Auxiliary material - related to predicate calculus and undecidability.

Clausal logic

Peter Flach

University of Bristol
**Propositional clausal logic**

✓ expressions that can be true or false

**Relational clausal logic**

✓ constants and variables refer to objects

**Full clausal logic**

✓ functors aggregate objects

**Definite clause logic = pure Prolog**

✓ no disjunctive heads
“Somebody is married or a bachelor if he is a man and an adult.”

Propositional clausal logic: syntax

- **Married**;
- Bachelor: Man and Adult.

- **Married** ∨ Bachelor ∨ ¬ Man ∨ ¬ Adult
Persons are happy or sad

\[ \text{happy; sad : } \text{person} . \]

No person is both happy and sad

\[ : \text{person, happy, sad} . \]

Sad persons are not happy

\[ : \text{person, sad, happy} . \]

Non-happy persons are sad

\[ \text{sad; happy : } \text{person} . \]
Herbrand base: set of atoms

\{married, bachelor, man, adult\}

Herbrand interpretation: set of true atoms

\{married, man, adult\}

A clause is false in an interpretation if all body-literals are true and all head-literals are false...

bachelor \ :- \ man, adult.

...and true otherwise: the interpretation is a model of the clause.

: married, bachelor.
A clause $C$ is a **logical consequence** of a program (set of clauses) $P$ iff every model of $P$ is a model of $C$.

Let $P$ be

```
married; bachelbr:-man, adult.
man.
:-bachelbr.
```

- $married$ is a logical consequence of $P$;
- $married$ is a logical consequence of $P$;
- $bachelbr$ is a logical consequence of $P$;
- $bachelbr$ is not a logical consequence of $P$;
- $bachelbr$ is a logical consequence of $P$.

**Exercise 2.2**
Propositional resolution
Propositional resolution is

- **sound**: it derives only logical consequences.
- **incomplete**: it cannot derive arbitrary tautologies like \( a \rightarrow \neg a \ldots \)
- \( \ldots \) but **refutation-complete**: it derives the empty clause from any inconsistent set of clauses.

**Proof by refutation**: add the negation of the assumed logical consequence to the program, and prove inconsistency by deriving the empty clause.
Exercise 2.4
Exercise 2.5

Direct proof:

- friendly :- happy
- happy :- has_friends

Proof by refutation:

- (friendly :- has_friends) \Rightarrow
- (friendly \lor \neg has_friends) \Rightarrow
- (\neg friendly) \land (has_friends) \Rightarrow
- :- friendly and has_friends
“Peter likes anybody who is his student.”

\[
\text{likes(} \text{peter}, S \text{)} :\text{-student_of(S, peter).}
\]

Relational clausal logic: syntax
A substitution maps variables to terms:

\{ S \rightarrow \text{maria} \}

A substitution can be applied to a clause:

\text{likes(likes(peter,maria):-student_of(maria,peter))}.

The resulting clause is said to be an instance of the original clause, and a ground instance if it does not contain variables.

Each instance of a clause is among its logical consequences.
**Herbrand universe**: set of ground terms (i.e. constants)

\{peter, maria\}

**Herbrand base**: set of ground atoms

\{likes(peter, peter), likes(peter, maria), likes(maria, peter), likes(maria, maria), student_of(peter, peter), student_of(peter, maria), student_of(maria, peter), student_of(maria, maria)\}

**Herbrand interpretation**: set of true ground atoms

\{likes(peter, maria), student_of(maria, peter)\}

An interpretation is a **model** for a clause if it makes all of its ground instances **true**

\[\text{likes(peter, maria)} : : \text{student_of(maria, peter)}.\]
\[\text{likes(peter, peter)} : : \text{student_of(peter, peter)}.\]
Relational resolution

-likes(peter,N)
-likes(peter,S):-student_of(S,peter)

{S→N}

:-student_of(N,peter)
student_of(S,T):-follows(S,C),teaches(T,C)

{S→N,T→peter}

:-follows(N,C),teaches(peter,C)
follows(maria,ai_techniques)

{N→maria,C→ai_techniques}

:-teaches(peter,ai_techniques)
teaches(peter,ai_techniques)
"Everybody loves somebody."

\[
\text{loves}(x, \text{person\_loved\_by}(x)).
\]

\[
\text{loves}(\text{peter}, \text{person\_loved\_by}(\text{peter})).
\]

\[
\text{loves}(\text{anna}, \text{person\_loved\_by}(\text{anna})).
\]

\[
\text{loves}(\text{paul}, \text{person\_loved\_by}(\text{paul})).
\]

...
**Every mouse has a tail**

\[
\text{tail_of(tail(X),X):-mouse(X).}
\]

**Somebody loves everybody**

\[
\text{loves(person_who_loves_everybody,XX).}
\]

**Every two numbers have a maximum**

\[
\text{max_of(max(X,Y),max(X,Y)):-number(X),number(Y).}
\]
**Herbrand universe**: set of ground terms

\{0, s(0), s(s(0)), s(s(s(0))), \ldots \}

**Herbrand base**: set of ground atoms

\{\text{plus}(0,0,0), \text{plus}(s(0),0,0), \ldots, \\
\text{plus}(0,s(0),0), \text{plus}(s(0),s(0),0), \ldots, \\
\ldots, \\
\text{plus}(s(0),s(s(0)),s(s(s(0))))\}

**Herbrand interpretation**: set of **true** ground atoms

\{\text{plus}(0,0,0), \text{plus}(s(0),0,s(0)), \text{plus}(0,s(0),s(0))\}

Some programs have only infinite models

\text{plus}(0,X,X).

\text{plus}(s(X),Y,s(Z)) :- \text{plus}(X,Y,Z).
\[\text{plus}(X,Y,s(Y)) \]
and
\[\text{plus}(s(V),W,s(s(V)))\]
unify to
\[\text{plus}(s(V),s(V),s(s(V)))\]

\[\text{length}([X|Y],s(0)) \]
and
\[\text{length}([V],V)\]
unify to
\[\text{length}([s(0)],s(0))\]

\[\text{larger}(s(s(X),X)\]
and
\[\text{larger}(V,s(V))\]
do **not** unify (occur check!)
<table>
<thead>
<tr>
<th><strong>Propositional</strong></th>
<th><strong>Relational</strong></th>
<th><strong>Full clausal logic</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Herbrand universe</strong></td>
<td>${a, b}$ (finite)</td>
<td>${a, f(a), f(f(a)), \ldots}$ (infinite)</td>
</tr>
<tr>
<td><strong>Herbrand base</strong></td>
<td>${p, q}$</td>
<td>${p(a,a), p(b,a), \ldots}$ (finite)</td>
</tr>
<tr>
<td><strong>clause</strong></td>
<td>$p \leftarrow q.$</td>
<td>$p(X,Z) \leftarrow q(X,Y), p(Y,Z).$</td>
</tr>
<tr>
<td><strong>Herbrand models</strong></td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>${p}$</td>
<td>${p(a,a)}$</td>
<td>${p(a,f(a)), q(a)}$</td>
</tr>
<tr>
<td>${p, q}$</td>
<td>${p(a,a), p(b,a), q(b,a)}$</td>
<td>${p(f(a), f(f(a))), q(f(a))}$</td>
</tr>
<tr>
<td><strong>Meta-theory</strong></td>
<td>sound</td>
<td>sound</td>
</tr>
<tr>
<td></td>
<td>refutation-complete</td>
<td>refutation-complete</td>
</tr>
<tr>
<td></td>
<td>decidable</td>
<td>decidable</td>
</tr>
</tbody>
</table>

**Summary**
Exercise 2.12
married\text{,:bachelor}:-adult\text{_man}.

\text{adult\_man}.

married:-adult\_man,\text{notbachelor}.

bachelor:-adult\_man,\text{notmarried}.

From indefinite to general clauses
“Everyone has a mother, but not every woman has a child.”

\[
\forall Y \exists X \text{ mother_of}(X,Y) \land \neg \forall Z \exists W \text{ woman}(Z) \rightarrow \text{ mother_of}(Z,W)
\]

- **push negation inside**

\[
\forall Y \exists X \text{ mother_of}(X,Y) \land \exists Z \forall W \text{ woman}(Z) \land \neg \text{ mother_of}(Z,W)
\]

- **drop quantifiers (Skolemisation)**

\[
\text{ mother_of(mother}(Y),Y) \land \text{ woman}(\text{ childless_woman}) \land \neg \text{ mother_of}(\text{ childless_woman},W)
\]

- **(convert to CNF and) rewrite as clauses**

\[
\text{ mother_of(mother}(Y),Y). \\
\text{ woman}(\text{ childless_woman}). \\
\neg \text{ mother_of}(\text{ childless_woman},W).
\]