EFFECTS-SYNTAX

BOUND VARIABLES: None

FREE VARIABLES: Rule, Network-Syntax-Error-Message, Rule-Syntax-Error-Message21, Network-Syntax-Analysis-Mssg-Code

ROUTINE: CHECK-RULE-SYNTAX

OBJECTIVE: CHECK-RULE-SYNTAX examines the syntax of an ATN state-transition rule.

SYNTAX ANALYSIS: The COND function constitutes three clauses which test possible illegal occurrences of s-expressions used to represent an ATN state-transition rule, namely, an empty list and an atom. If the ATN state-transition rule is a list, then each syntactic component or segment is examined by calling the routine, CHECK-RULE-SYNTAX-SEGMENTS.

A syntactically legal ATN state-transition rule can be any of the following:

(IF <test> >>> <new node>) or

(IF <test> >>> <new node> AFTER <side effects>)

Variables reflecting the results of the ATN Syntax Analysis are initialized appropriately.

FUNCTIONS USED: COND, NULL, SETQ, NOT, ATOM, CHECK-RULE-SYNTAX-SEGMENTS

BOUND VARIABLES: None

FREE VARIABLES: Rule, Network-Syntax-Error-Message, Network-Syntax-Analysis-Mssg-Code, Error-Exists-in-Network-Syntax, Rule-Syntax-Error-Message1, Rule-Syntax-Error-Message25

ROUTINE: CHECK-NETWORK-SYNTAX

OBJECTIVE: CHECK-NETWORK-SYNTAX is an iterative routine that examines the syntax of an Augmented Transition Network (ATN).

SYNTAX ANALYSIS: A test is initially made to determine if the entire ATN is missing. If an ATN is existing, then further testing is done to check each syntactic segment of the ATN, namely, the State-Description, the Node Name, and the State-Transition Rule. ATN syntactic component fetching and testing are done iteratively.

Variables reflecting the results of the ATN Syntax Analysis are initialized appropriately.

FUNCTIONS USED: COND, NULL, SETQ, RETURN, CHECK-STATE-DESCRIPTION-SYNTAX, CHECK-NODE-NAME-IN-STATE-DESCRIPTION, CHECK-RULE-SYNTAX, GO

BOUND VARIABLES: List-of-State-Descriptions represents a network.

List-of-Rules represents a state-description minus the network node name.

FREE VARIABLES: Network, Network-Syntax-Error-Message, Network-Syntax-Error-Messagel, Network-Syntax-Analysis-Mssg-Code, Error-Exists-in-Network-Syntax, State-Description, Newstate, Rule

ROUTINE: INITIALIZE-NETWORK-SYNTAX-ERROR-MESSAGES

OBJECTIVE: INITIALIZE-NETWORK-SYNTAX-ERROR-MESSAGES assigns each possible Augmented Transition Network (ATN) syntax error message to its corresponding Augmented Transition Network syntax error message code.

FUNCTION USED: SETQ

BOUND VARIABLES: None

FREE VARIABLES: Network-Syntax-Error-Message1 to Network-Syntax-Error-Message9

ROUTINE: INITIALIZE-RULE-SYNTAX-ERROR-MESSAGES

OBJECTIVE: INITIALIZE-RULE-SYNTAX-ERROR-MESSAGES assigns each possible Augmented Transition Network (ATN) rule syntax error message to its corresponding Augmented Transition Network rule syntax error message code.

FUNCTION USED: SETQ

BOUND VARIABLES: None

FREE VARIABLES: Rule-Syntax-Error-Message1 to Rule-Syntax-Error-Message34

ROUTINE: INITIALIZE-NETWORK-SYNTAX-SEGMENTS

OBJECTIVE: INITIALIZE-NETWORK-SYNTAX-SEGMENTS assigns dummy values to each variable which represents each syntactic segment of the Augmented Transition Network (ATN).

FUNCTION USED: SETQ

BOUND VARIABLES: None

FREE VARIABLES: State-Description, Newstate, Rule

ROUTINE: GETF

OBJECTIVE: GETF is a routine that fetches the property value of the property name FEATURES. The classification type of a word appears on the property list of the word under the property name FEATURES.

FUNCTION USED: GET

BOUND VARIABLE: The-Features represents the property value of the property name FEATURES.

FREE VARIABLES: None

ROUTINE: INTERSECTION

OBJECTIVE: INTERSECTION accepts two lists and determines a list containing only the elements that are in both of the two lists.

FUNCTIONS USED: COND, NULL, MEMBER, CAR, CONS, INTERSECTION, CDR

BOUND VARIABLES: List1 represents a list of s-expressions.

List2 represents a list of s-expressions.

FREE VARIABLES: None

ROUTINE: TESTF

OBJECTIVE: TESTF returns T only if a given node has all of the given features. TESTF can handle either a single feature or a list of features.

FUNCTIONS USED: COND, NULL, ATOM, SETQ, LIST, EQUAL, LENGTH, INTERSECTION, GETF

BOUND VARIABLES: Node represents a word whose features are to be examined.

Features represents the characteristics of a given word.

FREE VARIABLES: None

ROUTINE: PRINT-THE-PARENT-AND-THE-CHILDREN

OBJECTIVE: PRINT-THE-PARENT-AND-THE-CHILDREN outputs the property value of the child and the property value of the parent.

FUNCTIONS USED: TERPRI, PRINØ, XTAB, PRINT, GET

BOUND VARIABLES: None

FREE VARIABLES: The-Child, The-Parent

ROUTINE: ATTACH

OBJECTIVE: ATTACH is a routine that connects nodes by way of placing appropriate values on a property list.

FUNCTIONS USED: PUTPROP, APPEND, GET, LIST, SETQ, PRINT-THE-PARENT-AND-THE-CHILDREN

BOUND VARIABLES: The-Parent represents the atom name of the property value The-Children and the property name Children.

The-Child represents the atom name of the property value The-Parent and the property name Parent.

The-Children represents a list of property values.

FREE VARIABLES: None

ROUTINE: SELECT

OBJECTIVE: SELECT is a recursive routine that takes two lists as arguments and returns the first thing in the in the first list that is also in the second.

FUNCTIONS USED: COND, NULL, MEMBER, CAR, SELECT

BOUND VARIABLES: S-exprl represents a symbolic expression.

S-expr2 represents a symbolic expression.

FREE VARIABLES: None

ROUTINE: GENNAME

OBJECTIVE: GENNAME is a routine that generates a unique new atom each time it is evaluated. When the value of the argument of GENNAME is, say, Parse-Noun-Group, then the new atom is of the form Parse-Noun-Group-n where the number n increases by one each time the function is used with Parse-Noun-Group as the value of its argument. GENNAME is not a standard LISP function, however, it is defined using the LISP functions IMPLODE, EXPLODE, and GENSYM.

FUNCTIONS USED: IMPLODE, APPEND, EXPLODE, CDDDDR, GENSYM

BOUND VARIABLE: Name represents the atom to be connected into a unique new atom.

FREE VARIABLES: None

ROUTINE: PRINT-THE-REGISTER-AND-THE-VALUE-OF-THIS-NODE

OBJECTIVE: PRINT-THE-REGISTER-AND-THE-VALUE-OF-THIS-NODE outputs the register name and its corresponding content with respect to an Augmented Transition Network node.

FUNCTIONS USED: PRINØ, XTAB, PRINT

BOUND VARIABLES: None

FREE VARIABLES: Register, Value

ROUTINE: SETR

OBJECTIVE: SETR is a routine that places the value of a node into a register. It is accomplished by placing the property value of an atom name into a property name. SETR uses This-Node as a free variable.

FUNCTION USED: PUTPROP, PRINT-THE-REGISTER-AND-THE-VALUE-OF-THIS-NODE

BOUND VARIABLE: Register represents a property name.

Value represents a property value.

FREE VARIABLE: This-Node

ROUTINE: GETR

OBJECTIVE: GETR is a routine that retrieves the value of a node from a register. It is accomplished by fetching the property value from its property name.

FUNCTION USED: GET

BOUND VARIABLE: Register represents a property name

FREE VARIABLE: This-Node

ROUTINE: ADDR

OBJECTIVE: ADDR is a routine that uses SETR and GETR to add a new element to a register.

FUNCTIONS USED: SETR, CONS, GETR

BOUND VARIABLES: Register represents a property name.

Value represents a property value.

FREE VARIABLES: None

ROUTINE: INTERPRET

OBJECTIVE: INTERPRET is an iterative routine that accepts a network, fetches a state-description, and tests a state-transition rule. It works by maintaining the proper values for certain variables, e.g. Parent-Node, Features, This-Node, and Hold. More importantly, INTERPRET will also have Network, an atom whose value is an augmented transition network description.

FUNCTIONS USED: SETQ, COND, EQUAL, GO, ASSOC, CDR, CAR, EVAL, CADR, CDDDDR, MAPCAR, LAMBDA, NOT, TESTF, ATTACH, RETURN

BOUND VARIABLES: Network represents the ATN description.

Parent-Node will represent the property value of the property name Parent upon invoking the routine ATTACH.

The classification type of a word appears on the property list of the word under the property name Features.

Newstate represents the name of some node in the ATN description.

Oldstate represents the most recent value of Newstate.

State-description represents a list which contains an ATN node name and one or more state-transition rules. It specifies what courses to take in the process of driving a path through the network.

Rule represents a description of what must be done when an ATN arc has been taken.

Hold temporarily holds a list of words remaining to be analyzed.

This-Node represents the word under inspection.

FREE VARIABLES: Remaining-Words, Current-Word

ROUTINE: PRINT-THE-NETWORK-SYNTAX-ERRORS

OBJECTIVE: PRINT-THE-NETWORK-SYNTAX-ERRORS outputs the state-description, newstate, rule, the augmented transition network syntax error message number, and the augmented transition network syntax error message. The possible syntax error messages are Network-Syntax-Error-Message1 to Network-Syntax-Error-Message9 and Rule-Syntax-Error-Message1 to Rule-Syntax-Error-Message34.

FUNCTIONS USED: PRINT, TERPRI

BOUND VARIABLES: None

FREE VARIABLES: State-Description, Newstate, Rule, Network-Syntax-Analysis-Mssg-Code, and Network-Syntax-Error-Message

ROUTINE: RECORD

OBJECTIVE: RECORD is a FEXPR routine that creates function definitions such as Parse-Word, Parse-Noun-Group, and Parse-Clause.

FUNCTIONS USED: SETQ, INITIALIZE-NETWORK-SYNTAX-SEGMENTS, PUTPROP, PRINØ, XTAB, PRINT, TERPRI, SP, CHECK-NETWORK-SYNTAX, COND, PRINT-THE-NETWORK-SYNTAX-ERRORS, CONS, EVAL, SUBST, CAR

BOUND VARIABLE: The-Aug-Tran-Network represents the augmented transition network (ATN) description.

FREE VARIABLES: Network-Syntax-Error-Message, Error-Exists-in-Network-Syntax, Network-Syntax-Analysis-Mssg-Code, The-Network-Title, Network

ROUTINE: RECORD-PARSE-WORD

OBJECTIVE: RECORD-PARSE-WORD contains the Augmented Transition Network (ATN) description to analyze an English word. It invokes the FEXPR routine RECORD to create an instance of a function definition called PARSE-WORD.

FUNCTION USED: RECORD

BOUND VARIABLES: None

FREE VARIABLES: None

ROUTINE: RECORD-PARSE-NOUN-GROUP

OBJECTIVE: RECORD-PARSE-NOUN-GROUP contains the Augmented Transition Network (ATN) description to analyze an English Noun Group. It invokes the FEXPR routine RECORD to create an instance of a function definition called PARSE-NOUN-GROUP.

FUNCTION USED: RECORD

BOUND VARIABLES: None

FREE VARIABLES: None

ROUTINE: RECORD-PARSE-CLAUSE

OBJECTIVE: RECORD-PARSE-CLAUSE contains the Augmented Transition Network (ATN) description to analyze an English Clause. It invokes the FEXPR routine RECORD to create an instance of a function definition called PARSE-CLAUSE.

FUNCTION USED: RECORD

BOUND VARIABLES: None

FREE VARIABLES: None

ROUTINE: IDENTIFY

OBJECTIVE: IDENTIFY is the main routine called by the user to execute the ATN Interpreter and the ATN Syntax Analyzer. It contains a dictionary where English words are listed alphabetically and each property of each word is defined. The routine IDENTIFY states the English clause to be parsed by the ATN Interpreter.

FUNCTIONS USED: INITIALIZE-NETWORK-SYNTAX-ERROR-MESSAGES, INITIALIZE-RULE-SYNTAX-ERROR-MESSAGES, SETQ, DEFPROP, RECORD-PARSE-WORD, RECORD-PARSE-NOUN-GROUP, RECORD-PARSE-CLAUSE, COND, GO, NOT, PRINT, TERPRI

BOUND VARIABLES: Network represents the Augmented Transition Network description.

The-Network-Title represents the ATN name.

State-Description represents a list which contains an ATN node name and one or more state-transition rules. It specifies what courses to take in the process of driving a path through the network.

Newstate represents the name of some node in the ATN description.

Rule represents a description of what must be done when an ATN arc has been taken.

Remaining-Words represents a list of words remaining to be analyzed.

Current-Word represents the current word which is being analyzed.

Last-Parsed represents the word parsed most recently.

English-Clause represents the English clause to be parsed by the ATN Interpreter.

Network-Syntax-Error-Message represents a message provided by the ATN Syntax Analyzer reflecting the outcome of analyzing the ATN syntactically.

Error-Exists-in-Network-Syntax represents a boolean variable which signifies whether or not an ATN syntax error has been discovered.

Network-Syntax-Analysis-Mssg-Code represents any of Network-Syntax-Error-Message1 to Network-Syntax-Error-Message9 and Rule-Syntax-Error-Message1 to Rule-Syntax-Error-Message34.

Network-Syntax-Error-Message1 to Network-Syntax-Error-Message9 represents possible occurrences of ATN syntax errors.

Rule-Syntax-Error-Message1 to Rule-Syntax-Error-Message34 represents possible occurrences of ATN state-transition rule syntax errors.

FREE VARIABLES: None

APPENDIX B

THE AUGMENTED TRANSITION NETWORK (ATN) INTERPRETER AND SYNTAX ANALYZER SOURCE CODE

The following listing is the LISP source code for the ATN Syntax Analyzer to syntactically investigate an ATN description and the ATN Interpreter to parse an English clause.

```
(DEFUN CAAR (LIST1)
        (CAR (CAR LIST1)))


(DEFUN CADR (LIST1)
        (CAR (CDR LIST1)))


(DEFUN CADDR (LIST1)
        (CAR (CDR (CDR LIST1))))


(DEFUN CADDDR (LIST1)
        (CAR (CDR (CDR (CDR LIST1)))))


(DEFUN CADDDDR (LIST1)
        (CAR (CDR (CDR (CDR (CDR LIST1))))))
```

```
(DEFUN CADDDDDR (LIST1)
       (CAR (CDR (CDR (CDR (CDR (CDR (CDR LIST1)))))))


(DEFUN CDAR (LIST1)
       (CDR (CAR LIST1)))


(DEFUN CDDR (LIST1)
       (CDR (CDR LIST1)))


(DEFUN CDDDR (LIST1)
       (CDR (CDR (CDR LIST1))))


(DEFUN CDDDDR (LIST1)
       (CDR (CDR (CDR (CDR LIST1)))))
```

```
(DEFUN CDDDDDR (LIST1)

        (CDR (CDR (CDR (CDR (CDR LIST1))))))



(DEFUN REMOVE-NESTING-IN-LIST (S-EXPR)

        (COND ((NULL S-EXPR) NIL)
              ((ATOM S-EXPR) (LIST S-EXPR))
              (T (APPEND (REMOVE-NESTING-IN-LIST (CAR S-EXPR))
                         (REMOVE-NESTING-IN-LIST (CDR S-EXPR))))))
```



```
(DEFUN CHECK-STATE-DESCRIPTION-SYNTAX ()

        (COND ((NULL STATE-DESCRIPTION)
               (SETQ NETWORK-SYNTAX-ERROR-MESSAGE NETWORK-SYNTAX-ERROR-MESSAGE2)
               (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
                     'NETWORK-SYNTAX-ERROR-MESSAGE2)
               (SETQ ERROR-EXISTS-IN-NETWORK-SYNTAX T))
              ((NOT (ATOM STATE-DESCRIPTION))
               (COND ((EQUAL (LENGTH STATE-DESCRIPTION) 1)
                      (SETQ NETWORK-SYNTAX-ERROR-MESSAGE
                            NETWORK-SYNTAX-ERROR-MESSAGE3)
                      (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
                            'NETWORK-SYNTAX-ERROR-MESSAGE3)
                      (SETQ ERROR-EXISTS-IN-NETWORK-SYNTAX T))))
              (T (SETQ NETWORK-SYNTAX-ERROR-MESSAGE
                       NETWORK-SYNTAX-ERROR-MESSAGE4)
                 (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
                       'NETWORK-SYNTAX-ERROR-MESSAGE4)
                 (SETQ ERROR-EXISTS-IN-NETWORK-SYNTAX T))))
```

```
(DEFUN CHECK-NODE-NAME-IN-STATE-DESCRIPTION ()
       (PROG (OLD-NEWSTATE OLD-STATE-DESCRIPTION)

             (COND ((NULL NEWSTATE)
                    (SETQ NETWORK-SYNTAX-ERROR-MESSAGE
                          NETWORK-SYNTAX-ERROR-MESSAGE5)
                    (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
                          'NETWORK-SYNTAX-ERROR-MESSAGE5)
                    (SETQ ERROR-EXISTS-IN-NETWORK-SYNTAX T))
                   ((NOT (ATOM NEWSTATE))
                    (COND ((EQUAL (LENGTH NEWSTATE) 1)
                           (SETQ OLD-NEWSTATE NEWSTATE)
                           (SETQ OLD-STATE-DESCRIPTION STATE-DESCRIPTION)
                           (COND ((NULL (REMOVE-NESTING-IN-LIST OLD-NEWSTATE))
                                  (SETQ NETWORK-SYNTAX-ERROR-MESSAGE
                                        NETWORK-SYNTAX-ERROR-MESSAGE6)
                                  (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
                                        'NETWORK-SYNTAX-ERROR-MESSAGE6)
                                  (SETQ ERROR-EXISTS-IN-NETWORK-SYNTAX T))
                                 ((EQUAL (LENGTH (REMOVE-NESTING-IN-LIST
                                                  OLD-NEWSTATE)) 1)
                                  (SETQ NEWSTATE
                                        (CAR (REMOVE-NESTING-IN-LIST
                                              OLD-NEWSTATE)))
                                  (SETQ STATE-DESCRIPTION
                                        (SUBST NEWSTATE
                                               OLD-NEWSTATE
                                               OLD-STATE-DESCRIPTION))
                                  (SETQ NETWORK
                                        (SUBST STATE-DESCRIPTION
                                               OLD-STATE-DESCRIPTION
                                               NETWORK))
                                  (SETQ NETWORK-SYNTAX-ERROR-MESSAGE
                                        NETWORK-SYNTAX-ERROR-MESSAGE7)
                                  (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
                                        'NETWORK-SYNTAX-ERROR-MESSAGE7)
                                  (PRINT-THE-REPAIRED-NETWORK))
```

```
                                    (T (SETQ NETWORK-SYNTAX-ERROR-MESSAGE
                                             NETWORK-SYNTAX-ERROR-MESSAGE9)
                                       (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
                                             'NETWORK-SYNTAX-ERROR-MESSAGE9)
                                       (SETQ ERROR-EXISTS-IN-NETWORK-SYNTAX T))))
                                 (T (SETQ NETWORK-SYNTAX-ERROR-MESSAGE
                                          NETWORK-SYNTAX-ERROR-MESSAGE8)
                                    (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
                                          'NETWORK-SYNTAX-ERROR-MESSAGE8)
                                    (SETQ ERROR-EXISTS-IN-NETWORK-SYNTAX T)))))))


(DEFUN LIST-DIFFERENCE (LIST1 LIST2)

        (COND ((NULL LIST1) NIL)
              ((MEMBER (CAR LIST1) LIST2)
               (LIST-DIFFERENCE (CDR LIST1) LIST2))
              (T (CONS (CAR LIST1) (LIST-DIFFERENCE (CDR LIST1) LIST2)))))



(DEFUN REPLACE-RULE-SYNTAX-SEGMENT-WITH-KEYWORD-IF ()
        (PROG ()

              (RETURN (SUBST IF OLD-RULE-SYNTAX-SEGMENT1 OLD-RULE))))
```

```
(DEFUN INSERT-KEYWORD-IF-INTO-NETWORK-TRAN-RULE ()
       (PROG ()

             (RETURN (CONS IF OLD-RULE))))



(DEFUN REPLACE-RULE-SYNTAX-SEGMENT-WITH-STATE-TRAN-SYMBOL ()
       (PROG ()

              (RETURN (SUBST STATE-TRANSITION-SYMBOL
                             RULE-SYNTAX-SEGMENT3
                             OLD-RULE))))
```



```
(DEFUN INSERT-STATE-TRAN-SYMBOL-INTO-NETWORK-TRAN-RULE ()
       (PROG (SUB-RULE DIFFERENCE-OF-LISTS)

             (SETQ SUB-RULE (MEMBER RULE-SYNTAX-SEGMENT3 RULE))
             (SETQ DIFFERENCE-OF-LISTS (LIST-DIFFERENCE RULE SUB-RULE))
             (SETQ SUB-RULE (CONS STATE-TRANSITION-SYMBOL SUB-RULE))
             (RETURN (APPEND DIFFERENCE-OF-LISTS SUB-RULE))))



(DEFUN REPLACE-RULE-SYNTAX-SEGMENT-WITH-TRAN-RULE-NEW-NODE ()
       (PROG ()

             (RETURN (SUBST TRAN-RULE-NEW-NODE
                            RULE-SYNTAX-SEGMENT4
                            OLD-RULE))))
```

```
(DEFUN REPLACE-RULE-SYNTAX-SEGMENT-WITH-KEYWORD-AFTER ()
       (PROG ()

             (RETURN (SUBST AFTER RULE-SYNTAX-SEGMENT5 OLD-RULE))))



(DEFUN REPAIR-STATE-DESCRIPTION-AND-NETWORK ()

       (SETQ STATE-DESCRIPTION
             (SUBST RULE OLD-RULE OLD-STATE-DESCRIPTION))
       (SETQ NETWORK
             (SUBST STATE-DESCRIPTION
                    OLD-STATE-DESCRIPTION
                    NETWORK)))



(DEFUN REPAIR-VIA-KEYWORD-IF-REPLACEMENT ()

       (SETQ RULE (REPLACE-RULE-SYNTAX-SEGMENT-WITH-KEYWORD-IF))
       (REPAIR-STATE-DESCRIPTION-AND-NETWORK)
       (SETQ NETWORK-SYNTAX-ERROR-MESSAGE RULE-SYNTAX-ERROR-MESSAGE2)
       (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE 'RULE-SYNTAX-ERROR-MESSAGE2))
```

```
(DEFUN REPAIR-VIA-KEYWORD-IF-INSERTION ()

        (SETQ RULE (INSERT-KEYWORD-IF-INTO-NETWORK-TRAN-RULE))
        (REPAIR-STATE-DESCRIPTION-AND-NETWORK)
        (SETQ NETWORK-SYNTAX-ERROR-MESSAGE RULE-SYNTAX-ERROR-MESSAGE3)
        (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE 'RULE-SYNTAX-ERROR-MESSAGE3))



(DEFUN REPAIR-VIA-TEST-REPLACEMENT ()

        (SETQ RULE (SUBST 'T RULE-SYNTAX-SEGMENT2 OLD-RULE))
        (REPAIR-STATE-DESCRIPTION-AND-NETWORK)
        (SETQ NETWORK-SYNTAX-ERROR-MESSAGE RULE-SYNTAX-ERROR-MESSAGE20)
        (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
              'RULE-SYNTAX-ERROR-MESSAGE20))



(DEFUN REPAIR-VIA-STATE-TRAN-SYMBOL-REPLACEMENT ()

        (SETQ RULE (REPLACE-RULE-SYNTAX-SEGMENT-WITH-STATE-TRAN-SYMBOL))
        (REPAIR-STATE-DESCRIPTION-AND-NETWORK)
        (SETQ NETWORK-SYNTAX-ERROR-MESSAGE RULE-SYNTAX-ERROR-MESSAGE9)
        (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE 'RULE-SYNTAX-ERROR-MESSAGE9))
```

```
(DEFUN REPAIR-VIA-STATE-TRAN-SYMBOL-INSERTION ()

        (SETQ RULE (INSERT-STATE-TRAN-SYMBOL-INTO-NETWORK-TRAN-RULE))
        (REPAIR-STATE-DESCRIPTION-AND-NETWORK)
        (SETQ NETWORK-SYNTAX-ERROR-MESSAGE RULE-SYNTAX-ERROR-MESSAGE10)
        (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
              'RULE-SYNTAX-ERROR-MESSAGE10))



(DEFUN REPAIR-VIA-TRAN-RULE-NEW-NODE-REPLACEMENT ()

        (SETQ RULE (REPLACE-RULE-SYNTAX-SEGMENT-WITH-TRAN-RULE-NEW-NODE))
        (REPAIR-STATE-DESCRIPTION-AND-NETWORK)
        (SETQ NETWORK-SYNTAX-ERROR-MESSAGE RULE-SYNTAX-ERROR-MESSAGE14)
        (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
              'RULE-SYNTAX-ERROR-MESSAGE14))



(DEFUN REPAIR-VIA-KEYWORD-AFTER-REPLACEMENT ()

        (SETQ RULE (REPLACE-RULE-SYNTAX-SEGMENT-WITH-KEYWORD-AFTER))
        (REPAIR-STATE-DESCRIPTION-AND-NETWORK)
        (SETQ NETWORK-SYNTAX-ERROR-MESSAGE RULE-SYNTAX-ERROR-MESSAGE19)
        (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
              'RULE-SYNTAX-ERROR-MESSAGE19))
```

```
(DEFUN PRINT-THE-REPAIRED-NETWORK ()

       (PRINT 'THE-REPAIRED-AUGMENTED-TRANSITION-NETWORK-ABOVE-IS)
       (SP NETWORK))



(DEFUN EXAMINE-KEYWORD-IF-IN-LIST ()
       (PROG (TEMP-RULE-SYNTAX-SEGMENT1)

             (SETQ TEMP-RULE-SYNTAX-SEGMENT1
                   (REMOVE-NESTING-IN-LIST RULE-SYNTAX-SEGMENT1))
             (COND ((NULL TEMP-RULE-SYNTAX-SEGMENT1)
                    (SETQ NETWORK-SYNTAX-ERROR-MESSAGE
                          RULE-SYNTAX-ERROR-MESSAGE27)
                    (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
                          'RULE-SYNTAX-ERROR-MESSAGE27)
                    (RETURN NIL))
                   ((EQUAL (LENGTH TEMP-RULE-SYNTAX-SEGMENT1) 1)
                    (COND ((EQUAL (CAR TEMP-RULE-SYNTAX-SEGMENT1) 'T)
                           (REPAIR-VIA-KEYWORD-IF-INSERTION)
                           (PRINT-THE-REPAIRED-NETWORK)
                           (RETURN T))
                          (T (REPAIR-VIA-KEYWORD-IF-REPLACEMENT)
                             (PRINT-THE-REPAIRED-NETWORK)
                             (RETURN T))))
                   (T (SETQ NETWORK-SYNTAX-ERROR-MESSAGE
                            RULE-SYNTAX-ERROR-MESSAGE28)
                      (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
                            'RULE-SYNTAX-ERROR-MESSAGE28)
                      (RETURN NIL))))))
```

```
(DEFUN CHECK-KEYWORD-IF-IN-TRANSITION-RULE ()
       (PROG (RULE-SYNTAX-SEGMENT1 OLD-RULE-SYNTAX-SEGMENT1
              IF OLD-RULE OLD-STATE-DESCRIPTION)

             (SETQ RULE-SYNTAX-SEGMENT1 (CAR RULE))
             (SETQ OLD-RULE-SYNTAX-SEGMENT1 RULE-SYNTAX-SEGMENT1)
             (SETQ IF 'IF)
             (SETQ OLD-RULE RULE)
             (SETQ OLD-STATE-DESCRIPTION STATE-DESCRIPTION)

             (COND ((NULL RULE-SYNTAX-SEGMENT1)
                    (SETQ NETWORK-SYNTAX-ERROR-MESSAGE
                          RULE-SYNTAX-ERROR-MESSAGE26)
                    (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
                          'RULE-SYNTAX-ERROR-MESSAGE26)
                    (RETURN NIL))
                   ((NOT (ATOM RULE-SYNTAX-SEGMENT1))
                    (RETURN (EXAMINE-KEYWORD-IF-IN-LIST)))
                   (T (COND ((EQUAL RULE-SYNTAX-SEGMENT1 IF)
                             (RETURN T))
                            ((EQUAL RULE-SYNTAX-SEGMENT1 'T)
                             (REPAIR-VIA-KEYWORD-IF-INSERTION)
                             (PRINT-THE-REPAIRED-NETWORK)
                             (RETURN T))
                            (T (REPAIR-VIA-KEYWORD-IF-REPLACEMENT)
                               (PRINT-THE-REPAIRED-NETWORK)
                               (RETURN T)))))))
```

```
(DEFUN CHECK-TEST-IN-TRANSITION-RULE ()
       (PROG (RULE-SYNTAX-SEGMENT2 TEMP-RULE-SYNTAX-SEGMENT2
              OLD-RULE OLD-STATE-DESCRIPTION)

             (COND ((NULL (CDR RULE))
                    (SETQ NETWORK-SYNTAX-ERROR-MESSAGE
                          RULE-SYNTAX-ERROR-MESSAGE4)
                    (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
                          'RULE-SYNTAX-ERROR-MESSAGE4)
                    (RETURN NIL))
                   (T (SETQ RULE-SYNTAX-SEGMENT2 (CADR RULE))
                      (SETQ OLD-RULE RULE)
                      (SETQ OLD-STATE-DESCRIPTION STATE-DESCRIPTION)
                      (COND ((NULL RULE-SYNTAX-SEGMENT2)
                             (SETQ NETWORK-SYNTAX-ERROR-MESSAGE
                                   RULE-SYNTAX-ERROR-MESSAGE5)
                             (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
                                   'RULE-SYNTAX-ERROR-MESSAGE5)
                             (RETURN NIL))
                            ((NOT (ATOM RULE-SYNTAX-SEGMENT2))
                             (SETQ TEMP-RULE-SYNTAX-SEGMENT2
                                   (REMOVE-NESTING-IN-LIST
                                    RULE-SYNTAX-SEGMENT2))
                             (COND ((NULL TEMP-RULE-SYNTAX-SEGMENT2)
                                    (SETQ NETWORK-SYNTAX-ERROR-MESSAGE
                                          RULE-SYNTAX-ERROR-MESSAGE6)
                                    (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
                                          'RULE-SYNTAX-ERROR-MESSAGE6)
                                    (RETURN NIL))
                                   ((AND (EQUAL (LENGTH
                                                 TEMP-RULE-SYNTAX-SEGMENT2) 1)
                                         (EQUAL (CAR TEMP-RULE-SYNTAX-SEGMENT2)
                                                'T))
                                    (REPAIR-VIA-TEST-REPLACEMENT)
                                    (PRINT-THE-REPAIRED-NETWORK)
                                    (RETURN T))
                                   (T (RETURN T))))))
```

```
                        (T (COND ((EQUAL RULE-SYNTAX-SEGMENT2 'T)
                                  (RETURN T))
                                 (T (SETQ NETWORK-SYNTAX-ERROR-MESSAGE
                                          RULE-SYNTAX-ERROR-MESSAGE7)
                                    (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
                                          'RULE-SYNTAX-ERROR-MESSAGE7)
                                    (RETURN NIL)))))))))


(DEFUN EXAMINE-STATE-TRAN-SYMBOL-IN-LIST ()
       (PROG (TEMP-RULE-SYNTAX-SEGMENT3)

             (SETQ TEMP-RULE-SYNTAX-SEGMENT3
                   (REMOVE-NESTING-IN-LIST RULE-SYNTAX-SEGMENT3))
             (COND ((NULL TEMP-RULE-SYNTAX-SEGMENT3)
                    (SETQ NETWORK-SYNTAX-ERROR-MESSAGE
                          RULE-SYNTAX-ERROR-MESSAGE30)
                    (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
                          'RULE-SYNTAX-ERROR-MESSAGE30)
                    (RETURN NIL))
                   ((EQUAL (LENGTH TEMP-RULE-SYNTAX-SEGMENT3) 1)
                    (COND ((OR (ASSOC (CAR TEMP-RULE-SYNTAX-SEGMENT3) NETWORK)
                               (EQUAL (CAR TEMP-RULE-SYNTAX-SEGMENT3) WIN)
                               (EQUAL (CAR TEMP-RULE-SYNTAX-SEGMENT3) LOSE))
                           (REPAIR-VIA-STATE-TRAN-SYMBOL-INSERTION)
                           (PRINT-THE-REPAIRED-NETWORK)
                           (RETURN T))
                          (T (REPAIR-VIA-STATE-TRAN-SYMBOL-REPLACEMENT)
                             (PRINT-THE-REPAIRED-NETWORK)
                             (RETURN T))))
                   (T (SETQ NETWORK-SYNTAX-ERROR-MESSAGE
                            RULE-SYNTAX-ERROR-MESSAGE31)
                      (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
                            'RULE-SYNTAX-ERROR-MESSAGE31)
                      (RETURN NIL)))))
```

```
(DEFUN EXAMINE-ATOM-IF-STATE-TRAN-SYMBOL ()
       (PROG ()

             (COND ((EQUAL RULE-SYNTAX-SEGMENT3 STATE-TRANSITION-SYMBOL)
                    (RETURN T))
                   (T (COND ((OR (ASSOC RULE-SYNTAX-SEGMENT3 NETWORK)
                                 (EQUAL RULE-SYNTAX-SEGMENT3 WIN)
                                 (EQUAL RULE-SYNTAX-SEGMENT3 LOSE))
                             (REPAIR-VIA-STATE-TRAN-SYMBOL-INSERTION)
                             (PRINT-THE-REPAIRED-NETWORK)
                             (RETURN T))
                            (T (REPAIR-VIA-STATE-TRAN-SYMBOL-REPLACEMENT)
                               (PRINT-THE-REPAIRED-NETWORK)
                               (RETURN T))))))))
```

```
(DEFUN CHECK-STATE-TRANSITION-SYMBOL-IN-TRAN-RULE ()
       (PROG (RULE-SYNTAX-SEGMENT3 OLD-RULE-SYNTAX-SEGMENT3
              STATE-TRANSITION-SYMBOL
              WIN LOSE OLD-RULE OLD-STATE-DESCRIPTION)

             (COND ((NULL (CDDR RULE))
                    (SETQ NETWORK-SYNTAX-ERROR-MESSAGE
                          RULE-SYNTAX-ERROR-MESSAGE8)
                    (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
                          'RULE-SYNTAX-ERROR-MESSAGE8)
                    (RETURN NIL))
                   (T (SETQ RULE-SYNTAX-SEGMENT3 (CADDR RULE))
                      (SETQ OLD-RULE-SYNTAX-SEGMENT3 RULE-SYNTAX-SEGMENT3)
                      (SETQ STATE-TRANSITION-SYMBOL '>>> )
                      (SETQ WIN 'WIN)
                      (SETQ LOSE 'LOSE)
                      (SETQ OLD-RULE RULE)
                      (SETQ OLD-STATE-DESCRIPTION STATE-DESCRIPTION)
                      (COND ((NULL RULE-SYNTAX-SEGMENT3)
                             (SETQ NETWORK-SYNTAX-ERROR-MESSAGE
                                   RULE-SYNTAX-ERROR-MESSAGE29)
                             (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
                                   'RULE-SYNTAX-ERROR-MESSAGE29)
                             (RETURN NIL))
                            ((NOT (ATOM RULE-SYNTAX-SEGMENT3))
                             (RETURN (EXAMINE-STATE-TRAN-SYMBOL-IN-LIST)))
                            (T (RETURN (EXAMINE-ATOM-IF-STATE-TRAN-SYMBOL))))
                                                                        ))))
```

```
(DEFUN EXAMINE-NEW-NODE-IN-LIST ()
       (PROG (TEMP-RULE-SYNTAX-SEGMENT4 TRAN-RULE-NEW-NODE)

             (COND ((EQUAL (LENGTH RULE-SYNTAX-SEGMENT4) 1)
                    (SETQ TEMP-RULE-SYNTAX-SEGMENT4
                          (REMOVE-NESTING-IN-LIST RULE-SYNTAX-SEGMENT4))
                    (COND ((NULL TEMP-RULE-SYNTAX-SEGMENT4)
                           (SETQ NETWORK-SYNTAX-ERROR-MESSAGE
                                 RULE-SYNTAX-ERROR-MESSAGE13)
                           (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
                                 'RULE-SYNTAX-ERROR-MESSAGE13)
                           (RETURN NIL))
                          ((EQUAL (LENGTH TEMP-RULE-SYNTAX-SEGMENT4) 1)
                           (SETQ TRAN-RULE-NEW-NODE
                                 (CAR TEMP-RULE-SYNTAX-SEGMENT4))
                           (COND ((OR (ASSOC TRAN-RULE-NEW-NODE NETWORK)
                                      (EQUAL TRAN-RULE-NEW-NODE WIN)
                                      (EQUAL TRAN-RULE-NEW-NODE LOSE))
                                  (REPAIR-VIA-TRAN-RULE-NEW-NODE-REPLACEMENT)
                                  (PRINT-THE-REPAIRED-NETWORK)
                                  (RETURN T))
                                 (T (SETQ NETWORK-SYNTAX-ERROR-MESSAGE
                                          RULE-SYNTAX-ERROR-MESSAGE15)
                                    (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
                                          'RULE-SYNTAX-ANALYSIS-MESSAGE15)
                                    (RETURN NIL))))
                          (T (SETQ NETWORK-SYNTAX-ERROR-MESSAGE
                                   RULE-SYNTAX-ERROR-MESSAGE16)
                             (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
                                   'RULE-SYNTAX-ERROR-MESSAGE16)
                             (RETURN NIL))))
                   (T (SETQ NETWORK-SYNTAX-ERROR-MESSAGE
                            RULE-SYNTAX-ERROR-MESSAGE17)
                      (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
                            'RULE-SYNTAX-ERROR-MESSAGE17)
                      (RETURN NIL)))))
```

```
(DEFUN CHECK-NEW-NODE-IN-TRANSITION-RULE ()
       (PROG (RULE-SYNTAX-SEGMENT4 OLD-RULE-SYNTAX-SEGMENT4
              WIN LOSE OLD-RULE OLD-STATE-DESCRIPTION)

             (COND ((NULL (CDDDR RULE))
                    (SETQ NETWORK-SYNTAX-ERROR-MESSAGE
                          RULE-SYNTAX-ERROR-MESSAGE11)
                    (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
                          'RULE-SYNTAX-ERROR-MESSAGE11)
                    (RETURN NIL))
                   (T (SETQ RULE-SYNTAX-SEGMENT4 (CADDDR RULE))
                      (SETQ OLD-RULE-SYNTAX-SEGMENT4 RULE-SYNTAX-SEGMENT4)
                      (SETQ WIN 'WIN)
                      (SETQ LOSE 'LOSE)
                      (SETQ OLD-RULE RULE)
                      (SETQ OLD-STATE-DESCRIPTION STATE-DESCRIPTION)
                      (COND ((NULL RULE-SYNTAX-SEGMENT4)
                             (SETQ NETWORK-SYNTAX-ERROR-MESSAGE
                                   RULE-SYNTAX-ERROR-MESSAGE12)
                             (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
                                   'RULE-SYNTAX-ERROR-MESSAGE12)
                             (RETURN NIL))
                            ((NOT (ATOM RULE-SYNTAX-SEGMENT4))
                             (RETURN (EXAMINE-NEW-NODE-IN-LIST)))
                            (T (COND ((OR (ASSOC RULE-SYNTAX-SEGMENT4 NETWORK)
                                          (EQUAL RULE-SYNTAX-SEGMENT4 WIN)
                                          (EQUAL RULE-SYNTAX-SEGMENT4 LOSE))
                                      (RETURN T))
                                     (T (SETQ NETWORK-SYNTAX-ERROR-MESSAGE
                                              RULE-SYNTAX-ERROR-MESSAGE18)
                                        (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
                                              'RULE-SYNTAX-ERROR-MESSAGE18)
                                        (RETURN NIL)))))))))))
```

```
(DEFUN EXAMINE-KEYWORD-AFTER-IN-LIST ()
       (PROG (TEMP-RULE-SYNTAX-SEGMENT5)

             (SETQ TEMP-RULE-SYNTAX-SEGMENT5
                   (REMOVE-NESTING-IN-LIST RULE-SYNTAX-SEGMENT5))
             (COND ((NULL TEMP-RULE-SYNTAX-SEGMENT5)
                    (SETQ NETWORK-SYNTAX-ERROR-MESSAGE
                          RULE-SYNTAX-ERROR-MESSAGE33)
                    (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
                          'RULE-SYNTAX-ERROR-MESSAGE33)
                    (RETURN NIL))
                   ((EQUAL (LENGTH TEMP-RULE-SYNTAX-SEGMENT5) 1)
                    (REPAIR-VIA-KEYWORD-AFTER-REPLACEMENT)
                    (PRINT-THE-REPAIRED-NETWORK)
                    (RETURN T))
                   (T (SETQ NETWORK-SYNTAX-ERROR-MESSAGE
                            RULE-SYNTAX-ERROR-MESSAGE34)
                      (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
                            'RULE-SYNTAX-ERROR-MESSAGE34)
                      (RETURN NIL)))))
```

```
(DEFUN CHECK-KEYWORD-AFTER-IN-TRANSITION-RULE ()
       (PROG (RULE-SYNTAX-SEGMENT5 OLD-RULE-SYNTAX-SEGMENT5
              AFTER
              OLD-RULE OLD-STATE-DESCRIPTION)

              (SETQ RULE-SYNTAX-SEGMENT5 (CADDDDR RULE))
              (SETQ OLD-RULE-SYNTAX-SEGMENT5 RULE-SYNTAX-SEGMENT5)
              (SETQ AFTER 'AFTER)
              (SETQ OLD-RULE RULE)
              (SETQ OLD-STATE-DESCRIPTION STATE-DESCRIPTION)
              (COND ((NULL RULE-SYNTAX-SEGMENT5)
                     (SETQ NETWORK-SYNTAX-ERROR-MESSAGE
                           RULE-SYNTAX-ERROR-MESSAGE32)
                     (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
                           'RULE-SYNTAX-ERROR-MESSAGE32)
                     (RETURN NIL))
                    ((NOT (ATOM RULE-SYNTAX-SEGMENT5))
                     (RETURN (EXAMINE-KEYWORD-AFTER-IN-LIST)))
                    (T (COND ((EQUAL RULE-SYNTAX-SEGMENT5 AFTER)
                              (RETURN T))
                             (T (REPAIR-VIA-KEYWORD-AFTER-REPLACEMENT)
                                (PRINT-THE-REPAIRED-NETWORK)
                                (RETURN T))))))))
```

```
(DEFUN CHECK-SIDE-EFFECTS-SYNTAX ()
       (PROG (RULE-SYNTAX-SEGMENT6 LIST-OF-SIDE-EFFECTS
              NO-ERRORS-EXIST-IN-LIST-OF-SIDE-EFFECTS)

             (SETQ LIST-OF-SIDE-EFFECTS (CDDDDDR RULE))

             FETCH-A-SIDE-EFFECT
             (SETQ RULE-SYNTAX-SEGMENT6 (CAR LIST-OF-SIDE-EFFECTS))
             (COND ((NULL RULE-SYNTAX-SEGMENT6)
                    (SETQ NETWORK-SYNTAX-ERROR-MESSAGE
                          RULE-SYNTAX-ERROR-MESSAGE22)
                    (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
                          'RULE-SYNTAX-ERROR-MESSAGE22)
                    (RETURN NIL))
                   ((NOT (ATOM RULE-SYNTAX-SEGMENT6))
                    (COND ((NULL (REMOVE-NESTING-IN-LIST RULE-SYNTAX-SEGMENT6))
                           (SETQ NETWORK-SYNTAX-ERROR-MESSAGE
                                 RULE-SYNTAX-ERROR-MESSAGE23)
                           (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
                                 'RULE-SYNTAX-ERROR-MESSAGE23)
                           (RETURN NIL))
                          (T (SETQ NO-ERRORS-EXIST-IN-LIST-OF-SIDE-EFFECTS
                                   T))))
                   (T (SETQ NETWORK-SYNTAX-ERROR-MESSAGE
                            RULE-SYNTAX-ERROR-MESSAGE24)
                      (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
                            'RULE-SYNTAX-ERROR-MESSAGE24)
                      (RETURN NIL)))
             (SETQ LIST-OF-SIDE-EFFECTS (CDR LIST-OF-SIDE-EFFECTS))
             (COND ((NULL LIST-OF-SIDE-EFFECTS)
                    (GO EXIT-CHECK-SIDE-EFFECTS-SYNTAX))
                   (T (GO FETCH-A-SIDE-EFFECT)))

             EXIT-CHECK-SIDE-EFFECTS-SYNTAX
             (RETURN NO-ERRORS-EXIST-IN-LIST-OF-SIDE-EFFECTS)))
```

```
(DEFUN CHECK-RULE-SYNTAX-SEGMENTS ()

       (COND ((AND (CHECK-KEYWORD-IF-IN-TRANSITION-RULE)
                   (CHECK-TEST-IN-TRANSITION-RULE)
                   (CHECK-STATE-TRANSITION-SYMBOL-IN-TRAN-RULE)
                   (CHECK-NEW-NODE-IN-TRANSITION-RULE))
              (COND ((NOT (NULL (CDDDDR RULE)))
                     (COND ((CHECK-KEYWORD-AFTER-IN-TRANSITION-RULE)
                            (COND ((NULL (CDDDDDR RULE))
                                   (SETQ NETWORK-SYNTAX-ERROR-MESSAGE
                                         RULE-SYNTAX-ERROR-MESSAGE21)
                                   (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
                                         'RULE-SYNTAX-ERROR-MESSAGE21)
                                   NIL)
                                  (T (CHECK-SIDE-EFFECTS-SYNTAX))))))
                    (T)))))
```



```
(DEFUN CHECK-RULE-SYNTAX ()

       (COND ((NULL RULE)
              (SETQ NETWORK-SYNTAX-ERROR-MESSAGE RULE-SYNTAX-ERROR-MESSAGE1)
              (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
                    'RULE-SYNTAX-ERROR-MESSAGE1)
              (SETQ ERROR-EXISTS-IN-NETWORK-SYNTAX T))
             ((NOT (ATOM RULE))
              (SETQ ERROR-EXISTS-IN-NETWORK-SYNTAX
                    (NOT (CHECK-RULE-SYNTAX-SEGMENTS))))
             (T (SETQ NETWORK-SYNTAX-ERROR-MESSAGE RULE-SYNTAX-ERROR-MESSAGE25)
                (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
                      'RULE-SYNTAX-ERROR-MESSAGE25)
                (SETQ ERROR-EXISTS-IN-NETWORK-SYNTAX T))))
```

```
(DEFUN CHECK-NETWORK-SYNTAX ()
       (PROG (LIST-OF-STATE-DESCRIPTIONS LIST-OF-RULES)

             (COND ((NULL NETWORK)
                    (SETQ NETWORK-SYNTAX-ERROR-MESSAGE
                          NETWORK-SYNTAX-ERROR-MESSAGE1)
                    (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE
                          'NETWORK-SYNTAX-ERROR-MESSAGE1)
                    (SETQ ERROR-EXISTS-IN-NETWORK-SYNTAX T)
                    (RETURN ERROR-EXISTS-IN-NETWORK-SYNTAX)))
             (SETQ LIST-OF-STATE-DESCRIPTIONS NETWORK)

             FETCH-A-STATE-DESCRIPTION
             (SETQ STATE-DESCRIPTION (CAR LIST-OF-STATE-DESCRIPTIONS))
             (CHECK-STATE-DESCRIPTION-SYNTAX)
             (COND (ERROR-EXISTS-IN-NETWORK-SYNTAX
                    (RETURN ERROR-EXISTS-IN-NETWORK-SYNTAX)))

             (SETQ NEWSTATE (CAR STATE-DESCRIPTION))
             (CHECK-NODE-NAME-IN-STATE-DESCRIPTION)
             (COND (ERROR-EXISTS-IN-NETWORK-SYNTAX
                    (RETURN ERROR-EXISTS-IN-NETWORK-SYNTAX)))

             (SETQ LIST-OF-RULES (CDR STATE-DESCRIPTION))

             FETCH-A-TRANSITION-RULE
             (SETQ RULE (CAR LIST-OF-RULES))
             (CHECK-RULE-SYNTAX)
             (COND (ERROR-EXISTS-IN-NETWORK-SYNTAX
                    (RETURN ERROR-EXISTS-IN-NETWORK-SYNTAX)))
             (SETQ LIST-OF-RULES (CDR LIST-OF-RULES))
             (COND ((NULL LIST-OF-RULES)
                    (GO FETCH-NEXT-SET-OF-STATE-DESCRIPTIONS))
                   (T (GO FETCH-A-TRANSITION-RULE)))
```

```
        FETCH-NEXT-SET-OF-STATE-DESCRIPTIONS
        (SETQ LIST-OF-STATE-DESCRIPTIONS (CDR LIST-OF-STATE-DESCRIPTIONS))
        (COND ((NULL LIST-OF-STATE-DESCRIPTIONS)
               (GO EXIT-CHECK-NETWORK-SYNTAX))
              (T (GO FETCH-A-STATE-DESCRIPTION)))

        EXIT-CHECK-NETWORK-SYNTAX
        (RETURN ERROR-EXISTS-IN-NETWORK-SYNTAX)))



(DEFUN INITIALIZE-NETWORK-SYNTAX-ERROR-MESSAGES ()

        (SETQ NETWORK-SYNTAX-ERROR-MESSAGE1
              '(ERROR FOUND *** THE AUGMENTED TRANSITION NETWORK IS MISSING))
        (SETQ NETWORK-SYNTAX-ERROR-MESSAGE2
              '(ERROR FOUND *** THE NETWORK STATE DESCRIPTION IS AN EMPTY LIST))
        (SETQ NETWORK-SYNTAX-ERROR-MESSAGE3
              '(ERROR FOUND *** THE NUMBER OF NETWORK STATE DESCRIPTION SYNTAX
                            SEGMENTS IS NOT TWO OR MORE))
        (SETQ NETWORK-SYNTAX-ERROR-MESSAGE4
              '(ERROR FOUND *** THE NETWORK STATE DESCRIPTION IS AN ATOM))
        (SETQ NETWORK-SYNTAX-ERROR-MESSAGE5
              '(ERROR FOUND *** THE NODE NAME IN THE ORIGINAL NETWORK
                            IS AN EMPTY LIST))
        (SETQ NETWORK-SYNTAX-ERROR-MESSAGE6
              '(ERROR FOUND *** THE NODE NAME IN THE ORIGINAL NETWORK
                            IS A NESTED EMPTY LIST))
        (SETQ NETWORK-SYNTAX-ERROR-MESSAGE7
              '(ERROR FOUND BUT REPAIR MADE *** THE NODE NAME IN THE ORIGINAL
                                                NETWORK HAS BEEN REPAIRED))
        (SETQ NETWORK-SYNTAX-ERROR-MESSAGE8
              '(ERROR FOUND *** THE NODE NAME IN THE ORIGINAL NETWORK
                            IS A LIST OF DIFFERENT S-EXPRESSIONS))
        (SETQ NETWORK-SYNTAX-ERROR-MESSAGE9
              '(ERROR FOUND *** THE NODE NAME IN THE ORIGINAL NETWORK IS A
                            NESTED LIST OF DIFFERENT S-EXPRESSIONS)))
```

```
(DEFUN INITIALIZE-RULE-SYNTAX-ERROR-MESSAGES ()

       (SETQ RULE-SYNTAX-ERROR-MESSAGE1
             '(ERROR FOUND *** THE TRANSITION RULE IN THE NETWORK
                               IS AN EMPTY LIST))
       (SETQ RULE-SYNTAX-ERROR-MESSAGE2
             '(ERROR FOUND BUT REPAIR MADE *** THE KEYWORD IF HAS REPLACED
                                               THE RULE SYNTAX SEGMENT))
       (SETQ RULE-SYNTAX-ERROR-MESSAGE3
             '(ERROR FOUND BUT REPAIR MADE *** THE KEYWORD IF HAS BEEN
                                               INSERTED INTO THE NETWORK
                                               TRANSITION RULE))
       (SETQ RULE-SYNTAX-ERROR-MESSAGE4
             '(ERROR FOUND *** THE TEST IN THE TRANSITION RULE IS MISSING))
       (SETQ RULE-SYNTAX-ERROR-MESSAGE5
             '(ERROR FOUND *** THE TEST IN THE TRANSITION RULE IS
                               AN EMPTY LIST))
       (SETQ RULE-SYNTAX-ERROR-MESSAGE6
             '(ERROR FOUND *** THE TEST IN THE TRANSITION RULE IS
                               A NESTED EMPTY LIST))
       (SETQ RULE-SYNTAX-ERROR-MESSAGE7
             '(ERROR FOUND *** THE TEST IN THE TRANSITION RULE IS AN
                               ILLEGAL ATOM))
       (SETQ RULE-SYNTAX-ERROR-MESSAGE8
             '(ERROR FOUND *** THE STATE TRANSITION SYMBOL IN THE
                               TRANSITION RULE IS MISSING))
       (SETQ RULE-SYNTAX-ERROR-MESSAGE9
             '(ERROR FOUND BUT REPAIR MADE *** THE STATE TRANSITION SYMBOL
                                               HAS REPLACED THE RULE SYNTAX
                                               SEGMENT))
       (SETQ RULE-SYNTAX-ERROR-MESSAGE10
             '(ERROR FOUND BUT REPAIR MADE *** THE STATE TRANSITION SYMBOL
                                               HAS BEEN INSERTED INTO THE
                                               NETWORK TRANSITON RULE))
       (SETQ RULE-SYNTAX-ERROR-MESSAGE11
             '(ERROR FOUND *** THE NEW NODE IS MISSING FROM THE
                               NETWORK TRANSITION RULE))
```

```
(SETQ RULE-SYNTAX-ERROR-MESSAGE12
      '(ERROR FOUND *** THE NEW NODE IN THE NETWORK TRANSITION RULE
                        IS AN EMPTY LIST))
(SETQ RULE-SYNTAX-ERROR-MESSAGE13
      '(ERROR FOUND *** THE NEW NODE IN THE NETWORK TRANSITION RULE
                        IS A NESTED EMPTY LIST))
(SETQ RULE-SYNTAX-ERROR-MESSAGE14
      '(ERROR FOUND BUT REPAIR MADE *** THE NEW NODE IN THE NETWORK
                                        TRANSITION RULE HAS REPLACED
                                        THE RULE SYNTAX SEGMENT))
(SETQ RULE-SYNTAX-ERROR-MESSAGE15
      '(ERROR FOUND *** THE TRANSITION RULE NEW NODE IN A LIST IS NOT
                        A NETWORK NEW NODE))
(SETQ RULE-SYNTAX-ERROR-MESSAGE16
      '(ERROR FOUND *** THE NEW NODE IN THE TRANSITION RULE IS A
                        NESTED LIST OF DIFFERENT S-EXPRS))
(SETQ RULE-SYNTAX-ERROR-MESSAGE17
      '(ERROR FOUND *** THE NEW NODE IN THE TRANSITION RULE IS A LIST
                        OF DIFFERENT S-EXPRS))
(SETQ RULE-SYNTAX-ERROR-MESSAGE18
      '(ERROR FOUND *** THE NEW NODE IN THE TRANSITION RULE IS NOT A
                        NETWORK NEW NODE))
(SETQ RULE-SYNTAX-ERROR-MESSAGE19
      '(ERROR FOUND BUT REPAIR MADE *** THE KEYWORD AFTER HAS REPLACED
                                        THE RULE SYNTAX SEGMENT))
(SETQ RULE-SYNTAX-ERROR-MESSAGE20
      '(ERROR FOUND BUT REPAIR MADE *** THE TRANSITION RULE TEST
                                        WHICH IS A BOOLEAN ATOM T IN A
                                        NESTED LIST HAS BEEN REPLACED
                                        BY A BOOLEAN ATOM T))
(SETQ RULE-SYNTAX-ERROR-MESSAGE21
      '(ERROR FOUND *** THE SIDE EFFECTS ARE MISSING FROM THE
                        NETWORK TRANSITION RULE))
(SETQ RULE-SYNTAX-ERROR-MESSAGE22
      '(ERROR FOUND *** THE SIDE EFFECTS IN THE NETWORK TRANSITION
                        RULE IS AN EMPTY LIST))
(SETQ RULE-SYNTAX-ERROR-MESSAGE23
      '(ERROR FOUND *** THE SIDE EFFECTS IN THE NETWORK TRANSITION
                        RULE IS A NESTED EMPTY LIST))
```

```
(SETQ RULE-SYNTAX-ERROR-MESSAGE24
      '(ERROR FOUND *** THE SIDE EFFECTS IN THE NETWORK TRANSITION
                        RULE IS AN ATOM))
(SETQ RULE-SYNTAX-ERROR-MESSAGE25
      '(ERROR FOUND *** THE TRANSITION RULE IN THE NETWORK IS
                        AN ATOM))
(SETQ RULE-SYNTAX-ERROR-MESSAGE26
      '(ERROR FOUND *** THE FIRST SYNTAX SEGMENT OF THE TRANSITION
                        RULE IS AN EMPTY LIST))
(SETQ RULE-SYNTAX-ERROR-MESSAGE27
      '(ERROR FOUND *** THE FIRST SYNTAX SEGMENT OF THE TRANSITION
                        RULE IS A NESTED EMPTY LIST))
(SETQ RULE-SYNTAX-ERROR-MESSAGE28
      '(ERROR FOUND *** THE FIRST SYNTAX SEGMENT OF THE TRANSITION
                        RULE IS NOT THE KEYWORD IF))
(SETQ RULE-SYNTAX-ERROR-MESSAGE29
      '(ERROR FOUND *** THE THIRD SYNTAX SEGMENT OF THE TRANSITION
                        RULE IS AN EMPTY LIST))
(SETQ RULE-SYNTAX-ERROR-MESSAGE30
      '(ERROR FOUND *** THE THIRD SYNTAX SEGMENT OF THE TRANSITION
                        RULE IS A NESTED EMPTY LIST))
(SETQ RULE-SYNTAX-ERROR-MESSAGE31
      '(ERROR FOUND *** THE THIRD SYNTAX SEGMENT OF THE TRANSITION
                        RULE IS A LIST OF DIFFERENT S-EXPRESSIONS))
(SETQ RULE-SYNTAX-ERROR-MESSAGE32
      '(ERROR FOUND *** THE FIFTH SYNTAX SEGMENT OF THE TRANSITION
                        RULE IS AN EMPTY LIST))
(SETQ RULE-SYNTAX-ERROR-MESSAGE33
      '(ERROR FOUND *** THE FIFTH SYNTAX SEGMENT OF THE TRANSITION
                        RULE IS A NESTED EMPTY LIST))
(SETQ RULE-SYNTAX-ERROR-MESSAGE34
      '(ERROR FOUND *** THE FIFTH SYNTAX SEGMENT OF THE TRANSITION
                        RULE IS NOT THE KEYWORD AFTER)))
```

```
(DEFUN INITIALIZE-NETWORK-SYNTAX-SEGMENTS ()

        (SETQ STATE-DESCRIPTION
              '(THE STATE-DESCRIPTION IS UNKNOWN *** THE STATE-DESCRIPTION
                                                     ASSIGNMENT HAS NOT YET BEEN
                                                     MADE))
        (SETQ NEWSTATE '(THE NEWSTATE IS UNKNOWN *** THE NEWSTATE ASSIGNMENT
                                                     HAS NOT YET BEEN MADE))
        (SETQ RULE '(THE RULE IS UNKNOWN *** THE RULE ASSIGNMENT HAS NOT YET
                                             BEEN MADE)))



(DEFUN GETF (THE-FEATURES)

        (GET THE-FEATURES 'FEATURES))



(DEFUN INTERSECTION (LIST1 LIST2)

        (COND ((NULL LIST1) NIL)
              ((MEMBER (CAR LIST1) LIST2)
               (CONS (CAR LIST1) (INTERSECTION (CDR LIST1) LIST2)))
              (T (INTERSECTION (CDR LIST1) LIST2))))
```

```
(DEFUN TESTF (NODE FEATURES)

       (COND ((NULL FEATURES))
             ((ATOM FEATURES)
              (SETQ FEATURES (LIST FEATURES))))
       (EQUAL (LENGTH FEATURES)
              (LENGTH (INTERSECTION FEATURES (GETF NODE)))))



(DEFUN PRINT-THE-PARENT-AND-THE-CHILDREN ()

       (TERPRI) (TERPRI)
       (PRINC 'THE-PARENT-IS)
       (XTAB 7)
       (PRINT (GET THE-CHILD 'PARENT))
       (PRINC 'THE-CHILD-IS)
       (XTAB 8)
       (PRINT (GET THE-PARENT 'CHILDREN)))



(DEFUN ATTACH (THE-CHILD THE-PARENT)
       (PROG (THE-CHILDREN)

             (PUTPROP THE-CHILD THE-PARENT 'PARENT)
             (SETQ THE-CHILDREN
                   (APPEND (GET THE-PARENT 'CHILDREN) (LIST THE-CHILD)))
             (PUTPROP THE-PARENT THE-CHILDREN 'CHILDREN)
             (PRINT-THE-PARENT-AND-THE-CHILDREN)))
```

```
(DEFUN SELECT (S-EXPR1 S-EXPR2)

        (COND ((NULL S-EXPR1) NIL)
              ((MEMBER (CAR S-EXPR1) S-EXPR2) (CAR S-EXPR1))
              (T (SELECT (CDR S-EXPR1) S-EXPR2))))



(DEFUN GENNAME (NAME)

        (IMPLODE (APPEND (EXPLODE NAME)
                         (CDDDDR (EXPLODE (GENSYM NAME))))))



(DEFUN PRINT-THE-REGISTER-AND-THE-VALUE-OF-THIS-NODE ()

        (PRINO 'THE-REGISTER-IS)
        (XTAB 5)
        (PRINT REGISTER)
        (PRINO 'THE-VALUE-IS)
        (XTAB 8)
        (PRINT VALUE))



(DEFUN SETR (REGISTER VALUE)

        (PUTPROP THIS-NODE VALUE REGISTER)
        (PRINT-THE-REGISTER-AND-THE-VALUE-OF-THIS-NODE)
        VALUE)
```

```
(DEFUN GETR (REGISTER)
        (GET THIS-NODE REGISTER))


(DEFUN ADDR (REGISTER VALUE)
        (SETR REGISTER (CONS VALUE (GETR REGISTER))))
```

```
(DEFUN INTERPRET (NETWORK PARENT-NODE FEATURES)
       (PROG (NEWSTATE OLDSTATE STATE-DESCRIPTION RULE HOLD THIS-NODE)

             (SETQ HOLD REMAINING-WORDS)
             (SETQ CURRENT-WORD (CAR REMAINING-WORDS))
             (SETQ THIS-NODE (GENNAME (CAR NETWORK)))
             (SETQ NETWORK (CDR NETWORK))
             (SETQ NEWSTATE (CAAR NETWORK))

             GET-STATE-DESCRIPTION
             (COND ((EQUAL NEWSTATE 'WIN) (GO WIN))
                   ((EQUAL NEWSTATE 'LOSE) (GO LOSE))
                   (T (SETQ STATE-DESCRIPTION (ASSOC NEWSTATE NETWORK))))

             TEST-TRANSITION-RULE
             (COND ((SETQ STATE-DESCRIPTION (CDR STATE-DESCRIPTION))
                    (SETQ RULE (CAR STATE-DESCRIPTION)))
                   (T (SETQ OLDSTATE NEWSTATE)
                      (SETQ NEWSTATE 'LOSE)
                      (GO GET-STATE-DESCRIPTION)))

             (COND ((EVAL (CADR RULE))
                    (SETQ OLDSTATE NEWSTATE)
                    (SETQ NEWSTATE (CADDDR RULE))
                    (COND ((CDDDDR RULE)
                           (MAPCAR '(LAMBDA (X) (EVAL X)) (CDR (CDDDDR RULE)))))
                    (GO GET-STATE-DESCRIPTION))
                   (T (GO TEST-TRANSITION-RULE)))

             WIN
             (COND ((NOT (TESTF THIS-NODE FEATURES)) (GO LOSE)))
             (ATTACH THIS-NODE PARENT-NODE)
             (SETQ LAST-PARSED THIS-NODE)
             (RETURN THIS-NODE)

             LOSE
             (SETQ REMAINING-WORDS HOLD)
             (SETQ CURRENT-WORD (CAR REMAINING-WORDS))
             (RETURN NIL)))
```

```
(DEFUN PRINT-THE-NETWORK-SYNTAX-ERRORS ()

        (PRINT 'THE-STATE-DESCRIPTION-IS)
        (TERPRI)
        (PRINT STATE-DESCRIPTION)
        (TERPRI) (TERPRI)
        (PRINT 'THE-NEWSTATE-IS)
        (TERPRI)
        (PRINT NEWSTATE)
        (TERPRI) (TERPRI)
        (PRINT 'THE-RULE-IS)
        (TERPRI)
        (PRINT RULE)
        (TERPRI) (TERPRI)
        (PRINT 'THE-NETWORK-SYNTAX-ANALYSIS-MSSG-CODE-IS)
        (TERPRI)
        (PRINT NETWORK-SYNTAX-ANALYSIS-MSSG-CODE)
        (TERPRI) (TERPRI)
        (PRINT 'THE-NETWORK-SYNTAX-ERROR-MESSAGE-IS)
        (TERPRI)
        (PRINT NETWORK-SYNTAX-ERROR-MESSAGE)
        (TERPRI) (TERPRI))
```

```
(DEFUN RECORD FEXPR (THE-AUG-TRAN-NETWORK)

        (SETQ NETWORK-SYNTAX-ERROR-MESSAGE
              '(NO ERRORS FOUND *** THE ORIGINAL AUGMENTED TRANSITION NETWORK
                                   ABOVE IS SYNTACTICALLY CORRECT))
        (SETQ ERROR-EXISTS-IN-NETWORK-SYNTAX NIL)
        (SETQ NETWORK-SYNTAX-ANALYSIS-MSSG-CODE NIL)

        (INITIALIZE-NETWORK-SYNTAX-SEGMENTS)

        (SETQ THE-NETWORK-TITLE (CAR THE-AUG-TRAN-NETWORK))
        (PUTPROP THE-NETWORK-TITLE THE-AUG-TRAN-NETWORK 'NETWORK)
        (PRINO 'THE-ORIGINAL-AUGMENTED-TRANSITION-NETWORK-TITLE-IS)
        (XTAB 5)
        (PRINT THE-NETWORK-TITLE)
        (TERPRI)

        (SETQ NETWORK (GET THE-NETWORK-TITLE 'NETWORK))
        (SETQ NETWORK (CDR NETWORK))
        (PRINT 'THE-ORIGINAL-AUGMENTED-TRANSITION-NETWORK-IS)
        (SP NETWORK)
        (TERPRI)

        (CHECK-NETWORK-SYNTAX)
        (COND (ERROR-EXISTS-IN-NETWORK-SYNTAX
                (PRINT-THE-NETWORK-SYNTAX-ERRORS))
              (T (PRINT 'THE-FINAL-NETWORK-SYNTAX-ANALYSIS-MSSG-CODE-IS)
                 (TERPRI)
                 (PRINT NETWORK-SYNTAX-ANALYSIS-MSSG-CODE)
                 (TERPRI)
                 (PRINT 'THE-FINAL-SYNTAX-ANALYSIS-MESSAGE-FOR-THE-NETWORK-IS)
                 (TERPRI)
                 (PRINT NETWORK-SYNTAX-ERROR-MESSAGE)
                 (TERPRI) (TERPRI) (TERPRI) (TERPRI) (TERPRI)
                 (SETQ THE-AUG-TRAN-NETWORK (CONS THE-NETWORK-TITLE NETWORK))
                 (PUTPROP THE-NETWORK-TITLE THE-AUG-TRAN-NETWORK 'NETWORK)
                 (EVAL (SUBST (CAR THE-AUG-TRAN-NETWORK) 'NAME
                              '(DEFUN NAME (PARENT-NODE FEATURES)
                                      (INTERPRET (GET 'NAME 'NETWORK)
                                                 PARENT-NODE
                                                 FEATURES)))))))
```

```
(DEFUN RECORD-PARSE-WORD ()

        (RECORD PARSE-WORD
                (S1 (IF T >>>   WIN
                        AFTER
                        (SETQ THIS-NODE CURRENT-WORD)
                        (SETQ REMAINING-WORDS (CDR REMAINING-WORDS))
                        (COND (REMAINING-WORDS
                               (SETQ CURRENT-WORD
                                     (CAR REMAINING-WORDS)))
                              (T (SETQ CURRENT-WORD NIL))))))
                                                          )
```



```
(DEFUN RECORD-PARSE-NOUN-GROUP ()

        (RECORD PARSE-NOUN-GROUP
                (S1 (IF (PARSE-WORD THIS-NODE 'DETERMINER)
                        >>> S2
                        AFTER
                        (SETR 'NUMBER (SELECT '(SINGULAR PLURAL)
                                              (GETF LAST-PARSED)))
                        (SETR 'DETERMINER (SELECT '(DEFINITE INDEFINITE)
                                                  (GETF LAST-PARSED)))))
                (S2 (IF (PARSE-WORD THIS-NODE 'ADJECTIVE)
                        >>> S2
                        AFTER
                        (ADDR 'ADJECTIVES LAST-PARSED))
                    (IF (PARSE-WORD THIS-NODE 'NOUN)
                        >>> WIN
                        AFTER
                        (SETR 'NUMBER (SELECT '(SINGULAR PLURAL)
                                              (GETF LAST-PARSED)))
                        (SETR 'NOUN LAST-PARSED))))
                                                  )
```

```
(DEFUN RECORD-PARSE-CLAUSE ()

       (RECORD PARSE-CLAUSE
              (S1 (IF (PARSE-NOUN-GROUP THIS-NODE NIL)
                      >>> S2
                      AFTER
                      (SETR 'SUBJECT LAST-PARSED)))
              (S2 (IF (PARSE-WORD THIS-NODE '(VERB TENSED))
                      >>> S3
                      AFTER (SETR 'VERB (GET LAST-PARSED 'ROOT))))
              (S3 (IF (AND (EQUAL (GETR 'VERB) 'BE)
                           (PARSE-WORD THIS-NODE 'PASTPARTICIPLE))
                      >>> S4
                      AFTER
                      (SETR 'OBJECT (GETR 'SUBJECT))
                      (SETR 'SUBJECT NIL)
                      (SETR 'VERB LAST-PARSED))
                  (IF (AND (TESTF (GETR 'VERB) 'TRANSITIVE)
                           (PARSE-NOUN-GROUP THIS-NODE NIL))
                      >>> S4
                      AFTER (SETR 'OBJECT LAST-PARSED))
                  (IF (OR (TESTF (GETR 'VERB) 'INTRANSITIVE)
                          (GETR 'OBJECT))
                      >>> S4))
              (S4 (IF (AND (GETR 'SUBJECT)
                           (NULL REMAINING-WORDS))
                      >>> WIN)
                  (IF (AND (NOT (GETR 'SUBJECT))
                           (EQUAL CURRENT-WORD 'BY)
                           (PARSE-WORD THIS-NODE NIL))
                      >>> S5)
                  (IF (NOT (GETR 'SUBJECT))
                      >>> S4
                      AFTER
                      (SETR 'SUBJECT 'SOMEONE)))
              (S5 (IF (PARSE-NOUN-GROUP THIS-NODE NIL)
                      >>> S4
                      AFTER
                      (SETR 'SUBJECT LAST-PARSED))))
                                                    )
```

```
(SETQ *OTLL* 112)
(SETQ *LEVEL* 1000)
(SETQ *LINE-LENGTH* 120)
(SETQ *SCREEN-LENGTH* 1000)



(DEFUN IDENTIFY ()
       (PROG (NETWORK THE-NETWORK-TITLE STATE-DESCRIPTION NEWSTATE RULE
              REMAINING-WORDS CURRENT-WORD LAST-PARSED ENGLISH-CLAUSE

              NETWORK-SYNTAX-ERROR-MESSAGE ERROR-EXISTS-IN-NETWORK-SYNTAX
              NETWORK-SYNTAX-ANALYSIS-MSSG-CODE

              NETWORK-SYNTAX-ERROR-MESSAGE1 NETWORK-SYNTAX-ERROR-MESSAGE2
              NETWORK-SYNTAX-ERROR-MESSAGE3 NETWORK-SYNTAX-ERROR-MESSAGE4
              NETWORK-SYNTAX-ERROR-MESSAGE5 NETWORK-SYNTAX-ERROR-MESSAGE6
              NETWORK-SYNTAX-ERROR-MESSAGE7 NETWORK-SYNTAX-ERROR-MESSAGE8
              NETWORK-SYNTAX-ERROR-MESSAGE9

              RULE-SYNTAX-ERROR-MESSAGE1  RULE-SYNTAX-ERROR-MESSAGE2
              RULE-SYNTAX-ERROR-MESSAGE3  RULE-SYNTAX-ERROR-MESSAGE4
              RULE-SYNTAX-ERROR-MESSAGE5  RULE-SYNTAX-ERROR-MESSAGE6
              RULE-SYNTAX-ERROR-MESSAGE7  RULE-SYNTAX-ERROR-MESSAGE8
              RULE-SYNTAX-ERROR-MESSAGE9  RULE-SYNTAX-ERROR-MESSAGE10
              RULE-SYNTAX-ERROR-MESSAGE11 RULE-SYNTAX-ERROR-MESSAGE12
              RULE-SYNTAX-ERROR-MESSAGE13 RULE-SYNTAX-ERROR-MESSAGE14
              RULE-SYNTAX-ERROR-MESSAGE15 RULE-SYNTAX-ERROR-MESSAGE16
              RULE-SYNTAX-ERROR-MESSAGE17 RULE-SYNTAX-ERROR-MESSAGE18
              RULE-SYNTAX-ERROR-MESSAGE19 RULE-SYNTAX-ERROR-MESSAGE20
              RULE-SYNTAX-ERROR-MESSAGE21 RULE-SYNTAX-ERROR-MESSAGE22
              RULE-SYNTAX-ERROR-MESSAGE23 RULE-SYNTAX-ERROR-MESSAGE24
              RULE-SYNTAX-ERROR-MESSAGE25 RULE-SYNTAX-ERROR-MESSAGE26
              RULE-SYNTAX-ERROR-MESSAGE27 RULE-SYNTAX-ERROR-MESSAGE28
              RULE-SYNTAX-ERROR-MESSAGE29 RULE-SYNTAX-ERROR-MESSAGE30
              RULE-SYNTAX-ERROR-MESSAGE31 RULE-SYNTAX-ERROR-MESSAGE32
              RULE-SYNTAX-ERROR-MESSAGE33 RULE-SYNTAX-ERROR-MESSAGE34)
```

```
(INITIALIZE-NETWORK-SYNTAX-ERROR-MESSAGES)
(INITIALIZE-RULE-SYNTAX-ERROR-MESSAGES)

(SETQ ENGLISH-CLAUSE '(THE BOY SAT))
(SETQ REMAINING-WORDS ENGLISH-CLAUSE)

(DEFPROP A (DETERMINER INDEFINITE SINGULAR) FEATURES)
(DEFPROP AN (DETERMINER INDEFINITE SINGULAR) FEATURES)
(DEFPROP APPLE (NOUN SINGULAR) FEATURES)
(DEFPROP ATE (VERB TENSED) FEATURES)
(DEFPROP ATE EAT ROOT)
(DEFPROP BIG (ADJECTIVE) FEATURES)
(DEFPROP BOY (NOUN SINGULAR) FEATURES)
(DEFPROP EAT (TRANSITIVE) FEATURES)
(DEFPROP EATEN (PASTPARTICIPLE) FEATURES)
(DEFPROP EATS (VERB TENSED) FEATURES)
(DEFPROP EATS EAT ROOT)
(DEFPROP RED (ADJECTIVE) FEATURES)
(DEFPROP SAT (VERB TENSED) FEATURES)
(DEFPROP SAT SIT ROOT)
(DEFPROP SIT (INTRANSITIVE) FEATURES)
(DEFPROP THE (DETERMINER DEFINITE SINGULAR) FEATURES)
(DEFPROP WAS (VERB TENSED) FEATURES)
(DEFPROP WAS BE ROOT)
```

```
(RECORD-PARSE-WORD)

(COND ((NOT ERROR-EXISTS-IN-NETWORK-SYNTAX)
       (RECORD-PARSE-NOUN-GROUP))
      (T (GO PRINT-FINAL-ERROR-MESSAGE)))

(COND ((NOT ERROR-EXISTS-IN-NETWORK-SYNTAX)
       (RECORD-PARSE-CLAUSE))
      (T (GO PRINT-FINAL-ERROR-MESSAGE)))

(COND ((NOT ERROR-EXISTS-IN-NETWORK-SYNTAX)
       (PRINT 'THE-ENGLISH-CLAUSE-TO-BE-PARSED-IS)
       (TERPRI)
       (PRINT ENGLISH-CLAUSE)
       (TERPRI) (TERPRI) (TERPRI) (TERPRI) (TERPRI)
       (PRINT 'THE-RESULTS-OF-THE-INTERPRETING-PROCESS-ARE)
       (TERPRI)

       (PARSE-CLAUSE 'ROOT NIL))
      (T (GO PRINT-FINAL-ERROR-MESSAGE)))

PRINT-FINAL-ERROR-MESSAGE
(COND (ERROR-EXISTS-IN-NETWORK-SYNTAX
       (PRINT 'THE-FINAL-NETWORK-SYNTAX-ERROR-MESSAGE-IS)
       (TERPRI)
       (PRINT '(FINAL ERROR MESSAGE *** THE ENGLISH CLAUSE SYNTAX
                                        IDENTIFICATION PROCESS HAS
                                        BEEN SUSPENDED DUE TO
                                        AUGMENTED TRANSITION
                                        NETWORK SYNTAX ERRORS))))
                                                              ))
```

APPENDIX C

THE AUGMENTED TRANSITION NETWORK (ATN) COMPILER COMPONENT DESCRIPTION

The following constitute the description of the routines and the local data for the ATN Compiler.

ROUTINE: CADR

OBJECTIVE: CADR is routine that takes a list as its argument and evaluates it according to the following sequence: CDR, CAR.

FUNCTIONS USED: CAR, CDR

BOUND VARIABLE: Listl represents the list to be evaluated.

FREE VARIABLES: None

ROUTINE: CADDDR

OBJECTIVE: CADDDR is a routine that takes a list as its argument and evaluates it according to the following sequence: CDR, CDR, CDR, CAR.

FUNCTIONS USED: CAR, CDR

BOUND VARIABLE: Listl represents the list to be evaluated.

FREE VARIABLES: None

ROUTINE: CDDR

OBJECTIVE: CDDR is a routine that takes a list as its argument and evaluates it according to the following sequence: CDR, CDR.

FUNCTION USED: CDR

BOUND VARIABLE: Listl represents the list to be evaluated.

FREE VARIABLES: None

ROUTINE: CDDDDR

OBJECTIVE: CDDDDR is a routine that takes a list as its argument and evaluates it according to the following sequence: CDR, CDR, CDR, CDR.

FUNCTION USED: CDR

BOUND VARIABLE: Listl represents the list to be evaluated.

FREE VARIABLES: None

ROUTINE: GETF

OBJECTIVE: GETF is a routine that fetches the property value of the property name FEATURES. The classification type of a word appears on the property list of the word under the property name FEATURES.

FUNCTION USED: GET

BOUND VARIABLE: The-Features represents the property value of the property name FEATURES.

FREE VARIABLES: None

ROUTINE: INTERSECTION

OBJECTIVE: INTERSECTION accepts two lists and determines a list containing only the elements that are in both of the two lists.

FUNCTIONS USED: COND, NULL, MEMBER, CAR, CONS, INTERSECTION, CDR

BOUND VARIABLES: List1 represents a list of s-expressions.

List2 represents a list of s-expressions.

FREE VARIABLES: None

ROUTINE: TESTF

OBJECTIVE: TESTF returns T only if a given node has all of the given features. TESTF can handle either a single feature or a list of features.

FUNCTIONS USED: COND, NULL, ATOM, SETQ, LIST, EQUAL, LENGTH, INTERSECTION, GETF

BOUND VARIABLES: Node represents a word whose features are to be examined.

Features represents the characteristics of a given word.

FREE VARIABLES: None

ROUTINE: PRINT-THE-PARENT-AND-THE-CHILDREN

OBJECTIVE: PRINT-THE-PARENT-AND-THE-CHILDREN outputs the property value of the child and the property value of the parent.

FUNCTIONS USED: TERPRI, PRINØ, XTAB, PRINT, GET

BOUND VARIABLES: None

FREE VARIABLES: The-Child, The-Parent

ROUTINE: ATTACH

OBJECTIVE: ATTACH is a routine that connects nodes by way of placing appropriate values on a property list.

FUNCTIONS USED: PUTPROP, APPEND, GET, LIST, SETQ, PRINT-THE-PARENT-AND-THE-CHILDREN

BOUND VARIABLES: The-Parent represents the atom name of the property value The-Children and the property name Children.

The-Child represents the atom name of the property value The-Parent and the property name Parent.

The-Children represents a list of property values.

FREE VARIABLES: None

ROUTINE: SELECT

OBJECTIVE: SELECT is a recursive routine that takes two lists as arguments and returns the first thing in the in the first list that is also in the second.

FUNCTIONS USED: COND, NULL, MEMBER, CAR, SELECT

BOUND VARIABLES: S-expr1 represents a symbolic expression.

S-expr2 represents a symbolic expression.

FREE VARIABLES: None

ROUTINE: GENNAME

OBJECTIVE: GENNAME is a routine that generates a unique new atom each time it is evaluated. When the value of the argument of GENNAME is, say, Parse-Noun-Group, then the new atom is of the form Parse-Noun-Group-n where the number n increases by one each time the function is used with Parse-Noun-Group as the value of its argument. GENNAME is not a standard LISP function, however, it is defined using the LISP functions IMPLODE, EXPLODE, and GENSYM.

FUNCTIONS USED: IMPLODE, APPEND, EXPLODE, CDDDDR, GENSYM

BOUND VARIABLE: Name represents the atom to be connected into a unique new atom.

FREE VARIABLES: None

ROUTINE: PRINT-THE-REGISTER-AND-THE-VALUE-OF-THIS-NODE

OBJECTIVE: PRINT-THE-REGISTER-AND-THE-VALUE-OF-THIS-NODE outputs the register name and its corresponding content with respect to an Augmented Transition Network node.

FUNCTIONS USED: PRINØ, XTAB, PRINT

BOUND VARIABLES: None

FREE VARIABLES: Register, Value

ROUTINE: SETR

OBJECTIVE: SETR is a routine that places the value of a node into a register. It is accomplished by placing the property value of an atom name into a property name. SETR uses This-Node as a free variable.

FUNCTION USED: PUTPROP, PRINT-THE-REGISTER-AND-THE-VALUE-OF-THIS-NODE

BOUND VARIABLE: Register represents a property name.

Value represents a property value.

FREE VARIABLE: This-Node

ROUTINE: GETR

OBJECTIVE: GETR is a routine that retrieves the value of a node from a register. It is accomplished by fetching the property value from its property name.

FUNCTION USED: GET

BOUND VARIABLE: Register represents a property name

FREE VARIABLE: This-Node

ROUTINE: ADDR

OBJECTIVE: ADDR is a routine that uses SETR and GETR to add a new element to a register.

FUNCTIONS USED: SETR, CONS, GETR

BOUND VARIABLES: Register represents a property name.

Value represents a property value.

FREE VARIABLES: None

ROUTINE: COMPILE

OBJECTIVE: COMPILE is a MACRO routine that represents the Augmented Transition Network (ATN) compiler function.

FUNCTIONS USED: SETQ, CADR, CDDR, SUBST, EVAL, CONS, MAPCAR, APPENDM, LIST, PRINT, TERPRI, SP, RETURN

BOUND VARIABLES: Description represents the ATN description to be compiled.

Name represents either Parse-Word, Parse-Noun-Group, or Parse-Clause.

Body represents the ATN description minus the ATN name.

Program represents the compiled ATN description.

Beginning represents the initial block of the compiled ATN description.

Middle represents the middle block of the compiled ATN description.

End represents the terminal block of the compiled ATN description.

The-Compiled-ATN represents the function definition bearing the compiled ATN description.

FREE VARIABLES: None

ROUTINE: COMPILE-PARSE-WORD

OBJECTIVE: COMPILE-PARSE-WORD is a routine that contains the ATN description Parse-Word. It invokes the MACRO routine COMPILE in order to translate the ATN description into a form that LISP understands directly.

FUNCTION USED: COMPILE

BOUND VARIABLES: None

FREE VARIABLES: None

ROUTINE: COMPILE-PARSE-NOUN-GROUP

OBJECTIVE: COMPILE-PARSE-NOUN-GROUP is a routine that contains the ATN description Parse-Noun-Group. It invokes the MACRO routine COMPILE in order to translate the ATN description into a form that LISP understands directly.

FUNCTION USED: COMPILE

BOUND VARIABLES: None

FREE VARIABLES: None

ROUTINE: COMPILE-PARSE-CLAUSE

OBJECTIVE: COMPILE-PARSE-CLAUSE is a routine that contains the ATN description Parse-Clause. It invokes the MACRO routine COMPILE in order to translate the ATN description into a form that LISP understands directly.

FUNCTION USED: COMPILE

BOUND VARIABLES: None

ROUTINE: IDENTIFY

OBJECTIVE: IDENTIFY is the main routine called by the user to execute the ATN Compiler and the ATN Interpreter. It contains a dictionary where English words are listed alphabetically and each property of each word is defined. The routine IDENTIFY states the English Clause to be parsed by the ATN Interpreter.

FUNCTIONS USED: SETQ, DEFPROP, COMPILE-PARSE-WORD, COMPILE-PARSE-NOUN-GROUP, COMPILE-PARSE-CLAUSE, PRINT, TERPRI, PARSE-CLAUSE

BOUND VARIABLES: Remaining-Words represents a list of words remaining to be analyzed.

Current-Word represents the current word which is being analyzed.

Last-Parsed represents the word most recently parsed.

English-Clause represents the English clause to be parsed by the ATN Interpreter.

FREE VARIABLES: None

APPENDIX D

THE AUGMENTED TRANSITION NETWORK (ATN) COMPILER SOURCE CODE

The following is the LISP source code for the ATN Compiler to translate an ATN description.

```
(DEFUN CADR (LIST1)
       (CAR (CDR LIST1)))

(DEFUN CADDDR (LIST1)
       (CAR (CDR (CDR (CDR LIST1)))))

(DEFUN CDDR (LIST1)
       (CDR (CDR LIST1)))

(DEFUN CDDDDR (LIST1)
       (CDR (CDR (CDR (CDR LIST1)))))

(DEFUN GETF (THE-FEATURES)
       (GET THE-FEATURES 'FEATURES))
```

```
(DEFUN INTERSECTION (LIST1 LIST2)

        (COND ((NULL LIST1) NIL)
              ((MEMBER (CAR LIST1) LIST2)
               (CONS (CAR LIST1) (INTERSECTION (CDR LIST1) LIST2)))
              (T (INTERSECTION (CDR LIST1) LIST2))))

                        .

                        .


(DEFUN TESTF (NODE FEATURES)
                        .
        (COND ((NULL FEATURES))
              ((ATOM FEATURES)
               (SETQ FEATURES (LIST FEATURES))))
        (EQUAL (LENGTH FEATURES)
               (LENGTH (INTERSECTION FEATURES (GETF NODE)))))



(DEFUN PRINT-THE-PARENT-AND-THE-CHILDREN ()

        (TERPRI) (TERPRI)
        (PRINC 'THE-PARENT-IS)
        (XTAB 7)
        (PRINT (GET THE-CHILD 'PARENT))
        (PRINC 'THE-CHILD-IS)
        (XTAB 8)
        (PRINT (GET THE-PARENT 'CHILDREN)))
```

```
(DEFUN ATTACH (THE-CHILD THE-PARENT)
       (PROG (THE-CHILDREN)

             (PUTPROP THE-CHILD THE-PARENT 'PARENT)
             (SETQ THE-CHILDREN
                   (APPEND (GET THE-PARENT 'CHILDREN) (LIST THE-CHILD)))
             (PUTPROP THE-PARENT THE-CHILDREN 'CHILDREN)
             (PRINT-THE-PARENT-AND-THE-CHILDREN)))



(DEFUN SELECT (S-EXPR1 S-EXPR2)

       (COND ((NULL S-EXPR1) NIL)
             ((MEMBER (CAR S-EXPR1) S-EXPR2) (CAR S-EXPR1))
             (T (SELECT (CDR S-EXPR1) S-EXPR2))))



(DEFUN GENNAME (NAME)

       (IMPLODE (APPEND (EXPLODE NAME)
                        (CDDDDR (EXPLODE (GENSYM NAME))))))
```

```
(DEFUN PRINT-THE-REGISTER-AND-THE-VALUE-OF-THIS-NODE ()

        (PRINO 'THE-REGISTER-IS)
        (XTAB 5)
        (PRINT REGISTER)
        (PRINO 'THE-VALUE-IS)
        (XTAB 8)
        (PRINT VALUE))



(DEFUN SETR (REGISTER VALUE)

        (PUTPROP THIS-NODE VALUE REGISTER)
        (PRINT-THE-REGISTER-AND-THE-VALUE-OF-THIS-NODE)
        VALUE)



(DEFUN GETR (REGISTER)

        (GET THIS-NODE REGISTER))



(DEFUN ADDR (REGISTER VALUE)

        (SETR REGISTER (CONS VALUE (GETR REGISTER))))
```

```
(DEFUN COMPILE MACRO (DESCRIPTION)
  (PROG (NAME BODY PROGRAM BEGINNING MIDDLE END THE-COMPILED-ATN)

        (SETQ NAME (CADR DESCRIPTION))
        (SETQ BODY (CDDR DESCRIPTION))

        (SETQ BEGINNING
              (SUBST NAME
                     'REPLACE
                     '(PROG (THIS-NODE HOLD)
                            (SETQ HOLD REMAINING-WORDS)
                            (SETQ CURRENT-WORD (CAR REMAINING-WORDS))
                            (SETQ THIS-NODE (GENNAME 'REPLACE)))))

        (SETQ
         MIDDLE
         (EVAL
          (CONS 'APPENDM
          (MAPCAR
           '(LAMBDA
             (STATE)
             (LIST 'QUOTE (LIST (CAR STATE)
                   (CONS 'COND
                         (APPEND
                          (MAPCAR
                           '(LAMBDA
                             (CLAUSE)
                             (APPENDM (LIST (CADR CLAUSE))
                                      (COND ((CDDDDR CLAUSE)
                                             (CDR (CDDDDR CLAUSE))))
                                      (LIST (LIST 'GO (CADDDR CLAUSE)))))
                           (CDR STATE))
                          '((T (GO LOSE)))))))))
            BODY))))
```

```
        (SETQ END
              '(WIN (COND ((NOT (TESTF THIS-NODE FEATURES)) (GO LOSE)))
                    (ATTACH THIS-NODE PARENT-NODE)
                    (SETQ LAST-PARSED THIS-NODE)
                    (RETURN THIS-NODE)
                    LOSE
                    (SETQ REMAINING-WORDS HOLD)
                    (SETQ CURRENT-WORD (CAR REMAINING-WORDS))
                    (RETURN NIL)))

        (SETQ PROGRAM (APPENDM BEGINNING MIDDLE END))

        (SETQ THE-COMPILED-ATN
              (LIST 'DEFUN NAME '(PARENT-NODE FEATURES) PROGRAM))
        (PRINT 'THE-COMPILED-AUGMENTED-TRANSITION-NETWORK-IS)
        (TERPRI)
        (SP THE-COMPILED-ATN)
        (TERPRI) (TERPRI) (TERPRI)

        (RETURN THE-COMPILED-ATN)))


(DEFUN COMPILE-PARSE-WORD ()

       (COMPILE PARSE-WORD
                (SI (IF T >>> WIN
                        AFTER
                        (SETQ THIS-NODE CURRENT-WORD)
                        (SETQ REMAINING-WORDS (CDR REMAINING-WORDS))
                        (COND (REMAINING-WORDS
                               (SETQ CURRENT-WORD
                                     (CAR REMAINING-WORDS)))
                              (T (SETQ CURRENT-WORD NIL))))))
                                                          )
```

```
(DEFUN COMPILE-PARSE-NOUN-GROUP ()

       (COMPILE PARSE-NOUN-GROUP
                (S1 (IF (PARSE-WORD THIS-NODE 'DETERMINER)
                        >>> S2
                        AFTER
                        (SETR 'NUMBER (SELECT '(SINGULAR PLURAL)
                                              (GETF LAST-PARSED)))
                        (SETR 'DETERMINER (SELECT '(DEFINITE INDEFINITE)
                                                  (GETF LAST-PARSED)))))
                (S2 (IF (PARSE-WORD THIS-NODE 'ADJECTIVE)
                        >>> S2
                        AFTER
                        (ADDR 'ADJECTIVES LAST-PARSED))
                    (IF (PARSE-WORD THIS-NODE 'NOUN)
                        >>> WIN
                        AFTER
                        (SETR 'NUMBER (SELECT '(SINGULAR PLURAL)
                                              (GETF LAST-PARSED)))
                        (SETR 'NOUN LAST-PARSED))))
                                                   )
```

```
(DEFUN COMPILE-PARSE-CLAUSE ()

       (COMPILE PARSE-CLAUSE
             (S1 (IF (PARSE-NOUN-GROUP THIS-NODE NIL)
                     >>> S2
                     AFTER
                     (SETR 'SUBJECT LAST-PARSED)))
             (S2 (IF (PARSE-WORD THIS-NODE '(VERB TENSED))
                     >>> S3
                     AFTER (SETR 'VERB (GET LAST-PARSED 'ROOT))))
             (S3 (IF (AND (EQUAL (GETR 'VERB) 'BE)
                          (PARSE-WORD THIS-NODE 'PASTPARTICIPLE))
                     >>> S4
                     AFTER
                     (SETR 'OBJECT (GETR 'SUBJECT))
                     (SETR 'SUBJECT NIL)
                     (SETR 'VERB LAST-PARSED))
                 (IF (AND (TESTF (GETR 'VERB) 'TRANSITIVE)
                          (PARSE-NOUN-GROUP THIS-NODE NIL))
                     >>> S4
                     AFTER (SETR 'OBJECT LAST-PARSED))
                 (IF (OR (TESTF (GETR 'VERB) 'INTRANSITIVE)
                         (GETR 'OBJECT))
                     >>> S4))
             (S4 (IF (AND (GETR 'SUBJECT)
                          (NULL REMAINING-WORDS))
                     >>> WIN)
                 (IF (AND (NOT (GETR 'SUBJECT))
                          (EQUAL CURRENT-WORD 'BY)
                          (PARSE-WORD THIS-NODE NIL))
                     >>> S5)
                 (IF (NOT (GETR 'SUBJECT))
                     >>> S4
                     AFTER
                     (SETR 'SUBJECT 'SOMEONE)))
             (S5 (IF (PARSE-NOUN-GROUP THIS-NODE NIL)
                     >>> S4
                     AFTER
                     (SETR 'SUBJECT LAST-PARSED))))
                                                    )
```

```
(SETQ *OTLL* 112)
(SETQ *LEVEL* 1000)
(SETQ *LINE-LENGTH* 120)
(SETQ *SCREEN-LENGTH* 1000)
```

```
(DEFUN IDENTIFY ()
       (PROG (REMAINING-WORDS CURRENT-WORD LAST-PARSED ENGLISH-CLAUSE)

             (SETQ ENGLISH-CLAUSE '(THE BIG BOY ATE THE RED APPLE))
             (SETQ REMAINING-WORDS ENGLISH-CLAUSE)

             (DEFPROP A (DETERMINER INDEFINITE SINGULAR) FEATURES)
             (DEFPROP AN (DETERMINER INDEFINITE SINGULAR) FEATURES)
             (DEFPROP APPLE (NOUN SINGULAR) FEATURES)
             (DEFPROP ATE (VERB TENSED) FEATURES)
             (DEFPROP ATE EAT ROOT)
             (DEFPROP BIG (ADJECTIVE) FEATURES)
             (DEFPROP BOY (NOUN SINGULAR) FEATURES)
             (DEFPROP EAT (TRANSITIVE) FEATURES)
             (DEFPROP EATEN (PASTPARTICIPLE) FEATURES)
             (DEFPROP EATS (VERB TENSED) FEATURES)
             (DEFPROP EATS EAT ROOT)
             (DEFPROP RED (ADJECTIVE) FEATURES)
             (DEFPROP SAT (VERB TENSED) FEATURES)
             (DEFPROP SAT SIT ROOT)
             (DEFPROP SIT (INTRANSITIVE) FEATURES)
             (DEFPROP THE (DETERMINER DEFINITE SINGULAR) FEATURES)
             (DEFPROP WAS (VERB TENSED) FEATURES)
             (DEFPROP WAS BE ROOT)

             (COMPILE-PARSE-WORD)
             (COMPILE-PARSE-NOUN-GROUP)
             (COMPILE-PARSE-CLAUSE)

             (PRINT 'THE-ENGLISH-CLAUSE-TO-BE-PARSED-IS)
             (TERPRI)
             (PRINT ENGLISH-CLAUSE)
             (TERPRI) (TERPRI) (TERPRI) (TERPRI) (TERPRI)
             (PRINT 'THE-RESULTS-OF-THE-INTERPRETING-PROCESS-ARE)
             (TERPRI)

             (PARSE-CLAUSE 'ROOT NIL)))
```

NATURAL LANGUAGE ANALYSIS VIA AUGMENTED TRANSITION NETWORKS (ATN)

by

JOSE MA. U. LAZARO, JR.

B.S. in Physics, Ateneo de Manila University, Philippines, 1978

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

in

COMPUTER SCIENCE

Department of Computer Science

Kansas State University
Manhattan, Kansas

1982

ABSTRACT

This report serves as documentation of a project which implements in LISP an Interpreter and a Compiler for Augmented Transition Networks (ATN) based on Chapters 19 and 20 of the book entitled "LISP", authored by Patrick Henry Winston and Berthold Klaus Paul Horn. The ATN Syntax Analyzer has been designed and implemented as an auxiliary feature to the ATN Interpreter. ATN Syntax Error Recovery routines have been included to test subsequent ATN syntax errors and possibly make syntax corrections in order to save the entire ATN.

Chapter 19 of "LISP", entitled "Interpreting Augmented Transition Networks", examines an Interpreter for ATN descriptions to parse certain English sentences. The implementation of an ATN Interpreter involves auxiliary functions to capture the idea of ATN node traversal in order to recognize word classes of an English sentence. Registers which are represented as properties of the corresponding node are responsible for taking notes of ATN descriptions and referring to them later.

The ATN Interpreter has demonstrated that ATNs certainly capture aspects of English Syntax. LISP has made the understanding of ATNs an easy process. Lastly, registers have added power to ATN descriptions.

The ATN Syntax Analyzer, attached to the ATN Interpreter, consists of a hierarchy of routines to facilitate a systematic

way of investigating syntax errors. A significant operation which is Error Recovery has been included so that subsequent syntax errors can be identified and that the ATN need not be discarded immediately. The ATN Syntax Analyzer's Error Recovery System may, for instance, decide to insert missing ATN syntactic components or to replace invalid ATN syntactic components with the proper ATN syntactic components. The methods of properly identifying, diagnosing, and possibly correcting any ATN syntax errors are accomplished by the availability of layers of Error Recovery routines. In that sense, the ATN Syntax Analyzer supports abstraction and hierarchical structuring of the routines very systematically.

Chapter 20 of "LISP", entitled "Compiling Augmented Transition Networks", examines a Compiler for ATNs. The ATN Compiler translates source-language descriptions into descriptions that LISP understands directly by ripping apart a transition-specifying parcel and reassembling it into a COND clause. Several clauses for each ATN node are amalgamated into a single COND structure where CARs, CDRs, CONS, and APPENDs abound.

Compiling is a symbol-manipulating task and in that respect, LISP has proven to be eminently suited. The ATN Compiler complements the ATN Interpreter since it has made the ATN description more readable to readers who are well-acquainted with LISP.