The Common Framework Initiative for algebraic specification and development of software: recent progress

Donald Sannella

Laboratory for Foundations of Computer Science
University of Edinburgh, UK

Abstract. The Common Framework Initiative (CoFI) is an open international collaboration which aims to provide a common framework for algebraic specification and development of software. The central element of the Common Framework is a specification language called Casl for formal specification of functional requirements and modular software design which subsumes many previous algebraic specification languages. This paper is a brief summary of progress on CoFI during the period 1998–2001, when CoFI received funding from the European Commission as a Working Group under the Esprit programme.

1 Introduction

Algebraic specification is one of the most extensively-developed approaches in the formal methods area. The most fundamental assumption underlying algebraic specification is that programs are modelled as *many-sorted algebras* consisting of a collection of sets of data values together with functions over those sets. This level of abstraction is commensurate with the view that the correctness of the input/output behaviour of a program takes precedence over all its other properties. Another common element is that specifications of programs consist mainly of logical *axioms*, usually in a logical system in which equality has a prominent role, describing the properties that the functions are required to satisfy. A wide variety of different approaches to algebraic specification take these two principles as their starting point.

The past 25 years has seen a great deal of research on the theory and practice of algebraic specification. Overviews of this material include [5, 16, 23, 54, 88, 89, 95]. Developments on the foundational side have been balanced by work on applications, but despite a number of success stories, industrial adoption has so far been limited. The proliferation of *algebraic specification languages* is seen as a significant obstacle to the dissemination and use of these techniques. Despite extensive past collaboration between the main research groups involved and a high degree of agreement concerning the basic concepts, the field has given the appearance of being extremely fragmented, with no *de facto* standard specification language, let alone an international standard. Moreover, although many tools supporting the use of algebraic techniques have been developed in the academic community, none of them has gained wide acceptance, at least partly because of their isolated usability: each tool uses a different specification language.

Since late 1995, work has been underway in an attempt to remedy this situation. The *Common Framework Initiative* (abbreviated CoFI) is an open international collaboration which aims to provide a common framework for algebraic specification and development of software. The Common Framework is intended to be attractive to researchers in the field as a common basis for their work, and to ultimately become attractive for use in industry. The central element of the Common Framework is a specification language called Casl (the Common Algebraic Specification Language), intended for formal specification of functional requirements and modular software design and subsuming many previous specification languages. Development of prototyping and verification tools for Casl leads to them being interoperable, i.e. capable of being used in combination rather than in isolation.

CoFI began as an unfunded collaboration, but modest funding was obtained from the European Commission under the Esprit programme during the period October 1998 – April 2001 to support CoFI meetings and CoFI-related travel within Europe (project 29432, CoFI WG). Before this point, most effort in CoFI had concentrated on the design of Casl, which was complete in almost all respects in 1998. Building on this progress, the goals of the CoFI Working Group were: to coordinate the completion of and disseminate the Common Framework; to demonstrate its practical applicability in industrial contexts; and to establish the infrastructure needed for future European collaborative research in algebraic techniques. Activity is organized under six Task Groups, as follows:

- Language Design (coordinator: Bernd Krieg-Brückner until April 2001, since then Peter Mosses)
- Semantics (coordinator: Andrzej Tarlecki)
- Methodology (coordinator: Michel Bidoit)
- Tools (coordinator: Hélène Kirchner until April 2001, since then Bernd Krieg-Brückner and Till Mossakowski)
- Reactive Systems (coordinator: Egidio Astesiano, joined by Heinrich Hussmann from October 1999)
- External Relations (coordinator: Peter Mosses)

Overall coordination of CoFI was by Peter Mosses until August 1998, since then by Don Sannella.

This paper is a summary of progress on CoFI since 1998, with pointers to publications and other material produced by CoFI members during this period. Previous summaries of CoFI are [66, 71, 86]. Presentations of Casl are [2, 17, 31, 68]. The Casl tutorial [17] is especially recommended for newcomers, with more details, rationale and examples available from [2].

2 Language Design

In October 1998, the Language Design of CASL version 1.0 was published [29]. Subsequent work on polishing the design concerned syntactic extensions for literals, syntactic annotations for parsing and precedence of (mixfix) operator and predicate symbols, semantic annotations for proof obligations arising from various kinds of conservative extensions [81], details of concrete syntax affecting the form and position of comments and annotations, and details concerning the semantics of architectural specifications. A modification to version 1.0 incorporating many of these minor adjustments was released in July 1999 [30] and further adjustments were incorporated in CASL version 1.0.1 [31], released in March 2001. No further revisions are planned. The design of CASL v1.0.1 has been formally approved by IFIP WG1.3, see below under External Relations. The Semantics, Tools, and Methodology Task Groups, as well as the IFIP WG1.3 reviewers, who also made detailed comments on a previous version of CASL, all provided essential feedback regarding language design proposals. The rationale for aspects of the language design has been presented in [2, 18, 24, 25].

Another major effort has been the development of a library of Basic Datatypes for CASL which has also produced further methodological insight. Apart from a specification of natural, integer and rational numbers and their standard algebraic properties, the usual types such as sets, collections and lists have been defined. Moreover, a first attempt has been made at a first-order specification of real numbers, which appears to be the first of its kind. These types have also been related to approximate numeric types as used in computers. The library of basic datatypes has been revised several times following feedback from the members of the Language Design and Methodology Task Groups. The final result is documented in [82, 84].

Various sublanguages of CASL— total, many-sorted, equational — have been defined, often corresponding closely to embeddings of the specification languages of other frameworks into CASL [55, 56, 59, 63]. The relation of CASL to other specification languages such as ACT ONE, ASF, HEP-theories, LSL, OBJ3 and (functional) CafeOBJ has been clarified [63, 67, 69]. The logic underlying CASL has been translated to first-order logic (or second-order logic, when sort generation constraints are considered). This allows the re-use of first-order and higher-order theorem provers for CASL [63].

A procedure for the stepwise approval of sublanguages and language extensions has been established. The proposals for sublanguages and a higher-order extension [41] have received preliminary approval such that the definition of a proper semantics is now the next step. Work on parametrised and polymorphic (higher-order) types is ongoing [90] and there is a proposal for an extension of CASL with a mechanism for late binding [1].

Work on object-oriented and reactive system extensions to CASL has been carried out by the Reactive Systems Task Group, see below.

3 Semantics

The activities of the Semantics Task Group have centered around the development of a complete, formal mathematical semantics for CASL [32]. This development was accompanied by some work on the borderline between semantic and design issues for CASL, devoted to the careful study of some concepts introduced in the formalism: partiality has been presented in detail in [25], architectural specifications in [18], and the details of the category of signatures in [57, 93]. The early version of the semantics allowed us to identify some problems with the language design, which led to CASL version 1.0.1, as described above. A serious revision of the entire semantics is about to be completed [11]. Perhaps the most visible changes have involved the semantics of CASL architectural specifications, as presented in [49, 91, 92], due to problems pointed out for instance in [42, 48].

One key issue in the development of the final version of the semantics was to make it fully institution-independent, as studied in [61] for structured specifications and extended to architectural specifications in [92]. See [65] for a study of the impact of institution independence on development of tools for CASL.

With the semantics essentially ready, work has started on proof systems and proof support for CASL. This includes a study of foundations for translation of CASL into some well-supported logical frameworks [20] and verification of architectural specifications [43]; some key aspects of the logical system underlying CASL have been thoroughly investigated as well [19] cf. [36, 37]. An important stream of work addressed the issues of a careful, semantics-based comparison of CASL with other specification formalisms [8, 20, 59, 63].

We have also discussed some possible extensions of CASL, notably by higher-order operations, where a complete formal proposal is under development on the basis of [41], and behavioural equivalence of higher-order models (represented by pre-logical and lax logical relations) was investigated [44, 45, 72]. Further work has taken place on the borderline between semantics and methodology [10].

4 Methodology

The aim of the Methodology Task Group is to enrich the formalism designed under the Common Framework Initiative with ideas on the methodology of system specification and development that we would like to support, encourage and propagate. It is expected that these methodological views will influence the use of the CASL language and help its dissemination, and they have already influenced the overall design of the CASL language.

The activities of the Methodology Task Group have been organized around the following issues:

User manual and tutorials. A CASL User Manual is currently under preparation. It is intended to reuse the corresponding material for the preparation of a CASL Electronic Tutorial that will be made available on the Web. The

Casl User Manual will provide guidelines for writing specifications of individual system modules and for using the various features of the Casl language. Preliminary draft versions have been used for Casl tutorials in Berlin during ETAPS 2000 and in Genova during ETAPS 2001 [17]. Another, independent, Casl tutorial has been presented at a NASA formal methods workshop in June 2000 [79]. Related to this are studies on how to write consistent specifications [60, 83], on the proper use of Casl features [26], and on the addition of annotations to Casl specifications [81]. Further studies discuss the use of Casl during the software development specification and design phases [14, 27].

Case studies. A Web site with case studies illustrating the use and the benefits of Casl in various projects has been set up [9] to which CoFI participants have contributed. Extensive work has been done on basic datatypes for Casl in collaboration with the Language Design Task Group, see above for details. In addition, there have been studies on the use of Casl for computational and geometric modeling [40, 50–53] and multimedia presentations [13]. Casl has also been proposed for use as a general meta-notation in semantic descriptions [70].

Formal software development based on algebraic specifications. Architectural specifications, one of the most novel concepts of Casl, have been further studied [18]. Several studies have investigated refinement in various frameworks [44, 45, 87]. Further work related to Casl and the software development process are [7, 15, 38, 39].

5 Tools

Casl offers a flexible syntax including *mixfix* notation, which requires advanced parsing technology. Several Casl tools described in [22] have been built using the algebraic specification formalism Asf+Sdf and the Asf+Sdf Meta-Environment. Casl supports user-defined syntax which is non-trivial to process, and Asf+Sdf offers a powerful parsing technology called Generalized LR. Its interactive development environment facilitates rapid prototyping complemented by early detection and correction of errors. A number of core technologies developed for the Asf+Sdf Meta-Environment have been reused in the context of Casl.

Interoperability of Casl and existing tools is a major goal of the Tools Task Group. The first step has been to propose an exchange format that can be accepted as input and produced as output by every tool. The starting idea was to adopt basically abstract syntax trees with annotations providing specific information to communicate with various tools (parsers, rewrite engines, proof tools, etc.). An instantiation of a generic format developed for the representation of Asf+Sdf specifications and terms provides a Casl-specific exchange format. In [21], the abstract data type of Annotated Terms (*ATerms*) is defined and their design, implementation and application are discussed. A comprehensive procedural interface enables creation and manipulation of ATerms in C or Java. The ATerm implementation is based on maximal subterm sharing and automatic garbage collection. A binary exchange format for the concise representation of

ATerms (with sharing preserved) allows the fast exchange of ATerms between applications. Work is also in progress to provide XML as an external exchange format, with translations back and forth between XML and ATerms. Based on these low-level formats, high-level formats such as CasFix [22] (for abstract syntax trees of Casl specifications), CasEnv (for global environments containing signature information etc.) and FCasEnv (a flattened version of CasEnv, for use with tools that do not support structured specifications) have been developed. Formats for storing proofs and developments will follow.

One main achievement has been the integration of several tools in the Casl Tool Set CATS [62]. This combines a parser, a static checker, a LaTeX pretty printer, facilities for printing signatures of specifications and structure graphs of Casl specifications, with links to various verification and development systems. To experiment with Casl specifications, the CATS system provides different user interfaces: a Web-based interface, and a compact stand-alone version (with both a command-line and a window interface). A repository with successfully and unsuccessfully parsed specifications is under development.

Existing rewrite engines provide a good basis for prototyping (parts of) Casl specifications. The problem of executing some Casl specifications using the rewrite engine provided by the Elan system, which implements rewriting in a very efficient way, is addressed in [47]. The class of Casl specifications that can be executed are those having equational axioms (including possibly associative-commutative operators), that are oriented as conditional rewrite rules. The equality predicate is used to express the congruence on terms whilst the equivalence connective allows defining the congruence on expressions built over predicates. Subsorting and partiality features are not considered for now, but basic and structured Casl specifications are supported. The mapping from Casl to Elan is performed by translating the Casl abstract syntax into the abstract syntax developed for Elan. The current implementation needs the Casl Tool Set to parse a Casl specification and to generate the “flattened” FCasEnv format. By using the translation tool and then the Elan compiler, an executable program is produced which computes normal forms with respect to a given Casl specification.

The standalone version of CATS also contains an encoding into several other logics. The encoding transforms a Casl specification into second-order logic step by step. First, partiality is encoded via error elements inhabiting a supersort; second, subsorting is encoded via injections; and third, sort generation constraints are expressed via second-order induction axioms. It is possible to stop after the first or second step if one wants to use a tool supporting subsorting or sort generation constraints directly. For details, see [63], where alternative encodings are also described. In this way, CATS allows Casl to interface with a large number of first- and higher-order theorem provers.

The HOL-Casl system, being built on top of CATS, uses the encoding of Casl into second-order logic to connect Casl to the Isabelle theorem prover and the generic graphical user interface IsaWin. This approach to encoding Casl in proof systems such as Isabelle allows verification and program transformation [63].

Various verification tools have already been developed for algebraic specifications, and can be reused for specific subsets of Casl: equational, conditional, full first-order logic with total functions, total functions with subsorts, partial functions, etc. The INKA system provides an integrated specification and theorem proving environment for a sub-language of Casl that excludes partial functions (with the encoding provided by CATS, it will also be useable with full Casl). CATS has been connected to the development graph management component of the INKA theorem proving system. Structured Casl specifications in the CasEnv format are translated to development graphs [7, 64]. The development graph supports the management of theories and proof obligations that arise from Casl specifications in a theorem prover-independent way. Moreover, it provides an efficient way of managing change, allowing re-use of those parts of proofs that are not affected by the change of a specification.

All tools developed in the Tools Task Group are made available to the community, after validation by the group. A Web page for tools describing on-going work, giving access to available tools, and giving guidelines on how to propose a new tool, is available at `http://www.tzi.de/cofi/Tools`.

6 Reactive Systems

The aim of the Reactive Systems Task Group is to develop an extension of the Common Framework to deal with reactive, concurrent and parallel systems; object-oriented techniques for dealing with reactiveness have also been considered. The specification framework deals with all phases from requirement to design, including the intermediate steps.

From the beginning, the goal has been to have an extension which is: based on state-of-the art techniques; compatible and integrated with the Casl language proposal; mathematically rigorous; able to deal with a wide range of significant systems; sufficiently friendly for practical use by a wide community; and guided and complemented by considerations concerning methodology and tools.

The technical work of the Task Group has proceeded in two tracks:

Track 1: Autonomous Extensions of CASL towards Reactive Systems. These extensions are individual contributions aligned to the Common Framework. The following proposals have been developed within the Task Group or in close connection to it.

CASL-Charts: A formalism integrating Casl and state charts has been developed for giving discrete models of the functional and dynamic behaviour of reactive systems [77, 78].

State-based CASL extension: An extension of Casl to deal with internal states has been developed [12].

Integration of CASL with process specification languages: Proposals have been made to bring Casl together with most of the known approaches to formal specification of communicating processes: CASL/CCS [85]; CASL/CSP; CASL/Petri Nets.

CASL-LTL/Design: A method for the design specification of concurrent systems has been developed, based on a formalism of (structured) conditional specifications defining processes in terms of labelled transition systems [73].

CoFI-LTL/Requirements: This is essentially a many-sorted first-order temporal logic [3], cf. [34].

JTN (Java Targeted Notation): A visual formal notation was developed for the specification of reactive, parallel, concurrent systems. It is essentially a restricted subcase of CASL-LTL/Design, which is automatically translatable into Java [33].

Track 2: Coordinated Effort. In a joint effort of the members of the Task Group, work centered around the OMG standard language UML, the Unified Modelling Language. The motivation for this decision was that UML is an industry standards for specification and design of complex systems, including reactive systems, so that a link to UML will definitely enhance the practical accessibility of CASL, and also that UML already contains a number of graphical formalisms dealing with concurrency and reactiveness. It was observed that the UML standard has reached a level of maturity which is close to formal methods, but is missing a solid semantic foundation [46, 94].

The basic idea of the joint work is to adopt CASL as a language for annotating UML static and dynamic diagrams, enhancing and possibly replacing the Object Constraint Language of UML. In order to achieve an integration between CASL and UML, a systematic translation between UML diagrams and CASL specifications was defined and documented. Class diagrams, representing static aspects of the system, are translated into standard CASL. Statechart diagrams are dealt with by a translation into CASL-LTL [3, 34, 74]. First attempts were also made to translate sequence diagrams into CASL-LTL. Finally, a multiview approach was worked out for the semantics of UML, integrating static and dynamic aspects. The core idea of this approach is to use CASL as a metalanguage which helps to express the semantics of various diagrams in a uniform and mathematically rigorous language [6, 75]. This way, an important contribution also to the better formal underpinning of the semantics of UML diagrams was produced [76], cf. [4].

7 External Relations

The main tasks concerning external relations were establishing the relationship between CASL and previous frameworks, developing a tutorial online presentation of CASL, and liaison with IFIP Working Group 1.3.

Relationship between CASL and previous frameworks. At the level of specification-in-the-small, ACT ONE, ASF+SDF, (functional) CafeOBJ, LSL and OBJ3 [59, 63, 69] have been examined. The outcome was that apart from the rather pathological case of algebras with empty carriers, all of these languages can be translated to sublanguages of CASL. Moreover, the translation is always straightforward, with the exception of OBJ3's retracts (their translation to CASL had previously been studied [59]). For some languages, the corresponding sublanguage of CASL is indicated in [55, 63]. At the level of specification-in-the-large, a first informal look had previously been taken at LSL, ACT TWO, ASF+SDF, and OBJ3 [58]. Although some structuring constructs of these languages cannot be translated literally to CASL, it is always possible to find a circumscription in CASL. CoFI WG has not had sufficient resources for defining embeddings of other existing languages into CASL. It is hoped that a forthcoming informal sketch of the relationship between basic specifications in CASL and various other languages may stimulate further detailed investigations, leading to the provision of translators from other languages into CASL.

Developing a tutorial online presentation of CASL. It was decided to combine the CASL Tutorial and the CASL User's Manual into a single publication presented via different media as described above under Methodology. The CASL Tutorial will be based on the same explanations and examples as given in the User's Manual, supplemented by exercises (with interactivity provided by means of some of the available CASL tools). The examples and main points from the Tutorial/User's Manual have already been used in presentations of CASL at ETAPS in 2000 and 2001 [17].

Liaison with IFIP WG1.3. Following a review of a previous version of CASL, IFIP WG1.3 was asked to review the final design of CASL version 1.0.1 in May 2000, which was presented to the IFIP WG1.3 meeting at Stanford in June 2000. The report of the IFIP referees [35] was made available to CoFI and presented at the IFIP WG1.3 meeting in Genova in March 2001. On the basis of the report of the reviewers and the subsequent response of the CASL designers, IFIP WG1.3 decided to formally approve the design of CASL version 1.0.1. The IFIP WG1.3 reviewers made some useful recommendations concerning the enhancement of the documents describing CASL, especially concerning the need for a rough indication of the relationship between CASL and existing languages at the level of basic specifications.

8 Dissemination and take-up

One of the principal goals of CoFI WG has been wide dissemination and awareness of the Common Framework. The main target audience at this stage has been fellow researchers in the use of Formal Methods, with some awareness among relevant industrial groups as a secondary goal. Activities include:

Publication of the scientific results in a variety of forms: in conferences and journals, via the CoFI web pages [28], and on CD ROMs distributed with the FM'99 and LFM 2000 proceedings, with the former including some prototype CASL tools and the latter including an annotated list of CoFI-related URLs [80]. Publication of a book on CASL is planned.

Tutorials and presentations on CASL and COFI at a variety of venues and to a range of audiences. The tutorial material [17, 79] has been used in teaching at a number of universities.

Workshops organized jointly with other relevant workshops (WADT in 1999 and 2001) and/or as satellites of major conferences (FM in 1999, ETAPS in 1999, 2000 and 2001). This boosts the visibility of the workshops, providing opportunities for fellow researchers to learn about (and perhaps join) COFI.

It is rather early to expect significant industrial take-up. But a concrete start in this direction is represented by recent initiatives at Zühlke Engineering AG in Zürich where LSL has been used in the past. An internal study comparing LSL and CASL led to the decision to switch from LSL to CASL for future projects, and industrial training material on formal methods based on CASL is in preparation.

CASL will play a central role in the German multi-site "MultiMedia Instruction in Safe and Secure Systems" project, funded by the BMBF, which will develop a range of educational materials concerning formal specifications, covering an M.Sc. curriculum in Safe and Secure Systems. Logic and algebraic specification (as embodied in CASL) will be the major foundation and CASL will be the specification language for all issues involving data.

It is worth noting that the connection with COFI and CASL is highlighted by Springer in its marketing material for [5], suggesting that some degree of "market penetration" has already been achieved. Several presentations at WADT/COFI 2001 reported projects where CASL had been adopted for practical use. Together with some presentations concerning CASL itself, this gave the impression that CASL is already recognized as a well-established *de facto* standard language for algebraic specification.

9 Invitation

COFI has accomplished a great deal since its inception, but more remains to be done. It is an open collaboration, and new participants are welcome to join at any time. Anybody who wishes to contribute is warmly invited to visit the COFI web site at `http://www.brics.dk/Projects/CoFI/` where all COFI documents are freely available. Announcements of general interest to COFI participants are broadcast on the low-volume mailing list `cofi-list@brics.dk` and each Task Group has its own mailing list; see the COFI web site for subscription instructions. All of these mailing lists are moderated.

Acknowledgements Thanks to the participants of COFI for all their unselfish contributions; without them there would be no COFI activity to describe. Special thanks to the coordinators of the various COFI Task Groups, listed in the introduction. Extra special thanks to Peter Mosses, who got COFI off the ground on behalf of IFIP WG1.3 and COMPASS WG and acted as overall coordinator until mid-1998. The work described here was supported by the ESPRIT-funded COFI Working Group (project 29432, COFI WG).

References

1. D. Ancona, M. Cerioli and E. Zucca. Extending CASL by late binding. *Recent Trends in Algebraic Development Techniques: Selected Papers from WADT'99*, Bonas. Springer LNCS 1827, 53–72 (2000).
2. E. Astesiano, M. Bidoit, H. Kirchner, B. Krieg-Brückner, P. Mosses, D. Sannella and A. Tarlecki. CASL: The common algebraic specification language. *Theoretical Computer Science* (2002), to appear.
3. E. Astesiano, M. Broy and G. Reggio. Algebraic specification of concurrent systems. In E. Astesiano, H.-J. Kreowski and B. Krieg-Brückner (eds.), *Algebraic Foundations of Systems Specifications*, 467–520. Springer (1999).
4. E. Astesiano, M. Cerioli and G. Reggio. Plugging data constructs into paradigm-specific languages: towards an application to UML. *Proceedings of AMAST 2000*, Iowa City. Springer LNCS 1816, 273–292 (2000).
5. E. Astesiano, H.-J. Kreowski and B. Krieg-Brückner (eds.). *Algebraic Foundations of Systems Specification.* Springer (1999).
6. E. Astesiano and G. Reggio. UML as heterogeneous multiview notation: Strategies for a formal foundation. *Proc. of OOPSLA'98 Workshop "Formalizing UML. Why? How?"* (1998).
7. S. Autexier, D. Hutter, H. Mantel and A. Schairer. Towards an evolutionary formal software-development using CASL. *Recent Trends in Algebraic Development Techniques: Selected Papers from WADT'99*, Bonas. Springer LNCS 1827, 73–88 (2000).
8. H. Baumeister. Relating abstract datatypes and Z-schemata. *Recent Trends in Algebraic Development Techniques: Selected Papers from WADT'99*, Bonas. Springer LNCS 1827, 366–382 (2000).
9. H. Baumeister. CASL case studies. `http://www.informatik.uni-muenchen.de/~baumeist/CoFI/case.html` (2001).
10. H. Baumeister and D. Bert. Algebraic specification in CASL. *Software Specification Methods: An Overview Using a Case Study*, FACIT: Formal Approaches to Computing and Information Technology. Springer (2000).
11. H. Baumeister, M. Cerioli, A. Haxthausen, T. Mossakowski, P. Mosses, D. Sannella and A. Tarlecki. CASL: The common algebraic specification language. semantics. Version 1.0, to be completed (2001).
12. H. Baumeister and A. Zamulin. State-based extension of CASL. *Proc. IFM 2000* (2000).
13. D. Bert and S. Lo Presti. Algebraic specification of operator-based multimedia scenarios. *Recent Trends in Algebraic Development Techniques: Selected Papers from WADT'99*, Bonas. Springer LNCS 1827, 382–399 (2000).
14. M. Bidoit. From requirements to software in CASL. Invited talk given at WADT'99, Bonas (1999).
15. M. Bidoit, R. Hennicker, F. Tort and M. Wirsing. Correct realizations of interface constraints with OCL. *Proc. UML'99*. Springer LNCS 1723 (1999).
16. M. Bidoit, H.-J. Kreowski, P. Lescanne, F. Orejas and D. Sannella (eds.). *Algebraic System Specification and Development: A Survey and Annotated Bibliography.* Springer LNCS 501 (1991).
17. M. Bidoit and P. Mosses. A gentle introduction to CASL. Tutorial, WADT/CoFI Workshop at the 4th European Joint Conferences on Theory and Practice of Software (ETAPS 2001), Genova (2001). Available from `http://www.lsv.ens-cachan.fr/~bidoit/CASL/`.

18. M. Bidoit, D. Sannella and A. Tarlecki. Architectural specifications in CASL. *Formal Aspects of Computing* (2002), to appear. A preliminary version appeared under the same title in *Proc. 7th Intl. Conference on Algebraic Methodology and Software Technology, AMAST '98*, Manaus. Springer LNCS 1548, 341–357 (1998).
19. T. Borzyszkowski. Generalized interpolation in CASL. *Information Processing Letters* 76:19–24 (2000).
20. T. Borzyszkowski. Higher-order logic and theorem proving for structured specifications. *Recent Trends in Algebraic Development Techniques: Selected Papers from WADT'99*, Bonas. Springer LNCS 1827, 401–418 (2000).
21. M. van den Brand, H. de Jong, P. Klint and P. Olivier. Efficient annotated terms. *Software, Practice & Experience* 30:259–291 (2000).
22. M. van den Brand and J. Scheerder. Development of parsing tools for CASL using generic language technology. *Recent Trends in Algebraic Development Techniques: Selected Papers from WADT'99*, Bonas. Springer LNCS 1827, 89–105 (2000).
23. M. Cerioli, M. Gogolla, H. Kirchner, B. Krieg-Brückner, Z. Qian and M. Wolf (eds.). *Algebraic System Specification and Development: Survey and Annotated Bibliography.* 2nd edition. Monographs of the Bremen Institute of Safe Systems 3. Shaker (1998).
24. M. Cerioli, A. Haxthausen, B. Krieg-Brückner and T. Mossakowski. Permissive subsorted partial logic in CASL. *Proceedings of AMAST'97*, Sydney. Springer LNCS 1349, 91–107 (1997).
25. M. Cerioli, T. Mossakowski and H. Reichel. From total equational to partial first order logic. In E. Astesiano, H.-J. Kreowski and B. Krieg-Brückner (eds.), *Algebraic Foundations of Systems Specifications*, 31–104. Springer (1999).
26. M. Cerioli and G. Reggio. Basic CASL at work: a compass for the labyrinth of partiality, subsorting and predicates. Presented at WADT'99, Bonas (1999).
27. C. Choppy and G. Reggio. Using CASL to specify the requirements and the design: A problem specific approach. *Recent Trends in Algebraic Development Techniques: Selected Papers from WADT'99*, Bonas. Springer LNCS 1827, 104–123 (2000).
28. CoFI. The Common Framework Initiative for algebraic specification and development, electronic archives. Notes and Documents accessible from `http://www.brics.dk/Projects/CoFI/`.
29. CoFI Language Design Task Group. CASL – The CoFI Algebraic Specification Language – Summary, version 1.0. `Documents/CASL/Summary-v1.0`, in [28] (1998).
30. CoFI Language Design Task Group. CASL – The CoFI Algebraic Specification Language – Summary, version 1.0 (revised). `Documents/CASL/v1.0/Summary`, in [28] (1999).
31. CoFI Language Design Task Group. CASL – The CoFI Algebraic Specification Language – Summary, version 1.0.1. `Documents/CASL/v1.0.1/Summary`, in [28] (2001).
32. CoFI Semantics Task Group. CASL – The CoFI Algebraic Specification Language – Semantics. `Documents/CASL/Semantics` (version 0.96), in [28] (1999).
33. E. Coscia and G. Reggio. JTN: A Java-targeted graphic formal notation for reactive and concurrent systems. *Fundamental Approaches to Software Engineering (FASE'99), European Joint Conferences on Theory and Practice of Software*, Amsterdam. Springer LNCS 1577, 77–97 (1999).
34. G. Costa and G. Reggio. Specification of abstract dynamic datatypes: A temporal logic approach. *Theoretical Computer Science* 173(2):513–554 (1997).
35. H. Ehrig, J. Meseguer, U. Montanari, F. Orejas, F. Parisi-Presicce and M. Wirsing. Recommendations on the revised design of CASL. Internal IFIP WG1.3 document (2001).

36. R. Gumb. Model sets in a nonconstructive logic of partial terms with definite descriptions. *Automated Reasoning with Analytic Tableaux and Related Methods, TABLEAUX 2000*. Springer LNAI 1847, 268–278 (2000).
37. R. Gumb. An extended joint consistency theorem for a nonconstructive logic of partial terms with definite descriptions. *Studia Logica*. To appear.
38. M. Haveraaen. Domain specific languages and software architectures – a challenge for Casl. Presented at CoFI meeting, Amsterdam (1999).
39. M. Haveraaen. A 2-tiered software process model for utilizing Casl. Technical Report 208, Department of Informatics, University of Bergen (2000).
40. M. Haveraaen, H.A. Friis and T.A. Johansen. Formal software engineering for computational modeling. *Nordic Journal of Computing* 6:241–270 (1999).
41. A. Haxthausen, B. Krieg-Brückner and T. Mossakowski. Subsorted partial higher-order logic as an extension of Casl. CoFI Note L-10, in [28] (1998).
42. P. Hoffman. Semantics of architectural specifications (in Polish). Master's thesis, Warsaw University (2000).
43. P. Hoffman. Verifying architectural specifications. *Recent Trends in Algebraic Development Techniques: Selected Papers from WADT 2001*, Genova. Springer LNCS, to appear (this volume).
44. F. Honsell, J. Longley, D. Sannella and A. Tarlecki. Constructive data refinement in typed lambda calculus. *Proc. 3rd Intl. Conf. on Foundations of Software Science and Computation Structures (FOSSACS 2000), European Joint Conferences on Theory and Practice of Software*, Berlin. Springer LNCS 1784, 149–164 (2000).
45. F. Honsell and D. Sannella. Pre-logical relations. *Information and Computation* (2002), to appear. A preliminary version appeared under the same title in *Proc. Computer Science Logic, CSL'99*, Madrid. Springer LNCS 1683, 546–561 (1999).
46. H. Hussmann, M. Cerioli, G. Reggio and F. Tort. Abstract data types and UML models. Technical Report DISI-TR-99-15, Università di Genova (1999).
47. H. Kirchner and C. Ringeissen. Executing Casl equational specifications with the Elan rewrite engine. CoFI Note T-9 (revised version), in [28] (2000).
48. B. Klin. An implementation of static semantics for architectural specifications in Casl (in Polish). Master's thesis, Warsaw University (2000).
49. B. Klin, P. Hoffman, A. Tarlecki, L. Schröder and T. Mossakowski. Checking amalgamability conditions for Casl architectural specifications. *Proc. Intl. Symp. on Mathematical Foundations of Computer Science, MFCS 2001*. Springer LNCS 2136, 451–463 (2001).
50. F. Ledoux, A. Arnould, P. Le Gall and Y. Bertrand. A high-level operation in 3D modeling: a Casl case study. Report LaMI 52, Université d'Evry-Val d'Essonne, Evry (2000).
51. F. Ledoux, A. Arnould, P. Le Gall and Y. Bertrand. Geometric modeling with Casl. *Recent Trends in Algebraic Development Techniques: Selected Papers from WADT 2001*, Genova. Springer LNCS, to appear (this volume).
52. F. Ledoux, J.-M. Mota, A. Arnould, C. Dubois, P. Le Gall and Y. Bertrand. Formal specification for a mathematics-based application domain: geometric modeling. Report LaMI 51, Université d'Evry-Val d'Essonne, Evry (2000).
53. F. Ledoux, J.-M. Mota, A. Arnould, C. Dubois, P. Le Gall and Y. Bertrand. Spécifications formelles du chanfreinage. In *Approches Formelles dans l'Assistance au Développement de Logiciels (AFADL), Nancy* (2001).
54. J. Loeckx, H.-D. Ehrich and M. Wolf. *Specification of Abstract Data Types*. Wiley (1996).
55. T. Mossakowski. Sublanguages of Casl. CoFI Note L-7, in [28] (1997).

56. T. Mossakowski. Two "functional programming" sublanguages of CASL. CoFI Note L-9, in [28] (1998).
57. T. Mossakowski. Colimits of order-sorted specifications. *Recent Trends in Algebraic Development Techniques: Selected Papers from WADT'97*, Tarquinia. Springer LNCS 1376, 316–332 (1998).
58. T. Mossakowski. Translating other specification languages into CASL. Presented at WADT'98, Lisbon (1998). Handwritten notes are available from the author.
59. T. Mossakowski. Translating OBJ3 to CASL: The institution level. *Recent Trends in Algebraic Development Techniques: Selected Papers from WADT'98*, Lisbon. Springer LNCS 1589, 198–214 (1999).
60. T. Mossakowski. How to write consistent CASL design specifications. CoFI Note M-8, in [28] (2000).
61. T. Mossakowski. Specification in an arbitrary institution with symbols. *Recent Trends in Algebraic Development Techniques: Selected Papers from WADT'99*, Bonas. Springer LNCS 1827, 252–270 (2000).
62. T. Mossakowski. CASL: From semantics to tools. *Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2000), European Joint Conferences on Theory and Practice of Software*, Berlin. Springer LNCS 1785, 93–108 (2000).
63. T. Mossakowski. Relating CASL with other specification languages: the institution level. *Theoretical Computer Science* (2002), to appear.
64. T. Mossakowski, S. Autexier and D. Hutter. Extending development graphs with hiding. *Fundamental Approaches to Software Engineering (FASE 2001), European Joint Conferences on Theory and Practice of Software*, Genova. Springer LNCS 2029, 269–283 (2001).
65. T. Mossakowski and B. Klin. Institution-independent static analysis for CASL. *Recent Trends in Algebraic Development Techniques: Selected Papers from WADT 2001*, Genova. Springer LNCS, to appear (this volume).
66. P. Mosses. CoFI: the common framework initiative for algebraic specification and development. *Proc. 7th Intl. Joint Conf. on Theory and Practice of Software Development*, Lille. Springer LNCS 1214, 115–137 (1997).
67. P. Mosses. CASL for ASF+SDF users. *ASF+SDF '97, Proc. 2nd Intl. Workshop on the Theory and Practice of Algebraic Specifications*, volume ASFSDF-97 of *Electronic Workshops in Computing*. British Computer Society, `http://www.ewic.org.uk/ewic/workshop/list.cfm` (1997).
68. P. Mosses. CASL: a guided tour of its design. *Recent Trends in Algebraic Development Techniques: Selected Papers from WADT'98*, Lisbon. Springer LNCS 1589, 216–240 (1999).
69. P. Mosses. CASL for CafeOBJ users. Chapter 6 of K. Futatsugi, A. T. Nakagawa and T. Tamai (eds.), *CAFE: An Industrial-Strength Algebraic Formal Method*, 121–144. Elsevier (2000).
70. P. Mosses. CASL and Action Semantics. In *AS 2000*, number NS-00-6 in Notes Series, 62–78, BRICS, Univ. of Aarhus (2000).
71. P. Mosses. CoFI: The Common Framework Initiative for algebraic specification and development. In G. Păun, G. Rozenberg and A. Salomaa (eds.), *Current Trends in Theoretical Computer Science: Entering the 21st Century*, 153–163. World Scientific (2001). An earlier version appeared in *Bull. EATCS* 59:127–132 (1996).
72. G. Plotkin, J. Power, D. Sannella and R. Tennent. Lax logical relations. *Proc. 27th Intl. Colloq. on Automata, Languages and Programming*, Geneva. Springer 1853, 85–102 (2000).

73. G. Reggio, E. Astesiano and C. Choppy. CASL-LTL: A CASL extension for dynamic reactive systems – summary. Technical Report DISI-TR-99-34, Università di Genova (1999).
74. G. Reggio, E. Astesiano, C. Choppy and H. Hussmann. Analysing UML active classes and associated state machines – a lightweight approach. *Fundamental Approaches to Software Engineering (FASE 2000), European Joint Conferences on Theory and Practice of Software*, Berlin. Springer LNCS 1783, 127–146 (2000).
75. G. Reggio, M. Cerioli and E. Astesiano. An algebraic semantics of UML supporting its multiview approach. *Proc. 2nd AMAST Workshop on Algebraic Methods in Language Processing (AMILP 2000)*, number 16 in Twente Workshop on Language Processing, Enschede (2000).
76. G. Reggio, M. Cerioli and E. Astesiano. Towards a rigorous semantics of UML supporting its multiview approach. *Fundamental Approaches to Software Engineering (FASE 2001), European Joint Conferences on Theory and Practice of Software*, Genova. Springer LNCS 2029, 171–186 (2001).
77. G. Reggio and L. Repetto. CASL-CHART: Syntax and semantics. Technical Report DISI-TR-00-1, Università di Genova (2000).
78. G. Reggio and L. Repetto. CASL-CHART: A combination of statecharts and of the algebraic specification language CASL. *Proceedings of AMAST 2000*, Iowa City. Springer LNCS 1816, 243–257 (2000).
79. M. Roggenbach. CASL tutorial at LFM 2000 (2000). Slides available at `http://www.tzi.de/~roba/tutorial.ps`.
80. M. Roggenbach. An annotated list of URLs related with CoFI and CASL. On a CD ROM distributed to participants of the Fifth NASA Langley Formal Methods Workshop (2000). Also available from `http://www.informatik.uni-bremen.de/~roba/URL.txt`.
81. M. Roggenbach and T. Mossakowski. Proposal of some annotations and literal syntax in CASL. CoFI Note L-11, in [28] (1999).
82. M. Roggenbach and T. Mossakowski. Basic datatypes in CASL. CoFI Note L-12, version 0.4.1, in [28] (2000).
83. M. Roggenbach and L. Schröder. Towards trustworthy specifications I: Consistency checks. *Recent Trends in Algebraic Development Techniques: Selected Papers from WADT 2001*, Genova. Springer LNCS, to appear (this volume).
84. M. Roggenbach, L. Schröder and T. Mossakowski. Specifying real numbers in CASL. *Recent Trends in Algebraic Development Techniques: Selected Papers from WADT'99*, Bonas. Springer LNCS 1827, 146–161 (2000).
85. G. Salaün, M. Allemand and C. Attiogbé. Formal combination of the CCS process algebra with the CASL algebraic specification language. Presented at WADT/CoFI 2001, Genova (2001).
86. D. Sannella. The Common Framework Initiative for algebraic specification and development of software. *Proc. 3rd Intl. Conf. on Perspectives of System Informatics (PSI'99)*, Novosibirsk. Springer LNCS 1755, 1–9 (2000).
87. D. Sannella. Algebraic specification and program development by stepwise refinement. *Proc. 9th Intl. Workshop on Logic-based Program Synthesis and Transformation, LOPSTR'99*, Venice. Springer LNCS 1817, 1–9 (2000).
88. D. Sannella and A. Tarlecki. Essential concepts of algebraic specification and program development. *Formal Aspects of Computing* 9:229–269 (1997).
89. D. Sannella and A. Tarlecki. *Foundations of Algebraic Specifications and Formal Program Development*. Cambridge Univ. Press, to appear.
90. L. Schröder and T. Mossakowski. HasCASL: Towards integrated specification and development of Haskell programs. Submitted for publication (2001).

91. L. Schröder, T. Mossakowski and A. Tarlecki. Amalgamation in Casl via enriched signatures. *Proc. 28th Intl. Colloq. on Automata, Languages and Programming.* Springer 2076, 993–1004 (2001).
92. L. Schröder, T. Mossakowski, A. Tarlecki, P. Hoffman and B. Klin. Semantics of architectural specifications in Casl. *Fundamental Approaches to Software Engineering (FASE 2001), European Joint Conferences on Theory and Practice of Software*, Genova. Springer LNCS 2029, 253–268 (2001).
93. E. Wagner. On the category of Casl signatures. Presented at WADT'99, Bonas (1999).
94. R. Wieringa, E. Astesiano, G. Reggio, A. Le Guennec, H. Hussmann, K. van den Berg and P. van den Broek. Is it feasible to construct a semantics for all of UML? Dynamic behaviour and concurrency. *ECOOP Workshop Reader: UML Semantics FAQ* (1999).
95. M. Wirsing. Algebraic specification. *Handbook of Theoretical Computer Science* (J. van Leeuwen, ed.). North-Holland (1990).