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.
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.
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.

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

$$\text{false} :\neg a_1, \ldots, a_k$$

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 $I$.

Example

$$KB = \begin{cases} & \text{false} :\neg a, b. \\ & a :\neg c. \\ & b :\neg c. \end{cases}$$

$$KB \models \neg c$$
Disjunctions

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

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

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

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

\[ KB = \begin{cases} 
\text{false} & : - \ a, b. \\
\hspace{1cm} a & : - \ c. \\
\hspace{1cm} b & : - \ d. 
\end{cases} \]

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

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

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

```
empty-course(X) :- course(X), \+
    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.
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.

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

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.

empty-course(X) :- course(X), \+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*

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

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

to allow for exceptions.

*Default rule* (R. Reiter)

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

Encode *birds fly*

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

to allow for exceptions.

*Default rule* (R. Reiter)

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

In general,

\[
\begin{array}{c}
\text{prerequisite } p : \text{justification } j \\
\text{conclusion } c
\end{array}
\]

applied to \( KB \) says:

\[
\begin{array}{c}
\text{conclude } c \text{ if } KB \models p \text{ and } j \text{ is } KB\text{-consistent} \\\
KB, j \not\models \text{false}
\end{array}
\]
Rules and defaults

Encode *birds fly*

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

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

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

to allow for exceptions.

*Default rule* (R. Reiter)

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

In general,

\[
\text{prerequisite } p : \text{justification } j
\]

\[
\text{conclusion } c
\]

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

Let $KB$ be

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

Conclude:
Birds and bees

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

Let \(KB\) be

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

Conclude:
\[
\text{fly}(robin) \quad \text{by default rule (\star)}
\]

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

Let $KB$ be

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

Conclude:

\[
\begin{align*}
\text{fly(robin)} & \quad \text{by default rule (⋆)} \\
\text{but not fly(penguin)}. \\
\text{An explanation of fly(bee) using (⋆) is} \\
\text{bird(bee)}
\end{align*}
\]
Birds and bees

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

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:

\[\text{fly}(\text{robin}) \quad \text{by default rule} \ (\star)\]

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

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

\[\text{bird}(\text{bee})\]

which we can block by adding to \(KB\) the rule

\[\text{false} : \text{bird}(\text{bee}).\]
Non-determinism

Conflicting defaults

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

\[
\text{pacifist}(X) \quad \text{hawk}(X)
\]
Non-determinism

Conflicting defaults

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

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

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

Let \( KB \) be

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

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

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

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

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

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

N.B. Checking finite failure can be as hard as the Halting Problem.
Normal default rules and inferring negations

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

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

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

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

\[ \quad \rightarrow \quad \neg \varphi \]

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

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

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

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

\[
\neg \varphi
\]

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

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

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)

<table>
<thead>
<tr>
<th>Deduction</th>
<th>deduce</th>
<th>modus ponens $\cong$ function app $f(a)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abduction</td>
<td>explain</td>
<td>choose input $a$ from assumables</td>
</tr>
<tr>
<td>Induction</td>
<td>generalise/program</td>
<td>choose rule/function $f$</td>
</tr>
</tbody>
</table>

From $\models$ as inclusion $\subseteq$

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

$KB$ 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 ...