UNIVERSITÀ DEGLI STUDI DI NAPOLI
PARTHENOPE

Artificial Intelligence

Propositional Logic

LESSON 11

prof. Antonino Staiano

M.Sc. In ''Machine Learning e Big Data'' - University Parthenope of Naples

Syntax

- The syntax of propositional logic defines the sentences that can be formed
- Atomic sentences, either
  - A single proposition symbol that denotes a given proposition, e.g., P, Q …
  - A proposition symbol with a fixed meaning: *True* and *False*
- Complex sentences
  - Consist of atomic or complex sentences connected by logical connectives
  - Logical connectives
    - ∧ *(and)*
    - ∨ *(or)*
    - ¬ *(not)*
    - ⟹ *(implication)*
    - ⟺ *(if and only if / logical equivalence)*

Syntax

- A formal grammar in Backus-Naur Form (BNF):

$$
\begin{aligned}
\textit{Sentence} &\rightarrow \textit{AtomicSentence} \mid \textit{ComplexSentence} \\
\textit{AtomicSentence} &\rightarrow \textit{True} \mid \textit{False} \mid P \mid Q \mid R \mid \ldots \\
\textit{ComplexSentence} &\rightarrow (\textit{Sentence}) \\
&\mid \neg \textit{Sentence} \\
&\mid \textit{Sentence} \wedge \textit{Sentence} \\
&\mid \textit{Sentence} \vee \textit{Sentence} \\
&\mid \textit{Sentence} \Rightarrow \textit{Sentence} \\
&\mid \textit{Sentence} \Leftrightarrow \textit{Sentence} \\
\text{OPERATOR PRECEDENCE} &: \neg, \wedge, \vee, \Rightarrow, \Leftrightarrow
\end{aligned}
$$

Figure 7.7 A BNF (Backus–Naur Form) grammar of sentences in propositional logic, along with operator precedences, from highest to lowest.

Semantics

- Semantics of logical languages
  - Meaning of a sentence
    - its truth value with respect to a particular model
  - Model
    - A possible assignment of truth values to all proposition symbols that appear in the sentence
      - *P: It is raining*
      - Q: It is a Tuesday
      - {P = true, Q = false}
- Example
  - The sentence $P \wedge Q \Longrightarrow R$ has $2^3 = 8$ possible models
    - e.g., {P = *True*, Q = *False*, R = *True*}

Semantics

- The semantics for propositional logic must specify how to compute the truth value of any sentence, given a model
- Atomic sentences
  - *True* is true in every model
  - *False* is false in every model
  - The truth value of every proposition symbol (atomic sentence) must be specified in the model
- Complex sentences
  - The truth value is defined recursively as a function of the simpler sentences they are composed of, and the truth table of the logical connectives they contain

Connectives Truth Tables

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*

Example $\neg P \wedge (Q \vee R)$

P	Q	R	$(Q \vee R)$	$\neg P \wedge (Q \vee R)$
false	false	false	false	false
false	false	true	true	true
false	true	false	true	true
false	true	true	true	true
true	false	false	false	false
true	false	true	true	false
true	true	false	true	false
true	true	true	true	false

Propositional Logic and Natural Language

- The truth table of *and*, *or* and *not* is intuitive but captures only a subset of their meaning in natural language
- Example
  - *He fell **and** broke his leg*
    - ***and*** includes a temporal and causal relation
      - i.e., *He broke his leg and fell* does not have the same meaning
  - A tennis match can be won or lost
    - Disjunctive or, usually denoted in logic by $\oplus$

Implication ($\rightarrow$)

P	Q	$P \rightarrow Q$
false	false	true
false	true	true
true	false	false
true	true	true

Propositional Logic and Natural Language

- The truth table of P ⇒ Q may not fit your intuitive understanding of *P implies Q* or *if P then Q*
  - *5 is odd implies Tokyo is the capital of Japan*: meaningless in natural language, *true* in propositional logic (*P* ⇒ *Q* does not assume causation or relevance between *P* and *Q*)
  - *5 is even implies 10 is even*: can be considered false in natural language, but is *true* in propositional logic (*P* ⇒ *Q* is true whenever *P* is false)

Biconditional ($\leftrightarrow$)

P	Q	$P \leftrightarrow Q$
false	false	true
false	true	false
true	false	false
true	true	true

Propositional Logic and Natural Language

- Interpretation of biconditional $P \Leftrightarrow Q$
  - Biconditional is *true* whenever both implication sides are *true*
  - In English, this can be written as *P if and only if Q*
- Example
  - In the Wumpus world
    - "A square is breezy only if a neighboring square has a pit" is expressed as a biconditional

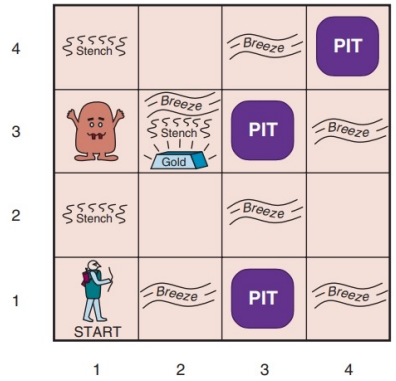
$$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$$

Example

1. Define a set of propositional symbols to represent the *Wumpus world*: the position of the agent, wumpus, pits, etc.
2. Define the model corresponding to the configuration below
3. Define the part of the initial agent's KB corresponding to its knowledge about the cave configuration in the figure below
4. Write a sentence for the propositions:

   (a) *If the wumpus is in room (3,1) then there is a stench in rooms (2,1), (4,1) and (3,2)*

   (b) *If there is a pit in room (1,3) then there is a breeze in rooms (1,2), (2,3) and (1,4)*



Example

Define a set of propositional symbols to represent the *Wumpus world*: the position of the agent, wumpus, pits, etc.

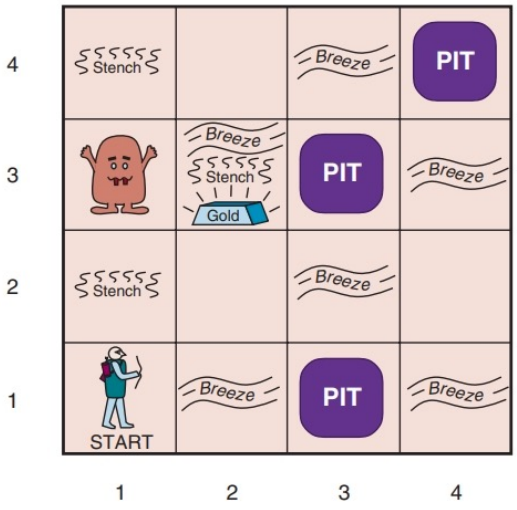
1. A possible choice of propositional symbols

$A_{1,1}$ ("the agent is in room (1,1)"), $A_{1,2}$, ..., $A_{4,4}$
$W_{1,1}$ ("the wumpus is in room (1,1)"), $W_{1,2}$, ..., $W_{4,4}$
$P_{1,1}$ ("there is a pit in room (1,1)"), $P_{1,2}$, ..., $P_{4,4}$
$G_{1,1}$ ("the gold is in room (1,1)"), $G_{1,2}$, ..., $G_{4,4}$
$B_{1,1}$ ("there is a breeze in room (1,1)"), $B_{1,2}$, ..., $B_{4,4}$
$S_{1,1}$ ("there is stench in room (1,1)"), $S_{1,2}$, ..., $S_{4,4}$

Example

Define the model corresponding to the configuration below

2. Model corresponding to the given configuration



- $A_{1,1}$ is true; $A_{1,2}$, $A_{1,3}$,... are false
- $W_{3,1}$ is true; $W_{1,1}$, $W_{1,2}$, ... are false
- $P_{1,3}$, $P_{3,3}$, $P_{4,4}$ are true; $P_{1,1}$, $P_{1,2}$, ... are false
- $G_{3,2}$ is true; $G_{1,1}$, $G_{1,2}$, ... are false
- $B_{1,2}$, $B_{1,4}$, ... are true; $B_{1,1}$, $B_{1,3}$, ... are false
- $S_{2,1}$, $S_{3,2}$, $B_{4,1}$ are true; $S_{1,1}$, $S_{1,2}$, ... are false

Example

3. What the agent knows in the starting configuration
   - I am in room (1,1) (starting position of the game)
   - There is no pit nor the Wumpus in room (1,1)
   - There is no gold in room (1,1)
   - I do not perceive a breeze nor a stench in room (1,1)
- The corresponding agent's KB in propositional logic (the set of sentences the agent believes to be true):
- $A_{1,1}$, $\neg A_{1,2}$, $\neg A_{1,3}$, . . . , $\neg A_{4,4}$ (16 sentences)
- $\neg W_{1,1}$, $\neg P_{1,1}$
- $\neg G_{1,1}$
- $\neg B_{1,1}$, $\neg S_{1,1}$

Example

Write a sentence for the proposition:

(a) *If the wumpus is in room (3,1) then there is a stench in rooms (2,1), (4,1) and (3,2)*

4. (a)

- One may think to translate the considered proposition using the implication connective

$$W_{3,1} \Rightarrow (S_{2,1} \wedge S_{4,1} \wedge S_{3,2})$$

- However, since there is only one wumpus, the opposite is also true:

$$(S_{2,1} \wedge S_{4,1} \wedge S_{3,2}) \Rightarrow W_{3,1}$$

- An equivalent, more concise way to express both sentences:

$$(S_{2,1} \wedge S_{4,1} \wedge S_{3,2}) \Leftrightarrow W_{3,1}$$

Example

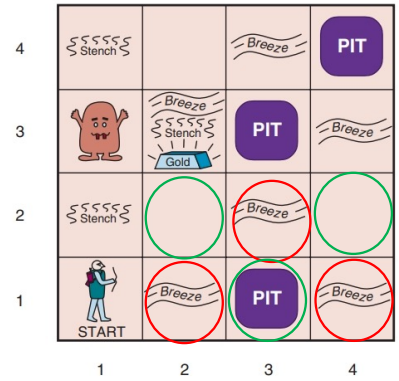
Write a sentence for the proposition:

(b) *If there is a pit in room (1,3) then there is a breeze in rooms (1,2), (2,3) and (1,4)*

4. (b)

- The presence of a pit in any square is a sufficient condition for the presence of a breeze in all the adjacent squares but is not a necessary condition
- For instance, a breeze in squares (1, 2), (2, 3), and (1, 4) can be caused by a pit in square (1, 3), but also by pits in squares (2, 2) and (2, 4)
- Therefore, the implication connective is the correct one for this proposition:

$$P_{1,3} \Rightarrow (B_{1,2} \wedge B_{2,3} \wedge B_{1,4})$$



Knowledge Base

- A set of sentences known by a knowledge-based agent it knows to be true
  - That is, some set of sentences in propositional logic that our AI knows about the world
    - We give that information to the AI and the AI would store it inside its knowledge base
  - Next, the AI uses that information in the KB to be able to draw conclusions about the rest of the world

Entailment

- To understand those conclusions, we use the notion of entailment

$$\alpha \vDash \beta$$

  - In every model in which sentence α is true, the sentence β is also true

- Ultimately, the entailment is that we're trying to encode into our computers

Entailment

- *If it didn't rain, Harry visited Hagrid today*
- *Harry visited Hagrid or Dumbledore today, but not both*
- *Harry visited Dumbledore today*
- *Harry did not visit Hagrid today*
- *It rained today*

Inference

- The process of deriving new sentences from old ones
  - Once we give
    - *If it didn't rain, Harry visited Hagrid today*
    - *Harry visited Hagrid or Dumbledore today, but not both*
    - *Harry visited Dumbledore today*
  - The AI can use an inference algorithm to figure that the following sentences must be true
    - *Harry did not visit Hagrid today*
    - *It rained today*

Example

- P: It is a Tuesday
- Q: It is raining
- R: Harry will go for a run

- KB: $(P \wedge \neg Q) \rightarrow R$ P $\neg Q$
- Inference: R

Inference Algorithms

- The inference algorithms answer the central question about entailment
  - Given some queries about the world, α
    - Does KB $\models \alpha$?
    - That is, using only the information inside of our KB, can we conclude that this sentence α is *true*?
- There are a couple of different algorithms for doing so
  - The simplest is model checking

Inference: Model checking

- Goal of inference
  - given a KB and a sentence α, deciding whether KB$\models \alpha$
- A simple inference algorithm: model checking
  - Application to propositional logic:
    - enumerate all possible models for sentences KB $\cup \{\alpha\}$
    - check whether α is *true* in every model in which KB is *true*
      - If so, KB entails α
      - Otherwise, KB does not entail α
- Implementation: truth tables

Model Checking: Example

- Determine whether {P ∨ Q, P ⇒ R, Q ⇒ R } ⊨ P ∨ R, using model checking

Propositional symbols			Premises			Conclusion
P	Q	R	$P \vee Q$	$P \Rightarrow R$	$Q \Rightarrow R$	$P \vee R$
false	false	false	false	true	true	false
false	false	true	false	true	true	true
false	true	false	true	true	false	false
false	true	true	true	true	true	true
true	false	false	true	false	true	true
true	false	true	true	true	true	true
true	true	false	true	false	false	true
true	true	true	true	true	true	true

- Yes, because the conclusion is *true* in every model in which the premises are *true* (grey rows)

Example

- P: It is a Tuesday Q: It is raining. R: Harry will go for a run.
- KB: (P ∧ ¬Q) → R P ¬Q
- Query: R

P	Q	R	KB
false	false	false	
false	false	true	
false	true	false	
false	true	true	
true	false	false	
true	false	true	
true	true	false	
true	true	true	

Example

- P: It is a Tuesday Q: It is raining. R: Harry will go for a run.
- KB: $(P \wedge \neg Q) \rightarrow R$ P $\neg Q$
- Query: R

P	Q	R	KB
false	false	false	false
false	false	true	false
false	true	false	false
false	true	true	false
true	false	false	false
true	false	true	true
true	true	false	false
true	true	true	false

Example

- P: It is a Tuesday Q: It is raining. R: Harry will go for a run
- KB: (P ∧ ¬Q) → R P ¬Q
- Query: R

P	Q	R	KB
false	false	false	false
false	false	true	false
false	true	false	false
false	true	true	false
true	false	false	false
true	false	true	true
true	true	false	false
true	true	true	false

Model Checking in Python

- Encoding the notation of
  - Propositional symbols
  - Connectives and, or, not, implication
- We consider a written in advance logic library
  - logic.py

Encoding symbols and the KB

```
from logic import *

# Create new classes, each having a name, or a symbol, representing each proposition.

rain = Symbol("rain")      # It is raining.

hagrid = Symbol("hagrid") # Harry visited Hagrid

dumbledore = Symbol("dumbledore") # Harry visited Dumbledore

# Save sentences into the KB

knowledge = And( # Starting from the "And" logical connective, becasue each proposition
          represents knowledge that we know to be true.

   Implication(Not(rain), hagrid), # ¬(It is raining) → (Harry visited Hagrid)

   Or(hagrid, dumbledore), # (Harry visited Hagrid) ∨ (Harry visited Dumbledore).

   Not(And(hagrid, dumbledore)), # ¬(Harry visited Hagrid ∧ Harry visited Dumbledore)
                                 i.e. Harry did not visit both Hagrid and Dumbledore.

   dumbledore # Harry visited Dumbledore. Note that while previous propositions
              contained multiple symbols with connectors, this is a proposition
              consisting of one symbol. This means that we take as a fact that, in this
              KB, Harry visited Dumbledore.

   )
```

Model Checking in practice

- To run the Model Checking algorithm, the following information is needed:
  - Knowledge Base, which will be used to draw inferences
  - A query, or the proposition that we are interested in whether it is entailed by the KB
  - Symbols, a list of all the symbols (or atomic propositions) used (in our case, these are rain, hagrid, and dumbledore)
  - Model, an assignment of truth and false values to symbols

Model checking

```
def check_all(knowledge, query, symbols, model): # If model has an assignment for each
symbol # (The logic below might be a little confusing: we start with a list of symbols.
The function is recursive, and every time it calls itself it pops one symbol from the
symbols list and generates models from it. Thus, when the symbols list is empty, we know
that we finished generating models with every possible truth assignment of symbols.)

if not symbols: # If knowledge base is true in model, then query must also be true

    if knowledge.evaluate(model):

      return query.evaluate(model)

      return True

else: # Choose one of the remaining unused symbols

 remaining = symbols.copy() p = remaining.pop() # Create a model where the symbol is
true

model_true = model.copy()

model_true[p] = True # Create a model where the symbol is false

model_false = model.copy() model_false[p] = False

# Ensure entailment holds in both models

return(check_all(knowledge, query, remaining, model_true) and check_all(knowledge,
query, remaining, model_false))
```

Knowledge Engineering

- This logic can be applied to a number of different types of problems
- Having a problem where some logical deduction can be used to solve it
  - What propositional symbols you might need in order to represent that information
  - What statements and propositional logic to use to encode that information that one knows
- Knowledge engineering
  - the process of trying to take a problem and figure out what propositional symbols to use, or how to represent it logically

Logic Puzzles

- Puzzle our way through the game trying to figure something out
- Example
  - *Gilderoy*, *Minerva*, *Pomona*, and *Severus* each belong to a different one of the four houses:
    - *Gryffindor*, *Hufflepuff*, ***Ravenclaw***, and *Slytherin* House
  - *Gilderoy* belongs to *Gryffindor* or ***Ravenclaw***
  - *Pomona* does not belong to *Slytherin*
  - *Minerva* belongs to *Gryffindor*
- Draw some conclusion about which person should be assigned to which house

Logic Puzzles

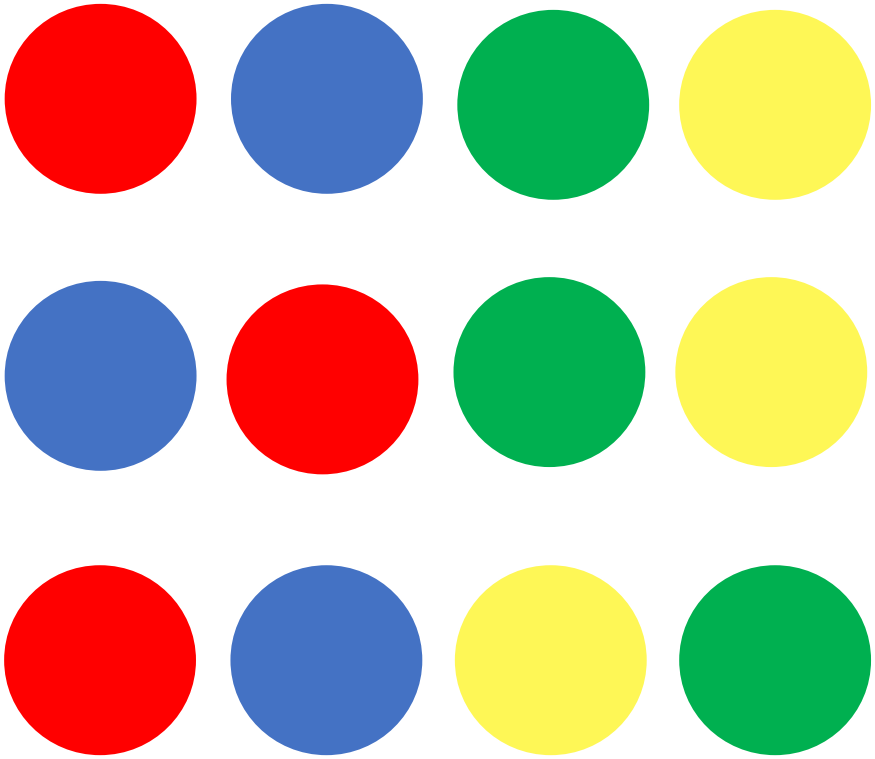
- Propositional symbols
  - *GilderoyGryffindor*
  - *GilderoyHufflepuff*
  - *GilderoyRavenclaw*
  - *GolderoySlytherin*
  - *MinervaGryffindor*
  - *MinervaHufflepuff*
  - *MinervaRavenclaw*
  - *MinervaSlytherin*
  - *PomonaGryffindor*
  - *PomonaHufflepuff*
  - *PomonaRavenclaw*
  - *PomonaSlytherin*
  - *SeverusGryffindor*
  - *SeverusHufflepuff*
  - *SeverusRavenclaw*
  - *SeverusSlytherin*

Logic Puzzles

- Using this type of knowledge, we can think about what types of logical sentences we can say about the puzzle

  - $PomonaSlytherin \Rightarrow \neg PomonaHufflepuff$
  - $MinervaRavenclaw \Rightarrow \neg GilderoyRavenclaw$
  - $GilderoyGryffindor \vee GilderoyRavenclaw$

Mastermind



Properties of model checking

- Soundness
  - It directly implements the definition of entailment
- Completeness
  - It works for any (finite) KB and α, and the corresponding set of models is finite
- Computational complexity
  - $O(2^n)$, where n is the number of proposition symbols appearing in KB and α
  - Infeasible when the number of proposition symbols is high
- Example
  - In the example of the Wumpus world, 96 proposition symbols have been introduced:
    - the corresponding truth table is made up of $2^{96} \approx 10^{28}$ rows