

For Friday

- Read Chapter 10, sections 1 and 2
- Prolog Handout 4

Length of a List

- Definition of length/2

length([], 0).

length([_ | Tail], N) :-
 length(Tail, N1),
 N is 1 + N1.

- Note: all loops must be implemented via recursion

Counting Loops

- Definition of sum/3
sum(Begin, End, Sum) :-
 sum(Begin, End, Begin, Sum).
sum(X, X, Y, Y).
sum(Begin, End, Sum1, Sum) :-
 Begin < End,
 Next is Begin + 1,
 Sum2 is Sum1 + Next,
 sum(Next, End, Sum2, Sum).

The Cut (!)

- A way to prevent backtracking.
- Used to simplify and to improve efficiency.

Negation

- Can't say something is NOT true
- Use a **closed world assumption**
- Not simply means “I can't prove that it is true”

Dynamic Predicates

- A way to write self-modifying code, in essence.
- Typically just storing data using Prolog's built-in predicate database.
- Dynamic predicates must be declared as such.

Using Dynamic Predicates

- assert and variants
- retract
 - Fails if there is no clause to retract
- retractall
 - Doesn't fail if no clauses

Resolution

- Propositional version.

$\{a \vee b, \neg b \vee c\} \vdash a \vee c$ OR $\{\neg a \Rightarrow b, b \Rightarrow c\} \vdash \neg a \Rightarrow c$

Reasoning by cases OR transitivity of implication

- First-order form

– For two literals p_j and q_k in two clauses

- $p_1 \vee \dots \vee p_j \vee \dots \vee p_m$

- $q_1 \vee \dots \vee q_k \vee \dots \vee q_n$

such that $\theta = \text{UNIFY}(p_j, \neg q_k)$, derive

$\text{SUBST}(\theta, p_1 \vee \dots \vee p_{j-1} \vee p_{j+1} \vee \dots \vee p_m \vee q_1 \vee \dots \vee q_{k-1} \vee q_{k+1} \vee \dots \vee q_n)$

Implication form

- Can also be viewed in implicational form where all negated literals are in a conjunctive antecedent and all positive literals in a disjunctive conclusion.

$$\neg p_1 \vee \dots \vee \neg p_m \vee q_1 \vee \dots \vee q_n \Leftrightarrow$$

$$p_1 \wedge \dots \wedge p_m \Rightarrow q_1 \vee \dots \vee q_n$$

Conjunctive Normal Form (CNF)

- For resolution to apply, all sentences must be in conjunctive normal form, a conjunction of disjunctions of literals

$$(a_1 \vee \dots \vee a_m) \wedge$$

$$(b_1 \vee \dots \vee b_n) \wedge$$

$$\dots \wedge$$

$$(x_1 \vee \dots \vee x_v)$$

- Representable by a set of clauses (disjunctions of literals)
- Also representable as a set of implications (INF).

Example

Initial

$$P(x) \Rightarrow Q(x)$$

$$\neg P(x) \Rightarrow R(x)$$

$$Q(x) \Rightarrow S(x)$$

$$R(x) \Rightarrow S(x)$$

CNF

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

$$P(x) \vee R(x)$$

$$\neg Q(x) \vee S(x)$$

$$\neg R(x) \vee S(x)$$

INF

$$P(x) \Rightarrow Q(x)$$

$$\text{True} \Rightarrow P(x) \vee R(x)$$

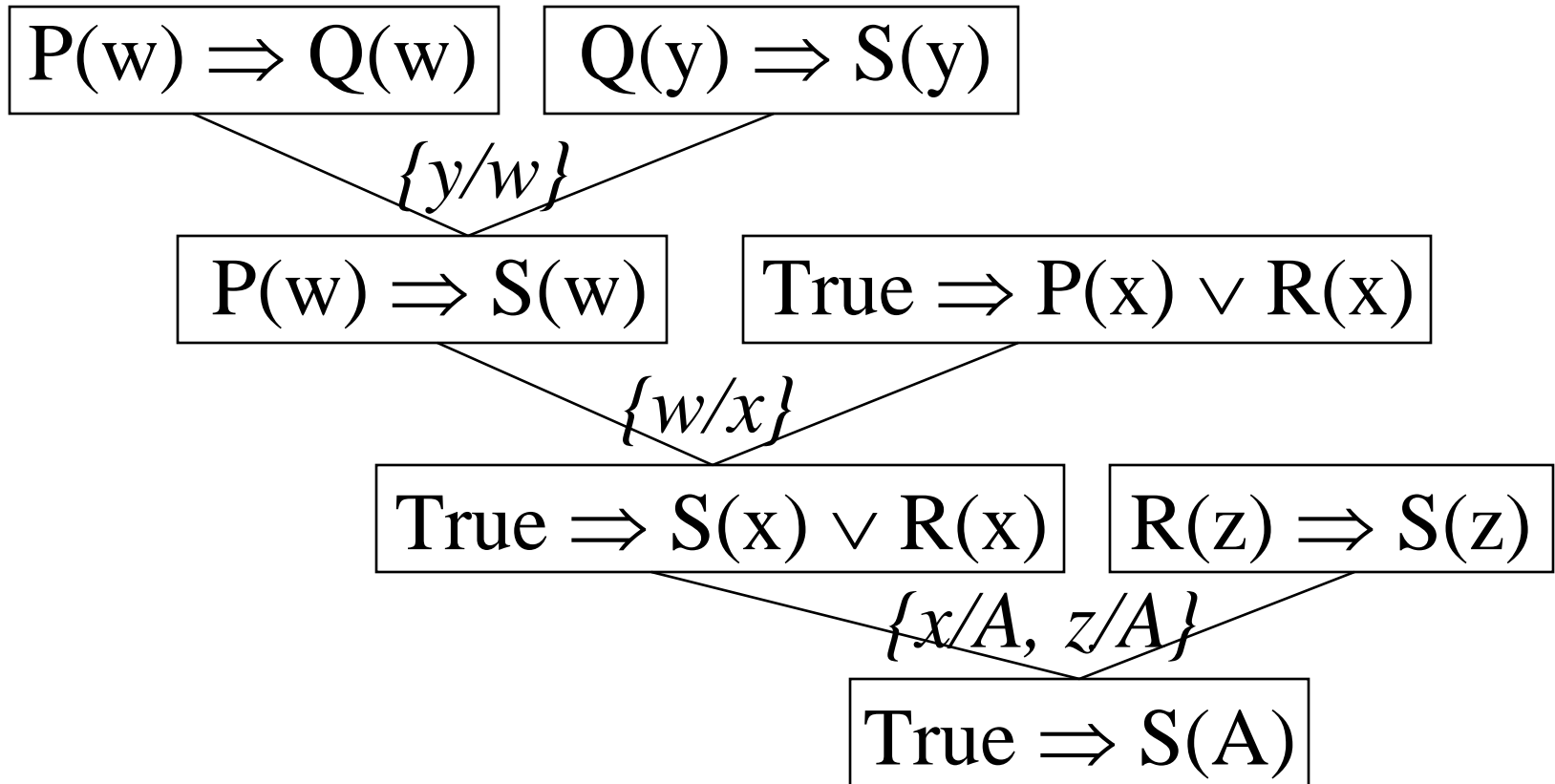
$$Q(x) \Rightarrow S(x)$$

$$R(x) \Rightarrow S(x)$$

Resolution Proofs

- INF (CNF) is more expressive than Horn clauses.
- Resolution is simply a generalization of modus ponens.
- As with modus ponens, chains of resolution steps can be used to construct proofs.
- Factoring removes redundant literals from clauses
 - $S(A) \vee S(A) \rightarrow S(A)$

Sample Proof

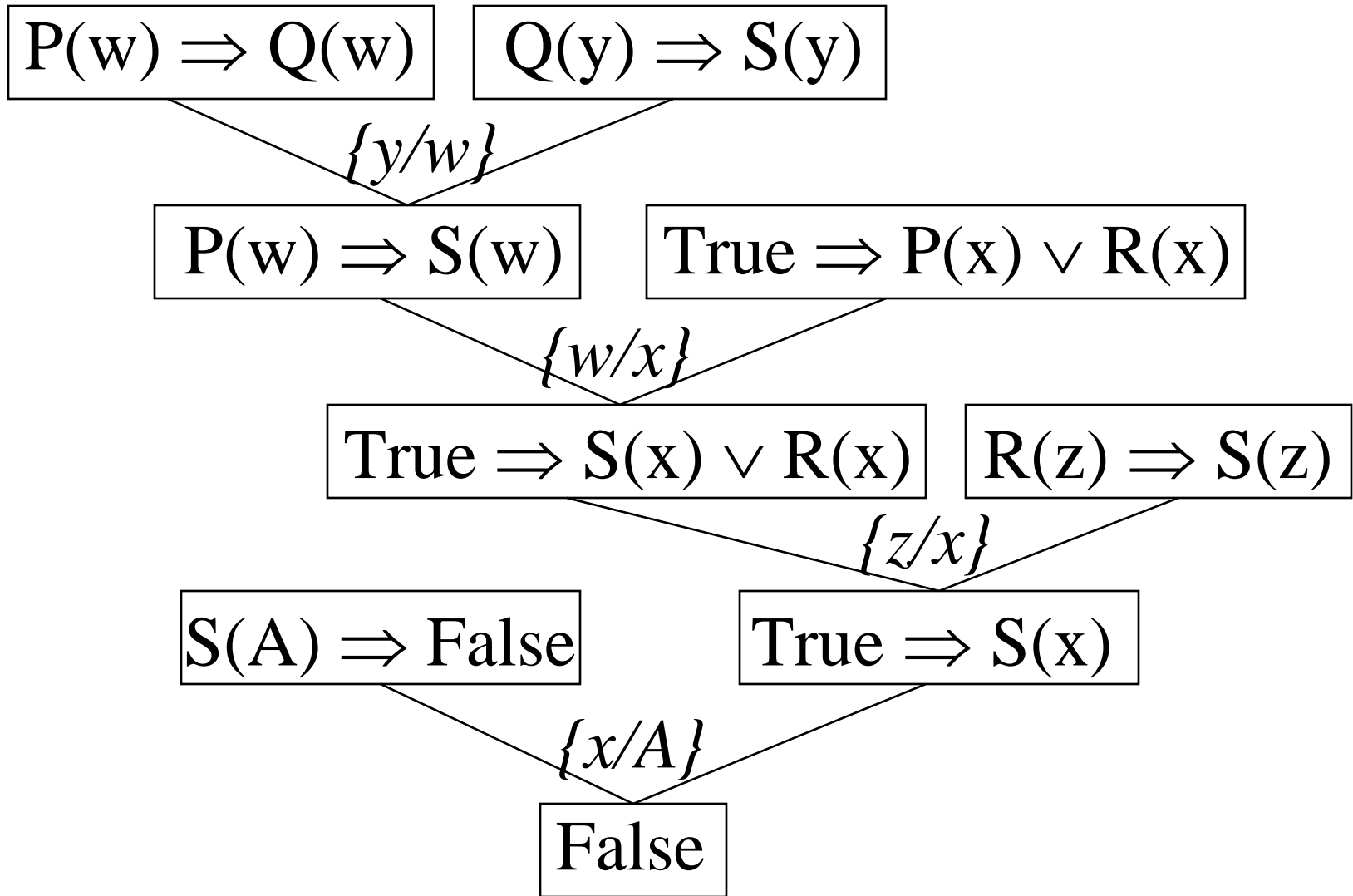


Refutation Proofs

- Unfortunately, resolution proofs in this form are still incomplete.
- For example, it cannot prove any tautology (e.g. $P \vee \neg P$) from the empty KB since there are no clauses to resolve.
- Therefore, use proof by **contradiction** (refutation, reductio ad absurdum). Assume the negation of the theorem P and try to derive a contradiction (False, the empty clause).

$$- (KB \wedge \neg P \Rightarrow \text{False}) \Leftrightarrow KB \Rightarrow P$$

Sample Proof



Resolution Theorem Proving

- Convert sentences in the KB to CNF (clausal form)
- Take the negation of the proposed theorem (query), convert it to CNF, and add it to the KB.
- Repeatedly apply the resolution rule to derive new clauses.
- If the empty clause (False) is eventually derived, stop and conclude that the proposed theorem is true.

Conversion to Clausal Form

- Eliminate implications and biconditionals by rewriting them.

$$p \Rightarrow q \rightarrow \neg p \vee q$$

$$p \Leftrightarrow q \rightarrow (\neg p \vee q) \wedge (p \vee \neg q)$$

- Move \neg inward to only be a part of literals by using deMorgan's laws and quantifier rules.

$$\neg(p \vee q) \rightarrow \neg p \wedge \neg q$$

$$\neg(p \wedge q) \rightarrow \neg p \vee \neg q$$

$$\neg \forall x p \rightarrow \exists x \neg p$$

$$\neg \exists x p \rightarrow \forall x \neg p$$

$$\neg \neg p \rightarrow p$$

Conversion continued

- Standardize variables to avoid use of the same variable name by two different quantifiers.

$$\forall x P(x) \vee \exists x P(x) \rightarrow \forall x_1 P(x_1) \vee \exists x_2 P(x_2)$$

- Move quantifiers left while maintaining order. Renaming above guarantees this is a truth-preserving transformation.

$$\forall x_1 P(x_1) \vee \exists x_2 P(x_2) \rightarrow \forall x_1 \exists x_2 (P(x_1) \vee P(x_2))$$

Conversion continued

- Skolemize: Remove existential quantifiers by replacing each existentially quantified variable with a Skolem constant or Skolem function as appropriate.
 - If an existential variable is not within the scope of any universally quantified variable, then replace every instance of the variable with the same unique constant that does not appear anywhere else.

$$\exists x (P(x) \wedge Q(x)) \rightarrow P(C_1) \wedge Q(C_1)$$

- If it is within the scope of n universally quantified variables, then replace it with a unique n-ary function over these universally quantified variables.

$$\forall x_1 \exists x_2 (P(x_1) \vee P(x_2)) \rightarrow \forall x_1 (P(x_1) \vee P(f_1(x_1)))$$

$$\forall x (\text{Person}(x) \Rightarrow \exists y (\text{Heart}(y) \wedge \text{Has}(x,y))) \rightarrow$$

$$\forall x (\text{Person}(x) \Rightarrow \text{Heart}(\text{HeartOf}(x)) \wedge \text{Has}(x, \text{HeartOf}(x)))$$

- Afterwards, all variables can be assumed to be universally quantified, so remove all quantifiers.

Conversion continued

- Distribute \wedge over \vee to convert to conjunctions of clauses

$$(a \wedge b) \vee c \rightarrow (a \vee c) \wedge (b \vee c)$$

$$(a \wedge b) \vee (c \wedge d) \rightarrow (a \vee c) \wedge (b \vee c) \wedge (a \vee d) \wedge (b \vee d)$$

– Can exponentially expand size of sentence.

- Flatten nested conjunctions and disjunctions to get final CNF

$$(a \vee b) \vee c \rightarrow (a \vee b \vee c)$$

$$(a \wedge b) \wedge c \rightarrow (a \wedge b \wedge c)$$

- Convert clauses to implications if desired for readability

$$(\neg a \vee \neg b \vee c \vee d) \rightarrow a \wedge b \Rightarrow c \vee d$$

Sample Clause Conversion

$$\forall x((\text{Prof}(x) \vee \text{Student}(x)) \Rightarrow (\exists y(\text{Class}(y) \wedge \text{Has}(x,y)) \wedge \exists y(\text{Book}(y) \wedge \text{Has}(x,y))))$$

$$\forall x(\neg(\text{Prof}(x) \vee \text{Student}(x)) \vee (\exists y(\text{Class}(y) \wedge \text{Has}(x,y)) \wedge \exists y(\text{Book}(y) \wedge \text{Has}(x,y))))$$

$$\forall x((\neg\text{Prof}(x) \wedge \neg\text{Student}(x)) \vee (\exists y(\text{Class}(y) \wedge \text{Has}(x,y)) \wedge \exists y(\text{Book}(y) \wedge \text{Has}(x,y))))$$

$$\forall x((\neg\text{Prof}(x) \wedge \neg\text{Student}(x)) \vee (\exists y(\text{Class}(y) \wedge \text{Has}(x,y)) \wedge \exists z(\text{Book}(z) \wedge \text{Has}(x,z))))$$

$$\forall x \exists y \exists z ((\neg\text{Prof}(x) \wedge \neg\text{Student}(x)) \vee ((\text{Class}(y) \wedge \text{Has}(x,y)) \wedge (\text{Book}(z) \wedge \text{Has}(x,z))))$$

$$(\neg\text{Prof}(x) \wedge \neg\text{Student}(x)) \vee (\text{Class}(f(x)) \wedge \text{Has}(x,f(x)) \wedge \text{Book}(g(x)) \wedge \text{Has}(x,g(x)))$$

Clause Conversion

$(\neg \text{Prof}(x) \wedge \neg \text{Student}(x)) \vee (\text{Class}(f(x)) \wedge \text{Has}(x, f(x)) \wedge \text{Book}(g(x)) \wedge \text{Has}(x, g(x)))$

$(\neg \text{Prof}(x) \vee \text{Class}(f(x))) \wedge$

$(\neg \text{Prof}(x) \vee \text{Has}(x, f(x))) \wedge$

$(\neg \text{Prof}(x) \vee \text{Book}(g(x))) \wedge$

$(\neg \text{Prof}(x) \vee \text{Has}(x, g(x))) \wedge$

$(\neg \text{Student}(x) \vee \text{Class}(f(x))) \wedge$

$(\neg \text{Student}(x) \vee \text{Has}(x, f(x))) \wedge$

$(\neg \text{Student}(x) \vee \text{Book}(g(x))) \wedge$

$(\neg \text{Student}(x) \vee \text{Has}(x, g(x)))$

Sample Resolution Problem

- Jack owns a dog.
- Every dog owner is an animal lover.
- No animal lover kills an animal.
- Either Jack or Curiosity killed Tuna the cat.
- Did Curiosity kill the cat?

In Logic Form

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

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

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

D) $\text{Kills}(\text{Jack}, \text{Tuna}) \vee \text{Kills}(\text{Curiosity}, \text{Tuna})$

E) $\text{Cat}(\text{Tuna})$

F) $\forall x (\text{Cat}(x) \Rightarrow \text{Animal}(x))$

Query: $\text{Kills}(\text{Curiosity}, \text{Tuna})$

In Normal Form

A1) Dog(D)

A2) Owns(Jack,D)

B) Dog(y) \wedge Owns(x,y) \Rightarrow AnimalLover(x)

C) AnimalLover(x) \wedge Animal(y) \wedge Kills(x,y)
 \Rightarrow False

D) Kills(Jack,Tuna) \vee Kills(Curiosity,Tuna)

E) Cat(Tuna)

F) Cat(x) \Rightarrow Animal(x)

Query: Kills(Curiosity,Tuna) \Rightarrow False

Resolution Proof

