Propositional Logic

Chapter 7

Outline

- Review
  - Knowledge-based agents
  - Logic in general
  - Propositional logic in particular – syntax and semantics
- Wumpus world
- Inference rules and theorem proving
  - Resolution
  - forward chaining
  - backward chaining

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; $x2+y > \{\}$ 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 Steelers won" and "the Bengals won" entails "Either the Steelers won or the Bengals won"

  – E.g., $x+y = 4$ entails $4 = x+y$

  – Entailment is a relationship between sentences (i.e., syntax) that is based on semantics

A	B	C	A ∧ B	A ∧ C	B ∧ C
F	F	F	F	F	F
F	F	T	F	F	F
F	T	F	F	F	F
F	T	T	F	F	T
T	F	F	F	F	F
T	F	T	F	T	F
T	T	F	T	F	F
T	T	T	T	T	T

A^C, C does not entail B∧C

A,B, Entails A∧B

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: Syntax

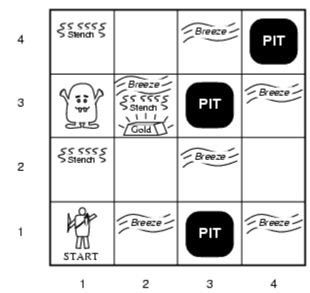
- Propositional logic is the simplest logic – illustrates basic ideas
- The proposition symbols P_1, P_2 etc are sentences
  - If S is a sentence, $\neg S$ is a sentence (negation)
  - If S_1 and S_2 are sentences, $S_1 \wedge S_2$ is a sentence (conjunction)
  - If S_1 and S_2 are sentences, $S_1 \vee S_2$ is a sentence (disjunction)
  - If S_1 and S_2 are sentences, $S_1 \Rightarrow S_2$ is a sentence (implication)
  - If S_1 and S_2 are sentences, $S_1 \Leftrightarrow S_2$ is a sentence (biconditional)

Propositional Logic: Semantics (truth tables for connectives)

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	*false*	*true*	*false*	*false*	*true*	*true*
false	*true*	*true*	*false*	*true*	*true*	*false*
true	*false*	*false*	*false*	*true*	*false*	*false*
true	*true*	*false*	*true*	*true*	*true*	*true*

Wumpus World PEAS description

- Performance measure
  - gold +1000, death -1000
  - -1 per step, -10 for using the arrow
- Environment
  - Squares adjacent to wumpus are smelly
  - Squares adjacent to pit are breezy
  - Glitter iff gold is in the same square
  - Shooting kills wumpus if you are facing it
  - Shooting uses up the only arrow
  - Grabbing picks up gold if in same square
  - Releasing drops the gold in same square



- Sensors: Stench, Breeze, Glitter, Bump, Scream
- Actuators: Left turn, Right turn, Forward, Grab, Release, Shoot

Wumpus world characterization

- Fully Observable
- Deterministic
- Episodic
- Static
- Discrete
- Single-agent?

Wumpus world characterization

- Fully Observable No – only local perception
- Deterministic Yes – outcomes exactly specified
- Episodic No – sequential at the level of actions
- Static Yes – Wumpus and Pits do not move
- Discrete Yes
- Single-agent? Yes – Wumpus is essentially a natural feature

Wumpus World continued

- Main difficulty: Agent doesn't know the configuration
- Reason about configuration
- Knowledge evolves as new percepts arrive and actions are taken.

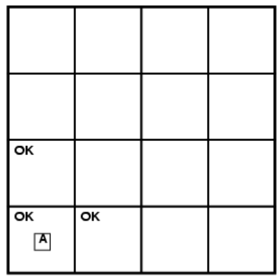
Wumpus Example

		stench	[Wumpus]	stench
		Glitter [gold]	stench, breeze	
0	[start]	breeze	[Pit]	breeze
	0			

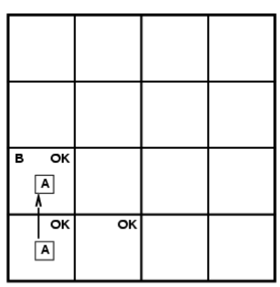
Examples of reasoning

- If the player is in square (1, 0) and the percept is breeze, then there must be a pit in (0,0) or a pit in (2,0) or a pit in (1,1).
- If the player is now in (0,0) [and still alive], there is not a pit in (0,0).
- If there is no breeze percept in (0,0), there is no pit in (0,1)
- If there is also no breeze in (0,1) then there is no pit in (1,1).
- Therefore, there must be a pit in (2,0)

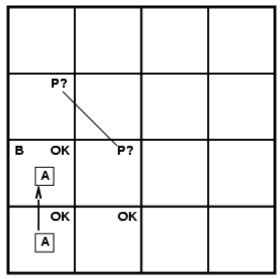
Exploring a wumpus world



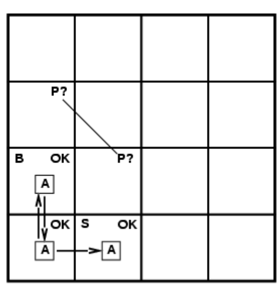
Exploring a wumpus world



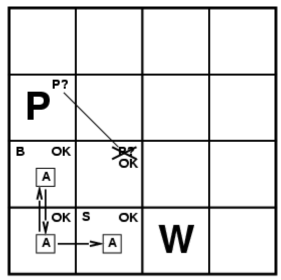
Exploring a wumpus world



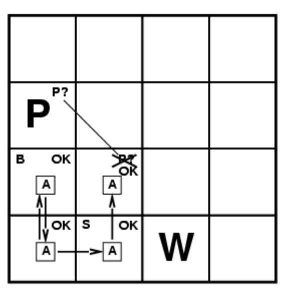
Exploring a wumpus world



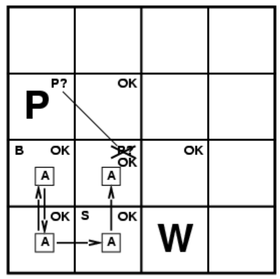
Exploring a wumpus world



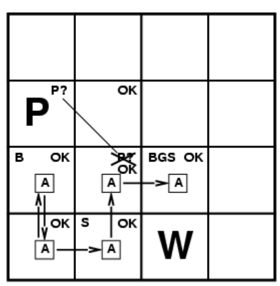
Exploring a wumpus world



Exploring a wumpus world



Exploring a wumpus world

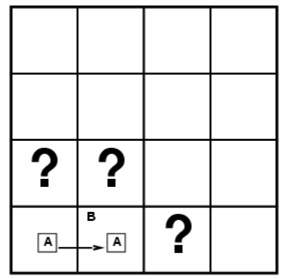


Entailment in the wumpus world

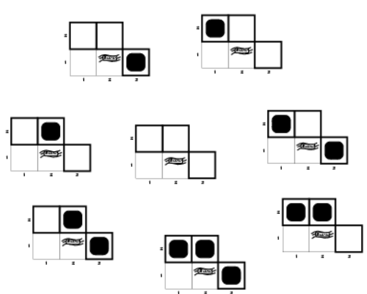
Situation after detecting nothing in [1,1], moving right, breeze in [2,1]

Consider possible models for *KB* assuming only pits

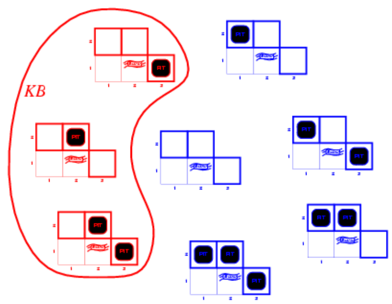
3 Boolean choices ⇒ 8 possible models



Wumpus models

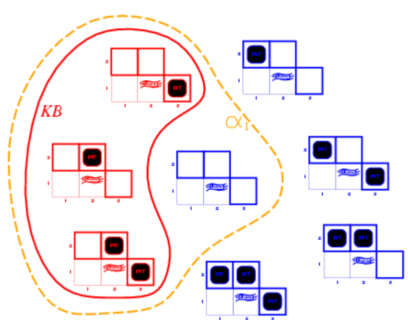


Wumpus models



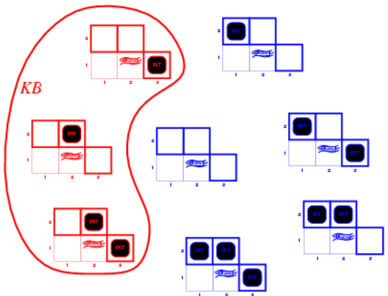
- *KB* = wumpus-world rules + observations

Wumpus models



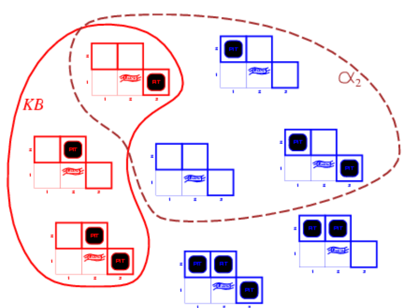
- *KB* = wumpus-world rules + observations
- α_1 = "[1,2] is safe", *KB* $\models \alpha_1$, proved by model checking

Wumpus models



- *KB* = wumpus-world rules + observations

Wumpus models



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

Logical Representation of Wumpus

Is there a pit in [i, j]?
Is there a breeze in [i, j]?

Pits cause breezes in adjacent squares.

Some 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"

$B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})$
$B_{2,1} \Leftrightarrow (P_{1,1} \lor P_{2,2} \lor P_{3,1})$
…

Inference-based agents in the wumpus world

A wumpus-world agent using propositional logic:

$\neg P_{1,1}$
$\neg W_{1,1}$
$B_{x,y} \Leftrightarrow (P_{x,y+1} \lor P_{x,y-1} \lor P_{x+1,y} \lor P_{x-1,y})$
$S_{x,y} \Leftrightarrow (W_{x,y+1} \lor W_{x,y-1} \lor W_{x+1,y} \lor W_{x-1,y})$
$W_{1,1} \lor W_{1,2} \lor \ldots \lor W_{4,4}$
$\neg W_{1,1} \lor \neg W_{1,2}$
$\neg W_{1,1} \lor \neg W_{1,3}$
…

$\Rightarrow$ 64 distinct proposition symbols, 155 sentences

Truth tables for inference

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	KB	α_1
false	*false*	*false*	*false*	*false*	*false*	*false*	*false*	*true*
false	*false*	*false*	*false*	*false*	*false*	*true*	*false*	*true*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*false*	*true*
false	*true*	*false*	*false*	*false*	*false*	*true*	<u>*true*</u>	<u>*true*</u>
false	*true*	*false*	*false*	*false*	*true*	*false*	<u>*true*</u>	<u>*true*</u>
false	*true*	*false*	*false*	*false*	*true*	*true*	<u>*true*</u>	<u>*true*</u>
false	*true*	*false*	*false*	*true*	*false*	*false*	*false*	*true*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*false*

Inference by enumeration

- Depth-first enumeration of all models is sound and complete
- For *n* symbols, time complexity is $O(2^n)$, space complexity is $O(n)$

Logical equivalence

- Two sentences are logically equivalent iff true in same models: $\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) \quad \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) \quad \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) \quad \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) \quad \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha \quad \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) \quad \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) \quad \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) \quad \text{de Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) \quad \text{de Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) \quad \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) \quad \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}
$$

Example Proof by Deduction

- Knowledge

  S1: $B_{22} \Leftrightarrow (P_{21} \vee P_{23} \vee P_{12} \vee P_{32})$ *rule*

  S2: $\neg B_{22}$ *observation*

- Inferences

  S3: $(B_{22} \Rightarrow (P_{21} \vee P_{23} \vee P_{12} \vee P_{32})) \wedge$
  $((P_{21} \vee P_{23} \vee P_{12} \vee P_{32}) \Rightarrow B_{22})$ *S1,bi elim*

  S4:

  S5:

  S6:

  S7:

Example Proof by Deduction

- Knowledge

  S1: $B_{22} \Leftrightarrow (P_{21} \vee P_{23} \vee P_{12} \vee P_{32})$ *rule*

  S2: $\neg B_{22}$ *observation*

- Inferences

  S3: $(B_{22} \Rightarrow (P_{21} \vee P_{23} \vee P_{12} \vee P_{32})) \wedge$
  $((P_{21} \vee P_{23} \vee P_{12} \vee P_{32}) \Rightarrow B_{22})$ *S1,bi elim*

  S4: $((P_{21} \vee P_{23} \vee P_{12} \vee P_{32}) \Rightarrow B_{22})$ *S3, and elim*

  S5: $(\neg B_{22} \Rightarrow \neg(P_{21} \vee P_{23} \vee P_{12} \vee P_{32}))$ *contrapos*

  S6: $\neg(P_{21} \vee P_{23} \vee P_{12} \vee P_{32})$ *S2,S5, MP*

  S7: $\neg P_{21} \wedge \neg P_{23} \wedge \neg P_{12} \wedge \neg P_{32}$ *S6, DeMorg*

Proof methods

- Proof methods divide into (roughly) two kinds:
  - Application of inference rules
    - Legitimate (sound) generation of new sentences from old
    - Proof = a sequence of inference rule applications
      Can use inference rules as operators in a standard search
    - Typically require transformation of sentences into a normal form
  - Model checking
    - truth table enumeration (always exponential in n)
    - improved backtracking, e.g., Davis--Putnam-Logemann-Loveland (DPLL)
    - heuristic search in model space (sound but incomplete)
      e.g., min-conflicts-like hill-climbing algorithms

Resolution

Conjunctive Normal Form (CNF)

conjunction of disjunctions of literals

clauses

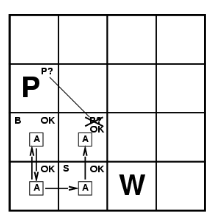
E.g., $(A \lor \neg B) \land (B \lor \neg C \lor \neg D)$

- Resolution inference rule (for CNF):

$$\frac{\ell_i \lor \ldots \lor \ell_k, \qquad m_1 \lor \ldots \lor m_n}{\ell_i \lor \ldots \lor \ell_{i-1} \lor \ell_{i+1} \lor \ldots \lor \ell_k \lor m_1 \lor \ldots \lor m_{j-1} \lor m_{j+1} \lor \ldots \lor m_n}$$

where ℓ_i and m_j are complementary literals.

E.g., $\dfrac{P_{1,3} \lor P_{2,2}, \qquad \neg P_{2,2}}{P_{1,3}}$



- Resolution is sound and complete for propositional logic

Resolution in Wumpus World

- There is a pit at 2,1 or 2,3 or 1,2 or 3,2
  - $P_{21} \vee P_{23} \vee P_{12} \vee P_{32}$
- There is no pit at 2,1
  - $\neg P_{21}$
- Therefore (by resolution) the pit must be at 2,3 or 1,2 or 3,2
  - $P_{23} \vee P_{12} \vee P_{32}$

Conversion to CNF

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

1. Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.

2. Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$.

3. Move $\neg$ inwards using de Morgan's rules and double-negation:

4. Apply distributivity law ($\wedge$ over $\vee$) and flatten:

Conversion to CNF

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

1. Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.
   $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

2. Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$.
   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$

3. Move $\neg$ inwards using de Morgan's rules and double-negation:
   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$

4. Apply distributivity law ($\wedge$ over $\vee$) and flatten:
   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$

$B_{22} \Leftrightarrow (P_{21} \vee P_{23} \vee P_{12} \vee P_{32})$

1. Eliminate $\Leftrightarrow$, replacing with two implications
   $(B_{22} \Rightarrow (P_{21} \vee P_{23} \vee P_{12} \vee P_{32})) \wedge ((P_{21} \vee P_{23} \vee P_{12} \vee P_{32}) \Rightarrow B_{22})$
2. Replace implication $(A \Rightarrow B)$ by $\neg A \vee B$
   $(\neg B_{22} \vee (P_{21} \vee P_{23} \vee P_{12} \vee P_{32})) \wedge (\neg(P_{21} \vee P_{23} \vee P_{12} \vee P_{32}) \vee B_{22})$
3. Move $\neg$ "inwards" (unnecessary parens removed)
   $(\neg B_{22} \vee P_{21} \vee P_{23} \vee P_{12} \vee P_{32}) \wedge ((\neg P_{21} \wedge \neg P_{23} \wedge \neg P_{12} \wedge \neg P_{32}) \vee B_{22})$
4. Distributive Law
   $(\neg B_{22} \vee P_{21} \vee P_{23} \vee P_{12} \vee P_{32}) \wedge (\neg P_{21} \vee B_{22}) \wedge (\neg P_{23} \vee B_{22}) \wedge (\neg P_{12} \vee B_{22}) \wedge (\neg P_{32} \vee B_{22})$

Last Step

- Sentences are now in CNF:
- (P1 v P2 v ~P3) ^ P4 ^ ~P5 ^ (P2 v P3)
- Create a separate clause corresponding to each conjunct
  - P1 v P2 v ~P3
  - P4
  - ~P5
  - P2 v P3

Simple Resolution Example

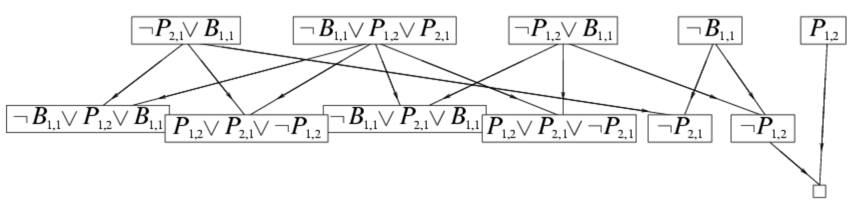
- When the agent is in 1,1, there is no breeze, so there can be no pits in neighboring squares
- Percept: ~B11
- Prove: ~P12.

Resolution example

- *KB* = $(B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1}$
- $\alpha = \neg P_{1,2}$

Resolution example

- *KB* = $(B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1}$
- $\alpha = \neg P_{1,2}$



Forward and backward chaining

- Horn Form (restricted)

  KB = conjunction of Horn clauses

  - Horn clause =
    - proposition symbol; or
    - (conjunction of symbols) ⇒ symbol
  - E.g., $C \wedge (B \Rightarrow A) \wedge (C \wedge D \Rightarrow B)$
- Modus Ponens (for Horn Form): complete for Horn KBs

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

- Can be used with forward chaining or backward chaining.
- These algorithms are very natural and run in linear time

Forward chaining

- Idea: fire any rule whose premises are satisfied in the *KB*,
  - add its conclusion to the *KB*, until query is found

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

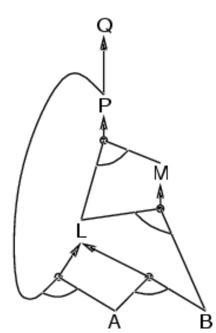
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

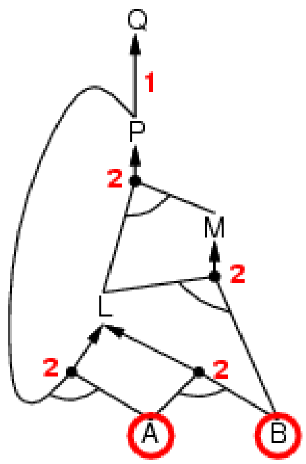
$A \wedge B \Rightarrow L$

A

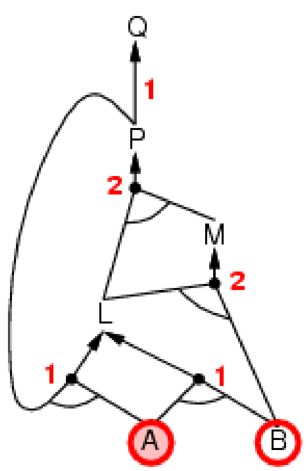
B



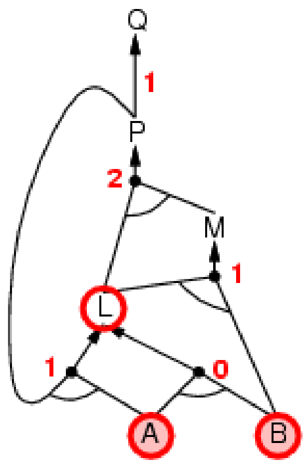
Forward chaining example



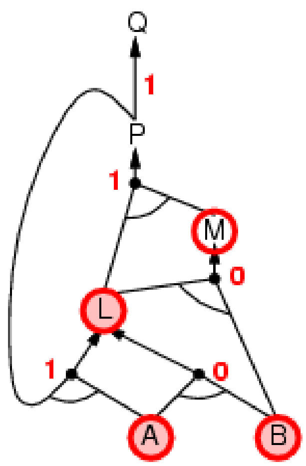
Forward chaining example



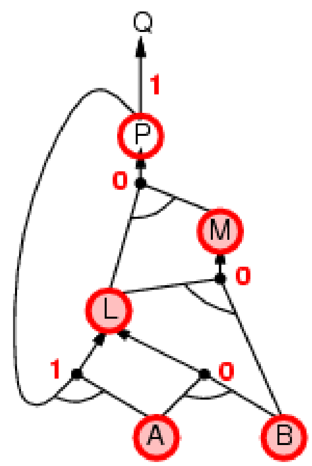
Forward chaining example



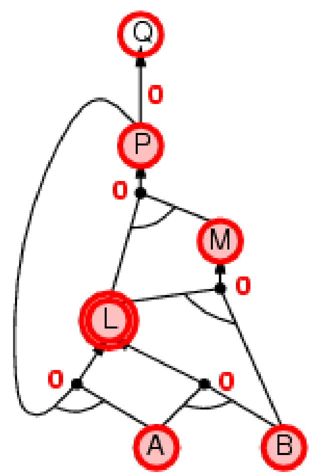
Forward chaining example



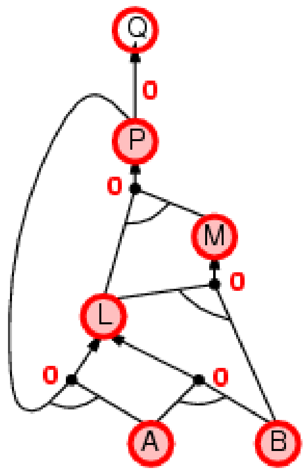
Forward chaining example



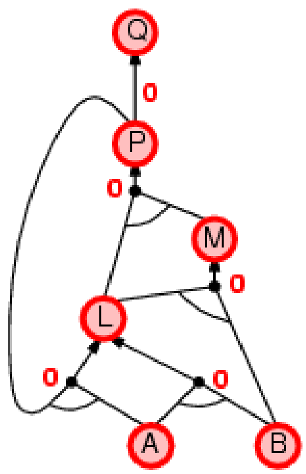
Forward chaining example



Forward chaining example



Forward chaining example



Backward chaining

Idea: work backwards from the query *q*:

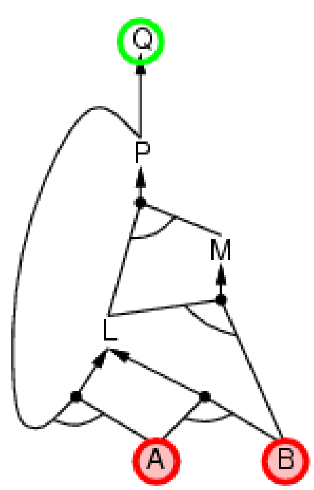
- to prove *q* by BC,
  - check if *q* is known already, or
  - prove by BC all premises of some rule concluding *q*

Avoid loops: check if new subgoal is already on the goal stack

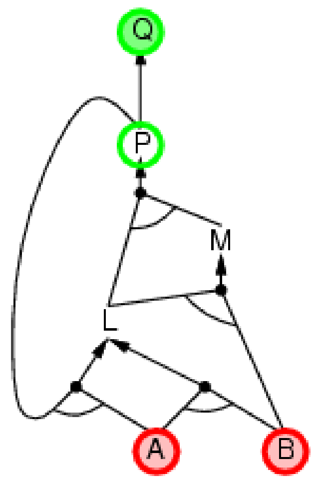
Avoid repeated work: check if new subgoal

1. has already been proved true, or
2. has already failed

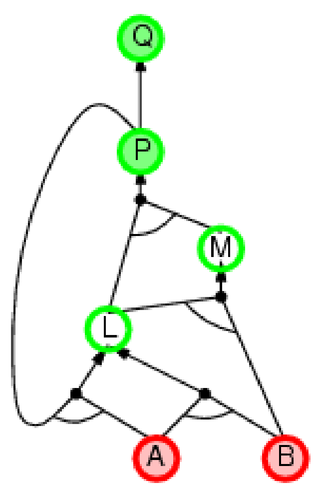
Backward chaining example



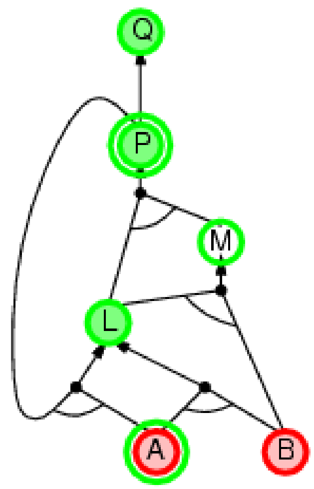
Backward chaining example



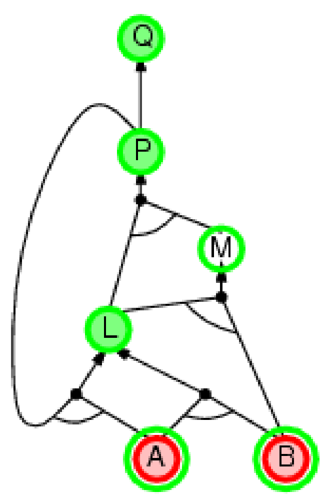
Backward chaining example



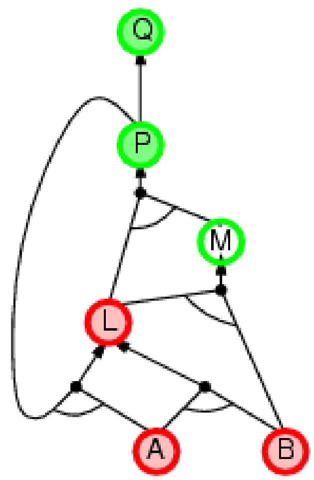
Backward chaining example



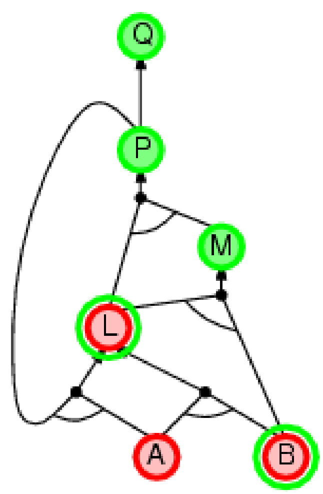
Backward chaining example



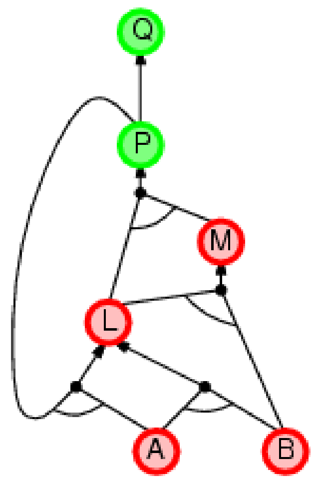
Backward chaining example



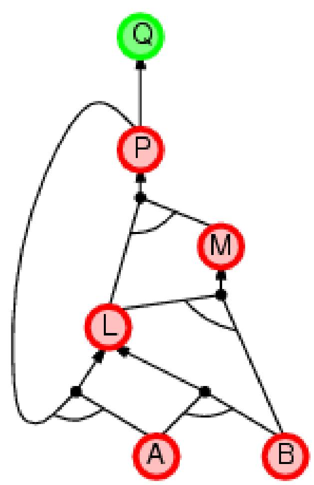
Backward chaining example



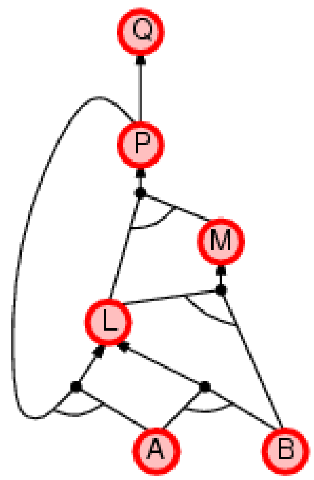
Backward chaining example



Backward chaining example



Backward chaining example



Forward vs. backward chaining

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

Efficient propositional inference

Two families of efficient algorithms for propositional inference:

Complete backtracking search algorithms

- DPLL algorithm (Davis, Putnam, Logemann, Loveland)
- Incomplete local search algorithms
  - `WalkSAT` algorithm

Expressiveness limitation of propositional logic

- KB contains "physics" sentences for every single square
- For every time *t* and every location [*x,y*],

$L^t_{x,y} \wedge FacingRight^t \wedge Forward^t \Rightarrow L^t_{x+1,y}$

- Rapid proliferation of clauses

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 wrt 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.
- Resolution is complete for propositional logic
  Forward, backward chaining are linear-time, complete for Horn clauses
- Propositional logic lacks expressive power