Horn Formulas

- Limited version of propositional logic.
- Boolean variables represent propositions that may be true or false.
- Literals are positive or negative variables.

- A clause of the formula can be one of two types:

  = Implications \( \text{body} \Rightarrow \text{head} \)
    where body is the conjunction of positive literals, and head is a single positive literal.
    Special case when body is empty, we have a fact.

  = Negative clauses
    a disjunction of negative literals.

- The goal of SAT is to find an assignment of values to variables that satisfies the formula if one exists, or to report unsatisfiable if none exists.
Greedy Algorithms

Example (thanks wikipedia)

Variables
\[
\begin{align*}
a & = \text{Fritz croaks} \\
b & = \text{Fritz eats flies} \\
c & = \text{Fritz chirps} \\
d & = \text{Fritz sings} \\
e & = \text{Fritz is a frog} \\
f & = \text{Fritz is a canary} \\
g & = \text{Fritz is green} \\
h & = \text{Fritz is yellow}
\end{align*}
\]

Formula
\[
\begin{align*}
(a \land b) & \Rightarrow e \\
(c \land d) & \Rightarrow f \\
e & \Rightarrow g \\
f & \Rightarrow h \\
\Rightarrow a \\
\Rightarrow b
\end{align*}
\]

SAT with Horn Formulas

Assignment:
\[
\begin{align*}
a & = T \\
b & = F \\
c & = F \\
d & = T \\
e & = T \\
f & = F \\
g & = T \\
h & = F
\end{align*}
\]

Satisfying? 
\[
\begin{align*}
T & \quad T & \quad T \\
T & \quad F & \quad T \\
F & \quad F & \quad T \\
T & \quad F & \quad F \\
T & \quad T & \quad T
\end{align*}
\]
def Greedy Horn SAT(variables, clauses):
    do some greedy work

    if satisfiable:
        return assignments
    else:
        return None
Remember SAT is looking for assignments that allow all clauses to be true.

Clause types:
- Implication: body $\Rightarrow$ head (e.g., $(a \land b) \Rightarrow e$)
- Negative clause: (e.g., $(\neg v \lor \neg e)$)

Think greedy.
What's the greedy step? What initialization is required?
Where's the repetition?
What's the expected runtime?
Greedy Algorithms  SAT with Horn Formulas

Greedy step choices

1. Set all variables to true, slowly assign some false. What are the effects on the algorithm?

2. Set all variables to false, slowly assign some true. What are the effects on the algorithm?

How are implications and negative clauses affected by each of the choices? Which should we use?
Greedy Algorithms

SAT with Horn Formulas

Greedy step: start with all variables True, slowly assign to false.

Implications all start out true.

Negative clauses all start out false.

Greedy step: pick some variable in negative clause to be false.

How do you choose the right one?

Repeat until all negative clauses are true.

Check all implications. If all true, SAT solution found.
If not all true, could there be some other assignment of false that works?

$\Rightarrow O(2^N)$ to check combinations
Greedy Algorithms

Greedy step: start with variables False, slowly assign to true.

Negative clauses start out all true.
Empty body implications start out false.
All other implications start out true.

Greedy step (immediately good for algorithm's goal):
Assign true to all variables in the head of an empty body implication. [If we don't do this the algorithm will fail.]

Repeat: Now we may have non-empty body implications where the body variables are all true and the head variable is false. We can't set the body variables to false, so make the head variable true. Repeat until all implications are true.

We may have broken some negative clauses, so check them.
def GreedyHornSAT(variables, clauses):
    for v in variables:
        v = false

    while some implication clause has true body and false head:
        head = true

    for c in negative clauses:
        if not satisfied c:
            return None

    return variable assignments
Greedy Algorithms      SAT with Horn Clauses

Correctness

If we assign true to a variable, it must be true, because we are following implications.

This is a minimalistic approach to the assignments. There might be other variables which could be assigned true, but they don’t have to be.

If the negative clauses are not satisfied, then there is no way to satisfy the formula.

If they are satisfied, then we have found a solution.
Greedy Algorithms

SAT with Horn Clauses

Runtime

\[ n = \text{number of variables} \]
\[ m = \text{number of literal entries in the formula} \]
\[ m \geq n \]

Initialize: \( O(n) \)

While implications:
- \( O(n) \) repeats,
- \( O(m) \) body variable checks for negative clauses:
- \( O(m) \) checks of variables

\[ \rightarrow O(m) \]
**Greedy Algorithms**  **SAT with Horn Formulas**

**Example:**

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
<th>h</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

**Initialize**

- $(a \land b) \Rightarrow e$
- $(c \land d) \Rightarrow f$
- $e \Rightarrow g$
- $f \Rightarrow h$

**Inference:**

- $e \Rightarrow g$
- $f \Rightarrow h$

**Check negative clauses:**

- $(\neg f \lor \neg e) = (\neg F \lor T) = (T \lor F) = T$
- $(\neg g \lor \neg h) = (\neg T \lor F) = (F \lor T) = T$

Satisfiable!