FORMAL VERIFICATION OF A LAZY ABSTRACTION MODEL CHECKER

Arthur Correnson
arthur.correnson@ens-rennes.fr

Lazy Abstraction Model Checking

Lazy Abstraction Model Checking is a verification technique to check that a program is safe to execute by exhaustively exploring all executions of an abstract model of the program. This model contains less information and is easier to verify. The model is incrementally refined during the verification process if it is found to abstract away too much information. We compute a model by encoding the semantics of the program as first-order formulas. Exploring the abstraction can then be reduced to satisfiability checking.

Goals

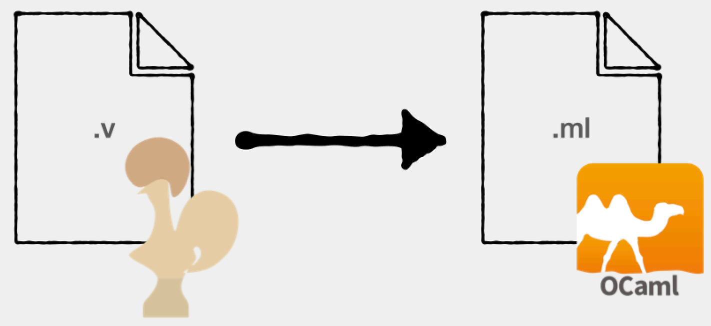
- Develop a lazy abstraction model checker that is:
  - reasonably efficient
  - formally proved to be correct

Method

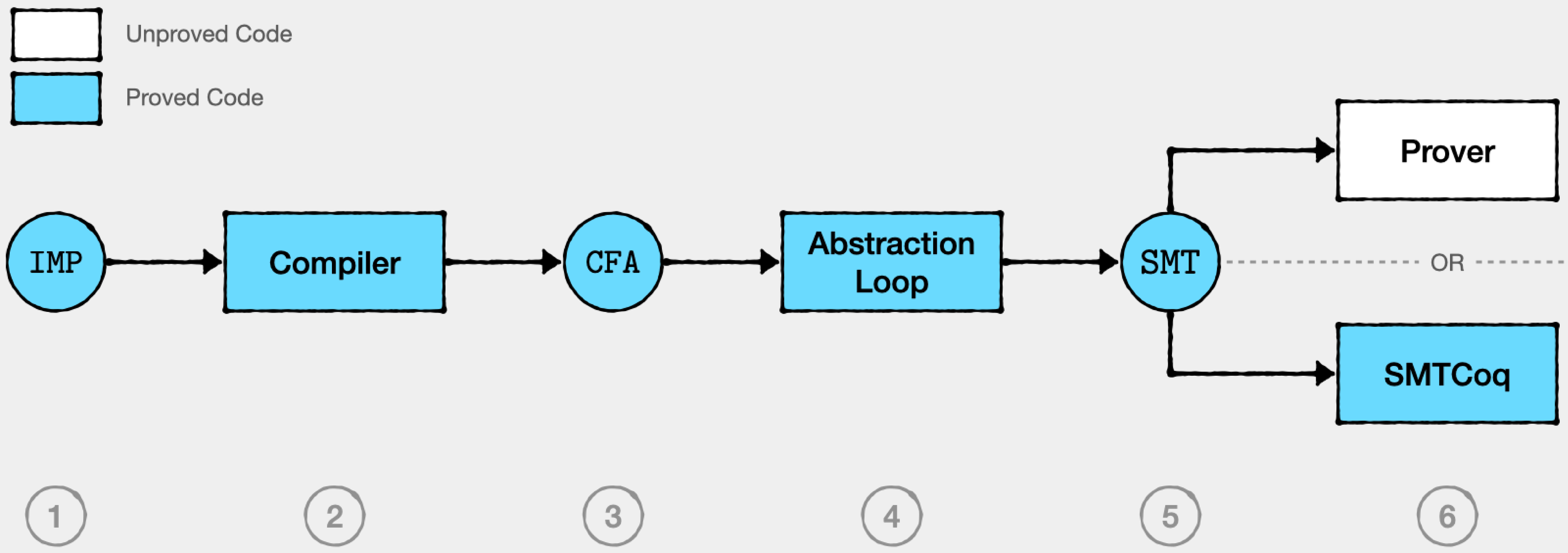
- Implementation of the model checker in the Coq proof assistant
- Formal proof of the model checker correctness. For any program p

$$\texttt{check}(p) = \texttt{Ok} \Rightarrow \forall (\pi : \texttt{path}), \neg \mathrm{ReachError}(p, \pi) \quad (1)$$
$$\texttt{check}(p) = \texttt{Error}(\pi) \Rightarrow \mathrm{ReachError}(p, \pi) \quad (2)$$

- The model checker is extracted into correct and executable OCaml code



Architecture of the Model Checker



(1) IMP Deep embedding of the source language (IMP) and formalization of its semantics.

(2) Compilation Semantics-preserving compiler from IMP to Control Flow Automaton (CFA). The proof is performed using simulation diagrams.

(3) CFA Implementation of Control Flow Automaton and formalization of their semantics.

(4) Abstraction Loop Exploration of an abstraction of the CFA. Generates first-order formulas and send requests to external provers.

(5) SMT Logic Deep embedding of a first-order logic and formalization of its semantics.

(6) Interface with provers Communication with external theorem provers. A certified validator (SMTCoq) can be used to verify the outputs of the provers at runtime.

A Purely Functional Algorithm

We model the internal state of the model checker as an immutable record `MC_state`. The progress of the model checker is reported by an algebraic datatype `MC_status`.

```
Inductive MC_status :=
  Next (_ : MC_state) | Error (_ : path) | Done.
```

A function `start` creates an initial state from a source program and a pure function `step` updates the internal state.

```
Definition start : IMP -> MC_state.
Definition step : MC_state -> MC_status.
```

Exploring the Abstraction

Given a set of states E and a program instruction i, $\mathrm{Post}(E, i)$ is the set of i-successors of E.

$$\mathrm{Post}(E, i) := \{s' \mid \exists s \in E, s \rightarrow_i s'\}$$

We implement an operator `post` that is proved to compute an over-approximation of Post for any instruction i and set of states encoded as a formula φ.

$$\mathrm{Post}([\![\varphi]\!], i) \subseteq [\![\texttt{post}(\varphi, i)]\!]$$

The `post` operator is called by the `step` function to explore the abstract model of the program.

Maintaining Invariants

The correctness of the model checking algorithm is ensured by means of invariants. We gather all the required invariants into a predicate Inv over `MC_state`. The initialization and iteration functions are proved to maintain the invariants:

$$\mathrm{Inv}(\texttt{start}(p)) \quad (3)$$
$$\mathrm{Inv}(s) \Rightarrow \texttt{step}(s) = \texttt{Next}(s') \Rightarrow \mathrm{Inv}(s') \quad (4)$$

Safety of the source program is proved to follow from Inv:

$$\mathrm{Inv}(s) \Rightarrow \texttt{step}(s) = \texttt{Done} \Rightarrow \forall (\pi : \texttt{path}), \neg \mathrm{ReachError}(p, \pi)$$

Similarly, we prove that errors are correctly reported:

$$\mathrm{Inv}(s) \Rightarrow \texttt{step}(s) = \texttt{Error}(\pi) \Rightarrow \mathrm{ReachError}(p, \pi)$$

Termination over Completeness

- We iterate the function `step` for a fixed number n of iterations
- A `Timeout` message is raised if the model checking does not terminate after n steps
- Termination is then enforced but we lose completeness

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY
UNIVERSITÄT DES SAARLANDES
ENS rennes école normale supérieure