Propositional Logic

- Boolean Logic - Can be used to do deductions in settings like the Wumpus world
- Syntax has following components:
  - Atomic Sentences / Atomic Formulas / Propositions : Can be TRUE/FALSE
    - Example : There is Breeze in [1,2] (Denoted by $B_{1,2}$)
      Agent is facing east (Denoted by FacingEast)
    - An Atomic Sentence can be TRUE / FALSE (some change over time, some do not)
  - Negation : There is NO Stench in [2,2] ($\neg S_{2,2}$)
  - Conjunction: There is NO stench in [3,2] AND Agent is Facing West ($\neg S_{3,2} \land$ FacingWest)
  - Disjunction : There is Stench in [2,3] OR Wumpus is NOT there in [3,3] ($S_{2,3} \lor \neg W_{3,3}$)
  - Implication : IF there is NO stench in [2,3] THEN Wumpus is NOT there in [3,3] ($\neg S_{2,3} \Rightarrow \neg W_{3,3}$)
  - Bi-Implication Agent is Facing east IFF Agent is NOT Facing West (FacingEast $\Leftrightarrow \neg$ FacingWest)

Propositional Logic : Syntax

$$
\begin{aligned}
\textit{Sentence} &\rightarrow \textit{AtomicSentence} \mid \textit{ComplexSentence} \\
\textit{AtomicSentence} &\rightarrow \textit{True} \mid \textit{False} \mid P \mid Q \mid R \mid \ldots \\
\textit{ComplexSentence} &\rightarrow (\, \textit{Sentence} \,) \\
&\mid \neg\, \textit{Sentence} \\
&\mid \textit{Sentence} \land \textit{Sentence} \\
&\mid \textit{Sentence} \lor \textit{Sentence} \\
&\mid \textit{Sentence} \Rightarrow \textit{Sentence} \\
&\mid \textit{Sentence} \Leftrightarrow \textit{Sentence} \\
\textsc{Operator Precedence} &: \neg, \land, \lor, \Rightarrow, \Leftrightarrow
\end{aligned}
$$

Propositional Logic : Semantics

- Tells us how to Assign TRUE / FALSE to sentences

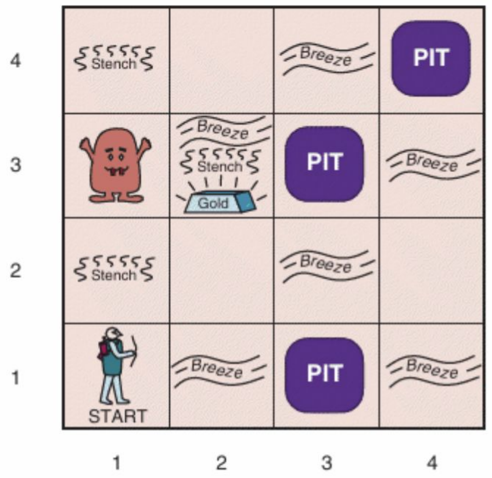
- Model specifies TRUE / FALSE for Atomic sentences
  - Example : If P,Q, R are the Atomic sentences them M = { P = TRUE, Q = TRUE, R = FALSE }
  - Models are just mathematical objects (Might or might not correspond to a real world instance)
    - Example : M = { $W_{2,2}$ = TRUE, $S_{3,2}$ = FALSE ....} is a model but does not correspond to any Wumpus world

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*

Modelling Wumpus World in Propositional Logic

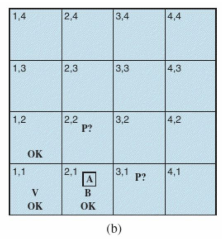
- First consider immutable aspects:
  - Those that do not change over time

- For every location [x,y], let us have the following atomic sentences:
  - $P_{x,y}$ is TRUE if there is a Pit in [x,y]
  - $W_{x,y}$ is TRUE if there is a Wumpus in [x,y] (either dead or alive)
  - $B_{x,y}$ is TRUE if it is Breezy in [x,y]
  - $S_{x,y}$ is TRUE if there is Stench in [x,y]
  - $L_{x,y}$ is TRUE if the Agent is located in
    ($L_{x,y}$ is not immutable, we will consider such atomic statements this



- Some sentences in our Knowledge Base:
  - There is no Pit in [1,1] $R_1 : \neg P_{1,1}$
  - A square is Breezy iff at least one of its neighbours has

    $R_2 : B_{1,1} \Leftrightarrow (P_{2,2} \lor P_{2,1})$ $R_3 : B_{2,1} \Leftrightarrow (P_{1,1} \lor P_{2,2} \lor P_{3,1})$

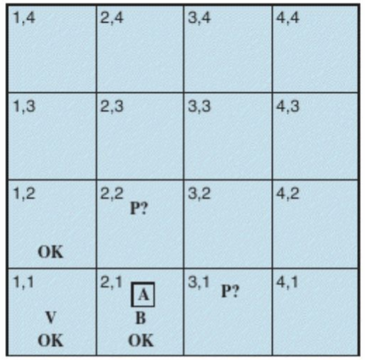


  - R_1 , R_2 and R_3 are TRUE in all Wumpus Worlds
  - Sentences TRUE in the given Wumpus World (known through percepts)

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

Inferences in Wumpus World in Propositional Logic

- Knowledge base (KB) has the following information:
  - $R_1 : \neg P_{1,1}$ $R_2 : B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
    $R_3 : B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
    $R_4 : \neg B_{1,1}$ $R_5 : B_{2,1}$



(b)

  - Can we infer $\neg P_{1,2}$ from the KB ?
  - Can we infer $\neg P_{2,2}$ from the KB ?

- EXERCISE:
  Design an Algorithm that takes a KB and a sentence α and checks whether KB $\vDash \alpha$ using the model checking method.

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	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*	<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*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*true*	*true*	*false*	*true*	*false*

Inferences in Propositional Logic via Model Checking

- We have a Sound and Complete Algorithm using the Model Checking Method
  - Sound : All Inferences are correct
  - Complete: Every KB $\models$ α can be inferred.
- But takes a lot of time
  - If there are n Atomic sentences under consideration, how many different models do we need to check?
- Checking if KB $\models \alpha$ for propositional logic is a Co-NP complete problem.
  - So any approach will take "a lot" of time.
- Is there a way to do Inferences without going through all models?
  - Yes, via Theorem Proving
  - Useful when the number of models is large but there is a short "proof"

Theorem Proving

- A different approach to making deductions.

- Does not go through all models
  - More Syntactic

- Done by applying Rules of Inferences
  - Example : If KB has $\alpha \vee \beta$ and $\neg\alpha$ then we can say KB $\vDash \beta$
    - This is based on the following Rule :

$$\frac{\alpha \vee \beta \qquad \neg\alpha}{\beta}$$

  - We can have many such rules

- Goal is to arrive at α starting from KB, using a sequence of application of such rules

Towards obtaining the rules

- Logical Equivalence:
  - α is equivalent to β (Denoted by $\alpha \equiv \beta$) if $M(\alpha) = M(\beta)$
    - Same as saying: For every model M: $M \models \alpha$ iff $M \models \beta$
    - $\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$

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

Towards obtaining the rules

- Validity / Tautology: α is a Validity if α is TRUE in all Models
    - $\alpha \vDash \beta$ iff $\alpha \Rightarrow \beta$ is a validity
    - Hence, to check if $\alpha \vDash \beta$ it is enough to check if $\alpha \Rightarrow \beta$ is a validity
- Satisfiability: α is Satisfiable if α is TRUE in some Model
    - SAT for Propositional Logic was the first problem that was proved to be NP-complete
- α is Valid iff $\neg\alpha$ is not satisfiable
    - Hence, $\alpha \vDash \beta$ iff $(\alpha \land \neg\beta)$ is not Satisfiable
- We will try to develop rules for making deductions

Inference Rules

- Modus Ponens :

$$\frac{\alpha \Rightarrow \beta \qquad \alpha}{\beta}$$

  - If (WumpusAhead ∧ WumpusAlive) ⇒ Shoot and (WumpusAhead ∧ WumpusAlive) are given then Shoot can be inferred

- AND - Elimination:

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

- These Rules are Sound
  - Proof: Consider all possible Truth values for α and β to verify

- Always have Sound rules (otherwise we will be making wrong deductions)

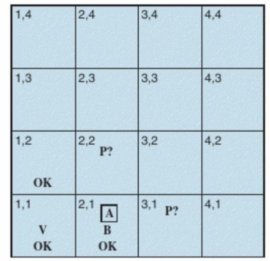
Inference Using Rules

- Knowledge base (KB) has the following information:

$R_1 : \neg P_{1,1}$ $R_2 : B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})$

$R_3 : B_{2,1} \Leftrightarrow (P_{1,1} \lor P_{2,2} \lor$ F

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



(b)

- How to infer $\neg P_{1,2}$ from the KB using Inference Rules ?
  - $R_6 : (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})$ [Biconditional elimination on R_2]
  - $R_7 : ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})$ [And-Elimination on R_6]
  - $R_8 : (\neg B_{1,1} \Rightarrow \neg (P_{1,2} \lor P_{2,1}))$ [Logical Equivalence of Contrapositives on R_7]
  - $R_9 : \neg (P_{1,2} \lor P_{2,1})$ [Modus Ponens on R_8 and R_4]
  - $R_{10} : \neg P_{1,2} \land \neg P_{2,1}$ [De Morgan's Law on R_9]
  - $R_{11} : \neg P_{1,2}$ [And-Elimination on R_{10}]

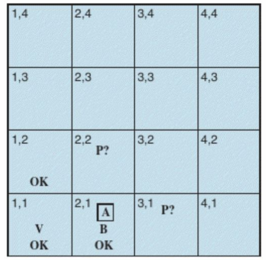
Algorithm for Inference using Rules

- Can be seen of as a Search Problem:
  - Initial State : Knowledge Base KB [Every state corresponds to a set of sentences]
  - Actions : All the inference rules applied to all the sentences that match the top half of the inference rule.
  - Result : of an Action is to add the sentence in the bottom half of the inference rule.
  - Goal : States that contain the sentence we are trying to prove.

- Monotonicity Property : If $KB \models \alpha$ then $KB \land \beta \models \alpha$
  - Search can ignore sentences that are not relevant to α
  - There are some Non-Monotonic Logics but we will not discuss it here

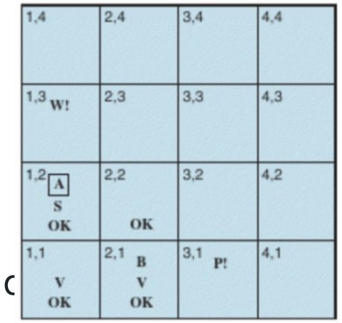
- This algorithm works ONLY IF we have a "complete" set of Inference Rules
  - How do we know we have sufficiently many inference rules?

Another Interesting Rule

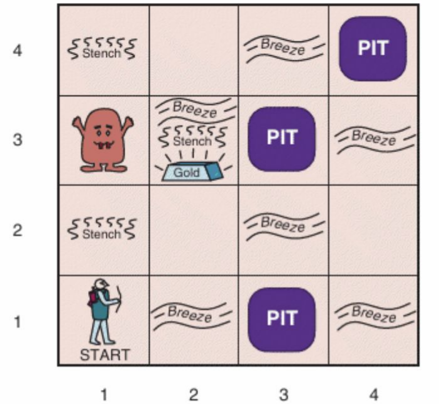


(b)

- R_1 : $\neg P_{1,1}$
  R_3 : $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
  R_5 : $B_{2,1}$
  R_7 : $((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
  R_9 : $\neg (P_{1,2} \vee P_{2,1})$
  R_{11} : $\neg P_{1,2}$
  R_2 : $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
  R_4 : $\neg B_{1,1}$
  R_6 : $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee$
  R_8 : $(\neg B_{1,1} \Rightarrow \neg (P_{1,2} \vee P_{2,1}))$
  R_{10} : $\neg P_{1,2} \wedge \neg P_{2,1}$



- Say we also have the following KB
  R_{12} : $\neg B_{1,2}$ R_{13} : $B_{1,2} \Leftrightarrow (P_{1,1} \vee$)
- Using similar reasoning as previous example, from R_{12} and ... get:
  R_{14} : $\neg P_{2,2}$ R_{15} : ... $P_{1,3}$
- Using Bi-conditional elimination on R_3 followed by And-elimination and MP on R_5
  R_{16} : $(P_{1,1} \vee P_{2,2} \vee P_{3,1})$



- Resolution Rule : $\neg P_{2,2}$ in R_{14} Resolves $P_{2,2}$ ... e
  R_{17} : ($P_{1,1}$ $\vee$)
- Resolution Rule : $\neg P_{1,1}$ in R_1 Resolves $P_{1,1}$ ... e
  R_{18} : $P_{3,1}$

Unit Resolution

- $$\frac{\ell_1 \vee \ell_2 \vee \dots \ell_{i-1} \vee \ell_i \vee \ell_{i+1} \vee \dots \ell_k \qquad m}{\ell_1 \vee \ell_2 \vee \dots \ell_{i-1} \vee \ell_{i+1} \vee \dots \ell_k} \quad \text{(Unit Resolution Rule)}$$
- Here, each ℓ_j is a literal, m is also a literal, and ℓ_i and m are complementary literals (one is the negation of the other)
- Called Unit Resolution because second sentence is a unit clause (contains one literal)
- Can we generalize the second sentence to a disjunction ?

Resolution

- $\ell_1 \lor \ell_2 \lor \dots \ell_{i-1} \lor \ell_i \lor \ell_{i+1} \lor \dots \ell_k \qquad m_1 \lor m_2 \lor \dots m_{j-1} \lor m_j \lor m_{j+1} \lor \dots m_n$
  ——————————————————————————— (Resolution Rule)
  $\ell_1 \lor \dots \ell_{i-1} \lor \ell_{i+1} \lor \dots \ell_k \lor m_1 \lor \dots m_{j-1} \lor m_{j+1} \lor \dots m_n$
- Where, each ℓ_i and m_j are complementary literals (one is the negation of the other)
- Example :
  $(P_{1,1} \lor P_{3,1}) \qquad (\neg P_{1,1} \lor \neg P_{2,21})$
  ———————————————————
  $(P_{3,1} \lor \neg P_{2,21})$
- Only one literal can be resolved at a time

  P V¬Q V R ¬PVQ
  ——————————
  ¬Q V R V Q

  P V¬Q V R ¬PVQ
  ——————————
  P V R V ¬P

  P V ¬Q V R ¬PVQ
  —————————— (Wrong)
  R

- Resolution Rule is Sound
- THIS SINGLE RULE COMPLETE

Conjunctive Normal Form

- Resolution applies only if the two sentences are in the form of Disjunctions (Clause)
  - How can it be Complete?
- Every Propositional Sentence can we written as a conjunction of Clauses

$$
\begin{aligned}
\mathit{CNFSentence} &\rightarrow \mathit{Clause}_1 \wedge \cdots \wedge \mathit{Clause}_n \\
\mathit{Clause} &\rightarrow \mathit{Literal}_1 \vee \cdots \vee \mathit{Literal}_m \\
\mathit{Literal} &\rightarrow \mathit{Symbol} \mid \neg \mathit{Symbol} \\
\mathit{Symbol} &\rightarrow P \mid Q \mid R \mid \ldots
\end{aligned}
$$

Conjunctive Normal Form

- Every Propositional Sentence can be written as a CNF sentence
- Example: $B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})$
    - $(B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})$
    - $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg(P_{1,2} \lor P_{2,1}) \lor B_{1,1})$
    - $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land ((\neg P_{1,2} \land \neg P_{2,1}) \lor B_{1,1})$
    - $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1})$

Resolution Algorithm

- To check whether KB ⊨ **α** , we will check whether (KB ∧ ¬**α**) is unsatisfiable
    - Convert (KB ∧ ¬**α**) into CNF and let S = Set of all clauses in the CNF
    - Apply the resolution rule whenever possible and keep on adding them to S
        - If at any point of time, Empty Clause is added to S then return **KB entails α**
        - If there are no new clauses that can be added to S, then return **KB does not entail α**

- Empty clause - Disjunction of 0 literals
    - Will be obtained when we resolve P and ¬P
    - Empty Clause is not satisfiable (because, for a disjunction to be true, at least one of the disjuncts should be true)
    - For any formula **α** if the algorithm derives an empty clause then **α** is not satisfiable
        - Proof : Rule ensures that if antecedant is satisfiable then consequent is also satisfiable But the last consequent is not satisfiable, hence its parent is not satisfiable Keep going up the tree and at the root we have **α** which will not be satisfiable

- Soundness: If the Algorithm returns KB entails **α** then it is actually the case
    - Algorithm returns KB entails **α** only when Empty Clause is derived for (KB ∧ ¬**α**)
    - This implies that (KB ∧ ¬**α**) is unsatisfiable, hence KB entails **α**