# Bounded Model Checking



[BiereCimattiClarkeZhu99]

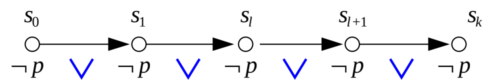
- uses SAT for model checking
  - historically not the first symbolic model checking approach
  - scales better than original BDD based techniques
- mostly incomplete in practice
  - validity of a formula can often not be proven
  - focus on counter example generation
  - only counter example up to certain length (the bound $k$) are searched



# Bounded Model Checking Safety



checking safety property $\mathbf{G}p$ for a bound $k$ as SAT problem:




$$I(s_0) \wedge T(s_0, s_1) \wedge \cdots \wedge T(s_{k-1}, s_k) \wedge \bigvee_{i=0}^{k} \neg p(s_i)$$

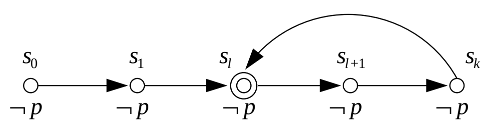
check occurrence of $\neg p$ in the first $k$ states



# Bounded Model Checking Liveness



generic counter example trace of length $k$ for liveness $\mathbf{F}p$



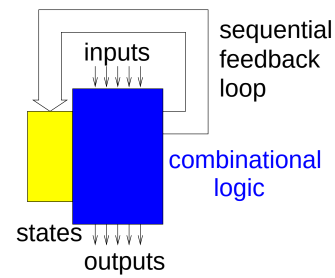

$$I(s_0) \wedge T(s_0, s_1) \wedge \cdots \wedge T(s_k, s_{k+1}) \wedge \bigvee_{l=0}^{k} s_l = s_{k+1} \wedge \bigwedge_{i=0}^{k} \neg p(s_i)$$

(however we recently showed that liveness can always be reformulated as safety [BiereArthoSchuppan02])



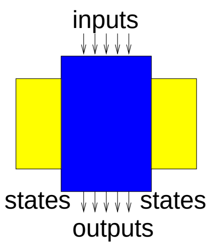
# Time Frame Expansion in HW






sequential circuit

# Time Frame Expansion in HW

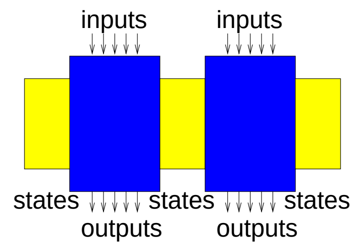



break sequential loop





# Time Frame Expansion in HW

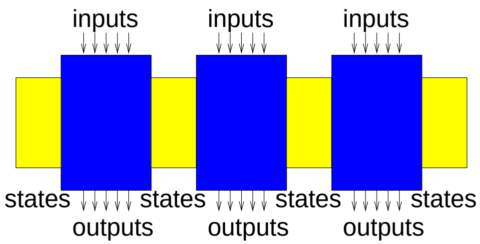



added 1st copy





# Time Frame Expansion in HW

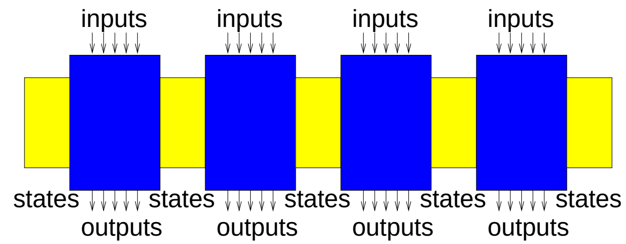



added 2nd copy





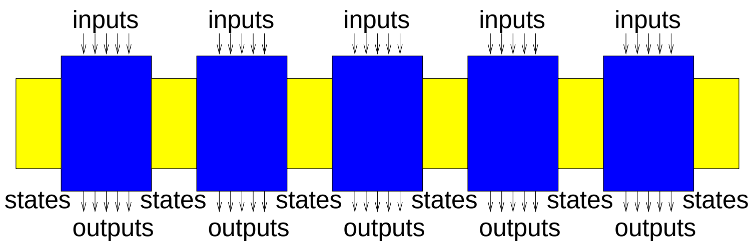
# Time Frame Expansion in HW




added 3rd copy

# Time Frame Expansion in HW



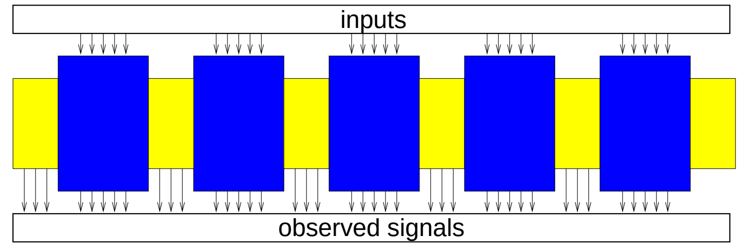



added 4th copy



# Time Frame Expansion in HW

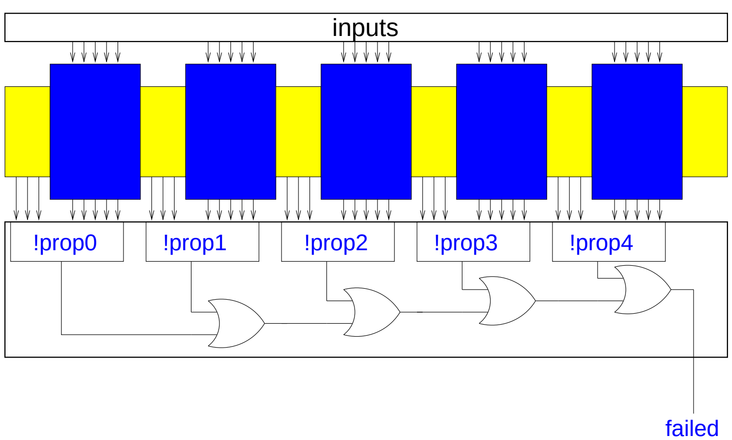







# Bounded Model Checking Safety in HW



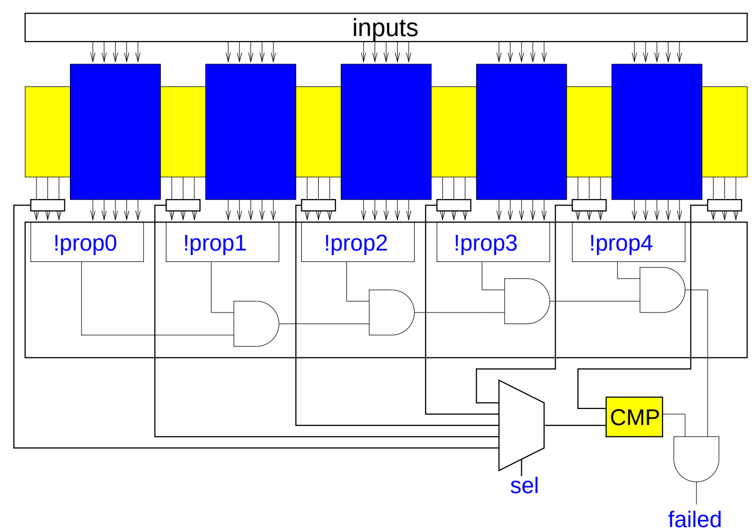



find inputs for which failed becomes true



# Bounded Model Checking Liveness in HW






find inputs for which failed becomes true

# Completeness in Bounded Model Checking



- find bounds on the maximal length of counter examples
  - also called **completeness threshold**
  - exact bounds are hard to find $\Rightarrow$ approximations
- induction
  - use inductive invariants as we have seen before
  - generalization of inductive invariants: **pseudo induction**
- use SAT for quantifier elimination as with BDDs (later)
  - then model checking becomes fixpoint calculation



# Measuring Distances



**Distance:** length of shortest path between two states

$$\delta(s,t) \equiv \min\{n \mid \exists s_0,\ldots,s_n[s = s_0, t = s_n \text{ and } T(s_i,s_{i+1}) \text{ for } 0 \le i < n]\}$$

(distance can be infinite if $s$ and $t$ are not connected)

**Diameter:** maximal distance between two connected states

$$d(T) \equiv \max\{\delta(s,t) \mid T^*(s,t)\}$$

with $T^*$ defined as the transitive reflexive hull of $T$.

**Radius:** maximal distance of a reachable state from the initial states

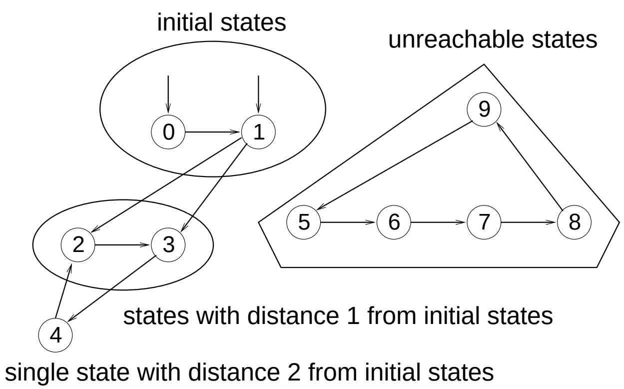
$$r(T,I) \equiv \max\{\delta(s,t) \mid T^*(s,t) \text{ and } I(s) \text{ and } \delta(s,t) \le \delta(s',t) \text{ for all } s' \text{ with } I(s')\}$$

(minimal number of steps to reach an arbitrary state in BFS)



# Diameter Example






diameter 4, radius 2

(*reachable diameter* 3, distance from 0 to 4 or max. distance between 2,3,4)



# Completeness Threshold for Safety



- a bad state is reached in at most $r(T,I)$ steps from the initial states
  - a bad state is a state violating the invariant to be proven
- thus, the radius is a completeness threshold for safety properties
- for safety properties the max. $k$ for doing bounded model checking is $r(T,I)$
- if no counter example of this length can be found the safety property holds

# How to determine the radius?



**reformulation:**

the radius is the max. length $r$ of a path leading from an initial state to a state $t$, such there is no other path from an initial state to $t$ with length less than $r$.

Thus radius $r$ is the minimal number which makes the following formula valid:

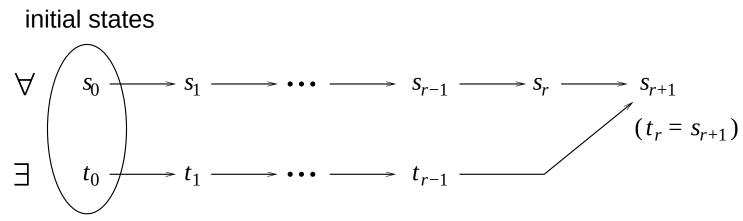
$$\forall s_0,\ldots,s_{r+1}[\;(I(s_0) \wedge \bigwedge_{i=0}^{r} T(s_i,s_{i+1})) \rightarrow$$
$$\exists\, n \leq r\;[\;\exists t_0,\ldots,t_n[\;I(t_0) \wedge \bigwedge_{i=0}^{n-1} T(t_i,t_{i+1}) \wedge t_n = s_{r+1}\;]\;]\;]$$

after replacing $\exists\, n \leq r \cdots$ by $\bigvee_{n=0}^{r} \cdots$ we get a **Quantified Boolean Formula** (QBF), which is much harder to prove un/satisfiable (PSPACE complete).



# Visualization of Reformulation






(we allow $t_{i+1}$ to be identical to $t_i$ in the lower path)



# Reoccurrence Radius/Diameter



- we can not find the real radius / diameter with SAT efficiently
- over approximation idea:
  - drop requirement that there is no shorter path
  - enforce *different* (no reoccurring) states on single path instead

**reoccurrence diameter:**

length of the longest path without reoccurring states

**reoccurrence radius:**

length of the longest initialized path without reoccurring states



# Determination of Reoccurrence Diameter



**reformulation:**

the reoccurrence radius is the length of the longest path from initial states without reoccurring states (one may further assume that only the first state is an initial state)

The reoccurring radius is the minimal $r$ which makes the following formula valid:

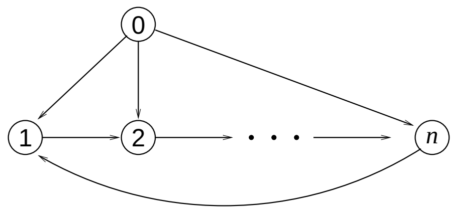
$$\forall s_0,\ldots,s_{r+1}[\;(I(s_0) \wedge \bigwedge_{i=0}^{r} T(s_i,s_{i+1})) \;\rightarrow \bigvee_{0 \leq i < j \leq r+1} s_i = s_j]$$

**this is a propositional formula and can be checked by SAT**

(exercise: reoccurrence radius/diameter is an upper bound on real radius/diameter)

## Bad Example for Reoccurrence Radius






radius 1, reoccurrence radius $n$



## Bounded Semantics with Loop



(E)LTL formula in NNF

let the path $\pi$ be a $(k,l)$ lasso

$$
\begin{array}{lll}
\pi \models_k^i p & \text{iff} & p \in L(\pi(i)) \\
\pi \models_k^i \neg p & \text{iff} & p \notin L(\pi(i)) \\
\pi \models_k^i f \wedge g & \text{iff} & \pi \models_k^i f \text{ and } \pi \models_k^i g \\
\pi \models_k^i \mathbf{X} f & \text{iff} & \begin{cases} \pi \models_k^l f & \text{if } i = k \\ \pi \models_k^{i+1} f & \text{else} \end{cases} \\
\pi \models_k^i \mathbf{G} f & \text{iff} & \bigwedge_{j=\min(i,l)}^{k} \pi \models_k^j f \\
\pi \models_k^i \mathbf{F} f & \text{iff} & \bigvee_{j=\min(i,l)}^{k} \pi \models_k^j f
\end{array}
$$



## Bounded Semantics without Loop



ELTL formula in NNF

there is no $l$ for which path $\pi$ is a $(k,l)$ lasso

$$
\begin{array}{lll}
\pi \models_k^i p & \text{iff} & p \in L(\pi(i)) \\
\pi \models_k^i \neg p & \text{iff} & p \notin L(\pi(i)) \\
\pi \models_k^i f \wedge g & \text{iff} & \pi \models_k^i f \text{ and } \pi \models_k^i g \\
\pi \models_k^i \mathbf{X} f & \text{iff} & \begin{cases} \text{false} & \text{if } i = k \\ \pi \models_k^{i+1} f & \text{else} \end{cases} \\
\pi \models_k^i \mathbf{G} f & \text{iff} & \text{false} \\
\pi \models_k^i \mathbf{F} f & \text{iff} & \bigvee_{j=i}^{k} \pi \models_k^j f
\end{array}
$$



## Bounded Semantics



- definition:

$$\pi \models_k f \quad :\Leftrightarrow \quad \pi \models_k^0 f$$

- bounded semantics aproximates real semantics:

$$\pi_k \models f \quad \Rightarrow \quad \pi \models f \qquad \text{for all } k$$

- (theoretical) completeness:

$$\text{if} \quad \pi \models f \quad \text{then there exists } k \text{ with} \quad \pi_k \models f$$

- **note:** negate original property first (e.g. $\mathbf{AG}p \mapsto \mathbf{EF}\neg p$)
  - ALTL $\longrightarrow$ ELTL
  - counter example $\longrightarrow$ witness
  - *bounded* witness is also a non-bounded witness

## Translation of Bounded Semantics to SAT



- two recursive translations from (E)LTL in NNF for fixed $k$:
  - ${}_l[\cdot]_k^i$ assumes $(k,l)$-loop
  - $[\cdot]_k^i$ assumes that no $(k,l)$-loop exists for all $l$
- add time frame expansion of transition relation:

$$I(s_0) \wedge T(s_0,s_1) \wedge \cdots \wedge T(s_{k-1},s_k)$$

- add $loop_k(l)$ constraint for looping translation: $loop_k(l) := T(s_k,s_l)$
- add $noloop_k$ constraint for non-looping translation:

$$noloop_k := \neg \bigvee_{l=0}^{k} loop_k(l)$$



## Looping Translation



$$\begin{aligned}
{}_l[p]_k^i &:= p(s_i) \\
{}_l[\neg p]_k^i &:= \neg p(s_i) \\
{}_l[f \wedge g]_k^i &:= {}_l[f]_k^i \wedge {}_l[g]_k^i \\
{}_l[\mathbf{X}\, f]_k^i &:= {}_l[f]_k^{next(i)} \\
{}_l[\mathbf{G}\, f]_k^i &:= \bigwedge_{j=min(l,i)}^{k} {}_l[f]_k^j \\
{}_l[\mathbf{F}\, f]_k^i &:= \bigvee_{j=min(l,i)}^{k} {}_l[f]_k^j
\end{aligned}$$

with

$$next(i) := \begin{cases} i+1 & \text{if } i<k \\ l & \text{else} \end{cases}$$



## Non-Looping Translation



$$\begin{aligned}
[p]_k^i &:= p(s_i) \\
[\neg p]_k^i &:= \neg p(s_i) \\
[f \wedge g]_k^i &:= [f]_k^i \wedge [g]_k^i \\
[\mathbf{X}\, f]_k^i &:= \begin{cases} [f]_k^{i+1} & \text{if } i<k \\ \mathit{false} & \text{else} \end{cases} \\
[\mathbf{G}\, f]_k^i &:= \mathit{false} \\
[\mathbf{F}\, f]_k^i &:= \bigvee_{j=i}^{k} [f]_k^j
\end{aligned}$$



## Translation



$$[K,f]_k := noloop_k \wedge [f]_k^0 \vee \bigvee_{l=0}^{k} loop_k(l) \wedge {}_l[f]_k^0$$

- **Theorem:** $K \models \mathbf{E} f \quad \Leftrightarrow \quad \exists k\, [K,f]_k$ satisfiable
- ${}_l[\cdot]_k^i$ and $[\cdot]_k^i$ are **linear** in $k$ if subformulae are shared
  - unique table for automatic sharing syntactically equivalent formulae
  - implemented as hash table (keys are pairs of formulae ids)
- more complex and quadratic translations for $\mathbf{R}$ and $\mathbf{U}$