## Classical Logic Operations

<table>
<thead>
<tr>
<th></th>
<th>(a \land b)</th>
<th>(a \lor b)</th>
<th>(a \leftarrow b)</th>
<th>(a \rightarrow b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>(a \neg\land b)</th>
<th>(a \neg\lor b)</th>
<th>(a \nleftarrow b)</th>
<th>(a \nrightarrow b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>(a \oplus b)</th>
<th>(a \equiv b)</th>
<th>(ite(a,b,c))</th>
<th>(\neg a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\(f\) is the value of the expression in parens
What is Satisfiability?

\[(\overline{v_0} \lor v_1 \lor \overline{v_2}) \land (\overline{v_1} \lor \overline{v_3}) \land (v_0 \lor v_3 \lor \overline{v_4} \lor \overline{v_5}) \land (v_3)\]

**CNF** (Conjunctive Normal Form)

- Boolean variables taking value 1 or 0.
- Clauses: disjunctions of variables or negations.
- Formula or instance of SAT: conjunction of clauses.

**Is there a truth assignment satisfying all clauses?**

Yes: \(v_0 = 0, v_1 = 0, v_3 = 1, \ldots\)

**Is there an efficient algorithm for finding a solution?**

Sometimes there is and sometimes we are not sure!...
Why CNF?

1. All expressions translate to CNF efficiently

<table>
<thead>
<tr>
<th>$O_x$</th>
<th>Equivalent CNF Expression</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>$(\overline{v_x})$</td>
<td>$v_x \iff 0$</td>
</tr>
<tr>
<td>1111</td>
<td>$(v_x)$</td>
<td>$v_x \iff 1$</td>
</tr>
<tr>
<td>0011</td>
<td>$(a \lor \overline{v_x}) \land (\overline{a} \lor v_x)$</td>
<td>$a \iff 1$</td>
</tr>
<tr>
<td>1100</td>
<td>$(a \lor v_x) \land (\overline{a} \lor \overline{v_x})$</td>
<td>$a \iff 0$</td>
</tr>
<tr>
<td>0101</td>
<td>$(b \lor \overline{v_x}) \land (\overline{b} \lor v_x)$</td>
<td>$b \iff 1$</td>
</tr>
<tr>
<td>1010</td>
<td>$(b \lor v_x) \land (\overline{b} \lor \overline{v_x})$</td>
<td>$b \iff 0$</td>
</tr>
<tr>
<td>0001</td>
<td>$(a \lor \overline{v_x}) \land (b \lor \overline{v_x}) \land (\overline{a} \lor \overline{b} \lor v_x)$</td>
<td>$v_x \iff (a \land b)$</td>
</tr>
<tr>
<td>1110</td>
<td>$(a \lor v_x) \land (b \lor v_x) \land (\overline{a} \lor \overline{b} \lor \overline{v_x})$</td>
<td>$v_x \iff (a \overline{n} b)$</td>
</tr>
<tr>
<td>0010</td>
<td>$(\overline{a} \lor \overline{v_x}) \land (b \lor v_x) \land (a \lor \overline{b} \lor v_x)$</td>
<td>$v_x \iff (a \not\rightarrow b)$</td>
</tr>
<tr>
<td>1101</td>
<td>$(a \lor \overline{v_x}) \land (\overline{b} \lor v_x) \land (\overline{a} \lor b \lor \overline{v_x})$</td>
<td>$v_x \iff (a \rightarrow b)$</td>
</tr>
<tr>
<td>0100</td>
<td>$(a \lor \overline{v_x}) \land (\overline{b} \lor \overline{v_x}) \land (\overline{a} \lor b \lor v_x)$</td>
<td>$v_x \iff (a \not\leftarrow b)$</td>
</tr>
<tr>
<td>1011</td>
<td>$(\overline{a} \lor v_x) \land (b \lor v_x) \land (a \lor \overline{b} \lor \overline{v_x})$</td>
<td>$v_x \iff (a \leftarrow b)$</td>
</tr>
<tr>
<td>1000</td>
<td>$(\overline{a} \lor v_x) \land (\overline{b} \lor \overline{v_x}) \land (a \lor b \lor v_x)$</td>
<td>$v_x \iff (a \lor b)$</td>
</tr>
<tr>
<td>0111</td>
<td>$(\overline{a} \lor v_x) \land (\overline{b} \lor \overline{v_x}) \land (a \lor \overline{b} \lor v_x)$</td>
<td>$v_x \iff (a \lor b)$</td>
</tr>
<tr>
<td>1001</td>
<td>$(a \lor \overline{b} \lor \overline{v_x}) \land (\overline{a} \lor b \lor \overline{v_x}) \land$</td>
<td>$v_x \iff (a \equiv b)$</td>
</tr>
<tr>
<td></td>
<td>$(\overline{a} \lor b \lor v_x) \land (a \lor b \lor v_x)$</td>
<td></td>
</tr>
<tr>
<td>0110</td>
<td>$(\overline{a} \lor b \lor v_x) \land (a \lor \overline{b} \lor v_x) \land$</td>
<td>$v_x \iff (a \oplus b)$</td>
</tr>
<tr>
<td></td>
<td>$(a \lor b \lor \overline{v_x}) \land (\overline{a} \lor \overline{b} \lor \overline{v_x})$</td>
<td></td>
</tr>
</tbody>
</table>
Why CNF?

\[ v_0 \equiv \left( (\overline{v_0} \equiv (v_1 \lor \overline{v_2})) \land (v_1 \lor \overline{v_2}) \land (\overline{v_2} \rightarrow \overline{v_3} \rightarrow v_4) \right) \]
Why CNF?

\[ v_0 \equiv \left( (\overline{v_0} \equiv (v_1 \lor v_2)) \land (v_1 \lor v_2) \land (\overline{v_2} \rightarrow \overline{v_3} \rightarrow v_4) \right) \]
Why CNF?

$v_0 \equiv ((\overline{v_0} \equiv (v_1 \lor \overline{v_2})) \land (v_1 \lor \overline{v_2}) \land (\overline{v_2} \rightarrow \overline{v_3} \rightarrow v_4))$

translates to...

$((v_0 \lor v_{x1}) \land (\overline{v_0} \lor \overline{v_{x1}}) \land$

$(v_2 \lor v_{x2}) \land (\overline{v_2} \lor \overline{v_{x2}}) \land$

$(v_3 \lor v_{x3}) \land (\overline{v_3} \lor \overline{v_{x3}}) \land$

$(v_{x1} \lor v_{x4}) \land (\overline{v_{x2}} \lor v_{x4}) \land (v_{x1} \lor v_{x2} \lor \overline{v_{x4}}) \land$

$(v_{x3} \lor v_{x5}) \land (\overline{v_{x4}} \lor v_{x5}) \land (\overline{v_{x3}} \lor v_{x4} \lor \overline{v_{x5}}) \land$

$(v_{x1} \lor v_{x4} \lor \overline{v_{x6}}) \land (v_{x1} \lor v_{x4} \lor \overline{v_{x6}}) \land (\overline{v_{x1}} \lor \overline{v_{x4}} \lor v_{x6}) \land (v_{x1} \lor v_{x4} \lor v_{x6}) \land$

$(v_{x2} \lor v_{x7}) \land (\overline{v_{x5}} \lor v_{x7}) \land (\overline{v_{x2}} \lor v_{x5} \lor \overline{v_{x7}}) \land$

$(v_{x4} \lor v_{x8}) \land (v_{x7} \lor \overline{v_{x8}}) \land (v_{x4} \lor \overline{v_{x7}} \lor v_{x8}) \land$

$(v_{x6} \lor \overline{v_{x9}}) \land (v_{x8} \lor \overline{v_{x9}}) \land (v_{x6} \lor \overline{v_{x8}} \lor v_{x9}) \land$

$(v_{x10} \lor \overline{v_{x10}}) \land (v_{x10} \lor v_{x10} \lor \overline{v_{x10}}) \land (v_{x10} \lor v_{x10} \lor v_{x10}) \land (v_{x10} \lor v_{x10} \lor v_{x10}) \land$

$(v_{x10})$
Why CNF?

2. There are a variety of techniques available for solving CNF formulas
   a. Resolution
   b. Extended Resolution
   c. Gröbner Bases
   d. Cutting planes
   e. Clausal learning
   f. Stochastic Local Search
   g. Survey propagation
   h. ...
2. There are a variety of techniques available for solving CNF formulas
   a. Resolution
   b. Extended Resolution
   c. Gr"obner Bases
   d. Cutting planes
   e. Clausal learning
   f. Stochastic Local Search
   g. Survey propagation
   h. ...

3. Probabilistic analysis of CNF formulas is easier than for other forms - the concept of random formula is natural for CNF and results are rich
Maximum Satisfiability

- Each clause has a positive integer weight.
- The weight of a formula w.r.t. an assignment: the sum of the weights of satisfied clauses.
- Find an assignment which maximizes formula weight.
Representations

Sets:
\[ \psi = \{ \{v_0, v_1, v_2\}, \{v_1, v_3\}, \{v_0, v_3, v_4, v_5\}, \{v_3\} \} \]
Representations

Sets:
\[ \psi = \{\{\overline{v_0}, v_1, \overline{v_2}\}, \{\overline{v_1}, \overline{v_3}\}, \{v_0, v_3, \overline{v_4}, \overline{v_5}\}, \{v_3\}\} \]

Resolution:
\[ \psi_v = \{c - \{v\} : c \in \psi, v \notin c\} \]
\[ \psi_{\overline{v}} = \{c - \{\overline{v}\} : c \in \psi, \overline{v} \notin c\} \]

\[ \{\{\overline{v_0}, v_1, \overline{v_2}\}, \{\overline{v_1}, \overline{v_3}\}, \{v_0, v_3, \overline{v_4}, \overline{v_5}\}, \{v_3\}\} \]

\[ v_0 = 1 \quad v_0 = 0 \]

\[ \{\{\overline{v_2}\}, \{\overline{v_1}, \overline{v_3}\}, \{v_3\}\} \quad \{\{\overline{v_1}, \overline{v_3}\}, \{v_3, \overline{v_4}, \overline{v_5}\}, \{v_3\}\} \]

\[ \psi|_{v_0} \quad \psi|_{\overline{v_0}} \]
Representations

Matrix:

\[
\begin{pmatrix}
v_0 & v_1 & v_2 & v_3 & v_4 & v_5 \\
c_0 & 0 & 1 & 0 & 0 & 0 & 0 \\
c_1 & 1 & -1 & -1 & -1 & 0 & 0 \\
c_2 & 0 & -1 & 0 & 0 & -1 & 1 \\
c_3 & 0 & -1 & 0 & -1 & 0 & -1 \\
\end{pmatrix}
\]

\((v_1) \land (v_0 \lor \overline{v}_1 \lor \overline{v}_2 \lor \overline{v}_3) \land (\overline{v}_1 \lor \overline{v}_4 \lor v_5) \land (\overline{v}_1 \lor \overline{v}_3 \lor v_5)\)
Representations

Matrix:

\[
\begin{pmatrix}
v_0 & v_1 & v_2 & v_3 & v_4 & v_5 \\
\begin{array}{cccccc}
c_0 & 0 & 1 & 0 & 0 & 0 \\
c_1 & 1 & -1 & -1 & -1 & 0 & 0 \\
c_2 & 0 & -1 & 0 & 0 & -1 & 1 \\
c_3 & 0 & -1 & 0 & -1 & 0 & -1
\end{array}
\end{pmatrix}
\]

\((v_1) \land (v_0 \lor \overline{v_1} \lor \overline{v_2} \lor v_3) \land (\overline{v_1} \lor \overline{v_4} \lor v_5) \land (\overline{v_1} \lor \overline{v_3} \lor v_5)\)

Decomposition:

\[
\begin{pmatrix}
v_3 & v_1 & v_5 & v_4 & v_0 & v_2 \\
\begin{array}{cccccc}
c_3 & -1 & -1 & 1 & | & 0 & 0 & 0 \\
c_0 & 0 & 1 & 0 & | & 0 & 0 & 0 \\
\hline
\end{array}
\end{pmatrix}
\]

\[
\begin{array}{cccccc}
c_1 & -1 & -1 & 0 & | & 0 & 1 & -1 \\
c_2 & 0 & -1 & -1 & | & 1 & 0 & 0
\end{array}
\]
Representations

Binary Decision Diagrams:

\{\{v_0, v_1, v_4\}, \{v_0, \overline{v}_1, \overline{v}_2, v_3, \overline{v}_4\}, \{v_0, \overline{v}_1, v_2, v_3\}, \{v_0, v_2, v_3, v_4\}, \{v_1, \overline{v}_2, v_3\}\}
Is point \( w \) stuck at 0?

- State of point \( w \) is observable only at point \( Y \).
- Choose input to “excite” \( w \) - point \( w \) set to 1 - \( \psi_1 = (A \land B) \).
- Same input should “sensitize” a path from \( w \) to \( Y \) - point \( v \) set to 0 - \( \psi_2 = (\overline{C} \lor (A \land B) \lor (A \land \overline{B})) \).
- Test vector is \( \psi_1 \land \psi_2 = (A \land B) \).
A reasonable functional specification of any 1-bit adder:

\[(X \iff (A \land \overline{B} \land \overline{C}) \lor (\overline{A} \land B \land \overline{C}) \lor (\overline{A} \land \overline{B} \land C) \lor (A \land B \land C)) \land \]
\[(Y \iff (A \land B) \lor (A \land C) \lor (B \land C)).\]
A reasonable functional specification of any 1-bit adder:

\[
X \iff (A \land \overline{B} \land \overline{C}) \lor (\overline{A} \land B \land \overline{C}) \lor (\overline{A} \land \overline{B} \land C) \lor (A \land B \land C) \land \\
Y \iff (A \land B) \lor (A \land C) \lor (B \land C)
\]

A proposed implementation of a 1-bit adder:

\[
\begin{align*}
(u & \iff (A \land \overline{B}) \lor (\overline{A} \land B)) \land \\
(v & \iff u \land \overline{C}) \land \\
(w & \iff A \land B) \land \\
(X & \iff (u \land \overline{C}) \lor (\overline{u} \land C)) \land \\
(Y & \iff w \lor v).
\end{align*}
\]

Call these formulas $\psi_S(A, B, C, X, Y)$ and $\psi_I(A, B, C, X, Y, u, v, w)$. 
Functional Verification of Hardware Design

A reasonable functional specification of any 1-bit adder:

\[(X \Leftrightarrow (A \land \overline{B} \land \overline{C}) \lor (\overline{A} \land B \land \overline{C}) \lor (\overline{A} \land \overline{B} \land C) \lor (A \land B \land C')) \land
(Y \Leftrightarrow (A \land B) \lor (A \land C') \lor (B \land C')).\]

A proposed implementation of a 1-bit adder:

\[
egin{align*}
(u & \Leftrightarrow (A \land \overline{B}) \lor (\overline{A} \land B)) \land \\
(v & \Leftrightarrow u \land C') \land \\
w & \Leftrightarrow A \land B) \land \\
(X & \Leftrightarrow (u \land \overline{C}) \lor (\overline{u} \land C')) \land \\
(Y & \Leftrightarrow w \lor v).
\end{align*}
\]

Call these formulas \(\psi_S(A, B, C, X, Y)\) and \(\psi_I(A, B, C, X, Y, u, v, w)\).

The theorem we are trying to prove is:

\[
\psi_S(A, B, C, X, Y) \Leftrightarrow \exists u, v, w : \psi_I(A, B, C, X, Y, u, v, w).
\]

That is, for each one of 32 patterns of binary values given to inputs \(A\), \(B\), and \(C\) and outputs \(X\) and \(Y\), there exists a set of values for internal points \(u\), \(v\), and \(w\) such that the value of \(\psi_S\) is the same as the value of \(\psi_I\). In other words, show

\[
\psi_S(\mathbf{u}) \Leftrightarrow \psi_I(\mathbf{v}, \mathbf{u}) \text{ is satisfiable}
\]

is unsatisfiable where \(\mathbf{u}\) is a vector of binary values for input variables and \(\mathbf{v}\) is a vector of values of internal points.
### Functional Verification - Temporal Logic

<table>
<thead>
<tr>
<th>Op. name</th>
<th>$(S, s_i) \models$</th>
<th>if and only if</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p$ (a variable)</td>
<td>$s_i(p) = 1$.</td>
<td></td>
</tr>
<tr>
<td><strong>not</strong></td>
<td>$\bar{\psi}_1$</td>
<td>$(S, s_i) \not\models \psi_1$</td>
</tr>
<tr>
<td><strong>and</strong></td>
<td>$\psi_1 \land \psi_2$</td>
<td>$(S, s_i) \models \psi_1$ and $(S, s_i) \models \psi_2$</td>
</tr>
<tr>
<td><strong>or</strong></td>
<td>$\psi_1 \lor \psi_2$</td>
<td>$(S, s_i) \models \psi_1$ or $(S, s_i) \models \psi_2$</td>
</tr>
<tr>
<td><strong>henceforth</strong></td>
<td>$\Box \psi_1$</td>
<td>$(S, s_j) \models \psi_1$ for all states $s_j, j \geq i$.</td>
</tr>
<tr>
<td><strong>eventually</strong></td>
<td>$\Diamond \psi_1$</td>
<td>$(S, s_j) \models \psi_1$ for some state $s_j, j \geq i$.</td>
</tr>
<tr>
<td><strong>next</strong></td>
<td>$\circ \psi_1$</td>
<td>$(S, s_{i+1}) \models \psi_1$.</td>
</tr>
<tr>
<td><strong>until</strong></td>
<td>$\psi_1 \mathcal{U} \psi_2$</td>
<td>For some $j \geq i$, $(S, s_i), (S, s_{i+1}), \ldots, (S, s_{j-1}) \models \psi_1$, and $(S, s_j) \models \psi_2$.</td>
</tr>
</tbody>
</table>

- $S = \{s_0, s_1, s_2, \ldots\}$ is a sequence of legal states and transitions.
- If $\psi$ evaluates to 1 for state $s_i \in S$ we say $(S, s_i) \models \psi$.
- We say $S \models \psi$ if and only if $(S, s_0) \models \psi$.
- Two LTTL formulas $\psi_1$ and $\psi_2$ are *equivalent* if, for all sequences $S$, $S \models \psi_1$ if and only if $S \models \psi_2$. 
Functional Verification - Set/Reset Latch

<table>
<thead>
<tr>
<th>Expressions</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\Box(s \land r))</td>
<td>No two inputs have value 1 simultaneously.</td>
</tr>
<tr>
<td>(\Box((s \land \overline{q}) \rightarrow ((s \mathcal{U} q) \lor \Box s)))</td>
<td>Input (s) cannot change if (s) is 1 and (q) is 0.</td>
</tr>
<tr>
<td>(\Box((r \land q) \rightarrow ((r \mathcal{U} \overline{q}) \lor \Box r)))</td>
<td>Input (r) cannot change if (r) is 1 and (q) is 1.</td>
</tr>
<tr>
<td>(\Box(s \rightarrow \Diamond q))</td>
<td>If (s) is 1, (q) will eventually be 1.</td>
</tr>
<tr>
<td>(\Box(r \rightarrow \Diamond \overline{q}))</td>
<td>If (r) is 1, (q) will eventually be 0.</td>
</tr>
<tr>
<td>(\Box((\overline{q} \rightarrow ((\overline{q} \mathcal{U} s) \lor \Box q))))</td>
<td>Output (q) rises to 1 only if (s) becomes 1.</td>
</tr>
<tr>
<td>(\Box((q \rightarrow ((q \mathcal{U} r) \lor \Box q))))</td>
<td>Output (q) drops to 0 only if (r) becomes 1.</td>
</tr>
</tbody>
</table>
Variables: at time $i$,

$v_1^i = \text{value of bit 1},$

$v_2^i = \text{value of bit 2}$

Does the 2-bit counter above reach state 11 in exactly 3 time steps?
BMC, Counter: The Formula

Force the property to hold:

\[(v_0^0 \land v_0^0) \land (v_1^1 \land v_1^1) \land (v_2^2 \land v_2^2) \land (v_3^3 \land v_3^3)\]
BMC, Counter: The Formula

Force the property to hold:

\[(v_1^0 \wedge v_2^0) \wedge (v_1^1 \wedge v_2^1) \wedge (v_1^2 \wedge v_2^2) \wedge (v_1^3 \wedge v_2^3)\]

Express the starting state:

\[\overline{v_1^0 \wedge v_2^0}\]
BMC, Counter: The Formula

Force the property to hold:

\[(v_1^0 \land v_2^0) \land (v_1^1 \land v_2^1) \land (v_1^2 \land v_2^2) \land (v_1^3 \land v_2^3)\]

Express the starting state:

\[v_1^0 \land v_2^0\]

Force legal transitions (repeat transition relation):

\[(v_1^1 \equiv \overline{v_2^1}) \land (v_1^1 \equiv v_1^0 \oplus v_2^0) \land (v_2^1 \equiv \overline{v_1^1}) \land (v_1^2 \equiv \overline{v_2^2}) \land (v_2^2 \equiv v_1^1 \oplus v_2^1)\]

\[(v_1^2 \equiv v_1^1 \oplus v_2^1) \land (v_2^3 \equiv \overline{v_1^2}) \land (v_1^3 \equiv v_1^2 \oplus v_2^2)\]
BMC, Counter: The Formula

Force the property to hold:

\[(v_1^0 \land v_2^0) \land (v_1^1 \land v_2^1) \land (v_1^2 \land v_2^2) \land (v_1^3 \land v_2^3)\]

Express the starting state:

\[\overline{v_1^0} \land \overline{v_2^0}\]

Force legal transitions (repeat transition relation):

\[(v_2^1 \equiv \overline{v_2^0}) \land (v_1^1 \equiv v_1^0 \oplus v_2^0) \land (v_2^2 \equiv \overline{v_2^1}) \land (v_2^2 \equiv v_1^1 \oplus v_2^1) \land (v_3^1 \equiv v_1^1 \oplus v_2^1) \land (v_3^2 \equiv v_1^2 \oplus v_2^2) \land (v_3^3 \equiv v_1^3 \oplus v_2^3)\]

Satisfied only by:

\[v_1^0 = 0, v_2^0 = 0, v_1^1 = 0, v_2^1 = 1, v_1^2 = 1, v_2^2 = 0, v_1^3 = 1, v_2^3 = 1\]
BMC, Counter: The Formula

Three repetitions of a function

\[
f_i = \overline{v_{i+1}} \land v_{i+3} \land (v_i \equiv v_{i+2})
\]
Equivalence Checking

“and” gate

“or” gate

“xor” gate

“adder” circuit of VLSI testing slide

Examples of AIGs. Edges with white circles are negated, and are not negated otherwise.
The beginning of the equivalency check - one circuit has been transformed to an AIG.
Node $a$ of the circuit has been merged with vertex $b$ of the AIG.
Equivalence Checking

The completed AIG.
Horn Formulas - Unit Resolution

\{\{v_0, \overline{v_1}, \overline{v_2}\}, \{\overline{v_0}\}, \{v_1, \overline{v_3}, \overline{v_5}\}, \{v_2\}, \{\overline{v_2}, v_4\}\}

Horn Solver (ψ)

/* Input ψ is a Horn formula */
/* Output is ‘unsatisfiable” or a model for ψ */
/* Locals: set of variables M */

Set \[ M \leftarrow \emptyset. \]
Repeat the following until no positive literal unit clauses are in ψ:

Choose \[ l \] from a positive literal unit clause \[ \{l\} \in \psi. \]
Set \[ M \leftarrow M \cup \{l\}. \]
Set \[ ψ \leftarrow \{c - \{l\} : c \in ψ, l \notin c\}. \]
If \[ \emptyset \in ψ, \] Output ‘unsatisfiable.”
Output \[ M. \]

Unique minimum satisfying assignment w.r.t. 1

All assignments satisfying above include \[ v_2 = v_4 = 1 \]
2-SAT Formulas - Implication Graph

\{ \{v_0, v_2\}, \{\overline{v_1}, \overline{v_2}\}, \{v_1, v_3\}, \{v_1, \overline{v_3}\}, \{\overline{v_1}, \overline{v_3}\}, \{\overline{v_0}, v_3\}\}
2-SAT Solver (ψ)
/* Input: set of sets 2-CNF formula ψ */
/* Output: “unsatisfiable” or a model for ψ */
/* Locals: variable s, set of variables M, M', set of sets formula φ */

Set φ ← ψ; Set s ← 1; Set M ← ∅; Set M' ← ∅; Set φ' ← ∅.

Repeat the following until some statement outputs a value:

Set ⟨φ, M’⟩ ← Unit Resolution (φ).

If ∅ ∈ φ then do the following:

If s has value 1 or φ' = ∅ then Output “unsatisfiable”.
Set s ← 1
Set M' ← ∅; φ ← φ'; l ← l'.
If l is a negative literal, Set M' ← ¬l.
Set φ ← {c − {l} : c ∈ φ, l /∈ c}.

Otherwise, if φ ≠ ∅ then do the following:
Set s ← 0.
Choose a literal l arbitrarily from a clause of φ.
Set M ← M ∪ M'.
Set M' ← ∅.
Set φ' ← φ.
Set l' ← l.
If l is a positive literal, Set M' ← {l}.
Set φ ← {c − {l} : c ∈ φ, l /∈ c}.

Otherwise, Output M ∪ M'.

☐
q-Horn Formulas - Monotone Decomposition

\[
\begin{pmatrix}
  v_0 & \ldots & v_y & v_{y+1} & \ldots & v_n \\
  c_0 & -1 & -1 & 1 & 0 & 0 & 0 \\
  \ldots & \text{Horn} & & \ldots & 0 & \ldots \\
  c_x & 0 & 1 & 0 & 0 & 0 \\
  c_{x+1} & -1 & -1 & 0 & 0 & 1 & -1 \\
  \ldots & \text{Non-Positive} & \text{2-SAT} \\
  c_m & 0 & -1 & -1 & 1 & 0 & 0
\end{pmatrix}
\]

q-Horn Solver \((\psi)\)

/* Input \(\psi\) is a q-Horn formula */
/* Output is “unsatisfiable” or a model for \(\psi\) */
/* Locals: set of variables \(M_1, M_2\) */

Set \(M \leftarrow \emptyset\).
Find a Monotone Decomposition of \(\psi\).
For the Horn part, find the unique minimum satisfying assignment \(M_1\).
If none exists, output “unsatisfiable.”
Remove all rows satisfied by \(M_1\).
For the 2-SAT part, find a satisfying assignment \(M_2\).
If none exists, output “unsatisfiable.”
Output \(M_1 \cup M_2\).
Linear Autark Formulas

*autark assignment:*

An assignment $t$ to a set of variables such that all clauses containing one or more of those variables are satisfied by $t$. 
Linear Autark Formulas

autark assignment:

An assignment $t$ to a set of variables such that all clauses containing one or more of those variables are satisfied by $t$.

example:

$$\{\{v_1, \overline{v}_2, v_3\}, \{\overline{v}_3, v_4, \overline{v}_5\}, \{\overline{v}_1, \overline{v}_2, v_5\}, \{v_1, v_5, v_6\}, \ldots\}$$

$v_1 = 1, v_2 = 0$ is an autark assignment assuming these so do appear elsewhere in the formula.
Linear Autark Formulas

**autark assignment:**

An assignment $t$ to a set of variables such that all clauses containing one or more of those variables are satisfied by $t$.

**example:**

\[
\{\{ v_1, \overline{v_2}, v_3 \}, \{ \overline{v_3}, v_4, \overline{v_5} \}, \{ \overline{v_1}, \overline{v_2}, v_5 \}, \{ v_1, v_5, v_6 \}, \ldots \}\]

$v_1 = 1, v_2 = 0$ is an autark assignment assuming these so do appear elsewhere in the formula.

**linear system:**

\[
\mathcal{M}_\psi \alpha \geq 0, \quad (1) \\
\alpha \neq 0.
\]

**where:**

- $\mathcal{M}_\psi$ is a $(0, \pm 1)$ representation of CNF formula $\psi$.
- $\alpha$ is an $n$ dimensional real vector with components $\alpha_1, \alpha_2, \ldots, \alpha_n$.

A solution to (1) implies an autark assignment for $\psi$:

- if $\alpha_i < 0$ then assign $v_i = 0$,
- if $\alpha_i > 0$ then assign $v_i = 1$,
- if $\alpha_i = 0$ then keep $v_i$ unassigned.
Minimally Unsatisfiable Formulas

Minimally Unsatisfiable Formula (CNF):

- Remove any clause to get a satisfiable formula.

Properties:

- If $\psi$ is minimally unsatisfiable with $n + 1$ clauses, then the number of variables in $\psi$ must be at most $n$.

- Every variable of a minimally unsatisfiable formula occurs positively and negatively in the formula.

- If $\psi$ is minimally unsatisfiable with $n$ variables and $n + 1$ clauses, then there exists a variable $v$ such that the literal $v$ occurs exactly one time in $\psi$ and the literal $\overline{v}$ occurs exactly one time in $\psi$.

- If $\psi$ is minimally unsatisfiable with $n$ variables and $n + 1$ clauses, then there exists a variable $v$ and a partition of the clauses of $\psi$ into two disjoint sets $\psi_v$ and $\psi_{\overline{v}}$ such that literal $v$ only occurs in clauses of $\psi_v$, literal $\overline{v}$ only occurs in clauses of $\psi_{\overline{v}}$ and no variable other than $v$ that is in $\psi_v$ is also in $\psi_{\overline{v}}$ and no variable other than $v$ that is in $\psi_{\overline{v}}$ is also in $\psi_v$. 

```
\begin{tikzpicture}
  \node (v0) at (0,0) [circle,fill,inner sep=2pt] {}; 
  \node (v1) at (1,0) [circle,fill,inner sep=2pt] {}; 
  \node (v2) at (2,0) [circle,fill,inner sep=2pt] {}; 
  \node (v0_bar) at (0,-1) [circle,fill,inner sep=2pt] {}; 
  \node (v1_bar) at (1,-1) [circle,fill,inner sep=2pt] {}; 
  \node (v2_bar) at (2,-1) [circle,fill,inner sep=2pt] {}; 
  \draw (v0) -- (v1); 
  \draw (v1) -- (v2); 
  \draw (v0) -- (v0_bar); 
  \draw (v1) -- (v1_bar); 
  \draw (v2) -- (v2_bar); 
  \foreach \i in {0,1,2} { 
    \node at (\i,0.5) {$v_\i$}; 
    \node at (\i,-0.5) {$\overline{v_\i}$}; 
  } 
  \foreach \i in {0,1,2} { 
    \node at (\i,-1.5) {$\{v_\i, \overline{v_\i}\}$}; 
  } 
  \node at (1.5,-0.5) {$\{v_0\}$}; 
\end{tikzpicture}
```
Properties of Formulas

A chorded loop - refutes q-Hornness
Resolution

If these two clauses are in the data base:
\[(v \lor l_a \lor \ldots \lor l_x), (\overline{v} \lor l_b \lor \ldots \lor l_y)\]

then add this clause (resolvent) to the data base:
\[(l_a \lor \ldots \lor l_x \lor l_b \lor \ldots \lor l_y)\]
Resolution

If these two clauses are in the data base:

\[(v \lor l_a \lor \ldots \lor l_x), (\overline{v} \lor l_b \lor \ldots \lor l_y)\]

then add this clause (resolvent) to the data base:

\[(l_a \lor \ldots \lor l_x \lor l_b \lor \ldots \lor l_y)\]

Davis Putnam Resolution

Add all possible resolvents for variable \(v\) then remove all clauses containing \(v\) or \(\overline{v}\)
Resolution Can Be Bad

Prove that it is impossible to assign \( n+1 \) pigeons to \( n \) holes without at least one hole containing two pigeons
Resolution Can Be Bad

Prove that it is impossible to assign $n+1$ pigeons to $n$ holes without at least one hole containing two pigeons

<table>
<thead>
<tr>
<th>Variables</th>
<th>Subscript Range</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_{i,k}$</td>
<td>$1 \leq i \leq n$ $1 \leq k \leq n+1$</td>
<td>$v_{i,k} = 1$ iff $k^{th}$ pigeon in hole $i$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Subscript Range</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(v_{1,k} \lor \ldots \lor v_{n,k})$</td>
<td>$1 \leq k \leq n+1$</td>
<td>Every pigeon in at least one hole</td>
</tr>
<tr>
<td>$(\overline{v_{i,l}} \lor \overline{v_{i,k}})$</td>
<td>$1 \leq l &lt; k \leq n+1$ $1 \leq i \leq n$</td>
<td>Each hole has no more than one pigeon</td>
</tr>
</tbody>
</table>

Every resolution proof requires generating exponentially many resolvents
Extended Resolution

Just add this:

\[ w \equiv f(x, y, \ldots, z) \]

where \( w \) is a variable not already in the formula and \( f \) is any Boolean function of variables that are in the formula.
Extended Resolution

Just add this:

$$w \equiv f(x, y, \ldots, z)$$

where $w$ is a variable not already in the formula and $f$ is any Boolean function of variables that are in the formula.

Example:

$$(w \lor x) \land (w \lor y) \land (\overline{w} \lor \overline{x} \lor \overline{y})$$

where $w$ is the variable that is not in the formula. This is equivalent to:

$$w \equiv (\overline{x} \lor \overline{y})$$

which means either $x$ and $y$ both have value 1 (then $w = 0$) or at least one of $x$ or $y$ has value 0 (then $w = 1$).
Extended Resolution

Just add this:

\[ w \equiv f(x, y, \ldots, z) \]

where \( w \) is a variable not already in the formula and \( f \) is any Boolean function of variables that are in the formula.

Example:

\[(w \lor x) \land (w \lor y) \land (\overline{w} \lor \overline{x} \lor \overline{y})\]

where \( w \) is the variable that is not in the formula. This is equivalent to:

\[ w \equiv (\overline{x} \lor \overline{y}) \]

which means either \( x \) and \( y \) both have value 1 (then \( w = 0 \)) or at least one of \( x \) or \( y \) has value 0 (then \( w = 1 \)).

Example, Pigeon Hole Formulas:

\[ w^{n-1}_{i,j} \equiv v_{i,j} \lor (v_{n,j} \land v_{i,n+1}), \quad 1 \leq i \leq n - 1, 1 \leq j \leq n. \]

All the \( w^{n-1}_{i,j} \) act like the \( v_{i,j} \) except that the maximum of \( i \) and \( j \) are reduced by 1. That is, if a unique mapping is possible and the modified formula is satisfied, one of \( w^{n-1}_{i,1}, w^{n-1}_{i,2}, \ldots, w^{n-1}_{i,n}, 1 \leq i \leq n - 1 \), will have value 1 and all \( z^{n-1}_{i,j} \lor z^{n-1}_{i,k} \) will also have value 1.
Gröbner Bases

Example:

\((v_1 \lor v_2 \lor v_3)\)

is represented by the equation

\[ v_1(1 + v_2)(1 + v_3) + v_2(1 + v_3) + v_3 + 1 = 0 \]

which may be rewritten

\[ v_1 v_2 v_3 + v_1 v_2 + v_1 v_3 + v_2 v_3 + v_1 + v_2 + v_3 + 1 = 0 \]
Gröbner Bases

Example:

\((v_1 \lor v_2 \lor v_3)\)

is represented by the equation

\[ v_1(1 + v_2)(1 + v_3) + v_2(1 + v_3) + v_3 + 1 = 0 \]

which may be rewritten

\[ v_1v_2v_3 + v_1v_2 + v_1v_3 + v_2v_3 + v_1 + v_2 + v_3 + 1 = 0 \]

Example:

\(v_1 \oplus v_2 \oplus v_3 \oplus v_4\)

is represented by

\[ v_1 + v_2 + v_3 + v_4 + 1 = 0. \]
Example:

\[(v_1 \lor \neg v_2) \land (v_2 \lor \neg v_3) \land (v_3 \lor \neg v_1)\]

The equations corresponding to the above are shown below as equations (1), (2), and (3). All equations below the line are derived as stated on the right.

\[
\begin{align*}
v_1v_2 & +v_2 & = 0 & (1) \\
v_2v_3 & +v_3 & = 0 & (2) \\
v_1v_3 & +v_1 & = 0 & (3)
\end{align*}
\]

\[
\begin{align*}
v_1v_2v_3 & +v_2v_3 & = 0 & (4) \iff v_3 \cdot (1) \\
v_1v_2v_3 & +v_3 & = 0 & (5) \iff (4) + (2) \\
v_1v_2v_3 & +v_1v_3 & = 0 & (6) \iff v_1 \cdot (2) \\
v_1v_2v_3 & +v_1 & = 0 & (7) \iff (6) + (3) \\
v_1v_2v_3 + v_1v_2 & = 0 & (8) \iff v_2 \cdot (3) \\
v_1v_2v_3 & +v_2 & = 0 & (9) \iff (8) + (1) \\
v_1 + v_2 & = 0 & (10) \iff (9) + (7) \\
v_1 + v_3 & = 0 & (11) \iff (5) + (7)
\end{align*}
\]

The solution is given by the bottom two equations which state that \(v_1 = v_2 = v_3\).
Stochastic Local Search

Example:

\[
\{\{v_1, \overline{v}_2, v_3\}, \{\overline{v}_3, v_4, \overline{v}_5\}, \{\overline{v}_1, \overline{v}_2, v_5\}, \{v_1, v_5, v_6\}, \ldots\}
\]
Stochastic Local Search

Example:

\[ \{ \{ v_1, \overline{v_2}, v_3 \}, \{ \overline{v_3}, v_4, \overline{v_5} \}, \{ \overline{v_1}, \overline{v_2}, v_5 \}, \{ v_1, v_5, v_6 \}, \ldots \} \]

Start with some assignment:

\[ v_1 = 0, v_2 = 1, v_3 = 0, v_4 = 1, v_5 = 0, v_6 = 0, \ldots \]

\[ \{ \{ v_1, \overline{v_2}, v_3 \}, \{ \overline{v_3}, v_4, \overline{v_5} \}, \{ \overline{v_1}, \overline{v_2}, v_5 \}, \{ v_1, v_5, v_6 \}, \ldots \} \]
Stochastic Local Search

Example:

\[
\{\{v_1, \overline{v}_2, v_3\}, \{\overline{v}_3, v_4, \overline{v}_5\}, \{\overline{v}_1, \overline{v}_2, v_5\}, \{v_1, v_5, v_6\}, \ldots\}
\]

Start with some assignment:

\[
v_1 = 0, v_2 = 1, v_3 = 0, v_4 = 1, v_5 = 0, v_6 = 0, \ldots
\]

\[
\{\{v_1, \overline{v}_2, v_3\}, \{\overline{v}_3, v_4, \overline{v}_5\}, \{\overline{v}_1, \overline{v}_2, v_5\}, \{v_1, v_5, v_6\}, \ldots\}
\]

Reverse a variable to reduce number of red clauses:

\[
v_1 = 1, v_2 = 1, v_3 = 0, v_4 = 1, v_5 = 0, v_6 = 0, \ldots
\]

\[
\{\{v_1, \overline{v}_2, v_3\}, \{\overline{v}_3, v_4, \overline{v}_5\}, \{\overline{v}_1, \overline{v}_2, v_5\}, \{v_1, v_5, v_6\}, \ldots\}
\]
Stochastic Local Search

Example:

\[
\{ \{v_1, \overline{v}_2, v_3\}, \{v_3, v_4, \overline{v}_5\}, \{v_1, \overline{v}_2, v_5\}, \{v_1, v_5, v_6\}, \ldots \}\]

Start with some assignment:

\[v_1 = 0, v_2 = 1, v_3 = 0, v_4 = 1, v_5 = 0, v_6 = 0, \ldots\]

\[
\{ \{v_1, \overline{v}_2, v_3\}, \{v_3, v_4, \overline{v}_5\}, \{v_1, \overline{v}_2, v_5\}, \{v_1, v_5, v_6\}, \ldots \}\]

Reverse a variable to reduce number of red clauses:

\[v_1 = 1, v_2 = 1, v_3 = 0, v_4 = 1, v_5 = 0, v_6 = 0, \ldots\]

\[
\{ \{v_1, \overline{v}_2, v_3\}, \{v_3, v_4, \overline{v}_5\}, \{v_1, \overline{v}_2, v_5\}, \{v_1, v_5, v_6\}, \ldots \}\]

Reverse a variable to reduce number of red clauses:

\[v_1 = 1, v_2 = 1, v_3 = 0, v_4 = 1, v_5 = 1, v_6 = 0, \ldots\]

\[
\{ \{v_1, \overline{v}_2, v_3\}, \{v_3, v_4, \overline{v}_5\}, \{v_1, \overline{v}_2, v_5\}, \{v_1, v_5, v_6\}, \ldots \}\]
Unsafe Constraints

A Search Space

Learned clauses
What Makes a Problem Hard?

Useful clauses are not learned early enough:
What Makes a Problem Hard?

Is any particular structure bad?
How Can We Make the Problem Easier?

Install the inferred constraints early

Install safe, uninferred constraints that are obtained from an analysis of the problem
  - for example, take advantage of problem symmetry

Install unsafe, uninferred constraints that are obtained from an analysis of solutions to smaller problems in the family
  - run the search to some depth past the hump
  - retract the unsafe constraints and search deeper