Available online at www.sciencedirect.com

SciVerse ScienceDirect

Electronic Notes in Theoretical Computer Science 285 (2012) 115–119

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Integrating Systems around the User: Combining Isabelle, Maple, and QEPCAD in the Prover's Palette

Laura I. Meikle[1] Jacques D. Fleuriot[2]

CISA, School of Informatics
University of Edinburgh
UK

Abstract

We describe the Prover's Palette, a general, modular architecture for combining tools for formal verification, with the key differentiator that the integration emphasises the role of the user. A concrete implementation combining the theorem prover Isabelle with the computer algebra systems Maple and QEPCAD-B is then presented. This illustrates that the design principles of the Prover's Palette simplify tool integrations while enhancing the power and usability of theorem provers.

Keywords: theorem proving, computer algebra, UI, Isabelle, Eclipse, QEPCAD, Maple.

1 Introducing the Prover's Palette

Interactive theorem proving today permits the verification of sophisticated theorems and complex algorithms, although the process remains cumbersome and time-consuming. We believe that in many instances, access to other tools such as Computer Algebra Systems (CAS) can accelerate this process. Whilst there have been significant advances in combining provers with external tools, as described in previous work [6], the increased complexity of current proof developments places new demands and challenges on tool integrations.

We begin with the premise that an integration's primary goal is to accelerate the proof development process. To date, this has been achieved primarily by integrations which can automatically simplify expressions and discharge subgoals [8][11]. With more complicated verification tasks—and with more mathematicians using

[1] Email: lauram@dai.ed.ac.uk
[2] Email: jdf@inf.ed.ac.uk

1571-0661
doi:10.1016/j.entcs.2012.06.010

provers—we believe that the process of formal proof can also benefit from tool integrations which are able to enhance a user's understanding of a problem. To achieve this, it is important that integrations support multiple modes of interaction: the user should be given the option to use external systems in a fully automated way, but they should also have the option to modify how that system is being used. A novice user can then benefit from a variety of proof assistants, even if he is not familiar with them, while the expert user is not hindered from accessing the full richness of these tools. This can be essential for some problems, as even the best fully automated integrations cannot always tune the parameters appropriately. Furthermore, by allowing visibility and modifiability to more aspects of the tool integration, the user is given greater flexibility in exploring the problem. We believe that a semi-interactive integration framework now has a vital role to play: the framework should support automatically configuring settings appropriate to a specific problem but also emphasise usability in exploring and changing these values, all while maintaining consistency between systems.

This philosophy has led us to develop the Prover's Palette, a user-centric approach to tool integrations. A concrete implementation that combines Isabelle and QEPCAD-B, assisting the user in reasoning about real nonlinear polynomials, has already been achieved [6]. In this paper, we apply our approach to a further systems integration in order to show that the design principles apply more generally and that our architecture can be easily extended to support more tools. In particular, we present an integration of Maple [10] into the Prover's Palette and show how its plotting capabilities can provide significant insight into theorems being proven. Our framework is available for download at `www.cognetics.org/proverspalette`.

2 System Design

The key idea of our approach is to unify multiple tools through a cohesive user interface (UI), allowing any relevant tool to be used at any point in the proof process. This is accomplished by constructing a UI "View" widget for each external tool. These Views sit alongside the proof script in the IDE, and:

- When the proof state changes, each View is notified by the framework and is updated appropriately, automatically acquiring a translation in its native language and a choice of appropriate default settings.
- Each View can be configured to run in "novice" mode, using sensibly-chosen default settings and running with a single user-click (some can be further configured to run automatically, appearing only when they yield a result).
- By exposing an optional sequence of "tabs", each View also allows full inspectability and customisation of the translation and settings.
- Each View presents the results of the external system's computation in an integrated form, commonly a command in the prover's language which can then be automatically inserted into the proof script.

We have implemented this design in Java, making each View a plug-in to the Eclipse Proof General Kit [1], leveraging its broker structure, and targeting the Isabelle theorem prover [9]. The Prover's Palette framework also uses libraries from the Feature Wizard [5] to support translation and the PG `ProofScriptEditor` API for post-processing insertion into the proof script [3].

The implementation supplies abstract superclasses which encapsulate common functionality, including defining the View and tabs, tracking and translating the proof state, modifying the proof state where necessary (e.g. expanding definitions, converting to PNF), and inserting results in various ways (via an oracle, by instantiation, etc). This minimises the additional code required to integrate a new tool with the Prover's Palette. We demonstrate this next by describing our implementation which adds Maple support to our system.

3 Using Maple in the Prover's Palette

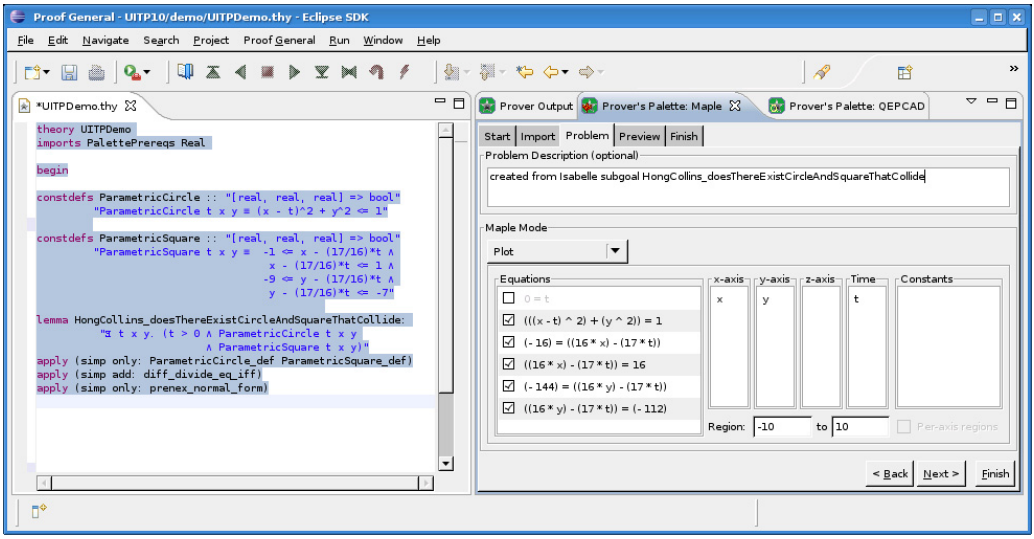

Figure 1. Reasoning with the Prover's Palette: Isabelle, QEPCAD and Maple in the IDE

As Maple is a powerful and popular CAS we selected it for integration into the Prover's Palette. We illustrate our approach—and this integration— through a real-world example making use of Maple's plotting capabilities. Figure 1 shows a screenshot of the prover IDE with the new Maple view.

The proof script in Figure 1 revisits a collision problem described by Collins and Hong [3]. This involves robot motion planning and queries whether a moving circle (1) and a moving square (2) will ever collide, specifically asking whether $\exists\, t\ x\ y.\ t \geq 0 \wedge (1) \wedge (2)$:

[3] The Prover's Palette currently inserts commands using Isabelle's "tactic" procedural mode, but support for the declarative "Isar" mode would be easy to add.

$$(1) \quad (x-t)^2 + y^2 \leq 1$$
$$(2) \quad -1 \leq x - \tfrac{17}{16}t \leq 1 \wedge -9 \leq y - \tfrac{17}{16}t \leq -7$$

Relying solely on a theorem prover, this can be a difficult proof, with the user having to contend with questions ranging from how the myriad lemmas should be applied through to whether the objects are defined correctly and whether the theorem is provable. Maple can give reassurances with regard to the latter two. With the Prover's Palette, the Maple View continually updates to reflect the current proof goal state: the user can then, for example, choose to plot these equations (on the "Problem" tab of the Maple View, as shown in Figure 1). The integration automatically converts the problem—stripping away quantifiers, breaking up conjuncts, and converting inequalities—then determines which equations are suitable for inclusion and which plot type and command is appropriate (e.g. `ImplicitPlot`, `ImplicitPlot3D`, `animate`) [4].

With the collision problem, the Maple integration defaults to a 2D animation. It has automatically determined the variables for each axis and deduced that five out of the six constituent equations should be plotted. In this plot configuration, $t = 0$ does not make sense; however the user is free to change the variables-to-axes mappings, and if t and x are swapped — so that t is plotted along the x-axis and we animate over time x — then the list of enabled equations is updated to include $t = 0$, although the plots may be unintuitive!

The user can then select the "Finish" button to send the plot command to Maple. Alternatively the user can go to the "Preview" tab to inspect the script to be sent, making edits where desired: applying colours or labels, or even adding additional equations. This tab also allows cancelling the external process at any time. Asking Maple to plot the collision problem results in a new window containing an interactive animation of the circle and square moving over time. Figure 2 shows a still of this animation where the circle and square are colliding, confirming that the problem is provable.

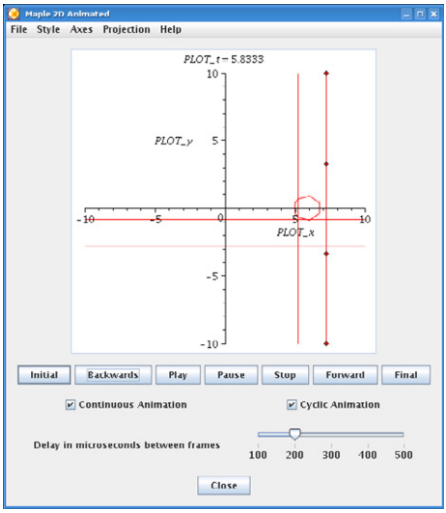

Figure 2. Maple Animated Plot

Following the visual insight afforded by Maple, the user could also invoke QEPCAD within the Prover's Palette environment. Previously we showed how QEPCAD can be used to confirm that the problem is "True" [6]. It can also generate witnesses ($t{=}\frac{96}{17}$, $x{=}\frac{96}{17}$ and $y = -1$) which the integration can instantiate into the proof with a single click.

[4] Due to space limitations, the automatic conversion and configuration to make problems suitable for use with Maple is not covered here. The interested reader is referred to [7].

4 Conclusion

By integrating Maple into the Prover's Palette we have shown that the framework can generalise to support other external tools. It would be interesting to further extend the system to support other theorem provers that can be made compatible with PG Kit; and with the advent of new Java-based prover IDEs such as I3P [4], we believe our system has promise for even wider integration. We have also shown that an interactive integration framework is capable of enhancing the user's understanding of proof problems. This supports our hypothesis that the user is most empowered when a palette of tightly integrated—yet customisable—tools is conveniently at their disposal. A successful integration framework enables a deeper understanding of proof obligations, freeing the user from the burden of tedious, mundane proof details, but without overly restricting what is possible.

Acknowledgement

We would like to thank the reviewers for their useful comments. This work was funded by the EPSRC grant EP/E005713/1.

References

[1] Aspinall D., C. Lüth, and D. Winterstein, *A Framework for Interactive Proof*. Towards Mechanized Mathematical Assistants, Springer LNAI 4573 (2007), 161-175.

[2] Brown C. W., *QEPCAD B: a program for computing with semi-algebraic sets using CADs*, SIGSAM Bulletin, **37** (2003), 97-108.

[3] Collins G. E., and H. Hong, *Partial Cylindrical Algebraic Decomposition for Quantifier Elimination*, Journal of Symbolic Computation **12** (1991), 299-328.

[4] Gast H., I3P, www-pu.informatik.uni-tuebingen.de/i3p

[5] Heneveld A., *Using Features for Automated Problem Solving*, PhD Thesis, University of Edinburgh, 2007.

[6] Meikle L. I., and J. D. Fleuriot, *Combining Isabelle and QEPCAD-B in the Prover's Palette*, AISC/MKM/Calculemus (2008), 315-330.

[7] Meikle L. I., *The Formal Verification of Geometric Algorithms*, to appear as PhD Thesis, University of Edinburgh.

[8] Meng J., C. Quigley, and L. C. Paulson, *Automation for interactive proof: first prototype*, Inf. Comput., **204: 10** (2006), 1575-1596.

[9] Paulson L. C., *Isabelle: A Generic Theorem Prover*, LNCS **828** (1994).

[10] Redfern M., and D. Betounes, "Mathematical Computing: An Introduction to Programming Using Maple", Telos Press, 2002.

[11] Tiwari A., PVS-QEPCAD, www.csl.sri.com/users/tiwari/qepcad.html.