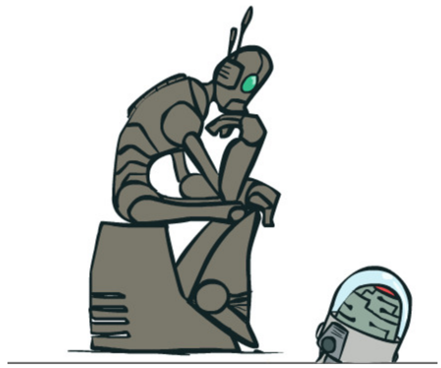
Artificial Intelligence

Chapter 7: Logical Agents



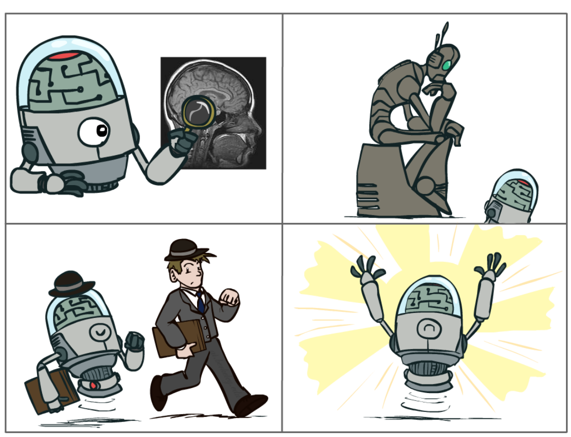
Updates and Additions: Dr. Siamak Sarmady
By: Tom Lenaerts
Université Libre de Bruxelles

What is AI?

The science of making machines that:

Think like people
cognitive science, neuroscience

Think rationally
long tradition, Aristotle... very difficult, how you end up acting is more important



Act like people
dating back to Alan Turing... general talk ... imitations ... it wasn't really leading us to build intelligence

Act rationally
This is what we most need

Outline

- Knowledge-based agents
- Wumpus world
- Logic in general
- Propositional and first-order logic
  - Inference, validity, equivalence and satifiability
  - Reasoning patterns
    - Resolution
    - Forward/backward chaining

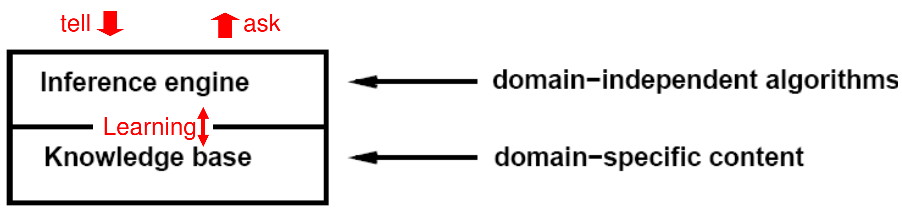
Knowledge Oriented Agents

- In this chapter we build agents that can:
  - Represent the world "using Logic"
  - Use logic to update representation of the world
  - Use the representations and logic to decide on their actions

- These agents will use Knowledge and Logic to live

Knowledge Base

- An expert system consists of a knowledge base (KB) and an inference engine (IE). The system does not have direct access to the outside world.
  - Depends on us to add knowledge and query information
  - Responds to us and does not act directly



- Sometimes the system is connected to environment directly and it can act. These are called logical agents. It can get information using percepts, reason and then act.

Knowledge

- **Knowledge Base:** is a set of representations of what the system knows about the world (objects and classes of objects, the fact about objects, relationships among objects, etc.).
- Each individual representation is called a sentence. The sentences are expressed in a knowledge representation language. The knowledge can sometimes be further divided into:
  - **Rules:** sentences in If…then… form that are used for reasoning
  - **Facts:** information which is gathered by perception, are used with rules to reason
- Examples of sentences:
  - The moon is made of green cheese
  - If A is true then B is true
  - A is false
  - All humans are mortal
  - Confucius is a human
- Normally, the KB is initialized with background knowledge, before interacting with the world (some rules)

Inference Engine (IE)

- **Inference:** when one ASKs questions of the KB, the answer should *follow* from what has been TELLed to the KB previously.
- The Inference engine derives new sentences from the input and KB
- The inference mechanism depends on the type representations in KB
- A KB agent operates as follows:
  1. It receives percepts from environment
  2. It computes what action it should perform (by IE and KB)
  3. It performs the chosen action (some actions are simply inserting inferred new facts into KB).

Generic KB-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
```

1. TELLs the knowledge base what it perceives
2. ASKs the KB what action should it perform.
   - Extensive reasoning done on existing state of the world, about the outcome of actions etc.
3. Once action is chosen, it is recorded using TELL and then it is executed
   - Necessary, so that KB knows the selected action is actually executed

- Details of representation lang. are abstracted by the two functions that interface between sensors-actuators and KB-inference sections i.e. MAKE-PERCEPT-SENTENCE and MAKE-ACTION-QUERY.
- Details of inference mechanism are hidden inside TELL and ASK.

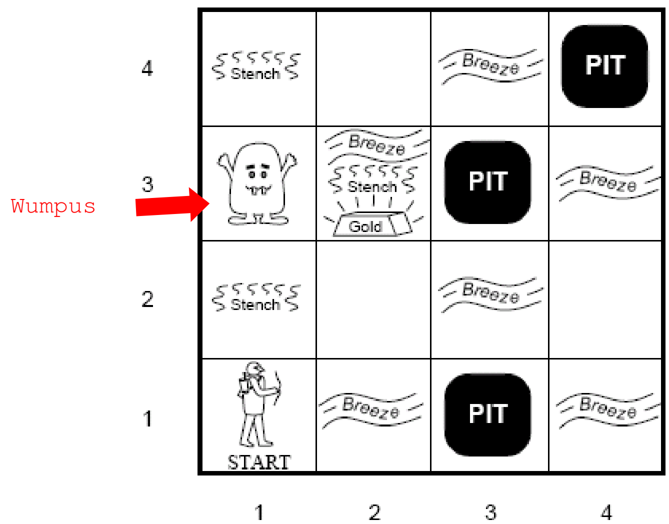
Abilities KB agent

A logical agent must be able to:

- Represent states and actions:
  - Convert what it percepts into logic sentences and give them to KB
  - Present actions it can perform in logic form
- Incorporate new percepts and update internal representation of the world:
  - Update the KB with the new percepts
- Deduce hidden properties of the world:
  - Deduce new sentences that present the world from those in KB (and those coming from percept)
- Deduce appropriate actions:
  - Reason using KB and select suitable actions

Wumpus World

A Typical Wumpus World



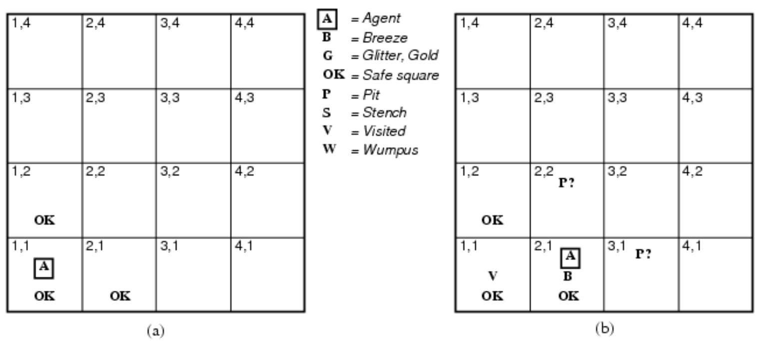
Wumpus World PEAS Description

- Performance measure
  - gold +1000, death -1000
  - -1 per step, -10 for using the arrow
- Environment and its rules
  - 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

- Observable?
  - No, only local perception
- Deterministic?
  - Yes, outcome 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.

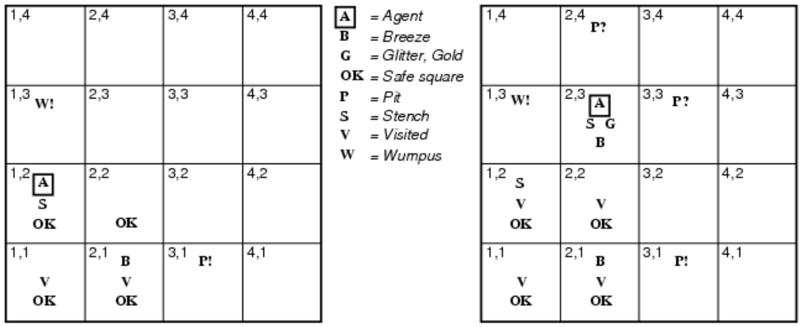
Exploring the Wumpus World



[1,1] The KB initially contains the rules of the environment.
The first percept is [*none, none,none,none,none*], move to safe cell e.g. [2,1]

[2,1] breeze which indicates that there is a pit in [2,2] or [3,1], return to [1,1] to try next safe cell i.e. [1,2]

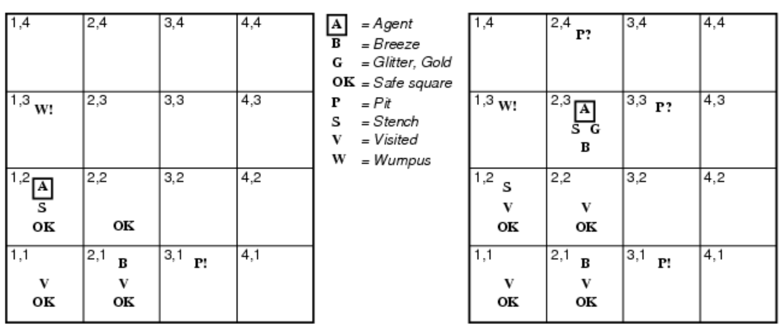
Exploring the Wumpus World



(a) (b)

[1,2] Stench in cell which means that wumpus is in [1,3] or [2,2]
YET … not in [1,1]
YET … not in [2,2] or stench would have been detected in [2,1]
THUS … wumpus is in [1,3] (you can shoot the wumpus)
THUS [2,2] is safe because of lack of breeze in [1,2] (so it wasn't a pit)
THUS pit in [3,1]
move to next safe cell [2,2] (you can also shoot the wumpus)

Exploring the Wumpus World



(a) (b)

[2,2] move to [2,3]
[2,3] detect glitter , smell, breeze
THUS pick up gold
THUS pit in [3,3] or [2,4]

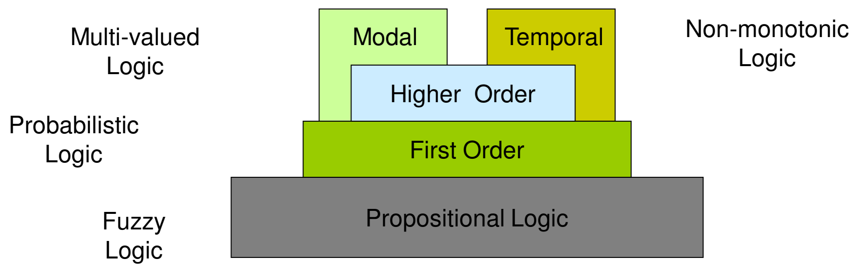
Exploring the Wumpus World

- Each time the agent reaches a conclusion, the **conclusion** is guaranteed to be correct, if the available **information** (in KB and from perception) is correct.
- In the coming sections we first talk about propositional logic.
- We then describe how we can build agents that can **represent information** and **draw conclusions** using Logic.

What is a logic?

- A formal language
  - Syntax – what expressions are legal (well-formed). A language consists of all those well formed expressions.
  - Semantics – what legal expressions mean
    - in logic the truth of each sentence with respect to each possible world.
- Examples in the language of arithmetic
  - x + 2 >= y is a sentence, x2 + y = is not a sentence (because it is not well formed)
  - x + 2 >= y is true in a world where x = 7 and y = 1
  - x + 2 >= y is false in a world where x = 0 and y = 6

Logic as a KR(Knowledge Representation) language



Entailment

- One thing follows from another

  $KB \models \alpha$

  KB متضمن α است یا α از KB منتج می شود...

  KB entails sentence α *if and only if* α is true in worlds where KB is true.

- E.g. x + y = 4 entails 4 = x + y
- Entailment is a relationship between sentences that is based on semantics.

- Think of entire KB as one big statement that is true or false
  - Sentences in the KB ANDed together to form one big sentence
- **ASKing KB a question:** does KB entail the sentence "there is no pit in [1, 1]"?
  - If so, add it to the KB. The new sentence is produced by **logical inference**

Models

- Logicians typically think in terms of models, which are formally structured worlds with respect to which truth can be evaluated.

- *m* is a model of a sentence α if α is true in *m* *(i.e. sentence* α is true with the values m specifies for the parts of α)
- Examples:
  - m_1(x=2,y=2) is a model of the sentence "x+y = 4". m_2(x=1,y=3) is also a model of this sentence.
  - m_1(P=true, Q=true, R=false) is a model of the sentence $P \wedge Q \wedge \neg R$
  - m_2(P=true, Q=true) is not a model of the sentence $P \wedge Q \wedge \neg R$

- $M(\alpha)$ is the set of all models of α (i.e. all the possible values of the parts of α)
- How many models can be defined over n propositional symbols?

Model Checking in Propositional Logic

- Assignment of a truth value – true or false – to every atomic sentence
- Examples:
  - Let A, B, C, and D be the propositional symbols
  - is m = {A=true, B=false, C=false, D=true} a model?
  - is m' = {A=true, B=false, C=false} a model?
- How many models can be defined over n propositional symbols?

Model Checking

- **Model Checking:** Assignment of a truth value – true or false – to every atomic sentence (aka. Checking the sentence for all models)

- One type of logical inference
  - Generate all possible models
  - If a models generates contradictions with sentences in the KB, KB is **false** in those models.
  - Is the ASKed sentence true, for **ALL** models in which KB is true? If so, KB entails the sentence

- Example
  - "2 + x = z" is in the KB
  - Models are all possible value pairs for x and z
  - In all models in which "2 + x = z" does not hold, the KB is false

Model Checking

- One type of logical inference is **model checking**
  - Generate all possible models
  - The KB is **false** in all models that generate contradictions with sentences in the KB
  - For **ALL** models in which KB is true, is the ASKed sentence also true? If so, KB entails the sentence

All possible models containing P, Q, R

Model	P	$P \wedge Q \wedge \neg R$	$P \wedge R$	KB
1	T	T	T	T
2	T	T	F	F
3	T	T	F	T
4	T	F	F	F
5	F	F	F	F
6	F	F	F	F
7	F	F	F	F
8	F	F	F	F

Which models contradict with KB?

2,4, 5, 6, 7,8

KB entails which of the three sentences?

$KB \models P \wedge Q \wedge \neg R$

$KB \models P$

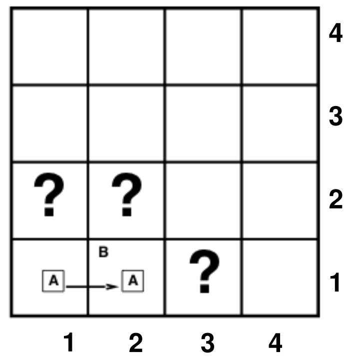
Wumpus world model

Note: assume we are only interested in Pits

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

Consider possible models for ?s assuming only pits

3 Boolean choices $\Rightarrow$ 8 possible models



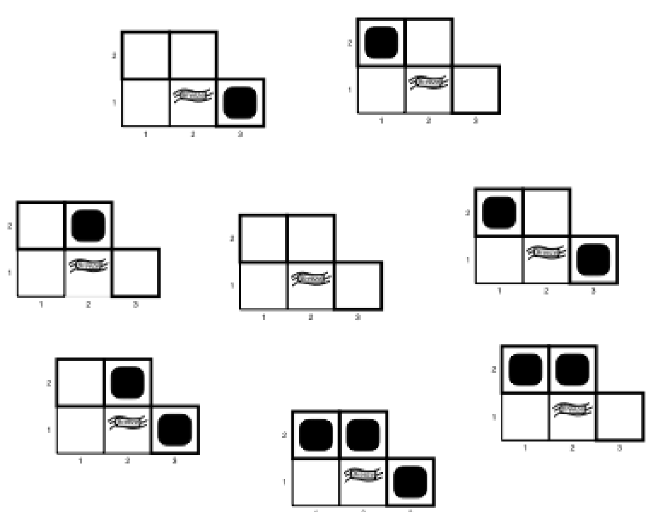
Wumpus world model

Situation after:

- detecting nothing in [1,1]
- moving right to [2,1]
- and perceiving breeze in [2,1]

If we only want to consider the location of Pits, what **possible values** could the three cells ([1,2], [2,2] and[3,1]) have?

3 Boolean Choices => 8 possible models

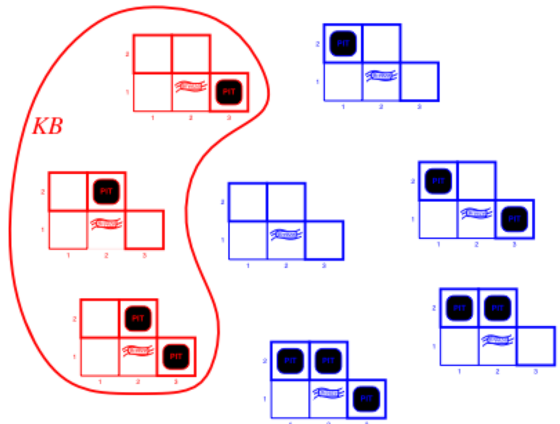


Wumpus world model

KB is the result of:

- Initial rules of wumpus world
- The observations of the two moves

Describe how KB is concluded…



No breeze at (1,1)
Breeze at (1,2)

KB = wumpus-world rules + observations

Wumpus world model

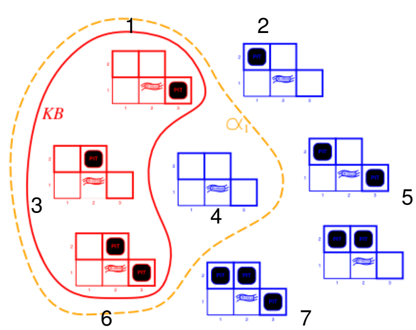
α_1= "[1,2] is not pit"

Can we entail α_1 from KB?

From the definition:

KB entails sentence α *if and only if* <u>α is true in worlds where KB is true</u> : *KB* ⊨ *α*

#	KB	$\alpha 1$
1	T	T
2	F	F
3	T	T
4	F	T
5	F	F
6	T	T
7	F	F
8	F	F



KB = wumpus-world rules + observations

α_1 = "[1,2] is safe", $KB \models \alpha_1$, proved by model checking

Wumpus world model

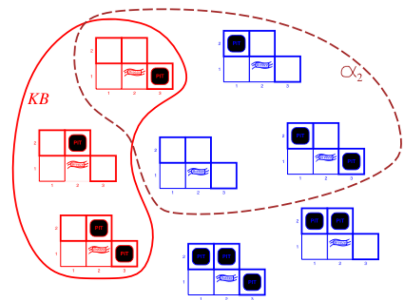
α_1= "[2,2] is not pit"

Can we entail α_1 from KB?

From the definition:

KB entails sentence α *if and only if* α is true in worlds where KB is true : $KB \models \alpha$

#	KB	$\alpha 1$
1	T	T
2	F	T
3	T	F
4	F	T
5	F	T
6	T	F
7	F	F
8	F	F



KB = wumpus-world rules $\cup$ observations

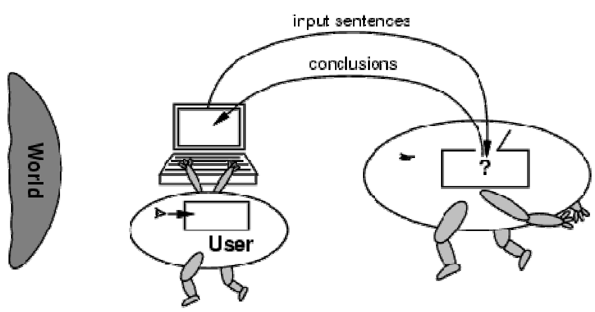
α_2 = "[2,2] is safe", $KB \not\models \alpha_2$

Logical inference

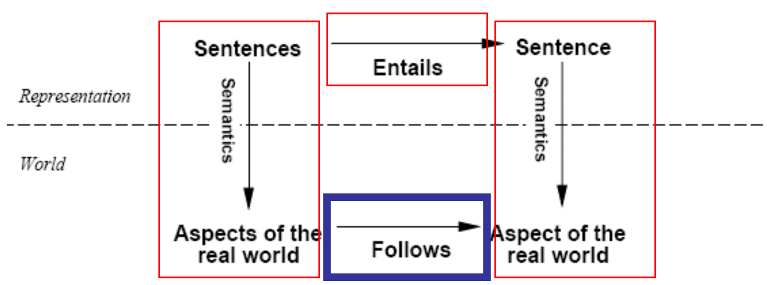
- **Inference:** is the process of deriving a specific sentence from a KB (where the sentence must be entailed by the KB)
  - KB $|-_i$ α = sentence α can be derived from KB by procedure I
- If an algorithm only derives entailed sentences it is called *sound* or *truth preserving*.
  - Otherwise it just makes things up!!!
  - *i is sound if whenever KB $|-_i$ α it is also true that KB $\models$ α*
- Completeness: the algorithm can derive every sentence that is entailed.
  - *i is complete if whenever KB $\models$ α it is also true that KB $|-_i$ α*
- The notion of entailment can be used for logic inference.
  - Model checking (see wumpus example): enumerate all possible models and check whether α is true.

No independent access to the world

- The reasoning agent often gets its knowledge about the facts of the world as a sequence of logical sentences and must draw conclusions only from them without independent access to the world.
- Thus it is very important that the agent's reasoning is sound!



Schematic perspective



If KB is true in the real world, then any sentence α derived from KB by a sound inference procedure is also true in the real world.

Propositional Logic

- Sometimes called "Boolean Logic"
  - Sentences are true (T) or false (F)
- Words of the syntax include propositional symbols…
  - P, Q, R, …
  - P = "I'm hungry", Q = "I have money",
    R = "I'm going to a restaurant"
- … and logical connectives
  - $\neg$ negation NOT
  - $\wedge$ conjunction AND
  - $\vee$ disjunction OR
  - $\Rightarrow$ implication IF-THEN
  - $\Leftrightarrow$ biconditional IF AND ONLY IF

Propositional Logic

- Atomic sentences
  - Propositional symbols
  - True or false
- Complex sentences
  - Groups of propositional symbols joined with connectives, and parenthesis if needed
  - $(P \wedge Q) \Rightarrow R$
  - Well-formed formulas following grammar rules of the syntax

Symbols of PL, Order of Precedence

Symbols

- Connectives: ¬, ∧, ∨, ⇒
- Propositional symbols, e.g., P, Q, R, …
- *True*, *False*

Order of Precedence

- ¬ ∧ ∨ ⇒
- Examples:
  - ¬ A ∨ B ⇒ C is equivalent to ((¬A)∨B)⇒C

Syntax of PL

- sentence → atomic sentence | complex sentence
- atomic sentence → Propositional symbol, *True*, *False*
- Complex sentence → ¬sentence
  | (sentence ∧ sentence)
  | (sentence ∨ sentence)
  | (sentence ⇒ sentence)

- Examples:
  - ((P ∧ Q) ⇒ R)
  - (A ⇒ B) ∨ (¬C)

Semantics of PL

- It specifies how to determine the truth value of any sentence in a model m
- The truth value of *True* is *True*
- The truth value of *False* is *False*
- The truth value of each atomic sentence is given by m
- The truth value of every other sentence is obtained recursively by using truth tables

P	Q	R	$P \wedge Q$	$(P \wedge Q) \Rightarrow R$
T	T	T	T	T
F	T	T	F	T
T	F	T	F	T
F	F	T	F	T
T	T	F	T	F
F	T	F	F	T
T	F	F	F	T
F	F	F	F	T

Propositional Logic Semantics

- Truth tables for all connectives
- Given each possible truth value of each propositional symbol, we can get the possible truth values of the expression

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
T	T	F	T	T	T	T
F	T	T	F	T	T	F
T	F	F	F	T	F	F
F	F	T	F	F	T	T

Propositional Logic Example

- Propositional symbols:
  - A = “The car has gas”
  - B = “I can go to the store”
  - C = “I have money”
  - D = “I can buy food”
  - E = “The sun is shining”
  - F = “I have an umbrella”
  - G = “I can go on a picnic”
- If the car has gas, then I can go to the store
  - $A \Rightarrow B$
- I can buy food if I can go to the store and I have money
  - $(B \wedge C) \Rightarrow D$
- If I can buy food and either the sun is not shining or I have an umbrella, I can go on a picnic
  - $D \wedge (\neg E \vee F) \Rightarrow G$

Proof (and inference) Methods

- Model checking (Enumeration of Models)
  - Truth table enumeration (exponential in n)
  - Improved backtracking (on all values of all variables)
  - Heuristic search in model space (sound but incomplete) e.g. min-conflicts like hill-climbing or genetic algorithms
- Applications 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 algorithm
  - Typically requires translation of sentences into a normal form

Wumpus World Sentences

For simplicity let's only consider breeze and pits:

Facts:

- 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]

- $\neg P_{1,1}$: means no pit in [1,1]
- $\neg B_{1,1}$: means no breeze in [1,1]
- $B_{2,1}$: means breeze in [2,1[

Rules:

- "Pits cause breezes in adjacent squares" (We should use bidirectional for rules and these should be written for every cell)

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

- A square is breezy if and only if there is an adjacent pit

A Simple Knowledge Base

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	R_1	R_2	R_3	R_4	R_5	KB
false	*false*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*false*	*false*
false	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*false*	*true*	*false*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*false*	*true*	*true*	*false*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*true*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*true*	*true*	*false*	*true*	*false*

A Simple Knowledge Base

We write all the possible states for the variables (all possible models) of cells we have info about…

Questions:

Can we entail ~$P_{1,2}$ from KB?

Can we entail $P_{2,2}$ from KB?

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	R_1	R_2	R_3	R_4	R_5	KB
false	*false*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*false*	*false*
false	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*false*	*true*	*false*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*false*	*true*	*true*	*false*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*true*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*true*	*true*	*false*	*true*	*false*

- **R1:** $\neg P_{1,1}$
- **R2:** $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
- **R3:** $B_{2,1}$ $(P_{1,1} \vee P_{2,2} \vee P_{3,1})$
- **R4:** $\neg B_{1,1}$
- **R5:** $B_{2,1}$

- First 3 sentences are valid in all games.
- Last 2 are the results of observations.

- KB consists of sentences R_1 thru R_5
- $R1 \wedge R2 \wedge R3 \wedge R4 \wedge R5$

Enumeration of Models

P: Set of propositional symbols in $\{KB, \neg\alpha\}$

n: Size of P

ENTAILS?(KB,α)

For each of the 2^n models on P do

If it is a model of $\{KB , \neg\alpha\}$ then return no

Return yes

The algorithm builds the truth table of both KB and $\neg\alpha$, then checks whether KB $\wedge \neg\alpha$ is not satisfiable. In next few slides we will see:

- Satisfiability is connected to inference via the following
  - KB $\vDash \alpha$ iff (KB $\wedge \neg\alpha$) is unsatisfiable
  - proof by contradiction

A Simple Knowledge Base

```
function TT-ENTAILS?(KB, α) returns true or false
   symbols ← a list of the proposition symbols in KB and α
   return TT-CHECK-ALL(KB, α, symbols, [])

function TT-CHECK-ALL(KB, α, symbols, model) returns true or false
   if EMPTY?(symbols) then
       if PL-TRUE?(KB, model) then return PL-TRUE?(α, model)
       else return true
   else do
       P ← FIRST(symbols); rest ← REST(symbols)
       return TT-CHECK-ALL(KB, α, rest, EXTEND(P, true, model)) and
                 TT-CHECK-ALL(KB, α, rest, EXTEND(P, false, model))
```

- Every known inference algorithm for propositional logic has a worst-case complexity that is exponential in the size of the input. (co-NP complete)

Inference with Truth Tables

- Sound
  - Only infers true conclusions from true premises
- Complete
  - Finds all facts entailed by KB
- Time complexity = $O(2^n)$
  - Checks all truth values of all symbols
- Space complexity = O(n)
  - Keeps one row of truth table at a time

Equivalence, Validity, Satisfiability

- A sentence is valid (Tautology) if it is true in all models
  - e.g. True, $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \Rightarrow B$
- Validity is connected to inference via the Deduction Theorem
  - $KB \models \alpha$ iff $(KB \Rightarrow \alpha)$ is valid
- A sentence is satisfiable if it is True in some model
  - e.g. $A \vee B$, C
- A sentence is unstatisfiable if it is True in no models
  - e.g. $A \wedge \neg A$
- Satisfiability is connected to inference via the following
  - $KB \models \alpha$ iff $(KB \wedge \neg\alpha)$ is unsatisfiable
  - proof by contradiction

Equivalence, Validity, Satisfiability

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

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

Inference Rules

- Inference rules
  - $\dfrac{(\alpha \Rightarrow \beta),\ \alpha}{\beta}$ Modus Ponens
  - $\dfrac{(\alpha \wedge \beta)}{\alpha}$ And-Elimination
  - $\dfrac{\alpha,\ \beta}{(\alpha \wedge \beta)}$
  - $\dfrac{(\alpha \vee \beta),\ \neg\beta}{\alpha}$
  - $\dfrac{(\alpha \vee \beta),\ (\neg\beta \vee \gamma)}{(\alpha \vee \gamma)}$

Inference with Rules

- Speeds up inference by using inference rules
- Use along with logical equivalences
- No need to enumerate and evaluate every truth value

Reasoning Patterns

- **And Elimination**

$$\frac{\alpha \wedge \beta}{\alpha}$$

- From a conjunction, any of the conjuncts can be inferred
- (WumpusAhead $\wedge$ WumpusAlive), WumpusAlive can be inferred

- **Modus Ponens**

$$\frac{\alpha \Rightarrow \beta, \quad \alpha}{\beta}$$

- Whenever sentences of the form $\alpha \Rightarrow \beta$ and α are given, then sentence β can be inferred
- (WumpusAhead $\wedge$ WumpusAlive) $\Rightarrow$ Shoot

  Given (WumpusAhead $\wedge$ WumpusAlive) , Shoot can be inferred

Wumpus World & Proof by Deduction (Inference Rules)

KB (R4): $\neg B_{1,1}$

KB (R2): $B_{1,1} \Leftrightarrow (P_{2,1} \vee P_{1,2})$

- Biconditional elimination

1. $(B_{1,1} \Rightarrow (P_{2,1} \vee P_{1,2})) \wedge ((P_{2,1} \vee P_{1,2}) \Rightarrow B_{1,1})$
   - And elimination
2. $(P_{2,1} \vee P_{1,2}) \Rightarrow B_{1,1}$
   - Contraposition
3. $\neg B_{1,1} \Rightarrow \neg(P_{2,1} \vee P_{1,2})$
   - Modus Ponens (with R4)
4. $\neg(P_{2,1} \vee P_{1,2})$
   - De Morgan’s Rule
5. $\neg P_{2,1} \wedge \neg P_{1,2}$

Evaluation of Deductive Inference

- Sound
  - Yes, because the inference rules themselves are sound (This can be proven using a truth table argument).
- Completeness
  - If we allow all possible inference rules, we're searching in an infinite space, hence not complete (successor function gives infinite options)
  - If we limit inference rules, we run the risk of leaving out the necessary one…
- Monotonic
  - If we have a proof, adding information to the DB will not invalidate the proof

$$\text{if } KB \vDash \alpha \;\; \text{then } \; KB \wedge \beta \vDash \alpha$$

- Can be efficient (more than model testing)
  - NP-Complete but (just like model testing) but in this method we can ignore irrelevant propositions of KB when searching for a proof while truth table encounter with exponential explosion of models

Resolution

- **Resolution:** is a method that provides a complete inference mechanism (search-based) using only one rule of inference
- **Unit resolution rule:** if we have a disjunctive sentence (clause) and a complement of a literal

$m = \sim l_i$

$$\frac{l_1 \vee \ldots \vee l_k, \qquad m}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k}$$

Can entail a sentence in which the complements are removed

- **Example:**

$$\frac{P_{1,1} \vee P_{3,1}, \qquad \sim P_{1,1}}{P_{3,1}}$$

- Complementary literals $P_{1,1}$ and $\neg P_{1,1}$ "cancel out"

Resolution

- Resolution rule:

$$\frac{l_1 \vee \ldots \vee l_k, \quad m_1 \vee \ldots \vee m_n}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n}$$

- In other form:
  - Given: $P_1 \vee P_2 \vee P_3 \ldots \vee P_n$, and $\neg P_1 \vee Q_1 \ldots \vee Q_m$
  - Conclude: $P_2 \vee P_3 \ldots \vee P_n \vee Q_1 \ldots \vee Q_m$
  - Complementary literals P_1 and $\neg P_1$ "cancel out"
- Example:

$$\frac{P_{1,1} \vee P_{3,1} \vee P_{2,1}, \quad \sim P_{1,1} \vee \sim P_{2,2} \vee P_{2,3}}{P_{3,1} \vee P_{2,1} \vee \sim P_{2,2} \vee P_{2,3}}$$

- Complementary literals P_1 and $\neg P_1$ "cancel out"
- **Why it works:** Consider 2 cases, P_1 is true, and P_1 is false

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) we entail that the pit must be at 2,3 or 1,2 or 3,2
  - $P_{23} \vee P_{12} \vee P_{32}$

Conjunctive Normal Form

- **Computational resolution:** To apply resolution mechanically, facts need to be in Conjunctive Normal Form (CNF).
- **CNF form:** A formula is in conjunctive normal form (CNF) or clausal normal form if it is a conjunction of clauses, where a clause is a disjunction of literals;
- **In simply language:** it is an AND of ORs.

$$(A \vee \sim B) \wedge (B \vee \sim C \vee \sim D)$$

- **K-CNF sentences:** Has exactly K literals in each conjunctional statement

$$(l_{1,1} \vee \ldots \vee l_{1,k}) \wedge \ldots \wedge (l_{n,1} \vee \ldots \vee l_{n,k})$$

Conversion to CNF

Example:

$B_{11} \Leftrightarrow (P_{12} \vee P_{21})$

1. Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$
   $(B_{11} \Rightarrow ((P_{12} \vee P_{21})) \wedge ((P_{12} \vee P_{21}) \Rightarrow B_{11}))$
2. Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$
   $(\neg B_{11} \vee P_{12} \vee P_{21}) \wedge (\neg(P_{12} \vee P_{21}) \vee B_{11}))$
3. Move $\neg$ in using deMorgan's rules and double negation
   $(\neg B_{11} \vee P_{12} \vee P_{21}) \wedge ((\neg P_{12} \wedge \neg P_{21}) \vee B_{11}))$
4. Distribute $\vee$ over $\wedge$
   $(\neg B_{11} \vee P_{12} \vee P_{21}) \wedge (\neg P_{12} \vee B_{11}) \wedge (\neg P_{21} \vee B_{11})$

   Set of clauses:
   $\{(\neg B_{11} \vee P_{12} \vee P_{21}), (\neg P_{12} \vee B_{11}) ,(\neg P_{21} \vee B_{11})\}$

Conversion to CNF

1. $B_{22} \Leftrightarrow (P_{21} \lor P_{23} \lor P_{12} \lor P_{32})$

2. Eliminate $\Leftrightarrow$, replacing with two implications
   $(B_{22} \Rightarrow (P_{21} \lor P_{23} \lor P_{12} \lor P_{32})) \land ((P_{21} \lor P_{23} \lor P_{12} \lor P_{32}) \Rightarrow B_{22})$

3. Replace implication $(A \Rightarrow B)$ by $\neg A \lor B$
   $(\neg B_{22} \lor (P_{21} \lor P_{23} \lor P_{12} \lor P_{32})) \land (\neg(P_{21} \lor P_{23} \lor P_{12} \lor P_{32}) \lor B_{22})$

4. Move $\neg$ "inwards" (unnecessary parens removed)
   $(\neg B_{22} \lor P_{21} \lor P_{23} \lor P_{12} \lor P_{32}) \land ((\neg P_{21} \land \neg P_{23} \land \neg P_{12} \land \neg P_{32}) \lor B_{22})$

4. Distributive Law
   $(\neg B_{22} \lor P_{21} \lor P_{23} \lor P_{12} \lor P_{32}) \land (\neg P_{21} \lor B_{22}) \land (\neg P_{23} \lor B_{22}) \land (\neg P_{12} \lor B_{22}) \land (\neg P_{32} \lor B_{22})$

(Final result has 5 clauses)

Proof using Resolution

- In order to show $KB \vDash \alpha$ we should show $KB \land \neg\alpha$ is unsatisfiable
  (proof by contradiction برهان خلف)

- Algorithm:
  1. First $KB \land \neg\alpha$ is converted into CNF form
  2. The resolution rule is applied to the resulting clauses (disjunctions). Each pair that has complementary literals is resolved to produce a new clause.
  3. The newly built clause is added to the set of clauses if it is not already added.
  4. The process continues until one of the following happens
     - There are no new clauses that can be added (by resolution). In this case α does not entail β.
     - *An application of the resolution rule derives the empty clause. In this case the KB $\land \neg\alpha$ is unsatisfiable and therefore α entails β.*

- If we prove that KB $\land \neg$P derives a contradiction (empty clause) and we know KB is true, then $\neg$P must be false, so P must be true!

Proof using Resolution (Example)

We can apply the resolution procedure to a very simple inference in the wumpus world.

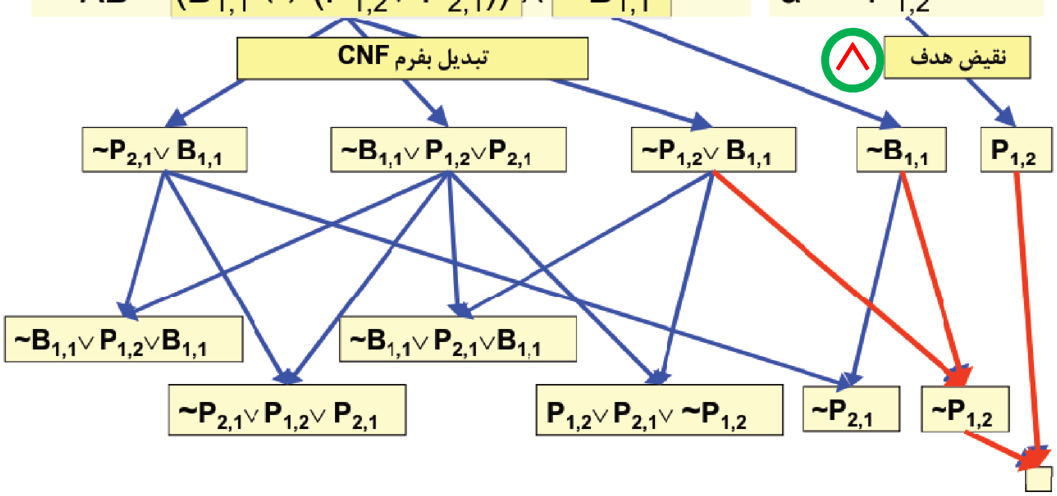
- When the agent is in [1,1], there is no breeze, so there can be no pits in the neighboring squares. The relevant knowledge base is

  KB = $R_2 \wedge R_4 = (B_{11} \Leftrightarrow (P_{12} \vee P_{21})) \wedge \neg B_{11}$

- We wish to prove α which is , say, $\neg P_{12}$
- We now should prove $KB \wedge \neg\alpha$ is un-satisfiable to prove α can be entailed
- We convert the $KB \wedge \neg\alpha$ into CNF form and continue the algorithm
- We reach an empty clause which means $KB \wedge \neg\alpha$ is un-satisfiable and therefore α can be entailed.

Proof using Resolution (Example)

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



Evaluation of Resolution

- Resolution is sound
  - Because the resolution rule is true in all cases
- Resolution is complete
  - Provided a complete search method is used to find the proof, if a proof can be found it will
  - **Note:** you must know what you're trying to prove in order to prove it!
- Resolution is exponential
  - The number of clauses that we must search grows exponentially…
  - However if we use Horn clauses, resolution will not have this problem

Horn Clauses

- A Horn Clause is a CNF clause with exactly one positive literal
  - The positive literal is called the head
  - The negative literals are called the body
  - Example $B_{11} \vee \neg P_{12} \vee \neg P_{21}$
- Why horn clauses?
  1. Because they are in fact implications or can be converted to them. KB rules are normally implications.
     - The above example is equal to $P_{12} \wedge P_{21} \Rightarrow B_{11}$ (using the rule $p \Rightarrow q \equiv {\sim}p \vee q$)
  2. Horn Clauses form the basis of forward and backward chaining which are easy and natural. The Prolog language is based on Horn Clauses.
  3. Deciding entailment with Horn Clauses is linear in the size of the knowledge base.

Reasoning with Horn Clauses

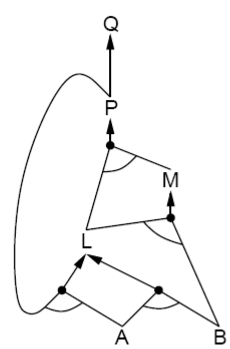
- Forward Chaining
  - For each new piece of data, using the rules, generate all new facts, until the desired fact is generated
  - Data-directed reasoning
- Backward Chaining
  - To prove the goal, find a clause that contains the goal as its head, and prove the body recursively
  - (Backtrack when you chose the wrong clause)
  - Goal-directed reasoning

Forward Chaining

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

- We are interested in Q
- The facts we have are A and B
- From bottom to up, we use the rules we have to find new facts.
- The graph shows just the useful part of a reasoning that produces what we need.

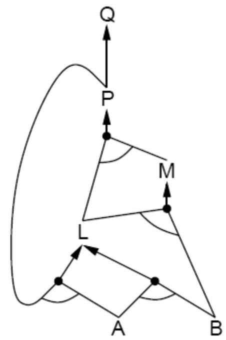
$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$



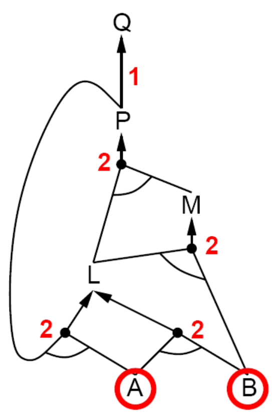
Forward Chaining

- AND-OR Graph
  - multiple links joined by an arc indicate conjunction – every link must be proved
  - multiple links without an arc indicate disjunction – any link can be proved

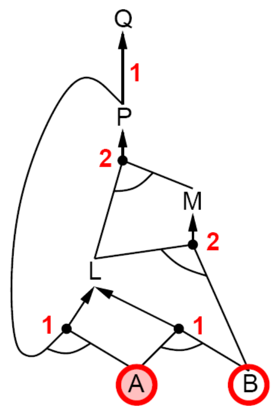
$$
\begin{aligned}
&P \Rightarrow Q\\
&L \wedge M \Rightarrow P\\
&B \wedge L \Rightarrow M\\
&A \wedge P \Rightarrow L\\
&A \wedge B \Rightarrow L\\
&A\\
&B
\end{aligned}
$$



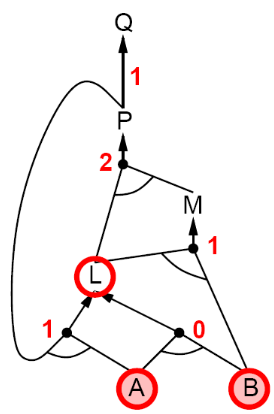
Forward Chaining



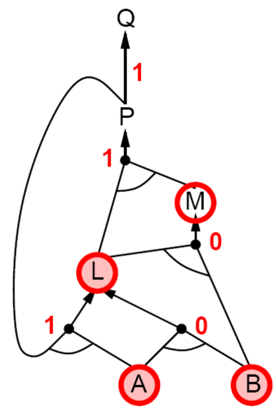
Forward Chaining



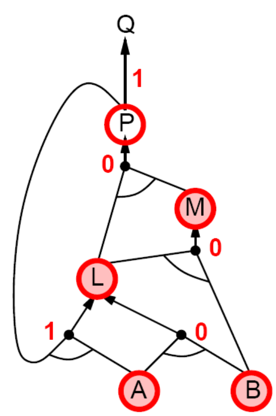
Forward Chaining



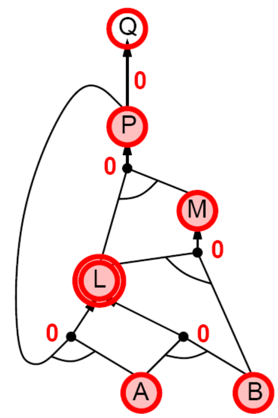
Forward Chaining



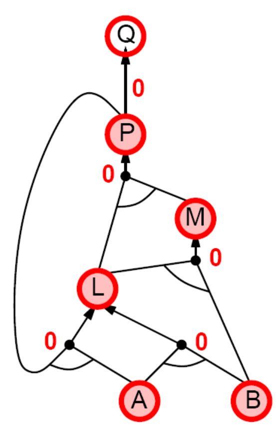
Forward Chaining



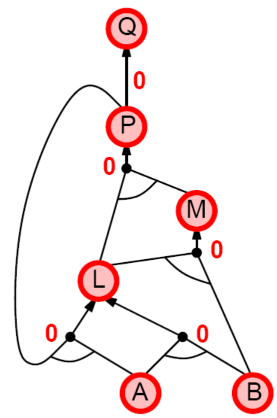
Forward Chaining



Forward Chaining



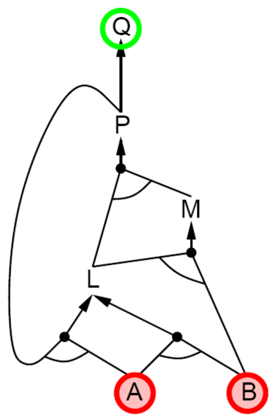
Forward Chaining



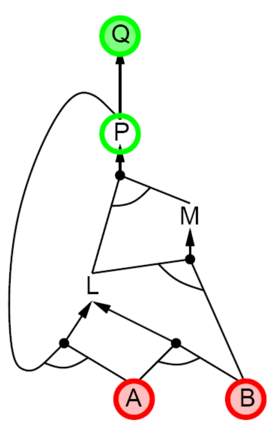
Backward Chaining

- Idea: work backwards from the query q:
  - To prove q by backward chaining:
    - Check if q is known already (in the list of KB).
    - Otherwise find those implications in the KB that conclude q.
    - If all the premises of one of those implications can be proved true (possibly by backward chaining) then q is true.
- Avoid loops
  - Check if new sub-goal is already on the goal stack
- Avoid repeated work: check if new subgoal
  - Has already been proved true, or
  - Has already failed

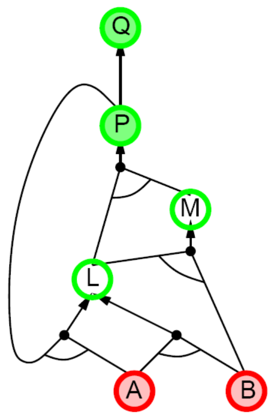
Backward Chaining



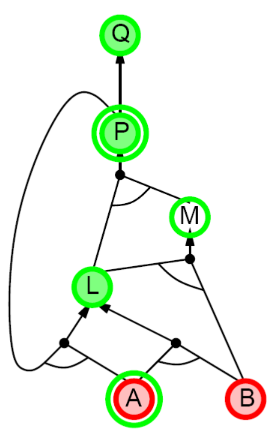
Backward Chaining



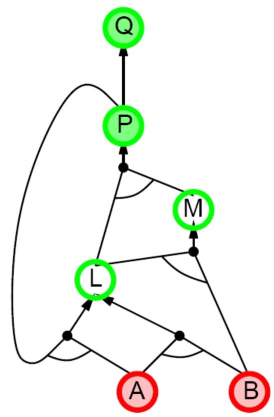
Backward Chaining



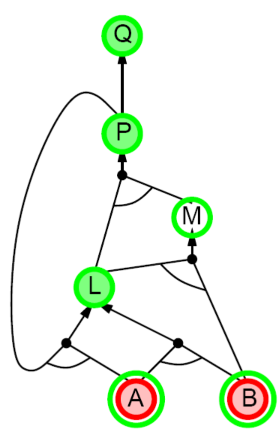
Backward Chaining



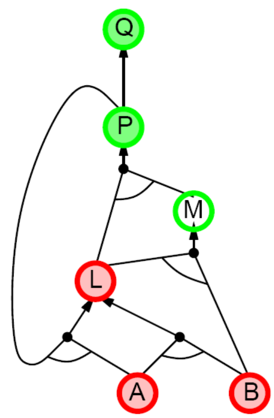
Backward Chaining



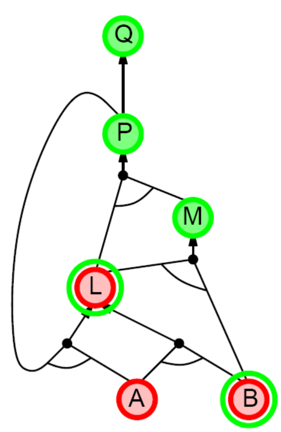
Backward Chaining



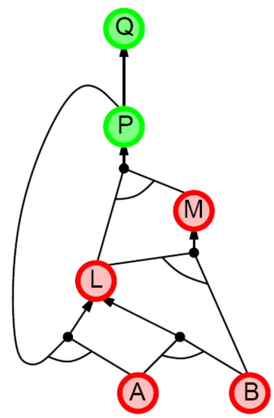
Backward Chaining



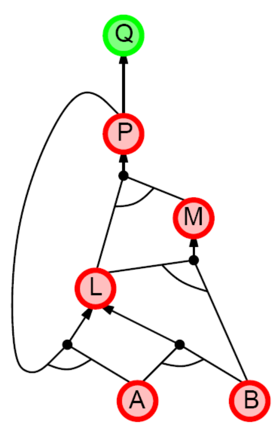
Backward Chaining



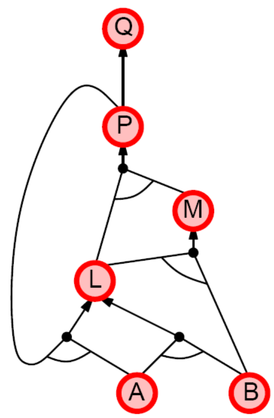
Backward Chaining



Backward Chaining



Backward Chaining



Forward Chaining vs. Backward Chaining

- Forward Chaining is data driven
  - Automatic, unconscious processing
    - object recognition
    - routine decisions
  - May do lots of work that is irrelevant to the goal
- Backward Chaining is goal driven
  - Appropriate for problem solving
  - E.g. "Is this sickness Malaria?", "Where are my keys?", "How do I start the car?"
- The complexity of BC can be much less than linear in size of the KB