Hano Strydom

Hanostrydom8@gmail.com

Abstract

These notes cover all the work done in class.  
Artificial Intelligence. A modern Approach (Fourth Edition)

ITRI 626 - Notes

Artificial Intelligence

Contents

[Table of Figures 1](#_Toc112582044)

[Tables 2](#_Toc112582045)

[Glossary 2](#_Toc112582046)

[Chapter 7 – Logical Agents 4](#_Toc112582047)

[Introduction 4](#_Toc112582048)

[7.1. Knowledge-Based Agents 5](#_Toc112582049)

[7.2. The Wumpus World 7](#_Toc112582050)

[7.3. Logic 10](#_Toc112582051)

[7.4. Propositional Logic: A very Simple Logic 15](#_Toc112582052)

[7.4.1. Syntax 15](#_Toc112582053)

[7.4.2. Semantics 16](#_Toc112582054)

[7.4.3. A Simple Knowledge Base 18](#_Toc112582055)

[7.4.4. A Simple Inference Procedure 19](#_Toc112582056)

[7.5. Propositional Theorem Proving 21](#_Toc112582057)

[7.5.1. Inference and Proofs 23](#_Toc112582058)

[7.5.2. Proof by resolution 26](#_Toc112582059)

[7.5.3. Horn Clauses and Definite Clauses 30](#_Toc112582060)

[7.5.4. Forward and Backward Chaining 32](#_Toc112582061)

# Table of Figures

[Figure 1 - Typical Wumpus World 8](#_Toc112580587)

[Figure 2 - Symbols in form of knowledge representation language 9](#_Toc112580588)

[Figure 3 - Possible Wumpus models 12](file:///C:\Users\hanos\OneDrive\Desktop\HonsNotes2022\AI\AI_NOTES.docx#_Toc112580589)

[Figure 4 - Wumpus Knowledge Base truth table 19](#_Toc112580590)

[Figure 5 - Standard Logical Equivalences 21](#_Toc112580591)

[Figure 6 - Full resolution rule 27](#_Toc112580592)

# Tables

[Table 1 - Soundness and Completeness 13](#_Toc112580582)

[Table 2 - Five Common Connectives 15](#_Toc112580583)

[Table 3 - Rules truth table 17](#_Toc112580584)

[Table 4 - P and Q truth table 17](file:///C:\Users\hanos\OneDrive\Desktop\HonsNotes2022\AI\AI_NOTES.docx#_Toc112580585)

# Glossary

|  |  |
| --- | --- |
| **Agent** | An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators |
| **Knowledge-based agents** | Use a process of **reasoning** over an internal **representation** of knowledge to decide what actions to take |
| **Logic** | A general class of representations to support knowledge-based agents |
| **Knowledge base / KB** | The knowledge base is a set of **sentences** |
| **Sentence** | Technical term. Related, but not identical to sentences in English and other natural languages |
| **Axiom** | When a sentence is taken as being given without being derived from other sentences |
| **Inference** | Deriving new sentences from the old sentences |
| **Knowledge Level** | Specify what the agent knows and what its goals are, in order to determine behaviour |
| **Implementation Level** | How the taxi works |
| **Declarative Approach to system building** | Starting with empty knowledge base, the agent designer can TELL sentences one by one until the agent knows how to operate in its environment |
| **Procedural Approach** | Encodes desired behaviors directly as program code |
| **Syntax** | Defines sentences of the language |
| **Semantics** | Defines the “meaning” of sentences  It is the truth of a sentence in each possible world  Defines the rules for determining the truth of a sentence with respect to a particular model |
| **Models** | Formal structured worlds out of which truth can be evaluated |
| **Possible Worlds** | Might be thought of as (potentially) real environments that the agent might or might not be in.  Models are mathematical abstractions, each of which has a fixed truth value for every relevant sentence |
| **Entailment** | Entailment means one sentence follows logically from another  Entailment is a relationship between sentences (i.e. syntax) based on semantics |
| **Sound or Truth Preserving** | An inference algorithm that derives only entailed sentences |
| **Grounding** | The connection between logical reasoning processes and the real environment in which the agent exists |
| **Learning** | Sentence construction process that produces general rules. |
| **Complex Sentences** | Constructed from simpler sentences using parentheses and operators called **logical** **connectives**. |
| **Model checking** | Enumerating models and showing that the sentence must hold in all models |
| **Logical equivalence** | Two sentences α and b are logically equivalent if they are true in the same set of models ( α ≡ b) |
| **Validity** | A sentence is valid if it is true in *all* models |
| **Proof** | A chain of conclusions that leads to a desired goal  Sequence of applications of inference rules |
| **Literal** | Proposition symbol or negation of a proposition symbol |
| **Clause** | Number of Literals that is connected with disjunctions |
| **Factoring** | The removal of multiple copies of literals |
| **Conjunctive normal form** | A sentence expressed as a conjunction of clauses is said to be in: |
| **Resolution Closure - RC(S)** | Set of all clauses derivable by repeated application of the resolution rule to clauses in S or their derivatives |
| **definite clause** | A disjunction of literals of which exactly one is positive |
| **Horn Clauses** | Disjunction of literals of which *at most one is positive* |
| **Forward-chaining algorithm** | Determine if single propositional symbol *Q* follows logically from knowledge base of Horn clauses |
| **Data-driven reasoning** | Reasoning in which the focus of attention starts with the known data |
| **Backward-chaining algorithm** | Works backward from the query |
| **Goal-directed** **reasoning** | Useful for answering specific questions such as “What shall I do now?” and “Where are my keys?” |

# Chapter 7 – Logical Agents

## Introduction

* **Agent**
  + An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators
* **Knowledge-based agents**
  + Use a process of **reasoning** over an internal **representation** of knowledge to decide what actions to take.
* **Problem-solving agents**
  + Know things in a very limited, inflexible sense.
  + They know what actions are available and what the results of performing a specific action from a specific state will be, but don’t know general facts.
  + **The atomic representations** used by problem-solving agents are also very **limiting**
* We develop **logic** as a general class of representations to support knowledge-based agents
* Consider representation of knowledge and reasoning process that brings knowledge to life
* People know things and perform reasoning
* Knowledge of problem-solving agents is very specific and inflexible
* Knowledge and reasoning are important in dealing with partially observable environments
* Sometimes hidden aspects of states must be inferred
* Knowledge-based agents are flexible

## 7.1. Knowledge-Based Agents

* Central component of knowledge-based agent:
  + Knowledge base / KB
    - Knowledge base is a set of **sentences.**
    - Each **sentence** is expressed in a language called a **knowledge representation language** and represents some assertion about the world
* **Axiom**
  + When a sentence is taken as being given without being derived from other sentences.
* **Knowledge Base operations:**
  + Add new sentences to the knowledge base
    - TELL
  + Query what is known:
    - ASK
* Both these operations may involve **inference**
  + **Inference**
    - Deriving new sentences from old sentences
  + Inference must obey the requirement that when one ASK a question of the knowledge base, the answer should follow from what has been told (or TELL) to the knowledge base previously
  + **Answers should follow logically from the knowledge base**
* Agents maintain a **knowledge base**, which may initially contain some **background knowledge**
* **When an Agent program is called, three things happen:**
  1. TELL the knowledge base what it perceives
  2. ASKS the knowledge base what action it should perform
     + In the process of answering this query, extensive reasoning may be done about the current state of the world, about the outcomes of possible action sequences and so on.
  3. Agent Program TELLS the knowledge base which action was chosen and returns the action so that it can be executed
* Details of representation language are hidden inside three functions that implement the interface between:
  + Sensors and Actuators on the one side
  + Core Representation and Reasoning
* Because of the definitions of TELL and ASK, however, the knowledge-based agent is not an arbitrary program for calculating actions. It is amenable to a description at the **knowledge level**, where we need specify only what the agent knows and what its goals are, in order to determine its behavior
* **Knowledge Level:**
  + Specify what the agent knows and what its goals are, in order to determine behavior
  + Example:
    - Goal: Taking passengers from San Francisco to Marin Country
    - Know Golden Gate Bridge is the only link between the locations
* **Implementation Level:**
  + How the taxi works
* **A knowledge-based agent can be built simply by TELLing it what it needs to know.**
* **Declarative Approach** to system building**:**
  + Starting with empty knowledge base, the agent designer can TELL sentences one by one until the agent knows how to operate in its environment
* **Procedural Approach:**
  + Encodes desired behaviors directly as program code
* In the 1970s and 1980s, advocates of the two approaches engaged in heated debates
* We now understand that a **successful agent** often **combines both declarative and procedural elements in its design** and that **declarative knowledge can often be compiled into more efficient procedural code**
* **A learning agent can be fully autonomous**

## 7.2. The Wumpus World

* What is the Wumpus World
  + A cave consisting of rooms connected by passageways
  + Wumpus lurks in the caves and eats anyone who enters the room
  + Wumpus can be shot with an arrow
  + Agent only has one arrow
  + Some rooms have bottomless pits that trap the agent
  + Must find a heap of gold / kill Wumpus
* **Task Environment (PEAS)**
  + **Performance Measure**
    - +1000 climbing out the cave with the gold
    - -1000 for falling into a pit / being eaten by Wumpus
    - -1 for each action taken
    - -10 for using the arrow
    - Game ends when the agent dies or climbs out of the cave
  + **Environment**
    - **4 x 4** grid of rooms, with walls surrounding the grid
    - **Blocks are read [column, row]**
    - Agents start in square [1,1], facing to the east
    - Location of gold and Wumpus are chosen randomly, with a uniform distribution, from squares other than the start square.
    - Each square other than the start square, can be a pit with a probability with 0.2
  + **Actuators**
    - Move forward
    - Turn Left
    - Turn Right
    - Grab Gold
    - Shoot arrow
    - Climb
  + **Sensors**
    - Stench – Squares directly adjacent to the Wumpus
    - Breeze - Squares directly adjacent to a pit
    - Glitter – A square with the gold
    - Bump – When an agent walks into a wall
    - Scream – When the Wumpus is killed
      * Example: [stench, breeze, none, none, none]

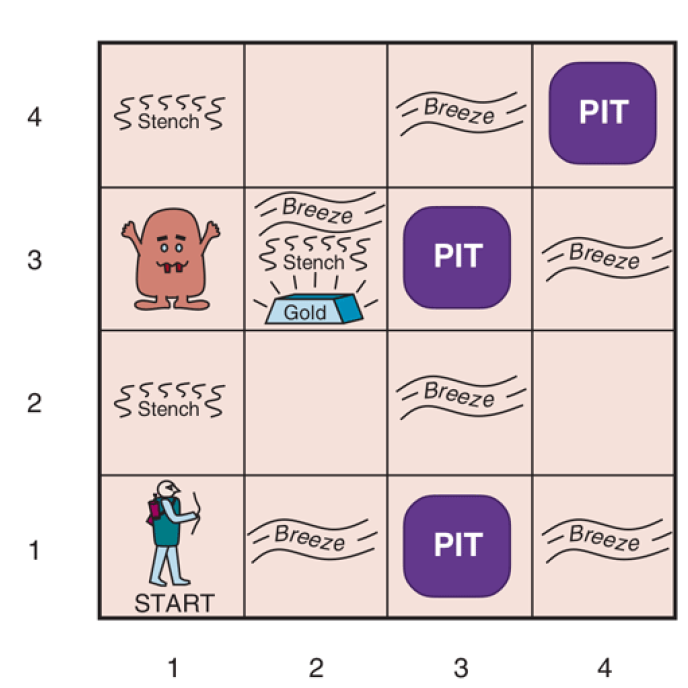
****

Figure 1 - Typical Wumpus World

* **Characteristic of Wumpus Environment**
  + Deterministic, discrete, static, and single agent, because the Wumpus doesn’t move
  + Sequential, because rewards may come only after many actions are taken
  + Partially observable, because some aspects of the state are not directly perceivable
    - The agent’s location, the Wumpus’s state of health, and the availability of an arrow
  + Locations of pits and Wumpus is unobserved, transition model of environment is known, and finding location of pits completes the agent’s knowledge of the state.
  + Alternatively, we could say that the transition model itself is unknown because the agent doesn’t know which Forward actions are fatal—in which case, discovering the locations of pits and Wumpus completes the agent’s knowledge of the transition model

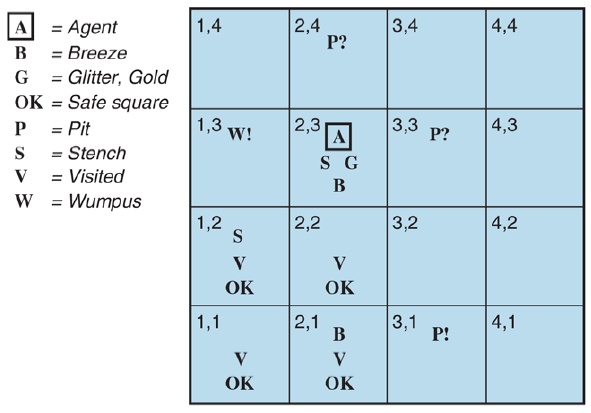
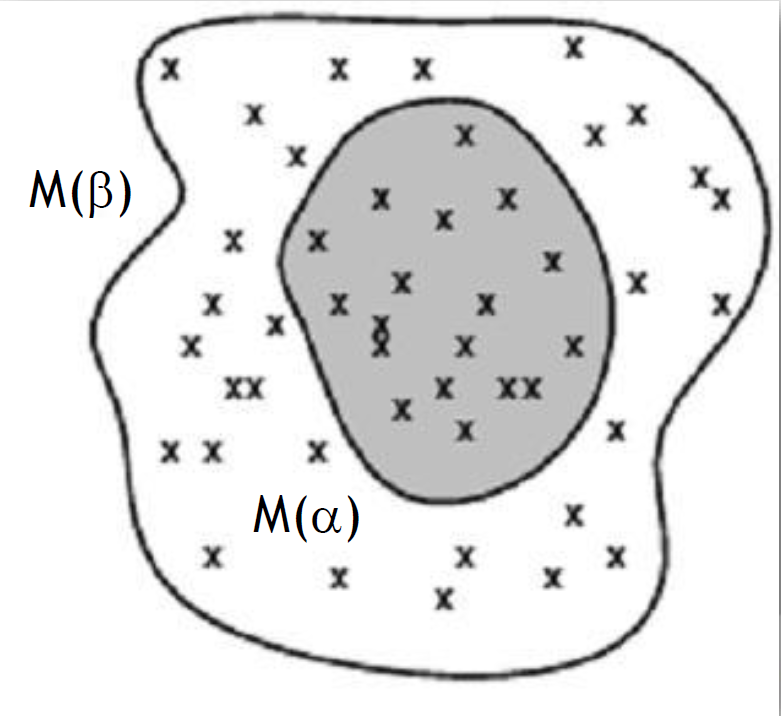


Figure 2 - Symbols in form of knowledge representation language

* Note that in each case for which the **agent draws a conclusion** from **the available information**, that conclusion is **guaranteed** to be **correct** if the **available information is correct**

## 7.3. Logic

* Knowledge base consists of **sentences**
* These **sentences** are expressed according to the syntax of representation language
  + It specifies all the sentences that are well formed.
  + 101x + 59 y = 250 **is well formed**
  + x50 + - 90z = y + = **is not well-formed**
* Logics are formal languages to express information and reach conclusions
* In standard logic, every sentence must be either true or false in each possible world
* **Syntax**
  + Defines sentences of the language
* **Semantics**
  + Defines the “meaning” of sentences
  + It is the **truth** of a sentence in **each** **possible** **world**
* **Language of Arithmetic:**
  + x + 2 ≥ y is a valid sentence
  + x + 2 ≥ { } is an invalid sentence
  + x + 2 ≥ y, true in a world where x = 7 and y = 1
  + x + 2 ≥ y, false in a world where x = 0 and y = 6
* Logicians think in terms of models
* **Models**
  + Formal structured worlds out of which truth can be evaluated
* **Possible Worlds**
  + Might be thought of as (potentially) real environments that the agent might or might not be in.
  + Models are mathematical abstractions, each of which has a fixed truth value for every relevant sentence.
  + E.g.
    - x Men and y Woman are sitting at a table playing bridges. The sentence x+y=4 is true when there are four people total.
* If a sentence is **true** in Model m, we say that:
  + **M satisfies α**
  + **M is a model of α**
  + **We say it a model of sentence α if α is true in m**
  + M(α) is the set of all models of α
* **Entailment**
  + Entailment means one sentence follows logically from another:
  + α ⊧ 
    - Sentence  follows logically from sentence α if and only if  is true in all models (worlds) where α is true
    - α ⊧  if and only if M(α) ⊆ M()
    - Note the direction of ⊆: if α ⊧ then α is a stronger assertion than : it rules out more possible worlds
  + Entailment is a relationship between sentences (i.e., syntax) based on semantics



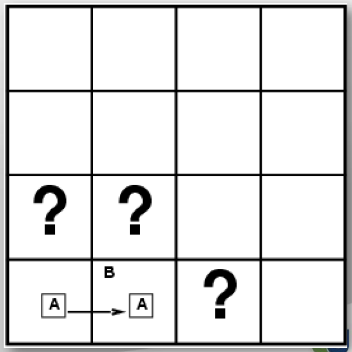
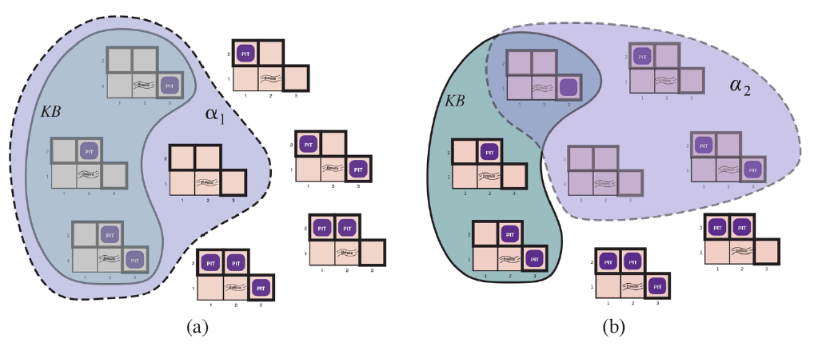
****

Figure 3 - Possible Wumpus models

* **Wumpus entailment example** 
  + The agent has detected nothing in [1,1] and a breeze in [2,1].
  + These percepts, combined with the agent’s knowledge of the rules of the Wumpus world, constitute the KB
  + The KB is false in models that contradict what the agent knows—for example, the KB is false in any model in which [1,2] contains a pit because there is no breeze in [1,1].
  + There are in fact just three models in which the KB is true
    - There can be a pit in [2,2], [3,1] or both
  + Let’s consider 2 possible conclusions:
    - α1 = “There is no put in [1,2]”
    - α2 = “There is no put in [2,2]”
  + Looking at Figure 3, we can see that every model in which KB is true, α1 is also true
    - Hence, KB ⊨ α1: there is no pit in [1,2].
  + We can see that in some models in which KB is true, α2 is false
    - Hence, KB does not entail α2
    - This means the agent cannot conclude that there is or is not a pit in [2,2]
  + The inference algorithm illustrated in Figure 3 is called **model checking**
    - It enumerates all possible models to check that α is true in all models in which KB is true, that is, that M(KB) ⊆ M(α)



* In understanding entailment and inference, it might help to think of the set of all consequences of **KB** as a haystack and of **α** as a needle
* Entailment is like the needle being in the haystack; inference is like finding it
* If an inference algorithm can derive from **KB**, we write:
  + KB ⊢ *i*α
    - Pronounced: “ α is derived from **KB** by *i*” or “*i* derives **α** from **KB**.”
* **Sound or Truth Preserving**
  + An inference algorithm that derives only entailed sentences
* Soundness is a highly desirable property.
  + An unsound inference procedure essentially makes things up as it goes along. It announces the discovery of nonexistent needles
* **Completeness**
  + Inference algorithm is complete if it can derive any sentence that is entailed
* We have described a reasoning process whose conclusions are guaranteed to be true in any world in which the premises are true; in particular, *if KB is true in the real world, then any sentence derived from KB by a sound inference procedure is also true in the real world*.

Table 1 - Soundness and Completeness

|  |  |
| --- | --- |
| KB ⊢ *i*α means sentence α can be derived from KB by procedure *i* | |
| Soundness | Procedure *i* is sound if KB ⊢ *i*α, then KB ⊨ α |
| Completeness | Procedure αis complete if KB ⊧ α, then KB ⊦ *i*α |



So, while an inference process operates on “syntax”—internal physical configurations such as bits in registers or patterns of electrical blips in brains—the process corresponds to the real-world relationship whereby some aspect of the real world is the case by virtue of other aspects of the real world being the case. This correspondence between world and representation is illustrated in the figure above

* **Grounding**
  + Connection between logical reasoning processes and the real environment in which the agent exists
    - In particular, how do we know that KB is true in the real world?
      * Simple answer: The agent’s sensors create the connection
  + The agent program creates a suitable sentence whenever there is a smell. Then, whenever that sentence is in the knowledge base, it is true in the real world. Thus, the meaning and truth of percept sentences are defined by the processes of sensing and sentence construction that produce them
  + What about the rest of the agent’s knowledge, such as its belief that Wumpuses cause smells in adjacent squares?
    - This is not a direct representation of a single percept, but a general rule—derived, perhaps, from perceptual experience but not identical to a statement of that experience. General rules like this are produced by a sentence construction process called **learning**, which is the subject of Part V.
  + Learning is fallible
    - It could be the case that Wumpuses cause smells except on February 29 in leap years, which is when they take their baths. Thus, KB may not be true in the real world, but with good learning procedures, there is reason for optimism

## 7.4. Propositional Logic: A very Simple Logic

### 7.4.1. Syntax

* Syntax of propositional logic defines the allowable sentences
* **Atomic sentences** consist of a single **propositional symbol**
* Each symbol stands for a proposition that can be true or false.
  + Symbols are uppercase and can have subscripts and face east.
    - E.g., P, Q, Ri,j, W1,3
  + Names are arbitrary, but often chosen to have mnemonic values
    - W1,3 for Wumpus in [1,3]
* There are two propositional symbols with fixed meanings:
  + *True* is the always-true proposition
  + *False* is the always-false proposition
* **Complex** **Sentences**
  + Constructed from simpler sentences using parentheses and operators called **logical** **connectives**.

Table 2 - Five Common Connectives

|  |  |
| --- | --- |
| ¬ (not) | * If S is a sentence, then ¬ S is a sentence * A sentence such as ¬W1,3 is called the **negation** of W1,3. A **literal** is either an atomic sentence (a **positive literal**) or a negated atomic sentence (a **negative literal**) |
| ∧ (and) | * If S1 and S2 are sentences, then S1 ∧ S2 is a sentence (conjunction) * A sentence whose main connective is ∧, such as W1,3 ∧ P3,1, is called a **conjunction**; its parts are the **conjuncts**. |
| ∨ (or) | * If S1 and S2 are sentences, then S1 ∨ S2 is a sentence (disjunction) * A sentence whose main connective is ∨, such as (W1,3 ∧ P3,1) ∨ W2,2, is a **disjunction**; its parts are W1.3 —in this example, (W1,3 ∧ P3,1) and W2,2. |
| ⇒ (implies) | * If S1 and S2 are sentences, then S1 ⇒ S2 is a sentence (implication) * A sentence such as (W1,3 ∧ P3,1) ⇒ ¬W2,2 is called an **implication** (or conditional). Its **premise** or **antecedent** is (W1,3 ∧ P3,1), and its **conclusion** or **consequent** is ¬W2,2. Implications are also known as **rules** or **if–then** statements. The implication symbol is sometimes written in other books ⊃ as or →. |
| ⇔ (if and only if) | * If S1 and S2 are sentences, then S1 ⇔ S2 is a sentence (if and only if) (biconditional) * The sentence W1,3 ⇔ ¬W2,2 is a **biconditional**. |

* Syntax Exercise | *R* for Rain, *S* for Snow
  + It is either raining or snowing
    - R ∨ S
  + It is both raining or snowing
    - R ∧ S
  + It is raining, but it is not snowing
    - R ∧ ¬S
  + It is not both raining and snowing
    - ¬ (R ∧ S)
    - ¬R ∧ ¬S
  + If it is not raining, then it is snowing
    - ¬R ⇒ S
  + It is raining if and only if it is not snowing
    - R ⇔ ¬ S

### 7.4.2. Semantics

* Semantics
  + Defines the rules for determining the truth of a sentence with respect to a particular model
* In propositional logic, each model specifies the **true value** (true or false) of each proposition symbol
* E.g.,
  + If the sentences in the knowledge base make use of the proposition symbols P1,2, P2,2, and P3,1 then one possible model is:
    - m1 = {P1,2 = *false*, P2,2 = *false*, P3,1 = *true*}
* The semantics for propositional logic must specify how to compute the truth value of *any* sentence, given a model
  + This is done recursively
* All sentences are constructed from atomic sentences and the five connectives
  + We need to specify how to compute the truth of atomic sentences and how to compute the truth of sentences formed with each of the five connectives
* Atomic Sentences:
  + *True* is true in every model and *False* is false in every model.
  + The truth value of every other proposition symbol must be specified directly in the model.
    - For example, in the model m1 given earlier, P1,2 is false.
* Complex Sentences:
  + 5 Rules to evaluate the truth, which hold for any subsentences S1 and S2 (atomic or complex) in any model *m*: T=textbook; S=slides; B=both 
    - (1B) ¬S is true if and only if S is false
    - (2B) S1∧S2 is true if and only if S1 is true and S2 is true
    - (3B) S1∨S2 is true if and only if S1 is true or S2 is true
    - (4S) S1⇒S2 is true if and only if S1 is false or S2 is true
    - (4S) S1⇒S2 is false if and only if S1 is true and S2 is false
    - (5S) S1⇔S2 is true if and only if S1⇒S2 is true and S2⇒S1 is true
    - (5T) S1⇔S2 is true if and only if S1 and S1 are both true of both false
* The rules above can also be expressed with **Truth Tables**
* Truth tables for the five connectives are given in Table 3

Table 3 - Rules truth table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| S1 | S2 | ¬S1 | S1 ∧ S2 | S1 ∨ S2 | S1 ⇒ S2 | S1 ⇔ S2 |
| 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 |

* Truth value of any sentence *s* can be computed with respect to any model *m* by a simple recursive evaluation (Simple recursive process evaluate an arbitrary sentence)
  + E.g.,

Table 4 - P and Q truth table

* + - Table

      Description automatically generated¬P1,2 ∧ (P2,2 ∨ P3,1 )

= true ∧ (true ∨ false)

= true ∧ true

= true

* The truth tables for “and,” “or,” and “not” are in close accord with our intuitions about the English words.
* Main point of possible confusion is that P ∨ Q is true when P is true or Q is true or both
* “Exclusive or” (“xor” for short), yields false when both disjuncts are true
  + no consensus on the symbol for “exclusive or”.
  + Some choices are ∨ or ≠ or ⊕.
* The truth table for ⇒ may not quite fit one’s intuitive understanding of “P implies Q” or “if P then Q.”
  + Propositional logic does not require any relation of *causation* or *relevance* between P and Q.
    - “5 is odd implies Tokyo is the capital of Japan” is a true sentence of propositional logic
  + Another point of confusion is that any implication is true whenever its antecedent is false
    - “5 is even implies Sam is smart” is true, regardless of whether Sam is smart.
* Biconditional
  + P ⇔ Q is true whenever both P ⇒ Q is true and Q ⇒ P are true
  + In English:
    - P if and only if Q
  + B1,1 ⇔ (P1,2 ∨ P2,1)
    - There is a breeze in [1,1] if and only if there is either a Pit in [1,2] or [2,1]

### 7.4.3. A Simple Knowledge Base

* Px,y is true if there is a pit in [x,y].
* Wx,y is true if there is a wumpus in [x,y], dead or alive
* Bx,y is true if there is a breeze in [x,y].
* Sx,y is true if there is a stench in [x,y].
* Lx,y is true if the agent is in location [x,y].
* We label each sentence Ri
  + R2 : B1,1 ⇔ (P1,2 ∨ P2,1) .
  + R3 : B2,1 ⇔ (P1,1 ∨ P2,2 ∨ P3,1)

### 7.4.4. A Simple Inference Procedure

* Our goal now is to decide whether KB ⊨ α for some sentence α
* Our first algorithm for inference is a model-checking approach that is a direct implementation of the definition of entailment:
  + enumerate the models, and check that α is true in every model in which *KB* is true.
* Models are assignments of *true* or *false* to every proposition symbol.
* Wumpus-world example, the relevant proposition symbols are:
  + B1,1 ,B2,1 ,P1,1 ,P1,2 ,P2,1 ,P2,2 and P3,1
  + That means there are 27 = 128 possible models
  + In those three models, P1,2 is false, hence there is no pit in [1,2]
  + P2,2 is true in two of the three models and false in one, so we cannot yet tell whether there is a pit in [2,2].



Figure 4 - Wumpus Knowledge Base truth table



* Let Pi,j be true if there is a pit in [i, j]
* Let Bi,j be true if there is a breeze in [i,j]
* R1 : ¬P1,1
* R2 : B1,1 ⇔ (P1,2 ∨ P2,1)
* R3 : B2,1 ⇔ (P1,1 ∨ P2,2 ∨ P3,1)
* R4 : ¬B1,1
* R5 : B2,1
* KB = R1 ∧ R2 ∧ R3 ∧ R4 ∧ R5
* KB ⊧ α1where α1= “[1,2] has no pit”



* If KB and α contain *n* symbols in all, then there are models 2n models
  + Time Complexity: O(2n)
  + Space Complexity: O(n)

## 7.5. Propositional Theorem Proving

* How to determine entailment:
  + Model Checking
    - Enumerating models and showing that the sentence must hold in all models
  + Theorem Proving
    - Applying rules of inference directly to the sentences in our knowledge base to construct a proof of the desired sentence without consulting models
* If the *number of models is large* but the *length of the proof is short*, then *theorem proving can be more efficient than model checking*
* Before we plunge into the details of theorem-proving algorithms, we will need some
* additional concepts related to entailment
  + Logical Equivalence
  + Validity
* **Logical equivalence:**
  + Two sentences α and  are logically equivalent if they are true in the same set of models
  + Written as: α ≡ 
    - Note that ≡ is used to make claims about sentences, while ⇔ is used as part of a sentence
* These equivalences play much the same role in logic as arithmetic identities do in ordinary mathematics
* **Alternative definition of equivalence:**
  + Any two sentences α and  are equivalent if and only if each of them entails the other:
    - α ≡ β if and only if α ⊨ β and β ⊨ α .

Figure 5 - Standard Logical Equivalences

* **Validity**
  + A sentence is valid if it is true in *all* models
    - P ∨ ¬P is Valid
    - A ⇒ A is valid
    - (A ∧ (A ⇒ B)) ⇒ B
  + Valid sentences is also known as **Tautologies**
    - They are *necessarily* true.
* Validity is connected to inference by the **Deduction Theorem**:
  + α╞ β if and only if (α ⇒ ) is valid
* **Satisfiable vs Unsatisfiable**
  + Satisfiability can be checked by enumerating the possible models until one is found that satisfies the sentence
    - A sentence is satisfiable if it is true in a model (at least 1)
      * R1 ∧ R2 ∧ R3 ∧ R4 ∧ R5
    - A sentence is unsatisfiable if it is true in no models
      * For example: A ∧ ¬A
  + Satisfiability is connected to inference by the following:
    - α╞ β if and only if (α ⇒ ¬) is unsatisfiable
  + Problem of determining the satisfiability of sentences in propositional logic:
    - **SAT** problem
      * The first problem proved to be NP complete
* Validity and satisfiability are connected:
  + α is valid if and only if ¬α is unsatisfiable;
  + Contrapositively, α is satisfiable if and only if ¬α is not valid
  + α╞ β if and only if the sentence (α ∧ ¬ β) is unsatisfiable.
* Proving β from α by checking the unsatisfiability of corresponds exactly to the standard mathematical proof technique of ***reductio ad absurdum*** (literally, “reduction to an absurd thing”).
  + Also called:
    - “Proof by **refutation”** and “Proof by **contradiction”**
  + One assumes a sentence β to be false and shows that this leads to a contradiction with known axioms α.
  + This contradiction is exactly what is meant by saying that the sentence (α ∧ ¬ β) is unsatisfiable.

### 7.5.1. Inference and Proofs

* **Inference rules** can be applied to derive a **proof**
* **Proof**
  + A chain of conclusions that leads to a desired goal.
  + Sequence of applications of inference rules
* **Inference Rules:**
  + **Standard pattern of inference**
  + **Searching for proofs an alternative to enumerating models (truth table)**
  + Search can go forward from initial knowledge base or
  + Search can go backward from goal sentence
  + Finding a proof can sometimes be highly efficient
  + Best known rule:
    - **Modus Ponens** (Latin for *mode that affirms*)
    - This means whenever any sentence of the form α ⇒  and α are given, then sentence can be inferred
    - (WumpusAhead ∧ WumpusAlive) ⇒ Shoot
    - If (WumpusAhead ∧ WumpusAlive) are given, then Shoot can be inferred
  + Another rule:
    - **And-Elimination**
    - or
    - This means that from a conjunction, any of the conjuncts can be inferred
    - From (WumpusAhead ∧ WumpusAlive), WumpusAlive can be inferred
* Truth values for α and shows **that Modus Ponens** and **And-Elimination** are **sound**
* All the logical equivalences of Figure 5 can be used as inferences rules
  + E.g.,
    - and

**Wumpus Example:**

**We start with the knowledge base below, and we want to show how to prove ¬P1,2, that there is no pit in [1,2].**

* R1: ¬P1,1
* R2: B1,1 ⇔ (P1,2 ∨ P2,1)
* R3: B2,1 ⇔ (P1,1 ∨ P2,2 ∨ P3,1)
* R4: ¬B1,1
* R5: B2,1

**First we take R2, because it is the only knowledge base with P1,2:**

* R2: B1,1 ⇔ (P1,2 ∨ P2,1)

**We then use the biconditional elimination rule:**

**If you replace α and  with the Wumpus variables:**

* R6: (B1,1 ⇒ (P1,2 ∨ P2,1)) ∧ ((P1,2 ∨ P2,1) ⇒ B1,1)

**Use R6 with the And-Elimination rule:**

* R7: ((P1,2 ∨ P2,1) ⇒ B1,1)

**Use R7 with the Contraposition equivalence:**

* R8: (¬B1,1 ⇒ ¬(P1,2 ∨ P2,1))

**Use R8 with Modus Ponens and R4:**

* R9: ¬(P1,2 ∨ P2,1)

**Use de Morgan rule:**

* ¬P1,2 ∧ ¬P2,1

**Now we can finally make the conclusion that there is not a pit in [1,2] nor [2,1]**

* How do we define a Proof Problem?
  + **Initial State:**
    - Initial knowledge base
  + **Actions:**
    - Set of actions consists of all the inference rules applied to all the sentences that match the top half of the inference rule
  + **Result:**
    - Result of an action is to add the sentence in the bottom half of the inference rule
  + **Goal:**
    - Goal is a state that contains the sentence we are trying to prove
* Searching for rules is an alternative to enumerating models
  + **Finding a proof can be more efficient because the proof can ignore irrelevant propositions, no matter how many of them are there.**
  + A simple truth-table algorithm would be overwhelmed by the exponential explosion of the models.
* One final property of logical systems:
  + **Monotonicity**
    - The set of entailed sentences can only increase as information is added to the knowledge base.
    - KB ⊨ α then KB ∧ β ⊨ α
  + Suppose the knowledge base contains the additional assertion β stating that there are exactly eight pits in the world, this helps the agent draw additional conclusions, but cannot invalidate any conclusion α already inferred, such that there is no pit in [1,2].
  + Monotonicity means that inference rules can be applied whenever suitable premises are found in the knowledge base—the conclusion of the rule must follow *regardless of what else is in the knowledge base*.

### 7.5.2. Proof by resolution

* Inference rules are *sound* but the question of completeness for the inference algorithms that use them have not been discussed
* Search algorithms such as iterative deepening search (page 81) are complete in the sense that they will find any reachable goal, but if the available inference rules are inadequate, then the goal is not reachable—no proof exists that uses only those inference rules.
* **Resolution**
  + Yields a complete inference algorithm when coupled with any complete search algorithm
* **Literal**
  + Proposition symbol or negation of a proposition symbol
* **Clause**
  + Number of Literals that is connected with disjunctions

**Wumpus resolution example:**

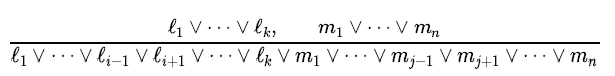
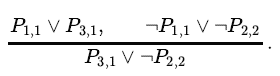
* The agent returns from [2,1] to [1,1] and then goes to [1,2], where it perceives a stench, but no breeze.
* We add the following facts to the knowledge base:
  + R11 : ¬B1,2
  + R12 : B1,2 ⇔ (P1,1 ∨ P2,2 ∨ P1,3)
* By the same process that led to R11 earlier, we can now derive the absence of pits in [2,2] and [1,3] (remember that [1,1] is already known to be pitless):
  + R13 : ¬P2,2
  + R14 : ¬P1,3
* We can also apply biconditional elimination to R3, followed by Modus Ponens with R5, to obtain the fact that there is a pit in [1,1], [2,2], or [3,1]:
  + R15: P1,1 ∨ P2,2 ∨ P3,1
* Now comes the first application of the resolution rule: the literal ¬P2,2 in R13 ***resolves*** with the literal P2,2 in R15 to give the **resolvent**:
  + R16: P1,1 ∨ P3,1
* In English; if there’s a pit in one of [1,1], [2,2], and [3,1] and it’s not in [2,2], then it’s in [1,1] or [3,1]. Similarly, the literal ¬P1,1 in R1 resolves with the literal P1,1 in R16 to give:
  + R17: P3,1
* In English: if there’s a pit in [1,1] or [3,1] and it’s not in [1,1], then it’s in [3,1].
* The last 2 inference steps are examples of the **Unit Resolution** inference rule
  + l is a literal and li  and *m* are complementary literals
* The unit resolution rule takes a **clause**—a disjunction of literals—and a literal and produces a new clause
* **Resolution rule is complete**
* A single literal can be viewed as a disjunction of one literal, also known as a **unit clause**
* **Full Resolution Rule:** 

Figure 6 - Full resolution rule

* *li* and *mj* are complementary literals.
* This says that resolution takes two clauses and produces a new clause containing all the literals of the two original clauses *except* the two complementary literals.



* You can resolve only one pair of complementary literals at a time. For example, we can resolve P and ¬P to deduce:

A picture containing text

Description automatically generated

* You can’t resolve on both P and Q at once to infer R.
* **One more technical aspect of the resolution rule:**
  + Resulting clause should contain only one copy of each literal
  + The removal of multiple copies of literals is called **factoring**
* If we resolve (A ∨ B), with (A ∨ ¬B), we obtain (A ∨ A) which is reduced to just A by factoring.
* The *soundness* of the resolution rule can be seen easily by considering the literal *mj* that is complementary to literal *mj* in the other clause. If *li* is true, then *mj* is false, and hence m1 ∨⋯∨ mj−1 ∨ mj+1 ∨⋯∨ mn must be true, because m1 ∨⋯∨ mn is given. If *li* is false, then ℓ1 ∨⋯∨ ℓi−1 ∨ ℓi+1 ∨⋯∨ ℓk must be true because ℓ1 ∨⋯∨ ℓk is given. Now *li* is either true or false, so one or other of these conclusions holds—exactly as the resolution rule states.
* What is more surprising about the resolution rule is that it forms the basis for a family of *complete* inference procedures. *A resolution-based theorem prover can, for any sentences* α *α and* β *in propositional logic, decide whether α ⊨* β. The next two subsections explain how resolution accomplishes this.

**Conjunctive normal form**

* The resolution rule applies only to clauses so it would seem to be relevant only to knowledge bases and queries consisting of clauses
* How, then, can it lead to a complete inference procedure for all of propositional logic?
  + *Every sentence of propositional logic is logically equivalent to a conjunction of clauses*
* *A sentence expressed as a conjunction of clauses is said to be in:*
  + ***Conjunctive normal form***
* B1,1 ⇔ (P1,2 ∨ P2,1)
  + Eliminate ⇔, replacing α ⇔ β with (α ⇒ β) ∧ (β ⇒ α)
    - (B1,1 ⇒ (P1,2 ∨ P2,1)) ∧ ((P1,2 ∨ P2,1) ⇒ B1,1) .
  + Eliminate ⇒, replacing α ⇒ β with ¬α ∨ β
    - (¬B1,1 ∨ P1,2 ∨ P2,1) ∧ (¬(P1,2 ∨ P2,1) ∨ B1,1)
  + Move ¬ inwards by repeated application of de Morgan’s rules and double negation elimination
    - (¬B1,1 ∨ P1,2 ∨ P2,1) ∧ ((¬P1,2 ∧ ¬P2,1) ∨ B1,1)
  + Apply the distributivity law (∧ over ∨ ) and simplify
    - (¬B1,1 ∨ P1,2 ∨ P2,1) ∧ (¬P1,2 ∨ B1,1) ∧ (¬P2,1 ∨ B1,1)

**A resolution algorithm**

* Inference procedures based on resolution work by using the principle of proof by contradiction
* To show that KB ⊨ α , we show that KB ∧ ¬α is unsatisfiable
* Convert KB ∧ ¬α to CNF
* The resolution rule is repeatedly applied to the resulting clauses
* Each pair that contains complementary literals is resolved to produce a new clause, which is added to the set if it is not already present. The process continues until one of two things happens:
  + There are no new clauses that can be added, in which case KB does not entail α
  + Two clauses resolve to yield the empty clause, in which case KB entails α
* The empty clause (a disjunction of no disjuncts) is equivalent to False
* The empty clause arises only from resolving two complementary unit clauses such as P and ¬P
* E.g.,
  + KB = R2 ∧ R4
  + KB = (B1,1 ⇔ (P1,2 ∨ P2,1)) ∧ ¬B1,1
  + We wish to prove α, which is, say ¬P1,2

**Completeness of resolution**

* Resolution Closure - RC(S)
  + Set of all clauses derivable by repeated application of the resolution rule to clauses in S or their derivatives.
  + RC(S) must be infinite
    - Because there are only finitely many distinct clauses that can be constructed out of the symbols P1, …, Pk that appear in S



* This theorem is proved by demonstrating its contrapositive:
  + If the closure *RC*(*S*) does *not* contain the empty clause, then S is satisfiable

### 7.5.3. Horn Clauses and Definite Clauses

* Completeness of resolution makes it a very important inference method
* The full power of resolution is not needed in many practical solutions
* Some real-world knowledge bases satisfy certain restrictions on the form of sentences they contain, which enables them to use a more restricted and efficient inference algorithm
* One such restricted form is the **definite clause:**
  + A disjunction of literals of which exactly one is positive
    - E.g.,
      * (¬L1,1 ∨ ¬Breeze ∨ B1,1) is a definite clause
      * (¬B1,1 ∨ P1,2 ∨ P2,1) is not a definite clause, because it has 2 positive clauses
* **Horn Clauses**
  + Disjunction of literals of which *at most one is positive*
  + ¬Shoot ∨ ¬WumpusAlive ∨ ¬WumpusAhead
  + All definite clauses are Horn clauses, as are clauses with no positive literals
    - These are called **goal clauses**
    - ¬W1,1 ∨ ¬W1,2 can be written as:

**Goal Clause**

* + - * W1,1 ∧ W1,2 ⇒ False
  + Horn clauses are closed under resolution:
    - If you resolve two Horn clauses, you get back a Horn clause
  + One more class:
    - *k*-CNF sentence
      * CNF sentence where each clause has at most *k* literals
* Every definite clause can be written as an implication whose premise is a conjunction of positive literals and whose conclusion is a single positive literal.
  + (¬L1,1 ∨ ¬Breeze ∨ B1,1) can be written as:
    - (L1,1 ∧ Breeze) ⇒ B1,1’
  + (Shoot ∨ ¬WumpusAlive∨ ¬WumpusAhead) can be written as:
    - (WumpusAlive ∧ WumpusAhead) ⇒ Shoot
* In **Horn Clauses:**
  + Premise is called the **Body**
  + Conclusion is called the **Head**
  + sentence consisting of a single positive literal, such as L1,1, is called a **fact**

**Fact**

* + - WumpusAlive written as:
      * True ⇒ WumpusAlive
* Inference with Horn clauses can be done through the forward-chaining and **backward-chaining** algorithms
* Decision if sentences follow logically from knowledge bases is linear in the size of the knowledge base

### 7.5.4. Forward and Backward Chaining

* **Forward-chaining algorithm**
  + Determine if single propositional symbol *Q* follows logically from knowledge base of Horn clauses
  + Start with known facts in the knowledge base
  + If all conditions of implication are known, then consequences can be added to the knowledge base
  + For example, if L1,1 and Breeze are known and (L1,1 ∧ Breeze) ⇒ B1,1 is in the knowledge base, then B1,1 can be added.
  + Process continues until *Q* is added or no further conclusions can be made
  + **Forward chaining is sound:**
    - Every inference is essentially an application of Modus Ponens
  + **Forward chaining is complete:**
    - Very entailed atomic sentence will be derived
  + The easiest way to see this is to consider the final state of the inferred table:
    - The table contains true for each symbol inferred during the process, and false for all other symbols
    - We can view the table as a logical model; moreover, every definite clause in the original KB is true in this model
  + We can conclude that the set of atomic sentences inferred at the fixed point defines a model of the original KB, furthermore, any atomic sentence that is entailed by the KB must be true in all its models and in this model, hence, every entailed atomic sentence must be inferred by the algorithm
  + Forward chaining is an example of the general concept of **data-driven** reasoning:
    - Reasoning in which the focus of attention starts with the known data
      * It can be used within an agent to derive conclusions from incoming percepts, often without a specific query in mind
* **Backward-chaining algorithm**
  + Works backward from the query
  + If the query is known to be true, then no work is needed otherwise the algorithm finds those implications in the knowledge base whose conclusion is q.
  + If all the premises of one of those implications can be proved true (by backward chaining), then is true.
  + The algorithm is essentially identical to the AND-OR-GRAPH-SEARCH algorithm
  + Backward chaining is a form of **goal-directed** reasoning.
    - Useful for answering specific questions such as “What shall I do now?” and “Where are my keys?”
    - Cost of backward chaining is much less than linear in the size of the knowledge base, because the process touches only relevant facts