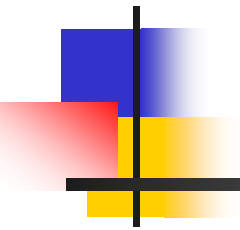
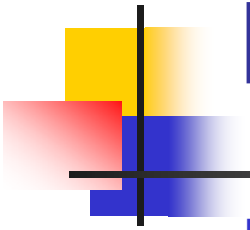
RN, Chapter 9

Predicate Calculus -- Inference





Logical Agents

- Reasoning [Ch 6]
- Propositional Logic [Ch 7]
- Predicate Calculus
  - Representation [Ch 8]
    - Syntax, Semantics, Expressiveness
    - Example: Circuits
  - Inference [Ch 9]
    - Resolution
- Implemented Systems [Ch 10]
- Planning [Ch 11]

Proof Process for Predicate Calculus

- Why not “Model Checking” approach?
- Still use

  Inference Rules

  to obtain new facts from exisiting info
- Still seeking…

If $\vdash$ is SOUND+COMPLETE,

$\Rightarrow$ $\vdash \equiv \models$

$\Rightarrow$ Computer can IGNORE SEMANTICS and just push symbols!

- Still exploiting

  SOUND inference rules

  MONOTONICITY

(Sound) Inference Rules

$$[MP]\quad \frac{P \Rightarrow Q \quad P}{Q}$$

$$[\forall E]\quad \frac{\forall x.\varphi(x)}{\varphi(\mathtt{A})} \text{ for any } \mathtt{A}$$

$$[\&I]\quad \frac{P \quad Q}{P \wedge Q}$$

$$[RC]\quad \frac{P \Rightarrow Q \quad Q \Rightarrow R}{P \Rightarrow R}$$

$$[MG]\quad \frac{P \Rightarrow Q \quad \neg P \Rightarrow Q}{Q}$$

$$[\vee D]\quad \frac{P \vee Q \quad \neg P}{Q}$$

$$[MT]\quad \frac{P \Rightarrow Q \quad \neg Q}{\neg P}$$

$$[\&E]\quad \frac{P \wedge Q}{P}$$

$$[\vee I]\quad \frac{P}{P \vee Q}$$

$$[\bowtie]\quad \frac{P \quad \neg P}{Contradiction}$$

. . .

Resolution Rule (PC)

- Simple example:

¬Man(x) v Mortal(x) Man(Socrates)
Mortal(Socrates)

using binding list λ ={ x/Socrates }

- Subst(Man(Socrates), λ) = Subst(Man(x), λ)
- Subst(Mortal(x), λ) = Mortal(Socrates)

- General:

$\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_n$ $\beta_1 \vee \beta_2 \vee \ldots \vee \beta_m$
$\alpha'_1 \vee \alpha'_2 \vee \ldots \vee \beta'_2 \vee \ldots \vee \beta'_m$

where there is a binding list, λ, s.t.

$Subst(\alpha_n, \lambda) = \neg Subst(\beta_1, \lambda)$

$Subst(\alpha_i, \lambda) = \alpha'_i \ \forall\ i$

$Subst(\beta_j, \lambda) = \beta'_j \ \forall\ j$

+ Subsumption:

if KB includes P(x) , remove "P(A)", "P(x) v Q(w)"

A v B, remove A v B v C

Requirement of Resolution

For Resolution to work, need:

1. Particular type of proof procedure to be “complete"

   A: Called Refutation Proof

   [... try to find contradiction...]

2. To express information as Conjunction of Disjunctions

   A: Called *Conjunctive Normal Form*

   [... is universal; can eliminate $\Rightarrow$, $\exists$, ...]

3. Process that takes two literals p, q and returns binding-list λ s.t.

   Subst(p, λ) = Subst(q, λ)

   A: Called Unification

   [... is well defined, and efficient, and ...]

1. Refutation Proof

- Deduction Theory

$KB \models \sigma$ *iff*
$KB \cup \neg\sigma$ is inconsistent *iff*
$KB \cup \neg\sigma \models \{\}$

- To prove σ:

Add $\neg\sigma$ to KB
(Attempt to) Prove a contradiction

- **Resolution is Refutation Complete (in PC)**
  If $KB \models \sigma$ then
  $\exists$ resolution proof of $\{\}$ from $KB \cup \neg\sigma$

2. Conversion to Conjunctive Normal Form

0: $\forall x\,[(\forall y\,P(x,\,y)) \Rightarrow \neg\forall y\,Q(x,\,y) \Rightarrow R(x,\,y)\,]$

1: **Eliminate implication, iff, . . .**

$\forall x\,[\neg(\forall y\,P(x,\,y)) \vee [\,\neg\forall y\,[\neg Q(x,\,y) \vee R(x,\,y)\,]]]$

2: **Move $\neg$ inwards**

$\forall x\,[(\exists y\,\neg P(x,\,y)) \vee [\,\exists y\,Q(x,\,y) \wedge \neg R(x,\,y)]]$

3: **Standarize variables**

$\forall x\,[(\exists y\,\neg P(x,\,y)) \vee [\,\exists z\,Q(x,\,z) \wedge \neg R(x,\,z)\,]]$

4: **Move quantifiers left**

$\forall x\,\exists y\,\exists z\,[\neg P(x,\,y)) \vee [\,Q(x,\,z) \wedge \neg R(x,\,z)\,]]$

5: **Skolemize (remove existentials); Drop $\forall$s**

$\neg P(x,\,F_1(x)) \vee [\,Q(x,\,F_2(x)) \wedge \neg R(x,\,F_2(x))\,]$

6: **Distribute $\wedge$ over $\vee$**

$[\neg P(x,\,F_1(x)) \vee Q(x,\,F_2(x))]$
$\wedge\,[\neg P(x,\,F_1(x)) \vee \neg R(x,\,F_2(x))]$

7: **Change to SET notation**

$\left\{\begin{array}{l} \neg P(x,\,F_1(x)) \vee Q(x,\,F_2(x)), \\ \neg P(x,\,F_1(x)) \vee \neg R(x,\,F_2(x)) \end{array}\right\}$

8: **Make variables unique**

$\left\{\begin{array}{l} \neg P(x_1,\,F_1(x_1)) \vee Q(x_1,\,F_2(x_1)), \\ \neg P(x_2,\,F_1(x_2)) \vee \neg R(x_2,\,F_2(x_2)) \end{array}\right\}$

Skolemizing

- To convert arbitrary PredicateCalculus to
  "Conjunctive Normal Form"
  need to eliminate ∃
- **A**: Just "name" it ... Using *new* name, to avoid conflicts
- Eg: "There is a rich person" ∃ x Rich(x)
  becomes Rich(G_1)
  where G_1 is a new "Skolem constant"
- Note: Rich(G_1) will NOT unify with
  Rich(Russ) nor
  Rich(β) for any β appearing in KB.
- Eg:

$$\exists k \frac{d}{dy}\left(k^y\right) = k^y \longrightarrow \frac{d}{dy}\left(e^y\right) = e^y$$

- Trickier when ∃ is inside ∀ ...

Skolemization #2

- "Everyone has a heart."
  - $\forall X$ person(X) $\Rightarrow$ $\exists Y$ heart(Y) & has(X, Y)
  - **Incorrect**: $\forall X$ person(X) $\Rightarrow$ heart($\mathbf{h_1}$) & has(X, $\mathbf{h_1}$)

    ... ?everyone has the SAME heart $\mathbf{h_1}$?
  - **Correct**:

    $\forall X$ person(X) $\Rightarrow$ heart(**h(X)**) & has(X, **h(X)**)

    where **h(.)** is a new symbol ("Skolem function")
- Skolem function arguments:

  all enclosing universally quantified variables
- Eg: $\forall A\ \forall B\ \exists C\ \forall D\ \exists E$ g(A,B, C, D, E)

  $\forall A\ \forall B\ \forall D$ g(A,B, $f_C(A,B)$, D, $f_E(A,B,D)$)

Skolemization #2

Skolemizing procedure (to remove existentials)

For each existential X,

let $Y_1, \ldots Y_m$ be the universally quantified variables that are quantified to the LEFT of X's "$\exists Y$".

Generate new function symbol, g_X, of *m* variables.

Replace each X with $g_X(Y_1, \ldots Y_m)$.

(Write $g_X()$ as g_X.)

$\forall Y \; \exists X \;\; \phi(X) \;\&\; \rho(Y) \;\rightarrow\; \forall Y \; \phi(\; \mathbf{g_X(Y)} \;) \;\&\; \rho(Y)$

$\exists X \; \forall Y \;\; \phi(X) \;\&\; \rho(Y) \;\rightarrow\; \forall Y \; \phi(\;\; \mathbf{g_X} \;\;) \;\&\; \rho(Y)$

Requirement of Resolution

For Resolution to work, need:

1. Particular type of proof procedure to be “complete"

   A: Called Refutation Proof

   [... try to find contradiction...]

2. To express information as Conjunction of Disjunctions

   A: Called *Conjunctive Normal Form*

   [... is universal; can eliminate $\Rightarrow$, $\exists$, ...]

3. Process that takes two literals p, q and returns binding-list λ s.t.

   Subst(p, λ) = Subst(q, λ)

   A: Called Unification

   [... is well defined, and efficient, and ...]

3. Unification (Specification)

Variables capitalized

- Fancy Match
- Unify(p, q) = σ
  - p, q: atomic propositions (w/variables)
  - σ : binding list ...
    - *Fail* or
    - $\{V_1/t_1, V_2/t_2, ..., V_n/t_n\}$ where
      - V_i's are distinct,
      - each t_j is { constant, variable, functional expr. }
      - no V_i appears in any t_j
- If non-Fail,
  Subst(p, σ) = Subst(q, σ)

  ... ie, σ makes p and q look the same

Substitution

- Substitution is set { V_1/t_1 V_2/t_2 ... V_n/t_n } where
  - V_i are distinct variables
  - t_i are terms that do not use any of the V_js
- Examples:

  { X / a }

  { X / a, Y / b, Z / f(a, W) }

  { X / W, Y / f(W), Z/W }

  ~~{ f(X) / a }~~

  ~~{ X / a, X / b }~~

  ~~{ X / f(X) }~~

  ~~{ X / f(Y), Y / g(q) }~~

  { X / f(g(q)), Y / g(q) }

Applying a Substitution

- Given $\begin{cases} t & - \text{a term} \\ \sigma & - \text{a substitution} \end{cases}$

"$t\sigma$" is the term resulting from applying substitution σ to term t.

f(a, h(Y,b), X)	{ X/b }	=	f(a, h(Y,b), b)
f(a, h(Y,b), X)	{ X/b Y/f(Z) }	=	f(a, h(f(Z),b), b)
f(a, h(Y,b), X)	{ X/Z Y/f(Z,a) }	=	f(a, h(f(Z,a),b), Z)
f(a, h(Y,b), X)	{ W/Z }	=	f(a, h(Y,b), X)

- σ need not include *all* variables in t;
  σ can include variables *not* in t

Composition of Substitutions

- Composition:

  $\sigma \circ \theta$ is composition of substitutions σ, θ

  For any term t, $t[\sigma \circ \theta] = (t\ \sigma)\ \theta$

- Example:

  $f(X)\ [\ \{X/Z\} \circ \{Z/a\}] = (\ f(X)\ \{X/Z\}\ )\ \{Z/a\}$

  $= f(Z)\ \{Z/a\}$

  $= f(a)$

- $\sigma \circ \theta$ is a substitution (usually)
- Eg:
  - $[\ \{X/a\} \circ \{Y/b\}\ ] = \{X/a, Y/b\}$
  - $[\ \{X/Z\} \circ \{Z/a\}\ ] = \{X/a, Z/a\}$

Unifiers

- t_1 and t_2 are unified by θ iff $t_1\,\theta = t_2\,\theta$
  Then θ is called a unifier

Examples: t_1 and t_2 are unifiable

t_1	t_2	unifer	term
f(b,c)	f(b,c)		
f(X,b)	f(a,Y)		
f(a,b)	f(c,d)		
f(a,b)	f(X,X)		
f(X,a)	f(Y,Y)		
f(g(U),d)	f(X,U)		
f(X)	f(g(X))		
f(X,g(X))	f(Y,Y)		
f(X)	f(Y)		

- Both t_1 and t_2 can have variables

Multiple Unifiers

Unifier for	$t_1 = f(X)$ and $t_2 = f(Y)$
θ	$t_1\theta = t_2\theta =$
$\{X/Y\}$	$f(Y)$
$\{Y/X\}$	$f(X)$
$\{Y/a \quad X/a\}$	$f(a)$
$\{Y/g(b,Z) \quad X/g(b,Z)\}$	$f(g(b(Z)))$
$\{X/Y \quad W/f(q,Z)\}$	$f(Y)$

- {Y/X} and {X/Y} make sense, but
  - { Y/a X/a } has irrelevant constant
  - { X/Y W/g } has irrelevant binding (W)
- Adding irrelevant bindings:

  ∞ unifiers!
- Is there a best one ?

Quest for Best Unifier

Wish list:

- No irrelevant constants
  - So {Y/X} preferred over { Y/a, X/a }
- No irrelevant bindings
  - So {Y/X} preferred over { Y/X, W/f(4,Z) }

INFERIOR unifier = composition of Good Unifier + another substitution

- Spse λ_1 has constant where λ_2 has varia

  Eg, λ_1 = { X/a, Y/a } , λ_2 = {X/Y}

  Then $\exists$ substitution μ s.t. $\lambda_2 \circ \mu = \lambda_1$

  Eg, μ = {Y/a} : {X/Y} $\circ$ {Y/a} = { X/a, Y/a }

- Spse λ_1 has extra binding over λ_2

  (Eg, λ_1 = {X/a, Y/b} , λ_2 = {X/a})

  Then $\exists$ substitution μ s.t. $\lambda_2 \circ \mu = \lambda_1$

  Eg, μ = {Y/b} : {X/a} $\circ$ {Y/b} = {X/a, Y/b}

Most General Unifier

- σ is a *mgu* for t_1 and t_2 iff
  - σ unifies t_1 and t_2, and
  - $\forall\ \mu$: unifier of t_1 and t_2,
    $\exists$ subsitution, θ, s.t. $\sigma \circ \theta = \mu$
    (Ie, for all terms t, $t\mu = (t\sigma)\theta$.)
- Example: $\sigma = \{\,\mathtt{X/Y}\,\}$ is mgu for `f(X)` and `f(Y)`
  Consider unifier $\mu = \{\,\mathtt{X/a}\quad\mathtt{Y/a}\,\}$.
  Use substitution $\theta = \{\,\mathtt{Y/a}\,\}$:

$$\begin{aligned}
\mathtt{f(X)}\mu &= \mathtt{f(X)\{X/a\ Y/a\}} \\
&= \mathtt{f(a)} \\
\\
\mathtt{f(X)}[\sigma \circ \theta] &= (\mathtt{f(X)}\sigma) \\
&= (\mathtt{f(X)\{X/} \\
&= \mathtt{f(Y)\{Y/a} \\
&= \mathtt{f(a)}
\end{aligned}$$

(μ is NOT a mgu, as $\neg\exists\theta'$ s.t. $\mu \circ \theta' = \sigma$!)

Similarly, $\mathtt{f(Y)}\mu = \mathtt{f(a)} = \mathtt{f(Y)}[\sigma \circ \theta]$

MGU – Example#2

- A mgu for

f(W, g(Z), Z)

f(X, Y, h(X))

is { X/W Y/g(h(W)) Z/h(W) }

MGU Procedure

```
Recursive Procedure MGU (x,y)
    If  x=y ⇒ Return ()
    If  Variable(x) ⇒ Return( MguVar(x,y) )
    If  Variable(y) ⇒ Return( MguVar(y,x) )
    If  Constant(x) or Constant(y) ⇒ Return( False )
    If  Not(Length(x) = Length(y)) ⇒ Return( False )
    g ← []
    For  i = 1..Length(x)  do
      s ← MGU( Part(x,i), Part(y,i) )
      g ← Compose(g,s)
      x ← Substitute(x,g)
      y ← Substitute(y,g)
    Return( g )
End MGU


Procedure MguVar (v,e)
    If    Includes(v,e) ⇒ Return( False )
    Else  Return( [v/e] )
End
```

MGU (con't)

- Notes:
  - If t_1 and t_2 are unifiable, then $\exists$ a mgu
  - Can be more than 1 mgu

    but they differ only in variable names
  - Not every unifier is a mgu.
  - A mgu uses constants only as necessary.
- Implementation: $\exists$ fast algorithm that computes a mgu of t_1 and t_2, if one exists;

  or reports failure.

- (Slow part is verifying legal substitution:

  none of v_i appear in any t_j.

  Avoid by resetting Prolog's *occurscheck* parameter.)

Example

- Natural Language

> *Jack owns a dog.*
> *Every dog owner is an animal lover.*
> *No animal lover kills an animal.*
> *Either Jack or Curiosity killed the cat (named Tuna).*
> *Did Curiosity kill the cat?*

- In predicate calculus:

> $\exists X$ dog(X) & owns(jack, X)
> $\forall X$ ($\exists$ Y dog(Y) & owns(X, Y)) $\Rightarrow$ animalLover(X)
> $\forall X$ animalLover(X) $\Rightarrow$ ($\forall Y$ animal(Y) $\Rightarrow \neg$kills(X, Y))
> kills(jack, tuna) v kills(curiosity, tuna)
> cat(tuna)
> $\forall X$ cat(X) $\Rightarrow$ animal(X)
> $\neg$kill(curiousity, tuna)

- Now what?

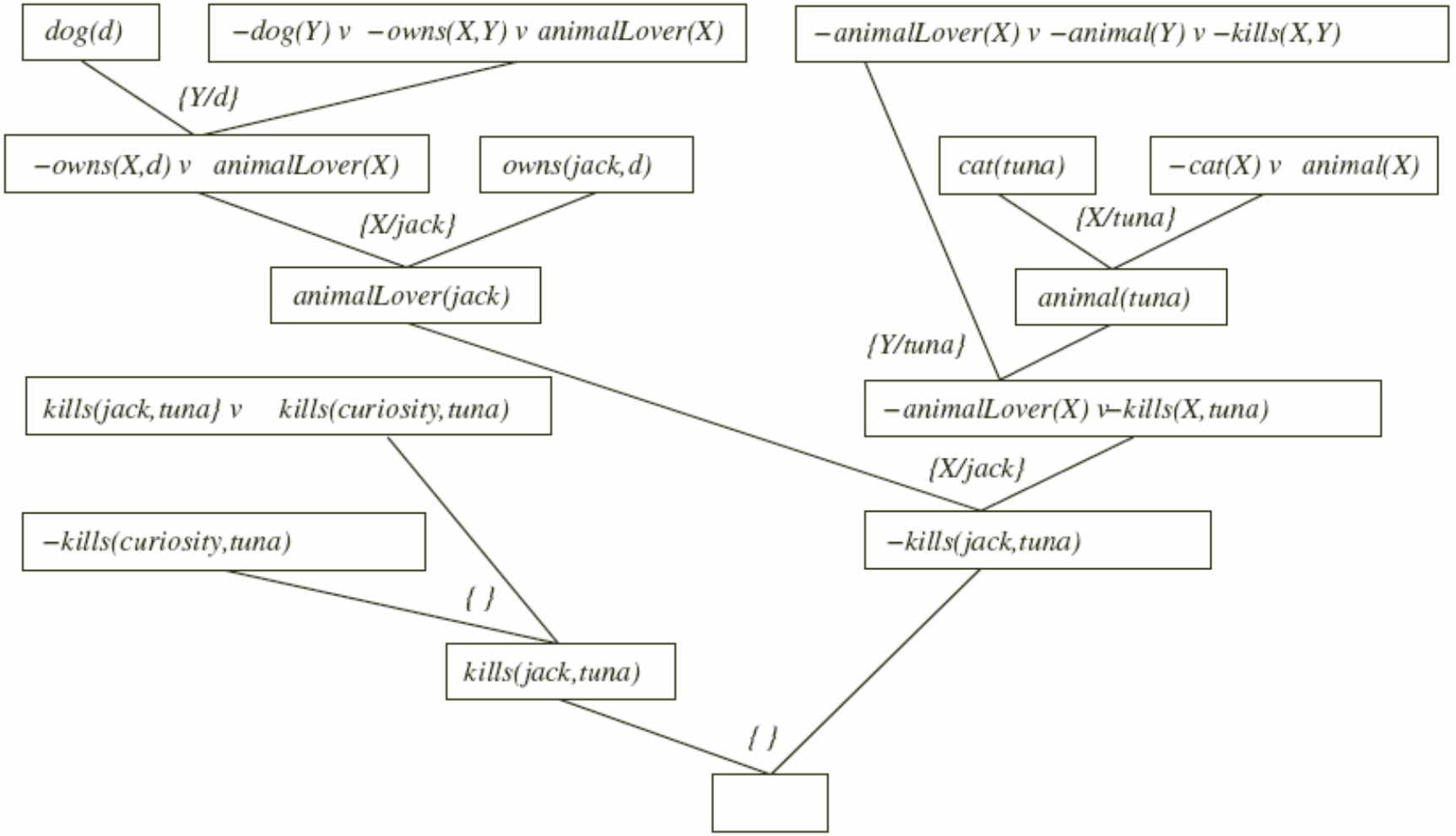
CNF Form

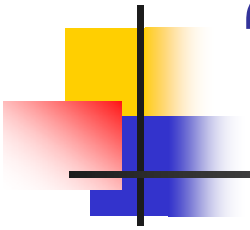
- ... in Conjunctive Normal Form

$\exists X$ dog(X) & owns(jack, X)
$\forall X$ ($\exists$ Y dog(Y) & owns(X, Y)) $\Rightarrow$ animalLover(X)
$\forall X$ animalLover(X) $\Rightarrow$ ($\forall Y$ animal(Y) $\Rightarrow$ $\neg$kills(X, Y))
kills(jack, tuna) v kills(curiosity, tuna)
cat(tuna)
$\forall X$ cat(X) $\Rightarrow$ animal(X)
$\neg$kill(curiousity, tuna)

- Note: Uniform structure
  Use new constants / functions: d
  for existentials ("Skolemizing")
  $\Rightarrow$ easier to refer to those objects

Refutation Proof by Resolution





“Tricks”

- 1. Refutation Proof

- 2. Normalization: put in CNF form
  - Skolemize ... remove ∃
    (by giving arbitrary, but unique name to ∃ objects)
  - remove quantifiers

- 3. Unification: matching variables/ terms
  to make literals look similar

Properties of Resolution

- **Sound**
  - $KB \vdash_{RR} \sigma$ only if

    σ is true in EVERY world in which KB holds.
- **Refutation Complete**
  - $KB \vdash_{RR} \sigma$ whenever

    σ is true in EVERY world in which KB holds.

    (as $\vdash_{Resln}$ is []-complete)
- **Semi-Decidable in Predicate Calculus**
  - $KB \vdash_{RR}^{?} \sigma$: If Yes, returns that answer eventually

    If No, may never return
- **Intractable**

  Exponential in |KB| for Propositional Logic

  (Linear for Proposition HORN)