CS4300 Logical Agents

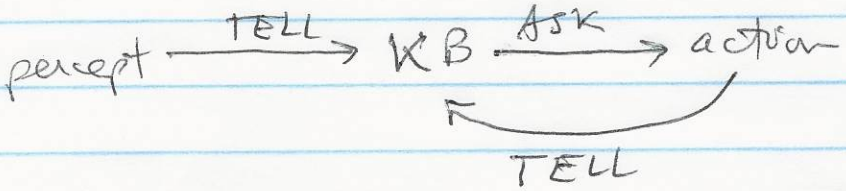
Knowledge base ≡ KB ≡ {sentences}
↑ ↑ true logical statements
agent's knowledge

axioms: always true
theorems: KB ⊢ T contingent

goal derive new theorems using percepts & KB

NS? KB as specific rules, functions, etc.
e.g., if percept(2) == 0 then action = FORWARD

percept ⟶ agent
- encode percept in KB (TELL)
- ask KB for action (ASK)
- record action in KB (TELL)



TELL: turn percept values into sentences
e.g., x=1, y=1, d=0, percept = [0,0,0,0,0]
⇒ B(1,1) = 0, St(1,1) = 0; G(1,1) = 0;
[more technically, $B_{11}=0$, $St_{11}=0$ $G^0_{11}=0$]

Ask: prove a theorem
e.g., FORWARD?
need a proof of FORWARD:

KB; $(At^1_{11}=1 \wedge B_{11}=0 \wedge St_{11}=0 \wedge Time=1 \wedge Dir^1=0) \rightarrow$ FORWARD;
$At^1_{11}=1$; $B_{11}=0$; $St_{11}=0$; Time=1; $Dir^1=0$
∴ FORWARD

ASK: GRAB?

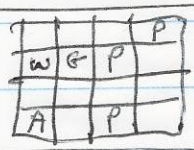
KB: $(AG^1_{11} = 1 \wedge HG^{a_1 d_1} = 0) \rightarrow GRAB$; $AT^1_{11} = 1$; $G^1_{11} = 1$; $AG^{a_1 d_1} = 0$

$AT^1_{11} = 1$

$\vdash GRAB$

So, agent asks for all actions to see which are OK according to KB, and then chooses 'best'

Wumpus World



Time	Loc	Percept	inference	action
1	1,1	00000	$\neg P21 \wedge \neg P12 \wedge \neg W21 \wedge \neg W12$	F
2	2,1	01000	$P31 \vee P22$; $\neg W31 \wedge \neg W22$	L
3	2,1	01000		L
4	2,1	01000		F
5	1,1	00000		R
6	1,1	00000		F
7	1,2	10000	$W22 \vee W13$; $\neg P22 \wedge \neg P13$ $\Rightarrow W13 \wedge P31$	R
8	1,2	10000		F
9	2,2	00000	$\neg P23 \wedge \neg P21$; $\neg W23 \wedge \neg W21$	F
10	3,2	01000	$P33 \vee P42 \vee P31$	L
11	3,2	01000		L
12	3,2	01000		F
13	2,2	00000		R
14	2,2	00000		F
15	2,3	11100	$G23 \wedge (P24 \vee P33)$	G

↑ how to do this reasoning?

syntax: symbols, parsing

semantics: truth assignments

model: a possible truth assignment where sentence is true

Consider $A \vee B = \alpha$

index	A	B	A ∨ B
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	1

} models for $A \vee B$

indexes 1, 2, 3 "model" α $M(\alpha) = \{1, 2, 3\}$

entailment $\alpha \models \beta$ β follows logically from α iff $M(\alpha) \subseteq M(\beta)$

Does $A \models (A \vee B)$

1 variable

$M(A) = \{1\}$

index	A
0	0
1	1

← models A

2 variables, A, B

index	A	B	A ∨ B
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	1

$M(A) = \{3, 4\}$

$M(A \vee B) = \{2, 3, 4\}$

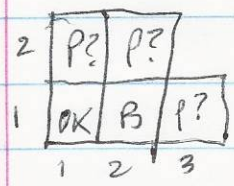
binary # whose value = index

so $M(A) \subseteq M(A \vee B)$

How about $(A \vee B) \models A$ $\{1, 2, 3\} \not\subseteq \{2, 3\}$ so NO

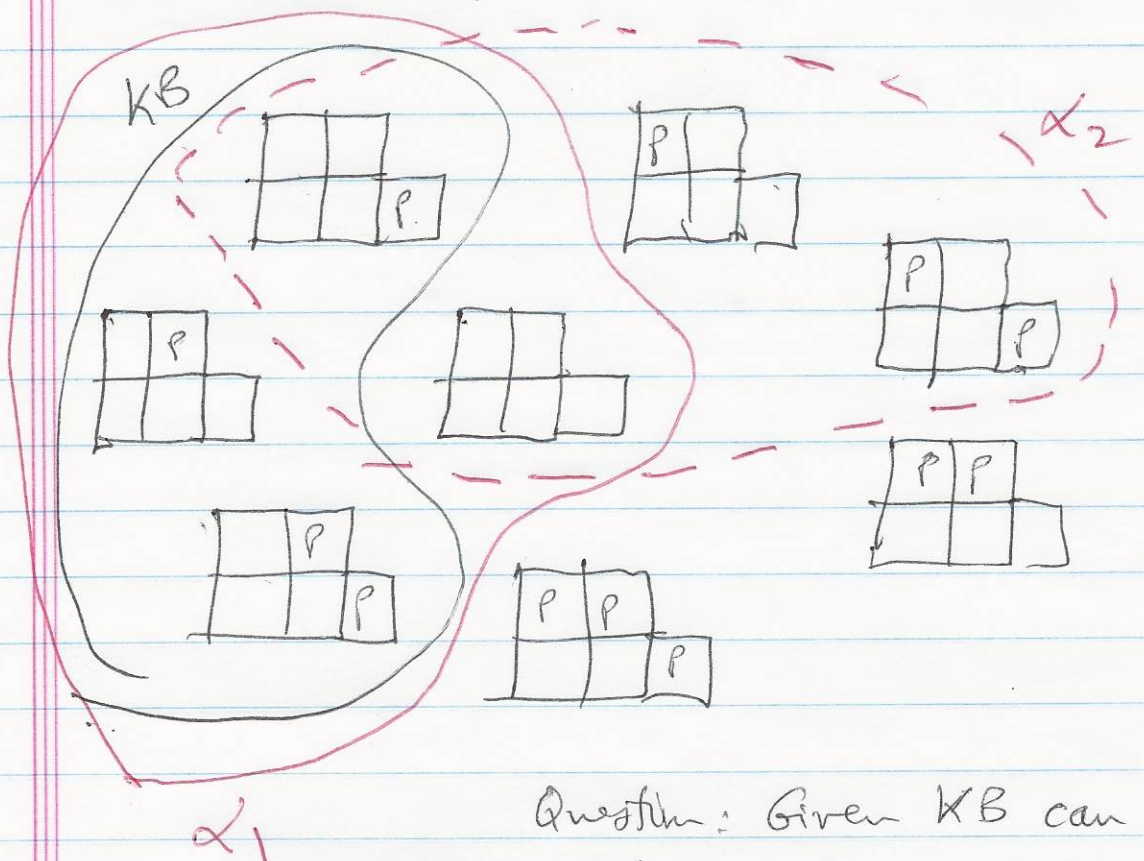
Consider Wumpus World

KB: nothing in 1,1 breeze in 2,1 + Wumpus world rules



	independent			dependent	
index	$P_{1,2}$	$P_{2,2}$	$P_{3,1}$	$\neg B_{1,1}$	$B_{2,1}$
0	0	0	0	1	0
1	0	0	1	1	1
2	0	1	0	1	1
3	0	1	1	1	1
4	1	0	0	0	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	0	1

each index corresponds to a scenario!



Question: Given KB can $\neg P_{1,2}$ be shown

$\alpha_1 \equiv \neg P_{1,2}$ $\alpha_2 \equiv \neg P_{2,2}$

$KB \models \alpha_1$ since $M(KB) \subseteq M(\alpha_1)$

$KB \not\models \alpha_2$ since $M(KB) \not\subseteq M(\alpha_2)$

Syntactic Proofs

$KB \vdash_i \alpha$

given inference rules i, α follows from KB by sequence of rule applications

KB	1. $\neg B_{1,1}$	from percept
	2. $B_{1,1} \leftrightarrow (P_{1,2} \vee P_{2,1})$	from WW rules
	3. $(P_{1,2} \vee P_{2,1}) \rightarrow B_{1,1}$	2; sufficient condition
	4. $\neg B_{1,1} \rightarrow \neg (P_{1,2} \vee P_{2,1})$	3; contrapositive
	5. $\neg (P_{1,2} \vee P_{2,1})$	1, 4; Modus Ponens
	6. $\neg P_{1,2} \wedge \neg P_{2,1}$	5; De Morgan's Law
	7. $\neg P_{1,2}$	6; meaning of $\wedge$
	8. $\neg P_{2,1}$	6; meaning of $\wedge$

p. 249 gives a set of rules

Technically speaking, there are steps missing before statement 3; what are they?

2a. $(B_{1,1} \rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \rightarrow B_{1,1})$ — 2; bicanditional elimination

3. $(P_{1,2} \vee P_{2,1}) \rightarrow B_{1,1}$ — 2a; meaning of $\wedge$

How could such a sequence be found automatically?

- Brute force
- BFS, ...

always combinatorics: syntax (rules) - semantics (truth values)

7/6

Propositional Logic (Propositional Calculus, Sentential Calculus)

syntax: grammar to define a legal sentence

What's a grammar? Technically! CS!

$G = (T, N, P, S)$

T: set of terminal symbols
N: set of nonterminal symbols
P: set of productions (rewrite rules)
S: start symbol

variable names

$T \equiv \{ True, False, P, Q, R, \ldots, [,], \neg, \wedge, \vee, \Rightarrow, \Leftrightarrow, (,) \}$

$N \equiv \{ Sentence, AtomicSentence, ComplexSentence \}$

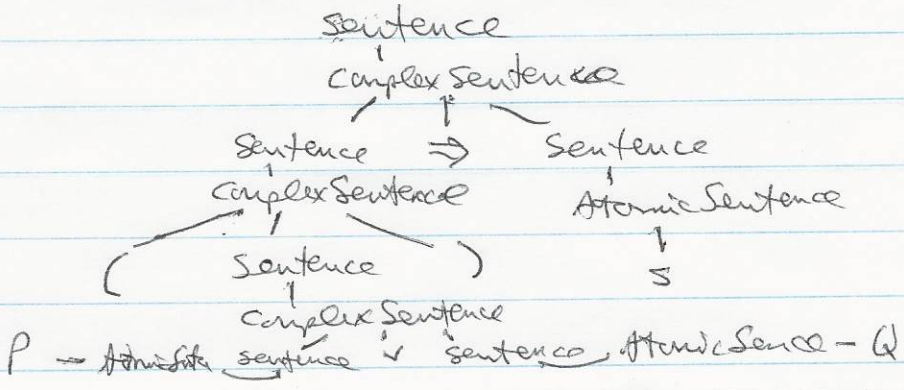
$P \equiv$ Sentence → AtomicSentence | ComplexSentence

AtomicSentence → True | False | P | Q | R | ... variable names

ComplexSentence → (Sentence)
| [Sentence]
| ¬ Sentence
| Sentence ∧ Sentence
| Sentence ∨ Sentence
| Sentence ⇒ Sentence
| Sentence ⇔ Sentence

$S \equiv$ Sentence

$(P \vee Q) \Rightarrow S$



$W_{1,1}$ is an Atomic Sentence
we do not have mechanism to use $W(1,1)$
$W(1,1)$ is a function $W: X \times Y \rightarrow \{0,1\}$

Semantics : truth of sentence given by truth of variables and operators

truth of:
Atomic Sentences : True if True
False if False
Variables must be specified

Complex Sentences:
$\neg P$ true iff P is false
$P \wedge Q$ true iff P is true and Q is true
$P \vee Q$ " " P is true or Q is true
$P \Rightarrow Q$ " unless P is True and Q is false
$P \Leftrightarrow Q$ " if both P & Q true or P & Q false

Truth Tables

precedence
1 first

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$	$\neg P \vee Q$
0	0	1	0	0	1	1	1 1
0	1	1	0	1	1	0	1 1
1	0	0	0	1	0	0	0 0 0
1	1	0	1	1	1	1	0 1 1

equivalent expressions

Some things true in all models

P	$P \vee \neg P$
0	1 10
1	1 1

$\models (P \vee \neg P)$ tautology

if contradiction in KB, then KB can imply anything
$(A \wedge \neg A) \Rightarrow B$ $\quad 0 \Rightarrow B$ true

Simple KB for WW

PXY true if Pit at XY
WXY " " Wumpus "
BXY " " Breeze "
SXY " " Stench "

Try to show $\neg P12$

R1: $\neg P11$
R2: $B11 \Leftrightarrow (P12 \vee P21)$
R3: $B21 \Leftrightarrow (P11 \vee P22 \vee P31)$
R4: $\neg B11$
R5: $B21$

Does $KB \models \neg P12$

7 proposition symbols: B11, B21, P11, P12, P21, P22, P31

$2^7 = 128$ models (assignments)
KB is true in 3 (How do we know?
$\neg P12$ is true in 64

assign key to each model

index	Val	B11	B21	P11	P12	P21	P22	P31
1	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	1
3	2	0	0	0	0	0	1	0
⋮								
34	33	0	1	0	0	0	0	1
35	34	0	1	0	0	0	1	0
36	35	0	1	0	0	0	1	1

7/9

algorithm on p. 248

easier to just generate combinations
then check truth value of sentence

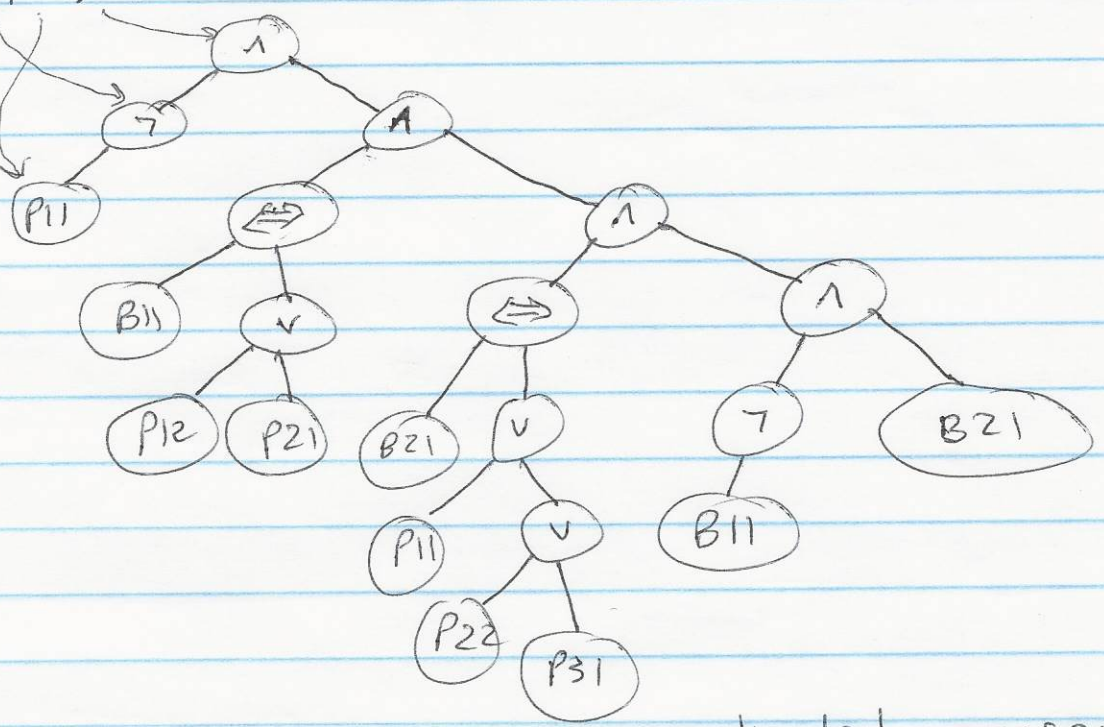
for val = 0 : 2^k - 1 % for k variables
    truth_vals = map_val_onto_k_bits
    if truth_vals substituted in sentence make
        sentence true
    then add index to models % e.g., val + 1

How to substitute into sentence ?
Worse : How to represent a (logical) Sentence ?

$R_1 \wedge R_2 \wedge R_3 \wedge R_4 \wedge R_5 \equiv KB$

$\equiv (\neg P11) \wedge (B11 \Leftrightarrow (P12 \vee P21)) \wedge (B21 \Leftrightarrow (P11 \vee P22 \vee P31)) \wedge (\neg B11) \wedge (B21)$

$\equiv$



⇒ complicated parser !

representation

propositions : complicated strings
connectives : all possible

7/10

(p. 253)

Use more regular (and simple) sentence structure

Conjunctive Normal Form (CNF)

literal: proposition or ¬proposition (variables)
clause: disjunction of literals (∨)
conjunction of clauses: and'd clauses (∧)

$$C_1 \wedge C_2 \wedge \dots \wedge C_n$$
$$C_i = D_{i1} \vee D_{i2} \vee \dots \vee D_{in_i}$$

Can convert complex sentence to CNF

1. $\alpha \Leftrightarrow \beta \equiv (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$
2. $\alpha \Rightarrow \beta \equiv \neg\alpha \vee \beta$
3. Move ¬ inward
   - $\neg(\neg\alpha) \equiv \alpha$
   - $\neg(\alpha \vee \beta) \equiv \neg\alpha \wedge \neg\beta$
   - $\neg(\alpha \wedge \beta) \equiv \neg\alpha \vee \neg\beta$
4. distribute out ∧'s and ∨'s
   - $(\alpha \wedge (\beta \vee \gamma)) \equiv (\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$
   - $(\alpha \vee (\beta \wedge \gamma)) \equiv (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$

Now given CNF, how to represent:

in Wumpus World, can know every possible variable (not their truth value – the variable name)

e.g., P11, P21, P31, ..., P44
B11, B21, B31, ..., B44
W11, W21, ... W44

Let integers 1:32 represent 1st 32 variables (P & B)
Then literals can be + or - integer to represent ~~not or true or false~~ literal

$1 \equiv P11 \qquad -1 \equiv \neg P11$
$17 \equiv B11 \qquad -17 \equiv \neg B11$

* note: we do not know the value of P11

$(\neg P11) \wedge (B11 \Leftrightarrow (P12 \vee P21)) \wedge (B21 \Leftrightarrow (P11 \vee P22 \vee P31)) \wedge (\neg B11) \wedge (B21)$
$-1 \wedge \qquad \wedge \qquad ? \qquad \wedge -17 \wedge 18$

$(B11 \Rightarrow (P12 \vee P21)) \wedge ((P12 \vee P21) \Rightarrow B11)$
$(\neg B11 \vee P12 \vee P21) \wedge (\neg(P12 \vee P21) \vee B11)$
$-17 \quad 5 \quad 2 \quad ((\neg P12 \wedge \neg P21) \vee B11)$

$(\neg P12 \vee B11) \wedge (\neg P21 \vee B11)$
$-5 \quad 17 \quad -2 \quad 17$

$S \equiv (-1) \wedge (-17, 5, 2) \wedge (-5, 17) \wedge (-2, 17) \wedge ( ) \wedge ( ) \wedge ( ) \wedge (-17) \wedge (18)$

S(1).con = [-1]
S(2).con = [-17, 5, 2]
S(3).con = [-5, 17]
:
S(9).con = [18]

} a conjunction of disjunctions

Algorithm (given a truth value combination

```
S = 1;      % assume true
for each conjunction Ci
    S = S ∧ Ci;
end
```

/? How to do this?

Given $C_i \equiv d_{i1} \vee d_{i2} \vee \ldots \vee d_{in_i}$

```
d = 0;      % assume false
for each disjunction dj
    d = d ∨ dj
end
```

/? How to do this?

```
index = |dj|
if sign(dj) < 0
   val = 1 - index^th bit
else
   val = index^th bit
end
```

This is Extra Credit algorithm

7/13

How to prove theorems?

$\alpha \models \beta$ iff $\alpha \Rightarrow \beta$ is valid

valid ≡ true in all models ($\emptyset \models \alpha \Rightarrow \beta$)

satisfiability : α is true in some model
(2^k possibilities ⇒ NP-complete)

another way:

$\alpha \models \beta$ iff $\alpha \wedge \neg\beta$ is unsatisfiable

reductio ad absurdum

resolution theorem proving

Resolution complete when coupled with complete search

unit resolution

$$(1)\ \frac{A \vee B \ \wedge \ \neg A}{B}$$

$$(1)\ \frac{\neg A \vee B \ \wedge \ A}{B}$$

$$(1)\ \frac{\neg A \vee B \vee C \ \wedge \ A}{B \vee C}$$

$$\frac{l_1 \vee l_2 \vee \dots \vee l_i \vee \dots l_k \ \wedge \ m}{l_1 \vee l_2 \vee \dots \vee l_{i-1} \vee l_{i+1} \vee \dots \vee l_k} \quad \text{where } m \equiv \neg l_i$$

$$\frac{l_1 \vee l_2 \vee \dots \vee l_k \ \wedge \ m_1 \vee m_2 \vee \dots \vee m_n}{l_1 \vee l_2 \vee \dots \vee l_{i-1} \vee l_{i+1} \vee \dots l_k \vee m_1 \vee m_2 \vee \dots \vee m_{j-1} \vee m_j \vee \dots m_n} \quad m_j \equiv \neg l_i$$

Eig., show $A \models A \vee B$

1. A
2. $\neg(A \vee B)$

$\xrightarrow{\text{to CNF}}$

1. A
2. $\neg A$
3. $\neg B$

---

4. Nil 1, 2; Res.

Truth Value

$A \rightarrow (A \vee B)$

A	B	A	→	(A	∨	B)
0	0	0	1			
0	1	0	1			
1	0	1	1	1	1	
1	1	1	1	1	1	

Algorithm Resolution to check $KB \models \alpha$

(1) Convert $KB \wedge \neg\alpha$ to CNF

(2) Each pair of clauses that contains complementary literals is resolved to produce a new clause which is added to the set

(3) Repeat (2) until
* no new clause found: $KB \not\models \alpha$
* Nil clause produced: $KB \models \alpha$

Note: any clause with complementary literals may be discarded: e.g., $P \vee \neg R \vee L_1 \vee \dots \vee L_K$

E.g., KB ≡ 1. $B_{11} \Leftrightarrow (P_{12} \vee P_{21})$
2. $\neg B_{11}$

$\alpha \equiv \neg P_{12}$

CNF:

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

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

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

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

⇓

1. $\neg B_{11} \vee P_{12} \vee P_{21}$
2. $B_{11} \vee \neg P_{12}$
3. $B_{11} \vee \neg P_{21}$

} from KB 1.

4. $\neg B_{11}$ } from KB 2
5. P_{12} } conjoin $\neg\alpha$ to KB

---

6. $\neg P_{12}$ Resolve 2 and 4
7. $\emptyset$ Resolve 5 and 6

Fluents Some aspects of the world change
e.g., HaveArrow, Location, HaveGold ...
use superscript for time

L_{11}^0 agent in [1,1] at time 0

$L_{xy}^t \rightarrow (Breeze^t \Leftrightarrow Bxy)$

$L_{11}^0 \wedge FacingEast^0 \wedge FORWARD^0 \Rightarrow (L_{21}^1 \wedge \neg L_{11}^1)$

gets complicated:

$Forward^t \Rightarrow (HaveArrow^t \Leftrightarrow HaveArrow^{t+1})$