VeriTLA$^+$: A Verification Environment for TLA$^+$ Applied To Service Descriptions

Stephan Merz (coordinator)
Charles Consel Damien Doligez Gilles Muller

1 Motivation

The objective of this project is to construct a verification environment for TLA$^+$ [6], a specification language for reactive and distributed algorithms and systems, and to validate it in the context of service descriptions for kernels of operating systems and for communication services, in particular, telephony. TLA$^+$ is based on ordinary (set-theoretic) mathematics to describe data structures, extended with elementary notations of programming languages such as records and arrays. Currently, tool support for TLA$^+$ is mainly provided by TLC, a model checker for finite-state instances of systems that allows users to validate and debug system specifications. However, due to the number of reachable states, many specifications can not be model checked on large enough instances to catch subtle bugs. A machine-checked mathematical proof can guarantee that a specification satisfies the properties asserted about it. We propose here to construct a verification environment for TLA$^+$ that will allow a user to carry out such proofs for high-level descriptions of algorithms and (distributed) systems. Because the environment will be based on theorem-proving techniques, the complexity of verification is not limited by the size of state spaces but only by the complexity of the system description.

We propose to validate the verification environment by applying it to models of services for operating systems and for telephony. These applications are typical for the class of systems targeted by TLA$^+$; they are of manageable complexity and should therefore provide a promising test-bed for our project. Instead of specifying services directly in TLA$^+$, we propose to use domain-specific languages (DSLs) to describe and configure these services. This approach allows us to verify a compilation scheme once and for all rather than to verify individual instances, amortizing the complexity of verification. Two of the teams participating in this project have ample experience with DSLs (Bossa and SPL), and these languages will provide us with actual case studies. If successful, VeriTLA$^+$ will help increase the confidence in these languages and their support tools. We are particularly interested in the genericity of the approach, which should be applicable to other DSLs, beyond the particular domain of application.

Despite several decades of research on computer-assisted reasoning, theorem proving is still often regarded as being of limited use for practical applications to system verification. Traditionally, two classes of theorem provers have been developed: on the one hand, automatic provers for first-order logic (e.g., resolution- or tableau-based) are very efficient at solving mathematical puzzles, but they do not provide support for theories essential for verification, such as arithmetic or set theory. At the other end of the spectrum one finds interactive proof assistants based on expressive logical languages, e.g. higher-order logics or type theory, that excel at supporting "meta-proofs" such as formalizations of programming language semantics but that provide only a very limited degree of automation. By itself, neither class of provers is useful for the verification of actual systems.

We believe that the situation has changed substantially, and that the project of building a verification platform, although still challenging, is worth reconsidering. An essential enabling factor has been progress on investigating combinations of different deductive tools. These combinations aim at providing expressive modeling languages for writing specifications while offering substantial automation for

practically relevant theories, such as elementary set theory and fragments of arithmetic over the integers and the reals. Furthermore, the combinations can be implemented such that the automatic proof procedures provide evidence of their reasoning that can be verified by a small kernel of trusted code, thus giving high assurances of correctness. By significantly raising the degree of automation, these integrated verification tools enable a user to concentrate on the high-level structure of verification, carried out interactively, while leaving the bulk of the low-level reasoning to automated back-ends.

Those of us with a background in verification techniques (CRISTAL and MOSEL projects) have experience in designing combinations of interactive and automated verification tools, and with the verification of distributed systems. Those of us specializing in building applications and providing adequate programming support (OBASCO and PHOENIX projects) have used formal notations and techniques to describe and analyze the systems we build. Together, we hope to make a noticable contribution by putting the pieces together and designing a practically usable, high-assurance verification environment. We will rely on our associated partners (from Microsoft Research, DIKU Copenhague, and the CASSIS project at INRIA Lorraine) to help us with their expertise on individual work packages.

2 Work Packages

We plan to organize the overall project into individual packages that are described in the following. For each package, we also list the main participants and give an estimate of its duration.

2.1 Proof Language and TLA$^+$ Parser

Participants: CRISTAL, MOSEL, Microsoft Research.

The TLA$^+$ language was originally designed for human readability, without any concern for tools. It contains many syntactic features, such as significance of indentation or fancy lists of binders, that make it easy for a user to write specifications but that complicate parsing. TLA$^+$ contains a preliminary language for writing hierarchical proofs [5], but it will have to be extended, in particular for referring to parts of formulas and expressions within a proof. Also, a syntax for the leaf nodes of a proof must be designed that is precise enough to identify the lemmas and methods used to establish the current assertion in order to limit the search space of a proof tool.

The TLA$^+$ support tools include a parser, but it lacks functionality—in particular, it can only parse entire specifications, not individual expressions or specification modules. The existing parser, which was primarily developed for the TLC model checker, does not support the proof language of TLA$^+$. This work package focuses on fully designing a proof language and on rewriting and extending the TLA$^+$ parser.

We estimate this work package to require 6–9 months; nevertheless, initial versions of the parser will be available quickly to enable us to design and implement the proof manager concurrently.

2.2 Proof Manager

Participants: CRISTAL, MOSEL, Microsoft Research.

We foresee an architecture in which the user mainly interacts with a proof manager component that keeps track of the hierarchical structure of the proof, of the dependencies between the subproofs. In particular, support is required for users to debug and reorganize verification projects, and the proof manager should indicate which parts remain valid and which need to be reproved.

The proof manager will not by itself incorporate deep logical reasoning; it will instead rely on the interactive and automatic back-end provers to certify the coherence of a proof. It will display the proof context, such as the available assumptions and definitions, the subgoal(s) to be proven, and the constants

introduced at the current node of the proof. Nevertheless, it may turn out to be beneficial to incorporate some elementary propositional and quantifier reasoning directly in the proof manager.

A first version of the proof manager, with an ASCII interface, will be developed and tested during the first year of the project duration. During the second year, work on the proof manager will focus on the design of a usable interface, possibly as an Eclipse plug-in, and the "client" projects (OBASCO and PHOENIX) will intervene to evaluate its usability on their models.

2.3 Isabelle/TLA$^+$

Participants: MOSEL.

The interactive back-end of our verification environment will be based on the proof assistant Isabelle [8], which is based on a generic "meta logic" in which different object logics can be encoded by suitable syntax, definitions, axioms, and proof rules. We have started to encode TLA$^+$ as an object logic for Isabelle and to instantiate the generic proof methods for this object logic. It will be an ongoing task of the project to enrich and validate this encoding. We must also develop a translator from the TLA$^+$ parser's output to the Isabelle encoding, as well as a link between the proof manager and Isabelle.

A preliminary version of Isabelle/TLA$^+$ will be available at the start of the project and can already be used for experimenting with initial models of service descriptions. A translation from TLA$^+$ to Isabelle should be completed within three months of the TLA$^+$ parser (work package 2.1). Further development of the encoding and of supporting proof methods will be an ongoing task that will benefit from input from the other work packages of this project.

2.4 Zenon Prover

Participants: CRISTAL.

Zenon is an automatic theorem prover based on the tableaux method. While the best results for proving theorems of predicate logic are currently achieved by resolution-based provers, tableaux have two important advantages. First, a tableaux-based prover can output a formal proof of the theorem instead of just a yes/no answer. Second, the tableaux method is easy to extend in order to handle non-standard logic rules and domain-specific theories.

Zenon already has proof output in the form of Coq scripts and Coq lambda-terms [10]. For this project we will need to add an Isabelle back-end to Zenon, which will output proofs for Isabelle/TLA$^+$. Given the similarities between Coq and Isabelle, we expect to finish this for the first version of Isabelle/TLA$^+$ very early in the project.

We will also need to add some reasoning rules to Zenon in order to handle the non-standard aspects of the TLA$^+$ logic. Then we will concentrate on extending Zenon with more rules to handle useful theories suited to specifications written in TLA$^+$, such as set theory, arithmetic, and the operators of TLA$^+$ (most particularly, array and function operators).

The logic part of this work should be done within one year of the start of the project. Adding useful theories is an open-ended part of the project, depending on the number and complexity of the theories, but we should at least get a good start on set theory and arithmetic within one or two years.

2.5 Integration of Automatic Back-Ends

Participants: CASSIS, CRISTAL, MOSEL.

The integration of automatic deductive tools is essential to make the environment usable in practice by relieving the user of low-level, routine reasoning. VeriTLA$^+$ will include at least two automatic proof components: the tableau prover Zenon (see work package 2.4) for predicate logic, and the SMT (Satisfiability Modulo Theories) solver haRVey [3] for reasoning within fixed theories such as equality

reasoning and arithmetic. In order to obtain a trustworthy combination, the integration will rely on a certification of the justifications provided by the back-ends. The back-ends can be used directly from the proof manager or as part of broader proof methods in Isabelle/TLA$^+$.

Zenon can already generate proof terms for Coq, and the implementation of an analogous module for Isabelle/TLA$^+$ should be straight-forward. Proof certification will be extended gradually to encompass the theories Zenon can handle (set theory, functions, arithmetic, etc.).

Within the QSL project at LORIA, members of MOSEL have been working on making haRVey proof-producing and on reconstructing Isabelle proofs from the proof traces that haRVey can generate [4]. This work will be extended (with occasional help by members of the CASSIS project who develop haRVey) to cover more expressive theories, including fragments of arithmetic and set theory. The recent adoption of the SMT-LIB input language [9] will allow us to experiment with other solvers of this kind.

Experimental versions of these combinations will be available within the first six months of the project. They will help us to fine-tune the design the overall verification environment.

2.6 Formalization of Bossa and SPL in TLA$^+$

Participants: all project partners.

Based on their existing operational semantics [7, 2], the domain-specific languages Bossa and SPL targeted in this project will be formalized in TLA$^+$. We will aim at preserving the structure of the DSLs in order to make the formalization accessible to domain experts. For each language construct, corresponding lemmas will be proven that aid in the verification of individual service constructions.

This work package will mainly be carried out during the first year of the project duration and will therefore not yet benefit from full support by the verification environment. It is a prerequisite for the subsequent stage of writing compilers from DSLs to TLA$^+$ (see section 2.7 below). It will also exercise the encoding of TLA$^+$ in Isabelle (section 2.3) and guide the further development of the automatic back-end tools (sections 2.4 and 2.5). In this way, this work package establishes the precise requirements on the verification environment from the users' point of view. We will use the resulting formalizations to reprove known properties of programs written in Bossa and SPL [1].

2.7 Compilation of DSLs to TLA$^+$

Participants: OBASCO, PHOENIX, DIKU.

Based on the formalizations of DSLs in TLA$^+$ described in work package 2.6, compilation schemes from the languages considered in this project to TLA$^+$ will be defined and implemented. In this way, ad-hoc encodings can be avoided and the degree of automation will be raised. From a verification point of view, the compilation process will be supported by formally verified lemmas that can be instantiated to prove correctness properties of services described in these DSLs.

This work package will take the better part of the second project year. It should be carried out by the experts in designing the DSLs, with the teams specializing in verification intervening as consultants.

3 Partners

We give some more detailed information on the project teams that participate in this project proposal, as well as on persons that have agreed to be associated partners of this project. We list the principal members of each team that will intervene in VeriTLA$^+$. Besides the existing staffing, we apply for the funding of two post-doctoral researchers (one year each) who will work for 6 months with either CRISTAL or MOSEL on the development of the verification environment and then with either OBASCO or PHOENIX for application and transfer.

3.1 Project MOSEL

The MOSEL project at INRIA Lorraine studies formal methods for the design and analysis of software-intensive systems. Their main focus is on concepts and applications of refinement for system design. MOSEL will act as the coordinating partner of VeriTLA$^+$; the following members of MOSEL will participate in this project:

- Stephan Merz is senior researcher at INRIA Lorraine and coordinates this proposal. He has been working on the theory and tool support for TLA and is developing an encoding of TLA$^+$ in the proof assistant Isabelle. He is also involved in proof certification for the SMT solver haRVey.
- Pascal Fontaine is ATER at Université Nancy 2. He has worked on automatic theorem proving methods for program verification and is one of the principal implementors of the SMT solver haRVey, which is based on a combination of decision procedures.
- Dominique Méry is professor at Université Henri Poincaré and the head of the MOSEL project. He has had longstanding experience in working with TLA$^+$ and with the use of refinement in system design.
- Leonor Prensa Nieto is assistant professor at ENSEM. She has extensively worked with the proof assistant Isabelle and has contributed to the integration of Isabelle and haRVey.

3.2 Project CRISTAL

The CRISTAL project at INRIA Rocquencourt investigates the design, implementation, and theoretical foundations of strongly-typed programming languages. Its members are also interested in verification techniques. The principal participant from CRISTAL in this proposal is Damien Doligez, INRIA research scientist, who oversees the development of the automatic theorem prover Zenon and is responsible for its instantiation for use with TLA$^+$.

3.3 Project OBASCO

The OBASCO project at IRISA and Ecole des Mines de Nantes investigates methods for adapting software to its uses by developing tools for building software architectures based on components and aspects. The main participant from OBASCO in this project proposal is Gilles Muller, Professor at Ecole des Mines in Nantes, who has developed domain-specific languages for kernels of operating systems. In particular, he is the driving force behind the development of Bossa.

3.4 Project PHOENIX

The PHOENIX team at INRIA Futurs and the LaBRI at Bordeaux develops principles, techniques and tools for the creation of multimedia communication services, based on domain-specific languages and a study of the layers underlying communication services to improve flexibility and performance. The following permanent members of PHOENIX will participate in this project:

- Charles Consel, professor at ENSEIRB and head of PHOENIX, and
- Laurent Réveillère, assistant professor at ENSEIRB.

Both are actively involved in the design and implementation of DSLs and compilation technology for specific applications.

3.5 Associate Partners

The following persons will be associate partners of VeriTLA$^+$:

- Julia Lawall at DIKU Copenhague has been a long-term associate of the OBASCO and PHOENIX teams and has worked on the application of analysis and compilation techniques to domain-specific languages.
- Silvio Ranise is a research scientist of the CASSIS project at INRIA Lorraine and is co-developing the SMT solver haRVey.
- Leslie Lamport and Yuan Yu work at the Silicon Valley laboratory of Microsoft Research. They are, respectively, the designer of TLA$^+$ and the chief architect of the TLA$^+$ tool environment.

3.6 Previous Cooperations

Damien Doligez and Stephan Merz have worked with Leslie Lamport and Yuan Yu for many years on the logic TLA and on tool support for TLA$^+$. They are currently preparing a proposal within the INRIA-MSR partnership, which complements the Cooperative Research Initiative proposed here: whereas the design of proof support tools for TLA$^+$ is at the center of both projects, the application to DSLs is specific to this proposal.

Pascal Fontaine and Stephan Merz have cooperated with Silvio Ranise in the context of the QSL project at LORIA, in particular towards the development of haRVey and its integration with Isabelle.

The MOSEL, OBASCO, and PHOENIX team, as well as Julia Lawall, have worked together within the ACI Sécurité Informatique CORSS, coordinated by Mamoun Filali at Toulouse (IRIT). The CORSS project has helped to identify the subject of the current proposal, but the application of TLA$^+$ and of a formal verification environment has not been within the scope of CORSS.

References

[1] Jean-Paul Bodeveix, Mamoun Filali, Julia Lawall, and Gilles Muller. Formal methods meet domain specific languages. In *Proc. 5th Intl. Conf. Integrated Formal Methods (IFM 2005)*, volume 3771 of *Lecture Notes in Computer Science*, pages 187–206, Eindhoven, The Netherlands, 2005. Springer-Verlag.

[2] Charles Consel, Fabien Latry, Laurent Réveillère, and Pierre Cointe. A generative programming approach to developing DSL compilers. In R. Glück and M. Lowry, editors, *Proc. 5th Intl. Conf. Generative Programming and Component Engineering (GPCE'05)*, volume 3676 of *Lecture Notes in Computer Science*, pages 29–46, Tallinn, Estonia, 2005. Springer-Verlag.

[3] D. Déharbe and S. Ranise. Light-weight theorem proving for debugging and verifying units of code. In *Software Engineering and Formal Methods (SEFM 2003)*, pages 220–228. IEEE Press, 2003. See also `http://www.loria.fr/equipes/cassis/softwares/haRVey/`.

[4] Pascal Fontaine, Jean-Yves Marion, Leonor Prensa Nieto, and Alwen Tiu. Expressiveness + automation + soundness: Towards combining SMT solvers and interactive proof assistants. In *12th Intl. Conf. Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2006)*, 2006. Submitted.

[5] Leslie Lamport. How to write a proof. *American Mathematical Monthly*, 102(7):600–608, 1995.

[6] Leslie Lamport. *Specifying Systems*. Addison-Wesley, Boston, Mass., 2002.

[7] Julia L. Lawall, Anne-Françoise Le Meur, and Gilles Muller. On designing a target-independent DSL for safe OS process-scheduling components. In *Proc. 3rd Intl. Conf. Generative Programming and Component Engineering (GPCE'04)*, volume 3286 of *Lecture Notes in Computer Science*, pages 436–455, Vancouver, Canada, 2004. Springer-Verlag.

[8] Tobias Nipkow, Lawrence Paulson, and Markus Wenzel. *Isabelle/HOL. A Proof Assistant for Higher-Order Logic*. Number 2283 in Lecture Notes in Computer Science. Springer-Verlag, 2002.

[9] Silvio Ranise and Cesare Tinelli. The SMT-LIB standard : Version 1.1, March 2005. See also `http://goedel.cs.uiowa.edu/smtlib/`.

[10] Coq Development Team. *The Coq Proof Assistant. Reference Manual*. INRIA, Project Logical, 8.0 edition, 2004. See also `http://coq.inria.fr/`.