Tutorial 1 : SAT USING Z3

CS60030 Formal Systems

PALLAB DASGUPTA,
FNAE, FASc,
A K Singh Distinguished Professor in AI,
Dept of Computer Science & Engineering
Indian Institute of Technology Kharagpur
Email: pallab@cse.iitkgp.ac.in
Web: http://cse.iitkgp.ac.in/~pallab

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

FMSAFE
FORMAL METHODS FOR SAFETY CRITICAL SYSTEMS

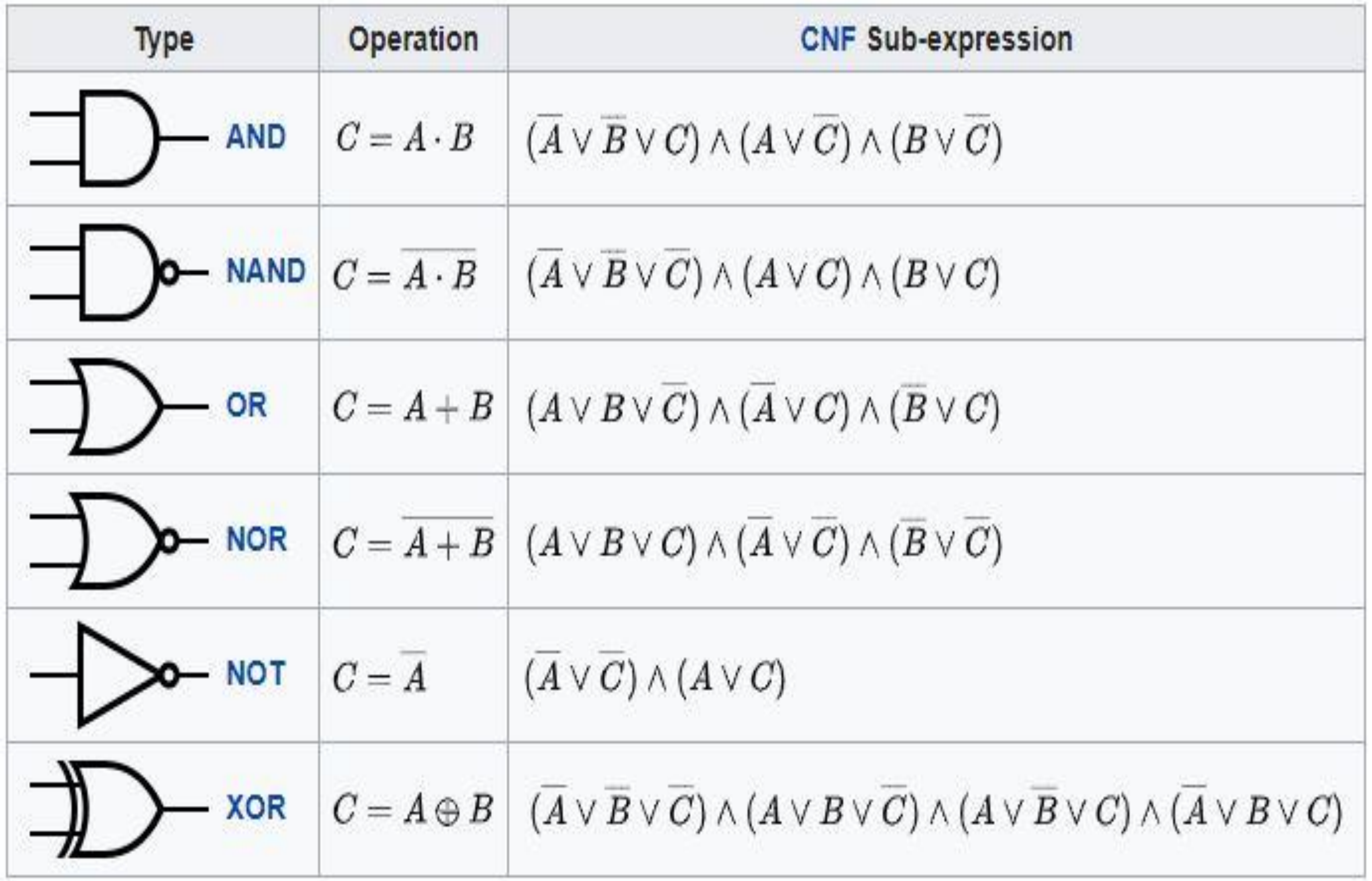
Hands-on Session (pre-requisites)

- **SAT solvers Z3 installed in the machines.**
  - Install Z3 in linux using : sudo apt install z3

CNF Representation of Combinational Logic Gates

Tseytin Transformation is a mechanism for converting any arbitrary combinational logic circuit into its equivalent Boolean formula in conjunctive normal form (CNF). For example, consider the following Boolean formula:

Characteristic function



Type	Operation	CNF Sub-expression
AND	$C = A \cdot B$	$(\overline{A} \vee \overline{B} \vee C) \wedge (A \vee \overline{C}) \wedge (B \vee \overline{C})$
NAND	$C = \overline{A \cdot B}$	$(\overline{A} \vee \overline{B} \vee \overline{C}) \wedge (A \vee C) \wedge (B \vee C)$
OR	$C = A + B$	$(A \vee B \vee \overline{C}) \wedge (\overline{A} \vee C) \wedge (\overline{B} \vee C)$
NOR	$C = \overline{A + B}$	$(A \vee B \vee C) \wedge (\overline{A} \vee \overline{C}) \wedge (\overline{B} \vee \overline{C})$
NOT	$C = \overline{A}$	$(\overline{A} \vee \overline{C}) \wedge (A \vee C)$
XOR	$C = A \oplus B$	$(\overline{A} \vee \overline{B} \vee \overline{C}) \wedge (A \vee B \vee \overline{C}) \wedge (A \vee \overline{B} \vee C) \wedge (\overline{A} \vee B \vee C)$

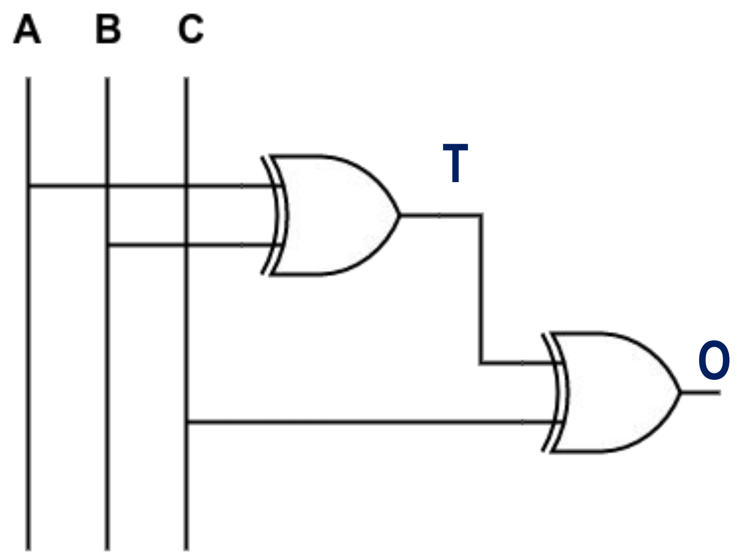
<u>AND Gate to CNF:</u>

$C \Leftrightarrow A \wedge B$

- $\equiv (C \Rightarrow A \wedge B) \wedge (A \wedge B \Rightarrow C)$
- $\equiv (\neg C \vee (A \wedge B)) \wedge (\neg(A \wedge B) \vee C)$
- $\equiv (\neg C \vee A) \wedge (\neg C \vee B) \wedge (\neg A \vee \neg B \vee C)$

SAT using Tseytin Transformation

Represent the combinational logic to SAT with Tseytin Transformation



$O = A \oplus B \oplus C$

$T \Leftrightarrow A \oplus B$

$(T \Rightarrow A \oplus B) \wedge ((A \oplus B) \Rightarrow T)$

$(\neg T \vee (A \oplus B)) \wedge (\neg(A \oplus B) \vee T)$

$(\neg T \vee ((A \vee B) \wedge (\neg A \vee \neg B))) \wedge (((\neg A \vee B) \wedge (A \vee \neg B)) \vee T)$

$(\neg T \vee A \vee B) \wedge (\neg T \vee \neg A \vee \neg B) \wedge (T \vee \neg A \vee B) \wedge (T \vee A \vee \neg B)$

$O \Leftrightarrow T \oplus C$

$(O \Rightarrow T \oplus C) \wedge ((T \oplus C) \Rightarrow O)$

$(\neg O \vee (T \oplus C)) \wedge (\neg(T \oplus C) \vee O)$

$(\neg O \vee ((T \vee C) \wedge (\neg T \vee \neg C))) \wedge (((\neg T \vee C) \wedge (T \vee \neg C)) \vee O)$

$(\neg O \vee T \vee C) \wedge (\neg O \vee \neg T \vee \neg C) \wedge (O \vee \neg T \vee C) \wedge (O \vee T \vee \neg C)$

DIMACS Format

- A file format which the SAT-solver takes as its input.
- A file can start with some comment lines. These are just text lines that start with a lower case "c". Example:

```
c This is a comment line
```

- After the initial comments, the next line of the file must tell how many variables (V a positive integer) and how many clauses (N a positive integer) in this CNF format:

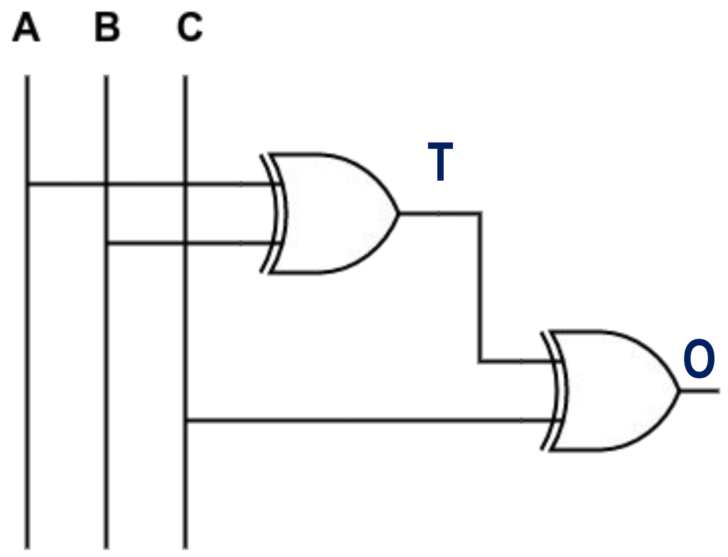
```
p cnf V N
```

- The next N lines of the file each specify one single clause. DIMACS format assumes your variables are x_1, x_2, x_3 ….. x_n. You specify a positive literal (like x_2 or x_7) in this clause with a positive integer (in this case, 2 or 7).
- Specify a negative, complemented literal with a negative integer(so $\neg x_5$ is -5 and $\neg x_{23}$ is -23).
- End each clause line with a 0.
- $(x_1 + \neg x_3)(x_2 + x_3 + \neg x_1)$ in DIMACS format looks like the following snippet.

```
c Comment line begins by ‘c’
c This is second comment line
p cnf 3 2
1 -3 0
2 3 -1 0
```

DIMACS Representation

Represent the following combinational logic gate using dimacs. You may use the Tseytin transformation to represent intermediate signals of the design.

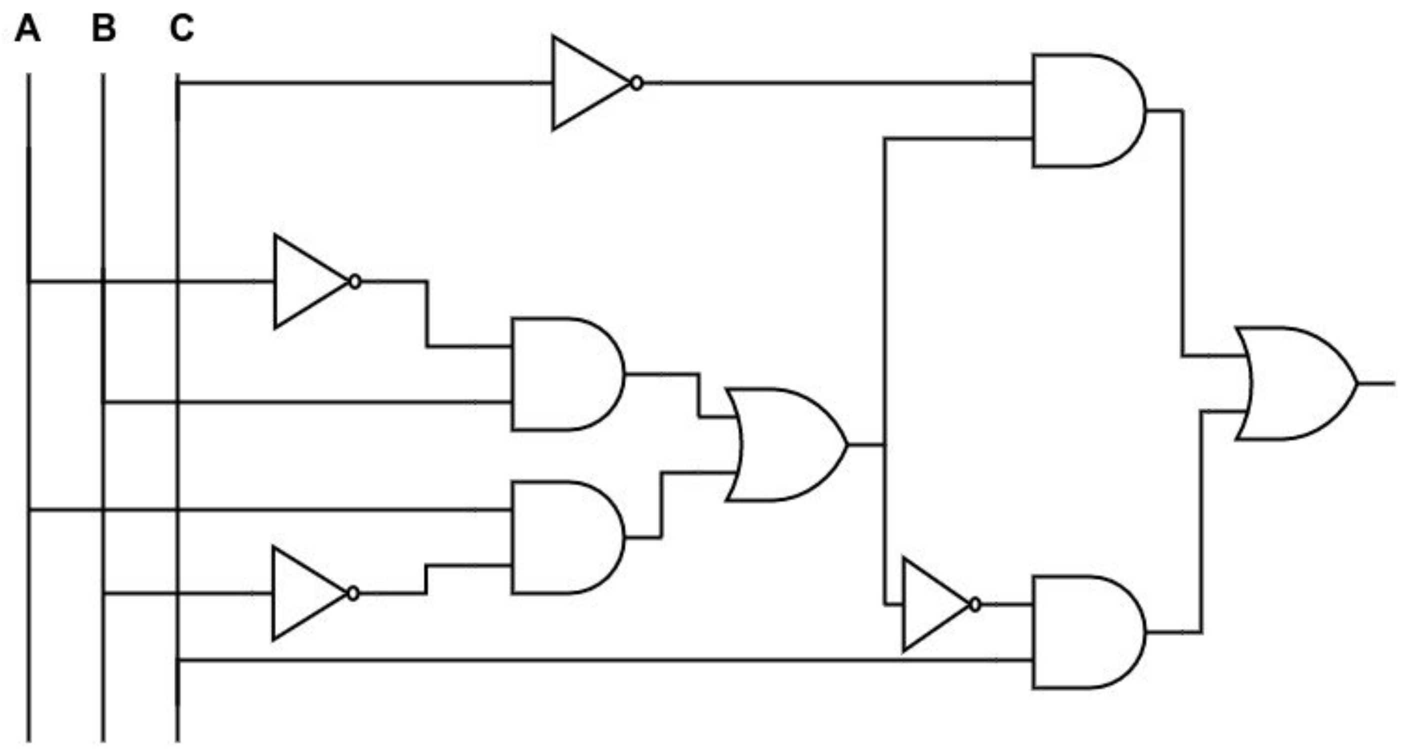


```
p cnf 5 8
-1 -2 -4 0
-1 2 4 0
1 -2 4 0
1 2 -4 0
-4 -3 -5 0
-4 3 5 0
4 -3 5 0
4 3 -5 0
```

Use the Z3 SAT solver to find satisfying assignments (or the lack of one) for the following cases:

- **A=1, B=1, Output = 0**
- **A=0, B=0, C=1**
- **Output = 1**
- **C=0, Output = 1**
- **A = B = C, Output = 0**

Equivalence Checking using SAT



Represent the given circuit in DIMACS format. Use the SAT solver to check if Circuit-1 and Circuit-2 are logically equivalent.

```
p cnf 13 27
-10 -1 0
10 1 0
-11 -2 0
11 2 0
-12 -3 0
12 3 0
-4 10 0
-4 2 0
-10 -2 4 0
-5 11 0
-5 1 0
-11 -1 5 0
-6 4 5 0
-4 6 0
-5 6 0
-6 -7 0
6 7 0
-8 7 0
-8 3 0
-3 -7 8 0
-9 12 0
-9 6 0
-6 9 -12 0
-13 8 9 0
13 -8 0
13 -9 0
13 0
```