Introduction to Bounded Model Checking

Armin Biere

Institute for Formal Models and Verification
Johannes Kepler University
Linz, Austria

FATS Seminar

ETH Zürich, Switzerland

Wednesday, October 28, 2009

A Personal History of Model Checking

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

EenBiere'05: SatELite

Kurshan'93: Localization

SheeranSinghStalmarck'00: k−Induction

QuielleSifakis'82: Model Checking

BallRajamani'01: SLAM

ClarkeEmersonSifakis:
Turing Award 2007

Holzmann'91: SPIN

GrafSaidi'97: Predicate Abstraction

Holzmann'81: On−The−Fly Reachability

ClarkeGrumbergJahLuVeith'03: CEGAR

Peled'94: Partial−Order−Reduction

What is Model Checking?

- mechanically check properties of models
- models:
  - finite automata, labelled transition systems
  - often requires automatic/manual abstraction techniques
- properties:
  - mostly 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

Reachability

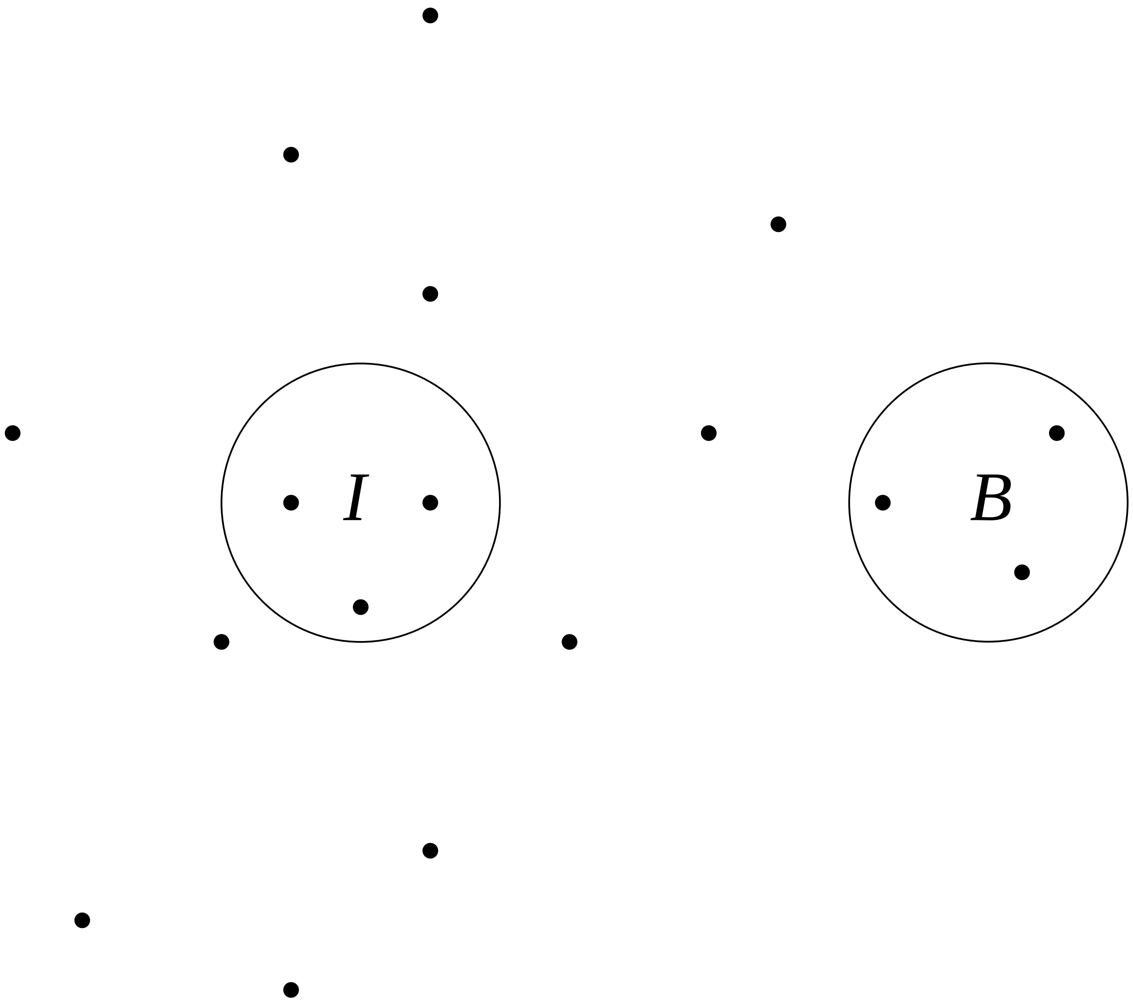
- set of states S, initial states I, transition relation T

- bad states B reachable from I via T?

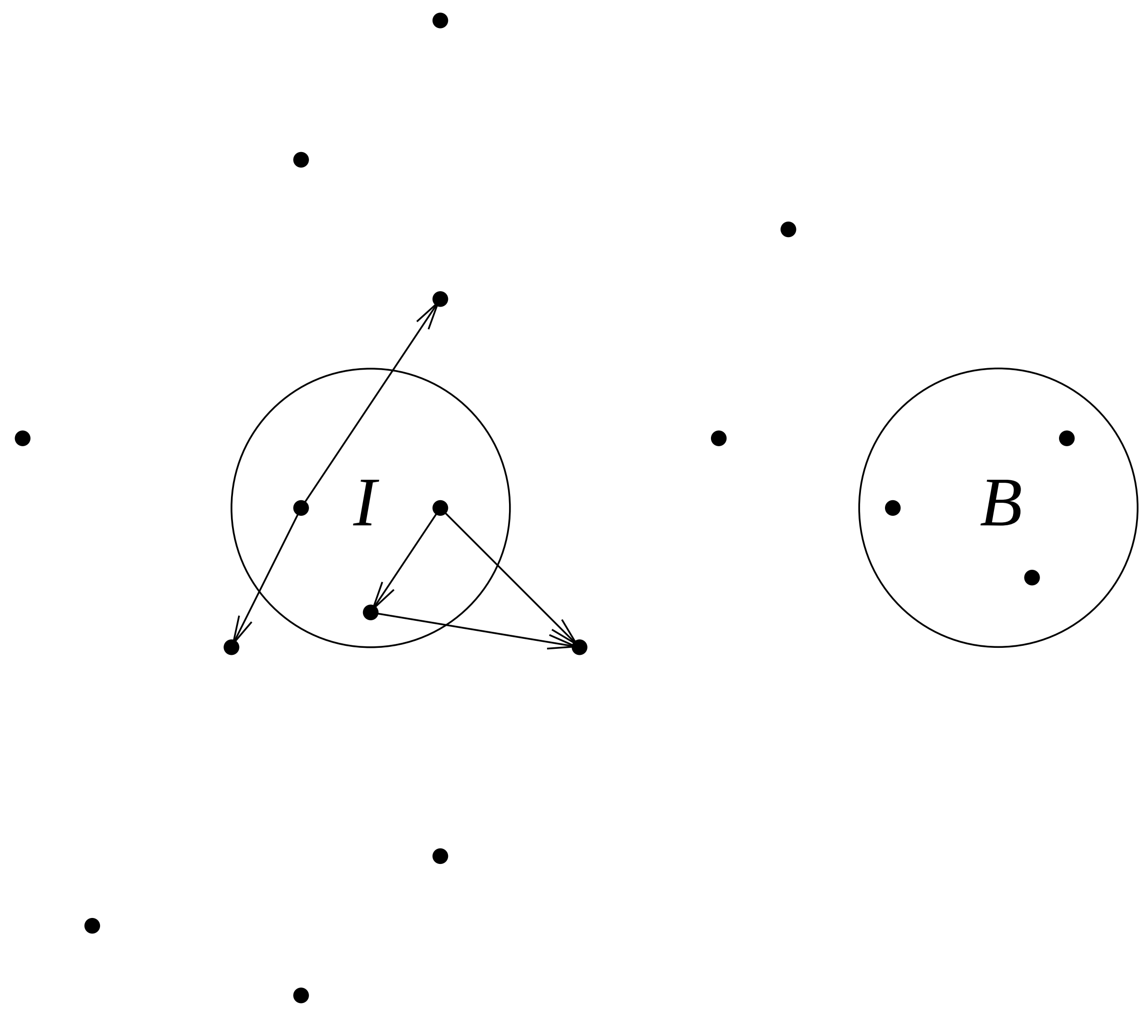
- symbolic representation of T (ciruit, program, parallel product)
  - avoid explicit matrix representations, because of the
  - state space explosion problem, e.g. n-bit counter: $|T| = O(n)$, $|S| = O(2^n)$
  - makes reachability PSPACE complete [Savitch'70]

- on-the-fly [Holzmann'81'] for protocols
  - restrict search to reachable states
  - simulate and hash reached concrete states

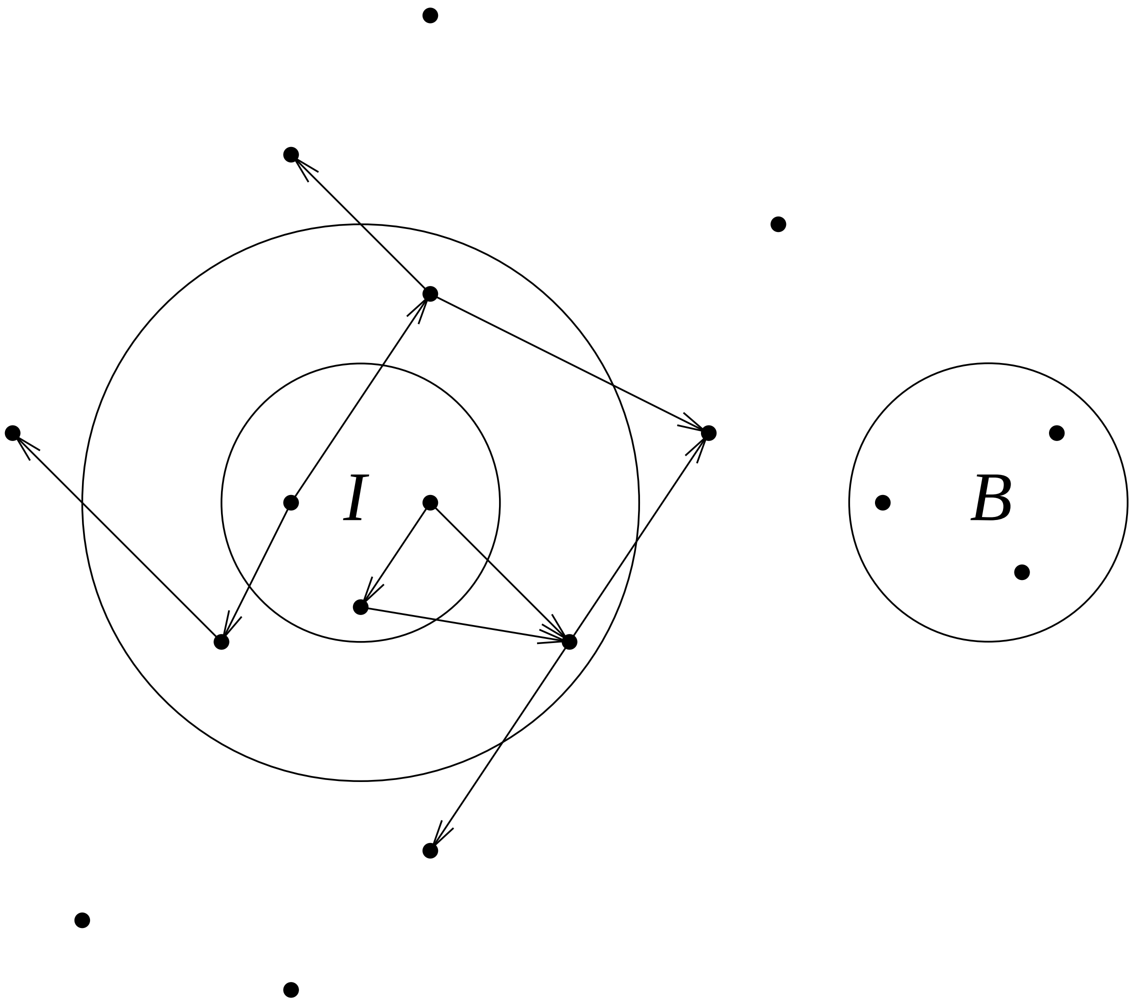
Forward Fixpoint: Initial and Bad States



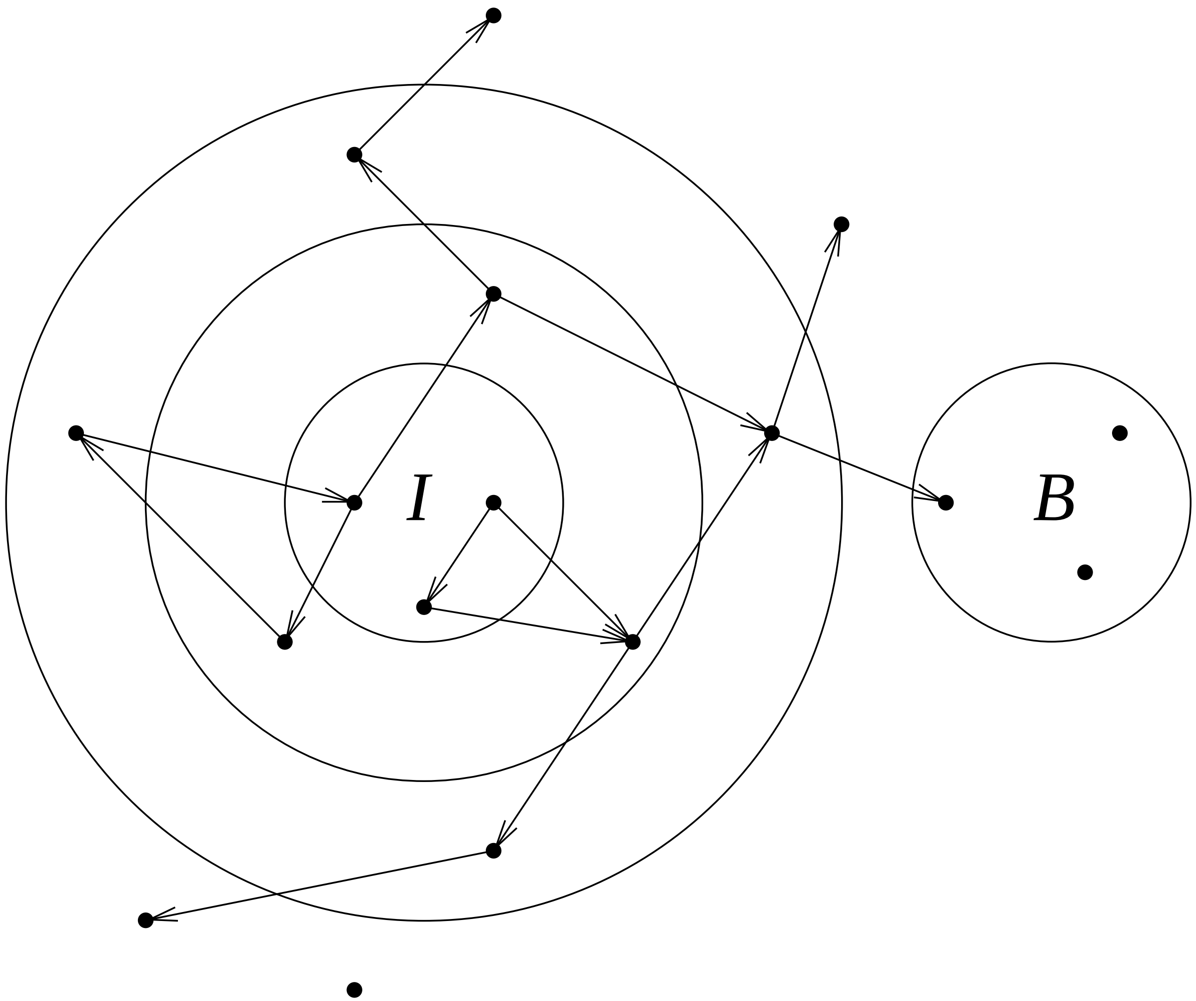
Forward Fixpoint: Step 1



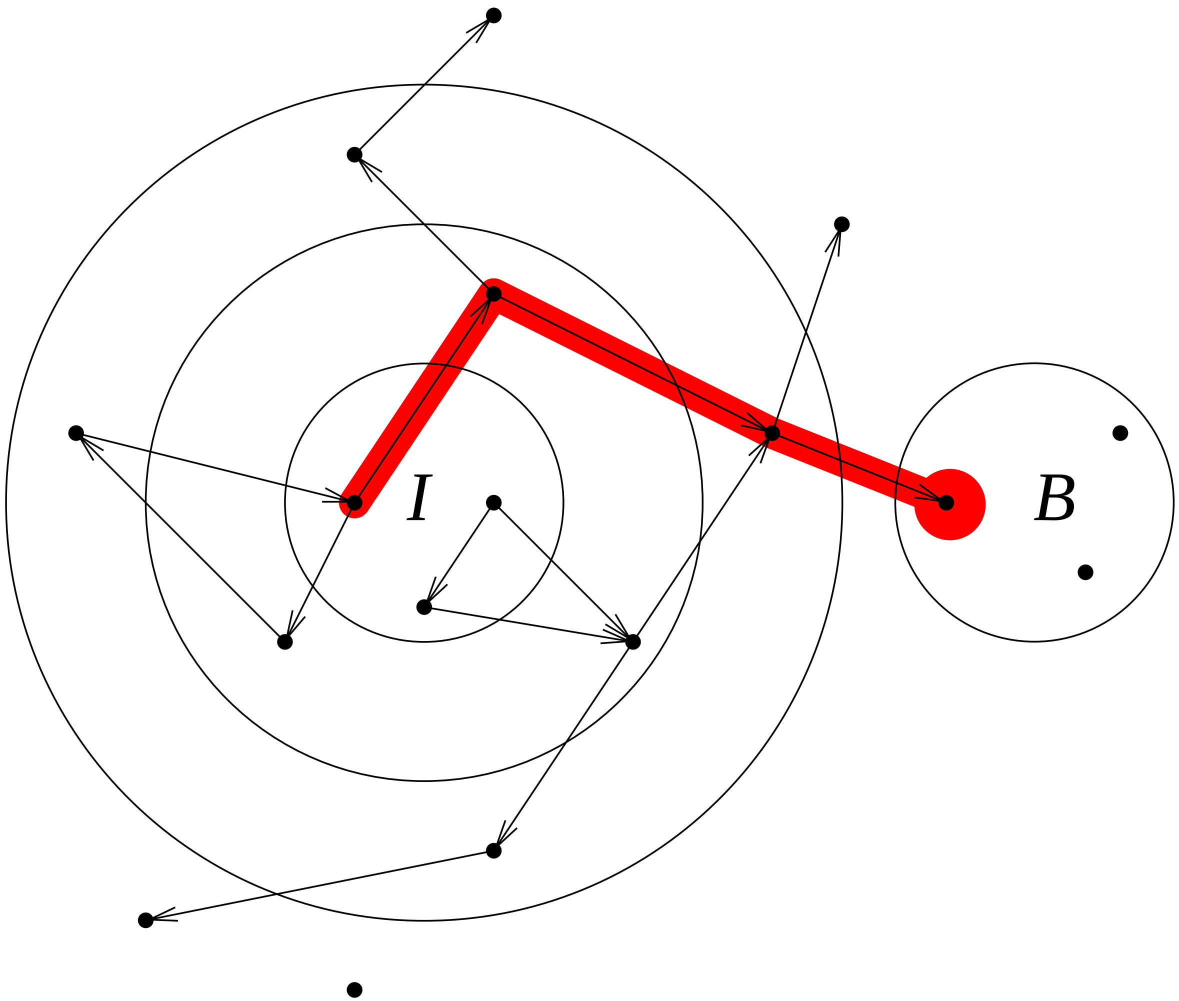
Forward Fixpoint: Step 2



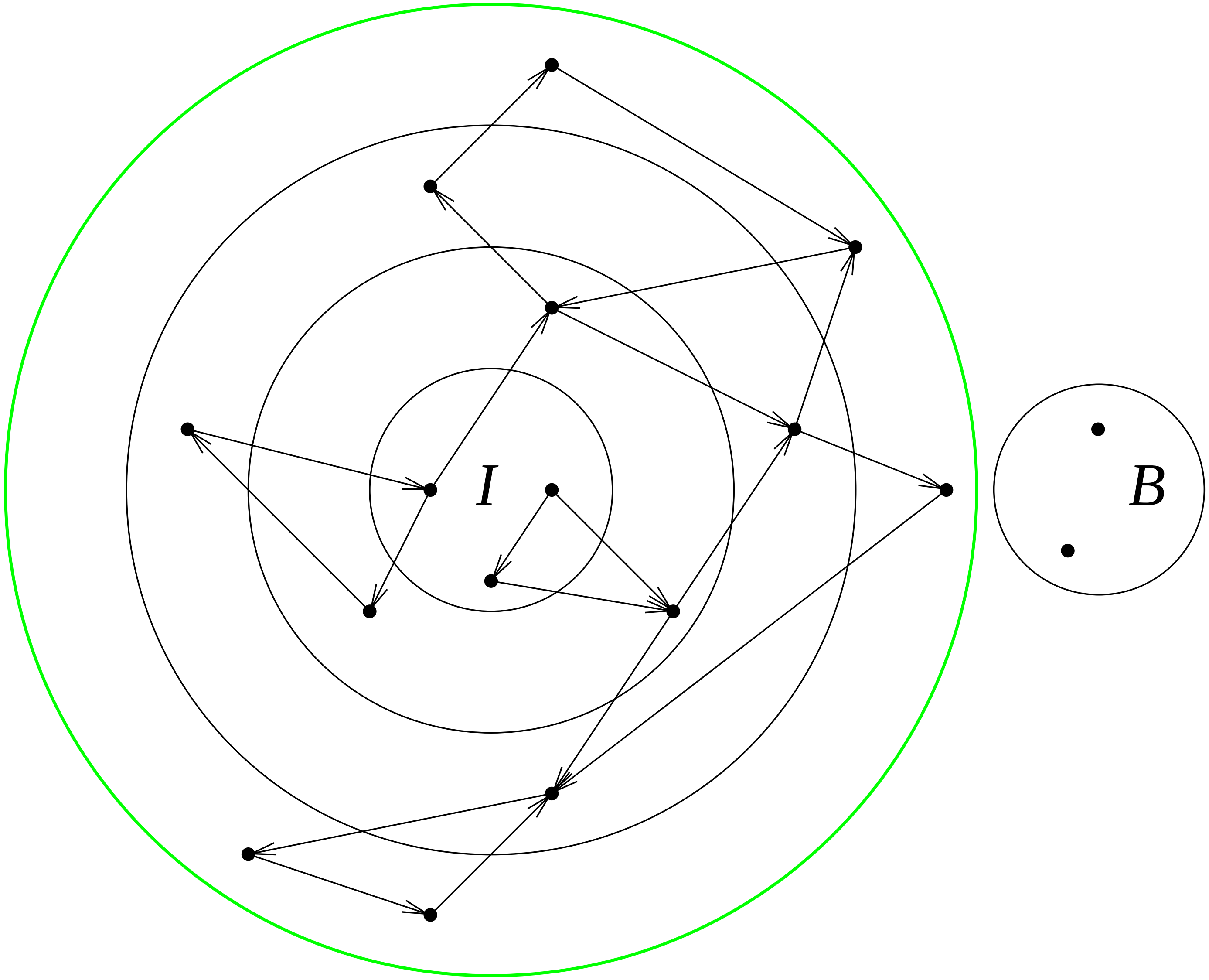
Forward Fixpoint: Step 3



Forward Fixpoint: Bad State Reached



Forward Fixpoint: Termination, No Bad State Reachable



initial states I, transition relation T, bad states B

<u>model-check</u>$^{\mu}_{\text{forward}}$ (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";

$S_C = S_N$;

$S_N = S_C \cup Img(S_C)$;

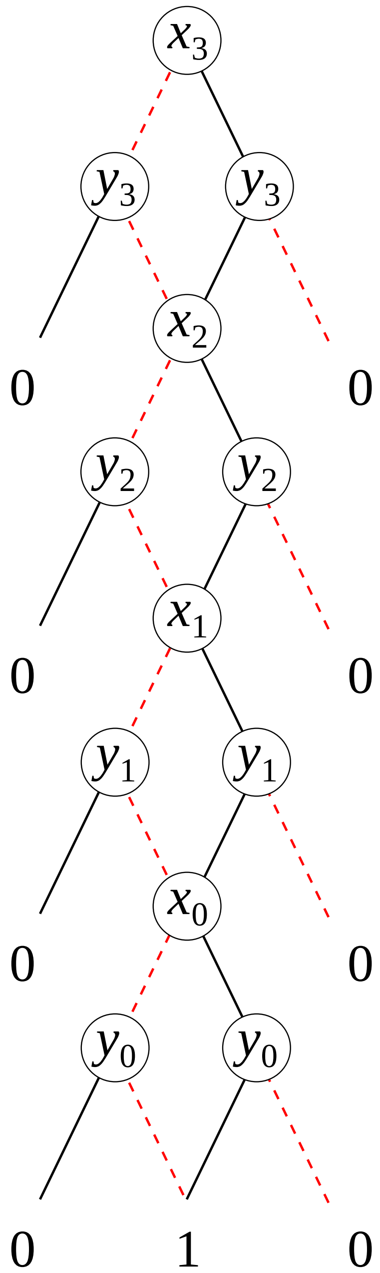
done;

return "no bad state reachable";

Symbolic Model Checking

- 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:

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

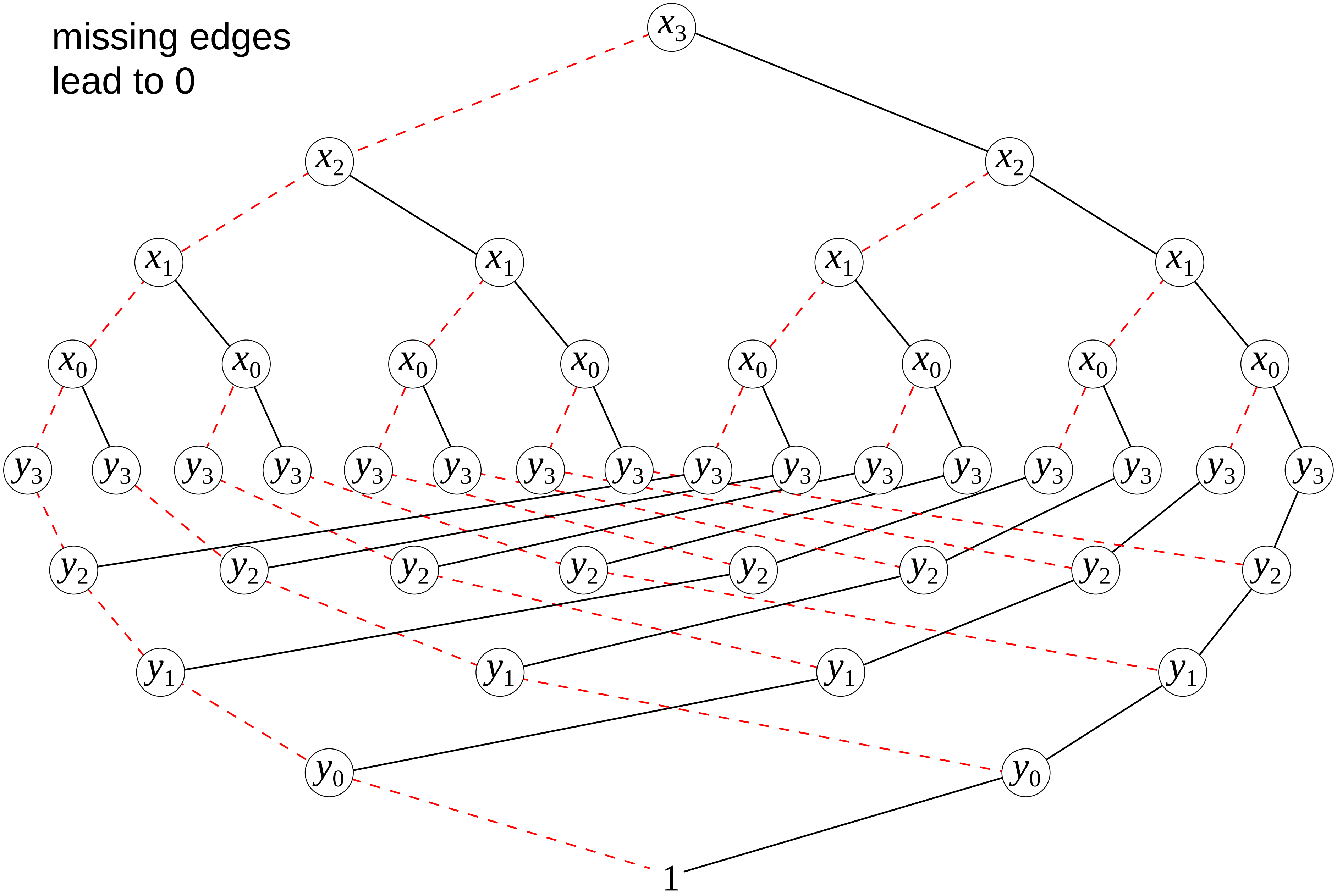
interleaved variable order:

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

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

Exponential BDD for Bit-Vector Comparison

missing edges
lead to 0



Unrolling of Forward Least Fixpoint Algorithm

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?	$B \cap S_N^1 \neq \emptyset$	$\exists s_0, s_1, s_2[I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge B(s_2)]$

Falsification Part of Fixpoint Algorithm

[BiereCimattiClarkeZhu-TACAS'99]

~~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? $S_C^2 \neq S_N^2$ $\exists s_0, s_1, s_2[I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge$~~
$\neg (I(s_2) \vee \exists t_0[I(t_0) \wedge T(t_0, s_2)])]$

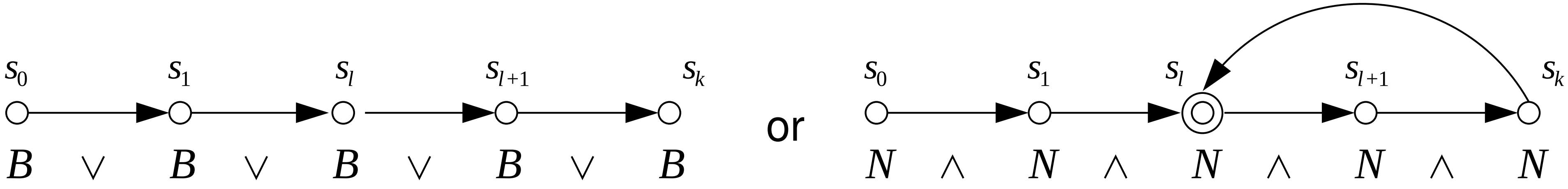
~~2: terminate? $S_C^2 = S_N^2$ $\forall s_0, s_1, s_2[I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \rightarrow$~~
~~$I(s_2) \vee \exists t_0[I(t_0) \wedge T(t_0, s_2)]]$~~

2: bad state? $B \cap S_N^1 \neq \emptyset$ $\exists s_0, s_1, s_2[I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge B(s_2)]$

Bounded Model Checking

[BiereCimattiClarkeZhu-TACAS'99]

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



- simple for safety properties: bad state B is reachable

$$\mathrm{BMC}(k): \qquad I(s_0) \wedge T(s_0, s_1) \wedge \cdots \wedge T(s_{k-1}, s_k) \wedge \bigvee_{i=0}^{k} B(s_i)$$

- harder for liveness properties cycle with no progress states N reachable

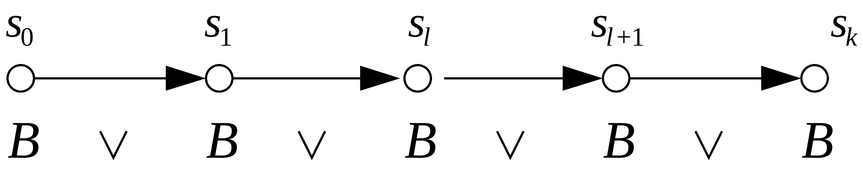
$$I(s_0) \wedge T(s_0, s_1) \wedge \cdots \wedge T(s_{k-1}, s_k) \wedge \bigwedge_{i=0}^{k} N(s_i) \wedge \exists l \; T(s_k, s_l)$$

- can also encode liveness into safety [BiereArthoSchuppan'01]

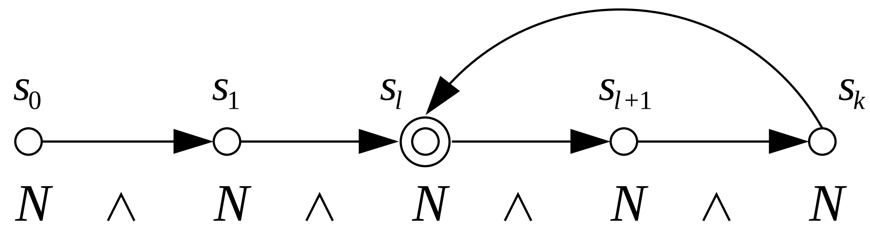
Bounded Model Checking

[BiereCimattiClarkeZhu-TACAS'99]

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



or



- simple for safety properties: bad state B is reachable

$$\mathrm{BMC}(k): \qquad I(s_0) \wedge T(s_0, s_1) \wedge \cdots \wedge T(s_{k-1}, s_k) \wedge \bigvee_{i=0}^{k} B(s_i)$$

- harder for liveness properties cycle with no progress states N reachable

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

- can also encode liveness into safety [BiereArthoSchuppan'01]

Bounded Model Checking State of the Art

see also Chapter 14 on BMC in Handbook of Satisfiability

- increase in efficiency of SAT solvers [Grasp,zChaff,MiniSAT,SatELite,...]

- 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, better encodings, e.g. [LatvalaBiereHeljankoJuntilla-FMCAD'04]

- other models, e.g. C/C++/Verilog [Kröning...], hybrid automata [Audemard...-BMC'04]

Induction with SAT

[SinghSheeranStalmarck'00]

- more specifically k-induction
  - does there exist k such that the following formula is *unsatisfiable*

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

  - if *unsatisfiable* and $\mathrm{BMC}(k)$ *unsatisfiable* then bad state unreachable

- bound on k: length of longest cycle free path = reoccurrence diameter

- $k = 0$ check whether $\neg B$ tautological (propositionally)

- $k = 1$ check whether $\neg B$ inductive for T

Interpolation in Model Checking

[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 many cases [HWMCC'08]

- 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)

$A \wedge B$ *unsatisfiable* then f is an interpolant iff

$$\text{(I1)} \quad A \Rightarrow f \qquad \text{and} \qquad \text{(I2)} \quad B \wedge f \Rightarrow \bot$$

an interpolating quadruple $(A,B)\; c\; [\,f\,]$ is well formed if

$$\text{(W1)} \quad V(c) \subseteq V(A) \cup V(B) \qquad \text{and} \qquad \text{(W2)} \quad V(f) \subseteq G \cup V(c) \qquad \text{with } G = V(A) \cap V(B)$$

an interpolating quadruple $(A,B)\; c\; [\,f\,]$ is valid if

$$\text{(V1)} \quad A \Rightarrow f \qquad \text{and} \qquad \text{(V2)} \quad B \wedge f \Rightarrow c$$

proof rules which produce well formed and valid interpolating quadruples:

$$\text{(R1)} \quad \frac{}{(A,B)\; c\; [\,c\,]} \; c \in A \qquad \frac{(A,B)\; c \,\dot\vee\, l\; [\,f\,] \quad (A,B)\; d \,\dot\vee\, \bar{l}\; [\,g\,]}{(A,B)\; c \vee d\; [\,f \wedge g\,]} \; |l| \in V(B) \quad \text{(R3)}$$

$$\text{(R2)} \quad \frac{}{(A,B)\; c\; [\,\top\,]} \; c \in B \qquad \frac{(A,B)\; c \,\dot\vee\, l\; [\,f\,] \quad (A,B)\; d \,\dot\vee\, \bar{l}\; [\,g\,]}{(A,B)\; c \vee d\; [\,f|_{\bar{l}} \vee g|_{l}\,]} \; |l| \not\in V(B) \quad \text{(R4)}$$

Mining Inductive Invariants

- through abstract interpretation resp. static analysis, or alternatively . . .

- randomly simulate model and extract potential invariants
  - signals / predicates which always hold
  - implications of signals / predicates that occur in the simulation / tests
  - equivalent signals (works well in sequential equivalence checking)

- prove them to be k-inductive
  - quite natural in sequential equivalence checking for circuits
  - synthesis algorithms also only see finitely many time steps

- how to obtain environment model / constraints / contracts?

Using Inductive Invariants

- inductive invariants help to speed-up both k-induction (and interpolation)

- let P be inductive: $I(s) \Rightarrow P(s)$ and $T(s,s') \wedge P(s) \Rightarrow P(s')$

- we want to prove that a bad state can not reached

- if $\mathrm{BMC}(k)$ is *unsatisfiable* it is enough to prove *unsatisfiability* of

$$
\begin{array}{l}
P(s_0) \wedge \\
P(s_1) \wedge \cdots \wedge P(s_k) \wedge \\
\overline{B(s_0)} \wedge \cdots \wedge \overline{B(s_{k-1})} \wedge \\
T(s_0,s_1) \wedge \cdots \wedge T(s_{k-1},s_k) \wedge B(s_k) \wedge \bigwedge\limits_{0 \le i < j \le k} s_i \neq s_j
\end{array}
$$

- this formula can become *unsatisfiable* much earlier, i.e. for smaller k, than

$$
\begin{array}{l}
\overline{B(s_0)} \wedge \cdots \wedge \overline{B(s_{k-1})} \wedge \\
T(s_0,s_1) \wedge \cdots \wedge T(s_{k-1},s_k) \wedge B(s_k) \wedge \bigwedge\limits_{0 \le i < j \le k} s_i \neq s_j
\end{array}
$$

Simple Path Constraints

[BiereBrummayer-FMCAD'08]

- bounded model checking: [BiereCimattiClarkeZhu'99]

$$I(s_1) \wedge T(s_1, s_2) \wedge \ldots \wedge T(s_{k-1}, s_k) \wedge \bigvee_{0 \le i \le k} B(s_i)$$

satisfiable?

- reoccurrence diameter checking: [BiereCimattiClarkeZhu'99]

$$T(s_1, s_2) \wedge \ldots \wedge T(s_{k-1}, s_k) \wedge \bigwedge_{1 \le i < j \le k} s_i \ne s_j$$

unsatisfiable?

- k-induction base case: [SheeranSinghStålmarck'00]

$$I(s_1) \wedge T(s_1, s_2) \wedge \ldots \wedge T(s_{k-1}, s_k) \wedge B(s_k) \wedge \bigwedge_{0 \le i < k} \neg B(s_i)$$

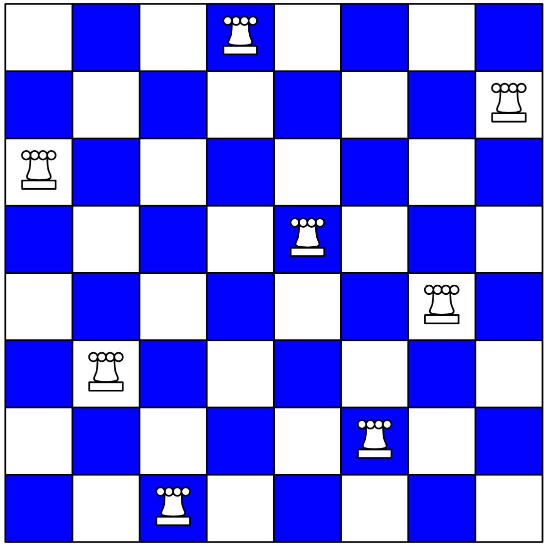
satisfiable?

- k-induction induction step: [SheeranSinghStålmarck'00]

$$T(s_1, s_2) \wedge \ldots \wedge T(s_{k-1}, s_k) \wedge B(s_k) \wedge \bigwedge_{0 \le i < k} \neg B(s_i) \wedge \bigwedge_{1 \le i < j \le k} s_i \ne s_j$$

unsatisfiable?

- classical concept in constraint programming:
  - k variables over a domain of size m supposed to have different values
  - for instance k-queen problem



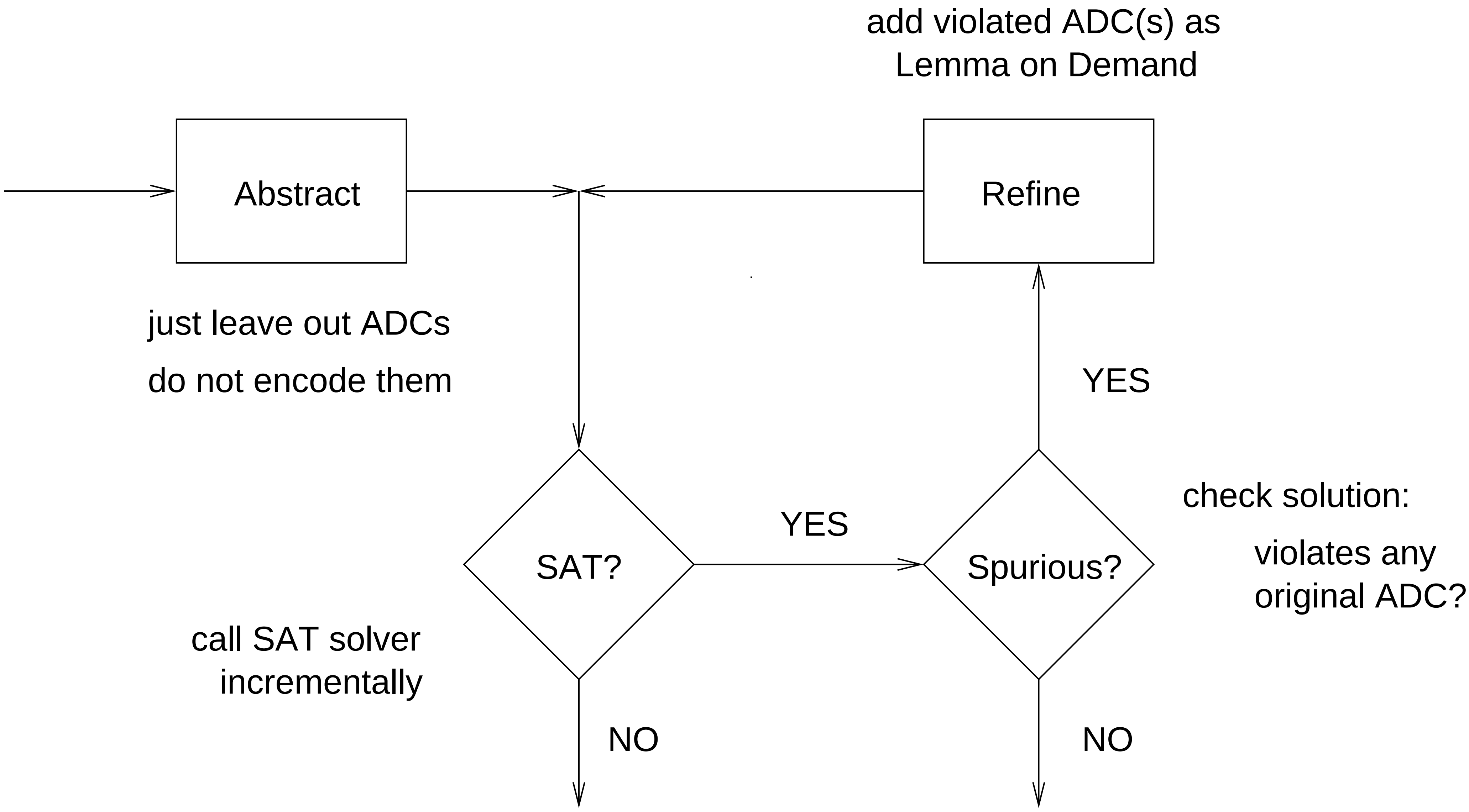
- propagation algorithms to establish arc-consistency
  - explicit propagators: [Régin'94]
    * $\mathrm{O}(k \cdot m)$ space
    * $\mathrm{O}(k^2 \cdot m^2)$ time
  - symbolic propagators: [GentNightingale'04] also [MarquesSilvaLynce'07]
    * one-hot CNF encoding with $\Omega(k \cdot m)$ boolean variables
- in model checking $k \ll m$ typically $k < 1000 \quad m = 2^n > 2^{100}$ n latches

Symbolic ADCs for Large Domains

- encoding bit-vector inequalities directly:
  - let u, v be two n-bit vectors, $d_0, \ldots, d_{n-1}$ fresh boolean variables

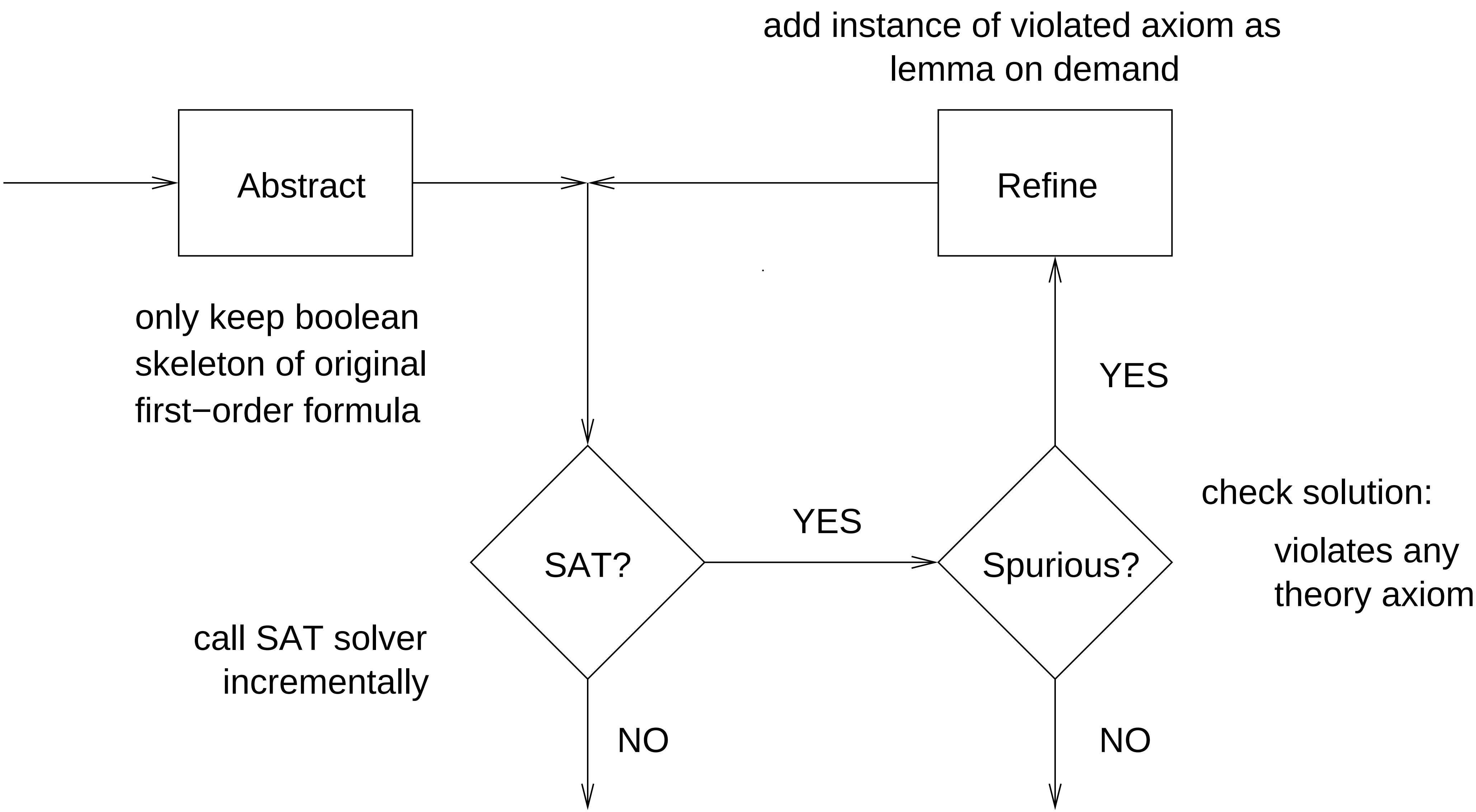
    $u \neq v$ is equisatisfiable to $(d_0 \vee \cdots \vee d_{n-1}) \wedge \bigwedge_{j=0}^{n-1} (u_j \vee v_j \vee \overline{d_j}) \wedge (\overline{u_j} \vee \overline{v_j} \vee \overline{d_j})$
  - can be extended to encode Ackermann Constraints + McCarthy Axioms
  - either **eagerly** encode all $s_i \neq s_j$ quadratic in k
  - or **refine** adding bit-vector inequalities on demand [EénSörensson-BMC'03]
- natively handle ADCs within SAT solver: main contribution in FMCAD'08
  - similar to theory consistency checking in lazy SMT vs. "lemmas on demand"
  - can be extended to also perform theory propagation
- sorting networks ineffective in our experience [KröningStrichman'03,JussilaBiere'06]

Lemmas on Demand for ADCs



Lemmas on Demand for Satisfiability Modulo Theory (SMT)

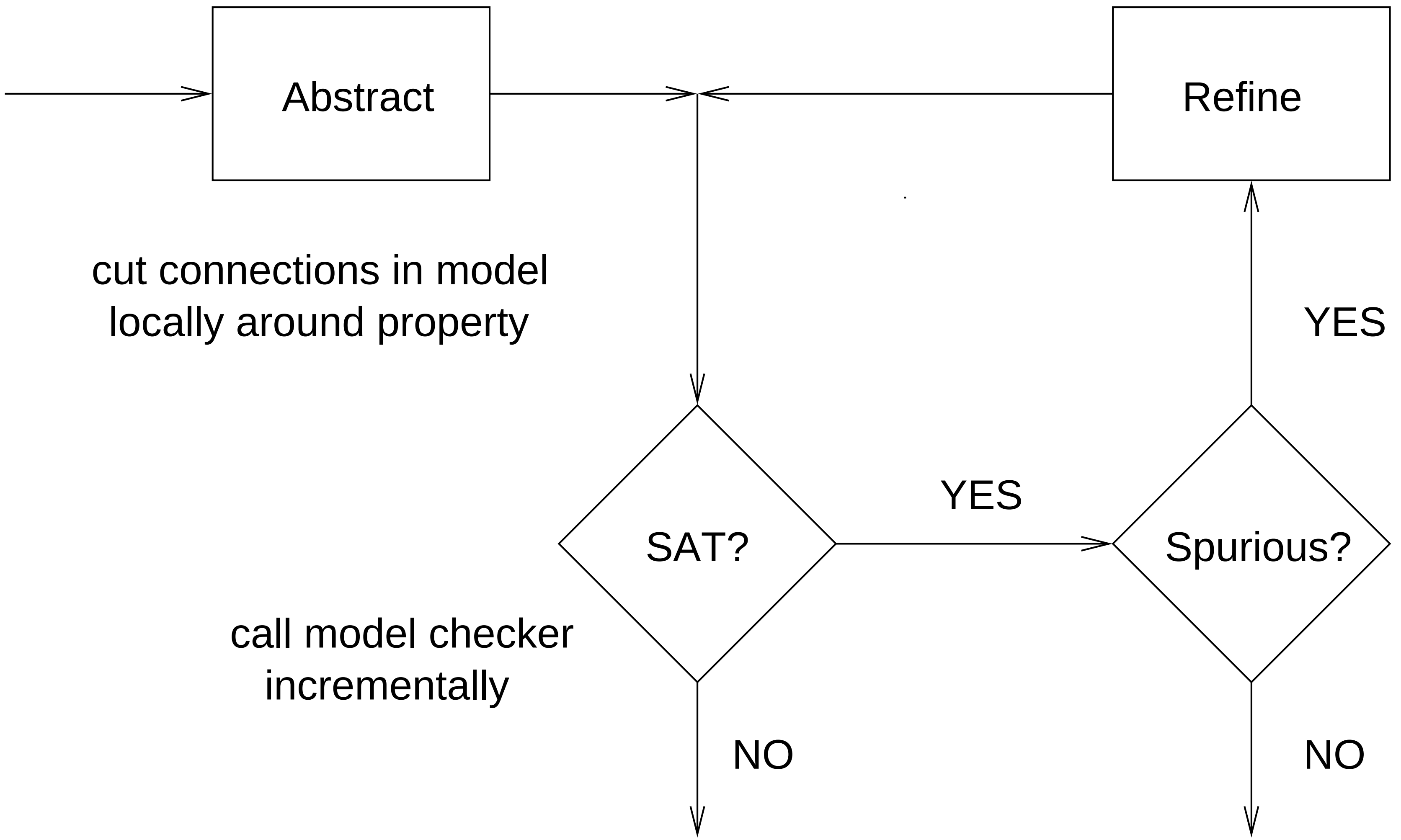
[DeMouraRueß-SAT'02] [BarrettDillStump-CAV'02] ...



Localization / Counter Example Guided Abstraction Refinement

Localization [Kurshan'93], Predicate Abstraction [GrafSaidi'97],
SLAM [BallRajamani'01], CEGAR [ClarkeGrumbergJhaLuVeith'03]

add more logic of model as
lemma on demand



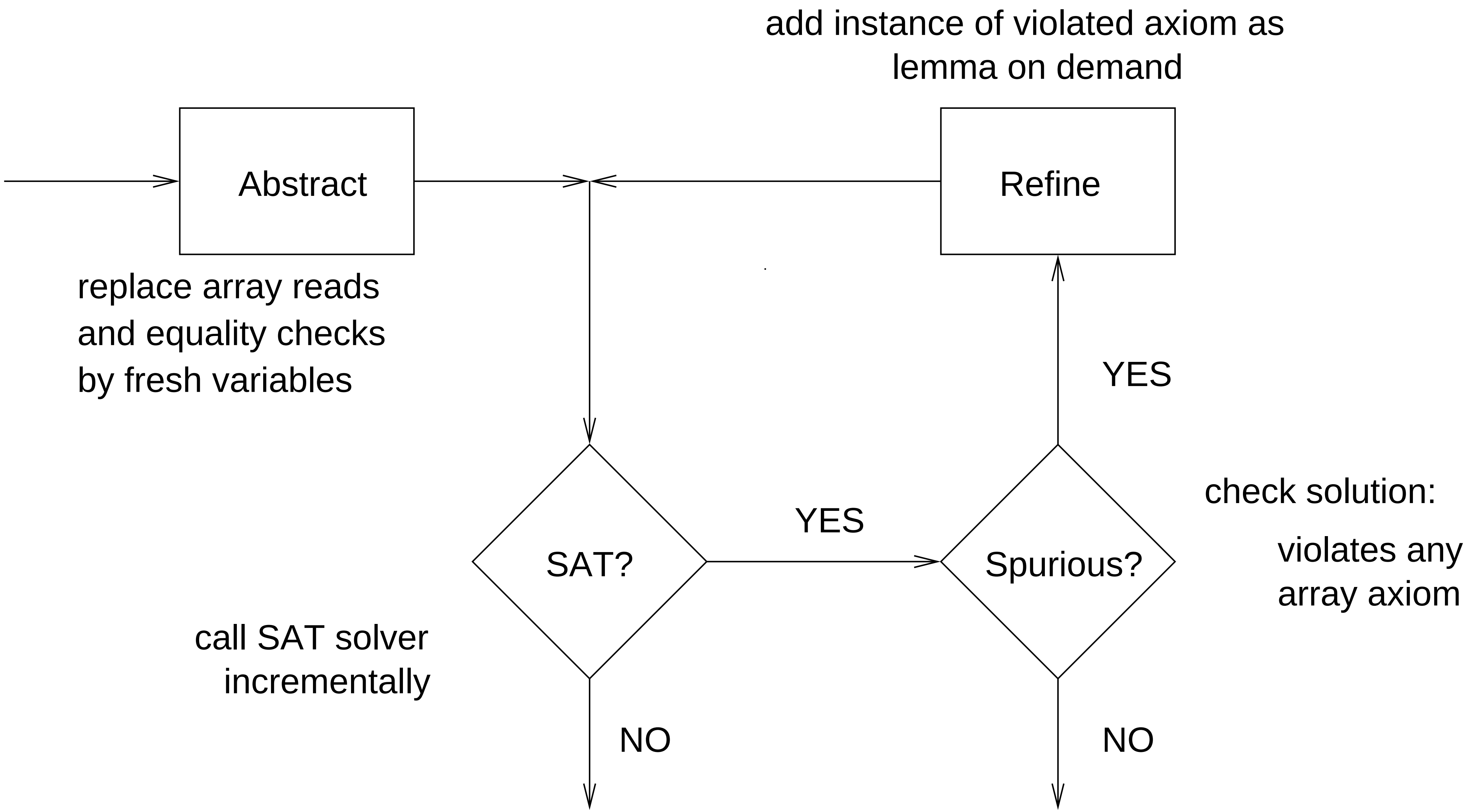
cut connections in model
locally around property

call model checker
incrementally

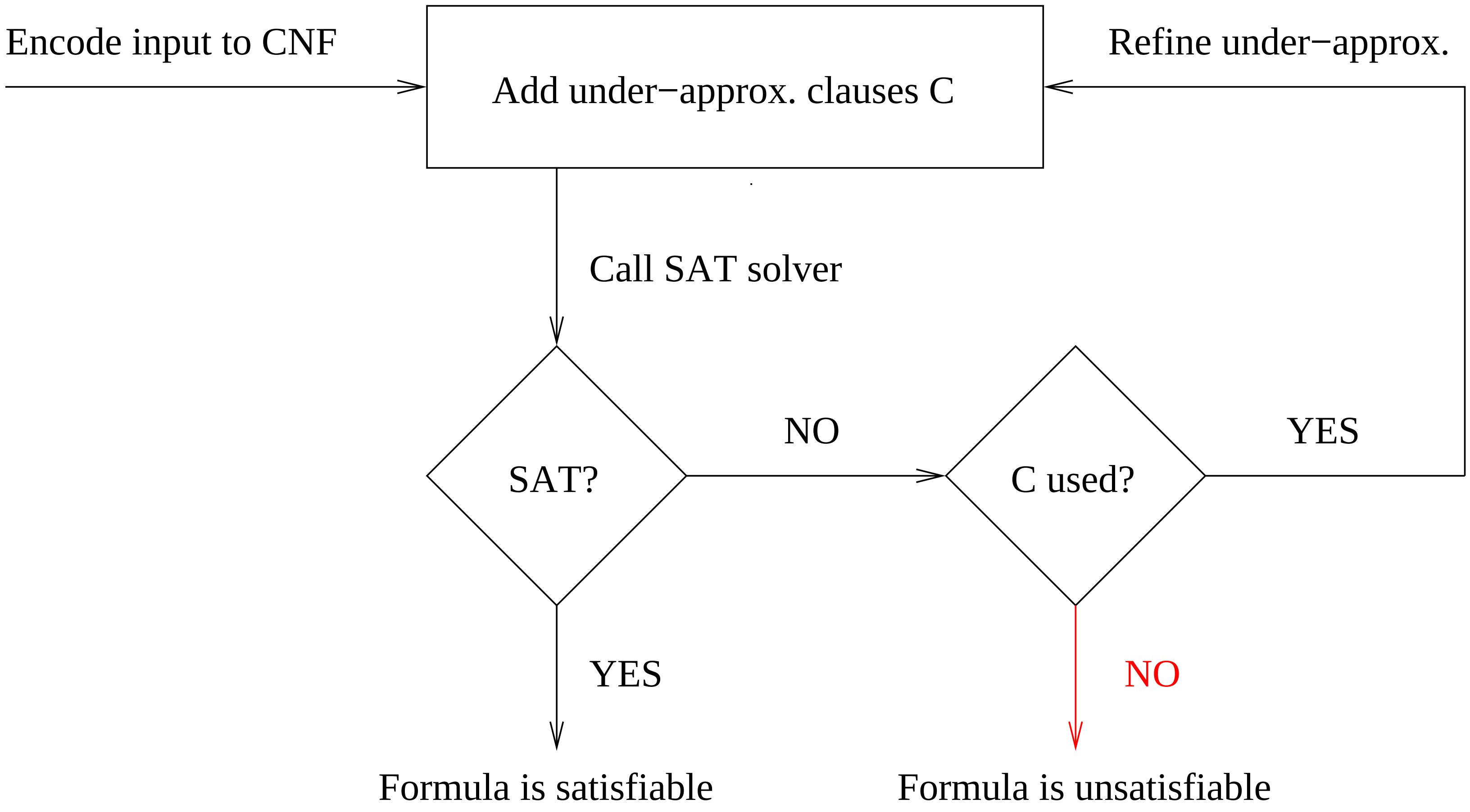
check solution:

impossible to
extend
to full model

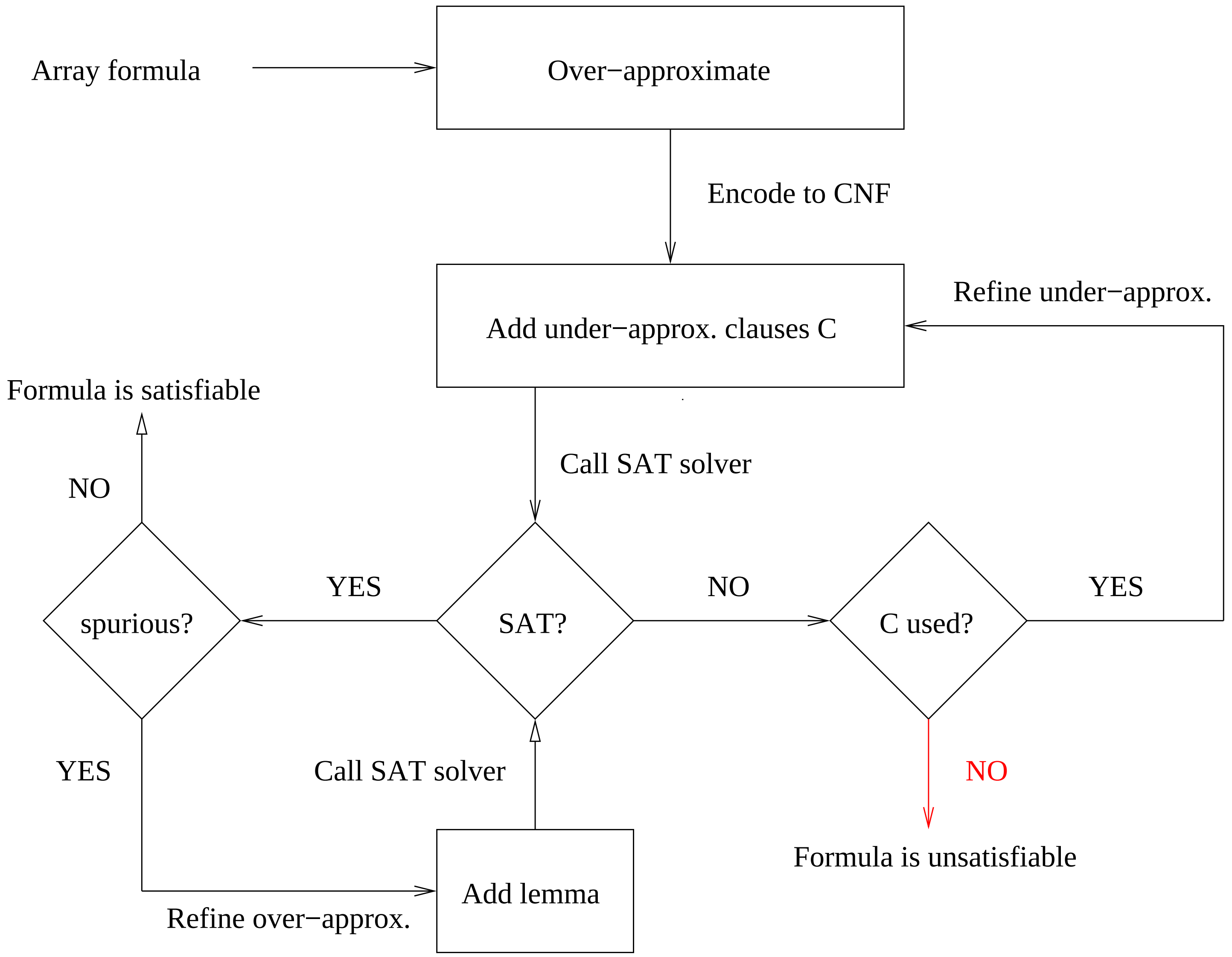
Lemmas on Demand for Extensional Arrays



Early Unsat Termination



Combining Over- and Under-Approximation

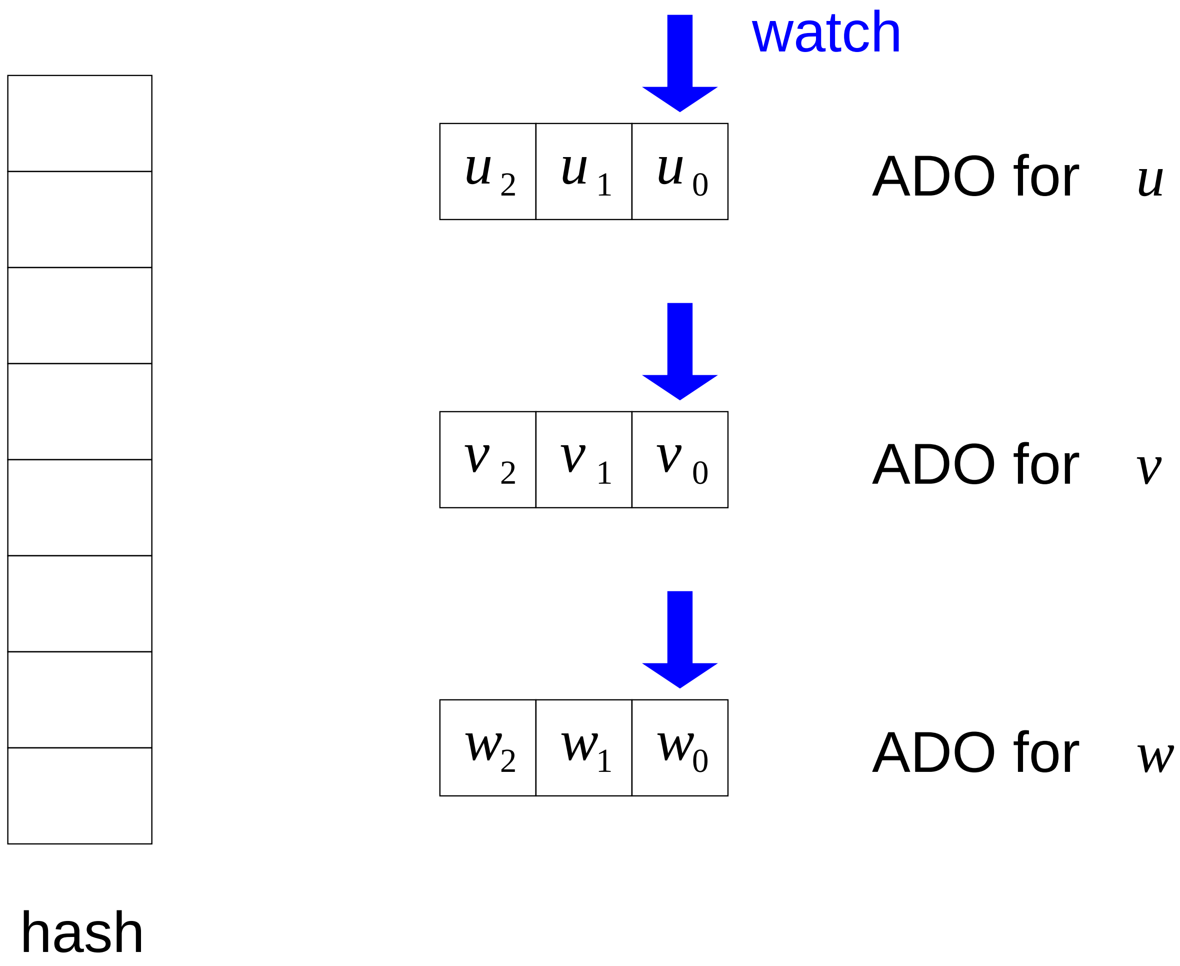


Lazy SMT

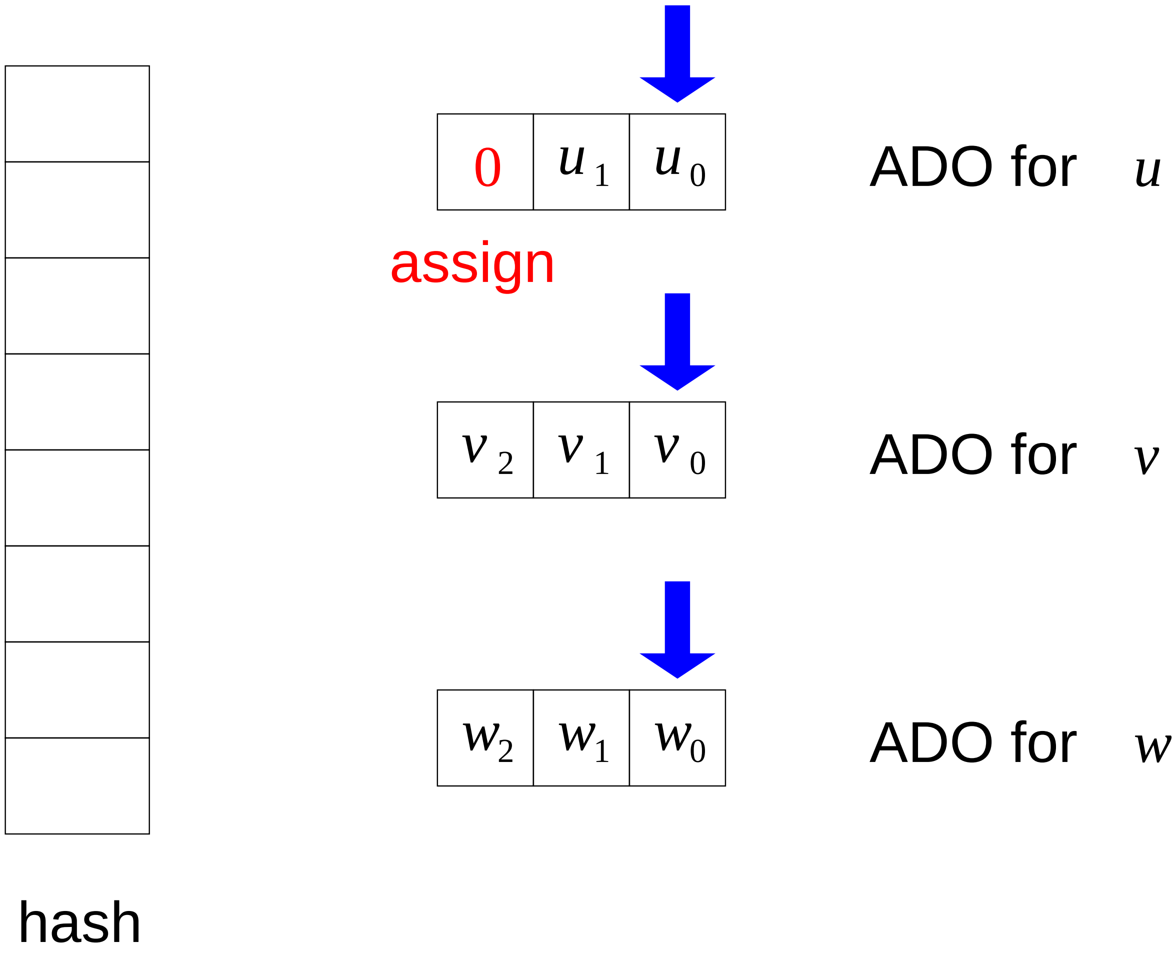
survey on lazy SMT [Sebastiani-JSAT'07]

- Lemmas on Demand are as lazy as it gets
  - SAT solver enumerates full models of propositional skeleton
  - abstracted lemmas are added / learned on demand
  - theory solver checks consistency of conjunction of theory literals
- on-the-fly consistency checking
  - additionally theory solver checks consistency of partial model as well
- theory propagation
  - theory solver even deduces and notifies SAT solver about implied values of literals
- generic framework: DPLL(T) [NieuwenhuisOliverasTinelli-JACM'06]

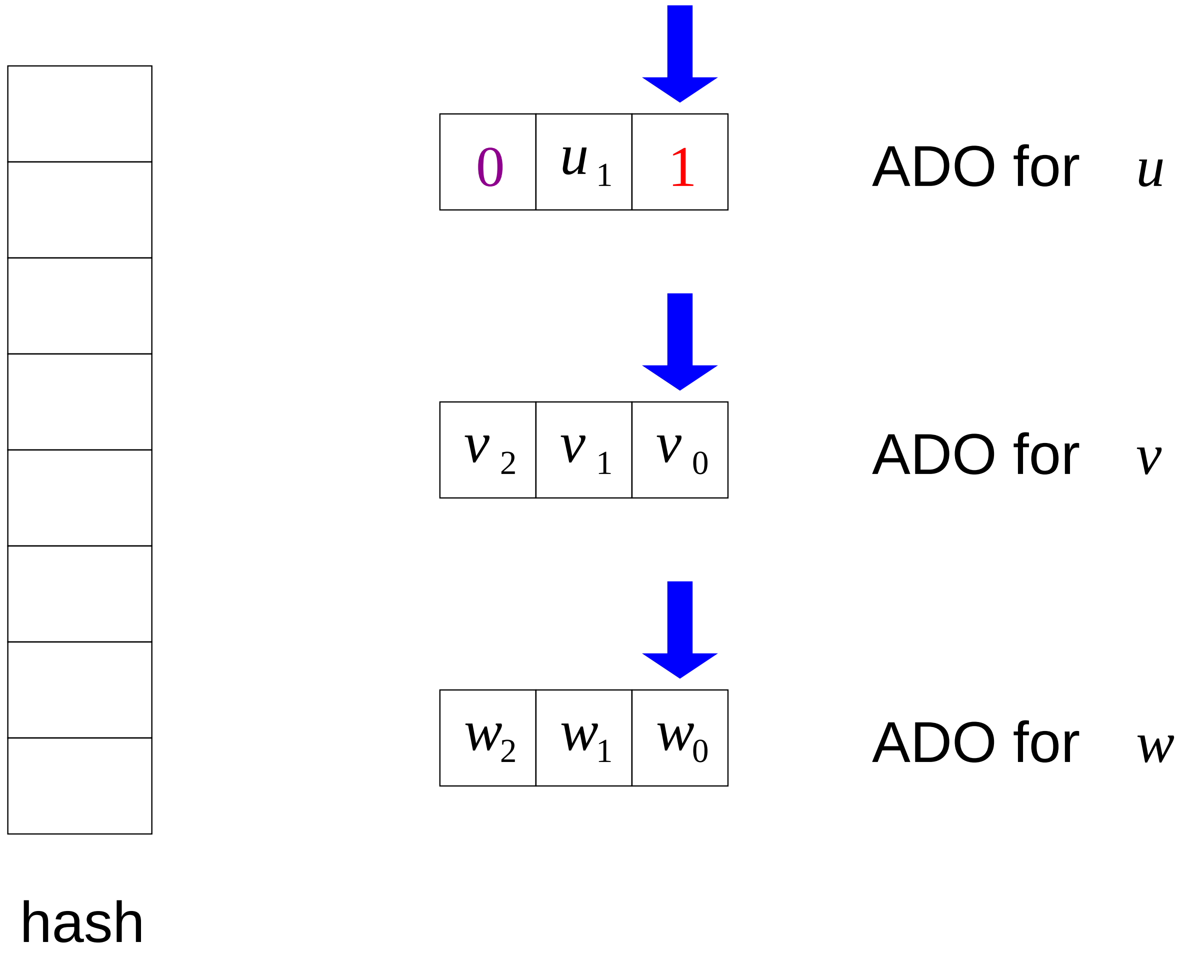
All Different Objects (ADOs)



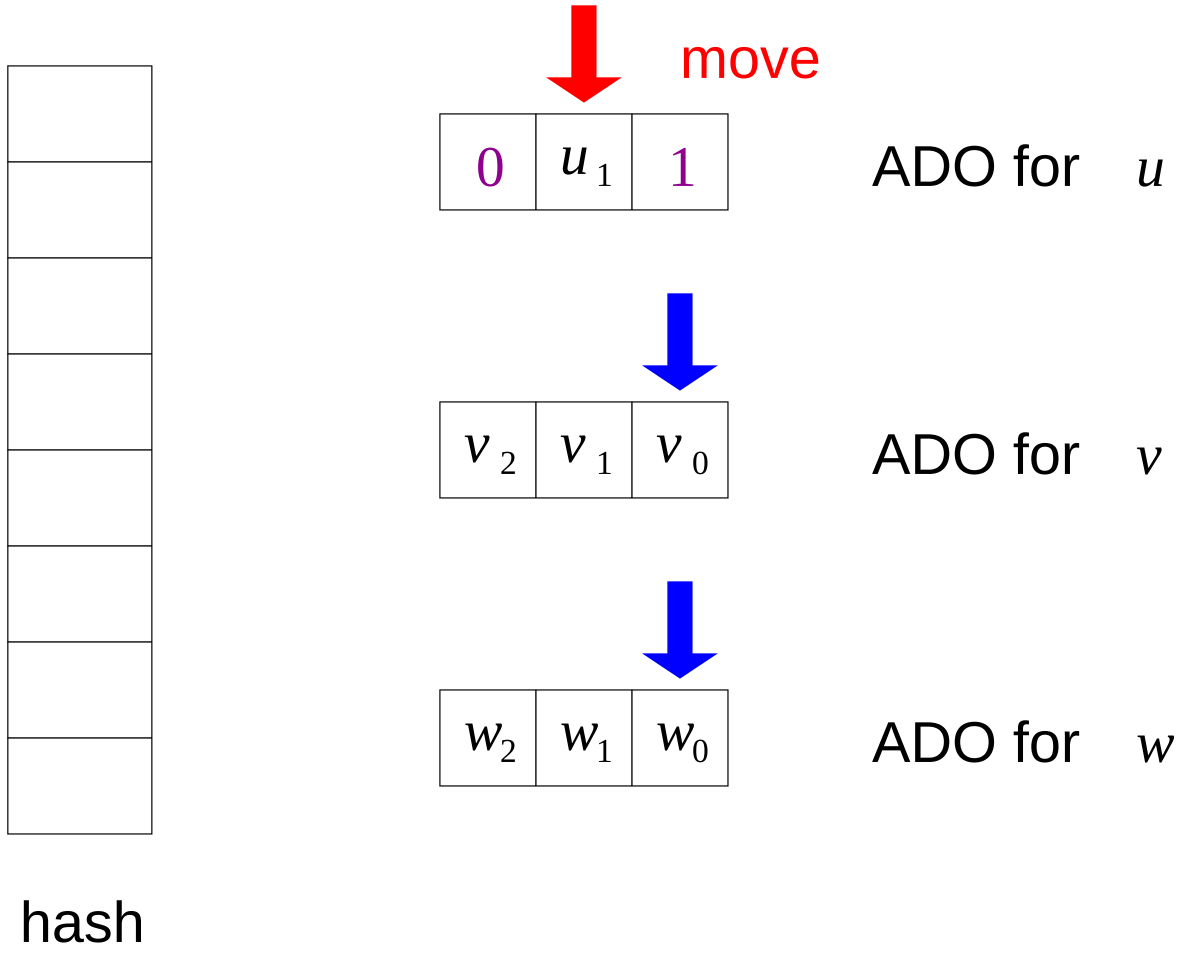
All Different Objects (ADOs)



All Different Objects (ADOs)

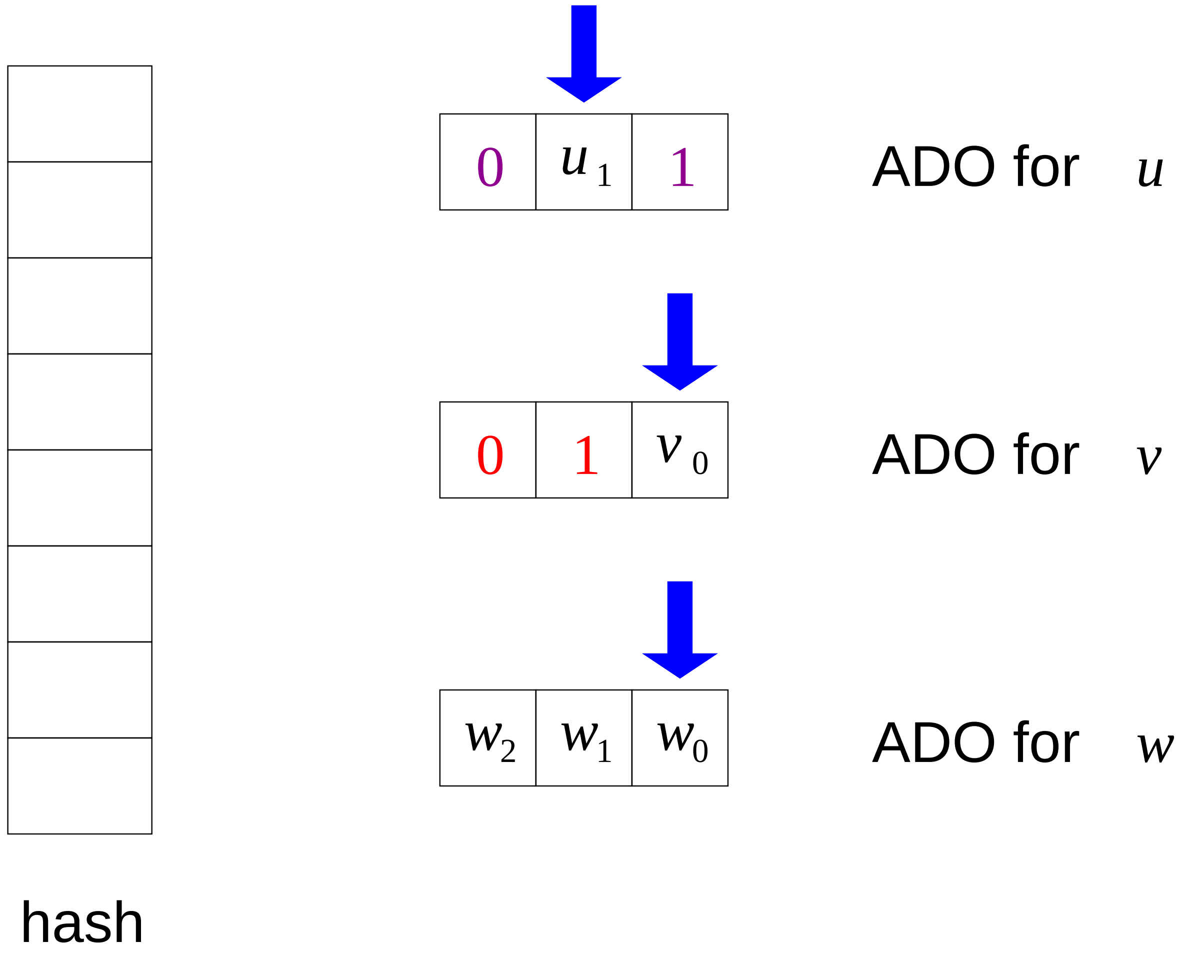


All Different Objects (ADOs)

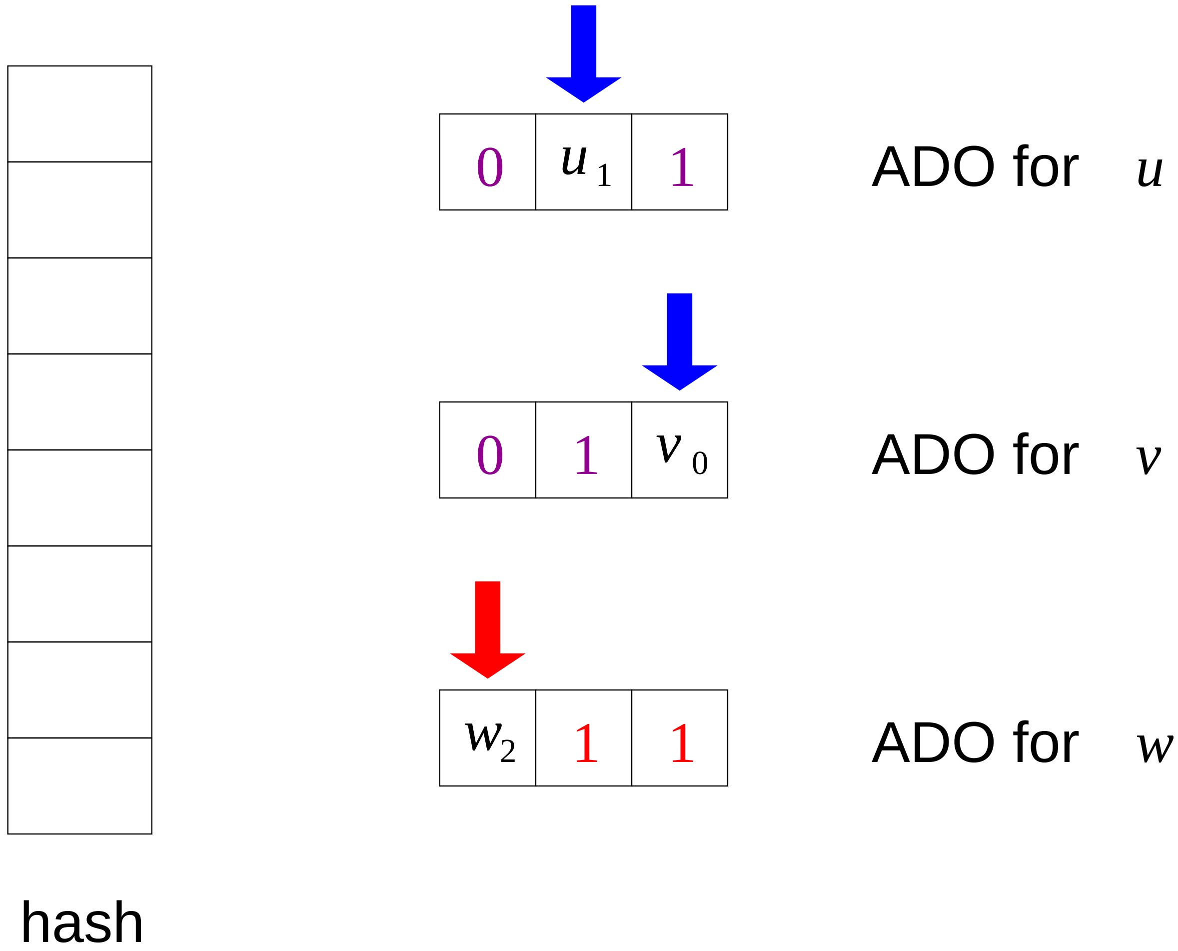


All Different Objects (ADOs)

[BiereBrummayer-FMCAD'08]

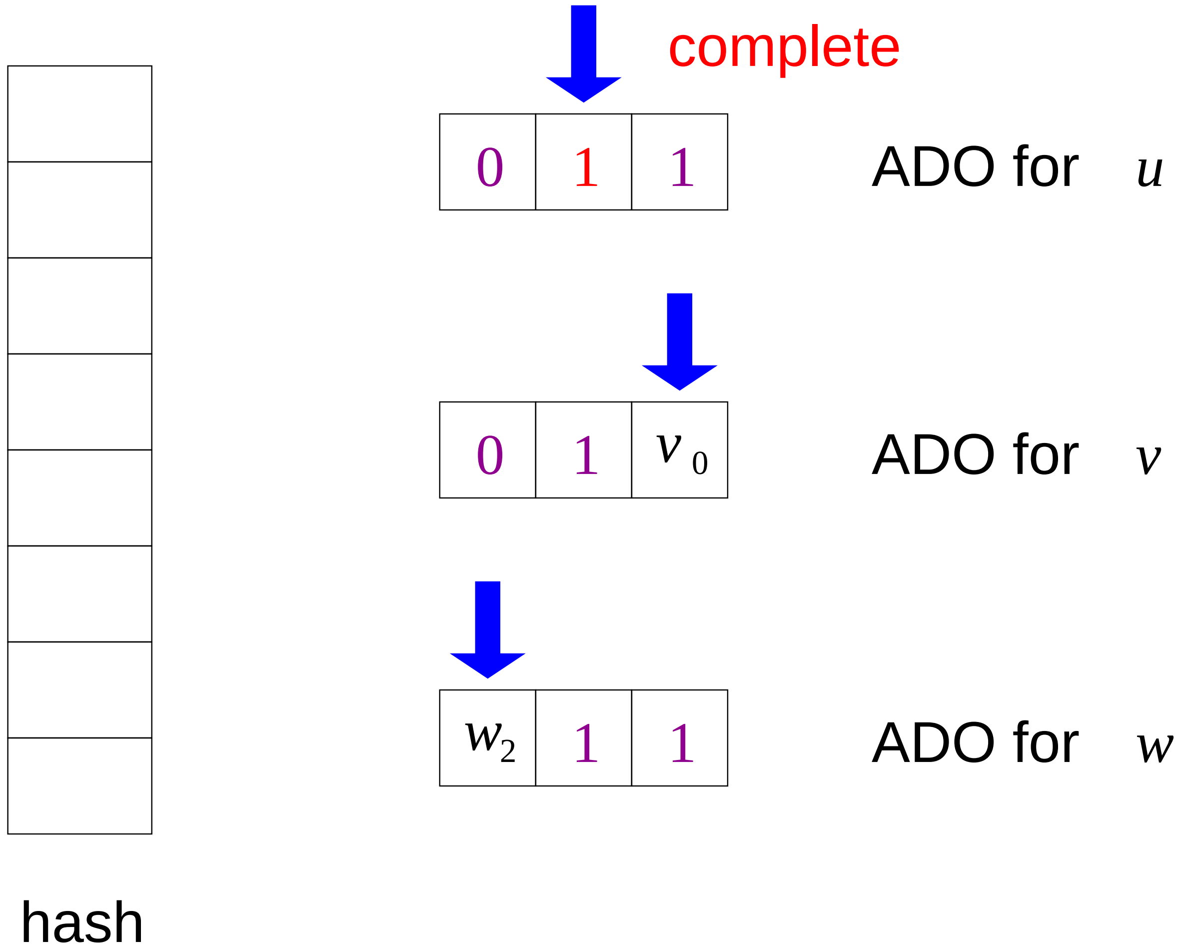


All Different Objects (ADOs)

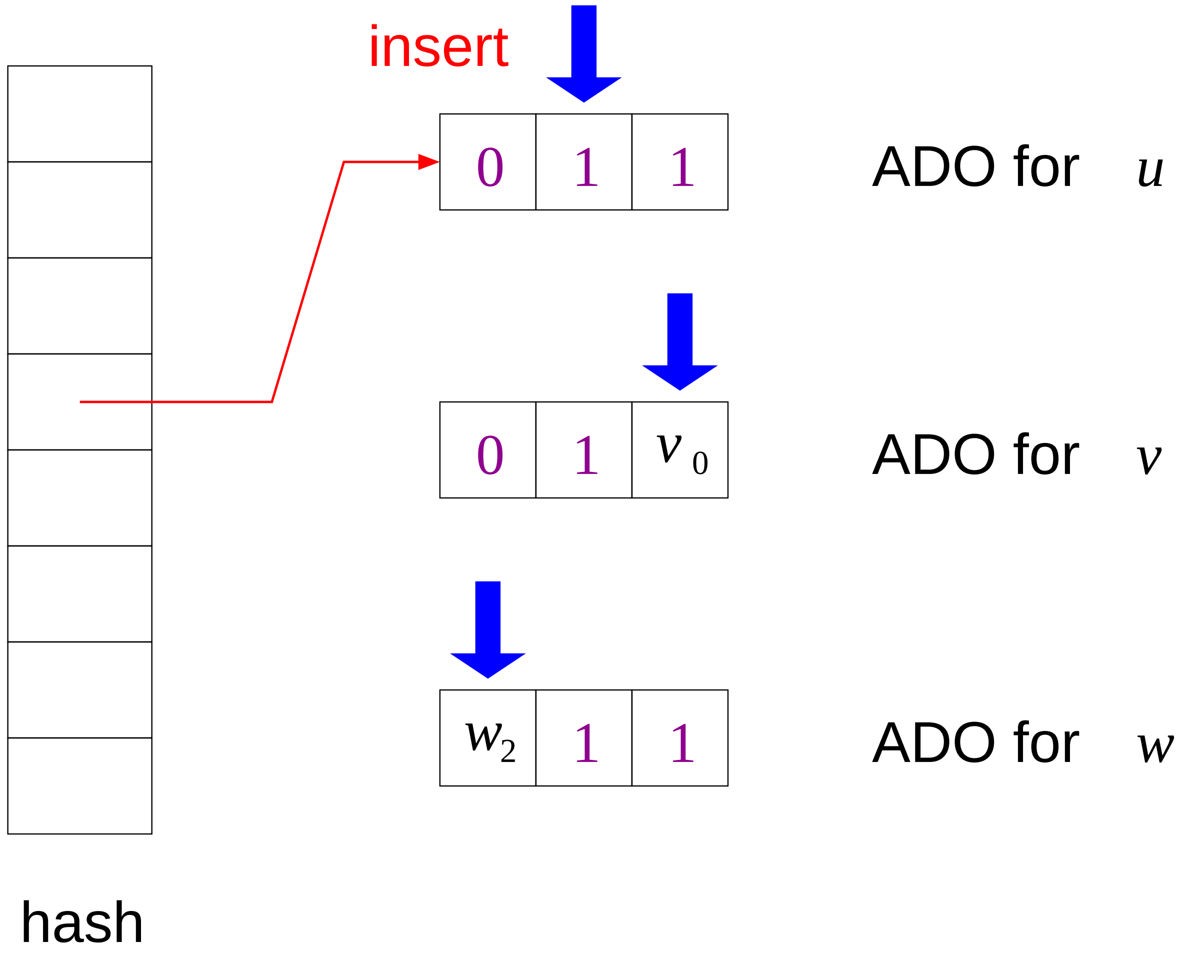


All Different Objects (ADOs)

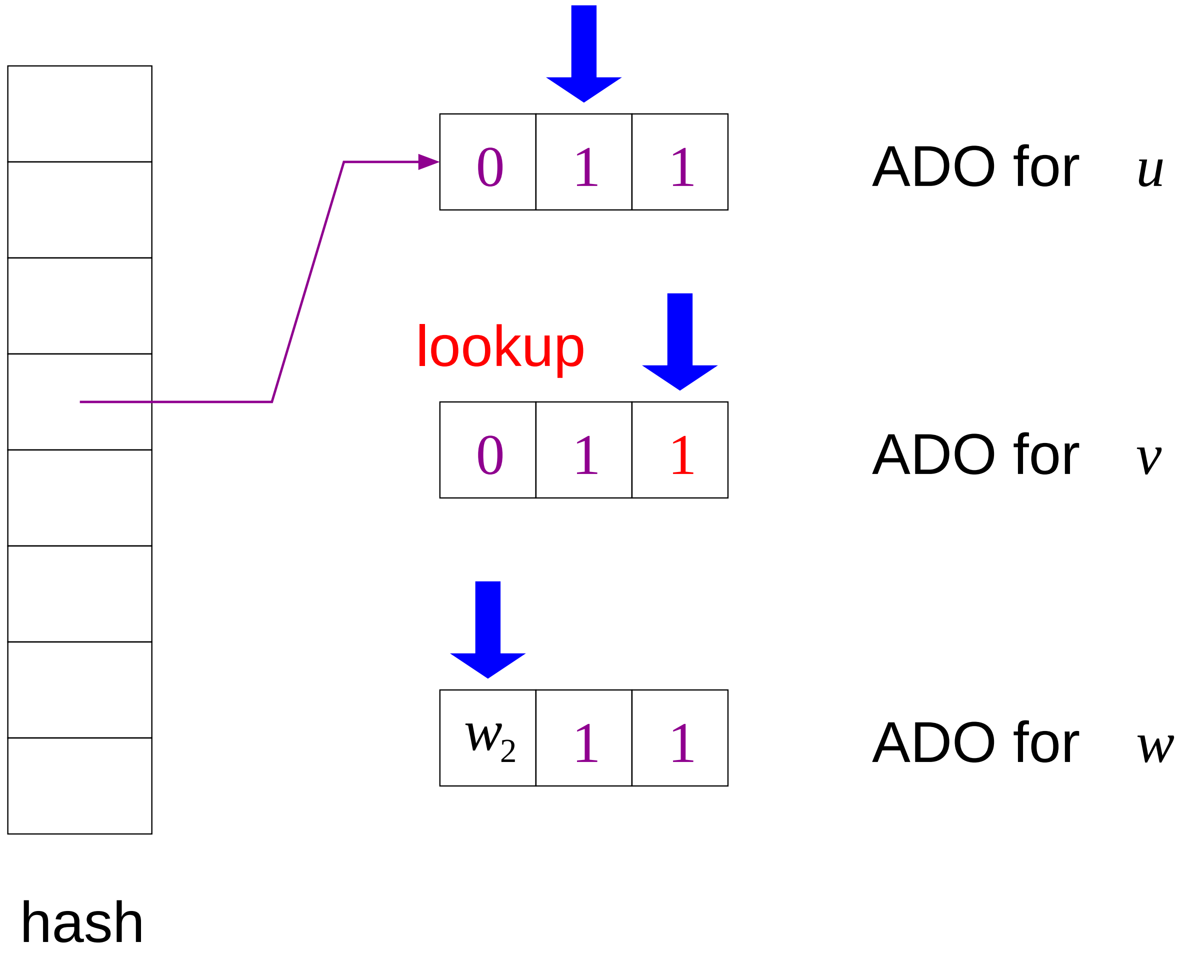
[BiereBrummayer-FMCAD'08]



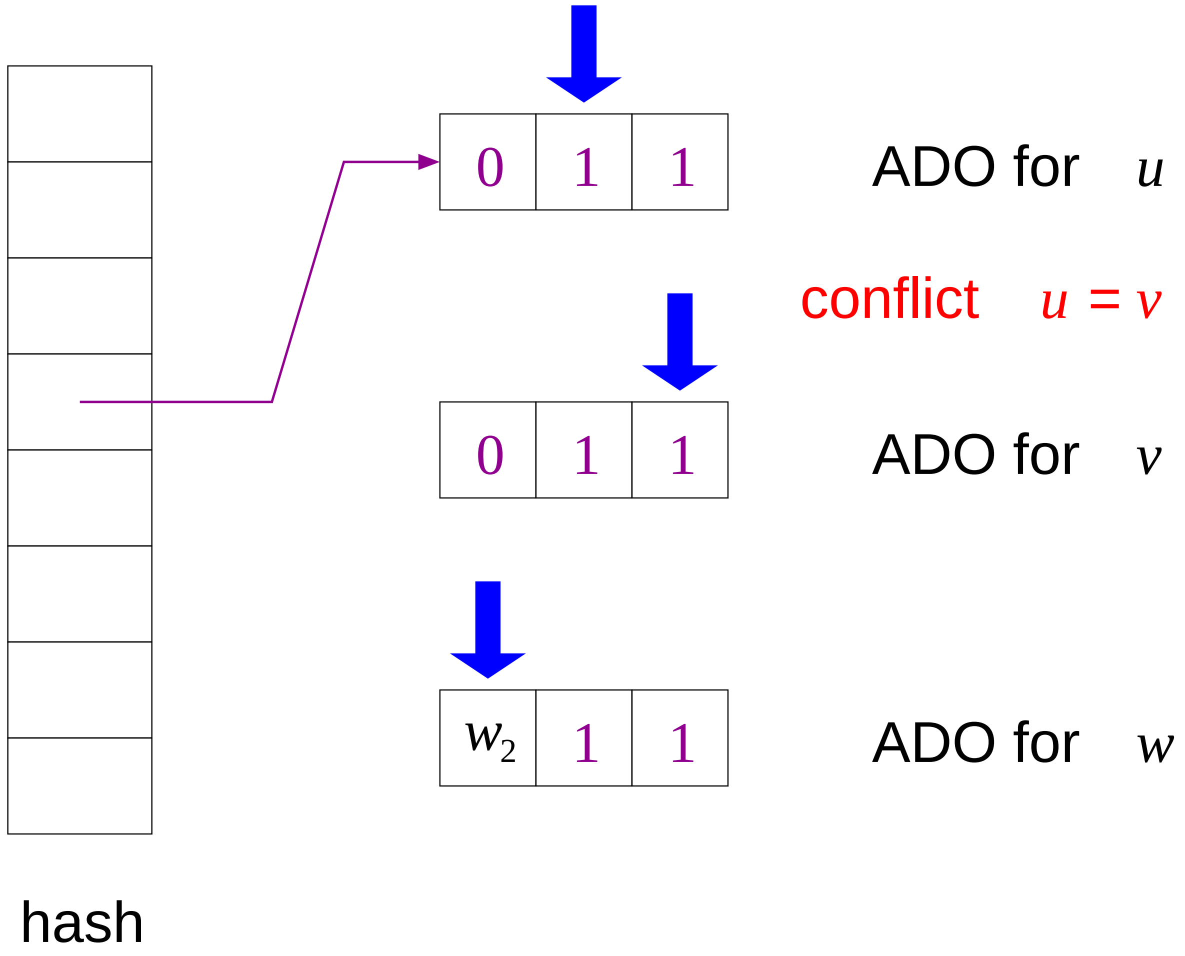
All Different Objects (ADOs)



All Different Objects (ADOs)



All Different Objects (ADOs)



Implementation Details

- ADO key is calculated from concrete bit-vector
  - by for instance XOR'ing bits word by word

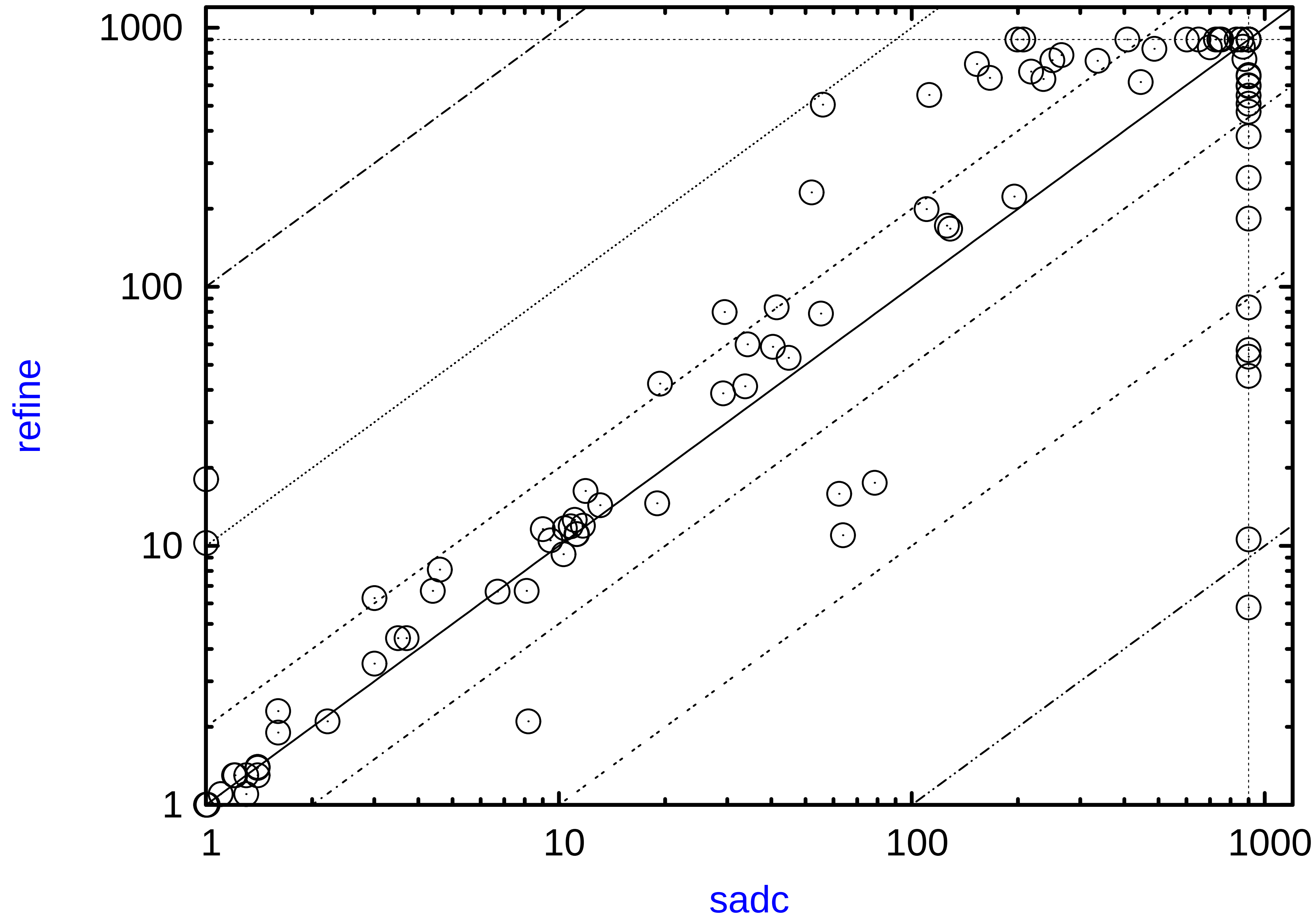
- ADOs watched by variables (not literals)
  - during backtracking all inserted ADOs need to be removed from hash table
  - save whether variable assignment forced ADO to be inserted
  - stack like insert/remove operations on hash table allow open addressing

- conflict analysis
  - all bits of the bit-vectors in conflict are followed
  - can be implemented by temporarily generating a pseudo clause

$$(u_2 \vee \overline{u}_1 \vee \overline{u}_0 \vee v_2 \vee \overline{v}_1 \vee \overline{v}_0)$$

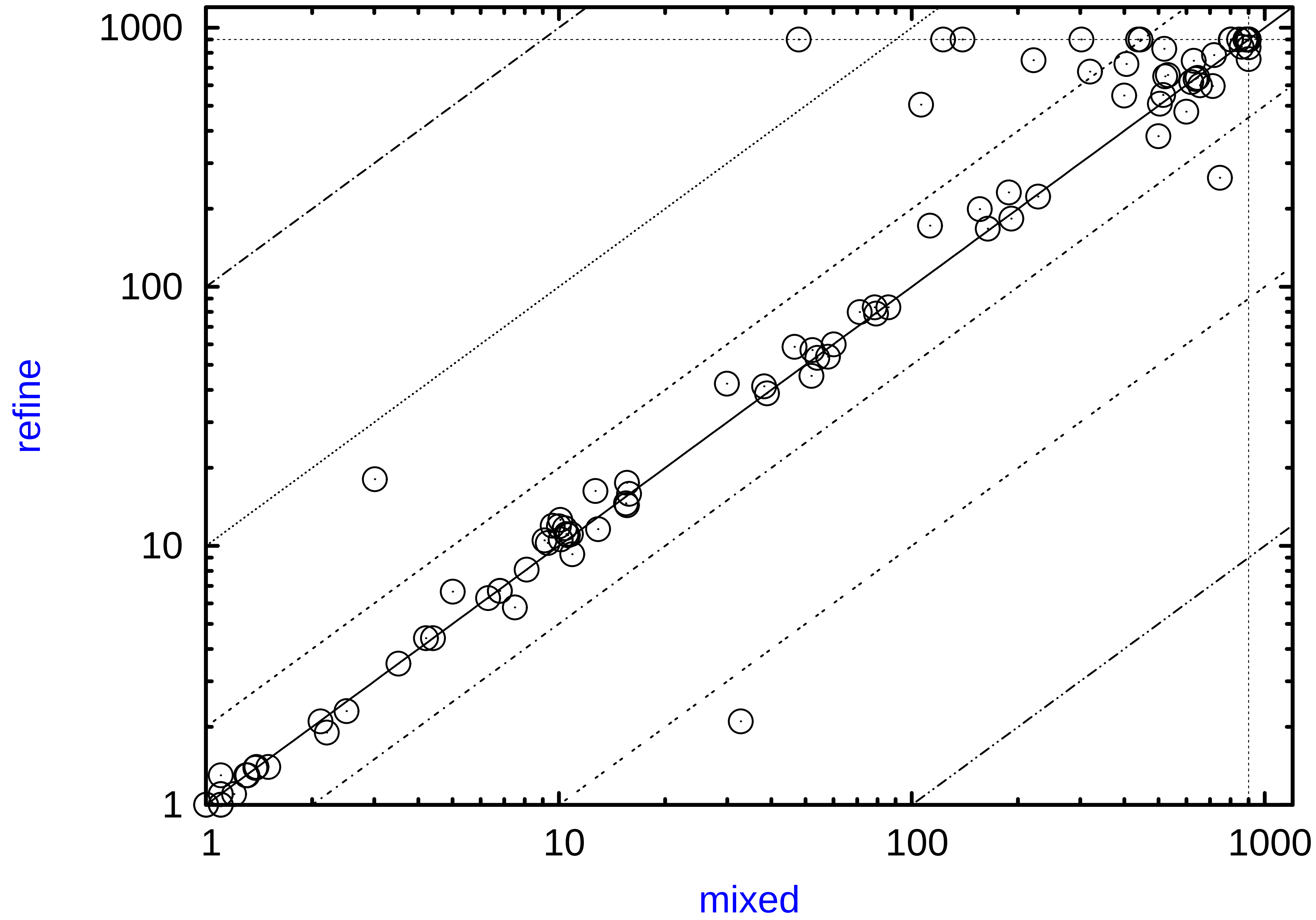
Symbolic ADCs versus Refine

[BiereBrummayer-FMCAD'08]



Mixed Approach versus Refine Only

[BiereBrummayer-FMCAD'08]



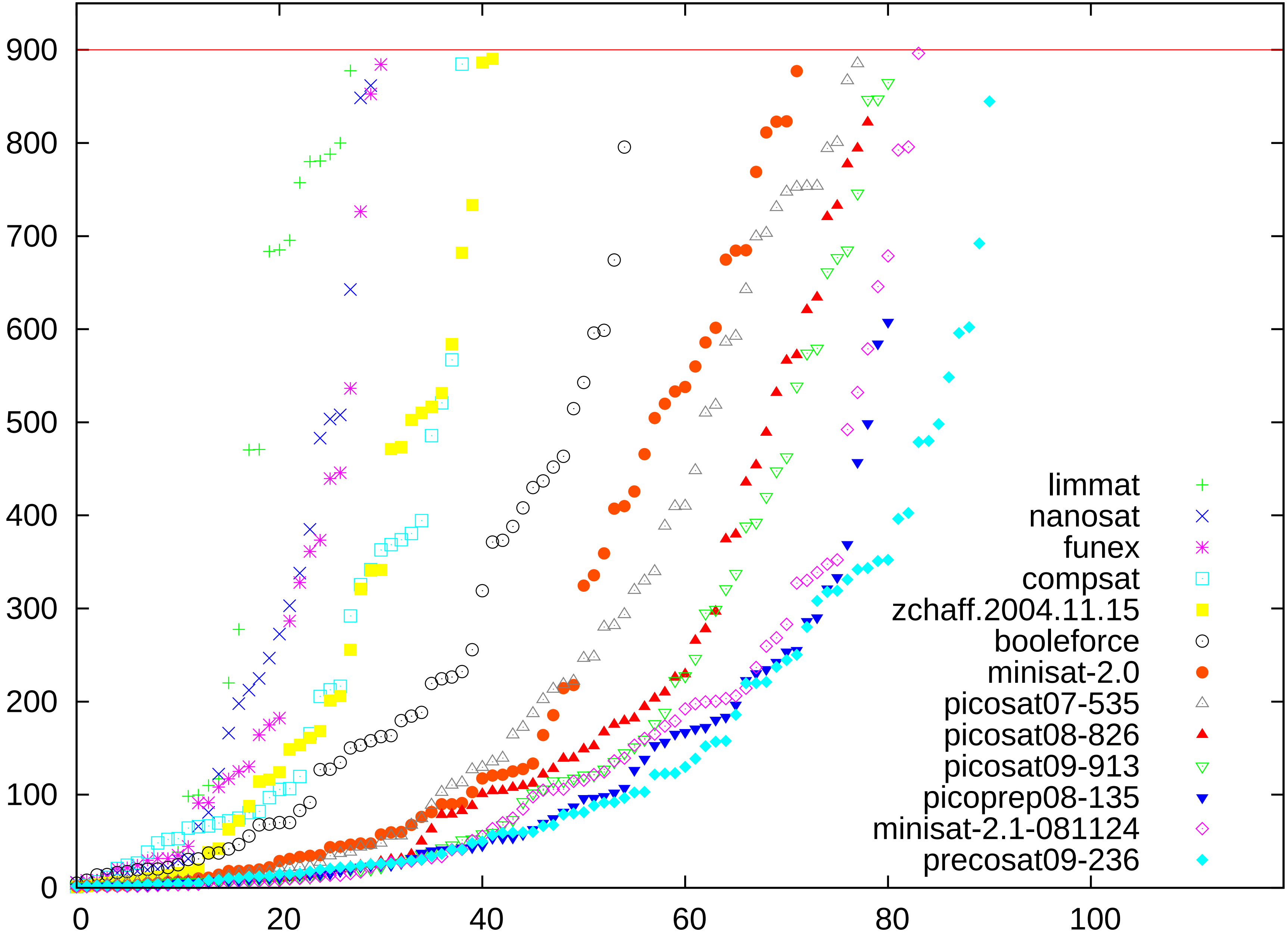
Conclusion on Symbolic All-Different Constraints

[BiereBrummayer-FMCAD'08]

- symbolic consistency checker for ADCs over bit-vectors
  - successfully applied to simple path constraints in model checking
  - similar to theory consistency checking in lazy SMT solvers
  - combination with eager refinement approach   lemmas on demand

- future work: ADC based BCP for bit-vectors
  - aka theory propagation in lazy SMT solvers
  - extensions to handle Ackermann constraints or even McCarthy axioms
  - one-way to get away from pure bit-blasting in BV

SAT Solver Progress (SAT'06 Race Instances)



Summary

- bounded model checking
  - routinely used in HW industry for falsification
  - need to improve word-level techniques for SW and HW verification / falsification
- SAT (and SMT) has seen tremendous improvements in recent years
  - was key enabler to make bounded model checking successful
  - many applications through the whole field of computer science
- still lots of opportunities:
  - parallel Model Checking / parallel SMT and SAT solving
  - portfolio and preprocessing (PrecoSAT was our first attempt)
  - make quantified boolean formula (QBF) reasoning work (QBF is PSPACE compl.)