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.

Many black and white dogs are border collies.

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 domain knowledge

representational efficiency
can represent in reasonably compact manner

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, axioms

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 Logic
  facts

First-Order Predicate Logic
  facts, relations 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
  proof algorithms
Propositional Logic

must define

syntax
semantics

Syntax

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

< complex-wff >
  == (not < 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 (not elephant))
Propositional Logic

Semantics

meanings for symbols (an interpretation)

in propositional logic,
symbols represent assertions
logical meaning is true or false

an interpretation is an assignment
of true or false to each symbol

each symbol corresponds to
a 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
Propositional Logic

**model set** \( M_w \)

associated with a <wff> w
is the set of interpretations
for which w is true

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

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

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

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

**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 $M_w$

equivalent to saying (v implies w) is valid

s entails s'

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

sound 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
Selected 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*
\[ a \implies (a \lor b) \]
\[ a \implies (b \lor a) \]

*double-negation elimination*
\[ (\lnot (\lnot a)) \implies a \]

*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*
\[ (\lnot (a \lor b)) \implies ((\lnot a) \land (\lnot b)) \]
\[ (\lnot (a \land b)) \implies ((\lnot a) \lor (\lnot b)) \]

*reduction*
\[ (\text{true or } a) \implies \text{true} \]
\[ (\text{false or } a) \implies a \]
\[ (\text{true and } a) \implies a \]
\[ (\text{false and } a) \implies \text{false} \]

*implication definition*
\[ (a \implies b) \implies ((\lnot a) \lor b) \]

*contradiction recognition*
\[ (a \land (\lnot a)) \implies \text{false} \]

*truth recognition*
\[ (a \lor (\lnot a)) \implies \text{true} \]
Propositional Logic

to prove validity of a wff $w$

check truth table

assume $\sim w$ and generate $\text{false}$ by

a proof as defined above

proof by contradiction

using $\sim \text{wff} == (\text{not wff})$

example

proof of $(p \implies p) \quad \text{-- by contradiction}$

1 $\sim(p \implies p)$ $;;;;$ assumption
2 $\sim(\sim p \lor p)$ $;;;;$ implication definition (1)
3 $(\sim \sim p \text{ and } \sim p)$ $;;;;$ deMorgan rule (2)
4 $(p \text{ and } \sim p)$ $;;;;$ double negation elimination (3)
5 $\text{false}$ $;;;;$ contradiction recognition (4)
Propositional Reasoning

Natural Deduction

assume w

generate x by rules of inference

remove assumption w by introducing (w implies x)

dexample

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 \( (S \text{ and } \sim w) \)
generate \textbf{false}

second is proof by contradiction
assume \( \sim(S \text{ implies } w) \)
Propositional Logic

proof by contradiction is a standard proof technique

if S entails w,
then (S and ~w) is not satisfiable

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

if S is not satisfied by a model,
then S and ~w is not satisfied

resolution theorem proving

turn S into clause form
i.e., a conjunction of disjunctive clauses

turn ~w into clause form
show that this allows one to derive false
(i.e., the empty clause)
Resolution Theorem Proving

turn S into clause form
i.e., conjunctive normal form
turn ~w into clause form
show that this allows one to derive false
i.e., the empty clause
using generalized resolution rule

Turn S into Clause Form

implication definition
demorgans rule
double-negation elimination
or distribution

(not (a implies (b and c)))
(not ((not a) or (b and c)))
((not (not a) and (not (b and c)))
((not (not a) and ((not b) or (not c)))
(a and ((not b) or (not c)))
===> yields 2 clauses
   [(a) ((not b) (not c))]

(a implies (b and c))
((not a) or (b and c))
(((not a) or b) and ((not a) or c))
===> [((not a) b) ((not a) c)]
Resolution Theorem Proving

apply single rule of inference

general resolution rule of inference

\[ X = (x_1, \ldots, x_k, \ldots, x_n) \]
\[ Y = (y_1, \ldots, y_l, \ldots, y_m) \]

where \( x_k = \sim y_l \)

(i.e., complementary literals)

\[ Z = (x_1, \ldots, x_{k-1}, x_{k+1}, \ldots, x_n, y_1, \ldots, y_{l-1}, y_{l+1}, \ldots, y_{m}) \]

eliminate copies of other literals from \( Z \)
eliminate clause \( Z \) with complementary literals

Derive empty clause
Example from prior page

1 (a)
2 ((not b) (not c))
3 ((not a) b)
4 ((not a) c)
5 (b) 1,3
6 (c) 1,4
7 ((not c)) 5,2
8 () 7,6
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 derive ( ) false
Resolution Theorem Proving

\[
\begin{align*}
&((\text{mythical implies } \neg\text{mortal}) \\text{and} \\
&\quad (\neg\text{mythical implies (mortal and mammal)}) \\text{and} \\
&\quad (\neg\text{mortal or mammal) implies horned}) \\text{and} \\
&\quad (\text{horned implies magical}) \\text{and} \\
&\quad \neg\text{horned})
\end{align*}
\]

apply all implication definitions

\[
\begin{align*}
&((\neg\text{mythical or } \neg\text{mortal}) \\
&\quad ((\text{mythical}) \text{ or (mortal and mammal)}) \\
&\quad ((\text{mortal and } \neg\text{mammal) or horned}) \\
&\quad (\neg\text{horned or magical}) \\
&\quad (\neg\text{horned}) \\
\end{align*}
\]
Resolution Theorem Proving

do distribution of disjunctions

\[
\begin{align*}
((\neg \text{mythical} \ \neg \text{mortal}) & \ (1) \\
(\text{mythical} \ \text{mortal}) & \ (2) \\
(\text{mythical} \ \text{mammal}) & \ (3) \\
(\text{mortal} \ \text{horned}) & \ (4) \\
(\neg \text{mammal} \ \text{horned}) & \ (5) \\
(\neg \text{horned} \ \text{magical}) & \ (6) \\
(\neg \text{horned}) & \ (7) \\
\end{align*}
\]

resolution rule of inference

simple (unit) resolution

\[
\begin{align*}
(a), \ (\neg a \ \text{or} \ b) & \implies (b) \\
(\neg a), \ (a \ \text{or} \ b) & \implies (b) \\
\end{align*}
\]

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

do resolutions

\[(7), (5) \implies \neg \text{mammal} \quad (8)\]

\[(7), (4) \implies \text{mortal} \quad (9)\]

\[(9), (1) \implies \neg \text{mythical} \quad (10)\]

\[(10), (3) \implies \text{mammal} \quad (11)\]

\[(11), (8) \implies \text{false}\]

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
Resolution Theorem Proving

refutation complete

to show \( S \implies x \)

by breadth-first search
show that \( S \cup \{\neg x\} \) leads to the empty clause

if exists, must be found at some finite depth

Horn Clause Reasoning

reasoning in worlds where knowledge has form

\[ a_1 \text{ and } a_2 \text{ and } \ldots a_m \implies c \]

horn clause is \((\neg a_1, \neg a_2, \ldots, \neg a_m, c)\)

positive unit clauses (facts) constitute state knowledge

want to know if some other positive literal follows
Horn Clause Reasoning

consider a known, positive unit clause a fact

Forward Chaining

data driven... given goal fact
have an “agenda” of facts
still to be processed
to process a fact, take it off the agenda
and remove it
from antecedents of horn clauses
if a horn clause had no antecedents,
consider the conclusion a new fact
when conclude a new fact,
if it is the goal fact, then return true
else add it to the agenda
if agenda is empty, return false

given (a,b => c) (c, d => e) (b, c => d) (f => e)
given a,b as facts  => c (with b) => d (with c) => e
Horn Clause Reasoning

consider a known, positive unit clause a fact

Backward Chaining

push desired conclusion to goal stack

Repeat until (goal stack empty)

    if top of goal stack is a fact, pop goal stack

    else if (are horn clauses that conclude top of stack)
        pop goal stack
        for each such horn clause h
            create a new stack with the antecedents
            of h pushed on top of goal stack
            select one of the new stacks as goal stack

    else if (exist other stacks)
        select one of the other stacks as goal stack

    else return false

return true

propositional PROLOG interpreter
Predicate Logic

propositional logic can represent any finite situation context

we have a sound, complete proof method for propositional logic

What is problem?

can only reason about facts.. propositions

((democratAndy and republicanJane) implies moreConservativeJaneAndy)

if have a million people need a quadrillion $(10^{12})$ rules to capture the general idea

there are no objects or relations

no generalizations
First-Order Predicate Logic (FOPL)

reasoning about predicates applied to objects
allowing generalizations

Literals

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

Facts

literals, where terms are all constants

\text{ground literals}

Examples

has-color(ball, red) \hspace{2cm} \text{has-color(< object >, <color>)}
on(ball, box) \hspace{2cm} \text{on(< obj >, < obj >)}
age(Tom, 37) \hspace{2cm} \text{age(< person >, < pos-integer >)}
democrat(Tom) \hspace{2cm} \text{democrat(<person>)}

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

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

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

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

\(f_s(y)\) => the value of \(x\) for each \(y\) that makes the scope true

\(E(x)[A(y)[\text{EQUAL}(x, (\text{plus}\ y\ 1))]]\)  ==>  
\(A(y)[\text{EQUAL}(a, (\text{plus}\ y\ 1))],\)

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), \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

\(~(A \text{ or } B) \iff (~A \text{ and } \neg B)\)

\((A \implies B) \iff (~A \text{ or } B)\)

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

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

multiple rules of inference

modus ponens
chain rule
quantification
introduction/elimination
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) ==> (~A or B)

E(x) [P(x) implies A(y) A(z) E(t) [Q(x, y, z, t)]]  ==>  E(x) [~P(x) or A(y) A(z) E(t) [Q(x, y, z, t)]]

(using ==> as application of rule of inference)

Step 2: reduce scope of negation

(move next to literals)

using

~~W ==> W

~E(x)(...) ==> A(x)~(...)

~A(x)(...) ==> E(x)~(...)

~( ... or ...) ==> (~(... and ~(...))

~( ... and ...) ==> (~(... or ~(...))
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)

\[
E(x) \left( \neg P(x) \lor A(y) A(z) E(t)[Q(x, y, z, t)] \right)
\]
\[\Rightarrow E(x) A(y) A(z) E(t)[\neg P(x) \lor Q(x, y, z, t)]\]

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)

\[
E(x) A(y) A(z) E(t)(\neg P(x) \lor Q(x, y z t))
\]
\[\Rightarrow A(y) A(z)[\neg P(a) \lor Q(x y z f_s(y z))]\]
Clause Normal Form

**Step 6:** drop universal quantification

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

**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)

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

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

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

under unifying substitution \( S \)

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

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 argument 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)
Unification algorithm

\texttt{unify(x, y, env) ;; returns an environment or ‘fail}

\begin{align*}
\text{if env} &= \text{‘fail, then return ‘fail} \\
\text{else if (x = y) then return env} \\
\text{else if variable?(x) return unify-var(x, y, env)} \\
\text{else if variable?(y) return unify-var(y, x, env)} \\
\text{else if (list?(x) and list?(y)) then} \\
&& \text{return unify(rest(x), rest(y), unify(first(x), first(y), env))} \\
\text{else return ‘fail}
\end{align*}

\texttt{unify-var(x, y, env)}

\begin{align*}
\text{if bound?(x, env) then return unify(val(x, env), y, env)} \\
\text{else if (variable?(y) and bound?(y, env))} \\
&& \text{return unify(x, val(y, env), env)} \\
\text{else if occurs?(x, y) return ‘fail} \\
\text{else return addbinding((x . y), env)}
\end{align*}
Unification

examples

\[ p(x, a) \quad p(b, a) \quad p(b, a) \]
\[ ((x \ . \ b)) \]

\[ p(x, b) \quad p(b, a) \quad \text{fail} \]

\[ p(a, x) \quad p(y, f(y)) \quad p(a, f(a)) \]
\[ ((y \ . \ a) \ (x \ . \ f(y))) \]

\[ p(a, x) \quad p(y, z) \quad p(a, x) \]
\[ ((y \ . \ a) \ (x \ . \ z)) \]

\[ p(x, x) \quad p(a, z) \quad p(a, a) \]
\[ ((x \ . \ a) \ (z \ . \ a)) \]
Resolution

general resolution rule of inference

\[ X = (x_1, ..., x_k, ..., x_n) \]
\[ Y = (y_1, ..., y_l, ..., y_m) \]

where \( x_k = \sim y_l \) (i.e., complementary literals)

complementary literals are those having
opposite sign
same predicate
argument lists that can
be unified with bindings ENV

resolvent Z is

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

where all \( x_i \) and \( y_j \) have variables substituted
with values from ENV

eliminate duplicate literals (factors) from Z
eliminate clause Z with complementary literals
under unification
Resolution

Examples

c= \[P(x \ y \ a)\]
c'= \[\sim P(b \ z \ z), \ Q(d \ e \ z)\]

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

\(R = [Q(d \ e \ a)]\)

(== modus ponens)

\(c= \sim M(x \ y \ c), \ P(x \ y \ a)\]
c'= \[\sim P(b \ z \ z), \ Q(d \ w \ z)\]

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

\(R = [\sim M(b \ a \ c), \ Q(d \ w \ a)]\)

(== 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
   \( S \cup \sim w \Rightarrow \text{Scurrent} \)
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 \( \text{Hs} \)

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

If set of clauses \( S \) is unsatisfiable, then there
exists a finite subset of \( \text{Hs} \) 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

Results by Robinson

lifting lemma

Let $C_1$ and $C_2$ be two clauses with no shared variables and let $C_1'$ and $C_2'$ be ground instances of $C_1$ and $C_2$. If $C'$ is a resolvent of $C_1'$ and $C_2'$, then there exists a clause $C$ such that (1) $C$ is a resolvent of $C_1$ and $C_2$ and (2) $C'$ is a ground instance of $C$.

Unification finds the most general unifier... so $C$ has the most ground instances possible.

Proof in generalized clauses is of same length as in ground clauses.
Resolution Theorem Proving

Example

Marcus was a man.
1 \text{man}(Marcus)

Marcus was a Pompeian
2 \text{Pompeian}(Marcus)

All Pompeians were Romans.
3 \forall x \left[ \text{Pompeian}(x) \implies \text{Roman}(x) \right]

Caesar was a Roman ruler.
4 \text{ruler}(Caesar)

All Romans were loyal to Caesar or hated him.
5 \forall x \left[ \text{Roman}(x) \implies \left( \text{loyalto}(x, \text{Caesar}) \text{ or } \text{hate}(x, \text{Caesar}) \right) \right]

Everyone is loyal to someone.
6 \forall x \ \exists y \left[ \text{loyalto}(x, y) \right]

Men try to assassinate rulers to whom they are not loyal.
7 \forall x \left[ \forall y \left[ \left( \text{man}(x) \text{ and } \text{ruler}(y) \right) \text{ and } \text{trytoassassinate}(x, y) \right) \implies \neg \text{loyalto}(x, y) \right] \right]

Marcus tried to assassinate Caesar.
8 \text{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)  =>  { [man(Marcus)] }
2  Pompeian(Marcus)  =>  { [Pompeian(Marcus)] }
3  A(x) (Pompeian(x) implies Roman(x))]
    =>  { [~(Pompeian x), (Roman x)] }
4  ruler(Caesar)  =>  { [ruler(Caesar)] }
5  A(x) ((Roman x) implies
       (loyalto(x Caesar) or hate(x Caesar))
       =>  {[~Roman(x), (loyalto(x Caesar),
                   hate(x Caesar)] }
6  A(x) E(y) loyalto(x y)]]  =>  { [loyalto(x f(x))] }
7  A(x) A(y) ((man(x) and ruler(y)) and
       trytoassassinate(x y)) implies ~loyalto(x y))
       =>  {[~man(x) , ~ruler(y),
                ~trytoassassinate(x y), ~loyalto(x y)] }
8  trytoassassinate(Marcus Caesar)
       { [trytoassassinate(Marcus Caesar)] }

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

G (negated)  =>  {[~hate(Marcus Caesar)] }
Resolution Theorem Proving

Step 3: Try to derive the null clause

5

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

3

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

2

\{[\text{loyalto(Marcus Caesar)}] \}

7

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

1

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

4

\{[\sim \text{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

\[ A(\ldots) \ [(P_1(\ldots) \text{ and } P_2(\ldots)\ldots) \text{ implies } P_c(\ldots)] \]

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 a 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 $S$

前进链式

后退链式

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: 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

\[
\begin{align*}
g1 & \quad g2 & \quad \text{goals} \\
r1 & \quad r3 & \quad \text{rules} \\
r5 & \\
f0 & \quad f1 & \quad f2 & \quad f3 & \quad f4 & \quad f6 & \quad f7 & \quad \text{facts}
\end{align*}
\]

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(a) above(a v1) clear(a) matches fact
stack: above(a, v1) no facts
unify with rule 1
v2 = a
stack: on(a, v3) matches fact on(a,b)
v3 = b

stack:
return {(v0,a),(v1,v3),(v2,a),(v3,b)} = {}
no variables in query