

# Tutorial on Model Checking

## Modelling and Verification in Computer Science

Armin Biere

Institute for Formal Models and Verification

Johannes Kepler University

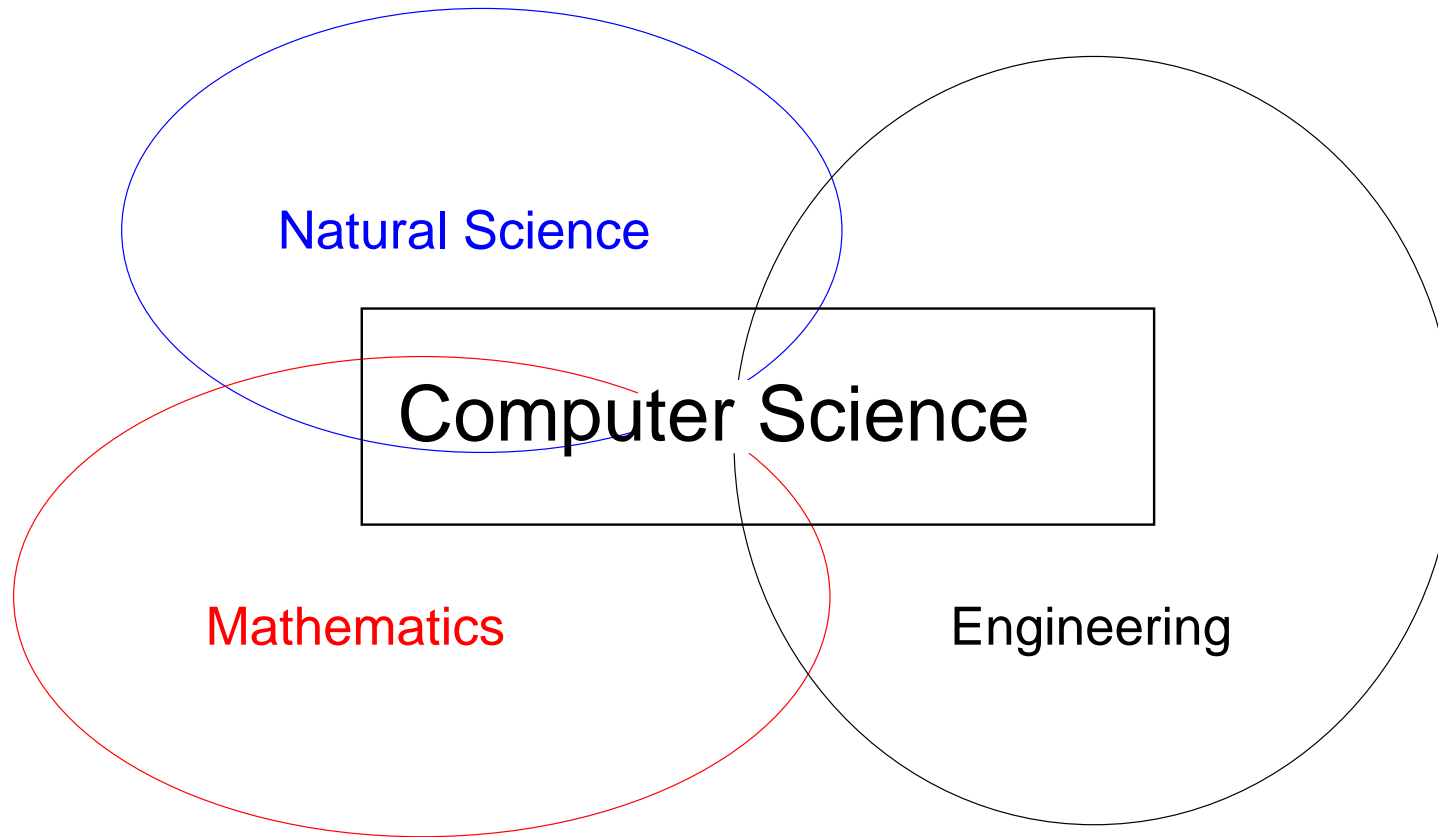
Linz, Austria

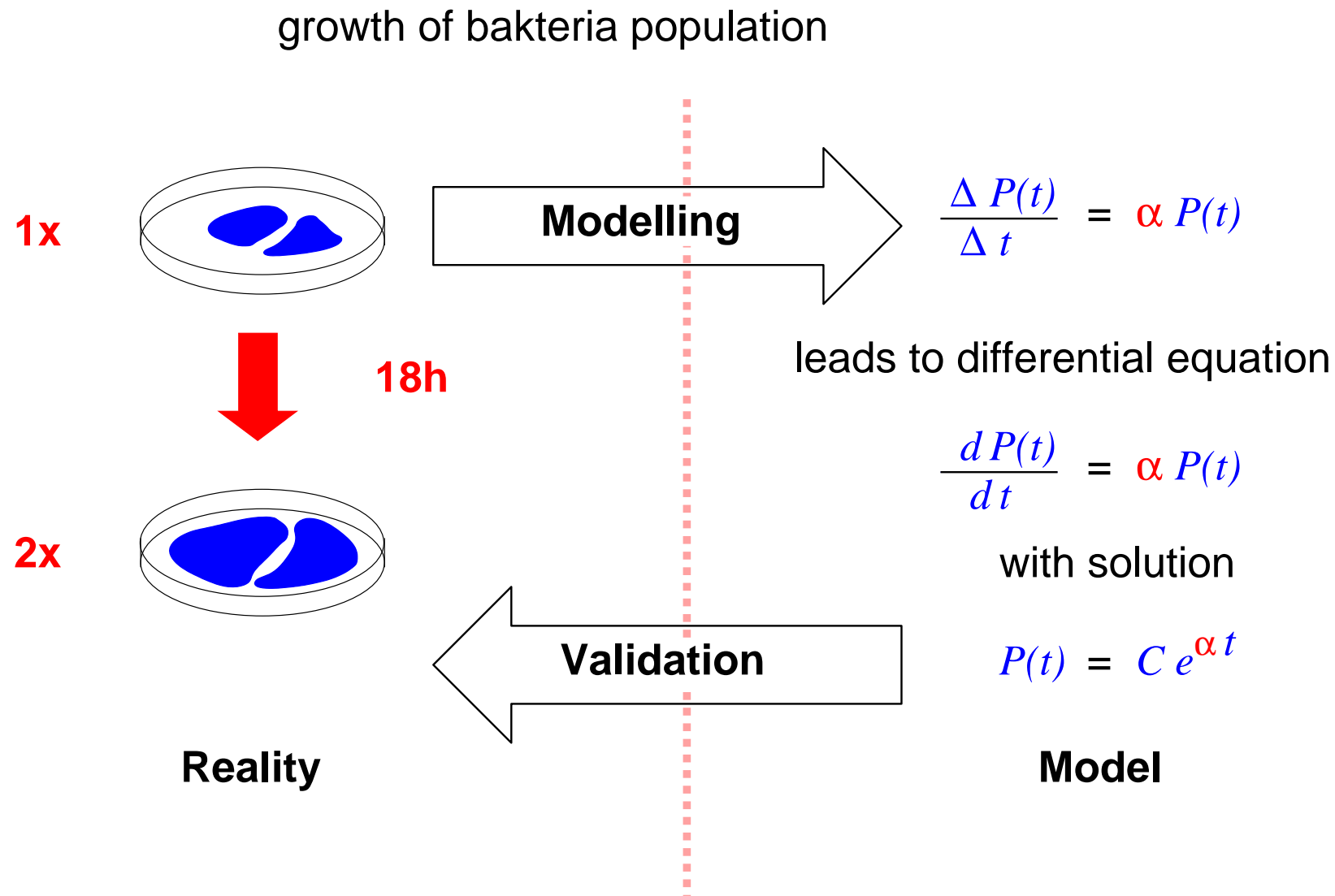
### AB'08

### Algebraic Biology

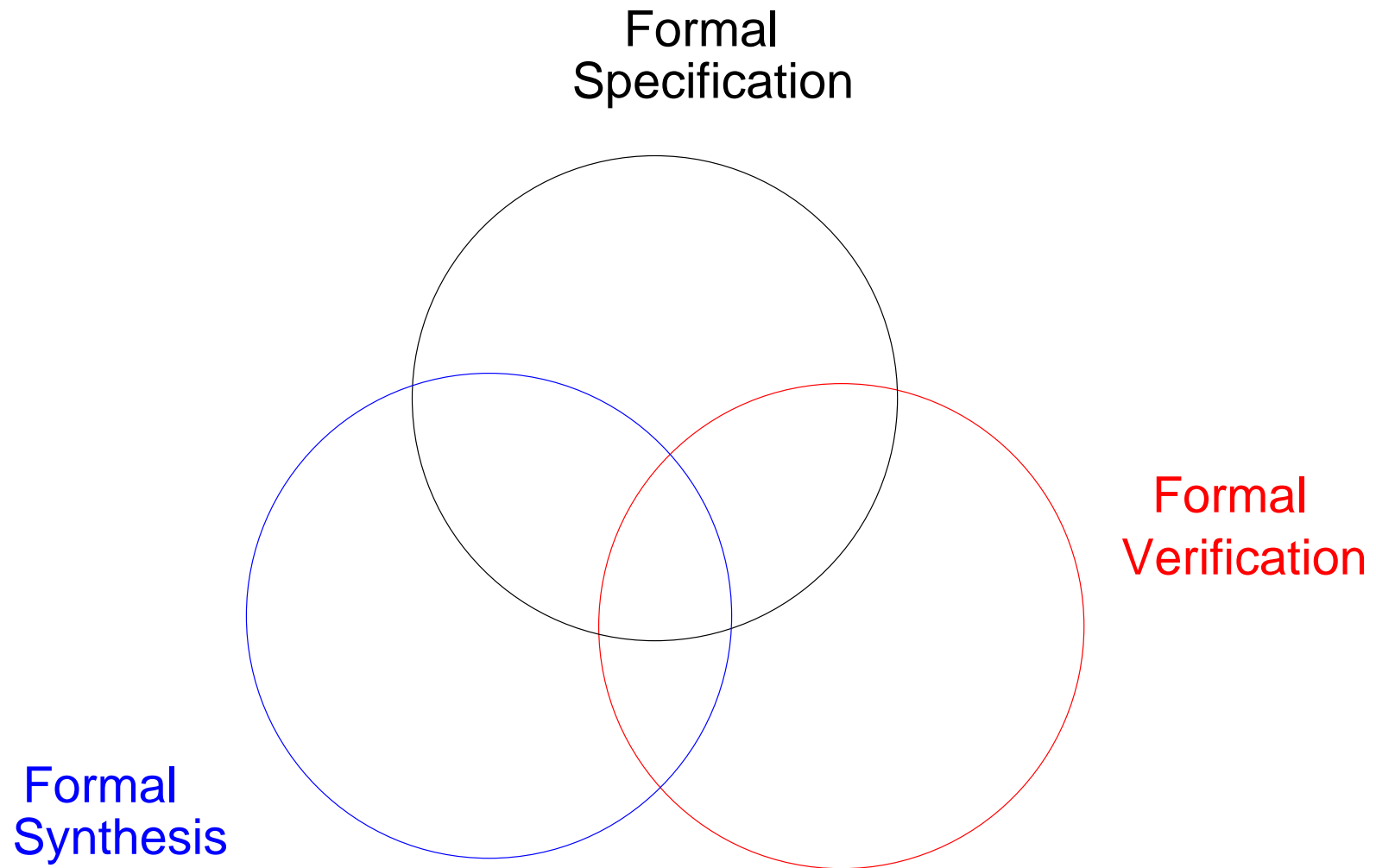
Castle of Hagenberg, Austria

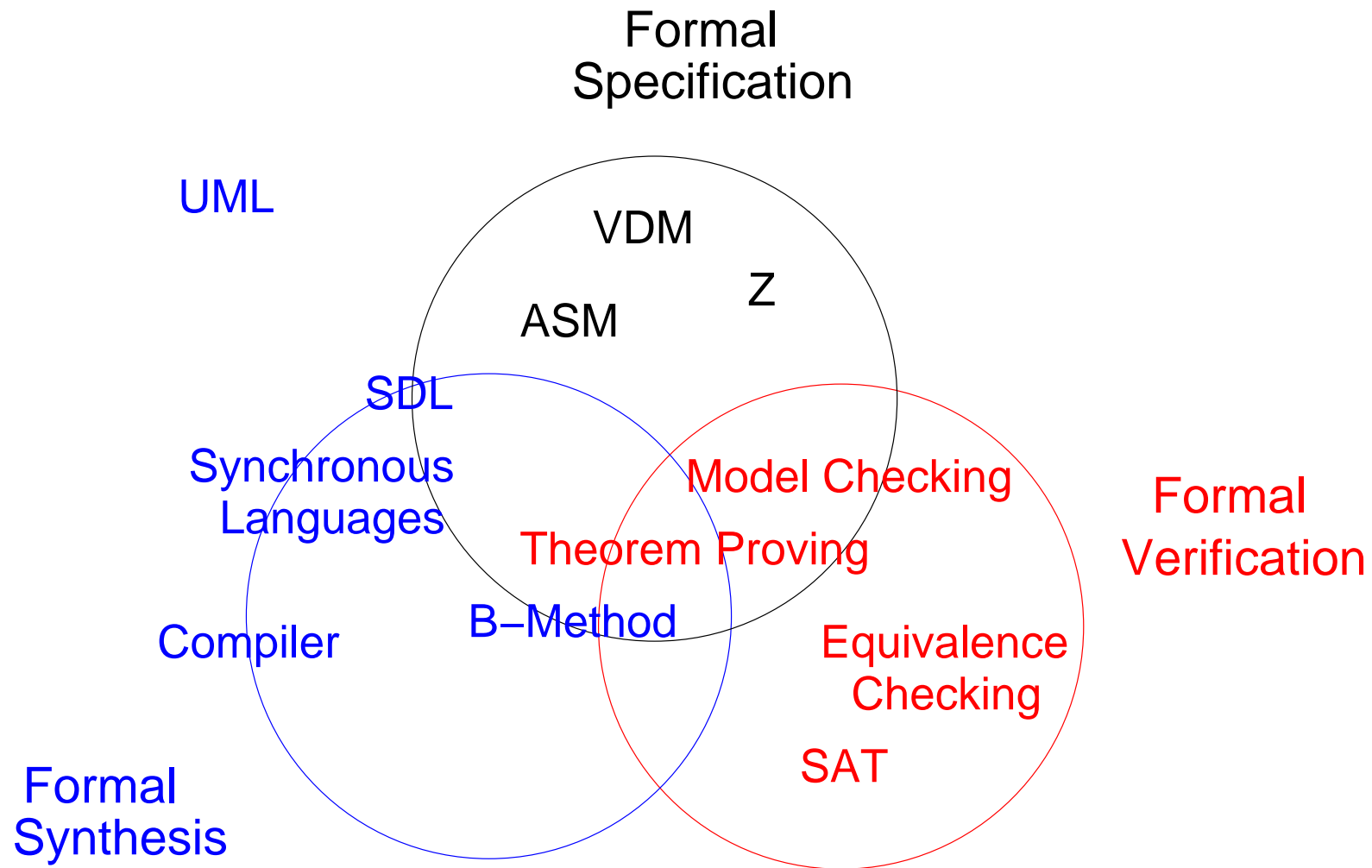
Friday, August 1, 2008





- programs and other digital systems are formal objects
  - they have precise mathematical models (denotational/operational)
  - Reality = Model  
(modulo complex semantics, compiler bugs, hardware failure, ...)
  - **properties of the models also hold in reality**
- proving properties of models is difficult
  - for Software in general **undecidable**
  - for Hardware in **NP** or **PSPACE**
- only valid for **functional properties** , not for **quantitative aspects**
  - availability, through put, latency, etc. are difficult to model precisely





- mechanically check properties of models
- models:
  - finite automata, labelled transition systems
  - often requires automatic/manual abstraction techniques
- properties:
  - only interested in *partial properties*
  - specified in temporal logic: CTL, LTL, etc.
  - safety: something bad should not happen
  - liveness: something good should happen
- automatic generation of counterexamples

BurchClarkeMcMillanDillHwang'90: Symbolic Model Checking

DavisPutnam'60: DP

CoudertMadre'89: Symbolic Reachability

McMillan'03: Interpolation

DavisLogemannLoveland'62: DPLL

Marques–SilvaSakallah'96: GRASP

Bryant'86: BDDs

BiereArthoSchuppan'01: Liveness2Safety

Pnueli'77: Temporal Logic

MoskewiczMadiganZhaoZhangMalik'01: CHAFF

McMillan'93: SMV

EenSorensson'03: MiniSAT

ClarkeEmerson'82: Model Checking

BiereCimattiClarkeZhu'99: Bounded Model Checking

Kurshan'93: Localization

EenBiere'05: SatELite

SheeranSinghStalmarck'00:  $k$ –Induction

QuielleSifakis'82: Model Checking

BallRajamani'01: SLAM

Holzmann'91: SPIN

GrafSaidi'97: Predicate Abstraction

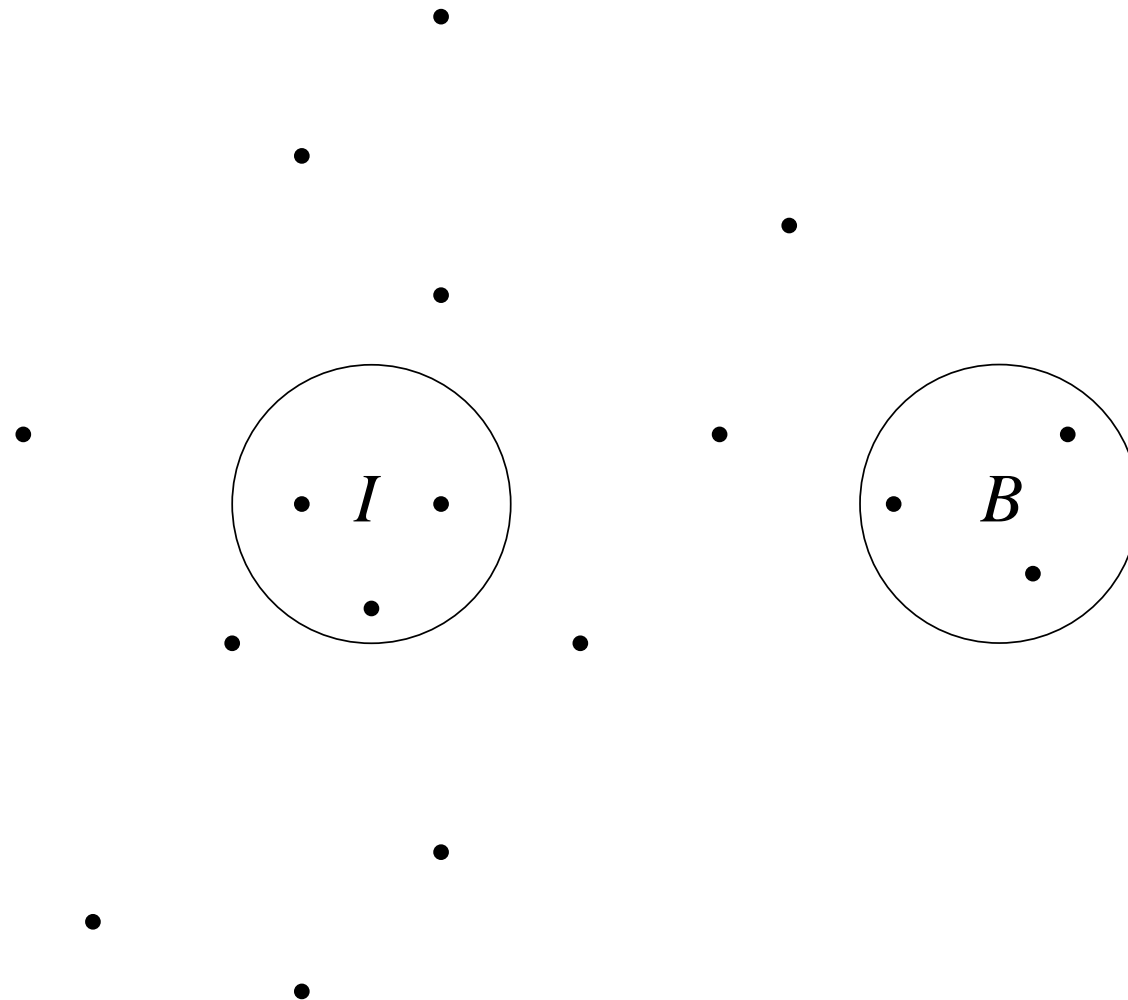
Holzmann'81: On–The–Fly Reachability

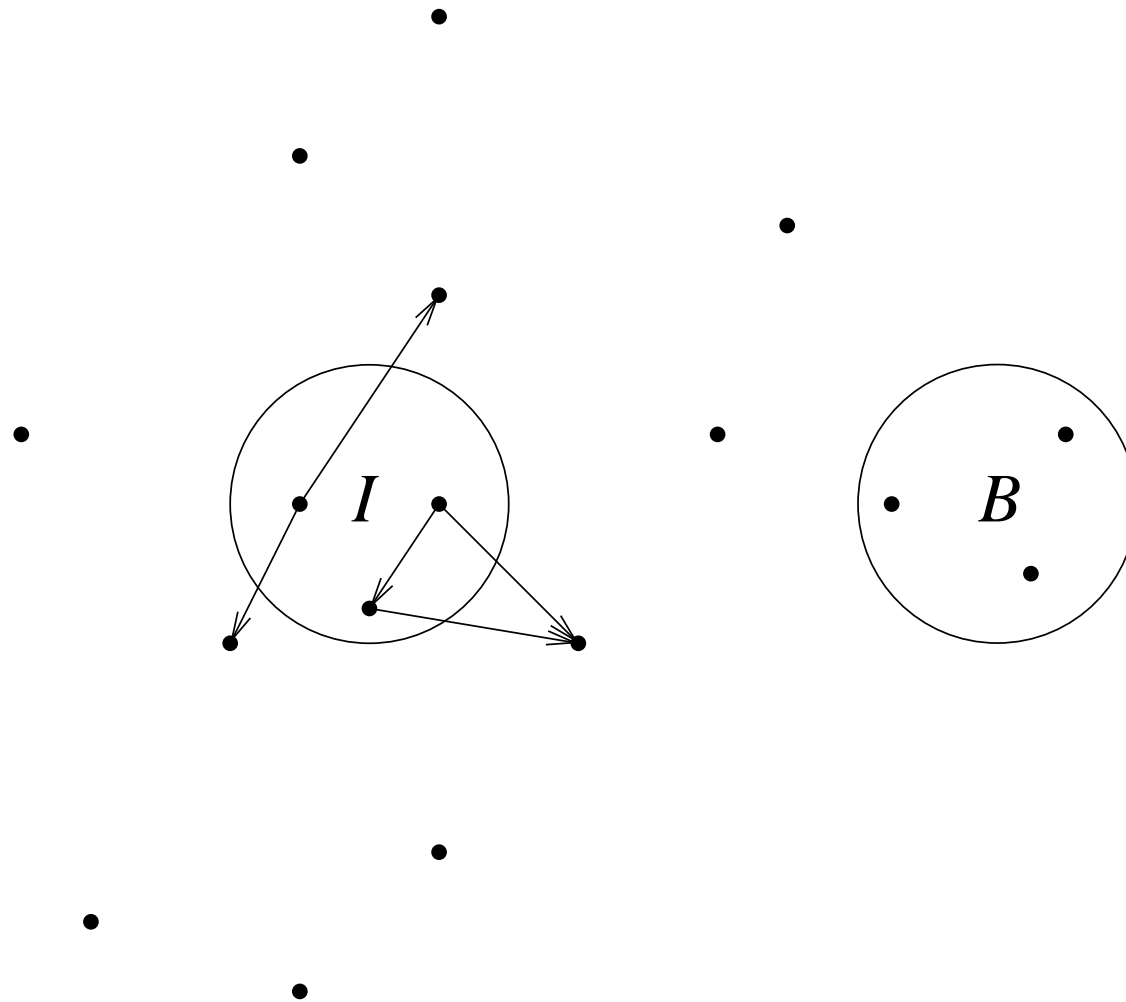
ClarkeGrumbergJahLuVeith'03: CEGAR

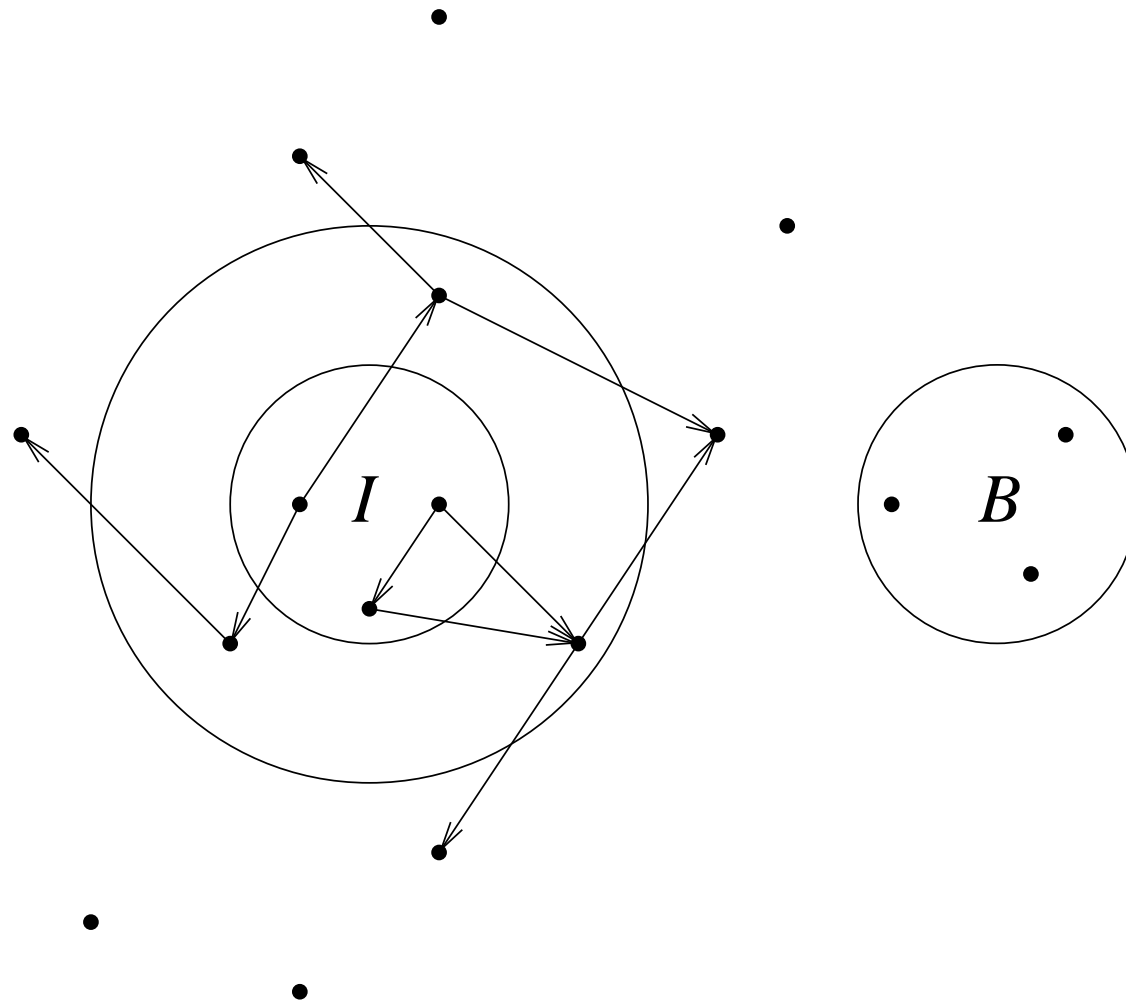
Peled'94: Partial–Order–Reduction

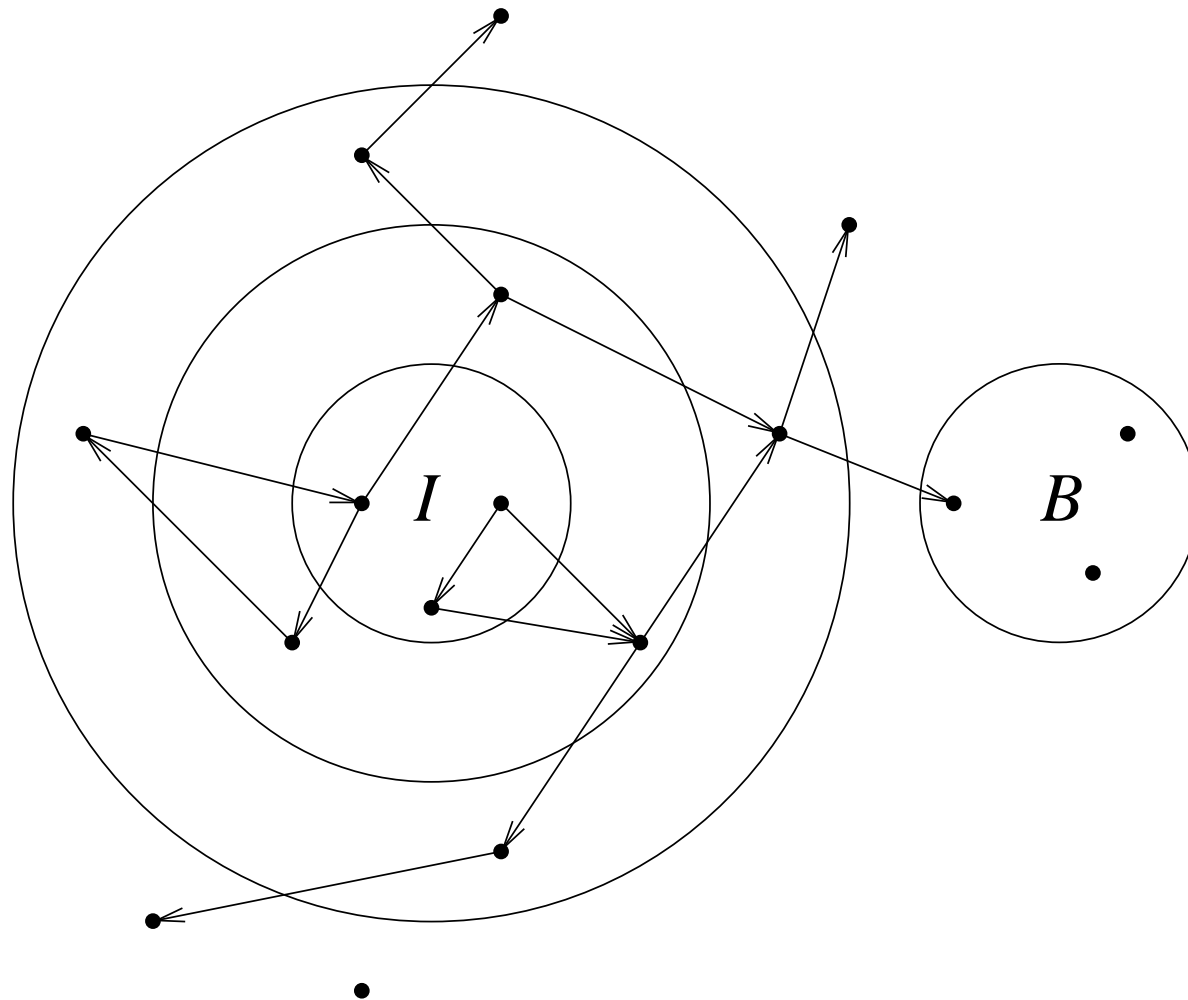


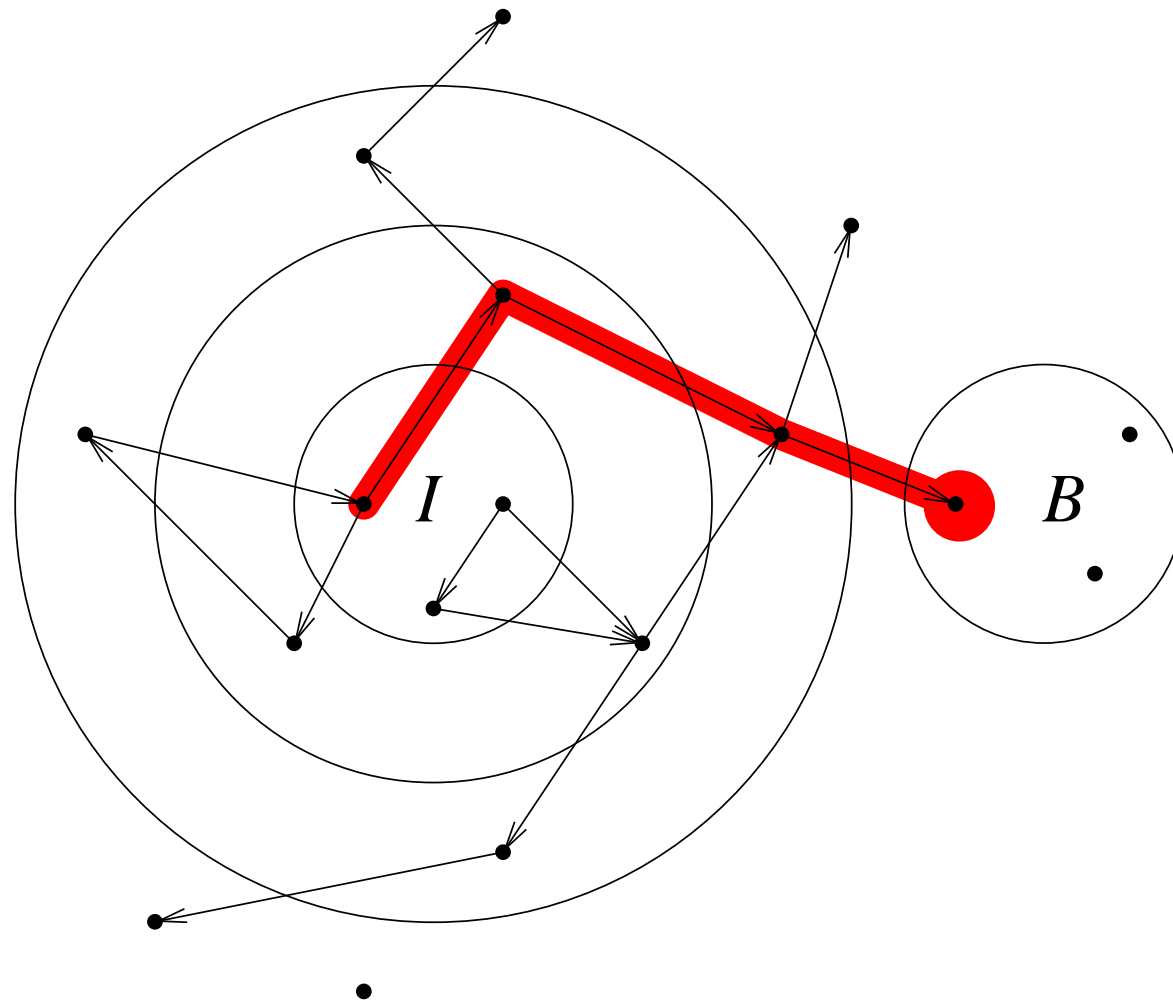
- set of states  $S$ , initial states  $I$ , transition relation  $T$
- bad states  $B$  reachable from  $I$  via  $T$ ?
- symbolic representation of  $T$  (circuit, program, parallel product)
  - avoid explicit matrix representations, because of the
  - state space explosion problem, e.g.  $n$ -bit counter:  $|T| = O(n)$ ,  $|S| = O(2^n)$
  - makes reachability PSPACE complete [Savitch'70]
- on-the-fly [Holzmann'81'] for protocols
  - restrict search to reachable states
  - simulate and hash reached concrete states

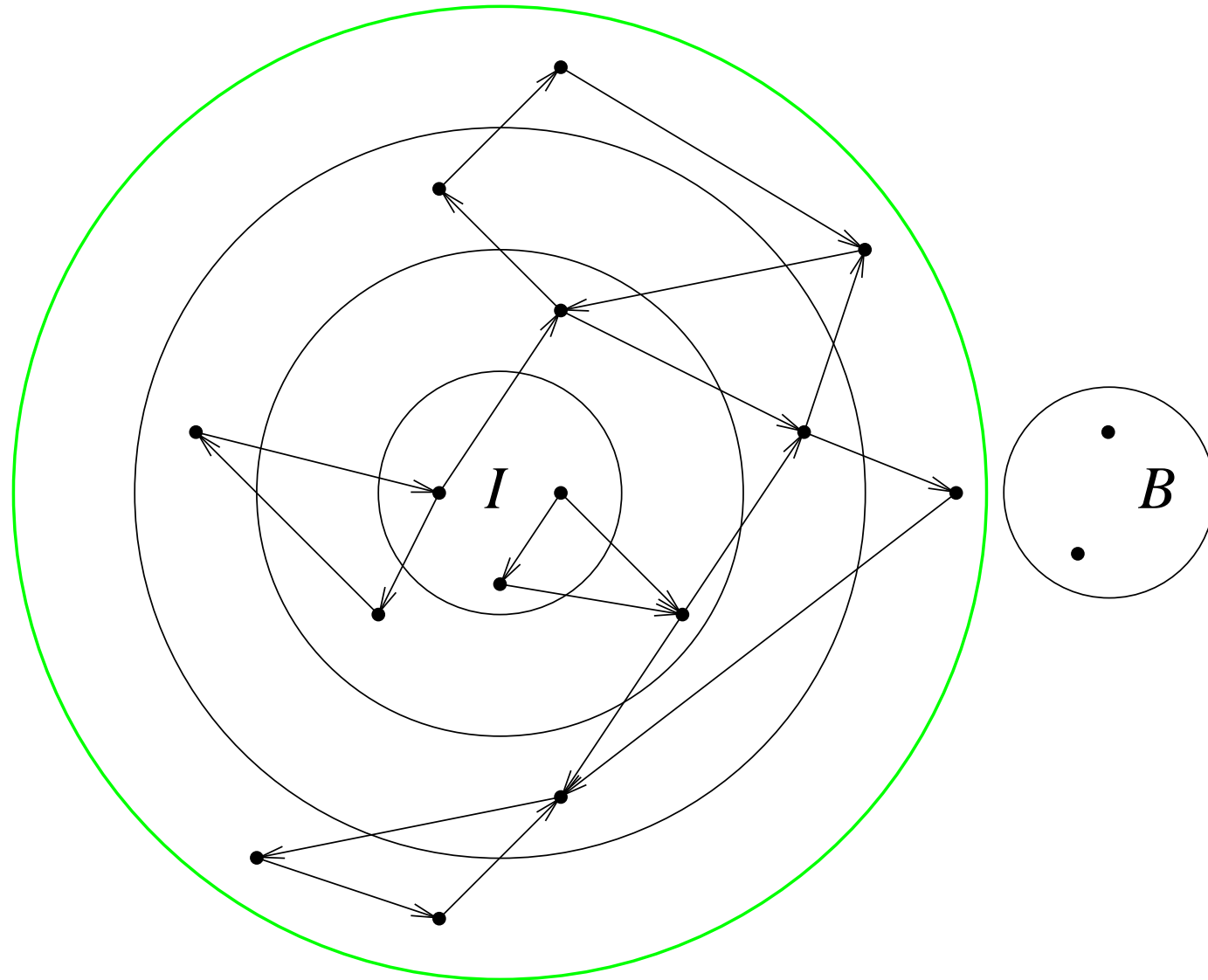












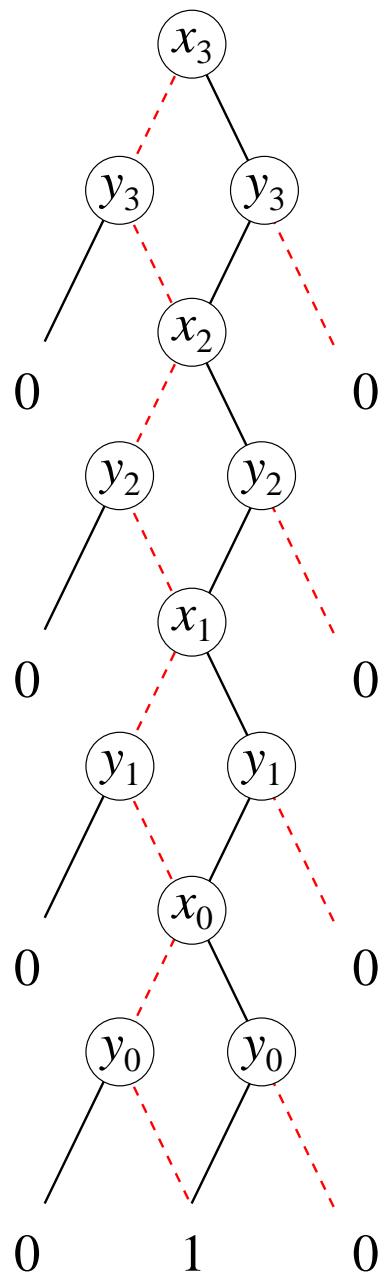
initial states  $I$ , transition relation  $T$ , bad states  $B$

```
model-check $\mu$ forward ( $I, T, B$ )  
   $S_C = \emptyset; S_N = I;$   
  while  $S_C \neq S_N$  do  
    if  $B \cap S_N \neq \emptyset$  then  
      return “found error trace to bad states”;  
     $S_C = S_N;$   
     $S_N = S_C \cup \text{Img}(S_C);$   
  done;  
  return “no bad state reachable”;
```



- algorithms to check more general properties [ClarkeEmerson'82], [QuielleSifakis'82]
  - uses temporal logic [Pnueli'77] as property specification language
  - model checkers are usually fully automatic
    - linear vs. branching time formalisms (CTL vs LTL) was hotly debated
  - either determine that property holds or ...
  - ... provide counter example for debugging purposes
- originally explicit (as in SPIN [Holzmann'91])
  - search works with concrete states,
  - **bottle neck:** number of states, that have to be stored
  - local (on-the-fly) and global algorithms (not on-the-fly)

- work with symbolic representations of states
  - symbolic representations are potentially exponentially more succinct
  - favors BFS: next frontier set of states in BFS is calculated symbolically
- originally “symbolic” meant model checking with BDDs  
[CoudertMadre’89/’90,BurchClarkeMcMillanDillHwang’90,McMillan’93]
- Binary Decision Diagrams [Bryant’86]
  - canonical representation for boolean functions
  - BDDs have fast operations (but image computation is expensive)
  - often blow up in space
  - restricted to hundreds of variables



boolean function/expression:

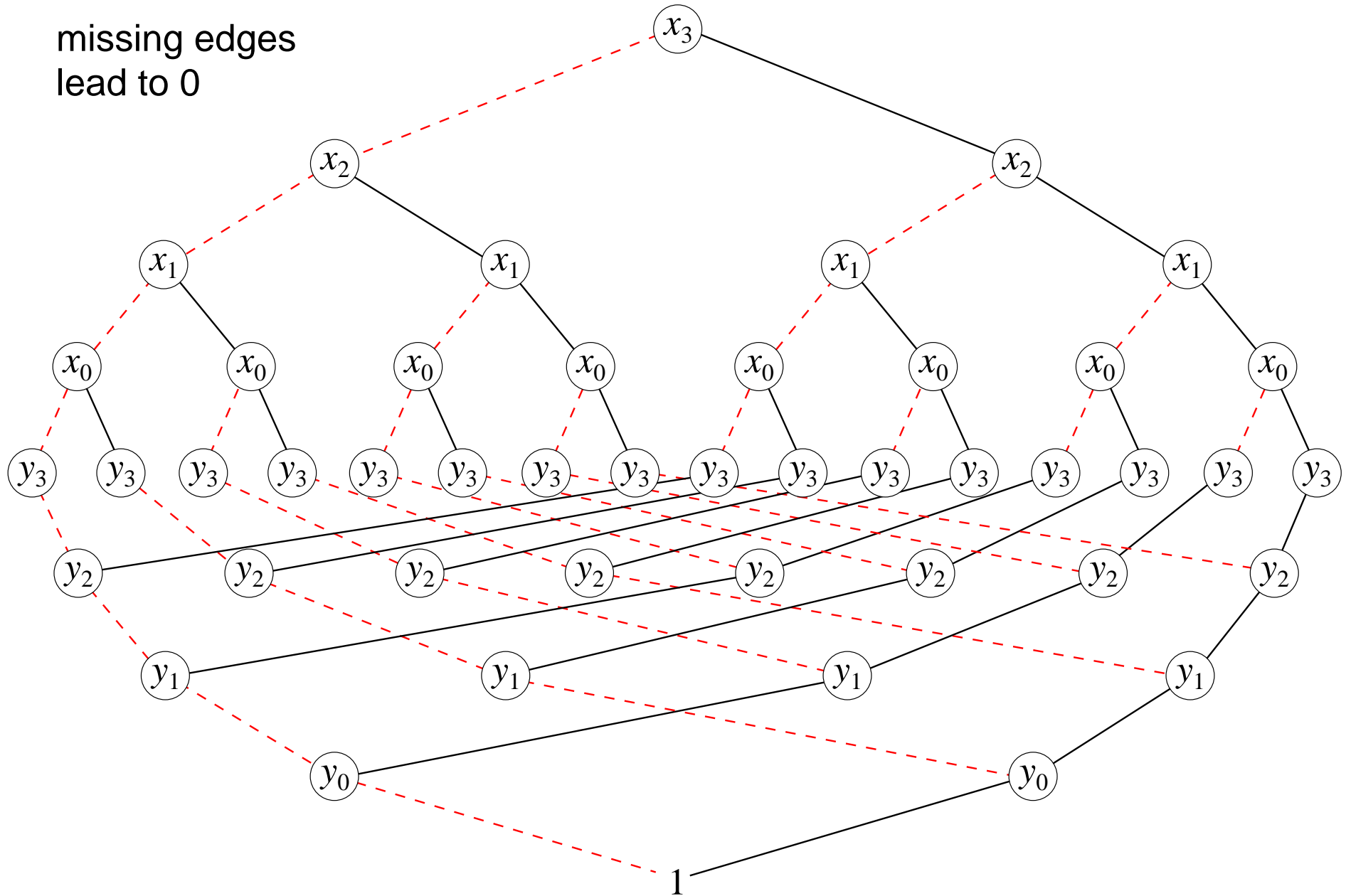
$$\bigwedge_{i=0}^{n-1} x_i = y_i$$

interleaved variable order:

$$x_3 > y_3 > x_2 > y_2 > x_1 > y_1 > x_0 > y_0$$

comparison of two  $n$ -bit-vectors needs  $3 \cdot n$  inner nodes for the interleaved variable order

missing edges  
lead to 0



0: continue?	$S_C^0 \neq S_N^0$	$\exists s_0 [I(s_0)]$
0: terminate?	$S_C^0 = S_N^0$	$\forall s_0 [\neg I(s_0)]$
0: bad state?	$B \cap S_N^0 \neq \emptyset$	$\exists s_0 [I(s_0) \wedge B(s_0)]$
1: continue?	$S_C^1 \neq S_N^1$	$\exists s_0, s_1 [I(s_0) \wedge T(s_0, s_1) \wedge \neg I(s_1)]$
1: terminate?	$S_C^1 = S_N^1$	$\forall s_0, s_1 [I(s_0) \wedge T(s_0, s_1) \rightarrow I(s_1)]$
1: bad state?	$B \cap S_N^1 \neq \emptyset$	$\exists s_0, s_1 [I(s_0) \wedge T(s_0, s_1) \wedge B(s_1)]$
2: continue?	$S_C^2 \neq S_N^2$	$\exists s_0, s_1, s_2 [I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge \neg (I(s_2) \vee \exists t_0 [I(t_0) \wedge T(t_0, s_2)])]$
2: terminate?	$S_C^2 = S_N^2$	$\forall s_0, s_1, s_2 [I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \rightarrow I(s_2) \vee \exists t_0 [I(t_0) \wedge T(t_0, s_2)]]$
2: bad state?	$B \cap S_N^2 \neq \emptyset$	$\exists s_0, s_1, s_2 [I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge B(s_2)]$

0: continue?  $s_C^0 \neq s_N^0 \quad \exists s_0 [I(s_0)]$

0: terminate?  $s_C^0 = s_N^0 \quad \forall s_0 [\neg I(s_0)]$

0: bad state?  $B \cap S_N^0 \neq \emptyset \quad \exists s_0 [I(s_0) \wedge B(s_0)]$

1: continue?  $s_C^1 \neq s_N^1 \quad \exists s_0, s_1 [I(s_0) \wedge T(s_0, s_1) \wedge \neg I(s_1)]$

1: terminate?  $s_C^1 = s_N^1 \quad \forall s_0, s_1 [I(s_0) \wedge T(s_0, s_1) \rightarrow I(s_1)]$

1: bad state?  $B \cap S_N^1 \neq \emptyset \quad \exists s_0, s_1 [I(s_0) \wedge T(s_0, s_1) \wedge B(s_1)]$

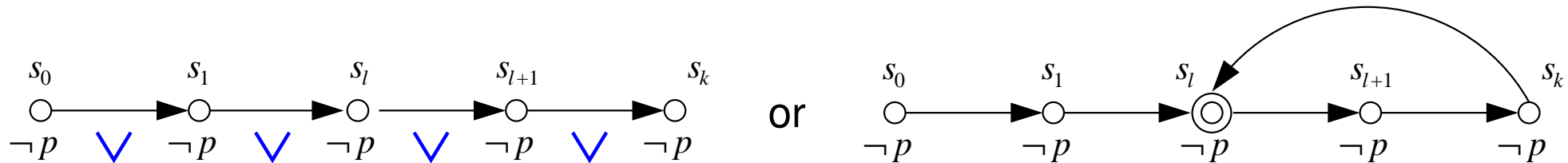
2: continue?  $s_C^2 \neq s_N^2 \quad \exists s_0, s_1, s_2 [I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge \neg (I(s_2) \vee \exists t_0 [I(t_0) \wedge T(t_0, s_2)])]$

2: terminate?  $s_C^2 = s_N^2 \quad \forall s_0, s_1, s_2 [I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \rightarrow I(s_2) \vee \exists t_0 [I(t_0) \wedge T(t_0, s_2)]]$

2: bad state?  $B \cap S_N^2 \neq \emptyset \quad \exists s_0, s_1, s_2 [I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge B(s_2)]$

[BiereCimattiClarkeZhu'99]

- look only for counter example made of  $k$  states (the bound)



- simple for safety properties  $p$  is invariantly true (e.g.  $p = \neg B$ )

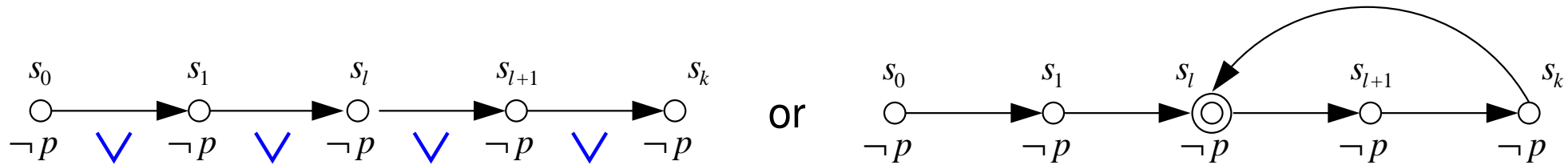
$$I(s_0) \wedge T(s_0, s_1) \wedge \cdots \wedge T(s_{k-1}, s_k) \wedge \bigvee_{i=0}^k \neg p(s_i)$$

- harder for liveness properties  $p$  is eventually true

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

[BiereCimattiClarkeZhu'99]

- look only for counter example made of  $k$  states (the bound)



- simple for safety properties  $p$  is invariantly true (e.g.  $p = \neg B$ )

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

- harder for liveness properties  $p$  is eventually true

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



- satisfiability checking (SAT)
  - of propositional/combinational problems (only boolean variables)
  - actually restricted to conjunctive normal form (CNF)
  - classical NP hard problem [Cook'71]
- key motivation of BMC
  - leverage capacity of SAT solvers
  - SAT solvers could handle 10000 variables in late 90'ties
  - compared to hundreds of variables with BDDs
- key insight: trade capacity for completeness

- increase in efficiency of SAT solvers [ZChaff,MiniSAT,SatELite,PicoSAT]
- SAT more robust than BDDs in bug finding  
(shallow bugs are easily reached by explicit model checking or testing)
- better **unbounded** but still SAT based model checking algorithms
  - $k$ -induction [SinghSheeranStalmarck'00]
  - interpolation [McMillan'03]
- 4th Intl. Workshop on Bounded Model Checking (BMC'06)
- other logics beside LTL, better encodings, e.g. [LatvalaBiereHeljankoJuntilla'04]
- other system models, such as hybrid automata

[SinghSheeranStalmarck'00]

- more specifically  $k$ -induction

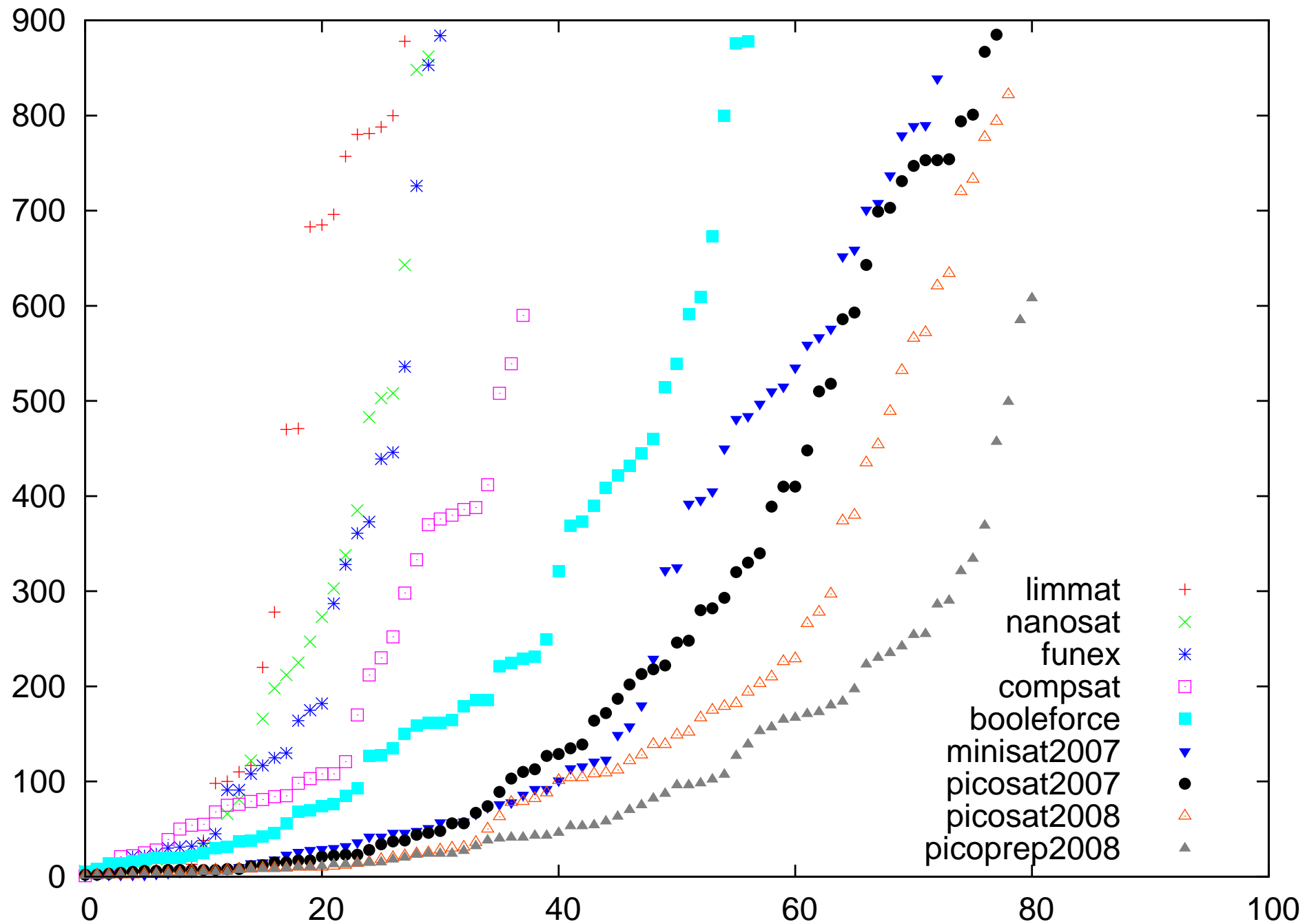
– does there exist  $k$  such that the following formula is *unsatisfiable*

$$\overline{B(s_0)} \wedge \cdots \wedge \overline{B(s_{k-1})} \wedge T(s_0, s_1) \wedge \cdots \wedge T(s_{k-1}, s_k) \wedge B(s_k) \wedge \bigwedge_{0 \leq i < j \leq k} s_i \neq s_j$$

– if *unsatisfiable* and  $\neg \text{BMC}(k)$  then **bad state unreachable**

- bound on  $k$ : length of **longest cycle free path**
- $k = 0$  check whether  $\neg B$  tautological (propositionally)
- $k = 1$  check whether  $\neg B$  inductive for  $T$

- Davis and Putnam procedure
  - DP: elimination procedure [DavisPutnam'60]
  - DPLL: splitting [DavisLogemannLoveland'62]
- modern SAT solvers are mostly based on DPLL
  - learning: GRASP [MarquesSilvaSakallah'96], ReISAT [BayardoSchrag'97]
  - watched literals, VSIDS: CHAFF [MoskewiczMadiganZhaoZhangMalik'01]
  - improved heuristics: MiniSAT [EenSorensson'03] actually Version from 2005
- preprocessing is a hot topic:
  - currently fastest solvers use SatELite style preprocessing [EenBiere'05] DP
- [www.satcompetition.org](http://www.satcompetition.org) since 2002



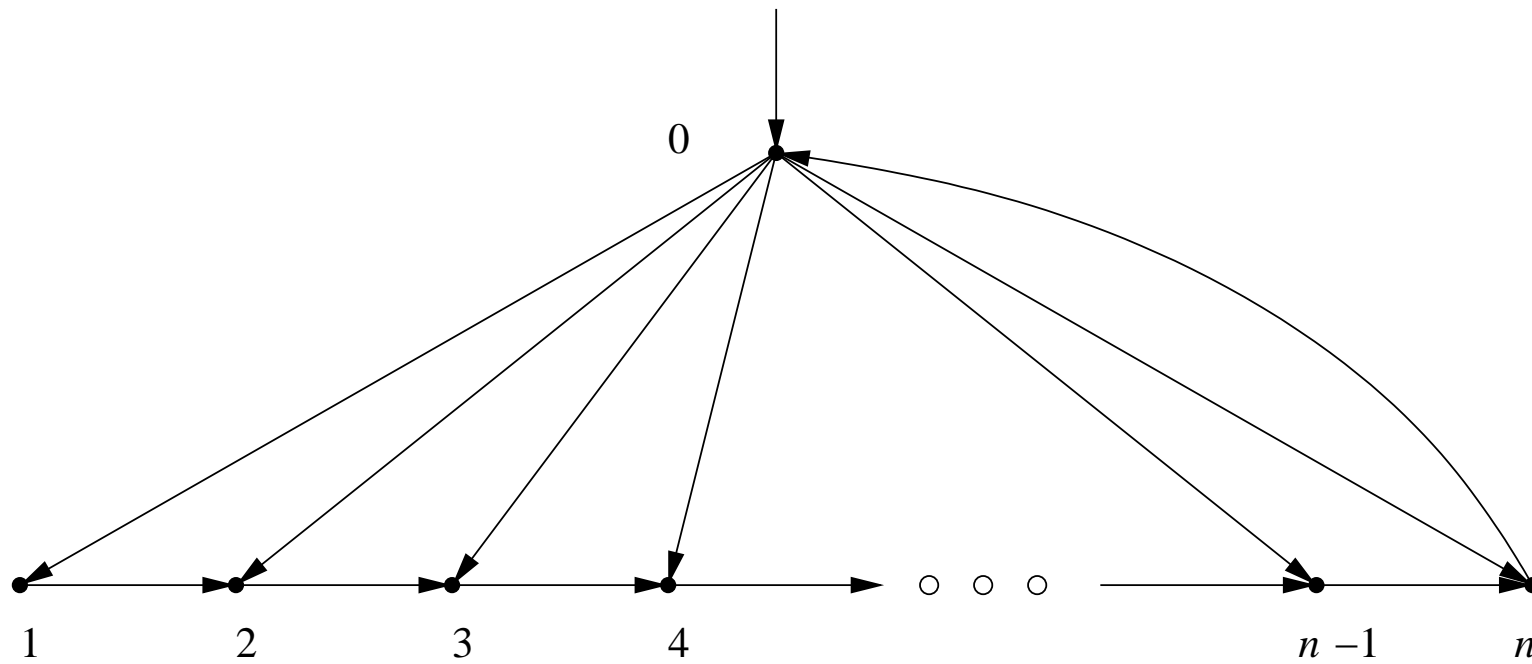
[McMillan'03]

- SAT based technique to overapproximate frontiers  $Img(S_C)$ 
  - currently most effective technique to show that bad states are unreachable
  - better than BDDs and  $k$ -induction in most cases [HWMCC'07]

- starts from a **resolution proof** refutation of a BMC problem with bound  $k + 1$

$$S_C(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge \cdots \wedge T(s_k, s_{k+1}) \wedge B(s_{k+1})$$

- result is a characteristic function  $f(s_1)$  over variables of the second state  $s_1$
  - these states do not reach the bad state  $s_{k+1}$  in  $k$  steps
  - any state reachable from  $S_C$  satisfies  $f$ :  $S_C(s_0) \wedge T(s_0, s_1) \Rightarrow f(s_1)$
- $k$  is bounded by the diameter (exponentially smaller than longest cycle free path)



length of longest shortest path  $O(n)$

diameter  $O(1)$

- further convergence between theorem proving and model checking
  - as pioneered by SLAM [BallRajamani'01] using
    - \* predicate abstraction [GrafSaidi'97] and
    - \* counter example guided abstraction refinement [ClarkeGrumbergJahLuVeith'03]
  - handle large software and hardware systems precisely
  - automate compositional reasoning, e.g. alias analysis
- improve Satisfiability Modulo Theory (SMT) procedures
  - What is the right way to handle bit-vectors, arrays?
  - Quantifiers, interpolation for bit-vectors and arrays?



- Satisfiability Solver (SAT) (standard NP hard problem)
  - improve heuristics, remove magic constants
  - more aggressive incremental preprocessing
  - effective incorporation of more powerful reasoning engines
- Quantified Boolean Formulas (QBF) (standard PSPACE hard problem)
  - new paradigms?
  - improve capacity and effectively apply QBF to real problems
- and do not forget testing, debugging, simulation

- Edmund M. Clarke, Orna Grumberg, and Doron Peled.  
*Model Checking*.  
MIT press, 1999.
- Gerard Holzmann.  
*The SPIN Model Checker*.  
Addison Wesley, 2004.
- Helmut Veith and Orna Grumberg, editors.  
*25 Years of Model Checking*, volume 5000 of *LNCS*. Springer, 2008.

- Mukul Prasad, Armin Biere, and Aarti Gupta.  
A survey on recent advances in SAT-based formal verification.  
*Software Tools for Technology Transfer (STTT)*, 7(2), 2005.
- Armin Biere.  
Bounded Model Checking.  
In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors,  
*Handbook of Satisfiability*,  
To be published by IOS Press.