**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.”

married; bachelor: -man, adult.

head = positive literals

body = negative literals

married ∨ bachelor ∨ -man ∨ -adult

Propositional clausal logic: syntax
Persons are happy or sad

happy; sad:-person.

No person is both happy and sad

:-person, happy, sad.

Sad persons are not happy

:-person, sad, happy.

Non-happy persons are sad

sad; happy:-person.

Exercise 2.1
**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**...

\texttt{bachelor:-man,adult}.

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

:-\texttt{married,bachelor}.

Propositional clausal logic: semantics
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; bachelor:-man, adult.
man.
:-bachelor.
```

\( married:-adult \) is a logical consequence of \( P \);

\( married:-bachelor \) is a logical consequence of \( P \);

\( bachelor:-man \) is not a logical consequence of \( P \);

\( bachelor:-bachelor \) is a logical consequence of \( P \).
Propositional resolution
Propositional resolution is

- **sound**: it derives only logical consequences.
- **incomplete**: it cannot derive arbitrary tautologies like \( a: \neg a \ldots \)
- …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.
happy; friendly:~teacher  friendly:~teacher, happy

friendly:~teacher  teacher; wise  teacher:~wise

teacher  friendly
Direct proof:

\[-\text{friendly} \text{ friendly} - \text{happy} \text{ happy} - \text{has\_friends}\]

\[\text{friendly} - \text{has\_friends}\]

\[-\text{happy} \text{ happy} - \text{has\_friends}\]

\[-\text{has\_friends} \text{ has\_friends}\]

\[\{}\]

Proof by refutation:

\[-(\text{friendly} - \text{has\_friends}) \Rightarrow\]
\[-(\text{friendly} \lor \neg \text{has\_friends}) \Rightarrow\]
\[(\neg\text{friendly}) \land (\text{has\_friends}) \Rightarrow\]
\[-\text{friendly} \text{ and } \text{has\_friends}\]

Exercise 2.5
“Peter likes anybody who is his student.”

```
likes(peter, S) :- 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:

likes(\text{peter}, \text{maria}) :\neg \text{student\_of(\text{maria}, \text{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**

\begin{align*}
\text{likes(peter, maria):} & \text{:- student_of(maria, peter).} \\
\text{likes(peter, peter):} & \text{:- student_of(peter, peter).}
\end{align*}
"Everybody loves somebody."

```
loves(x, person_loved_by(x)).
loves(peter, person_loved_by(peter)).
loves(anna, person_loved_by(anna)).
loves(paul, person_loved_by(paul)).
...
```

Full clausal logic: syntax
Every mouse has a tail

\[
\text{tail\_of}(\text{tail}(X), X) : \neg \text{mouse}(X).
\]

Somebody loves everybody

\[
\text{loves}(\text{person\_who\_loves\_everybody}, X).
\]

Every two numbers have a maximum

\[
\text{maximum\_of}(X, Y, \text{max}(X, Y)) : \neg \text{number}(X), \text{number}(Y).
\]

Exercise 2.9
**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).
Exercise 2.11

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

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

\[
\text{larger}(s(s(X), X) \\
\text{and} \\
\text{larger}(V, s(V)) \\
\text{do not unify (occur check!)}
\]
<table>
<thead>
<tr>
<th>Herbrand universe</th>
<th>Propositional —</th>
<th>Relational —</th>
<th>Full clausal logic</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>{a, b}</td>
<td>{a, b}</td>
<td>{a, f(a), f(f(a)),…}</td>
</tr>
<tr>
<td></td>
<td>(finite)</td>
<td>(finite)</td>
<td>(infinite)</td>
</tr>
<tr>
<td>Herbrand base</td>
<td>{p, q}</td>
<td>{p(a,a), p(b,a),…}</td>
<td>{p(a,f(a)), p(f(a), f(f(a))),…}</td>
</tr>
<tr>
<td></td>
<td>(finite)</td>
<td>(finite)</td>
<td>(infinite)</td>
</tr>
<tr>
<td>clause</td>
<td>p:-q.</td>
<td>p(X,Z):-q(X,Y), p(Y,Z) .</td>
<td>p(X,f(X)) :-q(X) .</td>
</tr>
<tr>
<td></td>
<td>Ø</td>
<td>Ø</td>
<td>Ø</td>
</tr>
<tr>
<td>Herbrand models</td>
<td>{p}</td>
<td>{p(a,a)}</td>
<td>{p(a,f(a)), q(a)}</td>
</tr>
<tr>
<td></td>
<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></td>
<td>(finite number of finite models)</td>
<td>(infinite number of finite or infinite models)</td>
<td></td>
</tr>
<tr>
<td>Meta-theory</td>
<td>sound</td>
<td>sound</td>
<td>sound (if unifying with occur check)</td>
</tr>
<tr>
<td></td>
<td>refutation-complete decidable</td>
<td>refutation-complete decidable</td>
<td>refutation-complete semi-decidable</td>
</tr>
</tbody>
</table>

Summary
Exercise 2.12
married; bachelor: ~adult_man.
adult_man.

married: ~adult_man, not bachelor.
bachelor: ~adult_man, not married.

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

∀Y∃X:mother_of(X,Y) ∧ ¬∀Z∃W:woman(Z) → mother_of(Z,W)

push negation inside

∀Y∃X:mother_of(X,Y) ∧ ∃Z∀W:woman(Z) ∧ ¬mother_of(Z,W)

drop quantifiers (Skolemisation)

mother_of(mother(Y), Y) ∧ woman(childless_woman) ∧ ¬mother_of(childless_woman, W)

(convert to CNF and) rewrite as clauses

mother_of(mother(Y), Y).
woman(childless_woman).
:- mother_of(childless_woman, W).