KRunner: Linking Rascal with K

Mark Hills[1] Paul Klint[2] Jurgen J. Vinju[3]

Centrum Wiskunde & Informatica
Amsterdam, The Netherlands

INRIA Lille Nord Europe
Lille, France

Using the K framework [6,15], it is possible to define the semantics of programming languages and language calculii. This includes the semantics of a number of "real-world" or paradigmatic languages and language subsets, such as Verilog [12], KOOL [8], SILF [6,9], and KERNELC [14], a core of the C language. These definitions have been used for a number of purposes, including to provide semantics-based interpreters, program analysis tools, and verification environments such as matching logic [13].

Like most semantics frameworks, K focuses on assigning semantics to the abstract syntax of a program, not to its concrete syntax. Because of this, the current K tool suite [5,1] provides very little support for language front-ends, instead assuming that programs will be given in (or transformed into) a format easily consumed by Maude [4]. Front-ends are then created on an ad-hoc basis, using a number of different lexers, parsers, and pretty-printers. Graphical front-ends are also not directly supported, meaning that the typical user of a definition either uses Maude directly, uses some other console-based tool (such as an execution script), or uses a custom graphical front-end. This leads to a potentially poor user experience where, for instance, the user needs to work backwards from the given error messages, potentially through the generated version of the program, back to her original program, in order to find the actual source of an error message. This also leads to difficulties in distributing language definitions, which may require a number of tools to be

[1] Email: Mark.Hills@cwi.nl
[2] Email: Paul.Klint@cwi.nl
[3] Email: Jurgen.Vinju@cwi.nl

bundled with the definition or installed separately.

The solution explored in this abstract is to provide the language front-end and user interface integration using the Rascal meta-programming language [11,10]. Rascal provides a number of features needed to build front-ends that can work with K language specifications. For lexing and parsing, Rascal provides a grammar notation which can be used to generate scannerless GLL-based parsers. Generated parse trees can then be manipulated using matching over concrete syntax patterns and standard (not parsing-specific) features of Rascal, including structure-shy traversals, string interpolations, rich built-in data types (e.g., sets, relations, lists, and tuples), pattern matching, user-defined algebraic data types, and higher-order functions. Rascal integration with the Eclipse IDE via IMP [3,2] provides an IDE for source programs in the defined language as well as interactive features for running and analyzing programs and for displaying results (output values, discovered errors, etc).

This work is based on the RLSRunner tool [7], which focused on providing support for K (or earlier, K-style) definitions running directly in Maude. The tool described here is similar to RLSRunner, but is intended to instead work directly with K language definitions, instead of only working with definitions already converted into Maude format. This should make it easier to take advantage of current work on K, including (potentially) execution engines outside of Maude.

References

[1] `http://code.google.com/p/k-framework/`.

[2] P. Charles, R. M. Fuhrer, S. M. S. Jr., E. Duesterwald, and J. J. Vinju. Accelerating the Creation of Customized, Language-Specific IDEs in Eclipse. In *Proceedings of OOPSLA'09*, pages 191–206. ACM, 2009.

[3] P. Charles, R. M. Fuhrer, and S. M. Sutton. IMP: A Meta-Tooling Platform for Creating Language-Specific IDEs in Eclipse. In *Proceedings of ASE'07*, pages 485–488. ACM Press, 2007.

[4] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott, editors. *All About Maude - A High-Performance Logical Framework, How to Specify, Program and Verify Systems in Rewriting Logic*, volume 4350 of *LNCS*. Springer-Verlag, 2007.

[5] T. F. Şerbănuţă and G. Roşu. K-Maude: A Rewriting Based Tool for Semantics of Programming Languages. In *Proceedings of WRLA 2010*, volume 6381 of *LNCS*, pages 104–122. Springer-Verlag, 2010.

[6] M. Hills, T. F. Şerbănuţă, and G. Roşu. A Rewrite Framework for Language Definitions and for Generation of Efficient Interpreters. In *Proceedings of WRLA'06*, volume 176(4) of *ENTCS*, pages 215–231. Elsevier, 2007.

[7] M. Hills, P. Klint, and J. Vinju. RLSRunner: Linking Rascal with K for Program Analysis. In *Proceedings of SLE'11*, LNCS. Springer-Verlag, 2011. To Appear.

[8] M. Hills and G. Roşu. KOOL: An Application of Rewriting Logic to Language Prototyping and Analysis. In *Proceedings of RTA'07*, volume 4533 of *LNCS*, pages 246–256. Springer-Verlag, 2007.

[9] M. Hills and G. Roşu. A Rewriting Logic Semantics Approach To Modular Program Analysis. In *Proceedings of RTA'10*, volume 6 of *Leibniz International Proceedings in Informatics*, pages 151 – 160. Schloss Dagstuhl - Leibniz Center of Informatics, 2010.

[10] P. Klint, T. van der Storm, and J. Vinju. RASCAL: A Domain Specific Language for Source Code Analysis and Manipulation. In *Proceedings of SCAM'09*, pages 168–177. IEEE, 2009.

[11] P. Klint, T. van der Storm, and J. Vinju. EASY Meta-programming with Rascal. In *Post-Proceedings of GTTSE'09*, volume 6491 of *LNCS*, pages 222–289. Springer-Verlag, 2011.

[12] P. O. Meredith, M. Katelman, J. Meseguer, and G. Roşu. A formal executable semantics of verilog. In *Proceedings of MEMOCODE 2010*, pages 179–188. IEEE Computer Society, 2010.

[13] G. Roşu, C. Ellison, and W. Schulte. Matching Logic: An Alternative to Hoare/Floyd Logic. In *Proceedings of AMAST'10*, volume 6486 of *LNCS*, pages 142–162. Springer-Verlag, 2011.

[14] G. Roşu, W. Schulte, and T. F. Şerbănuţă. Runtime Verification of C Memory Safety. In *Proceedings of RV 2009*, volume 5779 of *LNCS*, pages 132–151. Springer-Verlag, 2009.

[15] G. Roşu and T. Şerbănuţă. An Overview of the K Semantic Framework. *Journal of Logic and Algebraic Programming*, 79(6):397–434, 2010.