Equality

It is often useful to be able to say that two objects are equal. Introduce a special predicate symbol “=” of arity 2 to represent this. We write \( x = y \) instead of \( = (x, y) \).

Semantically, we want \( = \) to be interpreted as a binary relation, but not any old binary relation! We require that for every structure \( M \),

\[
_\mathcal{M} = \{(x, x) \mid x \in A^\mathcal{M}\}
\]

That is, \( (x, y) \in _\mathcal{M} \) iff \( x \) and \( y \) are the same object.
Warning: the semantics of predicate logic knows nothing about your intended application. The following is also a perfectly good structure for the chess language:

\[ a^M = \{0\} \]
\[ \text{black} - \text{king}^M = \text{white} - \text{king}^M = 0 \]
\[ \text{colour}(0) = 0 \]
\[ \text{King}^M = \text{Queen}^M = \{0\} \]
\[ \text{Adjacent}^M = \{(0,0)\} \]

Note that many names refer to the same object.

More generally, for a given language, there are many models that have properties that we do not want.

For example, in the chess domain, there are models \( M \) in which

1. \((a1, h8) \in \text{Adjacent}^M\) (The bottom left corner square is adjacent to the top right corner square).
2. \((bk, bk) \in \text{On}^M\) (The black king is on itself.)
3. \((bk, a1) \in \text{On}^M\) and \((wq, a1) \in \text{On}^M\) (Both the black king and the white queen are on square \( a1 \).)
4. \((bk, a1) \in \text{On}^M\) and \((bk, b7) \in \text{On}^M\) (The black king is on square \( a1 \) and on the square \( b7 \).)
If we want to reason about chess positions, we need to distinguish structures that correspond to sensible chess positions from structures that do not.

We can do this by finding a set of formulas $\Gamma$ such that if a structure $M$ corresponds to a chess position then $M \models \Gamma$ (i.e., $M \models \phi$ for all $\phi \in \Gamma$.)

If $\Gamma$ has this property, then if $\Gamma \models \psi$ then $\psi$ is true in all chess positions.

Some properties of chess positions:

1. Everything is either a square, a piece or a colour.
2. No square is adjacent to itself.
3. If two squares are adjacent, then they have different colours.
4. The only things that can be on a square are pieces.
5. There is at most one thing on any square.
6. Each piece is on at most one square.
Using the predicates \texttt{square}(X), \texttt{piece}(X), \texttt{adjacent}(X, Y), \texttt{On}(X, Y) and \texttt{colour}(X, Y), we can capture these as ...

1. $\forall X (\texttt{square}(X) \lor \texttt{piece}(X) \lor \texttt{colour}(X))$
2. $\neg \exists X (\texttt{square}(X) \land \texttt{adjacent}(X, X))$
3. $\forall X \forall Y \forall U \forall V ((\texttt{square}(X) \land \texttt{square}(Y) \land \texttt{adjacent}(X, Y) \land \texttt{colour}(X, U) \land \texttt{colour}(Y, V)) \rightarrow \neg (U = V))$
4. $\forall X \forall Y ((\texttt{square}(X) \land \texttt{On}(Y, X)) \rightarrow \texttt{piece}(Y))$
5. $\forall X \forall Y \forall Z ((\texttt{square}(X) \land \texttt{On}(Y, X) \land \texttt{On}(Z, X)) \rightarrow Y = Z)$
6. $\forall X \forall Y \forall Z ((\texttt{piece}(X) \land \texttt{On}(X, Y) \land \texttt{On}(X, Z) \land \texttt{square}(Y) \land \texttt{square}(Z)) \rightarrow Y = Z)$.

\textbf{A clausal form for predicate logic}

A \textit{literal} of predicate logic is either an atomic formula or its negation, i.e., a formula of the form $P(t_1, \ldots, t_m)$ or $\neg P(t_1, \ldots, t_m)$.

A formula of predicate logic is said to be a \textit{clause} if it is a sentence of the form

$$\forall x_1 \forall x_2 \ldots \forall x_k (L_1 \lor \ldots \lor L_m)$$

where the $L_i$ are literals.

(There are no free variables. If all the $L_i$ are ground, then we may have no quantifiers at the front.)
Just as in propositional logic, a *Horn Clause* is a clause with at most one positive atom.

We may write these in the forms:

\[ \forall x_1 \forall x_2 \ldots \forall x_k (B_1 \land \ldots \land B_n \rightarrow H) \]
\[ \forall x_1 \forall x_2 \ldots \forall x_k (\top \rightarrow H) \]
\[ \forall x_1 \forall x_2 \ldots \forall x_k (B_1 \land \ldots \land B_n \rightarrow \bot) \]

where the \( B_i \) and \( H \) are positive atoms.

Example (*): the following is a set of Horn clauses:

\[ \forall X \forall Y (\text{edge}(X,Y) \rightarrow \text{path}(X,Y)) \]
\[ \forall X \forall Y \forall Z (\text{edge}(X,Y) \land \text{path}(Y,Z) \rightarrow \text{path}(X,Z)) \]
\[ \top \rightarrow \text{edge}(a,b) \]
\[ \top \rightarrow \text{edge}(b,c) \]
\[ \top \rightarrow \text{edge}(c,a) \]
\[ \top \rightarrow \text{edge}(b,d) \]
\[ \text{path}(c,d) \rightarrow \bot \]
**Definition:** A *ground term* is a term that does not contain any variables, i.e., contains only constants and function symbols. A *ground atom* is an atom \( P(t_1, \ldots, t_n) \) such that each \( t_i \) is a ground term.

Examples:

- black-king, colour(black-king) are ground terms
- \( X \), colour(colour(X)) are not ground terms.
- adjacent(black-king,black-queen) is a ground atom
- adjacent(black-king,X) is not a ground atom

The *Herbrand base* of a set \( \Gamma \) of clauses is the set of all ground terms formed using only constants and function symbols that occur in \( \Gamma \).

Example:

The Herbrand base of example (*) is \( \{a, b, c, d\} \).

If we add the clause \( \forall X \text{On}(X, f(X)) \), we also need to the Herbrand base all the terms we can construct using the function symbol \( f \), so the Herbrand base is

\[
\{a, b, c, d, f(a), f(b), f(c), f(d), \\
f(f(a)), f(f(b)), f(f(c)), f(f(d)), \\
f(f(f(a))), \ldots\}
\]
An instance (with respect to the Herbrand base $B$) of a clause $\forall x_1 \ldots \forall x_n (L_1 \lor \ldots \lor L_n)$ is a formula obtained by substituting a ground term in $B$ for each variable $x_i$ and removing the quantifiers.

Example: if $B = \{a, b, c\ldots\}$ then

$$\forall X \forall Y \forall Z (\text{edge}(X, Y) \land \text{path}(Y, Z) \rightarrow \text{path}(X, Z))$$

has instances

- $\text{edge}(a, a) \land \text{path}(a, a) \rightarrow \text{path}(a, a)$
- $\text{edge}(a, b) \land \text{path}(b, a) \rightarrow \text{path}(a, a)$
- $\text{edge}(a, b) \land \text{path}(b, c) \rightarrow \text{path}(a, c)$

etc .... When there is also a function symbol $f$ in the Herbrand base, we also have instances such as

- $\text{edge}(a, f(b)) \land \text{path}(f(b), f(f(c))) \rightarrow \text{path}(a, f(f(c)))$

Write $\text{Instances}(\Gamma)$ for the set of all instances of a set of clauses $\Gamma$ (with respect to its own Herbrand base.)

Remark: If the set of clauses $\Gamma$ contains only a finite set of constant symbols then the Herbrand base is finite, the clause has a finite set of instances.

If $\Gamma$ contains a function symbol, then the Herbrand base is infinite, and the clause has an infinite set of instances.
Let $\Gamma$ be a set of predicate logic clauses.

By treating each ground atom as a propositional constant, we can view $\text{Instances}(\Gamma)$ as a set of propositional clauses.

**Theorem:** $\Gamma$ is satisfiable if and only if $\text{Instances}(\Gamma)$ is satisfiable.

This reduces the satisfiability problem for predicate logic to the satisfiability problem for propositional logic.

In particular, if all the clauses in $\Gamma$ are Horn clauses, and there are no function symbols, then the algorithm HORN can be applied to $\text{Instances}(\Gamma)$ to determine satisfiability of $\Gamma$. 