Extension of Domain-Specific Language and its Operational Semantics*

William Steingartner[1,*,†], Davorka Radaković[2,†] and Boris Dovčík[1,†]

[1]*Technical University of Košice, Faculty of Electrical Engineering and Informatics, Košice, Slovakia*

[2]*University of Novi Sad, Faculty of Sciences, Department of Mathematics and Informatics, Division of Informatics, 21000 Novi Sad, Serbia*

Abstract

A strong foundation in formal methods for software engineering is important in educating novice programmers. Semantic modeling is the necessary foundation for formal methods. The increasing utilization of domain-specific languages (DSL) underscores their importance in practical applications. Therefore, incorporating them into the curriculum of the Semantics of Programming Languages represents a notable departure from the traditional way of teaching the semantics of (mostly) imperative languages. Previously, we developed a practical way to visualize natural semantics calculations for a selected domain-specific language for motion control of a robot in an orthogonal grid. The main goal of this article is to present a tool designed and developed by us, which is intended for interactive visualization of the chosen semantic method and its calculations. In this article, we extend the previous visualization with an extended version of the selected DSL to give a robot a more compact movement by angle (which is multiple of 90 degrees) in the desired direction. In this way, we teach students the basics of formal semantics and how to extend DSL, i.e. to extend syntax by new commands and maximize students' learning efficiency. The findings of this paper would be of interest to educators and students.

Keywords

formal semantics, natural semantics, visualization, experiments, NextUI

1. Introduction

To novice computer science students it is significant to comprehend how the program is performed and what steps are taken. Accordingly, they must grasp the specific steps through which a program is executed by the machine. The principles based on formal methods guarantee the correctness of the designed and developed systems [1, 2]. The operational semantics is used to describe these processes. It contains two approaches – natural and structural operational semantics [3, 4, 5]. Teaching novice students to formal methods in theoretical computer science is a challenging task. Due to the COVID-19 pandemic, and lack of face-to-face teaching interaction, another help to online lessons is diverse visualizations of algorithms and scientific concepts. The use of software tools enabling interaction and visualization of calculations [6, 7, 8, 9, 10] can be viewed as a contemporary teaching method that enhances its appeal. Throughout the project, our motivation came from the results in the following areas: visualization of algorithms and methods, operational semantics, domain-specific languages, and logic. Since before the COVID-19 pandemic at the Faculty of Electrical Engineering and Informatics, Technical University of Košice, we have developed several software tools [11, 12, 13] (for our teaching subject) that help students grasp the subject's material. Developed tools were intended for students studying primarily the course Semantics of Programming Languages, in addition to related courses focused on the formal semantics of languages, properties of programming languages, or introduction

SQAMIA 2024: Workshop on Software Quality, Analysis, Monitoring, Improvement, and Applications, September 9–11, 2024, Novi Sad, Serbia

*You can use this document as the template for preparing your publication. We recommend using the latest version of the ceurart style.

*Corresponding author.

†These authors contributed equally.

william.steingartner@tuke.sk (W. Steingartner); davorkar@dmi.uns.ac.rs (D. Radaković); boris.dovcik@student.tuke.sk (B. Dovčík)

0000-0002-2852-9403 (W. Steingartner); 0000-0001-8480-3211 (D. Radaković)

to programming. This paper is an extension to the work in [14] where we presented a visualization of the natural semantics of selected DSL and we follow some principles from [15]. In this paper, we present a new teaching tool for visualizing natural semantics calculations for a selected domain-specific language for motion control of a robot in an orthogonal grid with extensions of the angle of movement. The main goal of this article is to present a tool designed and developed by us, which is intended for interactive visualization of the chosen semantic method and its calculations. We formulated the natural semantics of the DSL language for robot control based on research results, and we verified and proved its equivalence with existing semantic methods within our research in [16]. In this work, we build on previous results achieved in science and research, as well as in technological development.

The structure of this paper is the following: the next section reviews related work. Further, we present the syntax and semantics of the extended robot language (Section 3). Section 4 then presents the application implementation and its graphical user interface. Lastly, Section 5 outlines the conclusion on the presented topic.

2. Related Works

Numerous authors implement visual tools designed to aid students in mastering subjects. They employ visualization software or create tools to augment students' learning experiences, aiming to bridge theoretical concepts with practical applications. This approach fosters a clearer comprehension of the applied procedures and principles taught in the course.

An example of a visualization tool for teaching compilers VCOCO (Visible COmpiler Compiler) is [17]. It depends on COCO/R [18, 19] that is responsible for creating analyzers, and it creates visual representations that can be manipulated by students.

Another use of DSL for robots, Mernik demonstrated in [20]. It shows how different language compositions can be realized using LISA and Robot DSL, another research area in Software Language Engineering.

The RISC Algorithm Language (RISCAL) is a language for the formal modeling of theories and algorithms given in [14], and it supports full (first-order) predicate logic on top of a rich basis of types (integers, tuples, records, arrays, sets, recursive types) with corresponding operations. This educational tool includes the automatic generation of checkable verification conditions from algorithms, the visualization of the execution of procedures and the evaluation of formulas illustrating the computation of their results, and the generation of Web-based student exercises and assignments from RISCAL specifications. Further, the usage and implementation of RISCAL, a model checker for mathematical theories and algorithms based on a variant of First-order logic within models, is reported in [21]. They reported enhancing education in computer science and mathematics, in particular in academic courses on logic, formal methods, and formal modeling (see [22]).

Tsimbolynets and Perháč present in [6, 11] a tool developed to visualize the translation process of a program written in a simple imperative programming language by structural operational semantics, also known as small-step operational semantics. It is OS-independent; therefore, it is implemented as a web application like our visualization tool.

The paper [12] introduces a language definition tool that utilizes a metamodel specification rather than grammar as the basis of the language definition. Inspired by the Java tool YAJCo, it employs typical object-oriented techniques, defining the metamodel as classes in Python. The parsing process results in a graph of objects, exemplified through a case study involving a basic imperative programming language.

3. The Extended Robot Language

The extended robot control language is a domain-specific language used for describing the movement of a robot in a defined orthogonal two-dimensional system. We note that the first idea about this robot language was presented in [23].

3.1. Syntax of the Extended Robot Language

The language of the robot is very simple. It contains commands for controlling the robot's rotation, and movement, returning to the initial state, skip command, and a pattern for the sequencing of commands. Therefore, the simplest way of defining syntax is using the Backus-Naur Form, for which two syntactic domains (categories) were defined:

$$\begin{array}{ll} n \in \mathbf{Num} & \text{— numerals,} \\ S \in \mathbf{Statm} & \text{— statements.} \end{array}$$

The language of the robot uses concepts used in LOGO or Karel the Robot which makes this language closer to real languages. Then the syntax is as follows:

$$S ::= \mathbf{turn\ left} \mid \mathbf{turn\ right} \mid \mathbf{forward} \mid \mathbf{forward}\ n \mid \mathbf{reset} \mid \mathbf{skip} \mid S; S.$$

The statements **turn left** and **turn right** express a change in an angle of movement by 90 degrees to left or right, respectively. The statements **forward** and **forward** n express the movement of the robot in a direction that is defined by the angle. Argument n specifies the number of steps of the movement. The statement **reset** represents an instantaneous return to the initial position specified by the user at the beginning. The **skip** statement is an empty statement, and its role is mostly in proof of semantic equivalence. This syntax is now extended by new commands:

$$S ::= \ldots \mathbf{turn\ left}\ n \mid \mathbf{turn\ right}\ n.$$

These statements change the angle of movement by $n \cdot 90$ degrees in the desired direction.

3.2. Semantics of the Extended Robot Language

As semantic domains, we consider here standard numerical sets for naturals with zero ($\mathbb{N}_0$) and integers ($\mathbb{Z}$) as given in [14, 24].

Here, the configuration in natural semantics shall be extended. Configuration must express the current position with the direction. Direction is given by an angle of movement in the orthogonal system. We consider the following directions (given as angles in degrees):

- **0** represents the up,
- **90** represents the right,
- **180** represents the down,
- **270** represents the left movement direction.

In this model, the robot can turn only by multiples of 90 degrees, and the values of angles we consider as elements of the set $\mathbf{Angle} = \mathbb{Z}$. A configuration is an element of the semantic domain defined as follows:

$$\mathbf{Config} = \mathbf{Point} \times \mathbf{Angle}.$$

Then elements of this semantic domain are tuples consisting of a point and actual angle: (p, α). The semantic function is then defined as follows

$$\mathscr{S}' : \mathbf{Statm} \to (\mathbf{Config} \rightharpoonup \mathbf{Config})$$

and the function for every statement is given by

$$\mathscr{S}' [\![S]\!] (p, \alpha) = \begin{cases} (p', \alpha'), & \text{if } \langle S, (p, \alpha) \rangle \to (p', \alpha'), \\ \perp, & \text{otherwise.} \end{cases}$$

Here, the tuple $\langle S, (p, \alpha) \rangle$ is a new configuration in natural semantics for the extended language, and $\langle S, (p, \alpha) \rangle \to (p', \alpha')$ is a new transition relation. The semantic rules for statements were defined in [4] and [5], for the new statements we define new rules as the following:

$$\frac{\langle \textbf{turn left}, (p,\alpha)\rangle \to (p,\alpha'') \quad \langle \textbf{turn left}\ m, (p,\alpha'')\rangle \to (p,\alpha')}{\langle \textbf{turn left}\ n, (p,\alpha)\rangle \to (p,\alpha')}\ (\text{turn-l-n}_{\text{ns}})$$

$$\frac{\langle \textbf{turn right}, (p,\alpha)\rangle \to (p,\alpha'') \quad \langle \textbf{turn right}\ m, (p,\alpha'')\rangle \to (p,\alpha')}{\langle \textbf{turn right}\ n, (p,\alpha)\rangle \to (p,\alpha')}\ (\text{turn-r-n}_{\text{ns}})$$

For both given rules, we formulate the following:

- $\alpha'' = (\alpha \oplus \mathbf{90}) \bmod \mathbf{360}$,
- $\alpha' = (\alpha'' \oplus \mathbf{90} \otimes \mathscr{N}[\![m]\!]) \bmod \mathbf{360}$, which corresponds to

$$\alpha' = (\alpha \oplus \mathbf{90} \otimes \mathscr{N}[\![n]\!]) \bmod \mathbf{360},$$

for $\mathscr{N}[\![n]\!] = \mathscr{N}[\![m]\!] \oplus \mathbf{1}$, where $\mathscr{N}[\![n]\!], \mathscr{N}[\![m]\!] \in \mathbb{N}_0$.

In the case of both commands, we can observe the following equivalences:

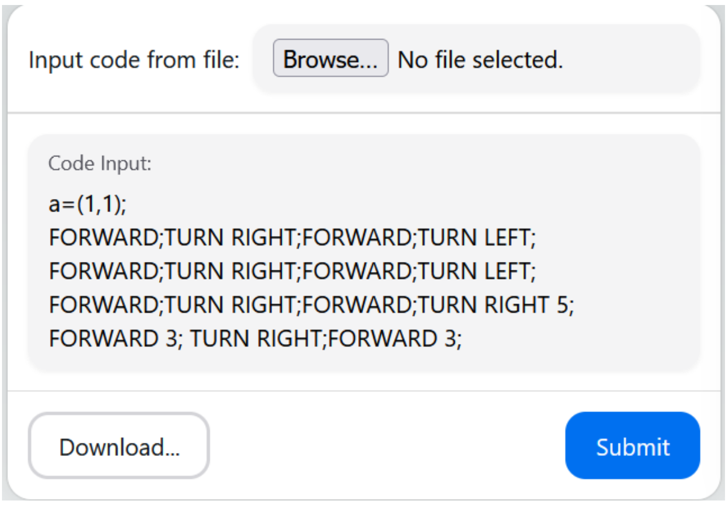
$$\text{if } \alpha \in \{\mathbf{0},\mathbf{90},\mathbf{180},\mathbf{270}\}, \text{ then } \langle \textbf{turn left}\ n, (p,\alpha)\rangle \equiv \langle \textbf{skip}, (p,\alpha)\rangle,$$
$$\text{if } \alpha \in \{\mathbf{0},\mathbf{90},\mathbf{180},\mathbf{270}\}, \text{ then } \langle \textbf{turn right}\ n, (p,\alpha)\rangle \equiv \langle \textbf{skip}, (p,\alpha)\rangle,$$

where $\mathscr{N}[\![n]\!] \bmod \mathbf{4} = \mathbf{0}$. We note that $\mathscr{N}[\![\cdot]\!]$ is a semantic function that sends numerals to naturals (or zero):

$$\mathscr{N} : \mathbf{Num} \to \mathbb{N}_0.$$

4. Implementation

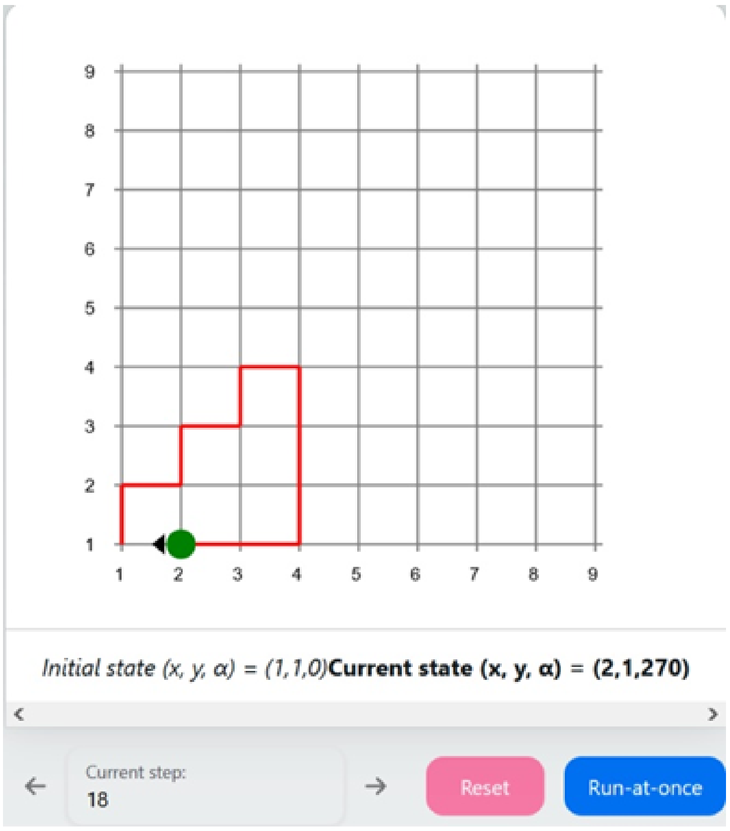
The application Operics for visualizing the operational semantics of robot language is available online at [25]. Next.js framework and the NextUI library were used to design our GUI. The application Operics utilizes the microservices architecture. Visualization logic is split between the frontend, which provides an interface for a user to interact with the application, and the backend, which logic is separated into two modules, one for grammar processing and the other for subsequent interpretation of input and output generation.



```
a=(1,1);
FORWARD;TURN RIGHT;FORWARD;TURN LEFT;
FORWARD;TURN RIGHT;FORWARD;TURN LEFT;
FORWARD;TURN RIGHT;FORWARD;TURN RIGHT 5;
FORWARD 3; TURN RIGHT;FORWARD 3;
```

Figure 1: The input component

The UI component (Figure 1) provides two ways of inputting code: through the text area and file upload. The robot grid (Figure 2) is the interpreter of possible states on the interface. At first, it is initialized with the creation of the whole grid and the robot on it.



*Initial state (x, y, α) = (1,1,0)***Current state (x, y, α) = (2,1,270)**

Figure 2: The grid component

4.1. The Interpretation Module

The interpretation module simplified controller architecture is shown in Figure 3. The *TreeBuilderService* creates a derivation tree of the natural semantics in the form of an object Tree. Method *applyNaturalSemantics* is extended for the rule processing of extended robot control language. Methods of the *GeneratorService* which is run after *TreeBuilderService* use a constructed *Tree* object and generate the desired output format. The extended state of the robot inherits from the basic state of the robot and extends it by the rotation angle. Subsequently, state transitions, natural semantics in the form of LaTeXcode, or graphic visualization in the form of a PDF file are generated based on the language version used. The endpoints were extended by adding/alternative postfix to the existing paths creating two new endpoints for extended robot control language.

As an important and (no less) interesting element of controlling the robot, we mention that the user can step the robot's movement himself, or start automatic stepping (run-at-once) from any point, i.e. from the beginning of the program, and during stepping let the robot catch up to end of the program. After performing the program, user receives a visible output consisting of more information. First of all, the grid with the full path of robot is visualized (this is what the grid component provides, see Figure 2. Formally, the robot is in individual states during its run (its activity) until it reaches the end (result) state. The list of all reached states is provided as another result (Figure 4).

Last but not least, a tree of natural semantics is also provided as a result, in which individual transitional sessions (big steps) are expressed, with the help of which the robot gets from the initial state to the final state. An example of such tree is in Figure 5.

We note that the individual results from the Operics program as presented in Figures 2, 4 and 5 express the solution or the result of the interpretation of the program expressed in Figure 1.

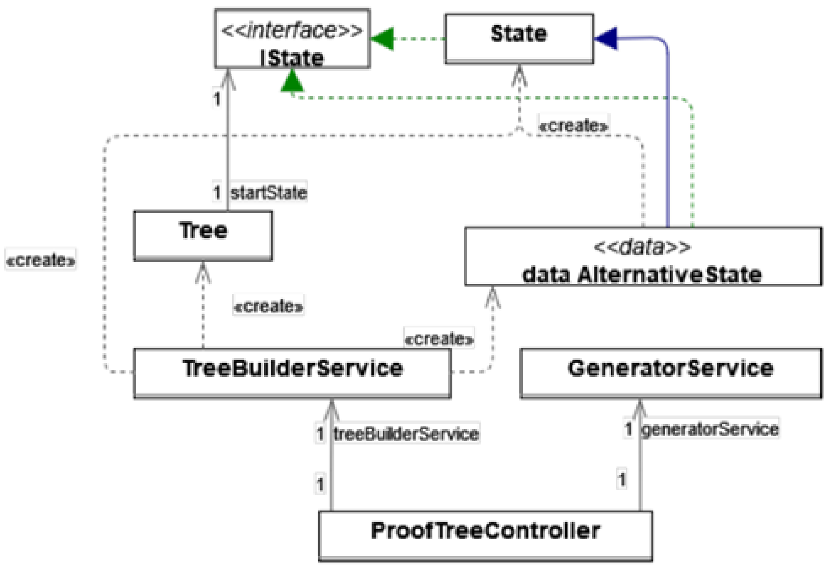
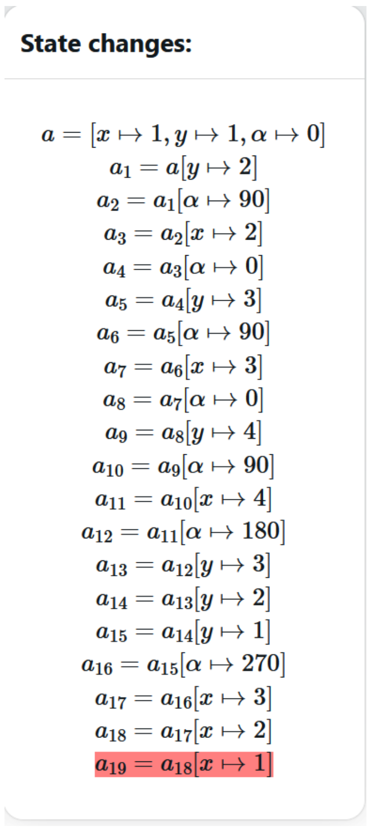


Figure 3: A simplified extended domain model of the interpretation module



State changes:

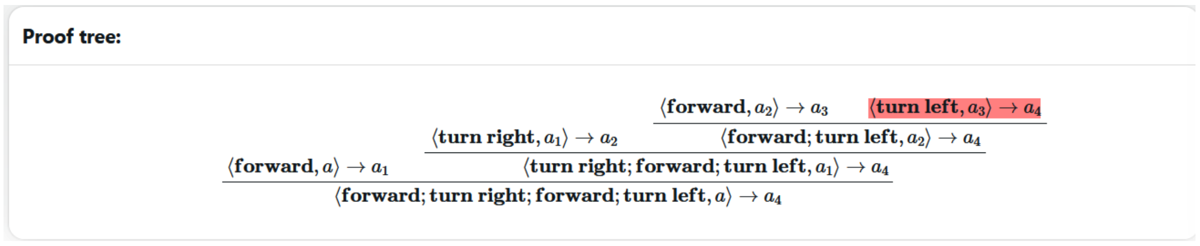
$$
\begin{aligned}
a &= [x \mapsto 1, y \mapsto 1, \alpha \mapsto 0] \\
a_1 &= a[y \mapsto 2] \\
a_2 &= a_1[\alpha \mapsto 90] \\
a_3 &= a_2[x \mapsto 2] \\
a_4 &= a_3[\alpha \mapsto 0] \\
a_5 &= a_4[y \mapsto 3] \\
a_6 &= a_5[\alpha \mapsto 90] \\
a_7 &= a_6[x \mapsto 3] \\
a_8 &= a_7[\alpha \mapsto 0] \\
a_9 &= a_8[y \mapsto 4] \\
a_{10} &= a_9[\alpha \mapsto 90] \\
a_{11} &= a_{10}[x \mapsto 4] \\
a_{12} &= a_{11}[\alpha \mapsto 180] \\
a_{13} &= a_{12}[y \mapsto 3] \\
a_{14} &= a_{13}[y \mapsto 2] \\
a_{15} &= a_{14}[y \mapsto 1] \\
a_{16} &= a_{15}[\alpha \mapsto 270] \\
a_{17} &= a_{16}[x \mapsto 3] \\
a_{18} &= a_{17}[x \mapsto 2] \\
a_{19} &= a_{18}[x \mapsto 1]
\end{aligned}
$$

Figure 4: A listing of state changes

4.2. The module for the graphical web interface

The robot movement is visualized on the robot grid using the react-konva library. The initial state of the robot and the current state of the robot are shown under the grid. Depending on the language version used they are shown either as configuration of points x and y:

$$\langle x, y \rangle$$


Proof tree:

$$\dfrac{\langle \mathbf{forward}, a \rangle \to a_1 \qquad \dfrac{\langle \mathbf{turn\ right}, a_1 \rangle \to a_2 \qquad \dfrac{\langle \mathbf{forward}, a_2 \rangle \to a_3 \qquad \langle \mathbf{turn\ left}, a_3 \rangle \to a_4}{\langle \mathbf{forward}; \mathbf{turn\ left}, a_2 \rangle \to a_4}}{\langle \mathbf{turn\ right}; \mathbf{forward}; \mathbf{turn\ left}, a_1 \rangle \to a_4}}{\langle \mathbf{forward}; \mathbf{turn\ right}; \mathbf{forward}; \mathbf{turn\ left}, a \rangle \to a_4}$$

Figure 5: A proof tree for first step in Figure 2

or for an extended version with an angle of rotation

$$\langle x, y, \alpha \rangle.$$

The rotation of the robot is shown as an arrow that points to the direction described by the angle of the movement. Calling **reset** command and moving the robot through the same paths multiple times can be confusing. For that reason when reset command is used the the color of the path is changed enabling thus easier recognition of current and past paths.

5. Conclusion

In this paper, we focused on the formulation and definition of semantic methods for describing the meaning of programs in a simple domain-specific language for controlling a robot in an orthogonal system with extensions of the angle of movements. We presented a software tool for formulating semantic methods, serving as an educational aid to enhance its appeal. As mentioned in [14] regarding the previous version of this tool, our application also presents opportunities for further development. We also would like to take advantage of the flexibility of this DSL to extend it with loops, to define conditionals, and to introduce some kind of state machine so our DSL for controlling the robot could become closer to real languages.

Acknowledgments

This work was supported by national KEGA project 030TUKE-4/2023 – “Application of new principles in the education of IT specialists in the field of formal languages and compilers”, granted by the Cultural and Education Grant Agency of the Slovak Ministry of Education. An international cooperation was supported by CEEPUS Network CZ-1503-04-2324 (Umbrella) – “Development of Computational Thinking”.

References

[1] S. Szymoniak, S. Kesar, How can best practices of cybersecurity include artificial intelligence within smart cities, in: The Leading Role of Smart Ethics in the Digital World, Universidad de La Rioja, 2024, pp. 135–142.

[2] L. Jancik, M. Kvet, Analytical tool for oracle sql statements, in: 2024 35th Conference of Open Innovations Association (FRUCT), Tampere, Finland, 2024, pp. 275–283. doi:10.23919/FRUCT61870.2024.10516421.

[3] F. Nielson, H. Nielson, Semantics with Applications. An Appetizer, Springer-Verlag London Limited, 2007.

[4] W. Steingartner, V. Novitzká, Natural semantics for domain-specific language, in: L. Bellatreche, M. Dumas, P. Karras, R. Matulevičius, A. Awad, M. Weidlich, M. Ivanović, O. Hartig (Eds.), New Trends in Database and Information Systems, Springer International Publishing, Cham, 2021, pp. 181–192.

[5] W. Steingartner, V. Novitzká, Metódy formálnej sémantiky programovacích jazykov [Methods of Formal Semantics of Programming Languages], Technical University of Košice, 2022. (in Slovak).

[6] V. Tsimbolynets, J. Perháč, Visualization Tool for Structural Operational Semantics of Simple Imperative Languages, IPSI Transactions on Internet Research 19 (2023) 66–74.

[7] S. Szymoniak, Time Influence on Security Protocol, in: H. K. Raian Ali, L. Maciaszek (Eds.), Proceedings of the 16th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE), ScoTePress, 2021, pp. 181–188.

[8] W. Schreiner, F.-X. Reichl, Mathematical Model Checking Based on Semantics and SMT, IPSI Transactions on Internet Research 16 (2020) 4–13.

[9] G. Savić, M. Segedinac, M. Čeliković, I. Luković, A Meta-model for Key Performance Indicators in Higher Education, IPSI Bgd Transactions on Internet Research 19 (2023) 76–91.

[10] W. Schreiner, W. Steingartner, Visualizing Logic Formula Evaluation in RISCAL, Technical Report, Research Institute for Symbolic Computation (RISC), Johannes Kepler University, Linz, Austria, 2018.

[11] V. Tsimbolynets, J. Perháč, Visualization of imperative programs translation with structural operational semantics, in: 2022 IEEE 16th International Scientific Conference on Informatics (Informatics), 2022, pp. 328–333. doi:10.1109/Informatics57926.2022.10083474.

[12] S. Chodarev, S. Ilyas, Metamodel-based language definition with python, IPSI Transactions on Internet Research 19 (2023) 32–38.

[13] D. Mihályi, M. Peniašková, J. Perháč, J. Mihelič, Web-based Questionnaires for Type Theory Course, Acta Electrotechnica et Informatica 17 (2017) 35–42.

[14] W. Steingartner, D. Radaković, R. Zsiga, Some Aspects about Visualization of Natural Semantics for a Selected Domain-Specific Language, IPSI Transactions on Internet Research 19 (2023) 46–54.

[15] J. Dostal, X. Wang, W. Steingartner, P. Nuangchalerm, Digital Intelligence – New Concept in Context of Future of School Education, in: L. Chova, A. Martinez, I. Torres (Eds.), 10th International Conference of Education, Research and Innovation (ICERI2017), ICERI Proceedings, 2017, pp. 3706–3712. 10th Annual International Conference of Education, Research and Innovation (ICERI), Seville, SPAIN, NOV 16-18, 2017.

[16] W. Steingartner, V. Novitzká, W. Schreiner, Proof of equivalence of semantic methods for a selected domain-specific language, Journal of Applied Mathematics and Computational Mechanics 23 (2024) 79–92. URL: https://doi.org/10.17512/jamcm.2024.2.07. doi:10.17512/jamcm.2024.2.07.

[17] R. D. Resler, M. D. Deaver, VCOCO: A visualisation tool for teaching compilers, in: Annual Conference on Innovation and Technology in Computer Science Education, ser. ITiCSE '98, 1998, pp. 199–202.

[18] H. Mössenböck, The Compiler Generator Coco/R User Manual, Johannes Kepler University Linz, Institute of System Software, 1990.

[19] H. Mössenböck, A generator for production quality compilers, in: D. Hammer (Ed.), Compiler Compilers, Springer, Berlin, Heidelberg, 1991, pp. 42–55.

[20] M. Mernik, An object-oriented approach to language compositions for software language engineering, Journal of Systems and Software 86 (2013) 2451–2464.

[21] W. Schreiner, Theorem and algorithm checking for courses on logic and formal methods, in: W. N. Pedro Quaresma (Ed.), Theorem Proving Components for Educational Software (ThEdu'18), volume 290 of *EPTCS*, Open Publishing Association, 2019, pp. 56–75.

[22] W. Schreiner, Logic and semantic technologies for computer science education, in: 2019 IEEE 15th International Scientific Conference on Informatics, IEEE, 2019, pp. 415–420.

[23] M. Pereira, M. Mernik, D. Cruz, P. Rangel Henriques, Program comprehension for domain-specific languages, Comput. Sci. Inf. Syst. 5 (2008) 1–17. doi:10.2298/CSIS0802001P.

[24] W. Steingartner, V. Novitzká, Operational semantics in a domain-specific robot control language: A pedagogical use case, Computer Science and Information Systems 21 (2024) 1077–1095.

[25] Operics, Operics: Online tool for natural semantics, n.d. URL: https://operics-frontend-g7mvk5367q-od.a.run.app/en/natural, accessed: 2024-07-14.