Counter-Example Generation in Symbolic Abstract Model-Checking

Gordon Pace, Nicolas Halbwachs & Pascal Raymond
Vérimag, Grenoble, France

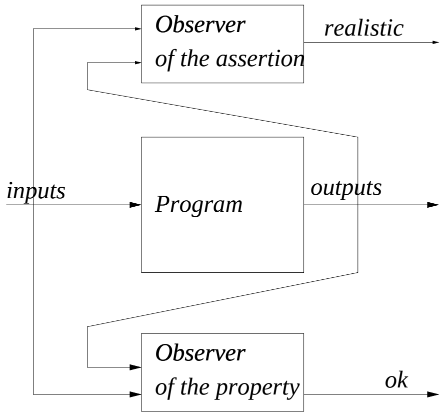
July 2001

Motivation

Model checking techniques work well in practice, but problems arise when ...

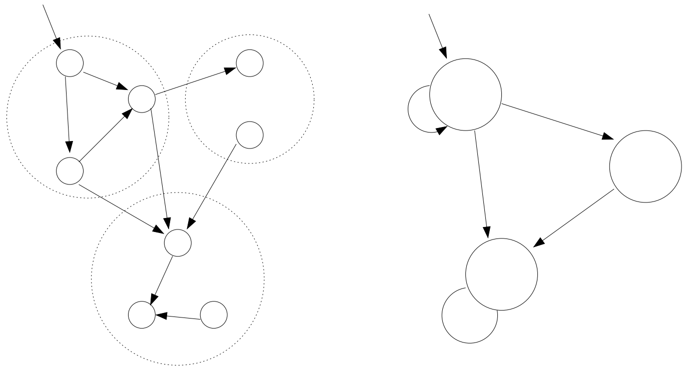
- The model does not satisfy the property being verified, and
- we want to see counter-examples, but
- abstraction is used, or
- some information is not to be seen by the user.

Lesar



Typical Verification Picture

Concrete model (C) Abstract model (A)



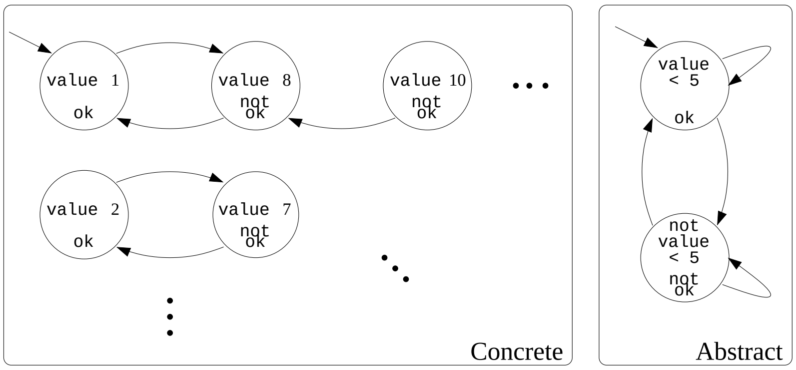
Big, or possibly infinite Smaller, and finite

Satisfy: $\mathit{safe}(A) \Rightarrow \mathit{safe}(C)$

But not: $\mathit{safe}(A) \Leftrightarrow \mathit{safe}(C)$

Example

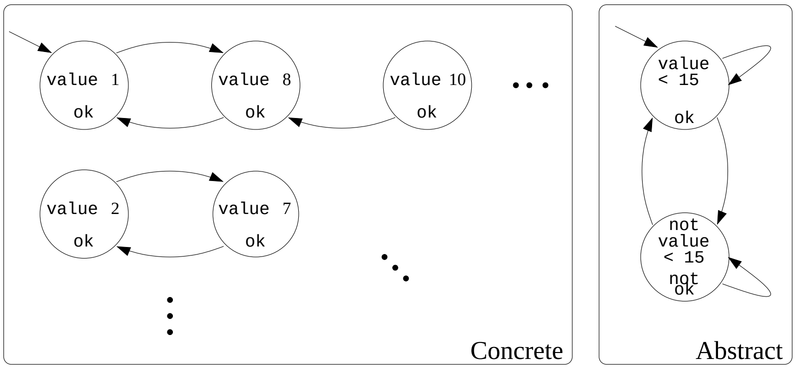
```
value = 1 -> (17*pre value) mod 9
ok    = value < 5
```



Counter-example in the abstract model, with a corresponding concrete one.

Example

```
value = 1 -> (17*pre value) mod 9
ok    = value < 15
```



Counter-example in the abstract model, but no concrete one.

What to do with an abstract counter-example

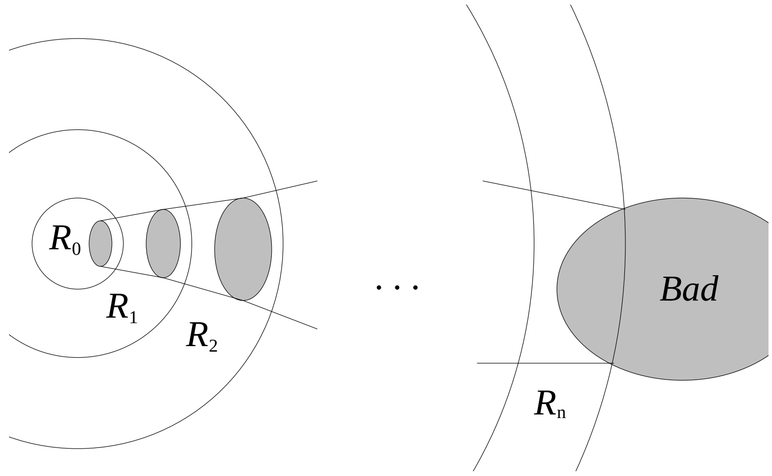
- Are there any concrete counter-examples corresponding to an abstract one?
- If so, can we generate them?
- If none exist, are there longer abstract counter-examples with corresponding concrete counter-examples?

What has already been done

Clarke et al, Lakhnech et al: The inverse of the abstraction function (or an approximation), and trace-set emptiness are computable.

Păsăreanu et al: Searching for deterministic abstract counter-examples, and using simulation on the (closed) model.

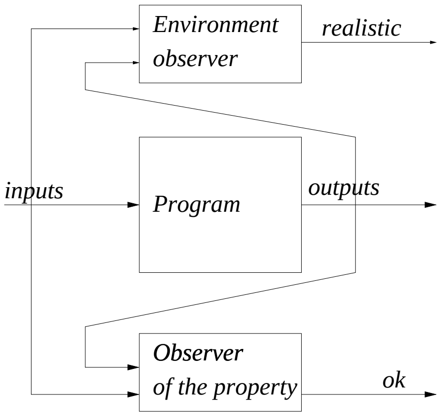
Ball et al: Use path simulation with heuristics.



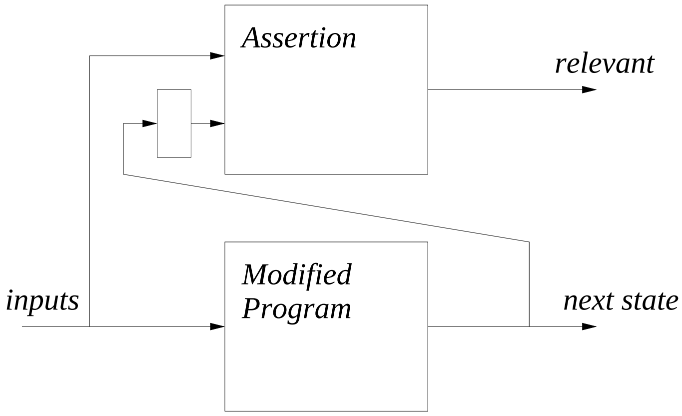
Approach

- Remove calculation of counter-examples from model-checker,
- do not lose information (eg throw away information that the user knows nothing about)
- We just keep the information $D_i = R_i \cap Bad$

Lurette



Lurette



Unabstracted programs

```
vars concrete;

step     = 0 -> 1+pre step
concrete = program(input,pre concrete)
relevant = case step of
             0: D0(input,concrete)
             1: D1(input,concrete)
              ...
             n: Dn(input,concrete)
             default: true
```

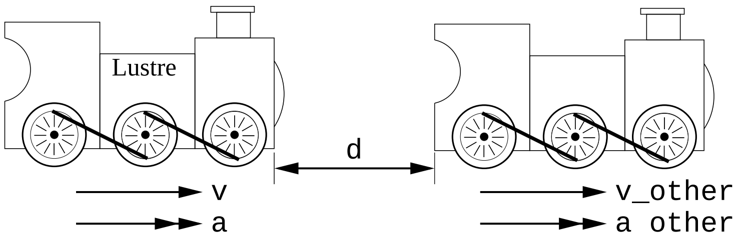
Abstracted programs

```
vars concrete, abstract;

step        = 0 -> 1+pre step
abstraction = abstraction_relation(abstract, concrete)
concrete    = program(input,pre concrete)
relevant    = abstraction and
                case step of
                  0: D0(input,abstract)
                  1: D1(input,abstract)
                   ...
                  n: Dn(input,abstract)
                  default: true
```

Example

```
value = 1 -> (17*pre value) mod 9
ok    = value < 15

step        = 0 -> 1+pre step
abstraction = (condition == value < 15)
value       = 1 -> (17*pre value) mod 9
relevant    = abstraction and
                case step of
                  0: condition
                  1: not condition
                  default: true
```

Another example [Bensalem et al, 1999]




```
danger = (pre v + a_max >= brake)
      or (pre v + a_max >  init_d ->pre d)

a = if danger then -brake else a_other
v = max (0, a + pre v)
d = v_other - v + (init_d -> pre d)

ok = d > 0 and v <= brake
```

Another example: Assertions

```
assertions:
  brake  >= 0
  init_d >  0
  brake  >  a_max
  v_other == 0 -> v_other >= 0
```

Another example: Counter-Example

```
a_max   = 1
brake   = 2
init_d  = 1
a_other = 2
v_other = 0
```

The other train is exceeding the accelleration limit!

Ariane V Application

- Medium sized model of control the flight control software.
- 40 state variables including numerical variables.
- Counter-examples typically of length 20-30 were found,
- from which it was clear that the problems lay with weak environment assertions.
- With large models, the constraint solver usually blew up before the model-checker had reached its limits.

Critique

- (+) No counter-example information is lost via approximations or hiding of information.
- (+) Using information *already calculated* by the model checker and
- (+) with test input generation tools that *already exist.*
- (+) Heuristics for path feasability left up to the test tool (their proper place).

Critique

- (−) Obviously, still incomplete.
- (−) Even if no concrete counter-examples of length n are found, longer ones may still exist.
- (−) The method works best with systems where the abstracted data does not have a large effect on the property.

Conclusion and Pending Questions

- Concrete counter-example generation can be reduced to a constraint solving, or test input generation problem.
- Which heuristics for test input generation algorithms work best for this problem?
- What to do with the 'longer concrete counter-examples may exist' problem?