B.Y. Choueiry

Title: First-Order Logic
AIMA: Chapter 7 (Sections 7.4 and 7.9)

Introduction to Artificial Intelligence
CSCE 476-876, Spring 2003
URL: `www.cse.unl.edu/~choueiry/S03-476-876`

Berthe Y. Choueiry (Shu-we-ri)
`choueiry@cse.unl.edu, (402)472-5444`

Summary

- First-order logic:
  - basic elements
  - syntax
  - semantics
- Examples

Outline

Wumpus world in FOL

- Situation calculus
  - frame problem
  - qualification problem
  - ramification problem
- Simple reflex agent: pure reflex + internal state
- Model-based agent
- Goal-based agent

Logical Agents for the Wumpus World (I)

Define an interface between the environment and the agent

Perceptual sentence $\begin{cases} \text{percept} \\ \text{time at which it occurred} \end{cases}$

Percepts: Stench, Breeze, Glitter, Bump, Scream

Perceptual sentence at time step 5:

$$Percept([Stench, Breeze, Glitter, None, None], 5)$$

Interfacing with FOL KBs

function KB-AGENT(*percept*) **returns** an *action*
  static: *KB*, a knowledge base
    t, a counter, initially 0, indicating time

  TELL(*KB*, MAKE-PERCEPT-SENTENCE(*percept*, *t*))
  action ← ASK(*KB*, MAKE-ACTION-QUERY(*t*))
  TELL(*KB*, MAKE-ACTION-SENTENCE(*action*, *t*))
  t ← *t* + 1
  return *action*

- $Tell(KB, Percept([Smell, Breeze, Glitter, None, None], 5))$
- Does the KB entail any particular actions at $t = 5$?
  Make-action-query at $t = 5$: $Ask(KB, \exists\ a\ Action(a, 5))$
  Ask returns a binding list $\{a/Grab, a/..., a/...\}$
- $Tell$ informs KB which action is taken
- If σ is a possible binding of variables
  $Ask(KB, S)$ returns some/all σ such that $KB \models S\sigma$

Representing change in the world

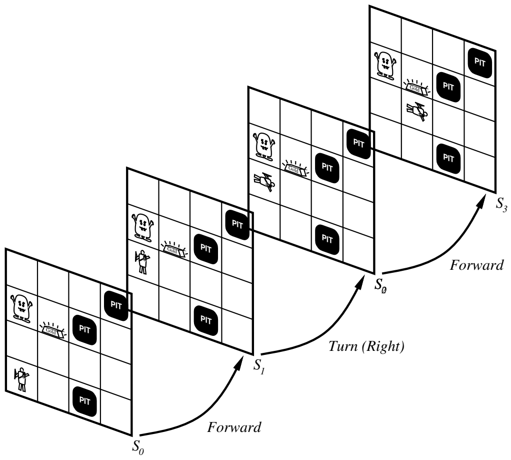
All percepts are added to KB
Percepts history provides a representation of the world
$\longrightarrow$ design rules to infer <u>actions</u> from past and present percepts

Problem: becomes tedious (big number of perceptual sentences)
Solution: maintain an internal model of the world
Problem: the world changes

Situation calculus:
— adds a situation argument to each non-eternal predicate
  e.g., $Holding(Gold, Now)$ rather than just $Holding(Gold)$
— world is represented as a sequence of situations
— world evolves according to **diachronic** rules

Situation calculus (I)

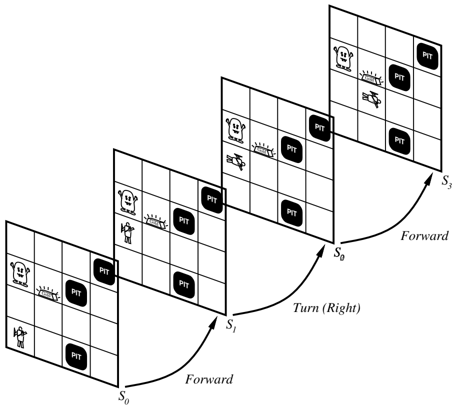
Each situation is a snapshot of the state of the world
Situations are generated from previous ones by <u>actions</u>



- Relations and properties that are situation dependent:
  $At(Agent, [1,1], S_0) \wedge At(Agent, [1,2], S_1)$
- Relations and properties not subject to change: $Wall[0,1]$

Situation calculus (II)

S_{i+1} resulting from applying *Action* to S_i:
$Result(Action, S_i) = S_{i+1}$



Situations are connected by the *Result* function:

$Result(Forward, S_o) = S_1$

$Result(TurnRight, S_1) = S_2$

etc.

Situation calculus (III)

Actions: described by stating their effects
Effects: properties of the new situation

Effect axiom: describe changes due to action

$\forall\, s\ AtGold(s) \Rightarrow Holding(Gold, Result(Grab, s))$

$\forall\, x, s\ \neg Holding(x, Result(Release, s))$

Frame axiom: describe non-changes due to action

$\forall\, a, x, s\ \ Holding(x, s) \wedge (a \neq Release) \Rightarrow Holding(x, Result(a, s))$

$\forall\, s\ \ HaveArrow(s) \Rightarrow HaveArrow(Result(Grab, s))$

Situation calculus (IV)

Effect axioms + Frame axioms:

provide a complete description of how the world evolves
in response to the agent's actions
can be combined in a single axiom..

Successor-state axiom

$$\forall\, a, x, s\ \ Holding(x, Result(a, s)) \Leftrightarrow \quad (Holding(x, s) \wedge (a \neq Release)) \vee (AtGold(s) \wedge a = Grab)$$

→ needed for every predicate that may change its value over time

→ lists all the ways in which the predicate can become $\begin{cases} \text{true} \\ \text{false} \end{cases}$

Situation calculus (V) Famous problems

Frame : find an elegant way to handle non-change
1– representation: avoid frame axioms (successor-state axioms)
2– inference: avoid repeated "copy-overs" to keep track of state

Qualification : difficulty of defining the circumstances under which an action is guaranteed to work
true descriptions of real actions require endless caveats: what if gold is slippery or nailed down or . . .

Ramification : proliferation of implicit consequences of actions
real actions have many secondary consequences—what about the dust on the gold. . .

Logical Agents for the Wumpus World (II)

Three architectures:

1. Reflex
classify percepts, and act accordingly
2. Model-based
construct an internal representation of the world and use it to act
3. Goal-based
form goals and try to achieve them

Goal-based usually $\subseteq$ model-based agents

Simple reflex agent (I)

Direct connection :

Rules directly connect percepts to actions
(similar reflexes, instincts)

$$\forall\, s, b, u, c, t \;\; Percept([s, b, Glitter, u, c], t) \Rightarrow Action(Grab, t)$$

Rules for perception : more flexible, more useful form

$$\forall\, b, g, u, c, t \;\; Percept([Stench, b, g, u, c], t) \Rightarrow Stench(t)$$

$$\forall\, s, g, u, c, t \;\; Percept([s, Breeze, g, u, c], t) \Rightarrow Breeze(t)$$

$$\forall\, s, b, u, c, t \;\; Percept([s, b, Glitter, u, c], t) \Rightarrow AtGold(t)$$

$+$ connection from predicates to action choices
$\forall t AtGold(t) \Rightarrow Action(Grab, t)$

Simple reflex agent (II) Limitations

Ideally: retrieve gold, if possible, and climb out

Pure reflex: no representation of the world

Reflex with internal state: do we have the gold already?
$\forall\, t \;\; AtGold(t) \;\wedge\; \neg Holding(Gold, t) \;\Longrightarrow\; Action(Grab, t)$

Problems:

$Holding(Gold, t)$ cannot be observed
Unable to avoid infinite loops (example)

Deducing simple properties

Synchronic rules: (same time)

relate properties of world to other properties of world:

$\forall\, l, t \;\; At(Agent, l, t) \wedge Stench(t) \Rightarrow Smelly(l)$

$\forall\, l, t \;\; At(Agent, l, t) \wedge Breeze(t) \Rightarrow Breezy(l)$

$\rightarrow Smelly, Breezy$ do not need a situation argument

Causal rules: infer effect from cause

$\forall x, y Pit(x) \wedge Adjacent(x, y) \Rightarrow Breezy(y)$

does not say whether squares far away from the pits can be breezy

Diagnostic rules: infer cause from effect

$\forall y Breezy(y) \Rightarrow \exists\, x Pit(x) \wedge Adjacent(x, y)$

does not say where the pit is

Causal rules and diagnostic rules: incomplete

Model-based Agent

Diagnostic reasoning vs. model-based reasoning

Diagnostic rules: do not necessarily derive the strongest possible conclusions from the available information

Model-based reasoning: explicit model of a process and what are the effects/symptoms

> *If the axioms correctly and completely describe the way the world works and the way the percepts are produced, then the inference procedure will be correctly infer the strongest possible description of the world state give the available percepts.*

Preferences among actions

One way: use a simple scale, $Great, Good, Medium, Risky, Deadly$

Action-value system: rules state how desirable actions are

$$\forall\, a, s\ Great(a, s) \Rightarrow Action(a, s)$$

$$\forall\, a, s\ Good(a, s) \wedge (\neg \exists b Great(b, s)) \Rightarrow Action(a, s)$$

$$\forall\, a, s\ Medium(a, s) \wedge (\neg \exists b Good(b, s) \vee Great(b, s)) \Rightarrow Action(a, s)$$

$$\forall\, a, s\ Risky(a, s) \ldots \wedge (\neg \exists b Medium(b, s)\ veeGood(b, s) \vee Great(b, s))..$$

Toward a goal-based agent

Presence of an explicit goal allows the agent to work out a sequence of actions that will achieve the goal.

Three ways:

Inference : write axioms, generate all possible sentences
good solutions+ wasteful solutions

Search : best-first search to find a path to the goal
requires translating agent's knowledge into operators and state representations

Planning : a special-purpose reasoning system to reason about actions

Making plans (I)

Initial condition in KB:

$At(Agent, [1, 1], S_0)$

$At(Gold, [1, 2], S_0)$

Query: $Ask(KB, \exists\, s\ Holding(Gold, s))$

i.e., in what situation will I be holding the gold?

Answer: $\{s/Result(Grab, Result(Forward, S_0))\}$

i.e., go forward and then grab the gold

This assumes that the agent is interested in plans starting at S_0 and that S_0 is the only situation described in the KB

Making plans (II)

Represent plans as action sequences $[a_1, a_2, \ldots, a_n]$

$PlanResult(p, s)$ is the result of executing p in s

Then the query $Ask(KB, \exists\, p\ Holding(Gold, PlanResult(p, S_0)))$ has the solution $\{p/[Forward, Grab]\}$

Definition of $PlanResult$ in terms of $Result$:

$\forall s PlanResult([], s) = s$

$\forall a, p, s PlanResult([a|p], s) = PlanResult(p, Result(a, s))$

Planning systems are special-purpose reasoners designed to do this type of inference more efficiently than a general-purpose reasoner

Summary

First-order logic
— objects and relations are semantic primitives
— syntax: constants, functions, predicates, equality, quantifiers

Increased expressive power: sufficient to define Wumpus world

Situation calculus
— conventions for describing actions and change in FOL
— can formulate planning as inference on a situation calculus KB

Title: Inference in First-Order Logic
AIMA: Chapter 9 (Sections 9.1 to 9.4)

Introduction to Artificial Intelligence
CSCE 476-876, Spring 2003
URL: `www.cse.unl.edu/~choueiry/S03-476-876`

Berthe Y. Choueiry (Shu-we-ri)
`choueiry@cse.unl.edu`, (402)472-5444

Summary

First-order logic
— objects and relations are semantic primitives
— syntax: constants, functions, predicates, equality, quantifiers

Situation calculus
— conventions for describing actions and change in FOL
— can formulate planning as inference on a situation calculus KB

Inference in First-Order Logic?

Normal forms: some approaches to inference use syntactic operations on sentences, often expressed in standardized forms

Conjunctive Normal Form (CNF)

$$\textit{conjunction} \text{ of } \underbrace{\textit{disjunctions} \text{ of } \textit{literals}}_{\textit{clauses}}$$

E.g., $(A \vee \neg B) \wedge (B \vee C \neg D)$

Horn Form

conjunction of *Horn clauses* (clauses with ≤ 1 positive literal)
E.g., $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$
Often written as set of implications: $B \Rightarrow A$ and $(C \wedge D) \Rightarrow B$

Inference in Propositional Logic

Rules:

1. Modus Ponens
2. And-elimination
3. And-introduction
4. Or-introduction
5. Double-negation elimination
6. Unit-resolution
7. Resolution

Inference:

- sound and complete
- NP-complete in general
- polynomial for Horn clauses (only with MP!!)

Outline

- Proofs in FOL
- Unification
- Forward chaining
- Backward chaining

Important result:
Resolution with refutation is a complete inference procedure for FOL.

Inference rules for FOL

1. Modus Ponens (MP)
2. And-elimination
3. And-introduction (AI)
4. Or-introduction
5. Resolution
6. +3 additional rules to handle FOL sentences with quantifiers
   (a) Existential elimination
   (b) Existential introduction
   (c) Universal elimination (UE)

MP+AI+UE $\equiv$ Generalized Modus Ponens

Substitution (a.k.a. binding list)

$$Subst(\{x/Sam, y/Pam\}, Likes(x, y)) = Likes(Sam, Pam)$$

Universal elimination (UE)

For any sentence α, variable v, and ground term g:

$$\frac{\forall\, v\ \alpha}{Subst(\{v/g\}, \alpha)}$$

Example:

Sentence = $\forall\, x\ Likes(x, IceCream)$
Variable = x, constant = Ben, binding list= {x/Ben}
we can infer: $Likes(Ben, IceCream)$

(A universal sentence is the conjunction of all its possible instantiations)

Existential elimination

For any sentence α, variable v, and constant symbol term k:

$$\frac{\exists\, v\ \alpha}{Subst(\{v/k\}, \alpha)}$$

Example:

Sentence = $\exists\, x\ Kill(x, Victim)$
Variable = x, constant = Murdered, binding list= {x/Murderer}
we can infer: $Kill(Murderer, Victim)$

warning: Murderer does <u>not</u> appear in KB
e.g., $Kill(Victim, Victim)$ is not a logical consequence
of the sentence

Existential introduction

For any sentence α, variable v, and ground term g,
warning: v does not appear α, g <u>does</u> appear in α

$$\frac{\alpha}{\exists\, v\ Subst(\{g/v\}, \alpha)}$$

Example:

Sentence = $Likes(Jerry, IceCream)$
Variable = x, ground term = Jerry
we can infer: $\exists\, x\ Likes(x, IceCream)$

Proofs

Sound inference: find α such that $KB \models \alpha$.

Proof process is a search, operators are inference rules

Examples:

Modus Ponens (MP)

$$\frac{\alpha, \quad \alpha \Rightarrow \beta}{\beta} \qquad \frac{At(Lynn, UNL) \quad At(Lynn, UNL) \Rightarrow Smart(Lynn)}{Smart(Lynn)}$$

And-Introduction (AI)

$$\frac{\alpha \quad \beta}{\alpha \wedge \beta} \qquad \frac{Smart(Lynn) \quad CSMajor(Lynn)}{Smart(Lynn) \wedge CSMajor(Lynn)}$$

Universal Elimination (UE)

$$\frac{\forall x \alpha}{Subst(v/\tau, \alpha)} \qquad \frac{\forall x At(x, UNL) \Rightarrow Smart(x)}{At(Pat, UNL) \Rightarrow Smart(Pat)}$$

τ must be a ground term (*i.e.*, no variables)

Proof procedure

Application of inference rules

— simply matches the premise patterns of the rules
to the sentences in the KB and

— adds their suitably instantiated conclusion patterns to the KB

Bob is a buffalo	1. $Buffalo(Bob)$
Wilber is a pig	2. $Pig(Wilber)$
Buffaloes outrun pigs	3. $\forall x, y Buffalo(x) \wedge Pig(y) \Rightarrow Faster(x, y)$
Bob outruns Wilber	
AI 1 & 2	4.
UE 3, $\{x/Bob, y/Wilber\}$	5.
MP 4 & 5	6.

Search with primitive inference rules

Search components:

- Operators: inference rules
- States: sets of sentences
- Goal tests: check if query sentence is in the state

Problems:
long proofs, with huge branching factors (especially for UE)
potentially explosive search space

Opportunity:
Pattern: And-Introduction, U-elimination, Modus-Ponens
$\longrightarrow$ define a single operator for 3-steps

Unification (I)

$Unify$:
— takes two atomic sentences p and q and
— returns a substitution in which p and q 'look the same'
— (if not possible, returns $fail$.)

$$Unify(p, q) = \theta \text{ where } Subst(\theta, p) = Subst(\theta, q)$$

Example 1:

$$p = Likes(John, x)$$
$$q = Likes(y, IceCream)$$
$$Unify(p, q) = \{John/y, x/IceCream\}$$

Example 2:

$$Unify(Likes(John, x), Likes(x, Jane)) = fail$$

Unification (II)

p	q	σ(substitution)
$Knows(John, x)$	$Knows(John, Jane)$	
$Knows(John, x)$	$Knows(y, OJ)$	
$Knows(John, x)$	$Knows(y, Mother(y))$	

Idea: Given a rule,
unify rule premises with known facts,
apply unifier to conclusion

E.g., if we know q and $Knows(John, x) \Rightarrow Likes(John, x)$
then we conclude $Likes(John, Jane)$
$Likes(John, OJ)$
$Likes(John, Mother(John))$

Unification (III) Most general unifier

MGU is the substitution that makes the least commitment about the bindings of the variables

$Unify(Knows(John, x), Knows(y, z))$ can return:
$\{y/John, x/z\}$,
$\{y/John, x/John, z/John\}$,
etc..

Unify must return the most general unifier: $\{y/John, x/z\}$

Generalized Modus Ponens (GMP) (I)

Idea: a generalization of MP to do in a single blow AI+UE+MP

$$\frac{p_1{}',\ \ p_2{}',\ \ldots,\ p_n{}',\ \ (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{Subst(\theta, q)}$$

$$\begin{aligned} \text{Example: } p_1{}' &= \quad Faster(Bob, Pat) \\ p_2{}' &= \quad Faster(Pat, Steve) \\ p_1 \wedge p_2 \Rightarrow q \ &= \quad Faster(x, y) \wedge Faster(y, z) \Rightarrow Faster(x, z) \\ \theta &= \quad \{x/Bob, y/Pat, z/Steve\} \\ Subst(\theta, q) &= \quad Faster(Bob, Steve) \end{aligned}$$

Generalized Modus Ponens (II)

All sentences in KB must be in a form that matches one of the premises of the Modus-Ponens rule:
— a single atomic sentence, or
— (conjunction of atomic sentences) $\Rightarrow$ (atomic sentence)

$\longrightarrow$ KB must be of <u>Horn Normal Form</u>
($i.e.$, conjunction of Horn clauses)

Reminder: Horn Normal Form
conjunction of *Horn clauses* (clauses with ≤ 1 positive literal)
e.g., $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$
Often written as set of implications: $B \Rightarrow A$ and $(C \wedge D) \Rightarrow B$

Generalized Modus Ponens (III)

$\longrightarrow$ KB must be of Horn Normal Form
(*i.e.*, conjunction of Horn clauses)

- Sentences converted to Horn clauses as entered to KB using $\exists$-elimination and And-elimination
  Example: $\exists x Dog(x) \land Likes(John, x)$ becomes
  $Dog(Rebel)$ and $Likes(John, Rebel)$
- All variables assumed universally quantified:
  Example: $\forall\, x\ Likes(x, IceCream)$ is written $Likes(x, IceCream)$

GMP is a sound procedure, basis for:
— forward chaining and
— backward chaining

Forward chaining

When a new fact p is added to the KB
for each rule such that p unifies with a premise
if the other premises are known
then add the conclusion to the KB and continue chaining

Forward chaining is data-driven
e.g., inferring properties from percepts
inference is not directed toward solving a particular task

Forward chaining: example

Add facts 1, 2, 3, 4, 5, 7 in turn.

1. $Buffalo(x) \wedge Pig(y) \Rightarrow Faster(x, y)$
2. $Pig(y) \wedge Snail(z) \Rightarrow Faster(y, z)$
3. $Faster(x, y) \wedge Faster(y, z) \Rightarrow Faster(x, z)$
4. $Buffalo(Bob)$
5. $Pig(Wilber)$
6. ...
7. $Snail(Lynn)$
8. ...
9. ...

Backward chaining:

When a query q is asked
  if a matching fact q' is known, return the unifier
  for each rule whose consequent q' matches q
    attempt to prove each premise of the rule
    by backward chaining

→ Some added complications in keeping track of the unifiers

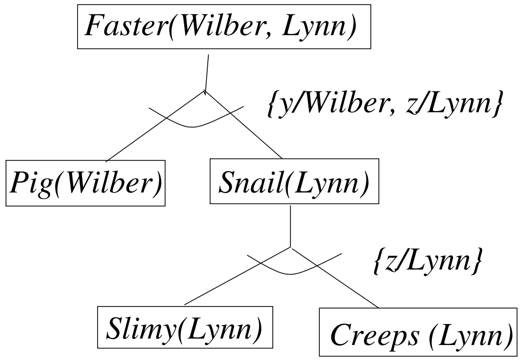
→ More complications help to avoid infinite loops

→ Two versions: find <u>any</u> solution, find <u>all</u> solutions

Backward chaining is the basis for <u>logic programming</u>, e.g., Prolog

Backward chaining: example

1. $Pig(y) \land Snail(z) \Rightarrow Faster(y, z)$
2. $Slimy(z) \land Creeps(z) \Rightarrow Snail(z)$
3. $Pig(Wilber)$
4. $Slimy(Lynn)$
5. $Creeps(Lynn)$



Title: Inference in First-Order Logic (II)
AIMA: Chapter 9 (Sections 9.5 to 9.6 pp.283)

Introduction to Artificial Intelligence
CSCE 476-876, Spring 2003
URL: `www.cse.unl.edu/~choueiry/S03-476-876`

Berthe Y. Choueiry (Shu-we-ri)
`choueiry@cse.unl.edu`, `(402)472-5444`

Outline:

- Completeness of GMP
- Generalization(s) of the resolution inference rule
- Resolution proof procedure
- Resolution + refutation proof procedure
- Conversion to normal form

Completeness of GMP:

Consider KB:

$$\forall x\ P(x) \Rightarrow Q(x)$$
$$\forall x\ \neg P(x) \Rightarrow R(x)$$
$$\forall x\ Q(x) \Rightarrow S(x)$$
$$\forall x\ R(x) \Rightarrow S(x)$$

Can we prove $S(A)$? GMP fails but..

Either, we have $P(A) \Rightarrow Q(A) \Rightarrow S(A)$
or, we have $\neg P(A) \Rightarrow R(A) \Rightarrow S(A)$

So KB $\models S(A)$, but GMP cannot allow derive it
$\longrightarrow$ GMP is an incomplete procedure

Reason: 2nd sentence is not a Horn clause

Task : find a <u>complete</u> proof procedure for FOL

Existence : established by Gödel in 1930/1931

There exists a set of inference rules that allows a complete proof procedure R

if $KB \models \alpha$ then $KB \vdash_R \alpha$

Resolution algorithm : Robinson, 1965

Modus Ponens: does not allow us to derive new <u>implications</u>

$$\frac{\alpha, \quad \alpha \Rightarrow \beta}{\beta}$$

Resolution rule: two forms, implications or disjunctions

$$\frac{\neg\alpha \Rightarrow \beta, \quad \beta \Rightarrow \gamma}{\neg\alpha \Rightarrow \gamma} \quad \text{or equivalently} \quad \frac{\alpha \vee \beta, \quad \neg\beta \vee \gamma}{\alpha \vee \gamma}$$

is more powerful than MP: allows us to derive a new implication by transitivity

Note: Resolution is a generalization of Modus Ponens:

$$\frac{\alpha, \quad \alpha \Rightarrow \beta}{\beta} \quad \text{is equivalent to} \quad \frac{True \Rightarrow \alpha, \quad \alpha \Rightarrow \beta}{True \Rightarrow \beta}$$

Resolution rule

$$\frac{\neg\alpha \Rightarrow \beta, \quad \beta \Rightarrow \gamma}{\neg\alpha \Rightarrow \gamma} \quad \text{or equivalently} \quad \frac{\alpha \vee \beta, \quad \neg\beta \vee \gamma}{\alpha \vee \gamma}$$

but also says:

either β is true and then γ necessarily true

or β is false then necessarily α is true

So, necessarily $\alpha \vee \gamma$ is true

Note: Premises have exactly two disjuncts

$\longrightarrow$ generalized versions: disjuncts of any length

Generalized resolution (disjunctions)

Given:

Premise 1: $p_1 \vee \ldots \vee p_j \vee \ldots \vee p_m$

Premise 2: $q_1 \vee \ldots \vee q_k \vee \ldots \vee q_n$

If $p_j = \neg q_k$, then we can derive the consequence:

$$p_1 \vee \ldots p_{j-1} \vee p_{j+1} \ldots \vee p_m \vee q_1 \vee q_{k-1} \vee q_{k+1} \ldots \vee q_n$$

More generally:

If Unify$(p_i, \neg q_k) = \theta$, from Premises 1, 2, we can derive:

$Subst(\theta, (p_1 \vee \ldots p_{j-1} \vee p_{j+1} \ldots \vee p_m \vee q_1 \ldots q_{k-1} \vee q_{k+1} \ldots \vee q_n))$.

Thus, the generalized resolution (disjunctions):

$$\frac{p_1 \vee \ldots p_j \ldots \vee p_m \quad q_1 \vee \ldots q_k \ldots \vee q_n}{Subst(\theta, (p_1 \vee \ldots p_{j-1} \vee p_{j+1} \ldots \vee p_m \vee q_1 \ldots q_{k-1} \vee q_{k+1} \ldots \vee q_n))}$$

Generalized resolution (disjunctions)

$$\frac{p_1 \vee \ldots p_j \ldots \vee p_m \quad q_1 \vee \ldots q_k \ldots \vee p_n}{Subst(\theta, (p_1 \vee \ldots p_{j-1} \vee p_{j+1} \ldots \vee p_m \vee q_1 \ldots q_{k-1} \vee q_{k+1} \ldots \vee q_n))}$$

Similarly, **Generalized resolution (implications)**

$$\frac{p_1 \wedge \ldots p_j \ldots \wedge p_{n_1} \Rightarrow r_1 \vee \ldots r_{n_2} \qquad s_1 \wedge \ldots \wedge s_{n_3} \Rightarrow q_1 \vee \ldots q_k \ldots \vee q_{n_4}}{Subst(\theta, (p_1 \wedge \ldots p_{j-1} \wedge p_{j+1} \wedge p_{n_1} \wedge s_1 \wedge \ldots s_{n_3} \Rightarrow r_1 \vee ..r_{n_2} \vee q_1 \vee \ldots q_{k-1} \vee q_{k+1} \vee \ldots \vee q_{n_4}))}$$

Conversion to conjunctive/implicative normal forms

Conjunctive normal form:
KB is a conjunction of disjunctions

'Implicative' normal form:
KB is a conjunction of implications

Consider KB:

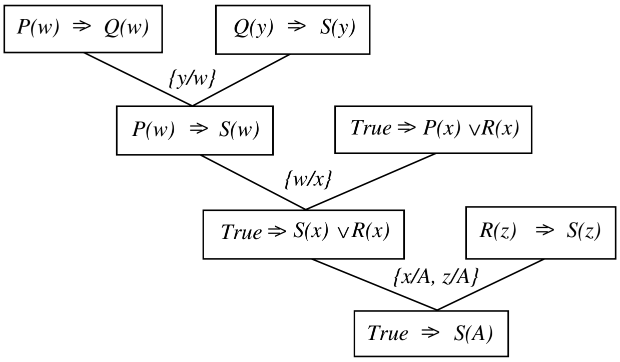
$$\forall x\ P(x) \Rightarrow Q(x)$$
$$\forall x\ \neg P(x) \Rightarrow R(x)$$
$$\forall x\ Q(x) \Rightarrow S(x)$$
$$\forall x\ R(x) \Rightarrow S(x)$$

Conjunctive Normal Form	Implicative Normal Form
historically more common	commonly used in AI
$\neg P(w) \vee Q(w)$	$P(w) \Rightarrow Q(w)$
$P(x) \vee R(x)$	$True \Rightarrow P(x) \vee R(x)$
$\neg Q(y) \vee S(y)$	$Q(y) \Rightarrow S(y)$
$\neg R(z) \vee S(z)$	$R(z) \Rightarrow S(z)$

Resolution proofs

$$
\begin{aligned}
P(w) &\Rightarrow Q(w) \\
True &\Rightarrow P(x) \vee R(x) \\
Q(y) &\Rightarrow S(y) \\
R(z) &\Rightarrow S(z)
\end{aligned}
$$

Prove $S(A)$ using resolution inference rule and forward chaining (backward chaining is OK too)



Technically, $True \Rightarrow S(A) \vee S(A)$
Use factoring rule: $True \Rightarrow S(A)$

Question

Is resolution is a complete proof procedure?

Consider empty KB, try proving: $P \vee \neg P$ (valid sentence)

Since KB is empty, no way to use KB to prove the sentence

$\longrightarrow$ use resolution with refutation
$\longrightarrow$ resolution with refutation is a complete proof procedure

Refutation

a.k.a. proof by contradiction, reductio ad absurdum

Idea: want to proof P ? assume P false, and prove contradiction

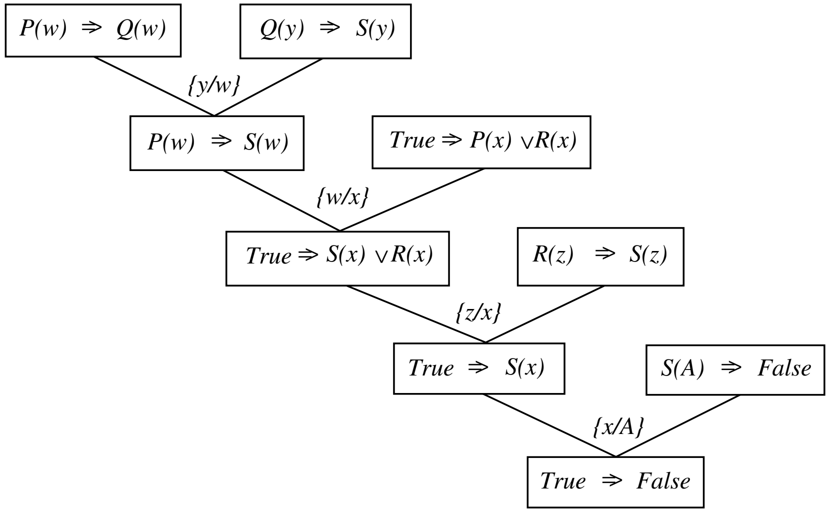
$$(KB \wedge \neg P \Rightarrow False) \Leftrightarrow (KB \Rightarrow P)$$

Note: $\neg P$ in 'implicative' normal form is written: $P \Rightarrow False$

Resolution + refutation: Example

KB (Implicative Normal Form):

$$\begin{aligned} P(w) &\Rightarrow Q(w) \\ True &\Rightarrow P(x) \vee R(x) \\ Q(y) &\Rightarrow S(y) \\ R(z) &\Rightarrow S(z) \\ S(A) &\Rightarrow False \end{aligned}$$



Conversion to Normal Form

7/8 rules to execute in sequence (easy to understand, hard to remember):

1. Eliminate implications
2. Move $\neg$ inwards
3. Standardize variables
4. Move quantifiers left
5. Skolemize
6. Distribute $\wedge$ over $\vee$
7. Flatten nested conjunctions and disjunctions
8. Convert disjunctions to implications

1. **Eliminate implications**:

$$p \Rightarrow q \quad \text{becomes} \quad \neg p \vee q$$

2. **Move $\neg$ inwards** :
negation is allowed only on atoms, use Morgan's laws

$\neg(p \vee q)$	becomes	$\neg p \wedge \neg q$
$\neg(p \wedge q)$	becomes	$\neg p \vee \neg q$
$\neg \forall x, p$	becomes	$\exists x, \neg p$
$\neg \exists x, p$	becomes	$\forall x, \neg p$
$\neg\neg p$	becomes	p

3. **Standardize variables**:
   (to avoid confusion when dropping quantifiers)
   Example: $(\forall x P(x)) \vee (\exists x Q(x))$ becomes $(\forall x P(x)) \vee (\exists y Q(y))$

4. **Move quantifiers left**:
   Keep the order in which they appear!
   Example: $(p \vee \forall\ x\ q)$ becomes $\forall\ x\, p \vee q$

this is true because p is guaranteed not to contain X (step 3)

4. **Skolemization**:

this is the process of moving existential quantifiers by elimination

Simple case, as in existential elimination (remember?)
$\exists\ x P(x)$ translates into $P(A)$, A constant, not appearing in KB

More complex case: Everyone has a heart
$\forall x\ Person(x) \Rightarrow \exists\ y\ Heart(y) \wedge Has(x, y)$
Apply 'simple' existential elimination, replace y with a constant H
$\forall x\ Person(x) \Rightarrow Heart(H) \wedge Has(x, H)$
which means: everyone has the same heart H :—(

Skolem function: $F(x)$ heart of x, F is a function whose name does not appear elsewhere in the KB
$\forall x\ Person(x) \Rightarrow Heart(F(x)) \wedge Has(x, F(x))$

Original sentence: $\forall x\ Person(x) \Rightarrow \exists\, y\ Heart(y) \wedge Has(x, y)$

Skolemized sentence:
$\forall x\ Person(x) \Rightarrow Heart(F(x)) \wedge Has(x, F(x))$

Skolemization: replace the existentially quantified variable by a Skolem function applied to all the variables universally quantified outside the existentially quantified variable.

$\longrightarrow$ After Skolemization, existentially quantified variables are eliminated, so we can drop universal quantifiers as all variables are now only universally quantified..

6. **Distribute $\wedge$ over $\vee$** :
   $(a \wedge b) \vee c$ becomes $(a \vee c) \wedge (b \vee c)$

7. **Flatten nested conjunctions and disjunctions**:
   $(a \wedge b) \wedge c$ becomes $(a \wedge b \wedge c)$
   $(a \vee b) \vee c$ becomes $(a \vee b \vee c)$

8. **Convert disjunctions to implications**: (optional)
   $(\neg a \vee \neg b \vee c \vee d)$ becomes $a \wedge b\ \Rightarrow\ c \vee d$

Summary

Propositional logic:

7 Inference rules: complete and sound
general forms: NP-Complete
Horn clauses: tractable

FOL: (predicate logic)
Generalized Modus Ponens (And-intro+Universal elimination+MP):
— Requires KB to be in Horn Normal Form, otherwise incomplete
— forward+backward chaining

Resolution + Refutation:
Complete proof procedure
Sentences need to be in Normal Form (conjunctive or implicative) using a 7/8-step recipe