Available online at www.sciencedirect.com
SCIENCE DIRECT®
Electronic Notes in Theoretical Computer Science
ELSEVIER Electronic Notes in Theoretical Computer Science 110 (2004) 177–180
www.elsevier.com/locate/entcs

The Action Environment — Tool Demonstration —

Mark van den Brand

Department of Software Engineering, CWI
Kruislaan 413, NL-1098 SJ Amsterdam, The Netherlands
and
Instituut voor Informatica en Electrotechniek, Hogeschool van Amsterdam
Weesperzijde 190, NL-1097 DZ Amsterdam, The Netherlands

Jørgen Iversen, Peter D. Mosses

BRICS & Department of Computer Science [1]
University of Aarhus, IT-parken, Aabogade 34, DK-8200 Aarhus N, Denmark

Introduction

When writing semantic descriptions of programming languages it is highly desirable to reuse descriptions of constructs from previous language descriptions. This is usually not possible without adaptations, due to the lacking modularity of the formalism used. In [3] Doh and Mosses proposed organizing language descriptions such that each construct is described in a separate module, which promotes reuse. They used Action Semantics [5] and ASF+SDF [4] as description language, but experience showed that the notation was too cumbersome, so we developed the ASDF language, which is tailor-made for writing action semantic descriptions (ASDs) of single language constructs. Here we will demonstrate an environment which supports working with ASDF modules.

[1] Basic Research in Computer Science (`www.brics.dk`), funded by the Danish National Research Foundation.

doi:10.1016/j.entcs.2004.06.002

ASDF

Using ASF+SDF for writing ASDs was tedious because one has to explicitly import modules that were obviously needed and write many superfluous keywords. Furthermore some parts of the description which logically belong together were not grouped together.

The new formalism ASDF alleviates these problems. An ASDF module consists of a name, possibly some explicit imports of other modules, and a sequence of sections. Typically a module has three sections: a section where the abstract syntax of the language construct is defined, a section where auxiliary notation for use in semantic functions is defined and a section where the semantic function mapping the language construct to an action is defined. Figure 1 shows two examples of ASDF modules.

Environment

The Action Environment has a GUI (see Fig. 1) where the focus is to give the user an overview of the open ASDF modules. This is done by visualising the import graph and by listing the hierarchical module names in a collapsible tree structure. Working with the module text is done using Emacs with an extension that allows structure editing.

When a module text contains a sort symbol, the unique module which declares that sort is imported, allowing most imports to be left implicit, cf. Fig. 1. The transitive closure of the modules imported by a module define an abstract syntax of a language, and it is possible to parse terms of this language over the module. Furthermore the mapping from the abstract syntax to actions, defined by the semantic functions, can be executed such that the environment becomes a language-to-actions compiler.

A full ASDF description of a complete programming language involves many small modules, and the demo will show the user can locate and work with the modules of interest.

Implementation

The Action Environment is built on top of the ASF+SDF Meta-Environment [2]. This was possible due to the configurability and the layered architecture of the ASF+SDF Meta-Environment; we simply added an ASDF layer on top of it and reconfigured the menus of the environment. The ASDF layer consists of components for parsing ASDF modules, for generating ASF+SDF modules from ASDF modules and for retrieving information like module name etc. from ASDF modules. Figure 2 shows how the architecture of the environment is divided into layers. All communication between the different layers and components in the layers is handled by the ToolBus [1].

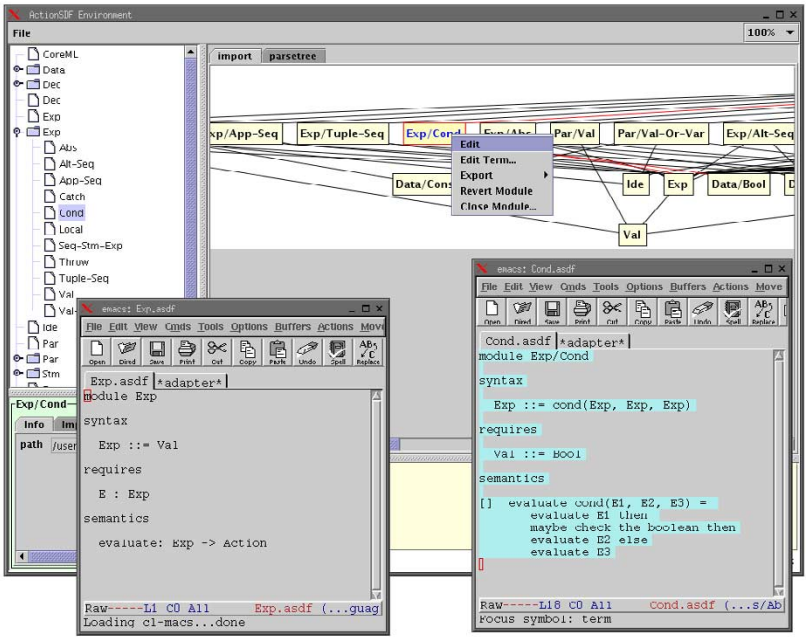

Fig. 1. The Action Environment

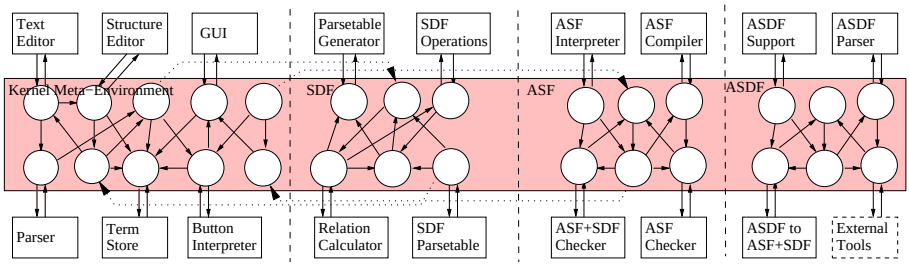

Fig. 2. The layered architecture of the Action Environment

The concrete syntax of a language and its mapping to abstract syntax are defined directly in ASF+SDF. We are planning to implement support for working simultaneously with ASF+SDF and ASDF modules in the Action Environment such that full language descriptions are supported.

Future work

The tool has already been used to support the development of a description of Standard ML.

Our next goal is to use the Action Environment as the front end of a compiler generator. The Action Environment supports the definition of the

abstract syntax of a language and the mapping from this abstract syntax to actions, so we need to connect an action-to-target-language compiler. An existing type-checker for actions is currently being integrated with the Action Environment.

References

[1] J. A. Bergstra and P. Klint. The discrete time ToolBus – A software coordination architecture. *Sci. Comput. Programming*, 31(2-3):205–229, 1998.

[2] M. G. J. van den Brand, P. Moreau, and J. J. Vinju. Environments for term rewriting engines for free! In *RTA 2003*, LNCS Vol. 2706, pages 424–435. Springer, 2003.

[3] K.-G. Doh and P. D. Mosses. Composing programming languages by combining action-semantics modules. *Sci. Comput. Programming*, 47(1):3–36, 2003.

[4] J. Heering, P. R. H. Hendriks, P. Klint, and J. Rekers. The syntax definition formalism SDF: Reference manual. *SIGPLAN Notices*, 24(11):43–75, 1989.

[5] P. D. Mosses. *Action Semantics*. Cambridge Tracts in Theoretical Computer Science 26. Cambridge University Press, 1992.