

DE LA RECHERCHE À L'INDUSTRIE

cea



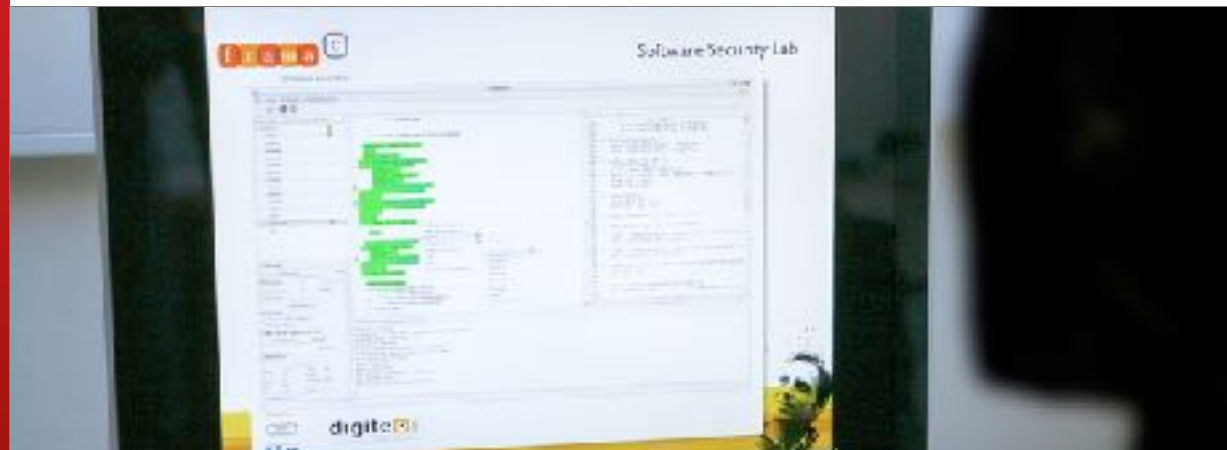
list

www.cea.fr

From 80% to 99%

An Industrial Collaboration for Automating Frama-C/WP

L. Correnson



FRAMA-C & SPARK DAY 2017

DE LA RECHERCHE À L'INDUSTRIE

cea



list

www.cea.fr

From ~~80%~~^{50%} to ~~99%~~^{100%}

An Industrial Collaboration for Automating
Frama-C/WP

(nearly)

L. Correnson



FRAMA-C & SPARK DAY 2017

DE LA RECHERCHE À L'INDUSTRIE



From ~~80%~~^{50%} to ~~99%~~^{100%}

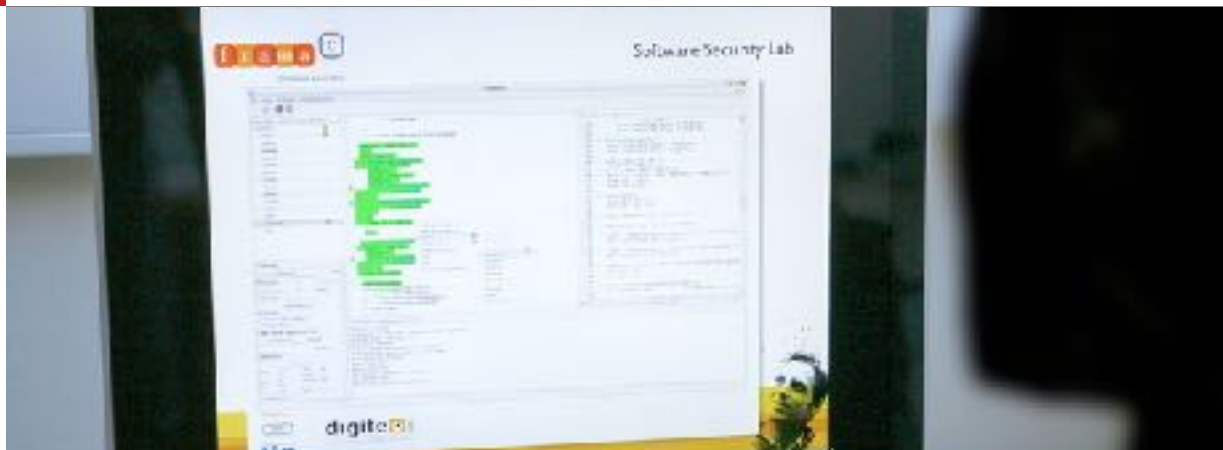
An Industrial Collaboration for Automating
Frama-C/WP

(nearly)

L. Correnson



J. Souyris



list

www.cea.fr

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

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

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

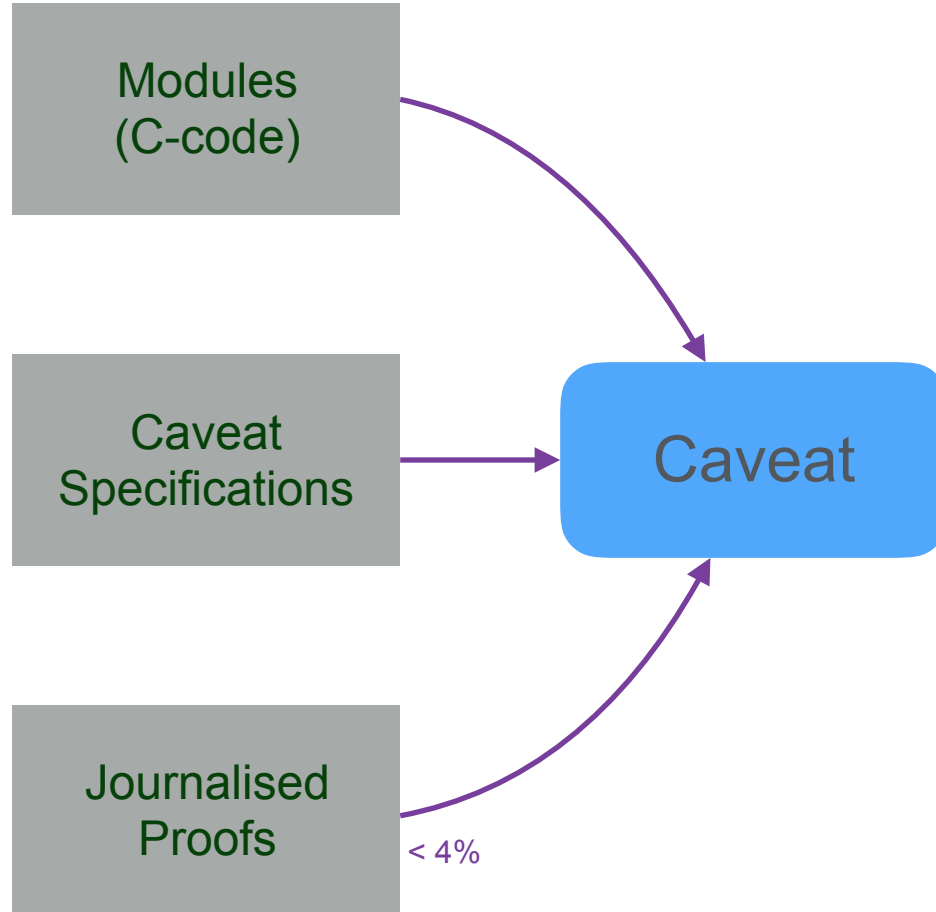
THE SITUATION (2010)

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

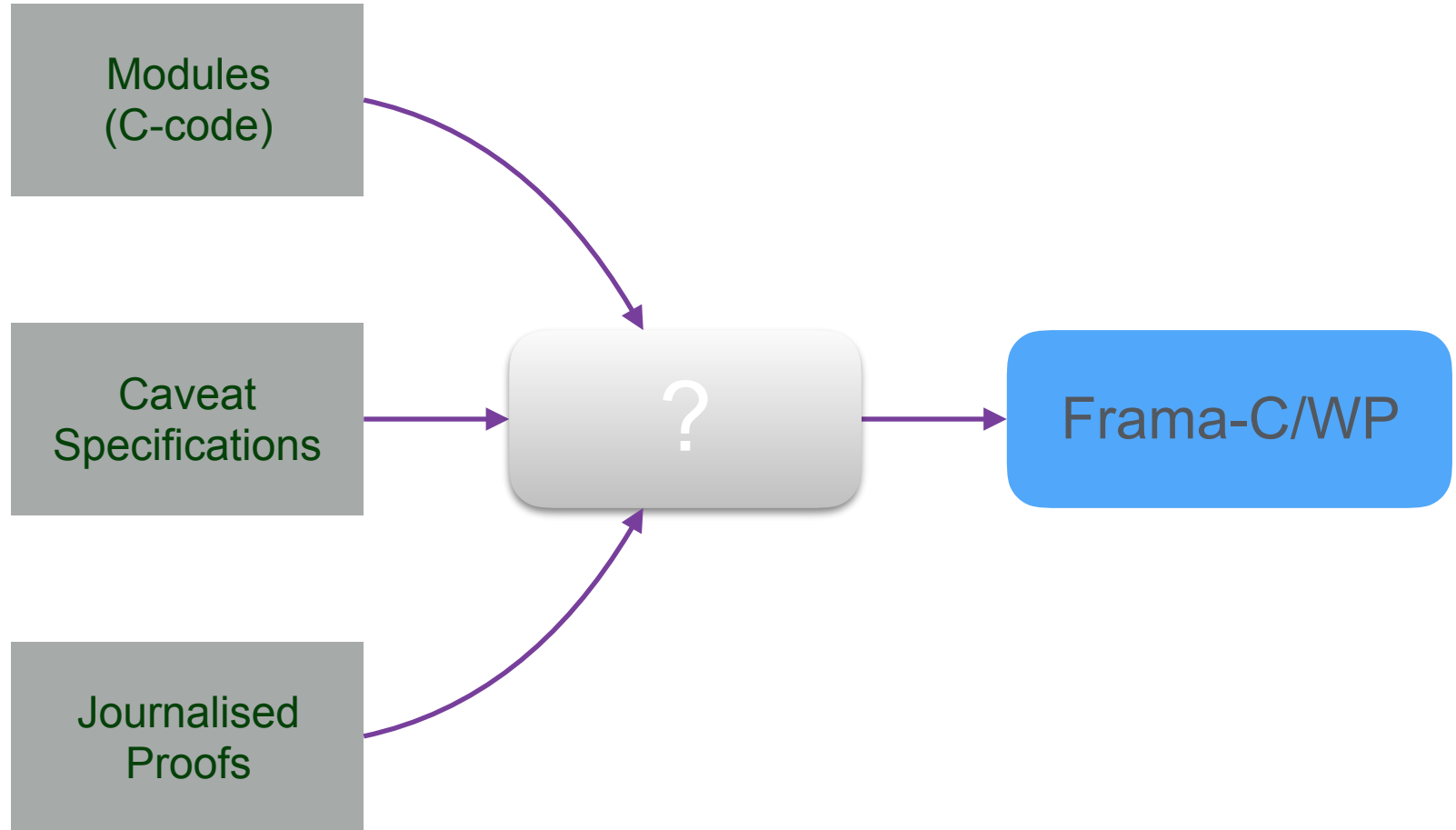
OBJECTIVES

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

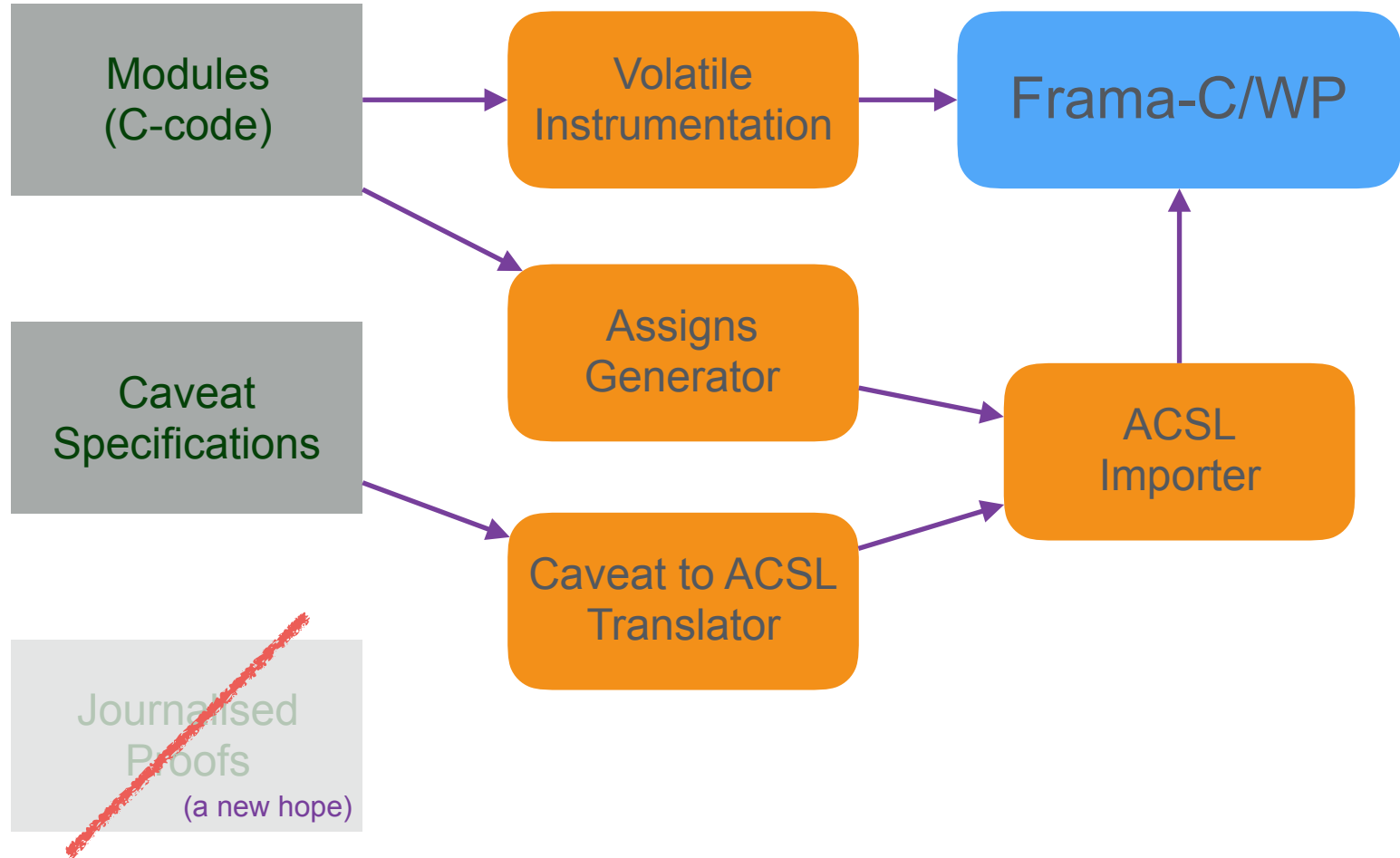
AVAILABLE ARTEFACTS



MIGRATION PROJECT



MIGRATION PROJECT



THE GAP !

Module	#VC	WP	Alt-Ergo	Coq	Failed	Success
Bench #1	13	4	9	-	-	100.0 %
Bench #2	54	11	13	30	-	100.0 %
Bench #3	35	6	8	17	4	88.6 %
Bench #4	Timeout					

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 (*furios*)

« ... »

CEA (*confused*)

« ... ok, ok, let's try something! »

INTRODUCING QED IN FRAMA-C/WP

Module	#VC	WP	Alt-Ergo	Coq	Failed	Success
Bench #1	11	11				100 %
Bench #2	25	25				100 %
Bench #3	22	18	3		1	95 %
Bench #4	172	116	56			100 %

INTRODUCING QED IN FRAMA-C/WP

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



(Published 2014)

Abstract. We propose a framework for manipulating in an efficient way terms and formulae 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.

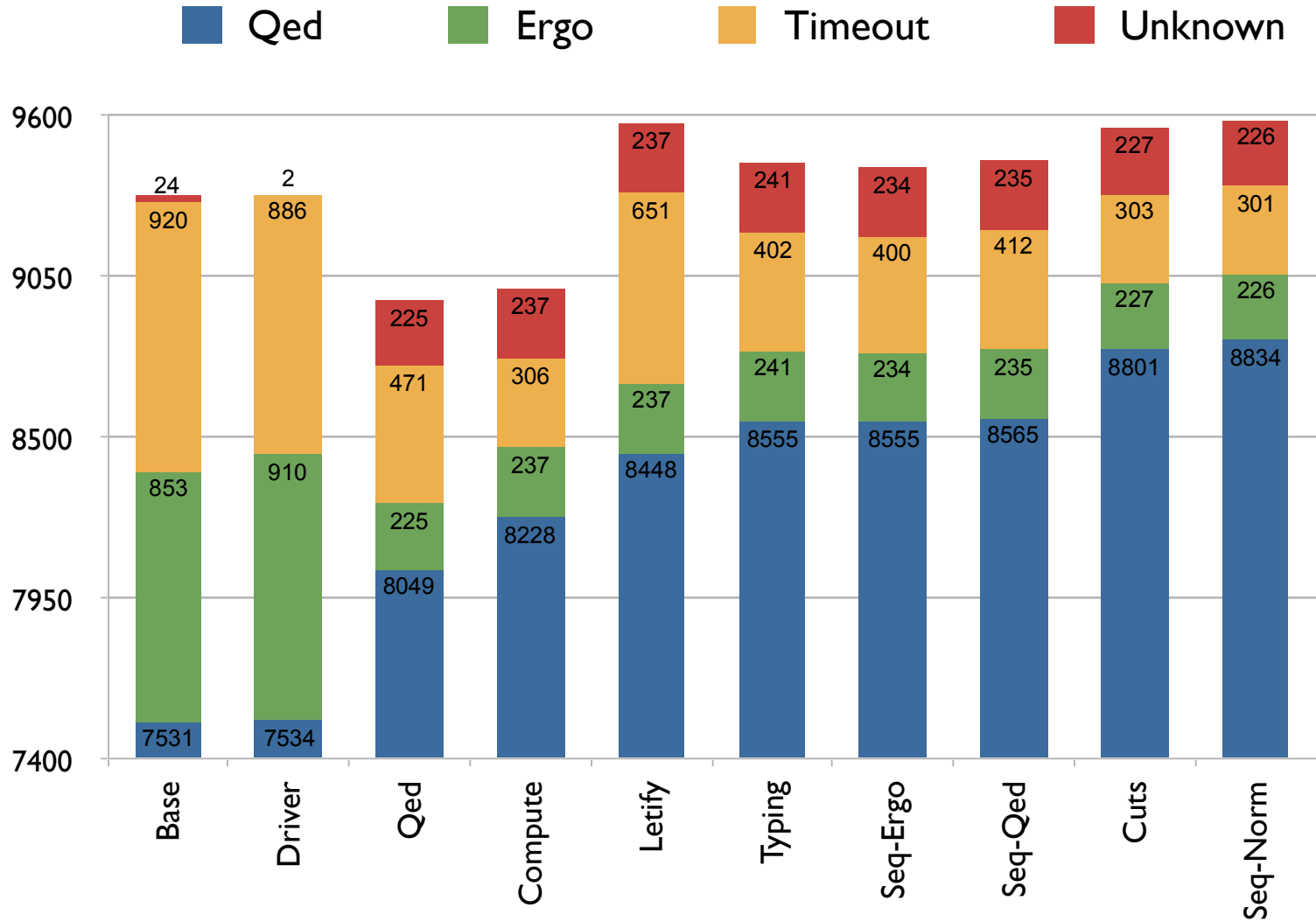
The **Wp** plug-in actually compile C and ACSL constructs into an internal logic representation that is finally exported to SMT solvers and other theorem provers. Thus, we need an internal system to represent and manipulate first-order logical formulae. This is exactly what **Qed** has been designed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

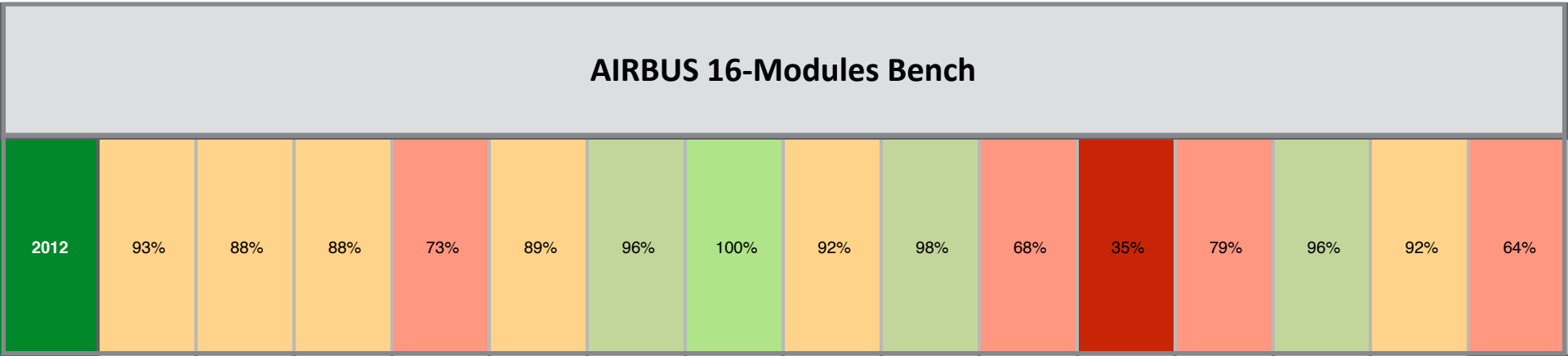
MATURING QED & WP



2011 — 2012

NUPW AUTOMATION

AIRBUS 16-Modules Bench



— 2012 —

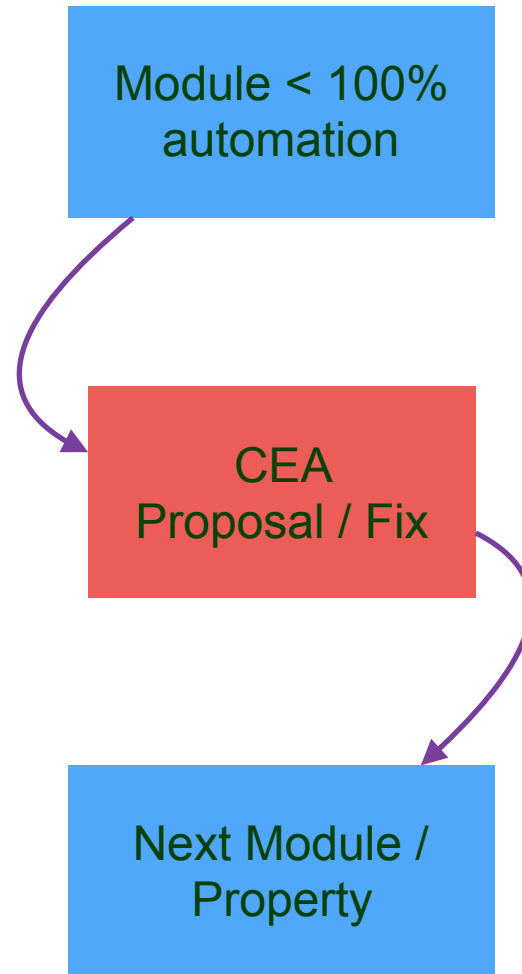
AIRBUS (*ambitious*)

« Seems that 100% is reachable... »

CEA (*volunteer*)

« Go! we setup a task force! »

SYSTEMATIC APPROACH



20 GitLab Issues

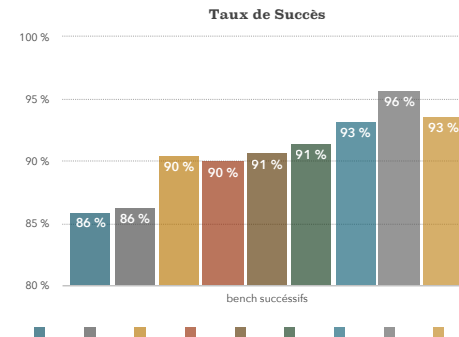
3 CEA Research-Engineers

Continuous Progression Reports

L. Correnson
P. Baudin
F. Bobot

- #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 initialiseurs 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 benches
- #19. (WP) séparation des formelles entre les modèles Hoare et Typed
- #20. (Qed) perte de partage

NUPW – TRAVAUX 2014



#	Tâche	Closed	#	Tâche	Closed
#1	Typage pour Alt-Ergo	✓	#21	Initialiseurs Compacts	✓
#2	Egalité booléenne	✓	#26	Typage des tableaux pour Alt-Ergo	✓
#3	Egalité conditionnelle	✓	#27	Introduction des existentielles	✓
#4	Modèle Caveat	✓	#28	Filtrage agressif des hypothèses	✓
#5	Assigns de structures	☐	#31	Pré-requis des simulés inappropriés	✓
#6	Flots d'égalités dans les branches	✓	#32	Simplification des expr. constantes	☐
#7	Egalité de structures	☐			☐
#8	Variable globales constantes	✓			☐
#9	Driver des séquences	✓			☐
#10	Répétition de séquences	✓			☐
#11	Initialisation des grands tableaux	✓			☐
#12	ACSL let-in de propriétés	☐			☐
#13	Contraintes des entiers C	✓			☐
#14	Bug sur tableaux infinis	✓			☐
#15	Exit behavior (related to global const)	✓			☐
#16	Conversions bits-à-bits	✓			☐
#17	Références initialement valides	✓			☐
#18	Export des résultats de WP	☐			☐
#19	Séparation entre Hoare et Typed	✓			☐
#20	Perte de partage en sortie de Qed	✓			☐

NUPW AUTOMATION

AIRBUS 16-Modules Bench

2012	93%	88%	88%	73%	89%	96%	100%	92%	98%	68%	35%	79%	96%	92%	64%
2014	100%	99%	99%	100%	95%	100%	100%	97%	96%	81%	95%	100%	100%	95%	85%

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 »

— 2017 —

GITLAB TO RESCUE

The screenshot shows a web browser window displaying the GitLab interface for a project named 'Support'. The browser's address bar shows 'git.frama-c.com'. The navigation menu includes 'Project', 'Activity', 'Repository', 'Graphs', 'Issues 15', and 'Wiki'. The project's logo is a circular icon with blue and white waves. Below the logo, the title 'Support' is followed by the subtitle 'Issue Tracking System for NUPW'. The interface includes buttons for 'Star', 'Fork', 'SSH', and 'Leave project'. A commit history bar shows a recent commit by '42ea891b' titled 'Update README.md'. At the bottom, there are buttons for 'Add Changelog', 'Add License', and 'Set up CI'.

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](#)

GITLAB TO RESCUE


The screenshot shows a web browser window displaying the GitLab interface for a project named 'Distribution'. The browser's address bar shows 'git.frama-c.com'. The navigation bar includes a sidebar with 'AIRBUS / Distribution', a search bar, and a user profile. The main content area features the project logo, name 'Distribution', and subtitle 'NUPW Distribution'. Below this are buttons for 'Star', 'Fork', 'SSH', and 'Leave project'. A section for 'Files (37 MB)', 'Commits (183)', 'Branches (2)', and 'Tags (14)' is visible, along with buttons for 'Add Changelog', 'Add License', 'Add Contribution guide', and 'Set up CI'. The current commit is identified as 'a225b6e4 Release NUPW v4.04'. The left sidebar contains sections for 'NUPW Distribution', 'NUPW Support', and 'Misc', each with a list of links.

git.frama-c.com

AIRBUS / Distribution

This project Search

Project Activity Repository Graphs Issues 0 Merge Requests 1

 Distribution
NUPW Distribution

Star 0 Fork 0 SSH + Leave project

Files (37 MB) Commits (183) Branches (2) Tags (14) Add Changelog Add License Add Contribution guide Set up CI

a225b6e4 Release NUPW v4.04

NUPW Distribution

- Download Zip NUPW v4.04
- Build info
- All versions

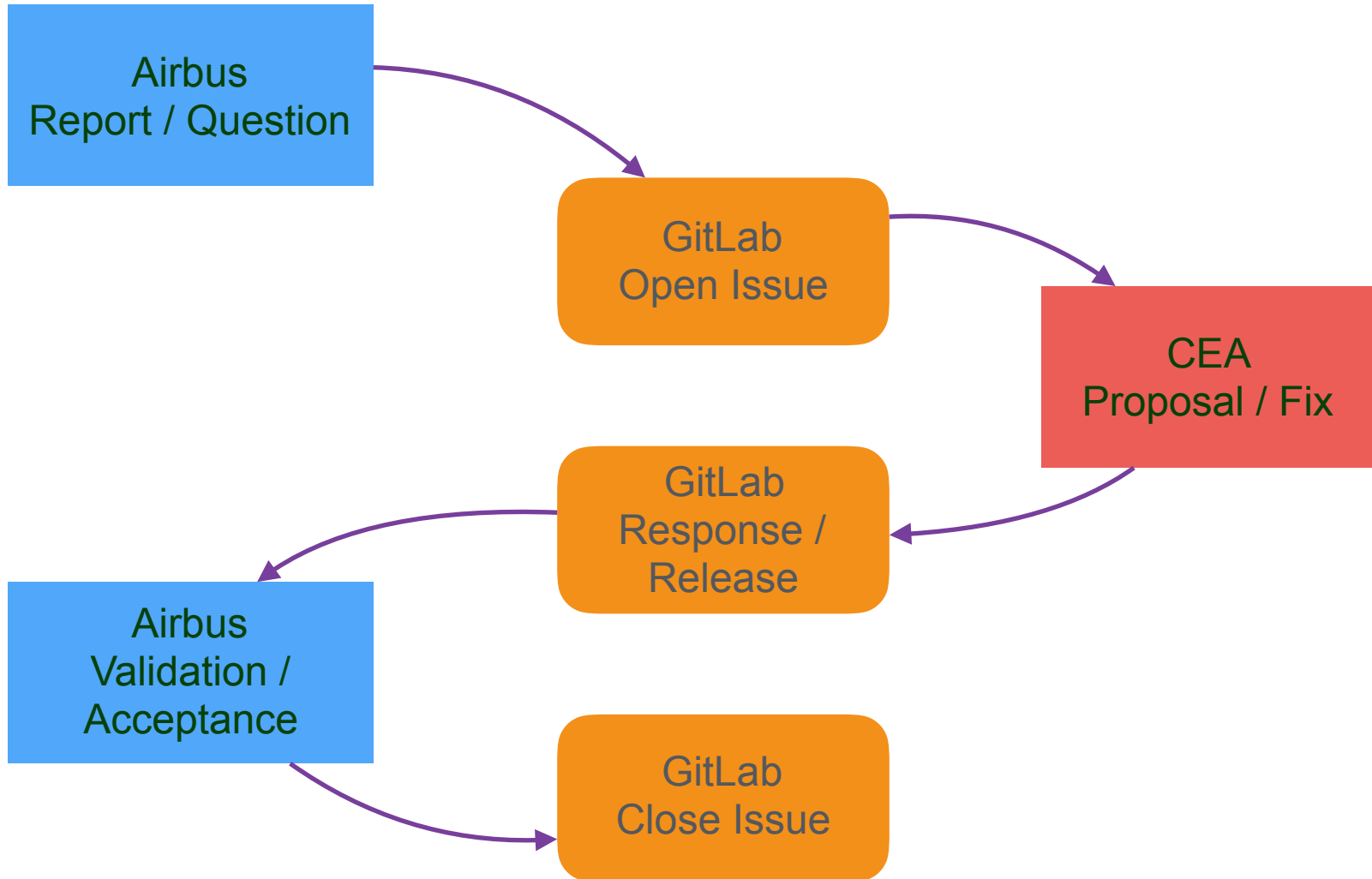
NUPW Support

- Support Site

Misc

- Git Recommendations

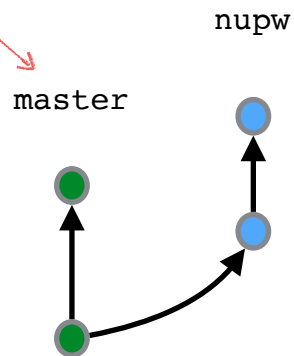
« SUPPORTING » MODE



MIXING « R » & « D » VIA GIT

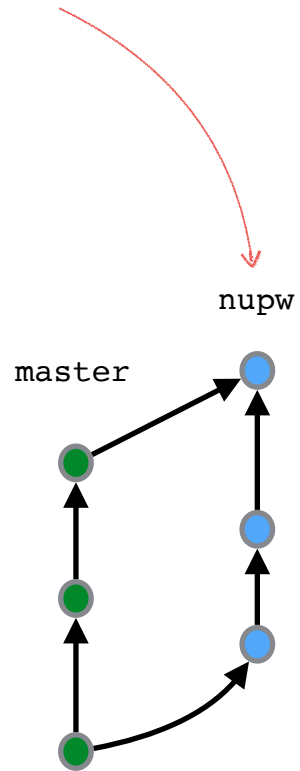
Research Experiment(s)

Dedicated Development(s)

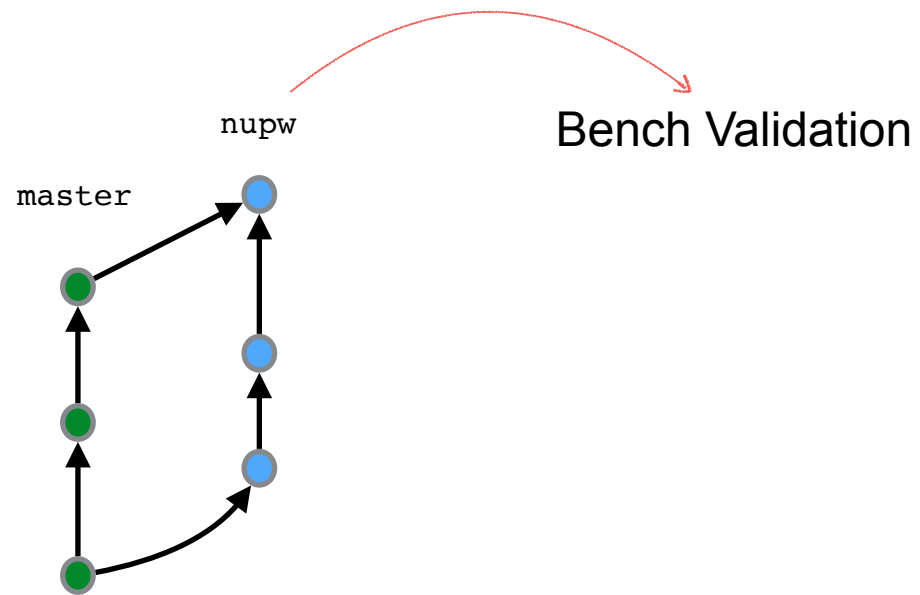


MIXING « R » & « D » VIA GIT

Platform Upgrades

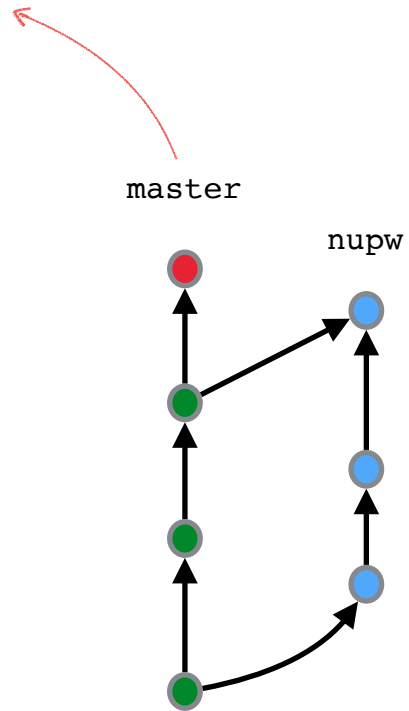


MIXING « R » & « D » VIA GIT



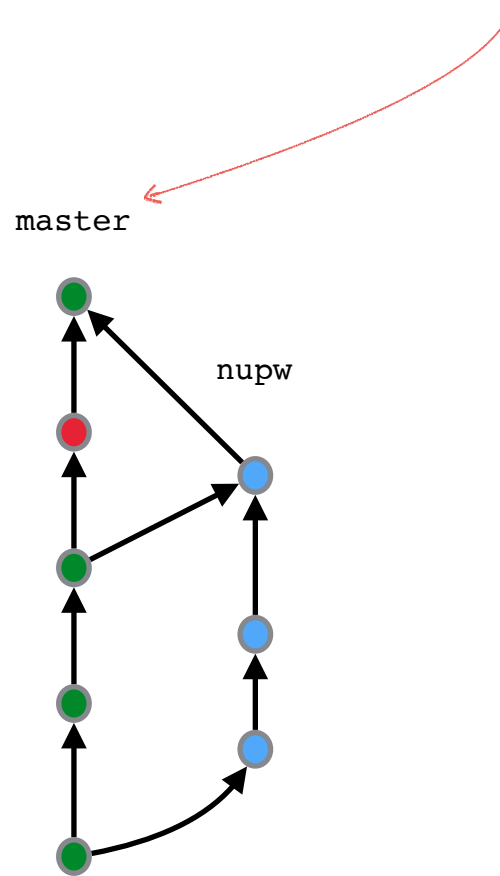
MIXING « R » & « D » VIA GIT

Public Release



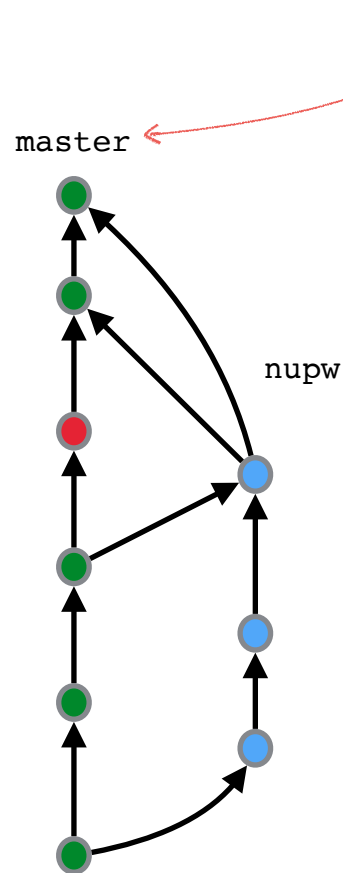
MIXING « R » & « D » VIA GIT

Backport — « kernel » part

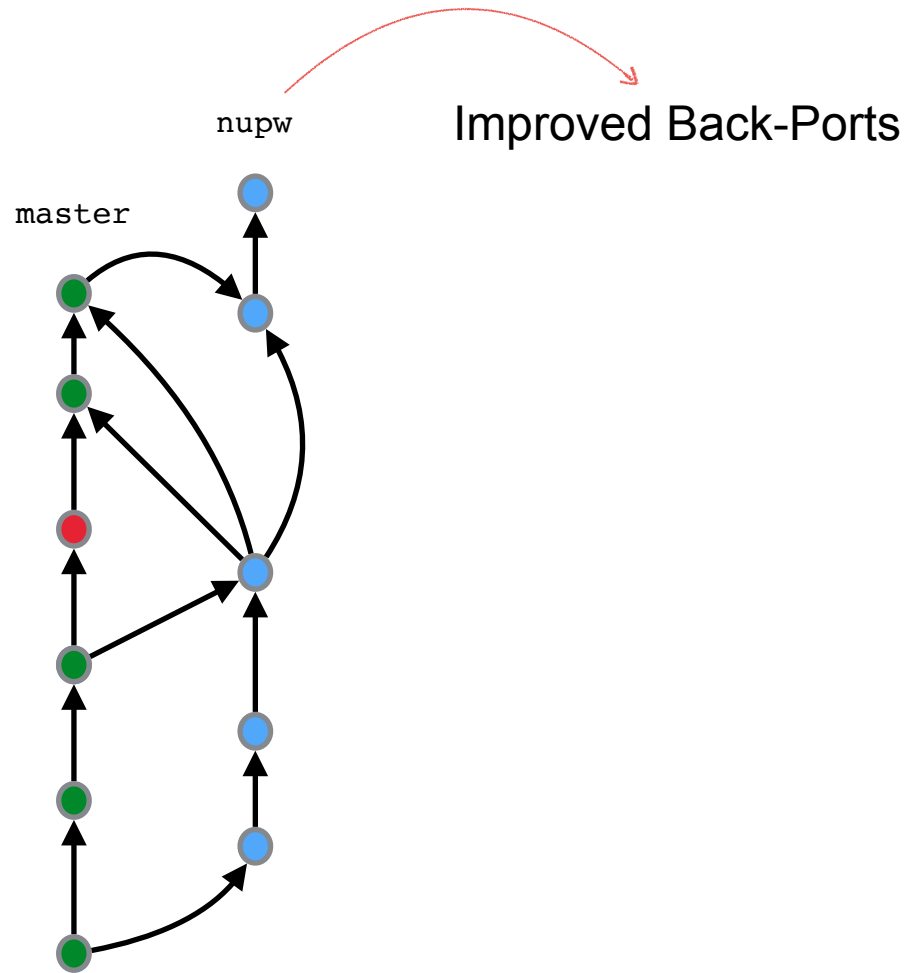


MIXING « R » & « D » VIA GIT

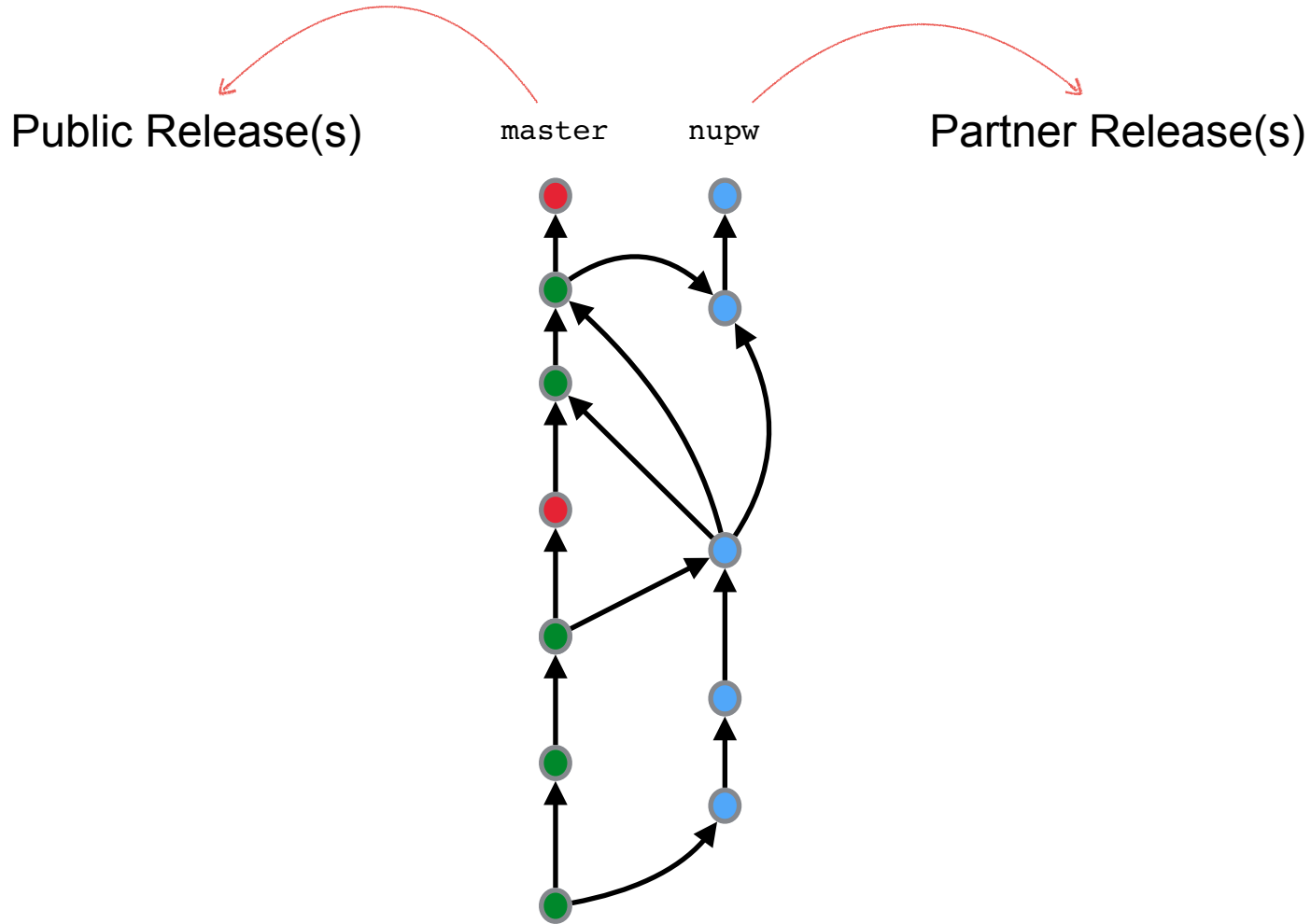
Backport — per « Plugin » parts



MIXING « R » & « D » VIA GIT



MIXING « R » & « D » VIA GIT



THE TECHNICAL PARTS

- Qed as an intermediate representation
- Inlined normalisation to avoid combinatorial explosion
- Maximal in-memory sharing preservation
- Agressive constant & equality propagation
- Integer modulo computations
- Integer bitwise simplifications
- Loop unrolling heuristics
- Hypothesis compaction & erasing
- Hypothesis generalisation (eg. 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)
- ...

NUPW AUTOMATION

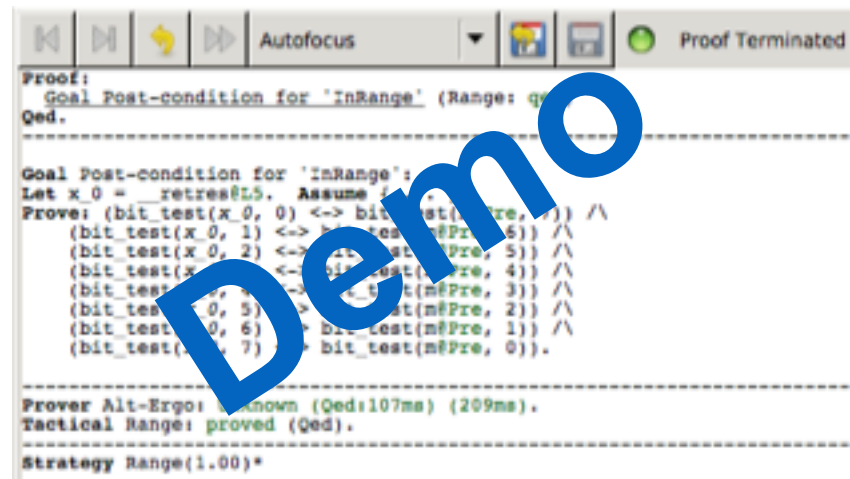
AIRBUS 16-Modules Bench

2012	93%	88%	88%	73%	89%	96%	100%	92%	98%	68%	35%	79%	96%	92%	64%
2014	100%	99%	99%	100%	95%	100%	100%	97%	96%	81%	95%	100%	100%	95%	85%
2016 (auto / tip)	100%	100%	90 / 99%	100%	100%	100%	100%	99 / 100%	97 / 100%	95 / 100%	100%	100%	100%	96 / 100%	94 / 95%

2012 — 2017

THE LAST 1% TO BE PROVED

Frama-C/WP
Interactive Proof Transformer



The screenshot shows a software interface for a proof transformer. At the top, there is a toolbar with navigation icons (back, forward, refresh, search) and a status indicator that says "Proof Terminated". Below the toolbar, the text "Proof:" is followed by "Goal Post-condition for 'InRange' (Range: q)". The main area displays a list of logical goals to be proved, each involving bit tests on variables x_0 and m#Pre. The goals are: (bit_test(x_0, 0) <-> bit_test(m#Pre, 7)) /\ (bit_test(x_0, 1) <-> bit_test(m#Pre, 6)) /\ (bit_test(x_0, 2) <-> bit_test(m#Pre, 5)) /\ (bit_test(x_0, 3) <-> bit_test(m#Pre, 4)) /\ (bit_test(x_0, 4) <-> bit_test(m#Pre, 3)) /\ (bit_test(x_0, 5) <-> bit_test(m#Pre, 2)) /\ (bit_test(x_0, 6) <-> bit_test(m#Pre, 1)) /\ (bit_test(x_0, 7) <-> bit_test(m#Pre, 0)). Below the goals, the text "Prover Alt-Ergo: Known (Qed:107ms) (209ms). Tactical Range: proved (Qed)." is displayed. At the bottom, "Strategy Range(1.00)*" is shown.

```
Proof:
Goal Post-condition for 'InRange' (Range: q)
Qed.
-----
Goal Post-condition for 'InRange':
Let x_0 = __retres@L5. Assume {
Prove: (bit_test(x_0, 0) <-> bit_test(m#Pre, 7)) /\
(bit_test(x_0, 1) <-> bit_test(m#Pre, 6)) /\
(bit_test(x_0, 2) <-> bit_test(m#Pre, 5)) /\
(bit_test(x_0, 3) <-> bit_test(m#Pre, 4)) /\
(bit_test(x_0, 4) <-> bit_test(m#Pre, 3)) /\
(bit_test(x_0, 5) <-> bit_test(m#Pre, 2)) /\
(bit_test(x_0, 6) <-> bit_test(m#Pre, 1)) /\
(bit_test(x_0, 7) <-> bit_test(m#Pre, 0)).
-----
Prover Alt-Ergo: Known (Qed:107ms) (209ms).
Tactical Range: proved (Qed).
-----
Strategy Range(1.00)*
```

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

TIMELINE (CONCLUSION)

2011

2014

2017

R&D | CAVEAT

R&D | NEW UNIT PROOF WORKSHOP

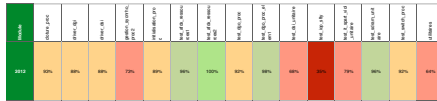
R&D | LLR

SUPPORT



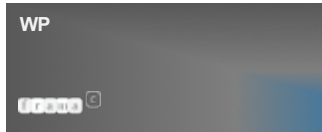
2008

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



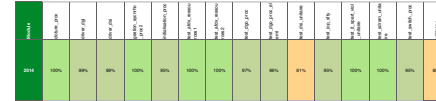
2011

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



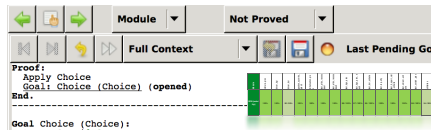
2012

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



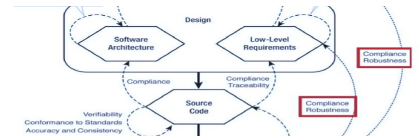
2014

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! »