Goal: l1-b; Show KB ∪ {¬ l1-b} is unsatisfiable

In computer:
{ light1_broken ← sw_up ∧ power ∧ unlit_light1. 
  sw_up. 
  power ← lit_light2. 
  unlit_light1. 
  lit_light2. }

In user's mind:
- light1_broken: light #1 is broken
- sw_up: switch is up
- power: there is power in the building
- unlit_light1: light #1 isn't lit
- lit_light2: light #2 is lit

Conclusion: light1_broken

- The computer doesn't know the meaning of the symbols
- The user can interpret the symbol using their meaning
CNF and DNF

How to convert a propositional formula to CNF and DNF.

Two main techniques.

One involves using the substitution theorem and a number of equivalences, including:

1. The definition of implication $P \rightarrow Q \equiv \neg P \lor Q$

2. The definition of equivalence $P \leftrightarrow Q \equiv (P \rightarrow Q) \land (Q \rightarrow P)$
3. Idempotency \((F \land F) \equiv F, \ (F \lor F) \equiv F\)

4. Commutativity \((F \land G) \equiv (G \land F), \ (F \lor G) \equiv (G \lor F)\)

5. Associativity \( ((F \land G) \land H) \equiv (F \land (G \land H)) \equiv (F \land G \land H) \)
\( ((F \lor G) \lor H) \equiv (F \lor (G \lor H)) \equiv (F \lor G \lor H) \)

6. Absorption \((F \land (F \land G)) \equiv F, \ (F \lor (F \land G)) \equiv F\)

7. Distributivity \((F \land (G \lor H)) \equiv ((F \land G) \lor (F \land H))\)
\( (F \lor (G \land H)) \equiv ((F \lor G) \land (F \lor H))\)
8. Double Negation \( \neg \neg F = F \)

9. De Morgan’s Laws
\( \neg (F \land G) = (\neg F \lor \neg G) \)
\( \neg (F \lor G) = (\neg F \land \neg G) \)

10. Tautology Laws
\( (F \lor G) \in F \) if \( F \) is a tautology
\( (F \land G) \in G \) if \( F \) is a tautology

11. Unsatisfiability Laws
\( (F \lor G) \in G \) if \( F \) is unsatisfiable
\( (F \land G) \in F \) if \( F \) is unsatisfiable
To convert $F$ to CNF,

1. Push negation inwards, only

$$\neg F \equiv F$$

$$(\neg F \lor (F \land \neg H))$$

$$(\neg (F \lor (F \land \neg H)))$$

until no subformulas of the form on the LHS of these rules occur.
2. Substitute in F each occurrence of a subformula of the form

\[(F \lor (\neg HH)) \Rightarrow ((F \lor G) \land (F \lor H))\]

\[(((F \lor G) \lor H)) \Rightarrow ((F \lor H) \land (G \lor H))\]

until no such subformulas occur.

The resulting formula is a CNF (except for possible occurrence of tautologies).

(End of method)
Another technique is based on truth tables. For formula $F$, to convert to DNF, take each row of its truth-table and build a conjunction with a positive literal of the corresponding variable if assigned $1 (\top)$, and a negative literal of the corresponding variable if assigned $0 (\bot)$. To convert $F$ to CNF, interchange the roles of $0$ and $1$. 
and I, and construct a disjunction for each row of the truth table.

Example. Convert to DNF and CNF the following formula:

\[ F = ((\neg A \rightarrow B) \land ((A \land \neg C) \leftrightarrow B)) \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>((\neg A \rightarrow B))</th>
<th>((A \land \neg C))</th>
<th>((A \land \neg C) \leftrightarrow B)</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</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>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
The equivalent DNF formula is:

\[(A \land \neg B \land C) \lor (A \land B \land \neg C)\]

The equivalent CNF formula is:

\[(A \lor B \lor C) \land (A \lor B \lor \neg C) \land (A \lor \neg B \lor C) \land (A \lor \neg B \lor \neg C) \land (\neg A \lor B \lor C) \land (\neg A \lor B \lor \neg C) \land (\neg A \lor \neg B \lor C) \land (\neg A \lor \neg B \lor \neg C)\]

which is very opaque.

But this is also an equivalent CNF formula:

\[A \land (B \lor \neg C) = A \land (B \lor C) \land (\neg B \lor \neg C)\]
Let's try the other method (for CNF):

\[ F = (\neg A \rightarrow B) \land (A \land \neg C) \rightarrow B \] 

\[ \equiv (\neg A \lor B) \land (\neg (A \land \neg C) \lor B) \land (B \lor (A \land \neg C)) \] 

\[ \equiv (A \lor B) \land (\neg (A \land \neg C) \lor B) \land (\neg B \lor (A \land \neg C)) \] 

\[ \equiv (A \lor B) \land ((\neg A \lor C) \lor (A \land B)) \land (B \lor (A \land \neg C)) \] 

\[ \equiv (A \lor B) \land (\neg A \lor B \lor C) \land (A \land \neg B \lor C) \]
\[ A \land \neg (A \lor B \lor C) \land \neg (B \lor \neg C) \]

\[ A \land \neg (B \lor C) \land \neg (B \lor \neg C) \]

**Resolution**

The (propositional) resolution rule:

Assume a formula in **clausal form**;

Start with \( F \in \text{CNF} \), so

\[ F = (L_{1,1} \lor \cdots \lor L_{1,n_1}) \land \cdots \land (L_{k,1} \lor \cdots \lor L_{k,n_k}) \]
(F is a conjunction of k clauses, where
the ith clause contains \( n_i \) literals).

Write each clause as a set, so \( F \) is
expressed as a set of clauses:

\[
F = \{ L_1, \ldots, L_{n_1}, \ldots, L_k, L_{n_1}, \ldots, L_{n_k} \}
\]

**Definition.** **Resolvent.**

\( C_i \) and \( C_j \) are clauses. Then \( R \) is a resolvent of
\( C_i \) and \( C_j \) if there is a literal \( L \in C_i \) such
that \( L \in C_i \) and \( R = (C_1 - \{ L \}) \cup (C_2 - \{ L \}) \).

\[ L = \begin{cases} \neg A_i & \text{if } L = A_i \\ A_i & \text{if } L = \neg A_i \end{cases} \]

Graphical notation \( \{ A, \neg c \} \rightarrow \{ A, B, C \} \)

The resolution \( \rightarrow \) \{ A, B \} of \( \{ A, \neg c \} \) and \( \{ A, B, C \} \)

Theorem. Let \( F \) be a CNF formula, represented
as a set of clauses. Let $R$ be a resolvent of two clauses $C_1$ and $C_2$ in $F$. Then, $F$ and $F \cup \{R\}$ are equivalent.

**Definition.** Let $F$ be a set of clauses. Then $\text{Res}(F)$ is defined as:

$$\text{Res}(F) = F \cup \{R | R \text{ is a resolvent of two clauses in } F\}$$

Furthermore, define


\[ \text{Res}(F) = F \]
\[ \text{Res}_{n+1}(F) = \text{Res}_n(\text{Res}_n(F)) \quad \text{for } n \geq 0, \text{ and} \]

finaly let

\[ \text{Res}_\#(F) = \bigcup_{n \geq 0} \text{Res}_n(F). \]

It can be proved that for every finite \( F \), there is a \( k \) s.t.

\[ \text{Res}_k(F) = \text{Res}_\#(F). \]

**Def:** The empty resolvent is the resolvent of \( C_1 = \{ L \} \) and \( C_2 = \{ \emptyset \}. \)
This is also called the empty clause, and is

\[ \{ \} \]

The Resolution Theorem (of propositional logic)

[J.A. Robinson proved this for FO2 around 1960.]

A clause set \( F \) is unsatisfiable iff \( \emptyset \in \text{Res}^k(F) \).

Example (Ex 33 Sch"oning) Verify resolution,

show that \((A \land B \land C) = \emptyset\) is a consequence of
The clause set \( F = \{ \neg A, B, \neg B, C, \neg A, \neg C, \neg A, B, C \} \)

Recall: \( F \models \varphi \) iff \( \neg F \models \neg \varphi \) iff \( F \cup \{ \neg \varphi \} \) is unsatisfiable.

5. Negate \( \varphi \) and show that \( F \cup \{ \neg \varphi \} \) is unsat.

\( \neg \varphi = \{ \neg A, \neg B, \neg C \} \). Show that

\( F \cup \{ \neg A, B, \neg B, C, \neg A, \neg C, \neg A, B, C, \neg A, \neg B, \neg C \} \)

is unsat., i.e., by brute force, show that


\[ \text{Res}^* (\text{FUNC}_G) = \square. \]

\[
\{ A, \neg C \} \quad \{ A, B, C \} \quad \{ \neg A \}, \neg B, \neg C \quad \{ \neg A, B \} \quad \{ \neg B, C \}
\]

\[
\{ A, B \} \quad \{ \neg B, \neg C \}
\]

\[
\{ B \} \quad \{ \neg B \}
\]

\[
\square
\]
Show by resolution that the following formula is unsatisfiable:

\[ F = (P \lor Q) \land (P \lor \neg Q) \land (\neg P \lor Q) \land (\neg P \lor \neg Q) \]

\[
\{ P, Q \} \cdot \{ P, \neg Q \} \cdot \{ \neg P, Q \} \cdot \{ \neg P, \neg Q \}
\]

\[
\{ P \} \quad \{ \neg P \}
\]

Another proof of the same:
\{p, q\} \quad \{p, \neg q\} \quad \{\neg p, q\} \quad \{\neg p, \neg q\}

This edge shows that this is a refutation graph (Schönig).

Some authors (e.g., Lühling) duplicate reused clauses.
Another example. Show that
\[ F = \{\{A, B, \neg C\}, \{\neg A\}, \{A, B, C\}, \{A, \neg B\}\} \]
is not satisfiable.

\[ \{A, B, \neg C\} \quad \{A, B, C\} \quad \{A, \neg B\} \quad \{\neg A\} \]

This is an example of a resolution graph. In this case, the resolution graph is a tree.
Exercise 34 [Schöning]

Using resolution, show that

\[ F = (\neg B \land C \land D) \lor (\neg B \land \neg D) \lor (C \land D) \lor B \] is a tautology.

We show that \( \neg F \) is unsatisfiable.

\[ \neg F = (B \lor C \lor \neg D) \land (B \lor D) \land (\neg (C \land D)) \land \neg B, \text{ in clausal form,} \]

\[ \{ B, C, \neg D \} \quad \{ B, D \} \quad \{ \neg C, \neg D \} \quad \{ \neg B \} \]

\[ \begin{array}{c}
\{ \neg C \} \\
\{ \neg D \} \\
\{ \neg B \} \\
\{ \} \\
\end{array} \]
A formula $F$ in CNF is a Horn formula if every disjunction in $F$ contains at most one positive literal.

We distinguish three kinds of Horn clauses:

1. $\Box$, a fact (one positive literal)
2. $\lor \Box, \ldots \lor \Box$, a rule (one positive literal & at least one negative literal)
3. $\lor \Box, \ldots \lor \Box$ [i.e., $\lor (\Box, \ldots, \Box]$], an integrity constraint (or indefinite (only negative literals) clause)
Horn clauses are often written in implicative form.

(reminder/defn.: \( 1 \) stands for any tautology
\( 0 = n = n \) contradiction.)

1. \( 1 \rightarrow A \) (fact, usually written as \( A \))

2. \( B_1 \land \cdots \land B_k \rightarrow H \) (rule)

3. \( B_1 \land \cdots \land B_k \rightarrow 0 \) (integrity constraint)

There is an efficient (linear-time) algorithm to test satisfiability of Horn formulas. Here it's:
Input: a Horn formula $F$ (i.e., a tautology).

1. Mark every occurrence of an atomic formula $A$ in $F$ if there is a clause $(\neg A)$ in $F$.

2. While there is a clause $C$ in $F$ of form $\text{②}$ or $\text{③}$, where $B_1, \ldots, B_k$ are already marked (and $C$ is not yet marked),
   - if $C$ is of form $\text{②}$,
     then mark every occurrence of $B_k$.
   - else output "unsatisfiable" and halt.

3. Output "satisfiable" and halt.

(The satisfying assignment is given by the marking, in that variable $A_i = 1$ iff $A_i$ has a mark.)
[The above algorithm is sound & complete. No proof given. It provides the
minimal model for $\mathcal{F}$.]  

Exercise 22 [Schöning]  Give an example of

a formula $\varphi$ that does not have an equivalent

 Horn formula.

A $\lor B$, because this formula does not have a single

 minimal model.

Homework (Ex. 21 [Schöning]):  

Apply the marking algorithm to

 $\varphi = (\neg A \lor \neg B \lor \neg D) \land \neg E \land (\neg (\neg A) \land \neg B \lor (\neg \neg D)) \lor$
If the formula is unsatisfiable, please also show that by resolution.