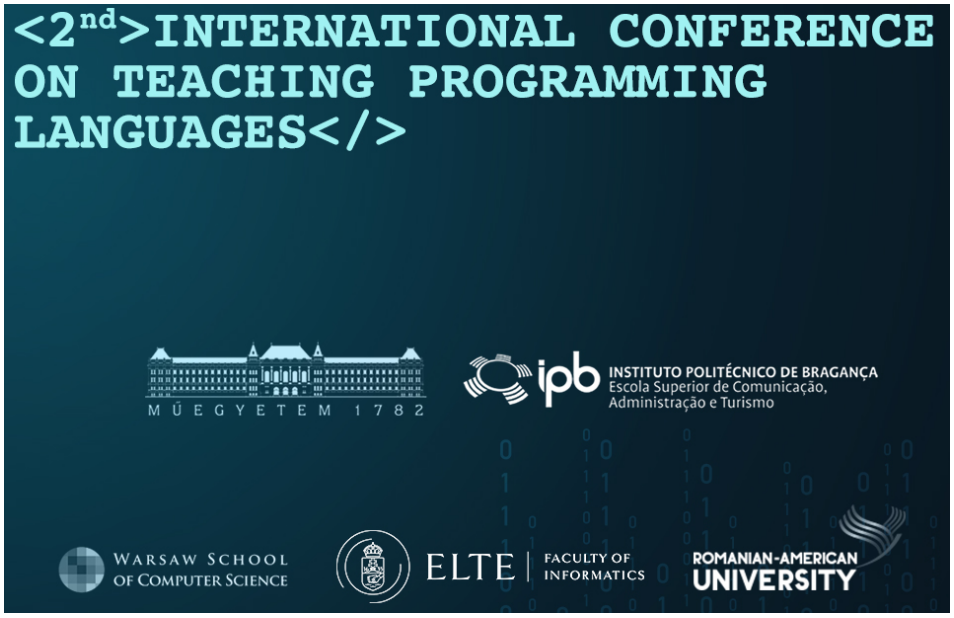
2nd International Conference on Teaching Programming Languages

Friday, 5 April 2024



Book of Abstracts

After having our first hybrid conference last year, we are delighted to have a second occasion to work together. This year it became clear, that the original topic of teaching computer programming languages can be as diverse as any other. The initial target audience consisting of university and secondary school teachers already needs to encompass current advances from the most ancient topics, such as functional programming (exemplified by Lisp descendants like Erlang and Elixir), to the most contemporary statistical methods used in artificial intelligence applications.
To answer this demand and hoping that we will meet the challenge to hold suitable tracks filled with all the interesting sessions in the future, we wish every participant to have fruitful discussions.

The Organisers

Editors:
Dr Carlos Rompante Cunha, EsACT-Polytechnic Institute of Bragança, Mirandela, Portugal
Dr Grzegorz Słowiński, Warsaw School of Computer Science, Warsaw, Poland
Dr Péter Somogyi, Budapest University of Technology and Economics, Hungary

Contents

1

Bringing Code to Life: Interactive Simulations for Learning Programming

Author: Andrei Luchici[1]

[1] *Romanian-American University, School of Computer Science for Business Management, Bucharest, Romania*

Corresponding Author: andrei.luchici@rau.ro

In our contemporary world, digital technology is ubiquitous and constantly advancing. This progression is mirrored by a dynamic evolution of programming languages. Newcomers to the field often grapple with these rapidly changing landscapes and technological demands, which can hinder the establishment of a solid foundational understanding and delay productive engagement. To navigate these challenges, we introduce an innovative approach to teaching programming: interactive simulations. By engaging with simulations, learners can immediately apply their knowledge to replicate complex physical, socio-economic, or biological systems, thereby bypassing the arduous initial learning phase. This hands-on method ignites interest and anchors learning in real-world contexts, paving the way for a pedagogy underpinned by clear, achievable, and measurable objectives (SMART). Ultimately, we posit that simulations present a compelling pedagogical alternative for cultivating programming understanding, fostering a more intuitive and purpose-driven educational experience. Our presentation will showcase a variety of practical applications and explore the educational framework behind interactive simulations, demonstrating how they can revolutionize the teaching of programming by providing an immersive and goal-oriented learning experience.

2

Integrating ChatGPT for Learning Enhancement: Elevating AI Capabilities in Your WebApp

Author: Gabriel Garais[1]

[1] *Romanian-American University, School of Computer Science for Business Management, Bucharest, Romania*

Corresponding Author: gabriel.garais@rau.ro

This presentation explores the transformative potential of integrating ChatGPT into educational web applications to enhance learning experiences and outcomes. Through practical examples and case studies, we demonstrate how ChatGPT's advanced AI capabilities can personalize education, automate administrative tasks, and provide instant, scalable support for students and educators alike. We delve into the technical and ethical considerations of embedding conversational AI in educational settings, offering insights into the future of AI-driven education.

3

Go Functional! - The Elixir of Programming

Author: Péter Hanák[1]

Co-author: Viktor Gergely [1]

[1] *Budapest University of Technology and Economics, Hungary*

Corresponding Author: hanak@emt.bme.hu

As Niklaus Wirth stated in 1976, 'Algorithms + Data Structures = Programs'. In the world of functional programming, algorithms are functions, and data is immutable. Each function performs a transformation on immutable data passed to it as parameters, returning the result of the transformation. With no mutable data, side effects are effectively excluded.
Repetitions can be realized by recursion; the correctness of a recursive function can be proved by induction. Repetitions can also be achieved with higher-order functions and comprehensions that transform the elements of a collection; in such cases, recursion remains hidden from the programmer.
Recursive algorithms work best on recursive data structures. In modern functional languages, like Elixir, recursive data structures –linear and tree-recursive –can be constructed, with links hidden from the programmer.
Elixir, first published in 2012, is a modern, open-source functional programming language, developed with practical aspects and requirements taking precedence over theoretical ones. The code generated by the Elixir compiler runs on Erlang's virtual machine, BEAM. As the latter supports concurrency, Elixir includes language constructs that make it easy to create and handle lightweight processes, as well as pass messages between processes.
Proponents argue –and we agree –that the first programming language a student learns should be a functional language. Elixir is near-ideal for this purpose: an Elixir program is easy to write, load and run. Importantly, Elixir's ecosystem includes a Livebook web application (similar to Python's Jupyter Notebook), making the teaching and learning programming easier as instruction cells and interactive evaluation cells can alternate in the learning material.
With Elixir, what a student has learned about arithmetic, expressions, math functions and math variables at school need not be unlearned –as is often the case when instructions and procedures with side-effects and mutable data are introduced with imperative programming.
In our presentation, we'll discuss some important features of functional programming with Elixir, while illustrating its use with Livebook.

4

Edux, Your Elixir Education Sandbox

Author: Viktor Gergely[1]

[1] *Budapest University of Technology and Economics, Hungary*

Corresponding Author: lustalista@gmail.com

Since 2021, Elixir has been the functional language taught as part of the Declarative Programming course at BME. Since the first such semester, there has been a continuous demand from students to try out the language using a sandbox without the need of installing any development environment on their laptops. Some web-based public sandbox solutions already existed for Prolog and Erlang, but not for Elixir. The idea came to reality during the summer of 2023 in the form of the Edux application (an Elixir sandbox for educational purposes). Edux comes with a simple and easy-to-use web interface built on HTML, CSS and JavaScript where you can type in your Elixir module in a textbox, and immediately compile and execute it in an interactive Elixir shell. Edux also provides an option to run your custom function calls without compiling any source code.

Besides the core system, a lot of open-source libraries have been released for Elixir by the community of Elixir enthusiasts. Edux serves as an example for an open-source application written in Elixir to run Elixir. By making the source code publicly available, students can get an insight into the simplicity of writing concurrent web applications based on their studies of sequential functional programming in Elixir.

9

Programming Is Not a Game ... Is It?

Author: Rui Pedro Lopes[1]

[1] *ESTiG, Bragança Polytechnic University, Bragança, Portugal*

Corresponding Author: rlopes@ipb.pt

Learning a programming language is a multi-challenge process, where students are required to use a specific syntax, semantics and set of rules to implement algorithms, step-by-step specifications of procedures. They are required to structure a problem in sub-problems or modules, that they need to tackle without loosing the full picture from sight, and work backwards to merge them into a solid solution for the initial problem. This requires strong motivation and effort from the students. Gamification can help with this and, at the same time, contribute to high levels of engagement and persistency, leading to a cycle of try-fail until success. "Programming is not a game···is it?" describes a gamification approach for autonomous learning and constant feedback for learning distributed systems programming. It is composed of a set of timed challenges, where the students can use any resource to complete a programming contest. A student companion, running in a docker container in the student's computer, provides constant monitoring and feedback as well as the goals, guiding the student towards success.

5

The potential of ChatGPT in Teaching/Learning Programming

Author: Carlos Rompante Cunha[1]

[1] *EsACT Bragança Polytechnic University, Mirandela, Portugal*

Corresponding Author: crc@ipb.pt

Artificial Intelligence (AI), which for several years was associated with the domain of laboratory R&D and the back office of large organizations, has recently exploded, with the branch of generative AI.
AI has reached the masses, the common user, who uses it and tries to understand it and take advantage of it.
The implications and impacts of this "new AI"are enormous and multifaceted.
Education systems are ecosystems that will have to know how to live with this new reality. From opinions prohibiting its use to those who see it as another tool available to everyone, it is important to reflect on how we should manage this new reality which, like all new realities, brings challenges and opportunities.
This communication briefly reflects on this new reality and how it could impact the teaching and learning of programming languages.

6

Learning Programming: University vs. Business Courses

Author: Grzegorz Kowalski[1]

[1] *Warsaw School of Computer Science, Warsaw, Poland*

Corresponding Author: gkowalski@ms.wwsi.edu.pl

This presentation will explore my experiences with teaching programming, focusing on the differences between university-style and business course styles.
We'll look into the advantages and disadvantages of each approach and conclude with what I believe to be the most effective way to teach programming based on these comparisons.

7

Teaching Functional Programming

Author: István Bozó[1]

[1] *Faculty of Informatics, Eötvös Loránd University, Hungary*

Corresponding Author: bozo_i@inf.elte.hu

Teaching functional programming at the Faculty of Informatics (ELTE) has a long history. Bachelor computer science students take the course in their first semester, and there are several related courses at master level too.

Several years ago our faculty started a two-semester programme for students from other faculties who are interested in learning informatics. Functional programming is also included in this programme as Programming with functions.

In this presentation I will share my experiences of the challenges I face during teaching these courses.

8

Interactive Teaching of Programming Language Theory with a Proof Assistant

Author: Péter Bereczky[1]

Co-author: Dániel Horpácsi [1]

[1] *Faculty of Informatics, Eötvös Loránd University, Hungary*

Corresponding Author: berpeti@inf.elte.hu

The teaching of programming language theory has a long track record at ELTE Faculty of Informatics. Traditionally, formal semantics and type systems of programming languages, similarly to other theory-oriented subjects, were taught with the pen-and-paper method. However, modern proof assistants call for replacing this old-fashioned way of teaching with novel and interactive methods that bring deeper understanding, provide a better learning experience and build technical skills in applying formal methods. From 2019 on we implemented this paradigm change in multiple programming language theory subjects for the benefit of both the students and the instructors. In this talk, we present the main challenges of bringing machine-checked proofs into the practice classes and share our experiences from the past five years of teaching the Formal semantics course.