The SMoLCS ToolSet *

Egidio Astesiano, Gianna Reggio, Franco Morando **

Dipartimento di Informatica e Scienze dell'Informazione, Università di Genova, Italy

SMoLCS is a formal method, based on algebraic specifications, for specifying *dynamic systems* at different levels of abstraction (from initial requirements till complete design); here by dynamic system we mean a system able to evolve along the time (e.g. a concurrent/parallel/distributed program, a net of computers, a hydroelectric central or an information system).

Formally, a dynamic system is a labelled transition system, which is a triple (S, L, T), where S is a set of states, L a set of labels and T a set of triples (s, l, s'), called transitions, with $s, s' \in S$ and $l \in L$. The states model the intermediate situations of a dynamic system, the labels its possible interactions with the external world and the transitions its possible activity in the sense that each transition corresponds to an action capability. Thus the behaviour of the dynamic system starting from a given state is represented by a labelled transition tree, called execution tree.

Since we need to consider different types of dynamic systems, different ways to compose them and different data structures, we organize all that in an overall algebraic structure, called dynamic algebra. A dynamic algebra is a many-sorted algebra with predicates on a dynamic signature, i.e. a signature where some sorts (dynamic sorts) correspond to states of lts's, and each of them has an associated sort of labels and a ternary predicate corresponding to the transitions.

SMoLCS specifications are pairs consisting of a dynamic signature and a set of axioms, distinguished in: *design specifications* (with conditional axioms and initial semantics), which determine *one* dynamic system by describing abstractly its structure and its activity, including the concurrent cooperations among its components; *requirement specifications* (with axioms of first-order logic extended by temporal combinators and loose semantics) which determine *one class* of dynamic systems by stating their properties, like liveness and safety. Moreover, it is formally defined when a design specification correctly implements a requirement specification. The specifications are written in a user-friendly language METAL, supporting both the design and the requirement level.

A software toolset has been developed to support the development of the SMoLCS specifications, consisting of: a checker of the static correctness of the METAL specifications (METAL-CHECKER); a rapid prototyper (SMoLCS-RP) which, given a design specification of a dynamic system and a state, generates the execution tree starting from such state; and a graphic user interface for SMoLCS-RP (TREE WALKER).

METAL-CHECKER takes in input a SMoLCS specification and gives the detailed list of errors found during the parsing (lexical, syntax and type errors); it also disambiguates overloadings and signals ambiguous formulae. Moreover it translates a

---

* This work has been partly supported by "Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo" of C.N.R. (Italy) and by HCM–Medicis.

** E.Astesiano and G.Reggio have developed the SMoLCS method; F.Morando has developed the basic algorithm for the prototyper and the overall tool architecture.

correct specification into the appropriate internal format for SMoLCS-RP, which generates the execution tree starting from a given state. The user may choose different strategies for building the tree. TREE WALKER shows parts of such tree while SMoLCS-RP is still producing it; thus also dynamic systems with infinite execution tree may be investigated.

The generation of the execution tree requires the solution of queries in an algebraic theory with conditional equations and predicates. Usually, these solutions are obtained by using conditional narrowing (CNA) or similar approaches. Such algorithms are unsuitable for our purposes since they tend to fall into endless proofs too frequently; hence we devised a more specialized algorithm called SCNA, which is a restriction of CNA proven to be complete for a meaningful decidable subclass of SMoLCS specifications called separable. SCNA separates logical deduction and functional evaluation to provide an effective complete replacement of CNA. SMoLCS-RP is able to solve, in a reasonable amount of time, systems of equations in the domain of separable specifications. Thus SMoLCS-RP can build the execution tree and also prototype the static subparts of the specifications (data structures) by evaluating operations and predicates on particular arguments.

The TREE WALKER uses Diagram Server (a tool for visualizing graphs, developed by P. Di Battista and his group at the University of Rome) for visualizing the tree topology and the information associated with the nodes and the arcs. The user may in this phase decide which nodes do not need to be further expanded ("frozen" nodes). The frozen nodes may be melted successively and they become eligible for further expansions. The format with which this information is visualized is defined in a highly-parameterized way by means of unparsing schemas allowing to generate an output following the preferences of the user. The toolset has been mainly developed in Prolog, and there is a small part written in C for the interface with Diagram Server. The Prolog code is written for more than 90% following the DEC-10 standard and thus, in principle, it might be easily portable on other Prolog interpreters and compilers. For what concerns the TREE WALKER graphical interface, we have used the Widget set supplied by the Motif interface of the Quintus Prolog V3.1. The toolset has been developed on a HP9000/700 under UNIX HP-UX A.08.07 with Quintus Prolog 3.1 interfaced with Motif 1.0 and X11 Rel. 3. The graphical interface uses, together with Motif, also Diagram Server V1.1; thus in this stage the toolset is available on machines that support Quintus Prolog 3.1 and Diagram Server V1.1.

The toolset software has been developed by F.Morando, A.Capani, F. Parodi, L.Campora and G.Delzanno. A.Giovini († 93) also contributed with F.Morando to the overall architecture.

References and documentations about SMoLCS are available by anonymous ftp at `ftp.disi.unige.it` in the directory `pub/smolcs`; for further information and any question e-mail to `smolcs@disi.unige.it`.

This article was processed using the LaTeX macro package with LLNCS style