Bounded Model Checking

PALLAB DASGUPTA
FNAE, FASc, FIETE,
Professor,
Dept of Computer Science & Engineering
Indian Institute of Technology Kharagpur
Email: pallab@cse.iitkgp.ac.in
Web: http://cse.iitkgp.ac.in/~pallab

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

FMSAFE
FORMAL METHODS FOR SAFETY CRITICAL SYSTEMS

BMC outline

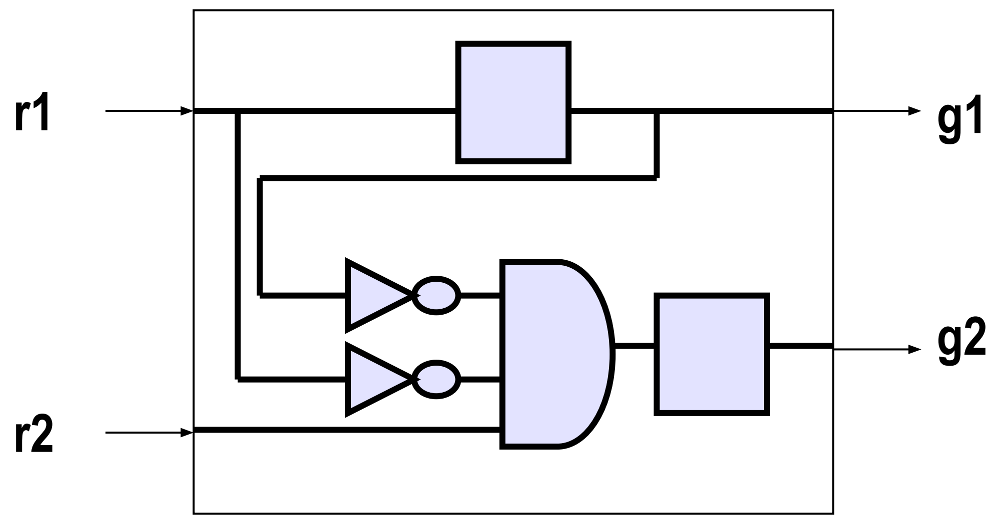
Given:

- The specification. For example, a property in formal logic.
- The design, as a finite state machine.
- A bound, *k*, on length of a run.
    - In bounded model checking, only runs of bounded length *k* or less are considered.

- Translation to SAT:
    - We unfold the negation of the property into Boolean clauses over different time steps
    - We unfold the state machine into Boolean clauses over the same number of time steps
    - We check whether the clauses are together satisfiable

Example: *Priority Arbiter*

Implementation:



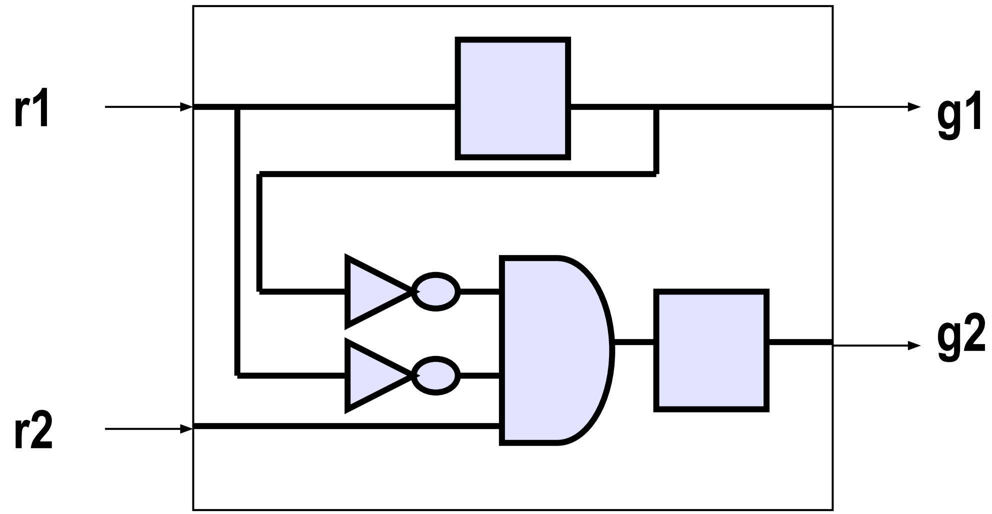
Initial state: g1=0, g2=1

Specification:

- When r1 is high, g1 must be asserted for the next two cycles
- In Linear Temporal Logic: $G(r1 \Rightarrow Xg1 \wedge XXg1)$
- In SystemVerilog Assertion (SVA): r1 |-> ##1 g1 ##1 g1

Example: *Priority Arbiter*

Implementation:



<u>**Transition Relation**</u>:

g2' □ r2 ∧ ¬ r1 ∧ ¬ g1

g1' □ r1

Initial state: g1=0, g2=1

Specification:

- In Linear Temporal Logic:
  G(r1 ⇒ Xg1 ∧ XXg1)
- In SystemVerilog Assertion (SVA):
  r1 |-> ##1 g1 ##1 g1

Negation of specification (counter-example):

- In Linear Temporal Logic: F(r1 ∧ (¬Xg1 ∨ ¬XXg1))
- In SVA, we look for: (r1 ##1 !g1) or (r1 ##2 !g1)

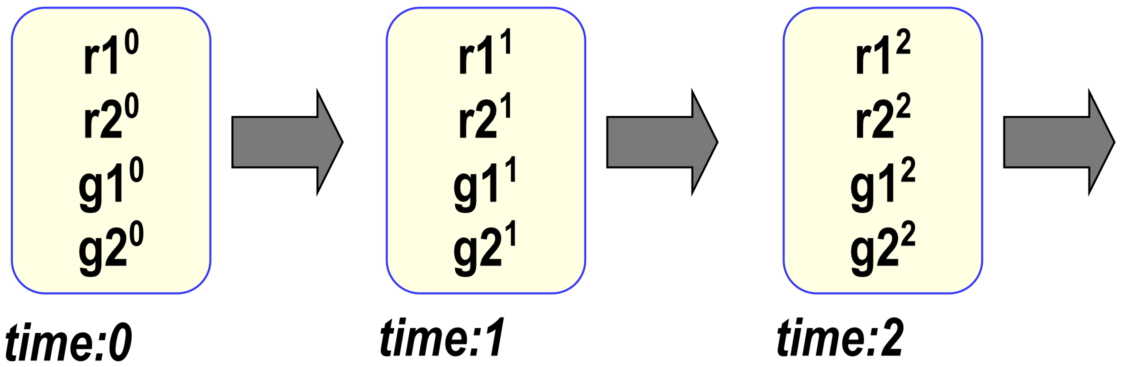
Strategy: Unfold transition relation one step at a time and check whether a counterexample exists

Variables in Temporal Worlds

Negation of specification:

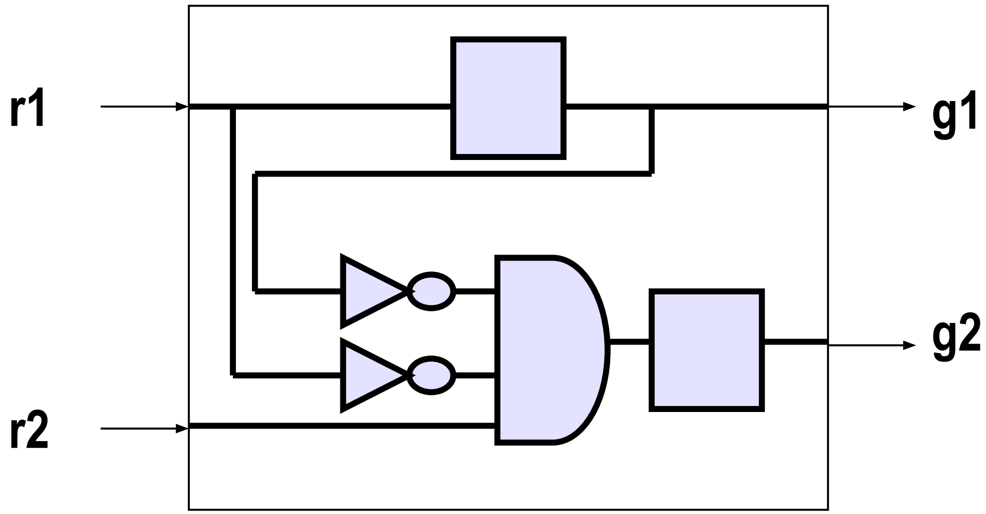
- In Linear Temporal Logic: $F(r1 \wedge (\neg Xg1 \vee \neg XXg1))$
- In SVA: (r1 ##1 !g1) or (r1 ##2 !g1)

<u>Variable naming convention</u>



$$\forall t \, [\, r1^t \wedge \neg\, g1^{t+1} \wedge \neg\, g1^{t+2} \,]$$

Iteration-1: *Bound* = 2



Negated Property: (r1 ##1 !g1) or (r1 ##2 !g1)

Is there a counter-example of length = 2?

Clauses from Transition Relation:

C_1^1: $r2^0 \wedge \neg r1^0 \wedge \neg g1^0 \Rightarrow g2^1$

C_2^1: $r1^0 \Rightarrow g1^1$

Clauses from Initial State:

I: $g2^0 \wedge \neg g1^0$

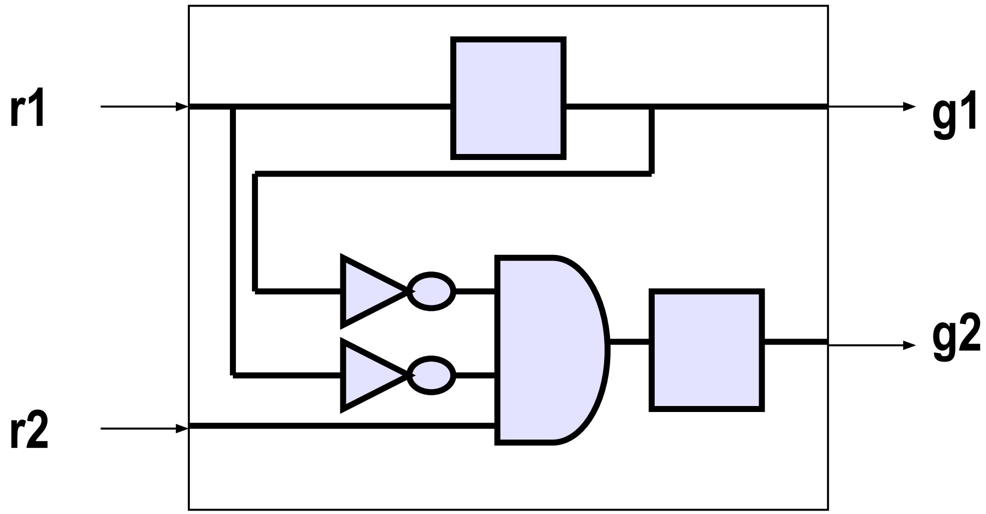
Clauses from Negated Property:

Z^1: $r1^0 \wedge \neg g1^1$

SAT Check: Is $Z^1 \wedge I \wedge C_1^1 \wedge C_2^1$ satisfiable?

Answer: No, since Z^1 conflicts with C_2^1

Iteration-2: *Bound* = 3



Negated Property: (r1 ##1 !g1) or (r1 ##2 !g1)

Is there a counter-example of length = 3?

Clauses from Transition Relation:

C_1^1: $r2^0 \wedge \neg r1^0 \wedge \neg g1^0 \Rightarrow g2^1$

C_2^1: $r1^0 \Rightarrow g1^1$

C_1^2: $r2^1 \wedge \neg r1^1 \wedge \neg g1^1 \Rightarrow g2^2$

C_2^2: $r1^1 \Rightarrow g1^2$

Clauses from Initial State:

I: $g2^0 \wedge \neg g1^0$

Clauses from Negated Property:

Z^2: $(r1^0 \wedge (\neg g1^1 \vee \neg g1^2)) \vee (r1^1 \wedge \neg g1^2)$

SAT Check: Is $Z^2 \wedge I \wedge C_1^1 \wedge C_2^1 \wedge C_1^2 \wedge C_2^2$ satisfiable?

Yes: Witness: $r1^0 = 1$, $r1^1 = 0$, $g1^1 = 1$, $g1^2 = 0$, rest are don't cares

Conclusion: We have found a counter-example!!

BMC is a bug hunting method

- We are checking only for bounded paths (paths which have at most k+1 distinct states)
  - So if the property is violated by only paths with more than k+1 distinct states, we would not find a counter-example using bounded model checking
  - If we do not find a counter-example using bounded model checking we are not sure that the property holds

- However, if we find a counter-example, then we are sure that the property is violated since the generated counter-example is never spurious (that is, it is always a concrete counter-example)

Formal Methodology

- Bound on path length *k*
- Clauses describing the design, M :
  - Initial state: $I(s_0)$
  - Unrolled transition relation: $\Lambda_{i=0..k-1}\ \rho(s_i, s_{i+1})$
- Loop clause: $loop_k = V_{i=0..k}\ \rho(s_k, s_i)$
- $[\ f\ ]_{i,k}$ means that (negated) property *f* is true at state s_i
- For a counter-example to exist on the design, $(\ M\ \Lambda\ [\ f\ ]_{i,k}\ )$ must be satisfiable

Translation of properties to clauses – *some basic forms*

$[f]_{i,k}$ means sequence *f* is true at state s_i

##1 f is true at state s_i of a run iff sequence f matches from s_{i+1} on that run. Formally:

$$[\#\#1\ f]_{i,k} = (i < k) \wedge [f]_{i+1,k}$$

##[0:m] f is true at state s_i of a run iff sequence f matches from some future state s_j within k steps. Formally:

$$[\#\#[0:m]\ f]_{i,k} = \vee_{j=i..m} [f]_{j,k}$$

f[*0:m] is true at state s_i of a run iff sequence f matches from all states reachable in k iterations and the run loops

$$[f[*0:m]]_{i,k} = \wedge_{j=i..m} [f]_{j,k} \wedge loop_k \qquad \text{where} \qquad loop_k = \vee_{i=0..k}\ \rho(s_k, s_i)$$

These are recursive formulations, allowing the translation of complex sequence expressions