Logical Agents

- When we use search to solve a problem we must
  - Capture the knowledge needed to formalize the problem
  - Apply a search technique to solve problem
  - Execute the problem solution

Knowledge Representation

- Knowledge-based agents
- Wumpus world
- Logic
  - Sentence, syntax, semantics, validity, satisfiability
  - Proof by Enumeration and Inference
- Proposition Logic
  - Inference Rules, Normal Forms
- Automating theorem proving
  - Forward chaining
  - Backward chaining
  - Resolution

A Brief History of Reasoning

450 B.C	Stoics	Propositional logic, inference (maybe)
322 B.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	Godel	$\exists$ Complete algorithm for FOL
1930	Herbrand	Complete algorithm for FOL (reduce to propositional)
1931	Godel	$\neg\exists$ complete algorithm for arithmetic
1960	Davis/Putnam	“practical” algorithm for propositional logic
1965	Robinson	“practical” algorithm for FOL - resolution

Role of Knowledge Representation

- We previously talked about applications of search but not about methods of formalizing the problem.
- Now we look at extended capabilities to general logical reasoning.
- The first step is the role of “knowledge representation” in AI.
- Formally:
  - The intended role of knowledge representation in artificial intelligence is to reduce problems of intelligent action to search problems.
  - “A good description, developed within the conventions of a good KR, is an open door to problem solving; a bad description, using a bad representation, is a brick wall preventing problem solving.”

Knowledge Bases

- *Knowledge base* (*KB*) is a set of sentences in a formal language

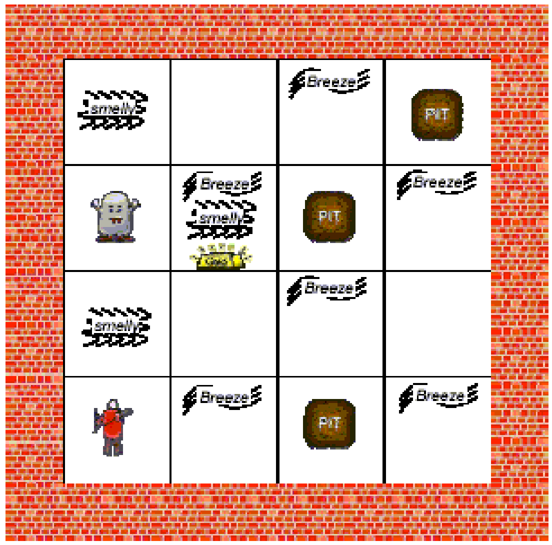
- Declarative approach to building an agent
  TELL it what it needs to know
- Then it can ASK itself what to do
  - Answers follow from KB – *entailment*
- Agents can be viewed at the knowledge level
  - What they know, regardless of implementation
- Or at the implementation level
  - Data structures in KB and algorithms that manipulate them

A Knowledge-Based Agent

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

- We will
  - Describe properties of languages to use for logical reasoning
  - Describe techniques for deducing new information from current information
  - Apply search to deduce (or learn) specifically needed information

The Wumpus World Environment



Percepts

- Percept = [Stench, Breeze, Glitter, Bump, Scream]
  - Stench in Wumpus square and adjacent (L, R, U, D) squares
  - Breeze in squares adjacent to pit
  - Glitter in gold square
  - Bump when walk into wall or obstacle
  - Everyone hears scream when Wumpus is defeated
  - Agent cannot perceive its own location

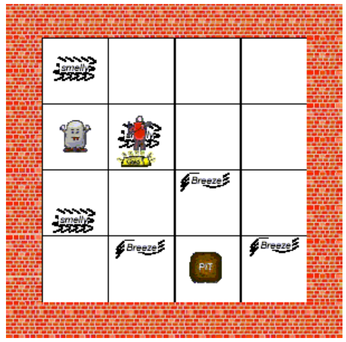
Goals

- Actions = [goforward, turnleft, turnright, grab, shoot, climb]
- Agent is defeated upon entering room with pit or live Wumpus
- Agent's goal is to find gold, return to [1,1], and climb out of cave
- Score (-1) for each action attempted, (-1000) for being defeated, (+1000) for leaving cave with gold

Wumpus World Properties

- Is the world deterministic?
  - Yes
- Is the world fully accessible?
  - No, only local perception
- Is the world static?
  - Yes (for now), Wumpus and pits do not move
- Is the world discrete?
  - Yes
- Single-agent?
  - Yes, Wumpus is a domain feature

Sample Run



- Now we look at
- How to represent facts / beliefs
  "There is a pit in (2,2) or (3,1)"
- How to make inferences
  "No Breeze in (1,2), so pit in (3,1)"

Formal Representation Through Logic

- Logic: a formal language for representing information such that conclusions can be drawn
  - Sentence: individual piece of knowledge
    English sentence forms one piece of knowledge in English language
    A statement in Java forms one piece of knowledge in Java programming language
  - Syntax: form used to represent sentences
    Syntax of Java indicates legal combinations of symbols (a = 2+3;)
    Syntax alone does not indicate meaning
  - Semantics: mapping from sentences to facts in the world
    They define the truth of a sentence in a world. Add the values of 2 and 3, store the result in the memory location indicated by variable a
  - Proof System: a way of manipulating syntactic expressions (enumeration or inference) to get other syntactic expressions (which will tell us something new)

A few Formal Logics

Language	Ontological Commitment	Epistemological Commitment
Propositional Logic	facts	true/false/unknown
Predicate Logic	facts, objects, relations	true/false/unknown
Temporal Logic	facts, objects, relations, time	true/false/unknown
Probability Theory	facts	degree of belief in [0.0 .. 1.0]
Fuzzy Logic	degree of truth	degree of belief in [0.0 .. 1.0]

- Which Logic to use?
  - Ontological commitment: What exists? Facts? Objects? Time? Beliefs?
  - Epistemological commitment: What are the states of knowledge?

Definitions

- The meaning of a sentence is a mapping onto the "real world".
  - This mapping is an interpretation.
- A sentence is valid (necessarily true, tautology) iff true under all possible interpretations.

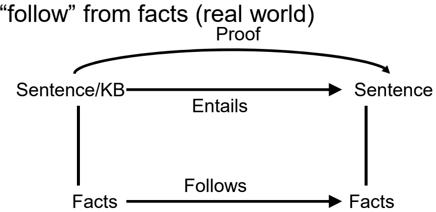
  $A \vee \neg A$
  - A could be:
    - Stench at [1,1]
    - 2+3=5
  - These statements are not valid.

    $A \wedge \neg A$

    $A \vee B$
- The last statement is satisfiable, meaning there exists at least one interpretation that makes the statement true. The previous statement is unsatisfiable.

Entailment

- The proof capability relies on relationships between items in the language:
  - Sentences "entail" sentences (representation level)
  - Facts "follow" from facts (real world)



- Entail/Follow means the new item is true if the old items are true
- A collection of sentences, or knowledge base (*KB*), entail a sentence α
  - $KB \models \alpha$
- *KB* entails the sentence α if and only if α is true in all worlds where the *KB* is true

Entailment Examples

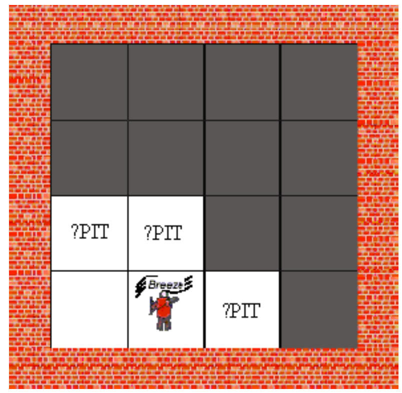
- KB
  - The Giants won
  - The Reds won
- Entails
  - The Giants won and the Reds won

- KB
  - To get a perfect score your program must be turned in today
  - I always get perfect scores
- Entails
  - I turned in my program today

- Entailment is a relationship between sentences (*syntax*) that is based on *semantics*.

Entailment in the Wumpus World

- Situation after detecting nothing in (1,1) and moving right with a breeze in (2,1)
- Consider ALL possible models for ?s assuming only pits
  - 3 possibilites ⇒ 8 models

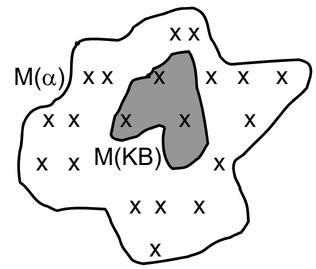


Proof Methods

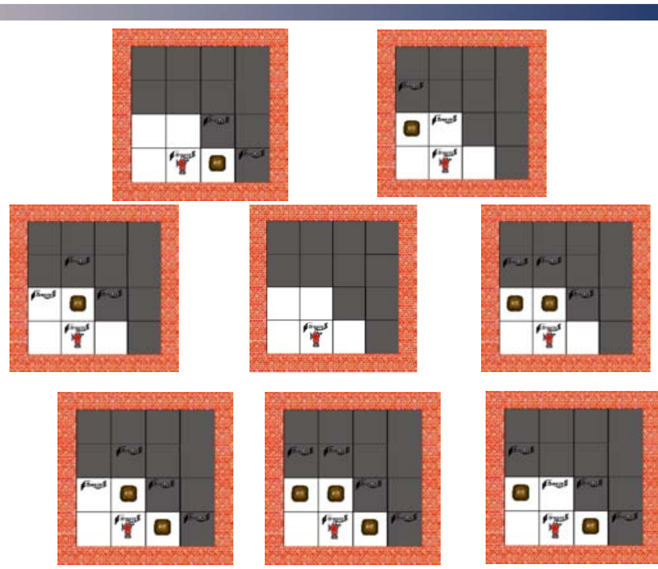
- Proof methods divide into (roughly) two kinds:
  - Model checking
    - Truth table enumeration (sound and complete for propositional logic sentence) (exponential in *n*)
    - Improved backtracking
    - Heuristic search in model space
      - Show that all interpretations in which the left hand side of the rule is true, the right hand side is also true (sound but incomplete)
  - Application of inference rules
    - Sound generation of new sentences from old
    - Proof = a sequence of inference rule applications
      - Can use inference rules as operators in a standard search algorithm.
      - Typically requires translation of sentences into a normal form

Proof by Enumeration (Models)

- Logicians think in terms of models, which are formally structured worlds with respect to which truth can be evaluated
- We say *m* is a model of a sentence α if α is true in *m*
- $M(\alpha)$ is the set of all models which hold for α
- The set M(KB) is an enumeration of the entailment of a KB
- Then KB $\models \alpha$ if and only if $M(KB) \subseteq M(\alpha)$
  KB = Giants won and Reds won
  α = Giants won

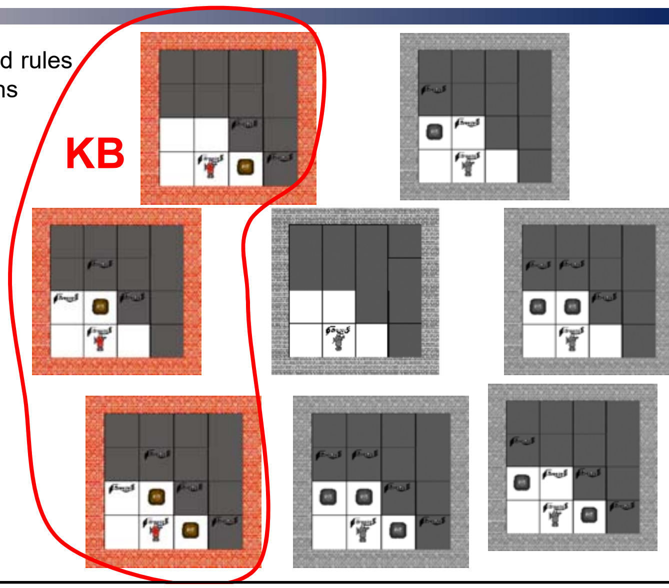


Wumpus Models



Wumpus Models

KB = Wumpus-world rules
+ observations

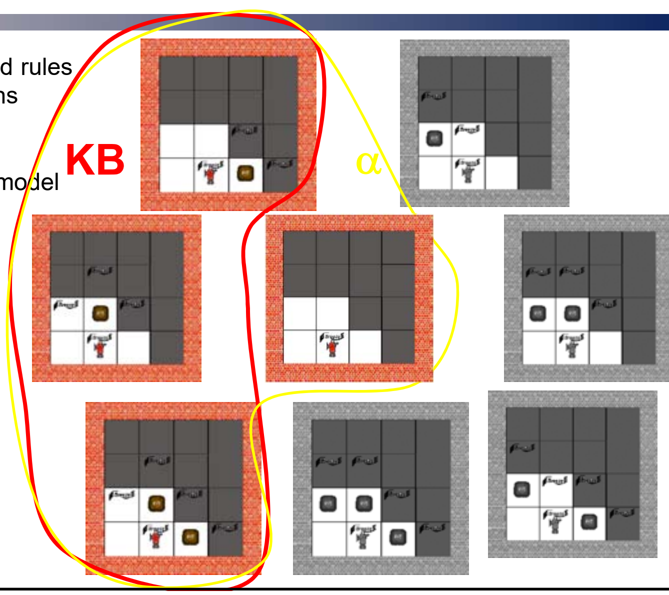


Wumpus Models

KB = Wumpus-world rules
+ observations

α = "(1,2) is safe"

KB $\models \alpha$, proved by model checking



Wumpus Models

KB = Wumpus-world rules
+ observations

α_2 = "(2,2) is safe"

KB $\not\models \alpha_2$

Proof by Inference

- *KB* $\vdash_i \alpha$: sentence α can be derived from *KB* by procedure *i*
- Inference can be used two different ways:
  1. Generate new sentences that are entailed by KB
  2. Determine whether or not sentence is entailed by KB
- A sound inference procedure generates only entailed sentences.
  - Whenever *KB* $\vdash_i \alpha$, then *KB* $\models \alpha$ is true
- A complete inference procedure can generate all sentences in the knowledge base.
  - Whenever *KB* $\models \alpha$, then *KB* $\vdash_i \alpha$
- Modus ponens yes   Abduction no

$$\frac{A,\ A \Rightarrow B}{B} \qquad \frac{B,\ A \Rightarrow B}{A}$$

Propositional Logic

- Propositional logic is the simplest logic – illustrates basic ideas
- Proposition symbols *P*, *Q*, etc., are unit atoms and sentences
- If S_1 and S_2 are sentences then so are $\neg S_1$, $S_1 \wedge S_2$, $S_1 \vee S_2$, $S_1 \Rightarrow S_2$, $S_1 \Leftrightarrow S_2$
  - Negation, conjunction, disjunction, implication, and biconditional
- An interpretation *i* consists of an assignment of truth values to all proposition symbols $i(s)$
  An interpretation is a “possible world”
  Given 3 proposition symbols P, Q, and R, there are 8 interpretations
  Given *n* proposition symbols, there are 2^n interpretations
- Models are worlds in which a particular sentence is true under at least one interpretation
- The true/false value of propositions and combinations of propositions can be calculated using a truth table

Propositional Logic

- For propositional logic, a row in the truth table is one interpretation
- A logic is “monotonic” as long as entailed sentences are preserved as more knowledge is added

P	Q	¬P	P∧Q	P∨Q	P⇒Q	P⇔Q
T	T	F	T	T	T	T
T	F	F	F	T	F	F
F	T	T	F	T	T	F
F	F	T	F	F	T	T

Rules of Inference for Propositional Logic

Logical Identities and Equivalences		
Idempotency	$A \vee A \equiv A$	$A \wedge A \equiv A$
Commutativity	$A \vee B \equiv B \vee A$	$A \wedge B \equiv B \wedge A$
	$A \Leftrightarrow B \equiv B \Leftrightarrow A$	
Associativity	$(A \vee B) \vee C \equiv A \vee (B \vee C)$	$(A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$
Absorption	$A \vee (A \wedge B) \equiv A$	$A \wedge (A \vee B) \equiv A$
Distributivity	$A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$	$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$
Tautology	$A \vee B \equiv A$ if A is a tautology	$A \wedge B \equiv B$ if A is a tautology
Unsatisfiability	$A \vee B \equiv B$ if A is unsatisfiable	$A \wedge B \equiv A$ if A is unsatisfiable
De Morgan's laws	$\neg(A \wedge B) \equiv \neg A \vee \neg B$	$\neg(A \vee B) \equiv \neg A \wedge \neg B$
Contradiction	$\neg A \wedge A \equiv F$	$\neg A \vee A \equiv T$
Identity	$A \vee F \equiv A$	$A \wedge T \equiv A$
Domination	$A \vee T \equiv T$	$A \wedge F \equiv F$
Double-Negation Elimination	$\neg\neg A \equiv A$	

Rules of Inference for Propositional Logic

Logical Identities and Equivalences		
Implication	$A \Rightarrow B \equiv \neg A \vee B$	
	$A \Leftrightarrow B \equiv (A \wedge B) \vee (\neg A \wedge \neg B)$	
	$A \Leftrightarrow B \equiv (\neg A \vee B) \wedge (A \vee \neg B)$	
Modus ponens	$A, A \Rightarrow B \models B$	
Modus tollens	$A \Rightarrow B, \neg B \models \neg A$	
And introduction	$A, B \models A \wedge B$	
Or introduction	$A \models A \vee B \vee C \vee D \vee \ldots$	
And elimination	$A \wedge B \wedge C \wedge \ldots \wedge Z \models A$	
Unit Resolution	$A \vee B, \neg B \models A$	$(A \vee B) \wedge \neg B \Rightarrow A$
Resolution	$A \vee B, \neg B \vee C \models A \vee C$	$\neg A \Rightarrow B, B \Rightarrow C \models \neg A \Rightarrow C$

Normal Forms

- Other approaches to inference use syntactic operations on sentences, often expressed in standardized forms
- Conjunctive Normal Form (CNF) *conjunction* of disjunction of literals
  $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$
- Disjunctive Normal Form (DNF) *disjunction* of conjunction of literals
  $(A \wedge B) \vee (A \wedge \neg C) \vee (A \wedge \neg D) \vee (\neg B \wedge \neg C) \vee (\neg B \wedge \neg D)$
- Horn Form (restricted) *conjunction* of *Horn clauses* (clauses with only 1 positive literal)
  $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$
- Often written as a set of implications or Implicative Normal Form (INF):
  $B \Rightarrow A$
  $(C \wedge D) \Rightarrow B$

Propositional Conversion to CNF

$B \Leftrightarrow A \vee C$

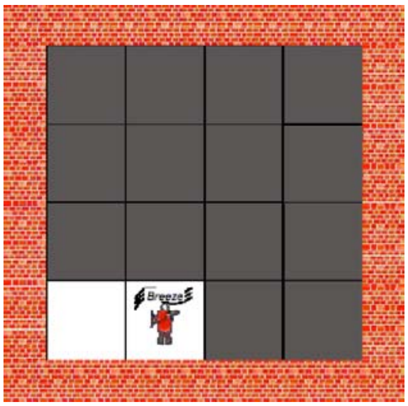
1. Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$
   $(B \Rightarrow A \vee C) \wedge (A \vee C \Rightarrow B)$
2. Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$
   $(\neg B \vee A \vee C) \wedge (\neg(A \vee C) \vee B)$
3. Move $\neg$ inwards using de Morgan's rules and double-negation
   $(\neg B \vee A \vee C) \wedge ((\neg A \wedge \neg C) \vee B)$
4. Apply distributivity law ($\vee$ over $\wedge$) and flatten
   $(\neg B \vee A \vee C) \wedge (\neg A \vee B) \wedge (\neg C \vee B)$

Proof Methods

- Proof methods divide into (roughly) two kinds:
  - Model checking
    - Truth table enumeration (sound and complete for propositional logic sentence) (exponential in *n*)
    - Improved backtracking
    - Heuristic search in model space
      - Show that all interpretations in which the left hand side of the rule is true, the right hand side is also true (sound but incomplete)
  - Application of inference rules
    - Sound generation of new sentences from old
    - Proof = a sequence of inference rule applications
      - Can use inference rules as operators in a standard search algorithm.
      - Typically requires translation of sentences into a normal form

Wumpus World Sentences

- Imagine we are at a stage in the game where we have had the following experiences. What is in our knowledge base?
- What can we deduce about the world?



- Given the initial world:
  R1: $\neg P_{1,1}$
- Also know:
  R2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
  R3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
- And we found out:
  R4: $\neg B_{1,1}$
  R5: $B_{2,1}$

Proof by Truth Table/Enumeration

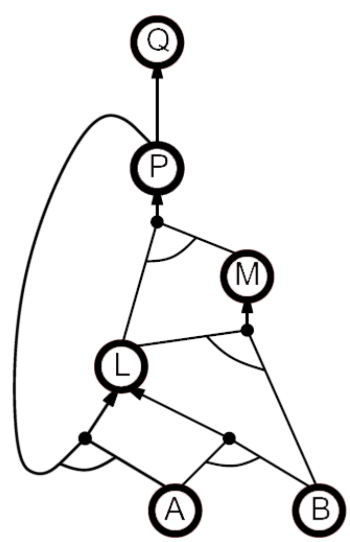
$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	R1	R2	R3	R4	R5	*KB*
F	F	F	F	F	F	F	T	T	T	T	F	F
F	F	F	F	F	F	T	T	T	F	T	F	F
:	:	:	:	:	:	:	:	:	:	:	:	:
F	T	F	F	F	F	F	T	T	F	T	T	F
F	T	F	F	F	F	T	T	T	T	T	T	T
F	T	F	F	F	T	F	T	T	T	T	T	T
F	T	F	F	F	T	T	T	T	T	T	T	T
F	T	F	F	T	F	F	T	F	F	T	T	F
:	:	:	:	:	:	:	:	:	:	:	:	:
T	T	T	T	T	T	T	F	T	T	F	T	F

KB ⇔R1 ∧ R2 ∧ R3 ∧ R4 ∧ R5

- Depth-first enumeration of all models is sound and complete
- $O(2^n)$ for n symbols; problem is co-NP-complete

Representing Horn Clauses as And-Or Graphs

- $P \Rightarrow Q$
- $L \wedge M \Rightarrow P$
- $B \wedge L \Rightarrow M$
- $A \wedge P \Rightarrow L$
- $A \wedge B \Rightarrow L$
- A
- B



Forward and Backward Chaining

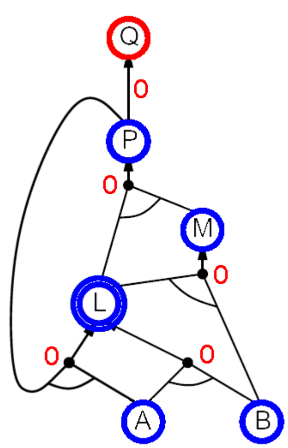
- Modus Ponens: complete for Horn KBs

$$\frac{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n, \quad \alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n \Rightarrow \beta}{\beta}$$

- Can be used in a forward or backward chaining manner
- Are very natural and run in linear time

Forward Chaining (FC)

- Fire any rule whose premises are satisfied in the KB, add its conclusion to the KB until query is found



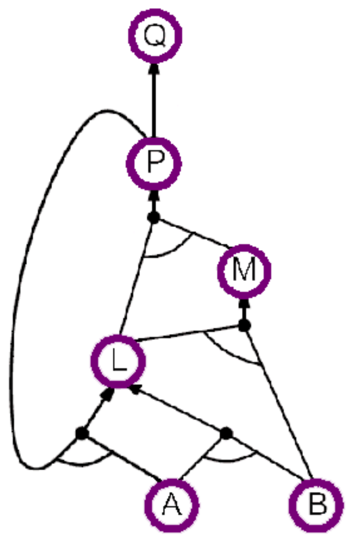
Proof of Completeness

- FC derives every atomic sentence that is entailed by *KB*
  1. FC reaches a fixed point where no new atomic sentences are derived
  2. Consider this final state as a model *m*, assigning true/false to symbols
  3. Every clause in the original *KB* is true in *m*
     Proof: Suppose a clause $a_1 \wedge \ldots \wedge a_k \Rightarrow b$ is false in *m*. Then $a_1 \wedge .. \wedge a_k$ is true in *m* and *b* is false in *m.* Therefore the algorithm has not reached a fixed point
  4. Hence *m* is a model of *KB*
  5. If $KB \models q$, *q* is true in every model of *KB*, including *m*

Backward Chaining (BC)

- Work backwards from the query q:
  to prove q by BC
  check if q is known already, or
  prove by BC all premises of some rule concluding q
- Avoid loops: check if new subgoal is already on the goal stack
- Avoid repeated work: check if new subgoal
  - Has already been proved true, or
  - Has already failed

Backward Chaining Example



Forward Chaining vs. Backward Chaining

- FC is data driven, automatic, unconscious processing
  - Object recognition, routing decisions
- May do lots of work that is irrelevant to the goal
- BC is goal-driven, appropriate for problem-solving
  - Where are my keys? How do I get into the PhD program?
- Complexity of BC can be much less than linear in size of KB

Resolution

- Resolution inference rule (for CNF) complete for propositional logic

$$\frac{\alpha_1 \vee \ldots \vee \alpha_n \qquad \beta_1 \vee \ldots \vee \beta_m}{\alpha_1 \vee .. \vee \alpha_{i-1} \vee \alpha_{i+1} \vee .. \vee \alpha_n \vee \beta_1 \vee .. \vee \beta_{j-1} \vee \beta_{j+1} \vee .. \vee \beta_m}$$

  where α_i and β_j are complementary literals,
  $\neg\alpha_i \equiv \beta_j$

- $A \vee B \quad \neg B$
  A
  ____
- (We will talk about this in detail with FOPC)

Propositional Logic Summary

- Logical agents apply inference to a knowledge base to derive new information and make decisions
- Basic concepts of logic:
  - Syntax: formal structure of sentences
  - Semantics: truth of sentences with respect to models
  - Entailment: necessary truth of one sentence given another
  - Inference: deriving sentences from other sentences
  - Soundness: derivations produce only entailed sentences
  - Completeness: derivations can produce all entailed sentences
- Wumpus world requires the ability to represent partial and negated information, reason by cases, etc.
- Forward, backward chaining are linear-time, complete for Horn clauses
- Resolution is complete for propositional logic

Review

- Knowledge Representation
- Logic
- Propositional Logic
- Forward and Backward Chaining Proofs for Horn Clauses

- Next class
  - First-Order Predicate Calculus
  - Resolution by Refutation
  - Expert Systems