Building automated and foundational verification tools with **Lithium**

Swiss Verification Day, 10.1.24

Michael Sammler

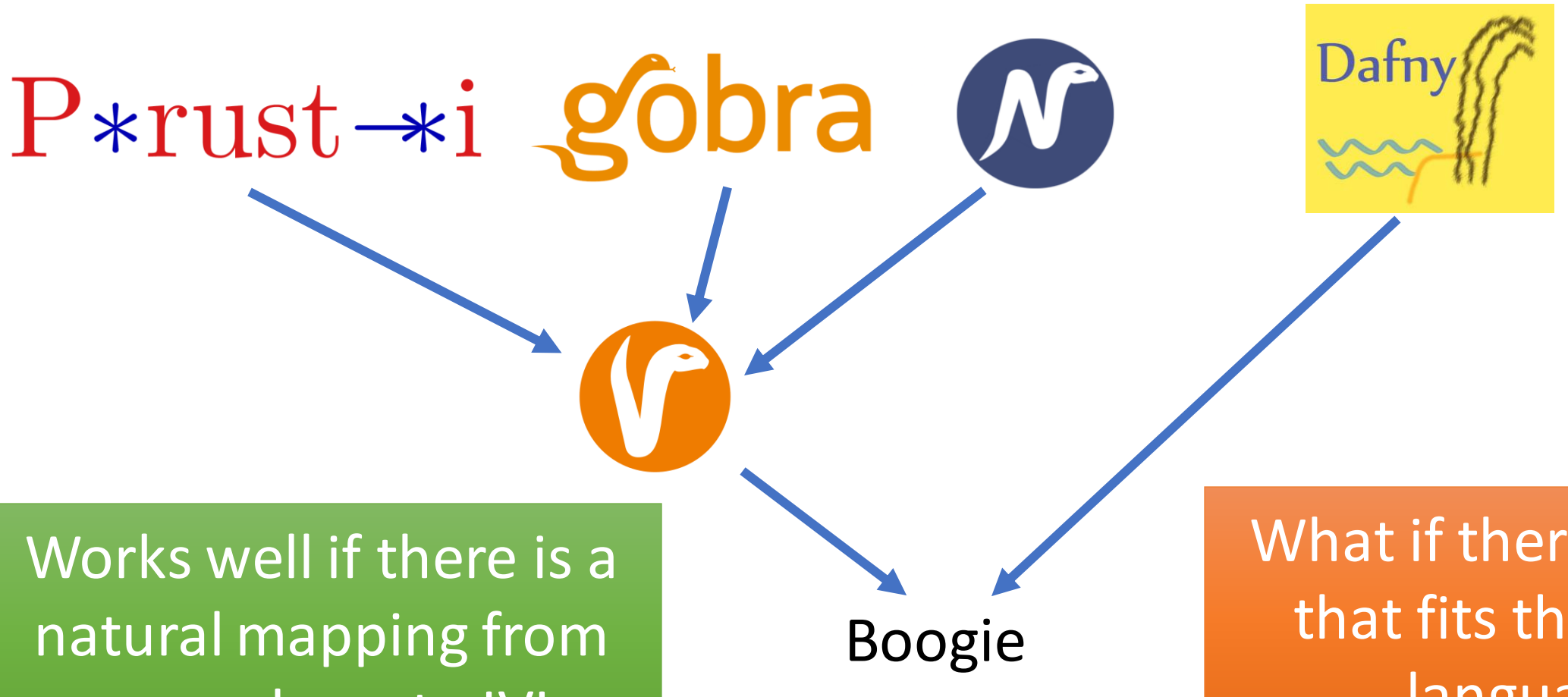
ETH zürich

Let's build a verification tool!

Building a verification tool from scratch is a lot of work!

How can we get a reusable foundation for building verification tools?

Common approach
Translate to an intermediate verification language (IVL)



Works well if there is a natural mapping from source lang. to IVL

What if there is **no** IVL that fits the source language?

Common approach
Translate to an intermediate verification language (IVL)

Source language
detailed model of C (e.g., taking addresses of local variables, int-ptr-casts, ...)

Proof search
based on type assignments

?

automated separation logic reasoning

Source language
assembly language with flat memory model

Proof search
based on points-to assertions

Challenge: Very different languages, memory models, and proof search procedures

~~Common approach~~

Translate to an intermediate verification language (IVL)

Key idea

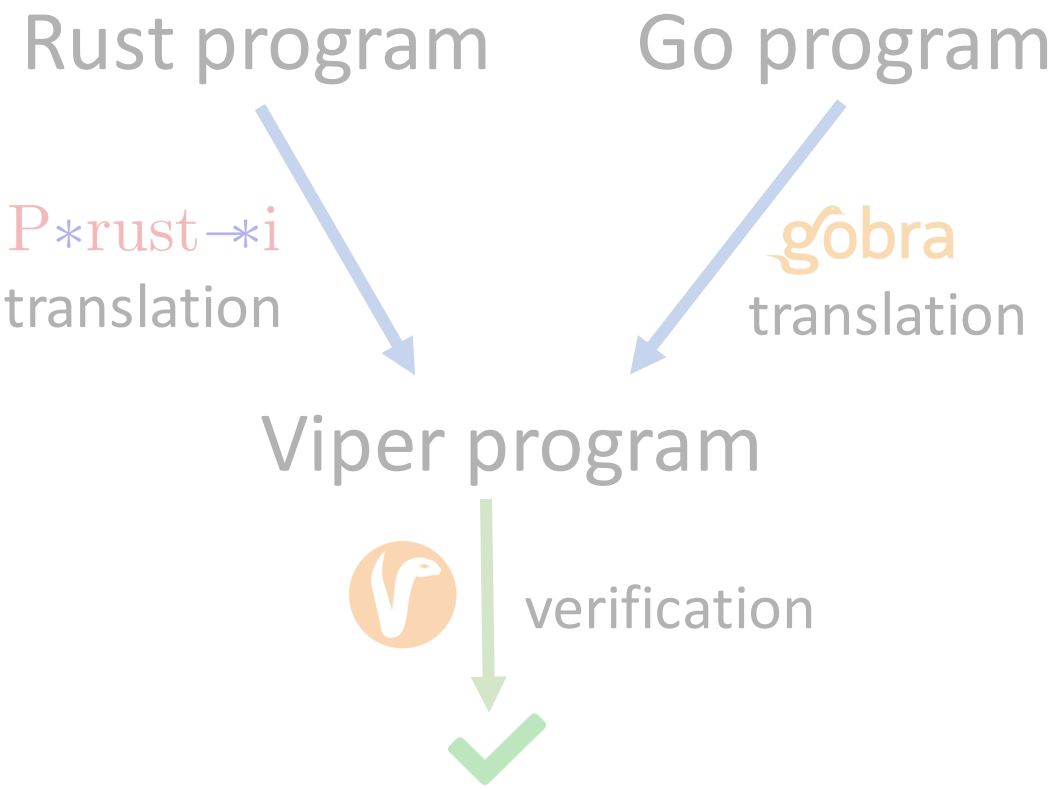
Create a DSL for expressing the **verification algorithm**

Lithium

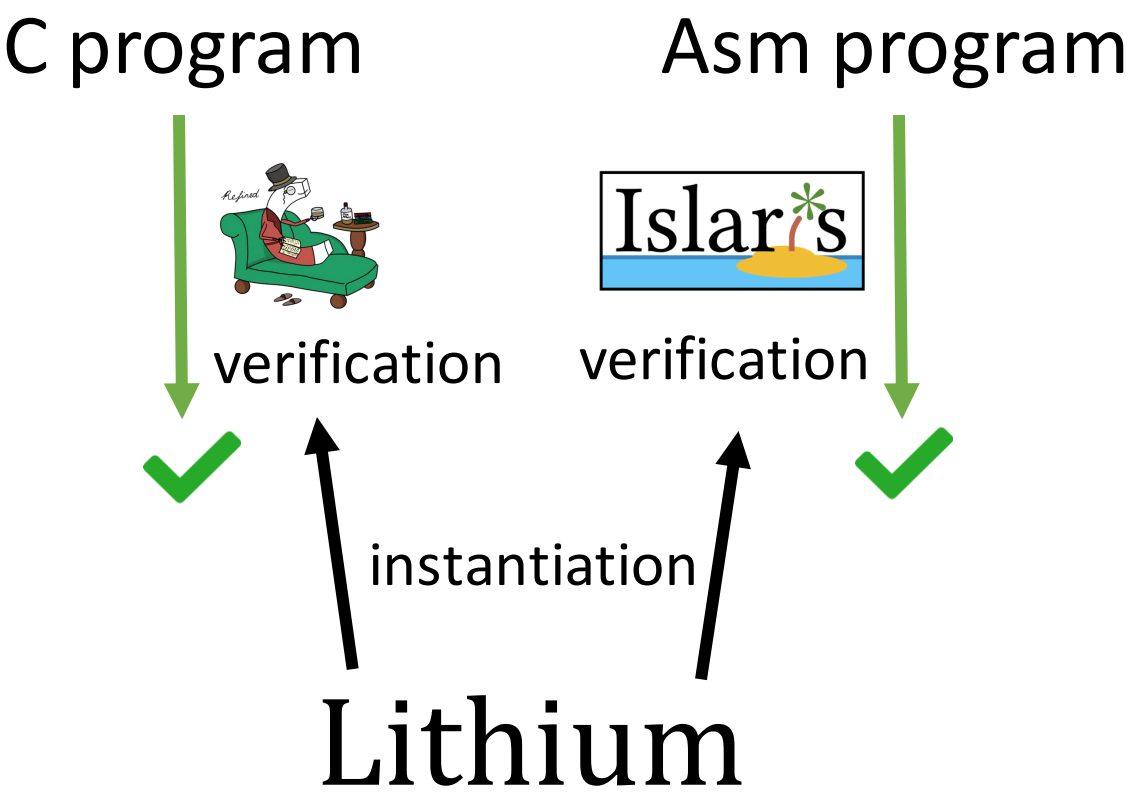
RefinedRust

Islaris

Common approach
Translate to an intermediate verification language



Key idea
Create a DSL for expressing the **verification algorithm**



Lithium

Atoms describe the assertions manipulated during verification (e.g. points-to predicate or type assignments)

$a \mapsto_M b$

$v \triangleleft_{\mathsf{v}} \&_{\mathsf{own}}(\tau)$

Functions encode the proof search procedure

Atom	$A ::= \ldots$
Function	$F ::= A_1 <: A_2 \mid \ldots$
Goal	$G ::= \mathbf{exhale}\ H; G \mid \mathbf{inhale}\ H; G \mid \forall x.\, G \mid \exists x.\, G \mid$ $\mathbf{done} \mid x \leftarrow F; G \mid \mathbf{return}_G\ x \mid \ldots$
Left-goal	$H ::= A \mid \ulcorner \phi \urcorner \mid H * H \mid \exists x.\, H(x) \mid \square H$

Provided by the user

Lithium primitives

proving and assuming assertions

(Lithium)

universal and existential quantifiers

Adaptable to the source language

Lithium

Shallowly embedded in the Coq proof assistant and based on Iris
⇒ **Foundational**

Comes with an interpreter
⇒ **Automated**

Let's build a verification tool with **Lithium**!

Thank you for your attention!

Interested in learning more?

Let's chat!

see also Part I of my thesis
(available at https://people.mpi-sws.org/~msammler/)

Lithium and the tutorial are distributed with RefinedC:

https://gitlab.mpi-sws.org/iris/refinedc/