SpeX: a rewriting-based formal-specification environment*

Ionuţ Ţuţu

Simion Stoilow Institute of Mathematics
of the Romanian Academy, Romania
ittutu@gmail.com

The development of new formal-specification languages is often a necessary yet challenging, even arduous, task. Declarative logical frameworks, such as LF [5], MMT [8], and RL [6], facilitate this process by means of highly expressive meta-languages and tools through which a wide array of logical systems and calculi can be represented and reasoned about. This representational approach makes it easy to provide generic tool support for newly developed formalisms, but it requires both language developers and end-users to be familiar with the logical framework of choice. On the other hand, systems such as the $\mathbb{K}$ framework [9] (which deals primarily with the design and analysis of programming languages) and Hets [7] (the Heterogeneous Tool Set, which provides an integrating framework of multiple logical systems, together with proof tools and logic translations) feature a clear separation between the meta-language utilized by system developers and the specification language and tools offered to end-users.

In this work, we explore a similar route to that of $\mathbb{K}$ and Hets in order to develop a rewriting-based environment, called SpeX, for working with formal specifications. This includes, for example, tool support for parsing and for analysing specifications, as well as automatically generated interpreters. However, unlike $\mathbb{K}$, the environment we propose targets specification languages and is inherently heterogeneous; and unlike Hets, for which specifications are built over logical systems formalized as institutions [3] by means of a fixed set of structuring constructs [10], SpeX admits a much weaker notion of 'language', enabling us to capture, for instance, comorphisms of structured institutions [11] where the structuring mechanisms can change as well along language translations.

Despite these small advancements, the basic functionality of SpeX is modest compared to any of the tools and frameworks mentioned above. Its main asset is the environment's potential to be easily extended in order to accommodate new specification languages or features, many of which may be experimental. That is, the purpose of SpeX is distinctly academic, aiming to help bridge the gap between the theory and practice of formal specification and verification by providing researchers in the area with an environment that encourages prototyping and testing ideas and techniques even from early stages of development. To that end, we introduce a suite of software libraries, all implemented in Maude [1], that support the integration of new formal-specification languages.

* This work was supported by a grant of the Romanian Ministry of Education and Research, CCCDI – UEFISCDI, project number PN-III-P2-2.1-PED-2019-0955, within PNCDI III.

From an architectural standpoint, SpeX consists of a small supervisory kernel that manages input/output operations and, most importantly, hosts a number of information processors – one for each specification language that is integrated into the environment. Some processors are concrete, pertaining to a given logical system (say, equational or first-order logic), while others are generic, allowing various combinations of specification-building operators to be defined on top of base logical systems that meet certain requirements. Therefore, for any instance of SpeX, the capabilities of the environment are dictated by the processors and corresponding languages it hosts. At most one of those processors can actively take part in a user interaction at a given time, and the active processor may change as a result of the interaction, hence SpeX may react differently (even to the same input) depending on which processor is currently selected.

To illustrate the approach and the steps needed in order to extend the environment, we consider a new language based on hidden algebra [4] that allows for the specification of hierarchical compositions of behavioural objects [2]. This includes the development of new parsers and analysis tools for hidden-algebra declarations and for hierarchical compositions, as well as preliminary proof-theoretic support, all of which are integrated within SpeX.

References

1. Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Talcott, C.L. (eds.): All About Maude – A High-Performance Logical Framework, How to Specify, Program and Verify Systems in Rewriting Logic, Lecture Notes in Computer Science, vol. 4350. Springer (2007)
2. Diaconescu, R.: Behavioural specification for hierarchical object composition. Theoretical Computer Science **343**(3), 305–331 (2005)
3. Goguen, J.A., Burstall, R.M.: Institutions: Abstract model theory for specification and programming. Journal of the ACM **39**(1), 95–146 (1992)
4. Goguen, J.A., Malcolm, G.: A hidden agenda. Theoretical Computer Science **245**(1), 55–101 (2000)
5. Harper, R., Honsell, F., Plotkin, G.D.: A framework for defining logics. Journal of the ACM **40**(1), 143–184 (1993)
6. Martí-Oliet, N., Meseguer, J.: Rewriting logic as a logical and semantic framework. In: Meseguer, J. (ed.) First International Workshop on Rewriting Logic and its Applications, WRLA 1996. Electronic Notes in Theoretical Computer Science, vol. 4, pp. 190–225. Elsevier (1996)
7. Mossakowski, T., Maeder, C., Lüttich, K.: The heterogeneous tool set (Hets). In: Beckert, B. (ed.) Proceedings of 4th International Verification Workshop in connection with CADE-21. CEUR Workshop Proceedings, vol. 259. CEUR-WS.org (2007)
8. Rabe, F., Kohlhase, M.: A scalable module system. Information and Computation **230**, 1–54 (2013)
9. Rosu, G.: Matching logic. Logical Methods in Computer Science **13**(4) (2017)
10. Sannella, D., Tarlecki, A.: Specifications in an arbitrary institution. Information and Computation **76**(2/3), 165–210 (1988)
11. Ţuţu, I.: Comorphisms of structured institutions. Information Processing Letters **113**(22-24), 894–900 (2013)