CADP
A Protocol Validation and Verification Toolbox[⋆]

Jean-Claude Fernandez, Hubert Garavel, Alain Kerbrat,
Laurent Mounier, Radu Mateescu, and Mihaela Sighireanu

Inria/Vérimag, Miniparc Zirst, rue Lavoisier, 38330 Montbonnot St-Martin, France

1 Introduction

Cadp (Cæsar/Aldébaran Development Package) is a toolbox for protocol engineering. It offers a wide range of functionalities, ranging from interactive simulation to the most recent formal verification techniques. A first presentation of Cadp can be found in [FGM+92]. Work on Cadp started in 1985 and the first version of the toolbox (version A) was released in 1990. The latest official version (version Y) was released in May 1994. An improved version (version Z) is in preparation, for which beta-releases are available.

The Cadp toolbox contains several closely interconnected components:
Aldébaran, Bcg, Cæsar, Cæsar.adt, Open/Cæsar and Xtl. All these components are accessible through a unified graphical user-interface developed in the Eucalyptus project. We first present the overall functionalities of the toolbox, followed by individual presentations of each component.

More recently, a prototype, named Tgv (Test Generation using Verification techniques) [FJJ+96], for the automatic generation of test suites has been developed within the Cadp toolbox.

The Cadp toolbox has been installed in 130 sites[2] and used for a number of case studies, e.g. [KB95, GM96], including several industrial applications, such as the verification of the bus arbiter of Bull's PowerScale™ architecture.

2 Description languages and compilers

The Cadp toolbox accepts three different input formalisms:

– It accepts high-level protocol descriptions written in the Iso language Lotos [International Standard 8807]. The toolbox contains two compilers Cæsar and Cæsar.adt. They translate Lotos descriptions into C code which can be used for simulation, verification and testing purpose.

⋆ This work has been supported in part by the European Commission, under project ISC-CAN-65 "EUCALYPTUS-2: A European/Canadian Lotos Protocol Tool Set".

[2] The toolbox is distributed free of charge to universities and academic research centers (under a license agreement). E-mail: `caesar@imag.fr`

- It accepts low-level protocol descriptions specified as Labelled Transition Systems (Lts, for short), i.e., finite state machines with transitions labelled by action names.
- As an intermediate step, the Cadp toolbox accepts networks of communicating automata, i.e., finite state machines running in parallel and connected together using Lotos parallel composition and hiding operators.

The latest releases of the Cadp toolbox devote a growing importance to the concept of intermediate formats and programming interfaces, which allow the Cadp tools to be applied to protocol description written in other languages than Lotos (e.g., Sdl with the Geode compiler, etc.).

3 Validation and verification functionalities

The Cadp toolbox allows to cover most of the development cycle of a protocol by offering an integrated set of functionalities. These functionalities (and tools) are interactive or random simulation (Open/Cæsar), partial and exhaustive deadlock detection (Open/Cæsar and Aldébaran), test sequences generation (Tgv), verification of behavioural specifications with respect to a bisimulation relation (Aldébaran), verification of branching-time temporal logic specifications (Evaluator and Xtl).

All the validation and verification tools are based on a same principle consisting in the exploration of an Lts describing the exhaustive behaviour of the protocol under analysis. This Lts can be accessed through several representations: *The explicit representation* consists in the exhaustive list of the states and transitions of the Lts. A compact format (Bcg) is available to encode explicit representations efficiently. *The implicit representation* consists in a C library providing a set of functions allowing a dynamic exploration of the Lts. It is well adapted to perform “on the fly” verification, avoiding the generation of the whole Lts. *The symbolic representation* consists in a set of Binary Decision Diagrams (Bdd) encoding the transition relation of the Lts. It can be built from program’s description of higher level than the Lts level, thus allowing to take advantage of the Bdd structure sharing capabilities.

4 Presentations of the toolbox components

1. Aldébaran [FKM93] allows the comparison and the reduction of Ltss modulo various equivalence relations (such as strong bisimulation, observational equivalence, delay bisimulation, τ^*a bisimulation, branching bisimulation, and safety equivalence) and preorder relations (such as simulation preorder and safety preorder). The verification algorithms used in Aldébaran are based either on the Paige-Tarjan algorithm for computing the relational coarsest partition, or on the “on-the-fly” techniques proposed by Fernandez-Mounier, or on symbolic Lts representation using Binary Decision Diagrams (Bdds). Aldébaran has diagnosis capabilities that provide the user with explanations when two Ltss are found not to be related.

2. Bcg (Binary-Coded Graphs) is both a format for the representation of explicit Ltss and a collection of libraries and programs dealing with this format. Compared to Ascii-based formats for Ltss, the Bcg format uses a binary representation with compression techniques resulting in much smaller (up to 20 times) files. Bcg is independent from any source language but keeps track of the objects (types, functions, variables) defined in the source programs. The following tools are currently available for this format: Bcg_Io performs conversions between the Bcg format and a dozen of other formats; Bcg_Open establishes a gateway between the Bcg format and the Open/Cæsar environment; Bcg_Draw provides a 2-dimension graphical representation of Bcg graphs with an automatic layout of states and transitions; Bcg_Edit is an interactive editor which allows to modify manually the display generated by Bcg_Draw.
3. Cæsar [GS90] is a compiler which translates Lotos descriptions into Ltss. Cæsar proceeds in several steps, first translating the Lotos description to compile into an intermediate Petri Net model, which provides a compact representation of the control and data flows. Then, the Lts is produced by performing reachability analysis on this Petri net. Cæsar only handles Lotos specifications with static control features, which is usually sufficient for most applications. The current version of Cæsar allows the generation of large Ltss (some million states) within a reasonable lapse of time. The efficient compiling algorithms of Cæsar can also be exploited in the framework of the Open/Cæsar environment.
4. Cæsar.adt [Gar89] is a compiler that translates the data part of Lotos descriptions into libraries of C types and functions. Each Lotos sort or operation is translated into an equivalent C type or function. One must indicate to Cæsar.adt which Lotos operations are "constructors" and which are not (fairly obvious, in practice). Cæsar.adt does not allow non-free constructors ("equations between constructors"). Translation of large programs (several hundreds of lines) is usually achieved in a few seconds. Cæsar.adt can be used in conjunction with Cæsar, but it can also be used separately to compile and execute efficiently large abstract data types descriptions.
5. Open/Cæsar is an extensible programming environment for the design of applications working with the implicit representation of Ltss. Currently, several languages/compilers are connected to the Open/Cæsar environment, including: the Cæsar and Cæsar.adt compilers, the Bcg_Open gateway for explicit graphs, the Exp.Open gateway for networks of communicating automata, etc. Various application programs have already been written in the Open/Cæsar framework, including two interactive simulators (with shell-like and X-window interfaces), a random execution tool, a deadlock detection tool based on G. Holzmann's technique, a reachability analysis tool (with τ^*a on-the-fly reduction), a sequence-searching tool, an on-the-fly evaluator for branching-time μ-calculus, etc.
6. Xtl (eXecutable Temporal Language) is a functional-like programming language designed to allow an easy, compact implementation of various temporal logic operators. These operators are evaluated over an Lts encoded in the Bcg format. Besides the usual predefined types (booleans, integers, etc.) The Xtl language defines special types, such as sets of states, transitions, and labels of the Lts.

It offers primitives to access the informations contained in states and labels, to obtain the initial state, and to compute the successors and predecessors of states and transitions. The temporal operators can be easily implemented using these functions together with recursive user-defined functions working with sets of states and/or transitions of the Lts. A prototype compiler for Xtl has been developed, and several temporal logics like Hml, Ctl, Actl and Ltac have been easily implemented in Xtl.

References

[FGM$^+$92] Jean-Claude Fernandez, Hubert Garavel, Laurent Mounier, Anne Rasse, Carlos Rodríguez, and Joseph Sifakis. A Toolbox for the Verification of LOTOS Programs. In Lori A. Clarke, editor, *Proceedings of the 14th International Conference on Software Engineering ICSE'14 (Melbourne, Australia)*, pages 246–259. ACM, May 1992.

[FJJ$^+$96] Jean-Claude Fernandez, Claude Jard, Thierry Jéron, Laurence Nedelka, and César Viho. Using On-the-Fly Verification Techniques for the Generation of Test Suites. In R. Alur and T. A. Henzinger, editors, *Proceedings of the 8th International Conference on Computer-Aided Verification (Rutgers University, New Brunswick, NJ, USA)*, volume 1102 of *Lecture Notes in Computer Science*. Springer Verlag, 1996. Also available as INRIA Research Report RR-29XX.

[FKM93] Jean-Claude Fernandez, Alain Kerbrat, and Laurent Mounier. Symbolic Equivalence Checking. In C. Courcoubetis, editor, *Proceedings of the 5th Workshop on Computer-Aided Verification (Heraklion, Greece)*, volume 697 of *Lecture Notes in Computer Science*. Springer Verlag, June 1993.

[Gar89] Hubert Garavel. Compilation of LOTOS Abstract Data Types. In Son T. Vuong, editor, *Proceedings of the 2nd International Conference on Formal Description Techniques FORTE'89 (Vancouver B.C., Canada)*, pages 147–162. North-Holland, December 1989.

[GM96] Hubert Garavel and Laurent Mounier. Specification and Verification of various Distributed Leader Election Algorithms for Unidirectional Ring Networks. *Science of Computer Programming*, 1996. Special issue on Industrially Relevant Applications of Formal Analysis Techniques. Full version available as INRIA Research Report RR-29XX.

[GS90] Hubert Garavel and Joseph Sifakis. Compilation and Verification of LOTOS Specifications. In L. Logrippo, R. L. Probert, and H. Ural, editors, *Proceedings of the 10th International Symposium on Protocol Specification, Testing and Verification (Ottawa, Canada)*, pages 379–394. IFIP, North-Holland, June 1990.

[KB95] Alain Kerbrat and Slim Ben Atallah. Formal Specification of a Framework for Groupware Development. In G. v. Bochmann, R. Dssouli, and O. Rafiq, editors, *Proceedings of the 8th International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols FORTE'95 (Montreal, Quebec, Canada)*, October 1995. Short paper.

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