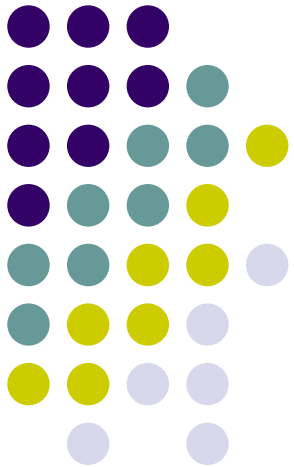
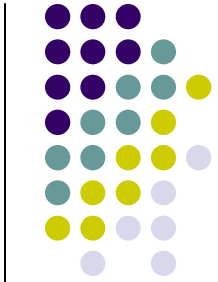
Restoring Circuit Structure From SAT Instances

Jarrod A. Roy, Igor L. Markov,
and Valeria Bertacco

University of Michigan at Ann Arbor

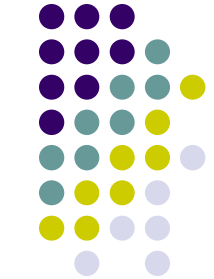


Outline



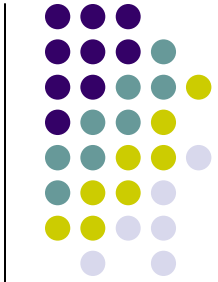
- Motivation
- Preliminaries
- A Generic Circuit Detection Algorithm
- AND-OR-NOT Circuit Conversion
- Additional Gate Types
- Empirical Results
- Conclusions and Further Work

Motivation



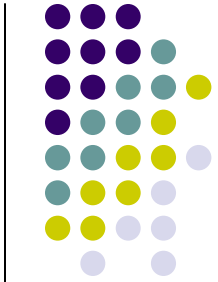
- SAT solvers have become a staple tool in EDA flows due to recent breakthroughs
- Circuit techniques improve performance on fully circuit derived instances
  - order of magnitude speedup
  - require circuit structure *a priori*
- Literature assumes “structure lost” in CNF
  - We find assumption *not necessarily* true

Motivation



- If we are converting to CNF, don't we already know the circuit structure?
  - True, no need waste time in this case
- On the other hand, we observe non-trivial structure from several sources
  - property checking: part circuit, part constraints
  - mathematical constructions: DIMACS `Pret` "encoded 2-colouring forced to be UNSAT"
- Automatically detecting structure and benefiting from it makes solvers more applicable for EDA

Converting Circuits to SAT



- All logic gates have a characteristic function
  - defines compatible assignments of inputs and outputs
- Converting a Circuit to CNF-SAT instance requires one variable per wire and several clauses per gate
- The conversion of gates to clauses is the encoding of each gate's characteristic function in CNF
  - we call it the CNF-signature of the gate

$$z = NAND(x_1, \ldots, x_j) \equiv \left[\prod_{i=1}^{j} (x_i + z)\right] \left(\sum_{i=1}^{j} \overline{x_i} + \overline{z}\right)$$

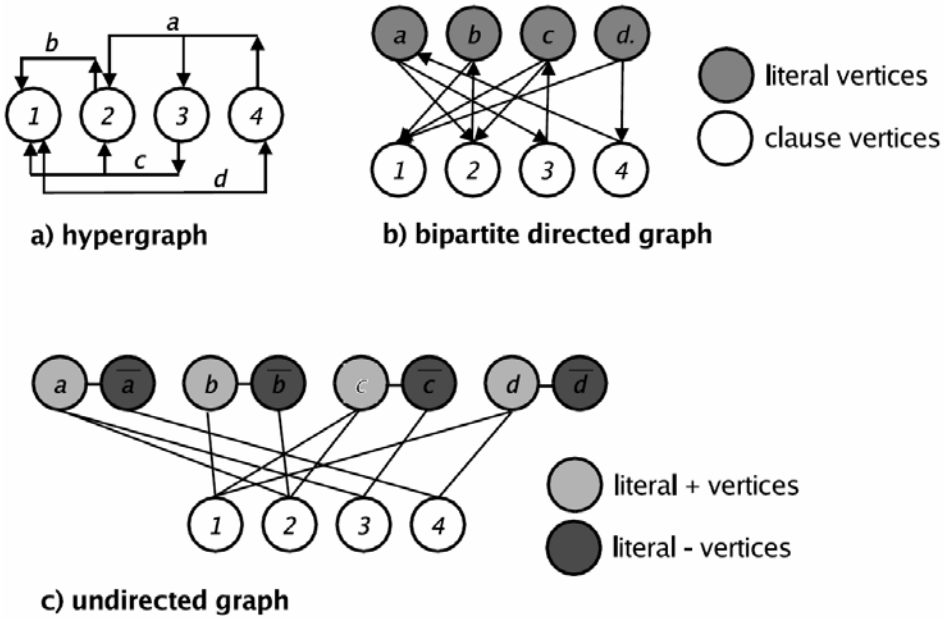
$$z = NOR(x_1, \ldots, x_j) \equiv \left[\prod_{i=1}^{j} (\overline{x_i} + \overline{z})\right] \left(\sum_{i=1}^{j} x_i + z\right)$$

A Generic Circuit Detection Algorithm

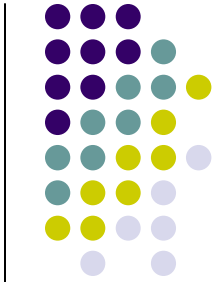
- Convert the CNF instance to an undirected graph
- Convert the CNF-signature of the gate to match to an undirected graph
- Use subgraph isomorphism to match instances of the gate

Conversions of the clauses

(b+d+c)(c+a+b')(a+c')(d+a')



A Generic Circuit Detection Algorithm



- To piece together the circuit, create a maximal independent set (MIS) instance
  - one node per detected gate
  - an edge between nodes if the gates are incompatible (signatures overlap, etc.)

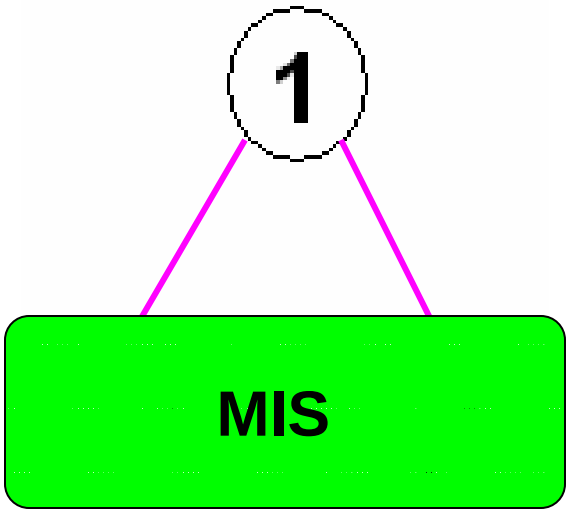
(a’+b)(a’+c)(a’+d)(b’+a)(b’+c)

(a+b’+c’)(a+b’+d’)(b+a’+c’)

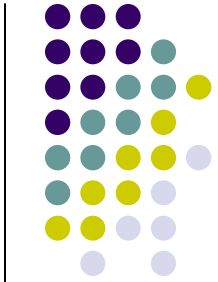
Encodes (1) a=AND(b,c),

(2) a=AND(b,d), and (3) b=AND(a,c)

Only (2) and (3) are compatible.



AND-OR-NOT Circuit Conversion



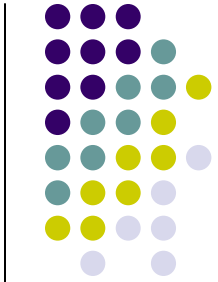
- Generic alg requires solving NP-hard problems
- Is there a more efficient way, possibly for a slightly more restricted problem?
- Yes: We prove the mapping from AND-OR-NOT circuits to CNF unique, no incompatible gate matches
  - Proof examines each clause of the CNF and shows it must have come from a specific gate
  - Proof suggests efficient linear time algorithm
    - based on pattern-matching of clauses

Easily Detectable Gate Types

Gate type	Difficulty of restoring circuit structure
OR *and* AND	Straightforward pattern-matching
NOR *and* NAND	Pattern-matching with back-tracking
NOT, XOR *and* XNOR	Can be detected by straightforward pattern-matching, but w/o orientation, which can only be determined in the context of other gate types
MAJ3	More advanced pattern matching with back-tracking

Table 1: The relative difficulty of detecting particular types of logic gates in CNF-SAT formulas. Note that this is not an exhaustive listing of detectable gates.

Spotlight on XOR/XNOR

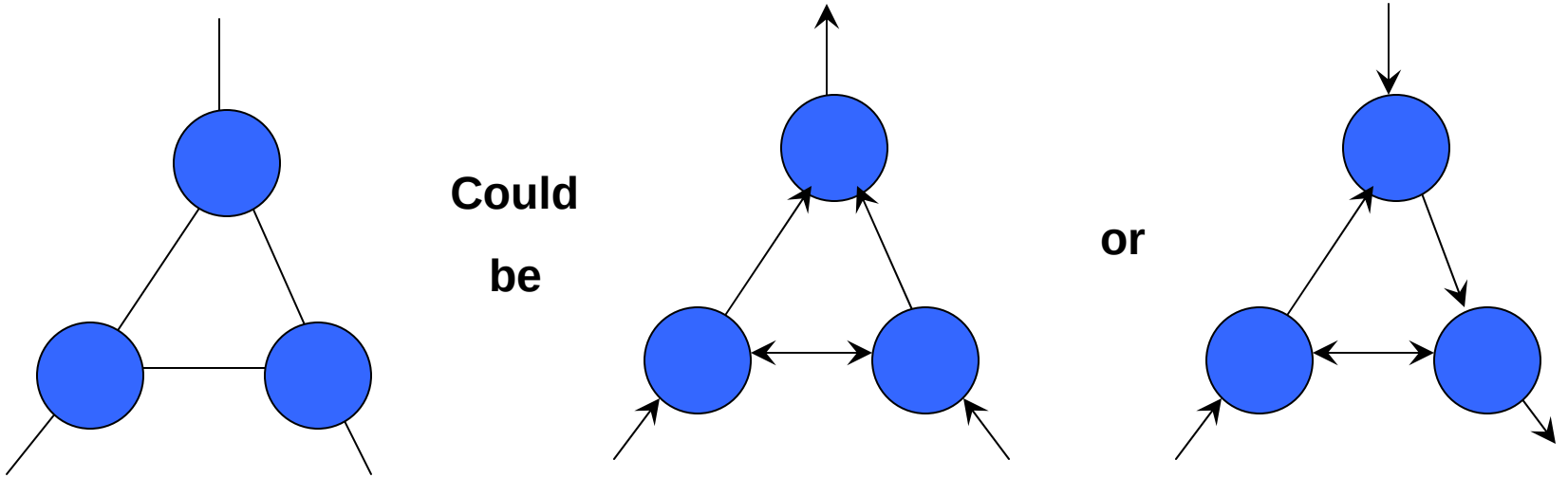


- XOR/XNOR gates are inherently unoriented
  - CNF-signatures are symmetric
  - the 2-input XOR gate
    a = XOR(b,c) has CNF signature
    (a'+b+c) (a+b'+c) (a+b+c') (a'+b'+c')

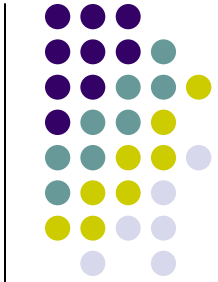
- Their detection is not all that difficult, but orientation requires proper context
  - With proper context, orientation
    can be propagated in a BFS like fashion

Spotlight on XOR/XNOR

- What happens without context?
  - Multiple valid interpretations; happens with a chain of XORS



Empirical Results



- Implemented detection of AND, OR, NAND, NOR, XOR, XNOR, NOT and MAJ3 gates
- Tested for the presence of structure in DIMACS, SAT2002 and Velev benchmarks
- Results show:
  - Much structure detected
    - sometimes in unexpected places
    - preserved by simplification
  - Technique is fast and scales well
    - small fraction of solving runtime

Structure in Standard Benchmarks

Benchmark series	% variables in simple gates	% clauses in simple gates	% variables in XOR/XNORs	% clauses in XOR/XNORs	Detection runtime (s)	# of benchmarks	# of variables	# of clauses
Bf	54.29%	22.12%	1.18%	0.54%	0.43	4	5793	16566
Dubois	0%	0%	100%	100%	0.09	13	1275	3400
Hanoi	43.22%	10.19%	0%	0%	0.37	2	2041	12272
Parity	33.17%	13.58%	88.35%	68.42%	2.68	30	24267	83330
Pret	0%	0%	100%	100%	0.09	8	840	2240
Ssa	47.25%	18.57%	1.45%	0.69%	1.09	8	7228	17669
XOR-Chain	0%	0%	100%	99.55%	0.38	27	4554	12126

Before SAT Preprocessor Hypre

After Hypre

Benchmark series	% variables in simple gates	% clauses in simple gates	% variables in XOR/XNORs	% clauses in XOR/XNORs	Detection runtime (s)	Hypre runtime (s)	% variables remaining	% clauses remaining
Bf	75.31%	46.41%	0.33%	0.12%	0.23	0.49	22.31%	31.17%
Dubois	0%	0%	100%	100%	0.09	0.05	100%	100%
Hanoi	52.24%	12.56%	0%	0%	0.24	0.23	64.97%	66.35%
Parity	30.70%	17.09%	100%	83.24%	2.13	1.47	54.99%	67.01%
Pret	0%	0%	100%	100%	0.09	0.03	100%	100%
Ssa	58.21%	29.86%	9.25%	4.74%	0.17	0.34	8.91%	8.33%
XOR-Chain	0%	0%	100%	100%	0.38	0.17	99.41%	99.55%

Scalability Results

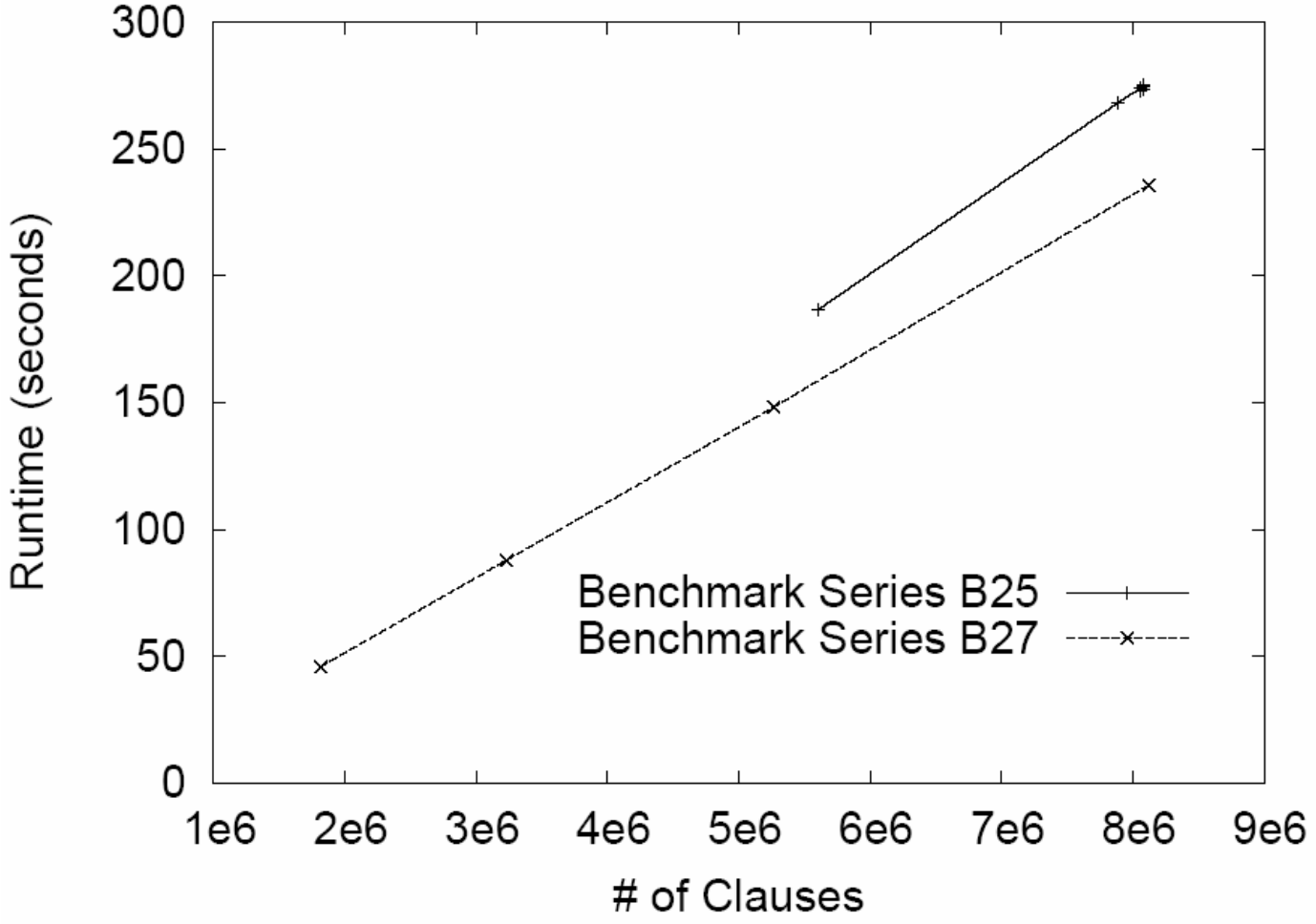


Figure 4: Runtime vs. SAT instance size on Velev's B25 and B27 series [18] of benchmarks.

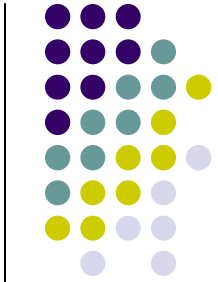
Comparison with Solving Runtime

Circuit	ZChaff	Im-plicit	Ex-plicit	Simu-lation	Ex-traction
9Vliw001	1057	567	793	93	10
9Vliw004	953	804	1011	92	10
9Vliw005	3126	740	1314	88	10
9Vliw007	140	286	855	110	11
9Vliw008	1450	239	1914	114	12
9Vliw009	1006	784	829	117	10
9Vliw010	867	329	1897	96	10
9Vliw015	2209	985	1270	97	10
9Vliw017	1007	175	913	109	10
9Vliw019	2936	849	1448	129	10
9Vliw021	1666	1069	1345	99	10
9Vliw024	1375	965	1282	107	9

Circuit Structure Extraction Time

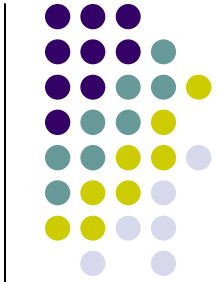
Circuit Based Technique Runtime = Simulation + (Implicit or Explicit Learning)

Conclusions



- Much circuit structure can be extracted efficiently
  - orientation can be difficult
- Tests show structure pops up in many places, possibly unbeknownst to the user
- Circuit-based SAT techniques vastly improve solving when given this structure
- Logical next step: extend general SAT solvers to make use of this structure

Further Work



- How difficult is it to detect other gate types such as AOI/OAI, ITE, etc.?
  - Recent work shows AOI/OAI as difficult as MAJ3
- Examine other methods for orienting inherently unoriented gate types
  - Guess and propagate, ….
- Is the original orientation of the circuit necessary, or will any valid orientation do?
  - If so, is the original orientation just better than other valid ones?