# SPARK 2014 and GNATprove

**Roadmap and Challenges** 

## Roadmap

- May 29 2013: GPL 2013 release (called SPARK-HiLite)
- June 2013: hi-lite project on Open-DO moved to spark2014 project (also public)
- June 2013: finalization of SPARK 2014 RM
  - most of SPARK 2005 supported (not yet supported: RavenSPARK + Object Oriented programs)
  - generation of Global is not described in RM
- November 2013: beta release
- Q1 2014: release 1 of SPARK 2014
  - flow analysis, non-aliasing analysis, proof

## Assumptions (1/3)

#### • Why?

- Allow mixing different verification methods
- Allow mixing of SPARK and non-SPARK code
- Allow mixing of Ada and C Code
- How?
  - Two phases
  - Modular generation of explicit assumptions
  - Aggregation of assumptions and verification results
- Existing approaches
  - Evidential Tool Bus (SRI)
  - Frama-C collaboration of plug-ins

## Assumptions (2/3)

procedure F
 (V : in out Integer) is
begin
 if V > 0 then
 ...
 G (V, Z);
 ...
 end if;
end F;

#### Tool output:

file.adb:12:7: precondition of G proved
file.adb:15:6: postcondition of F proved
file.adb:12:7: postcondition of G assumed

### Assumptions (3/3)



Christakis, Müller, Wüstenholz: Collaborative Verification and Testing with Explicit Assumptions, FM 2012

## **Object Oriented Code**

- Support for behavioral subtyping only
- Check Liskov Substitutability Principle (LSP)
  - weaker Pre and stronger Post
  - Less Global Input and Global In\_Out
  - same Global Ouput
- Subprogram checked against Pre/Post
- Which contract for dispatching?
  - Pre / Post ?
  - Pre'Class / Post'Class ?
- Global'Class / Depends'Class ?

#### RavenSPARK

- RavenSCAR is a subset of Ada for safe usage of tasking features (schedulability)
  - only top-level tasks
  - fixed priorities
- RavenSPARK is a subset of RavenSCAR compatible with SPARK
  - tasks communicate only through protected objects
- Proof of protected objects & tasks (seems "simple")
- Proof of manipulation of protected objects (similar as "volatile")

## Data Invariants (1/2)

#### • Subtype predicates

- "strong" invariant
- Add a predicate to a type that should always be true
- Will support only limited form in SPARK:
  - Cannot mention global variables
- Support in GNATprove seems straightforward
  - insert assumptions/assertions where needed

## Data Invariants (2/2)

#### • Type invariants

- "weak" invariant
- Add a predicate to a type that can be temporarily broken by "primitive operations" (functions) of that type
- In SPARK, invariant should not depend on global variables
- Sufficient to enrich precondition/postcondition?
- Do we need more restrictions?

## Non-Linear Arithmetic (1/2)

#### Two lines of work:

#### 1. Axiomatisation + Heuristics

- produce Why encoding/axiomatisation for non-linear operations
- improve Alt-Ergo's provability and performance based on practical problems

2. Keep good interface with multiple provers

- non-linear arithmetic is an active research area
- decision procedures for SMT solvers using bitvectors, computer algebra
- implemented in Z3, CVC4, Boolector, Alt-Ergo?

## Non-Linear Arithmetic (2/2)

#### • Axioms + Heuristics

- Advantages: tailored for industrial problems, shortterm bang for buck
- Disadvantages: possibly fragile, prover specific
- Decision procedures
  - Advantages: more predictable, based on fundamental knowledge, long-term solution
  - Disadvantages: possibly too time consuming, may not work on industrial problems
- Compare Simplifier vs Victor

A bit of both?

#### Counterexamples (1/3)

000	🗙 GPS – (Search project) – search.adb	
<u>File Edit Navigate VCS Proje</u>	ect <u>B</u> uild <u>D</u> ebug <u>T</u> ools <u>C</u> odePeer Prove <u>W</u> indow <u>H</u> elp	0
1 🗀 🔁 🛭 🖘 🕐 🛛	🚽 📑 🔤 default 🔽 🖉 🕸 😤 🍘 ▷	
Project ♥	<pre>     UserSimov,MHild Costron Incomparation Control (A control</pre>	, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1) 0, 0, 0, 0, 0, 0, 0, 0, 1) , , , , , , , , , , , , , , , , , , ,
Project Outline		
	Messages Locations Car nees Entity	
		h.

### Counterexamples (2/3)

1. generate VCs with labels

goal Incorrect: (forall x: int. ("model:0":x) <> 0)

2. call alt-ergo with switch -model

alt-ergo -model file.why

3. extract equalities with literals from model

x = X1(arith): [0 [int]]

4. display extracted values in GPS

Concrete model instead of propositional one? Partial model when timeout reached?

#### Counterexamples (3/3)

```
(set-logic AUFLIA)
 (declare-fun x () Int)
 (assert (= x 0))
 (check-sat)
 (get-value (x))
 (exit)
```

#### Why

goal Incorrect:
 (forall x: int.
 ("model:0":x) <> 0)

<b>∀</b> x, y : int.	x <> 0	x = 0	x < 0	x < y
Z3	x = 0	x = 1	x = 0	x = 0, y = 0
CVC4	x = 0	x = 1	x = 0	x = 0, y = 0
Alt-Ergo	x = 0	x ∈ ]-∞;-1] ∪ [1; +∞[	x ∈ [0;+∞[	x > (y - 1)
Riposte	x = 0	x = -1	x = 0	x = 0, y = 0
Sireum Kiasan	x = 0	x = 1	x = 0	x = 0, y = 0

## Floating-points (1/2)

- mathematical reals are used to model floating points in proof
- difference between executable semantics and proof semantics
- false positives and negatives
- Way out: use floating point semantics and proof tools with floating point support (Gappa, Alt-Ergo + Gappa)

Boldo, Clément, Filliâtre, Meyero, Melquiond, Weis: Wave equation numerical resolution: a comprehensive mechanized proof of a C program. *Journal of Automated Reasoning, 2013* 

## Floating-points (2/2)

- Floating point semantics also for assertions, is it a limitation?
- Is NaN allowed?
- Is +/-Inf allowed?
- Can we have a type "float" in Why3 (programs)?

## **Multi-prover Approach**

#### Benefits

- increase provability (portfolio approach)
- help during debugging (detect prover shortcomings, generate counterexamples)

#### • SMT solvers

- encoding is important, ongoing work
- use of SMT built-in types as much as possible
- careful use of triggers

#### • First-order provers

- Why to Spass, E-prover, Vampire
- possibly more: Paradox, Equinox...
- need more investigation on practical problems

#### **Axiomatized Units**

- User can define Why3 theories for Ada units
  - To improve efficiency (containers)
  - To improve expressivity (sum\_of, permutation...)
- Works for generic packages
   Uses Why3 clone
- User can start from auto-generated stubs
  - Generate expected signature for Ada elements
  - Generate complete translation of Ada types
  - One namespace per Ada declaration
- Theories provided for SPARK Libraries

#### **Bridge to Manual Provers**

O     O     Why3 Interactive Proof Session				interest in
Context	Theories/Goals	Status Tir 🗕 95	axiom H3 : to_int3 pos1 < 10	
<ul> <li>Unproved goals</li> </ul>	マ 🖏 split_goal	· · · · · · · · · · · · · · · · · · ·	axiom H4	Isabelle (Securet)
O All goals	🚊 1. precondition	950 950	to_int5	
Provers	🗢 ờ Search_linear_search_pre	95 96	to_int5 val_	onsetta) - atr_Ai
Alt Ergo (0.04)		96 96	constant res : bool	
ALCEIGO (0.54)	∽ 🖏 split_goal	96	3	
Alt-Ergo (0.96)	1. assertion	96 96	5	
Coq (8.3pl4)	2. assertion	96 96	5 axiom H5 : res = of_int6 1 \(\Lambda\) res 7	Elle Edit Navigation Iry lactics lemplates Queries Display Compile Windows Help
Transformations	✓ Ø Search_linear_search_def	96	<pre>constant search_linear_search</pre>	
🖏 Split	VC for def	97	constant search_linear_search	
	v 🖏 split_goal	97 97	1 2 axiom H6 :	earch_package_Search_linear_search_def_WP_parameter_def_2.v
	1. precondition	97	3 search_linear_search_result	
Tools		97: 97:	search_ineal_search_result	b) = false).
n Edit	a sesertion	Ø [" 97	6 axiom H7 : ustemp_standard: 7	
Replay	4 assertion	97	goal WP_parameter_def : to_ and	<pre>(* Why3 assumption *) Definition in concess (rec. found2:bool) (alises</pre>
Cleaning	5. assertion		- End	(rec found3 = (rec found al)).
Remove	6. postcondition	2 14	4 let def "GP_Subp:search.ads:25"	
Nemove	7. precondition	(2)	6 ensures { ( <mark>[Searchelement</mark> 6 ensures { ( "GP_Reason:VC_POS	Parameter dummy/: tib.
💮 Clean	8. precondition	(2) 14 14 14	7 ( "GP_Pretty_Ada:A (Linear_Searc	(* Why3 goal *)
Proof monitoring	9. precondition	(2) 14		Theorem WP parameter def : (0%Z <= (to int3 va) Welcome to CoqIDE, an Integrated Developmen
Waiting: 0 Scheduled: 0	10. assertion		<pre>D (let _temp_standard_search_3) 1 ensures {( result = (Integer.bo)</pre>	(pos = (of_int1 1%2)) -> (((to_int1 pos) < 1) = [] [] [] [] [] [] [] [] [] [] [] [] []
Running: 1 \	🚺 11. assertion	<sup>15</sup> <sup>15</sup> <sup>15</sup>	2 3 ) in (	(((to_int1 pos1) + (offset1 a))%Z - 1%Z)%Z)) sion 8.3pl4 (June 2012)
Miterrupt	12. assertion	(2) 15- 15- 15- 15- 15- 15- 15- 15- 15- 15-	4 (let_temp_standard_search_5	forall (res:bool) (resl:index), ((res = (of i
	🚉 13. postcondition	Ø     15     15	6 (((1<= search_linear_search	s foract (search_thear search_result):index).
	14. precondition	<sup>15</sup> <sup>15</sup> <sup>15</sup>	7 True) else ( 3 False)) )}	((search_linear_searchresult = res) /\
	15. precondition	Ø     15     1     1     1	) ) := (	(search_linear_search_result1 = res1)) ->
	16. assertion	20 16 16	1 (let_temp_standard_search_4 :	((to int3 val)) = true) ->
	17. assertion		2 ensures {( result = (andb (Inter 3	((to_int4 search_linear_searchresult) = 1
	▶ 🞑 18. assertion			= intros hl pos h2 h3 posl h4 h5 res resl (h6,h7)
	4	▶ file:	search_package//search_packag	a search_theal_search_result search_theal_s
				Qed.
				Ready line: 1 Char: 1 Coolde started
				cine. I Chai. I Coque statted