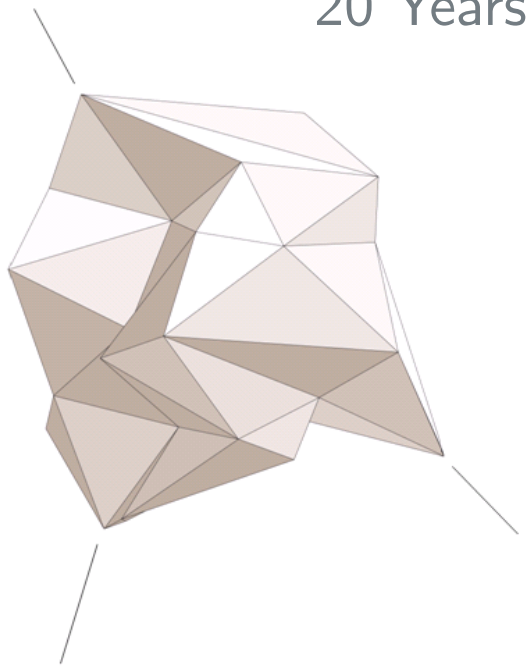
ITP 2014

Are we there yet?

20 Years of industrial theorem proving with SPARK

July 15, 2014



Rod Chapman
roderick.chapman@gmail.com

Florian Schanda
florian.schanda@altran.com

ALTRAN

Apologies from Rod

Rod has sadly left the company at the end of May - Flo will be giving the talk on his behalf.

Introduction

Purpose and structure of this talk:

- Briefly cover our experiences over the last 20 years
- Swap between three broad groups:

Project	Significant SPARK projects
Feedback	How user feedback changed the tools
Open	Interactions with free software

The Early Days

1987

- SPADE developed at University of Southampton
  - Intermediate language “FDL”
  - Verification condition generator
  - Checker (interactive proof tool, written in PROLOG)

The Early Days

1987

- SPADE developed at University of Southampton
  - Intermediate language “FDL”
  - Verification condition generator
  - Checker (interactive proof tool, written in PROLOG)
- Followed by SPARK for Ada’83
  - “Examiner” to produce FDL
  - Checker
  - Simplifier

Project: SHOLIS

1995

- Assists naval crew with the safe operation of helicopters at sea
- Shows safety limits on wind vectors, ship's roll and pitch, etc.

ALTRAN

Project: SHOLIS

1995

- No operating system and no COTS libraries of any kind
- 27 kloc (logical) of SPARK code, 54 kloc of information-flow contracts, and 29 kloc of proof contracts
- 9000 VCs
- 75.5% proven automatically by the Simplifier
- 2200 remaining VCs proved manually using the Checker

Project: C130J

1995

- Lockheed-Martin C130J is the latest generation of the "Hercules" transport aircraft
- Mission Computer implemented in SPARK, and was subject to a large verification effort in the UK

Project: C130J

1995

- Originally started as Ada code, but was converted to SPARK
- Only flow analysis and testing to meet DO-178B Level A
- This was already very successful (used only 20% of testing budget)
- Later, UK MoD demanded proof to meet DEFSTAN 00-55
- Original spec (in Parnas-Tables) converted to pre- and post-conditions
- Proof effort was completed successfully (sorry – no stats available!)

Feedback: SHOLIS and C130J

2002

- Automatic proof using Simplifier saved a lot of time. Special tuning for common SPARK VCs was added:
  - Unwrapping and instantiation of $\forall$ conclusions, in particular those for array type-safety
  - Unsigned wraparound arithmetic - very common in low-level device drivers, ring buffers, cryptography, etc.
  - Basic interval arithmetic for integer expressions

Feedback: SHOLIS and C130J

2002

- We noticed that the "proof friendliness" of the code has by far the largest impact
- This correlates with general "good ideas": simplicity, low information-flow coupling, good use of abstraction
- We started to set goals for projects to hit a particular level of automatic proof (95% automatically discharged)

Project: Tokeneer

2003 – 2008

- NSA-funded demonstrator of high-security software engineering
- Developed to meet Common Criteria EAL5
- System specification and security properties in Z, and implementation in SPARK
- Small system (and budget), about 10 kloc logical, producing 2623 VCs
- 2513 of those were proven automatically (95.8 %), with only 43 left to the interactive proof and 67 discharged by review
- Open source since 2008, including all project documentation, plans, etc. Go and download!

Feedback: User rules

2006

- Replaying checker proofs is very fragile
- Looking at VCs and deciding "seems OK" is not very formal, and is also fragile
- We added support for user-supplied axioms for the Simplifier

Feedback: User rules

2006

Pros:

- Easy to use
- Rules are much less fragile
- Can be formally reviewed without understanding the code
- Can be checked by tools (Checker, Victor and Riposte)

Feedback: User rules

2006

Pros:

- Easy to use
- Rules are much less fragile
- Can be formally reviewed without understanding the code
- Can be checked by tools (Checker, Victor and Riposte)

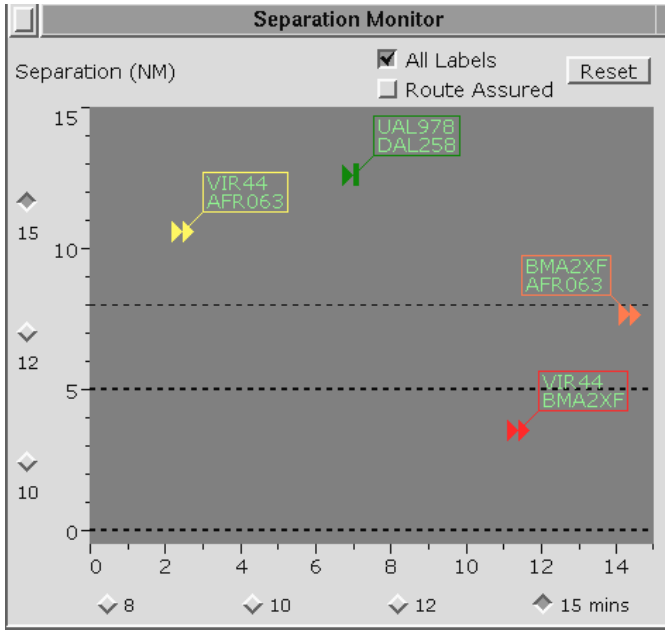
Cons:

- Opens the door to unsound proofs

Project: iFACTS

2006 → today

- iFACTS augments tools for en-route air-traffic controllers in the UK
- Provides electronic flight-strip management, trajectory prediction and medium-term conflict detection



Project: iFACTS

2006 → today

- In full operational service since 2011
- Formal specification in Z
- Written in SPARK – 250 kloc
- 153,000 VCs of which 98.76% are discharged automatically (user rules and review for the rest)

Feedback: Parallelisation

2007

Cheap, multi-core CPUs are now widely available.

- We noticed that all VCs produced by SPARK are independent (not part of the original design, but clearly a good idea anyway)
- Obvious improvement was to invoke more than one instance of the Simplifier
- If you sort the VCs based on file size, you get a near-optimal linear speedup
- Results are dramatic: difference between a 12 hour and 3 hour proof is significant (overnight $\rightarrow$ during the day)

Open: AdaCore Partnership

2009

AdaCore ALTRAN

PARTNERSHIP

- All SPARK tools released under the GNU GPL (version 3)
- This was a huge step for us (closed code base since 1987)
- New SPARK tools are jointly developed by Altran and AdaCore

Open: Victor

2009

- Paul Jackson (University of Edinburgh)
- Simplifier (a rewrite engine) has trouble with some "obvious" proofs:

$$a \vee b \implies b \vee a$$

- Translates FDL VCs to SMTLIB (and runs an SMT solver, for example Alt-Ergo, CVC4, etc.)
- Now part of the official SPARK release

Open: Riposte

2011

- Martin Brain (University of Oxford)
- Failed proof attempts might be real bugs, or incompleteness in prover – determining which costs time (€)
- Transforms FDL VCs into answer-set instances (and runs an answer-set solver, for example clasp) and provides counter-examples
- Now part of the official SPARK release

Project: SPARKSkein

2010

- Common misconception "Ada is slower than C because of all this safety stuff..."
- Implementation of Skein (a contender for SHA3, sadly not the winner) in SPARK
- Clean implementation (for example instead of macros, we just use normal procedures)
- After some improvements in the gcc backend, the C and SPARK implementations are equally fast

Project: SPARKSkein

2010

- Absence of RTE proved: originally 23 of 367 VCs proved via Checker, now 100% is proved automatically using Victor or Riposte
- Proof was difficult: non-linear arithmetic and modular types
- We found an arithmetic-overflow bug in the C reference implementation (since the SPARK implementation closely mirrored it)
- Released as free software (GPLv3)

Feedback: Cachesimp

2012

Time used to do proof is not just a number, it has significant impact on how the tools are used. For example:

2 weeks You structure your project around this. Proof failures are extremely costly and are major setbacks.

1 day You structure your workday around this. Proof failures may result in delayed delivery.

10 minutes You just have another cup of tea. Proof failures are detected so early, they never become an issue.

Feedback: Cachesimp

2012

- Proof on iFACTS used to take 3 hours on a modern GNU/Linux computer, regardless of size of code change
- Cachesimp is a simple tool ($\approx$ 250 lines of code) that hashes a VC and queries a central server if the answer is already known
- Memcached is effectively a hash table with a simple text protocol; originally implemented and used by LiveJournal to cache queries
- Incremental (previous answers cached) + distributed (many clients can connect simultaneously)
- Average 29-fold speedup, most code changes can now be fully analyzed in less than 30 minutes

Open: Isabelle

2011

- Stefan Berghofer (secunet)
- Isabelle/HOL plugin to read SPARK VCs
- SPARK always allowed proof functions, and this is an elegant and formal way to give them meaning
- Used to implement and formally verify a BigNum library and a cryptography library
- Part of the official Isabelle release
- Also free software (both the plugin, and the above libraries)

Project: Muen

2013

- Reto Buerki and Adrian-Ken Rueegsegger (both HSR – University of Applied Sciences Rapperswil)
- Separation kernel for Intel x86/64 platform
- Written in SPARK (2463 logical), and assembly (256 lines)
- Proof of absence of RTE, all 666 VCs are discharged automatically
- Again, free software (GPLv3)

The Future: SPARK2014

SPARK2014 is a complete re-design:

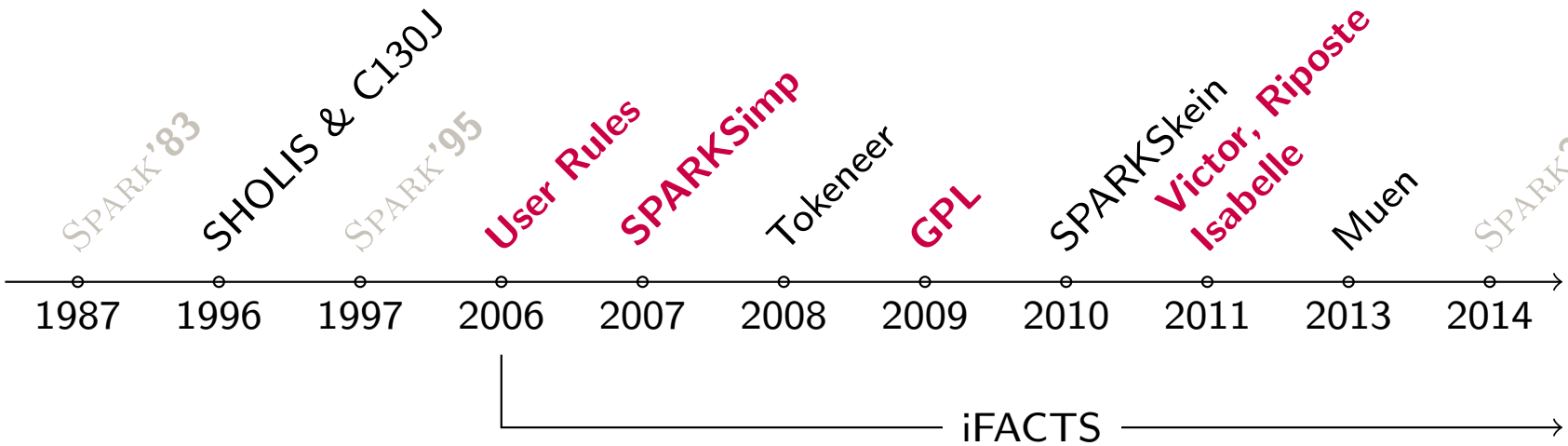
- Based on the GNAT front-end (the gcc Ada front-end)
- WhyML as the intermediate language (instead of FDL)
- SMT solvers (CVC4, Alt-Ergo, etc.) as the automatic proof tools (instead of Simplifier)
- Support for Isabelle, Coq, etc. (instead of Checker)
- Still free software (GPLv3)

The Future: SPARK2014

Not just the tooling is different:

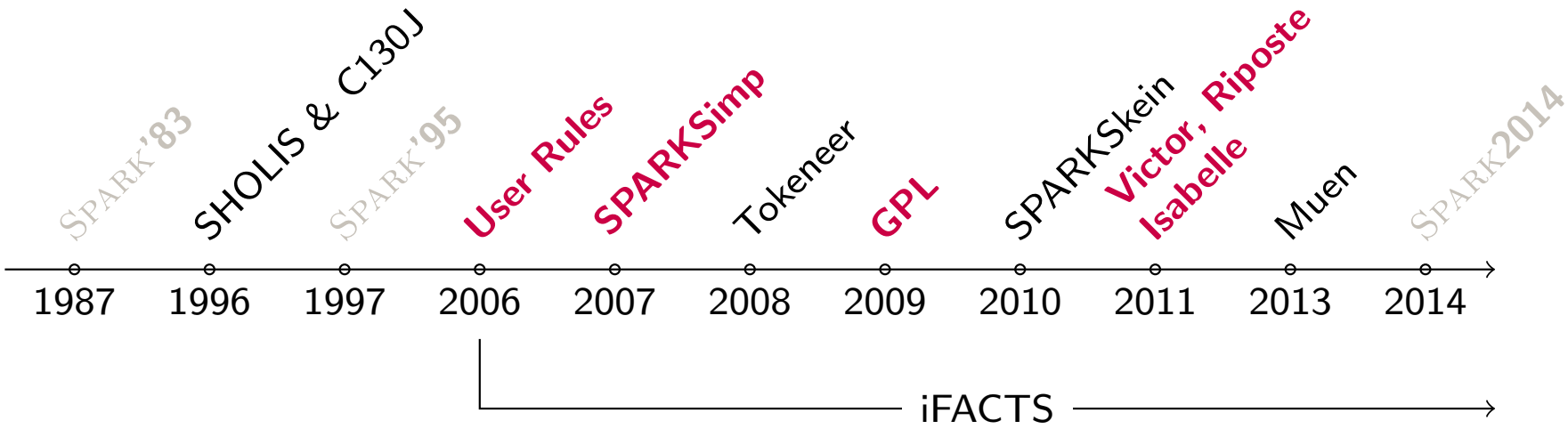
- Larger subset of Ada
- Contracts are part of the language, not "special comments"
- Executable contracts allow combination of test and proof: this will help with DO-178C
- Sound IEEE-754 floating point support (instead of reals)

Conclusion



- SPARK has been used for a long time in the industry
- ITP is used for groundbreaking work, but automatic provers are always in demand
- Free software - `http://spark-2014.org`

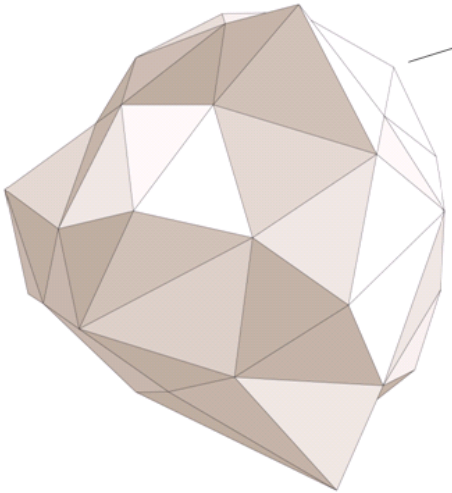
Conclusion



- SPARK has been used for a long time in the industry
- ITP is used for groundbreaking work, but automatic provers are always in demand
- Free software - `http://spark-2014.org`

Thank you for listening - any questions?

INNOVATION MAKERS



ALTRAN