

SAT Solvers

Virendra Singh

Professor

Computer Architecture and Dependable Systems Lab

Department of Electrical Engineering

Indian Institute of Technology Bombay

<http://www.ee.iitb.ac.in/~viren/>

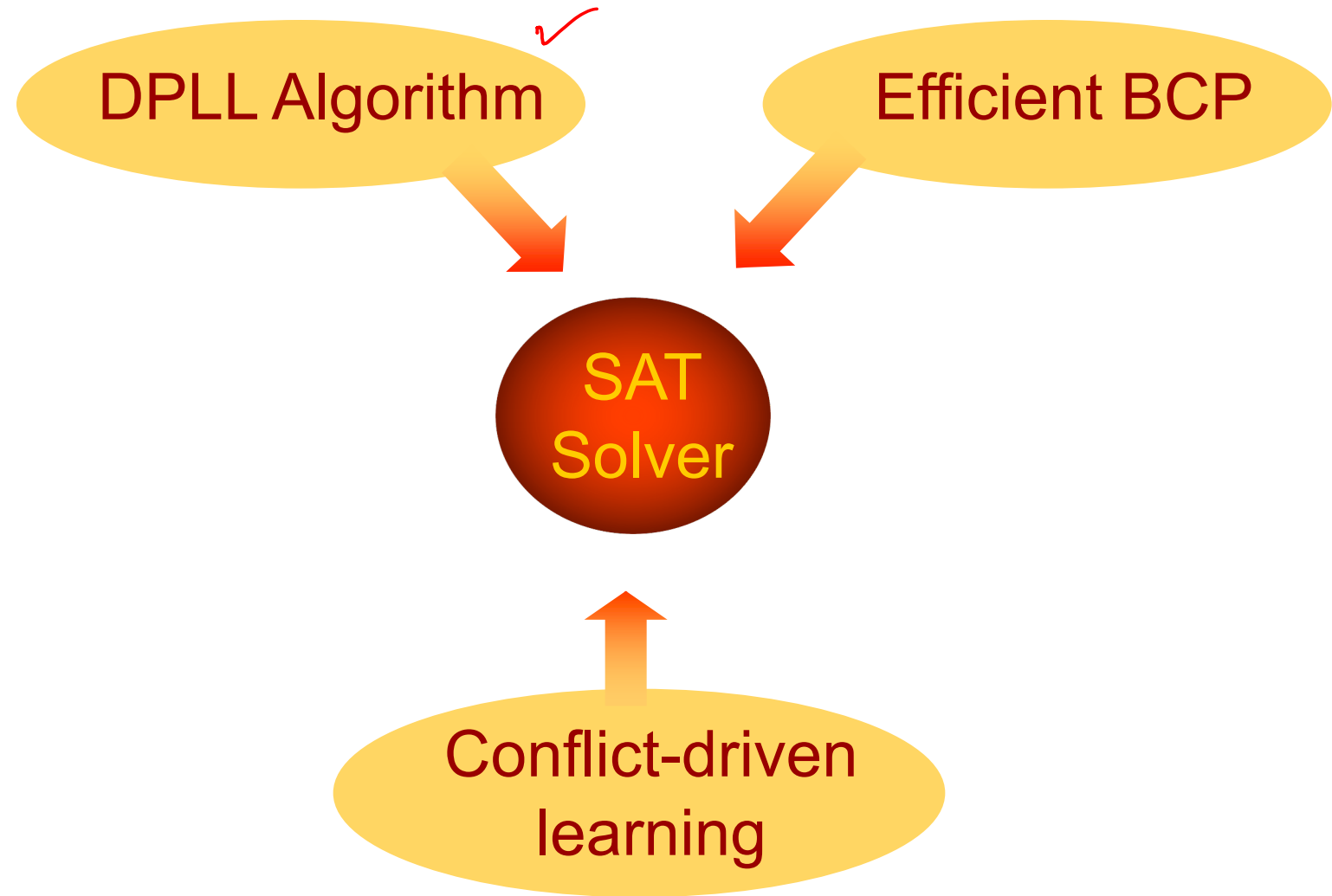
E-mail: viren@ee.iitb.ac.in

EE-224: Digital Systems



Lecture 15-A: 27 September 2020 **CADSL**

Anatomy of a Modern SAT Solver



Stallmarck's Method (SM) in CNF

- Recursive application of the **branch-merge rule** to each variable with the goal of identifying common conclusions

$$\varphi = (\overset{\checkmark}{a} + \overset{\checkmark}{b})(\overset{\checkmark}{\neg a} + \overset{\checkmark}{c})(\overset{\checkmark}{\neg b} + \overset{\checkmark}{d})(\overset{\checkmark}{\neg c} + \overset{\checkmark}{d})$$

(Note: In the original image, 'a' is green with '0' below it, 'b' is blue with '1' below it, '¬a' is red with '1' below it, 'c' is green with '1' below it, '¬b' is blue with '0' below it, 'd' is red with '1' below it, '¬c' is red with '1' below it, and 'd' is red with '1' below it. Red checkmarks are above each clause.)

Try $a = 0$: $(a = 0) \Rightarrow (b = 1) \Rightarrow (d = 1)$ $C(a = 0) = \{a = 0, b = 1, d = 1\}$

Try $a = 1$: $(\underline{a} = 1) \Rightarrow (c = 1) \Rightarrow (d = 1)$ $C(a = 1) = \{a = 1, c = 1, d = 1\}$

$$C(a = 0) \cap C(a = 1) = \underline{\underline{\{d = 1\}}}$$

Any assignment to variable a implies $d = 1$.
Hence, $d = 1$ is a **necessary** assignment !

Recursion can be of arbitrary depth



Recursive Learning (RL) in CNF

- Recursive evaluation of clause satisfiability requirements for identifying common assignments

$$\varphi = (a + b)(\neg a + d)(\neg b + d)$$

Try $a = 1$:

$$(a = 1) \Rightarrow (d = 1)$$

$$C(a = 1) = \{a = 1, d = 1\}$$

Try $b = 1$:

$$(b = 1) \Rightarrow (d = 1)$$

$$C(b = 1) = \{b = 1, d = 1\}$$

$$C(a = 1) \cap C(b = 1) = \{d = 1\}$$

Every way of satisfying $(a + b)$ implies $d = 1$.
Hence, $d = 1$ is a **necessary** assignment !

Recursion can be of arbitrary depth



Recursive Learning within DP

$$\varphi = (a + b + \dot{c})(\neg a + d + \dot{e})(\neg \dot{b} + d + \dot{c})$$

$$c=0 \quad e=0$$

$$d=1$$

Implications:

$$\begin{aligned} (a=1) \wedge (e=0) &\Rightarrow (d=1) \\ (b=1) \wedge (c=0) &\Rightarrow (d=1) \end{aligned}$$

consensus

$$(b + c + e + d)$$

consensus

$$(c=0) \wedge ((e=0) \wedge (c=0)) \Rightarrow (d=1)$$

Clausal form:

$$\frac{\bar{c} \cdot \bar{e} \rightarrow d}{(\bar{c} \cdot \bar{e} + d)} = (c + e + d)$$

$$(c + e + d)$$

(c + e + d) Unit clause !

Clause provides explanation
for necessary assignment $d=1$



SM vs. RL

- Both complete procedures for SAT
- Stallmarck's method:
 - hypothetical reasoning based on variables
- Recursive learning:
 - hypothetical reasoning based on clauses
- Both can be integrated into backtrack search algorithms

Local Search

- Repeat M times:
 - Randomly pick complete assignment
 - Repeat K times (and while exist unsatisfied clauses):
 - Flip variable that will satisfy largest number of unsat clauses

$$\varphi = (a + b)(\neg a + c)(\neg b + d)(\neg c + d)$$

$$\varphi = (\overset{\checkmark}{a} + \overset{\checkmark}{b})(\neg \overset{\checkmark}{a} + \overset{\checkmark}{c})(\neg \overset{\checkmark}{b} + \overset{\checkmark}{d})(\neg \overset{\checkmark}{c} + \overset{\checkmark}{d})$$

$$\varphi = (\overset{\checkmark}{a} + \overset{\checkmark}{b})(\neg \overset{\checkmark}{a} + \overset{\checkmark}{c})(\neg \overset{\checkmark}{b} + \overset{\checkmark}{d})(\neg \overset{\checkmark}{c} + \overset{\checkmark}{d})$$

Pick random assignment

$$\underline{a=1, b=1, c=1, d=0}$$

Flip assignment on d

Instance is **satisfied** !



Comparison

- Local search is incomplete
 - If instances are known to be SAT, local search can be competitive
- Stallmarck's Method (SM) and Recursive Learning (RL) are in general slow, though robust
 - SM and RL can derive too much **unnecessary** information
- For most EDA applications **backtrack search (DP)** is currently the most promising approach !
 - **Augmented with techniques for inferring new clauses/implicates (i.e. learning) !**

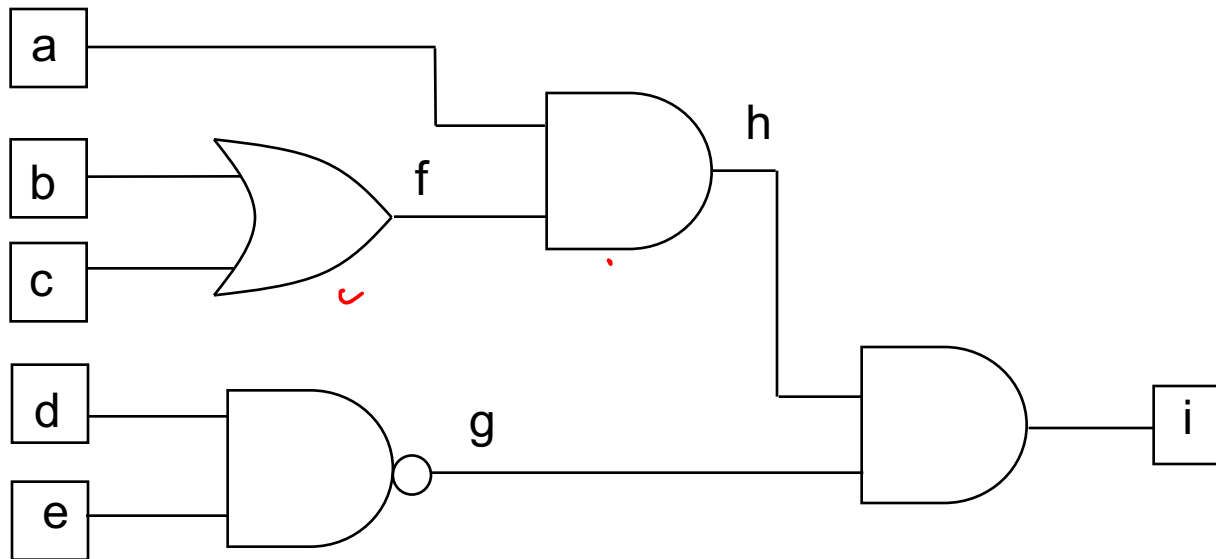


SAT Solvers Today

- Capacity: ✓
 - Formulas upto a *million variables* and *3-4 million clauses* can be solved in *few hours*
 - Only for *structured instances* e.g. derived from real-world circuits & systems
- Tool offerings:
 - ◆ Public domain
 - GRASP : Univ. of Michigan
 - SATO: Univ. of Iowa
 - zChaff: Princeton University ✓
 - BerkMin: Cadence Berkeley Labs.
 - ◆ Commercial
 - PROVER: Prover Technologies



Solving Circuit Problems as SAT

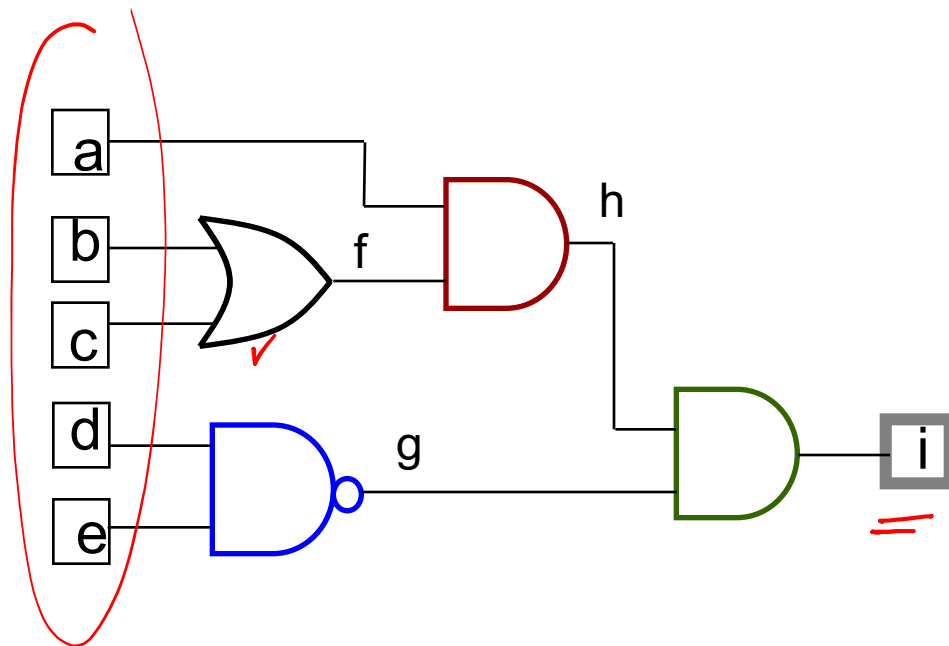


Input Vector Assignment ?

➡ Primary Output 'i' to 1 ?

Solving circuit problems as SAT

- Set of clauses representing function of each gate
- Unit literal clause asserting output to '1'

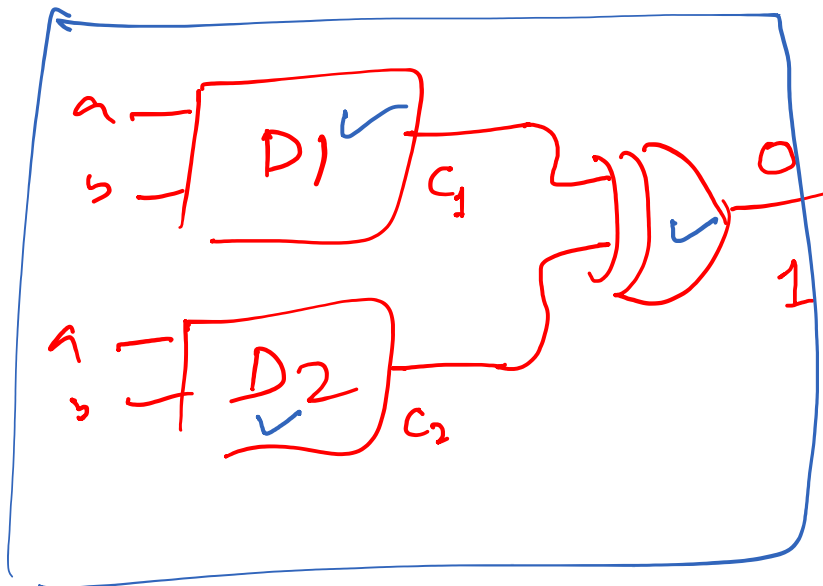


$$\begin{aligned}
 &\checkmark (\bar{b} + f)(\bar{c} + f)(b + c + \bar{f}) \\
 &(d + g)(e + g)(\bar{d} + \bar{e} + \bar{g}) \\
 &(a + \bar{h})(f + \bar{h})(\bar{a} + \bar{f} + h) \\
 &(h + \bar{i})(g + \bar{i})(\bar{h} + \bar{g} + i) \\
 &\quad \quad \quad \underline{(i)}
 \end{aligned}$$

Application of SAT

VERIFICATION

EQUIVALENCE CHECKING



$$D1 \stackrel{?}{=} D2$$

$$(SAT \text{ exp of } D1) \cdot (SAT \text{ exp } D2) \cdot (SAT \text{ exp XOR}) = 1$$

If input assignment exist

$$D1 \neq D2$$

If does not exist

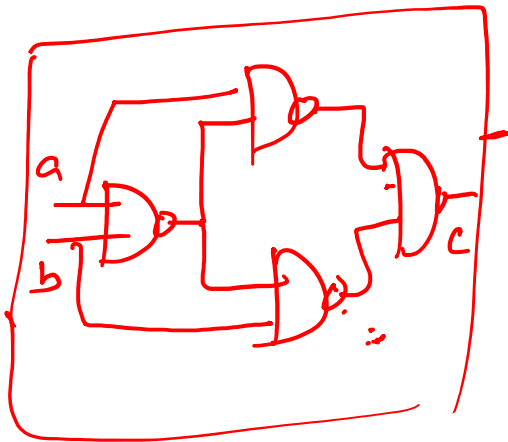
$$D1 \equiv D2$$

Application of SAT

$$a\bar{b} + \bar{a}b$$

✓

$\equiv ?$



Thank You

