LECTURE NOTES– MODULE III

KNOWLEDGE REPRESENTATION – LOGICAL AGENTS

ECS302

ARTIFICIAL INTELLIGENCE

3/4 B.Tech B3,B12

PRANEEL ANKIREDDY
ASSISTANT PROFESSOR
DEPARTMENT OF CSE,GIT,GU

Module III Lecture Notes – PART- I
[*Knowledge And Reasoning*]

Syllabus

***Knowledge Representation*:** Logical Agents: Knowledge based agents, the wumpus world, and logic.

Propositional Logic: A very simple logic, reasoning patterns in propositional logic.

.

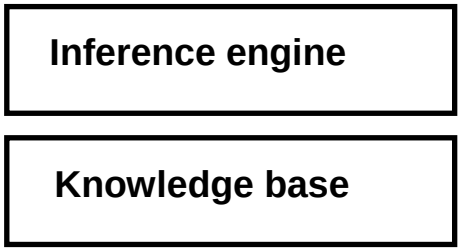
Concepts

- Knowledge-based agents
- Example: The wumpus world
- Logic in general—models and entailment
- Propositional (Boolean) logic
- Equivalence, validity, satisfiability
- Inference rules and theorem proving
  - forward chaining
  - backward chaining
  - resolution
- First Order Logic

Knowledge Based Agents

A **knowledge-based agent** needs a KB and an inference mechanism. It operates by storing sentences in its **knowledge** base, inferring new sentences with the inference mechanism, and using them to deduce which actions to take. ... The interpretation of a sentence is the fact to which it refers.

Knowledge Bases:



domain−independent algorithms

domain−specific content

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

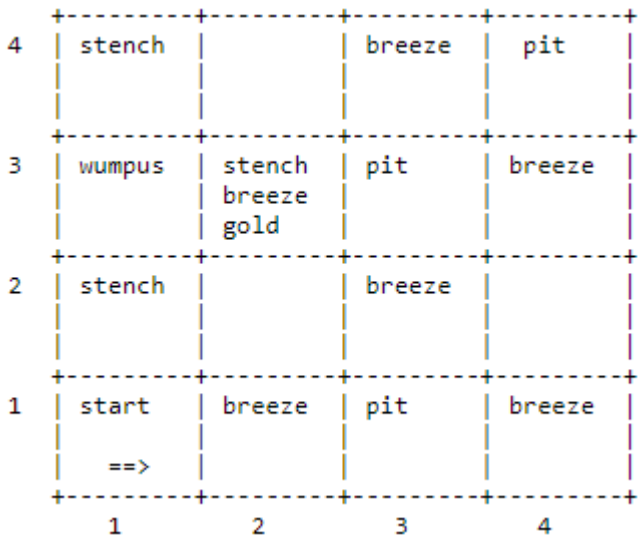
Then it can Ask itself what to do—answers should follow from the KB Agents can be viewed at the knowledge level i.e., what they know, regardless of how implemented or at the implementation level i.e.,data structures in KB and algorithms that manipulate them.

The Wumpus World:

A variety of "worlds" are being used as examples for Knowledge Representation, Reasoning, and Planning. Among them the Vacuum World, the Block World, and the Wumpus World.

The Wumpus World was introduced by Genesereth, and is discussed in Russell-Norvig. The Wumpus World is a simple world (as is the Block World) for which to represent knowledge and to reason.

It is a cave with a number of rooms, represented as a 4x4 square.



Rules of the Wumpus World

The **neighborhood** of a node consists of the four squares north, south, east, west of the given square. In a square the agent gets a vector of percepts, with components

Stench, Breeze, Glitter, Bump, Scream

For example [Stench, None, Glitter, None, None]

- Stench is perceived at a square iff the wumpus is at this square or in its neighborhood.
- Breeze is perceived at a square iff a pit is in the neighborhood of this square.
- Glitter is perceived at a square iff gold is in this square
- Bump is perceived at a square iff the agent goes Forward into a wall
- Scream is perceived at a square iff the wumpus is killed anywhere in the cave

An agent can do the following actions (one at a time):

Turn(Right), Turn(Left), Forward, Shoot, Grab, Release, Climb

- The agent can go Forward in the direction it is currently facing, or Turn Right, or Turn Left. Going Forward into a wall will generate a Bump percept.
- The agent has a single arrow that it can Shoot. It will go straight in the direction faced by the agent until it hits (and kills) the wumpus, or hits (and is absorbed by) a wall.
- The agent can Grab a portable object at the current square or it can Release an object that it is holding.
- The agent can Climb out of the cave if at the Start square. The Start square is (1,1) and initially the agent is facing east. The agent dies if it is in the same square as the wumpus. The objective of the game is to kill the wumpus, to pick up the gold, and to climb out with it.

Representing our Knowledge about the Wumpus World

Percept(x,y)

where x must be a percept vector and y must be a situation. It means that at situation y the agent perceives x. For convenience we introduce the following definitions:

- Percept([Stench,y,z,w,v],t) = > Stench(t)
- Percept([x,Breeze,z,w,v],t) = > Breeze(t)
- Percept([x,y,Glitter,w,v],t) = > AtGold(t)

Holding(x,y)

where x is an object and y is a situation. It means that the agent is holding the object x in situation y.

Action(x,y)

where x must be an action (i.e. Turn(Right), Turn(Left), Forward, ..) and y must be a situation. It means that at situation y the agent takes action x.

At(x,y,z)

where x is an object, y is a Location, i.e. a pair [u,v] with u and v in {1,2,3,4}, and z is a situation. It means that the agent x in situation z is at location y.

Present(x,s)

means that object x is in the current room in the situation s.

Result(x,y)

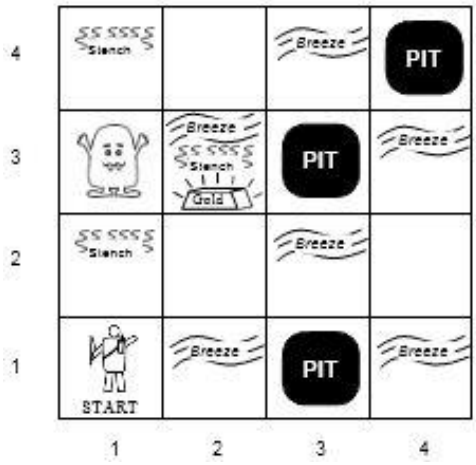
It means that the result of applying action x to the situation y is the situation Result(x,y). Notethat Result(x,y) is a term, not a statement. For example we can say

- Result(Forward, S0) = S1
- Result(Turn(Right),S1) = S2

These definitions could be made more general. Since in the Wumpus World there is a single agent, there is no reason for us to make predicates and functions relative to a specific agent. In other "worlds" we should change things appropriately.

Wumpus World : PEAS Description

Performance measure
Gold +1000, death−1000
−1perstep,−10forusingthearrow
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
Actuators
Left turn, Right turn, Forward, Grab, Release, Shoot
Sensors
Breeze, Glitter, Smell



Wumpus World Characterization

Observable: NO – Only Local Perception

Deterministic: Yes – Outcomes Exactly Specified

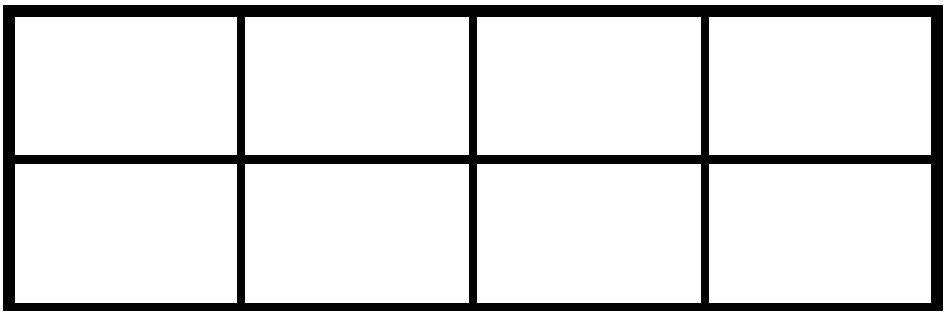
Episodic: No – Sequential at the level of actions

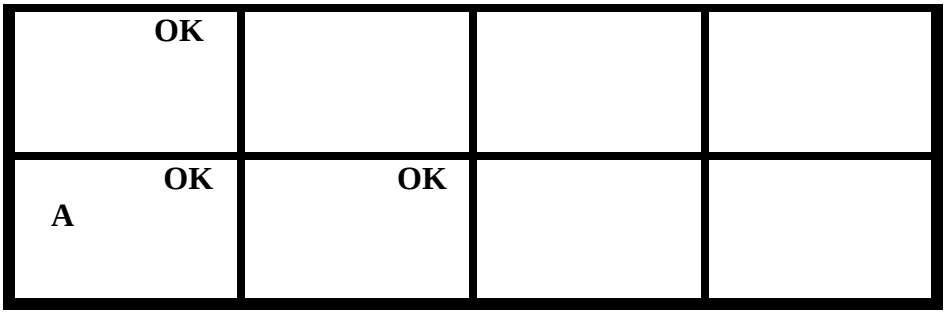
Static: No – Wumpus and Pits Do not Move

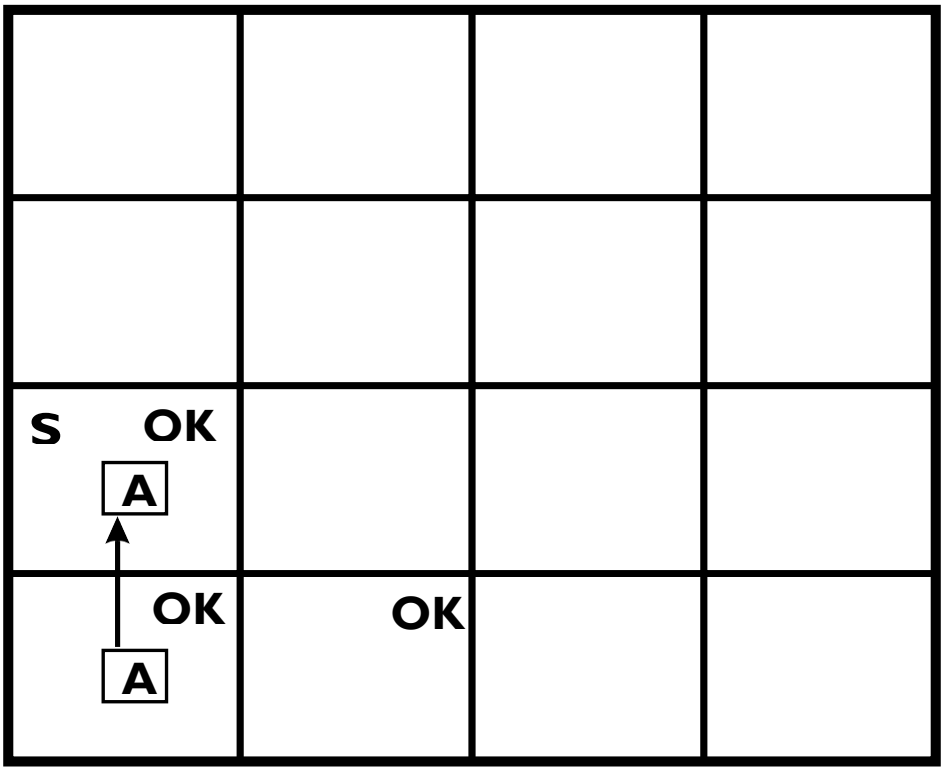
Discrete: Yes

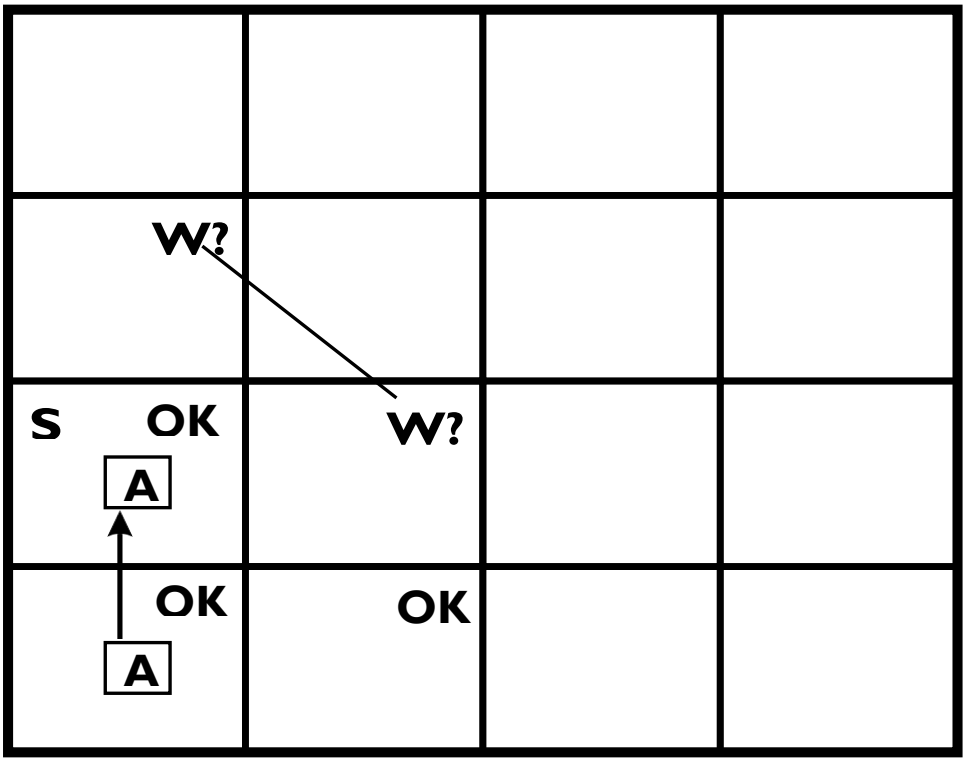
Single agent: Yes – Wumpus is essentially a Natural Feature

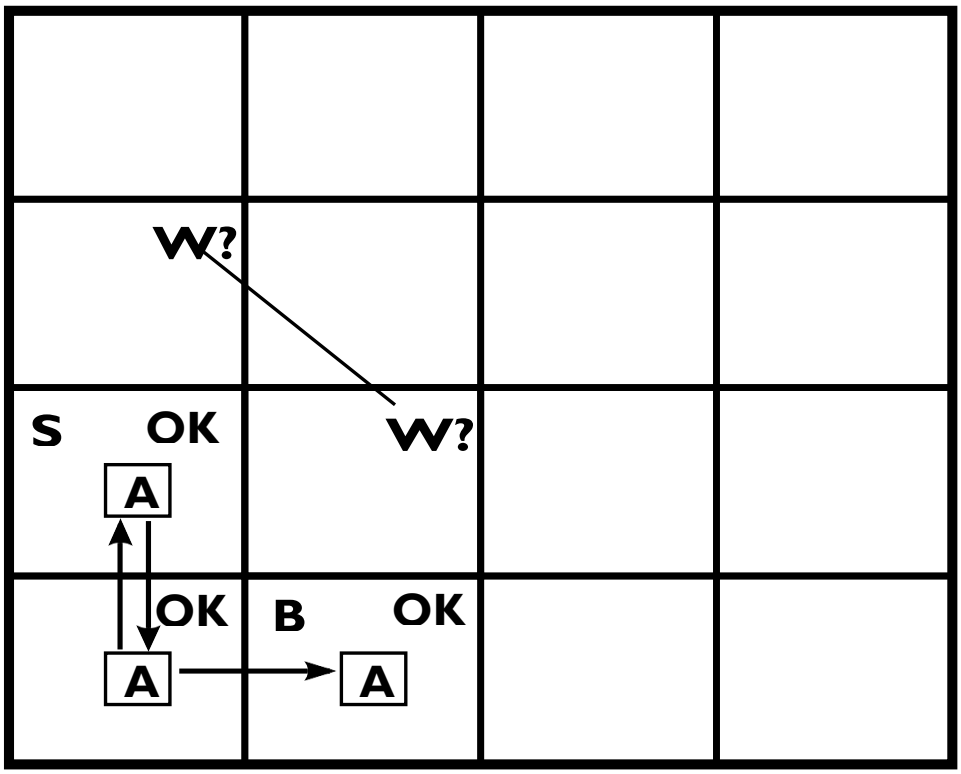
Exploring a Wumpus World

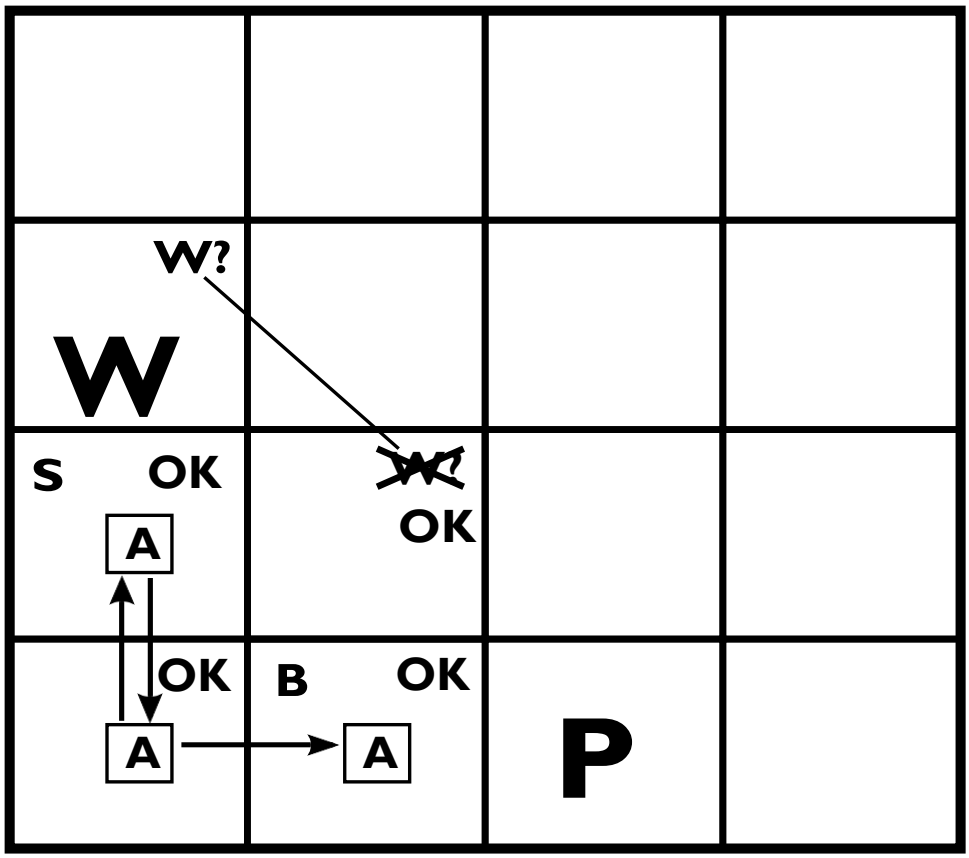


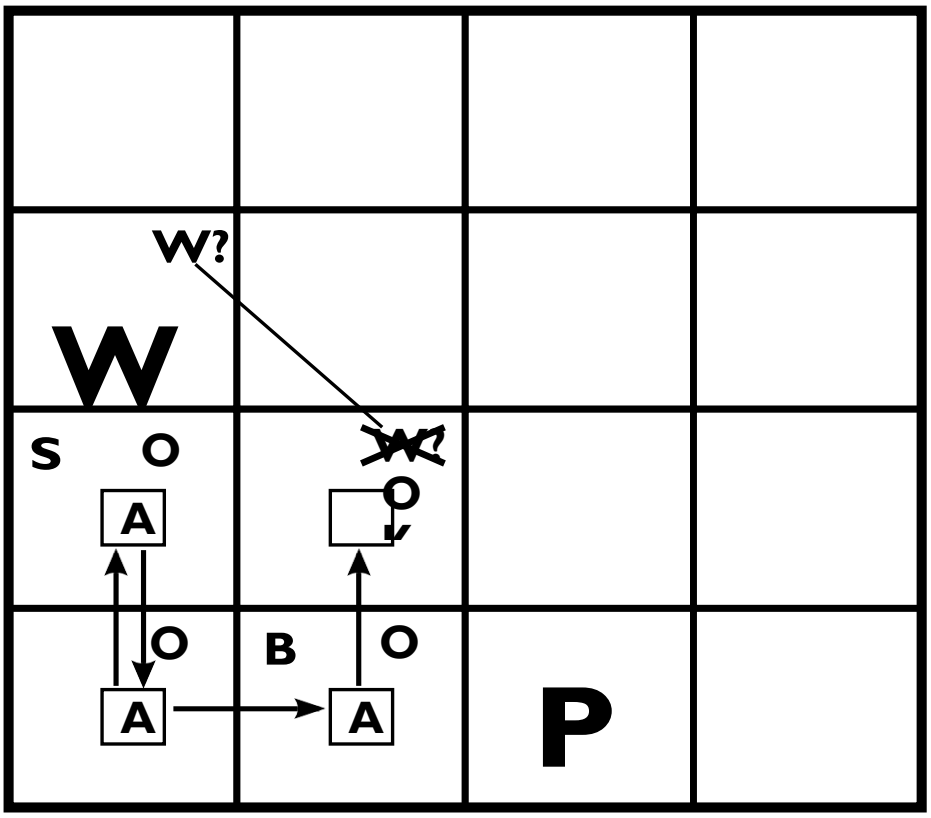

OK
OK
OK
A

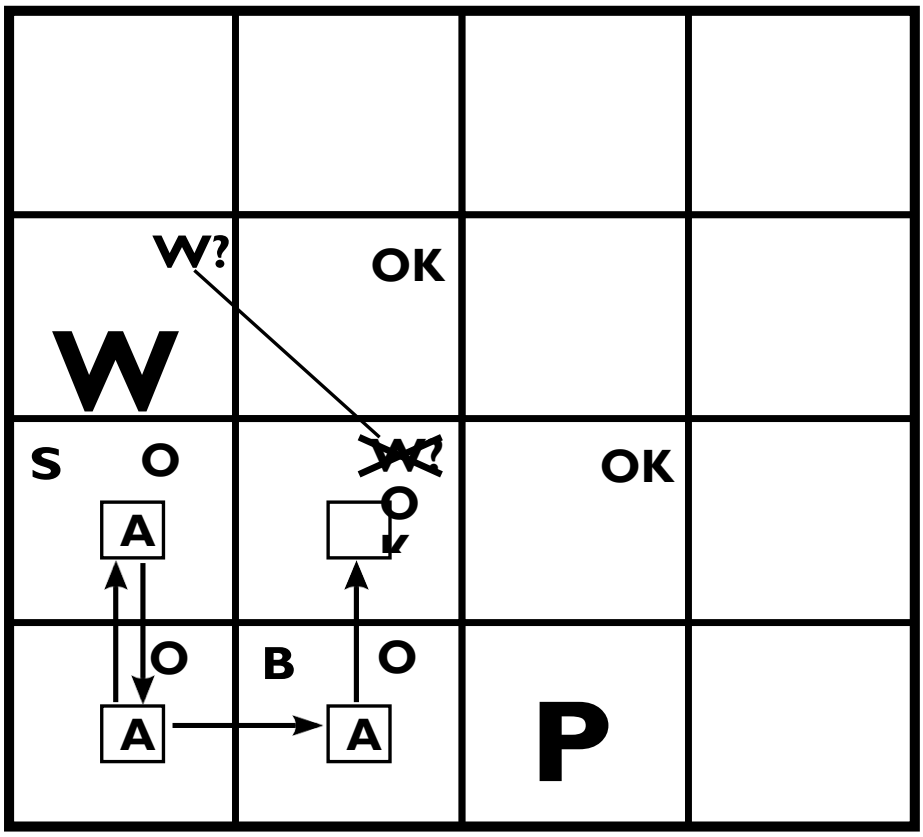

S
OK
A
OK
OK
A

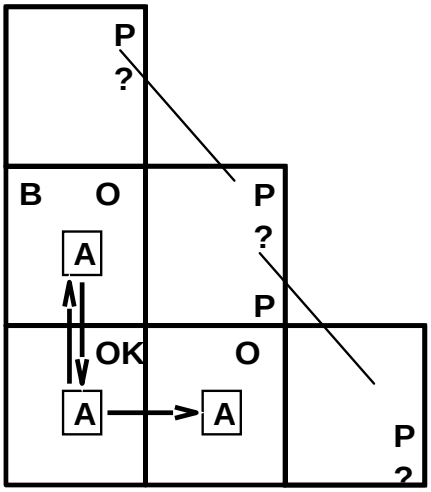

W?
S
OK
W?
A
OK
OK
A


W?
S
OK
W?
A
OK
B
OK
A
A


W?
W
S
OK
W?
OK
A
OK
B
OK
A
A
P


W?
W
S
O
A
O
A
B
O
A
P





Breeze in (1,2) and(2,1)
⇒ no safe actions
Assumingpitsuniformlydistributed,
(2,2) has pit w/ prob 0.86, vs. 0.31

Smell in (1,1)
⇒ cannot move
Can use a strategy ofcoercion:

S

A

shoot straight ahead
wumpuswasthere ⇒dead⇒safe
wumpuswasn't there ⇒ safe

Logic in general

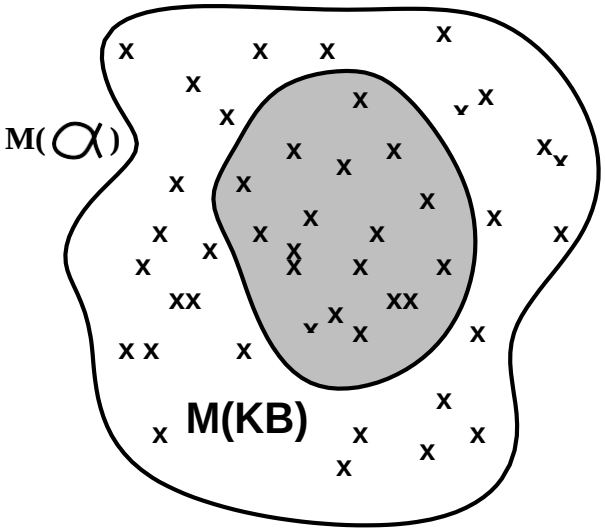
Logics are formal languages for representing information such that conclusions can be drawn
Syntaxdefinesthesentencesinthelanguage
Semantics define the "meaning" ofsentences;
i.e., define truth of a sentence in a world
E.g., the language of arithmetic
$x+2\geq y$ is a sentence; $x2+y >$ is not a sentence
$x+2\geq y$ istrueiffthenumber$x+2$ isnolessthanthenumber$y$ $x+2\geq y$ 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 fromanother:
$KB \models \alpha$
A knowledge base KB entails a sentenceα if and only if α is true in all worlds where KB is true.
E.g.,theKBcontaining"There'sapitahead"and"There'sgoldtotheleft" entails "Either there's a pit ahead or gold to the left"
E.g., $x+y=4$ entails $4=x+y$
Entailment is a relationship between sentences (i.e.,syntax) that is based on semantics

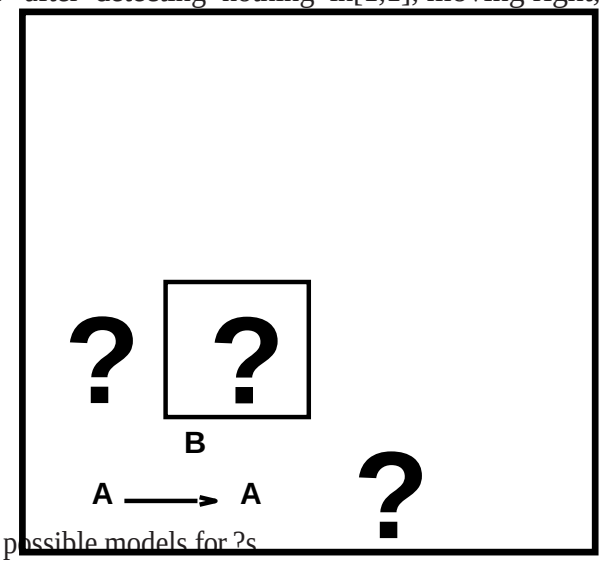
Models

Logicians typically think in terms of models, whichareformally 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 of α
Then $KB \models \alpha$ if and only if $M(KB) \subseteq M(\alpha)$
E.g. KB = {there's a pit ahead, there's gold to the left }
α = there's gold to the left



Entailment in the Wumpus World

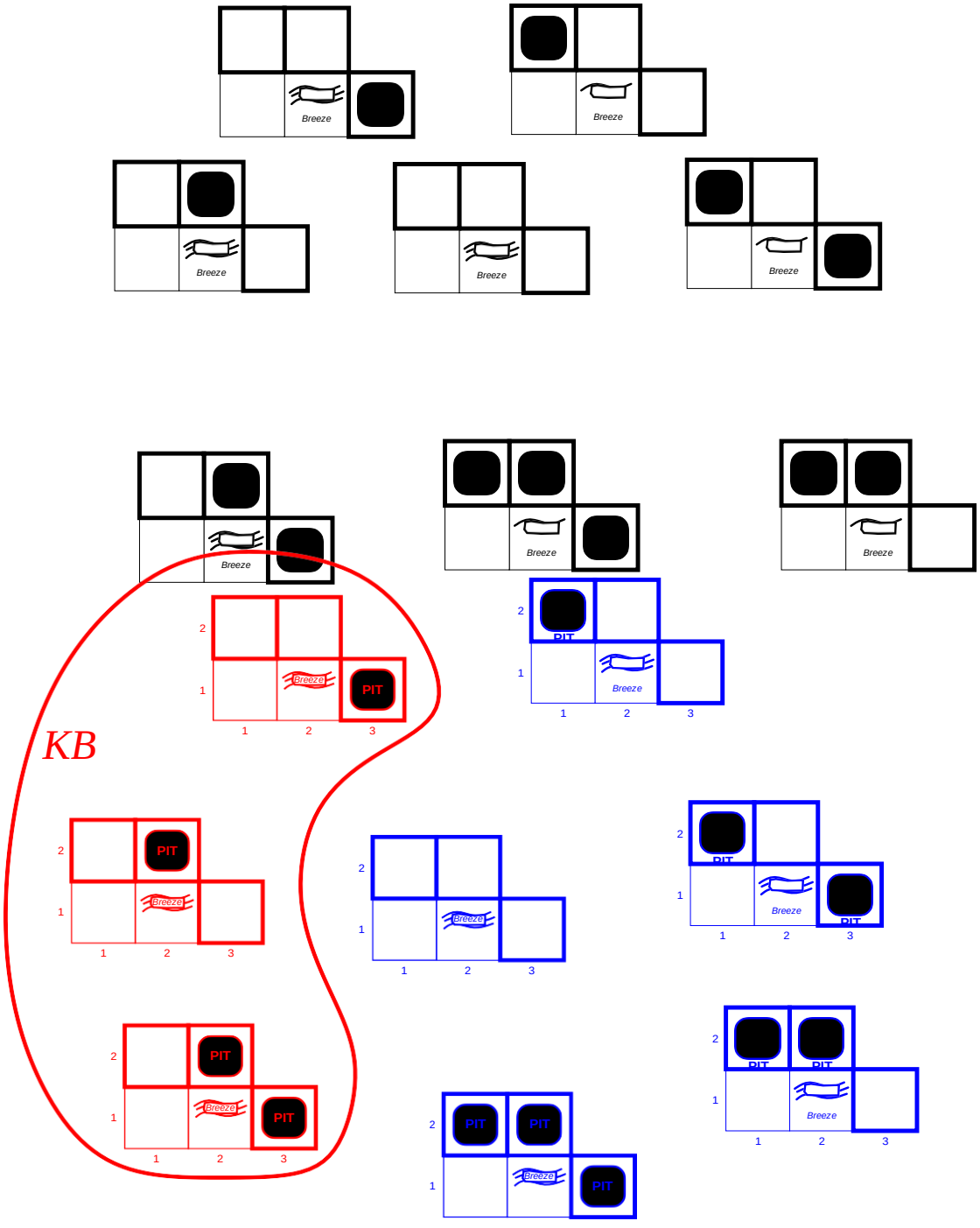
Situation after detecting nothing in[1,1], moving right, breeze in [2,1]



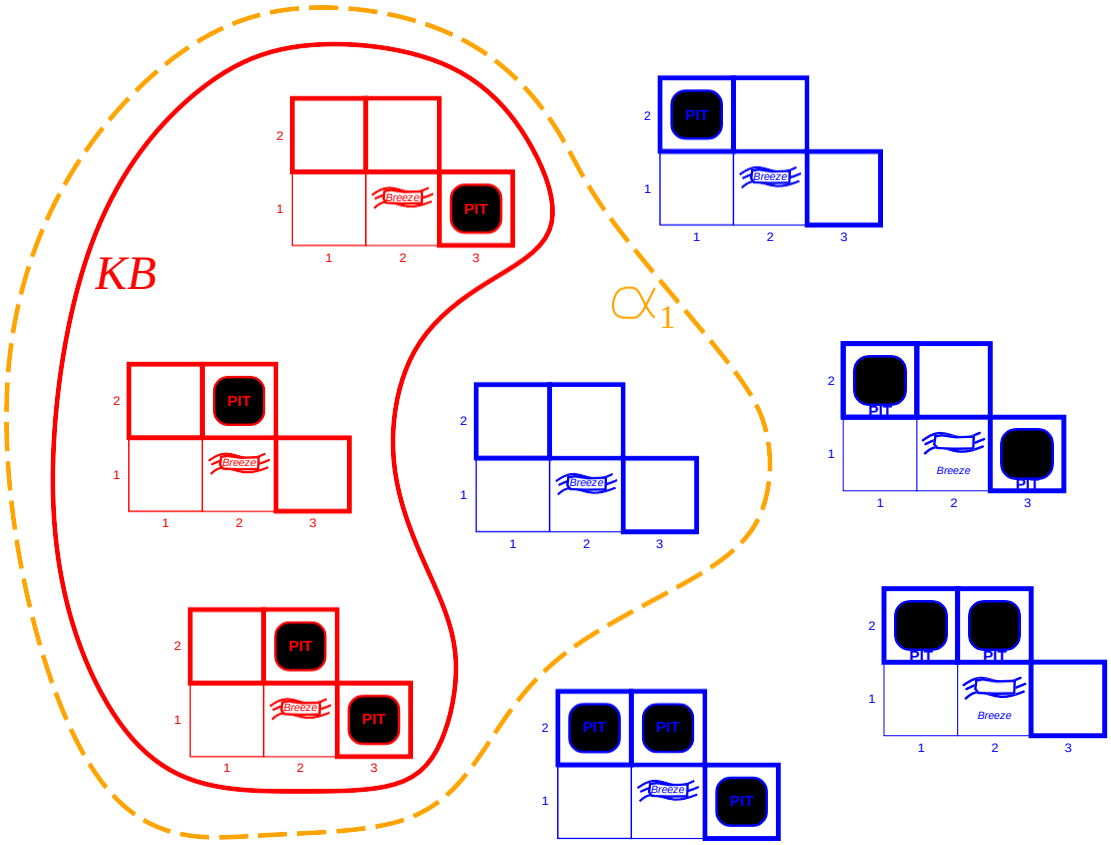
Consider possible models for ?s assuming only pits

3Booleanchoices ⇒ 8 possible models

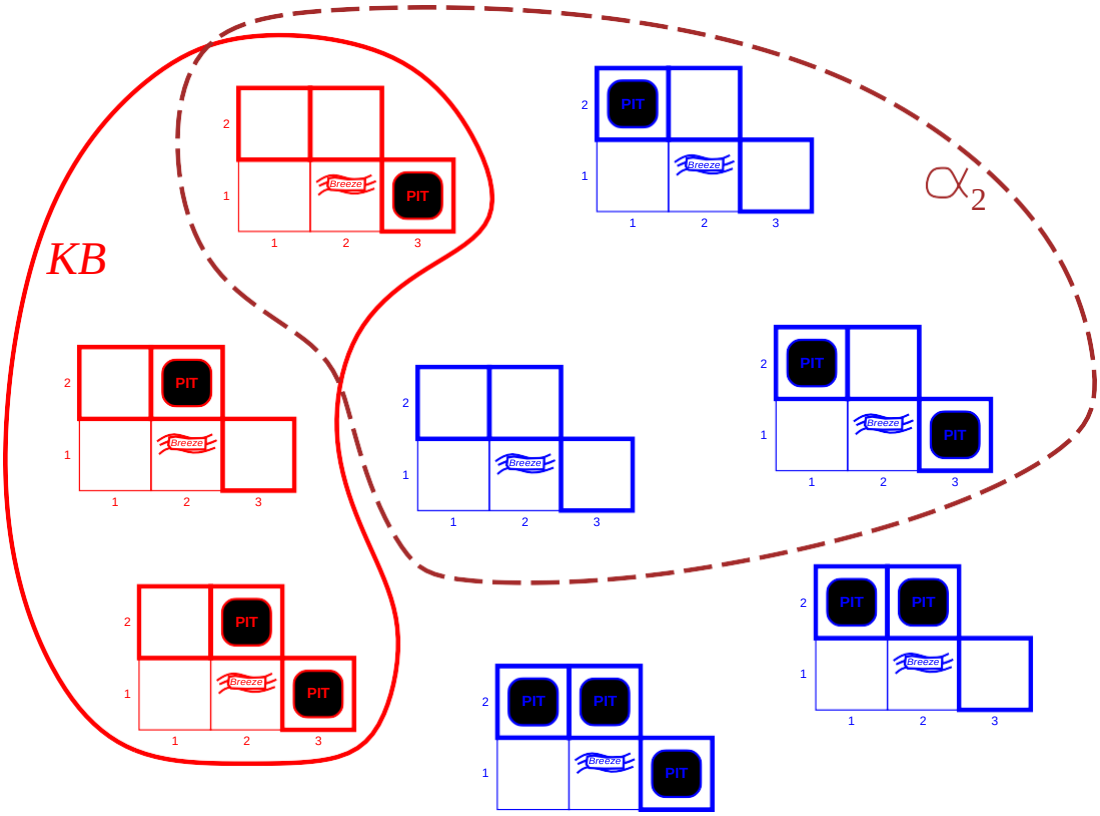
Wumpus Models



KB = wumpus-world rules + observations



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



KB = wumpus-world rules + observations
α_2 = "[2,2] is safe", $KB \models \alpha_2$

Inference

$KB \vdash_i \alpha$ = sentence α can be derived from KB by procedure i
Consequences of KB are a hay stack; α is a needle.
Entailment=needle in hay stack;
inference=finding it
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:wewilldefine a logic(first-orderlogic)whichisexpressiveenough tosayalmostanything of interest, and forwhichthereexists a soundand complete inference procedure.
Thatis, theprocedurewill answer anyquestionwhoseanswerfollowsfrom 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

Each model specifies true/false for each proposition symbol

E.g.

$P_{1,2}$	$P_{2,2}$	$P_{3,1}$
TRUE	TRUE	FALSE

(With these symbols, 8 possible models, can be enumerated automatically.) Rules for evaluating truth with respect to a model m:

$\neg S$	is true iff	S	is false			
$S_1 \wedge S_2$	is true iff	S_1	is true and	S_2	is true	
$S_1 \vee S_2$	is true iff	S_1	is true or	S_2	is true	
$S_1 \Rightarrow S_2$	is true iff	S_1	is false or	S_2	is true	
i.e.,	is false iff	S_1	is true and	S_2	is false	
$S_1 \Leftrightarrow S_2$	is true iff	$S_1 \Rightarrow S_2$	is true and	$S_2 \Rightarrow S_1$	is true	

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 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].

$\neg P_{1,1}$

$\neg B_{1,1}$

$B_{2,1}$

How do we encode "pits cause breezes in adjacent squares"?

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

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

In other words, "a square is breezy if and only if there is an adjacent pit"

Truth Tables for Inference

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	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*

Enumerate rows (different assignments to symbols), if KB is true in row, check that α is too

Inference by Enumeration

Depth-first enumeration of all models is sound and complete

```
function TT-Entails?(KB, α) returns true or false
  inputs: KB, the knowledge base, a sentence in propositional logic
          α, the query, a sentence in propositional logic
  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))
```

$O(2^n)$ for n symbols; problem is co-NP-complete

Logical Equivalence

Two sentences are logically equivalent iff they are true in the same models:

$\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$

$(\alpha \wedge \beta)$	$\equiv$	$(\beta \wedge \alpha)$	commutativity of $\wedge$
$(\alpha \vee \beta)$	$\equiv$	$(\beta \vee \alpha)$	commutativity of $\vee$
$((\alpha \wedge \beta) \wedge \gamma)$	$\equiv$	$(\alpha \wedge (\beta \wedge \gamma))$	associativity of $\wedge$
$((\alpha \vee \beta) \vee \gamma)$	$\equiv$	$(\alpha \vee (\beta \vee \gamma))$	associativity of $\vee$
$\neg(\neg\alpha)$	$\equiv$	α	double-negation elimination
$(\alpha \Rightarrow \beta)$	$\equiv$	$(\neg\beta \Rightarrow \neg\alpha)$	Contraposition
$(\alpha \Rightarrow \beta)$	$\equiv$	$(\neg\alpha \vee \beta)$	implication elimination
$(\alpha \Leftrightarrow \beta)$	$\equiv$	$((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$	Bi conditional elimination
$\neg(\alpha \wedge \beta)$	$\equiv$	$(\neg\alpha \vee \neg\beta)$	De Morgan

¬(α∨β)	≡	(¬α ∧¬β)	De Morgan		
(α ∧(β ∨γ))	≡	((α ∧β)∨(α ∧γ))	distributivityof	∧	over ∨
(α ∨(β ∧γ))	≡	((α ∨β)∧(α ∨γ))	distributivityof	∨	over ∧

Validity And Satisfiability

A sentence is valid 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:

$KB \models \alpha$ if and only if $(KB \Rightarrow \alpha)$ is valid

A sentence is satisfiable if it is true in some model e.g., $A \vee B$, C

A sentence is un satisfiable 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 un satisfiable i.e., prove α by *reductio ad absurdum*

Proof Methods

Proof methods divide into (roughly)two kinds:

Application of inference rules

- Legitimate(sound)generationofnewsentencesfromold
- Proof = a sequence of inference rule applications canuseinferencerules as operators in a standard search algorithm
- Typically require translation of sentences into a normalform

Model checking

- Truth table enumeration (always exponential in n)
- Improvedbacktracking,e.g.,Davis–Putnam–Loge Mann–Loveland
- Heuristic search inmodelspace(soundbutincomplete) e.g.,min-conflicts-likehill-climbing algorithms

Forward and Backward Chaining

Horn Form (restricted)

KB = conjunction of Horn clauses

Horn clause =

- proposition symbol; or

– (conjunctionofsymbols) ⇒symbol Example KB: C∧(B ⇒A) ∧ (C∧D ⇒B)

Modus Ponens (for Horn Form): complete for Horn KBs

$\alpha_1, \ldots, \alpha_n, \quad \alpha_1 \wedge \cdots \wedge \alpha_n \Rightarrow \beta \beta$

Can be used with forwardchaining or backwardchaining.

These algorithms are very natural and run in linear time

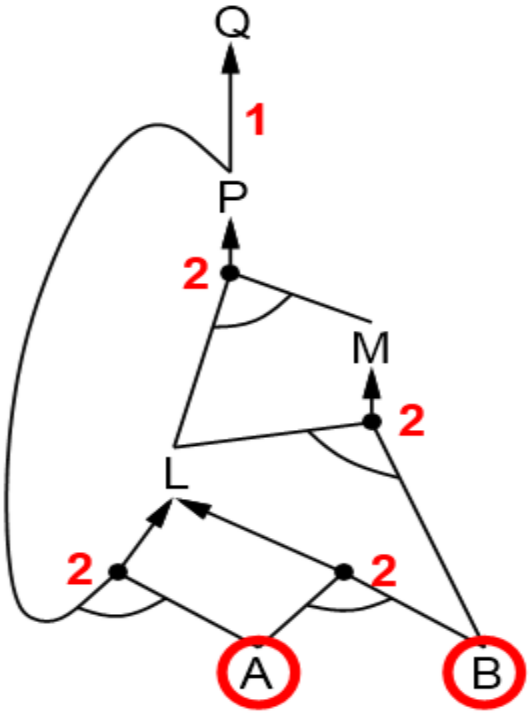
ForwardChaining

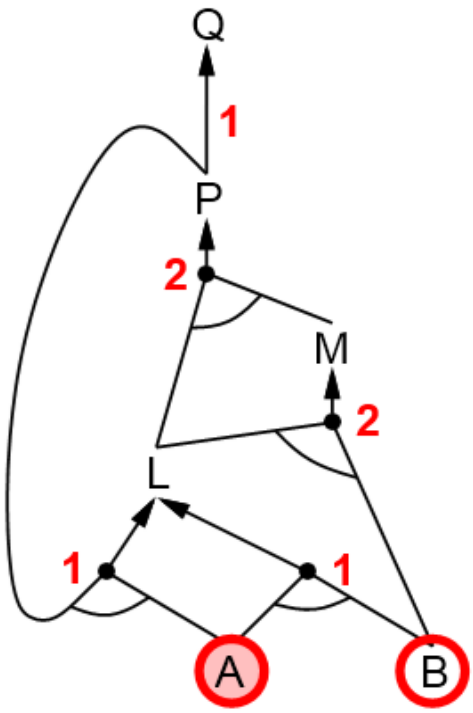
Idea: fireany rule whose premisesaresatisfiedinthe*KB*, additsconclusiontothe*KB*,untilqueryisfound

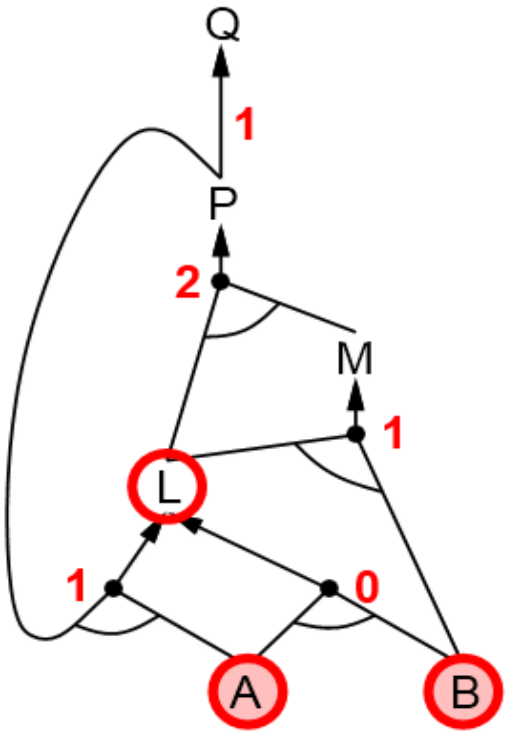
ForwardChaining Algorithm

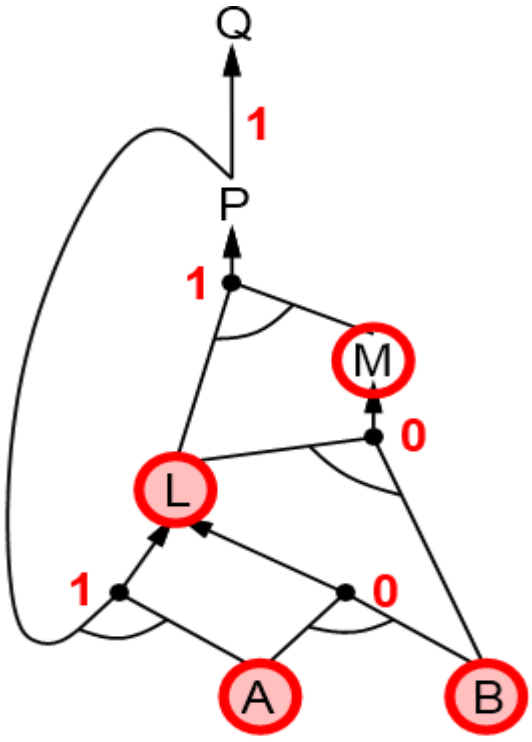
```
function PL-FC-ENTAILS?(KB, q) returns true or false
  inputs: KB, the knowledge base, a set of propositional Horn clauses
          q, the query, a proposition symbol
  local variables: count, a table, indexed by clause, initially the number of premises
                   inferred, a table, indexed by symbol, each entry initially false
                   agenda, a list of symbols, initially the symbols known to be true in KB

  while agenda is not empty do
      p ← POP(agenda)
      if p = q then return true
      unless inferred[p] do
          inferred[p] ← true
          for each Horn clause c in whose premise p appears do
              decrement count[c]
              if count[c] = 0 then
                  PUSH(HEAD[c], agenda)
  return false
```

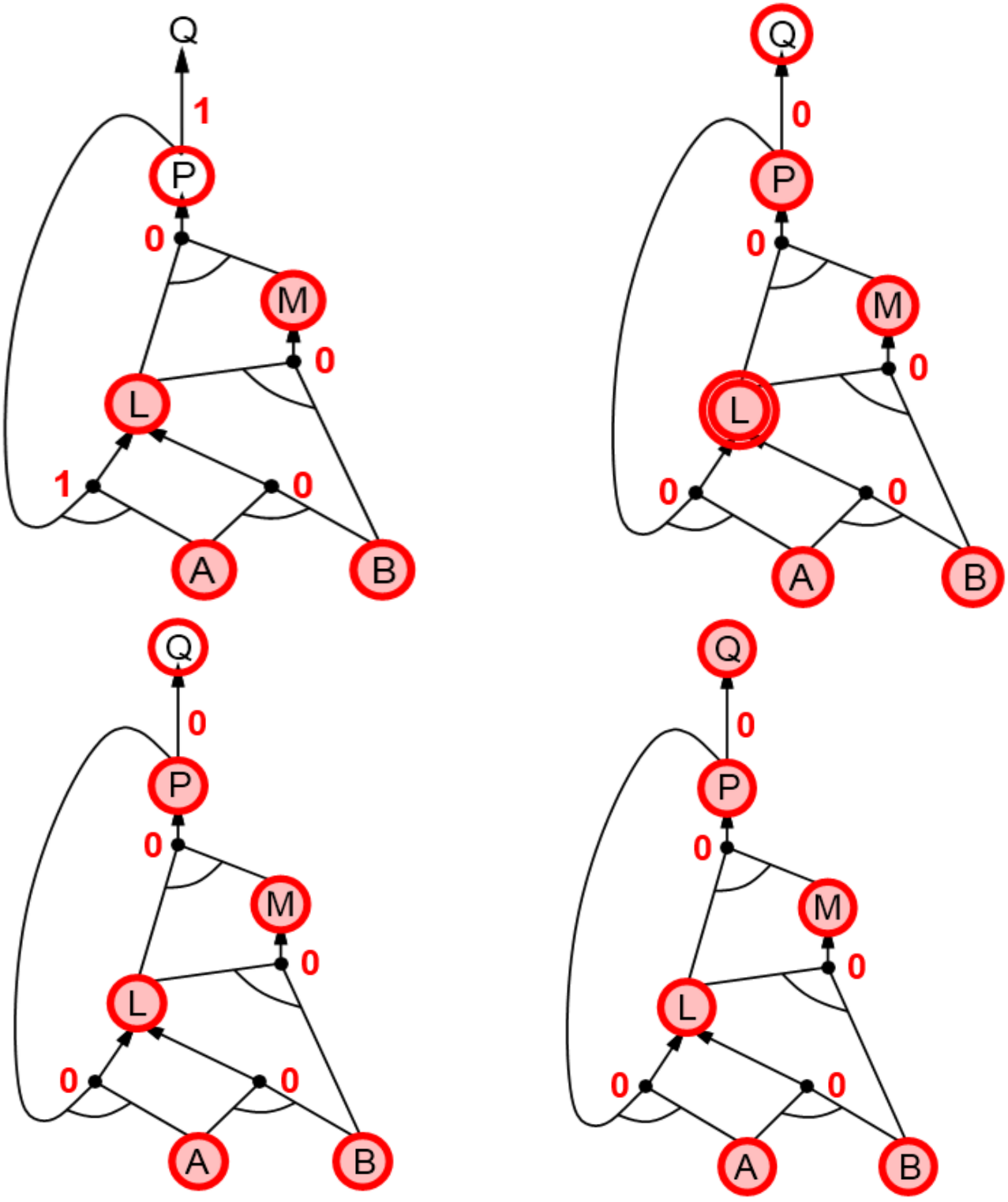
ForwardChaining Example


Q
1
P
2
M
2
L
2
2
A
B


Q
1
P
2
M
2
L
1
1
A
B


Q
1
P
2
M
1
L
1
0
A
B


Q
1
P
1
M
0
L
1
0
A
B


Q
1
P
0
M
0
L
1
0
A
B
Q
0
P
0
M
0
L
0
0
A
B
Q
0
P
0
M
0
L
0
0
A
B
Q
0
P
0
M
0
L
0
0
A
B

Proof of Completeness

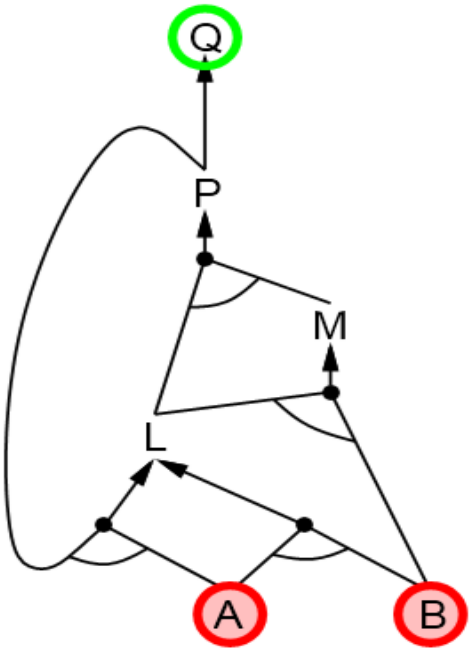
FC derives every atomic sentence that is entailed by *KB*

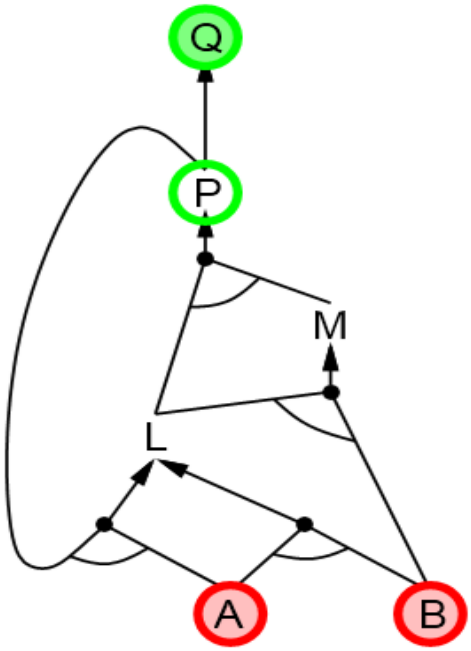
1. FC reaches a fixed point where no new atomic sentences are derived
2. Consider the final state as a model *m*, assigning true/false to symbols
3. Every clause in the original *KB* is true in *m*
   i. Proof: Suppose a clause $a_1 \wedge \ldots \wedge a_k \Rightarrow b$ is false in *m* Then $a_1 \wedge \ldots \wedge a_k$ 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 $KB \models q$, then *q* is true in every model of *KB*, including *m*
   a. General idea: construct any model of *KB* by sound inference, check α

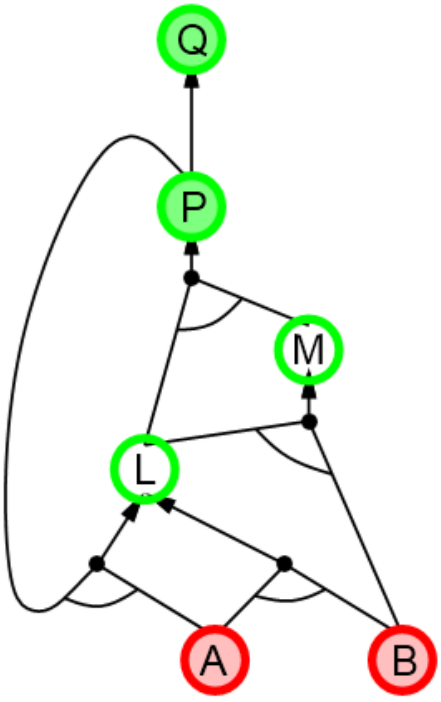
Backward Chaining

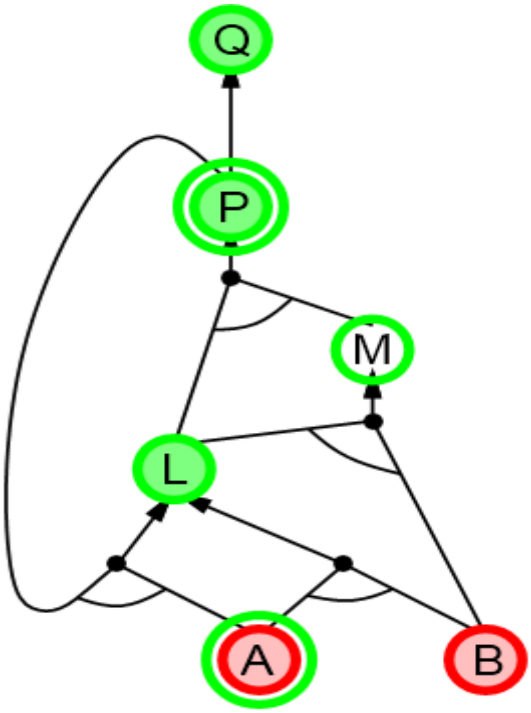
Idea: work backwards from the query *q*: to prove *q* by BC,
check if *q* is known already, or
prove by BC all premises of some rule concluding *q*
Avoid loops: check if new sub goal is already on the goal stack Avoid repeated work: check if new sub goal

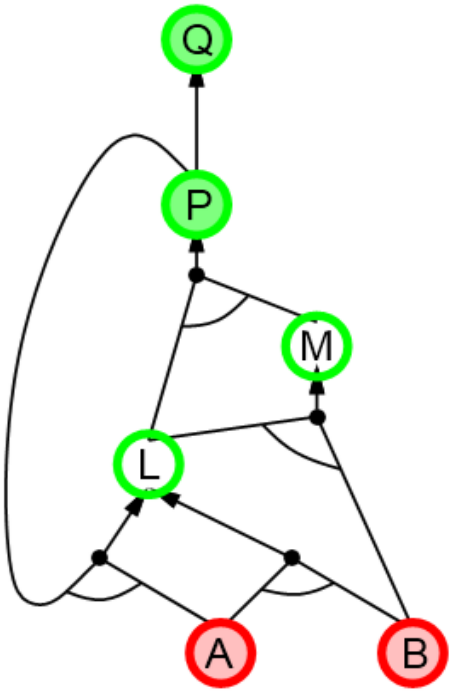
1. has already been proved true, or
2. has already failed

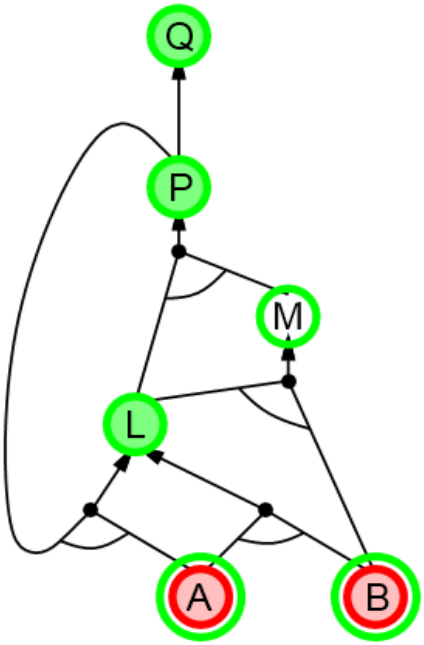

Q
P
M
L
A
B

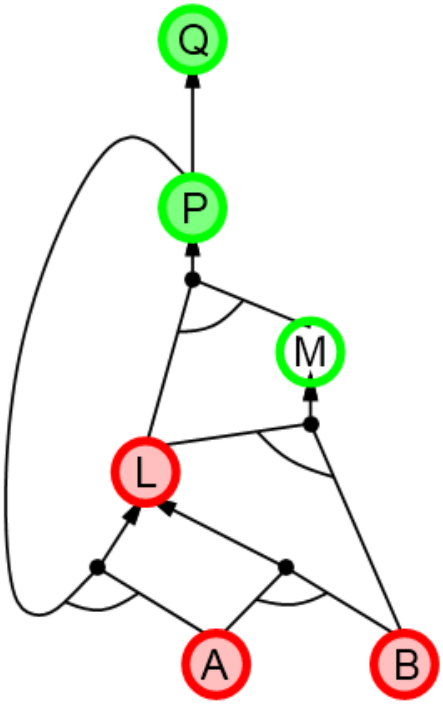

Q
P
M
L
A
B

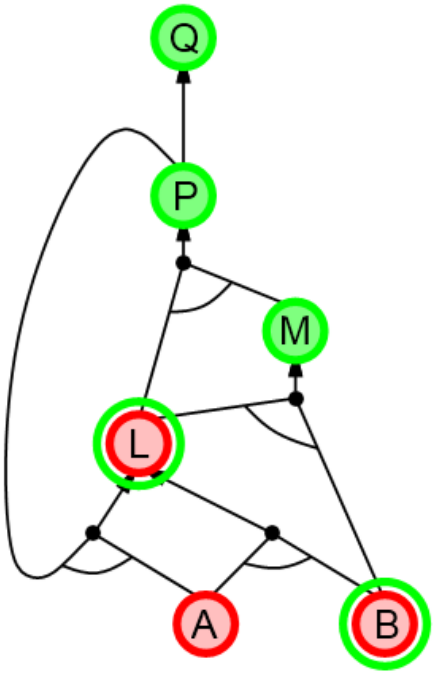

Q
P
M
L
A
B

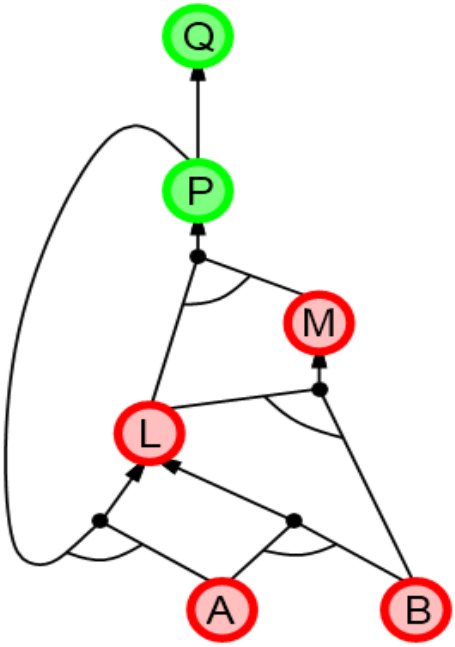

Q
P
M
L
A
B

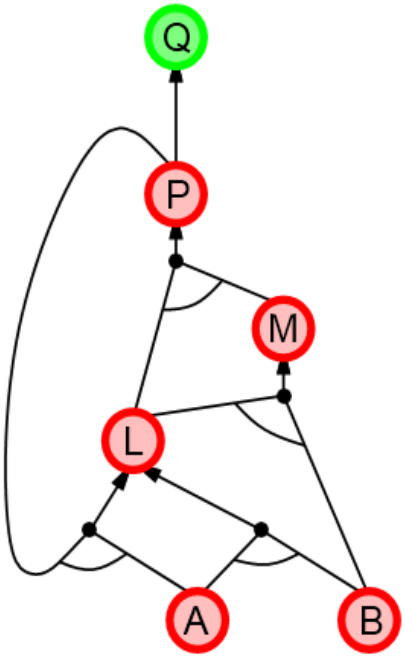

Q
P
M
L
A
B

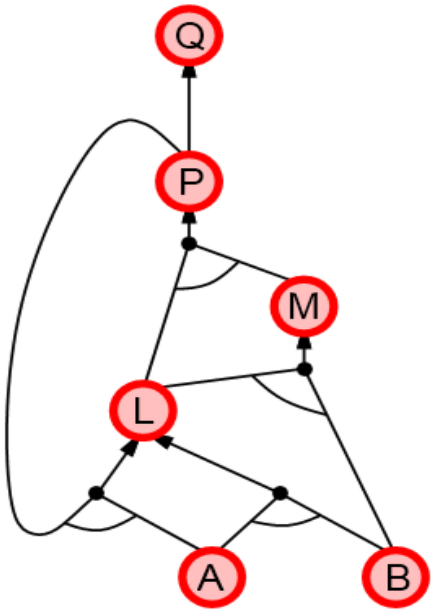

Q
P
M
L
A
B


Q
P
M
L
A
B


Q
P
M
L
A
B


Q
P
M
L
A
B


Q
P
M
L
A
B



Forward vs Backward Chaining

FC is data-driven, cf. 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

Resolution

Conjunctive Normal Form (CNF—universal)

conjunction of disjunctions ofliterals

s_____,_____x

clauses

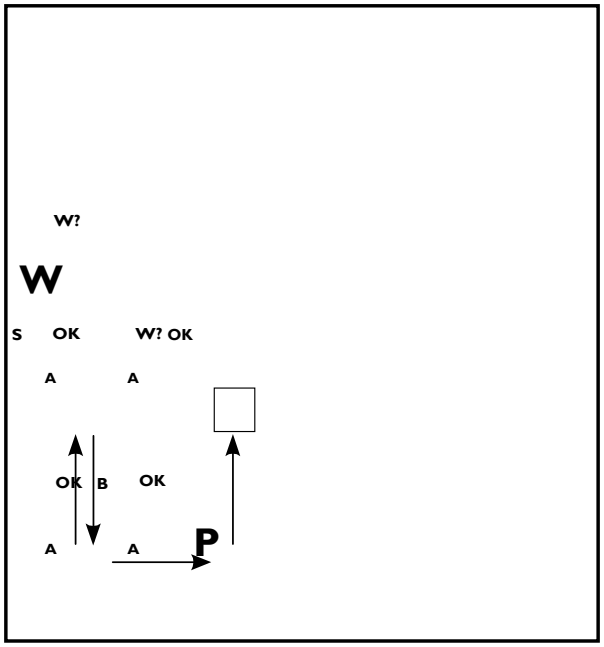
E.g., $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

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

$$\frac{\ell_1 \vee \cdots \vee \ell_i \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_j \vee \cdots \vee m_n}{\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n}$$

where $\ell_i \equiv \neg m_j$ are complementary literals. E.g.,

$$\frac{W_{1,3} \vee W_{2,2}, \quad \neg W_{2,2}}{W_{1,3}}$$



Resolution is sound and complete for propositional logic

Conversion to CNF

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

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

Resolution Algorithm

Proof by contradiction, i.e., show $KB \wedge \neg\alpha$ un satisfiable

function PL-Resolution(KB,α) returns *true* or *false*

inputs: KB, the knowledge base, a sentence in propositional logic

α, the query, a sentence in propositional logic

clauses ←the set of clauses in the CNF representation of $KB \wedge \neg\alpha$

new ←{}

loop do

for each C_i, C_j in *clauses* do

resolvents ←PL-Resolve(C_i,C_j)

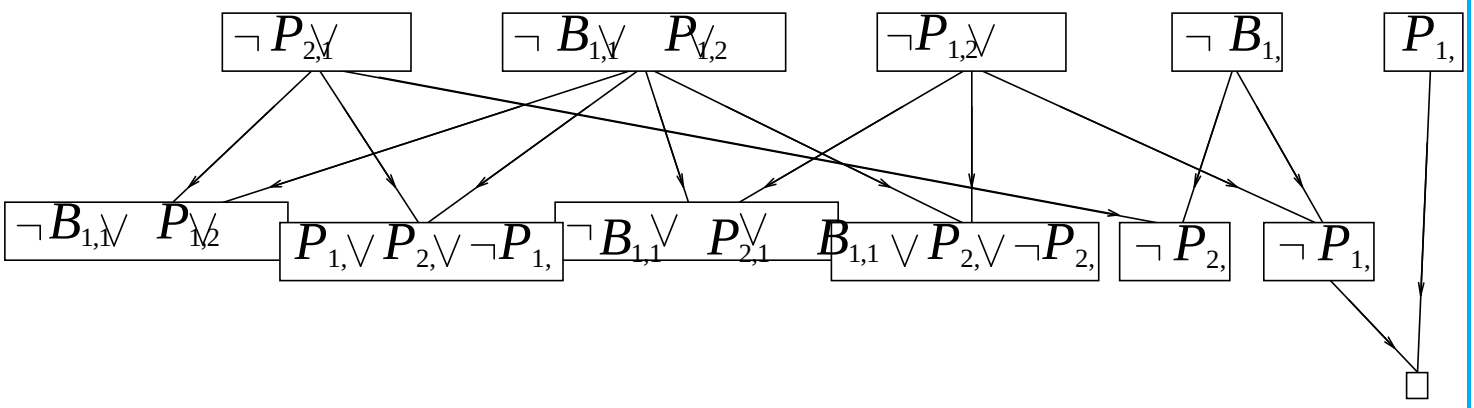
if *resolvents* contains the empty clause then return *true*

new ← *new* ∪ *resolvents*

if *new* ⊆ *clauses* then return *false*
clauses ← *clauses* ∪ *new*

Resolution Example

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



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
- Propositional logic lacks expressive power

Pros and Cons of Propositional Logic

- Propositional logic is declarative: pieces of syntax correspond to facts.
- Propositional logic allows partial/disjunctive/negated information (unlike most data

structures and databases)

- Propositional logic is compositional: meaning of B1,1 ∧ P1,2 is derived from meaning of B1,1 and of P1,2
- Meaning in propositional logic is context-independent.(unlike natural language, where meaning depends on context)
- Propositional logic has very limited expressive power (unlike natural language)

  E.g., cannot say "pits cause breezes in adjacent squares"
  except by writing one sentence for each square.