Consistency Checking of All Different Constraints over Bit-Vectors within a SAT Solver

Armin Biere, Robert Brummayer

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

FMCAD'08

Formal Methods in Computer Aided Design

Portland, Oregon, USA

Thursday, November 20, 2008

- bounded model checking: [BiereCimattiClarkeZhu'99]

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

- reoccurrence diameter checking: [BiereCimattiClarkeZhu'99]

$$T(s_1, s_2) \wedge \ldots \wedge T(s_{k-1}, s_k) \;\wedge \bigwedge_{1 \leq i < j \leq k} s_i \neq s_j$$ unsatisfiable?

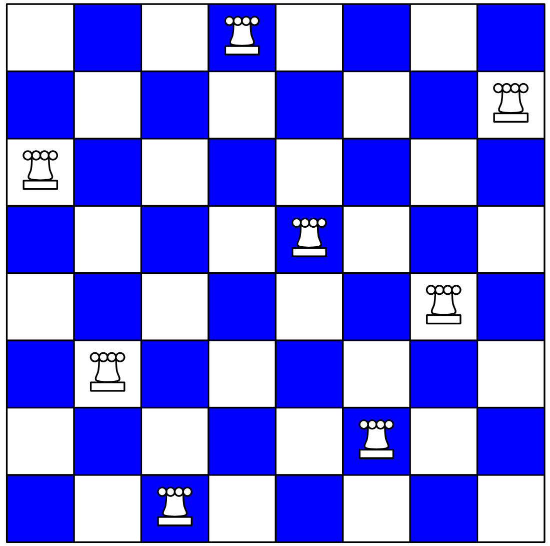
- k-induction base case: [SheeranSinghStålmarck'00]

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

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

$$T(s_1, s_2) \wedge \ldots \wedge T(s_{k-1}, s_k) \;\wedge B(s_k) \wedge \bigwedge_{0 \leq i < k} \neg B(s_i) \;\wedge \bigwedge_{1 \leq i < j \leq k} s_i \neq s_j$$ unsatisfiable?

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

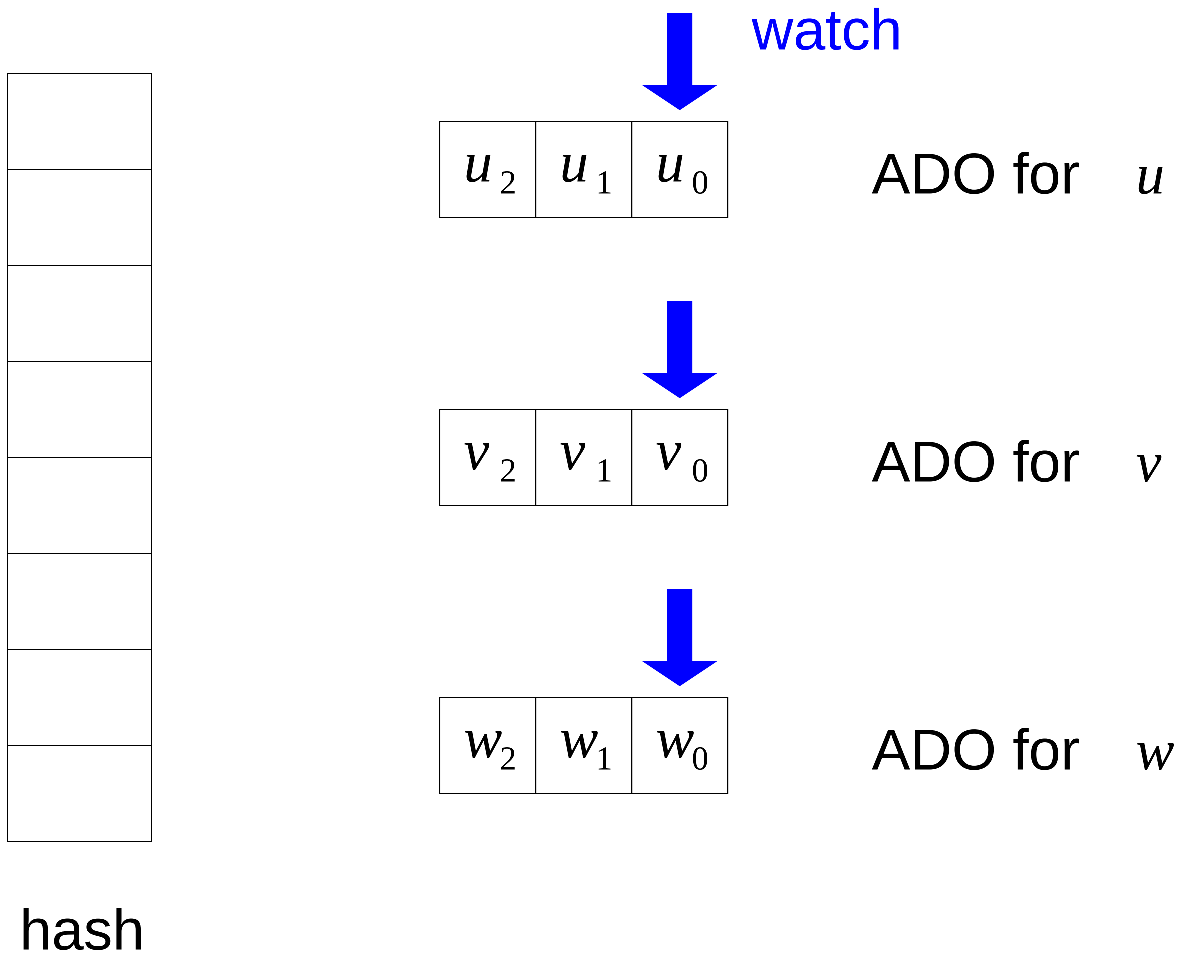


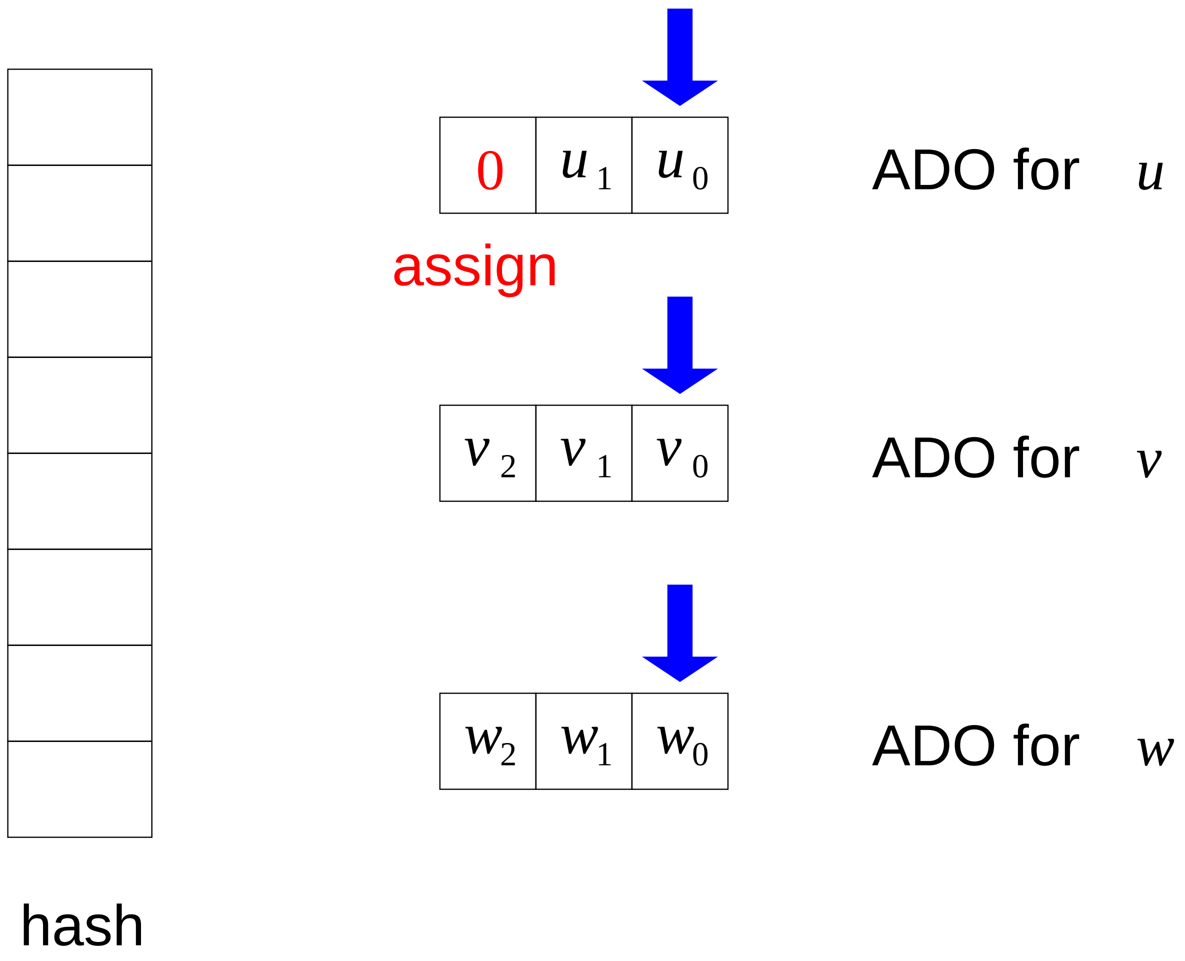
- propagation algorithms to establish arc-consistency
  - explicit propagators: [Régin'94]
    - $\mathrm{O}(k \cdot m)$ space
    - $\mathrm{O}(k^2 \cdot m^2)$ time
  - symbolic propagators: [GentNightingale'04] also [MarquesSilvaLynce'07]
    - one-hot CNF encoding with $\Omega(k \cdot m)$ boolean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

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

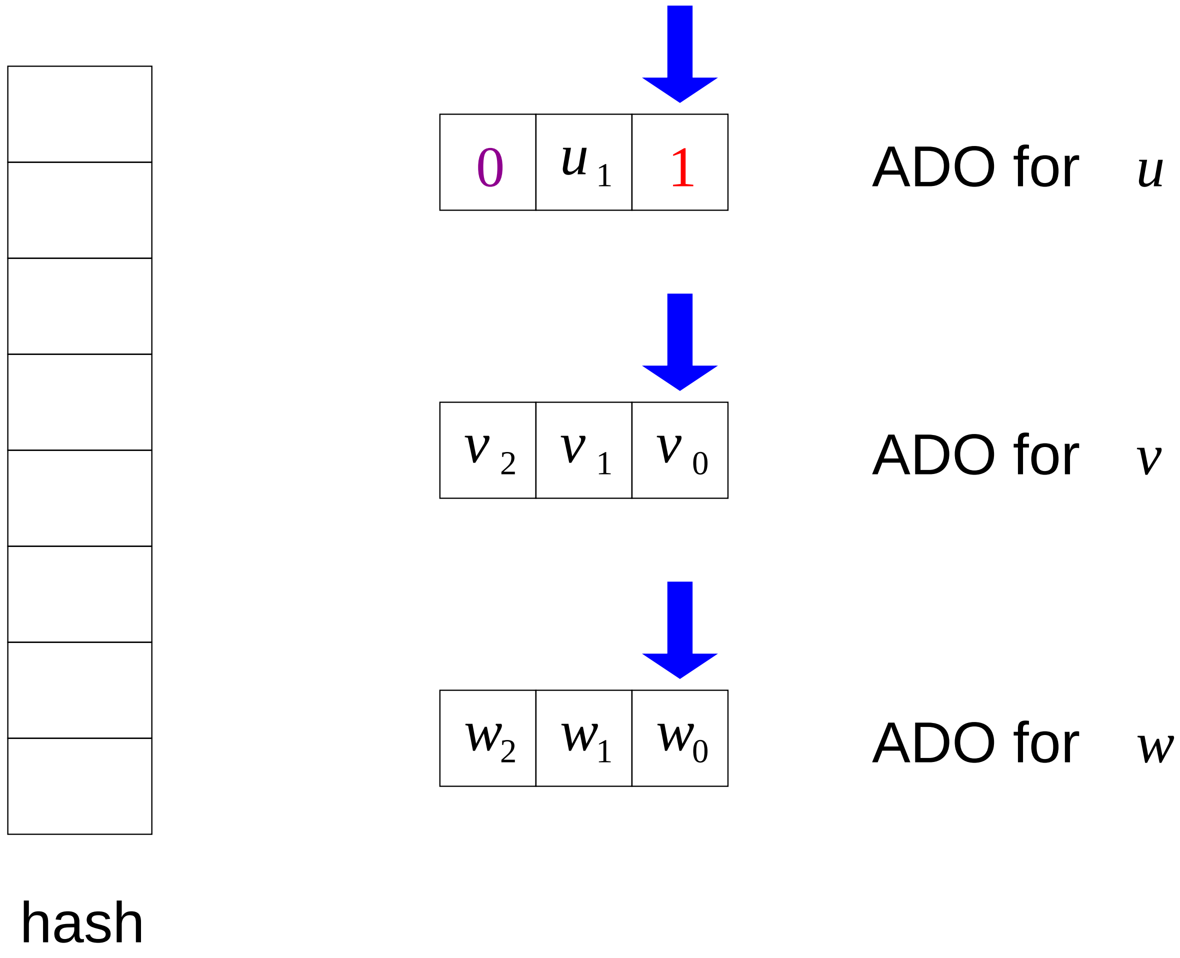
    $u \neq v$ is equisatisfiable to $(d_0 \vee \cdots \vee d_{n-1}) \wedge \bigwedge_{j=0}^{n-1} (u_j \vee v_j \vee \overline{d_j}) \wedge (\overline{u_j} \vee \overline{v_j} \vee \overline{d_j})$
  - can be extended to encode Ackermann Constraints + McCarthy Axioms
  - either **eagerly** encode all $s_i \neq s_j$ quadratic in k
  - or **refine** adding bit-vector inequalities on demand [EénSörensson'03]
- natively handle ADCs within SAT solver: our main contribution
  - 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]

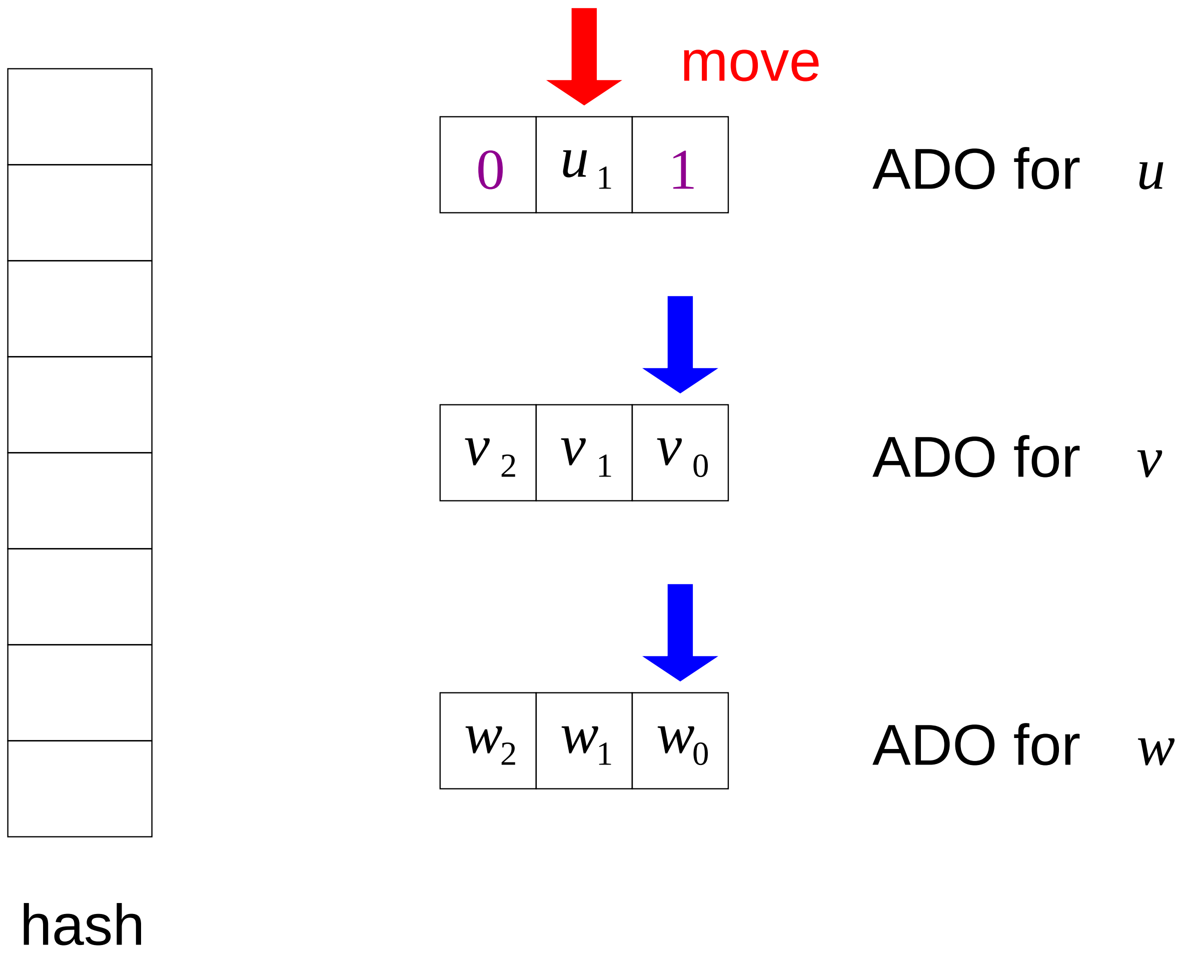
All Different Objects (ADOs)

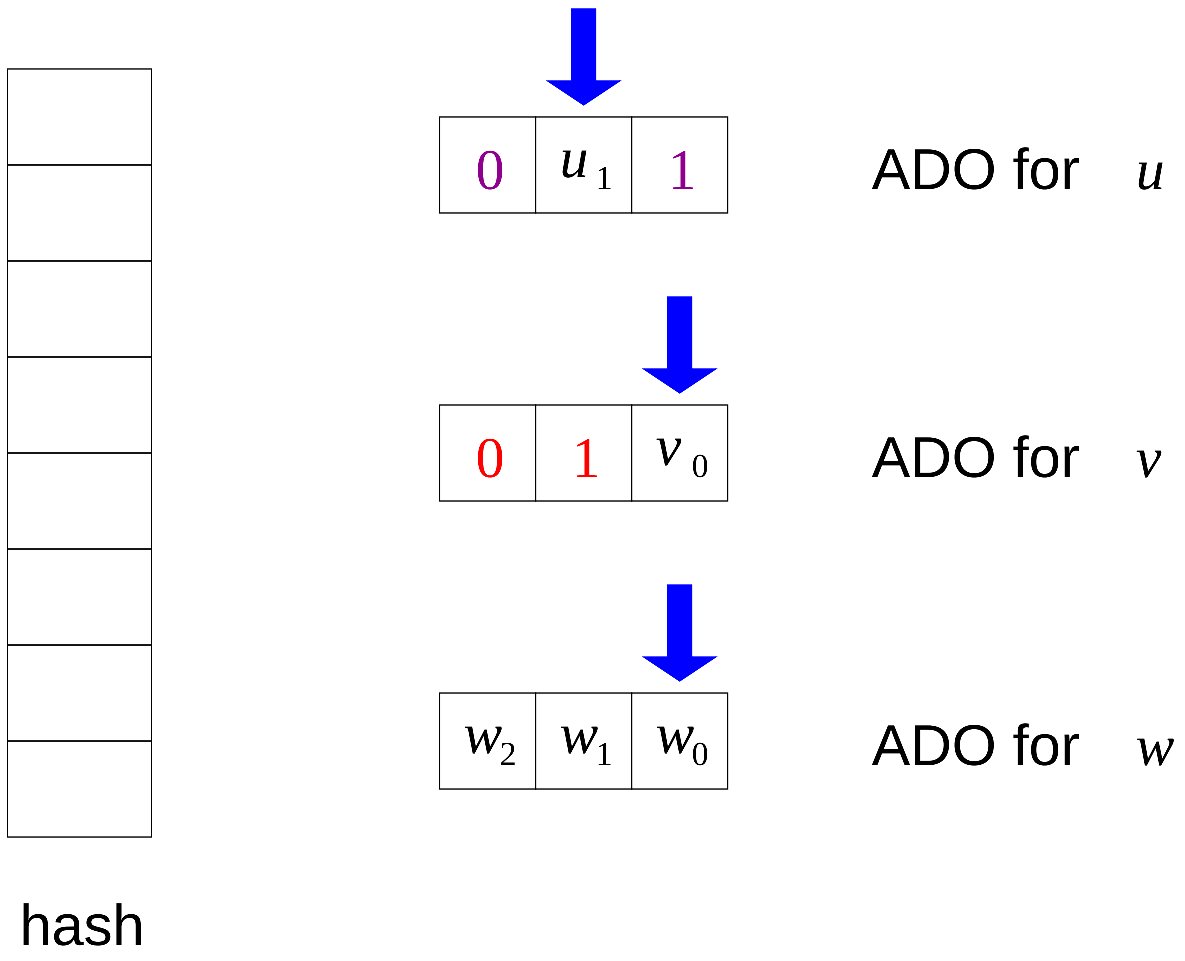


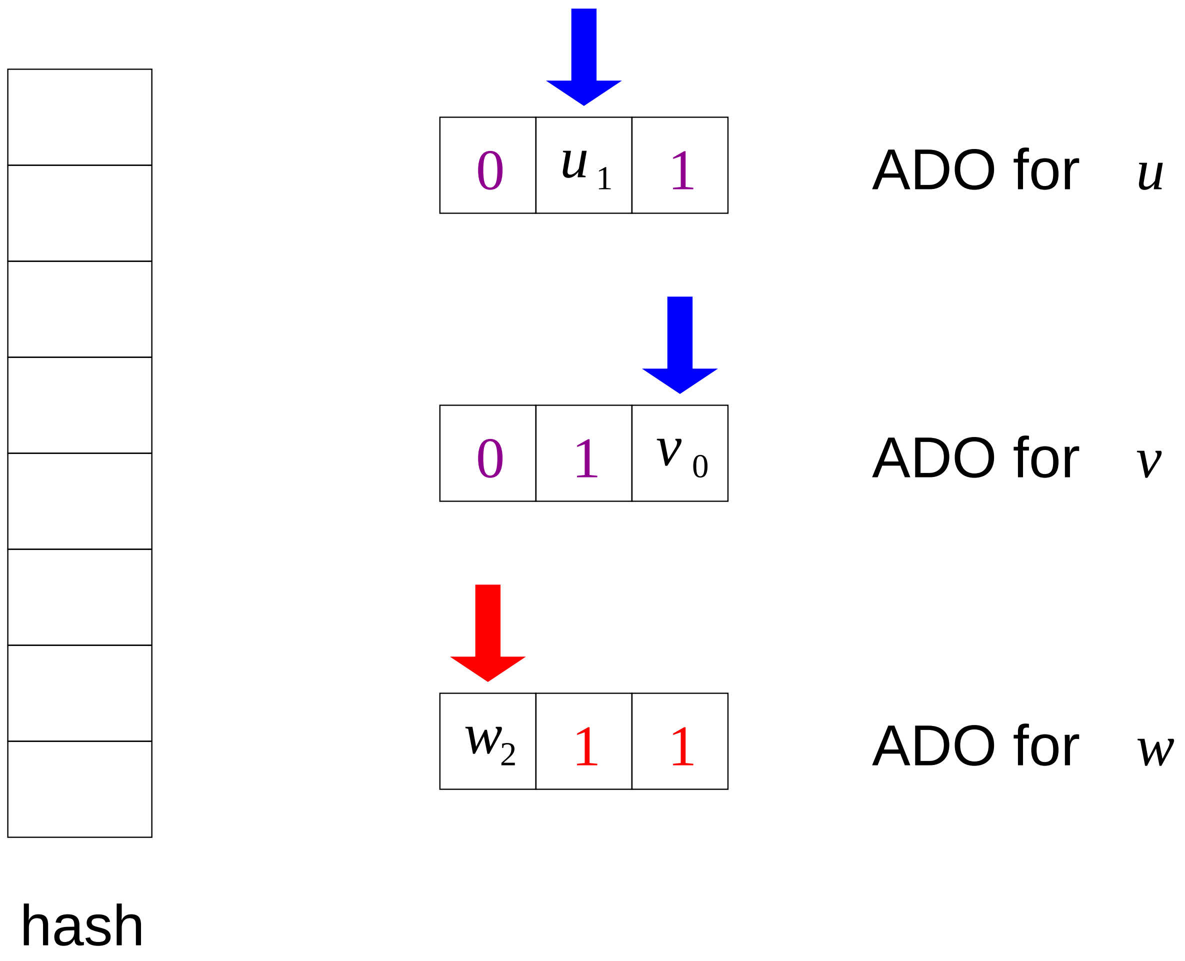

0
u_1
u_0
ADO for u
assign
v_2
v_1
v_0
ADO for v
w_2
w_1
w_0
ADO for w
hash

All Different Objects (ADOs)

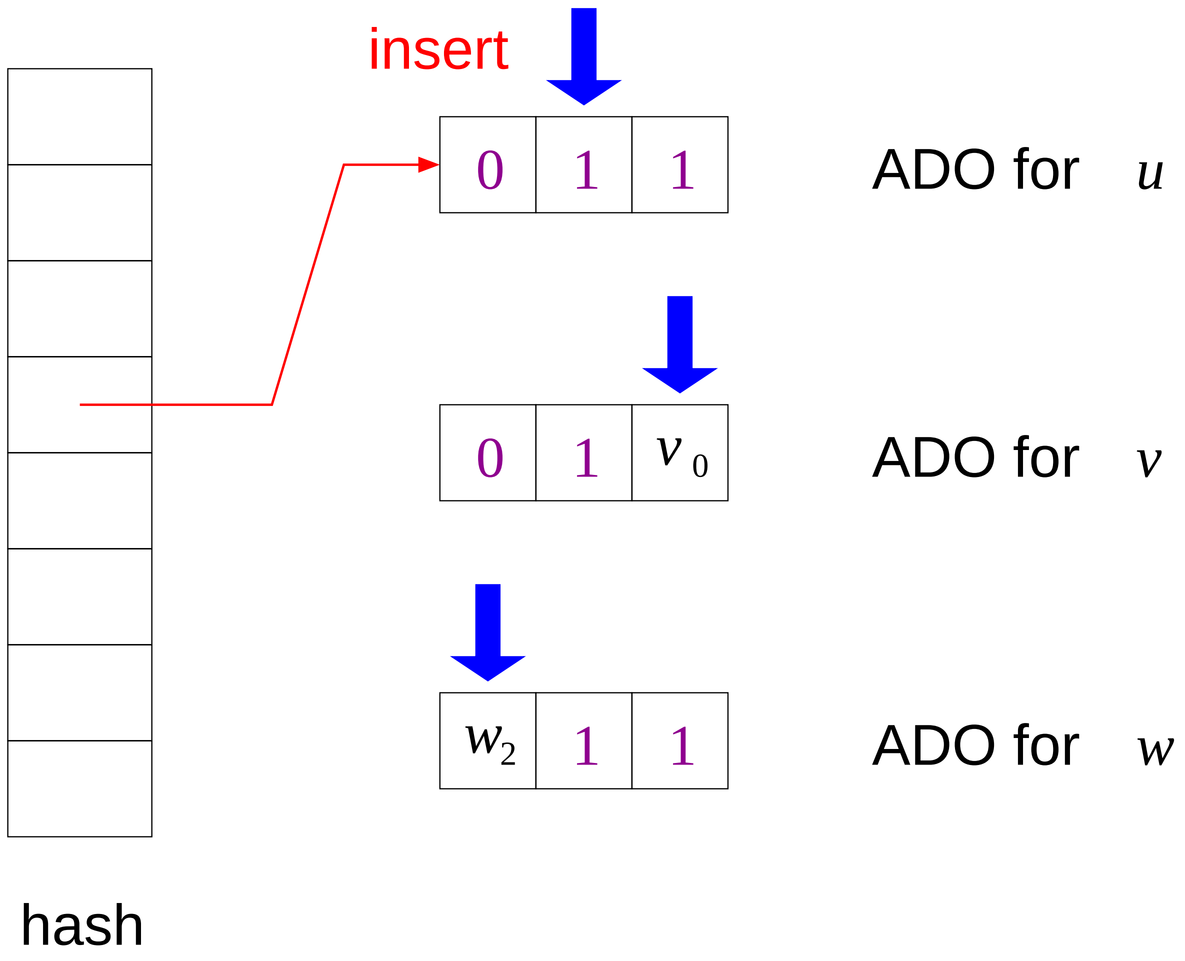



move
0
u_1
1
ADO for u
v_2
v_1
v_0
ADO for v
w_2
w_1
w_0
ADO for w
hash

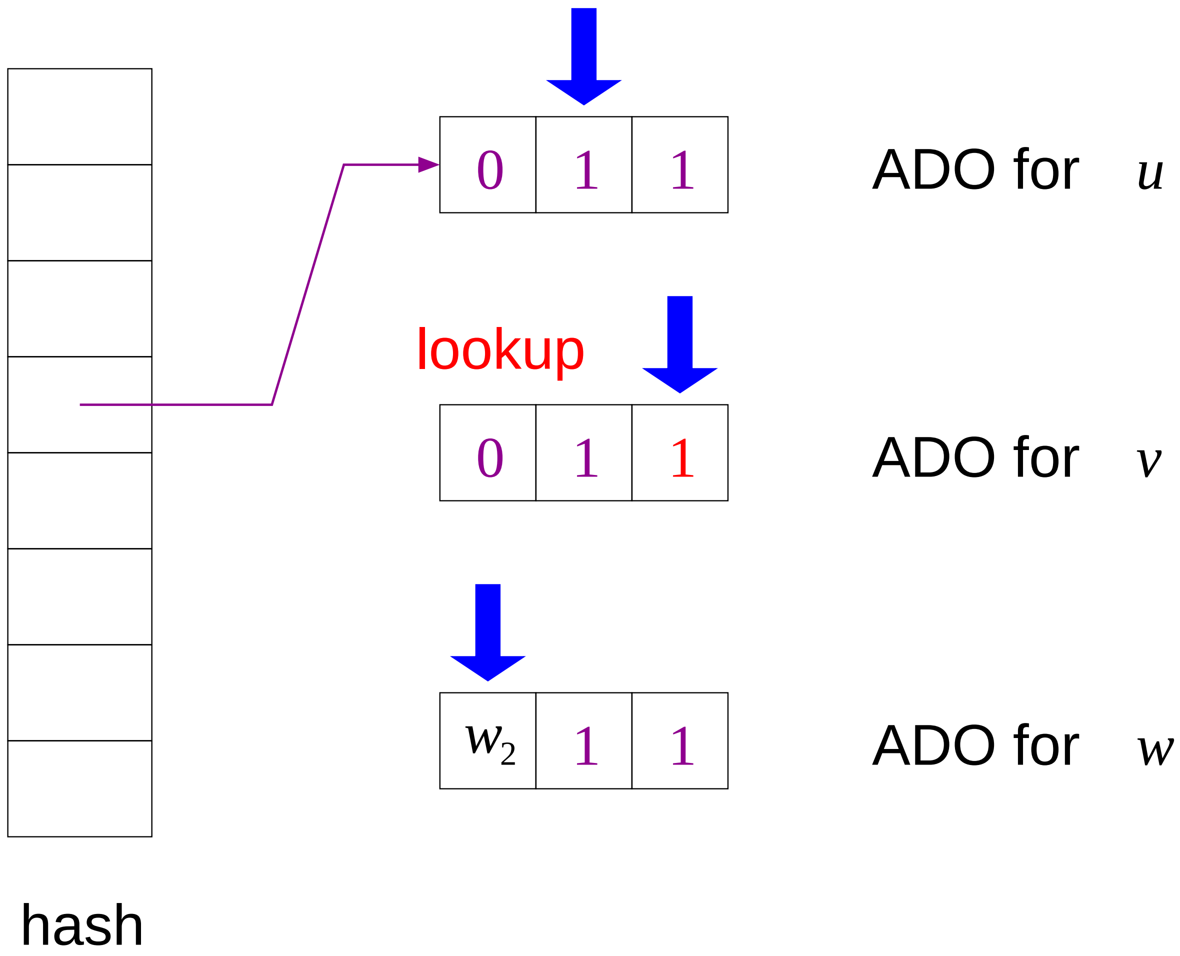

0
u_1
1
ADO for u
0
1
v_0
ADO for v
w_2
w_1
w_0
ADO for w
hash

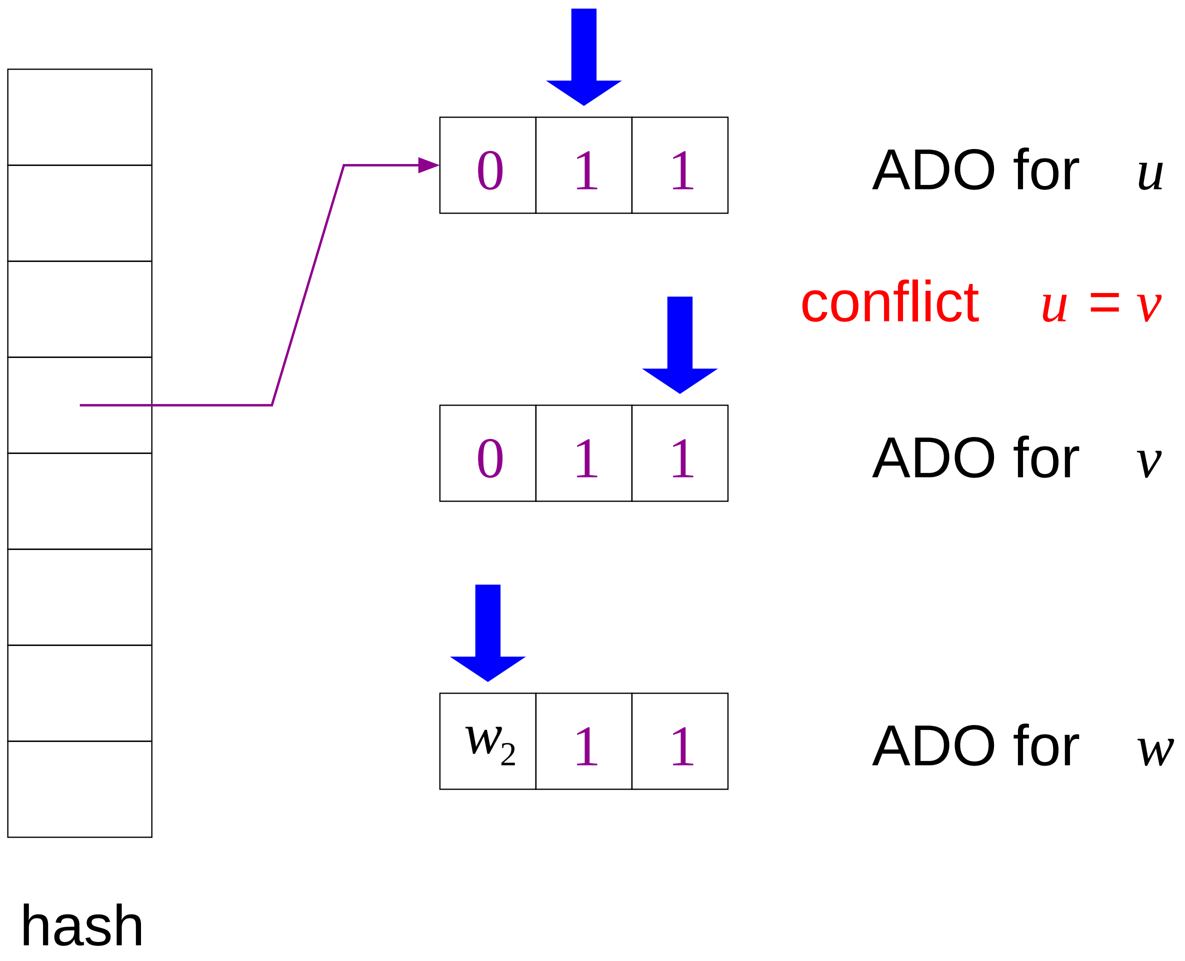

0
u_1
1
ADO for u
0
1
v_0
ADO for v
w_2
1
1
ADO for w
hash

complete
0 1 1
ADO for u
0 1 v_0
ADO for v
w_2 1 1
ADO for w
hash


insert
0
1
1
ADO for u
0
1
v_0
ADO for v
w_2
1
1
ADO for w
hash

All Different Objects (ADOs)




0 1 1
ADO for u
conflict $u = v$
0 1 1
ADO for v
w_2 1 1
ADO for w
hash

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

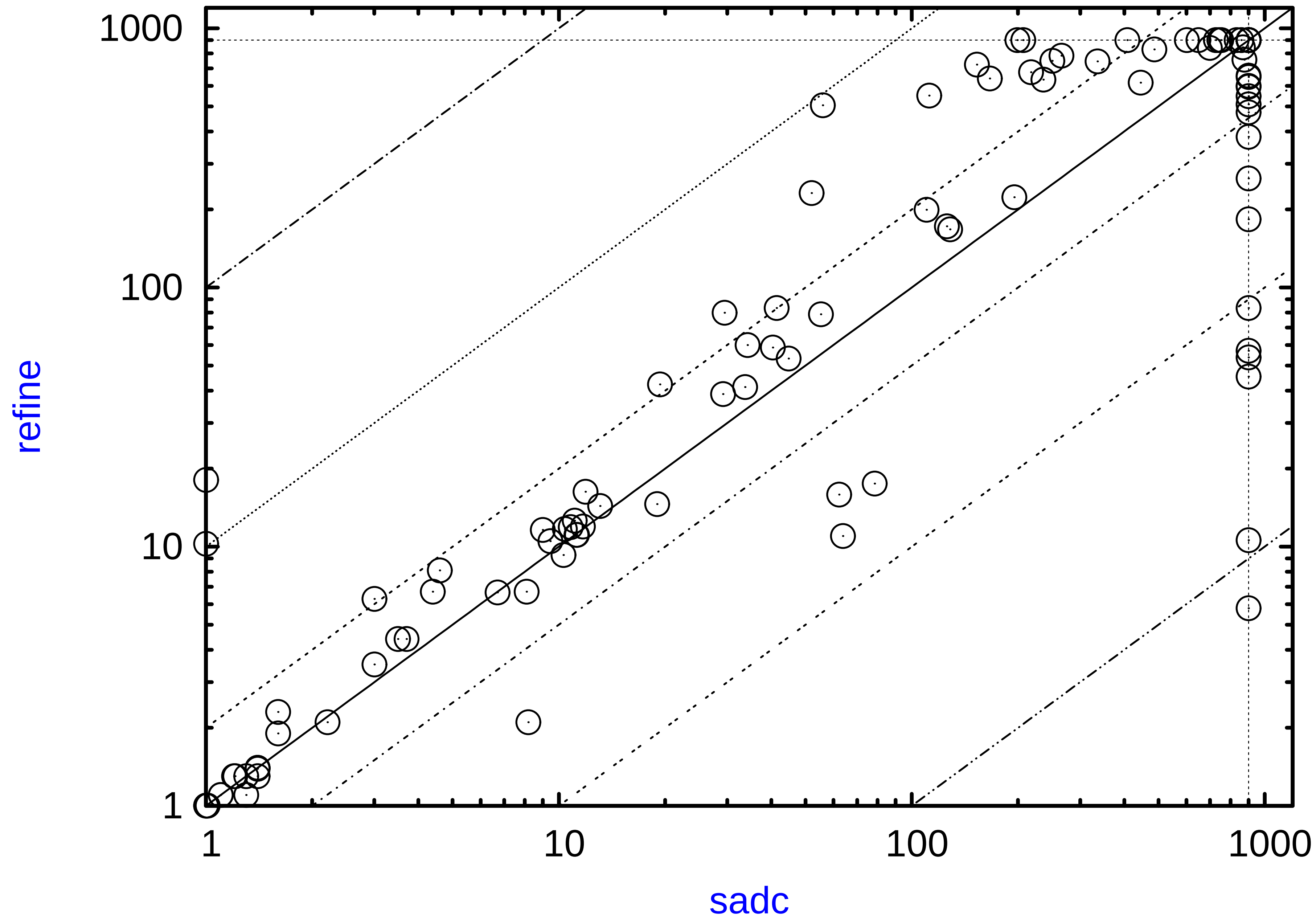
- conflict analysis
  - all bits of the bit-vectors in conflict are followed
  - can be implemented by temporarily generating a pseudo clause

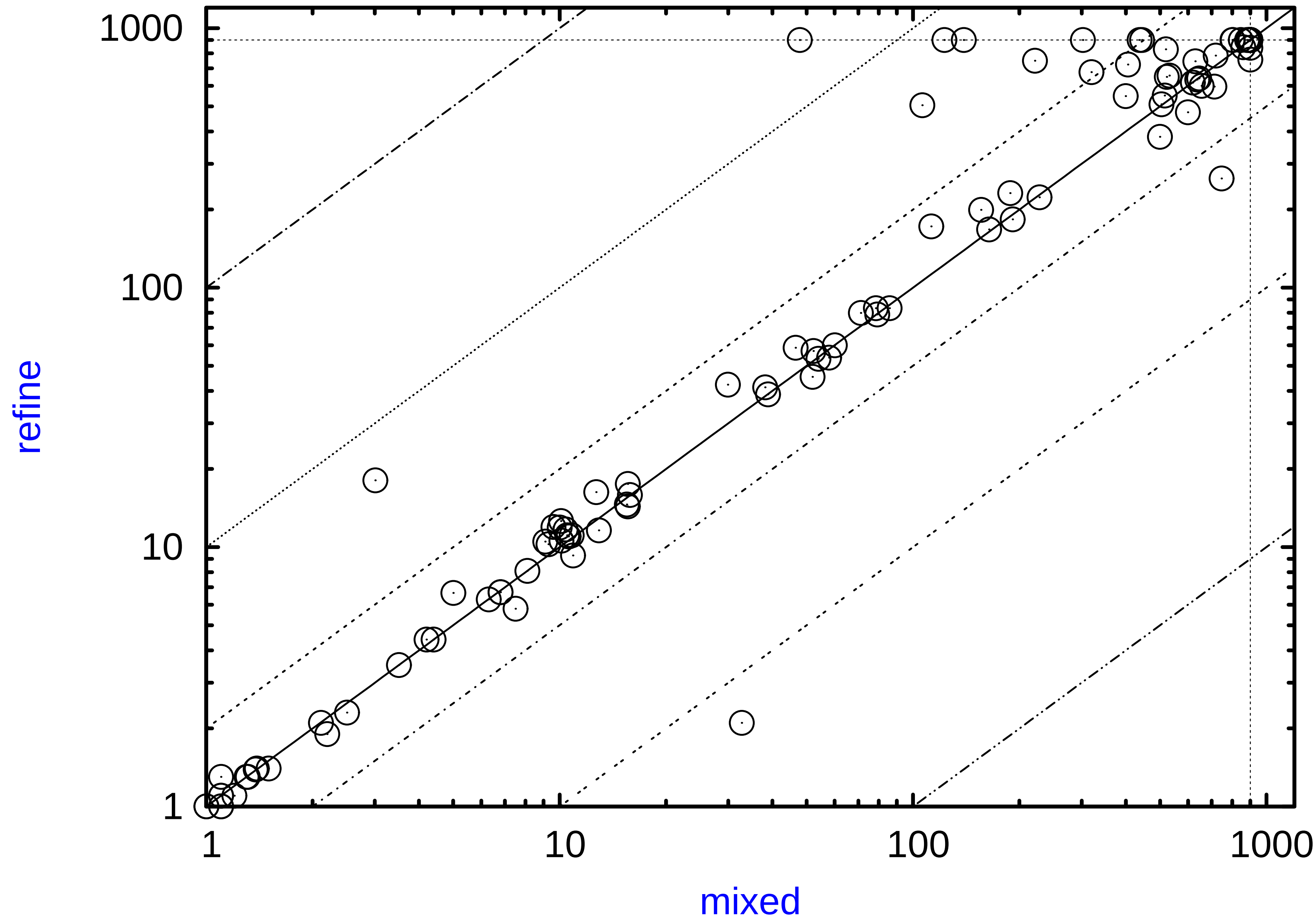
$$(u_2 \vee \overline{u}_1 \vee \overline{u}_0 \vee v_2 \vee \overline{v}_1 \vee \overline{v}_0)$$

		solved		*inconclusive*		*unsatisfiable*	*time*	*space*	*steps*
	complete		*unsolved*		*satisfiable*		10^3sec	GB	
mixed	y	259	85	38	182	39	96	23.2	9736
refine	y	250	94	32	179	39	101	23.0	9698
sadc	y	244	100	36	171	37	103	17.2	9131
eager	y	242	102	27	177	38	102	31.9	9438
none	n	267	77	56	179	32	87	16.6	10877
base	n	283	61	96	187	0	70	28.9	15187

only checked up to $k = 100$ (at most 100 steps per instance)

three possible outcomes: *inconclusive*, *satisfiable*, or *unsatisfiable*


refine
1000
100
10
1
1
10
100
1000
sadc


1000
100
10
1
1
10
100
1000
refine
mixed

- 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