Horn clauses and negations

An integrity constraint is a clause of the form

\[ \text{false} :\!-\! \text{a}_1, \ldots, \text{a}_k \]

where each \( \text{a}_i \) 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

$$\text{false} :- \text{a}_1, \ldots, \text{a}_k$$

where each $\text{a}_i$ 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

```
false :- a1,...,ak
```

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

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

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

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

\( KB \models \neg c \)
Satisfiability

Every set of definite clauses is satisfiable (i.e., has a model). Not so with Horn clauses

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

Example

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

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

Every set of definite clauses is satisfiable (i.e., has a model). Not so with Horn clauses

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

Every set of definite clauses is satisfiable (i.e., has a model). 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

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

\( KB \models \neg c \lor \neg d \)
Satisfiability and disjunctions

Every set of definite clauses is satisfiable (i.e., has a model). 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

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

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

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 **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.

\[
\text{empty-course}(X) \leftarrow \text{course}(X), \\
\neg \text{enrolled}(_,X).
\]

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.

\[ \text{empty-course}(X) \leftarrow \text{course}(X), \neg \text{enrolled}(\_, X). \]

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*

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

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)

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

Encode *birds fly*

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

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

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

to allow for exceptions.

*Default rule* (R. Reiter)

In general,

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

\[
\text{conclusion } c
\]

applied to \( KB \) says:

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

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

Encode *birds fly*

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

to allow for exceptions.

Default rule (R. Reiter)

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

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

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

\[
\begin{aligned}
& KB, j \not\models false \\
& j \text{ is true in some model of } KB
\end{aligned}
\]
Birds and bees

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

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

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

Let \( KB \) be

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

Conclude:

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

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

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

Let \( KB \) be

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

Conclude:

- fly(robin) by default rule (\( \ast \))

but not fly(penguin).

An explanation of fly(bee) using (\( \ast \)) is

- bird(bee)
Birds and bees

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

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

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

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

Conflicting defaults

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

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

\[
\text{quaker}(\text{nixon}).
\]

\[
\text{republican}(\text{nixon}).
\]

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

Applying one default to Nixon makes the other inapplicable.

\( KB \) has two incompatible extensions, breaking

\[
\text{least fixed point (provability model) for Horn clauses.}
\]
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 \( \neg \varphi \)

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

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

N.B. Checking finite failure can be as hard as the Halting Problem.
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

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

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

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

A default rule is *normal* if its justification is its conclusion

\[
\begin{array}{c}
p : c \\
\hline
\end{array}
\]

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

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

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

\[
\text{– } \neg \varphi
\]

*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>Mode</th>
<th>Action</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deduction</td>
<td>deduce</td>
<td>modus ponens ≈ function app $f(a)$</td>
</tr>
<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>

- Deduction: Deduce
- Abduction: Explain
- Induction: Generalise/Program

- **Typed functional prog ≈ proof**

- **KB** satisfies $\iff$ Mod($\text{KB}$) $\subseteq$ Mod($\text{g}$) $\iff$ $\text{Mod}(\text{KB}) \neq \emptyset$
3 modes of inference (C.S. Peirce)

<table>
<thead>
<tr>
<th>Deduction</th>
<th>deduce</th>
<th>typed functional prog ≈ proof</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abduction</td>
<td>explain</td>
<td>modus ponens ≈ function app</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$f(a)$</td>
</tr>
<tr>
<td>Induction</td>
<td>generalise/program</td>
<td>choose input $a$ from</td>
</tr>
<tr>
<td></td>
<td></td>
<td>assumables</td>
</tr>
<tr>
<td></td>
<td></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 . . .