From 80% to 99%
An Industrial Collaboration for Automating Frama-C/WP

L. Correnson
From 80% to 99%
An Industrial Collaboration for Automating Frama-C/WP
(nearly)

L. Correnson
From 80% to 99% (nearly)
An Industrial Collaboration for Automating Frama-C/WP

L. Correnson

J. Souyris

FRAMA-C & SPARK DAY 2017
ONCE UPON A TIME

Caveat
... FROM UNIT TESTS TO UNIT PROOFS ...

Caveat Tool (late 2000s)

- Formal Specifications
- Automated Proof (by Rewriting)
- Interactive Proof Transformation
Caveat @ Airbus (2005)

- Replacement for Unit Tests
- Complete Behaviours as Test Plan
- Dedicated Memory Model
- Limited Aliasing & Coding Rules
- Runtime-Error-Free Hypothesis
- Qualified for DO 178 B
...WHEN FINALLY CAME

WP
A PARTNERSHIP STORY

From Caveat to NUPW

2010 — 2017
THE SITUATION (2010)

- Caveat Obsolescence
- Caveat Limitations
- Promising Frama-C
THE SITUATION (2010)

- Caveat Expertise
- Deployed Unit Proofs
- Extended with Alt-Ergo
- 96% Automation Rate
- Aliasing Limitations (Review)
OBJECTIVES

- 16 modules bench
- Automation Rate > 96%
- Caveat in-place Replacement
AVAILABLE ARTEFACTS

- Modules (C-code)
- Caveat Specifications
- Journalised Proofs

< 4%
MIGRATION PROJECT

- Modules (C-code)
- Caveat Specifications
- Journalised Proofs
- Frama-C/WP

Diagram: Flowchart showing the migration project phases.
MIGRATION PROJECT

Modules (C-code) → Volatile Instrumentation → Frama-C/WP

Caveat Specifications → Assigns Generator → ACSL Importer

Caveat to ACSL Translator

Journalised Proofs (a new hope)

AIRBUS NUPW  2010 — 2012
### 3.3.3 Le module seq.v

Ce module est dédié à la normalisation automatique des buts sur les séquences en appliquant les axiomes définis par l'axiomatique passée au greffon dans les sources du cas d'étude.

Le module définit deux tactiques:

- **seqrule**: normalise les séquences dans le but
- **callto**: réduit un but de la forme \((S, f) = (T, g)\) en \((S = T)\) et \(f = g\) puis simplifie et normalise les sous-buts résultants.

Encore une fois, ces tactiques ne sont que des cas très particuliers des étapes de recherche effectuées par Alt-Ergo de manière automatique, mais sur la base des axiomes fournis dans les sources du cas d'étude.

### 3.4 Résultats

Le WP est configuré avec les options spécifiques suivantes:

- **-wp-script cloture_proc_cav.script**: base de script de preuves édités dans Coq-IDE
- **-wp-coq-lib nup, hint, seq**: inclusion des modules externes
- **-wp-timeout 3600**: les tactiques automatiques peuvent être assez longues à calculer, en pratique, quelques minutes suffisent pour l'ensemble de l'étude
- **-wp-depth 300**: pour autoriser un horizon de recherche très profond adapté aux longues séquences dans Alt-Ergo
- **-wp-timeout 20**: on ne laisse pas Alt-Ergo chercher trop longtemps inutilement.

Les résultats obtenus sont les suivants:

<table>
<thead>
<tr>
<th>Module</th>
<th>#VC</th>
<th>WP</th>
<th>Alt-Ergo</th>
<th>Coq</th>
<th>Failed</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bench #1</td>
<td>13</td>
<td>4</td>
<td>9</td>
<td>-</td>
<td>-</td>
<td>100.0 %</td>
</tr>
<tr>
<td>Bench #2</td>
<td>54</td>
<td>11</td>
<td>13</td>
<td>30</td>
<td>-</td>
<td>100.0 %</td>
</tr>
<tr>
<td>Bench #3</td>
<td>35</td>
<td>6</td>
<td>8</td>
<td>17</td>
<td>4</td>
<td>88.6 %</td>
</tr>
<tr>
<td>Bench #4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Timeout</td>
</tr>
</tbody>
</table>

**Automation < 50%**
AIRBUS (*a bit anxious*)

« Hey, CEA, you shall do something, aren’t you? »

CEA (*embarrassed*)

« … ah, really, you don’t like Coq? »
AIRBUS (furious)

« ... »

CEA (confused)

« ... ok, ok, let’s try something! »
# INTRODUCING QED IN FRAMA-C/WP

<table>
<thead>
<tr>
<th>Module</th>
<th>#VC</th>
<th>WP</th>
<th>Alt-Ergo</th>
<th>Coq</th>
<th>Failed</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bench #1</td>
<td>11</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td>100 %</td>
</tr>
<tr>
<td>Bench #2</td>
<td>25</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td>100 %</td>
</tr>
<tr>
<td>Bench #3</td>
<td>22</td>
<td>18</td>
<td>3</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Bench #4</td>
<td>172</td>
<td>116</td>
<td>56</td>
<td></td>
<td></td>
<td>95 %</td>
</tr>
</tbody>
</table>

— 2012 —
Qed. Computing What Remains to Be Proved

Loïc Correnson
CEA, LIST, Software Safety Laboratory
PC 174, 91191 Gif-sur-Yvette France
firstname.lastname@cea.fr

Abstract. We propose a framework for manipulating in an efficient way terms and formulæ in classical logic modulo theories. Qed was initially designed for the generation of proof obligations of a weakest-precondition engine for C programs inside the Frama-C framework, but it has been implemented as an independent library. Key features of Qed include on-the-fly strong normalization with various theories and maximal sharing of terms in memory. Qed is also equipped with an extensible simplification engine. We illustrate the power of our framework by the implementation of non-trivial simplifications inside the Wp plug-in of Frama-C. These optimizations have been used to prove industrial, critical embedded softwares.

1 Introduction
In the context of formal verification of critical softwares, the recent fantastic improvement of automated theorem provers and SMT solvers[1] opens new routes. Inside the Frama-C [2] platform, we have developed the Wp plug-in to implement an efficient weakest precondition calculus to formally prove a C program against its specification. The specification is written in terms of the “ANSI-C Specification Language” (ACSL) [3], which is a first-order logic system with dedicated constructs to express C properties such as pointer validity and floating point operations.

The Wp plug-in actually compile C and ACSL constructs into an internal logic representation that is finally exported to SMT solvers and other theorem provers. Thus, we need an internal system to represent and manipulate first-order logical formulæ. This is exactly what Qed has been designed for.

Designing such a library is not difficult in itself. Some datatype is needed for expressing terms and properties, combined with pretty-printing facilities to export them into several languages. This is what we implemented in our early prototypes.

However, experimental results shown that a formula can not be naively build then translated and finally sent to an external back-end prover. We actually observed limitations of such a naive approach on real life examples from critical embedded software:

– SMT solvers are quite efficient, but they are sensitive to the amount of hypotheses they receive. Having a proof for $A \rightarrow B$ does not mean you will have a proof for $A \land A' \rightarrow B$.
AIRBUS 16-Modules Bench

<table>
<thead>
<tr>
<th>Year</th>
<th>2012</th>
</tr>
</thead>
<tbody>
<tr>
<td>%</td>
<td></td>
</tr>
<tr>
<td>93%</td>
<td>88%</td>
</tr>
<tr>
<td>88%</td>
<td>89%</td>
</tr>
<tr>
<td>73%</td>
<td>89%</td>
</tr>
<tr>
<td>96%</td>
<td>100%</td>
</tr>
<tr>
<td>92%</td>
<td>98%</td>
</tr>
<tr>
<td>68%</td>
<td>35%</td>
</tr>
<tr>
<td>79%</td>
<td>96%</td>
</tr>
<tr>
<td>92%</td>
<td>96%</td>
</tr>
<tr>
<td>64%</td>
<td></td>
</tr>
</tbody>
</table>

— 2012 —
AIRBUS *(ambitious)*

« Seems that 100% is reachable… »

CEA *(volunteer)*

« Go! we setup a task force! »
SYSTEMATIC APPROACH

Module < 100% automation

CEA Proposal / Fix

Next Module / Property
20 GitLab Issues
3 CEA Researcher-Engineers
Continuous Progression Reports

#1. (Qed) typing lets with alt-ergo
#2. (Qed) traduction des booléens
#3. (Qed) remontée des conditionnelles
#4. (WP) modèle Caveat
#5. (WP) assigns des tableaux et structures
#6. (Qed) control-flow au travers des ifs
#7. (WP) comparaison des structures
#8. (WP) global-const
#9. (NUPW) driver des séquences
#10. (NUPW) répétition dans les séquences
#11. (Qed) preuve des initialisateurs de tableaux
#12. (WP) \let dans les prédicats ACSL
#13. (WP) affaiblissement des quantificateurs sur les entiers
#14. (WP) warning sur les tableaux infinis
#15. (WP) exit behavior des simulés
#16. (Qed) conversion des décalages d’entiers
#17. (WP) références initialement valides
#18. (FRAMA-C) consolidation des benchs
#19. (WP) séparation des formelles entre les modèles Hoare et Typed
#20. (Qed) perte de partage

L. Correnson
P. Baudin
F. Bobot

NUPW — TRAVAUX 2014

Taux de Succès

<table>
<thead>
<tr>
<th>Tâche</th>
<th>Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>✔️</td>
</tr>
<tr>
<td>#2</td>
<td>✔️</td>
</tr>
<tr>
<td>#3</td>
<td>✔️</td>
</tr>
<tr>
<td>#4</td>
<td>✔️</td>
</tr>
<tr>
<td>#5</td>
<td>✔️</td>
</tr>
<tr>
<td>#6</td>
<td>✔️</td>
</tr>
<tr>
<td>#7</td>
<td>✔️</td>
</tr>
<tr>
<td>#8</td>
<td>✔️</td>
</tr>
<tr>
<td>#9</td>
<td>✔️</td>
</tr>
<tr>
<td>#10</td>
<td>✔️</td>
</tr>
<tr>
<td>#11</td>
<td>✔️</td>
</tr>
<tr>
<td>#12</td>
<td>✔️</td>
</tr>
<tr>
<td>#13</td>
<td>✔️</td>
</tr>
<tr>
<td>#14</td>
<td>✔️</td>
</tr>
<tr>
<td>#15</td>
<td>✔️</td>
</tr>
<tr>
<td>#16</td>
<td>✔️</td>
</tr>
<tr>
<td>#17</td>
<td>✔️</td>
</tr>
<tr>
<td>#18</td>
<td>✔️</td>
</tr>
<tr>
<td>#19</td>
<td>✔️</td>
</tr>
<tr>
<td>#20</td>
<td>✔️</td>
</tr>
</tbody>
</table>
### AIRBUS 16-Modules Bench

<table>
<thead>
<tr>
<th>Year</th>
<th>Module 1</th>
<th>Module 2</th>
<th>Module 3</th>
<th>Module 4</th>
<th>Module 5</th>
<th>Module 6</th>
<th>Module 7</th>
<th>Module 8</th>
<th>Module 9</th>
<th>Module 10</th>
<th>Module 11</th>
<th>Module 12</th>
<th>Module 13</th>
<th>Module 14</th>
<th>Module 15</th>
<th>Module 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>2012</td>
<td>93%</td>
<td>88%</td>
<td>88%</td>
<td>73%</td>
<td>89%</td>
<td>96%</td>
<td>100%</td>
<td>92%</td>
<td>98%</td>
<td>68%</td>
<td>35%</td>
<td>79%</td>
<td>96%</td>
<td>92%</td>
<td>92%</td>
<td>64%</td>
</tr>
<tr>
<td>2014</td>
<td>100%</td>
<td>99%</td>
<td>99%</td>
<td>100%</td>
<td>95%</td>
<td>100%</td>
<td>100%</td>
<td>97%</td>
<td>96%</td>
<td>81%</td>
<td>95%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>95%</td>
<td>85%</td>
</tr>
</tbody>
</table>

2012 — 2014
AIRBUS

« We need to deploy! »
« For Caveat migration and for new projects! »

CEA (a bit nervous)

« Now? »
CEA (apart)

« we need an organisation »
GITLAB TO RESCUE

CEA Support for NUPW

Issue Tracker Usage:
• Please submit one separate issue per question
• Use assignment to identify who has something to do on each issue
• An issue shall be closed by its author only

NUPW Distributions

• Release Board
• Distrib Site
MIXING « R » & « D » VIA GIT

Research Experiment(s) → Dedicated Development(s)

master → nupw
MIXING « R » & « D » VIA GIT

Platform Upgrades

nupw

master
MIXING « R » & « D » VIA GIT

Bench Validation
MIXING « R » & « D » VIA GIT

Public Release

master

nupw
MIXING « R » & « D » VIA GIT

Backport — « kernel » part
MIXING « R » & « D » VIA GIT

Backport — per « Plugin » parts
MIXING « R » & « D » VIA GIT

Improved Back-Ports

master

nupw
MIXING « R » & « D » VIA GIT

Public Release(s)  master  nupw  Partner Release(s)
THE TECHNICAL PARTS

☑ Qed as an intermediate representation
☑ Inlined normalisation to avoid combinatorial explosion
☑ Maximal in-memory sharing preservation
☑ Aggressive constant & equality propagation
☑ Integer modulo computations
☑ Integer bitwise simplifications
☑ Loop unrolling heuristics
☑ Hypothesis compaction & erasing
☑ Hypothesis generalisation (e.g. init parts)
☑ Systematic branch pruning
☑ Type synthesis for Alt-Ergo
☑ Small bugs in Alt-Ergo (few, indeed)
☑ External drivers (closer to SMT)
☑ Dedicated simplifications (beyond SMT)
☑ Proof obligation understanding & retro-engineering
☑ User-defined simplifications
☑ Elicit Caveat Aliasing (no more manual review)
☑ Kernel-aware runtime errors (overflows & downcasts)

…
<table>
<thead>
<tr>
<th>Year</th>
<th>2012</th>
<th>2014</th>
<th>2016 (auto / tip)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>93%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>88%</td>
<td>99%</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>88%</td>
<td>99%</td>
<td>90 / 99%</td>
</tr>
<tr>
<td></td>
<td>73%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>89%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>96%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>92%</td>
<td>97%</td>
<td>99 / 100%</td>
</tr>
<tr>
<td></td>
<td>98%</td>
<td>96%</td>
<td>97 / 100%</td>
</tr>
<tr>
<td></td>
<td>68%</td>
<td>81%</td>
<td>95 / 100%</td>
</tr>
<tr>
<td></td>
<td>35%</td>
<td>95%</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>79%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>96%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>92%</td>
<td>95%</td>
<td>96 / 100%</td>
</tr>
<tr>
<td></td>
<td>64%</td>
<td>85%</td>
<td>94 / 95%</td>
</tr>
</tbody>
</table>

2012 — 2017
THE LAST 1% TO BE PROVED

Frama-C/WP
Interactive Proof Transformer

Demo

— 2017 —
THE LAST 1% TO BE PROVED

Frama-C/WP
Interactive Proof Transformer

- Goal exploration / view
- Memory model interpretation
- User-defined Tactics
- User-defined Heuristics
- Replay Scripts
- Code & Spec reconciliation

In Frama-C Phosphorus
2008
A 10 year investment in static code analysis, the Caveat tool from CEA is used in production to validate safety-critical code in the A380 program, and a few years later on the A350 and A400M

2011
Obsolescence management triggers the investigation of tooling renewal, and the identification of compatibility and performance challenges in proposed solutions

WP

2012
Teams at CEA List complete the development of the Frama-C/WP plugin, complete with migration helpers: together they form the new unit proof workshop NUPW

2014
Efficient reasoning techniques dramatically boost the level of proof automation, and bring NUPW to performance-parity with legacy tooling

2016
The Frama-C/WP plugin is extended to provide advanced interactive features that support the proof engineering phases, while Airbus and List teams setup regression baselines and training courses

2017
An enhanced support contract accompanies the deployment of NUPW to operational teams

Airbus engineers complete the design of a formal language for Low-Level Requirements, setting the stage for the design of innovative static and dynamic verifications
AIRBUS & CEA (together)

« That’s an R&D partnership! »