Artificial Intelligence

academic year 2023/2024

Giorgio Fumera

Pattern Recognition and Applications Lab
Department of Electrical and Electronic Engineering
University of Cagliari (Italy)

Knowledge Representation and Inference: Logical Languages

Suggested textbook

S. Russell, P. Norvig, *Artificial Intelligence – A Modern Approach*, 4th Ed., Pearson, 2021 (or a previous edition)

Introduction

Example problems

Consider the following problems, and assume that your goal is to design **rational agents**, in the form of computer programs, capable of **autonomously** solving them.

Automatic theorem proving

Write a computer program capable to **prove** or to **refute** the following statement:

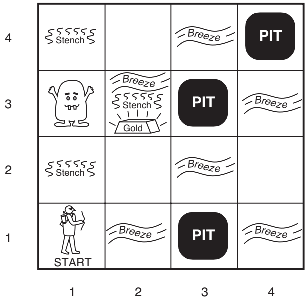
Goldbach's conjecture (1742)

For any even number $p \geq 2$, there exists at least one pair of prime numbers q and r (identical or not) such that

$$q + r = p$$

Game playing

Write a computer program capable of playing the **wumpus game**, a text-based computer game (G. Yob, c. 1972) used in a modified version as an AI's toy-problem. A basic version:



- **the wumpus world**: a cave made up of connected rooms, bottomless pits, a heap of gold, and the **wumpus**, a beast that eats anyone who enter its room
- **goal**: starting from room (1,1), find the gold and go back to (1,1), without falling into a pit or hitting the wumpus
- main **rules of the game**:
  - the content of any room is known only **after** entering it
  - in rooms neighboring the wumpus and pits a **stench** and a **breeze** is perceived, respectively

Knowledge-based systems

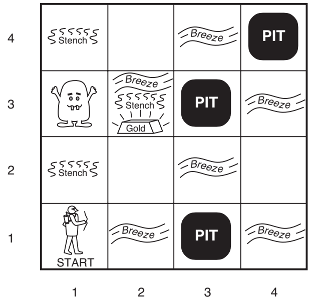
Humans usually solve problems like the ones above by combining two **high-level** capabilities: abstract **knowledge representation** and **reasoning**.

Knowledge-based systems aim at **mechanizing** the above human capabilities:

- representing knowledge about the world
- reasoning to derive new knowledge and to **guide action**

An example: playing the wumpus game

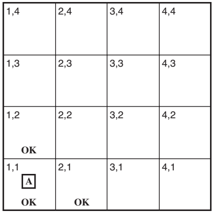
Consider the following initial configuration of the **wumpus game**, and remember that the player knows the content of any room only after entering it:

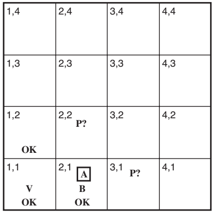


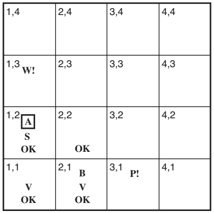
If **you** were the player, how would **you** reason to **decide** the next move to do at each game step?

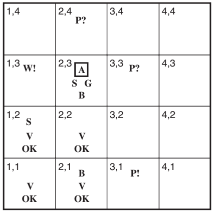
An example: playing the wumpus game

Sketch of a possible **reasoning process** for deciding the next move, starting from the configuration previously shown (some moves are omitted).

a)


b)


c)


d)


$\boxed{\textbf{A}}$	= *Agent*
B	= *Breeze*
G	= *Glitter, Gold*
OK	= *Safe square*
P	= *Pit*
S	= *Stench*
V	= *Visited*
W	= *Wumpus*

Main approaches to AI system design

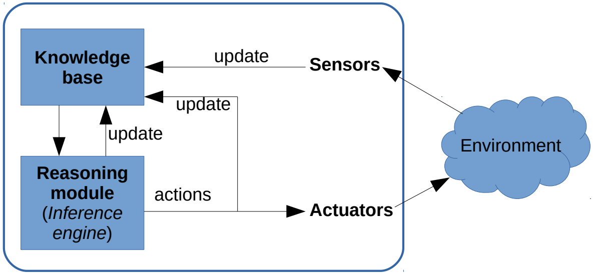
Procedural: the desired behavior (actions) are encoded directly as program code (no explicit knowledge representation and reasoning).

Declarative: explicit representation, in a **knowledge base**, of

- **background knowledge** (e.g., the rules of the wumpus game)
- knowledge about a specific **problem instance** (e.g., what the agent knows about a specific wumpus cave it is exploring)
- the agent's **goal**

Actions are then derived by **reasoning**.

Architecture of knowledge-based systems



Main feature: **separation** between **knowledge representation** and **reasoning**

- the **knowledge base** contains all the agent's knowledge about its environment, in **declarative** form
- the **inference engine** implements a reasoning process (algorithm) to derive new knowledge and to make decisions

Main applications in computer science and AI

Artificial Intelligence:

- Expert systems (medicine, engineering, finance, etc.)
- Automatic theorem provers

Computer science:

- Logic programming languages (**Prolog**, etc.)
- Databases (relational calculus, SQL)
- Semantic Web
- Satisfiability (SAT) problems:
  hardware/software design, planning (travel, logistics, ...), etc.

A short introduction to logic

- What is logic?
- Propositions and argumentations
- Logical (formal) languages
- Logical reasoning

Logic

Logic is one of the main tools used in AI for

- **knowledge representation**: logical languages
  - propositional logic
  - predicate (first-order) logic
- **reasoning**: inference rules and algorithms

Some of the main contributions:

- Aristotle (4th cent. BC): the "laws of thought"
- G. Boole (1815–64): Boolean algebra (propositional logic)
- G. Frege (1848–1925): predicate logic
- K. Gödel (1906–78): incompleteness theorem

Logic

Definition (a possible one)

> ***Logic*** *is the study of conditions under which an* ***argumentation*** *(reasoning) is* ***correct****.*

This definition involves the following concepts:

- **argumentation**: a set of statements consisting of some **premises** and one **conclusion**, e.g.:
  All men are mortal; Socrates is a man;
  then, Socrates is mortal
- **correctness**: an argumentation is said to be correct when its conclusion cannot be false when all its premises are true
- **proof**: a procedure to assess correctness

Propositions

Natural language is very complex and vague, and therefore difficult to formalize. Logic considers argumentations made up of only a subset of statements: **propositions** (**declarative statements**).

Definition

*A **proposition** is a statement expressing a concept that can be either **true** or **false**.*

Example

- *Socrates is a man*
- *Two and two makes four*
- *If the Earth had been flat, then Columbus would have not reached America*

A counterexample: *Read that book!*

Simple and complex propositions

Definition

A proposition is:

- ***simple***, *if it does not contain simpler propositions*
- ***complex***, *if it is made up of simpler propositions connected by* ***logical connectives***

Example

Simple propositions:

- *Socrates is a man*
- *Two and two makes four*

Complex propositions:

- *A tennis match can be won* ***or*** *lost*
- ***If*** *the Earth had been flat,* ***then*** *Columbus would have not reached America*

Argumentations

How to determine whether a proposition is true or false?
This is a **philosophical** question.
Logic does not address this question: it only analyzes the **structure** of an argumentation.

Example

All men are mortal; Socrates is a man;
then, Socrates is mortal.

Informally, the **structure** of this argumentation is:

all P are Q; x is P; then x is Q.

Its correctness depends only on its structure, **whatever** P, Q and x mean, that is, **regardless** of whether the corresponding propositions "all P are Q", "x is P" and "x is Q" are true or false.

Formal languages

Logic provides **formal languages** for representing (the structure of) propositions, in the form of **sentences**.
A formal language is defined by a **syntax** and a **semantics**.

Definition

- **syntax** (grammar): rules that define what sentences are “well-formed”
- **semantics**: rules that define the “meaning” of “well-formed” sentences

Examples of formal languages:

- **arithmetic**: **propositions** about numbers
- **programming languages**: **instructions** to be executed by a computer (for imperative languages like C)

Natural vs logical (formal) languages

In **natural languages**:

- syntax is not rigorously defined
- semantics defines the "content" of a statement, i.e., "what it refers to in the real world"

Example (syntax)

- *The book is on the table*: syntactically correct statement, with a clear semantics
- *Book the on is table the*: syntactically incorrect statement, no meaning can be attributed to it
- *Colorless green ideas sleep furiously*:[1] syntactically correct, but what does it mean?

[1] N. Chomsky, *Syntactic Structures*, 1957

Natural vs logical (formal) languages

Logical languages:

- syntax: formally defined
- semantics: rules that define the **truth value** of each well-formed sentence with respect to each possible **model** (a possible "world" represented by that sentence)

Example (arithmetic)

- **Syntax**: $x + y = 4$ is a well-formed sentence, $x4y+ =$ is not
- **Model**: the symbol '4' represents the natural number four, 'x' and 'y' any pair of natural numbers, '$+$' the sum operator, etc.
- **Semantics**: $x + y = 4$ is true for $x = 1$ and $y = 3$, for $x = 2$ and $y = 2$, etc.

Logical entailment

Logical reasoning is based on the relation of **logical entailment** between sentences, that defines when a sentence **logically follows** from another one.

Definition

*The sentence α **entails** the sentence β, if and only if, in every model in which α is true, also β is true. In symbols:*

$$\alpha \models \beta$$

Example (from arithmetic)

$$x + y = 4 \quad \models \quad x = 4 - y \ ,$$

because in every model (i.e., for any assignment of numbers to x and y) in which $x + y = 4$ is true, also $x = 4 - y$ is true.

Logical inference

Definition

Logical inference*: the process of deriving conclusions from premises*

Inference algorithm*: a procedure that derives sentences (conclusions) from other sentences (premises), in a given formal language*

Formally, the fact that an inference algorithm A derives a sentence α from a set of sentences ("knowledge base") KB is written as:

$$KB \vdash_A \alpha$$

Properties of inference algorithms

Definition

Soundness *(truth-preservation): if an inference algorithm derives **only** sentences entailed by the premises, i.e.:*

$$\text{if } KB \vdash_A \alpha, \text{then } KB \models \alpha$$

Completeness*: if an inference algorithm derives **all** the sentences entailed by the premises, i.e.:*

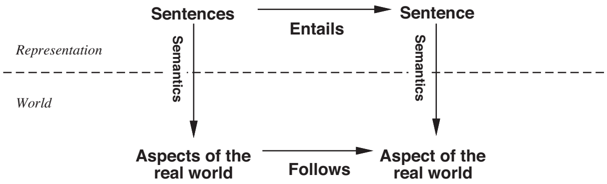
$$\text{if } KB \models \alpha, \text{then } KB \vdash_A \alpha$$

A **sound** algorithm derives conclusions that are guaranteed to be true in any world in which the premises are true.

Properties of inference algorithms

Inference algorithms operate only at the **syntactic** level:

- sentences are physical configurations of an agent (e.g., bits in registers)
- inference algorithms construct new physical configurations from previous ones
- logical reasoning should ensure that new configurations constructed by inference algorithms represent aspects of the world that actually follow from the ones represented by starting configurations

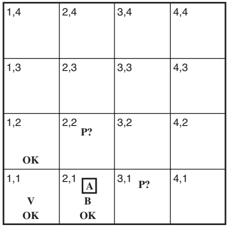


Applications of inference algorithms

In AI inference is used to answer two main kinds of questions:

- does **a given** conclusion α logically follows from the agent's knowledge *KB*? (i.e., $KB \models \alpha$?)
- what are **all** the conclusions that logically follow from the agent's knowledge? (i.e., find all α such that $KB \models \alpha$)

Example (the wumpus world)



- does a breeze in room (2,1) entail the presence of a pit in room (2,2)?
- what conclusions can be derived about the presence of pits and of the wumpus in each room, from the current knowledge?

Inference algorithms: model checking

The definition of **entailment** can be directly applied to construct a simple inference algorithm, named **Model checking**:
Given a set of premises, KB, and a sentence α, **enumerate** all possible models and **check** whether α is true in **every** model in which KB is true.

Example (arithmetic)

- $KB : \{x + y = 4\}$
- $\alpha : y = 4 - x$

Is the inference $\{x + y = 4\} \vdash y = 4 - x$ correct?

Model checking: enumerate all possible pairs of numbers x, y, and check whether $y = 4 - x$ is true whenever $x + y = 4$ is.

The issue of *grounding*

A knowledge base *KB* (set of sentences considered true) is just "syntax" (a physical configuration of the agent):

- what is the connection between a *KB* and the real world?
- how does one know that *KB* is true in the real world?

This is the same **philosophical** question met before. For humans:

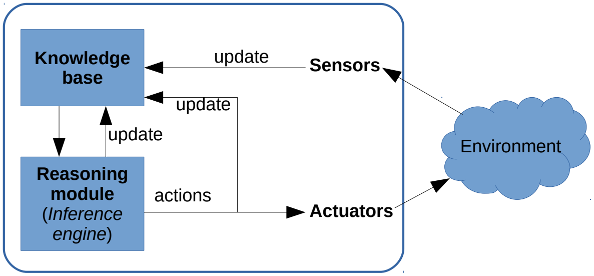
- a set of beliefs (set of statements considered true) is a physical configuration of our brain
- how do we know that our beliefs are true in the real world?

A simple answer can be given for agents (e.g., computer programs or robots): the connection is created by

- **sensors**, e.g.: perceiving a breeze in the wumpus world
- **learning**, e.g., when a breeze is perceived, there is a pit in some adjacent room

Of course, both perception and learning are **fallible**.

Architecture of knowledge-based systems revisited



If **logical languages** are used:

- **knowledge base**: a set of sentences in a given logical language
- **inference engine**: an inference algorithm for the same logical language

This course will focus on logical languages for knowledge representation.

Logical languages

Propositional logic

- the simplest logical language
- an extension of Boolean algebra (G. Boole, 1815–64)

Predicate (or **first-order**) **logic**

- a more expressive and concise extension of propositional logic
- seminal work: G. Frege (1848–1925)

Propositional Logic

Syntax

- **Atomic sentences**
  - either a **propositional symbol** that denotes a given proposition (usually written in capitals), e.g.: P, Q, ...
  - or a propositional symbol with fixed meaning: *True* and *False*
- **Complex sentences** consist of atomic or (recursively) complex sentences connected by **logical connectives** (corresponding to natural language connectives like `and`, `or`, `not`, etc.)
- **Logical connectives** (only the commonly used ones are shown – different notations exist):

$\wedge$	(and)
$\vee$	(or)
$\neg$	(not)
$\Rightarrow$	(implication)
$\Leftrightarrow$	(if and only if / logical equivalence)

Syntax

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

$$
\begin{array}{rcl}
\mathit{Sentence} & \rightarrow & \mathit{AtomicSentence} \mid \mathit{ComplexSentence} \\
\mathit{AtomicSentence} & \rightarrow & \mathit{True} \mid \mathit{False} \mid \mathit{Symbol} \\
\mathit{Symbol} & \rightarrow & P \mid Q \mid R \mid \ldots \\
\mathit{ComplexSentence} & \rightarrow & \neg \mathit{Sentence} \\
 & \mid & (\ \mathit{Sentence} \land \mathit{Sentence}\ ) \\
 & \mid & (\ \mathit{Sentence} \lor \mathit{Sentence}\ ) \\
 & \mid & (\ \mathit{Sentence} \Rightarrow \mathit{Sentence}\ ) \\
 & \mid & (\ \mathit{Sentence} \Leftrightarrow \mathit{Sentence}\ )
\end{array}
$$

Semantics

Semantics of propositional logic:

- **model** of a sentence: a possible assignment of truth values to all propositional symbols that appear in it
- **meaning** of a sentence: its **truth value** with respect to a particular **model**

Example

The sentence $P \wedge Q \Rightarrow R$ has $2^3 = 8$ possible models.
For instance, one model is $\{P = \mathit{True}, Q = \mathit{False}, R = \mathit{True}\}$.

Note: models are abstract mathematical objects with no unique connection to the real world (e.g., P may stand for **any** proposition in natural language).

Semantics

- **Atomic sentences**:
  - *True* is true in every model
  - *False* is false in every model
  - the truth value of every propositional symbol (atomic sentence) must be specified in the model
- **Complex sentences**: their truth value is **recursively** determined as a function of the simpler sentences they are made up of, and of the **truth tables** of the logical connectives they contain

Truth tables of commonly used connectives

The semantics of logical connectives is **defined** by their 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

Note: $P \Rightarrow Q$ reads as "P implies Q, or "if P then Q".

Example

Determining the truth value of $\neg P \wedge (Q \vee R)$ in all possible models, i.e., for all possible assignment of truth values to P, Q and 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

The semantics of the implication connective

The semantics (truth table) of the $\land$, $\lor$, $\neg$ and $\Leftrightarrow$ connectives is intuitive, since it is analogous to the usual meaning of the corresponding terms "and", "or", "not" and "to be equivalent to" of natural language (with some exceptions for "and" and "or", discussed later).

The truth table of the implication ($\Rightarrow$) connective may be less intuitive, instead. It corresponds to the concept of "**sufficient** but **not necessary** condition". That is to say, $P \Rightarrow Q$ can be "translated" in natural language as:

> *P is a **sufficient** but **not necessary** condition for Q to be true*

The semantics of the implication connective

Examples:

- *Five is prime and greater than two implies it is odd*
  Being prime and greater than two is a sufficient condition for a number to be odd; it is however not necessary, since there are also non-prime numbers that are odd (e.g., nine). Therefore, using the symbols P, G and O to represent the propositions "five is prime", "five is greater than two" and "five is odd", the above complex proposition can be represented as: $(P \wedge G) \Rightarrow O$
- *A pit in square* $(1,3)$ *implies a breeze in square* $(2,3)$ In the wumpus game it is known that the presence of a pit in square $(1,3)$ causes a breeze in square $(2,3)$; this condition is however not necessary, since also a pit in square $(3,3)$, or $(2,2)$, or $(2,4)$, causes a breeze in square $(2,3)$. Therefore, using the symbols P, and B to represent the propositions "there is a pit in square $(1,3)$", and "there is a breeze in square $(2,3)$", the above complex proposition can be represented as: $P \Rightarrow B$

The semantics of the implication connective

Consider again the truth table of the implication connective:

P	Q	$P \Rightarrow Q$
false	false	true
false	true	true
true	false	false
true	true	true

Note that, when P is false, then $P \Rightarrow Q$ is true, **regardless** of the truth value of Q. This can be understood by considering again the meaning of "sufficient but not necessary condition".
For instance, the following statements are true:

- *Four is prime and greater than two implies it is odd*
- *Nine is prime and greater than two implies it is odd*

The reason is that a statement of the form "if P then Q" makes the claim that Q is true **only if** P is true, otherwise **it does not make any claim** about Q. In other words, the above statement can be restated as if P is true, then I am claiming that also Q is true; otherwise, **I am making no claim**. Accordingly, the only way for $P \Rightarrow Q$ to be false is when P is true and Q is false.

The semantics of the implication connective

The meaning of the implication connective is also analogous to that of the operator $\subset$ (subset) in set theory.
Consider two sets $\mathcal{P}$ and $\mathcal{Q}$ such that $\mathcal{P} \subset \mathcal{Q}$, e.g., $\mathcal{P}$ can be the set of prime numbers that are greater than two, and $\mathcal{Q}$ can be the set of odd numbers.
By definition, any element of $\mathcal{P}$ is also an element of $\mathcal{Q}$, whereas the opposite is not necessarily true, i.e., there are elements of $\mathcal{Q}$ that are **not** elements of $\mathcal{P}$ (such as the number nine).
Accordingly, for any entity (e.g., any number) x, the mathematical sentence $\mathcal{P} \subset \mathcal{Q}$ can be restated as "if $x \in \mathcal{P}$ then $x \in \mathcal{Q}$". Using the propositional symbols P and Q to represent the propositions stating that a given entity x (e.g., the number five) belongs to a set $\mathcal{P}$ and to a set $\mathcal{Q}$, respectively, the above sentence can be represented in propositional logic as $P \Rightarrow Q$.

Propositional logic and natural language

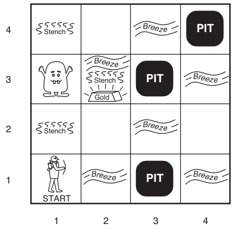
The truth table of $\wedge$ and $\vee$ is intuitive, but it captures only a **subset** of the meaning of the terms "and" and "or" in natural language.

Example

- *He felt down* ***and*** *broke his leg.*
  Here the term "and" includes a **temporal** and a **causal** relation, that is not represented by the logical connective $\wedge$ (*He broke his leg and felt down* does not have the same meaning)
- *A tennis match can be won* ***or*** *lost.*
  Here the term "or" has a **disjunctive** meaning (the corresponding Boolean operator is usually denoted by $\oplus$), which is different from the **conjunctive** meaning of $\vee$ (a tennis match cannot be **simultaneously** won and lost)

Exercise

1. Define a set of propositional symbols to represent the wumpus world: the position of the agent, the 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 no breeze is perceived in room (1,1), then there is no pit in room (1,2)*
   (b) *If the wumpus is in room (3,1) then there is a stench in rooms (2,1), (4,1) and (3,2)*



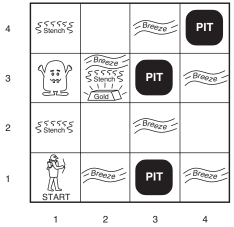
Solution of exercise 1

A possible choice of propositional symbols:

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

Solution of exercise 2

Model corresponding to the considered 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

Solution of exercise 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):

- $A_{1,1}$, $\neg A_{1,2}$, $\neg A_{1,3}$, $\dots$, $\neg A_{4,4}$ (16 sentences)
- $\neg W_{1,1}$, $\neg P_{1,1}$
- $\neg G_{1,1}$
- $\neg B_{1,1}$, $\neg S_{1,1}$

Solution of exercise 4(a)

If no breeze is perceived in room (1,1), then there is no pit in room (1,2)

It is easy to see that the absence of breeze in a room is a sufficient condition for the absence of a pit in any adjacent room. It is however not a necessary condition, since, even if a breeze is present, e.g., in room (1,1), it may be due to a pit in room (2,1), whereas room (1,2) can still contain no pit.

The above proposition can therefore be correctly represented by an implication:

$$\neg B_{1,1} \Rightarrow \neg P_{1,2}$$

Solution of exercise 4(b)

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

One may be tempted to translate the above proposition using the implication connective ($\Rightarrow$):

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

However, since there is only one wumpus, its presence in room (3,1) is not only a sufficient condition, but also a necessary one, for a stench to be present in the considered rooms. In other words, the opposite is also true:

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

The correct way to represent the above statement is therefore to use the equivalence connective:

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

Inference: model checking

Goal of inference: given a KB and a sentence α, deciding whether KB$\models \alpha$.

A simple inference algorithm: **model checking** (see above).

Application to propositional logic:

- enumerate all possible models for sentences KB$\bigcup\{\alpha\}$
- check whether α is true in every model in which KB is true

Implementation: **truth tables**.

Model checking: an example

Determine whether $\{P \lor Q, P \Rightarrow R, Q \Rightarrow R\} \models P \lor R$, using model checking.

Propositional symbols			Premises			Conclusion
P	Q	R	$P \lor Q$	$P \Rightarrow R$	$Q \Rightarrow R$	$P \lor R$
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

The answer is **yes**, because the conclusion is true in every model in which the premises are true (grey rows).

Properties of model checking

- **Soundness**: **yes**, since it directly implements the definition of entailment
- **Completeness**: **yes**, since it works for any (finite) KB any and α, and the corresponding set of models is finite
- **Computational complexity**: $O(2^n)$, where n is the number of propositional symbols appearing in KB and α

The drawback of model checking is its **exponential** computational complexity, which makes it infeasible when the number of propositional symbols is high.

Example

In the exercise about the wumpus world, 96 propositional symbols have been introduced: the corresponding truth table is made up of $2^{96} \approx 10^{28}$ rows.

Inference rules and algorithms

To avoid the exponential computational complexity of model checking, **practical** inference algorithms based on **inference rules** have been devised.

An inference rule represents a **standard pattern of inference**: it implements a **simple reasoning step** whose soundness can be easily proven, that can be applied to a set of premises having a **specific** structure to derive a conclusion.

Given a set of premises KB and a hypothetical conclusion α, **inference algorithms** try to find a **proof** $KB \vdash_A \alpha$ (if any), i.e., a **sequence** of applications of inference rules that leads from KB to α.

Exercise 1

Construct the agent's initial KB for the wumpus game.

The KB should contain:

- the rules of the game: the agent starts in room (1,1); there is a breeze in rooms adjacent to pits, etc.
- rules to decide the agent's move at each step of the game

Note that the KB must be **updated** at each step of the game:

1. adding percepts in the current room (from **sensors**)
2. **reasoning** to derive new knowledge about the position of pits and wumpus
3. **reasoning** to decide the next move
4. updating the agent's position after a move

Exercise 1

Rules of the wumpus game:

- *the agent starts in room (1,1)*:
  $A_{1,1} \land \neg A_{1,2} \land \ldots \land \neg A_{4,4}$
- *there is a breeze in rooms adjacent to pits*:
  $P_{1,1} \Rightarrow (B_{2,1} \land B_{1,2})$,
  $P_{1,2} \Rightarrow (B_{1,1} \land B_{2,2} \land B_{1,3})$, ...
  (**one** proposition in natural language, **sixteen** sentences in propositional logic – one for each room)
- *there is only one wumpus*:
  $(W_{1,1} \land \neg W_{1,2} \land \neg W_{1,3} \land \ldots \land \neg W_{4,4}) \lor$
  $(\neg W_{1,1} \land W_{1,2} \land \neg W_{1,3} \land \ldots \land \neg W_{4,4}) \lor \ldots$
  (**one** proposition in natural language, **sixteen** sentences in propositional logic – one for each room)
- ...

Often, **one** concise proposition in natural language needs to be represented by **many** complex sentences in propositional logic.

Exercise 1

How to **update** the KB to account for the **change** of the agent's position after each move? E.g., $A_{1,1}$ is true in the starting position, and becomes false after the first move:

- adding $\neg A_{1,1}$ makes the KB contradictory, since $A_{1,1}$ is still present ...
- ... but inference rules do not allow **removing** sentences

Solution: using a **different** propositional symbol for each time step, e.g., $A^t_{i,j}$, $t = 1, 2, \ldots$

- initial KB: $A^1_{1,1}$, $\neg A^1_{1,2}$, $\ldots \neg A^1_{4,4}$
- if the agent moves to (1,2), the following sentences must be added to the KB: $\neg A^2_{1,1}$, $A^2_{1,2}$, $\neg A^2_{1,3}$ $\ldots$, $\neg A^2_{4,4}$; and so on

Things get complicated ...

Exercise 2

The following argumentation (an example of **syllogism**) is intuitively correct; prove its correctness using propositional logic:
All men are mortal; Socrates is a man; then, Socrates is mortal.

Three **distinct** propositional symbols must be used:
P (*All men are mortal*), Q (*Socrates is a man*), R (*Socrates is mortal*)

Therefore:

- premises: $\{P, Q\}$
- conclusion: R

Do the premises **entail** the conclusion, i.e., $\{P, Q\} \models R$?

Model checking easily allows one to prove that the answer is **no**: in the model $\{P = \mathit{True}, Q = \mathit{True}, R = \mathit{False}\}$, the premises are **true** but the conclusion is **false**.

What's wrong?

Limitations of propositional logic

Main problems:

- **limited expressive power**
- **lack of conciseness**

Example (wumpus world)

Even small knowledge bases (in natural language) require a large number of propositional symbols and sentences.

Example (syllogisms)

Inferences involving the **structure** of **atomic** sentences (e.g., *All men are mortal, . . .*) cannot be made.

From propositional to predicate logic

The description of many domains of interest for real world applications (e.g., mathematics, philosophy, AI) involve the following elements of natural language:

- **nouns** denoting **individuals** or **objects**, e.g.: the wumpus, pits, Socrates, Plato, the number four, etc.
- **predicates** denoting **properties** of individuals (or objects) or **relations** between them, e.g.: Socrates **is a man**, five **is prime**, four **is lower than** five; the sum of two and two **equals** four
- **functions** that indirectly denote an individual (or object) between objects in terms of other ones, e.g.: the **father of Mary**, **one plus three** is even
- facts involving **some** or **all** individuals or objects of a given set, e.g.: **all** squares neighboring the wumpus are smelly; **some** numbers are prime

These elements cannot be represented in propositional logic, and require the more expressive **predicate logic**.

Predicate Logic

Models

In predicate logic a **model** consists of:

- a **domain of discourse**, i.e., the set of all objects or individuals mentioned in the propositions, e.g.:
  - the set of natural numbers
  - a set of individuals: Socrates, Plato, ...
- **relations** between domain elements, **explicitly** represented as the set of tuples among which a relation holds, e.g.:
  - being a prime number (unary relation): $\{1, 2, 3, 5, 7, 11, \dots\}$
  - being greater than (binary relation): $\{(2,1), (3,1), \dots\}$
  - being equal to (binary relation): $\{(1,1), (2,2), \dots\}$

  (unary relations are also called **properties**)
- **functions** mapping tuples of domain elements to a single one, e.g.:
  - plus: $(1,1) \rightarrow 2$, $(1,2) \rightarrow 3$, ...
  - father of: John $\rightarrow$ Mary, ...

Note that relations and functions are defined **extensionally**, i.e., by explicitly enumerating the corresponding tuples.

Syntax

The basic elements are symbols that are used to represent domain elements, relations and functions:

- **constant symbols** denote domain elements (objects or individuals), e.g.: *One*, *Two*, *Three*, *John*, *Mary*
- **predicate symbols** denote relations, e.g.: *GreaterThan*, *Prime*, *Sum*, *Father*
- **function symbols** denote functions, e.g.: *Plus*, *FatherOf*

Syntax

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

$$
\begin{array}{rcl}
\textit{Sentence} & \rightarrow & \textit{AtomicSentence} \\
 & | & (\textit{Sentence Connective Sentence}) \\
 & | & \textit{Quantifier Variable}, \ldots \textit{Sentence} \\
 & | & \neg\ \textit{Sentence} \\
\textit{AtomicSentence} & \rightarrow & \textit{Predicate}(\textit{Term}, \ldots) \\
\textit{Term} & \rightarrow & \textit{Function}(\textit{Term}, \ldots) \mid \textit{Constant} \mid \textit{Variable} \\
\textit{Connective} & \rightarrow & \Rightarrow \mid \wedge \mid \vee \mid \Leftrightarrow \\
\textit{Quantifier} & \rightarrow & \forall \mid \exists \\
\textit{Constant} & \rightarrow & \textit{John} \mid \textit{Mary} \mid \textit{One} \mid \textit{Two} \mid \ldots \\
\textit{Variable} & \rightarrow & a \mid x \mid s \mid \ldots \\
\textit{Predicate} & \rightarrow & \textit{GreaterThan} \mid \textit{Father} \mid \ldots \\
\textit{Function} & \rightarrow & \textit{Plus} \mid \textit{FatherOf} \mid \ldots
\end{array}
$$

Semantics: interpretations

Remember that semantics defines the truth of well-formed sentences, related to a particular model.
In predicate logic this requires an **interpretation**: defining which domain elements, relations and functions are referred to by symbols.

Examples:

- *One*, *Two* and *Three* denote the natural numbers 1, 2, 3;
  John and *Mary* denote the individuals John and Mary
- *GreaterThan* denotes the binary relation “to be greater than” ($>$) between numbers;
  Father denotes the fatherhood relation between individuals
- *Plus* denotes the function mapping a pair of numbers to the number corresponding to their sum

Semantics: terms

Terms are logical expressions denoting domain elements.
A term can be:

- **simple**: a constant symbol, e.g.: *One*, *Two*, *John*
- **complex**: a function symbol applied (possibly, **recursively**) to other terms, e.g.:
  FatherOf(*Mary*)
  Plus(*One*, *Two*)
  Plus(*One*, *Plus*(*One*, *One*))

Note:

- assigning a constant symbol to every domain element is not required (domains can be even infinite): only elements **explicitly** mentioned in propositions (e.g., *Socrates*) should be assigned a constant symbol
- a domain element can be denoted by more than one symbol

Semantics: atomic sentences

Atomic sentences are the simplest kind of proposition: a predicate symbol applied to a list of terms.

Examples:

- $GreaterThan(Two, One)$
- $Prime(Two)$,
- $Prime(Plus(Two, Two))$
- $Sum(One, One, Two)$
- $Father(John, Mary)$
- $Father(FatherOf(John), FatherOf(Mary))$

Note that there is no formal distinction between predicate and function symbols. There is however a syntactic difference: functions are **terms**, and therefore can only appear as **arguments** of predicates and (recursively) functions; predicates cannot appear as arguments of predicates or functions, instead.

Semantics: atomic sentences

Definition

An atomic sentence is true, in a given model and under a given interpretation, if the relation referred to by its predicate symbol holds between the objects referred to by its arguments (terms)

Example

According to the above model and interpretation:

- $GreaterThan(One, Two)$ is false
- $Prime(Two)$ is true
- $Prime(Plus(One, One))$ is true
- $Sum(One, One, Two)$ is true
- $Father(John, Mary)$ is true

Semantics: complex sentences

Complex sentences are obtained as in propositional logic, using logical connectives.

Examples:

- $Prime(Two) \land Prime(Three)$
- $\neg Sum(One, One, Two)$
- $GreaterThan(One, Two) \Rightarrow (\neg GreaterThan(Two, One))$
- $Father(John, Mary) \lor Father(Mary, John)$

Semantics (truth value) is determined as in propositional logic.
Examples: the second sentence above is false, the others are true.

Semantics: quantifiers

Quantifiers allow one to express propositions involving **collections** of domain elements, **without** enumerating them **explicitly**.

Two main quantifiers are used in predicate logic:

- **universal** quantifier, e.g.:
  All *men are mortal*
  All *rooms neighboring the wumpus are smelly*
  All *even numbers are not prime*
- **existential** quantifier, e.g.:
  Some *numbers are prime*
  Some *rooms contain pits*
  Some *men are philosophers*

Quantifiers require a new kind of term: **variable symbols**, usually denoted with lowercase letters.

Semantics: universal quantifier

Example

Assume that the **domain** is the set of natural numbers.
All natural numbers are greater or equal than one

$$\forall x \; GreaterOrEqual(x, One)$$

Semantics: universal quantifier

The semantics of a sentence $\forall x\ \alpha(x)$, where $\alpha(x)$ is a sentence containing the variable x, is:

$\alpha(x)$ is true for **each** domain element in place of x

Example

If the domain is the set of natural numbers,

$$\forall x\ \mathit{GreaterOrEqual}(x, \mathit{One})$$

states that the following (infinite) sentences are **all** true:
$\mathit{GreaterOrEqual}(\mathbf{One}, \mathit{One})$
$\mathit{GreaterOrEqual}(\mathbf{Two}, \mathit{One})$
. . .

Semantics: universal quantifier

Consider the proposition:

all even numbers greater than two are not prime

A common mistake is to represent it as:

$$\forall x \; \mathit{Even}(x) \wedge \mathit{GreaterThan}(x, \mathit{Two}) \wedge (\neg \mathit{Prime}(x))$$

The above sentence actually states that:

all numbers are even, greater than two, and are not prime

which is different from the intended meaning.

Semantics: universal quantifier

The correct sentence can be obtained by restating the original proposition as:

> ***for all*** *x*, ***if*** *x is even and greater than two,*
> ***then*** *x is not prime*

This proposition states a sufficient condition, and therefore can be represented as an **implication**:

$$\forall x \ (Even(x) \land GreaterThan(x, Two)) \Rightarrow (\neg Prime(x))$$

In general, propositions where “all” refers to all the domain elements **that satisfy some condition** should be represented using an **implication**.

Semantics: universal quantifier

Consider again this sentence:

$$\forall x \; (Even(x) \land GreaterThan(x, Two)) \Rightarrow (\neg Prime(x))$$

Claiming that it is true corresponds to claim that also sentences like the following ones are true:

$$(Even(One) \land GreaterThan(One, Two)) \Rightarrow (\neg Prime(One))$$

Note that the antecedent of the implication is false (the number one is not even, nor it is greater than the number two). This is not contradictory, since implications with false antecedents are true **by definition** (see again the truth table of $\Rightarrow$).

Semantics: existential quantifier

Example

Assume that the domain is the set of natural numbers.

- *Some numbers are prime*

$$\exists x \; \mathit{Prime}(x)$$

  This is read as: *there exists some x such that x is prime*

- *Some numbers are not greater than three, and are even*

$$\exists x \; \neg \mathit{GreaterThan}(x, \mathit{Three}) \wedge \mathit{Even}(x)$$

Semantics: existential quantifier

Consider a proposition like the following:

> *some odd numbers are prime*

A common mistake is to represent it using an implication:

$$\exists x \; Odd(x) \Rightarrow Prime(x)$$

The above sentence actually states that:

> *there exists some number such that, if it is odd, then it is prime*

which is **weaker** than than the original proposition, since it would be true (by definition of $\Rightarrow$) also if there were no odd numbers (i.e., if the antecedent $Odd(x)$ were false for all domain elements).

Semantics: existential quantifier

The sentence corresponding to the original proposition can be obtained by restating the latter as:

there exists *some x such that x is odd* ***and*** *x is prime*

The above proposition can be represented using a **conjunction**:

$$\exists x \ \mathit{Odd}(x) \land \mathit{Prime}(x)$$

In general, propositions stating several properties about “some” domain element should be represented using a **conjunction**.

Semantics: nested quantifiers

A sentence can contain more than one quantified variable.
If the quantifier is the same for all variables, e.g.:

$$\forall x (\forall y (\forall z \ldots \alpha[x, y, z, \ldots] \ldots))$$

then the sentence can be rewritten more concisely as:

$$\forall x, y, z \ldots \alpha[x, y, z, \ldots]$$

For instance, in the domain of natural numbers, the proposition:

> *If a number is greater than another number, then also the successor of the former is greater than the latter*

can be represented by the following sentence (using the function *Successor*):

$$\forall x, y \; GreaterThan(x, y) \Rightarrow GreaterThan(Successor(x), y)$$

Semantics: nested quantifiers

If a sentence contains both universally and existentially quantified variables, its meaning depends on the **order** of quantification.
In particular, $\forall x(\exists y\ \alpha[x, y])$ and $\exists y(\forall x\ \alpha[x, y])$ are **not** equivalent, i.e., they are not true under the **same** models.

For instance,

$$\forall x\ \exists y\ Loves(x, y)$$

states that (i.e., is true under any model in which) *everybody loves somebody*.

Instead,

$$\exists y\ \forall x\ Loves(x, y)$$

states that *there is someone who is loved by everyone*, whose meaning is different from the proposition above, i.e., these two sentences can be true under **different** sets of models.

Semantics: connections between quantifiers

The quantifiers $\forall$ and $\exists$ are connected with each other through **negation**, just like in natural language.

For instance, asserting that *Every natural number is greater than or equal to zero* is the same as asserting that *There does not exist any natural number which is not greater than or equal to zero*.

The two propositions above can be respectively translated into the following, **equivalent** sentences, whose domain is assumed to be the set of natural numbers:

$$\forall x \; GreaterOrEqual(x, Zero)$$
$$\neg\,(\exists x \; \neg GreaterOrEqual(x, Zero))$$

Semantics: connections between quantifiers

In general, since $\forall$ is equivalent to a conjunction over all domain elements, and $\exists$ is equivalent to a disjunct, they obey De Morgan's rules (shown below on the left, in the usual form involving two propositional variables):

$$\begin{array}{rcl|rcl}
\neg P \land \neg Q & \Leftrightarrow & \neg(P \lor Q) & \forall x(\neg\alpha[x]) & \Leftrightarrow & \neg(\exists x\ \alpha[x]) \\
\neg(P \land Q) & \Leftrightarrow & (\neg P) \lor (\neg Q) & \neg(\forall x\ \alpha[x]) & \Leftrightarrow & \exists x(\neg\alpha[x]) \\
P \land Q & \Leftrightarrow & \neg(\neg P \lor \neg Q) & \forall x\ \alpha[x] & \Leftrightarrow & \neg(\exists x(\neg\alpha[x])) \\
P \lor Q & \Leftrightarrow & \neg(\neg P \lor \land Q) & \exists x\ \alpha[x] & \Leftrightarrow & \neg(\forall x\ (\neg\alpha[x]))
\end{array}$$

Exercises

Represent the following propositions using sentences in predicate logic, including the definition of the domain

1. All men are mortal; Socrates is a man; Socrates is mortal
2. All rooms neighboring a pit are breezy (wumpus game)
3. Peano-Russell's axioms of arithmetic, that define natural numbers (nonnegative integers):
   - P1 zero is a natural number
   - P2 the successor of any natural number is a natural number
   - P3 zero is not the successor of any natural number
   - P4 no two natural numbers have the same successor
   - P5 any property which belongs to zero, and to the successor of every natural number which has the property, belongs to all natural numbers

Exercises

4. Assume that the goal is to prove that West is a criminal (using suitable inference algorithms):

   The law says that it is a crime for an American to sell weapons to hostile countries. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.

   Note that in a knowledge-based system the first proposition above encodes the **general** knowledge about the problem at hand (analogously to the rules of chess and of the wumpus game), whereas the second proposition encodes a **specific** problem instance (analogously to a specific configuration of a chessboard or of the wumpus maze).

Solution of exercise 1

Domain and symbols:

- domain: any set including all men
- constant symbols: *Socrates*
- predicate symbols: *Man* and *Mortal*, unary predicates; e.g., $Man(Socrates)$ means that Socrates is a man.

The sentences are:

$$\forall x \; Man(x) \Rightarrow Mortal(x)$$
$$Man(Socrates)$$
$$Mortal(Socrates)$$

Solution of exercise 2

A **possible** choice of domain and symbols:

- domain: row and column coordinates
- constant symbols: 1, 2, 3, 4
- predicate symbols:
  - *Pit*, binary predicate; e.g., $P(1,2)$ means that there is a pit in room (1,2)
  - *Adjacent*, predicate with four terms; e.g., $\mathit{Adjacent}(1,1,1,2)$ means that room (1,1) is adjacent to room (1,2)
  - *Breezy*, binary predicate; e.g., $\mathit{Breezy}(2,2)$ means that there is a breeze in room (2,2)

Solution of exercise 2

The required sentence can be obtained by restating the considered proposition as:

> *if a room contains a pit, then all the adjacent rooms are breezy*

Note that the latter proposition represents a **sufficient** condition for all the rooms adjacent to given one to be breezy, but not a necessary one, since the same rooms can be breezy also due to the presence of pits in other rooms. The latter proposition can therefore be represented as an **implication**:

$$\forall x, y \; Pit(x, y) \Rightarrow (\forall p, q \; Adjacent(x, y, p, q) \Rightarrow Breezy(p, q))$$

Solution of exercise 2

One could however argue that the original proposition does not **completely** represent the corresponding rule of the wumpus game, that can be stated as follows:

> *a room is breezy, if and only if at least one of the adjacent rooms contains a pit*

It is easy to see that the above proposition states a **necessary** and **sufficient** condition, which can be represented as an **equivalence**:

$$\forall x, y \; Breezy(x, y) \Leftrightarrow (\exists p, q \; Adjacent(x, y, p, q) \land Pit(p, q))$$

Solution of exercise 3

A **possible** choice of domain and symbols:

- domain: any set including all natural numbers (e.g., the set of real numbers)
- constant symbols: Z, denoting the number zero
- predicate symbols:
  - N, unary predicate denoting the fact of being a natural number; e.g., $N(Z)$ means that zero is a natural number
  - Eq, binary predicate denoting equality; e.g., $Eq(Z, Z)$ means that zero equals zero
  - P denoting any **given** property
- function symbols: S, mapping a natural number to its successor; e.g., $S(Z)$ denotes one, $S(S(Z))$ denotes two

Solution of exercise 3

P1 $N(Z)$

P2 $\forall x\ N(x) \Rightarrow N(S(x))$

P3 $\neg(\exists x\ Eq(Z, S(x)))$

P4 $\forall x, y\ Eq(S(x), S(y)) \Rightarrow Eq(x, y)$

P5 $(P(Z) \wedge \forall x((N(x) \wedge P(x)) \Rightarrow P(S(x)))) \Rightarrow$
$(\forall x\ (N(x) \Rightarrow P(x)))$

Solution of exercise 4

A **possible** choice of domain and symbols:

- domain: a set including different individuals (among which Colonel West), nations (among which America and Nono), and missiles
- constant symbols: *West*, *America* and *Nono*
- predicate symbols:
  - *Country*$(\cdot)$, *American*$(\cdot)$, *Missile*$(\cdot)$, *Weapon*$(\cdot)$, *Hostile*$(\cdot)$: respectively, being: a country, an American citizen, a missile, a weapon, hostile
  - $Enemy(< who >, < to\ whom >)$: being enemies
  - $Owns(< who >, < what >)$: owning something
  - $Sells(< who >, < what >, < to\ whom >)$: selling something to someone
- no function symbols are necessary

Solution of exercise 4

The law says that it is a crime for an American to sell weapons to hostile nations:
$\forall x, y, z \; (American(x) \land Country(y) \land Hostile(y) \land Weapon(z) \land$
$\quad Sells(x, y, z)) \Rightarrow Criminal(x)$

The second proposition can be conveniently split into simpler ones:

Nono is a country...:
$Country(Nono)$

...Nono is an enemy of America (which is also a country)...:
$Enemy(Nono, America)$
$Country(America)$

...Nono has some missiles...:
$\exists x \; Missile(x) \land Owns(Nono, x)$

...all Nono's missiles were sold to it by Colonel West:
$\forall x \; (Missile(x) \land Owns(Nono, x)) \Rightarrow Sells(West, Nono, x)$

Solution of exercise 4

A human would intuitively say that the above propositions in natural language imply that West is a criminal.
However, it is not difficult to see that the above sentences in predicate logic are **not** sufficient to prove this.
The reason is that humans exploit **background knowledge** (or **common sense**) that is **not** represented **explicitly** in the above propositions. In particular, there are two “missing links”:

- an enemy nation is hostile
- a missile is a weapon

To use such additional knowledge, it must be **explicitly** represented by sentences in predicate logic:

- $\forall x, y \ (Country(x) \land Enemy(x, America)) \Rightarrow Hostile(x)$
- $\forall x \ Missile(x) \Rightarrow Weapon(x)$

Knowledge engineering

Knowledge engineering is the process of constructing a KB.

It consists of investigating a specific domain, identifying the relevant concepts (**knowledge acquisition**), and formally representing them.

This requires the interaction between

- a **domain expert** (DE)
- a **knowledge engineer** (KE), who is expert in knowledge representation and inference, but usually **not** in the domain of interest

A possible approach, suitable for **special-purpose** KBs (in predicate logic), is the following.

Knowledge engineering

1. Identify the task:
   - what range of queries will the KB support?
   - what kind of facts will be available for each problem instance?
2. Knowledge acquisition: eliciting from the domain expert the **general** knowledge about the domain (e.g., the rules of chess)
3. Choice of a **vocabulary**: what concepts have to be represented as objects, predicates, functions?
   The result is the domain's **ontology**, which affects the complexity of the representation and the inferences that can be made.
   E.g., in the wumpus game pits can be represented either as objects, or as unary predicates on squares.

Knowledge engineering

4. Encoding the domain's general knowledge acquired in step 2 (this may require to revise the vocabulary of step 3)
5. Encoding a specific problem instance (e.g., a specific chess game)
6. Posing queries to the inference procedure and getting answers
7. Debugging the KB, based on the results of step 6

Applications of predicate logic and inference algorithms

Encoding condition-action rules to recommend actions, based on a data-driven approach:

- **production systems** (production: condition-action rule)
- **expert systems**

Applications of predicate logic and inference algorithms

Logic programming languages, in particular **Prolog**, used for:

- rapid prototyping
- symbol processing applications (compilers, natural language parsers, etc.)
- developing expert systems

Example of a Prolog **clause**:

```
criminal(X) :- american(X), weapon(Y), sells(X,Y,Z),
               hostile(Z).
```

Running a program consists of proving a sentence (**query**) using a specific inference algorithm, e.g.:

- `?- criminal(west)`
  produces `Yes`
- `?- criminal(A)`
  produces `A=west, Yes`

Applications of predicate logic and inference algorithms

Theorem provers, used for:

- assisting (not replacing) mathematicians
- proof checking
- verification and synthesis of hardware and software
  - hardware design (e.g., entire CPUs)
  - programming languages (syntax)
  - software engineering (verifying program specifications, e.g., RSA public key encryption algorithm)

Beyond classical logic

Classical logic is based on two principles:

- **bivalence**: there exist only two truth values, *true* and *false*
- **determinateness**: each proposition has only one truth value

But: how to deal with propositions like the following ones?

- *Tomorrow will be a sunny day*: is this true or false, **today**?
- *John is tall*: is this “completely” true (or false)?
  This kind of problem is addressed by **fuzzy logic**
- Goldbach’s conjecture: *Every even number is the sum of a pair of prime numbers*
  Can we say this is either true or false, even if **no proof** has been found yet?

Expert Systems

Historical notes

One of the main applications of knowledge-based systems in Artificial Intelligence:

- encoding human experts' problem-solving knowledge in **specific** application domains for which **no single algorithmic solution exists** (e.g., medical diagnosis)
- commonly used as **decision support systems**, i.e., to support (not to replace) experts' decisions
- **problem-independent** architecture for knowledge representation and reasoning
- knowledge representation: `IF...THEN...` rules

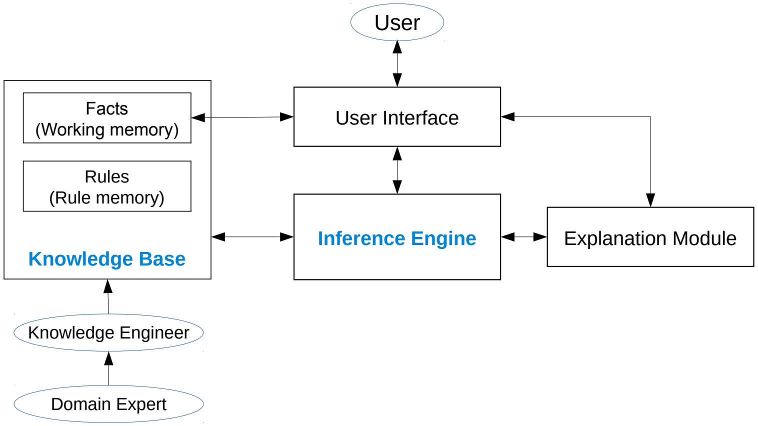
Historical notes

- Main motivation: limitations of **general** problem-solving approaches pursued in AI until the 1960s
- First expert systems: 1970s
- Widespread use in the 1980s: many commercial applications
- Used in niche/focused domains since the 1990s

Main current applications of Expert Systems

- Medical diagnosis
- Geology, botany (e.g.: rock and plant classification)
- Help desk
- Finance
- Military strategies
- Software engineering (e.g., design patterns)

Expert System architecture



Designing the Knowledge Base of Expert Systems

Two main, distinct roles:

- **knowledge engineer**
- **domain expert**

Main issues:

- defining suitable data structures for representing **facts** (specific **problem instances**) in **working memory**
- suitably **eliciting** experts' knowledge (general knowledge) and encoding it as `IF...THEN...` rules in **rule memory**

How Expert Systems work

The inference engine implements an inference algorithm which is triggered by the addition of new facts in the working memory:

```
while there is some active rule do
    select one active rule (using conflict resolution strategies)
    execute the actions of the selected rule
```

Three kinds of actions exist:

- **modifying** one fact in the working memory
- **adding** one fact to the working memory
- **removing** one fact from the working memory

Expert System shells: CLIPS

- C Language Integrated Production System (CLIPS)
  `https://www.clipsrules.net/`
- Developed since 1984 by the Johnson Space Center, NASA
- Now maintained independently from NASA as public domain, open source, free software
- Currently used by government, industry, and academia
- Main features:
  - interpreted, functional language (object-oriented extension)
  - specific data structures and instructions/functions for expert system implementation
  - interfaces to other languages (C, Python, etc.)

Other Expert System shells

- A free shell: Drools (Business Rules Management System) `http://www.drools.org`
- Several commercial shells are also available