2.4 Towards an Evolutionary Formal Software-Development Using CASL

Serge Autexier[1], Dieter Hutter[2], Heiko Mantel[2], and Axel Schairer[2]

(1) Saarland University, P.O. Box 151150, D-66041 Saarbrücken, Germany

(2) German Research Center for Artificial Intelligence, Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany

Email: (1) `serge@ags.uni-sb.de`

(2) {`hutter,mantel,schairer`}`@dfki.de`

It has long been recognised that specifications in the large are only manageable if they are built in a structured way. Specification languages, like CASL [2], provide various mechanisms to combine basic specifications to extensive structured specifications. Analogously verification tools have to provide appropriate mechanisms to structure the correspondent logical axiomatisations and different inference mechanisms. In practice, a formal program development is an evolutionary process [4]. Specification and verification are mutually intertwined. Failed proofs give rise to changes of the specification which in turn will render previously found proofs invalid. For practical purposes it is indispensable to restrict the effects of such changes to a minimum in order to preserve as much proof effort as possible during a change of the specification.

The aims of this paper are twofold. First, we present the notion of a development graph which forms the proof theoretical basis for the representation of formal developments. A development graph is a directed graph with different kinds of directed links. The nodes in a development graph consists of a consequence relation (specifying the underlying logic) and a set of local axioms. Directed links between nodes are labelled by consequence morphisms and allow one to relate theories to each other. Four kinds of directed links occur in a development graph: Global definition links include the mapped axioms of the source node as additional axioms to the target node while global theorem links postulate that the mapped axioms are theorems within the target node. These global theorem links allow for a representation of properties like "satisfies" or "implements". In order to support an efficient management of change, the global links are decomposed into local links between theories, which enables a fine grained localisation of the effects of changes in a specification.

In a second part, we present how CASL specifications are transformed into development graphs via an explicit representation of the specification's structure, called theory representations (TR), cf. for example [3]. It makes explicit the structure and content of the specification text that is relevant for the deductive process in a canonicalized way. These TRs are then translated into the corresponding development graphs. When changing the specification text, a new TR is computed and is compared to the previous one. Differences between the two are then translated into the necessary changes to the actual development graph. When these changes take effect the management of change defined over development graphs ensures that the overall development enters a new consistent state. Thus the representation of the specification's structure via TRs connects Casl specifications and their representation in the development graph.

The described framework has been implemented within the INKA 5.0 system and is the central part to represent and manage formal software developments. The development graph is an experiment in providing a general structure meeting the requirements arising from the representation of structured specifications, the evolutionary aspects of the specification process and the need for a sophisticated and strong deductive support to prove properties about specifications.

References

[1] S. Autexier, D. Hutter, H. Mantel, A. Schairer: System Description: INKA 5.0 - A Logic Voyager. In H. Ganzinger, CADE-16, Springer, LNAI 1632, 1999

[2] CoFi-task group on language design, ESPRIT working group 29432, EU, 1998. CoFI Webpage: http://www.brics.dk/Projects/CoFI/

[3] R. Harper, D. Sannella, A. Tarlecki: Structured presentations and logic representations. In Annals of Pure and Applied Logic, 67:113-160, 1994

[4] D. Hutter et al.: Verification Support Environment (VSE), Journal of High Integrity Systems, Vol. 1, 1996.