Tutorial 1 : Abstraction and Refinement of Concurrent Programs and Formal Specification
A Practical View

Dominique Cansell[2,3]
cansell@loria.fr, Dominique Méry[2]
mery@loria.fr, and Christophe Tabacznyj[1,2]
tabaczny@loria.fr

[1] CEA
CENTRE D'ETUDES DE SACLAY
DEIN / SLA - Bt 528
F 91191 GIF-SUR-YVETTE CEDEX
FRANCE
[2] LORIA Université Henri Poincaré Nancy 1
BP 239
54506 Vandoeuvre-ls-Nancy Cedex
[3] Universit de Metz
Ile du Saulcy
57045 Metz
France

Formal methods allow to model systems and systems properties by providing accurate mathematical notations (type theory, set theory, ...). Implementations can be derived from a formal specification using methods based on refinement. Therefore, from a pragmatic industrial point of vue, the dual work based on abstraction is very important too for verifying safety critical systems, but also for addressing questions like maintenance, reverse ingineering of codes, modifications of programming language, code evolution, inspection of open codes to ensure their correctness with respect to the specification, program comprehension... The tutorial will sketch practical issues related to abstraction and refinement techniques for concurrent programming.

Following P. Cousot's Abstract Interpretation[6, 7, 5], abstraction is a theory of approximation of semantics. It formalizes the idea that a projection of the semantics on a simpler domain can help to compute properties (rather than to prove them). Properties given by the effective computation of the approximate semantics are closely related to the projection domain. All questions can not be solved, but all answers are always correct. Static analysis uses abstract interpretation to derive a computable semantics from the standard semantics. We will present such a framework specialized to the comprehension of sequential programs and study both the geometry of computation and how abstraction is related to it. Then, we will give some insights on how to extend the framework to concurrent programs.

Refinement of programs or refinement of specifications are techniques used in real case studies and tools[1, 8]; the B method developed by Abrial, is a suitable framework for developing distributed algorithms[4, 3, 2]. We will show how the B method may be used to develop distributed solutions to classical problems of distributed computations and we will give relationship to others approaches like actions systems, UNITY, TLA. The tutorial will be illustrated by demonstrations using the Atelier B environment.

References

1. J.-R. Abrial. *The B book - Assigning Programs to Meanings*. Cambridge University Press, 1996.
2. J.-R. Abrial. Extending b without changing it (for developing distributed systems). In H. Habrias, editor, *1st Conference on the B method*, pages 169–190, November 1996.
3. J.-R. Abrial and L. Mussat. Introducing dynamic constraints in B. In D. Bert, editor, *B'98 :Recent Advances in the Development and Use of the B Method*, volume 1393 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998.
4. J.R. Abrial. Development of the abr protocol. ps file, february 1999.
5. P. Cousot. *Calculational System Design*, chapter The Calculational Design of a Generic Abstract Interpreter. NATO ASI Series F. Amsterdam: IOS Press, 1999.
6. P. Cousot and R. Cousot. Abstract interpretation frameworks. *Journal of Logic and Computation*, 2(4):511–547, 1992.
7. P. Cousot and R. Cousot. Refining model checking by abstract interpretation. 6(1):69–96, January 1999.
8. STERIA - Technologies de l'Information, Aix-en-Provence (F). *Atelier B, Manuel Utilisateur*, 1998. Version 3.5.