

# Predicate Logic

propositional logic can represent any  
finite situation context

have sound, complete proof method  
for propositional logic

## What is problem?

only have single facts

(onblockablock implies overblockablockb)

must have this for all pairs of blocks

there are no relations

no generalizations

# First-Order Predicate Calculus (FOPC)

reasoning about predicates applied to objects  
(relations)  
allowing generalizations

## Literals

$\langle \text{literal} \rangle ::= \langle \text{predicate} \rangle (\langle \text{term}_1 \rangle \dots \langle \text{term}_n \rangle) \mid \sim \langle \text{literal} \rangle$

## Facts

literals, where terms are constants  
ground literals

## Examples

red(ball)                       $\langle \text{color} \rangle (\langle \text{object} \rangle)$

on(ball box)                       $\langle \text{loc-rel} \rangle (\langle \text{obj} \rangle \langle \text{obj} \rangle)$

age(Tom 37)                       $\text{age}(\langle \text{person} \rangle \langle \text{pos-integer} \rangle)$

often adopt such a "typed calculus"

# FOPC

## General Statements

introduced through use of  
variables  
quantification

## Quantification

### Universal

$$A(\langle \text{var} \rangle)[\langle \text{wff} \rangle]$$

where  $\langle \text{var} \rangle$  is a variable,  
possibly occurring in terms of  $\langle \text{wff} \rangle$

$$A(x)[\text{EQUAL}(x x)]$$

### Existential

$$E(\langle \text{var} \rangle)(\langle \text{wff} \rangle)$$

where  $\langle \text{var} \rangle$  is a variable  
possibly occurring in terms of  $\langle \text{wff} \rangle$

$$E(x)[\text{EQUAL}(x x)]$$

# FOPL

## Quantification

### scope

range of application of quantifier

given by syntax

inside surrounding brackets

$$E(x)[(A(y)[EQUAL(x\ y)])]$$
$$A(x)[E(y)[EQUAL(x\ y)]]$$

### semantics

$$A(\text{var})[\text{scope}]$$

true if scope is true for every element  
of domain of variable var

$$E(\text{var})[\text{scope}]$$

true if scope is true for at least one element  
of domain of variable var

# FOPL

## functional terms

$\langle \text{term} \rangle ::= \text{constant} \mid \text{variable} \mid$   
 $(\langle \text{function} \rangle \langle \text{term} \rangle^*)$

$A(y)[E(x)[EQUAL(x \text{ (plus } y \ 1))]]$

map objects to an object in some domain(s)

## skolem functions

used to replace existential quantifiers  
within scope of universal quantifiers

$A(y)[E(x)[EQUAL(x \text{ (plus } y \ 1))]]$

rewritten as

$A(y)[EQUAL(f_s(y) \text{ (plus } y \ 1))]$

$f_s(y) \Rightarrow$  the value of  $x$  for each  $y$   
that makes the scope true

$E(x)[A(y)[EQUAL(x \text{ (plus } y \ 1))]] \Rightarrow$   
 $A(y)[EQUAL(a \text{ (plus } y \ 1))],$   
where  $a$  is  $f_s()$

# FOPL

## interpretation

assignment of meaning (reference)  
to symbols in wff(s)

constants, variables ==  
to domains of objects

predicates ==  
from domains of objects  
to {true, false}

functions ==  
from domains of objects  
to a domain of objects

## example

$A(x)[E(y)[Equal(x, y)]]$

x and y to integers

Equal to relation over pairs of integers

$\{(1,1) (2,2), \dots\}$

# FOPL

## **satisfaction**

interpretation such that wff(s) is true

## **satisfiability**

existence of interpretation satisfying wff(s)

## **validity**

wff satisfied by all interpretations

same as for propositional logic,  
just definition of interpretation has changed

# FOPC

## Tasks

tell if a wff logically follows from  
a given set of satisfiable wffs

## Notion

logically follows from (is entailed by)

A wff  $w$  logically follows from a set  $S$  of wffs  
iff  $w$  is satisfied in every  
interpretation satisfying (all wffs of)  $S$

## Proof

truth table check ?

large, potentially infinite, domains

theorem proving

# FOPC

## Theorem Proving

### Computational Issues

complexity of wff syntax

equivalence of wffs

$$\sim(A \text{ or } B) \iff (\sim A \text{ and } \sim B)$$

$$(A \implies B) \iff (\sim A \text{ or } B)$$

$$A(x)[\sim W] \iff \sim E(x)[W]$$

multiple rules of inference

modus ponens

chain rule

quantification

introduction/elimination

# FOPC

## Theorem Proving

How can we deal  
with the above complications?

eliminate connectives

eliminate quantification

### **Clause Normal Form**

use single rule of inference

### **Resolution**

# Clause Normal Form

$E(x) (P(x) \text{ implies } A(y) A(z) E(t) Q(x y z t))$

**Step 1:** eliminate implications

$(A \text{ implies } B) \implies (\sim A \text{ or } B)$

$E(x) [P(x) \text{ implies } A(y) A(z) E(t) [Q(x y z t)]] \implies$   
 $E(x) [\sim P(x) \text{ or } A(y) A(z) E(t) [Q(x y z t)]]$

(using  $\implies$  as application of rule of inference)

**Step 2:** reduce scope of negation

(move next to literals)

using

$\sim\sim W \implies W$

$\sim E(x)(\dots) \implies A(x)\sim(\dots)$

$\sim A(x)(\dots) \implies E(x)\sim(\dots)$

$\sim(\dots \text{ or } \dots) \implies (\sim(\dots) \text{ and } \sim(\dots))$

$\sim(\dots \text{ and } \dots) \implies (\sim(\dots) \text{ or } \sim(\dots))$

# Clause Normal Form

## Step 3: standardize variables

replace variables with unique names  
for each quantification

## Step 4: move all quantifiers to left, in order of occurrence (prenex form)

$$\begin{aligned} E(x) (\sim P(x) \text{ or } A(y) A(z) E(t) Q(x y z t)) \\ \implies E(x) A(y) A(z) E(t) [\sim P(x) \text{ or } Q(x y z t)] \end{aligned}$$

## Step 5: eliminate existential quantification

if variable not in scope of universal  
quantification, replace by a constant

otherwise, replace by function  
of enclosing universal variables  
(skolemization)

$$\begin{aligned} E(x) A(y) A(z) E(t) (\sim P(x) \text{ or } Q(x y z t)) \\ \implies A(y) A(z) [\sim P(a) \text{ or } Q(x y z f_s(y z))] \end{aligned}$$

# Clause Normal Form

**Step 6:** drop universal quantification

$$\begin{aligned} A(y) A(z)[\sim P(a) \text{ or } Q(x \ y \ z \ f_s(y \ z))] \\ \implies (\sim P(a) \text{ or } Q(x \ y \ z \ f_s(y \ z))) \end{aligned}$$

**Step 7:** put in conjunctive normal form  
(conjunction of disjuncts)

$$\begin{aligned} ((w \text{ and } w') \text{ or } z) \implies \\ ((w \text{ or } z) \text{ and } (w' \text{ or } z)) \end{aligned}$$

**Step 8:** flatten and eliminate ANDs and ORs  
(put into clause form)

$$\begin{aligned} (\sim P(a) \text{ or } Q(x \ y \ z \ f_s(y \ z))) \\ \implies \{ [\sim P(a), Q(a \ y \ z \ f_s(y \ z))] \} \end{aligned}$$

now have a set of clauses,  
each clause a set of literals

# Resolution Theorem Proving

Any wff in FOPC (without equality)  
can be rewritten in clause form.  
(Davis, Putnam, 1960)

If the original is satisfiable, then  
the resultant clause form is satisfiable.

## Resolution Inference Rule

complementary literals

pair of literals

same predicate, one negated, one not  
arguments can be unified

Given 2 clauses with complementary literals

$c = [P(..), c_1, \dots, c_n]$  and  
 $c' = [\sim P(..), c'_1, \dots, c'_m]$

under unifying substitution  $S$

form resolvent  $R = [c_1, \dots, c_n, c'_1, \dots, c'_m]$

after applying substitution  $S$ .

# Unification

matching where both sides can have variables

unification is like matching except must check  
to see if either of elements is a variable

first rename variables to avoid unwanted constraints

is a linear process, go through arguments  
one at a time, unifying elements as go

## example

father(?x, tom) and father(harry, ?y)

these argument lists  
can be unified under the substitution

((?x . harry) (?y . tom))

unifier -- father(harry, tom)

# Resolution

## Examples

$$c = [P(x \ y \ a)]$$
$$c' = [\sim P(b \ z \ z), Q(s \ t \ z)]$$

then  $S = \{(x \ . \ b) (y \ . \ z) (z \ . \ a)\}$  and

$$R = [Q(s \ t \ a)]$$

(=== modus ponens)

$$c = [\sim M(x \ y \ c), P(x \ y \ a)]$$

$$c' = [\sim P(b \ z \ z), (Q \ s \ t \ z)]$$

then  $S = \{(x \ . \ b) (y \ . \ z) (z \ . \ a)\}$  and

$$R = [\sim M(b \ a \ c), Q(s \ t \ a)]$$

(=== transitivity of implication,  
chain rule)

# Resolution Theorem Proving

## Basic Proof Method (British Museum Algorithm)

To prove  $w$  logically follows from  $S$ ,

1. Form set of clauses corresponding to  
 $S \cup \sim w \quad == \quad S_{\text{current}}$
2. Until [null clause derived] (contradiction)
  - i) form all resolvents  $R$  from  $S_{\text{current}}$
  - ii) eliminate factors in new clauses  
(literals equivalent under unification)
  - ii) update  $S_{\text{current}}$  by  $R$

Herbrand Base  $H_s$

Replace clauses of  $S$  with substitutions  
for all variables for all values  
in respective domains (could be infinite).

If set of clauses  $S$  is unsatisfiable, then there  
exists a finite subset of  $H_s$  that is unsatisfiable.  
(Herbrand, 1930)

If set  $S$  is unsatisfiable, then application of  
finite number of steps based on unification  
and resolution will yield a contradiction.  
(Robinson, 1965)

# Resolution Theorem Proving

## Example

Marcus was a man.

1      man(Marcus)

Marcus was a Pompeian

2      Pompeian(Marcus)

All Pompeians were Romans.

3      A(x) [Pompeian(x) implies Roman(x)]

Caesar was a Roman ruler.

4      ruler(Caesar)

All Romans were loyal to Caesar or hated him.

5      A(x) [Roman (x) implies  
(loyalto(x Caesar) or hate(x Caesar))]

Everyone is loyal to someone.

6      A(x) E(y) [loyalto(x y)]

Men try to assassinate rulers to whom they are not loyal.

7      A(x) [A(y) [(man(x) and ruler(y))  
and trytoassassinate(x y))  
implies ~loyalto(x y)]]

Marcus tried to assassinate Caesar.

8      trytoassassinate(Marcus Caesar)

# Resolution Theorem Proving

Suppose want to prove: Marcus hated Caesar.

Goal: hate(Marcus Caesar)

## Step 1: Put 1-8 in clause form

- 1 man(Marcus)  $\Rightarrow$  { [man(Marcus)] }
- 2 Pompeian(Marcus)  $\Rightarrow$  { [Pompeian(Marcus)] }
- 3 A(x) (Pompeian(x) implies Roman(x))  
 $\Rightarrow$  { [ $\sim$ (Pompeian x), (Roman x)] }
- 4 ruler(Caesar)  $\Rightarrow$  { [ruler(Caesar)] }
- 5 A(x) ((Roman x) implies  
(loyalto(x Caesar) or hate(x Caesar))  
 $\Rightarrow$  { [ $\sim$ Roman(x), (loyalto(x Caesar),  
hate(x Caesar)) ] }
- 6 A(x) E(y) loyalto(x y)]  $\Rightarrow$  { [loyalto(x f(x))] }
- 7 A(x) A(y) ((man(x) and ruler(y)) and  
trytoassassinate(x y)) implies  $\sim$ loyalto(x y))  
 $\Rightarrow$  { [ $\sim$ man(x) ,  $\sim$ ruler(y),  
 $\sim$ trytoassassinate(x y),  $\sim$ loyalto(x y)] }
- 8 trytoassassinate(Marcus Caesar)  
{ [trytoassassinate(Marcus Caesar)] }

## Step 2: Negate Goal, and put in clause form:

G (negated)  $\Rightarrow$  { [ $\sim$ hate(Marcus Caesar)] }

# Resolution Theorem Proving

**Step 3:** Try to derive the null clause

5            G

{[ $\sim$ Roman(Marcus), loyalto(Marcus Caesar)] }

3

{[ $\sim$ Pompeian(Marcus), loyalto(Marcus Caesar)] }

2

{[loyalto(Marcus Caesar)] }

7

{[ $\sim$ man(Marcus),  $\sim$ ruler(Caesar),  
 $\sim$ trytoassassinate(Marcus Caesar)] }

1

{[ $\sim$ ruler(Caesar),  $\sim$ trytoassassinate(Marcus Caesar)] }

4

{[ $\sim$ trytoassassinate(Marcus Caesar)] }

8

□

# Resolution Theorem Proving

basic method == breadth first search  
(British Museum Algorithm)

efficiency improvements

Update Scurrent

Development

Means-ends Heuristics  
unit preference  
fewest literals

## Update Scurrent

Discard True Clauses  
complementary literals within a single clause

Discard Subsumptions  
discard clause C, if exists clause D such that  
D, under substitution, is a subset of C  
(C subsumes D)

# Resolution Theorem Proving

## Efficiency

restrict resolution application (select operators)

## Development

### 1. ancestry filtering

a clause used to form resolvent is either

- i) an input clause
- ii) a direct descendant of input clause
- iii) A and B, where A is ancestor of B

### 2. set of support

only resolve with clauses of goal  
or clauses derived from goal

### 3. vine form (input or linear resolution)

only i or ii of 1 above (incomplete method)

# Rule-Based Representation

one way to reduce complexity is to restrict language

## conjunctive rules

$A(\dots) [(P1(\dots) \text{ and } P2(\dots)\dots) \text{ implies } Pc(\dots)]$

a conjunction of positive literals  
(conditions, antecedents)  
implies a single positive literal (conclusion)

no functional terms  
only universal quantification (dropped)

## aka **Horn Clauses**

in clause form, consist of negative literals for antecedents and positive literal for conclusion

## **Rule-Based Knowledge Bases** **Expert Systems**

# Rule-Based Reasoning

Assume a set of facts S  
positive literals  
constant terms

determine what follows from S  
if w follows from S

forward chaining

backward chaining

## Forward Chaining

match conditions of a rule R to facts of S  
if succeed, add conclusion of R to S  
under uniform substitution  
given by matching

# Forward Chaining

## example

### rules

r1:  $\text{on}(x, y) \Rightarrow \text{above}(x, y)$

r2:  $\text{on}(x, y) \text{ and } \text{above}(y, z) \Rightarrow \text{above}(x, z)$

r3:  $\text{clear}(x) \text{ and } \text{above}(x, y) \Rightarrow \text{top}(x)$

### facts

$\text{on}(a, b)$

$\text{on}(b, c)$

$\text{clear}(a)$

## forward chaining

r1:  $(x.a)(y.b) \Rightarrow \text{above}(a,b)$

r1:  $(x.b)(y.c) \Rightarrow \text{above}(b,c)$

r2:  $(x.a)(y.b)(z.c) \Rightarrow \text{above}(a,c)$

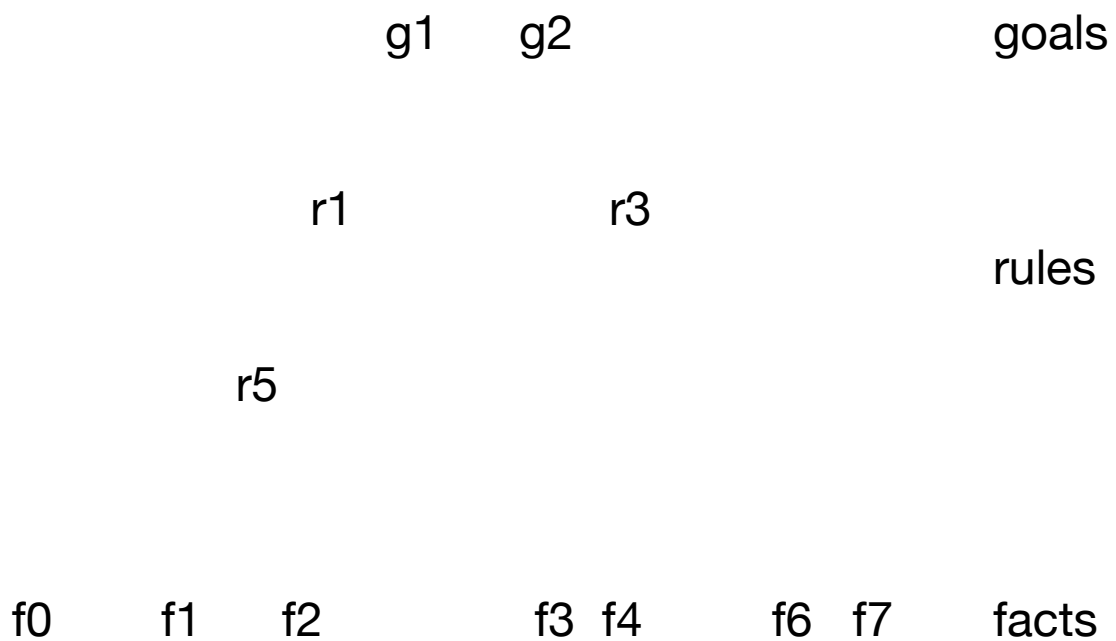
r3:  $(x.a)(y.b) \Rightarrow \text{top}(a)$

r3:  $(x.a)(y.c) \Rightarrow \text{top}(a)$  .... redundant

# Backward Chaining

given global rule base and a fact base,  
and a stack of goals

find a proof of the goals



place goals on a stack in some order

# Backward Chaining

Backward(stack)

```
if (stack is empty)
    return true
else let g be top goal on stack
    pop stack
    if FindFact(g, stack) return true
    else return FindRule(g, stack)
```

FindFact(g, stack)

```
For (all facts of fact base)
    if can match g with next fact f
        then if Backward((subst(stack))) return true
return false
```

FindRule(g, stack)

```
For (all rules of rule base)
    if can unify g with conclusion of next rule r
        then if Backtrack(pushall(subst(conditions(r)),
                                subst(stack))) return true
return false
```

subst applies new substitution to elements of argument

# Backward Chaining

## example

stack at each call to Backward

stack: top(a)

no facts  
unify with rule r3  
replace variables

stack: clear(a) above(a v1)

clear(a) matches fact

stack: above(a, v1)

no facts  
unify with rule 1

stack: on(a, v2)

matches fact on(a,b)

stack:

return true

# FOPC

now have the apparatus for reasoning

how do we represent knowledge in a domain

## Domain-specific Ontologies

$D_i$  object domains (sets of constants)

e.g., numbers, persons, products, stores

$P_i$  n-ary predicates

$D_1 \times D_2 \times \dots \times D_n \rightarrow \{\text{true}, \text{false}\}$

$f_i$  n-ary functions

$D_1 \times D_2 \times \dots \times D_n \rightarrow D_r$

$A_i$  domain axioms

basic inferences for the domain

# Ontologies

## example

people, family

object domains

person, integer, gender, date

predicates

parent(< person> < person>)

gender(< person> < gender>)

father(< person> < person>)

...

age(< person> < integer>)

older(< person> < person>)

birthday(< person> < date>)

functions

age-of(< person>) -> < integer>

axioms

$A(x,y)[((\text{parent}(x\ y) \text{ and } \text{gender}(x\ \text{male})) \text{ implies } \text{father}(x\ y))]$

$A(x, y)[\text{parent}(x, y) \text{ implies } \text{older}(x, y)]$

# Knowledge Engineering

defining an adequate ontology  
for a particular domain

adequacy defined with respect to  
questions one wants answered

## General Ontologies

measures

weight, volume, conversions

time

points, intervals

space

points, lines, planes, volumes, areas

actions/events/processes

situation calculus

objects

classes, subclasses, inheritance

mental objects/processes

ideas, thoughts, dreams, forgetting

# Ontology

## Actions

### situation calculus

Holds(wff, situation)

represent actions as functions on situations

Result(a, s)

situation that results from

performing action a in situation s

General form for operator/action representation:

$$A(\text{vars})[(\text{Holds}(\text{wff}, s) \text{ and } \dots) \quad \{\text{preconditions}\} \\ \Rightarrow (\text{Holds}(\text{wff}, \text{Result}(a, s)) \text{ and } \dots)] \\ \quad \{\text{effects}\}$$

where s is a situation and a is an action

## Example

$$A(x,y,s)[ \text{Holds}((\text{clear}(x) \text{ and } \text{clear}(y)),s) \\ \Rightarrow \text{Holds}((\text{on}(x,y) \text{ and } \text{clear}(x) \text{ and } \sim\text{clear}(y)), \\ \text{Result}(\text{put-on}(x,y), s))] ]$$

# Time

moments

zero duration

intervals

positive duration

given an I, Start(I) and End(I) moments

time scale

range and precision

of a sequence of time points

e.g., day in minutes, minute in seconds

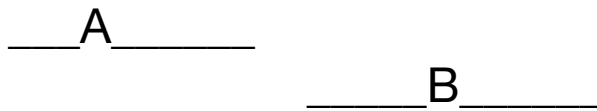
points related by  $<$ ,  $\leq$ ,  $=$ ,  $\geq$ ,  $>$

function

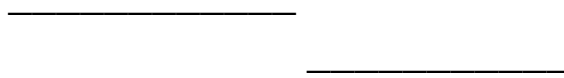
Time(moment)  $\rightarrow$  point on time scale

# Time

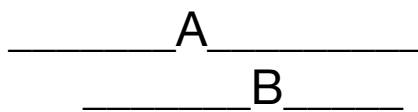
predicate, relations between time intervals



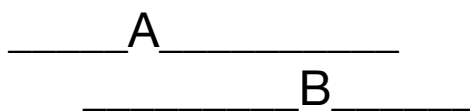
Before(A, B), After(B, A)



Meets(A, B)



During(B, A) {StartsWith, EndsWith, Same}



Overlap(A, B) , OverLap(B, A)

# Objects and Categories

## Structured Representations

Semantic Networks

Frames

general slot-filler structures  
(no implied ontology)

## Motivations

Computational

graph search  
compiled inference

Psychological

associations  
prototypicality

# Semantic Networks

representation of  
human associative memory  
human taxonomic knowledge

structured logic representation

binary relations

objects/concepts as nodes  
relations label arcs

(on a b)

on  
**a** -----> **b**

(red barn) ==> (has-color barn red)

has-color  
**barn** -----> **red**

# Semantic Networks

to represent more complex  
n-ary relations ( $n > 2$ )

create a relation token node  
and name each argument of  
relation with its case name

## Example

(buy-from tom mary clock 10)

### buy-from

|            |             |              |
|------------|-------------|--------------|
| is-a       |             | price        |
|            | <b>bf1</b>  | <b>10</b>    |
| buyer      | seller      | item         |
| <b>tom</b> | <b>mary</b> | <b>clock</b> |

# Semantic Networks

used to represent

conceptual hierarchies  
property inheritance

## Conceptual Hierarchy

is-a

element-of

subset-of

token-of

;; confusions

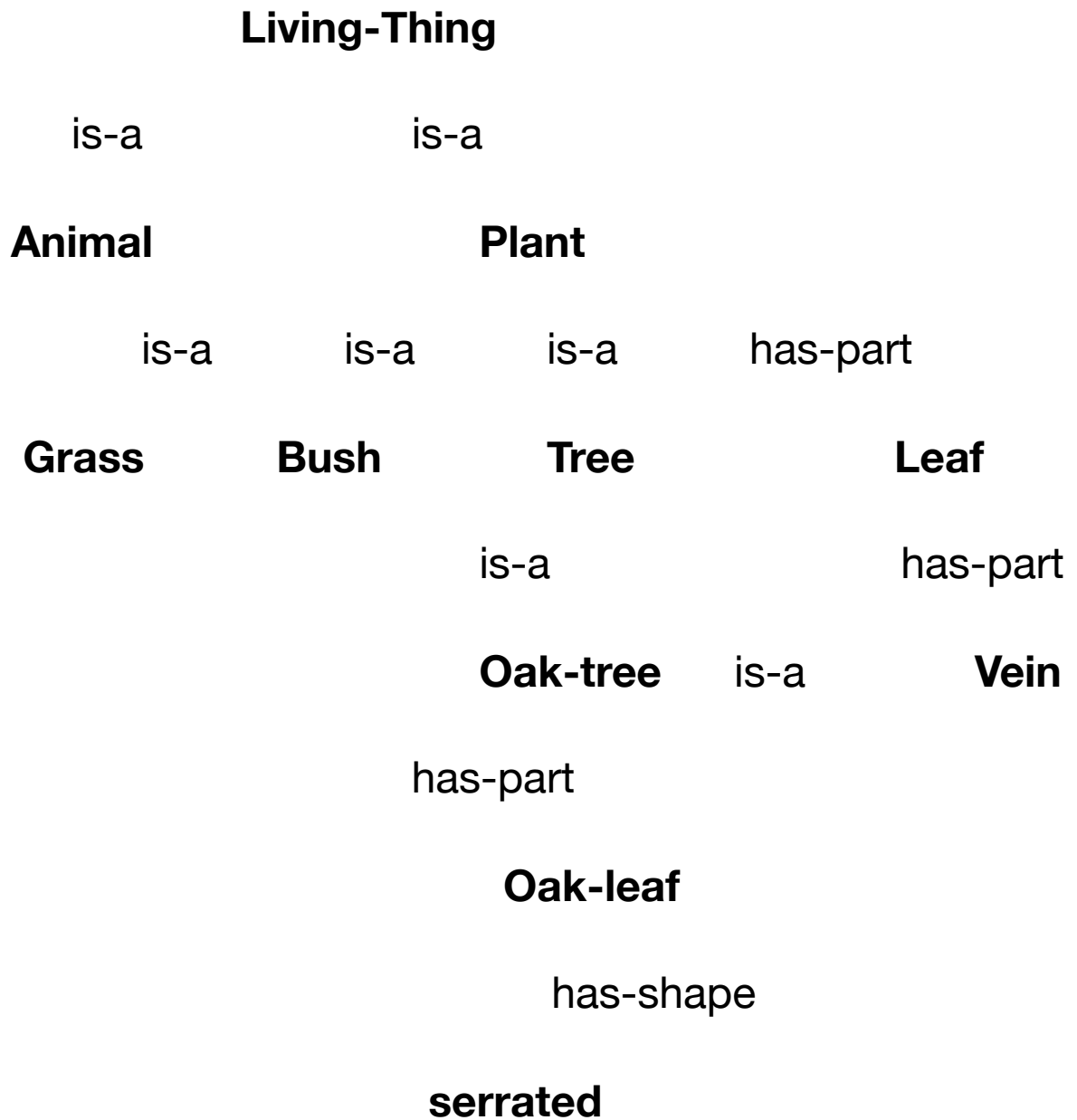
## Property Inheritance

has-part

has-feature

# Semantic Network

## Example



tree search and inheritance

# Frames

one view: extension of semantic networks  
procedural attachment

another: representation of  
common situations

objects / object structures  
events / event sequences

## Frame Features

### slots

each frame is a set of slots  
each representing a binary relation

each slot has set of facets  
representing value or  
modifiers of relation

# Frames

## Additional Features

### **procedural values**

specialized inference procedures  
for determining the value  
of particular slots

### **procedural attachment (demons)**

if-altered, if-added demons  
to-establish procedures

### **default slot values**

# Frames

## Example

```
(def-frame rectilinear-object
  (is-a (value (planar-object)))
  (height (default 100)
    (trigger
      (when-set
        (if height == width
          then (set-slot type square)
          else (set-slot type rect))))))
)
(width (default 100)
  (trigger
    (when-set
      (if height == width
        then (set-slot type square)
        else (set-slot type rect))))))
)
(type (default square))
(area (value (! (* height width))))))
```

# Frames

common event scenarios  
scripts

```
(def-frame restaurant-script
  (is-a (value (event-script)))
  (sequence
    (default
      (entering
        seating
        ordering
        eating
        paying
        leaving))))))
```

```
(def-frame entering
  (is-a (value (event)))
  (agent (default person))
  (time)
  (from-loc)
  (to-loc))
```

# Reasoning

## Semantic Nets

generalization/specialization  
hierarchy traversal

analogy  
associative distance  
and intersection

logical inference  
partitioned networks

## Frames

inheritance  
hierarchy traversal

situational matching  
completion by defaults

script matching  
exception notification