FINITE-STATE GRAMMATICAL MODEL AND PARSER FOR AIR TRAFFIC CONTROLLER'S COMMANDS

Jorge L. Ortiz, Ph.D, PE
Electrical & Computer Engineering Department
College of Engineering
University of Puerto Rico – Mayaguez, P.O.Box 9042
Mayaguez, Puerto Rico 00681-9042

ABSTRACT

This paper presents a grammatical model for the air traffic controller's (ATC) commands using finite-state transition networks (FSTN). The grammatical representation is used by a syntactic parser and recognizer for the analysis of the grammatical structure of the commands. A grammatical description using FSTN is proposed for the ATC's commands assigning word categories and syntactic structure that can be followed by a syntactic parser for recognition and parsing.

This paper, also, presents an innovative model using "skip loops" for the implementation of a syntactic parser using finite-state transition networks to delete and remove incorrect or out of syntax words. These words could be the effect of mixed streams of words or errors in the conversion from spoken language to characters. The skip loop is an arc that allows the finite-state transition automata (FSTA) to delete a word that does not match the grammatical structure of the sentence, and continue the recognition process without affecting the syntactical definition of the sentence. This particular approach is especially useful in areas such as the command language of the air traffic controller (ATC)

The model uses FSTN with skip loops to model and recognize ATC's command language. The use of skip loops allows to delete words that may be present in the statement that are unrecognized or that do not fit into the grammatical structure of the ATC's language. This technique facilitates the recognition of the statement minimizing the possibility of declaring the statement as ill-formed. Two syntactic parser prototypes are implemented using Prolog and CLIPS.

These techniques are useful in applications like military tactical environments that are exposed to rapidly changing commands, streams of information, and different sources of background noise. Many critical decisions have to be made extracting the correct information from multiple input streams making difficult and uncertain the selection of the correct input information. The method presented introduces a certain degree of intelligence using current AI techniques to obtain an intelligent syntactic parsing of the input information. The parser syntax can be defined to dynamically adjust its model to follow a particular stream of information that sounds or looks appropriate for the particular context. The purpose of the parser will be to model the process that resembles the human ability to follow a single dialog in an environment where there are many conversations and background noise.

AUTHOR'S BIOGRAPHY

Jorge L. Ortiz is professor at the Electrical & Computer Engineering Department at University of Puerto Rico-Mayaguez. He earned a Ph.D. in Electrical Engineering from University of Houston, TX. He has occupied positions like Associate Dean of Engineering and Associate Director of the ECE Department. Dr. Ortiz has participated in research projects with the Naval Air Warfare Center - TSD in Orlando, Florida. His current research interests are artificial intelligence, natural language processing, and expert systems.

FINITE-STATE GRAMMATICAL MODEL AND PARSER FOR AIR TRAFFIC CONTROLLER'S COMMANDS

Jorge L. Ortiz, Ph.D, PE
Electrical & Computer Engineering Department
College of Engineering
University of Puerto Rico – Mayaguez, P.O. Box 9042
Mayaguez, Puerto Rico 00681-9042

INTRODUCTION

Applications like military tactical environments are exposed to rapidly changing commands, streams of information, and different sources of background noise. Many critical decisions have to be made extracting the correct information from multiple input streams making difficult and uncertain the selection of the correct input information. The method presented introduces a certain degree of intelligence using current AI techniques to obtain an intelligent syntactic parsing of the input information. The parser syntax can be defined to change or adjust its model to follow a specific stream of information that sounds or looks appropriate for the particular context. The aim of the parser will be to model the process that resembles the human ability to follow a single dialog in an environment where there are many ongoing conversations.

A finite-state transition automata (FSTA) is used in applications such as language parsing by describing the grammar structure using finite-state transition networks. Parsers are classified in two main types as syntactic or semantic (Cole 1996). Syntactic parsers describe and determine the different combinations of textual elements in a sentence, while the semantic parser works with the interpretation or meaning of the sentence. The FSTN allows describing the syntactic structure of the grammar in any language such as English.

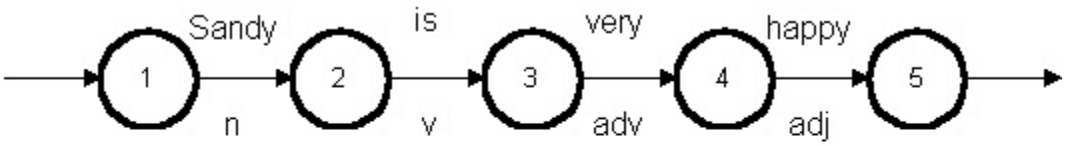
The process by which the syntactic structure of an expression is determined is known as syntactic parsing (Mathews 1998). Syntactic parsers manipulate the declarative knowledge of the grammar to determine if a sentence is correct or not. Recognition is the process of identifying a string of words as syntactically well formed. Parsing goes beyond the recognition process by associating a syntactic structure to those expressions that have been recognized. In summary, recognition and parsing are processes that determine whether a particular sentence or stream of words is a valid expression or not.

The syntactic parsing process replaces each word by its lexical category and checks if the transformed stream corresponds to one of the possible grammatically correct sentences. Using this process the input stream will be accepted as a well formed or as ill-formed sentence. The simplest parsing procedure could be to list all possible sentences and checks any input stream with all the possible sentence combinations on the list. Of course, there is infinite number of combinations and the task is impractical to achieve. There are several structures or techniques (Cole 1996) to represent grammars such as functional Unification Grammar, Tree Adjunction Grammar, Lexicon Functional Grammar, and Finite-State Grammar. The finite–state grammar model is a very flexible technique that uses finite-state transition networks to define the structure of the grammar in a graphical way and later translated to a logic program as Prolog language for its implementation.

SKIP LOOPS SYNTACTIC PARSER

What happens when the sentence has other words that do not match the structure of the grammar described by the FSTN? It will not be recognized and classified as ill-formed statement. Assume there is a language recognition system that translates spoken language to text and that for any reason the system catches a word out of context from a different speaker and it is mixed

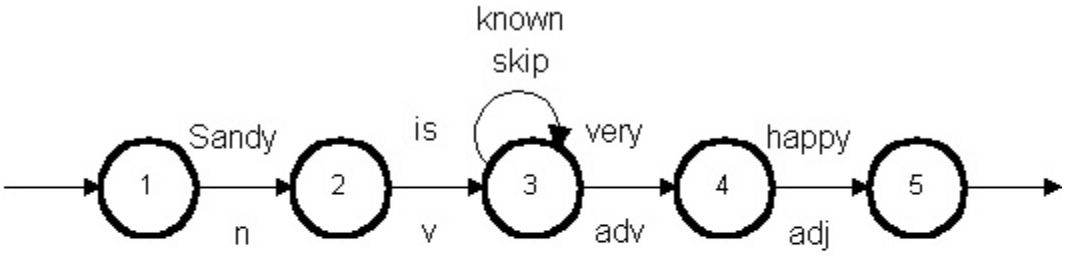
with the stream of word that is desired to process. For example, "sandy is very happy". The finite-state transition diagram for this statement grammar is shown below (see Figure 1). The syntactic structure of the statement shows a noun (n), verb (v), adverb (adv), and adjective (adj). The statement is correct according to its definition in the FSTN.



Figue 1. Finite -State Transition Diagram for the Statement "Sandy is very happy."

Assume you get an error in the conversion from spoken language to text. The statement that you get from the converter could be: "sandy is known very happy." The word "known" could be an error in the conversion from spoken language to text or could be a word from another speaker mixed with the original stream of words. Using the grammar described above (Figure 1), the sentence will be classified as ill formed as it really is. However, it is a desirable characteristic of the recognizer/parser system to be able to delete words that do not belong to the specific application. In situations that the word that do not match the defined structure of the sentence it can be recognized and parsed successfully if the word "known" is ignored or "skipped". Humans hear statements that may have inserted words that do not match the context of the statement and are able to filter the unnecessary word and process the information using the correct statement. This paper proposes to use "skip loops" embedded into the finite-state transition grammar to skip words that do not match the structure of the grammar. The skip loop is an arc that allows the FSTA to consume a word that does not match the grammatical structure of the sentence or may be is an unknown word, and continues the recognition of the input stream without affecting the syntactical definition of the sentence. This particular approach is useful in specialized areas such as the air traffic controller's environment where a special technical language is used. The illustration below (see Figure 2) shows how a skip loop will be positioned in node 3 to skip the word "known" and continue with the remaining of the sentence and completing its recognition.

Skip loops are added to all the nodes in the diagram to be able to skip words out of context in any position in the sentence. In this way a sentence like: "Sandy not is away very what happy" can be recognized if the FSTA



Figue 2. Finite-State Transition Diagram for the Statement "Sandy is **known** very happy."

skips the words that do not belong to the configuration of the network or at the same time it does not match the grammatical construction of the defined syntax. This allows the syntactic parser to fill in all the grammatical word positions and obtain a well-formed statement.

AIR TRAFFIC CONTROLLER'S FINITE-STATE GRAMMAR MODELS AND PARSER

The air traffic controllers' commands to direct the commercial and military air traffic are part of a very specialized language. Traffic controllers use a combination of English language words and numbers that do not math the formal English grammar. It is a specialized command language where names and measurements are spoken without using determinants, conjunctions, and other syntactic parts of the English language. Some reasons for this special language is to make the communication between the air traffic controller and the pilot of the aircraft strait forward, short, and only including the necessary information to understand commands without including redundant or unnecessary words to interpret the message that wants to be delivered. This can be classified as a task-oriented grammar, however there are thousands of different statements that the ATC can generate, but with the use of finite-state transition networks a grammatical structure for the commands is described in this paper to identify all the traffic controllers commands. Looking at a sample of ATC's statements there are different grammatical structures that they use for sending information to the pilots. Here are some examples of the statements used:

1. Four one one five miles.
2. Seven one six final radar contact ten miles.
3. Diamond six zero three four point zero miles.
4. Six zero zero begin descent.
5. Wolf one zero six on course.

The above statements may have little or no meaning to common people, but for air traffic controllers, these are the statements they use to communicate information to he aircraft pilots. Trying to identify a grammatical structure or syntax for these sentences, syntax categories like subject in sentences 3 and 5 can be recognized. These are the words "Diamond" and "Wolf" that are used as the names assigned to each aircraft. After the subject, a series of three digit numbers are used for identification purposes of what have been called the subject. These three digits are used as a numerical identification of the aircraft and will be defined in this paper as "aircraft identification number" (idn). After the subject and the identification numbers, a command or series of command words (cw) completes the statement like in sentences 4 and 5. In others cases, a command word or a series of command words are followed by a measurement (mw). Statement number 2 is an example of this case. Here, a series of one to three numerals or a fraction are mentioned followed by some kind of unit, like miles in this case. The syntactic structure of the statements are represented using FSTN (see Figures 4 to 8).

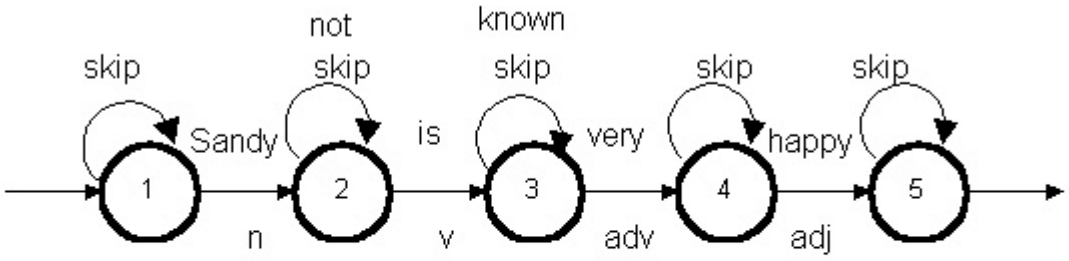


Figure 3. Finite-State Transition Diagram for the Statement "Sandy **not** is **known** very happy."

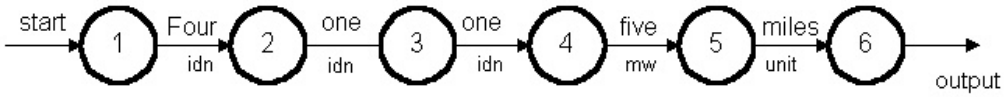


Figure 4. Finite-State Transition Diagram for the Statement "Four one one five miles."

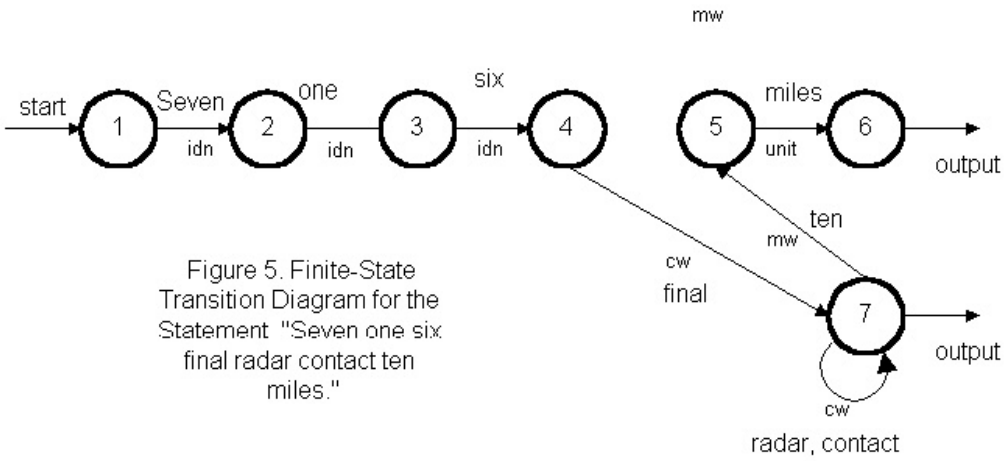


Figure 5. Finite-State Transition Diagram for the Statement "Seven one six final radar contact ten miles."

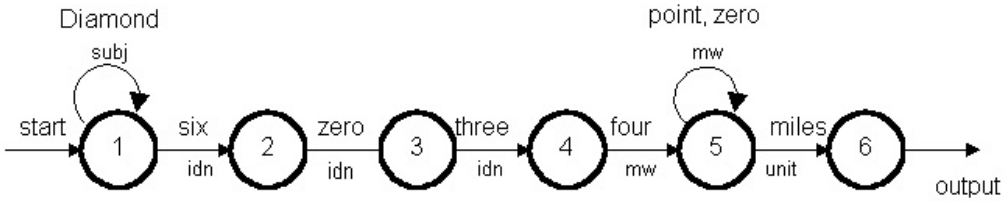


Figure 6. Finite-State Transition Diagram for the Statement "Six zero three four miles."

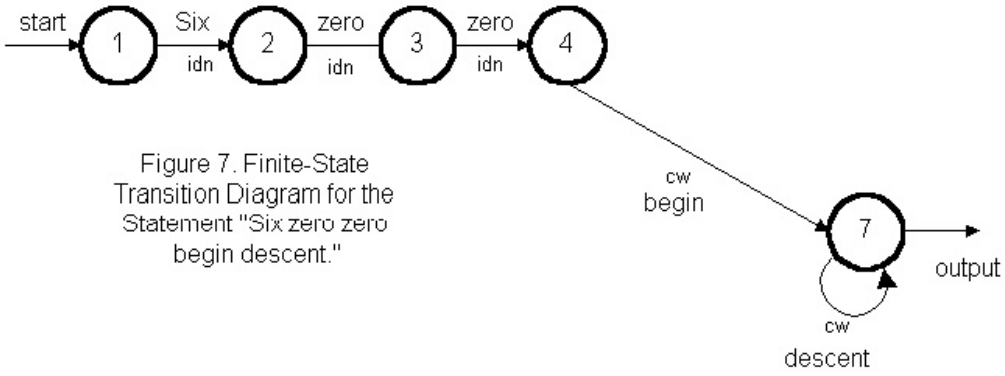


Figure 7. Finite-State Transition Diagram for the Statement "Six zero zero begin descent."

RESULTS

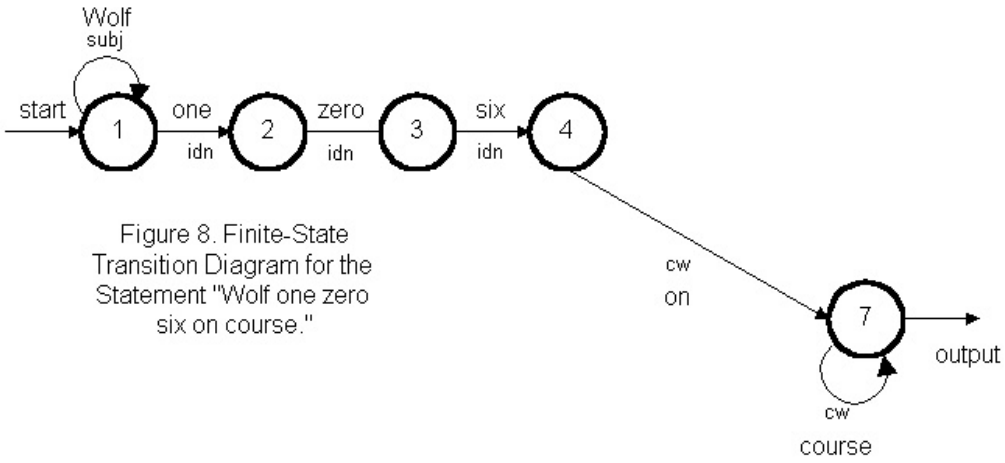


Figure 8. Finite-State Transition Diagram for the Statement "Wolf one zero six on course."

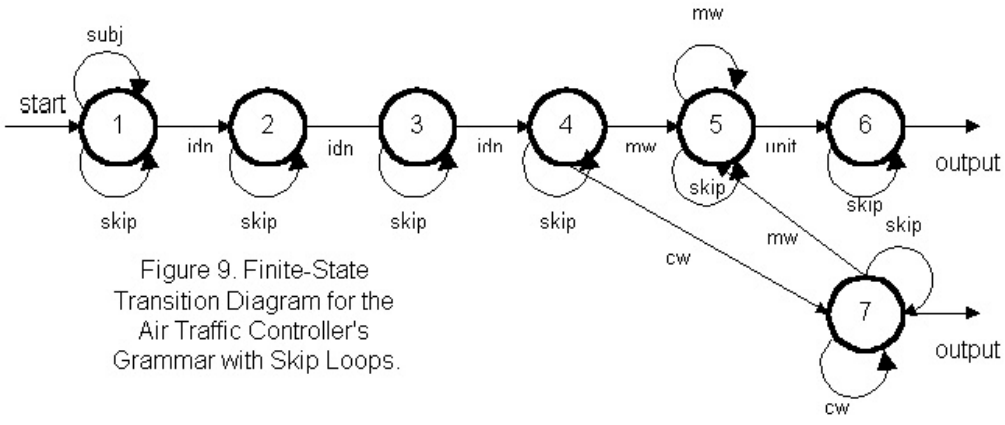


Figure 9. Finite-State Transition Diagram for the Air Traffic Controller's Grammar with Skip Loops.

Each of these statements has a particular finite-state transition network that describes its grammatical structures. These FSTN diagrams can be combined to make a unique and more general model for the ATC sentence syntax structure (see Figure 9). This model shows a more general and unique grammatical structure that can be used for all the five commands shown. The finite–state transition automata based on this network can recognize any of the structures found in the five statements discussed before. Additionally, in cases where extra words are found that does not match the syntactic structure of the statement it can be skipped using the skip loops defined before. The skip loops are included in all the nodes as a way to skip words that are not recognized as part of the grammatical structure or syntax of the sentence. Task-oriented grammars are usually known to relax its rules by ignoring portions of the input that do not fit its grammar. However, in this case, skip loops go beyond the relaxing concept because it allows to skip words that are either unrecognized or syntactically incorrect without loosing flexibility and without violating grammatical rules.

Two prototypes of the system have been prepared to test the grammar models and parser. One of the prototypes is written in Prolog and the other uses CLIPS (C Language Integrated Production System). The example shows a window (see Figure 10) with the results testing several of the commands using CLIPS. The first example "diamond six zero three four point zero miles" is a correct statement or command and is accepted by the parser and outputs the grammar for the statement as "subj idn idn idn mw mw unit". The second example inputs the statement "diamond six **what** zero three point **uups** zero miles" with two unexpected words (**what** and **uups**) that do not fit the grammar for the ATC's commands developed. The parser makes the corrections to the statement and filters the incorrect words outputting the correct statement and its grammar. Another useful example is the statement "wolf one zero **ahh heey** zero on course" is inputted and corrected getting the correct statement and grammar from the system output as shown in the figure.

CONCLUSION

A grammatical model for the air traffic controller's command language using finite-state transition networks has been presented. Also, a robust finite-state syntactic parser framework has been presented. Finite-state transition networks with skip loops are used to model the grammatical scheme for air traffic controller's language. Using skip loops allows to delete or ignore words that may be present in the statement that are unrecognized or that do not fit into the grammatical structure of the air traffic controller's language. The skip loops have the effect of deleting the incorrect word from the input stream. This technique facilitates the recognition of the statement minimizing the possibility of declaring the statement as ill-formed due to an incorrect word into the statement that do not match the predefined grammar structure. This is a result of an ongoing research for the ATC's commands recognition. The author has shown that these techniques are possible using two prototypes implemented in Prolog and CLIPS.

ACKNOWLEDGEMENT

The finite-state transition parser with skip loops and the ATC's grammatical models were developed by Dr. Jorge L. Ortiz during his participation in the U.S.Navy/ASEE Summer Faculty Fellowship Program in summer 1999. The author wants to express his gratitude to Mr. Paul R. Little at the Naval Air Warfare Center - TSD in Orlando, for making possible his participation in this program.

REFERENCES

Matthews, Clive.1998. An Introduction to Natural Language Processing through Prolog:

Roche, Emmanuel, and Shabes, Yves. 1997. Finite-State Language Processing: The MIT Press.

Clocksin,W.F., Mellish,C.S. 1994. Programming in Prolog. Fourth Edition: Springer.

Cole,Ronald A. 1996. Survey of the State of the Art in Human Language Technology: National Science Foundation.. http://cslu.cse.ogi.edu//HLTsurvey

Addison_Wesley Longman Limited.Gazda,G.,and Mellish,C. 1996. Natural Language Processing in Prolog: University of Sussex at Brighton, England.

Luger, G., Stubblefield,W. 1997. Artificial Intelligence: Structures and Strategies for Complex Problem Solving.:Addison Wesly Longman.

Suereth, Russel. 1997. Developing Natural Languages Interfaces: Processing Human Conversations: Mc-Graw-Hill.

Dougherty, Ray C. 1994. Natural Language Computing: An English Generative Grammar in Prolog: Lawrence Erlbaum Associates.

Figure 10. Sample Parser Output Using Clips.