Horn clauses

- A **literal** is an atomic formula or its negation
- A **clause** is a disjunction of literals
- A **Horn clause** is a clause with exactly one positive literal
- A **Horn formula** is a conjunctive normal form formula whose clauses are all Horn
Example

• Prolog:
  
  ```
  c:- a, b.
  a.
  b.
  ```

• Horn formula:
  
  \[ [c \land \neg a \land \neg b] \lor a \lor b \]
  
  \[ [c, \neg a, \neg b] \quad [a] \quad [b] \]
Example

• Horn formula: \[ [c, \neg a, \neg b] \quad [a] \quad [b] \]

• Let us proof \( c \)
• The goal clause is \( \neg c \)

\[ [c, \neg a, \neg b] \quad [a] \quad [b] \quad [\neg c] \]

• By resolution we obtain the empty clause, and hence proof \( c \)
Resolution

- Resolution is a single inference rule
- It takes two clauses, and produces a new clauses
- The new clauses is implied by the two old clauses
  - The two old clauses need to have complementary literals
  - The new clause contains all the literals of both old clauses except the complementary ones
Resolution example

[c, ¬a, ¬b] [a] [b] [¬c]
Resolution example

\[ [c, \neg a, \neg b] \quad [a] \quad [b] \quad [\neg c] \]
Resolution example

\[ [c, \neg a, \neg b] \quad [a] \quad [b] \quad [\neg c] \]

\[ [c, \neg b] \quad [b] \quad [\neg c] \]
Resolution example

\[
\begin{align*}
[c, \neg a, \neg b] & \quad [a] \quad [b] \quad [\neg c] \\
[c, \neg b] & \quad [b] \quad [\neg c]
\end{align*}
\]
Resolution example

\[ [c, \neg a, \neg b] \quad [a] \quad [b] \quad [\neg c] \]

\[ [c, \neg b] \quad [b] \quad [\neg c] \]

\[ [c] \quad [\neg c] \]
Resolution example

\[
\begin{array}{c}
[c, \neg a, \neg b] \quad [a] \quad [b] \quad [\neg c] \\
[c, \neg b] \quad [b] \quad [\neg c] \\
[c] \quad [\neg c]
\end{array}
\]
Resolution example

\[
\begin{align*}
[c, \neg a, \neg b] & \quad [a] \quad [b] \quad [\neg c] \\
[c, \neg b] & \quad [b] \quad [\neg c] \\
[c] & \quad [\neg c] \\
[] & 
\end{align*}
\]
Why Horn clauses?

- Resolution of two Horn clauses always results in a Horn clause
- Resolution of a goal clause and a definite clause is always a goal clause
- Horn clauses have better computational properties than normal clauses
- Prolog is based on computing with Horn clauses
Alfred Horn

- The name *Horn clause* comes from Alfred Horn, who discovered the significance of such clauses.