ACTA TECHNICA CORVINIENSIS – Bulletin of Engineering
Tome V (Year 2012). FASCICULE 2 [April–June]. ISSN 2067-3809

[1.] Maja LUTOVAC, [2.] Goran FERENC, [3.] Vladimir KVRGIĆ, [4.] Jelena VIDAKOVIĆ, [5.] Zoran DIMIĆ

ROBOT PROGRAMMING SYSTEM BASED ON L-IRL PROGRAMMING LANGUAGE

[1.] LOLA INSTITUTE, KNEZA VIŠESLAVA 70A, BELGRADE, SERBIA

ABSTRACT: *Contemporary robot languages should be simple for usage, compact, portable and easily integrated into complex production systems. L-IRL (Lola Industrial Robot Language) is a robot Programming Language that was initially Pascal-based programming language. Further development is intended to add new characteristics such as portability and to become easy to learn and use. L-IRL is programming language based on the procedural paradigm and it is basis of the offline part of the robot programming system. Developed Graphical User Interface of offline part provides user friendly usage. New language parser is formed as LR(1) type parser (parser that reads input from left to right) and written with tools such as Bison and Lex with the C++ programming language. The proposed solution gives logical and functional separation between different phases of parsing and compiling. L-IRL is using XML as one of the main communication tools between different elements of the system. XML is used as meta language for system specification file and as object code of the compiler so that new software solution is more compact and portable.*
KEYWORDS: *L-IRL, robot language, XML*

INTRODUCTION

The main aim is to create a programming language that is simple and easy to use and at the same time compact and portable. L-IRL (Lola Industrial Robot Language) is programming language based on the procedural paradigm. L-IRL is the basis of the offline part of the robot programming system. It is a Pascal-based programming language with syntax and some functional modifications. The further developments of L-IRL were to obtain more compact, portable solution so that the programming language can be easily to learn and use. More details about existing and previous versions of L-IRL can be found in [3, 6].

Aside from basic elements which exist in other programming languages L-IRL contains specific structures and language constructs based on DIN 66312 standard. Special language constructs proposed by the standard are special geometric data types, geometric expressions and move statements which are used for robot movement control. Currently defined robot movements are PTP (Point to Point) movements and movements along mathematically defined paths (CP movements) as well as approximate PTP and CP motions. Currently present mathematically defined paths are line and the circle. By specifying parameters of the move statement it is possible to change path, speed, and acceleration, orientation of the end-effectors and other characterizes of the movement.

Language parser is developed using software tools Lex and Bison which generate lexical and syntax analyzer of the source code. In previous version, parser and compiler were developed using recursive descent approach. In this 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. RapidXML parser for C++ was used for parsing and generating XML code.

Offline programming system is a development environment of the language compiler for robot programming in an L-IRL programming language. GUI of the offline part of robot control system combines review of the robot system specifications, editor, whose input is a program written in a robot language L-IRL as well as graphical representation and simulation of motions robot's end-effectors.

For its main use, robot programming, handling many language's constructs such as: dynamic arrays, parallel executions blocks and system variables definition using XML is simplified. Portability is enabled by developing interpreter and object code compiler based on XML. After generation, it is sent on to the real-time parts of the system [2, 4]. Real-time part of the system is control system based on Real-Time Linux platform on which is set OROCOS open architecture software system, designed specifically for creating applications of this type.

GRAPHICAL USER INTERFACE OF THE OFFLINE PART OF THE ROBOT PROGRAMMING SYSTEM

In Figure 1 is given the appearance of a graphical user interface of the offline part of the robot programming system, through which the user enters a program, written in an L-IRL Programming Language, and generates as output another program with the same meaning but written in a XML.

Left part of the main window (Figure 2) contains information about the robot system specification. It includes the ability to create a new variable, to change the value of the already defined variables and to delete variables from the system specifications.

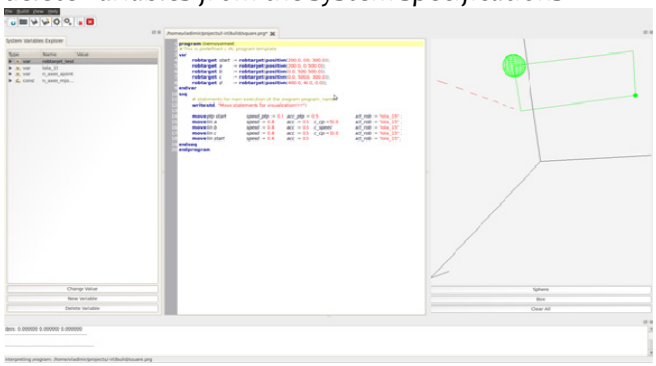

Figure 1. Graphical user interface of the robot programming system offline part

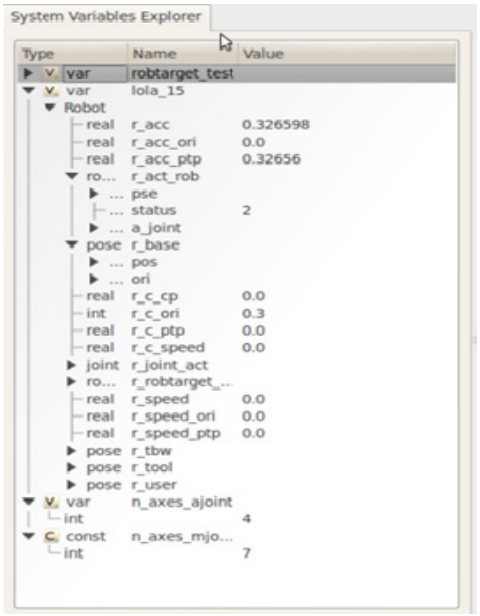

Figure 2. System specification

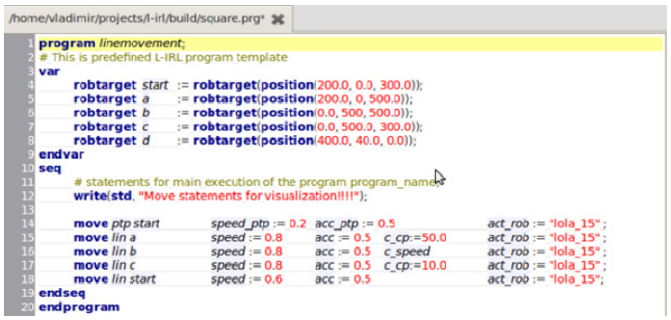

Figure 3. Editor of the robot programming system offline part

Main window (Figure 3) takes as input program written in L-IRL programming language. Program continue to compiles and the output is the program written in XML which have exactly the same values of parameters and gives same information like as input.

Right part of the graphical user interface (Figure 4) presented the simulation of robot movements and graphical representations of successive positions of the robot end-effectors. Before starting the robot, it is important to check the program written in language L-IRL which contains various commands of robot motion and to prevent the execution of programs that contains errors in the given path or position that may damage the system or robot or made unwanted movement.

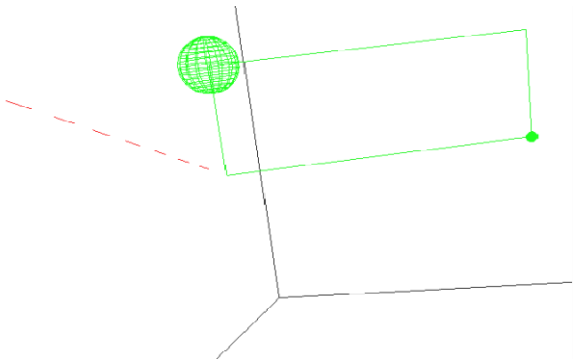

Figure 4. Graphical representations of robot movements

L-IRL Language Program Structure

L-IRL is procedural language with rigid syntax and every program obeys certain structural rules. Program written in L-ILR robot language includes the following sections [3, 6]:

- *Part for linking with extern files – import something*
- *Part for declaration of constants*
- *Part for declaration of user defined types*
- *Variable declaration part*
- *Part for defining procedures and functions*
- *Main execution block.*

L-IRL language defines standard data types (bool, int, real, char, string) and geometric data types:

- *Position - describes the position in millimeters of the coordinate system in space relative to the reference coordinate system. It is given with three real numbers that describe the coordinate values of x, y and z directions*
- *Orientation - given with three real numbers that describe the orientation angles in degrees of the coordinate system in space relative to the reference coordinate system*
- *Pose - consists of components pos and ori, which are position and orientation type, respectively*
- *Mainjoint - describes position and orientation of the end effectors in robot coordinates (internal coordinates)*
- *Addjoint - defines the additional axis of the robot*
- *Joint - consists of components m_jonit and add_joint, which are mainjoint and addjoint type, respectively*
- *Robtarget - describes the position and orientation of the end effectors and the status of the robot axes*

Derived data types, such as records or arrays, can also be defined. Expressions in L-IRL are typical expressions built from operators, variable/constant access and other conventional. Several types of statements are available in L-IRL such as assignments, conditional branches (if and case), loops (for, while and repeat), function and procedure blocks; L-IRL contains previously defined functions and procedures (@ - homogeneous transformation, inv – inverse homogeneous transformation, sin, cos, etc.)

L-IRL contains robot motion statements (move statements) and statements that manage parallel execution (wait, signal, etc) The move instructions consist of parameters for determining the type of path (PTP – point to point, LIN - linear or CIRCLE), the target point (and an additional secondary in a circular motion), speed and name of the robot. There are optional parameters that can be specified as (ACC, ACC PTP, C ORI, C, speed, speed, etc.), which define the characteristics of the movement. [3].

DTD And System Specification

System specification

System specification file contains information about the number and names of the robots that participated in robotic operations. System specification file is used for describing behavior of certain built-in structures of the language and also to represent system variables of the program in execution, like position of robot, speed, acc, etc . System specification file – sysspec.xml is written in XML.

Beside the source code in L-IRL language, in order to interpret and compile the source code, system specification file needs to be present in the execution folder. Compiler translates source code of the program in L-IRL together with system specification file into XML based object code.

Besides compiler of L-IRL, this file is using bay the robot controller, which updates the system variables values when program is executing.

DTD – Document Type Definitions

System specification file – sysspec.xml is written in XML which is defined by special syntax rules i.e. Meta language. Since this file can be changed by L-IRL programmer it is necessary to define rules of the XML syntax used in this file. Approach used for describing the rules of the XML syntax is called Document Type Definitions [1] or DTD.

A document type definition provides a list of the elements, attributes, notations, and entities contained in a document, as well as their relationships to one another. DTDs specify a set of rules for the structure of a document. DTDs can be included in the file that contains the document they describe, or they can be linked from an external URL. Such external DTDs can be shared by different documents and Web sites or other software. DTDs provide a means for applications, organizations, and interest groups to agree upon, document, and enforce adherence to markup standards.

XML Based Object Code

XML is one of the main communication tools between different elements of the robot programming system. After generation of the XML object code on the offline part of the system, it is sent on to the real-time parts of the system [2, 4].

Abstract syntax tree (AST) matches DOM tree of the XML object code document. Interpretation of the object code by real-time virtual machine is achieved by traversing a DOM tree of the XML document and executing statements and expressions coded within tree nodes. This way the resulting object code has same semantics as source code of the program.

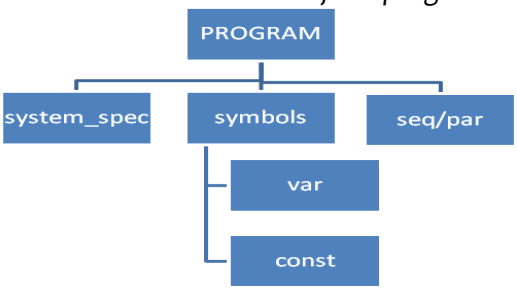

Figure 5. Object code basic structure

Basic structure of the object code is divided in three basic parts. Object code starts with root node <program> which has three child nodes (Figure 5):

- <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. Syntax check of the object code is enabled with the use of XML Schema. This way the syntax check is same as validation of the object code using XML Schema validation.

Basic Statements

Basic statements of the object code are statements which define structure of generated XML file.

Generated XML file has node <program> as root node. As mentioned earlier, root node has three child bodes: <system spec>, <symbols> and <seq> or <par>. 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.

Jump Statements

In order to generate object code for some statements (loops, if statement,) labels need to be defined. Labels define target address for jump instructions and they are defined with tag: <label>. Unconditional jump instruction is defined with tag <jmp>. As required attribute, jump instruction takes value of the target label. Conditional jump is performed with the instruction <jmpif>. Similar to unconditional jump instruction, label value has to be provided in the form of attribute. Jump condition is provided as a child node of the <jmpif> node.

Special Statements

Move statements are provided with special syntax for defining all move parameters. They are defined with tag: <move>. They are provided with special syntax due to define parameters that represent all necessary information for robot movement control. They consist of parameters for determining the type of path - PTP, LIN, CP which is followed by data of geometric type describing the target and, if necessary, extra point of the path. With these parameters, there are additional parameters (ACC, ACC PTP, C ORI, C SPEED, SPEED etc.) that are used optionally and they define the characteristics of the movement [3].

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 used to represent the specification of the robot, as well as for representing object code that is the main way of communication between different parts of the system for robot control.

CONCLUSIONS

This paper introduced a new approach for designing and developing offline part of the robot programming system with special emphasis on development of interpreter and compiler. The new version is more compact and portable. Graphical user interface provide simple usage and programming environment. One use of XML as object code is shown in this paper. XML code is generated using RapidXML parser for C++. Further work is focused on development of real-time virtual machine which interprets XML object code.

REFERENCES

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

[2] *Kaplarević V., Milićević M., Vidaković J., Kvrgić V. (2011), New approach for designing robot programming system based on L-IRL programming language, 10th Anniversary International Conference on Accomplishments in Electrical and Mechanical Engineering an Information Technology DEMI, Banja Luka, ISBN 978-99938-39-36-1, pp. 873-876*

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

[4] *Milićević M., Kaplarević V., Dimić Z., Cvijanović V., Bućan M. (2011), Development of distributed control system for robots control based on real-time LINUX platform, 10th Anniversary International Conference on Accomplishments in Electrical and Mechanical Engineering an Information Technology DEMI, Banja Luka, ISBN 978-99938-39-36-1, pp. 813-818*

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

ACTA TECHNICA CORVINIENSIS – BULLETIN of ENGINEERING

ISSN: 2067-3809 [CD-Rom, online]