Counter-example Guided Abstraction Refinement

(IF311 : "Formal methods in software design")

Frédéric Herbreteau (fh@labri.fr)

Bordeaux INP

Motivation

Big models are out of the scope of model-checking:

- variables that range over **huge domains** (32-bits integers, ...)
- variables that are **unbounded** (integers $\mathbb{N}, \mathbb{Z}$, arrays $[0, n] \to E$ for $n \in \mathbb{N}$, ...)

Goal: build an **abstract** model that allows to prove the **actual** algorithm, **automatically**

Example: verification of a C program (1/2)

Example

Check that `lock()` is never blocking (i.e. the lock is free when the program needs to get it)

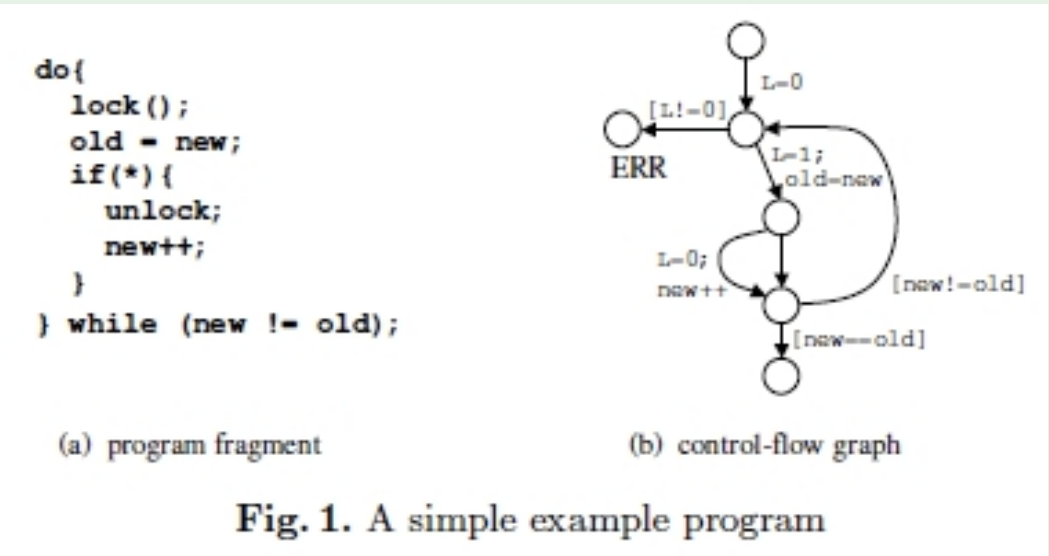


Fig. 1. A simple example program

source: R. Jhala and K.L. McMillan, *Lazy Abstraction with Interpolants.*, Computer-Aided Verification, 2006

Example: verification of a C program (2/2)

Example

```
--algorithm Concrete {
  variables old \in 0 .. 4294967295; (* 32bits pointers *)
            new \in 0 .. 4294967295;
            l = FALSE;

  macro lock()   { await (l = FALSE); l := TRUE }
  macro unlock() { l := FALSE }

  {
l1: while (new /= old) {
l2:   assert( l = FALSE );  (* The lock is free *)
l3:   lock();
      old := new;
      either {
l4:     unlock();
        new := new+1
      }
      or { skip }
    }
  }
}
```

TLC "cannot handle a **number this big**" 4294967295

Outline

Abstraction

Refinement

Automatization: the CEGAR approach

Plan

Abstraction

Refinement

Automatization: the CEGAR approach

Predicate abstraction

Goal: from a model M, with semantics T, define a finite and small abstract semantics $\widehat{T}$ of M s.t. $T \preceq \widehat{T}$

- Given a set P of **predicates**, define $\sim_P$ as:

$$s \sim_P s' \qquad \text{iff} \qquad \forall p \in P.\ s \models p \Leftrightarrow s' \models p$$

Definition

The **predicate abstraction** of T is the transition system $\widehat{T_{\sim_P}}$

- Predicates are **defined** from the **variables** in the model (PlusCal, C program,...)
- A **decision procedure** is needed to know if a state $s \models p$ (s in the semantics of the program)

Example: verification of a C program (...)

Example

Let $P = \emptyset$, and let **not abstract** `pc`

- How many states are there in $\widehat{T_{\sim P}}$? How many of them are **reachable**?
- Give states s_1, s_2 and s_3 s.t. $s_1 \sim_P s_2$, but s.t. $s_1 \not\sim_P s_3$
- Build $\widehat{T_{\sim P}}$

(See : `abstract1.tla`)

Initial coarse abstraction

Example

```
Failure of assertion

Counter example:

Initial predicates
  pc = l1

Action line 11, col 7
  pc = l2
```

- **Initial abstraction** $\widehat{T_{\sim\emptyset}}$: control-flow graph

  (See : `mcmillan-cav06-abstract1.smv`)

- The assertion is violated by $\widehat{T_{\sim\emptyset}}$, what can be deduced for T?

- What can be **deduced** from the **counter-example**?

Plan

Abstraction

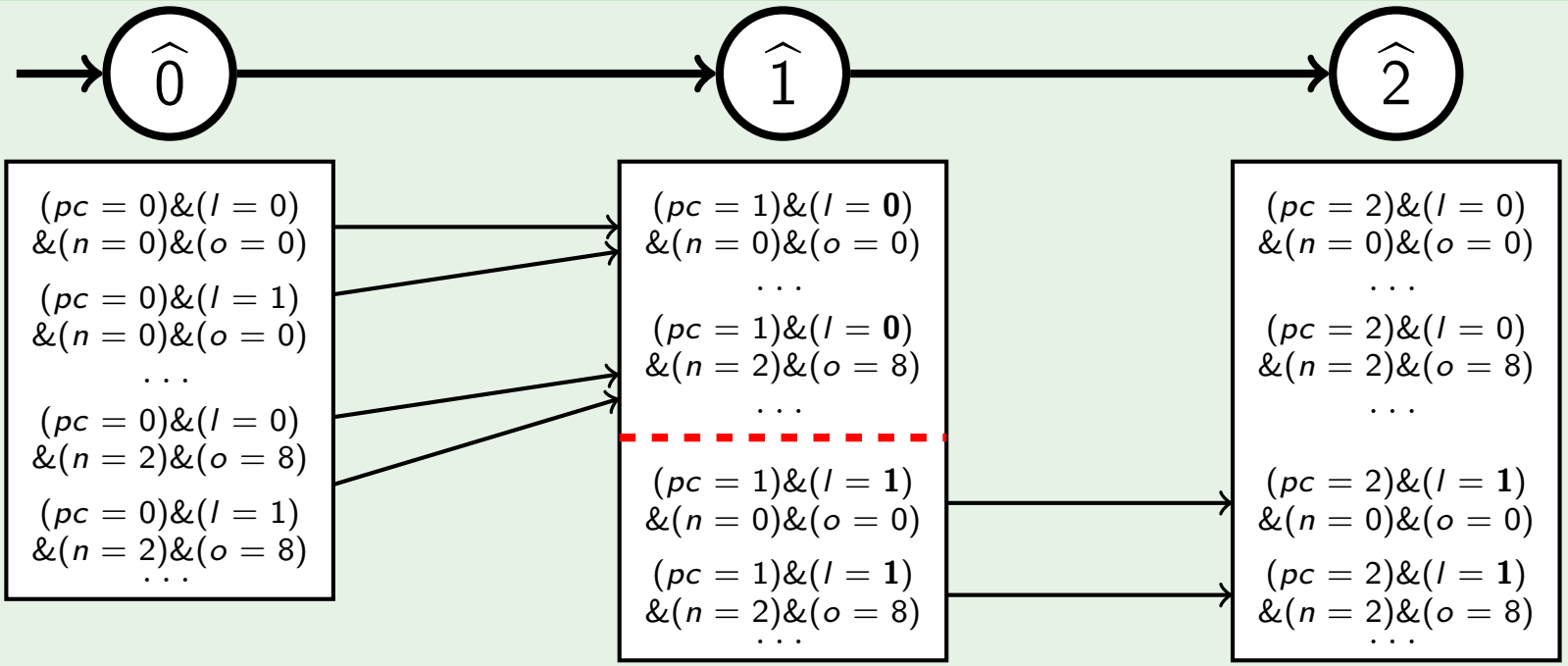
Refinement

Automatization: the CEGAR approach

A spurious counter-example

- The abstract state $\widehat{1}$ can be splitted in two **disjoint parts**: a **reachable** one and a **co-reachable** one

Example



- Why is there a **spurious counter-example**?

Interpolation

Definition

An **interpolant** of inconsistent formulas A and B is a formula Υ satisfying:

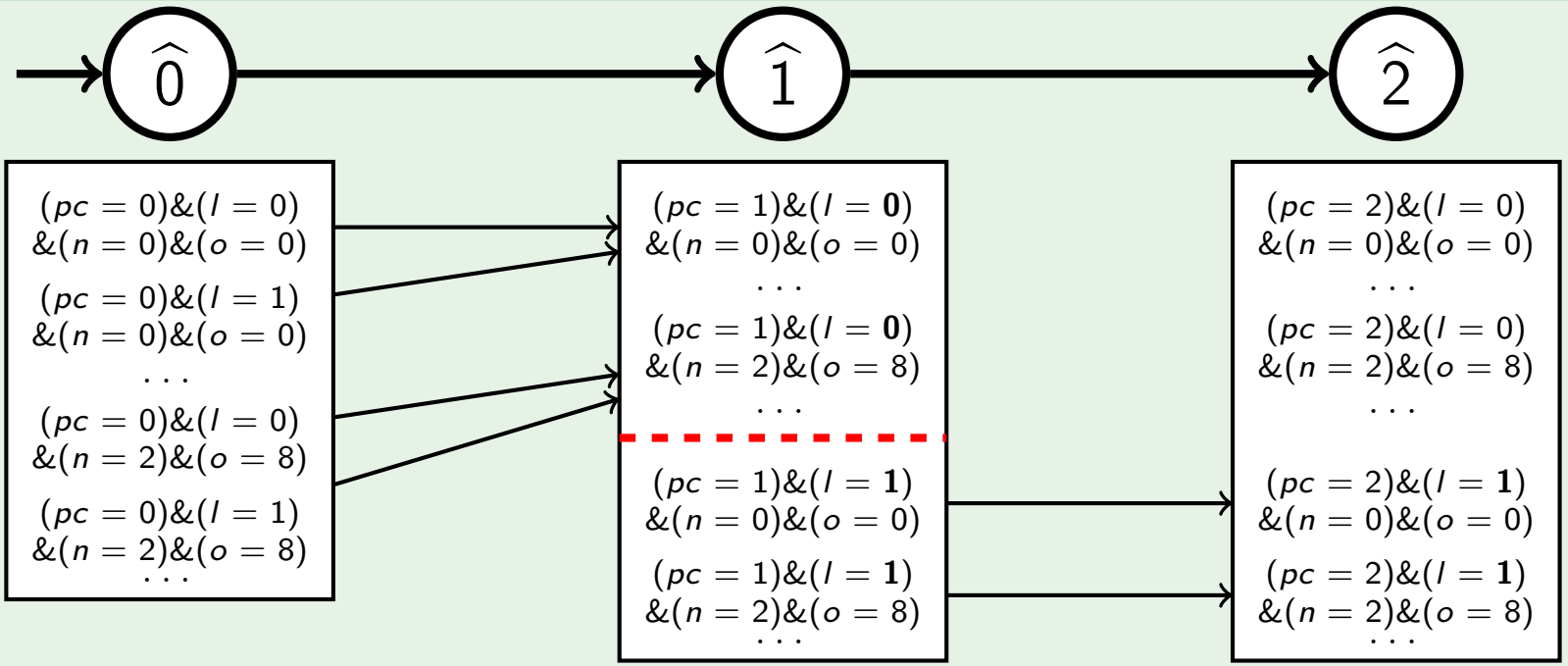
- $A \Rightarrow \Upsilon$ and $\Upsilon \wedge B \equiv \mathit{FALSE}$
- and $\mathit{Var}(\Upsilon) \subseteq \mathit{Var}(A) \cap \mathit{Var}(B)$

Theorem

Any two inconsistent **first-order** formulas A and B admit an interpolant (Craig's theorem).

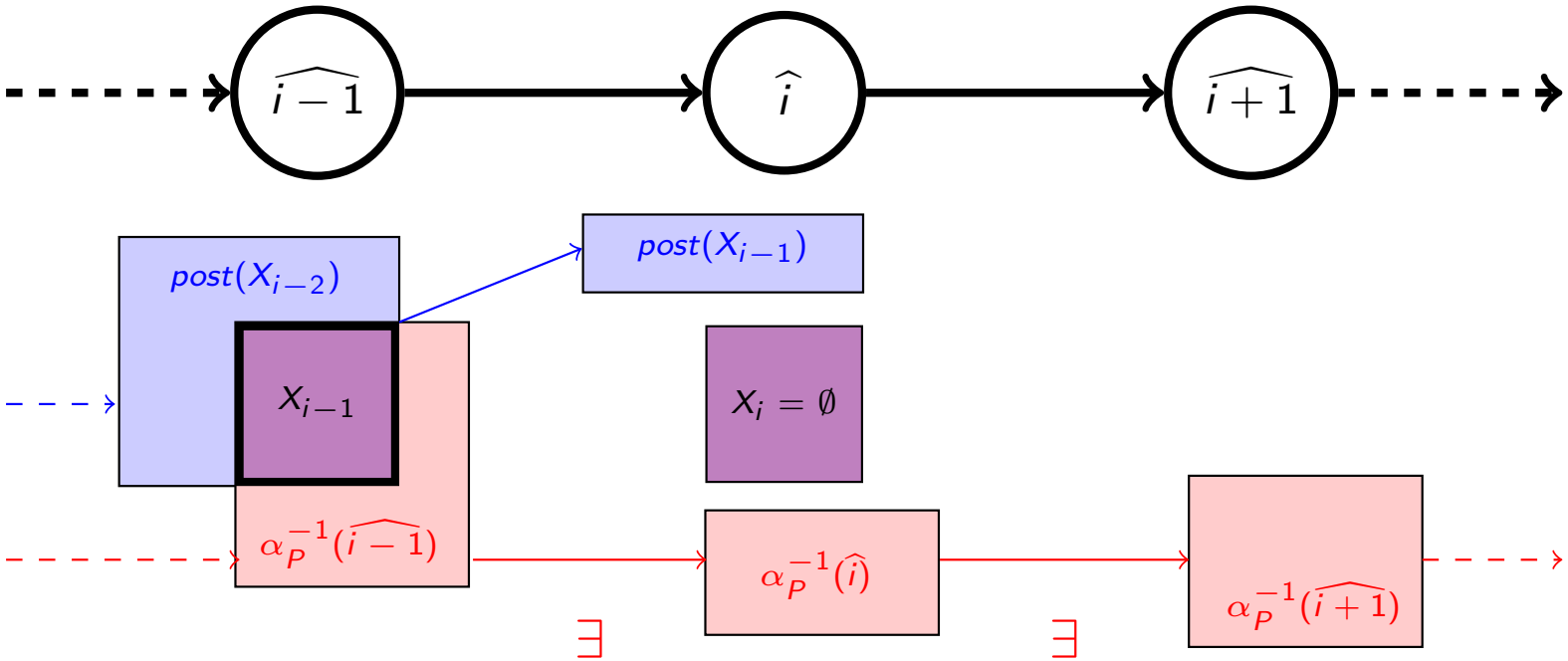
Refinement: an example

Example



- Compute an **interpolant** of $Post(\widehat{0}) \cap \widehat{1}$ and $Pre(\widehat{2}) \cap \widehat{1}$
- How can it be used to **refine** the abstract model?

Refinement: the properties



Υ **interpolant** of $post(X_{i-1})$ and $\alpha_P^{-1}(\widehat{i})$

Theorem

- $T \preceq \widehat{T_{\sim P \cup \{\Upsilon\}}} \preceq \widehat{T_{\sim P}}$
- The **spurious counter-example** is not a run of $\widehat{T_{\sim P \cup \{\Upsilon\}}}$

Example: verification of a C program (...)

Example

```
Failure of assertion

Counterexample:

Initial predicates
  pc = l1, l_is_FALSE = TRUE

Action line 12
  pc = l2, l_is_FALSE = TRUE

Action line 17
  pc = l3, l_is_FALSE = TRUE

Action line 22
  pc = l1, l_is_FALSE = FALSE

Action line 12
  pc = l2, l_is_FALSE = FALSE
```

- Compute the **refined** model for $\Upsilon \equiv (l = \mathit{FALSE})$ (`abstract2.tla`)
- Show that the new counter-example to the left is **spurious**
- Propose a **new interpolant** Υ' and build $\widehat{T_{\sim_{P \cup \{\Upsilon, \Upsilon'\}}}}$ (`abstract3.tla`)
- What can be **concluded**? What is the **size** of $\widehat{T_{\sim_{P \cup \{\Upsilon, \Upsilon'\}}}}$?

Plan

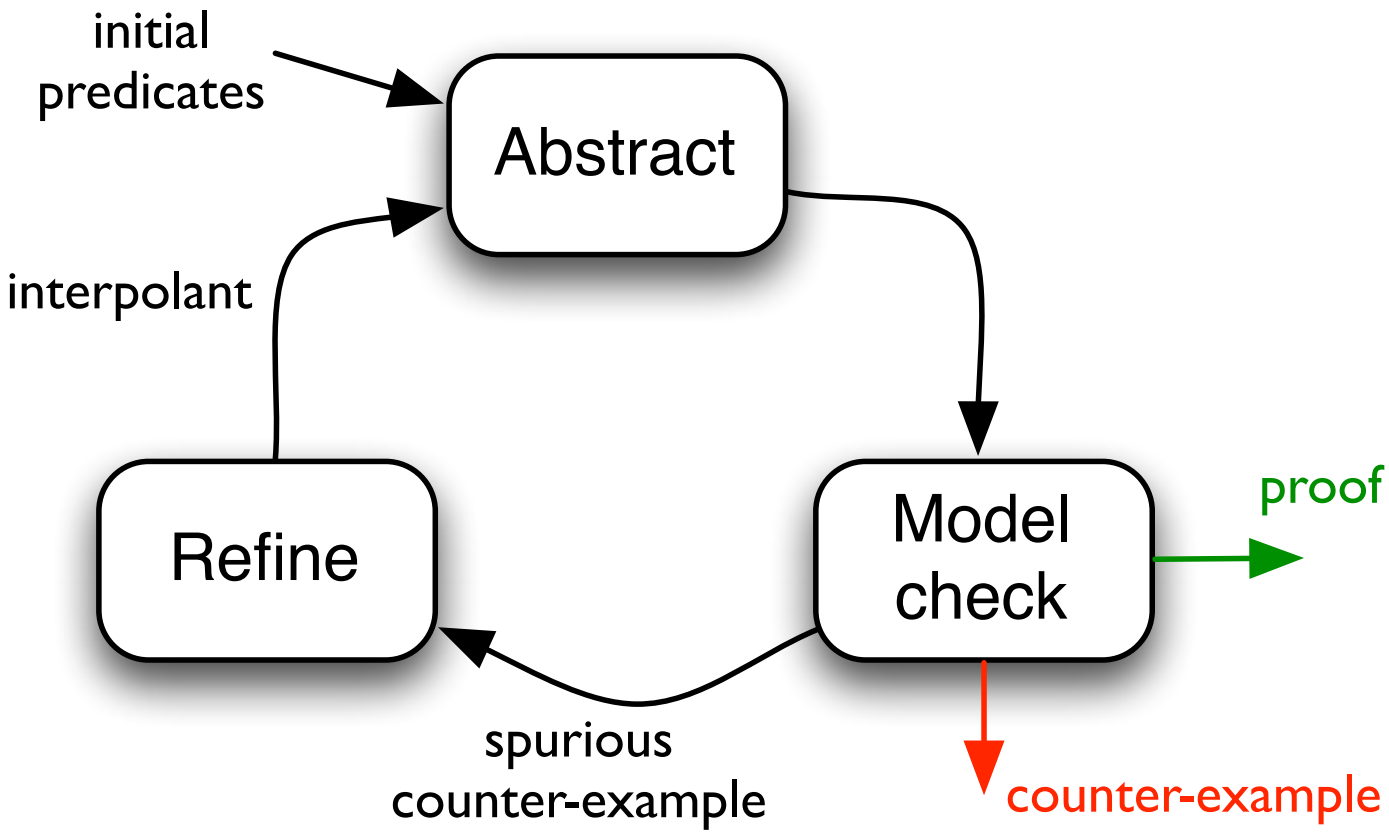
Abstraction

Refinement

Automatization: the CEGAR approach

The CEGAR Approach

CEGAR= CounterExample-Guided Abstraction Refinement



May not terminate (program verification undecidable)

Some tools that implement CEGAR

- **SLAM**, Microsoft (`http://research.microsoft.com/en-us/projects/slam/`), used for the verification of the windows drivers
- **BLAST**, UCLA/EPFL (`http://mtc.epfl.ch/software-tools/blast/index-epfl.php`), verification of C programs, introduced lazy abstraction
- **Moped**, U. Stuttgart (`http://www.fmi.uni-stuttgart.de/szs/tools/moped/`), verification of Java programs

Reference : E.M. Clarke, O. Grumberg, S. Jha, Y. Lu and H. Veith, *Counterexample-guided abstraction refinement for symbolic model checking*, JACM 50(5): 752-794, 2003.

Conclusion

- **simulation** and **bisimulation** relations are powerful tools to scale up model-checking to large/infinite models
- CEGAR allows to compute simulation relations **automatically**
- Predicate **abstraction** is one technique among others (domains of values, cartesian abstraction, lazy interpolation,...)
- **Interpolation** is algorithmically hard. Active field of **research** (SAT/SMT solving, language-theory, graphs of counter-examples,...)