The GNATprove tool for SPARK and other similar tools[1]

Paul Jackson
Paul.Jackson@ed.ac.uk

University of Edinburgh

Formal Verification
Autumn 2023

[1] Slides mostly by Florian Schanda, formerly at Altran UK

Tool architecture

User view

SPARK ⟶ gnatprove ⟶ Verdict

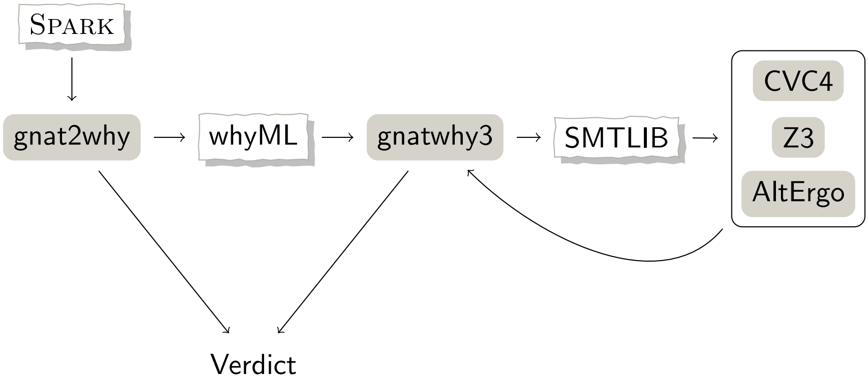
gnat2why

Translation to WhyML

- SPARK is an extremely complicated language
  - Would be huge amount of work to generate VCs directly from SPARK
- Much easier to translate first to the smaller, intermediate language WhyML
  - Simpler control flow
  - Simpler types
- VC generation is based on this IL

Tool architecture

More detailed view...



gnat2why

Translation to WhyML

```
function Example
   (A, B : Natural)
    return Natural
is
   R : Natural;
begin
   if A < B then
      R := A + 1;
   else
      R := B - 1;
   end if;
   return R;
end Example;
```

→

```
let example (a: int) (b: int)
    requires { a >= 0 /\ a <= 2147483647 }
    requires { b >= 0 /\ b <= 2147483647 }
    returns { r -> r >= 0 /\
                   r <= 2147483647 }
  = let r = ref 0 in
    if a < b then
       r := a + 1
    else
       r := b - 1;
    (!r)
```

gnat2why

Translation to WhyML

Actual translation embeds lots of extra information.
For $r = a/b$:

```
( ( "GP_Sloc:overflow.adb:7:7" ( #"overflow.adb" 7 0 0#
overflow__example__result.int__content <- ( (
#"overflow.adb" 7 0 0# "GP_Sloc:overflow.adb:7:16"
"GP_Shape:return__div" "keep_on_simp" "model_vc"
"GP_Reason:VC_OVERFLOW_CHECK" "GP_Id:1"
(Standard__integer.range_check_(( #"overflow.adb" 7 0 0#
"GP_Reason:VC_DIVISION_CHECK" "GP_Id:0"
"GP_Sloc:overflow.adb:7:16" "GP_Shape:return__div"
"keep_on_simp" "model_vc" (Int_Division.div_
(Overflow__example__a.a) (Overflow__example__b.b))
))) ) ); #"overflow.adb" 7 0 0# raise Return__exc ) );
#"overflow.adb" 3 0 0# raise Return__exc )
```

But we eventually get nice output...

overflow.adb:7:16: medium: divide by zero might fail (e.g. when B = 0)
overflow.adb:7:16: medium: overflow check might fail

gnat2why

Kinds of inserted checks

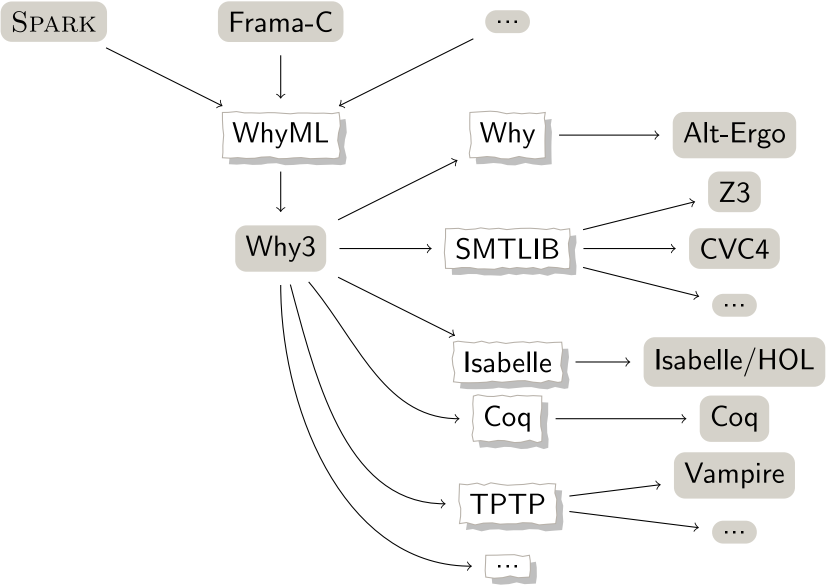
- Some checks are user defined
  - (user asserts, postconditions)
- Ada RM defines basic checks
  (overflow, range, index, division by zero, discriminants, etc.)
- SPARK RM defines more (LSP checks, loop variants and invariants, etc.)

gnatwhy3

Generates VCs in the SMT-LIB logic AUFBVFPDTNIRA

- Boolean
- Integer
- Reals
- Quantifiers
- Arrays
- Uninterpreted functions
- Bitvectors
- IEEE-754 Floating Point
- Algebraic Datatypes

Why3 ecosystem



Boogie ecosystem

Boogie is an intermediate-level verification language from Microsoft Research.

Front-ends include

Spec# for C#

Dafny Simple imperative language with heap data.

- Popular in teaching
- Recent application to secure web apps (*Ironclad*) and distributed systems (*Ironfleet*)

VCC For low-level concurrent C.

- Used to verify 60klines Hyper-V hypervisor.

SDV Microsoft's Static Driver Verifier

- Checks driver - Windows kernel interactions

Back-end analysis tools include:

Boogie tool generates VCs for SMT solvers Z3, CVC5, Yices2

Corral Bounded loop unrolling – no use of invariants.

- Used in SDV.

Other WP-based verification tools

Viper Language and tool suite

- Native support for permissions & ownership reasoning (e.g. in style of separation logic)
- Front-ends for Java, OpenCL, Rust, Python, Go, ...

Stainless for Scala, a JVM-based functional and OO language

F* A dependently-typed functional programming language using SMT solvers to prove specifications

OpenJML for Java

- JML is *Java Modelling Language*, an assertion language
- Descendent of ESC/Java system