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?
Boogie works well if there is a natural mapping from source language 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

Source language
assembly language with flat memory model

Proof search
based on points-to assertions

automated separation logic reasoning

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
Common approach
Translate to an intermediate verification language

Rust program  Go program
P_rust → i translation  gôbra translation
Viper program
verification

Key idea
Create a DSL for expressing the verification algorithm

C program  Asm program
verification  verification

Lithium
Lithium

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

\[ a \leftrightarrow^M b \]

Functions encode the proof search procedure

\[ v \triangleleft_v \&_{\text{own}}(\tau) \]

Provided by the user

Lithium primitives

Adaptable to the source language

Atoms: \[ A ::= \ldots \]
Function: \[ F ::= A_1 <: A_2 \mid \ldots \]
Goal: \[ G ::= \text{exhale } H; \ G \mid \text{inhale } H; \ G \mid \forall x. G \mid \exists x. G \mid \text{done } x \leftarrow F; \ G \mid \text{return}_G x \mid \ldots \]
Left-goal: \[ H ::= A \mid \exists x. H(x) \mid \square H \]

proving and assuming assertions
universal and existential quantifiers

(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!

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/