An Overview of the Eucalyptus Toolbox

H. Garavel
INRIA Rhône-Alpes / VERIMAG
ZIRST, 655, avenue de l'Europe
F-38330 Montbonnot Saint Martin
France
hubert.garavel@inria.fr

Abstract

This article presents the essential features of a protocol engineering environment, the EUCALYPTUS *toolbox, which has been developed or improved in the framework of two successive European-Canadian projects* EUCALYPTUS-*1 and* EUCALYPTUS-*2. This toolbox is based on the formal description technique* LOTOS *standardized by* ISO. *It offers a wide range of functionalities, including simulation, compilation, verification and test case generation for* LOTOS *descriptions.*

1 Introduction

The development of telecommunication protocols and distributed systems can be improved by the use of formal methods supported by appropriate software tools.

Formal description techniques such as the LOTOS language standardized by ISO [29] have been defined to allow a precise and unambiguous description of complex reactive systems. The design of LOTOS was motivated by the need for a language with a high abstraction level and a strong mathematical basis, which could be used for the description and analysis of complex systems. The LOTOS language features two clearly separated parts:

- The *data part* of LOTOS, used to describe data structures, is based on the well-known theory of algebraic abstract data types. In this approach, data structures are described by *sorts*, which represent value domains, and *operations*, which are mathematical functions defined on these domains. The meaning of operations is defined by algebraic *equations*. Sorts, operations, and equations are grouped in modules called *types*, which can be combined together using importation, renaming, parametrization, and actualization. The underlying semantics is that of initial algebras [9].
- The *control part* of LOTOS, used to describe dynamic behaviours, is based on the process algebra approach for concurrency, and appears to combine the best features of CCS [37, 38] and CSP [27]. LOTOS relies on a small set of basic operators, which represent primitive concepts of concurrent systems (sequential composition, non-deterministic choice, guard, parallel composition, rendez-vous, etc.) These operators are used to build algebraic terms that describe the behaviour of concurrent systems; the approach is compositional, since complex behaviours can be obtained by combining elementary ones.

During the past decade, a number of software engineering tools for LOTOS have been developed, often in large European projects. However, as such tools are innovative and intrinsically complex, the development and maintainance requires a time scale that exceeds by far the duration of short-term, finalized projects. We believe that robust and workable implementations can only be obtained from a sustained, long-term effort.

As an example of such a long-term effort, we present the results of two successive European-Canadian projects, named EUCALYPTUS-1 and EUCALYPTUS-2, the name EUCALYPTUS being an acronym for *European-Canadian Lotos Protocol Tool Set*. Four partners have been involved in these projects:

- The research project SPECTRE of INRIA (Grenoble, France);
- The University of Liège (Belgium), Institut d'Electricité Montefiore, Département Systèmes et Automatique;
- The University of Montréal (Québec, Canada), Départment Informatique et Recherche Opérationnelle;
- The University of Ottawa (Ontario, Canada), Department of Computer Science, Telecom-

munications Software Engineering Research Group.

Eucalyptus-1 is a two-year project, which started on January 1st, 1993. Eucalyptus-2 is a two-year followup project, which started on January 1st, 1995. Both projects have been supported by the European Commission (under contracts Esprit Ec-Ca 001 and Isc-Can-65), the Natural Sciences and Engineering Research Council of Canada (Nserc), and the Telecommunications Research Institute of Ontario (Trio).

Capitalizing on the experience acquired by the various partners in using the Lotos language for large-size applications and developing Lotos tools, the Eucalyptus-1 and Eucalyptus-2 projects have been taking aim at producing a complete, workable protocol engineering toolbox for Lotos. This "Eucalyptus toolbox" consists of several tools, which have been developed by the partners during the project, or improved from pre-existing Lotos environments, such as Grenoble's Cadp [13, 12] and Ottawa's XEludo [45]. Additionally, two other research teams, although not members of the Eucalyptus consortium, have decided to integrate their own tools in the Eucalyptus toolbox:

- The research project Pampa of Irisa (Rennes, France);
- Institut National des Télécommunications (Int, Evry, France).

This article presents the essential features of the Eucalyptus toolbox. It is organized as follows. Section 2 describes the functionalities of the toolbox from an external point of view. Section 3 takes a more internal view by presenting each component of the toolbox separately. Section 4 introduces various tools, which although not integrated yet in the toolbox, should become part of it in a near future. Section 5 demonstrates the practical usefulness of the Eucalyptus toolbox by giving a comprehensive list of applications and bibliographic references. Finally, Section 6 formulates some concluding remarks.

2 Functionalities of the EUCALYPTUS toolbox

Although the Eucalyptus toolbox groups different tools developed by different partners, extensive efforts have been done to achieve a smooth integration, by making tools compatible with each other, by developing gateways that allow different tools to interoperate, and by providing a unified user-interface.

From the user point of view, the Eucalyptus toolbox is primarily viewed through its graphical user-interface (Gui) based on X-windows. Functionally, the Gui is simple: it is divided into two main windows. The left sub-window presents the files present in the current Unix directory. There are different types of files (e.g., Lotos programs, C programs, test sequences, etc.) By clicking on a particular file, the list of operations (e.g. code generation, simulation, verification, etc.) permitted for this type of file is proposed. When an operation is invoked, its results are displayed in the right sub-window.

From 1994 to 1995, a first version of the Gui (see Figure 1) was developed by all the Eucalyptus partners, using the XtPanel interface builder developed by Steve Cole and Dave Nichols (Stanford University). However, as various problems (portability, maintainability, and speed) were faced with XtPanel, a second version of the Gui (see Figure 2) was developed using the popular Tcl-Tk interface builder. This new version solves all the aforementioned problems and should be made available by September 1996.

The following functionalities can be accessed using the Gui:

Extensions to LOTOS: It is possible to enrich Lotos with new constructs (especially, compact notations for type definitions) by using a preprocessor which allows macro-definitions to be defined and expanded into standard Lotos.

Analysis: The Eucalyptus toolbox contains front-end tools performing lexical, syntactic, and static semantics analysis according to the Lotos standard [29].

Report generation: There are tools for pretty-printing Lotos descriptions and generating cross-references (for process names, gate names, type names, etc.)

Code generation: There are compilers to translate Lotos types and process definitions into C code that can be executed and/or embedded in application programs.

Simulation: The toolbox supports various forms of simulation, such as interactive simulation (step-by-step execution with backtracking), symbolic expansion (in which input values are handled symbolically), goal-oriented simulation, and random execution.

Exhaustive verification: The toolbox allows to generate the Labelled Transition System (Lts) corresponding to a Lotos description. Ltss with millions of states and transitions can be generated, within the limits of memory available. Ltss are stored using a compact format with ad hoc data compression techniques.

These LTSs can be analyzed and verified in several ways. They can be minimized and compared modulo various equivalence (bisimulations) and preorder relations. They can also be validated using properties expressed as formulas of temporal logic or μ-calculus. Connections with external verification tools are also provided.

Compositional verification: Due to the well-known state explosion problem, exhaustive generation of LTSs is not always possible. The EUCALYPTUS toolbox allows to divide a LOTOS description into parallel processes, to generate the LTSs corresponding to these processes, to minimize these LTSs modulo a bisimulation relation, and to build the LTS for the whole system by recombining these reduced LTSs. In practice, this "divide and conquer" approach often gives satisfactory results.

"On the fly" verification: As an alternative approach to avoid the state explosion problem, the EUCALYPTUS toolbox allows certain properties to be verified without generating the whole LTS first. "On the fly" verification techniques range from simple properties, such as deadlock detection and search of particular execution sequences, up to more elaborated properties such as "on the fly" comparisons of LTSs modulo bisimulation relations and "on the fly" evaluation of branching-time μ-calculus formulas.

Graph drawing: The toolbox contains several tools to display the LTSs generated from LOTOS descriptions. For small LTSs (e.g., with less than one hundred states), these tools generate automatically a PostScript representation, using heuristics for 2-dimensional or 3-dimensional layout. If necessary, these automatically-generated pictures can be improved by hand using an interactive editor available with the toolbox. Interfacing with external viewers such as AUTOGRAPH [42] is also provided.

Test generation: From the LOTOS descriptions, one can automatically generate test sequences, which will be used to assess the conformity of real implementations with respect to the original descriptions. Several strategies for test generation are already available, others are expected to be integrated soon.

Trace analysis: Finally, the EUCALYPTUS toolbox allows to verify whether a given trace (execution sequence) can be obtained from a LOTOS description. This allows to validate test suites and their verdicts with respect to a LOTOS description representing the behavior of the system under test.

3 Tools already available

At present, the EUCALYPTUS toolbox contains the following tools:

APERO (Liège): The data part of LOTOS is based upon a highly abstract model (multi-sorted algebras), which yields considerable expressive power to the language but turns out to be too fundamental with respect to the usual needs in dynamic system descriptions. Moreover, this model lacks some theoretical computability properties (decidability), so that existing software tools (simulators, compilers, verifiers, etc.) have to put some restrictions on the LOTOS descriptions which they handle.

Furthermore, when taking a look at existing LOTOS descriptions, it appears that the description of the needed data types is often very large. This lack of concision had already been identified by G. Scollo in 1986 who proposed to extend the language definition with shorthand notations. However, most of the extensions proposed by G. Scollo were not included in the standardized definition of the language. Thus, the problem remained unsolved as system descriptions using the extended language were neither internationally accepted by the scientific community nor tractable by LOTOS related tools.

APERO (a loose acronym for "Act-one PrEpROcessor") [41] offers a technical solution to this problem. It behaves as a pre-processor that translates the aforementioned shorthand notations into standard LOTOS, thus allowing simple and compact specification of common data structures in LOTOS while keeping compatibility with the standard language, and therefore with existing tools. This pre-processing functionality is combined with an extension of the data type library of LOTOS, adding generic definitions for common data structures such as records or enumerations. This extended library contains an infinite number of definitions; for each LOTOS description, APERO generates a subset containing only the needed ones.

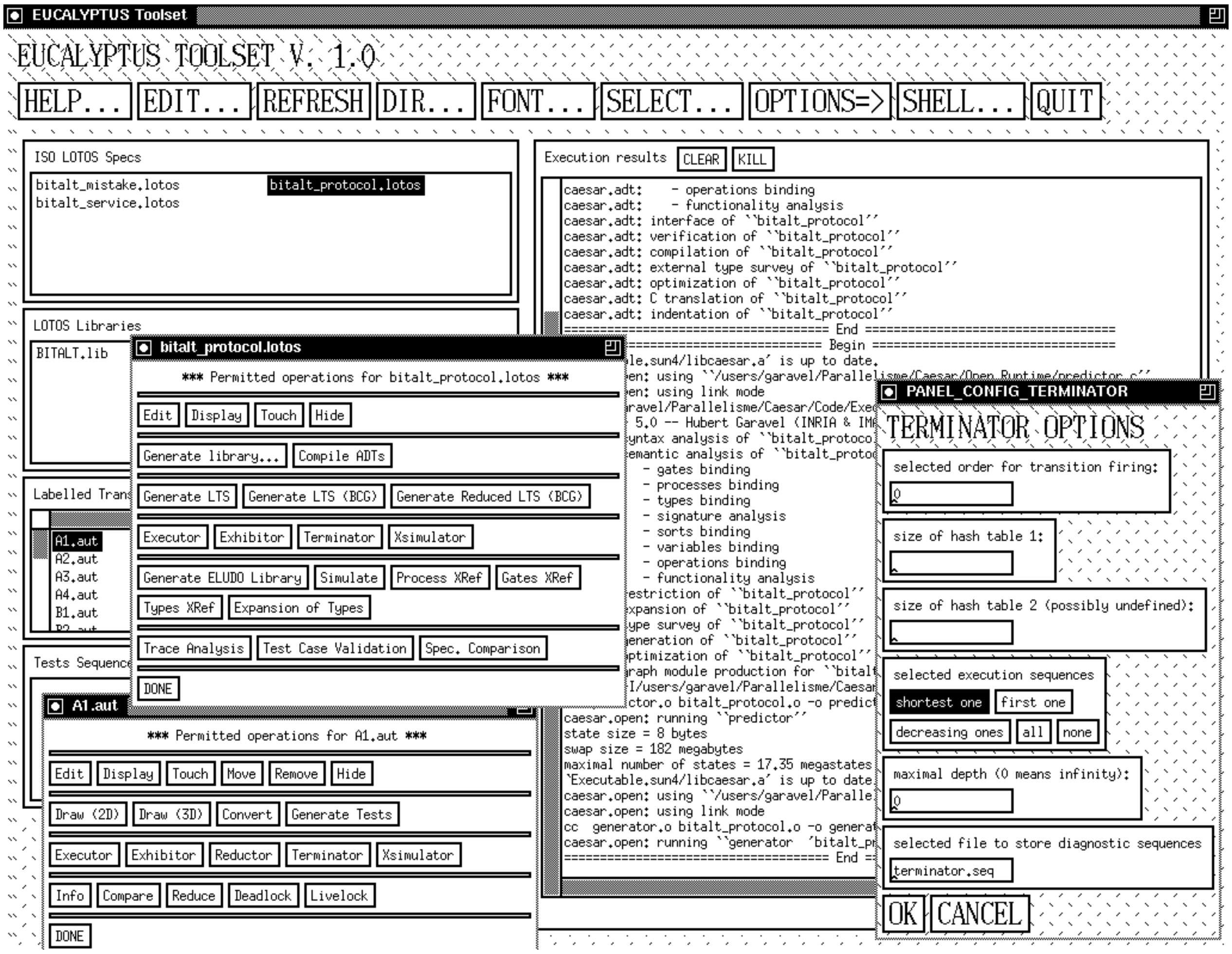

Figure 1: Version 1 of the EUCALYPTUS graphical user-interface

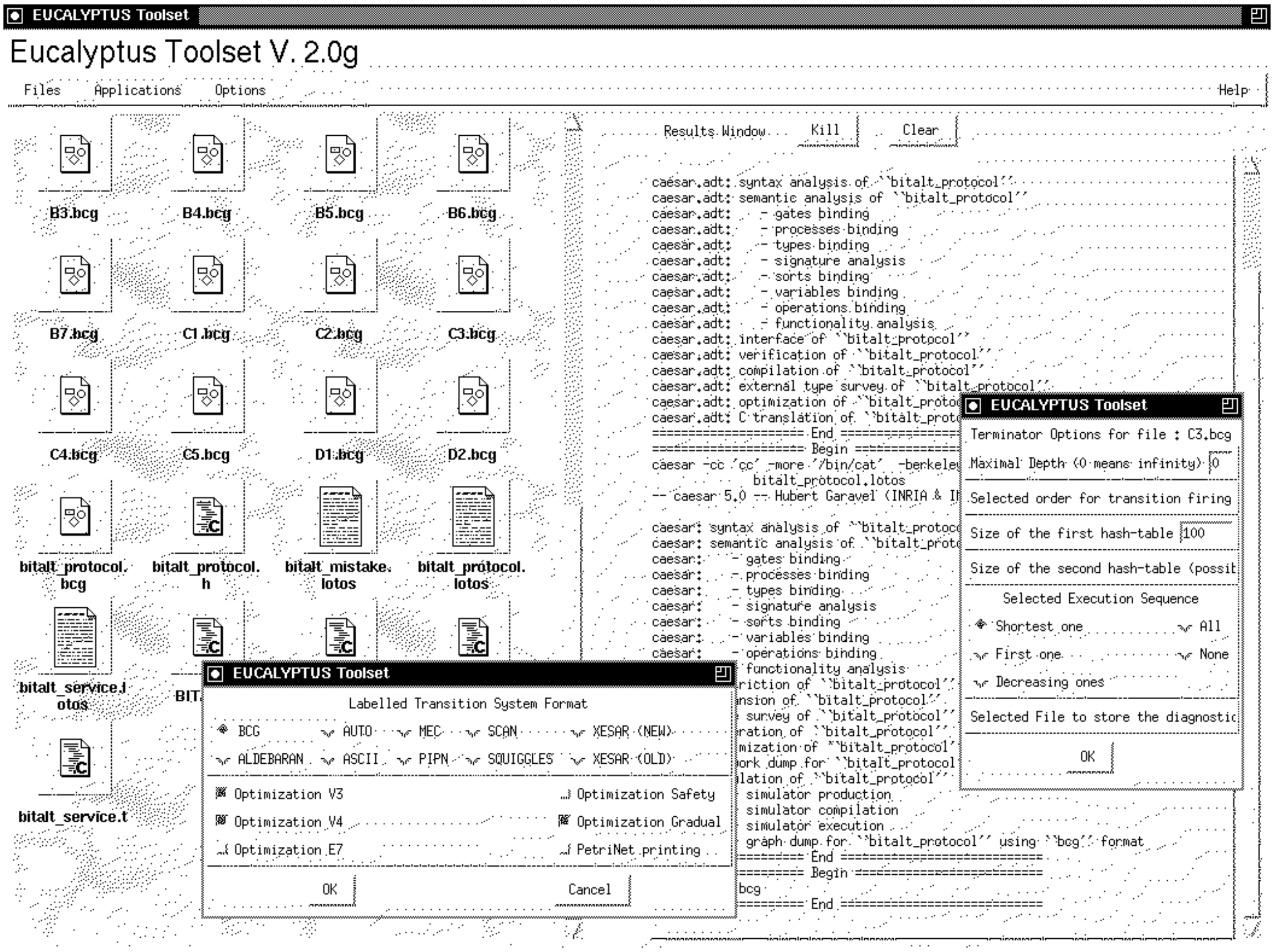

Figure 2: Version 2 of the EUCALYPTUS graphical user-interface

CÆSAR (Grenoble): Cæsar [18, 22] is a compiler that translates Lotos descriptions into Ltss. Cæsar translation algorithms proceed 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 descriptions with static control features: process recursion is not allowed on the left and right hand part of "`|[...]|`", nor on the left hand part of "`>>`" and "`[>`". In spite of these restrictions, the subset of Lotos accepted by Cæsar is large and 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. These Ltss can be generated either in the Bcg format or in other formats used by various verification tools.

Moreover, the efficient compiling algorithms of Cæsar can also be exploited in the framework of the Open/Cæsar environment (see below).

CÆSAR.ADT (Grenoble): Cæsar.adt [18, 23] is a compiler that translates the data part of Lotos descriptions into libraries of C types and functions. Each Lotos sort is translated into an equivalent C type and each Lotos operation is translated into an equivalent C function (or macro-definition). Cæsar.adt also generates C functions for comparing and printing abstract data types values, as well as iterators for the sorts the domain of which is finite. The user can also decide to provide hand-written C code for some Lotos sorts and/or operations.

The user must indicate to Cæsar.adt which Lotos operations are "constructors" and which are not. Cæsar.adt does not allow non-free constructors ("equations between constructors"). However, it is always possible to transform a Lotos description in order to remove equations between constructors.

Translation of large programs (thousands 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.

ALDEBARAN (Grenoble): Aldébaran [11, 16, 15] is a tool for comparing and reducing Ltss (or networks of communicating Ltss) modulo various equivalence relations (such as strong bisimulation, observational equivalence, delay bisimulation, $\tau^* a$ bisimulation, branching bisimulation, and safety equivalence) and preorder relations (such as simulation preorder and safety preorder). For instance, one can check that the Lts of a protocol (generated using Cæsar) is equivalent (modulo various abstraction criteria) to the Lts representing the service of this protocol. Aldébaran has diagnostic capabilities that provide the user with explanations (execution sequences) when two Ltss are found to be not equivalent.

The verification techniques implemented 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).

OPEN/CÆSAR (Grenoble): Open/Cæsar is an extensible environment for designing programs performing simulation, execution, verification (partial, on-the-fly, etc.), and test generation. It allows these programs to be developed in a simple and modular framework, independently from any particular description language. Open/Cæsar is open and documented, so that users can extend the environment by adding their own libraries and programs.

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.

Several application programs are currently available within the Open/Cæsar framework, including:

- Evaluator is an on-the-fly evaluator for branching-time μ-calculus;
- Executor is a random execution tool;
- Exhibitor searches for execution sequences matching a given pattern defined by a regular expression;
- Generator performs reachability analysis and generates an Lts;
- Reductor is similar to Generator, but performs on-the-fly reduction modulo the $\tau^* a$ equivalence;
- Simulator is an interactive simulator with a command-line interface;
- Terminator is a deadlock detection tool based on G. Holzmann's *bit space* technique [28];

- XSIMULATOR is another interactive simulator with a graphical user interface based on X-WINDOWS.

BCG (Grenoble): BCG (Binary-Coded Graphs) [19, 43] 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. Due to its extended functionalities, the BCG format is more complex than ASCII-based formats; however, a large set of programming interfaces, object code libraries, and programs makes this extra complexity almost invisible to the user. Several tools are currently available for this format:

- BCG_IO performs conversions between the BCG format and a dozen of other formats, including the common format FC2 [35] designed in the European project CONCUR-2, which provides for interfacing external verification tools such as the CONCURRENCY WORKBENCH [7] and AUTO/AUTOGRAPH [42];
- BCG_OPEN establishes a gateway between the OPEN/CÆSAR environment and the BCG format, thus allowing all OPEN/CÆSAR tools to be applied to BCG graphs;
- BCG_DRAW provides a graphical representation of BCG graphs with an automatic, 2-dimensional layout of states and transitions;
- BCG_EDIT is an interactive editor which allows to modify manually the graphical representations generated by BCG_DRAW.

ELUDO (Ottawa): The LOTOS environment under development at the University of Ottawa is named XELUDO (an acronym for "Environnement LOTOS de l'Université d'Ottawa") [45]. The main currently operational tools in XELUDO are ISLA, SELA, and GOAL (see below). XELUDO includes common facilities available to all the tools, including:

1. A graphical interface for ASCII terminals, based upon the "curses" library of UNIX;
2. A graphical interface based upon X-WINDOWS;
3. A LOTOS translator, which is a standalone program that performs initial verification and preprocessing on the LOTOS description to be analyzed under XELUDO. It may be executed from inside XELUDO or as and independent LOTOS specifier's development tool. The main functions supported by the translator are the following:
   - Lexical, syntactical and static semantics analysis;
   - Translation to a Prolog internal representation suitable for the XELUDO tools, which are written in Prolog; the translation of the LOTOS description into its equivalent Prolog form is done only once;
   - Generation of cross-references of processes, gates and types;
   - Creation of a user-defined type library, to replace the default standard library;
   - Output of a parse tree of the description;
   - Pretty-printing of the LOTOS description.

ISLA (Ottawa): ISLA [25] provides a step-by-step execution mode which allows to simulate the sequence of possible actions that are permitted by a LOTOS description. The execution of a LOTOS description can be represented as a tree, where the root of the tree is the description itself, the intermediate nodes are behavior expressions and the branches of the tree represent LOTOS actions.

The user may choose to simulate the whole description at once, or only parts of it (certain processes). At each step, during simulation, the user is prompted with a menu of possible next actions. The user chooses the next action to be executed and, if the selected action requires data to be supplied by the environment (the user plays the role of the environment), then data must be entered for the simulation to continue.

A menu-driven facility prompts the user with appropriate choices for data. Also, at any point during simulation, the user may ask to see the current behavior of the system or the behavior that will result by executing one of the possible next actions.

Furthermore, ISLA displays the complete set of execution paths that have been exercised by the user during the current simulation session, in the form of a tree. This allows her to check, for example, where in the execution tree a guard was evaluated and where certain value identifiers were instantiated. So, if certain chosen values did not lead to the desired sequence, the user can back up to a point where a different value can be entered. Therefore, the user may return to a previous execution point, and redo execution from that point with different choices.

It is also possible to save the sequence of actions, executed up to some point in the tree, in the memory or in an external file, thereby gaining the possibility of continuing the simulation, at a later time, from where it was left off.

TESTGEN (Evry): TESTGEN [5] is a tool for generating optimal test sequences from the LOTOS description of a protocol, in order to check the conformance of a protocol implementation to its formal description. The classical technique of Unique Input/Output sequences is adapted to the LTSs obtained from LOTOS specifications by defining a new concept: the Unique Event sequence. This approach, combined with powerful optimization techniques (Rural Chinese Postman Tour and Greedy Overlap) produces optimal test sequences, which check the conformance of a protocol implementation by performing a minimal cost tour of the reference LTS. This method overcomes the limited controllability and observability of the protocol implementation by an external tester.

TETRA (Montréal): TETRA (TEst and TRace Analyzer) [3, 4, 8] compares a given trace of interactions with a reference description written in LOTOS, checking whether an execution history of the LOTOS description could produce the given trace. The tool allows for two modes of analysis:

- off-line trace analysis, where the reference description is compiled together with the traces to be analyzed, which are written in the form of LOTOS processes;
- on-line trace analysis, which compiles the reference description alone and analyses the interactions of a trace one after the other as they are received from another site executing/simulating the implementation under test. The result of a trace analysis is either "valid trace" or "invalid trace".

In the latter case, an optional error diagnostic facility provides indications about possible causes of the discrepancy between the trace and the LOTOS description according to various error hypotheses.

Beside trace analysis, TETRA also has an option to validate test cases and their verdicts with respect to a reference description which defines the expected behavior of the tested system. It establishes in this case whether or not the branches of a test case conform to the reference description. Diagnostic analysis for erroneous branches is possible as well.

VISCOPE (Rennes): VISCOPE [31] is a tool for automatic display (in 2- or 3-dimensions) of LTSs. Based on partial-order theory, it generates a PostScript representation with a layout of states and transitions that make concurrent actions visible, most notably by preserving the "diamonds" in the LTSs, which are created by the interleaving of concurrent actions. A new version of VISCOPE is currently under development, which should provide full compatibility with the BCG format and the BCG_EDIT tool.

4 Tools to be integrated

There exist other tools, which, although available separately, are not integrated yet in the EUCALYPTUS toolbox. We mention them since some of them are likely to be integrated in a near future:

BRIDGE (Ottawa): BRIDGE is a tool that strengthens the toolbox integration, by allowing combined use of complementary tools. It provides a convenient gateway between verification and simulation tools.

The verification tools ALDÉBARAN, BCG, and OPEN/CÆSAR can discover quickly "faulty" execution sequences, i.e., execution sequences that do not correspond to a permitted behaviour. However, as these tools operate at the level of the LTS model, the correspondence between faulty execution sequences and source LOTOS descriptions is (partially) lost, due to the sophisticated, optimized translations performed by the CÆSAR/CÆSAR.ADT compilers.

The BRIDGE tool (used in conjunction with OPEN/CÆSAR's EXHIBITOR tool) solves this problem by allowing faulty execution sequences to be re-executed step by step using the interactive simulator ISLA. As ISLA maintains a direct correspondence between the simulation steps and the source LOTOS descriptions, it is very easy for the user to understand the reason of the error.

SELA (Ottawa): SELA [24] is a symbolic expander for LOTOS. The step-by-step execution mode provided by ISLA is very useful, but it is also time consuming.

SELA allows one to compute the tree of all possible next actions from the current point, or any given point in the tree. This is known as the symbolic execution tree because expressions are computed in terms of (not necessarily) bounded value identifiers. In terms of LOTOS theory, the calculation of this tree is called 'expansion'. When generating a symbolic tree, guards and predicates, whose values depend on interactions with the environment, are assumed to be true. In addition, the user is required to set limits on the depths and widths of the symbolic tree to be generated.

Although calculation of the symbolic tree may not terminate, it can yield finite initial subtrees of an infinite monolithic LOTOS description equivalent to the original one.

GOAL (Ottawa): GOAL [26] is a tool providing "Goal-Oriented Execution" for LOTOS. Both ISLA and SELA suffer from the well-known problem of state explosion: for most practical LOTOS descriptions, execution trees grow very quickly.

Goal-oriented execution attempts to relieve this problem. In this type of execution, the tool attempts to find execution sequence(s) leading to a certain action or sequence of actions (these are the "goals"). The LOTOS description is scanned statically to find where the action(s) can be found. Then the inference rules are applied, taking into consideration this information. The result are sets of execution sequences reaching the goals.

TGV (Grenoble and Rennes): TGV (Test Generation based on Verification) [14] takes as inputs an LTS and an automaton formalizing the behavioural part of a test purpose and generates a test case in the standard TTCN format. This translation is performed in successive steps.

The first step transforms the LTS into a graph representing the observable behaviour of the protocol description in the testing environment, taking into account the concurrency which may be produced by asynchronous interaction between the tester and the implementation. Additional transformations are also performed during this step: abstraction of unobservable internal actions and determinization.

The next step, the kernel of TGV, is based upon a depth-first search algorithm, which produces a *test graph* containing all informations needed in TTCN test cases.

The last step takes as input the *test graph*, extracts from the transition labels the message parameters and produces the constraint part. The remaining graph is unfolded into a tree describing the behavioural part of the test case. Finally the constraint and behavioural parts of the test case are translated into the graphical format TTCN GR.

XTL (Grenoble): 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 generated from a source program and encoded in the BCG format.

Beside the usual predefined types (booleans, integers, characters, strings, etc.), the XTL language allows to access all types and functions defined in the source program and provides special types, such as states, transitions, sets of states, sets of transitions, and labels of the LTS. It offers primitives to access the informations contained in states and labels: this allows to express "basic predicates" (i.e. containing no temporal operators) defined over the states and labels of the LTS. There also exist predefined functions to access the initial state and the successors and predecessors of states and transitions, thus allowing the exploration of the transition relation.

Therefore, the XTL language can be viewed as a high-level query language for accessing all information contained in BCG graphs. The temporal operators of various temporal logics can be implemented as recursively defined user XTL functions operating on sets of states and transitions. A prototype compiler for XTL has been developed, and several temporal logics like HML, CTL, ACTL and LTAC have already been implemented in XTL.

5 Applications

The EUCALYPTUS tools have been used in many different application fields. We give here a (non-exhaustive) list of case-studies, most of which have led to scientific publications:

Cryptography: message authentication algorithm [30, 40, 46];

Embedded software: car overtaking protocol [10], Airbus 330/340 Flight Warning Computer [20], railyard systems [17];

Network protocols: bus instrumentation protocol FIP [1], OSI95 transport service [34], bounded retransmission protocol [36], Internet transport protocol TCP [44];

Telephony: transit node message router [39], Plain Ordinary Telephone Service (SICS, Sweden), feature interactions in telephony systems [33];

Distributed systems: REL/REL reliable atomic multicast protocol [2], groupware protocols [32], distributed leader election algorithms [21],

Hardware protocols: bus arbiter of Bull's POWERSCALE architecture [6].

6 Conclusion

To be effective, formal description techniques need to be supported by software engineering tools, which allow to compile, simulate, verify and generate tests. Based on LOTOS, a powerful and sound formal language, the EUCALYPTUS toolbox offers these functionalities. It gathers a number of tools, which have been either developed or improved in the framework of two successive European/Canadian projects EUCALYPTUS-1 and EUCALYPTUS-2. These tools have been interfaced and integrated into a unified graphical user-interface. Other tools will be added to the toolbox in a near future. The EUCALYPTUS tools have been assessed on realistic case-studies, some of them in industrial context. Finally, beside the toolbox development, it is worth mentioning the significant role played by the EUCALYPTUS-2 project in the definition of EXTENDED LOTOS, a revised version of LOTOS, which is currently under elaboration within ISO.

Acknowledgements

As the EUCALYPTUS toolbox integrates a number of tools from various institutions, this article compiles a number of tool descriptions provided by several persons. The author is grateful to those who provided the material for the basis of a unified presentation of the EUCALYPTUS toolbox. He would also like to thank all the scientists, engineers, and students who have contributed, directly or indirectly, to the EUCALYPTUS-1 and EUCALYPTUS-2 projects, namely:

In Grenoble: Marius Bozga, Dr. Jean-Claude Fernandez, Jean-Michel Frume, Dr. Hubert Garavel, Dr. Alain Kerbrat, Dr. Laurent Mounier, Radu Mateescu, Renaud Ruffiot, Dr. Joseph Sifakis, Mihaela Sighireanu, and Louis-Pascal Tock;

In Evry: Pr. Ana Cavalli and Toma Macavei;

In Liège: France Bierbaum, Pr. André Danthine, Jean-Charles Henrion, Michel Jankowski, Dr. Guy Leduc, Luc Léonard, and Charles Pecheur;

In Ottawa: Xavier Etchevers, Mark Jorgensen, Pr. Luigi Logrippo and Jacques Sincennes;

In Montréal: Omar Bellal, Pr. G. v. Bochmann, and Daniel Ouimet;

In Rennes: Dr. Claude Jard and Dr. Thierry Jeron.

References

[1] Pierre Azema, Khalil Drira, and François Vernadat. A Bus Instrumentation Protocol Specified in LOTOS. In Juan Quemada, José Manas, and Enrique Vázquez, editors, *Proceedings of the 3rd International Conference on Formal Description Techniques FORTE'90 (Madrid, Spain)*. North-Holland, November 1990.

[2] Simon Bainbridge and Laurent Mounier. Specification and Verification of a Reliable Multicast Protocol. Technical Report HPL-91-163, Hewlett-Packard Laboratories, Bristol, U.K., October 1991.

[3] G.v. Bochmann and O. Bellal. Test Result Analysis with respect to Formal Specification. In *Proceedings of the 2nd International Workshop on Protocol Test Systems (Berlin, Germany)*, October 1989.

[4] G.v. Bochmann, D. Desbiens, M. Dubuc, D. Ouimet, and F. Saba. Test Result Analysis and Validation of Test Verdicts. In *Proceedings of the 3rd International Workshop on Protocol Test Systems (McLean, Virginia, USA)*, October 1990.

[5] Ana Cavalli, Sung Un Kim, and Patrick Maigron. Improving conformance testing for LOTOS. In Richard L. Tenney, Paul D. Amer, and M. Umit Uyar, editors, *Proceedings of the 6th International Conference on Formal Description Techniques FORTE'93 (Boston, MA, USA)*, pages 367–384. North-Holland, October 1993.

[6] Ghassan Chehaibar, Hubert Garavel, Laurent Mounier, Nadia Tawbi, and Ferruccio Zulian. Specification and Verification of the Power-Scale Bus Arbitration Protocol: An Industrial Experiment with LOTOS. In Reinhard Gotzhein and Jan Bredereke, editors, *Proceedings of the Joint International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols, and Protocol Specification, Testing, and Verification FORTE/PSTV'96 (Kaiserslautern, Germany)*. IFIP, Chapman & Hall, October 1996. Full version available as INRIA Research Report RR-2958.

[7] R. Cleaveland, J. Parrow, and B. Steffen. The Concurrency Workbench. In J. Sifakis, editor, *Proceedings of the 1st Workshop on Automatic Verification Methods for Finite State Systems (Grenoble, France)*, volume 407 of *Lecture Notes in Computer Science*, pages 24–37. Springer Verlag, June 1989.

[8] M. Dubuc, G.v. Bochmann, O. B. Bellal, and F. Saba. Translation from TTCN to LOTOS and the Validation of Test Cases. In Juan Quemada, José Manas, and Enrique Vázquez, editors, *Proceedings of the 3rd International Conference on Formal Description Techniques FORTE'90 (Madrid, Spain)*. North-Holland, November 1990.

[9] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 1 — Equations and Initial Semantics*, volume 6 of *EATCS Monographs on Theoretical Computer Science*. Springer Verlag, 1985.

[10] Patrik Ernberg, Lars-åke Fredlund, and Bengt Jonsson. Specification and Validation of a Simple Overtaking Protocol using LOTOS. In Kenneth R. Parker and Gordon A. Rose, editors, *Proceedings of the 4th International Conference on Formal Description Techniques FORTE'91*, pages 377–392. North-Holland, 1991.

[11] Jean-Claude Fernandez. An Implementation of an Efficient Algorithm for Bisimulation Equivalence. *Science of Computer Programming*, 13(2–3):219–236, May 1990.

[12] Jean-Claude Fernandez, Hubert Garavel, Alain Kerbrat, Radu Mateescu, Laurent Mounier, and Mihaela Sighireanu. CADP: A Protocol Validation and Verification Toolbox. In Rajeev Alur and Thomas A. Henzinger, editors, *Proceedings of the 8th Conference on Computer-Aided Verification (New Brunswick, New Jersey, USA)*, August 1996.

[13] 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.

[14] Jean-Claude Fernandez, Claude Jard, Thierry Jéron, 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)*, August 1996.

[15] 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.

[16] Jean-Claude Fernandez and Laurent Mounier. A Tool Set for Deciding Behavioral Equivalences. In *Proceedings of CONCUR'91 (Amsterdam, The Netherlands)*, August 1991.

[17] Lars-åke Fredlund and Fredrik Orava. An Experiment in Formalizing and Analysing Railyard Configurations. In Z. Brezočnik and T. Kapus, editors, *Proceedings of COST 247 International Workshop on Applied Formal Methods in System Design (Maribor, Slovenia)*. University of Maribor, Slovenia, June 1996.

[18] 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.

[19] Hubert Garavel. Binary Coded Graphs — Definition of the BCG Format (version 1.0). Rapport Technique à paraître, INRIA Rhône-Alpes, Grenoble, 1994.

[20] Hubert Garavel and René-Pierre Hautbois. *Experimenting LOTOS in Aerospace Industry*. In Teodor Rus and Charles Rattray, editors, *Theories and Experiences for Real-Time System Development*, volume 2 of *Amast Series in Computing*, chapter 11. World Scientific, 1994.

[21] Hubert Garavel and Laurent Mounier. Specification and Verification of various Distributed Leader Election Algorithms for Unidirectional Ring Networks. *Science of Computer Programming*, 1996. To appear.

[22] 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.

[23] Hubert Garavel and Philippe Turlier. CÆSAR.-ADT : un compilateur pour les types abstraits algébriques du langage LOTOS. In Rachida Dssouli and Gregor v. Bochmann, editors, *Actes du Colloque Francophone pour l'Ingénierie des Protocoles CFIP'93 (Montréal, Canada)*, 1993.

[24] B. Ghribi and L. Logrippo. A Validation Environment for LOTOS. In A. Danthine, G. Leduc, and P. Wolper, editors, *Proceedings of the 13th IFIP International Workshop on Protocol Specification, Testing and Verification (Liège, Belgium)*, pages 93–108. IFIP, North-Holland, May 1993.

[25] R. Guillemot, R. Haj-Hussein, and L. Logrippo. Executing Large LOTOS Specifications. In S. Aggarwal and K. Sabnani, editors, *Proceedings of the 8th International Workshop on Protocol Specification, Testing and Verification (Atlantic City, NJ, USA)*, pages 399–410. IFIP, North-Holland, 1988.

[26] M. Haj-Hussein, L. Logrippo, and J. Sincennes. Goal-Oriented Execution of LOTOS Specifications. In M. Diaz and R. Groz, editors, *Proceedings of the 4th International Conference on Formal Description Techniques FORTE'92 (Perros-Guirec, France)*, pages 311–327. North-Holland, 1992.

[27] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[28] Gerard J. Holzmann. *Design and Validation of Computer Protocols*. Software Series. Prentice Hall, 1991.

[29] ISO/IEC. LOTOS — A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour. International Standard 8807, International Organization for Standardization — Information Processing Systems — Open Systems Interconnection, Genève, September 1988.

[30] ISO/IEC. Approved Algorithms for Message Authentication — Part 2: Message Authenticator Algorithm. International Standard 8731-2, International Organization for Standardization — Banking, Genève, 1992.

[31] Claude Jard and Thierry Jéron. 3D layout of reachability graphs of communicating processes. In R. Tamassia and I. G. Tollis, editors, *Proceedings of the DIMACS Workshop Graph Drawing'94 (Princeton, New Jersey, USA)*, volume 894 of *Lecture Notes in Computer Science*. Springer Verlag, 1994. Bilingual version available as IRISA Research Report PI-854.

[32] 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.

[33] Henri Korver. Detecting Feature Interactions with CÆSAR/ALDEBARAN. *Science of Computer Programming*, 1996. To appear.

[34] Luc Léonard. *The LOTOS Specification of the Enhanced Transport Service*. In *The OSI95 Transport Service with Multimedia Support*, pages 239–244 and 398–515. Springer Verlag, 1994.

[35] Eric Madelaine and Robert de Simone. *FC2: Reference Manual Version 1.1*. INRIA, Sophia-Antipolis (France), July 1993.

[36] R. Mateescu. Formal Description and Analysis of a Bounded Retransmission Protocol. In Z. Brezočnik and T. Kapus, editors, *Proceedings of the COST 247 International Workshop on Applied Formal Methods in System Design (Maribor, Slovenia)*. University of Maribor, Slovenia, June 1996.

[37] Robin Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer Verlag, 1980.

[38] Robin Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[39] Laurent Mounier. A LOTOS Specification of a Transit-Node. Rapport SPECTRE 94-8, VERIMAG, Grenoble, March 1994.

[40] Harold B. Munster. LOTOS Specification of the MAA Standard, with an Evaluation of LOTOS.

NPL Report DITC 191/91, National Physical Laboratory, Teddington, Middlesex, UK, September 1991.

[41] Charles Pecheur. VLib: Infinite Virtual Libraries for LOTOS. In A. Danthine, G. Leduc, and P. Wolper, editors, *Proceedings of the 13th IFIP International Workshop on Protocol Specification, Testing and Verification (Liège, Belgium)*, pages 1–16. IFIP, North-Holland, May 1993.

[42] Valérie Roy and Robert de Simone. Auto/Autograph. In R. P. Kurshan and E. M. Clarke, editors, *Proceedings of the 2nd Workshop on Computer-Aided Verification (Rutgers, New Jersey, USA)*, volume 3 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 477–491. AMS-ACM, June 1990.

[43] Renaud Ruffiot. Définition et réalisation d'un atelier logiciel pour l'étude des systèmes de transitions. Mémoire d'ingénieur cnam, Institut IMAG, Grenoble, December 1994.

[44] Ina Schieferdecker. Abruptly-Terminated Connections in TCP – A Verification Example. In Z. Brezočnik and T. Kapus, editors, *Proceedings of the COST 247 International Workshop on Applied Formal Methods in System Design (Maribor, Slovenia)*. University of Maribor, Slovenia, June 1996.

[45] B. Stepien, J. Tourrilhes, and J. Sincennes. ELUDO: The University of Ottawa LOTOS Toolkit. Technical report, University of Ottawa, 1994. Obtainable by FTP on lotos.csi.uottawa.ca.

[46] Philippe Turlier. La compilation des types abstraits algébriques du langage LOTOS. Mémoire d'ingénieur cnam, Laboratoire de Génie Informatique — Institut IMAG, Grenoble, December 1992.