

# CS 161 Fundamentals of Artificial Intelligence

## Lecture 8

### Logical agents

Quanquan Gu

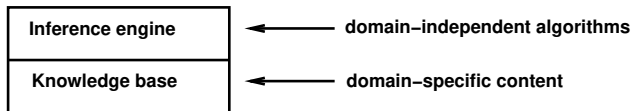
Department of Computer Science  
UCLA

Feb 14, 2023

# Outline

- Knowledge-based agents
- Wumpus world
- Logic in general—models and entailment
- Propositional (Boolean) logic
- Equivalence, validity, satisfiability
- Inference rules and theorem proving
  - forward chaining
  - resolution

# Knowledge bases



- **Knowledge base** = set of **sentences** in a **formal** language
- **Declarative** approach to building an agent (or other system):

**TELL** it what it needs to know

Then it can **ASK** itself what to do—answers should follow from the KB

# A simple knowledge-based agent

```
function KB-AGENT(percept) returns an action  
static: KB, a knowledge base  
         t, a counter, initially 0, indicating time  
  
TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))  
action ← ASK(KB, MAKE-ACTION-QUERY(t))  
TELL(KB, MAKE-ACTION-SENTENCE(action, t))  
t ← t + 1  
return action
```

The agent must be able to:

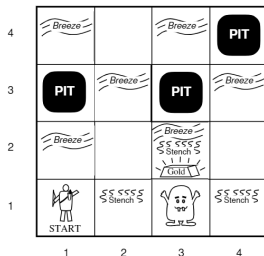
- Represent states, actions, etc.
- Incorporate new percepts
- Update internal representations of the world
- Deduce hidden properties of the world
- Deduce appropriate actions

# Wumpus World description

To put it simple

- ▶ Grid world
- ▶ Pit causes breeze in adjacent cells
- ▶ Wumpus causes stench in adjacent cells
- ▶ Find the gold

*Details are in next slide.*



# 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

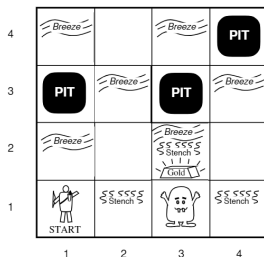
Grabbing picks up gold if in same square

Releasing drops the gold in same square

**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?? Yes—Wumpus and Pits do not move

Discrete?? Yes

Single-agent?? Yes—Wumpus is essentially a natural feature

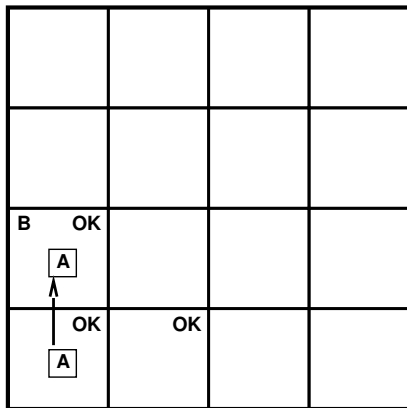
## Exploring a wumpus world

OK			
OK <div>A</div>	OK		

**A** = Agent; **B** = Breeze; **G** = Glitter, Gold; **OK** = Safe square;  
**P** = Pit; **S** = Stench; **V** = Visited; **W** = Wumpus

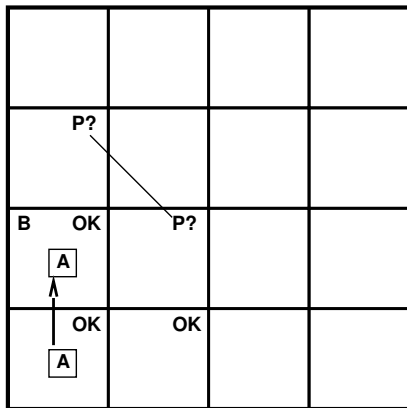


## Exploring a wumpus world



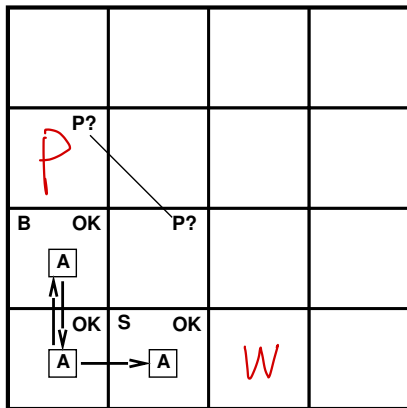
**A** = Agent; **B** = Breeze; **G** = Glitter, Gold; **OK** = Safe square;  
**P** = Pit; **S** = Stench; **V** = Visited; **W** = Wumpus

## Exploring a wumpus world



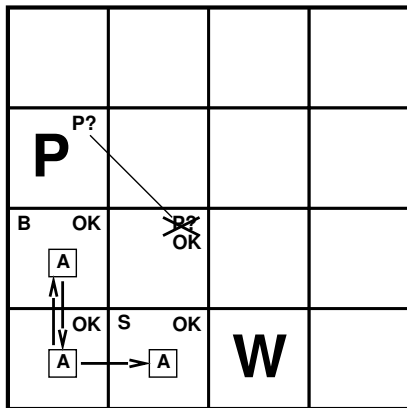
**A** = Agent; **B** = Breeze; **G** = Glitter, Gold; **OK** = Safe square;  
**P** = Pit; **S** = Stench; **V** = Visited; **W** = Wumpus

## Exploring a wumpus world



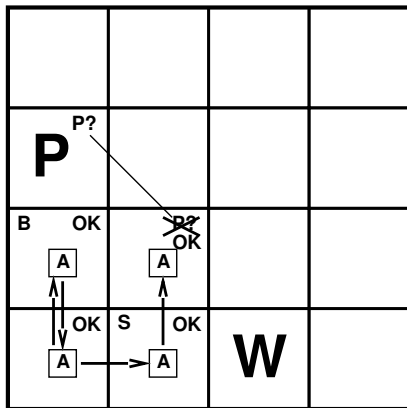
**A** = Agent; **B** = Breeze; **G** = Glitter, Gold; **OK** = Safe square;  
**P** = Pit; **S** = Stench; **V** = Visited; **W** = Wumpus

## Exploring a wumpus world

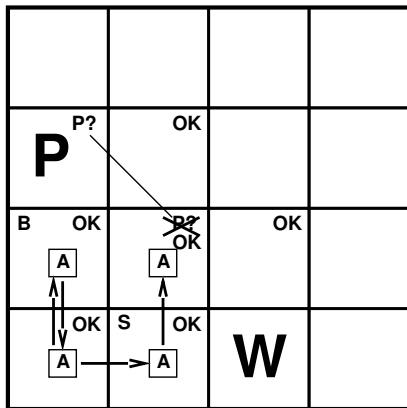


**A** = Agent; **B** = Breeze; **G** = Glitter, Gold; **OK** = Safe square;  
**P** = Pit; **S** = Stench; **V** = Visited; **W** = Wumpus

## Exploring a wumpus world

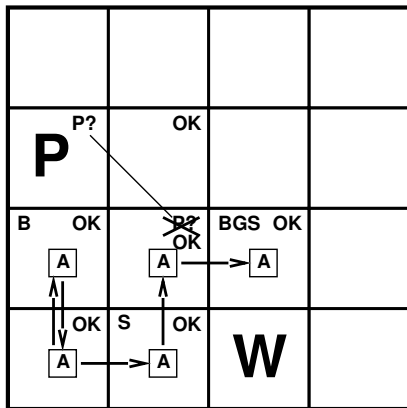


**A** = Agent; **B** = Breeze; **G** = Glitter, Gold; **OK** = Safe square;  
**P** = Pit; **S** = Stench; **V** = Visited; **W** = Wumpus



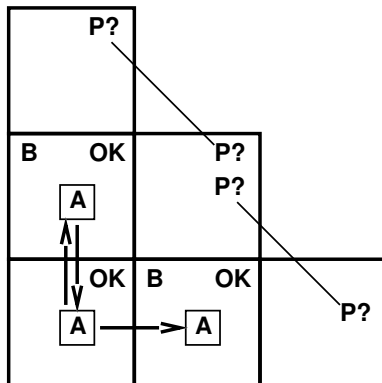
**A** = Agent; **B** = Breeze; **G** = Glitter, Gold; **OK** = Safe square;  
**P** = Pit; **S** = Stench; **V** = Visited; **W** = Wumpus

## Exploring a wumpus world



**A** = Agent; **B** = Breeze; **G** = Glitter, Gold; **OK** = Safe square;  
**P** = Pit; **S** = Stench; **V** = Visited; **W** = Wumpus

## Other tight spots

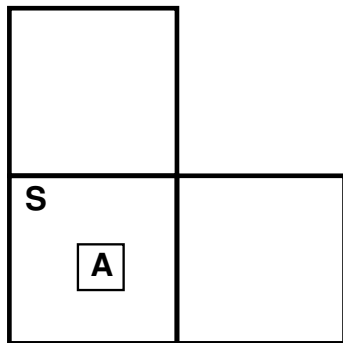


Breeze in (1,2) and (2,1)  
 $\Rightarrow$  no safe actions

We can calculate the probability!



## Other tight spots



Smell in (1,1)

$\Rightarrow$  cannot move

Can use a strategy of **coercion**:  
shoot straight ahead

wumpus was there  $\Rightarrow$  dead  $\Rightarrow$  safe

wumpus wasn't there  $\Rightarrow$  safe

# 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  $\alpha$

if and only if *models*

$\alpha$  is true in all worlds where  $KB$  is true

E.g., the KB containing “the Giants won” and “the Reds won” entails “Either the Giants won or the Reds won”

E.g.,  $x + y = 4$  entails  $4 = x + y$

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

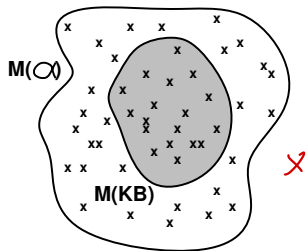
Note: brains process **syntax** (of some sort)

# Models

environment, world

- ▶ Logicians typically think in terms of **models**, which are formally structured worlds with respect to which truth can be evaluated
- ▶ We say  $m$  is a model of a sentence  $\alpha$  if  $\alpha$  is true in  $m$
- ▶  $M(\alpha)$  is the set of all models of  $\alpha$

Given sentences  $\alpha$  and  $\beta$ , we say  $\alpha$  **entails**  $\beta$  ( $\alpha \models \beta$ ), iff  $M(\alpha) \subset M(\beta)$ .  $KB \models \alpha$  if and only if  $M(KB) \subseteq M(\alpha)$

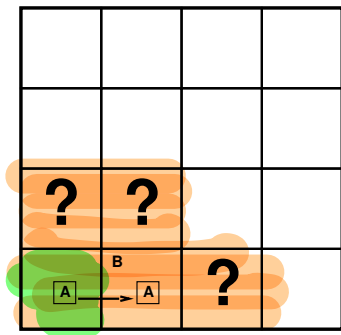


# Entailment in the wumpus world

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

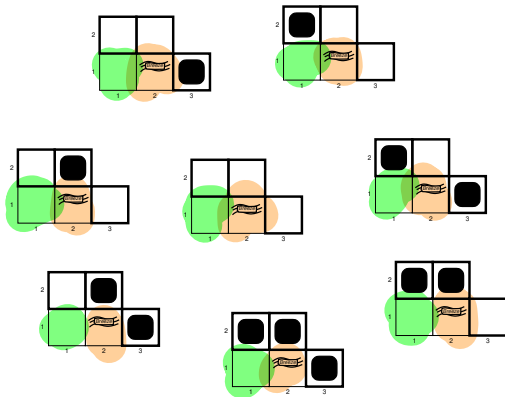
Consider possible models for ?s  
assuming only pits

3 Boolean choices  $\Rightarrow$  8 possible models

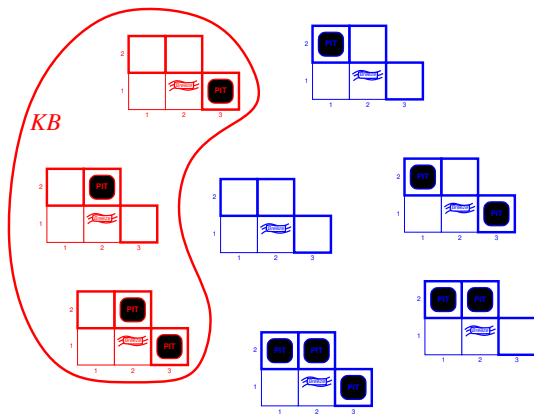


$$2^3 = 8$$

# Wumpus models

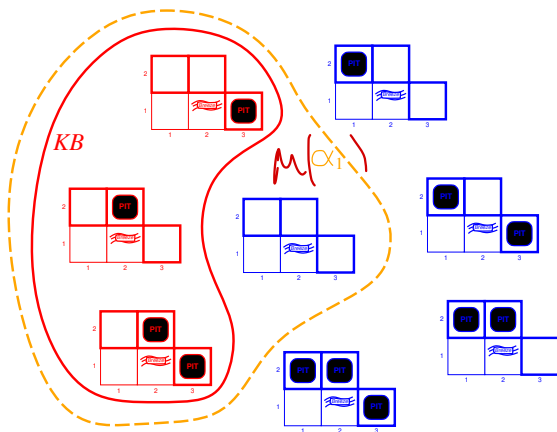


# Wumpus models



$KB$  = wumpus-world rules + observations

# Wumpus models

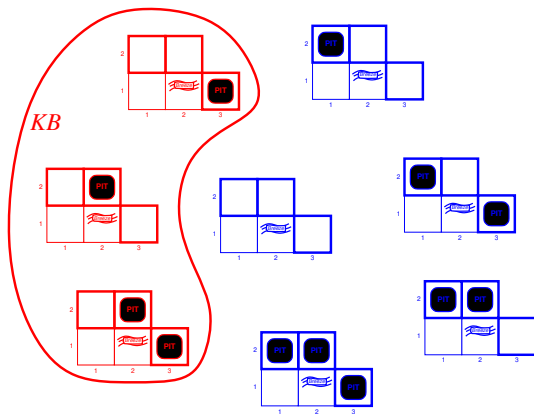


$KB$  = wumpus-world rules + observations

$\alpha_1$  = "no pit in  $[1,2]$ ",  $KB \models \alpha_1$ , proved by **model checking**:  
enumerates all possible models to check that  $\alpha$  is true in all  
models in which  $KB$  is true.

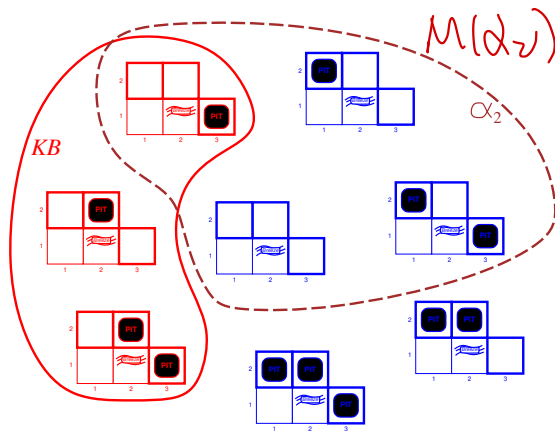


# Wumpus models



$KB$  = wumpus-world rules + observations

# Wumpus models



$KB$  = wumpus-world rules + observations

$\alpha_2$  = "no pit in [2,2]",  $KB \not\models \alpha_2$

# Inference

$KB \vdash_i \alpha$  = sentence  $\alpha$  can be derived from  $KB$  by some inference algorithm  $i$

For instance, model checking is an inference algorithm!

**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  $KB \vdash_i \alpha$

# Propositional logic: Syntax

- Propositional logic is the simplest logic, a.k.a., boolean logic
- The proposition symbols  $P_1$ ,  $P_2$  etc are **atomic sentences**
- We can generate **complex sentences** based on atomic sentences using **logical connectives**.

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.  $P_{1,2}$   $P_{2,2}$   $P_{3,1}$   $\beta_{1,2}$   $\beta_{2,2}$   $\beta_{2,3}$   
*true true false*

(With these symbols, 8 possible models, can be enumerated automatically.)

Rules for evaluating truth with respect to a model  $m$  (True: T; False: F)

$\neg S$	is T iff	$S$	is F
$S_1 \wedge S_2$	is T iff	$S_1$	is T <b>and</b> $S_2$ is T
$S_1 \vee S_2$	is T iff	$S_1$	is T <b>or</b> $S_2$ is T
$S_1 \Rightarrow S_2$	is T unless	$S_1$	is T <b>and</b> $S_2$ is F
$S_1 \Leftrightarrow S_2$	is T iff	$S_1 \Rightarrow S_2$	is T <b>and</b> $S_2 \Rightarrow S_1$ is T

Simple recursive process evaluates an arbitrary sentence, e.g.,

$$\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1}) = \text{false} \wedge (\text{true} \vee \text{false}) = \text{false}$$

*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  $[i, j]$ .

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

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

$$B_{2,1}$$

“Pits cause breezes in adjacent squares”

# 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”

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

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

“A square is breezy **if and only if** there is an adjacent pit”




# Truth tables for inference

$$R_1 : \neg P_{1,1} \quad R_2 : B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1}) \quad R_3 : B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$$

$$R_4 : \neg B_{1,1} \quad R_5 : B_{2,1}$$

$$KB : R_1 \wedge \cdots \wedge R_5$$



$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	$R_1$	$R_2$	$R_3$	$R_4$	$R_5$	$KB$
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
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
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	<u>true</u>
false	true	false	false	false	true	false	true	true	true	true	true	<u>true</u>
false	true	false	false	false	true	true	true	true	true	true	true	<u>true</u>
false	true	false	false	true	false	false	true	false	false	true	true	false
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
true	true	true	true	true	true	true	false	true	true	false	true	false

**Figure 7.9** A truth table constructed for the knowledge base given in the text.  $KB$  is true if  $R_1$  through  $R_5$  are true, which occurs in just 3 of the 128 rows (the ones underlined in the right-hand column). In all 3 rows,  $P_{1,2}$  is false, so there is no pit in  $[1,2]$ . On the other hand, there might (or might not) be a pit in  $[2,2]$ .

# Inference by enumeration

Depth-first enumeration of all models is sound and complete

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

  symbols  $\leftarrow$  a list of the proposition symbols in KB and  $\alpha$ 
  return TT-CHECK-ALL(KB,  $\alpha$ , symbols, { })
```

---

```
function TT-CHECK-ALL(KB,  $\alpha$ , symbols, model) returns true or false
  if EMPTY?(symbols) then
    if PL-TRUE?(KB, model) then return PL-TRUE?( $\alpha$ , model)
    else return true // when KB is false, always return true
  else do
    P  $\leftarrow$  FIRST(symbols)
    rest  $\leftarrow$  REST(symbols)
    return (TT-CHECK-ALL(KB,  $\alpha$ , rest, model  $\cup$  {P = true})
      and
      TT-CHECK-ALL(KB,  $\alpha$ , rest, model  $\cup$  {P = false}))
```

**Figure 7.10** A truth-table enumeration algorithm for deciding propositional entailment. (TT stands for truth table.) PL-TRUE? returns *true* if a sentence holds within a model. The variable *model* represents a partial model—an assignment to some of the symbols. The keyword “**and**” is used here as a logical operation on its two arguments, returning *true* or *false*.

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

# Logical equivalence

Two sentences are **logically equivalent** iff true in same models:

$\alpha \equiv \beta$  if and only if  $\alpha \models \beta$  and  $\beta \models \alpha$

$$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha) \quad \text{commutativity of } \wedge$$

$$(\alpha \vee \beta) \equiv (\beta \vee \alpha) \quad \text{commutativity of } \vee$$

$$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma)) \quad \text{associativity of } \wedge$$

$$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma)) \quad \text{associativity of } \vee$$

$$\neg(\neg\alpha) \equiv \alpha \quad \text{double-negation elimination}$$

$$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha) \quad \text{contraposition}$$

$$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta) \quad \text{implication elimination}$$

$$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination}$$

$$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta) \quad \text{De Morgan}$$

$$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta) \quad \text{De Morgan}$$

$$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) \quad \text{distributivity of } \wedge \text{ over } \vee$$

$$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) \quad \text{distributivity of } \vee \text{ over } \wedge$$

# 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**:

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

- A sentence is **satisfiable** if it is true in **some** model  
e.g.,  $A \vee B$ ,  $C$
- A sentence is **unsatisfiable** if it is true in **no** models  
e.g.,  $A \wedge \neg A$

Satisfiability is connected to inference via the following:

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

i.e., prove  $\beta$  by contradiction

$\neg(\alpha \Rightarrow \beta)$  is unsatisfiable  
 $\equiv \neg(\neg\alpha \vee \beta) \leftarrow$  implication elimination  
 $\equiv \alpha \wedge \neg\beta \leftarrow$  De Morgan law

# Proof methods

- Proof methods divide into (roughly) two kinds:

## Application of inference rules (detailed)

- 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 require translation of sentences into a **normal form**

## Model checking (brief)

truth table enumeration (always exponential in  $n$ )

improved backtracking, e.g., Davis–Putnam–Logemann–Loveland  
heuristic search in model space (sound but incomplete)

e.g., min-conflicts-like hill-climbing algorithms

- We use  $\frac{\alpha}{\beta}$  to imply that  $\alpha$  can infer  $\beta$ .

# Resolution

**Conjunctive Normal Form (CNF—universal)**

**conjunction** of **disjunctions** of **literals**  
**clauses**

*proposition symbol  
and its negation*

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

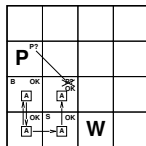
**Resolution** inference rule (for CNF):

$$\frac{\ell_1 \vee \dots \vee \ell_k, \quad m_1 \vee \dots \vee m_n}{\ell_1 \vee \dots \vee \ell_{i-1} \vee \ell_{i+1} \vee \dots \vee \ell_k \vee m_1 \vee \dots \vee m_{j-1} \vee m_{j+1} \vee \dots \vee m_n}$$

where  $\ell_i$  and  $m_j$  are **complementary literals**.

E.g.,

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



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  $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$ .

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

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:

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

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

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

  
CNF

# Resolution algorithm

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

**function** PL-RESOLUTION( $KB, \alpha$ ) **returns** *true* or *false*

**inputs:**  $KB$ , the knowledge base, a sentence in propositional logic  
 $\alpha$ , the query, a sentence in propositional logic

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

$new \leftarrow \{\}$

**loop do**

**for each**  $C_i, C_j$  **in**  $clauses$  **do**

$resolvents \leftarrow$  PL-RESOLVE( $C_i, C_j$ )

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

$new \leftarrow new \cup resolvents$

**if**  $new \subseteq clauses$  **then return** *false*

$clauses \leftarrow clauses \cup new$

↑  
CNF



## Resolution example

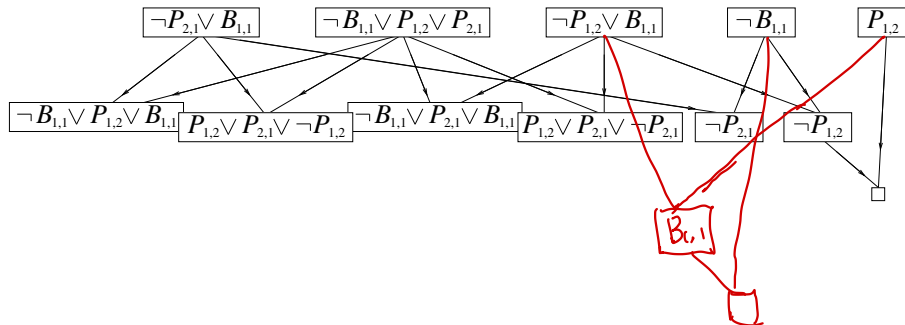
" $KB \models \neg P_{1,2}$ "  
 $\equiv KB \wedge P_{1,2}$  is unsatisfiable

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

First, transfer  $(B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1}))$  into CNF:

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

Second, apply PL-Resolution until empty clause appears!



# Forward chaining

- KB = **conjunction** of **Horn clauses**
  - **Horn clauses**: disjunction of literals of which at most one is positive
  - **Horn clauses** =
    - literal; or
    - (conjunction of symbols)  $\Rightarrow$  literal
- E.g.,  $C \wedge (B \Rightarrow A) \wedge (C \wedge D \Rightarrow B)$
- **Modus Ponens** (for Horn Form): complete for Horn KBs

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

- Can be used with **forward chaining**, which is very natural and runs 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
- We want to know whether  $Q$  is true? How to derive it?

$$P \Rightarrow Q$$

$$L \wedge M \Rightarrow P$$

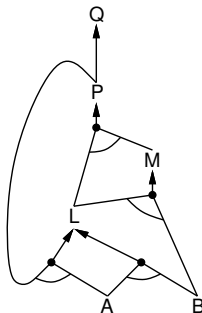
$$B \wedge L \Rightarrow M$$

$$A \wedge P \Rightarrow L$$

$$A \wedge B \Rightarrow L$$

$A$

$B$



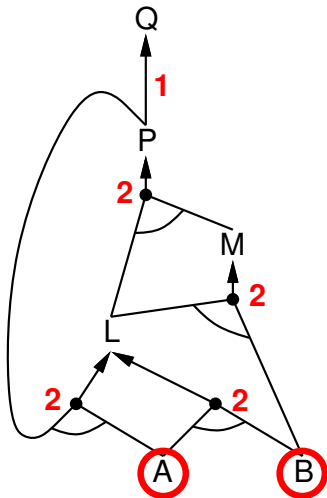
# Forward chaining algorithm

```
function PL-FC-ENTAILS?(KB, q) returns true or false
  inputs: KB, the knowledge base, a set of propositional definite clauses
           q, the query, a proposition symbol
  count  $\leftarrow$  a table, where count[c] is the number of symbols in c's premise
  inferred  $\leftarrow$  a table, where inferred[s] is initially false for all symbols
  agenda  $\leftarrow$  a queue of symbols, initially symbols known to be true in KB

  while agenda is not empty do
    p  $\leftarrow$  POP(agenda)
    if p = q then return true
    if inferred[p] = false then
      inferred[p]  $\leftarrow$  true
      for each clause c in KB where p is in c.PREMISE do
        decrement count[c]
        if count[c] = 0 then add c.CONCLUSION to agenda
  return false
```

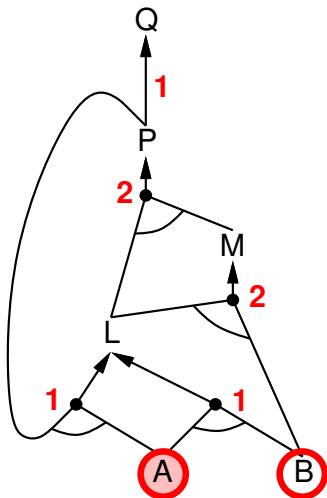
**Figure 7.15** The forward-chaining algorithm for propositional logic. The *agenda* keeps track of symbols known to be true but not yet “processed.” The *count* table keeps track of how many premises of each implication are as yet unknown. Whenever a new symbol *p* from the agenda is processed, the count is reduced by one for each implication in whose premise *p* appears (easily identified in constant time with appropriate indexing.) If a count reaches zero, all the premises of the implication are known, so its conclusion can be added to the agenda. Finally, we need to keep track of which symbols have been processed; a symbol that is already in the set of inferred symbols need not be added to the agenda again. This avoids redundant work and prevents loops caused by implications such as  $P \Rightarrow Q$  and  $Q \Rightarrow P$ .

## Forward chaining example



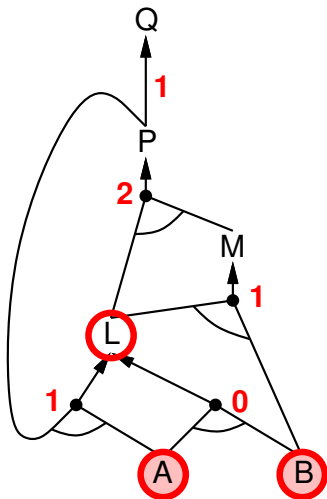
Start from  $A$

## Forward chaining example



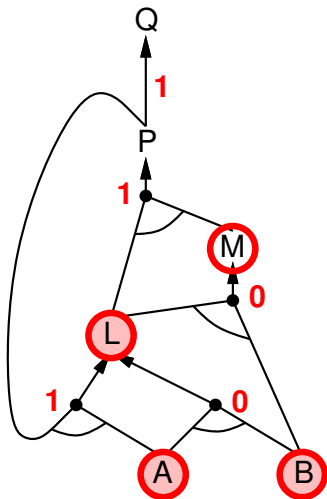
$A \wedge B \Rightarrow L$  decrease 1,  $A \wedge P \Rightarrow L$  decrease 1

## Forward chaining example



$count(A \wedge B \Rightarrow L) = 0$ , then add  $L$  into agenda

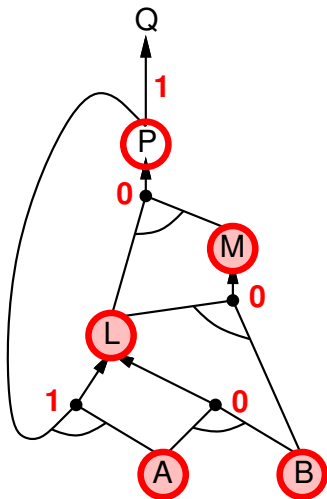
## Forward chaining example



$count(L \wedge B \Rightarrow M) = 0$ , then add  $M$  into agenda

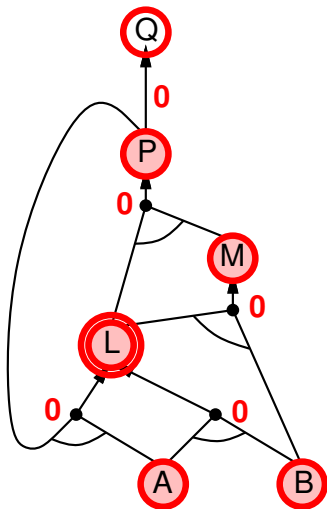


## Forward chaining example



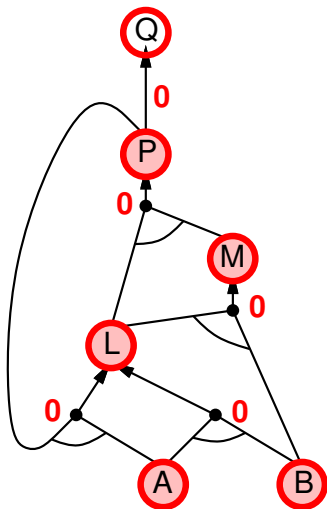
$\text{count}(L \wedge M \Rightarrow P) = 0$ , then add  $P$  into agenda

## Forward chaining example



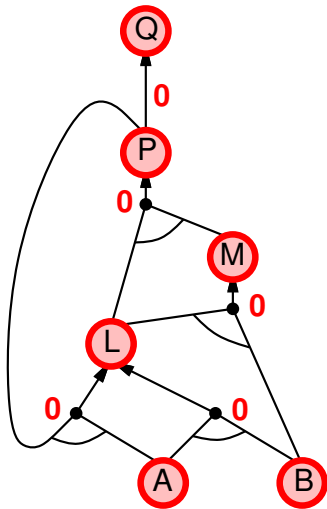
$L$  has already been inferred

## Forward chaining example



$Count(P \Rightarrow Q) = 0$ , Add  $Q$  into agenda

## Forward chaining example



# Forward chaining and Backward chaining

- Forward Chaining (FC) is **data-driven**, which may do lots of work that is irrelevant to the goal
- Another related algorithm is called Backward Chaining (BC), which is **goal-directed**. See the textbook for more details.

# 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

Propositional logic lacks expressive power

# Acknowledgment

The slides are adapted from Stuart Russell.