Foundations of Software Specification and Development: An Abstract Overview[*]

Andrzej Tarlecki[†]

The long-term goal of work in the area of algebraic (and not only algebraic) specification is to provide a formal basis to support systematic development of software from specifications by means of verified refinements. A considerable body of technical work has been devoted to this important goal, with many interesting concepts and quite a number of non-trivial results emerging. I intend to present my overall view of the area, primarily trying to give a general picture of what has been achieved and what has to be done, and venturing into some intriguing technical aspects of the work from time to time only.

To begin with, the very notion of *specification* deserves some discussion: how specifications are built, understood, verified, modified, used... I will pay special attention here to the issues of building specifications in a *structured* way: realistic specifications are quite big and complex, and so structuring specifications and exploiting this structure in everything we use them for is the only reasonable hope to make formal specifications practically usable.

It turns out that many of the issues arising here are in essence independent of the details of the underlying concepts many specification formalisms are so attached to. One can abstract away from the concepts of a logical formula used to capture specific program properties, of a model used to present a more abstract, semantic view of programs, of a signature used to describe static interfaces of systems and their components. To make this precise, a formalisation of the notion of a logical system has been proposed under the name of *institution* — I will address the powerful conceptual possibilities such a formalisation opens in the course as well.

I will then rely on a sufficiently abstract and general concept of a specification to address the issues of *program development*. The overall idea here is that this should proceed by gradual refinement steps, incorporating relatively minor design and programming decisions one at a time. This allows, at least in principle, formal verification of correctness of such steps, and the whole framework is set up in such a way that this in turn ensures correctness of the resulting program.

Refinement steps that involve *decomposition* of a problem into independent programming tasks deserve special attention. I will discuss relationship between such decompositions

[*]Abstract for a proposed course at ECI 2007 Summer School

[†]Institute of Informatics, Warsaw University; and Institute of Computer Science, Polish Academy of Sciences; Warsaw, Poland

and specification structure, as well as how such decompositions, if carried out and verified rigorously, lead to well-structured programs with precisely specified, modular components.

One key issue that arises naturally when specific examples are taken into account is that requirements specifications need not to (and so, should not be) considered literally, but only in so far as they constrain the externally observable program behaviour. One way to capture this if by considering *observational* interpretation of specifications and by allowing refinement steps to be correct up to such observable interpretations only.

These ideas will be largely presented without referring to any specific formalisms — in examples, when needed, I will rely on CASL, the Common Algebraic Specification Language, developed recently within CoFI, the Common Framework Initiative.

Overall plan of lectures

- **Underlying logical framework:** basic algebraic framework; first-order logic; elements of category theory; institutions.
- **Specifications:** basic and structured specifications; proof systems for specifications.
- **Program development:** simple refinements; constructor refinements; local constructions in global refinement steps; architectural specifications.
- **Observational approach:** observational/behavioural interpretation of specifications, refinements, and architectural specifications; local correctness and stability.
- **Heterogeneous specifications:** maps between institutions; focused heterogeneous specifications; distributed heterogeneous specifications.
- **Conclusions**.

Course materials:

Preliminary course transparencies are available at `http://www.mimuw.edu.pl/~tarlecki/ECI/`.

Prerequisites and literature:

Most of the material covered by the course can be found in the forthcoming book:

- D. Sannella, A. Tarlecki. *Foundations of Algebraic Specifications and Formal Program Development.*

Access to at least the relevant chapters will be provided.

Although the necessary prerequisites will be recalled, a rudimentary knowledge of basic concepts of universal algebra and category theory may be helpful. Apart from a number of standard textbooks, these can be learnt from:

- D. Sannella, A. Tarlecki. Algebraic preliminaries. In: *Algebraic Foundations of System Specification*, E. Astesiano, H.-J. Kreowski, B. Krieg-Brückner, eds., 13–30, Springer Verlag 1999.

- D. Sannella, A. Tarlecki. Category theory. In: *Foundations of Algebraic Specifications and Formal Program Development*, forthcoming.

The lectures will loosely follow the ideas presented in the following papers, which therefore should make a good background reading:

- A. Tarlecki. Abstract specification theory: an overview. In: *Models, Algebras, and Logics of Engineering Software*, M. Broy, M. Pizka, eds., NATO Science Series - Computer and Systems Sciences, Vol. 191, 43–79, IOS Press, 2003.

- D. Sannella, A. Tarlecki. Essential concepts of algebraic specification and program development. *Formal Aspects of Computing* 9(1997) 229–269.

- M. Bidoit, D. Sannella, A. Tarlecki. Architectural specifications in CASL. *Formal Aspects of Computing*, 13 (2002), 252–273.

- M. Bidoit, D. Sannella, A. Tarlecki. Global development via local observational construction steps. *Proc. MFCS'02*, Springer LNCS 2420, 1–24, Springer-Verlag 2002.

Institutions have been introduced in:

- J. Goguen, R. Burstall. Institutions: abstract model theory for specification and programming. *Journal of the Assoc.for Computing Machinery* 39(1):95–146, 1992.

The following book chapter provides an introduction to, motivation for, and some concepts and results built around this notion:

- A. Tarlecki. Institutions: an abstract framework for formal specifications. In: *Algebraic Foundations of System Specification*, E. Astesiano, H.-J. Kreowski, B. Krieg-Brückner, eds., 105–130, Springer-Verlag 1999.

Most of the above papers may be found at `http://www.mimuw.edu.pl/~tarlecki/ECI/`.

CoFI pages, `http://www.cofi.info/`, provide all the material on Casl and relate work. The following are not to be missed:

- M. Bidoit, P.D. Mosses. Casl *User Manual.* Springer LNCS 2900 (IFIP Series), 2004. With chapters by T. Mossakowski, D. Sannella, and A. Tarlecki.

- CoFI (The Common Framework Initiative). Casl *Reference Manual.* Springer LNCS 2960 (IFIP Series), 2004.