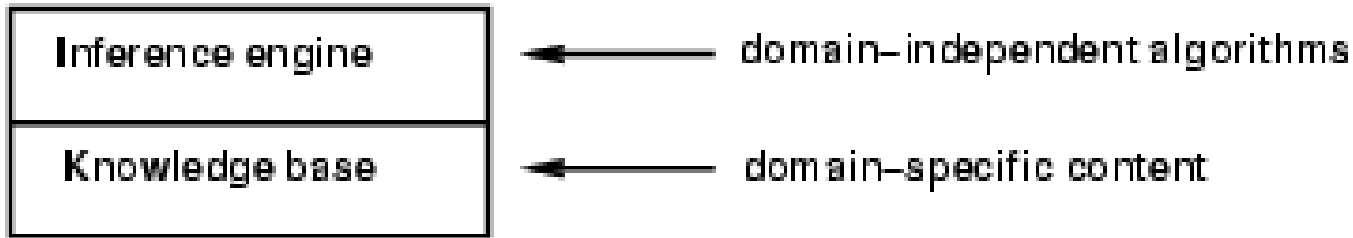


Logic

- Use mathematical deduction to derive new knowledge.
- Predicate Logic is a powerful representation scheme used by many AI programs.
- Propositional logic is much simpler (less powerful).

Knowledge bases



- Knowledge base = set of **sentences** in a **formal** language
- **Declarative** approach to building an agent (or other system):
 - Tell it what it needs to know
- Then it can **Ask** itself what to do - answers should follow from the KB
- Agents can be viewed at the **knowledge level**
 - i.e., what they know, regardless of how implemented
- Or at the **implementation level**
 - i.e., data structures in KB and algorithms that manipulate them

A simple knowledge-based agent

```
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
```

- The agent must be able to:
 - Represent states, actions, etc.
 - Incorporate new percepts
 - Update internal representations of the world
 - Deduce hidden properties of the world
 - Deduce appropriate actions

Logic in general

- **Logics** are formal languages for representing information such that conclusions can be drawn
- **Syntax** defines the sentences in the language
- **Semantics** define the "meaning" of sentences;
 - i.e., define **truth** of a sentence in a world
- E.g., the language of arithmetic
 - $x+2 \geq y$ is a sentence; $x^2+y > \{\}$ is not a sentence
 - $x+2 \geq y$ is true iff the number $x+2$ is no less than the number y
 - $x+2 \geq y$ is true in a world where $x = 7, y = 1$
 - $x+2 \geq y$ is false in a world where $x = 0, y = 6$

Entailment

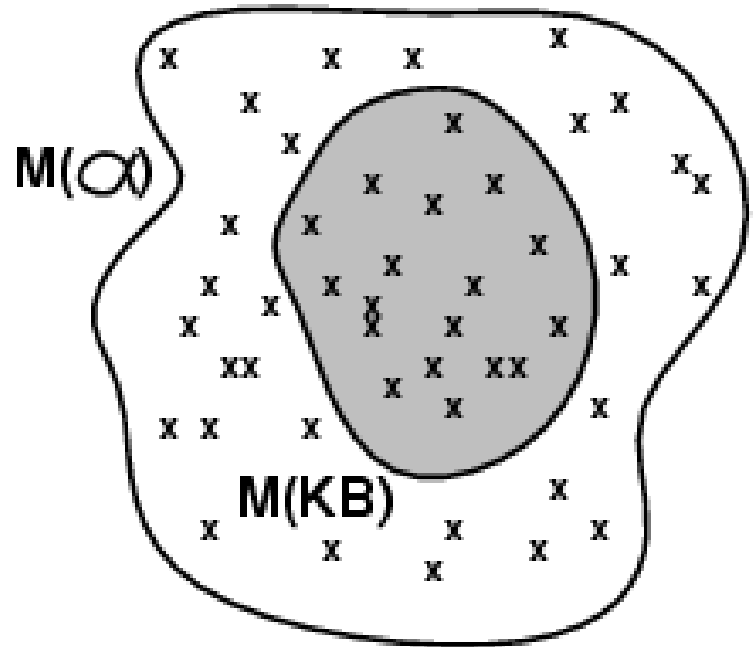
- Entailment means that one thing **follows from** another:

$$KB \models \alpha$$

- Knowledge base KB entails sentence α if and only if α is true in all worlds where KB is true
 - E.g., the KB containing “the Giants won” and “the Reds won” entails “Either the Giants won or the Reds won”
 - E.g., $x+y = 4$ entails $4 = x+y$
 - Entailment is a relationship between sentences (i.e., **syntax**) that is based on **semantics**

Models

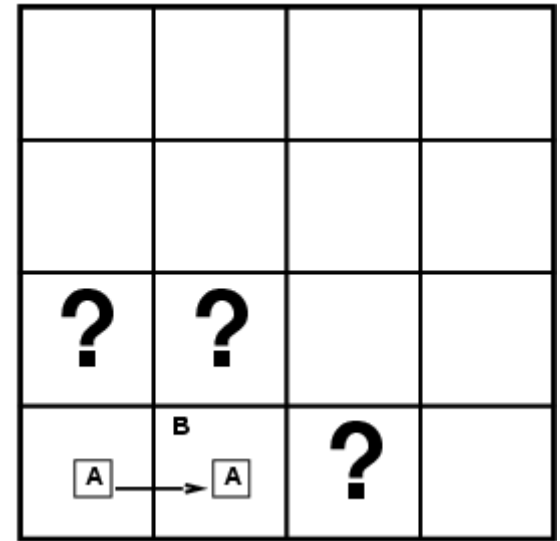
- Logicians typically think in terms of **models**, which are formally structured worlds with respect to which truth can be evaluated
 - A model is a “possible world”
- We say m is a **model** of a sentence α if α
- $M(\alpha)$ is the set of all models of α
- Then $KB \models \alpha$ iff $M(KB) \subseteq M(\alpha)$
 - E.g. $KB =$ Giants won and Reds won $\alpha =$ Giants won



Entailment in the wumpus world

Situation after detecting nothing
in [1,1], moving right, breeze
in [2,1]

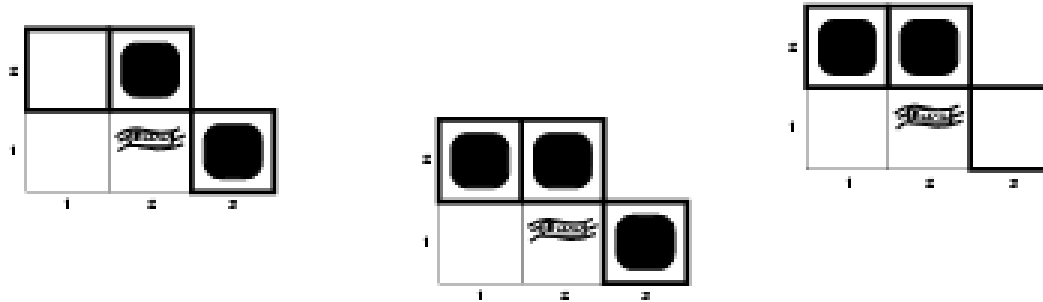
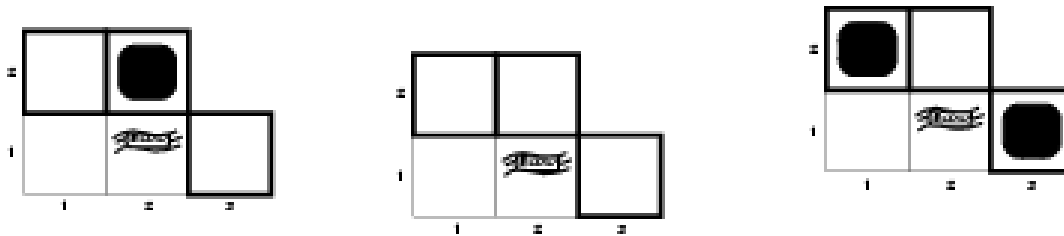
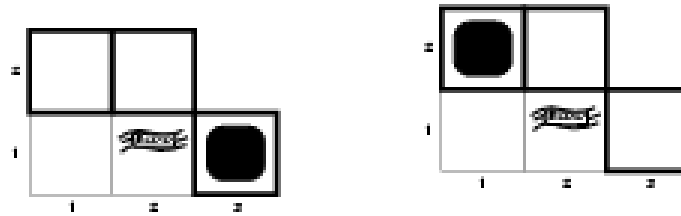
Consider possible models for
KB assuming only pits



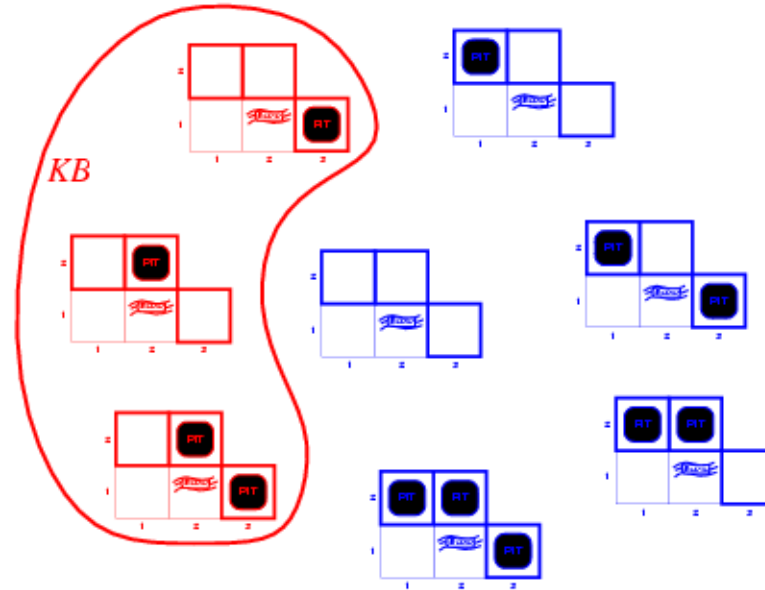
3 Boolean choices

$2^3 = 8$ possible models

Wumpus models

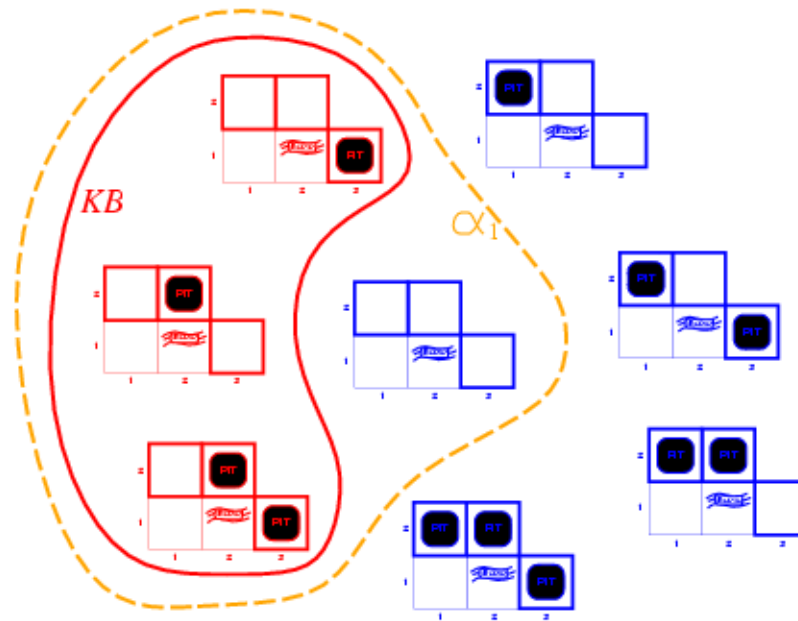


Wumpus models



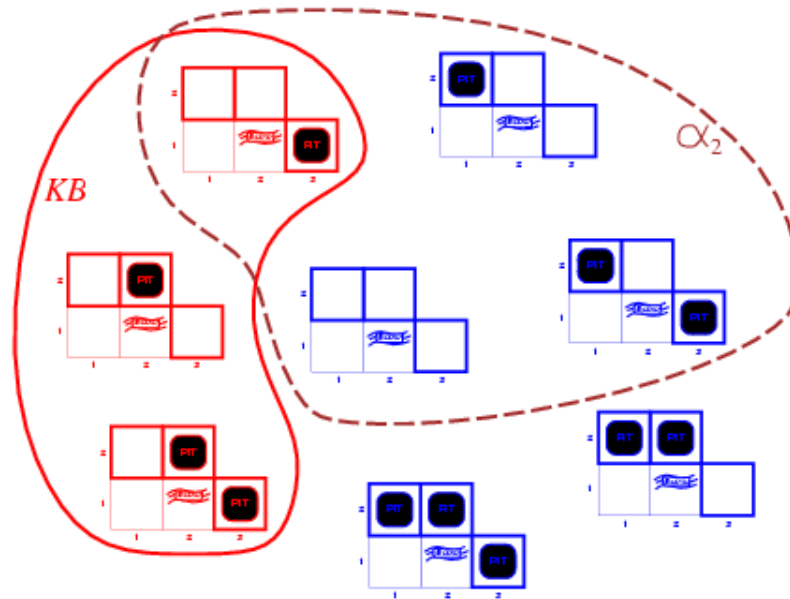
- $KB = \text{wumpus-world rules} + \text{observations}$

Wumpus models



- KB = wumpus-world rules + observations
- α_1 = "[1,2] is safe", $KB \models \alpha_1$, proved by model checking

Wumpus models



- KB = wumpus-world rules + observations
- α_2 = "[2,2] is safe", $KB \not\models \alpha_2$

Inference

- $KB \vdash_i \alpha$ = sentence α can be derived from KB by procedure i
- **Soundness**: i is sound if whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$
- **Completeness**: i is complete if whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$
- Preview: we will define a logic (first-order logic) which is expressive enough to say almost anything of interest, and for which there exists a sound and complete inference procedure.
- That is, the procedure will answer any question whose answer follows from what is known by the KB .

Propositional Logic

- Symbols represent *propositions* (facts).
- A proposition is either *TRUE* or *FALSE*.
- Boolean connectives can join propositions together into complex *sentences*.
- Sentences are statements that are either *TRUE* or *FALSE*.

Propositional Logic Syntax

- The constants *TRUE* and *FALSE*.
- Symbols such as *P* or *Q* that represent propositions.
- Logical connectives:
 - \wedge AND, conjunction
 - \vee OR, disjunction
 - \Rightarrow Implication , conditional (If then)
 - \Leftrightarrow Equivalence , biconditional
 - \neg Negation (unary)
 - () parentheses (grouping)

Truth Tables

P	Q	$\neg P$	$P \vee Q$	$P \wedge Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	TRUE
FALSE	TRUE	TRUE	TRUE	FALSE	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE
TRUE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE

Sentences

- *True, False* or any proposition symbol is a sentence.
- Any sentence surrounded by parentheses is a sentence.
- The disjunction, conjunction, implication or equivalence of 2 sentences is a sentence.
- The negation of a sentence is a sentence.

Examples

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

If P or Q is true, then R is true

$$P \Leftrightarrow (Q \wedge R)$$

If Q and R are both true, P must be true AND if Q or R is false then P must be false.

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

If P is false, then If Q is true R must be true.

Wumpus world sentences

Let $P_{i,j}$ be true if there is a pit in $[i, j]$.

Let $B_{i,j}$ be true if there is a breeze in $[i, j]$.

$$\neg P_{1,1}$$

$$\neg B_{1,1}$$

$$B_{2,1}$$

- "Pits cause breezes in adjacent squares"

$$B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})$$

$$B_{2,1} \Rightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$$

Sentence Validity

- A propositional sentence is valid (TRUE) if and only if it is true under all possible interpretations in all possible worlds (models, domains).
- For example:
If Today_Is_Monday Then We_Have_Class
The truth does not depend on whether today is Tuesday but on whether the relationship is true.

Satisfiability

A sentence is satisfiable if it is true in some model

e.g., $A \vee B, C$

A sentence is unsatisfiable if it is true in no model

e.g., $A \wedge \neg A$

Satisfiability is connected to inference via

$KB \models \alpha$ if and only if $(KB \wedge \neg \alpha)$ is unsatisfiable

i.e., prove α by *reductio ad absurdum*

Inference by enumeration

- Depth-first enumeration of all models is sound and complete

```
function TT-ENTAILS?(KB,  $\alpha$ ) returns true or false
```

```
  symbols  $\leftarrow$  a list of the proposition symbols in KB and  $\alpha$ 
```

```
  return TT-CHECK-ALL(KB,  $\alpha$ , symbols, [])
```

```
function TT-CHECK-ALL(KB,  $\alpha$ , symbols, model) returns true or false
```

```
  if EMPTY?(symbols) then
```

```
    if PL-TRUE?(KB, model) then return PL-TRUE?( $\alpha$ , model)
```

```
    else return true
```

```
  else do
```

```
    P  $\leftarrow$  FIRST(symbols); rest  $\leftarrow$  REST(symbols)
```

```
    return TT-CHECK-ALL(KB,  $\alpha$ , rest, EXTEND(P, true, model)) and  
          TT-CHECK-ALL(KB,  $\alpha$ , rest, EXTEND(P, false, model))
```

- For n symbols, time complexity is $O(2^n)$, space complexity is $O(n)$

Inference Rules

- There are many patterns that can be formally called *rules of inference* for propositional logic.
- These patterns describe how new knowledge can be derived from existing knowledge
 - knowledge is in the form of propositional logic sentences.
- Some patterns are common and have fancy names.

Inference Rule Notation

- When describing an inference rule, the *premise* specifies the pattern that must match our knowledge base and the *conclusion* is the new knowledge inferred.
- We will use the notation

premise \vdash *conclusion*

Inference Rules

- Modus Ponens: $x \Rightarrow y, x \vdash y$
- And-Elimination: $x_1 \wedge x_2 \wedge \dots \wedge x_n \vdash x_i$
- And-Introduction: $x_1, x_2, \dots, x_n \vdash x_1 \wedge x_2 \wedge \dots \wedge x_n$
- Or-Introduction: $x \vdash x \vee y \vee z \vee \dots$
- Double-Negation Elimination: $\neg\neg x \vdash x$
- Unit Resolution: $x \vee y, \neg x \vdash y$

Resolution Inference Rule

$$x \vee y, \neg y \vee z \vdash x \vee z$$

-or-

$$\neg x \Rightarrow y, y \Rightarrow z \vdash \neg x \Rightarrow z$$

Logic & Finding a Proof

- Given
 - a knowledge base represented as a set of propositional sentences.
 - a goal stated as a propositional sentence
 - a list of inference rules
- We can write a program to repeatedly apply inference rules to the knowledge base, in the hope of deriving the goal.

Example

It will snow OR there will be a test.

Dave is Darth Vader OR it will not snow.

Dave is not Darth Vader.

Will there be a test?

Solution

Snow = a Test = b Dave is Vader = c

Knowledge Base (these are all true):

$$a \vee b, \quad c \vee \neg a, \quad \neg c$$

By Resolution we know $b \vee c$ is true.

By Unit Resolution we know b is true.

There will be a test!

Developing a Proof Procedure

- Deriving (or refuting) a goal from a collection of logic facts corresponds to a very large search tree.
- A large number of *rules of inference* could be utilized.
- The selection of which rules to apply and when would itself be non-trivial.

Resolution & CNF

- *Resolution* is a single rule of inference that can operate efficiently on a special form of sentences.
- The special form is called *clause form* or *conjunctive normal form* (CNF), and has these properties:
 - Every sentence is a disjunction (or) of literals
 - All sentences are implicitly conjuncted (anded).

Propositional Logic and CNF

- Any propositional logic sentence can be converted to CNF. We need to remove all connectives other than OR (without modifying the meaning of a sentence)

Converting to CNF

- Eliminate implications and equivalence.
- Reduce scope of all negations to single term.
- Use associative and distributive laws to convert to a conjunct of disjuncts.
- Create a separate sentence for each conjunct.

Eliminate Implications and Equivalence

$x \Rightarrow y$ becomes $\neg x \vee y$

$x \Leftrightarrow y$ becomes $(\neg x \vee y) \wedge (\neg y \vee x)$

Reduce Scope of Negations

$\neg(\neg x)$ becomes x

$\neg(x \vee y)$ becomes $(\neg y \wedge \neg x)$

$\neg(x \wedge y)$ becomes $(\neg y \vee \neg x)$

**deMorgans Laws**

Convert to conjunct of disjuncts

Associative property:

$$(A \vee B) \vee C = A \vee (B \vee C)$$

Distributive property:

$$(A \wedge B) \vee C = (A \vee C) \wedge (B \vee C)$$

Using Resolution to Prove

- Convert all propositional sentences that are in the knowledge base to CNF.
- Add the contradiction of the goal to the knowledge base (in CNF).
- Use resolution as a rule of inference to prove that the combined facts can not all be true.

Proof by contradiction

- We assume that all original facts are TRUE.
- We add a new fact (the contradiction of sentence we are trying to prove is TRUE).
- If we can infer that FALSE is TRUE we know the knowledgebase is corrupt.
- The only thing that might not be TRUE is the negation of the goal that we added, so it must be FALSE. Therefore the goal is true.

Propositional Example: The Mechanics of Proof

Knowledge Base:

P

$(P \wedge Q) \Rightarrow R$

$(S \vee T) \Rightarrow Q$

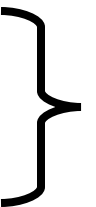
T

Goal:

R



*These represent the
facts we know to be
true.*



*This is what we want to
prove is true.*

Conversion to CNF

Sentence

CNF

P

P

$(P \wedge Q) \Rightarrow R$

$\neg P \vee \neg Q \vee R$

$(S \vee T) \Rightarrow Q$

$\neg S \vee Q$

$\neg T \vee Q$

T

T

Add Contradiction of Goal

- The goal is **R**, so we add **$\neg R$** to the list of facts, the new set is:

1. **P**
2. **$\neg P \vee \neg Q \vee R$**
3. **$\neg S \vee Q$**
4. **$\neg T \vee Q$**
5. **T**
6. **$\neg R$**

Resolution Rule of Inference

Recall the general form of resolution:

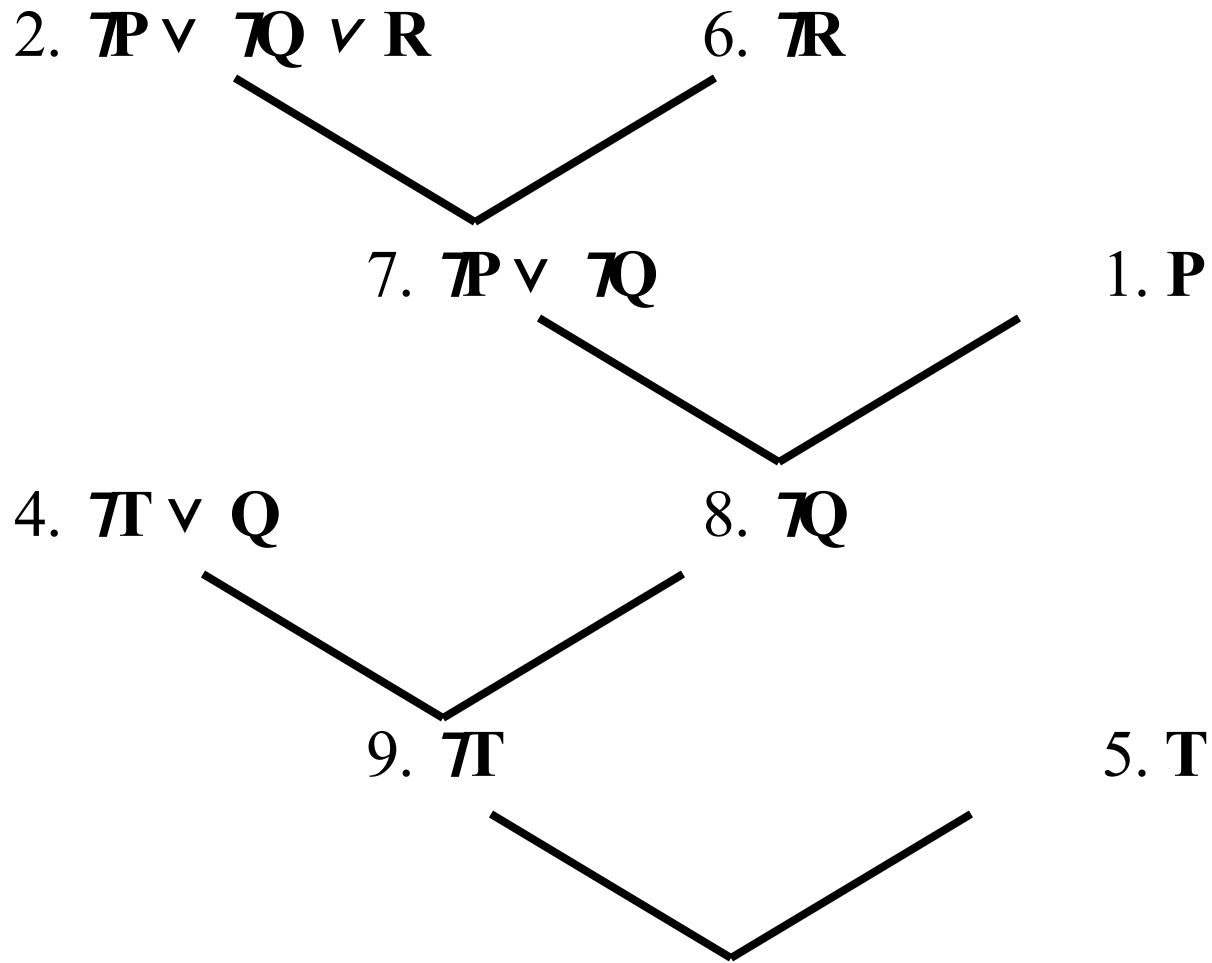
$$x_1 \vee x_2 \vee \dots \vee x_n \vee z, \quad y_1 \vee y_2 \vee \dots \vee y_m \vee \neg z \vdash \\ x_1 \vee x_2 \vee \dots \vee x_n \vee y_1 \vee y_2 \vee \dots \vee y_m$$

Applying Resolution

Fact 2 can be *resolved* with fact 6, yielding a new fact:

$$\begin{array}{ccc} \neg P \vee \neg Q \vee R & & \neg R \\ & \searrow & \swarrow \\ & \neg P \vee \neg Q & \end{array}$$

A new fact, call it fact 7.



*There is no way all
the clauses can be true!*

← *Null Clause*

A more intuitive look at the same example.

P: Joe is smart

Q: Joe likes hockey

R: Joe goes to RPI

S: Joe is Canadian

T: Joe skates.

- Original Sentences:
 - Joe is smart
 - If Joe is smart and Joe likes hockey, Joe goes to RPI
 - If Joe is Canadian or Joe skates, Joe likes hockey.
 - Joe skates.
- After conversion to CNF:
 - Fact 2: Joe is not smart, or Joe does not like hockey, or Joe goes to RPI.
 - Fact 3: Joe is not Canadian or Joe likes hockey.
 - Fact 4: Joe does not skate, or Joe likes hockey.

Joe is not smart, or Joe does not like hockey, or Joe goes to RPI

Joe does not go to RPI

*Joe is not smart or
Joe does not like hockey*

Joe is smart

*Joe does not skate, or
Joe likes hockey*

Joe does not like hockey

Joe does not skate

Joe skates

Null Clause

Propositional Logic Limits

- The expressive power of propositional logic is limited. The assumption is that everything can be expressed by simple facts.
- It is much easier to model real world objects using *properties* and *relations*.
- Predicate Logic provides these capabilities more formally and is used in many AI domains to represent knowledge.

Propositional Logic Problem

If the unicorn is mythical, then it is immortal, but if it is not mythical then it is a mortal mammal. If the unicorn is either immortal or a mammal, then it is horned. The unicorn is magical if it is horned.

Q: Is the unicorn mythical? Magical? Horned?