Knowledge Representation

- Attribute-Value pairs, frames, and semantic networks allow you to represent knowledge very effectively, but...
- ...accessing and reasoning with this knowledge is *ad hoc*.
- However, our reasoning does not seem *ad hoc*...we follow certain reasoning patterns or rules.
Rule-based Systems

- Rule-based systems try to mimic our reasoning steps with sets of if-then rules:
  
  \[
  \text{if is-fresh(coffee) then pour(coffee)}
  \]
  
  \[
  \text{if not is-fresh(coffee) then make(coffee)}
  \]

- This kind of reasoning was already studied by the ancient Greeks and is referred to as the \textit{modus ponens},

\[
\begin{align*}
\text{if A then B} \\
A = \text{true} \\
\therefore B = \text{true}
\end{align*}
\]

- Sometimes rules are also referred to as \textit{productions} or \textit{production rules}.

Rules:

If \text{<condition>} then \text{<action>}
Rule-based Systems

Computation step:
- The interpreter
  - selects a rule from the rulebase
  - applies the rule to the symbols in the working memory
  - updates the working memory

Rules can be selected in an arbitrary order only depending on the state of the computation.
Rule-based Systems

- A convenient framework for rule-based reasoning is **first-order logic**
- Rather than arbitrary data structures, first-order logic depends on:
  - Quantified Variables
  - Predicates
  - Logical Connectives
  - If-then Rules
First-Order Logic

- Quantified Variables
  - Universally quantified variables
    \[ \forall X \quad \text{for all objects } X \]
  - Existentially quantified variables
    \[ \exists Y \quad \text{there exists an object } Y \]
First-Order Logic

- **Predicates**
  - Predicates are functions that map their arguments into *true/false*
  - The signature of a predicate $p(X)$ is
    
    $$p: \text{Objects} \rightarrow \{ \text{true, false} \}$$
    
    with $X \in \text{Objects}$.
  - **Example: human($X$)**
    - $\text{human}: \text{Objects} \rightarrow \{ \text{true, false} \}$
    - $\text{human(tree)} = \text{false}$
    - $\text{human(paul)} = \text{true}$
  - **Example: mother($X,Y$)**
    - $\text{mother}: \text{Objects} \times \text{Objects} \rightarrow \{ \text{true, false} \}$
    - $\text{mother(betty,paul)} = \text{true}$
    - $\text{Mother(giraffe,peter)} = \text{false}$
First-Order Logic

- We can combine predicates and quantified variables to make statements on sets of objects
  - $\exists X [\text{mother}(X, \text{paul})]$  
    - there exists an object $X$ such that $X$ is the mother of Paul
  - $\forall Y [\text{human}(Y)]$  
    - for all objects $Y$ such that $Y$ is human
First-Order Logic

- Logical Connectives: and, or, not

  - $\exists F \forall C [\text{parent}(F,C) \text{ and } \text{male}(F)]$
    - There exists an object $F$ for all object $C$ such that $F$ is a parent of $C$ and $F$ is male.

  - $\forall X [\text{day}(X) \text{ and } (\text{comfortable}(X) \text{ or } \text{rainy}(X))]$
    - For all objects $X$ such that $X$ is a day and $X$ is either comfortable or rainy.
First-Order Logic

- If-then rules: \( A \rightarrow B \)
  - \( \forall X \forall Y [\text{parent}(X,Y) \text{ and } \text{female}(X) \rightarrow \text{mother}(X)] \)
    - For all objects \( X \) and for all objects \( Y \) such that if \( X \) is a parent of \( Y \) and \( X \) is female then \( X \) is a mother.
  - \( \forall Q [\text{human}(Q) \rightarrow \text{mortal}(Q)] \)
    - For all objects \( Q \) such that if \( Q \) is human then \( Q \) is mortal.
First-Order Logic

∀∅ [female(pam)]
∀∅ [female(liz)]
∀∅ [female(ann)]
∀∅ [female(pat)]

∀∅ [male(tom)]
∀∅ [male(bob)]
∀∅ [male(jim)]

∀∅ [parent(pam,bob)]
∀∅ [parent(tom,bob)]
∀∅ [parent(tom,liz)]
∀∅ [parent(bob,ann)]
∀∅ [parent(bob,pat)]
∀∅ [parent(pat,jim)]

∀X∀Y [parent(X,Y) and female(X) → mother(X)]
∀X∀Y [parent(X,Y) and male(X) → father(X)]
∀X∀Y ∀YZ [parent(X,Y) and parent(X,Z) and not same-person(Y,Z) → siblings(Y,Z)]

How about sister?
How about grandparent?

NOTE: if we only consider the persons mentioned here, then we are making use of the **closed world assumption**.
Prolog = Programming in Logic

Executable First-Order Logic

Facts:
\[ \forall \forall [\text{female}(pam)] \]
becomes
female(pam).

Rules:
\[ \forall \forall Y [\text{parent}(X,Y) \text{ and } \text{female}(X) \rightarrow \text{mother}(X)] \]
becomes
\[ \text{mother}(X) :\text{parent}(X,Y), \text{female}(X) \]

Observations:
- Think of :- as the ← arrow.
- Universal quantification is implied
- Only universally quantified rules are allowed
- Variables have to start with a capital letter
- Objects have to be all lower case letters
Prolog – Rules & Facts

We can execute this program by asking questions:

?- female(pam).
?- female(X). ∃X[female(X)]?
?- mother(pam).
?- father(Y).

Can we prove that ‘female(pam)’ is true?
Can we prove that there exists an object X that make ‘female(X)’ true?

etc

What about the ‘sameperson’ predicate?

female(pam).
female(liz).
female(ann).
female(pat).
male(tom).
male(bob).
male(jim).

parent(pam,bob).
parent(tom,bob).
parent(tom,liz).
parent(bob,ann).
parent(bob,pat).
parent(pat,jim).

mother(X) :- parent(X,Y), female(X).
father(X) :- parent(X,Y), male(X).
siblings(Y,Z) :- parent(X,Y), parent(X,Z), not(sameperson(Y,Z)).

facts

rules
Prolog – Rules & Facts

facts

isa(cardinal, bird).
isa(bluejay, bird).
isa(boy, human).
isa(girl, human).
isa(computer, artifact).
isa(airplane, artifact).
isa(bird, animal).
isa(human, animal).

has(bird, feathers).
has(bird, wings).
has(human, intelligence).
has(computer, intelligence).
has(airplane, wings).

rules

can_do(Thing, fly) :- has(Thing, wings).
can_do(Thing, think) :- has(Thing, intelligence).
can_do(Thing, live) :- isa(Thing, animal).

We can ask questions:

?- isa(cardinal,bird).
?- isa(bluejay,human).
?- can_do(human,think).

or:

?- isa(cardinal,X).
?- can_do(X,think).