Usage of XML and P code for Robot Motion Control

Maja Lutovac, Goran Ferenc, JelenaVidaković, Zoran Dimić, Vladimir Kvrgić
Robot controllers, Lola Institute, Belgrade, Serbia
{maja.lutovac, goran.ferenc, jelena.vidakovic, zoran.dimic, vladimir.kvrgic}@li.rs

Abstract—**L-IRL (Lola Industrial Robot Language) is procedural language for industrial robot programming. Offline and real-time programming system are main parts of robot programming system based on L-IRL programming language. System for robot programming is using object code as one of the main communication tools between different elements of the system. In this paper we have presented how object code of the compiler can be generated in the form of P code or XML code. This gives software solution that is compact, portable and easy to use with standardized object code that consist information for robot motion control. After generation of object code on the offline part of the system, it is sent to the real-time part of the system using CORBA protocol.**

Keywords- Industrial robot; Robot language; Robot motion control; Robot programming

I. INTRODUCTION

It is developed L-IRL programming language for robot programming system based on DIN 66312 standard for industrial robots. The programming system consists of two parts – offline and real-time part [1]. Those two parts communicate using CORBA protocol. Offline part sends object code to the real-time part of the robot programming system. It is of great importance to have compact and standardized object code that consist information for robot motion control. P code is used as object code because it is often much smaller than the same program translated into machine code. XML code is used because it is text-readable and hence user friendly, and it is easy to produce this type of object code.

In previous version, parser and compiler were developed using recursive descent approach [1]. New language parser is written with tools such as Bison and Lex with the C++ programming language. The new version of L-IRL parsing is based on LALR (Look Ahead Left to Right) algorithm which is directly incorporated into Bison. Using these software tools together with methodologies of the object oriented programming, efficient structural and logical refactoring of the source is achieved. This approach enables logical and functional separation between different phases of parsing and compiling.

In this paper it is presented usage of P code and XML as the object code in communication between offline and real-time part of the system for robot control. Program written in L-IRL programming language is translating into one of these two types of object code. Using the P code as object was chosen because the first version of offline system for robot programming has limited memory resources and P code as summarized binary code is compact and does not take much space in memory. In one of the latest version of the system it was introduced the possibility of using XML as object code.

II. P CODE

A. P Code Generator

The P code compiler translates program written in the L-IRL programming language [2] into P object code. The resulting compiler output is an executable program and it can be interpreted in real time [3]. The compiler is implemented in two passes [4]. In the first pass lexical, syntactic and semantic analysis is performed (Fig. 1). Lexical analyzer recognizes the language and symbols and transforms them into an internal representation which is faster for work. It represents the interface between the original program and syntax analyzer. The result of the lexical analysis is internal representation of the original program lexemes. Information about lexemes (symbols) is stored in the symbol table. Based on these representations, compiler performs syntactic analysis, semantic analysis and generation of intermediate code during the first pass. Syntax analysis analyzing a text made of a sequence of symbols and determines its grammatical structure with respect to a given L-IRL programming language grammar.

Semantic analysis performs semantic checks such as type checking, object binding or definite assignment, rejecting incorrect programs or issuing warnings. For testing during the semantic analysis it is necessary to implement the stack. At the end of the first pass program generates an intermediate code which is written into temporary file and which is used in the second pass during the solving addressing in advance and code optimization.

Code generation of some statements requires defining of labels, whose addresses are not known at compile time. Because of unresolved addresses in advance, in the first pass, compiler introduced pseudo code orders for labels and tables. Addressing in advance problem or address call that is not known is solved by introducing counter labels and label tables. Initially the value of label counter is set to 0. In each label call counter value is reached and its value is incremented by one.

B. P Code Advantages

Translating up to object code that is interpreted in real time, we get the portable compiler. In case of realization of an interpreter in another robot controller it would be enough to rewrite the interpreter of object code. P code is designed to match the original program logic and execution of robotic operations in real time. It was regarded as a special language with its own syntax, which defines the command code and code sentences. When designing the code syntax it is sought to code commands report directly to the concepts of L-IRL, and that the code corresponds to the syntax of the source language.

Compared to direct translation into native machine code, a two-stage approach involving translation into P code and execution by an interpreter or just-in-time compiler offers several advantages like portability, simple implementation and compact size.

It is much easier to write a small P code interpreter for a new machine than it is to modify a compiler to generate native code for the same machine.

Generating machine code is one of the more complicated parts of writing a compiler. By comparison, generating P code is much easier. This makes it useful for getting a compiler up and running quickly.

Since P code is based on an ideal virtual machine, a P code program is often much smaller than the same program translated to machine code. When the P code is interpreted, the interpreter can apply additional runtime checks that are difficult to implement with native code.

C. P Code Example

The following text provides an example of program written in the L-IRL robot language and Fig. 2 shows object code generated in the form of P code. Program contains two move instructions for robot motion. Based on values of speed, acceleration and position in move instructions parameters required for robot motion are calculated. They are translating into object code and over CORBA protocol send on the real-time system where object code is executing in real time.

```
program linemovement;
var
    robtarget start :=
    robtarget(position(200.0, 0.0, 300.0));
    robtarget a := robtarget(position(200.0, 0,
       500.0));
endvar
seq
    move ptp start speed_ptp:= 0.2
       acc_ptp := 0.5 act_rob := lola_15;
    move lin a speed := 0.8 acc := 0.5
       act_rob := lola_15;
endseq
endprogram
```

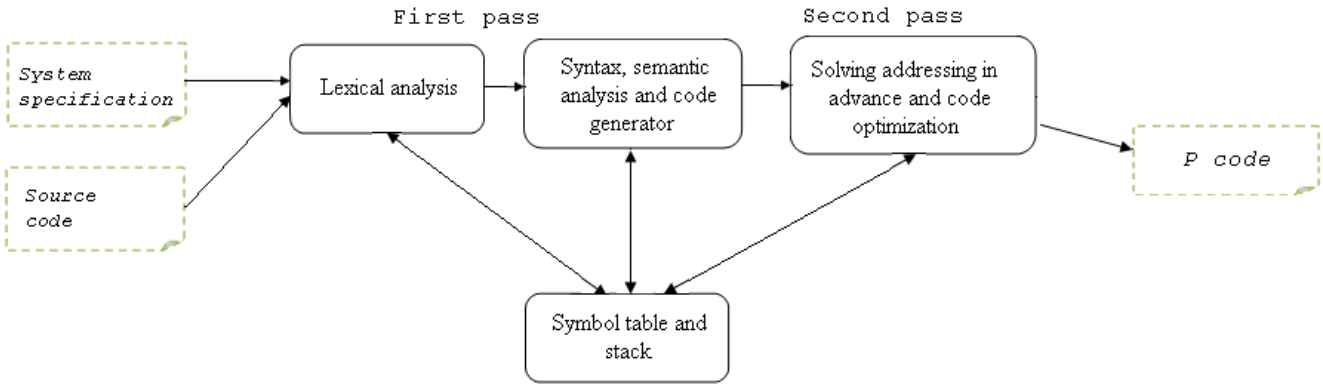

Figure 1. P code compiler

```
00 00 00 47 00 00 00 00 00 00 00 00 00 00 05 00 00 00 00 00 00 00 52 00 00 00 00 00 00 00 2E 00 00 00 08
00 00 E2 00 00 00 11 00 00 00 32 10 D1 E3 26 BC A0 B8 08 07 FD 91 41 BC C6 D0 12 7B 24 FD 84 3B E2 05 AD 2A
CE A3 BC 09 B1 65 3E 12 00 2A BB D0 D3 00 A3 E9 CE C4 3F 93 C7 4D 1A 03 13 C0 3F B3 2B EC 95 22 B8 6A 3E CF
54 99 DA 0B 5E BE F3 FB 31 2E 76 85 B5 BE 9A 89 32 D5 E1 E4 70 BF 99 83 71 40 59 E7 25 BF D3 81 5E 29 88 EF
3F D1 75 D0 49 0F FD 24 3F 11 7B 66 D3 A1 20 4D 3E C3 8A BA D8 58 52 57 40 11 89 B9 5B 10 BA 83 40 30 FF F2
EA 28 25 3F E3 22 86 FB 6C 58 14 BF 71 B7 BC 7F 33 18 90 BF 17 87 A7 00 00 00 00 00 00 00 00 00 00 00 00
00 00 31 30 31 32 00 00 01 39 00 00 00 01 00 00 00 F3 55 78 A3 AE 40 08 E9 14 FC 91 64 DD 3F F7 4D B3 B0 92
C8 40 2D 05 ED C9 9C D2 DC BF 4E 4B 3B 08 56 CB 9F 3E FE 75 C9 DE 06 BD 32 3F C3 3D 3B 49 48 10 BF BF BD 1D
```

Figure 2. Example of P code

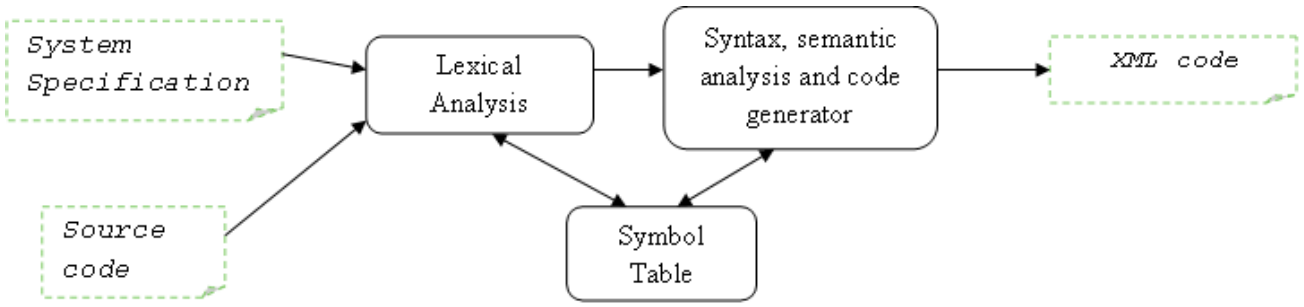

Figure 3. XML compiler

III. XML OBJECT CODE

A. XML Code Generator

Second option is to translate program written in the L-IRL programming language into XML object code. The resulting compiler output, like in case of P code, is an executable program and it can be interpreted in real time.

The compiler is performing lexical, syntactic and semantic analysis (Fig. 2). Like it is mentioned earlier lexical analysis converting a sequence of characters into a sequence of tokens and stores information about lexemes (symbols) in the symbol table. Then it follows syntax analysis, semantic analysis and generation of XML code. For generation of XML code it was used already developed XML parser, such as RapidXML parser [5].

XML object code has many useful properties such as well-defined tree structure. This structure was appropriate for developing our new interpreter, and also it is suitable for parsing using RapidXML parser.

Abstract syntax tree of the source code corresponds to DOM (Document Object Model) XML object code tree. Interpreting the object code on the real-time virtual machine consists of visiting DOM tree and it executes commands that are specified by nodes of the tree.

Basic structure of the object code is divided in three basic parts [6]. Object code starts with root node <program>.This has three child nodes:

- <system spec> - part which matches content of the system specification file,
- <symbols> - part which provides information about defined symbols and
- <seq> or <par> - part which defines sequential or parallel main execution block.

Each of these nodes defines separate section of the generated XML object program file. Node <system spec> defines section of XML object file that contains content of the system specification file. Content of this node is defined in compilation phase by adding the root node of the system specification file to the root node of object program. Second node <symbols> defines section which contains definitions of variables, constants, procedures and functions. Third child node of the root node, defines section which contains translated statements, jump statements, etc. Element <seq> defines sequential execution of the main execution block while <par> defines parallel execution of the main execution block.

Part of the object code which defines variables, constants, procedures and functions within the program consists of three sections defined by var node in which program variables are define, const which defines constants and function section in which the program functions and procedures are defined.

B. XML Advantages

XML is extensively used in web programming, but its use is found in other areas as well [7]. XML provides powerful and flexible tool for data exchange and its usage is increasing [8]. Usage of XML as object code is shown in this paper.

Using XML is a simplified representation of dynamic arrays, parallel execution units, functions and procedures as well as the definition of system variables.

Developing interpreter and object code compiler based on XML enables portability. XML is also used to represent the specification of the robot.

C. XML Example

The following text provides an example of move instruction written in the L-IRL robot language and part of the object code for move instruction generated in the form of XML.

```
move ptp robtarget(position(200.0, 0.0, 300.0))
speed_ptp := 0.2 acc_ptp := 0.5 act_rob := lola_15 ;
```

```
<move>
    <variable>
        <name>start</name>
    </variable>
    <type>ptp</type>
    <acc_ptp>
```

```
            <real>0.5</real>
        </acc_ptp>
        <act_rob>
            <variable>
            <name>lola_15</name>
            </variable>
        </act_rob>
        <c_cp/>
        <c_ptp/>
        <speed_ptp>
            <real>0.2</real>
        </speed_ptp>
        <time/>
        <until/>
</move>
```

IV. CONCLUSION

XML documents are text-readable and hence user friendly. However XML documents are relatively verbose in their content. They are large and can take a long time to process in real-time interpreter. Larger documents also can take long time to transmit. P code is smaller than the XML code, so memory usage and time required for sending and receiving information are improved. In both cases of usage P code or XML we get a portable compiler. This simplifies implementation of an interpreter in another robot controller, so it is sufficient to rewrite the interpreter of the object code only.

ACKNOWLEDGMENT

This work was created within the research project that is supported by the Ministry of Education Science, Republic of Serbia: "Development of the devices for pilots training and dynamic flight simulation of modern combat aircraft: 3 DoF centrifuge and 4 DoF spatial disorientation trainer".

REFERENCES

[1] M. Lutovac, G. Ferenc, V. Kvrgić, J. Vidaković, Z. Dimić, Robot programming system based on L-IRL programming language, Acta Technica Corviniensis – Bulletin of Engineering, Fascicule 2. April–June, pp. 27-30, 2012, ISSN: 2067-3809.

[2] V. Kvrgić, "Development of intelligent system for management and programming of industrial robots," (in Serbian), PhD dissertation, University of Belgrade, Faculty of Mechanical Engineering, 1998.

[3] M. Milićević, V. Kaplarević, Z. Dimić, V. Cvijanović, M. Bućan, "Development of distributed control system for robots control based on real-time LINUX platform," 10th Anniversary Int. Conf. on Accomplishments in Electrical and Mechanical Engineering an Information Technology DEMI, pp. 813-818, 2011.

[4] M. Pavlović, "High Level Programming Language for multi-robotic operations," (in Serbian), M. Sc thesis, Scholl of Electrical Engineering, University of Belgrade, 1994.

[5] RapidXML, http://rapidxml.sourceforge.net.

[6] V. Kaplarević, M. Milićević, J. Vidaković, V. Kvrgić, "New approach for dessigning robot programming system based on L-IRL programming language," 10th Anniversary Int. Conf. on Accomplishments in Electrical and Mechanical Engineering an Information Technology DEMI, pp.873-876, 2011.

[7] Elliotte Rusty Harold, XML Bible, IDG Books Worldwide, Inc W3C, Extensible Markup Language (XML) 1.0 (Fifth Edition), http://www.w3.org/TR/REC-xml/.

[8] T Sunnyvale, Krishna Sankar, Flexible XML parsing based on p code, United States Patent 7716576, Cisco Technology Inc. San Jose, CA (US) ,Issued May 20, 2002.