UDC 004.7

O. V. Bilenkyi
S. O. Kot

CREATING A DSL FOR CONTROLLING A ROBOTIC MANIPULATOR

Vinnytsia National Technical University

Анотація

У роботі запропоновано створення власної предметно-орієнтованої мови для спрощення програмування промислових роботів-маніпуляторів. За допомогою мета-фреймворку ANTLR4 реалізовано лексичний і синтаксичний аналіз команд. Транслятор, написаний мовою Python, конвертує високорівневі інструкції користувача у виклики операційної системи ROS та фреймворку MoveIt.

Ключові слова: *робототехніка, робот-маніпулятор, предметно-орієнтована мова, ANTLR4, ROS, MoveIt.*

Abstract

The paper proposes the creation of a custom domain-specific language to simplify the programming of industrial robotic manipulators. Lexical and syntactic analysis of commands is implemented using the ANTLR4 meta-framework. A Python-based translator converts high-level user instructions into calls for the Robot Operating System and the MoveIt framework.

Keywords*: robotics, robotic manipulator, domain-specific language (DSL), ANTLR4, ROS, MoveIt.*

Introduction

In today's world, there is a rapid trend towards automation and the implementation of robotic manipulators in various industries. According to data from the IFR (International Federation of Robotics) [1], approximately 542,000 new industrial robots were installed worldwide in 2024, bringing the total number to 4.66 million units. This trend is economically justified by leading countries: robots are capable of working around the clock; they perform monotonous, heavy, or dangerous work much more efficiently and significantly facilitate the scaling of production.

However, such rapid robotization creates a shortage of qualified personnel capable of integrating and maintaining these systems. Robotics is a complex multidisciplinary field that combines programming, mathematics, physics, electronics, and mechanics. That is why the implementation and use of robotic manipulators in educational institutions is a modern step toward the comprehensive and practical study of these disciplines.

Research Findings

Unlike CNC (computer numerical control) machines, where G-code is the de facto programming standard, no such unified standard exists for robotic manipulators. Each major manufacturer offers its own proprietary solution and closed ecosystem.

Using GPLs (general-purpose programming languages) to directly write motion programs is often impractical in a manufacturing environment. In such code, the business logic of the technological process is mixed with the low-level implementation of drive control, requiring the operator to have deep programming knowledge. Process engineers must be able to declaratively describe exactly what the robot should do, while the details of how to implement it should be handled by an abstraction. That is why creating a custom DSL (Domain-Specific Language) for controlling a robotic manipulator is a relevant and appropriate solution [2].

ANTLR4 [3] was chosen as the tool for developing a specialized DSL. This parser generator frees the developer from the tedious task of writing basic constructs for lexical and syntactic analyzers, and automatically constructs an AST (abstract syntax tree).

The language development process is divided into two logical levels. First, a grammar file was created, containing only the declarative rules of the language. This approach makes the architecture flexible and independent of the platform's target programming language. The syntactic and lexical rules describing movement and control commands are shown in Fig. 1.

```
grammar dsl;

program: 'DEF' ID '()' statement* 'END' EOF;

statement: ptpCommand
         | linCommand
         | velCommand
         | waitCommand
         | outCommand
         ;

ptpCommand: 'PTP' position;
linCommand: 'LIN' position;

position: '{' 'X' x=NUMBER ',' 'Y' y=NUMBER ',' 'Z' z=NUMBER (',' 'A' a=NUMBER ',' 'B' b=NUMBER ',' 'C' c=NUMBER)? '}';

velCommand: '$VEL.CP' '=' NUMBER;
waitCommand: 'WAIT' 'SEC' NUMBER;
outCommand: '$OUT' '[' NUMBER ']' '=' booleanValue;

booleanValue: 'TRUE' | 'FALSE';

ID: [a-zA-Z_][a-zA-Z_0-9]*;
NUMBER: '-'? [0-9]+ ('.' [0-9]+)?;
WS: [ \t\r\n]+ -> skip;
```

Fig. 1 – Code fragment of the grammar for the DSL

The developed DSL [4] converts user-written commands into calls for the open meta-operating system ROS (Robot Operating System) and the MoveIt framework. After defining the rules in the parser generator, a logic module based on the Python language was developed.

Using the “visitor” architectural pattern, a class is described that traverses the generated AST tree, interprets the commands of our language, and translates them into the corresponding movement planning and execution commands of the MoveIt framework. Figure 2 shows how a tree node calls the corresponding “visitor” method, which encapsulates complex interaction with ROS [5].

```
class RobotController(dslVisitor):
    def __init__(self):
        super().__init__()
        moveit_commander.roscpp_initialize(sys.argv)
        rospy.init_node('robot_dsl_engine', anonymous=True)

        self.manipulator = moveit_commander.MoveGroupCommander("manipulator")
        self.gripper = moveit_commander.MoveGroupCommander("gripper")

        self.manipulator.set_max_velocity_scaling_factor(0.5)

    def extract_pose(self, ctx):
        pose = Pose()
        pose.position.x = float(ctx.x.text)
        pose.position.y = float(ctx.y.text)
        pose.position.z = float(ctx.z.text)

        roll = math.radians(float(ctx.c.text)) if ctx.c else 0.0
        pitch = math.radians(float(ctx.b.text)) if ctx.b else 0.0
        yaw = math.radians(float(ctx.a.text)) if ctx.a else 0.0
```

Fig. 2 – Code fragment of the visitor for the DSL

Conclusions

The implementation of the developed domain-specific language simplifies the process of programming robotic systems by separating the high-level business logic of the technical task from the complex low-level implementation. By using the ANTLR4 meta-framework in combination with the ROS and MoveIt ecosystems, an architecture has been created that abstracts the specifics of manipulators and is easily scalable in the future.

The implemented version ensures standalone operation: it includes motion control via PTP protocol, linear interpolation, logical pauses, and control of the end-effector. Further development involves expanding the grammar using complex control structures, implementing real-time error handling, and testing the system on physical industrial manipulators.

REFERENCES

1. International Federation of Robotics (IFR). World Robotics 2024 Report: Industrial Robots. URL: https://ifr.org/worldrobotics (accessed: 15.03.2026).
2. Rizwan M. Programming for Reliability and Safety in Robotics: The Role of Domain-Specific Languages. Lund: Lund University, 2024.
3. Parr T. The Definitive ANTLR 4 Reference. Pragmatic Bookshelf, 2013. 328 p.
4. Wesselink B., de Vos K., Kurtev I. et al. RoboSC: a domain-specific language for supervisory controller synthesis of ROS applications. 2023 IEEE International Conference on Robotics and Automation (ICRA). London, 2023. P. 9090–9096.
5. Joseph L., Cacace J. Mastering ROS 2 for Robotics Programming: Design, build, and simulate complex robots using ROS 2. 4th ed. Birmingham: Packt Publishing, 2024. 642 p.

Біленький Олексій Васильович – студент групи 1ПІ-22Б, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, e-mail: olexiubilenkiu@ukr.net

Кот Сергій Олександрович – к.філ.н., доцент кафедри Іноземних мов, Вінницький національний технічний університет, м. Вінниця, e-mail: kot.sergii@vntu.edu.ua

Oleksii V. Bilenkyi – Faculty of Information Technologies and Computer Engineering

Sergii O. Kot – *PhD,* Associate Professor in the Department of Foreign Languages at Vinnytsia National Technical University, Vinnytsia, e-mail: kot.sergii@vntu.edu.ua