Supporting FLAT concepts in Learn-OCaml seeing is believing, programming is understanding

Artur Miguel Dias, António Ravara, Simão Melo de Sousa
{amd|aravara}@fct.unl.pt, desousa@di.ubi.pt
Universidade Nova de Lisboa, Universidade da Beira Interior
NOVA-LINCS

June 10, 2022

Introduction

We present the motivation and principles that lead to the development of support for exercises on Formal Languages and Automata Theory in Learn-OCaml, its integration with OFLAT, our web based teaching environment (mostly implemented in OCaml), its uses in the classroom in two Portuguese Universities, and in supporting students independent work.

A teaching challenge. Our context is courses on Formal Languages and Automata Theory (FLAT) for undergraduates. It is a demanding teaching situation: big enrolment numbers / reduced teaching staff; challenging concepts, often perceived by CS students as far from their interests.

Teaching principles. Our goals are mainly threefold: (1) put programming in the centre of the learning process; (2) engaging students with "modern" tools, complementary to the classical textbooks; (3) provide automatic feedback to their coursework and give the necessary information to the teacher to give precise feedback to the student in a timely manner.

We advocate a programming-centric approach: the majority of concepts we teach in our FLAT courses are algorithmic, usually presented in natural language, with set theoretic ways or pseudo-code. Our view, instead, is to write formally the algorithms, in an executable language close to their mathematical definition, in the cases this is possible – OCaml is an obvious choice, and this even more so when functional programming is introduced early on our courses.

Engaging students demands effective practical labs: interactive platforms to see the concepts in action. To comprehend the concepts, checking and creating examples plays a central role in the learning process. Feedback is essential, but given the very high ratio of teacher/student, automatic tools are a solution:

provide automatic evaluation and report; when teacher feedback is required, the environment should gather all needed information and point the teacher to the student needs.

Design principle. One central design principle is to offer as often as possible multiple and interchangeable views to a concept. An automaton, for instance, can be viewed/designed/manipulated as a graph in a graphical interface, but also structurally defined by a convenient textual representation or programmatically defined as an element of an adequate data structure and then manipulated by a provided API or student programmed functions.

This multimodality is a key and distinctive concept in our courseware. In order to support the programming modality and classroom management, we choose to use and extend the Learn-OCaml platform (LO for short) since it provides a rich environment for OCaml programming classes, including a feature-full online programming environment, automatic exercises/programs evaluation and grading, classroom management, etc.

Architecture and implementation

We developed a web base teaching environment that combines interactive and graphical aspects with grading facilities and class management. The support for FLAT concepts is done by: (1) providing LO with an adequate implementation, with the help of the extant library OCamlFLAT; (2) interfacing LO with an extant full-featured interactive graphical FLAT application (OFLAT).

Technically, the first point uses translation of FLAT concepts to the existing LO exercise formats, so no change to the LO core was required. The OCamlFLAT library needs to be loaded.

The second point requires the implementation of a software bus that makes possible for OFLAT and LO to transparently share and synchronize on what the student has done. The student should be able to define, for instance, a finite automaton in the graphical application and check it in LO, and vice-versa. OFLAT is a purely client-side web-application, mainly developed with js-of-ocaml, linking with the functions of OCamlFLAT and using the Cytoscape.js library (via OCaml bindings) to graphically present the concepts.

Conclusions and future work

The courseware is in use for two years in two Portuguese universities, with about 350 students per year. A comprehensive and rigorous assessment is still missing, but the informal feedback is very positive and makes clear needed improvements: further programming exercises and more animations.

Acknowledgements

The Tezos Foundation funded the initial version of OFLAT. The OCaml/INRIA Foundations funded the extension and integration to the Learn-OCaml Platform. In addition to the authors, collaborated in the project our students João Gonçalves (OCamlFLAT), Rita Macedo (OFLAT v1) and Eduardo Silva (OFLAT v2).