Completeness

- As explained earlier, Generalized Modus Ponens requires sentences to be in Horn form:
  - atomic, or
  - an implication with a conjunction of atomic sentences as the antecedent and an atom as the consequent.

- However, some sentences cannot be expressed in Horn form.
  - e.g.: $\forall x \neg$ bored_with_this_lecture (x)
  - Cannot be expressed in Horn form due to presence of negation.

Completeness

- A significant problem since Modus Ponens (with FC/BC) cannot automatically operate on such a sentence, and thus cannot use it in inference.
- Knowledge exists but cannot be used.
- Thus inference using Modus Ponens for ALL of first order logic is ***incomplete.***

Completeness

- However, Kurt Gödel in 1930-31 developed the **completeness theorem**, which shows that it is possible to find **complete** inference rules for First Order Logic.

- The theorem states:
  - any sentence entailed by a set of sentences can be proven from that set.

=> **Resolution Algorithm** which is a complete inference method.

Completeness

- The completeness theorem says that a sentence can be proved *if* it is entailed by another set of sentences.
- This is a big deal, since arbitrarily deeply nested functions combined with universal quantification make a potentially infinite search space.
- But entailment in first-order logic is only **semi-decidable**, meaning that if a sentence is not entailed by another set of sentences, it cannot necessarily be proven that it is not entailed.

Completeness in FOL

Procedure i is complete if and only if

$$KB \vdash_i \alpha \quad \text{whenever} \quad KB \models \alpha$$

Forward and backward chaining are complete for Horn KBs but incomplete for general first-order logic

E.g., from

$$\begin{array}{l} PhD(x) \;\Rightarrow\; HighlyQualified(x) \\ \neg PhD(x) \;\Rightarrow\; EarlyEarnings(x) \\ HighlyQualified(x) \;\Rightarrow\; Rich(x) \\ EarlyEarnings(x) \;\Rightarrow\; Rich(x) \end{array}$$

should be able to infer $Rich(Me)$, but FC/BC won't do it

Does a complete algorithm exist?

Historical note

450B.C.	Stoics	propositional logic, inference (maybe)
322B.C.	Aristotle	"syllogisms" (inference rules), quantifiers
1565	Cardano	probability theory (propositional logic + uncertainty)
1847	Boole	propositional logic (again)
1879	Frege	first-order logic
1922	Wittgenstein	proof by truth tables
1930	Gödel	$\exists$ complete algorithm for FOL
1930	Herbrand	complete algorithm for FOL (reduce to propositional)
1931	Gödel	$\neg\exists$ complete algorithm for arithmetic
1960	Davis/Putnam	"practical" algorithm for propositional logic
1965	Robinson	"practical" algorithm for FOL—resolution

Refutation Proof/Graph

```
¬parent(art,jon)      ¬ father(X, Y) ∨ parent(X, Y)
                \            /
     ¬ father (art, jon)      father (art,  jon)
                        \       /
                           []
```

Resolution

Entailment in first-order logic is only semidecidable:

can find a proof of α if $KB \models \alpha$

cannot always prove that $KB \not\models \alpha$

Cf. Halting Problem: proof procedure may be about to terminate with success or failure, or may go on for ever

Resolution is a refutation procedure:

to prove $KB \models \alpha$, show that $KB \wedge \neg\alpha$ is unsatisfiable

Resolution uses KB, $\neg\alpha$ in CNF (conjunction of clauses)

Resolution inference rule combines two clauses to make a new one:

$$\begin{tikzcd} C_1 \arrow[dr] & & C_2 \arrow[dl] \\ & C & \end{tikzcd}$$

Inference continues until an empty clause is derived (contradiction)

Resolution inference rule

Basic propositional version:

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

Full first-order version:

$$\frac{\begin{array}{c} p_1 \vee \ldots \; p_j \; \ldots \vee p_m, \\ q_1 \vee \ldots \; q_k \; \ldots \vee q_n \end{array}}{(p_1 \vee \ldots \; p_{j-1} \vee p_{j+1} \; \ldots p_m \vee q_1 \ldots \; q_{k-1} \vee q_{k+1} \; \ldots \vee q_n)\sigma}$$

where $p_j\sigma = \neg q_k\sigma$

For example,

$$\frac{\begin{array}{l} \neg Rich(x) \vee Unhappy(x) \\ Rich(Me) \end{array}}{Unhappy(Me)}$$

with $\sigma = \{x/Me\}$

Remember: normal forms

Other approaches to inference use syntactic operations on sentences, often expressed in standardized forms

<u>Conjunctive Normal Form</u> (CNF—universal)

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

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

"product of sums of simple variables or negated simple variables"

<u>Disjunctive Normal Form</u> (DNF—universal)

$$\textit{disjunction} \text{ of } \underbrace{\textit{conjunctions} \text{ of } \textit{literals}}_{\textit{terms}}$$

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

"sum of products of simple variables or negated simple variables"

<u>Horn Form</u> (restricted)

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 $(C \wedge D) \Rightarrow B$

Conjunctive normal form

Literal = (possibly negated) atomic sentence, e.g., $\neg Rich(Me)$

Clause = disjunction of literals, e.g., $\neg Rich(Me) \vee Unhappy(Me)$

The KB is a conjunction of clauses

Any FOL KB can be converted to CNF as follows:

1. Replace $P \Rightarrow Q$ by $\neg P \vee Q$
2. Move $\neg$ inwards, e.g., $\neg \forall x\, P$ becomes $\exists x\, \neg P$
3. Standardize variables apart, e.g., $\forall x\, P \vee \exists x\, Q$ becomes $\forall x\, P \vee \exists y\, Q$
4. Move quantifiers left in order, e.g., $\forall x\, P \vee \exists x\, Q$ becomes $\forall x \exists y\, P \vee Q$
5. Eliminate $\exists$ by Skolemization (next slide)
6. Drop universal quantifiers
7. Distribute $\wedge$ over $\vee$, e.g., $(P \wedge Q) \vee R$ becomes $(P \vee Q) \wedge (P \vee R)$
8. Separate into different clauses that contain ONLY disjunction
9. Standardize variables so that each clause uses different variables

Converting to clausal form

- The canonical (standard) form for resolution is **Conjunctive Normal Form** (conjunction of disjunctions), or equivalently, **Implicative Normal Form** (conjunction implies disjunction)
- Once you have the CNF, conversion to INF is relatively easy
- For example, say the CNF sentence is:
  - $\neg A(x) \vee \neg B(y) \vee C(y) \vee D(y,z)$
- Implicative Normal Form: Uses only positive literals
  - Group all the negative literals into the left side of the implication
  - $A(x) \wedge B(y) \rightarrow C(y) \vee D(y,z)$

Converting sentences to clausal form

1. Eliminate all $\leftrightarrow$ connectives

   $(P \leftrightarrow Q)$ rewritten as $((P \rightarrow Q) \wedge (Q \rightarrow P))$

2. Eliminate all $\rightarrow$ connectives

   $(P \rightarrow Q)$ rewritten as $(\sim P \vee Q)$

3. Reduce the scope of each negation symbol to a single predicate

   $\sim\sim P$ rewritten as P

   $\sim(P \vee Q)$ rewritten as $\sim P \wedge \sim Q$

   $\sim(P \wedge Q)$ rewritten $\sim P \vee \sim Q$

   $\sim(\forall x)P$ rewritten as $(\exists x)\sim P$

   $\sim(\exists x)P$ rewritten as $(\forall x)\sim P$

4. Standardize variables: rename all variables so that each quantifier has its own unique variable name

Skolemization Step 5 – Complicated by Quantifiers

$\exists x \, Rich(x)$ becomes $Rich(G1)$ where $G1$ is a new "Skolem constant"

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

More tricky when $\exists$ is inside $\forall$

E.g., "Everyone has a heart"

$$\forall \, x \;\; Person(x) \;\Rightarrow\; \exists \, y \;\; Heart(y) \wedge Has(x, y)$$

Incorrect:

$$\forall \, x \;\; Person(x) \;\Rightarrow\; Heart(H1) \wedge Has(x, H1)$$

Correct:

$$\forall \, x \;\; Person(x) \;\Rightarrow\; Heart(H(x)) \wedge Has(x, H(x))$$

where H is a new symbol ("Skolem function")

Skolem function arguments: all enclosing universally quantified variables

Final Steps of Converting sentences to clausal form

6. Remove universal quantifiers by (1) moving them all to the left end; (2) making the scope of each the entire sentence; and (3) dropping the "prefix" part

   Ex: $(\forall x)P(x) \Longrightarrow P(x)$

7. Put into conjunctive normal form (conjunction of disjunctions)

   $(P \wedge Q) \vee R \Longrightarrow (P \vee R) \wedge (Q \vee R)$

   $(P \vee Q) \vee R \Longrightarrow (P \vee Q \vee R)$

8. Split conjuncts into a separate clauses

9. Standardize variables so each clause contains only variable names that do not occur in any other clause

EXAMPLE: Convert the following FOL sentence to CNF by applying the 9 steps discussed earlier

$$(\forall x)\ [\ P(x) \rightarrow \ \{\ (\forall y)(P(y) \rightarrow P(f(x,y))) \wedge \neg(\forall y)(Q(x,y) \rightarrow P(y))\ \}\ ]$$

Examples: Converting FOL sentences to clause form…

Convert the sentence

1. $(\forall x)(P(x) \rightarrow ((\forall y)(P(y) \rightarrow P(f(x,y))) \wedge \neg(\forall y)(Q(x,y) \rightarrow P(y))))$

`(like A => B ^ C)`

2. Eliminate => $(\forall x)(\neg P(x) \vee ((\forall y)(\neg P(y) \vee P(f(x,y))) \wedge \neg(\forall y)(\neg Q(x,y) \vee P(y))))$

3. Reduce scope of negation

   $(\forall x)(\neg P(x) \vee ((\forall y)(\neg P(y) \vee P(f(x,y))) \wedge (\exists y)(Q(x,y) \wedge \neg P(y))))$

4. Standardize variables apart

   $(\forall x)(\neg P(x) \vee ((\forall y)(\neg P(y) \vee P(f(x,y))) \wedge (\exists z)(Q(x,z) \wedge \neg P(z))))$

Examples: Converting FOL sentences to clause form…

5. Eliminate existential quantification – Skolem function g introduced:
   $(\forall x)(\neg P(x) \vee((\forall y)(\neg P(y) \vee P(f(x,y))) \wedge (Q(x,g(x)) \wedge \neg P(g(x)))))$

6. Drop universal quantification symbols
   $(\neg P(x) \vee ((\neg P(y) \vee P(f(x,y))) \wedge (Q(x,g(x)) \wedge \neg P(g(x)))))$

7. Convert to conjunction of disjunctions
   $(\neg P(x) \vee \neg P(y) \vee P(f(x,y))) \wedge (\neg P(x) \vee Q(x,g(x))) \wedge (\neg P(x) \vee \neg P(g(x)))$

Examples: Converting FOL sentences to clause form…

8. Create separate clauses containing only disjunctions

   $\neg P(x) \vee \neg P(y) \vee P(f(x,y))$

   $\neg P(x) \vee Q(x,g(x))$

   $\neg P(x) \vee \neg P(g(x))$

9. Standardize variables so that each clause uses different variables

   $\neg P(x) \vee \neg P(y) \vee P(f(x,y))$

   $\neg P(z) \vee Q(z,g(z))$

   $\neg P(w) \vee \neg P(g(w))$

Resolution proof

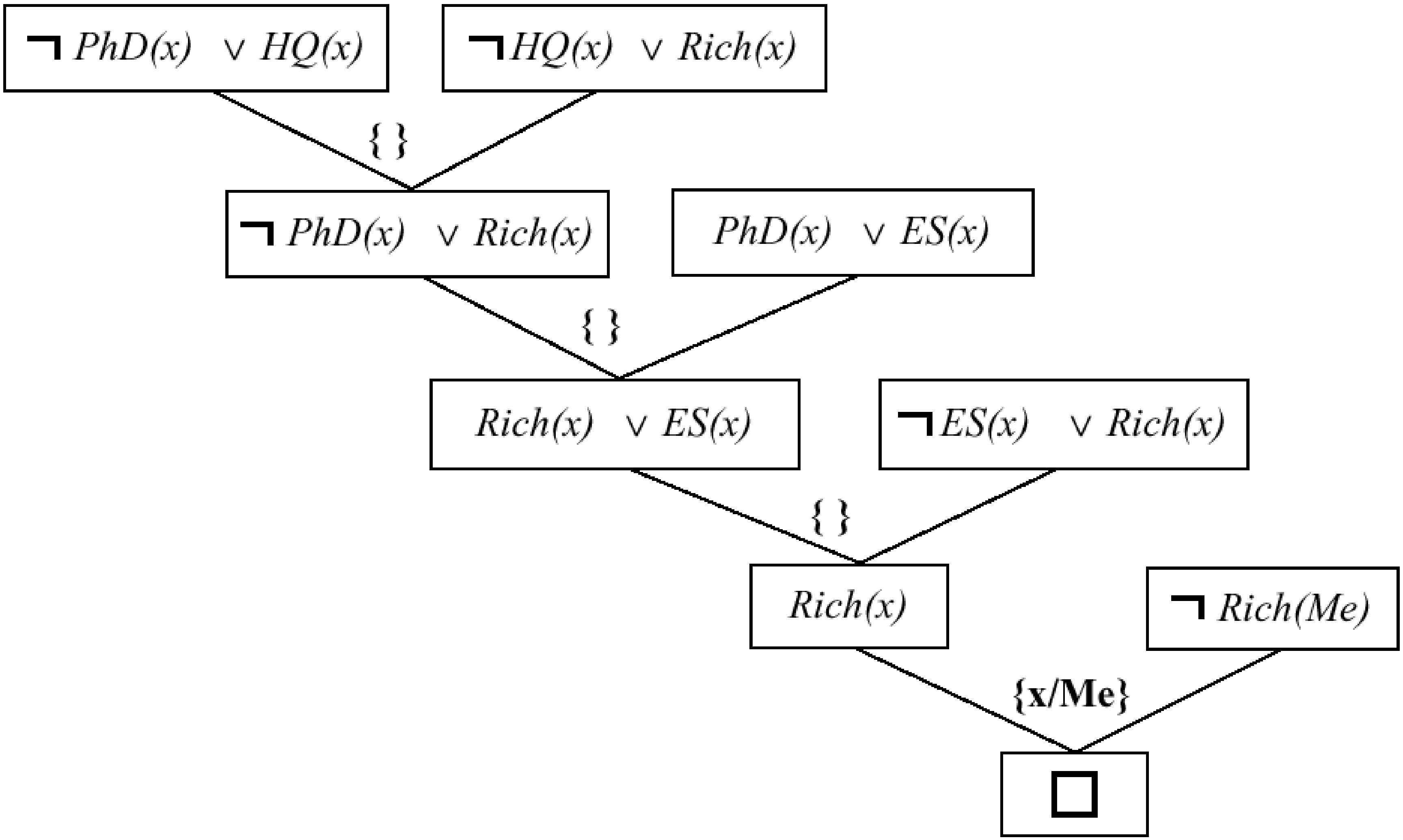
To prove α:

- negate it
- convert to CNF
- add to CNF KB
- infer contradiction

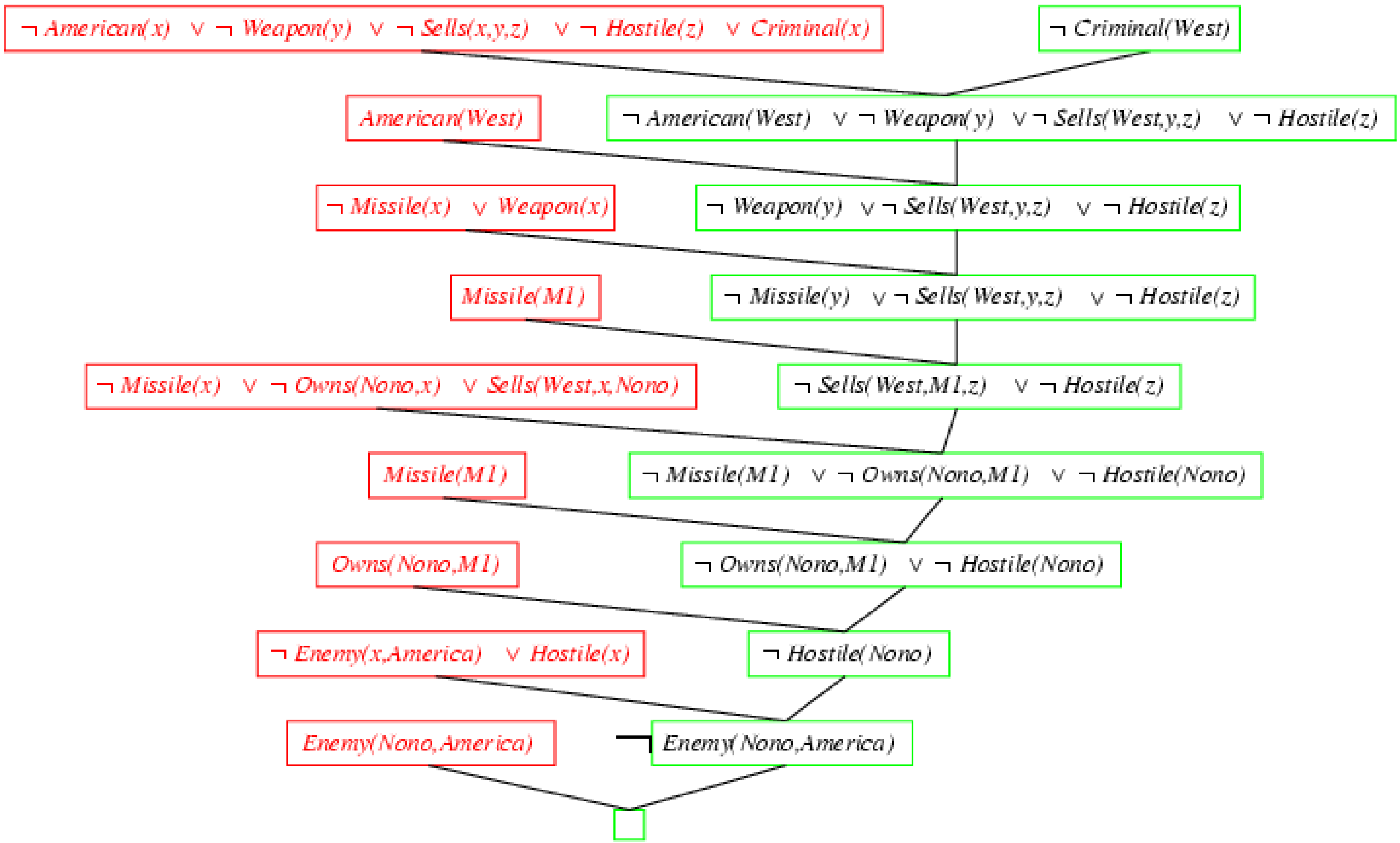
E.g., to prove $Rich(me)$, add $\neg Rich(me)$ to the CNF KB

$$\neg PhD(x) \vee HighlyQualified(x)$$
$$PhD(x) \vee EarlyEarnings(x)$$
$$\neg HighlyQualified(x) \vee Rich(x)$$
$$\neg EarlyEarnings(x) \vee Rich(x)$$

Resolution proof



Resolution proof: definite clauses



Inference in First-Order Logic

- Canonical forms for resolution

Conjunctive Normal Form (CNF)

$$\neg P(w) \vee Q(w)$$
$$P(x) \vee R(x)$$
$$\neg Q(y) \vee S(y)$$
$$\neg R(z) \vee S(z)$$

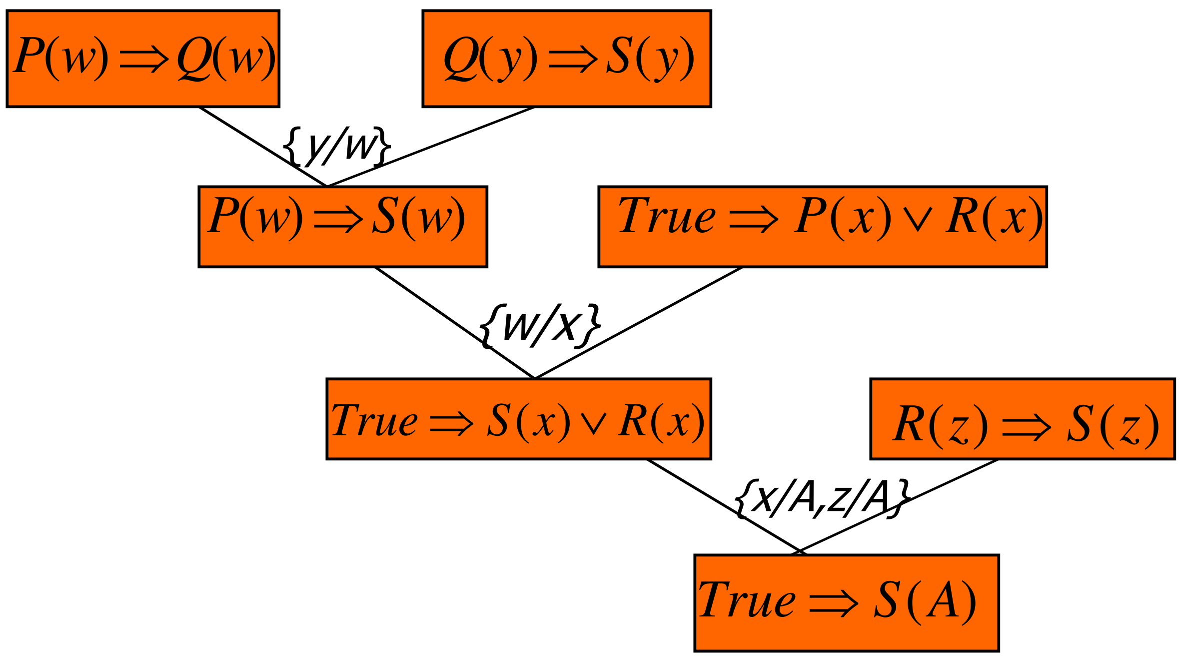
Implicative Normal Form (INF)

$$P(w) \Rightarrow Q(w)$$
$$True \Rightarrow P(x) \vee R(x)$$
$$Q(y) \Rightarrow S(y)$$
$$R(z) \Rightarrow S(z)$$

Inference in First-Order Logic

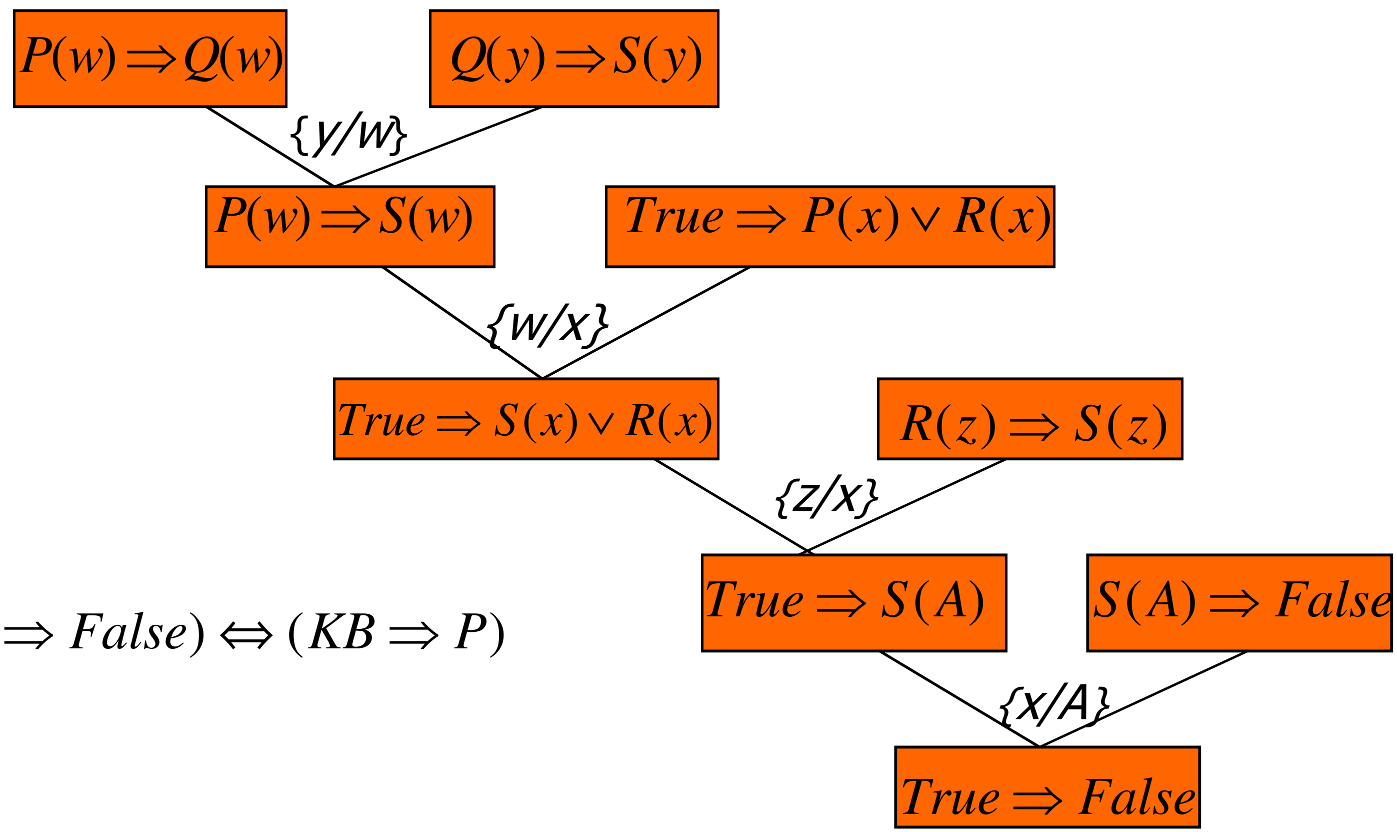
- Resolution Proofs

In a forward- or backward-chaining algorithm, just as Modus Ponens.



Inference in First-Order Logic

- Refutation



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

Example of Refutation Proof (in conjunctive normal form)

(1)	Cats like fish	$\neg cat\ (x) \vee likes\ (x,fish)$
(2)	Cats eat everything they like	$\neg cat\ (y) \vee \neg likes\ (y,z) \vee eats\ (y,z)$
(3)	Josephine is a cat.	$cat\ (jo)$
(4)	Prove: Josephine eats fish.	$eats\ (jo,fish)$

Backward Chaining

Negation of goal wff: $\neg$ eats(jo, fish)

$\neg$ eats(jo, fish) $\qquad\qquad$ $\neg$ cat(y) $\vee$ $\neg$likes(y, z) $\vee$ eats(y, z)

θ = {y/jo, z/fish}

$\neg$ cat(jo) $\vee$ $\neg$likes(jo, fish) $\qquad\qquad$ cat(jo)

$\theta = \varnothing$

$\neg$ cat(x) $\vee$ likes(x, fish) $\qquad\qquad$ $\neg$ likes(jo, fish)

θ = {x/jo}

$\neg$ cat(jo) $\qquad\qquad$ cat(jo)

$\perp$ (contradiction)

Forward chaining

cat (jo) ¬cat (X) ∨ likes (X,fish)

\ /

likes (jo,fish) ¬cat (Y) ∨ ¬likes (Y,Z) ∨ eats (Y,Z)

\ /

¬cat (jo) ∨ eats (jo,fish) cat (jo)

\ /

eats (jo,fish) ¬ eats (jo,fish)

\ /

[]

Question:

- When would you use forward chaining? What about backward chaining?

- A:
  - FC: If expert needs to gather information before any inferencing
  - BC: If expert has a hypothetical solution