Automating Repetitive Tasks for the Masses

Sumit Gulwani
Microsoft Research
sumitg@microsoft.com

Abstract

The programming languages (PL) research community has traditionally catered to the needs of professional programmers in the continuously evolving technical industry. However, there is a new opportunity that knocks our doors. The recent IT revolution has resulted in the masses having access to personal computing devices. More than 99% of these computer users are non-programmers and are today limited to being passive consumers of the software that is made available to them. Can we empower these users to more effectively leverage computers for their daily tasks? The formalisms, techniques, and tools developed in the PL and the formal methods research communities can play a pivotal role!

Categories and Subject Descriptors D.1.2 [*Software*]: Programming Techniques—Automatic Programming; K.3.1 [*Computing Milieux*]: Computers and Education—Computer Uses in Education

Keywords End-user Programming; Computer-aided Education; Programming by Examples; Programming by Natural Language; Program Synthesis; Domain-specific Languages; Search Algorithms

What do masses struggle with?

There are several domains of repetitive tasks that large populations of people (with access to computational devices) struggle with in their daily lives. These can be identified via study of help forums [6], analyzing search engine query logs [9], conducting user studies, or partnering with domain experts. Following are two broad areas with great need to automate repetitive tasks.

End-user Programming: Empowering IT workers

End users of computational devices, most of whom are non-programmers, often need to create small (and perhaps one-off) scripts to automate repetitive tasks. These users can easily specify their intent using examples and/or natural language. The opportunity here is to develop program synthesis techniques [3] that can translate the user's specification in the form of examples or natural language (which is often imprecise or ambiguous) into intended scripts. While program synthesis has been an old area of study, its application to end-user programming is very timely now, given the increased significance of this application, and the recent advances in computational power and algorithms to address the associated technical challenges.

Programming by Examples (PBE) Several data manipulation tasks, including extraction [7], transformation [6] and formatting [10], are amenable to programming by examples. The IT revolution has resulted in digitization of massive amounts of data. Data is available in documents of various types, e.g., text/log files, spreadsheets, webpages, xml, pdf, and images. These documents offer great flexibility in storing and organizing hierarchical data by combining presentation/formatting with the underlying data model. However, this makes it hard to manipulate and reason about the underlying data. PBE has the potential to convert data wrangling into a delightful experience, thereby enabling users to discover new actionable insights about their data.

Programming by Natural Language (PBNL) Natural language is a good fit for describing actions with side-effects as is the case in smartphone automation scripts [8]. Some data manipulation tasks such as filtering and summarization, are also best communicated using natural language [5].

Computer-aided Education: Empowering teachers & students

Education has been one of the least technically impacted sectors. With recent disruption in the form of online educational platforms, blended classrooms, and one-tablet-per-child programs, technology stands to play a transformative role. For instance, technology can assist with repetitive tasks in Education like problem generation and feedback generation, for a variety of subjects including programming, algorithms, logic, mathematics, and language learning [4]. This can facilitate interactive and adaptive pedagogies in both standard and online classrooms.

Problem generation Generating fresh problems that have specific solution characteristics (e.g., difficulty level, use of a certain set of concepts) is a tedious task for the teacher. Automating it can enable personalized workflows for students and also help prevent plagiarism (each student can be provided with a different problem with the same characteristics).

Feedback generation This involves identifying whether the student's solution is incorrect and, if so, the nature of the error and potential fix. Automating it can save teachers time, and enable consistency in grading. It can also be used to provide immediate feedback to students thereby improving student learning.

What role can PL technologies play?

Domain-specific languages (DSL) The classic problem (in the PL community) of designing an appropriate domain-specific language is relevant to both End-user programming and Education domains. Several PBE and PBNL systems involve an underlying DSL that describes the space of programs to which the user's specification is mapped. Such a DSL is based on abstractions that can be

POPL '15, January 15–17, 2015, Mumbai, India.

ACM 978-1-4503-3300-9/15/01.
http://dx.doi.org/10.1145/2676726.2682621

used to *succinctly* describe the various tasks that the users hope to accomplish. On one hand, the DSL should be expressive enough to describe real-world tasks in the domain, while, on the other hand, it should be restricted enough to enable efficient synthesis. In case of problem generation, declarative DSLs allow for describing various kinds of (syntactic or solution-based) features associated with the problems that the teacher aims to generate.

Symbolic reasoning and Search techniques Various symbolic reasoning and search techniques have been developed in the PL community as part of research on invariant generation and constraint solving. Such techniques can be brought to bear to deal with the challenging problem of searching for programs in a DSL that match the user's specification (in case of both PBE and PBNL). Such techniques can also be useful for generating models to queries expressed in declarative languages (in case of both End-user programming and Education domains).

Test input generation techniques Automated test input generation is a well-studied topic in the PL community with great practical utility. Interestingly, these techniques can be useful in both End-user programming and Education domains. In End-user programming, they can drive an active learning user interface that prompts the user about behavior on *distinguishing* inputs that differentiate the multiple programs learned from the user's imprecise specification [3]. In Education, they can be used to generate counterexample based feedback for solutions to *conceptual* problems including constructions (of automata, grammars, programs) and proofs (in algebra, geometry). These techniques can also be used to generate practice problems for *procedural* content such as mathematical procedures taught in middle/high school (e.g., addition, long division, Gaussian elimination) and algorithmic procedures taught in undergraduate computer science, where students are expected to demonstrate their understanding of certain classic algorithms on specific inputs (e.g., breadth-first search, insertion sort, or regular expression to automaton conversion) [4]. For instance, the various concepts in teaching the addition procedure (such as adding single digits, adding numbers without carry, with a single carry, or with a double carry) correspond to specific sets of paths inside the loopy procedure for adding two numbers represented as arrays of digits—test input generation techniques can be used to generate inputs that traverse such a given set of paths.

What are some future directions?

Meta-frameworks Developing a robust end-to-end experience for a given domain often requires critical domain knowledge and non-trivial implementation effort. A key future direction is to develop general frameworks that can enable easy implementation of domain-specific technologies (for related domains in PBE, PBNL, problem generation, or feedback generation), say from a declarative specification of the domain knowledge. For instance, can we develop frameworks that allow construction of efficient synthesizers from a mere description of the underlying DSL, similar to how declarative parsing frameworks allow compiler writers to construct a parser from a mere description of the syntax of the underlying language. The SyGuS [1], Rosette [11], and FlashExtract [7] frameworks are great initial efforts in this direction.

Data-driven research Leveraging large-scale field data to improve the robustness of systems is another key aspect. For instance, in case of Education domains, the student data can be leveraged to obtain different correct solutions to a problem (which in turn can be used to generate feedback [2]), or to discover effective learning pathways to guide problem selection. In case of End-user programming domains, the knowledge of common idiomatic computations (obtained from a large collection of programs constructed by end users) can be used to guide ranking of programs learned from user's imprecise or ambiguous specification.

Multi-modality and Interactivity Another useful direction is to develop *multi-modal* systems that take as input various forms of specifications such as examples, logical specifications, natural language, and speech. An orthogonal challenge is to develop *interactive* systems that support debugging and allow the user to split the task into multiple steps in case of failure to completely automate the task. There is good inspiration to be drawn from work on interactive theorem proving and work that combines different modes of analysis such as static/deductive and dynamic/inductive.

Acknowledgments

I thank Rishabh Singh and Ben Zorn for feedback on this article. I thank all my collaborators on projects in End-user programming and Education for believing in and driving the shared vision mentioned in this article.

References

[1] R. Alur, R. Bodík, G. Juniwal, M. M. K. Martin, M. Raghothaman, S. A. Seshia, R. Singh, A. Solar-Lezama, E. Torlak, and A. Udupa. Syntax-guided synthesis. In *FMCAD*, 2013.

[2] E. Fast, C. Lee, A. Aiken, M. S. Bernstein, D. Koller, and E. Smith. Crowd-scale interactive formal reasoning and analytics. In *UIST*, 2013.

[3] S. Gulwani. Dimensions in program synthesis. In *PPDP*, 2010.

[4] S. Gulwani. Example-Based Learning in Computer-Aided STEM Education. *CACM*, 2014.

[5] S. Gulwani and M. Marron. NLyze: Interactive programming by natural language for spreadsheet data analysis and manipulation. In *SIGMOD*, 2014.

[6] S. Gulwani, W. Harris, and R. Singh. Spreadsheet data manipulation using examples. *CACM*, 2012.

[7] V. Le and S. Gulwani. FlashExtract: A framework for data extraction by examples. In *PLDI*, 2014.

[8] V. Le, S. Gulwani, and Z. Su. Smartsynth: Synthesizing smartphone automation scripts from natural language. In *MobiSys*, 2013.

[9] O. Polozov and S. Gulwani. LaSEWeb: Automating search strategies over semi-structured web data. In *KDD*, 2014.

[10] M. Raza, S. Gulwani, and N. Milic-Frayling. Programming by example using least general generalizations. In *AAAI*, 2014.

[11] E. Torlak and R. Bodík. A lightweight symbolic virtual machine for solver-aided host languages. In *PLDI*, 2014.

Biography

Sumit Gulwani is a principal researcher at Microsoft Research, Redmond. He has expertise in formal methods and automated program analysis and synthesis techniques. His recent research interests lie in the cross-disciplinary areas of automating end-user programming (for systems like spreadsheets, smartphones, and robots), and building intelligent tutoring systems (for various subject domains including computer science, mathematics, and language learning). His programming-by-example work led to the Flash Fill feature in Microsoft Excel 2013 that is used by hundreds of millions of people. He was awarded the ACM SIGPLAN Robin Milner Young Researcher Award in 2014. He obtained his PhD in Computer Science from UC-Berkeley in 2005, and was awarded the ACM SIGPLAN Outstanding Doctoral Dissertation Award. He obtained his BTech in Computer Science and Engineering from IIT Kanpur in 2000, and was awarded the President's Gold Medal.