

Research of NC Code Interpreter Based on Theory of Finite Automaton

Ge Teng (Corresponding author)

School of mechanical engineering, University of Shanghai for Science & Technology

516 Jun Gong Street, Shanghai 200093, China

Tel: 86-159-2156-3397 E-mail: 178201427@qq.com

Jiang Zhifeng & Fang Jianglong

Shanghai capital numerical control co.ltd

470 Gui Ping street, Shanghai 200233, China

Received: February 17, 2012

Accepted: March 6, 2012

Published: April 1, 2012

doi:10.5539/mas.v6n4p38

URL: <http://dx.doi.org/10.5539/mas.v6n4p38>

Abstract

This paper analyse the function and structure of NC code interpreter in detail. Syntactic recognition model is constructed based on the theory of finite automaton, the interpreter process every single line of NC code with high speed and high accuracy by just single scanning, the efficiency of fault detection is improved.

Keywords: NC code interpreter, Finite automaton

1. Introduction and Motivation

CNC technology is the key technology of CNC machine tools, CNC machine tools is the foundation of factory automation. In NC system G-code interpreter is a very important module. CNC machine tools commonly use G-code to describe the machine processing information, such as tool path, the choice of the coordinate system and the opening of the coolant. G-code translated into the data block which NC system can identify is the main function of G-code interpreter.

There are two ways to decode the G-code basically: compiling and interpreting, the compilation system is fast and efficient, but requires larger memory, likely to cause system resources shortage and reduce overall system performance when using small linear interpolation and part program is relatively complex because of the interval time between compiling and processing. Interpretation system is slow, simple structure, without a lot of disk space, take up less system resources, the disadvantage is that requires higher CPU speed. Embedded processor used in NC system widedspreadly with low-cost and low-power, but its performance compared to PC has a significant gap. Therefore, the improvement of the efficiency of G-code interpreter through optimization has important practical significance.

2. Function of NC Code Interpreter

In the NC system, the interpreter as a separate module and its main function is translating the processing information such as the start and end point coordinates, spindle speed, feed rate and tool number contained by NC code text file into the language which computer can process and stored in a dedicated buffer. Main functions of the interpreter:

- 1) Lexical analysis: scanning the string for each character, and identify the basic syntax words.
- 2) Parsing: parsing the word which identified by lexical analysis according the rules and syntax of the NC instructions, check out some advanced syntax errors.
- 3) Semantic checking: semantic checking is the senior examination focus on the wrong meaning of words in the program and logic errors between program segments.
- 4) Construction of data structure: transform the information which extracted from program segments into the corresponding intermediate code for the interpretation of the NC program.

5) Analysis of fixed cycle: a series of preset operation commands called fixed cycle like threading cycle and drilling cycle which widely used in numerical control system, because of these operation commands the efficiency of NC programming is further enhanced.

6) Analysis of subroutine and macro: subroutines and macros can achieve software reuse, saving users' time of writing programs. In the face of these calls, the interpreter should be able to select the appropriate entry and exit to these statements conversion processing.

3. Realization of Interpreter

3.1 Theory of Finite Automata

Finite automata as an identification device that can accurately identify the regular set which defined by regular grammar and regular expression. The introduction of finite automata theory is to find special methods for constructing syntax program automatically.

A finite automaton can be described as a state transition diagram shown in Figure 2. State transition diagram is a directed graph, each node of the figure represents a state, each directed edge links one state to another. Finite automaton likes a composition composed by a controller with reading head and a input tape with characters. The reading head moves from left to right, it will cause the state of the controller to change and reading head move to right a sign bit whenever reads a character. The controller includes finite states, there is a conversion relationship between the each state. Whenever a state changes to another state by reading a character, it called state transition, and the state after the change is referred to as the subsequent state. According to state transition diagram which constructed by grammar the lexical analysis program can be generated.

3.2 General Frame of the Interpreter

In order to approach to modular design, two data buffer are needed: middle data buffer and output data buffer. The interpreter is divided into two parts i.e., lexical and grammatical analysis, formation of the middle data structure; Semantic analysis, program interpretation and formation of the output data structure.

The efficiency of interpreter influenced by the numbers of string scanning, the influence is not clear when just a few program segments to be processed, but for the process of large parts with complex camber usually adopt small linear interpolation that generating continuous small lines program segments which often up to tens of thousands, then the numbers of string scanning is the deciding factor of interpreter's efficiency. So the interpreter adopts single pass and its general frame (Figure 1) description as follows:

- 1) A line of NC program to be read for the analysis.
- 2) The string to be scanned through the lexical analysis model which generated by the state transition, and to generate the middle data structure stored in the middle data cache.
- 3) Syntax analysis and semantic analysis to be completed in accordance with intermediate data structures and parameters of the numerical control system. Also at this stage interpreter complete the operations and actions of parts of the NC program i.e., the compatibility G-code modal, the subroutine call.
- 4) The middle data structure would be interpreted by calling performance function, at this stage the interpretation of operations and actions of all the NC programs should be completed, such as reference point return, fixed-cycle analysis, etc., the final target output data structure would be generated finally.

3.3 The Specific Implementation

NC code is in accordance with regular syntax rules which are relatively simple. Morphology is part of the syntax, the morphology analysis should be combined with syntax analysis for efficiency and practicality.

NC code alphabet includes letter, number, symbol, etc.

$\Sigma = \{\text{letter, number, symbol}\}$

Letter = {AC1, AC2, AC3}

Number = {0-9}

Symbol = {+, -, ., :, (,), /, space}

Address character1(AC1) set = {O, P, Q, M, N, T, H, L, D};

Address character2(AC2) set = {X, Y, Z, U, V, W, A, B, C, I, J, K, R};

Address character3(AC3) set = {G, F, S};

The regular expression of Σ :

$$L(\square) = ((L([ODLMNPTQH][0-9]+)|L([UVWXYZIJKABCR]-?[0-9]+(\backslash.[0-9]+)?))|L([GFS][0-9]+(\backslash.[0-9]+)?))L(\backslash(.*\backslash)))*L(//;.*)?$$

$$= (((([ODLMNPTQH][0-9]+)|([GFS][0-9]+(\backslash.[0-9]+)?))|([UVWXYZIJKABCR]-?[0-9]+(\backslash.[0-9]+)?)) \backslash(.*\backslash)))*L(//;.*)?$$

This regular expression can be described as a state transition diagram shown in Figure 2.

Statements:

S0: Initial state, at this state read the first character of one line of the NC code;

S1: Address character1 state, when read a letter of character set1 come into this state, at this state the number followed by address character must be unsigned integer.

S2: Address character2 state, when read a letter of character set3 come into this state, the number followed by address character of this state can be an unsigned integer or unsigned floating-point number.

S3: Address character3 state, when read a letter of character set2 come into this state, the number followed by address character of this state can be an integer or unsigned floating-point number.

S4: State of "+,-" symbol, the number followed by address character of this state must be signed number.

S5: Word1 state, in this state that read a complete word (address character + integer).

S6: State of "." symbol, the number followed by address character of this state must be floating-point number.

S7: Word2 state, in this state that read a complete word (address character + unsigned integer/floating-point number).

S8: State of "/" symbols.

S9: Comments state, when read a ";" character or two consecutive "/" character come into this state, it indicates ignore all subsequent characters.

S10: Symbol "(" state, come into this state when read "(", when read ")" step out of this state and come into the S0 state, and ignore the comment characters within brackets.

S11: Error state, it is termination state, indicates that the word current read has lexical errors.

S12: Exit state, it is termination state, indicates the entire program has been scanned without lexical errors.

Parsing process (Figure 3):

- 1) Read one line of NC code, then the working position jump into the initial state S0.
- 2) Read a character, according to state transfer diagram for the state transition.
- 3) Whenever enter a new state interpreter need to judge i.e., the current state is S11, scanning of the string would be stopped with displaying error messages; the current state is S12, scanning of the string would be stopped with reading the next line; the current state is intermediate state and after stepping out of S5, S7 state that represent the automata read a effective word, then extract the information of the current word.
- 4) Pointer move to the right a sign bit, read next character, according to the state transition diagram for state transition.
- 5) Repeat steps 3 and 4 until check is finished.

Whenever the automata come into the state of the S1, S2, S3, the address character is recorded, when step out of the S5, S7 word state extract information and store it in a corresponding intermediate data structure for the initial syntax checking.

source code of information extraction:

```
switch(s)
{
    case 0:code='\0'; num=0; flag=0; break;           // Jump into the initial state S0
    case 1;;
    case 2;;
    case 3:code=*Gcode;num=0;flag=0;break;         //Address character state and process initialization
    case 4: if(sign_flag)flag=1;break;             //State of "+,-" symbol
```

```

case 6: note_flag=1;k=0.1;break;           //State of "." symbol
case 5: if(flag) num=num*10-(*string-48); //Enter the word1 state, numeric recorded
        else num=num*10+(*string-48);break;
case 7: if(note_flag)                     //Enter the word2 state, numeric recorded
        { if(flag) num=num-k*(*string-48); //floating-point number
          else num=num+k*(*string-48);k*=0.1; }
        else num=num*10+(*string-48);break; // unsigned integer
default:break;
}

```

Due to the binary data analysis is easier than analysis of string, the information extracted through the methods above will be stored in a middle data structure, and then based on the information of the middle data structure to achieve senior grammar, semantics analysis.

source code of middle data structure:

```

struct GCodeBlock
{
    int N;           // N Code
    double X;       //X/pause time
    double Y,Z,I,J; //coordinate parameters
    double K;       // K/repeating time
    double R;       //R
    double S;       // Spindle speed
    double F;       // feed speed
    int P;          //pause time /number of Subroutine calling
    int Q;          //depth of per feed
    int L;          //number of Subroutine
    .....
}

```

Informations of the intermediate data structure are sent to the module of semantic analysis and code generation which complete semantic analysis and interpretation of the code according to the extracted information of NC code.

Read the array of intermediate data structures into which through the semantic analysis, all the non-circular and non-linear interpolation motion commands are converted into circular and linear interpolation i.e., the reference point return, fixed cycle. Fixed cycle is the group trajectory composed by series of circular interpolation and linear interpolation, it can be transformed to the feed of linear interpolation and the circular interpolation parameters through calculation. The action commands are converted into circular interpolation and linear interpolation, the function commands are stored in the corresponding flag, and the X, Y, and Z coordinates have been converted to the value of the mechanical, they are stored in the final output data structure at last.

For different types of numerical control system, the meaning of functional word may be different, and then the interpretation of the NC code of the different numerical control system can be achieved by overloaded function. G84 is the threading cycle of FANUC system, but in SIEMENS system G331 is used for tapping and G332 for back, and then the different Performance Function is called for interpretation.

4. Summary

In this paper, structure and characteristics of the NC code is analysed, syntax analysis model was constructed based on the theory of finite automaton, this model can extract the process information with high speed and high accuracy by scanning a single line of NC code once, it has obvious advantages in using small linear interpolation and part program is relatively complex. The interpreter is versatile and compatible, because corresponding interpretation module is called while the NC program of different CNC system is interpreted. However this

syntax model does not apply to the macro analysis, because the automaton would be more complex and less efficient. These problems will be solved by practice.

References

- Ji, H., Li, Y., & Xiao, S. G. (2006). Design and Implementation of G-code Interpreter in Linux Environment. *Application Research of Computers*, 12, 200-202.
- Suh, S. H., & Cheon, S. U. (2002). A framework for an intelligent CNC and data model. *Int. J. Adv. Manufact. Technol.*, 19(10), 727-735. <http://dx.doi.org/10.1007/s001700200083>
- Wu, K. N., Li, B., & Chen, J. H. (2006). Implementation of NC code interpreter of open architecture NC system platform. *China Mechanical Engineering*, 17, 168-171.
- Wang, Q. K., & Li, W. (2009). Novel structure of NC program processor for CNC system. *Journal of Beijing University of Aeronautics and Astronautics*, 35(1), 122-125.
- Xu, X. W., Wang, L. H., & Rong, Y. M. (2006). STEP-NC and Function Blocks for Interoperable Manufacturing. *IEEE Transactions on Automation Science and Engineering*, 3(3). <http://dx.doi.org/10.1109/TASE.2005.862147>
- Zhang, C. R., Shan, C., & Wang, H. (2002). Design and realization of code interpreter for CNC system. *Journal of Shandong university (Engineering science)*, 6, 565-569.

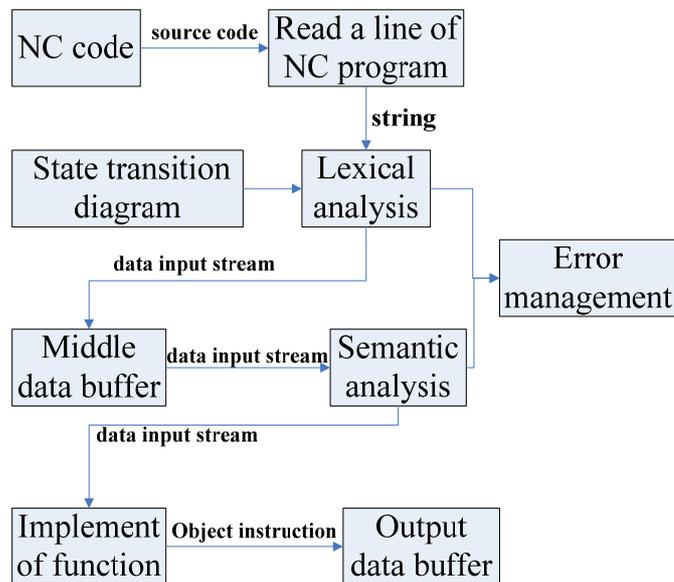


Figure 1. General frame

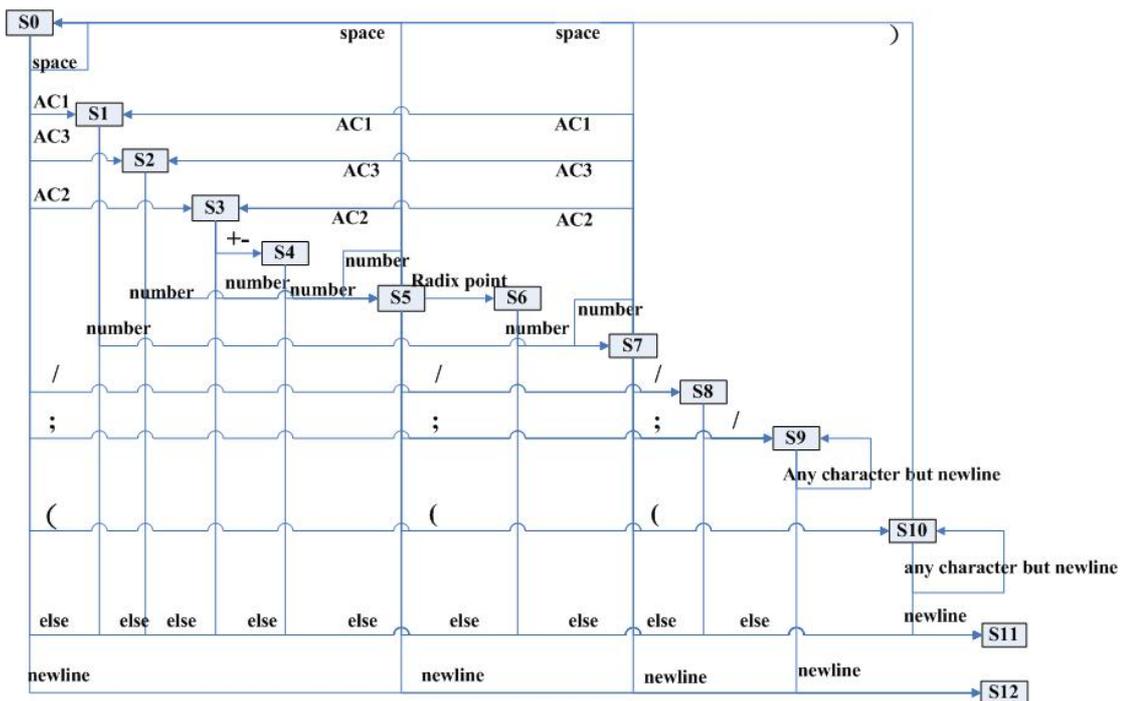


Figure 2. State transition diagram

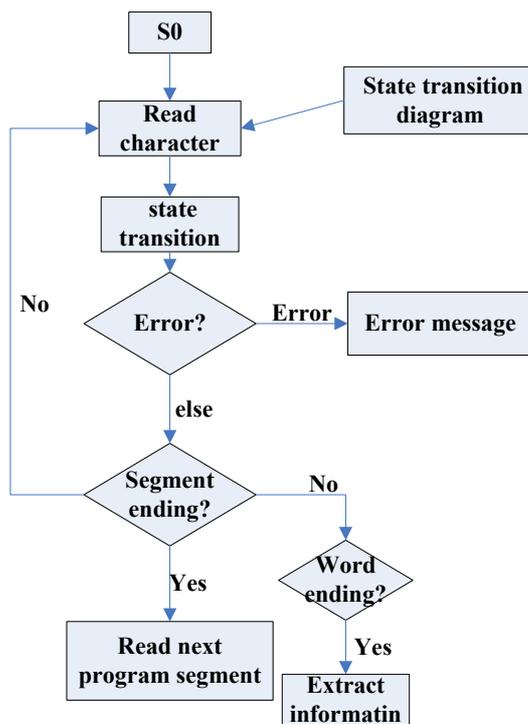


Figure 3. Flow chart of parsing