Efficient Modular SAT Solving for IC3

Sam Bayless*, Celina G. Val*,
Thomas Ball†, Holger H. Hoos*, Alan J. Hu*

*University of British Columbia
†Microsoft Research

Efficient Modular SAT Solving for IC3

We introduce an improved version of IC3, 'SMS-PDR', using our new 'SAT modulo SAT' solver:

	SMS-PDR	PDR	IC3
HWMCC'08	**596**/645	581/645	586/645
HWMCC'10	**749**/818	733/818	712/818
HWMCC'12	**92**/310	84/310	48/310

Solves more instances than both ABC's PDR, and the original IC3, on the 2008, 2010, and 2012 Hardware Model Checking Competition Benchmarks.

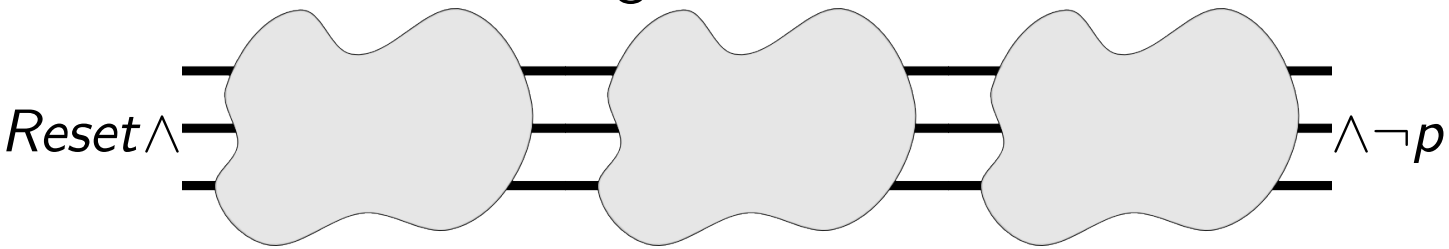
Efficient Modular SAT Solving for IC3

Outline:

1. Cube-blocking in IC3
2. Connection between cube-blocking and SMT
3. SAT modulo SAT solvers for cube-blocking
4. Results

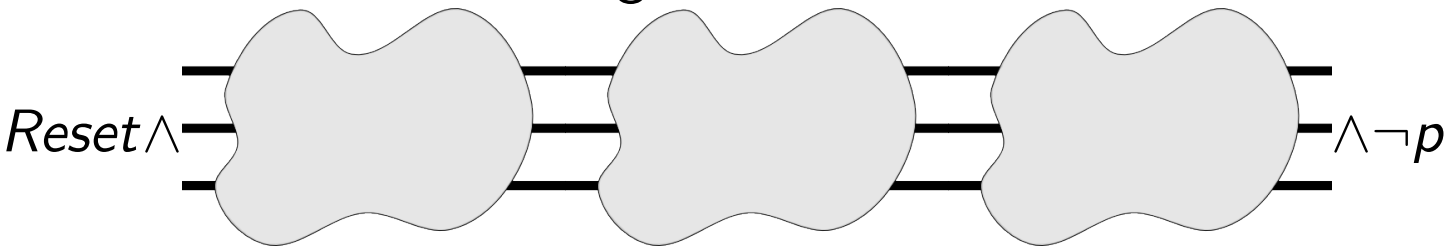
Cube Blocking in IC3

Bounded Model Checking:

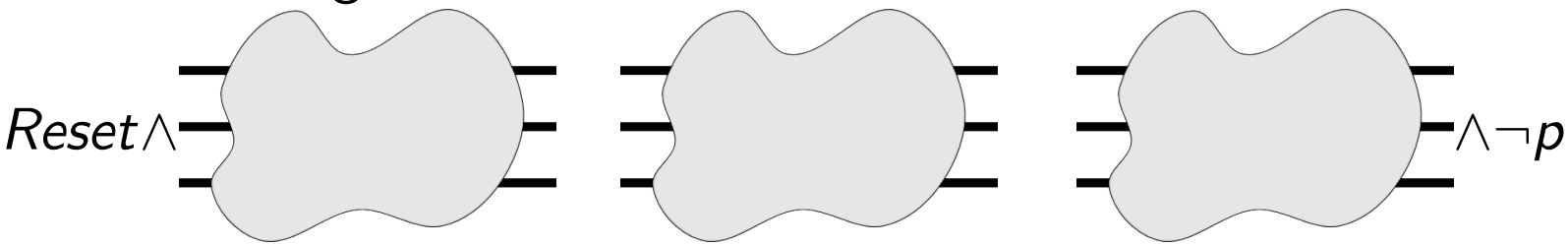


Cube Blocking in IC3

Bounded Model Checking:

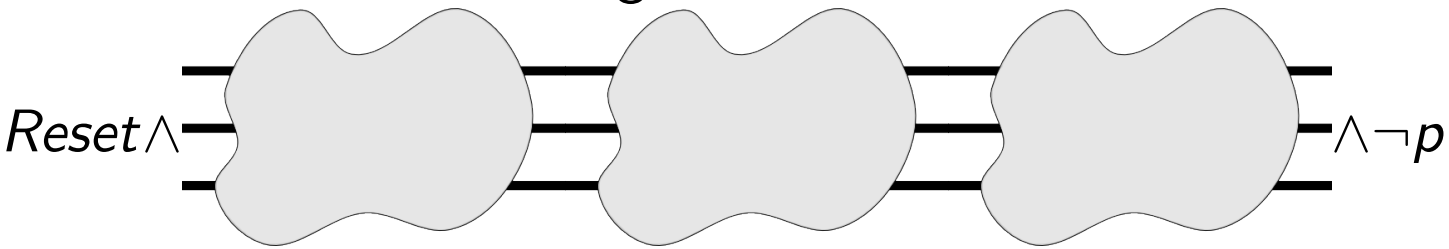


Cube Blocking in IC3:

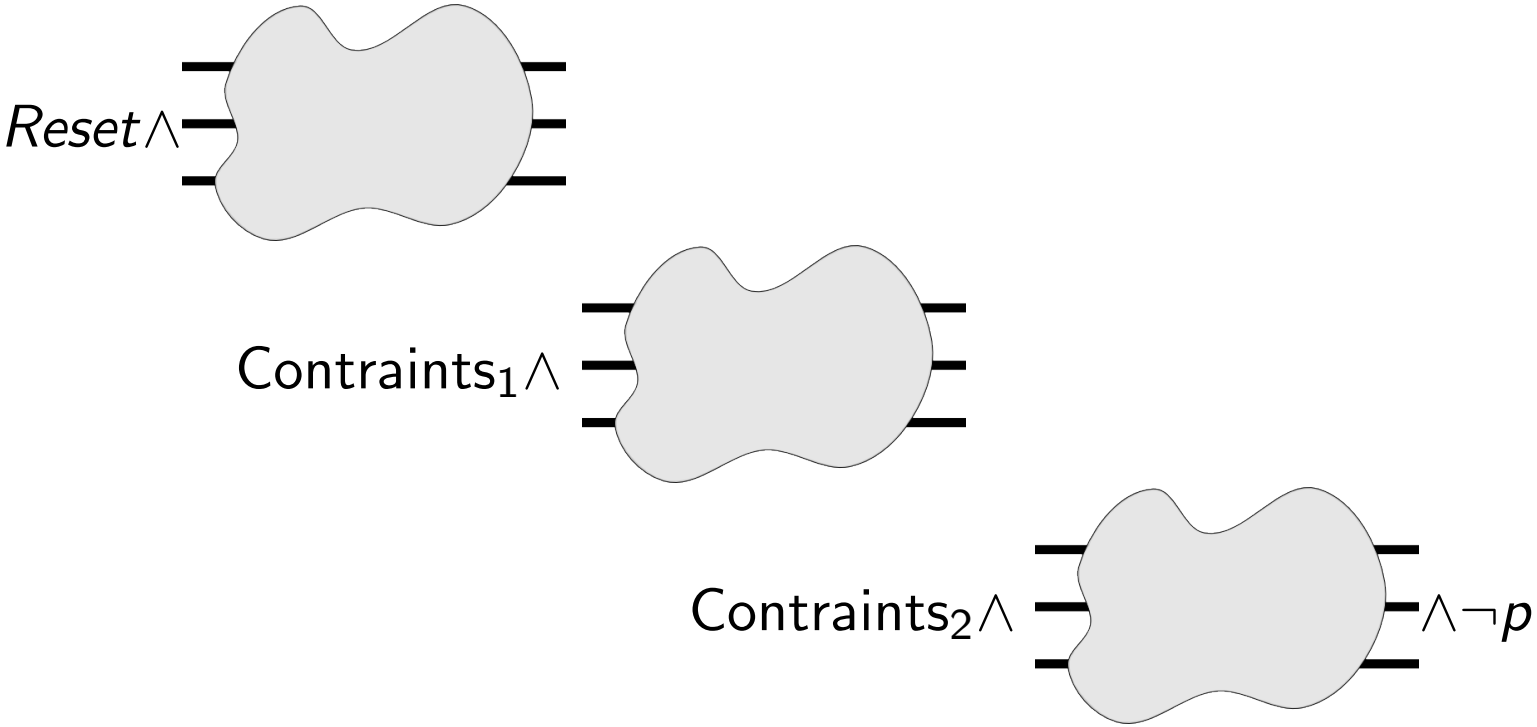


Cube Blocking in IC3

Bounded Model Checking:

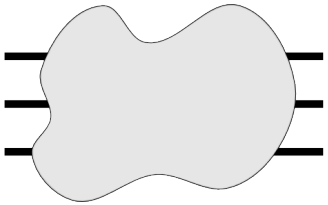


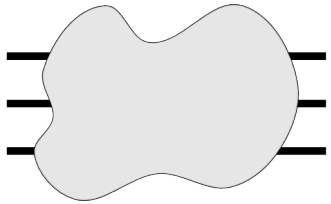
Cube Blocking in IC3:

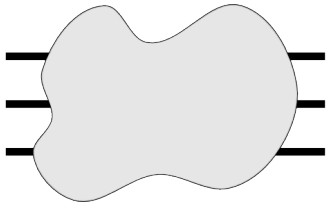


Cube Blocking in IC3

Cube Blocking in IC3:

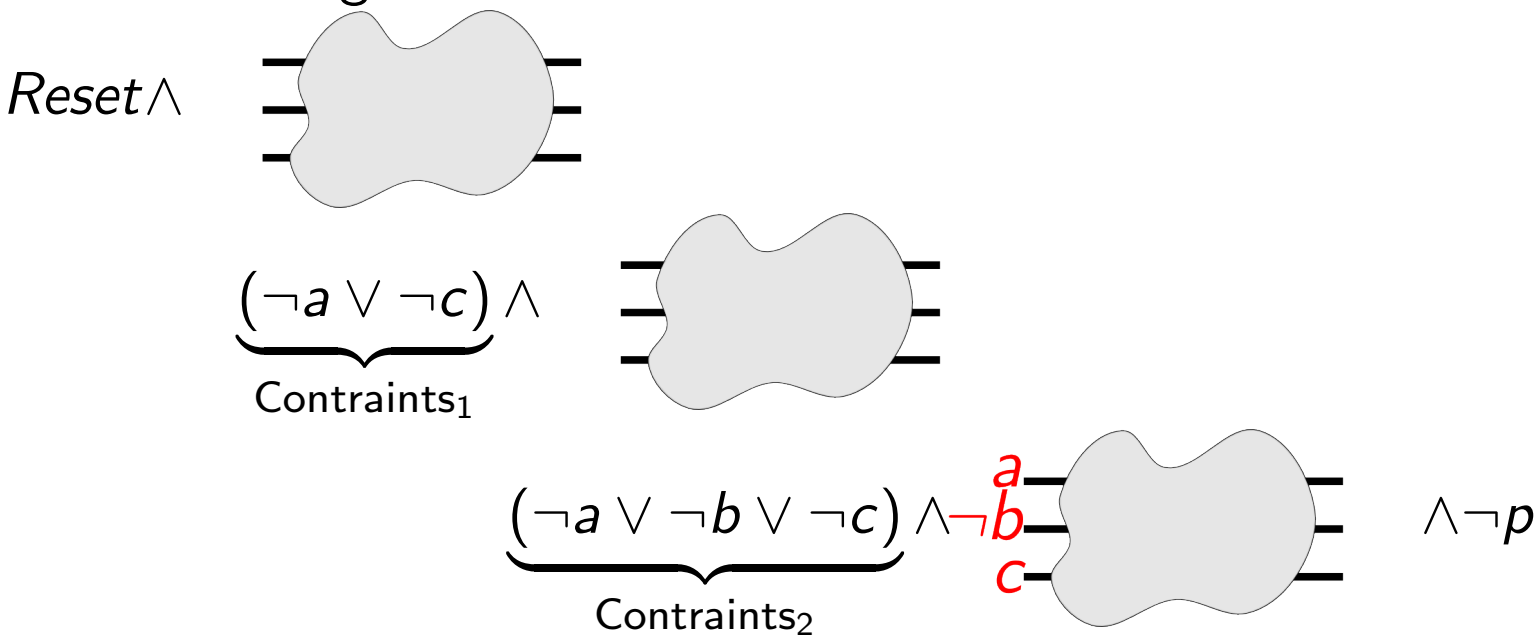
$Reset \wedge$ 

$\underbrace{(\neg a \vee \neg c)}_{\text{Contraints}_1} \wedge$ 

$\underbrace{(\neg a \vee \neg b \vee \neg c)}_{\text{Contraints}_2} \wedge$  $\wedge \neg p$

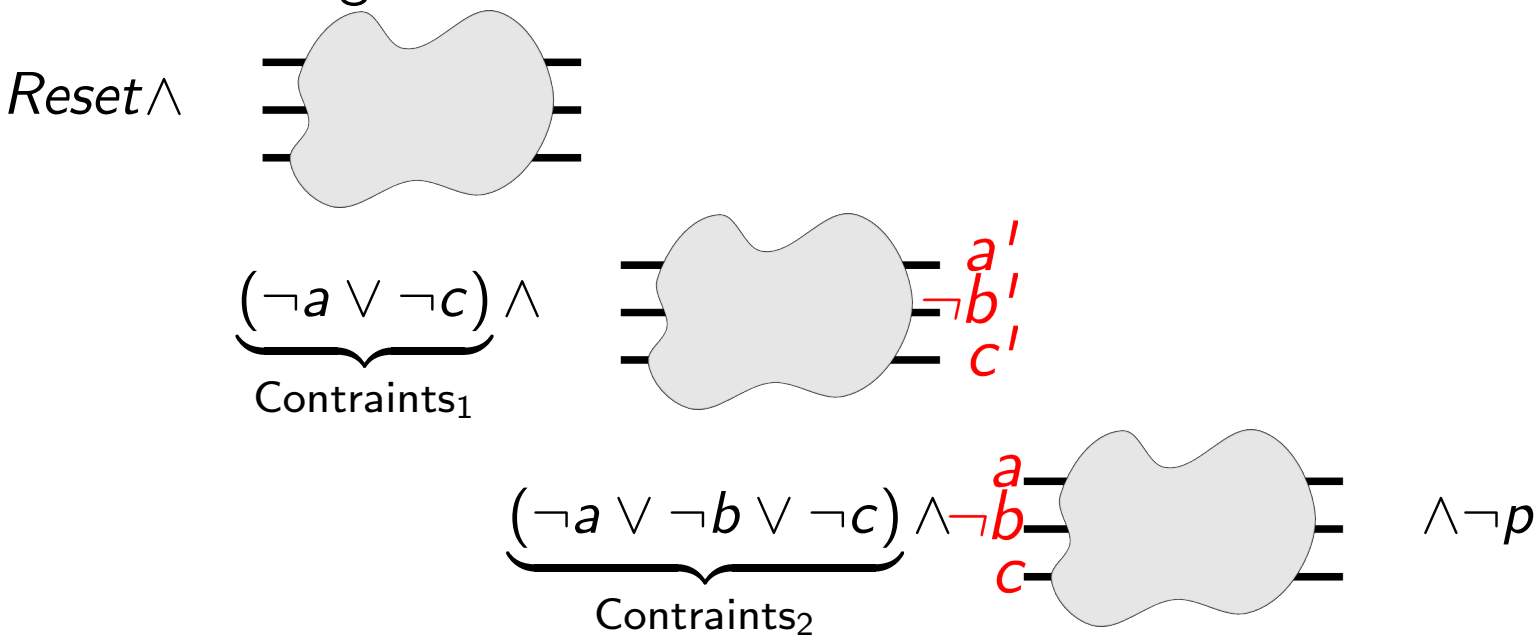
Cube Blocking in IC3

Cube Blocking in IC3:



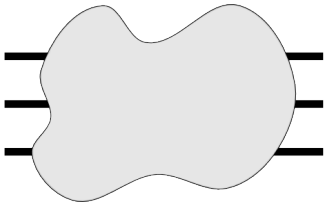
Cube Blocking in IC3

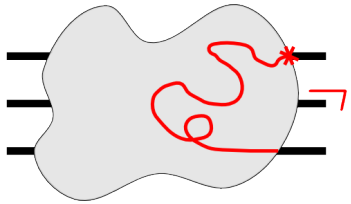
Cube Blocking in IC3:



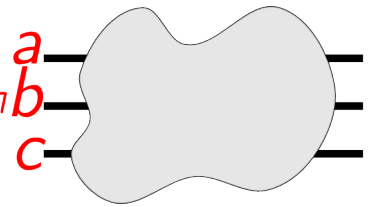
Cube Blocking in IC3

Cube Blocking in IC3:

$Reset \wedge$ 

$\underbrace{(\neg a \vee \neg c)}_{\text{Contraints}_1} \wedge$ 

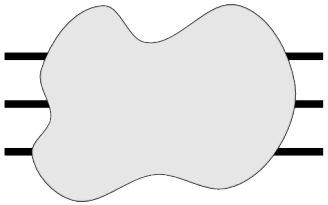
$\rightarrow (\neg a' \vee b')$

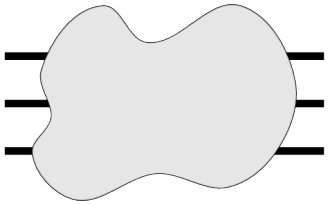
$\underbrace{(\neg a \vee \neg b \vee \neg c)}_{\text{Contraints}_2} \wedge \neg$ 

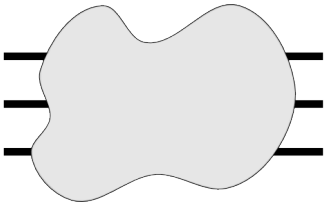
$\wedge \neg p$

Cube Blocking in IC3

Cube Blocking in IC3:

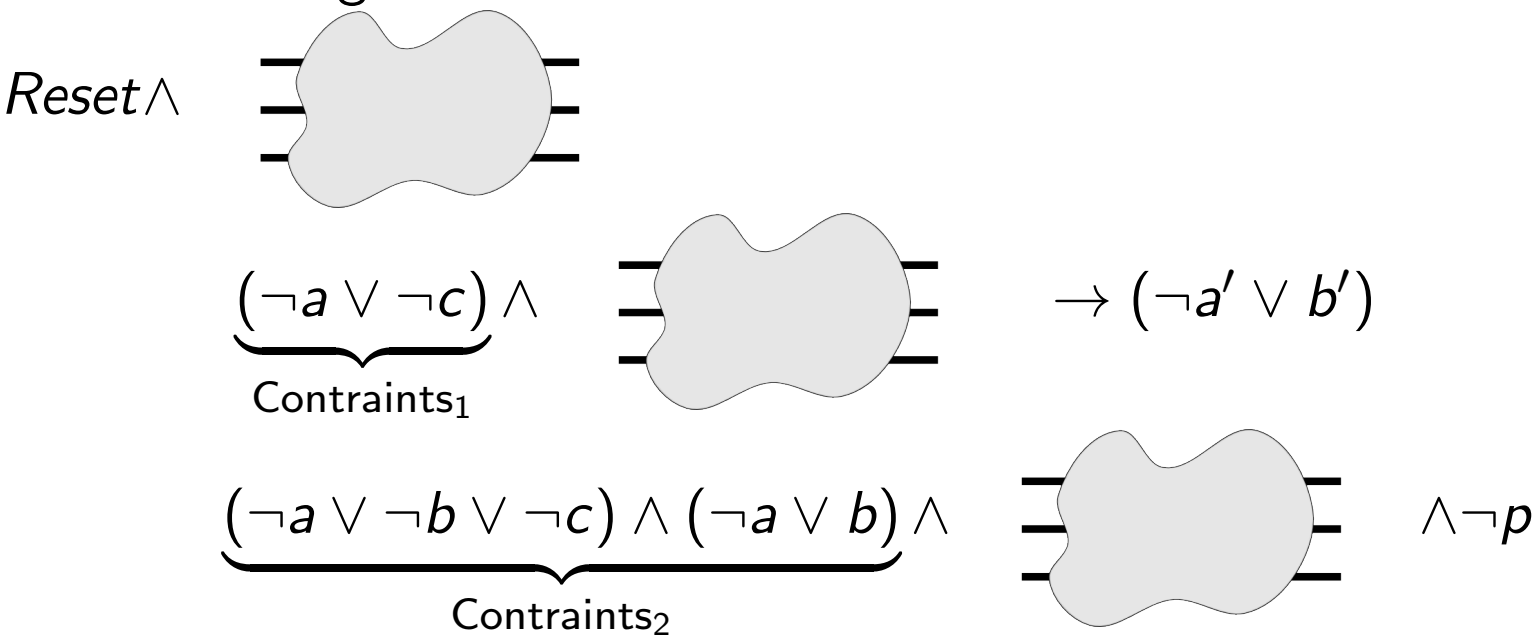
$Reset \wedge$ 

$\underbrace{(\neg a \vee \neg c)}_{\text{Contraints}_1} \wedge$  $\rightarrow (\neg a' \vee b')$

$\underbrace{(\neg a \vee \neg b \vee \neg c) \wedge (\neg a \vee b)}_{\text{Contraints}_2} \wedge$  $\wedge \neg p$

Cube Blocking in IC3

Cube Blocking in IC3:



Problem: Because we are solving each time frame independently, satisfying assignments to the inputs of one time frame might be trivially false (*i.e.*, refuted by unit propagation alone) when passed to the outputs of the preceding time frame.

SAT modulo Theory Solvers

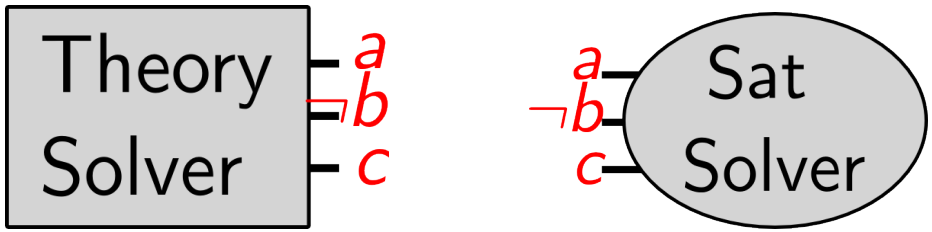
Solve formulas containing both propositional logic and statements in some other language (like linear arithmetic, real numbers, bit-vectors, . . .).

SAT modulo Theory Solvers

Solve formulas containing both propositional logic and statements in some other language (like linear arithmetic, real numbers, bit-vectors, . . .).

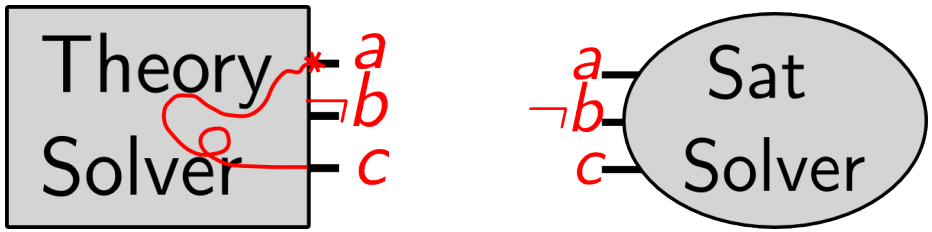
SAT modulo Theory Solvers

Solve formulas containing both propositional logic and statements in some other language (like linear arithmetic, real numbers, bit-vectors, . . .).



SAT modulo Theory Solvers

Solve formulas containing both propositional logic and statements in some other language (like linear arithmetic, real numbers, bit-vectors, . . .).



SAT modulo Theory Solvers

Efficient, lazy SMT[1] solvers apply unit propagation from the SAT solver to the theory solver *eagerly*

[1] R. Sebastiani. "Lazy satisfiability modulo theories". In: *Journal on Satisfiability, Boolean Modeling and Computation (JSAT)* 3 (2007), pp. 141–224.

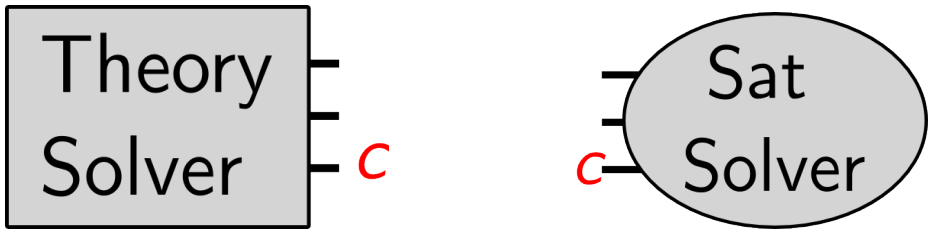
SAT modulo Theory Solvers

Efficient, lazy SMT[2] solvers apply unit propagation from the SAT solver to the theory solver *eagerly*

[2] R. Sebastiani. "Lazy satisfiability modulo theories". In: *Journal on Satisfiability, Boolean Modeling and Computation (JSAT)* 3 (2007), pp. 141–224.

SAT modulo Theory Solvers

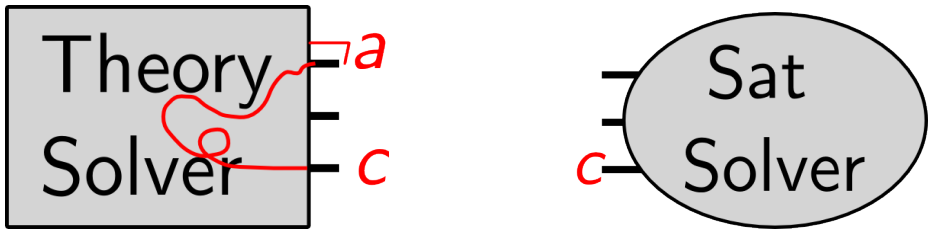
Efficient, lazy SMT[3] solvers apply unit propagation from the SAT solver to the theory solver *eagerly*



[3] R. Sebastiani. "Lazy satisfiability modulo theories". In: *Journal on Satisfiability, Boolean Modeling and Computation (JSAT)* 3 (2007), pp. 141–224.

SAT modulo Theory Solvers

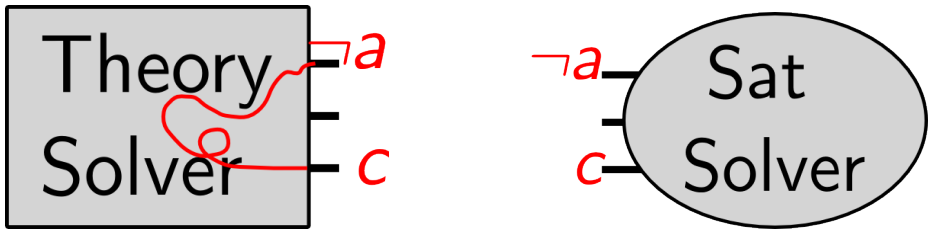
Efficient, lazy SMT[4] solvers apply unit propagation from the SAT solver to the theory solver *eagerly*



[4] R. Sebastiani. "Lazy satisfiability modulo theories". In: *Journal on Satisfiability, Boolean Modeling and Computation (JSAT)* 3 (2007), pp. 141–224.

SAT modulo Theory Solvers

Efficient, lazy SMT[5] solvers apply unit propagation from the SAT solver to the theory solver *eagerly*



[5] R. Sebastiani. "Lazy satisfiability modulo theories". In: *Journal on Satisfiability, Boolean Modeling and Computation (JSAT)* 3 (2007), pp. 141–224.

SAT modulo SAT

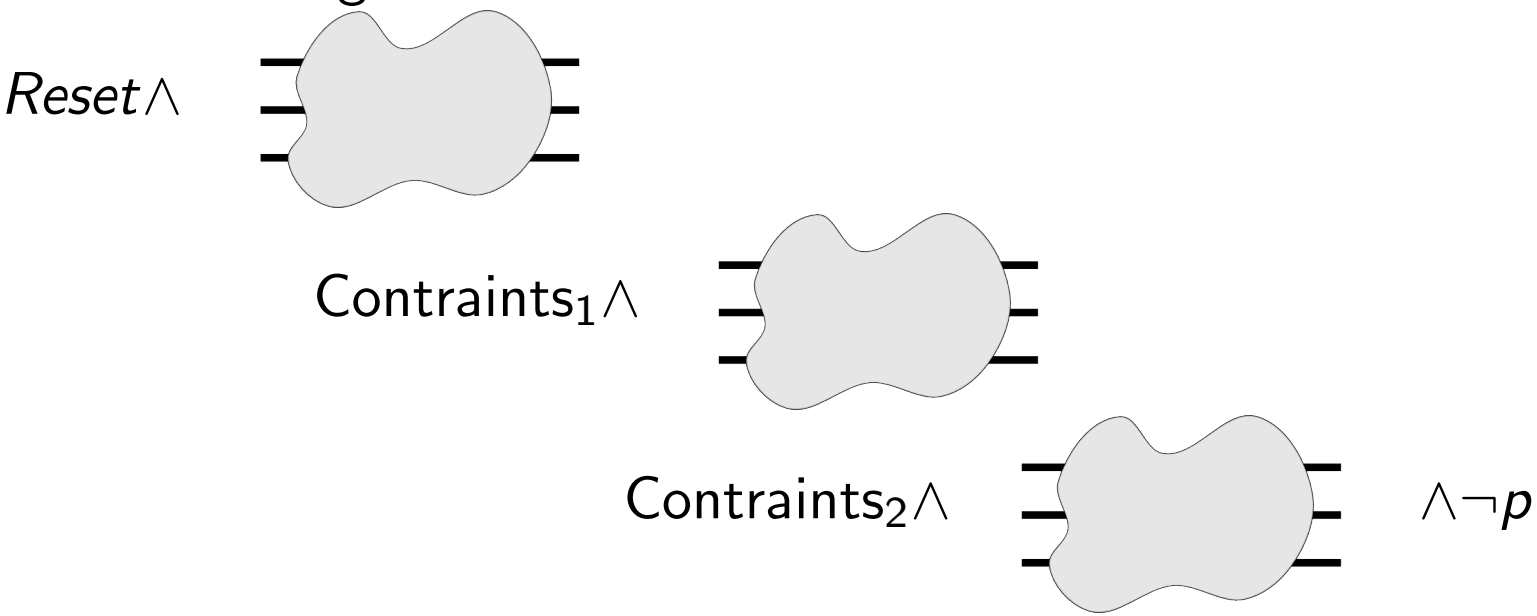
Observation: Incremental SAT solvers can support all the operations that lazy SMT theory solvers need

1. Eager unit propagation
2. Efficient learned clauses & lazy conflict analysis
3. Maintaining state in the theory solver

$\longrightarrow$ SAT is actually an *ideal* theory for lazy SMT solvers!
$\longrightarrow$ We can build a 'SAT modulo SAT' solver using all the tricks that lazy SMT solvers use!

Cube Blocking in IC3

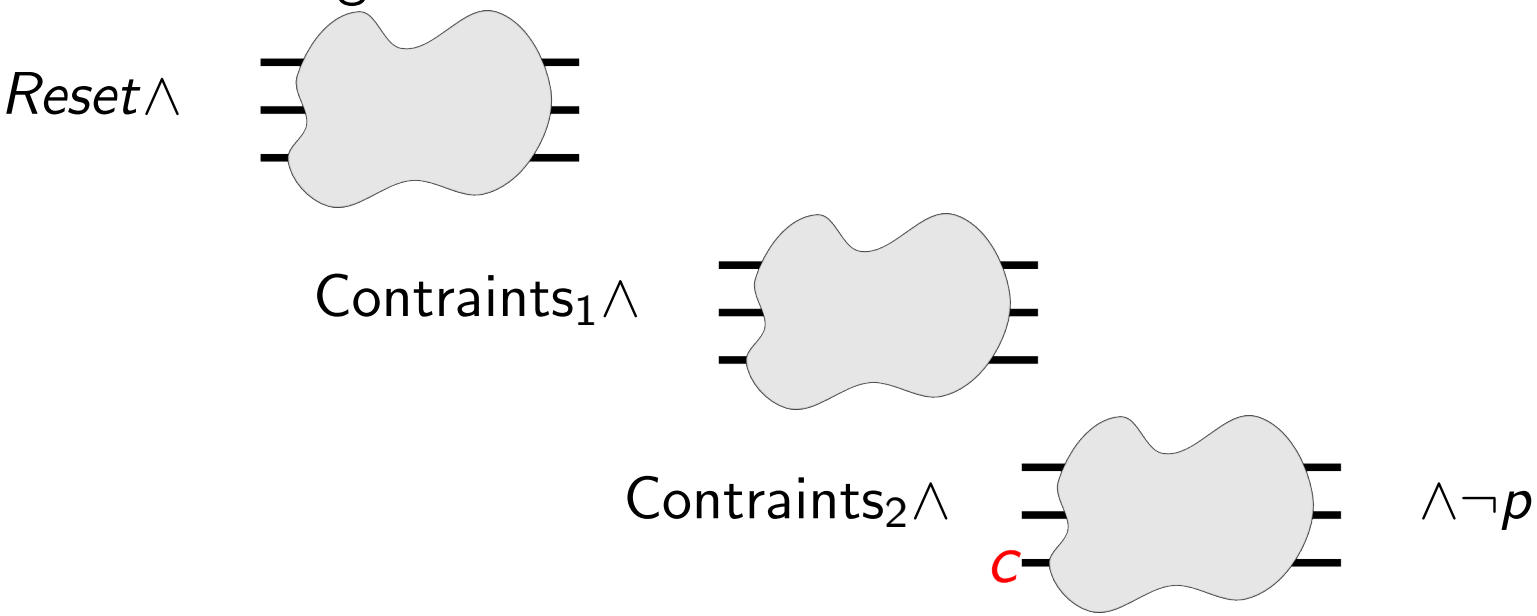
Cube Blocking:



$\longrightarrow$ Cube blocking is just solving a recursive SMT formula, where the 'theory' is the formula for the preceding time step.
$\longrightarrow$ We can use a SAT modulo SAT solver *directly* to apply cube blocking, giving us eager unit propagation and lazy conflict analysis!

Cube Blocking in IC3

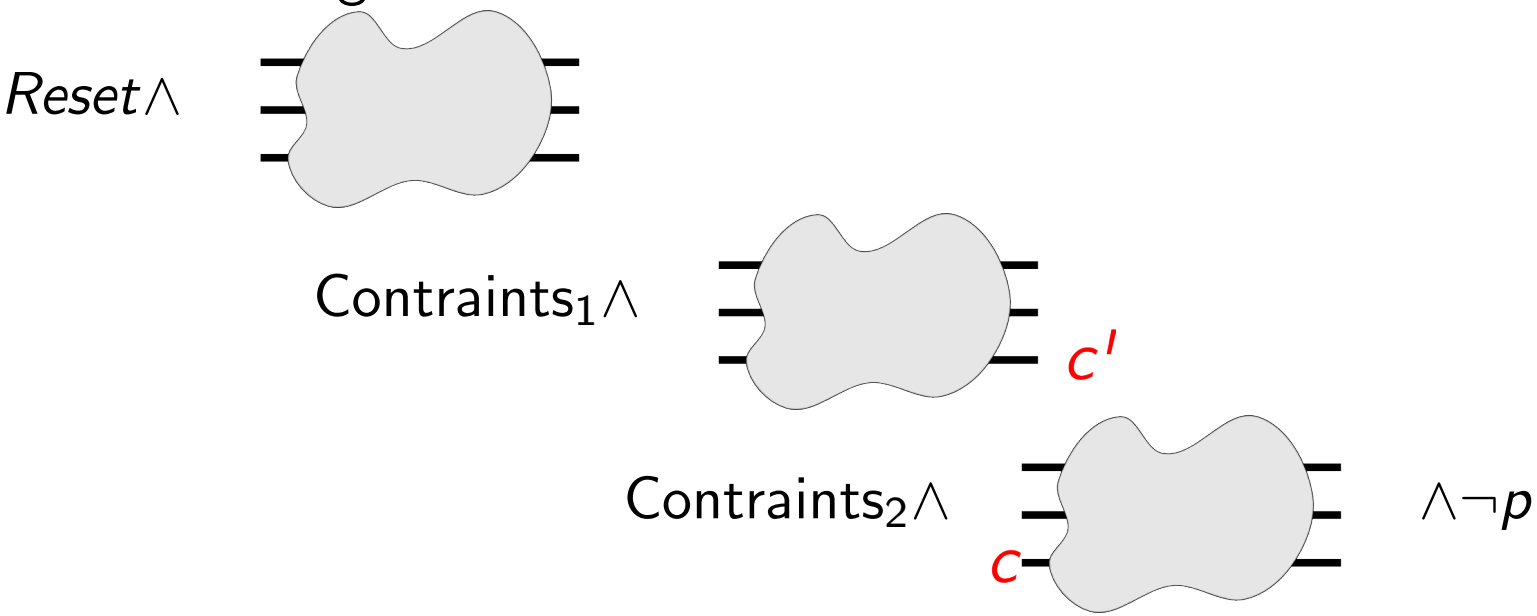
Cube Blocking:



$\longrightarrow$ Cube blocking is just solving a recursive SMT formula, where the 'theory' is the formula for the preceding time step.
$\longrightarrow$ We can use a SAT modulo SAT solver *directly* to apply cube blocking, giving us eager unit propagation and lazy conflict analysis!

Cube Blocking in IC3

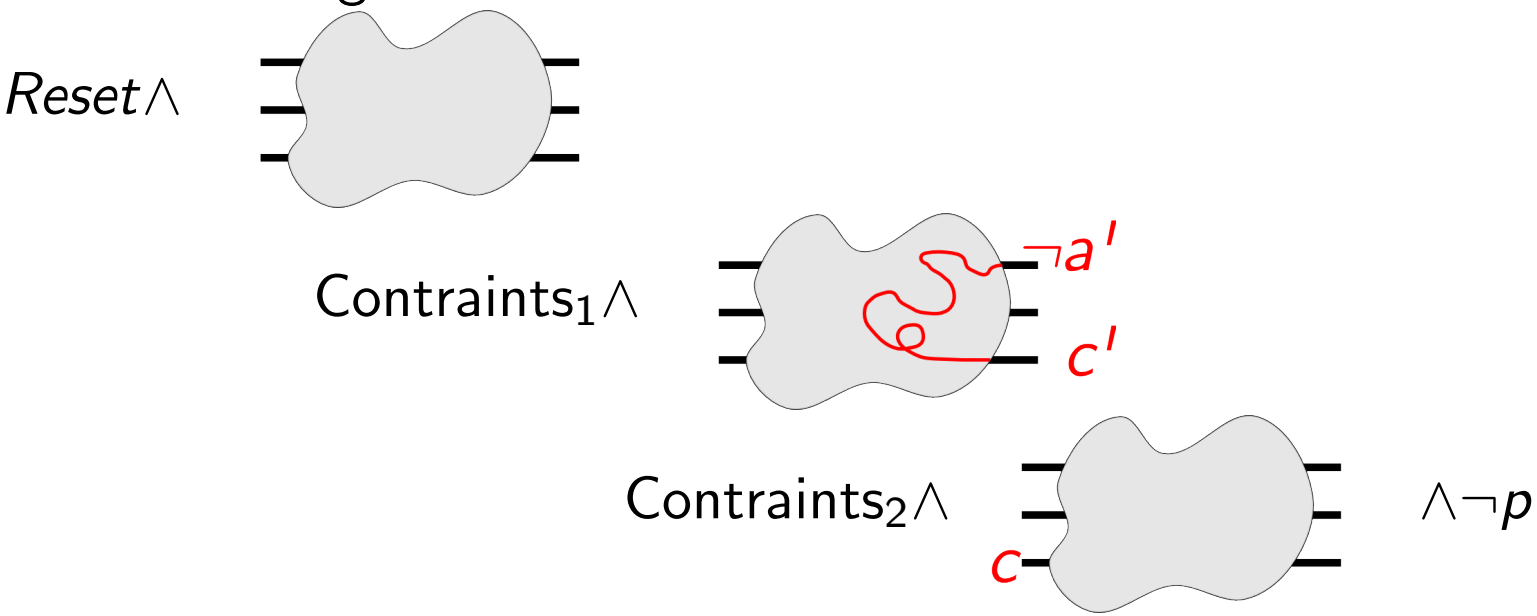
Cube Blocking:



$\longrightarrow$ Cube blocking is just solving a recursive SMT formula, where the 'theory' is the formula for the preceding time step.
$\longrightarrow$ We can use a SAT modulo SAT solver *directly* to apply cube blocking, giving us eager unit propagation and lazy conflict analysis!

Cube Blocking in IC3

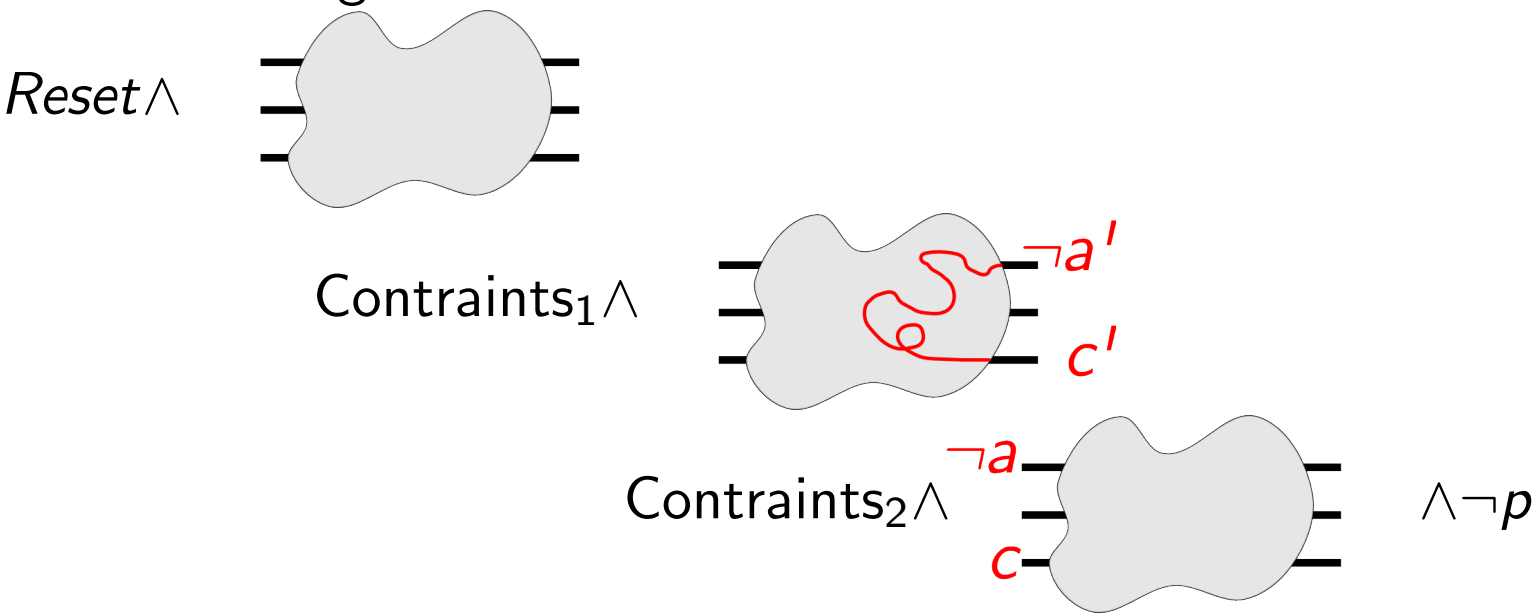
Cube Blocking:



$\longrightarrow$ Cube blocking is just solving a recursive SMT formula, where the 'theory' is the formula for the preceding time step.
$\longrightarrow$ We can use a SAT modulo SAT solver *directly* to apply cube blocking, giving us eager unit propagation and lazy conflict analysis!

Cube Blocking in IC3

Cube Blocking:



$\longrightarrow$ Cube blocking is just solving a recursive SMT formula, where the 'theory' is the formula for the preceding time step.
$\longrightarrow$ We can use a SAT modulo SAT solver *directly* to apply cube blocking, giving us eager unit propagation and lazy conflict analysis!

Results

Our 'SAT modulo SAT' version of PDR ('SMS-PDR') versus ABC's PDR and the original IC3:

	SMS-PDR	PDR	IC3
HWMCC'08	**596**/645	581/645	586/645
HWMCC'10	**749**/818	733/818	712/818
HWMCC'12	**92**/310	84/310	48/310

- Solves more instances than both IC3 and PDR on each benchmark
- Solves both more safe *and* unsafe instances than both IC3 and PDR on each benchmark
- Solves just one instance less than the entire virtual best solver from the 2008 competition

Results

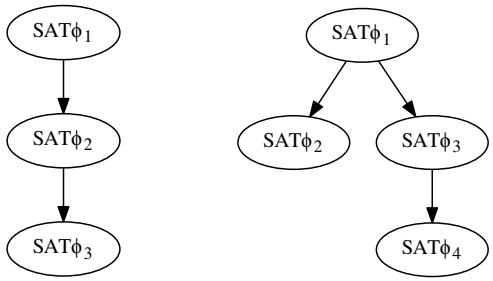
Our 'SAT modulo SAT' version of PDR ('SMS-PDR') versus ABC's PDR and the original IC3:

	SMS-PDR		PDR		IC3	
	SAT	UNSAT	SAT	UNSAT	SAT	UNSAT
HWMCC'08	**245**	**351**	242	339	240	346
HWMCC'10	**322**	**427**	317	416	308	404
HWMCC'12	**25**	**67**	21	63	14	34

- Solves more instances than both IC3 and PDR on each benchmark
- Solves both more safe *and* unsafe instances than both IC3 and PDR on each benchmark
- Solves just one instance less than the entire virtual best solver from the 2008 competition

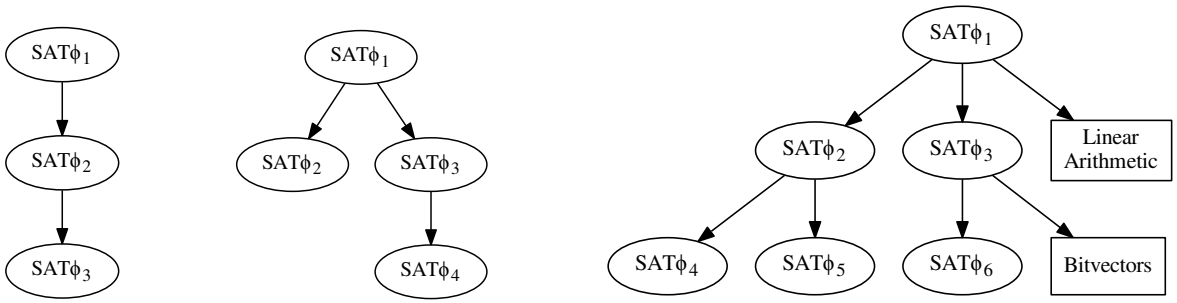
SAT modulo SAT

- SAT modulo SAT is not limited to just model checking
- Can be applied to many other types of formulas:
  - Not restricted to just unrolled circuits
  - Can handle partitions of arbitrary formulas
  - Can handle tree-structured formulas



SAT modulo SAT

- SAT modulo SAT is not limited to just model checking
- Can be applied to many other types of formulas:
  - Not restricted to just unrolled circuits
  - Can handle partitions of arbitrary formulas
  - Can handle tree-structured formulas
  - Can combine SAT and SMT theories



- The source code for our implementation of IC3, and for the new 'SAT modulo SAT' solver that it runs on, are open source & online at:

  `www.cs.ubc.ca/labs/isd/Projects/ModularSAT/`