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

Read Prolog Tutorial on course website

- Rule-based systems try to mimic our reasoning steps with sets of if-then rules:

```
if is-fresh(coffee) then pour(coffee)
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 *modus ponens,*

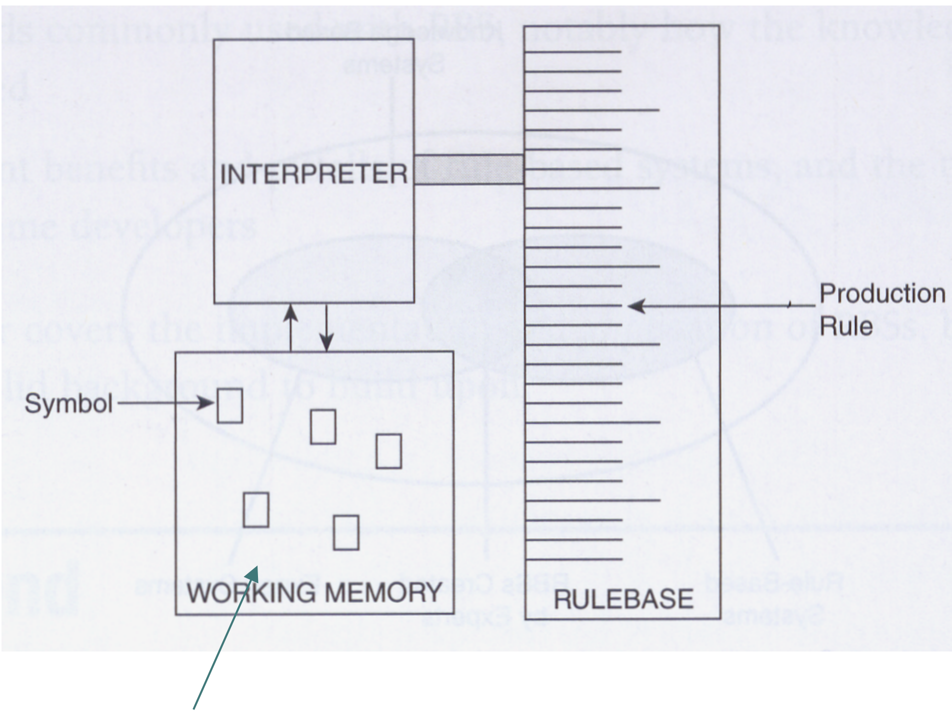
  if A then B
  A = true

  $\therefore$ B = true

- Sometimes rules are also referred to as *productions* or *production rules*.

Rules:
If <condition> then <action>

Rule-based Systems



Current State of the Reasoning (Computation)

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 (predicate 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

    $\forall X$ – for all objects X

  - Existentially quantified variables

    $\exists Y$ – 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: Objects $\rightarrow$ { true, false }

    with X $\in$ Objects.
  - Example: human(X)
    - human: Objects $\rightarrow$ { true, false }
    - human(tree) = false
    - human(paul) = true
  - Example: mother(X,Y)
    - mother: Objects $\times$ Objects $\rightarrow$ { true, false }
    - mother(betty,paul) = true
    - Mother(giraffe,peter) = false

First-Order Logic

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

First-Order Logic

- Logical Connectives: and, or, not
  - ∃F∀C[parent(F,C) and male(F)]
    - There exists an object F for all object C such that F is a parent of C and F is male.
  - ∀X[day(X) and (comfortable(X) or 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 → B
  - ∀X∀Y[parent(X,Y) and female(X) → mother(X,Y)]
    - 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.
  - ∀Q[human(Q) → mortal(Q)]
    - For all objects Q such that if Q is human then Q is mortal.

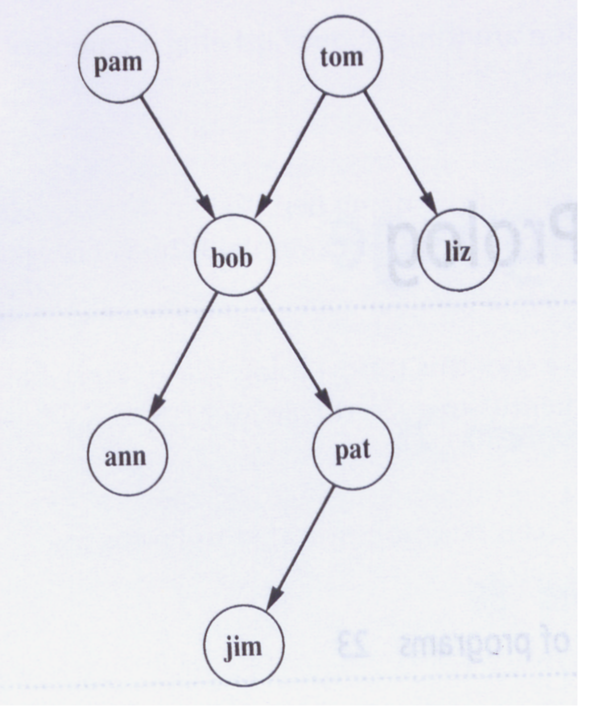
First-Order Logic

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

} Assertions

∀∅ [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 <u>closed world assumption</u>.

Prolog

Prolog = **Prog**ramming in **Log**ic

☞ Executable First-Order Logic

Facts:

∀∅ [female(pam)]

becomes

female(pam).

Rules:

∀X∀Y [parent(X,Y) and female(X) → mother(X)]

becomes

mother(X) :- parent(X,Y) , female(X)

And Connective

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

facts

```
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).
```

rules

```
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)).
```

We can execute this program by asking questions:

?- female(pam).
?- female(X). $\exists X[female(X)]?$
?- 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?

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).