How to Reason?

Q: How to reason?

Given KB, q, determine if $KB \models q$?

A: Select Inference Rule IR

Select Fact(s) $\{F_i\}$ from KB

Apply rule IR to Facts $\{F_i\}$

to get new Fact ϕ

...Add ϕ to KB

Repeat until find $\phi = q$

Issues: 1. Lots of Inference Rules

Which one to use, when?

2. Is overall system "complete"?

If $\exists$ answer, guaranteed to find it?

Resolution Rule (Propositional)

- Most Simple:

$$\frac{\begin{array}{ll} A \vee & B \\ & \neg B \end{array}}{A}$$

$$\frac{\begin{array}{ll} \texttt{Man} \vee & \texttt{Mouse} \\ & \neg\texttt{Mouse} \end{array}}{\texttt{Man}}$$

- Almost as Simple:

$$\frac{\begin{array}{lll} A \vee & B & \\ & \neg B & \vee\ C \end{array}}{A \vee \qquad\qquad C}$$

$$\frac{\begin{array}{lll} \texttt{Man} \vee & \texttt{Mouse} & \\ & \neg\texttt{Mouse} & \vee\ \texttt{CatFood} \end{array}}{\texttt{Man} \vee \qquad\qquad \texttt{CatFood}}$$

- General:

$$\frac{\begin{array}{lll} A_1 \vee A_2 \vee \cdots \vee A_{n-1} \vee & A_n & \\ & B_1 & \vee\ B_2 \vee \cdots \vee B_m \end{array}}{A_1 \vee A_2 \vee \cdots \vee A_{n-1} \vee \qquad B_2 \vee \cdots \vee B_m}$$

where $A_n = \neg B_1$

Verify Soundness

- **Modus Ponens:**

$$\begin{array}{c} \alpha \Rightarrow \beta \\ \alpha \\ \hline \beta \end{array}$$

Truth table:

	α	β	$\alpha \Rightarrow \beta$
*	T	T	T
	T	F	F
	F	T	T
	F	F	T

Consider all worlds where $\left\{ \begin{array}{c} \alpha \\ \alpha \Rightarrow \beta \end{array} \right\}$ both hold.

Observe: β holds here as well!

- **Resolution:**

$$\begin{array}{lll} \alpha \vee & \beta & \\ & \neg\beta \vee & \gamma \\ \hline \alpha & \vee & \gamma \end{array}$$

α	β	γ	$\alpha \vee \beta$	$\neg\beta \vee \gamma$	$\alpha \vee \gamma$
False	*False*	*False*	*False*	*True*	*False*
False	*False*	*True*	*False*	*True*	*True*
False	*True*	*False*	*True*	*False*	*False*
False	*True*	*True*	*True*	*True*	*True*
True	*False*	*False*	*True*	*True*	*True*
True	*False*	*True*	*True*	*True*	*True*
True	*True*	*False*	*True*	*False*	*True*
True	*True*	*True*	*True*	*True*	*True*

Sufficiency

- Subsumes:

$[MP]$	$\dfrac{p \Rightarrow q \quad p}{q}$	$\dfrac{\neg p \lor q \quad p}{q}$
$[MT]$	$\dfrac{p \Rightarrow q \quad \neg q}{\neg p}$	$\dfrac{\neg p \lor q \quad \neg q}{\neg p}$
$[RC]$	$\dfrac{p \Rightarrow q \quad q \Rightarrow r}{p \Rightarrow r}$	$\dfrac{\neg p \lor q \quad \neg q \lor r}{\neg p \lor r}$
$[MG]$	$\dfrac{p \Rightarrow q \quad \neg p \Rightarrow q}{q}$	$\dfrac{\neg p \lor q \quad p \lor q}{q}$
$[\bowtie]$		$\dfrac{\neg p \quad p}{\{\}}$

$\ldots$

- Is Resolution *sufficient*?
  Complete inference process?

Resolution Rule (PC)

- Simple Example:

$$\frac{\begin{array}{lcl} \neg\texttt{man(X)} & \vee & \texttt{mortal(X)} \\ \texttt{man(socrates)} & & \end{array}}{\texttt{mortal(socrates)}}$$

  using binding list $\sigma = \{ X/\texttt{socrates} \}$

  Notice:

  – *subst*(man(socrates), σ) = *subst*(man(X), σ)

  – *subst*(mortal(X), σ) = mortal(socrates)

- In General:

$$\frac{\begin{array}{lll} A_1 \vee A_2 \vee \cdots \vee A_{n-1} \vee & A_n & \\ & B_1 & \vee\ B_2 \vee \cdots \vee B_m \end{array}}{A'_1 \vee A'_2 \vee \cdots \vee A'_{n-1} \vee \qquad B'_2 \vee \cdots \vee B'_m}$$

  where there is a binding list, σ, for which

  subst(A_n, σ) = $\neg$*subst*(B_1, σ)

  subst(A_i, σ) = $A'_i \quad \forall i$

  subst(B_j, σ) = $B'_j \quad \forall j$

Requirements of Resolution

For Resolution to work, need:

1. Process that takes two literals p and q and returns binding-list σ s.t.

   $$\texttt{subst}(p, \sigma) = \texttt{subst}(q, \sigma)$$

   A: Called *Unification*

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

2. . . . to be "complete", needs particular type of proof procedure

   A: Called *Refutation Proof*

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

3. To express information as

   Conjunction of Disjunctions

   A: Called *Conjunctive Normal Form*

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

1. Unification (Specification)

- Fancy *Match*

- *Unify*(p, q) $= \sigma$

  - p, q: atomic propositions (w/variables)

  - σ: binding list —

    `Fail` or

    $\{\mathtt{x}_1/\mathtt{e}_1,\ \mathtt{x}_2/\mathtt{e}_2,\ \ldots,\ \mathtt{x}_n/\mathtt{e}_n\}$

    where

    $\mathtt{x}_i$'s are distinct,

    each $\mathtt{e}_j$ is $\left\{ \begin{array}{c} \text{constant} \\ \text{variable} \\ \text{functional expr.} \end{array} \right\}$

    no $\mathtt{x}_i$ appears in any $\mathtt{e}_j$.

- If non-`Fail`,

  subst(p, σ) = *subst*(q, σ)

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

2. Resolution is NOT Complete

- Resolution $\vdash_R$ smashes together clauses

  Given

  $KB = \{\ldots,\ \sigma \vee A,\ \ldots,\ \neg A \vee \rho,\ \ldots\}$,

  $\vdash_R$ can derive . . .

  $KB \vdash_R \sigma \vee \rho$

- But if $KB = \{\}$,

  $\vdash_R$ cannot derive anything

- But *tautologies* $(p \vee \neg p)$ always entailed

  $\{\} \models p \vee \neg p$

p	$p \vee \neg p$
+	+
0	+

  But

  $\{\} \not\vdash_R p \vee \neg p$

- Similarly

  $\{p\} \models p \vee p$

  But

  $\{p\} \not\vdash_R p \vee p$

. . .

Refutation

- Resolution can still be used for entailment! Using *Refutation Proof*:

- $KB \models \sigma$ means
  σ is true in all models, $\mathcal{M}(KB)$

...	σ	...
⋮	0	⋮
⋮	0	⋮
⋮	+	⋮
⋮	+	⋮
⋮	+	⋮
⋮	+	⋮

- Now consider $KB \cup \neg\sigma$

  It has NO models
  $\mathcal{M}(KB \cup \neg\sigma) = \{\}$

$\Rightarrow KB \cup \neg\sigma \models \mathcal{F}\text{alse}$

Refutation Proof

- **Deduction Theory**

$$
\begin{array}{ll}
KB \models \sigma & iff \\
KB \cup \{\neg\sigma\} \text{ is inconsistent} & iff \\
KB \cup \{\neg\sigma\} \models \mathcal{F}\text{alse} &
\end{array}
$$

- To prove σ:

Add $\neg\sigma$ to KB
Prove a Contradiction, $\mathcal{F}$alse

Refutation Complete

Def'n: $\vdash$ is Complete:

$$\forall\, KB,\ \sigma :\ KB \models \sigma \quad \Rightarrow \quad KB \vdash \sigma$$

$\vdash$ is REFUTATION Complete:

$$\forall\, KB :\ KB \models \{\} \quad \Rightarrow \quad KB \vdash \{\}$$

- Resolution $\vdash_R$ is REFUTATION COMPLETE

If $KB \models \sigma$
then $\exists$ resolution proof of $\mathcal{F}$alse
from $KB \cup \{\neg\sigma\}$

Proof: Let $RC(\Gamma)$ be deductive closure of Γ using Resl'n

Need only show: if $\{\} \notin RC(\Gamma)$,
then Γ is consistent ... *i.e.*, Γ has model.

Build model over variables $v_1, \ldots, v_k$:
For $i = 1..k$
$\star$ if $\exists c_j \in RC(\Gamma)$ s.t.
$\neg v_i \in c_j$ and assg'n to $v_1, \ldots, v_{i-1}$ false
then $v_i \leftarrow$ false
$\star$ otherwise $v_i \leftarrow$ true

This assignment $\{\pm v_1, \ldots, \pm v_k\}$ is model for Γ !

Using Refutation Resolution

- Given KB, σ

  Let $\Gamma = KB \cup \neg\sigma$

  Try to prove $\mathcal{F}$alse, using $\vdash_R$

  $\Gamma \vdash_R {}^{?} \mathcal{F}\text{alse}$

  If succeed, then $KB \models \sigma$

  If fail, then $KB \not\models \sigma$

- $\mathcal{F}$alse is EMPTY CLAUSE $\{\}$

- Problem:

  Resolution works by smashing CLAUSES!

  $\Rightarrow$ Need to encode KB, $\neg\sigma$ as *clauses*

- Solution: Can always be done!

3. Conversion to Conjunctive Normal Form

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

1: **Eliminate implication, iff, . . .**
$\forall \mathtt{x}\ [\neg(\forall \mathtt{y}\ P(\mathtt{x},\ \mathtt{y})) \vee [\,\neg\forall \mathtt{y}\,[\neg Q(\mathtt{x},\ \mathtt{y}) \vee R(\mathtt{x},\ \mathtt{y})\,]]]$

2: **Move $\neg$ inwards**
$\forall \mathtt{x}\,[(\exists \mathtt{y}\,\neg P(\mathtt{x},\ \mathtt{y})) \vee [\,\exists \mathtt{y}\, Q(\mathtt{x},\ \mathtt{y})\ \wedge\ \neg R(\mathtt{x},\ \mathtt{y})]]$

3: **Standarize variables**
$\forall \mathtt{x}\,[(\exists \mathtt{y}\,\neg P(\mathtt{x},\ \mathtt{y})) \vee [\,\exists \mathtt{z}\, Q(\mathtt{x},\ \mathtt{z})\ \wedge\ \neg R(\mathtt{x},\ \mathtt{z})\,]]$

4: **Move quantifiers left**
$\forall \mathtt{x}\,\exists \mathtt{z}\,\exists \mathtt{y}\,[\neg P(\mathtt{x},\ \mathtt{y})) \vee [\,Q(\mathtt{x},\ \mathtt{z})\ \wedge\ \neg R(\mathtt{x},\ \mathtt{z})\,]]$

5: **Skolemize (remove existentials); Drop $\forall$s**
$\neg P(\mathtt{x},\ \mathtt{F1(x)}) \vee [\,Q(\mathtt{x},\ \mathtt{F2(x)})\ \wedge\ \neg R(\mathtt{x},\ \mathtt{F2(x)})\,]$

6: **Distribute $\wedge$ over $\vee$**
$[\neg P(\mathtt{x},\ \mathtt{F1(x)}) \vee Q(\mathtt{x},\ \mathtt{F2(x)})]$
$\wedge\,[\neg P(\mathtt{x},\ \mathtt{F1(x)}) \vee \neg R(\mathtt{x},\ \mathtt{F2(x)})]$

7 : **Change to SET notation**
$\{\ \neg P(\mathtt{x},\ \mathtt{F1(x)})\ \ \vee Q(\mathtt{x},\ \mathtt{F2(x)}),$
$\neg P(\mathtt{x},\ \mathtt{F1(x)})\ \ \vee\ \ \neg R(\mathtt{x},\ \mathtt{F2(x)})\ \}$

8 : **Make variables unique**
$\{\ \neg P(\mathtt{x1},\ \mathtt{F1(x1)})\ \ \vee Q(\mathtt{x1},\ \mathtt{F2(x1)}),$
$\neg P(\mathtt{x2},\ \mathtt{F1(x2)})\ \ \vee\ \ \neg R(\mathtt{x2},\ \mathtt{F2(x2)})\ \}$

Normal form: Clausal

- **Eliminate implication, iff, ...**

  $\alpha \Rightarrow \beta \quad \mapsto \quad \neg\alpha \vee \beta$

- **Move $\neg$ inwards**

  $\neg\neg\alpha \quad \mapsto \quad \alpha$

  $\neg(\alpha \vee \beta) \quad \mapsto \quad \neg\alpha \wedge \neg\beta \qquad \neg\forall\, x\, \phi(x) \quad \mapsto \quad \exists\, x\, \neg\phi(x)$

  $\neg(\alpha \wedge \beta) \quad \mapsto \quad \neg\alpha \vee \neg\beta \qquad \neg\exists\, x\, \phi(x) \quad \mapsto \quad \forall\, x\, \neg\phi(x)$

- **Standarize variables** (Make all names unique:)

  $\forall x \phi(x) \wedge \exists x \rho(x) \quad \mapsto \quad \forall x \phi(x) \wedge \exists y \rho(y)$

- **Move quantifiers left**

  $\forall x\ \phi(x) \wedge \exists y \rho(y) \quad \mapsto \quad \forall x\, \exists y\ \phi(x) \wedge \rho(y))$

- **Skolemize** (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 "$\exists x$". 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.)

  $\forall y \exists x\ \phi(x) \wedge \rho(y) \quad \mapsto \quad \forall y\ \phi(\ \boxed{g_x(y)}\ ) \wedge \rho(y)$

  $\exists x \forall y\ \phi(x) \wedge \rho(y) \quad \mapsto \quad \forall y\ \phi(\ \boxed{g_x}\ ) \wedge \rho(y)$

- **Distribute $\wedge$ over $\vee$**

  $(x \wedge y) \vee z \quad \mapsto \quad (x \vee z) \wedge (y \vee z)$

- **Change to SET notation**

  $(x \vee z) \wedge (y \vee \neg z) \quad \mapsto \quad \{x \vee z, \quad y \vee \neg z\ \}$

- **Make Variables Unique**

  $$\left\{ \begin{array}{c} P(x)\ \vee\ Q(x) \\ R(x)\ \vee\ \neg W(x, y) \end{array} \right\} \mapsto \left\{ \begin{array}{c} P(x_1)\ \vee\ Q(x_1) \\ R(x_2)\ \vee\ \neg W(x_2, y) \end{array} \right\}$$

[R&N pp. 281–282]

Skolemizing

Q: To convert arbitrary PredicateCalculus to
"Conjunctive Normal Form"
need to eliminate $\exists$

A: Just "name" it
Using new name. . . to avoid conflicts

Eg: "There is a rich person."

$\exists X\ \texttt{rich}(X)$

becomes

$\texttt{rich}(\ \mathrm{g}_1\ )$

where g_1 is a new "Skolem constant"

Note: will NOT unify with

"$\texttt{rich(russ)}$" nor

"$\texttt{rich}(\ \chi\ )$" for any χ in KB.

Eg: $\exists k\ \frac{d}{dy}(k^y) = k^y$ becomes $\frac{d}{dy}(e^y) = e^y$

- Trickier when $\exists$ is inside $\forall$. . .

Skolemization #2

Eg: "Everyone has a heart."

$$\forall X\, \texttt{person}(X) \;\Rightarrow\; \exists Y\, \texttt{heart}(Y) \wedge \texttt{has}(X,Y)$$

Incorrect: $\forall X$ $\texttt{Person}(X) \Rightarrow \texttt{heart(h}_1\texttt{)}\wedge\texttt{has(x,H}_1\texttt{)}$

?everyone has the SAME heart $\texttt{h}_1$?

Correct: $\forall X \texttt{person}(X) \Rightarrow \texttt{heart(h(}X\texttt{))} \wedge \texttt{has(}X\texttt{,h(}X\texttt{))}$

where $\texttt{h}$ is a new symbol ("Skolem function")

- Skolem function arguments:
  all enclosing universally quantified variables

- **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 "$\exists X$". 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.)

$$\forall Y \exists X\ \phi(X) \wedge \rho(Y) \quad \mapsto \quad \forall Y\ \phi(\boxed{g_X(Y)}) \wedge \rho(Y)$$
$$\exists X \forall Y\ \phi(X) \wedge \rho(Y) \quad \mapsto \quad \forall Y\ \phi(\boxed{g_X}) \wedge \rho(Y)$$

Skolemization – Theorem

Theorem: Given theory T,
let $s(T)$ be "skolemized version" of T
replacing each existential with skolem function.
Then...
If T is consistent, then $s(T)$ is consistent.

Eg...

If $T_1 = \left\{ \begin{array}{c} \alpha_1 \\ \alpha_2 \\ \dots \\ \exists X \, \forall Y \;\; \varphi(X, Y) \\ \dots \end{array} \right\}$ is consistent

then $s(T_1) = \left\{ \begin{array}{c} \alpha_1 \\ \alpha_2 \\ \dots \\ \forall Y \;\; \varphi(\mathtt{c}_1, Y) \\ \dots \end{array} \right\}$ is consistent.

... if $s(T)$ is inconsistent, then T is inconsistent ...

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:

  $\exists x\ \texttt{Dog}(x)\ \wedge\ \texttt{Owns}(\texttt{Jack}, x)$

  $\forall x\ (\exists y\ \texttt{Dog}(y)\ \wedge\ \texttt{Owns}(x, y)) \Rightarrow \texttt{AnimalLover}(x)$

  $\forall x\ \texttt{AnimalLover}(x)\ \Rightarrow\ (\forall y\ \texttt{Animal}(y)\ \Rightarrow\ \neg\texttt{Kills}(x, y)\ )$

  `Kills(Jack, Tuna)` $\vee$ `Kills(Curiosity, Tuna)`

  `Cat(Tuna)`

  $\forall x\ \texttt{Cat}(x)\ \Rightarrow \texttt{Animal}(x)$

  $\neg$`Kill( Curiousity, Tuna)`

- Now what?
  - REFUTATION PROOF! (includes $\neg$ of goal)
  - Convert to "Clausal Form"
  - Resolve, seeking {}
  - (Return solution)

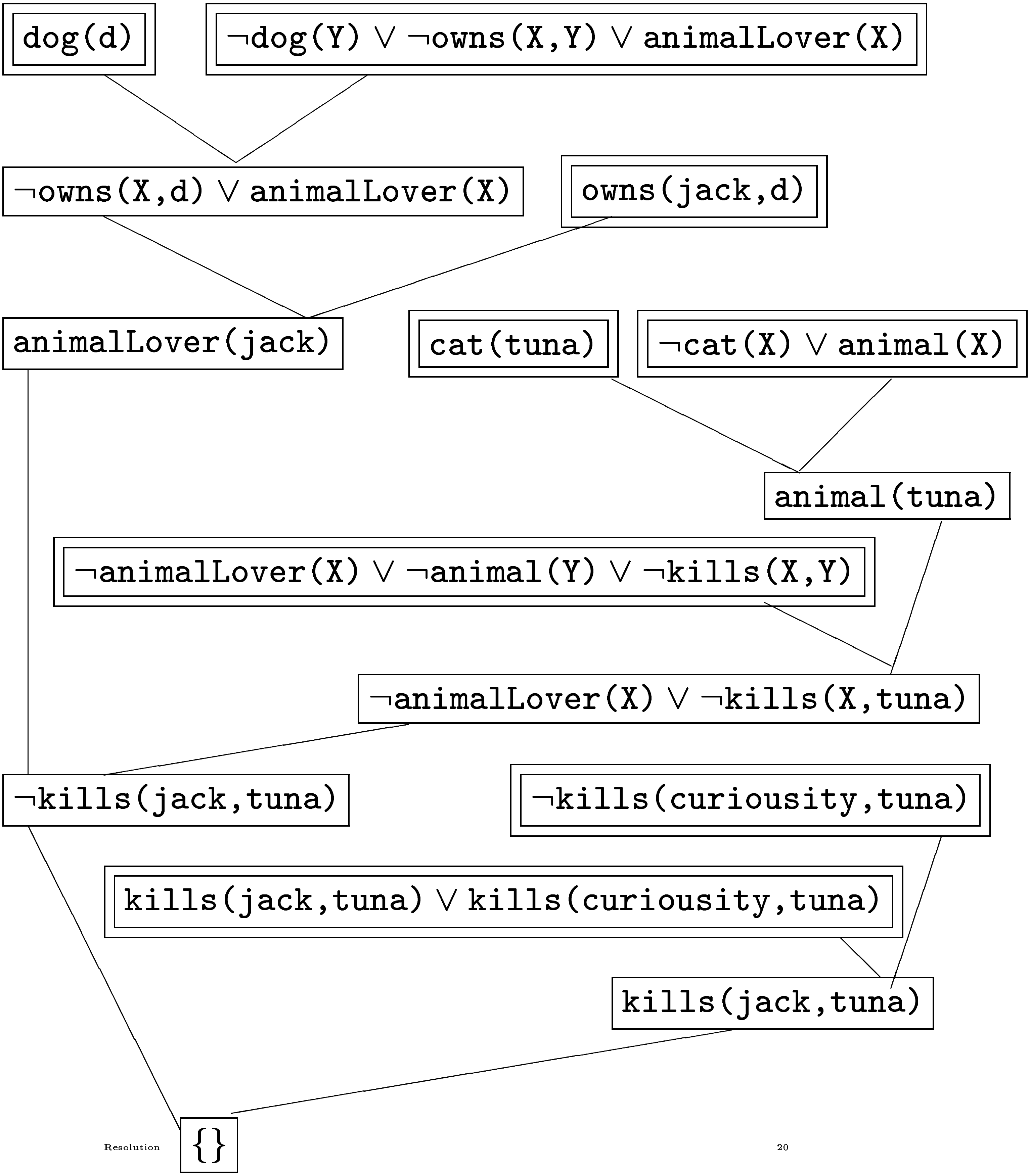
CNF Form

- . . . in Conjunctive Normal Form Form

  dog(d)
  owns(jack, d)
  ("d" is constant "naming" Jack's dog)
  $\neg$dog(Y) $\vee$ $\neg$owns(X, Y) $\vee$ AnimalLover(X)
  $\neg$animalLover(W) $\vee$ $\neg$animal(Y) $\vee$ $\neg$kills(W, Y)
  kills(jack, tuna) $\vee$ kills(curiosity, tuna)
  cat(tuna)
  $\neg$cat(Z) $\vee$ animal(Z)

  $\neg$kills(curiosity, tuna)

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


dog(d)
¬dog(Y) ∨ ¬owns(X,Y) ∨ animalLover(X)
¬owns(X,d) ∨ animalLover(X)
owns(jack,d)
animalLover(jack)
cat(tuna)
¬cat(X) ∨ animal(X)
animal(tuna)
¬animalLover(X) ∨ ¬animal(Y) ∨ ¬kills(X,Y)
¬animalLover(X) ∨ ¬kills(X,tuna)
¬kills(jack,tuna)
¬kills(curiousity,tuna)
kills(jack,tuna) ∨ kills(curiousity,tuna)
kills(jack,tuna)
{}

“Tricks”

1. **Refutation Proof**

2. **Normalization**: put in **CNF form**

   - **Skolemize** — remove $\exists$

     (by giving arbitrary, but unique name to $\exists$ objects)

   - remove quantifiers / $\Rightarrow$ / $\wedge$ / $\vee$ etc.

   [R&N pp. 281-282]

3. **Unification**: matching variables/ terms between clauses that look similar

- [Robinson 1965]

Comments on Resolution

- Formal Properties

  Sound, Complete

  Efficiency [exponential time]

- Strategies
  - Unit Preference
  - Ordered
  - Set of Support: Backward vs Forward Reasoning

  - Input
  - Linear

- Equality

. . . implementation $\Rightarrow$ Prolog

Inference Using Resolution

Given KB, σ

1. Convert KB to CNF: $CNF(KB)$
2. Convert $\neg\sigma$ to CNF: $CNF(\neg\sigma)$
3. $CNF(KB) \cup CNF(\neg\sigma) \vdash_R {}^? \{\}$

If succeed, then $KB \models \sigma$

If fail, then $KB \not\models \sigma$

As propositional:

⋆ sound

⋆ complete

⋆ decidable

But

⋆ Exponential time in general
(not "just" NP-hard)

⋆ Linear time for Horn clauses

⋆ Quadratic time for 2-CNF clauses

Properties of Resolution

+ Sound

  $KB \vdash_{RR} \sigma$ only if

  σ is true in EVERY world in which KB holds.

+ Complete

  $KB \vdash_{RR} \sigma$ whenever

  σ is true in EVERY world in which KB holds.

  (as $\vdash_R$ is $\Box$-complete)

– Semi-Decidable in Predicate Calculus

  $KB \overset{?}{\vdash}_{RR} \sigma$ $\begin{cases} \texttt{If Yes,} & \text{returns that answer eventually} \\ \texttt{If No,} & \text{may never return.} \end{cases}$

– Intractable

  Exponential in $|KB|$ for Propositional Logic

  (Linear for Proposition HORN)

- While complete, significant search problem!

  $\Rightarrow$ Many different search strategies:
  resolution strategies

Length of resolution proof?

- Can Resolution be FORCED
  to take **exponentially** many steps?
- Posed [Cook / Karp] $\approx$ 1971/72.
  ... Related to NP vs. $co - NP$ questions

  "Resolved" by Armin Haken 1985.
- Pigeon-Hole (PH) problem:
  Cannot place $n + 1$ pigeons in n holes (1/hole)
- PH takes **exponentially** many steps
  (for Resolution)
  no matter what order, strategy, ...
- Important:
  PH hidden in many practical problems
  Makes theorem proving/ reasoning expensive

  Contributed to recent move to model-based methods

Pigeon-Hole Principle

- $P_{i,j}$ for Pigeon i in hole j.

- Every pigeon is in some hole:

  $P_{1,1} \vee P_{1,2} \vee P_{1,3} \vee \ldots \vee P_{1,n}$

  $P_{2,1} \vee P_{2,2} \vee P_{2,3} \vee \ldots \vee P_{2,n}$

  $\vdots$

  $P_{(n+1),1} \vee P_{(n+1),2} \vee P_{(n+1),3} \vee \ldots \vee P_{(n+1),n}$

- Every pigeon is in at most one hole:

  $(\neg P_{1,1} \vee \neg P_{1,2})$, $(\neg P_{1,1} \vee \neg P_{1,3})$, $\ldots (\neg P_{1,(n-1)} \vee \neg P_{1,n})$

  $\vdots$

  $(\neg P_{2,1} \vee \neg P_{2,2}), \ldots, (\neg P_{2,(n-1)} \vee \neg P_{2,n})$

- Every hole has at most one pigeon:

  $(\neg P_{1,1} \vee \neg P_{2,1})$, $(\neg P_{1,1} \vee \neg P_{3,1})$, $\ldots$

  $(\neg P_{1,2} \vee \neg P_{2,2})$, $(\neg P_{1,2} \vee \neg P_{3,2})$, $\ldots$

  $\vdots$

Result

- Requires $O(n^3)$ clauses

Haken85: Resolution proof that
PH is inconsistent
requires dealing with at least exponential
number of clauses,
no matter how clauses are resolved!

$\Rightarrow$ **"Method can't count."**

- Can word in Predicate Calculus . . . same problem.

Generality; Choice Points

- As any theory can be translated to CNF and as resolution is □-complete,

  All deduction in terms of Resolution.

- As unification is functional,

  [MGU is unique up to variable names]

  only decision is

  Which (two literals in which) two clauses to (try to) Resolve?

- Eg:

  Insist on using a positive atomic literal:

  Forward reasoning

  Insist on using a negative atomic literal:

  Backward reasoning

  Insist on using an atomic literal:

  Unit Resolution (F or B)

  + Set of support, ancestry filtering, ordered(lock)

  . . .

Resolution Strategy I: Unit Preference

Goal: to find $\{\}$ (clasue w/ 0 literals)

- When $\gamma = \textit{Resolve}(\alpha,\ \beta)$
  $$|\gamma| = |\alpha| + |\beta| - 2$$
- If $|\alpha| = 4$ and $|\beta| = 3$, then $|\gamma| = 5$
  so $|\gamma| > |\alpha|,\ |\beta|$
  Is this progress?

But if $|\alpha| = 1$, then $|\textit{Resolve}(\alpha, \beta)| = |\beta| - 1$
PROGRESS towards 0 !

- **Unit Preference:**
  Given
  $$KB = \{\alpha,\ \beta,\ \ldots,\ \chi,\ \phi,\ \ldots\ \}$$

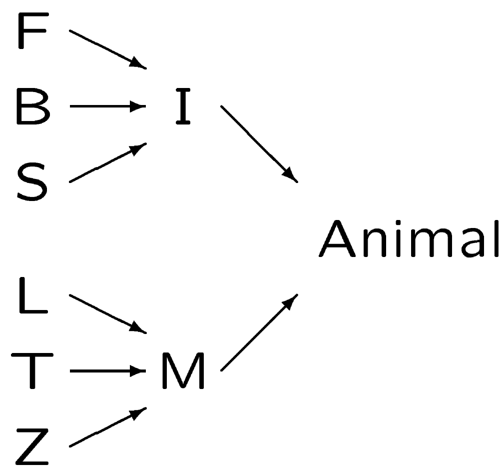
  May resolve α and β only if
  α is single literal ("unit clause")

- Does it work?

Unit Propagation
≈ Forward/Backward Reasoning

Query: `animal(ralph) ?`

```
  zebra(ralph)
¬fly(X)     ∨ insect(X)
¬bee(X)     ∨ insect(X)
¬spider(X)  ∨ insect(X)
¬insect(X)  ∨ animal(X)
¬lion(X)    ∨ mammal(X)
¬tiger(X)   ∨ mammal(X)
¬zebra(X)   ∨ mammal(X)
¬mammal(X)  ∨ animal(X)
```



```
¬f(X) ∨ i(X)
¬b(X) ∨ i(X)
¬s(X) ∨ i(X)
¬i(X) ∨ a(X)
¬l(X) ∨ m(X)
¬t(X) ∨ m(X)
¬z(X) ∨ m(X)
¬m(X) ∨ a(X)

¬a(r)
z(r)
```

```
¬f(X) ∨ i(X)
¬b(X) ∨ i(X)
¬s(X) ∨ i(X)
¬i(X) ∨ a(X)
¬l(X) ∨ m(X)
¬t(X) ∨ m(X)
¬z(X) ∨ m(X)
¬m(X) ∨ a(X)
z(r)
¬a(r)
```

Unit Resolution; Ordered Resolution

Can resolve P and Q only if . . .

Unit Preference: $|P| = 1$

STATUS: Not complete $\left\{ \begin{array}{cc} A \lor B, & A \lor \neg B \\ \neg A \lor B, & \neg A \lor \neg B \end{array} \right\}$

But . . . Refutation Complete for Horn clauses.

- Horn $\Leftrightarrow$ each clause has ≤ 1 positive literal

  Horn: $A \quad A \lor \neg B, \quad \neg B, \quad \neg A \lor \neg B, \quad \ldots$

  NotHorn: $A \lor B, \quad A \lor \neg Q \lor W$

Ordered Resolution: Literals in each clause are *ordered*:

$$P = \langle p_1 \lor p_2 \ \lor \ldots \rangle, \quad Q = \langle q_1 \lor q_2 \lor \ldots \rangle$$

. . . only if p_1 unifies with $\neg q_1$.

STATUS: Refutation complete for Horn

Resolution Strategies, II

Set of Support: Resolve P, Q only if $P \in S$
where $S \subset KB$ is "set of support".
...then add resolvent to S.

Complete if Consistent($KB - S$)

Backward Reasoning:
Initial Support: S = negated query $\neg\sigma$

Forward Reasoning:
Initial Support: S = original KB

Q: Which is better?

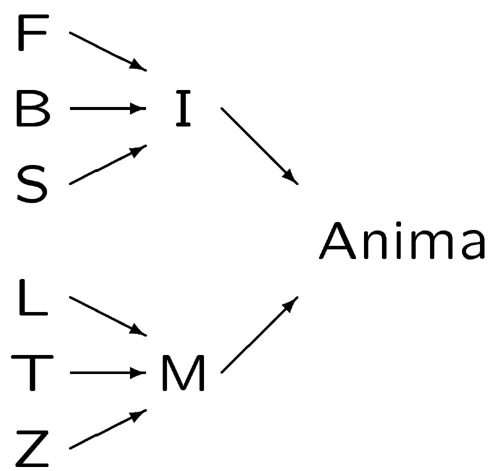
A: Depends on branching factor!

Set-of-Support: Backward Reasoning

Query: `animal(ralph) ?`

KB_1

```
zebra(ralph)
¬fly(X)    ∨ insect(X)
¬bee(X)    ∨ insect(X)
¬spider(X) ∨ insect(X)
¬insect(X) ∨ animal(X)
¬lion(X)   ∨ mammal(X)
¬tiger(X)  ∨ mammal(X)
¬zebra(X)  ∨ mammal(X)
¬mammal(X) ∨ animal(X)
```



```
z(r)
¬f(X) ∨ i(X)
¬b(X) ∨ i(X)
¬s(X) ∨ i(X)
¬i(X) ∨ a(X)
¬l(X) ∨ m(X)
¬t(X) ∨ m(X)
¬z(X) ∨ m(X)
¬m(X) ∨ a(X)
¬a(r)
```

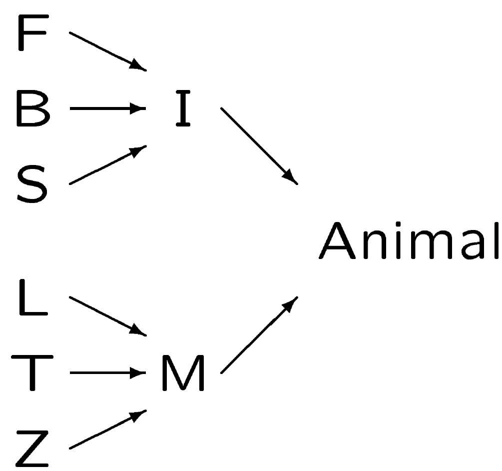
SofS

Set-of-Support: Forward Reasoning

Query: `animal(ralph) ?`

KB_1

```
zebra(ralph)
¬fly(X)    ∨ insect(X)
¬bee(X)    ∨ insect(X)
¬spider(X) ∨ insect(X)
¬insect(X) ∨ animal(X)
¬lion(X)   ∨ mammal(X)
¬tiger(X)  ∨ mammal(X)
¬zebra(X)  ∨ mammal(X)
¬mammal(X) ∨ animal(X)
```



```
z(r)
¬f(X) ∨ i(X)
¬b(X) ∨ i(X)
¬s(X) ∨ i(X)
¬i(X) ∨ a(X)
¬l(X) ∨ m(X)
¬t(X) ∨ m(X)
¬z(X) ∨ m(X)
¬m(X) ∨ a(X)
```

SofS {
```
¬a(r)
```

```
¬a(r)
```

SofS {
```
z(r)
¬f(X) ∨ i(X)
¬b(X) ∨ i(X)
¬s(X) ∨ i(X)
¬i(X) ∨ a(X)
¬l(X) ∨ m(X)
¬t(X) ∨ m(X)
¬z(X) ∨ m(X)
¬m(X) ∨ a(X)
```

Forward vs Backward Reasoning

- Here, both F- and B- reasoning were also
  Unit Preferences
  typically the case

- Here. . .
  Backward Reasoning required **8** steps
  Forward Reasoning required **3** steps

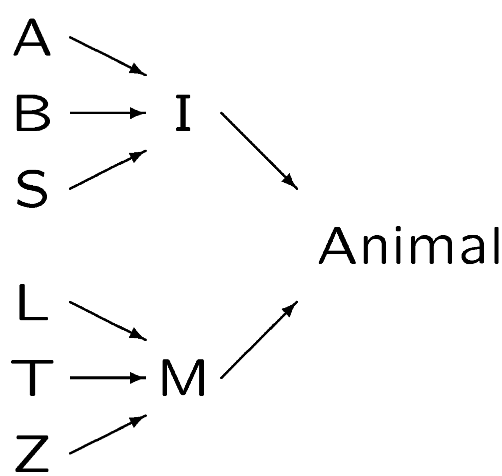
  Not always. . .

Forward vs Backward Reasoning

Query: `animal(ralph) ?`

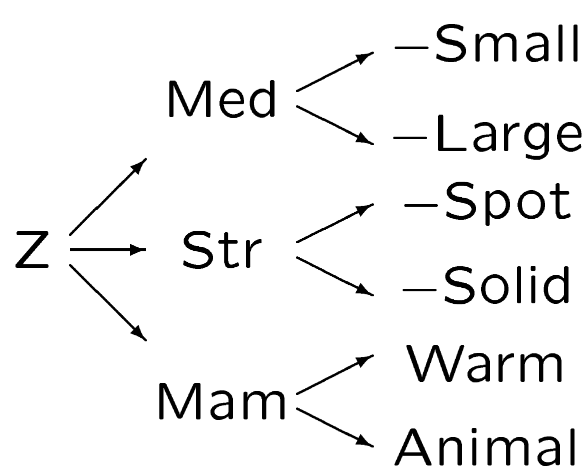
KB_1

```
 zebra(ralph)
¬ant(X)     ∨ insect(X)
¬bee(X)     ∨ insect(X)
¬spider(X)  ∨ insect(X)
¬insect(X)  ∨ animal(X)
¬lion(X)    ∨ mammal(X)
¬tiger(X)   ∨ mammal(X)
¬zebra(X)   ∨ mammal(X)
¬mammal(X)  ∨ animal(X)
     ...
```



KB_2

```
 zebra(ralph)
¬zebra(X)    ∨ medium(X)
¬zebra(X)    ∨ striped(X)
¬zebra(X)    ∨ mammal(X)
¬medium(X)   ∨ nonsmall(X)
¬medium(X)   ∨ nonlarge(X)
¬striped(X)  ∨ nonsolid(X)
¬striped(X)  ∨ nonspot(X)
¬mammal(X)   ∨ animal(X)
¬mammal(X)   ∨ warm(X)
     ...
```

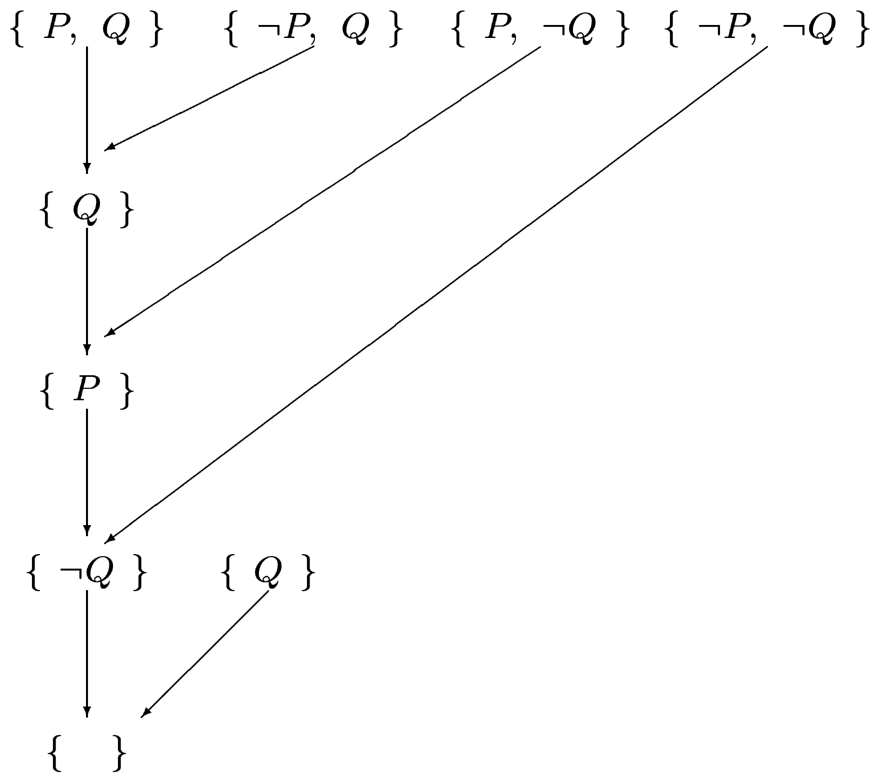


Resolution Strategies, III

Input Resolution: only if P in original KB

STATUS: Not complete.

Linear resolution: only if P in original KB or P is ancestor of Q in proof tree.



STATUS: Refutation complete

(if KB consistent, then $KB \cup \{\phi\}$ inconsistent iff LinRes, starting with ϕ, reaches $\{\}$)

Dealing with Equality

- Given axioms

```
russ = profG

happy(russ)
poor(profG)

confused(X) :- happy(X), poor(X).
```

expect to conclude

```
confused(russ)
```

- Prolog would *not*:
  Reduce `confused(russ)` to `poor(russ)`,
  but not match `poor(russ)` w/ `poor(profG)`.

? Could add rule:

```
poor(Y) :- poor(X), Y=X.
```

Comments on Equality

```
russ = profG.  happy(russ).  poor(profG).
confused(X) :- happy(X), poor(X).
```

- Need rule for each relation, function, . . .
- Rule

  `poor(Y) :- poor(X), X=Y.`

  would *NOT* work

  Reduce `confused(russ)` to `profG = russ`,
  NOT in knowledge base!

Fix:

```
X=Y :- Y=X.
X=X.
X=Z :- X=Y, Y=Z.
```

But. . .

```
poor(billGates)
  poor(russ), russ=billGates.
    russ=billGates
     billGates=russ
      russ=billGates
       billGates=russ
        ...
```

or worse:

```
russ=billGates
 russ=Y, Y=billGates
  profG=billGates
   profG=Y, Y=billGates
    Y=profG, Y=billGates
     russ=profG, russ=billGates
      russ=billGates
       ...
```

Sol'n: Need *lots* of control rules!

Wrap-Up wrt Equality

Note: `f(A)` does NOT unify with `f(B)`,
even if `A = B`

Eg: `Father(Russ) = Leonard`
`MorningStar = Venus`
`2+2 = 4`
...

Option#1: View "=" as std predicate

$$\forall x: \; x = x$$
$$\forall x, y: \; x = y \;\Rightarrow\; y = x$$
$$\forall x, y, z: \; x = y \;\wedge\; y = z \;\Rightarrow\; x = z$$

But also need...

$$\forall x, y: \; x = y \;\Leftrightarrow\; \mathtt{P}_1(x) = \mathtt{P}_1(y)$$
$$\forall x, y: \; x = y \;\Leftrightarrow\; \mathtt{P}_2(x) = \mathtt{P}_2(y)$$
...
$$\forall x_A, x_B, y_A, y_B: \; x_A = x_B \wedge y_A = y_B \;\Leftrightarrow\; \mathtt{F}_1(x_A, y_A) = \mathtt{F}_1(x_B, y_B)$$
...

for every predicate

+ search control problems...

Demodulation: For any terms x, y, z where $\mathtt{Unify}(x, z) = \theta$:

$$\frac{x = y, \;(\ldots z \ldots)}{(\ldots\mathtt{Subst}(\theta, y) \ldots)}$$

Paramodulation: ...do not know $x = y$,
but only "$x = y \;\vee\; P(x)$"