### Outline

1. **Modeling with Propositional Logic**
   - SAT Example: Equivalence Checking if-then-else Chains
   - SAT Example: Circuit Equivalence Checking

2. **Formal Definition SAT**

3. **Conjunctive Normal Form**
   - Definition
   - Tseitin Transformation
   - DIMACS CNF

### SAT Example II

- Represent procedures as independent Boolean variables
  
  \[
  \text{original} := \\
  \begin{align*}
  &\text{if } \neg a \land \neg b \text{ then } h \\
  &\text{else if } \neg a \text{ then } g \\
  &\text{else } f
  \end{align*}
  \]

  \[
  \text{optimized} := \\
  \begin{align*}
  &\text{if } a \text{ then } f \\
  &\text{else if } b \text{ then } g \\
  &\text{else } h
  \end{align*}
  \]

- Compile if-then-else chains into Boolean formulae

- Check equivalence of Boolean formulae

### "Compilation"

- \[
  \text{original} \equiv \begin{align*}
  &\neg a \land \neg b \text{ then } h \\
  &\text{else if } \neg a \text{ then } g \\
  &\text{else } f
  \end{align*}
  \]

- \[
  \text{optimized} \equiv \begin{align*}
  &a \text{ then } f \\
  &\text{else if } b \text{ then } g \\
  &\text{else } h
  \end{align*}
  \]

- \[
  (\neg a \land \neg b) \lor \neg (\neg a \land \neg b) \land (\neg a \land g \lor a \land f) \equiv a \land f \lor \neg a \land (b \land g \lor \neg b \land h)
  \]
How to Check (In)Equivalence?

Reformulate it as a satisfiability (SAT) problem:

Is there an assignment to \( a, b, f, g, h \), which results in different evaluations of original and optimized?

or equivalently:

Is the boolean formula \( \text{compile}(\text{original}) \neq \text{compile}(\text{optimized}) \) satisfiable?

Such an assignment provides an easy to understand counterexample.

SAT

SAT (Satisfiability) the classical NP-complete problem:

Given a propositional formula \( f \) over \( n \) propositional variables \( V = \{x, y, \ldots\} \).

Is there an assignment \( \sigma : V \rightarrow \{0, 1\} \) with \( \sigma(f) = 1 \) ?

Conjunctive Normal Form

Definition (Conjunctive Normal Form)

A formula in Conjunctive Normal Form (CNF) is a conjunction of clauses 

\[ C_1 \land C_2 \land \ldots \land C_n \]

each clause \( C \) is a disjunction of literals 

\[ C = L_1 \lor \ldots \lor L_m \]

and each literal is either a plain variable \( x \) or a negated variable \( \overline{x} \).

Example \( (a \lor b \lor c) \land (\overline{a} \lor \overline{b}) \land (\overline{a} \lor \overline{c}) \)

CNF for Parity Function is Exponential

- No merging in the Karnaugh map
- All clauses contain all variables
- CNF for parity with \( n \) variables has \( 2^n - 1 \) clauses

Better ideas?
For each gate produce complete input / output constraints as clauses.
Collect all constraints in a big conjunction.


Example of Tseitin Transformation: Circuit to CNF

\[ o \land (x \rightarrow a) \land (x \rightarrow c) \land (x \leftarrow a \land c) \land \ldots \]

\[ o \land (\pi \lor a) \land (\pi \lor c) \land (x \lor \pi \lor r) \land \ldots \]

Algorithmic Description of Tseitin Transformation


Tseitin Transformation

- For each non input circuit signal \( s \) generate a new variable \( x_s \).
- For each gate produce complete input / output constraints as clauses.
- Collect all constraints in a big conjunction.


Optimizations for the Tseitin Transformation

- Goal is smaller CNF (less variables, less clauses).
  - Extract multi argument operands (removes variables for intermediate nodes).
  - NNF: half of AND, OR node constraints may be removed due to monotonicity.
  - Use sharing.


DIMACS CNF

- DIMACS CNF format = standard format for CNF.
- Used by most SAT solvers.
- Plain text file with following structure:
  
  p  cad <# variables> <# clauses>
  <clause>: 0
  <clause>: 0
  ...

- One or more lines per clause.


DIMACS CNF

- Every clause is a list of numbers, separated by spaces.
- A clause ends with 0.
- Every number 1, 2, . . . corresponds to a variable.
- Variable names (e.g., \( a, b, \ldots \)) have to be mapped to numbers.
- A negative number corresponds to negation.
- Let \( a \) have number 5. Then \( \pi \) is -5.
Example SAT: Circuit Equivalence

Let's change the circuit!

Is the CNF satisfiable?

Satisfying assignment mapped to the circuit:

<table>
<thead>
<tr>
<th>variable</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>0</td>
</tr>
<tr>
<td>x</td>
<td>0</td>
</tr>
<tr>
<td>u</td>
<td>1</td>
</tr>
<tr>
<td>o</td>
<td>1</td>
</tr>
<tr>
<td>w</td>
<td>1</td>
</tr>
</tbody>
</table>

Example SAT: Circuit Equivalence

Output of the SAT solver:

SATISFIABLE

1 2 3 4 -5 -6 7 8 9

Values of the variables:

<table>
<thead>
<tr>
<th>variable</th>
<th>number</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>a</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>c</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>x</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>y</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>u</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>v</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>w</td>
<td>9</td>
<td>1</td>
</tr>
</tbody>
</table>

Caveat: there is more than one solution

Example SAT: Circuit Equivalence

Let's change the circuit!

Is the CNF satisfiable?

Satisfying assignment mapped to the circuit:

<table>
<thead>
<tr>
<th>variable</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td>1</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
</tr>
<tr>
<td>c</td>
<td>1</td>
</tr>
<tr>
<td>x</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
</tr>
<tr>
<td>y</td>
<td>0</td>
</tr>
<tr>
<td>u</td>
<td>1</td>
</tr>
<tr>
<td>v</td>
<td>1</td>
</tr>
<tr>
<td>w</td>
<td>1</td>
</tr>
</tbody>
</table>