Reasoning and Knowledge Representation

Basic KR Hypothesis:

intelligent behavior is based upon the symbolic representation of beliefs and goals

Given:

Spike is a border collie.

Border collies are black and white.

I saw a black and white dog.

and processes for realizing their implications.

Spike is black and white.
I saw a border collie.  (maybe) ????????
Knowledge Representation and Reasoning

Desirable Properties

representational adequacy
  able to capture all needed knowledge
    (for a given domain)

inferential adequacy
  able to create and manipulate
    the necessary knowledge structures

inferential efficiency
  able to realize needed inferences in time

acquisitional adequacy
  able to incorporate new, relevant knowledge

acquisitional efficiency
  able to incorporate new knowledge in time
Reasoning and Knowledge Representation

Issues

ontology/granularity

primitive objects and relations
abstractions

what to infer

attention, focus, coherence
goal-dependence

when to infer

when changes occur
when results needed
Knowledge Representation

knowledge level
  truth or belief preserving entailment

symbolic level
  operational representation of knowledge level

Logical Approaches to Symbolic Level

Propositional Calculus
  facts

First-Order Predicate Calculus
  facts and generalizations
  purely syntactic (symbolic) systems
  sound and complete
  inference procedures

Issues

form of statements (syntax)
truth of statements (semantics)
rules of inference
notions of proof
Propositional Logic

must define

syntax
semantics

Syntax

<wff> (statement, sentence)
   == true | false | < symbol > | < complex-wff >

< complex-wff >
   == ~< wff > | (< wff > < connective > < wff >)

< connective >
   == and | or | implies

<symbol>
   == sequence of letters and digits (no connectives)

wff examples

(happy or sad) (a or (b and true))
(ant implies ~elephant)
Propositional Logic

Semantics

meanings for symbols (an interpretation)
symbols represent facts

logical meaning is true or false

in propositional logic,
an *interpretation* is an assignment
of true or false to each symbol

each proposition can correspond
da fact about a situation
its value indicates whether true or false
in that situation (holds in)

meanings for complex wffs are
based on meanings of logical connectives
given by their truth tables
(which you all know)
Propositional Logic

**model set** \( M_w \)

associated with a \(<\text{wff}\>\) \( w \)

is the set of interpretations

for which \( w \) is true

\[ s = (p \text{ and } q) \]

\[ M_s = \{(\text{true}, \text{true})\} \]

\[ s' = ((p \text{ or } q) \text{ and } q) \]

\[ M_{s'} = \{(\text{true}, \text{true}), (\text{false}, \text{true})\} \]

**validity**

\( w \) is valid iff \( w \) is true in every interpretation

\( w \) is valid iff \( M_w \) is all possible interpretations

neither \( s \) nor \( s' \) is valid

**satisfiability**

\( w \) is true in some interpretation

\( M_w \) is not empty

both \( s \) and \( s' \) are satisfiable
Propositional Logic

entailment

v entails w if, whenever v is true, w is true
v entails w if $M_v$ is a subset of $M_w$

s entails s'

equivalent to saying (v implies w) is valid

proof

that w is entailed by S (a set of wffs)

assume S (conjunction) as initial lines of proof

add new line of proof, by applying a rule of inference to one or more existing lines

until generate w

sound rule of inference

inference rule such that if v is true and w can be generated from v by applying the rule then is w true
Propositional Inference Rules

*modus ponens*

\[ a, (a \implies b) \implies b \]

*and-elimination*

\[ (a \land b) \implies a \]
\[ (a \land b) \implies b \]

*or-introduction*

\[ a \implies (a \lor b) \]
\[ a \implies (b \lor a) \]

*or-reduction*

\[ (p \lor \neg p) \implies \text{true} \]

*double-negation elimination*

\[ \neg\neg a \implies a \]

*distribution*

\[ (a \lor (b \land c)) \implies ((a \lor b) \land (a \lor c)) \]
\[ (a \land (b \lor c)) \implies ((a \land b) \lor (a \land c)) \]

*DeMorgan*

\[ \neg(a \lor b) \implies (\neg a \land \neg b) \]
\[ \neg(a \land b) \implies (\neg a \lor \neg b) \]

*reduction*

\[ (\text{true} \lor a) \implies \text{true} \]
\[ (\text{false} \lor a) \implies a \]
\[ (\text{true} \land a) \implies a \]
\[ (\text{false} \land a) \implies \text{false} \]

*implication definition*

\[ (a \implies b) \implies (\neg a \lor b) \]

*contradiction recognition*

\[ a, \neg a \implies \text{false} \]
Propositional Logic

to prove validity of a wff \( w \)

check truth table

assume \( \neg w \) and generate \textbf{false} by

a proof as defined above

\textit{proof by contradiction}

depth example

proof of \((p \implies p)\) -- by contradiction

\begin{align*}
1 & \neg(p \implies p) \quad \text{;; assumption} \\
2 & \neg(\neg p \lor p) \quad \text{;; implication definition (1)} \\
3 & (\neg\neg p \land \neg p) \quad \text{;; deMorgan rule (2)} \\
4 & (p \land \neg p) \quad \text{;; double negation elimination (3)} \\
5 & p \quad \text{;; and elimination (4)} \\
6 & \neg p \quad \text{;; and elimination (4)} \\
7 & \textbf{false} \quad \text{;; contradiction recognition (5) (6)}
\end{align*}
Propositional Reasoning

Natural Deduction

assume w

generate x by rules of inference

remove assumption w by introducing (w implies x)

example

proof of (p implies p)  -- natural deduction

1 p ;;; assumption
2 (p implies p) ;;; assumption removal (1)
Propositional Logic

proof

that w is entailed by S (set of wffs)

given S,
through application of inference rules to S
generate w

given S,
through application of inference rules to (S and ~w)
generate false

second is proof by contradiction
assume ~(S implies w)
Propositional Logic

proof by contradiction is basis for a standard proof technique

if $S$ entails $w$,
then $S$ and $\neg w$ is not satisfiable

if $S$ is satisfied by a model then so is $w$
by definition of entails,
and so $S$ and $\neg w$ is not satisfied

if $S$ is not satisfied by a model,
then $S$ and $\neg w$ is not satisfied

resolution theorem proving

turn $S$ into clause form
i.e., a conjunction of disjunctive clauses
turn $\neg w$ into clause form
show that this derives false
(i.e., the empty clause)
Propositional Logic

If the unicorn is mythical, then it is not mortal.
If the unicorn is not mythical,
    then it is a mortal mammal.
If the unicorn is not mortal or a mammal,
    then it is horned.
The unicorn is magical if it is horned.

$S =$
{ 1. (mythical implies ~mortal)
  2. (~mythical implies (mortal and mammal))
  3. ((~mortal or mammal) implies horned)
  4. (horned implies magical)  }

Want to prove that the unicorn is horned.

Show
  ((1 and 2 and 3 and 4) and ~horned)
  can be rewritten to false
Resolution Theorem Proving

((mythical implies ~mortal) and
(~mythical implies (mortal and mammal)) and
(~mortal or mammal) implies horned) and
(horned implies magical) and
~horned)

apply all implication definitions, and elimination

((~mythical or ~mortal)
((mythical) or (mortal and mammal))
((mortal and~mammal) or horned)
(~horned or magical)
(~horned)
)
Resolution Theorem Proving

do distribution of disjunctions

\[ \neg\text{mythical or } \neg\text{mortal} \quad (1) \]
\[ \text{mythical or mortal} \quad (2) \]
\[ \text{mythical or mammal} \quad (3) \]
\[ \text{mortal or horned} \quad (4) \]
\[ \neg\text{mammal or horned} \quad (5) \]
\[ \neg\text{horned or magical} \quad (6) \]
\[ \neg\text{horned} \quad (7) \]

new rule of inference

simple (unit) resolution

\[ (a), (\neg a \text{ or } b) \implies (b) \]
\[ (\neg a), (a \text{ or } b) \implies (b) \]

(derived from modus ponens
and implication definition)
Resolution Theorem Proving

do resolutions

(7), (5) ==> (~mammal) (8)

(7), (4) ==> (mortal) (9)

(9), (1) ==> (~mythical) (10)

(10), (3) ==> (mammal) (11)

(11), (8) ==> false

resolution method for propositional calculus
is complete and sound

general resolution rule of inference

$X = (x_1, ..., x_k, ..., x_n)$

$Y = (y_1, ..., y_l, ..., y_m)$

where $x_k = \neg y_l$

(i.e., complementary literals)

$Z = (x_1, ..., x_{k-1}, x_{k+1}, ..., x_n, y_1, ..., y_{l-1}, y_{l+1}, ..., y_{m})$

eliminate copies of other literals from $Z$

eliminate clause $Z$ with complementary literals
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 Logic (FOPL)

reasoning about predicates applied to objects
(relations)
allowing generalizations

Literals

\[ <\text{literal}> ::= <\text{predicate}>(<\text{term}_1> \ldots <\text{term}_n>) | \sim<\text{literal}> \]

Facts

literals, where terms are constants
ground literals

Examples

\begin{align*}
\text{red(ball)} & \quad <\text{color}>(<\text{object}>) \\
\text{on(ball box)} & \quad <\text{loc-rel}>(<\text{obj} > <\text{obj}>) \\
\text{age(Tom 37)} & \quad \text{age(<person > <pos-integer>)}
\end{align*}

often adopt such a "typed calculus" .... ontology
FOPL

General Statements

introduced through use of
variables
quantification

Quantification

Universal

\[ A(< \text{var}>)[< \text{wff}>] \]

where \(< \text{var} >\) is a variable,
possibly occurring in terms of \(< \text{wff} >\)

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

Existential

\[ E(< \text{var} >)[< \text{wff}>] \]

where \(< \text{var} >\) is a variable
possibly occurring in terms of \(<\text{wff} >\)

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

\[ A(var)[scope] \]
true if scope is true for every element
of domain of variable var

\[ E(var)[scope] \]
true if scope is true for at least one element
of domain of variable var
FOPL

functional terms

\[
< \text{term} > ::= \text{constant} \mid \text{variable} \mid \\
(\langle \text{function} > < \text{term} > ^ {\ast})
\]

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

maps object domain to object domain

skolem functions

used to replace existential quantifiers within scope of universal quantifiers

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

rewritten as
\[A(y)[\text{EQUAL}(f_{s}(y) \ (\text{plus} \ y \ 1))]\]

\[f_{s}(y) \Rightarrow \text{the value of } x \text{ for each } y \text{ that makes the scope true}\]

\[E(x)[A(y)[\text{EQUAL}(x \ (\text{plus} \ y \ 1))]] \Rightarrow \\
A(y)[\text{EQUAL}(a \ (\text{plus} \ y \ 1))], \text{ where } a \text{ is } f_{s}()\]
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), \ldots\}
FOPL

model

interpretation such that wff(s) is true

satisfiability

existence of a model

validity

wff satisfied by all interpretations (models)

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

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
FOPL

Theorem Proving

Computational Issues

complexity of wff syntax

equivalence of wffs

\neg (A \lor B) \iff (\neg A \land \neg B)

(A \rightarrow B) \iff (\neg A \lor B)

A(x)[\neg W] \iff \neg E(x)[W]

\neg A(x)[W] \iff E(x)[\neg W]

multiple rules of inference

modus ponens
chain rule
quantification
    introduction/elimination
FOPL

Theorem Proving

How can we deal with the above complications?

eliminate connectives

eliminate quantification

Clause Normal Form

use single rule of inference

Resolution Theorem Proving
Clause Normal Form

\[ E(x) \ (P(x) \implies A(y) \ A(z) \ E(t) \ Q(x \ y \ z \ t)) \]

**Step 1:** eliminate implications

\[(A \implies B) \implies (\neg A \lor B)\]

\[ E(x) \ [P(x) \implies A(y) \ A(z) \ E(t) \ [Q(x \ y \ z \ t)]] \implies E(x) \ [
eg P(x) \lor A(y) \ A(z) \ E(t) \ [Q(x \ y \ z \ t)]] \]

(using \implies as application of rule of inference)

**Step 2:** reduce scope of negation

(move next to literals)

using

\[ \neg \neg W \implies W \]

\[ \neg E(x)(...) \implies A(x)\neg(\ldots) \]

\[ \neg A(x)(...) \implies E(x)\neg(\ldots) \]

\[ \neg(\ldots \text{ or } \ldots) \implies (\neg(\ldots) \text{ and } \neg(\ldots)) \]

\[ \neg(\ldots \text{ and } \ldots) \implies (\neg(\ldots) \text{ or } \neg(\ldots)) \]
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)

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

\[ \Rightarrow E(x) \ A(y) \ A(z) \ E(t)[\neg P(x) \text{ or } Q(x \ y \ z \ t)] \]

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)

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

\[ \Rightarrow A(y) \ A(z)[\neg P(a) \text{ or } Q(x \ y \ z \ f_5(y \ z))] \]
Clause Normal Form

Step 6: drop universal quantification

\[ A(y) \land A(z)[\neg P(a) \lor Q(x, y, z, f_s(y, z))] \]
\[ \implies (\neg P(a) \lor Q(x, y, z, f_s(y, z))) \]

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

\[ ((w \land w') \lor z) \implies ((w \lor z) \land (w' \lor z)) \]

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

\[ (\neg P(a) \lor Q(x, y, z, f_s(y, z))) \]
\[ \implies \{ [\neg P(a), Q(a, y, z, f_s(y, z))] \} \]

now have a set of clauses,
each clause a set of literals
Resolution Theorem Proving

Any wff in FOPL (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

\[ c = [P(..), c_1, ..., c_n] \text{ and } c' = [\neg P(..), c_1', ..., c_m'] \]

under *unifying substitution* \( S \)

form resolvent \( R = [c_1, ..., c_n, c_1', ..., c_m'] \)

after applying substitution \( S \).
Unification

matching where both sides can have variables

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

first rename variables to avoid unwanted constraints

is a linear process, go through terms one at a time, unifying 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' = \[ \neg P(b \ z \ z), \ Q(s \ t \ z) \]

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

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

(== modus ponens)

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

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

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

(== 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
   \[ S \cup \neg w \implies \text{Scurrent} \]
2. Until [null clause derived] (contradiction)
   i) form all resolvents \( R \) from \( \text{Scurrent} \)
   ii) eliminate factors in new clauses
       (literals equivalent under unification)
   ii) update \( \text{Scurrent} \) by \( R \)

Herbrand Base \( Hs \)

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

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

\( G \ (\text{negated}) \Rightarrow \{[\neg \text{hate(Marcus Caesar)]} \} \)
Resolution Theorem Proving

Step 3: Try to derive the null clause

5  G
   {[~Roman(Marcus), loyalto(Marcus Caesar)] }

3
   {[~Pompeian(Marcus), loyalto(Marcus Caesar)] }

2
   {[loyalto(Marcus Caesar)] }

7
   {[~man(Marcus), ~ruler(Caesar),
      ~trytoassassinate(Marcus Caesar)] }

1
   {[~ruler(Caesar), ~trytoassassinate(Marcus Caesar)]}

4
   {[~trytoassassinate(Marcus Caesar)] }

8
   []
Resolution Theorem Proving

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

efficiency improvements

Update Scurrent (as in breadth-first search)

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**

\[ A(....) \land [P_1(...) \land P_2(...)\ldots] \implies P_c(...)] \]

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

eexample

rules
r1: on(x, y) => above(x, y)

r2: on(x, y) and above(y, z) => above(x, z)

r3: clear(x) and above(x, y) => top(x)

facts
on(a, b)
on(b, c)
clear(a)

forward chaining
r1: (x.a)(y.b) => above(a,b)
r1: (x.b)(y.c) => above(b,c)
r2: (x.a)(y.b)(z.c) => above(a,c)
r3: (x.a)(y.b) => top(a)
r3: (x.a)(y.c) => top(a) .... redundant
Backward Chaining

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

find a proof of the goals

g1    g2    goals

r1    r3    rules

r5

f0    f1    f2    f3    f4    f6    f7    facts

place goals on a stack in some order
**Backward Chaining**

BackwardChain(goals, b)
   if (goals is empty)
      return b
   else     result = FindFact(goals, b)
      if result not fail then return result
      else return FindRule(goals, stack)

FindFact(goals, b)
   Let g be First(goals)
   For (all facts f of fact base)
      if can match g and f under bindings b creating b’
      then result = BackwardChain(Rest(goals), b’)
      if result not fail return result
   return fail

FindRule(goals, stack)
   Let g be First(goals)
   For (all rules r of rule base)
      if can unify g and conclusion of r with new variables
         under bindings b creating b’
      then result = BackwardChain(pushall(conditions(r), Rest(goals)))
      if result not fail then return result
   return fail
Backward Chaining

rules
r1: on(x, y) => above(x, y)
r2: on(x, y) and above(y, z) => above(x, z)
r3: clear(x) and above(x, y) => top(x)

facts
on(a, b)
on(b, c)
clear(a)

stack at each point

stack: top(a)  no facts
        unify with rule r3
        v0 = a

stack: clear(v0) above(v0 v1)  clear(a) matches fact

stack: above(v0, v1)  no facts
        unify with rule 1
        v2 = a

stack: on(v2, v3)  matches fact on(a, b)
        v3 = b

stack:

return {(v0, a), (v2, a), (v3, b)}  ==> {}  no variables in query
FOPC

now have the approaches to 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

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

$f_i$ n-ary functions

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

$A_i$ 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)[((parent(x y) and gender(x male))
implies father(x y))]
A(x, y)[parent(x, y) implies 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
**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
$\text{Time(moment)} \rightarrow \text{point on time scale}$
Time

predicate, relations between time intervals

___A_______    _____B_______

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

_____________

_____________

Meets(A, B)

________A__________

_______B_______

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

_______A___________

_______B_______

Overlap(A, B)  ,  Overlap(B, A)
Ontology

Actions

situation calculus

Holds(wff, situation)

represent actions as functions on situations

Result(action, situation) => situation
situation that results from
performing action a in situation s

General form for operator/action representation:

A(vars)[(Holds(wff, s) and .... ) {preconditions}
  => (Holds(wff, Result(a, s)) and ...)]
  {effects}
where s is a situation and a is an action

Example

A(x,y,s)[ Holds((clear(x) and clear(y)),s))
  ==>  Holds((on(x,y) and clear(x) and ~clear(y)),
              Result(put-on(x,y), s))]
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

<table>
<thead>
<tr>
<th>is-a</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>buy-from</td>
<td>10</td>
</tr>
</tbody>
</table>

buyer       seller       item

tom        mary        clock
Semantic Networks

used to represent

- conceptual hierarchies
- property inheritance

Conceptual Hierarchy

- is-a
- element-of
- subset-of
- ;; confusions
- token-of

Property Inheritance

- has-part
- has-feature
Semantic Network

Example

Living-Thing

is-a is-a

Animal
Plant

is-a is-a is-a has-part

Grass Bush Tree Leaf

is-a has-part

Oak-tree is-a Vein

has-part

Oak-leaf

has-shape

serrated

tree search and inheritance
Frames

extension of semantic networks

representation of common situations

situations / object structures
scripts / event sequences

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

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

scripts --- common event scenarios

(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