Horn Clauses

A *Horn clause* is a clause containing at most one positive literal.

A *definite clause* contains exactly one positive literal.

Examples of a Horn Clause

$$[\neg\textsc{Child}, \neg\textsc{Mail}, \textsc{Boy}]$$

Not a Horn Clause

$$[\textsc{Rain}, \textsc{Sleet}, \textsc{Snow}]$$

Horn Clauses (Cont)

$$p_1 \wedge \ldots \wedge p_n \rightarrow q$$

$$[\neg p_1, \ldots, \neg p_n, q]$$

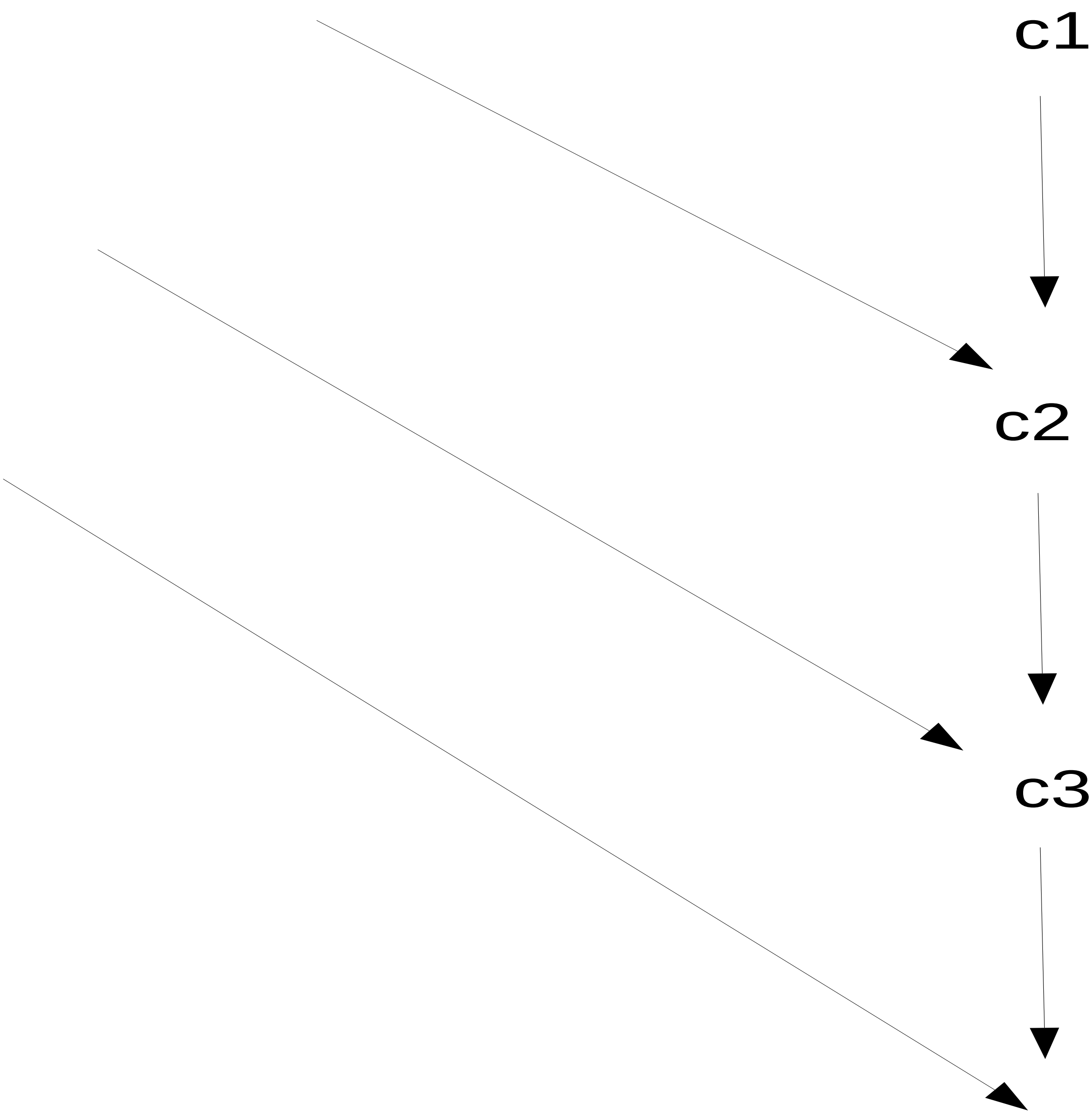
Positive Horn Clause

Negative Horn Clause

Some Observations

There is a derivation of a negative clause (including the empty clause) from a set of Horn clauses S iff there is one where each new clause in the derivation is a negative resolvent of the previous clause in the derivation and some element of S.

SLD Resolution Pattern



SLD Resolution

For any set S of clauses, an SLD derivation of a clause c from S is a sequence of clauses $c_1, c_2, \ldots, c_n$ such that $c_n = c, c_1 \in S$ and c_{i+1} is a resolvent of c_i and some clause of S. We write $S \vdash_{SLD} c$ if there is an SLD derivation of c from S.

$$\text{if } S \vdash_{SLD} [] \text{ then } S \vdash []$$

For Horn clauses:

$$\text{if } S \vdash_{SLD} [] \text{ iff } S \vdash []$$

Example

$$\textsc{Toddler}$$

$$\textsc{Toddler} \rightarrow \textsc{Child}$$

$$(\textsc{Child} \wedge \textsc{Male}) \rightarrow \textsc{Boy}$$

$$\textsc{Infant} \rightarrow \textsc{Child}$$

$$(\textsc{Child} \wedge \textsc{Female}) \rightarrow \textsc{Girl}$$

$$\textsc{Female}$$

Query

$$KB \models \textsc{Girl}$$

Another Example: Lists

Constant NIL Binary Function CONS, e.g., $\textsc{cons}(t_1, t_2)$

Definition of $\textsc{Append}(\textsc{x},\textsc{y},\textsc{z})$

$\textsc{Append}(\textsc{nil}, y, y)$

$\textsc{Append}(x, y, z) \rightarrow$
$\textsc{Append}(\textsc{cons}(w, x), y, \textsc{cons}(w, z))$

We wish to show that this entails the following:

$$\textsc{Append}(\textsc{cons}(\textsc{a}, \textsc{cons}(\textsc{b}, \textsc{nil})), \textsc{cons}(\textsc{c}, \textsc{nil}), \textsc{cons}(\textsc{a}, \textsc{cons}(\textsc{b}, \textsc{cons}(\textsc{c}, \textsc{nil}))))$$

Back-Chaining procedure

Input: a finite list of atomic sentences, $q_1, \ldots, q_n$
Output: `yes` or `no` depending on whether a given KB entails all of the q_i

SOLVE$[q_1, \ldots, q_n] =$
  If $n = 0$ then return `yes`
  For each clause $c \in KB$, do
    If $c = [q_1, \neg p_1, \ldots, \neg p_m]$ and
      $SOLVE[p_1, \ldots, p_m, q_2, \ldots, q_n]$
    then return `yes`
  end for
  Return `no`

Forward-Chaining procedure

Input: a finite list of atomic sentences, $q_1, \ldots, q_n$
Output: `yes` or `no` depending on whether a given KB entails all of the q_i

1. if all of the goals q_i are marked as solved, then return `yes`
2. check if there is a clause $[q_1, \neg p_1, \ldots, \neg p_n]$ in the KB, such that all of its negative atoms $\neg p_1, \ldots, \neg p_n$ are marked as solved, and such that the positive atom p is not marked as solved
3. if there is such a clause, mark p as solved and go to step 1
4. otherwise, return `no`