Horn clauses and negations

An integrity constraint is a clause of the form

\[ \text{false} ::= a_1, \ldots, a_k \]

where each \( a_i \) is an atom and \text{false} is a special atom that is false in all interpretations.
Horn clauses and negations

An integrity constraint is a clause of the form

\[
\text{false} :- a_1, \ldots, a_k
\]

where each \( a_i \) is an atom and false is a special atom that is false in all interpretations.

A Horn clause is either a definite clause or an integrity constraint.
Horn clauses and negations

An integrity constraint is a clause of the form

\[ \text{false} :- a_1, \ldots, a_k \]

where each \( a_i \) is an atom and \( \text{false} \) is a special atom that is false in all interpretations.

A Horn clause is either a definite clause or an integrity constraint.

The negation of a formula \( \alpha \), written \( \neg\alpha \), is a formula that is true in an interpretation \( I \) iff \( \alpha \) is false in \( I \).
Horn clauses and negations

An integrity constraint is a clause of the form

false :- a₁,...,ak

where each aᵢ is an atom and false is a special atom that is false in all interpretations.

A Horn clause is either a definite clause or an integrity constraint.

The negation of a formula α, written ¬α, is a formula that is true in an interpretation I iff α is false in I.

Example

\[ KB = \left\{ \begin{array}{l}
\text{false :- a,b.} \\
a :- c. \\
b :- c.
\end{array} \right\} \]

\[ KB \models \neg c \]
Disjunctions

Every set of definite clauses is satisfiable.
Not so with Horn clauses

\[ KB \models \varphi \iff KB, \neg \varphi \text{ is not satisfiable} \]

Horn-SAT is feasible, whereas 3-SAT is likely not.
Disjunctions

Every set of definite clauses is satisfiable.
Not so with Horn clauses

\[ KB \models \varphi \iff KB, \neg \varphi \text{ is not satisfiable} \]
\[ \iff KB, \neg \varphi \models false. \]
Disjunctions

Every set of definite clauses is satisfiable.
Not so with Horn clauses

$$KB \models \varphi \iff KB, \neg \varphi \text{ is not satisfiable}$$

$$\iff KB, \neg \varphi \models \text{false.}$$

The disjunction $\alpha \lor \beta$ of $\alpha$ and $\beta$ is a formula that is true in an interpretation $I$ iff at least one of $\alpha$ or $\beta$ is true in $I$.

Example

$$KB = \left\{ \begin{array}{l}
\text{false} :- a, b. \\
a :- c. \\
b :- d. \\
\end{array} \right\}$$

$$KB \models \neg c \lor \neg d$$
Disjunctions

Every set of definite clauses is satisfiable. Not so with Horn clauses

\[ KB \models \varphi \iff KB, \neg \varphi \text{ is not satisfiable} \]

\[ \iff KB, \neg \varphi \models \text{false}. \]

The disjunction \( \alpha \lor \beta \) of \( \alpha \) and \( \beta \) is a formula that is true in an interpretation \( I \) iff at least one of \( \alpha \) or \( \beta \) is true in \( I \).

Example

\[
KB = \begin{cases} 
\text{false :- a,b.} \\
a :- c. \\
b :- d.
\end{cases}
\]

\[ KB \models \neg c \lor \neg d \]

Horn-SAT is feasible, whereas 3-SAT is likely not.
Non-monotonicity

Logical consequence is **monotonic**: adding clauses doesn’t invalidate a previous conclusion

\[ KB \models \varphi \text{ implies } KB, \psi \models \varphi. \]
Non-monotonicity

Logical consequence is monotonically: adding clauses doesn’t invalidate a previous conclusion

\[ KB \models \varphi \text{ implies } KB, \psi \models \varphi. \]

Negation-as-failure leads to non-monotonicity: a conclusion can be invalidated by adding more clauses.

\[
\text{empty-course}(X) :- \text{course}(X),\nonumber
\backslash+\text{enrolled}(\_X).\nonumber
\]

Sometimes assume that a database of facts is complete. Any fact not listed is false.
Non-monotonicity

Logical consequence is **monotonic**: adding clauses doesn’t invalidate a previous conclusion

\[ KB \models \varphi \text{ implies } KB, \psi \models \varphi. \]

Negation-as-failure leads to **non-monotonicity**: a conclusion can be invalidated by adding more clauses.

\[
\text{empty-course}(X) :- \text{course}(X), \\backslash +\text{enrolled}(_,X).
\]

Sometimes assume that a database of facts is complete. Any fact not listed is false.

Example: Assume a database of video segments is complete.
Rules

Encode *birds fly*

\[
\text{fly}(X) :- \text{bird}(X).
\]

to allow for exceptions.
Rules and defaults

Encode *birds fly*

\[
\text{fly}(X) :\text{-} \text{bird}(X). \quad \%
\frac{\text{bird}(X)}{\text{fly}(X)}
\]

to allow for exceptions.

*Default rule* (R. Reiter)

\[
\text{bird}(X) : \text{fly}(X) \\
\frac{\text{fly}(X)}{\text{fly}(X)}
\]
Rules and defaults

Encode *birds fly*

\[
\text{fly}(X) :\text{-} \text{bird}(X). \quad \%
\frac{\text{bird}(X)}{\text{fly}(X)}
\]

to allow for exceptions.

*Default rule* (R. Reiter)

\[
\frac{\text{bird}(X): \text{fly}(X)}{\text{fly}(X)}
\]

In general, applied to \( KB \) says:

\[
\text{conclude } c \text{ if } KB \models p \text{ and } j \text{ is } KB\text{-consistent }
\]

\[
KB,j \not\models false
\]
Rules and defaults

Encode *birds fly*

\[
\text{fly}(X) :- \text{bird}(X). \quad \%
\]

\[
\frac{\text{bird}(X)}{\text{fly}(X)}
\]

to allow for exceptions.

*Default rule* (R. Reiter)

\[
\frac{\text{bird}(X) : \text{fly}(X)}{\text{fly}(X)}
\]

In general,

prerequisite \(p\) : justification \(j\)

\[
\text{conclusion } c
\]

applied to \(KB\) says:

conclude \(c\) if \(KB \models p\) and \(j\) is \(KB\)-consistent

\[
KB, j \not\models false
\]

\(j\) is true in some model of \(KB\)
Birds and bees

\[
\begin{align*}
\text{bird}(X) : \text{fly}(X) \\
\text{fly}(X)
\end{align*}
\]

Let \( KB \) be

\[
\begin{align*}
\text{bird}(\text{robin}). \\
\text{bird}(\text{penguin}). \\
\text{false} :- \text{fly}(\text{penguin}). \\
\text{fly}(\text{bee}).
\end{align*}
\]

Conclude:
Birds and bees

\[ (*) \quad \frac{\text{bird}(X) : \text{fly}(X)}{\text{fly}(X)} \]

Let \( KB \) be

- \( \text{bird}(\text{robin}). \)
- \( \text{bird}(\text{penguin}). \)
- \( \text{false} : - \text{fly}(\text{penguin}). \)
- \( \text{fly}(\text{bee}). \)

Conclude:

- \( \text{fly}(\text{robin}) \) by default rule \( (* \)

but not \( \text{fly}(\text{penguin}). \)
Birds and bees

Let $KB$ be

$\text{bird}(\text{robin})$.
$\text{bird}(\text{penguin})$.
false :- $\text{fly}(\text{penguin})$.
$\text{fly}(\text{bee})$.

Conclude:

$\text{fly}(\text{robin})$ by default rule $(\star)$

but not $\text{fly}(\text{penguin})$.

An explanation of $\text{fly}(\text{bee})$ using $(\star)$ is

$\text{bird}(\text{bee})$
Birds and bees

Let $KB$ be

- `bird(robin).`
- `bird(penguin).`
- `false :- fly(penguin).`
- `fly(bee).`

Conclude:

- `fly(robin)` by default rule ($\star$)

but not `fly(penguin)`.

An explanation of `fly(bee)` using ($\star$) is

- `bird(bee)`

which we can block by adding to $KB$ the rule

- `false :- bird(bee).`
Non-determinism

Conflicting defaults

\[
\begin{align*}
quaker(X): & \text{pacifist}(X) \\
pacifist(X) & \\
republican(X): & \text{hawk}(X) \\
\text{hawk}(X) &
\end{align*}
\]
Non-determinism

Conflicting defaults

\[
\begin{align*}
\text{quaker}(X) &: \text{pacifist}(X) \\
\text{pacifist}(X) \\
\text{republican}(X) &: \text{hawk}(X) \\
\text{hawk}(X)
\end{align*}
\]

Let \( KB \) be

\[
\begin{align*}
\text{quaker}(\text{nixon}). \\
\text{republican}(\text{nixon}). \\
\text{false} &: \text{pacifist}(X), \text{hawk}(X).
\end{align*}
\]
Non-determinism

Conflicting defaults

\[
\text{quaker}(X) : \text{pacifist}(X) \quad \text{republican}(X) : \text{hawk}(X) \\
\text{pacifist}(X) \quad \text{hawk}(X)
\]

Let $KB$ be

\[
\text{quaker}(	ext{nixon}). \\
\text{republican}(	ext{nixon}). \\
\text{false} :- \text{pacifist}(X), \text{hawk}(X).
\]

Applying one default to Nixon makes the other inapplicable.

$KB$ has two incompatible extensions, breaking

least fixed point (provability model) for Horn clauses.
Normal default rules and inferring negations

A default rule is *normal* if its justification is its conclusion \[ p : c \]

\[ \begin{array}{c}
\text{infer } c \text{ if it is consistent and } p \text{ is provable}
\end{array} \]
Normal default rules and inferring negations

A default rule is *normal* if its justification is its conclusion \( \frac{p : c}{c} \)

\( \vdash \) infer \( c \) if it is consistent and \( p \) is provable

*Closed World Assumption*: any unprovable atom \( \varphi \) is false

\[ \text{true} : \neg \varphi \]

\[ \frac{\neg \varphi}{\neg \varphi} \]
A default rule is normal if its justification is its conclusion

\[ \frac{p}{c} \]

– infer \( c \) if it is consistent and \( p \) is provable

Closed World Assumption: any unprovable atom \( \varphi \) is false

\[ \text{true} : \neg \varphi \]

Negation as failure: \( \varphi \) is false if attempting to prove \( \varphi \) fails finitely

\[ \text{naf}(P) :- (P,!\text{,fail}); \text{true}. \]
Normal default rules and inferring negations

A default rule is normal if its justification is its conclusion \( \frac{p : c}{c} \)

\[ \text{− infer } c \text{ if it is consistent and } p \text{ is provable} \]

*Closed World Assumption*: any unprovable atom \( \varphi \) is false

\[
\begin{align*}
\text{true} : \neg \varphi \\
\neg \varphi
\end{align*}
\]

*Negation as failure*: \( \varphi \) is false if attempting to prove \( \varphi \) fails finitely

\[
naf(P) :- (P,!,fail); \text{ true.}
\]

N.B. Checking finite failure can be as hard as the Halting Problem.
### 3 modes of inference (C.S. Peirce)

<table>
<thead>
<tr>
<th>Deduction</th>
<th>deduce</th>
<th>modus ponens (\equiv) function app (f(a))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abduction</td>
<td>explain</td>
<td>choose input (a) from <em>assumables</em></td>
</tr>
<tr>
<td>Induction</td>
<td>generalise/program</td>
<td>choose rule/function (f)</td>
</tr>
</tbody>
</table>
3 modes of inference (C.S. Peirce)

Deduction | deduce | modus ponens \( \cong \) function app \( f(a) \)
Abduction  | explain | choose input \( a \) from *assumables*
Induction  | generalise/program | choose rule/function \( f \)

From \( \models \) as inclusion \( \subseteq \)

\[
KB \models g \iff \text{Mod}(KB) \subseteq \text{Mod}(g)
\]

\[
KB \text{ satisfiable} \iff \text{Mod}(KB) \not\subseteq \text{Mod}(\text{false})
\]

\[
\iff \text{Mod}(KB) \neq \emptyset
\]

to weighing alternatives \( d \in D \) via probabilities given \( KB \)

\[
\text{prob}(d|KB) = \text{conditional probability of } d \text{ given } KB
\]

\( \leadsto \) Bayesian networks, from next week on.