Get Started with Open Source Formal Verification

Fabien Chouteau

Embedded Software Engineer at AdaCore

Mastodon : @DesChips@mamot.fr
GitHub : Fabien-Chouteau
Hackaday.io: Fabien.C

What is Formal Verification?

the act of proving or disproving the correctness of intended algorithms [...] using formal methods of mathematics [1]

[1] https://en.wikipedia.org/wiki/formal_verification

An example

```
y = 10 / (x - 10);
```

An example

```
y = 10 / (x - 10);
```

```
x - 10 != 0
```

An example

```
y = 10 / (x - 10);
```

```
x - 10 != 0
```

```
x != 10
```

An example

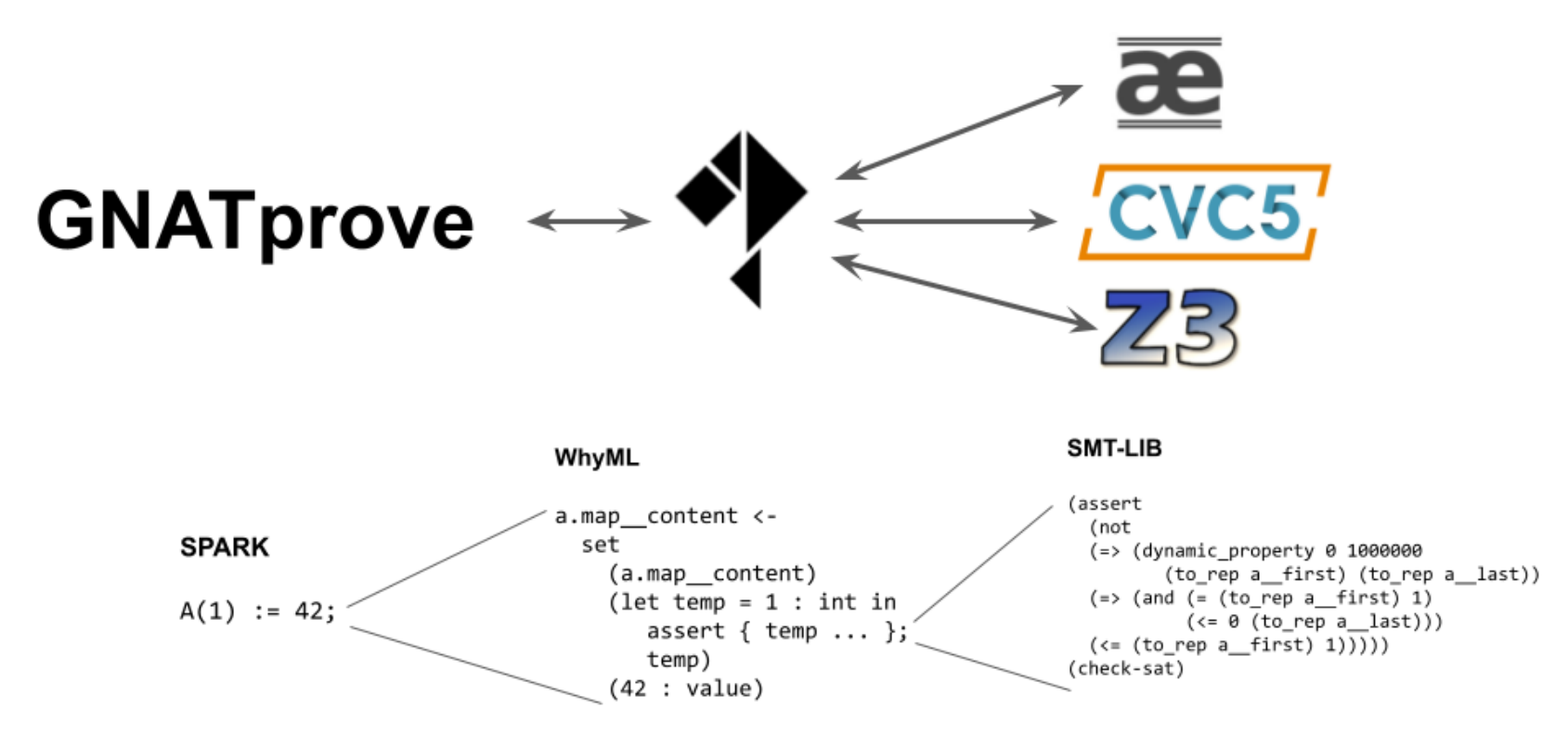
```
if (x != 10) {
  y = 10 / (x - 10);
} else {
  y = 42;
};
```

Spot the bugs

```
float * compute (int * tab, int size) {

   float tab2 [size];
   float * result;

   for (int j = 0; j <= size; ++j) {
       tab [j] = tab2 [j] / 10;
   }

   result = tab2;
   return result;
}
```

SPARK

SPARK - The Automatic Proof Toolkit



SPARK - The language

Why a subset of Ada?

```
type Percentage is new Float range 0.0 .. 1.0;
```

Why a subset of Ada?

```
type Stack is private;

function Is_Empty (S : Stack) return Boolean;
function Is_Full (S : Stack) return Boolean;

procedure Push (S : in out Stack; Value : Natural)
  with Pre  => not Is_Full (S),
       Post => not Is_Empty (S);
```

Why should I care about SPARK?

- No vulnerabilities for **any** possible inputs
- Proof of functional correctness
- Avoid some of the testing efforts

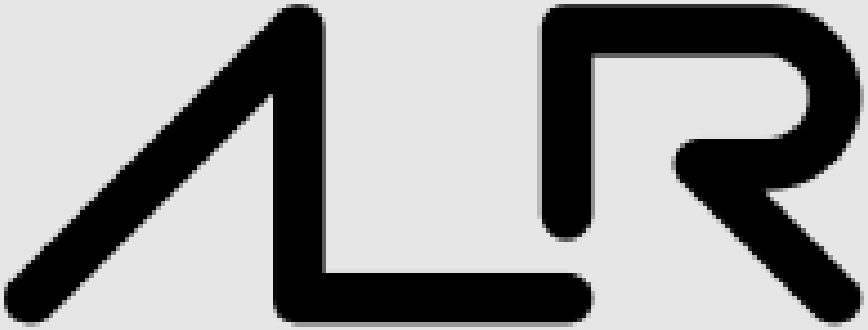
NVIDIA Security Team [2]:

- "Testing security is pretty much impossible"
- "provability over testing as a preferred verification method"
- "let's focus on other areas of security"

[2] https://www.adacore.com/papers/nvidia-adoption-of-spark-new-era-in-security-critical-software-development

Let's prove!

Download and install Alire



Download the Alire package manager from:

`https://alire.ada.dev`

Start a new crate

```
$ alr init --bin lets_prove
lets_prove initialized successfully.
```

```
$ cd lets_prove
```

Add gnatprove dependency

```
$ alr with gnatprove
```

Add some code

In src/lets_prove.adb:

```
with Ada.Text_IO;

procedure Lets_Prove
with SPARK_Mode
is
   X : constant Integer := Integer (Ada.Text_IO.Col);
   Y : Integer;
begin
   Y := 10 / (X - 10);

   Ada.Text_IO.Put_Line (Y'Img);
end Lets_Prove;
```

Run gnatprove

```
$ alr gnatprove
Phase 1 of 2: generation of Global contracts ...
Phase 2 of 2: flow analysis and proof ...

lets_prove.adb:9:12: medium: divide by zero might fail
    9 |   Y := 10 / (X - 10);
      |        ~~~^~~~~~~~~~
Summary logged in gnatprove.out
```

With counter examples

```
$ alr gnatprove --counterexamples=on
Phase 1 of 2: generation of Global contracts ...
Phase 2 of 2: flow analysis and proof ...

lets_prove.adb:9:12: medium: divide by zero might fail
    9 |   Y := 10 / (X - 10);
      |        ~~~^~~~~~~~~~
  e.g. when X = 10
Summary logged in gnatprove.out
```

Fix the code

```
with Ada.Text_IO;

procedure Lets_Prove
with SPARK_Mode
is
   X : constant Integer := Integer (Ada.Text_IO.Col);
   Y : Integer;
begin
   if X /= 10 then
      Y := 10 / (X - 10);
   else
      Y := 42;
   end if;

   Ada.Text_IO.Put_Line (Y'Img);
end Lets_Prove;
```

Run gnatprove again

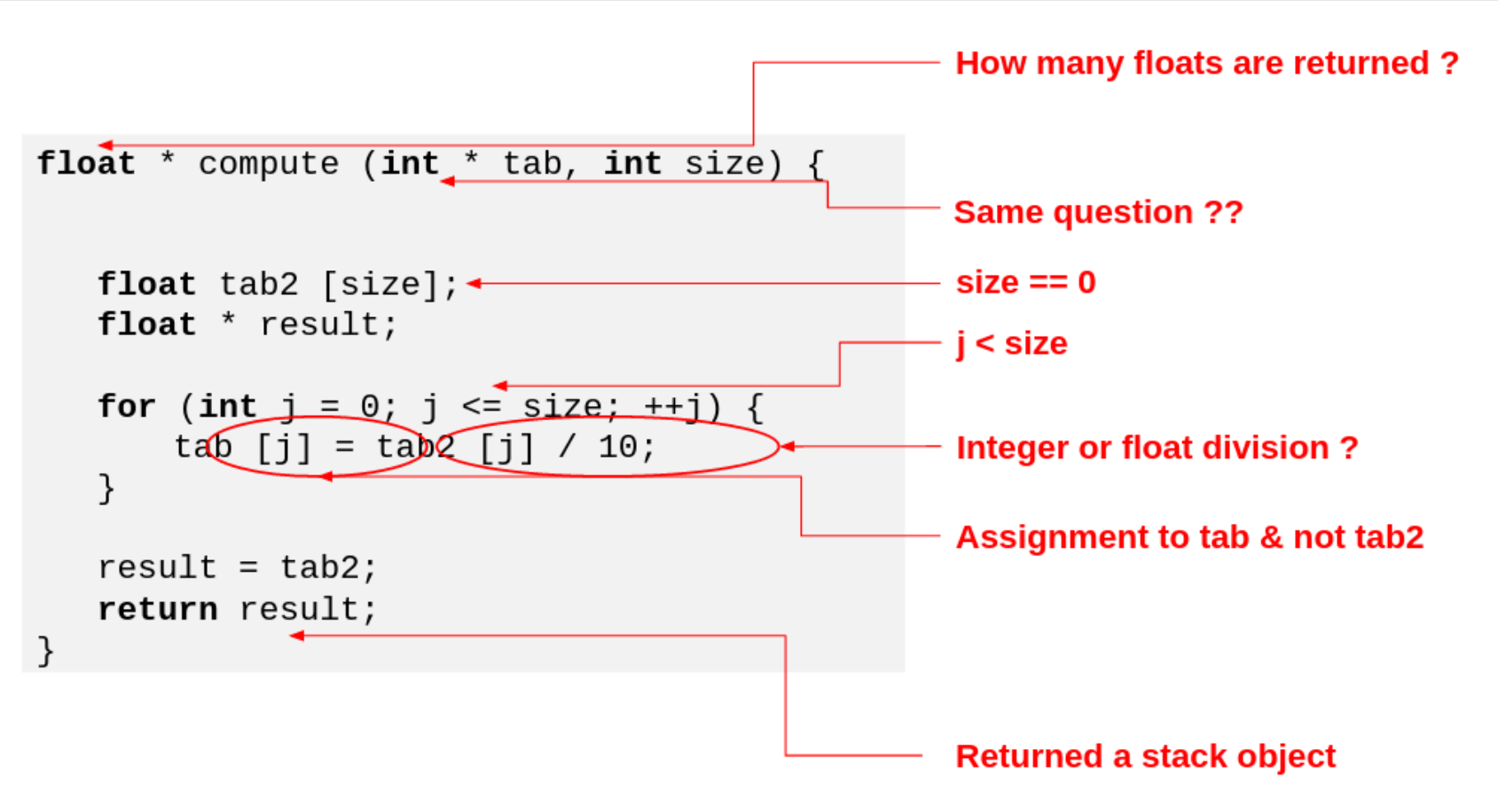
```
$ alr gnatprove
Phase 1 of 2: generation of Global contracts ...
Phase 2 of 2: flow analysis and proof ...
Summary logged in gnatprove.out
```

That's it you just proved your first program!

Resources

- learn.adacore.com: interactive learning website
- reddit.com/r/ada
- forum.ada-lang.io
- gitter.im/ada-lang/

The answer


float * compute (int * tab, int size) {
float tab2 [size];
float * result;
for (int j = 0; j <= size; ++j) {
tab [j] = tab2 [j] / 10;
}
result = tab2;
return result;
}
How many floats are returned ?
Same question ??
size == 0
j < size
Integer or float division ?
Assignment to tab & not tab2
Returned a stack object