

Artificial Intelligence

An introduction

Maik Kschischcho

Institute for Computer Science
University of Koblenz

Table of Contents

- ① Overview and history of AI
- ② Propositional logic
- ③ First order logic
- ④ Applications and limitations of logic in AI
- ⑤ Probability theory and probabilistic logic
- ⑥ Bayesian Networks
- ⑦ Outlook

Literature

Recommendations

- ① Russel, P. and Norvig, S., Artificial Intelligence: A Modern Approach., 4th ed., Pearson. 2021.
- ② Aggarwal, C.C. Artificial Intelligence: A textbook., Springer Nature Switzerland AG 2021.
<https://doi.org/10.1007/978-3-030-72357-6>
- ③ Ertel, W. Introduction to Artificial Intelligence., 2nd ed., Springer Nature Switzerland AG 2017.
<https://doi.org/10.1007/978-3-319-58487-4>

Assignments

- Oral or written exam, depending in the number of students

Recommendations

- Follow the lectures!
- Read!!
- Try to solve the problems!!!

Chap. 1 Overview and history of AI

What is this course about?

- Introduction to the deductive method (GOFAI: good old fashioned AI method)
- More recent probabilistic graphical models
- Bridging the gap between inductive AI and Machine Learning (ML)
- For ML and Deep Learning, see the specialized courses

Aims:

- Learn about the merits and limitations of deductive approaches
- Start thinking, how the GOFAI can be best combined with inductive ML methods
- Learn about Bayesian Networks
- Recap on probability (useful for ML as well)

Precursors

Myth, legend and fiction

Egyptian, greek and jewish mytholgy (Talos, Golem)



Figure: The death of Talos depicted on a 5th century BC krater now in the Jatta National Archaeological Museum in Ruvo di Puglia.

Early Artificial Intelligence (1943-1952)

- 1943: McCulloch and Walter Pitts propose a model of artificial neurons.
- 1949: Donald Hebb demonstrates an updating rule for modifying the connection strength between neurons (now called Hebbian learning).
- 1950: Alan Turing proposes a test for a machine's ability to exhibit intelligent behavior equivalent to human intelligence (Turing test).

Birth of Artificial Intelligence (1952-1956)

- 1952: Allen Newell and Herbert A. Simon create "Logic Theorist". This program proves 38 of 52 Mathematics theorems, and finds new and more elegant proofs for some theorems.
- 1956: At the Dartmouth Conference John McCarthy calls the new field 'Artificial Intelligence'

Enthusiastic phase (1956-1974)

- The researchers emphasize developing algorithms which can solve mathematical problems.
- 1966: Joseph Weizenbaum creates the first chatbot named as ELIZA.
- 1973: WABOT-1, the first intelligent humanoid robot is built in Japan.
- 1966: Failure of machine translation.
- 1969: Criticism of perceptrons (early, single-layer artificial neural networks) by Minsky and Papert.
- 1971–75: DARPA's frustration with the Speech Understanding Research program at Carnegie Mellon University.

First AI winter (1974-1980)

- Almost no funding, because of overselling during the enthusiastic phase.

Boom

1980–1987

- **Expert Systems:** A program that answers questions or solves problems about a specific domain of knowledge, using logical rules that are derived from the knowledge of experts.
- **Cyc:** Assemble a comprehensive ontology and knowledge base that spans the basic concepts and rules about how the world works (attack the **commonsense knowledge problem**).
- **Revival of neural networks:**
 - ▶ Hopfield Networks, which provide a model of human memory (John Hopfield, 1982).
 - ▶ Backpropagation algorithm (Paul Verbos, David Rumelhart, 1985).
 - ▶ Applications in optical character recognition and speech recognition.

Bust: second AI winter

1987–1993

- Expert systems turned out too difficult to maintain.
- Cheap PCs from IBM and Apple became more powerful than specialized AI machines.
- Sharp cuts in both academic and commercial research funding.

AI 1993-2011 |

Moores law: Speed and memory capacity of computers doubles every two years

- 1997: Deep Blue became the first computer chess-playing system to beat a world chess champion, Garry Kasparov.
- 2005: Stanford robot won the DARPA Grand Challenge by driving autonomously for 131 miles along an unrehearsed desert trail.
- 2011; IBM's question answering system, Watson, defeated the two greatest Jeopardy! quiz show champions.

Intelligent agents: a system that perceives its environment and takes actions which maximize its chances of success

- Influence of **decision theory** to AI (Pearl, Kaelbling, Newell).
- Probabilistic reasoning: Judea Pearl.

AI 1993-2011 II

- New tools from probability theory came to AI: Bayesian networks, hidden Markov models, information theory, stochastic modeling and classical optimization

Deep Learning and Big Data

2011-2020

Breakthroughs in Neural Networks:

- Hinton, Bengio, Le Cun and others realized that deeper networks learn representations of data and can avoid overfitting, even if shallow networks can represent the same functions in theory (Universal Approximation Theorem).
- Specialized architectures revolutionized image analysis (convolutional networks, resnets).
- Deep learning systems achieved enormous success in games like GO.
- Computation on GPU accelerators.

Breakthroughs in Big Data:

- Computational capacity to process huge amounts of data.
- Storage of huge amounts of data in the Internet.

AI era

2020-present

Attention, transformers and large language models:

- 2017 paper "Attention Is All You Need" by Vaswani et al, overcame problems with recurrent architectures.
- Transformers are based on multi-head attention mechanisms and form the basis of large language models (LLMs).

AI era

2020-present

Attention, transformers and large language models:

- 2017 paper "Attention Is All You Need" by Vaswani et al, overcame problems with recurrent architectures.
- Transformers are based on multi-head attention mechanisms and form the basis of large language models (LLMs).

Where are we going from here?

- Interpretability, explainability or even causality?
- What about deductive reasoning? Can it improve ML?
- Will there be a synthesis of induction and deduction?
- What is general artificial intelligence? How can a computer acquire world knowledge?

What is AI?

or what does it want to be

Aims: Design machines that can

- mimic human behaviour (intelligence),
- make decisions,
- learn from experience,
- reason about facts,
- solve problems.

Behaviour is not explicitly preprogrammed into the system, but the algorithms act in some way flexibly.

Agents in AI

AI studies agents in their environment

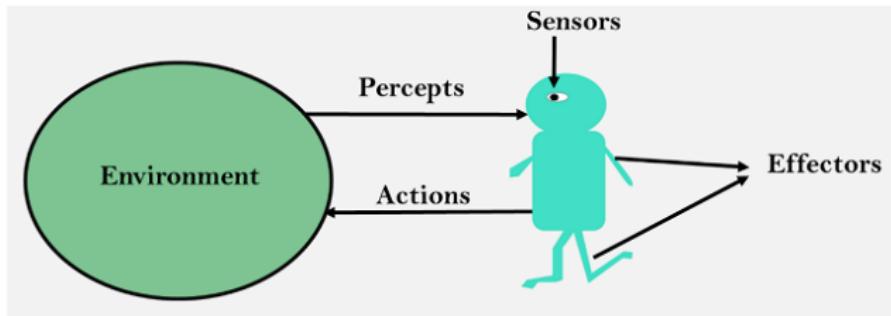


Figure: From <https://www.javatpoint.com/agents-in-ai>

An agent

- *perceives its environment through sensors*
- *acts upon that environment through actuators*

Agents

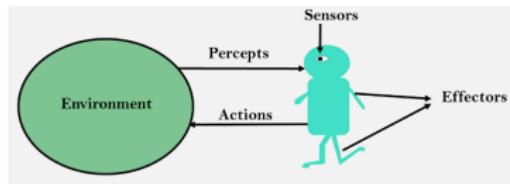


Figure: From <https://www.javatpoint.com/agents-in-ai>

Sensors: are devices detecting the state of and changes in the environment.

Actuators: are the component of machines that convert energy into motion (e.g. a muscle, an electric motor, gears, etc.).

Effectors: affect the environment (e.g. legs, wheels, arms, fingers, wings, fins, display screen, etc.).

Intelligent (rational) agents

A rational or intelligent agent can be characterized by the following rules:

- Rule i: AI agents must be able to perceive the environment.
- Rule ii: The observation must be used to make decisions.
- Rule iii: A decision should result in an action.
- Rule iv: The action must be a rational action.

Intelligent (rational) agents

A rational or intelligent agent can be characterized by the following rules:

- Rule i: AI agents must be able to perceive the environment.
- Rule ii: The observation must be used to make decisions.
- Rule iii: A decision should result in an action.
- Rule iv: The action must be a rational action.

Intelligent agent: autonomous entity which acts upon an environment for achieving a *goal*. It can learn from the environment to better achieve the goal.

What is meant with rational?

- there is a performance measure defining the success criterion
- the agent can process prior knowledge in addition to observations
- it can perform a sequence of best possible actions

PEAS

We can group the properties of an intelligent agent under the PEAS representation model:

P: Performance measure

E: Environment

A: Actuators

S: Sensors

Example

Self driving car:

- Performance: safety, time, legal drive, comfort
- Environment: roads, other vehicles, road signs, pedestrian
- Actuators: steering, accelerator, brake, signal, horn
- Camera, GPS, speedometer, odometer, accelerometer, sonar

Turing test

Alan Turing, MIND, 433, VOL. LIX. NO. 236. 1950

A machine passing the Turing test would be considered intelligent in a human like fashion.

Imitation game

- A computer (player A) and a human (player B) are placed in two different rooms.
- A human interrogator (player C) addresses each room with questions regarding any topic to which a human should be able to respond.
- If an interrogator would not be able to identify which is a machine and which is human, then the computer passes the test successfully.

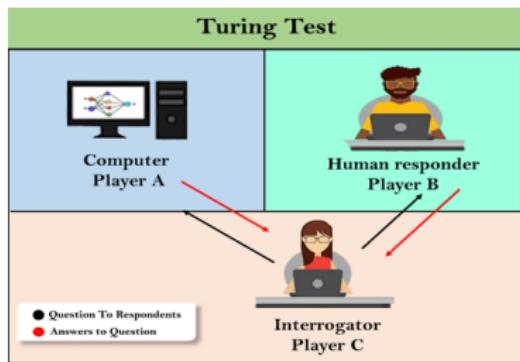


Figure: From
<https://www.javatpoint.com/agents-in-ai>

The Chinese room argument

Searle, John (1980), Behavioral and Brain Sciences, 3 (3): 417–457.

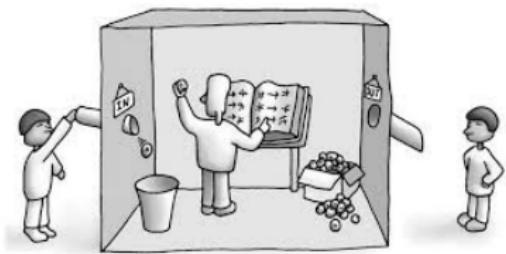


Figure: From Source: Wikicomms

- The person inside the room is provided a list of Chinese characters
- By using an instruction book explaining in detail the rules according to which strings (sequences) of characters may be formed, the person forms sentences.
- To the outside, it appears to be that the person in the room understands and speaks Chinese.
- BUT: The person doesn't understand any Chinese.

Strong AI

Searle, John (1980)

- **Strong AI:** "The appropriately programmed computer with the right inputs and outputs would thereby have a mind in exactly the same sense human beings have minds."
- According to Searl, there is a difference between
 - ▶ **simulating** a mind and
 - ▶ actually **having** a mind.

Strong AI

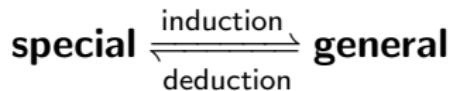
Searle, John (1980)

- **Strong AI:** "The appropriately programmed computer with the right inputs and outputs would thereby have a mind in exactly the same sense human beings have minds."
- According to Searl, there is a difference between
 - ▶ **simulating** a mind and
 - ▶ actually **having** a mind.

Is strong AI possible?

Two schools of thought in AI

Deduction and induction

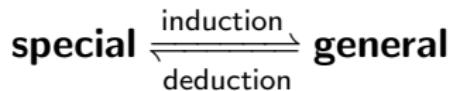


Induction

"I saw a couple of dogs yesterday.
Both had four legs. Therefore, all
dogs have four legs."

Two schools of thought in AI

Deduction and induction



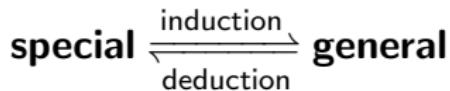
Induction

"I saw a couple of dogs yesterday. Both had four legs. Therefore, all dogs have four legs."

Deduction

"All canine animals have four legs. All dogs are canines. Therefore, dogs have four legs."

Deduction and induction



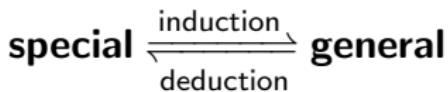
Induction

- might make logic mistakes
- inductive **learning**
- statistical approaches,
machine learning

Deduction

- mathematically accurate conclusions
- deductive **reasoning**
- logic reasoning and search methods

Deduction and induction



Induction

- Starts from examples (data)
- A **learning algorithm** is used to derive a model
- Reasoning about unseen examples requires **generalization**
- Approximations and probabilistic predictions

Deduction

- Starts with a **knowledge base** of assertions and hypotheses
- **Logical inferences** allow to reason about unknown facts
- Exact logical conclusions based on the knowledge base

An example

Spam filter for emails

Deductive

- Flag emails from blacklisted senders as spam
- Flag emails containing certain keywords or predefined word patterns as spam
- Uses a knowledge base of senders, keywords and word patterns

Inductive

- Flag spam by comparing email content with that of previous spam/no spam emails
- Uses a data base of emails labelled as spam/no spam and a machine learning algorithm

An example for a deduction

Medical expert system MYCIN (Shortliffe 2002)

- Knowledge base of bacteria and antibiotics, as well as a set of rules indicating their relationship
- Based on a physician's questions, it uses the knowledge base and the rules to make recommendations for specific patients

Advantages

- Recommendations are **explainable**
- Trustworthy, because the available experts knowledge (or hypotheses) about bacteria and antibiotics is represented in the knowledge base

Disadvantages

- Recommendations are limited by the available knowledge
- Will not work on unseen strains of bacteria

Example for an inductive system

OptAB (Wendland et al, 2024)

- Trained on data from patients with sepsis treated with different antibiotics
- Makes predictions about the disease course under different antibiotics treatments

Advantages

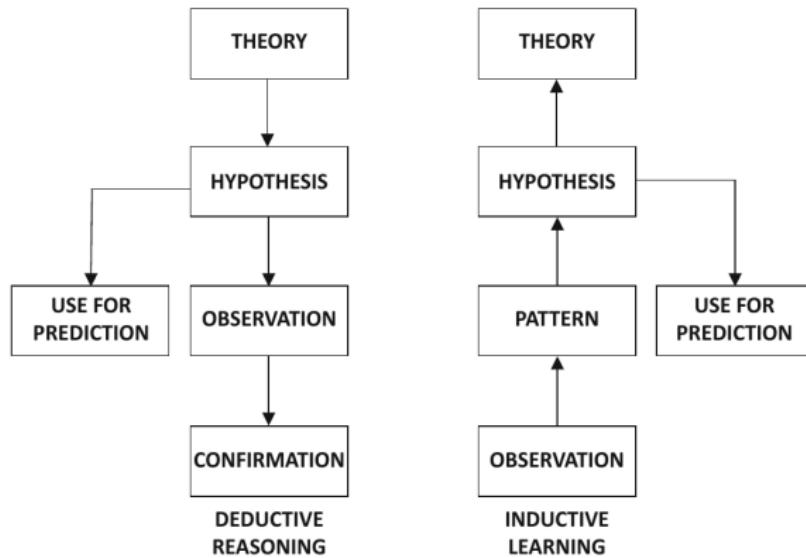
- Can recognize patterns in the data which were possibly unknown before
- Can generate new hypotheses
- Tested on unseen patients

Disadvantages

- Explainability is harder
- Mathematical proof that the predictions are always correct, based on available knowledge, is not possible

Summary

Deduction and Induction



- Deductive reasoning:
Symbolic AI
- Inductive reasoning:
Subsymbolic AI
(Machine Learning)

Figure 1.1: The two schools of thought in artificial intelligence

From Aggarwal, Artificial Intelligence, 2021, Fig.1, page 5

Chap. 2 Propositional logic

Symbolic AI vs. subsymbolic AI

- the area of “symbolic AI” is about formalisation of thought/reasoning through symbolic manipulation

Symbolic AI vs. subsymbolic AI

- the area of “symbolic AI” is about formalisation of thought/reasoning through symbolic manipulation
 - ▶ Relationships are modelled through “rules”

Symbolic AI vs. subsymbolic AI

- the area of “symbolic AI” is about formalisation of thought/reasoning through symbolic manipulation
 - ▶ Relationships are modelled through “rules”
 - ▶ The motivation behind this is not only “simulation” of intelligent processes but also “explanation” (white box approach)

Symbolic AI vs. subsymbolic AI

- the area of “symbolic AI” is about formalisation of thought/reasoning through symbolic manipulation
 - ▶ Relationships are modelled through “rules”
 - ▶ The motivation behind this is not only “simulation” of intelligent processes but also “explanation” (white box approach)
 - ▶ Not yet

Symbolic AI vs. subsymbolic AI

- the area of “symbolic AI” is about formalisation of thought/reasoning through symbolic manipulation
 - ▶ Relationships are modelled through “rules”
 - ▶ The motivation behind this is not only “simulation” of intelligent processes but also “explanation” (white box approach)
 - ▶ Not yet
- Another approach is “subsymbolic AI” or “inductive AI”

Symbolic AI vs. subsymbolic AI

- the area of “symbolic AI” is about formalisation of thought/reasoning through symbolic manipulation
 - ▶ Relationships are modelled through “rules”
 - ▶ The motivation behind this is not only “simulation” of intelligent processes but also “explanation” (white box approach)
 - ▶ Not yet
- Another approach is “subsymbolic AI” or “inductive AI”
 - ▶ Motivation is mainly about simulation/imitation

Symbolic AI vs. subsymbolic AI

- the area of “symbolic AI” is about formalisation of thought/reasoning through symbolic manipulation
 - ▶ Relationships are modelled through “rules”
 - ▶ The motivation behind this is not only “simulation” of intelligent processes but also “explanation” (white box approach)
 - ▶ Not yet
- Another approach is “subsymbolic AI” or “inductive AI”
 - ▶ Motivation is mainly about simulation/imitation
 - ▶ Example: neural networks

Symbolic AI vs. subsymbolic AI

- the area of “symbolic AI” is about formalisation of thought/reasoning through symbolic manipulation
 - ▶ Relationships are modelled through “rules”
 - ▶ The motivation behind this is not only “simulation” of intelligent processes but also “explanation” (white box approach)
 - ▶ Not yet
- Another approach is “subsymbolic AI” or “inductive AI”
 - ▶ Motivation is mainly about simulation/imitation
 - ▶ Example: neural networks
 - ▶ Approaches are usually black boxes

Symbolic AI vs. subsymbolic AI

- the area of “symbolic AI” is about formalisation of thought/reasoning through symbolic manipulation
 - ▶ Relationships are modelled through “rules”
 - ▶ The motivation behind this is not only “simulation” of intelligent processes but also “explanation” (white box approach)
 - ▶ Not yet
- Another approach is “subsymbolic AI” or “inductive AI”
 - ▶ Motivation is mainly about simulation/imitation
 - ▶ Example: neural networks
 - ▶ Approaches are usually black boxes
 - ▶ Very successful, but explainability is a problem

Symbolic AI vs. subsymbolic AI

- the area of “symbolic AI” is about formalisation of thought/reasoning through symbolic manipulation
 - ▶ Relationships are modelled through “rules”
 - ▶ The motivation behind this is not only “simulation” of intelligent processes but also “explanation” (white box approach)
 - ▶ Not yet
- Another approach is “subsymbolic AI” or “inductive AI”
 - ▶ Motivation is mainly about simulation/imitation
 - ▶ Example: neural networks
 - ▶ Approaches are usually black boxes
 - ▶ Very successful, but explainability is a problem
- For subsymbolic AI see the course “Machine learning” and “Deep Learning”

Symbolic AI vs. subsymbolic AI

- the area of “symbolic AI” is about formalisation of thought/reasoning through symbolic manipulation
 - ▶ Relationships are modelled through “rules”
 - ▶ The motivation behind this is not only “simulation” of intelligent processes but also “explanation” (white box approach)
 - ▶ Not yet
- Another approach is “subsymbolic AI” or “inductive AI”
 - ▶ Motivation is mainly about simulation/imitation
 - ▶ Example: neural networks
 - ▶ Approaches are usually black boxes
 - ▶ Very successful, but explainability is a problem
- For subsymbolic AI see the course “Machine learning” and “Deep Learning”

Symbolic AI vs. subsymbolic AI

- the area of “symbolic AI” is about formalisation of thought/reasoning through symbolic manipulation
 - ▶ Relationships are modelled through “rules”
 - ▶ The motivation behind this is not only “simulation” of intelligent processes but also “explanation” (white box approach)
 - ▶ Not yet
- Another approach is “subsymbolic AI” or “inductive AI”
 - ▶ Motivation is mainly about simulation/imitation
 - ▶ Example: neural networks
 - ▶ Approaches are usually black boxes
 - ▶ Very successful, but explainability is a problem
- For subsymbolic AI see the course “Machine learning” and “Deep Learning”

→ the foundation for symbolic AI is *logic*

Logic and formal systems

- Logics can be used to model reasoning processes

Logic and formal systems

- Logics can be used to model reasoning processes
- A (logical) statement is an abstract construct that is either T or F

Logic and formal systems

- Logics can be used to model reasoning processes
- A (logical) statement is an abstract construct that is either T or F
- Formal logic is about making statement about statements

Logic and formal systems

- Logics can be used to model reasoning processes
- A (logical) statement is an abstract construct that is either T or F
- Formal logic is about making statement about statements
- Example:

Anna is a student

All students are humans

→ Anna is a human

Logic and formal systems

- Logics can be used to model reasoning processes
- A (logical) statement is an abstract construct that is either T or F
- Formal logic is about making statement about statements
- Example:

Anna is a student

All students are humans

→ Anna is a human

- Analysis:

Logic and formal systems

- Logics can be used to model reasoning processes
- A (logical) statement is an abstract construct that is either T or F
- Formal logic is about making statement about statements
- Example:

Anna is a student

All students are humans

→ Anna is a human

- Analysis:

► Given that the statement “Anna is a student” is true

Logic and formal systems

- Logics can be used to model reasoning processes
- A (logical) statement is an abstract construct that is either T or F
- Formal logic is about making statement about statements
- Example:

Anna is a student

All students are humans

→ Anna is a human

- Analysis:
 - ▶ Given that the statement “Anna is a student” is true
 - ▶ Given that the statement “All students are humans” is true

Logic and formal systems

- Logics can be used to model reasoning processes
- A (logical) statement is an abstract construct that is either T or F
- Formal logic is about making statement about statements
- Example:

Anna is a student

All students are humans

→ Anna is a human

- Analysis:
 - ▶ Given that the statement “Anna is a student” is true
 - ▶ Given that the statement “All students are humans” is true
 - ▶ then the statement “Anna is a human” is a necessarily true statement

Logic and formal systems: structure

Every logic (=formal system) has the following components:

- ① Syntax: What are the possible statements?

Logic and formal systems: structure

Every logic (=formal system) has the following components:

- ① Syntax: What are the possible statements?
- ① Signature: What symbols are allowed? ($\Sigma = \{\text{Anna, human, student}\}$)

Logic and formal systems: structure

Every logic (=formal system) has the following components:

- ① Syntax: What are the possible statements?
 - ① Signature: What symbols are allowed? ($\Sigma = \{\text{Anna, human, student}\}$)
 - ② Grammar: how can symbols be combined in order to obtain complex statements?
 $(\text{student} \Rightarrow \text{human})$

Logic and formal systems: structure

Every logic (=formal system) has the following components:

- ① Syntax: What are the possible statements?
 - ① Signature: What symbols are allowed? ($\Sigma = \{\text{Anna, human, student}\}$)
 - ② Grammar: how can symbols be combined in order to obtain complex statements?
 $(\text{student} \Rightarrow \text{human})$
- ② Semantics: Which are the “true” statements? What is the relationship between true statements?

Logic and formal systems: structure

Every logic (=formal system) has the following components:

- ① Syntax: What are the possible statements?
 - ① Signature: What symbols are allowed? ($\Sigma = \{\text{Anna, human, student}\}$)
 - ② Grammar: how can symbols be combined in order to obtain complex statements?
 $(\text{student} \Rightarrow \text{human})$
- ② Semantics: Which are the “true” statements? What is the relationship between true statements?
 - ① Interpretations (or “possible worlds”): assign meaning to symbols of language.

Logic and formal systems: structure

Every logic (=formal system) has the following components:

- ① Syntax: What are the possible statements?
 - ① Signature: What symbols are allowed? ($\Sigma = \{\text{Anna, human, student}\}$)
 - ② Grammar: how can symbols be combined in order to obtain complex statements?
 $(\text{student} \Rightarrow \text{human})$
- ② Semantics: Which are the “true” statements? What is the relationship between true statements?
 - ① Interpretations (or “possible worlds”): assign meaning to symbols of language.
 - ② Models: What are the interpretations in which a statement is true?

Logic and formal systems: structure

Every logic (=formal system) has the following components:

- ① Syntax: What are the possible statements?
 - ① Signature: What symbols are allowed? ($\Sigma = \{\text{Anna, human, student}\}$)
 - ② Grammar: how can symbols be combined in order to obtain complex statements?
$$\text{student} \Rightarrow \text{human}$$
- ② Semantics: Which are the “true” statements? What is the relationship between true statements?
 - ① Interpretations (or “possible worlds”): assign meaning to symbols of language.
 - ② Models: What are the interpretations in which a statement is true?
 - ③ Entailment: when is one statement entailed (follows logically from) another?

Logic and formal systems: structure

Every logic (=formal system) has the following components:

- ① Syntax: What are the possible statements?
 - ① Signature: What symbols are allowed? ($\Sigma = \{\text{Anna, human, student}\}$)
 - ② Grammar: how can symbols be combined in order to obtain complex statements?
$$\text{student} \Rightarrow \text{human}$$
- ② Semantics: Which are the “true” statements? What is the relationship between true statements?
 - ① Interpretations (or “possible worlds”): assign meaning to symbols of language.
 - ② Models: What are the interpretations in which a statement is true?
 - ③ Entailment: when is one statement entailed (follows logically from) another?
- ③ Calculus: How can entailment be implemented?

Propositions and logical expressions

- **Atoms** are individual propositions which can not be further broken down into smaller constituents

Propositions and logical expressions

- **Atoms** are individual propositions which can not be further broken down into smaller constituents
- The propositions are represented by **symbols**

Propositions and logical expressions

- **Atoms** are individual propositions which can not be further broken down into smaller constituents
- The propositions are represented by **symbols**
- Propositions can be connected to form new propositions using a **formula**

Propositions and logical expressions

- **Atoms** are individual propositions which can not be further broken down into smaller constituents
- The propositions are represented by **symbols**
- Propositions can be connected to form new propositions using a **formula**

Propositions and logical expressions

- **Atoms** are individual propositions which can not be further broken down into smaller constituents
- The propositions are represented by **symbols**
- Propositions can be connected to form new propositions using a **formula**

Example

a = "The street ist wet."

b = "It is raining."

We can connect these two propositions to form a new proposition

c = "If it is raining, the street is wet."

The proposition c can be expressed as a formula

$b \Rightarrow a$

We will see, that logic processing is independent of the semantics of the propositional symbols.

Syntax of propositional logic I

Definition

Let Σ be a set of symbols and $\mathcal{O} = \{\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow\}$ be the set of logical operators. The sets \mathcal{O}, Σ and $\{T, F\}$ are pairwise disjoint. Σ is called the **signature** and its elements are the **proposition variables**. The set of propositional logic formulas is recursively defined via:

- T and F are (atomic) formulas.
- All proposition variables, i.e. all elements of Σ are (atomic) formulas.
- If a and b are formulas, then $\neg a$, (a) , $a \vee b$, $a \wedge b$, $a \Rightarrow b$, $a \Leftrightarrow b$ are also formulas.

Example

Given $\Sigma = \{a, b, c\}$ we can form the formulas

$$a \wedge b, \quad , \quad a \wedge b \wedge c, \quad a \vee c \wedge b, \quad (\neg a \wedge b) \Rightarrow (\neg c \vee a)$$

Syntax of propositional logic II

Definition

We read the symbols in the following way:

T :	"true"
F :	"false"
$\neg a$:	"not a" (negation)
$a \wedge b$:	"a and b" (conjunction)
$a \vee b$:	"a or b" (disjunction)
$a \Rightarrow b$:	"if a then b" (implication))
$a \Leftrightarrow b$:	"a if and only if b" (equivalence))

Semantics of logical formulas

There are only two truth values in propositional logic, T for "true" and F for "false".

Example

When is the formula $a \wedge b$ true? It depends, whether a and b are true. If $a = \text{"It is cold today."}$ and $b = \text{"It is raining."}$ and both are true, then $a \wedge b$ is true.

If, however, $b = \text{"It is dry."}$, but b is false, then $a \wedge b$ is false.

This means, that we have to assign T or F to the proposition variables, reflecting the state of the world.

Why we need interpretations I

Logic formulas alone can not represent any knowledge. We need to assign objects of the real world to the logic symbols to decide about the truth of a logic formula.

Example

A safety system with a 2 of 3 logic.

- Let Sig_1 , Sig_2 , and Sig_3 be three sensors measuring the same signal.
- If at least two sensors have values above a critical value, the system should indicate an error.

We introduce symbols describing the safety system

- $a =$ "Signal Sig_1 is above a critical value"
- $b =$ "Signal Sig_2 is above a critical value"
- $c =$ "Signal Sig_3 is above a critical value"
- $z =$ "The safety system indicates an error."

Why we need interpretations II

Example

Now, consider the following formulas

- $a \wedge c \Rightarrow z$: If Sig_1 is above critical ($a = T$) and Sig_3 is above critical ($c = T$) implies that system indicates an error ($z = T$),
- $\neg a \vee \neg b \Rightarrow z$: This states, that if Sig_1 is not above critical ($\neg a = T$) and Sig_3 is not above critical ($\neg c = T$), then the system should indicate an error ($z = T$).

For our problem, the formula $a \wedge c \Rightarrow z$ is true, but $\neg a \vee \neg b \Rightarrow z$ is false.

This example illustrates

- ① We need interpretations to assign a truth value to logical formulas
- ② A given set of formulas can be valid in different domains (worlds). The logical formulas are the same, but they consider different facts of the world.

Semantics of logical formulas

Interpretations

Definition

A mapping $I : \Sigma \rightarrow \{T, F\}$ assigning to every proposition variable $s \in \Sigma$ a truth value $I(s)$ is called an **interpretation**.

If there are n variables in a formula, there are 2^n different interpretations.

Semantics of logical formulas

Truth table for logical operators

a	b	(a)	$\neg a$	$a \wedge b$	$a \vee b$	$a \Rightarrow b$	$a \Leftrightarrow b$
T	T	T	F	T	T	T	T
T	F	T	F	F	T	F	F
F	T	F	T	F	T	T	F
F	F	F	T	F	F	T	T

The empty formular is true for all interpretations.

Semantics of logical formulas

Priority of logical operators

Order of evaluating more complex formulas

- ① Expression within parentheses (\cdot) are evaluated first.
- ② Order of priorities for unparenthesized formulas (from left to right):

\neg , , \wedge , , \vee , , \Rightarrow , , \Leftrightarrow

Semantics of logical formulas

Priority of logical operators

Order of evaluating more complex formulas

- ① Expression within parentheses (\cdot) are evaluated first.
- ② Order of priorities for unparenthesized formulas (from left to right):

$\neg, \quad, \wedge, \quad, \vee, \quad, \Rightarrow, \quad, \Leftrightarrow$

Example

- ① $(\neg(a \wedge b)) \Rightarrow (c \wedge d)$ means the same as $\neg(a \wedge b) \Rightarrow (c \wedge d)$ or
 $\neg(a \wedge b) \Rightarrow c \wedge d$

Semantics of logical formulas

Priority of logical operators

Order of evaluating more complex formulas

- ① Expression within parentheses (\cdot) are evaluated first.
- ② Order of priorities for unparenthesized formulas (from left to right):

$\neg, \quad, \wedge, \quad, \vee, \quad, \Rightarrow, \quad, \Leftrightarrow$

Example

- ① $(\neg(a \wedge b)) \Rightarrow (c \wedge d)$ means the same as $\neg(a \wedge b) \Rightarrow (c \wedge d)$ or
 $\neg(a \wedge b) \Rightarrow c \wedge d$
- ② $((a \wedge b) \vee c) \wedge (\neg d) \Rightarrow e$ means the same as $(a \wedge b \vee c) \wedge \neg d \Rightarrow e$

Models of formulas

Definition

An interpretation $I : \Sigma \rightarrow \{T, F\}$ that satisfies a formula s is called a **model** (world) of the formula s .

We will denote all models of a given formula s by $\mathcal{M}(s)$.

Example

Let $\Sigma = \{a, b\}$ be the signature (set of symbols) and $s = a \vee b$.

- One model of $s = a \vee b$ is $\{I(a) = T, I(b) = T\}$
- All models are given by the set $\mathcal{M}(a \vee b) = \{(I(a) = T, I(b) = T), (I(a) = T, I(b) = F), (I(a) = F, I(b) = T)\}$

Determining models of formulas using truth tables

For a given set of symbols Σ we can obtain the set $\mathcal{M}(s)$ by the following algorithm:

- ① Determine all interpretations $I \rightarrow \{t, f\}$ and assign a truth table.
- ② For all interpretations determine the truth value of s .
- ③ Delete all rows from the truth table with $I(s) = F$. Each remaining row of the truth table describes one model of s .

Example

$$s = \neg a \vee b$$

a	b	$\neg a \vee b$
T	T	T
T	F	F
F	T	T
F	F	T

$$\mathcal{M}(\neg a \vee b) =$$

$$\{(I(a) = T, I(b) = T), (I(a) = F, I(b) = T), (I(a) = F, I(b) = F)\}$$

Classification of logic formulas

Definition

A formula s is called

- **satisfiable**, if it is true for at least interpretation, i.e. $\mathcal{M}(s) \neq \emptyset$.
- **falsifiable**, if it is false for at least one interpretation, i.e. $\mathcal{M}(\neg s) \neq \emptyset$.
- **logically valid (true)**, if it is true for all interpretations, i.e. $\mathcal{M}(\neg s) = \emptyset$. True formulas are also called **tautologies**.
- **unsatisfiable** if is not true for any interpretation.

Iff s is a tautology, then $\neg s$ is unsatisfiable.

The formula s is satisfiable, iff $\neg s$ is falsifiable.

Semantic equivalence I

versus syntactic equivalence

Definition

Two formulas a and b are called **semantically equivalent** if they take on the same value for all their interpretations. We write $a \equiv b$.

Syntactic equivalence $a \Leftrightarrow b$ is just a syntactic object of the formal language of propositional logic, defined via the truth table. Semantic equivalence $a \equiv b$ is used to describe the same meaning of the propositions a and b .

Semantic equivalence II

versus syntactic equivalence

Example

The equivalence

$$a \Rightarrow b \equiv b \vee \neg a$$

can be verified from the truth table

a	b	$a \Rightarrow b$	$b \vee \neg a$
T	T	T	T
T	F	F	F
F	T	T	T
F	F	T	T

Semantic equivalence III

versus syntactic equivalence

Example

The equivalence

$$a \Leftrightarrow b \equiv (b \vee \neg a) \wedge (a \vee \neg b)$$

can also be verified from the truth table (reading exercise).

Equivalent logical expressions

Theorem

The junctors \wedge and \vee are commutative, associative and idempotent.

This means for example

$$a \wedge b \equiv b \wedge a \quad \text{and} \quad a \vee b \equiv b \vee a \quad (\text{commutativity})$$

$$a \vee (b \vee c) \equiv (a \vee b) \vee c \quad (\text{associativity})$$

$$a \wedge a \equiv a \quad (\text{idempotency})$$

Theorem (Equivalences)

$$\neg a \vee b \Leftrightarrow a \Rightarrow b \quad (\text{implication})$$

$$a \Rightarrow b \Leftrightarrow \neg b \Rightarrow \neg a \quad (\text{contraposition})$$

$$(a \Rightarrow b) \wedge (b \Rightarrow a) \Leftrightarrow (a \Leftrightarrow b) \quad (\text{equivalence})$$

$$\neg(a \wedge b) \Leftrightarrow \neg a \vee \neg b \quad (\text{De Morgan's law})$$

$$\neg(a \vee b) \Leftrightarrow \neg a \wedge \neg b \quad (\text{De Morgan's law})$$

$$a \vee (b \wedge c) \Leftrightarrow (a \vee b) \wedge (a \vee c) \quad (\text{distributive law})$$

$$a \wedge (b \vee c) \Leftrightarrow (a \wedge b) \vee (a \wedge c) \quad (\text{distributive law})$$

$$a \vee \neg a \Leftrightarrow t \quad (\text{tautology})$$

$$a \wedge \neg a \Leftrightarrow f \quad (\text{contradiction})$$

$$a \vee f \Leftrightarrow a$$

$$a \vee t \Leftrightarrow t$$

$$a \wedge f \Leftrightarrow f$$

$$a \wedge t \Leftrightarrow a$$

Clauses and conjunctive normal form I

Definition (Clauses and conjunctive normal form (CNF))

- ① A **literal** L is a variable or a negated variable.
- ② A clause K consists of a **disjunction**

$$L_1 \vee L_2 \vee \dots \vee L_m$$

of literals.

- ③ A formula is in **conjunctive normal form** if and only if it consists of a **conjunction**

$$K_1 \wedge K_2 \wedge \dots \wedge K_n$$

of clauses K_i .

Clauses and conjunctive normal form II

Example

The formulas

$$a \vee b \tag{1}$$

$$p \vee q \vee \neg r \tag{2}$$

are clauses. Here, the symbol r and its negation $\neg r$ is one literal.
Combining these by conjunctions provides a formula

$$(a \vee b) \wedge (p \vee q \vee \neg r)$$

in CNF.

Transformation into an equivalent CNF I

Theorem

Every propositional logic formula can be transformed into an equivalent conjunctive normal form.

Algorithm to transform a formula into a CNF

- ① Use the equivalences

$$\neg a \vee b \Leftrightarrow a \Rightarrow b \quad (\text{implication})$$

$$(a \Rightarrow b) \wedge (b \Rightarrow a) \Leftrightarrow (a \Leftrightarrow b) \quad (\text{equivalence})$$

to remove implications (\Rightarrow) and equivalences (\Leftrightarrow).

Transformation into an equivalent CNF II

- ② Use the equivalences

$$\neg(a \wedge b) \Leftrightarrow \neg a \vee \neg b \quad (\text{De Morgan's law})$$

$$\neg(a \vee b) \Leftrightarrow \neg a \wedge \neg b \quad (\text{De Morgan's law})$$

$$\neg(\neg a) = a$$

to move the negation symbolc (\neg) directly in front of each symbol.

- ③ Use the distributive laws

$$a \vee (b \wedge c) \Leftrightarrow (a \vee b) \wedge (a \vee c) \quad (\text{distributive law})$$

$$a \wedge (b \vee c) \Leftrightarrow (a \wedge b) \vee (a \wedge c) \quad (\text{distributive law})$$

to generate the CNF.

Transformation into an equivalent CNF III

Example

$$\begin{aligned}(p \vee \neg r) \Rightarrow q &\stackrel{(1)}{=} q \vee \neg(p \vee \neg r) \\ &\stackrel{(2)}{=} q \vee (\neg p \wedge r) \\ &\stackrel{(3)}{=} (q \vee \neg p) \wedge (q \vee r)\end{aligned}$$

Entailment

Assume we are given a knowledge base s as propositional logical formula. From this we want to deduce whether a query q is true or false.

Definition

A formula s entails a formula q (or q follows from s), if every model of s is also a model of q

$$\mathcal{M}(s) \subseteq \mathcal{M}(q)$$

Then we write $s \models q$

- This means that the truth of q is contained in the truth of s , but not necessarily vice versa. This means, that q can be true, even when $I(s) = F$.
- For a tautology y we have $I(y) = T$ for all interpretations I , i.e. $\emptyset \models y$. We write $\models y$.

Entailment II

An example

Example

The formula

$$s = a \wedge b \wedge (b \wedge c \vee a \wedge b \vee a \wedge c \Rightarrow z)$$

has two models, which are shown in the shortened truth table

a	b	c	z	s
T	T	T	T	T
T	T	F	T	T

Each formula, which is true for the two interpretations of a, b, c and z , follows from s . Examples are: $z, b \wedge z, a \vee b \vee c, c \wedge b \vee \neg c$.

The deduction theorem I

Theorem

$a \models b$ if and only if $\models a \Rightarrow b$

If a entails b (b follows from a), $a \Rightarrow b$ is a tautology.

The deduction theorem II

Proof.

Remember the truth table for the implication

a	b	$a \Rightarrow b$
T	T	T
T	F	F
F	T	T
F	F	T

- Assume, that $a \models b$ holds. This means that $\mathcal{M}(a) \subseteq \mathcal{M}(b)$, This means that the second line with $a \Rightarrow b = F$ can not be realized and thus $a \Rightarrow b = T$.
- Assume $a \Rightarrow b = T$. Then, the second line as again excluded and every model of a is again a model of b .



A first proof system

From the deduction theorem we learn:

- To show that $a \models b$ we can show that $a \Rightarrow b$ is a tautology, i.e., always true.
- This can be done automatically by checking the truth table.
- Drawback: For large formulas a and b , it can take a long time, because for a formula with a with n proposition variables there are 2^n different interpretations.
- We will use derivation to overcome this limitation.

Proof by contradiction

- The deduction theorem tells us that $a \models b$ can be proved by showing $a \Rightarrow b$
- Therefore, the negation $\neg(a \Rightarrow b)$ is unsatisfiable
- With

$$\neg(a \Rightarrow b) \equiv \neg(\neg a \vee b) \equiv a \wedge \neg b$$

we see that $a \wedge \neg b$ must be unsatisfiable.

Theorem (Proof by contradiction)

$a \models b$ if and only if $a \wedge \neg b$ is unsatisfiable.

If we have a query b and ask, whether b follows from a knowledge base a , we can add $\neg b$ to the knowledge base and derive a contradiction, because $b \wedge \neg b$ is unsatisfiable.

Derivation

- Idea: To show entailment $a \models b$ we replace the test of all interpretations using a truth table by a **syntactic manipulation**.
- This syntactic manipulation is called a **derivation** and we write $a \vdash b$.
- The rules of derivation form a **propositional calculus**.

Entailment: b follows from a ($a \models b$) is a **semantic** concept. It says, that for all models b are also models of a .

Derivation: is a syntactic approach to apply rules to derive b from a , i.e. to manipulate a syntactically to show $a \vdash b$

Soundness and completeness of syntactic calculi

When does a calculus produce correct answers?

Definition

A calculus is called **sound** or **correct**, if a derived proposition follows semantically. That is for formulas a and b we have

$$(a \vdash b) \Rightarrow (a \models b).$$

A calculus is called **complete** if all semantic consequences can be derived. That is for formulas a and b we have

$$(a \models b) \Rightarrow (a \vdash b).$$

Some notation I

for derivation rules

- We write

$$\{f_1, \dots, f_n\} \quad \text{or} \quad \begin{matrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{matrix}$$

for the formula $f \equiv f_1 \wedge f_2, \dots, \wedge f_n$ with conjunctions between the propositions.

- All propositions which are listed in these expression are set to true T .
- $\{f_1, \dots, f_n\} \models s$ if and only if for $I(f_1) = T, \dots, I(f_n) = T$ we also have $I(s) = T$.

Some notation II

for derivation rules

Using truth tables can be computationally expensive. Instead, we use a syntactic calculus. We write for a **derivation**

$$\frac{\begin{array}{c} f_1 \\ f_2 \\ \vdots \\ f_n \end{array}}{s}$$

$\{f_1, \dots, f_n\} \vdash s \quad \text{or}$

We call f_1, \dots, f_n the **premises** and s the **conclusion**.

Modus Ponens I

A simple rule of derivation

Theorem (Modus Ponens)

$$\frac{p \Rightarrow q}{\frac{p}{q}} \quad \text{or} \quad \{p \Rightarrow q, p\} \vdash q \quad \text{or} \quad \frac{\begin{array}{c} I(p \Rightarrow q) \equiv T \\ I(p) \equiv T \end{array}}{I(q) \equiv T}$$

"If q follows from p AND p is true, then q is also true."

The three notations are equivalent.

Modus Ponens II

A simple rule of derivation

Proof.

$I(p)$	$I(q)$	$I(p \wedge (p \Rightarrow q))$
T	T	T
T	F	F
F	T	F
F	F	F

The premises q and $p \Rightarrow q$ are true by assumption. Then, there is only one model and in this model q is also true. □

Task

Given:

- ① Set of true propositions (Axioms, knowledge base)
- ② A proposition which might be true or not (query)

Wanted: A proof that the query is true

Definition (Decidability)

A propositional calculus is called **decidable** if there is an algorithm which provides a decision whether any query against any set of axioms is true or false in a finite number of steps.

Remark: Finite can still mean that we need many steps.

The resolution rule

The modus ponens rule can be generalized. Assume we are given three clauses and let A , B and C be their literals. Then

$$\frac{A \vee B}{\neg A \vee C} \quad B \vee C$$

The derived clause is called the **resolvent**.

Proof.

Using truth tables. □

The general resolution rule I

Theorem

Given the two clauses $A \vee B_1 \vee \dots \vee B_n$ and $\neg A \vee C_1 \vee \dots \vee C_m$ with the complementary literals A and $\neg A$ are both true we can derive $B_1 \vee \dots \vee B_n \vee C_1 \vee \dots \vee C_m$.

$$\frac{A \vee B_1 \vee \dots \vee B_n}{\neg A \vee C_1 \vee \dots \vee C_m} \quad B_1 \vee \dots \vee B_n \vee C_1 \vee \dots \vee C_m$$

The general resolution rule II

Example

Assume $a = T$, $b = T$ and $c \vee \neg a \vee \neg b = T$. We want to show that C is true. We look for complementary literals and apply the resolution rule

$$\frac{\begin{array}{c} a \\ c \vee \neg a \vee \neg b \end{array}}{c \vee \neg b}$$

Now we apply the resolution rule to the resolvent

$$\frac{\begin{array}{c} b \\ c \vee \neg b \end{array}}{c}$$

Proof by contradiction I

Typically, the resolution rule is used in a different way

- Add the negated query to the axioms
- Generate the empty clause \emptyset which corresponds to a contradiction.
This means, in the last step we use

$$\frac{A \quad \neg A}{\emptyset}$$

Proof by contradiction II

Example

Assume $a = T$, $b = T$ and $c \vee \neg a \vee \neg b = T$. We want to show that C is true. To proof c we add $\neg c$ to the axioms.

$$\frac{\begin{array}{c} c \vee \neg a \vee \neg b \\ \neg c \end{array}}{\neg a \vee \neg b}$$

Now we apply the resolution to the resolvent and b

$$\frac{\begin{array}{c} \neg a \vee \neg b \\ b \end{array}}{\neg a}$$

Proof by contradiction III

Example

Now we combine this with a from the axioms to obtain the empty clause

$$\frac{\neg a \quad a}{\emptyset}$$

Resolutinal calculus

Resolution refutation

Given: Axioms (knowledge base) and the proposition (query) to be proofed.

- ① Formulate the axioms and the query as clauses and put these to the memory.
- ② Choose two clauses with complementary literals. If there are no such clauses, then stop (Proof is impossible).
- ③ Apply the resolution rule to the selected clauses with the complementray literals.
- ④ If the resolvent is the empty clause \emptyset , then Stop (Proof successfully finished). Otherwise, add the resolvent to the memory and go to step 2.

Result: Proof of the query (if it is possible)

Properties of resolution calculus

Theorem

The resolution calculus for the proof of unsatisfiability of formulas in conjunctive normal form is sound and complete.

- ① The resolution calculus is sound, because the resolution rule is correct and the proof by contradiction is correct. This means, that the query q follows from the axioms a via unsatisfiability of $a \wedge \neg q$, which is equivalent to $a \models q$.
- ② The resolution calculus is not per se complete. A counterexample is the proof of $q \vee \neg q$, which is not possible by resolution. However, we can proof this by contradiction and use the negation $\neg(q \vee \neg q) \equiv \neg q \wedge q$ and now we can use resolution to derive the empty clause \emptyset . In this sense the resolution calculus is complete, if the proof by contradiction is included.

Example I

Control of traffic lights

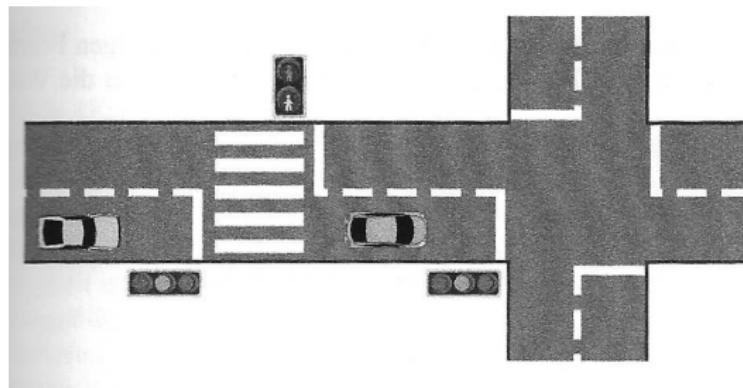


Figure: Two traffic lights and an on demand pedestrian crossing. From Lunze, KI für Ingenieure, Fig. 7.11

Specification: The on demand pedestrian crossing should not interrupt the traffic flow (the car friendly solution, just like Koblenz).

Example II

Control of traffic lights

Description of the state of the three traffic lights:

We use the propositions *pgreen*, *pyellow*, *pred* and *predyellow* and *cgreen*, *cyellow*, *cred* and *credyellow* with the following meaning

- *pgreen* = "The pedestrian traffic light is green for cars."
- *cgreen* = "The traffic light at the cross roads is green for cars."
- ...

Specification: The on demand pedestrian crossing should not interrupt the traffic flow. The formula

$$s \equiv \neg((pyellow \vee pred \vee predyellow) \wedge cgreen)$$

must always be true.

Example III

Control of traffic lights

Control:

- The traffic light at the cross roads follows the usual sequence $c\text{red}y\text{ellow} - c\text{green} - c\text{yellow} - c\text{red}$.
- The pedestrian traffic light can only switch from $p\text{green} - p\text{yellow}$ if the traffic light at the crossing is in state $c\text{yellow}$.

Example IV

Control of traffic lights

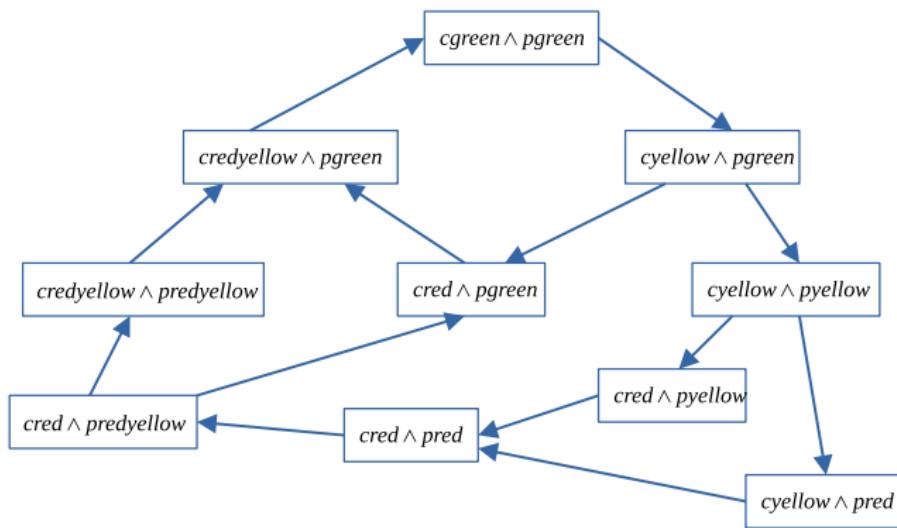


Figure: The state transition graph for the traffic lights.

Example V

Control of traffic lights

Additional formulas ensure that each traffic light can only be in one of the four states *green*, *yellow*, *red*, or *redyellow*:

$$\begin{aligned} & (cred \wedge \neg credyellow \wedge \neg cgreen \wedge \neg cyellow) \\ \vee & (\neg cred \wedge credyellow \wedge \neg cgreen \wedge \neg cyellow) \\ \vee & (\neg cred \wedge \neg credyellow \wedge cgreen \wedge \neg cyellow) \\ \vee & (\neg cred \wedge \neg credyellow \wedge \neg cgreen \wedge cyellow) \end{aligned}$$

and analogously for the pedestrian traffic light,

Example VI

Control of traffic lights

Model checking:

Add the negated specification

$$\neg s \equiv ((pyellow \vee pred \vee predyellow) \wedge cgreen)$$

one after the other to the formulas valid for each state and generate the empty clause using resolution.

Example VII

Control of traffic lights

Let the state be $cred \wedge pred$. We have to prove that the set of formulas

$$\begin{array}{c} cred \\ pred \\ \hline pyellow \vee pred \vee predyellow \\ cgreen \\ \dots \\ \dots \end{array} \left. \begin{array}{l} \text{the current state} \\ \\ \text{the negated spec} \\ \\ \text{each traffic light can only be in one state} \end{array} \right\}$$

leads to an empty clause \emptyset , i.e. a contradiction. The dots indicate the clauses expressing that each traffic light can only be in one state (see above).

Further applications

Some examples

- Automatic program verification: Increasingly complex software systems are now taking over tasks of more and more responsibility and security relevance.
- Software reuse. Programmers specify the state before and after a program was run and compare it to a software data base to select a module which fits the requirements.
- Automatic theorem proving in mathematics.
- Control system and safety system verification.
- ...

Chap. 3 First order logic

Limitations of Propositional Logic

Example

$a = \text{"Robot 7 is situated at the xy position (35, 79)"}$ can be used as a propositional logic variable. However, to specify the position of 100 different robots on a grid of 100×100 we would need $100^3 = 10^6$ different propositional variables.

First order predicate logic (PL1) can overcome this and other limitations.

Syntax I

Terms label the objects we are considering (Domain \mathcal{D}). Terms are constants, variables and functions.

- **Constants:** are concrete objects

Example: A certain machine, an alarm, a drug,...

Notation: Lower case letters

- **Variables:** represent any object of the domain

Notation: upper case letters X, Y, \dots

- **Functions:** $f(t_1, \dots, t_n)$ describe relations between objects. The arguments t_1, \dots, t_n are terms and f assigns these terms a value which is an object in the domain.

Example: $\text{target}(X)$ might assign an edge X in a directed graph to its target node.

Notation: Lower case italic letters.

Syntax II

Definition (Terms)

Let \mathcal{K} be a set of constants, \mathcal{V} be a set of variables and \mathcal{F} be a set of function symbols. These sets are pairwise disjoint. We define the set of **terms** recursively:

- All variables and constants are (atomic) terms.
- If t_1, \dots, t_n are terms and f an n -place function symbol, then $f(t_1, \dots, t_n)$ is also a term.

Syntax III

Predicates: are propositions about objects in the domain and describe properties. A predicate

$$P(t_1, \dots, t_n)$$

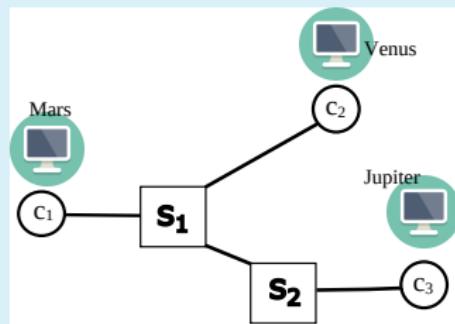
can be interpreted as the proposition: "The terms t_1, \dots, t_n have the property P ."

- If all arguments t_1, \dots, t_n of P are constants, the predicate is either true T or false.
- If one argument t_i is a variable, the truth value of the predicate is not determined.

Syntax IV

Example

First order logic description of a computer network



Communication network consisting of three computers c_1, c_2, c_3 named Mars, Venus, Jupiter and two switches s_1, s_2 .

- Constants: $\mathcal{K} = \{c_1, c_2, c_3, s_1, s_2, \text{Mars}, \text{Venus}, \text{Jupiter}\}$

Syntax V

Example

- The predicate $\text{Computer}(X)$ describes which elements are computers.
Here

$\text{Computer}(\text{Mars})$

$\text{Computer}(\text{Venus})$

$\text{Computer}(\text{Jupiter})$

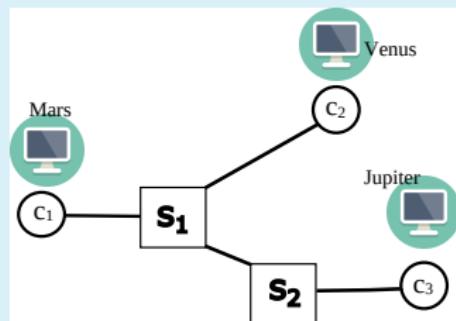
are three statements with truth value T .

- The predicate $\text{Edge}(X, Y)$ describes a connection between the nodes corresponding to the variables X and Y . This is a predicate which has no defined truth value, since X and Y are variables.

Syntax VI

Example

- The statements



$\text{Edge}(c_1, s_1)$
 $\text{Edge}(s_1, c_2)$
 $\text{Edge}(s_1, s_2)$
 $\text{Edge}(s_1, c_3)$

are all true, whereas
 $\text{Edge}(c_1, s_2)$ is false.

Syntax VII

Example

- The function f can be used to assign network nodes to the names of the computers: $c_1 = f(\text{Mars})$, $c_2 = f(\text{Venus})$, $c_3 = f(\text{Jupiter})$.

Syntax VIII

Definition

Let **Predicate logic formulas** are built as follows:

- If t_1, \dots, t_n are terms and P is a n-place predicate symbol, then $P(t_1, \dots, t_n)$ is an (atomic) formula.
- If a and b are formulas, then $\neg a$, (a) , $a \vee b$, $a \wedge b$, $a \Rightarrow b$, $a \Leftrightarrow b$ are also formulas.
- If X is a variable and a a formula, the $\forall X a$ and $\exists X a$ are also formulas. Here, \forall is the universal quantifier and \exists is the existential quantifier.
- $P(t_1, \dots, t_n)$ and $\neg P(t_1, \dots, t_n)$ are called literals.

Quantifiers

- $\forall X$ reads "for all X "
- $\exists X$ reads "there exists X "

Syntax IX

Example

- ① Let $N(X)$ denote the predicate "X is a natural number" and $\text{PositiveReal}(X)$ be the predicate "X is a positive real number", then we have

$$\forall X N(X) \Rightarrow \text{PositiveReal}(X)$$

"For all X : If X is a natural number, then it is a positive real number"

- ② For the existence quantor

$$\exists X \text{Edge}(c_1, X)$$

"There is a node having an incoming edge emanating from c_1 ."

Syntax X

Definition

Formulas in which every variable is in the scope of a quantifier are called **first-order sentences** or **closed formulas**. Variables which are not in the scope of a quantifier are called **free variables**.

Semantics I

In predicate logic, the meaning of formulas is recursively defined over the construction of the formula, in that we first assign constants, variables, and function symbols to objects in the real world.

Definition

An interpretation \mathcal{I} is defined as

- A mapping from the set of constants and variables $\mathcal{K} \cup \mathcal{V}$ to a set \mathcal{W} of names of objects in the world.
- A mapping from the set of function symbols to the set of functions in the world. Every n —place function symbol is assigned an n —place function.
- A mapping from the set of predicate symbols to the set of relations in the world. Every n —place predicate symbol is assigned an n —place relation.

Semantics II

Remark: Some authors use the term "arity of n " to indicate the number of arguments in functions or predicates, instead of n — place.

Semantics III

Example

Let k_1, k_2 and k_3 be constants. "plus" a two-place function symbol and gr a predicate symbol. The truth of the formula

$$p \equiv gr(\text{plus}(k_1, k_3), k_2)$$

depends on the interpretation \mathcal{I} . Let us choose one interpretation:

$$\mathcal{I}_1 : k_1 \mapsto 1, k_2 \mapsto 2, k_3 \mapsto 3, \text{plus} \mapsto +, gr \mapsto >$$

The formula above is mapped to

$$1 + 3 > 2 \quad \text{or} \quad 4 > 2$$

For all pairs $x > y$ with $x, y \in \{1, 2, 3, 4\}$ we can evaluate the formula $x > y$ and obtain the set $G = \{(4, 3), (4, 2), (4, 1), (3, 2), (3, 1), (2, 1)\}$. Since $(4, 2) \in \mathbb{I}_1$ we can say that $p = T$ under the interpretation \mathcal{I}_1 .

Example

For the interpretation

$$\mathcal{I}_2 : k_1 \mapsto 2, k_2 \mapsto 1, k_3 \mapsto 3, \text{plus} \mapsto -, \text{gr} \mapsto >$$

we find

$$2 - 3 - > 1 \quad \text{or} \quad - 1 > 1$$

We see that $p = F$ under the interpretation \mathcal{I}_2 .

Truth in PL1

Definition

- An atomic formula $P(t_1, \dots, t_n)$ is *true* (or short T) under the interpretation \mathcal{I} if, after interpretation and evaluation of all terms t_1, \dots, t_n and interpretation of the predicate P through the n -place relation r , it holds that

$$(\mathcal{I}(t_1), \dots \mathcal{I}(t_n)) \in r$$

- The truth of quantifierless formulas follow from the truth of atomic formulas— as in propositional calculus—through the semantics of the logical operators " \neg ", " \wedge ", " \vee ", " \Rightarrow " and " \Leftrightarrow ".
- A formula $\forall X P$ is true under the interpretation \mathcal{I} exactly when its is true given an arbitrary change of the interpretation for the variable X (and only for X).
- A formula $\exists X P$ is true under the interpretation \mathcal{I} exactly when there is an interpretation for X which makes the formula true.

- Semantic equivalence, satisfiability, tautology, unsatisfiability, model and semantic entailment carry over from propositional calculus to first order logic.
- The deduction theorem and semantic entailment hold analogously.

Example: Family Tree I

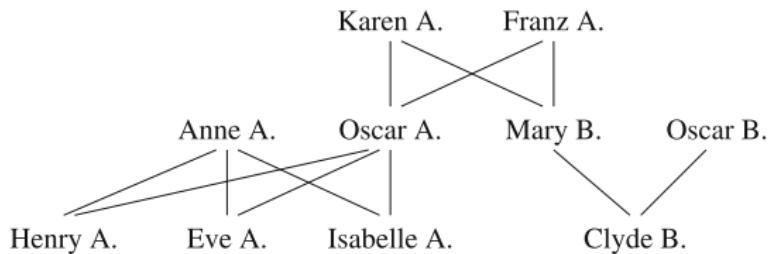


Fig. 3.1 A family tree. The edges going from Clyde B. upward to Mary B. and Oscar B. represent the element (Clyde B., Mary B., Oscar B.) as a child relationship

Figure: From Ertel, W., Introduction to AI, page 43

The tree represents family relationships. For example, the triple (Oscar A., Karen A., Franz A.) stands for the proposition "Oscar A. is a child of Karen A. and Franz A.".

Example: Family Tree II

All family relationships are expressed by the relation

$\text{child} = \{(Oscar\ A., Karen\ A., Franz\ A.), (Mary\ B., Karen\ A., Franz\ A.),$
 $(Henry\ A., Anne\ A., Oscar\ A.), (Eve\ A., Anne\ A., Oscar\ A.),$
 $(Isabelle\ A., Anne\ A., Oscar\ A.), (Clyde\ B., Mary\ B., Oscar\ B.)\}$

The one-place relation

$\text{Female} = \{\text{Karen}\ A., \text{Anne}\ A., \text{Mary}\ B., \text{Eve}\ A., \text{Isabelle}\ A.\}$

represents the female persons.

Example: Family Tree III

To establish formulas for family relationships we define a three-place predicate $\text{Child}(X, Y, Z)$ with the semantic

$$\mathcal{I}(\text{Child}(X, Y, Z)) = T \equiv (\mathcal{I}(X), \mathcal{I}(Y), \mathcal{I}(Z)) \in \text{child}.$$

For the interpretation

$$\mathcal{I}(\text{oscar}) = \text{Oscar A.}, \mathcal{I}(\text{eve}) = \text{Eve A.}, \mathcal{I}(\text{anne}) = \text{Anne A.}$$

the predicate $\text{Child}(\text{eve}, \text{oscar}, \text{anne})$ is true.

The formula

$$\forall X \forall Y \forall Z \text{Child}(X, Y, Z) \Leftrightarrow \text{Child}(X, Z, Y)$$

expresses the symmetry of the predicate Child in the last two elements.
Therefore, $\text{Child}(\text{eve}, \text{anne}, \text{oscar})$ is also true.

Example: Family Tree IV

The predicate Descendant can recursively be defined

$$\begin{aligned} \forall X \forall Y \text{Descendant}(X, Y) \Leftrightarrow \\ \exists Z \text{Child}(X, Y, Z) \vee (\exists U \exists V \text{Child}(X, U, V) \wedge \text{Descendant}(U, Y)) \end{aligned}$$

Example: Family Tree V

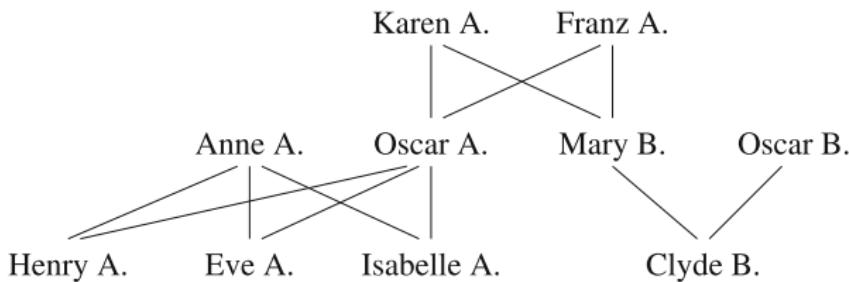


Fig. 3.1 A family tree. The edges going from Clyde B. upward to Mary B. and Oscar B. represent the element (Clyde B., Mary B., Oscar B.) as a child relationship

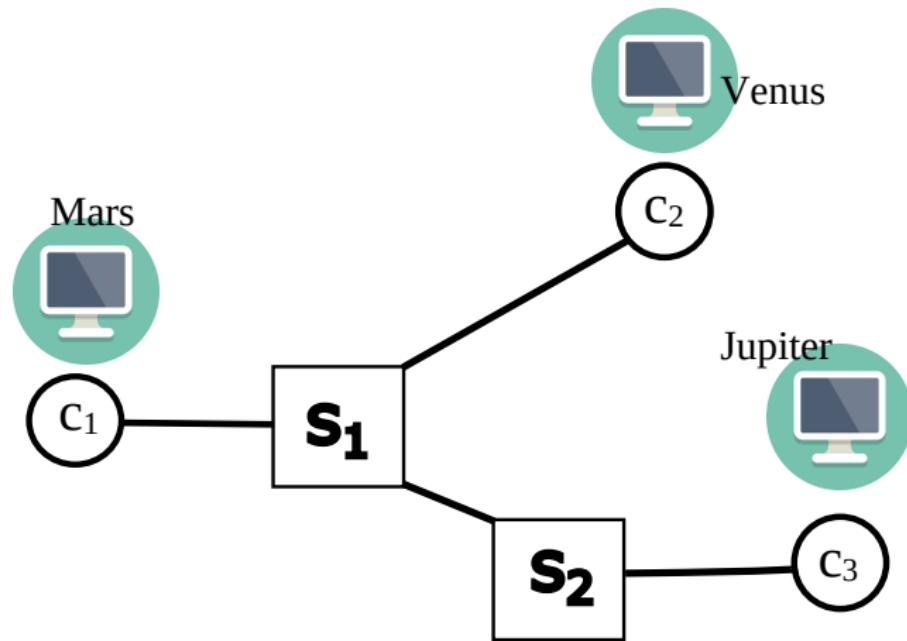
The knowledge about the family relationships in the above graph can be collected into the knowledge base KB .

Example: Family Tree VI

$KB \equiv \text{Female}(karen) \wedge \text{Female}(anne) \wedge \text{Female}(mary) \wedge \text{Female}(eve)$
 $\wedge \text{Female}(isabelle)$
 $\wedge \text{Child}(oscar, karen, franz) \wedge \text{Child}(mary, karen, franz)$
 $\wedge \text{Child}(eve, anne, oscar) \wedge \text{Child}(henry, anne, oscar)$
 $\wedge \text{Child}(isabelle, anne, oscar) \wedge \text{Child}(clyde, mary, oscar)$
 $\wedge (\forall X \forall Y \forall Z \text{ Child}(X, Y, Z) \Leftrightarrow \text{Child}(X, Z, Y))$
 $\wedge (\forall X \forall Y \text{Descendant}(X, Y) \Leftrightarrow$
 $\exists Z \text{Child}(X, Y, Z) \vee (\exists U \exists V \text{Child}(X, U, V) \wedge \text{Descendant}(U, Y)))$

We can now ask whether $\text{Child}(eve, oscar, anne)$ or $\text{Descendant}(eve, franz)$ can be derived from KB . This will be possible, when we have a calculus.

Example: Computer network (contd.) I



Example: Computer network (contd.) II

To connect two computers we need to have a path from the node of the first to the node of the second computer. We use the predicate Path:

$$\forall X \forall Y \text{Edge}(X, Y) \Rightarrow \text{Path}(X, Y)$$

$$\forall X \forall Y \forall Z \text{Path}(X, Y) \wedge \text{Edge}(Y, Z) \Rightarrow \text{Path}(X, Z)$$

The first formula states that two nodes X, Y are connected by a path, if there is an edge between them. The second formula state that two nodes X and Z are connected by a path, if there is a path between the nodes X, Y and an edge between Y and Z .

Two computers C_1 and C_2 are connected, if there is a path between their corresponding network nodes $f(C_1)$ and $f(R_2)$

$$\begin{aligned} \forall C_1 \forall C_2 \text{Computer}(C_1) \wedge \text{Computer}(C_2) \wedge \text{Path}(f(C_1), f(C_2)) \\ \Rightarrow \text{Connected}(C_1, C_2) \end{aligned}$$

Example: Computer network (contd.) III

If we know that there is path between two nodes we can also make the following statement

$$\forall X \forall Y (\text{Path}(X, Y) \Rightarrow \text{Edge}(X, Y) \vee \exists Z \text{Path}(X, Z) \wedge \text{Edge}(Z, Y))$$

expressing the fact there need to be certain edges in the graph, if there is a path between X and Y .

Equality in PL1

We want compare terms and define equality via a predicate " $=$ ", which we could also write $\text{eq}(x, y)$. We define equality as

$$\forall X \ X = X \quad (\text{reflexivity})$$

$$\forall X \forall Y \ X = Y \Rightarrow Y = X \quad (\text{symmetry})$$

$$\forall X \forall Y \forall Z \ X = Y \wedge Y = Z \Rightarrow X = Z \quad (\text{transitivity})$$

In addition we want functions to be unique. For a function f we have

$$\forall X \forall Y \ X = Y \Rightarrow f(x) = f(y) \quad \text{substitution axiom for functions}$$

In a similar way we require for predicate symbols

$$\forall X \forall Y \ X = Y \Rightarrow P(x) = P(y) \quad \text{substitution axiom for predicates}$$

Substitution of a variable by a term

Definition

We write $\Phi[X/t]$ for the formula that results when we replace every free occurrence of the variable X in Φ with the term t . Thereby we do not allow any variables in the term t that are quantified in Φ . In those cases variables must be renamed to ensure this.

Example

Consider the formula $\forall X X = Y$. Here, Y is a free variable. Replacing Y by the term $X + 1$ yields $\forall X X = X + 1$, which is incorrect. Instead, we correctly replace $\forall X X = Y + 1$ which has a different semantic.

Quantifiers and Normal Forms

The formula $\forall X P(X)$ is true, iff it is true for all interpretations of the variable X . One could write $P(c_1) \wedge P(c_2) \wedge \dots \wedge P(c_n)$ for all constants $c_1, \dots, c_n \in \mathcal{K}$. For $\exists X P(X)$ one could write $P(c_1) \vee P(c_2) \vee \dots \vee P(c_n)$. From this we find

$$\forall X \Phi \equiv \neg \exists X \neg \phi$$

Accordingly, universal and existential quantifiers are mutually replaceable.

Example

The propositions "Everybody wants to be loved." and "Nobody does not want to be loved" are equivalent.

Prenex normal form I

The quantifiers are not very easy to use for automatic reasoning, It is often useful to find equivalent formulas in a normal form.

Definition

A predicate logic formula Φ is in **prenex normal form** if for $Q_i \in \{\forall, \exists\}$, $i = 1, \dots, n$ we have

- $\Phi = Q_1 X_1 \cdots Q_n X_n \Psi$
- Ψ is a formula without any quantifiers.

Prenex normal form II

Examples

- Be careful with out of scope quantifiers: $\forall X A(X) \Rightarrow \exists X B(X)$.
Instead, we should write

$$\forall X A(X) \Rightarrow \exists Y B(Y).$$

Here, we can bring the quantifier to the front

$$\forall X \exists Y A(X) \Rightarrow B(Y).$$

Prenex normal form III

Examples

- To bring the quantifiers in front of

$$(\forall X A(X)) \Rightarrow \exists Y B(Y)$$

we write the formula in equivalent form

$$\neg(\forall X A(X)) \vee \exists Y B(Y)$$

Now we replace the universal quantifier

$$(\exists X \neg A(X)) \vee \exists Y B(Y)$$

and pull the two quantifiers to the front

$$\exists X \exists Y \neg A(X) \vee B(Y).$$

Prenex normal form IV

Examples

This formula is equivalent to

$$\forall X \exists Y A(X) \Rightarrow B(Y).$$

Remarks:

- First eliminate implications so that there are no negations to the quantifiers.
- In general, one can only pull quantifiers out if negations only exist directly on atomic sub-formulas.

Prenex normal form V

Theorem

Every predicate logic formula can be transformed into an equivalent formula in prenex normal form.

Elimination of existential quantifiers I

Skolemization

Existential quantifiers can be eliminated by replacing the variables under the scope of \exists by a function of the remaining variables. This is called **Skolemization**.

Example for illustration:

The formula

$$\forall U \forall V \exists X \forall W \exists Y \quad P(f(U), V, X) \vee Q(X, W, Y)$$

is in prenex normal form. The variable X is under the scope of the existential quantifier and depends on U and V , because these are still free at the level of this existential quantifier. We replace X by a Skolem function $g(U, V)$ (**new symbol!**) to eliminate the existential quantifier regarding correspondingt to X

$$\forall U \forall V \forall W \exists Y \quad P(f(U), V, g(U, V)) \vee Q(g(U, V), W, Y)$$

Elimination of existential quantifiers II

Skolemization

Analogously we replace Y by the skolem function $h(U, V, W)$

$$\forall U \forall V \forall W P(f(U), V, g(U, V)) \vee Q(g(U, V), W, h(U, V, W))$$

Now there are only universal quantifiers left. Now, we leave them out in the notation and obtain

$$P(f(U), V, g(U, V)) \vee Q(g(U, V), W, h(U, V, W)).$$

When a formula in prenex normal form is skolemized

Elimination of existential quantifiers III

Skolemization

- all existential quantifiers are eliminated from the outside inward, where a formula $\forall x_1 \dots \forall x_n \exists y \Phi$ is replaced by $\forall x_1 \dots \forall x_n \Phi(y/f(x_1, \dots, x_n))$, where f is a Skolem function.
- If an existential quantifier is far outside, such as in $\exists y P(y)$, then y must be replaced by constant (that is, by a zero-place function symbol).

Transformation of a formula to prenex normal form and skolemization

NormalFormTransformation(Formula)

- ① Transformation into prenex normal form:

Transformation of a formula to prenex normal form and skolemization

NormalForm Transformation (Formula)

- ① Transformation into prenex normal form:
 - ▶ Bring all the quantifiers to the left hand side

Transformation of a formula to prenex normal form and skolemization

NormalForm Transformation (Formula)

① Transformation into prenex normal form:

- ▶ Bring all the quantifiers to the left hand side
- ▶ Transformation of the remaining expression (without quantifiers) into conjunctive normal form (see propositional logic)

Transformation of a formula to prenex normal form and skolemization

NormalForm Transformation (Formula)

① Transformation into prenex normal form:

- ▶ Bring all the quantifiers to the left hand side
- ▶ Transformation of the remaining expression (without quantifiers) into conjunctive normal form (see propositional logic)
 - Elimination of equivalences and implications.

Transformation of a formula to prenex normal form and skolemization

NormalForm Transformation (Formula)

① Transformation into prenex normal form:

- ▶ Bring all the quantifiers to the left hand side
- ▶ Transformation of the remaining expression (without quantifiers) into conjunctive normal form (see propositional logic)
 - Elimination of equivalences and implications.
 - Repeated application of de Morgan's law and distributive law.

Transformation of a formula to prenex normal form and skolemization

NormalForm Transformation (Formula)

① Transformation into prenex normal form:

- ▶ Bring all the quantifiers to the left hand side
- ▶ Transformation of the remaining expression (without quantifiers) into conjunctive normal form (see propositional logic)
 - Elimination of equivalences and implications.
 - Repeated application of de Morgan's law and distributive law.
- ▶ Renaming of variables if necessary.

Transformation of a formula to prenex normal form and skolemization

NormalForm Transformation (Formula)

① Transformation into prenex normal form:

- ▶ Bring all the quantifiers to the left hand side
- ▶ Transformation of the remaining expression (without quantifiers) into conjunctive normal form (see propositional logic)
 - Elimination of equivalences and implications.
 - Repeated application of de Morgan's law and distributive law.
- ▶ Renaming of variables if necessary.
- ▶ Factoring out universal quantifiers.

Transformation of a formula to prenex normal form and skolemization

NormalForm Transformation (Formula)

① Transformation into prenex normal form:

- ▶ Bring all the quantifiers to the left hand side
- ▶ Transformation of the remaining expression (without quantifiers) into conjunctive normal form (see propositional logic)
 - Elimination of equivalences and implications.
 - Repeated application of de Morgan's law and distributive law.
- ▶ Renaming of variables if necessary.
- ▶ Factoring out universal quantifiers.

② Skolemization

Transformation of a formula to prenex normal form and skolemization

NormalForm Transformation (Formula)

① Transformation into prenex normal form:

- ▶ Bring all the quantifiers to the left hand side
- ▶ Transformation of the remaining expression (without quantifiers) into conjunctive normal form (see propositional logic)
 - Elimination of equivalences and implications.
 - Repeated application of de Morgan's law and distributive law.
- ▶ Renaming of variables if necessary.
- ▶ Factoring out universal quantifiers.

② Skolemization

- ▶ Replacement of existentially quantified variables by new Skolem functions.

Transformation of a formula to prenex normal form and skolemization

NormalForm Transformation (Formula)

① Transformation into prenex normal form:

- ▶ Bring all the quantifiers to the left hand side
- ▶ Transformation of the remaining expression (without quantifiers) into conjunctive normal form (see propositional logic)
 - Elimination of equivalences and implications.
 - Repeated application of de Morgan's law and distributive law.
- ▶ Renaming of variables if necessary.
- ▶ Factoring out universal quantifiers.

② Skolemization

- ▶ Replacement of existentially quantified variables by new Skolem functions.
- ▶ Deletion of resulting universal quantifiers.

Example I

Computer networks (contd.)

The formula

$$\forall X \forall Y (\text{Path}(X, Y) \Rightarrow \text{Edge}(X, Y) \vee \exists Z \text{Path}(X, Z) \wedge \text{Edge}(Z, Y))$$

expresses the fact there need to be certain edges in the graph, if there is a path between X and Y .

- ① Transformation into prenex normal form: Here we can shift all quantifiers to the left

$$\forall X \forall Y \exists Z (\text{Path}(X, Y) \Rightarrow \text{Edge}(X, Y) \vee \text{Path}(X, Z) \wedge \text{Edge}(Z, Y))$$

Example II

Computer networks (contd.)

Now we transform the expression without quantifiers into conjunctive normal form

$$\begin{aligned}\text{Path}(X, Y) &\Rightarrow \text{Edge}(X, Y) \vee \text{Path}(X, Z) \wedge \text{Edge}(Z, Y) \\ &\equiv \text{Edge}(X, Y) \vee \text{Path}(X, Z) \wedge \text{Edge}(Z, Y) \vee \neg\text{Path}(X, Y) \\ &\equiv (\text{Edge}(X, Y) \vee \text{Path}(X, Z) \vee \neg\text{Path}(X, Y)) \\ &\quad \wedge (\text{Edge}(X, Y) \vee \text{Edge}(Z, Y) \vee \neg\text{Path}(X, Y))\end{aligned}$$

- ② Skolemization To remove the existential quantifier for the variable Z we replace $Z/g(X, Y)$

$$\begin{aligned}&\text{Edge}(X, Y) \vee \text{Path}(X, g(X, Y)) \vee \neg\text{Path}(X, Y) \\ &\text{Edge}(X, Y) \vee \text{Edge}(g(X, Y), Y) \vee \neg\text{Path}(X, Y)\end{aligned}$$

Example III

Computer networks (contd.)

Here, we do not explicitly write \forall , but the all quantifier is implicitly assumed as well as the conjunctive connection of both clauses.

Proof calculi

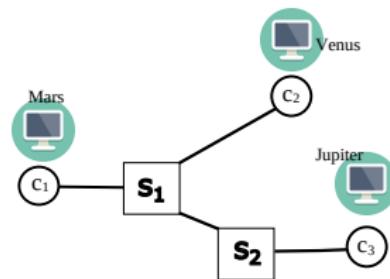
There are various proof calculi for PL1. We will focus on the resolution rule.

$$\frac{A \vee B_1 \vee \dots \vee B_n}{\neg A \vee C_1 \vee \dots \vee C_m} \quad \frac{}{B_1 \vee \dots \vee B_n \vee C_1 \vee \dots \vee C_m}$$

which remains true for PL1. However, there we need to add a process called **unification**, as illustrated by the following example.

Example I

Computer networks (contd.)



- We want to show that there is a path from node c_1 to node s_1 . We use the axioms (knowledge base)

$$\text{Edge}(c_1, s_1)$$

$$\text{Path}(X, Y) \vee \neg \text{Edge}(X, Y)$$

where the last formula is equivalent to the definition
 $\forall X \forall Y \text{ Edge}(X, Y) \Rightarrow \text{Path}(X, Y)$.

Example II

Computer networks (contd.)

- We can't apply the resolution rule, because $\text{Edge}(c_1, s_1)$ and $\neg\text{Edge}(X, Y)$ are not complementary.
- The last clause is only shorthand for $\forall X \forall Y \text{Path}(X, Y) \vee \neg\text{Edge}(X, Y)$ and is therefore valid for all interpretations.
- Therefore we can replace the variables by constants and we chose the replacements X/c_1 and Y/s_1 in the second clause.
- Now we can unify the two clauses to

$$\text{Edge}(c_1, s_1)$$

$$\text{Path}(c_1, s_1) \vee \neg\text{Edge}(c_1, s_1)$$

Example III

Computer networks (contd.)

- Resolution yields

$$\frac{\text{Edge}(c_1, s_1) \quad \neg \text{Edge}(c_1, s_1) \vee \text{Path}(c_1, s_1)}{\text{Path}(c_1, s_1)}$$

Unification I

Definition

Two literals are called **unifiable** if there is a substitution σ for all variables which makes the literals equal. Such a σ is called a **unifier**.

- Only variables can be substituted. Constants, function symbols or predicate name must not be replaced.
- A specific replacement of variables by a specific set of terms is called an **instantiation**.

Unification II

Example

The literals $P(X, f(Y))$ and $P(g(Z), f(a))$ can be unified in three different ways

$$\sigma_1 : \quad X/g(Z), \quad Y/a$$

$$\sigma_2 : \quad X/g(b), \quad Y/a, Z/b$$

$$\sigma_3 : \quad X/g(f(a)), \quad Y/a, Z/f(a)$$

and the unified formulas are

$$P(X, f(Y))_{\sigma_1} = P(g(Z), f(a)) = P(g(Z), f(a))_{\sigma_1}$$

$$P(X, f(Y))_{\sigma_2} = P(g(b), f(a)) = P(g(Z), f(a))_{\sigma_2}$$

$$P(X, f(Y))_{\sigma_3} = P(g(f(a)), f(a)) = P(g(Z), f(a))_{\sigma_3}$$

Unification III

Definition

A unifier is called the **most general unifier** if all other unifiers can be obtained from it by substitution of variables.

Example

For the literals $P(X, f(Y))$ and $P(g(Z), f(a))$ the unification $\sigma_1 : X/g(Z), Y/a$ is the most general unifier.

Unification IV

Unification algorithm:

Given: Two literals $A = P_A(s_1, \dots, s_m)$ and $B = P_B(t_1, \dots, t_m)$, which have different names for their variables.

- ① Check the predicate names and their arity. If the names are different $P_A \neq P_B$ or the predicates have a different number of arguments (arity), then the literals can not be unified.
- ② Unify the terms s_i and t_i for $i = 1, 2, \dots, n$ in the following way:
 - ▶ If s_i and t_i are constants with the same name, then they are unified. If they have different names, the literals A and B can not be unified.
 - ▶ If s_i is a variable and t_i a term (or vice versa), then the substitution s_i/t_i unifies them.

Unification V

- ③ If $s_i = f(u_1, \dots, u_p)$ and $t_i = g(v_1, \dots, v_q)$ are functions, then they can only be unified if they have the same names ($f = g$) and the same arity ($p = q$). Then, the two functions are unified by unifying their arguments u_i and v_i for all $i = 1, \dots, p$ one after the other using the method of step (2).

Result: The most general unifier for the literals A and B

Resolution rule for PL1 I

Theorem

Let σ be the most general unification (MGU) of the literals A and $\neg D$, i.e. $A_\sigma = D_\sigma$. From the true clauses $A \vee B_1 \vee \dots \vee B_n$ and $\neg D \vee C_1 \vee \dots \vee C_m$ containing these literals we can derive $(B_1 \vee \dots \vee B_n \vee C_1 \vee \dots \vee C_m)_\sigma$:

$$\frac{\begin{array}{c} A \vee B_1 \vee \dots \vee B_n \\ \neg D \vee C_1 \vee \dots \vee C_m \end{array} \quad \text{with} \quad A_\sigma = D_\sigma}{(B_1 \vee \dots \vee B_n \vee C_1 \vee \dots \vee C_m)_\sigma} \quad (3)$$

Remark: Both the parent clauses and the resolvent can contain variables.

Resolution rule for PL1 II

Theorem

The resolution rule is correct. That is, the resolvent is a semantic consequence of the two parent clauses.

For completeness, we need a small addition.

- It can happen, that some terms occur more than once in a clause or resolvent. For example, the resolvent could have the form $A \vee B \vee B \vee C$.
- Since $A \vee B \vee B \vee C \equiv A \vee B \vee C$ we can simplify this formula. This is called **factorization**.
- Factorization can also be applied to unified literals.

Resolution rule for PL1 III

Theorem (Factorization)

A clause can be factorized as

$$\frac{A \vee D \vee B_1 \dots \vee B_n \quad \text{with} \quad A_\sigma = D_\sigma}{(A \vee B_1 \dots \vee B_n)_\sigma} \quad (4)$$

where σ is the MGU of A and D .

Resolution rule for PL1 IV

Example (Russell paradox)

"There is a barber who shaves everyone who does not shave himself."

- This statement is unsatisfiable (Should the barber shave himself?)
- Formalization in PL1

$$\forall X \text{Shaves}(\text{barber}, X) \Leftrightarrow \neg \text{Shaves}(X, X)$$

- Transform this formula into a clause

$$\begin{aligned} & (\neg \text{Shaves}(\text{barber}, X) \vee \neg \text{Shaves}(X, X)) \\ & \quad \wedge (\text{Shaves}(\text{barber}, X) \vee \text{Shaves}(X, X)) \end{aligned}$$

Resolution rule for PL1 V

Example (Russell paradox)

- From this, we can not derive a contradiction. First, we need to unify both clauses by $\sigma : X/\text{barber}$ and obtain

$$\begin{aligned} & (\neg \text{Shaves}(\text{barber}, X) \vee \neg \text{Shaves}(X, X))_{\sigma_1} \\ & \equiv (\neg \text{Shaves}(\text{barber}, \text{barber}) \vee \neg \text{Shaves}(\text{barber}, \text{barber})) \\ & \equiv \neg \text{Shaves}(\text{barber}, \text{barber}) \\ & (\text{Shaves}(\text{barber}, X) \vee \text{Shaves}(X, X)) \\ & \equiv (\text{Shaves}(\text{barber}, \text{barber}) \vee \text{Shaves}(\text{barber}, \text{barber})) \\ & \equiv \text{Shaves}(\text{barber}, \text{barber}) \end{aligned}$$

Resolution rule for PL1 VI

Example (Russell paradox)

- Now we can apply resolution

$$\frac{\begin{array}{c} Shaves(\text{barber}, \text{barber}) \\ \neg Shaves(\text{barber}, \text{barber}) \end{array}}{\emptyset}$$

Resolution rule for PL1 VII

Theorem

The resolution rule (1) together with the factorization rule (2) is refutation complete. That is, by application of factorization and resolution steps, the empty clause can be derived from any unsatisfiable formula in conjunctive normal form.

Decidability I

Remember the proof task:

Proof task

Given the axioms (knowledge base) and a proposition (query), proof the query (if possible).

The theorem shows, that the resolution calculus is **not decidable**:

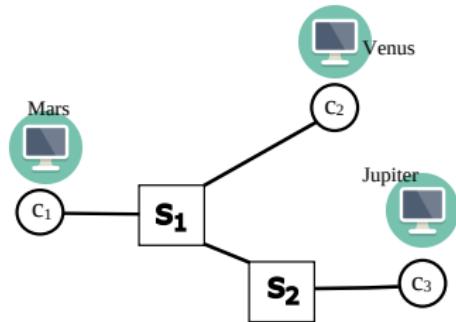
- Either, the empty clause can be generated (using proof by contradiction) after a finite number of steps and the query can be proofed, if it is a tautology (i.e. always true),
or
- the query is false and the proof doesn't terminate. Then, it is not clear whether the query is false or whether the proof system needs further resolution steps.

Decidability II

- Even, if the negation of the query can be used to partially overcome the problem of non-terminating algorithms, it can not decide, whether the query is satisfiable, but not a tautology (i.e. not general).

Example: Communication between computers (contd.) I

Communication network consisting of three computers c_1, c_2, c_3 named Mars, Venus, Jupiter and two switches s_1, s_2 .



Assertion: There is a connection between the computers Mars and Venus.

Connected(Mars, Venus)

Example: Communication between computers (contd.) II

Axioms:

$\text{Computer}(\text{Mars})$ (5)

$\text{Computer}(\text{Venus})$ (6)

$\text{Computer}(\text{Jupiter})$ (7)

$\text{Edge}(c_1, s_1)$ (8)

$\text{Edge}(s_1, c_2)$ (9)

$\text{Edge}(s_1, s_2)$ (10)

$\text{Edge}(s_2, c_3)$ (11)

$\neg \text{Edge}(X, Y) \vee \text{Path}(X, Y)$ (12)

$\neg \text{Path}(X, Y) \vee \neg \text{Edge}(Y, Z) \vee \text{Path}(X, Z)$ (13)

$\neg \text{Computer}(R_1) \vee \neg \text{Computer}(R_2) \vee \neg \text{Path}(f(R_1), f(R_2))$
 $\vee \text{Connected}(R_1.R_2)$ (14)

Example: Communication between computers (contd.) III

Here, f was the function which assigns node names to computer names (i.e. $c_1 = f(Mars)$).

- ① We use the so called **set of support** strategy which adds the negated assertion to the axioms:

$$\neg \text{Connected}(\text{Mars}, \text{Venus}) .$$

We use the substitution

$$\sigma_1 : R_1/\text{Mars}, R_2/\text{Venus}$$

to unify

$$\text{Connected}(\text{Mars}, \text{Venus}) \quad \text{and} \quad \text{Connected}(X, Y)$$

Example: Communication between computers (contd.) IV

The resulting complementary literals can be used in the first resolution step (see clause (12))

$$\frac{\begin{array}{l} \textit{Connected}(R_1.R_2) \vee \neg \textit{Computer}(R_1) \vee \neg \textit{Computer}(R_2) \vee \neg \textit{Path}(f(R_1), f(R_2)) \\ \neg \textit{Connected}(\textit{Mars}, \textit{Venus}) \end{array}}{(\neg \textit{Computer}(R_1) \vee \neg \textit{Computer}(R_2) \vee \neg \textit{Path}(f(R_1), f(R_2)))_{\sigma_1}}$$

which yields the resolvent (after unifying substitution)

$$\neg \textit{Computer}(\textit{Mars}) \vee \neg \textit{Computer}(\textit{Venus}) \vee \neg \textit{Path}(f(\textit{Mars}), f(\textit{Venus}))$$

Example: Communication between computers (contd.) V

- ② The last resolvent will be used in the next inference steps:

$$\frac{\begin{array}{c} \neg \text{Computer}(\text{Mars}) \vee \neg \text{Computer}(\text{Venus}) \vee \neg \text{Path}(f(\text{Mars}), f(\text{Venus})) \\ \text{Computer}(\text{Mars}) \end{array}}{\neg \text{Computer}(\text{Venus}) \vee \neg \text{Path}(f(\text{Mars}), f(\text{Venus}))}$$

and

$$\frac{\begin{array}{c} \neg \text{Computer}(\text{Venus}) \vee \neg \text{Path}(f(\text{Mars}), f(\text{Venus})) \\ \text{Computer}(\text{Venus}) \end{array}}{\neg \text{Path}(f(\text{Mars}), f(\text{Venus}))}.$$

- ③ Using $c_1 = f(\text{Mars})$ and $c_2 = f(\text{Venus})$ we obtain $\neg \text{Path}(c_1, c_2)$.
Now we use clause (10) and unify with the substitution

Example: Communication between computers (contd.) VI

$$\sigma_2 : X/c_1, \quad Y/c_2$$

to perform the next resolution step

$$\frac{\neg Path(c_1, c_2) \quad Path(X, Y) \vee \neg Edge(X, Y)}{(\neg Edge(X, Y))_{\sigma_2}} .$$

which yield after substitution σ_2 the clause $Edge(c_1, c_2)$. There is, however, no axiom which can be used in a resolution step with this last clause.

Example: Communication between computers (contd.) VII

- ④ Therefore, we use clause (11) with a complementary literal. With the unification

$$\sigma_3 : X/c_1, \quad Z/c_2$$

we have

$$\frac{\neg Path(c_1, c_2) \quad Path(X, Z) \vee \neg Path(X, Y) \vee \neg Edge(Y, Z)}{(\neg Path(X, Y) \vee \neg Edge(Y, Z))_{\sigma_3}} .$$

which yields $\neg Path(c_1, Y) \vee \neg Edge(Y, c_2)$. Please note, that here we have an example where the resolvent and also the parent clauses contain a variable.

Example: Communication between computers (contd.)

VIII

- ⑤ For the first literal in the last resolvent clause we find a complementary literal in clause (10). First, we need to rename the variables in (10)

$$\neg \text{Edge}(X_1, Y_1) \vee \text{Path}(X_1, Y_1)$$

and then we substitute

$$\sigma_4 : X_1/c_1, \quad Y_1/Y.$$

This yields a complementary literal for the last resolvent clause and we perform the resolution step

$$\frac{\begin{array}{c} \neg \text{Path}(c_1, Y) \vee \neg \text{Edge}(Y, c_2) \\ \text{Path}(c_1, Y) \vee \neg \text{Edge}(c_1, Y) \end{array}}{\neg \text{Edge}(Y, c_2) \vee \text{Edge}(c_1, Y)}.$$

Example: Communication between computers (contd.) IX

- ⑥ Now we substitute

$$\sigma_5 : Y/s_1$$

and use clause (7)

$$\frac{\begin{array}{c} \neg \text{Edge}(s_1, c_2) \vee \text{Edge}(c_1, s_1) \\ \text{Edge}(s_1, c_2) \end{array}}{\neg \text{Edge}(c_1, s_1)}.$$

- ⑦ Using axiom (6) we can obtain the empty clause

$$\frac{\begin{array}{c} \neg \text{Edge}(c_1, s_1) \\ \text{Edge}(c_1, s_1) \end{array}}{\emptyset}$$

and the assertion is proofed.

Chap. 4 Applications and limitations of logic in AI

Applications

Selected examples

- Planning: Plan the actions of a robotic
- Knowledge representation and reasoning
- Automatic theorem proofing in Mathematics

Planning

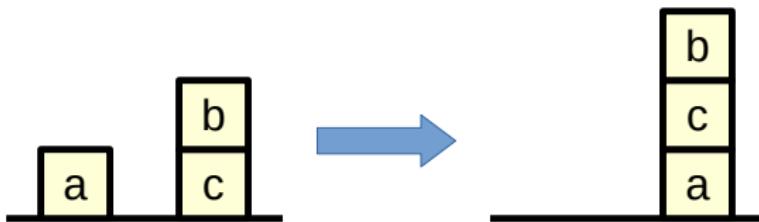


Figure: A planning task for a robot.

Planning tasks are characterized by

- an initial situation,
- actions, which can change the current situation
- a target situation.

STRIPS I

Stanford Research Institute Problem Solver

- describes a situation \mathcal{S} as a PL1 expression
- actions change a situation \mathcal{S} by deleting and adding clauses in \mathcal{S}
- actions can only be performed, if the situation \mathcal{S} fulfills certain constraints \mathcal{C}

Actions:

Name of the action:(parameters of the action)

Constraint: PL1 expression \mathcal{C} to be fulfilled by current \mathcal{S}

Action:

add: Clauses to be added to \mathcal{S}

delete: Clauses to be deleted from \mathcal{S}

STRIPS II

Stanford Research Institute Problem Solver

The constraint of the action is fulfilled if $\mathcal{S} \models \mathcal{C}$.

- The constraints \mathcal{C} are described by a set of clauses to be fulfilled in the situation \mathcal{S} .
- To proof $\mathcal{S} \models \mathcal{C}$ one adds $\neg\mathcal{C}$ to the set of clauses \mathcal{S} and generates an empty clause (proof by contradiction).

Planning the actions of a robot I



Figure: A planning task for a robot.

The planning task can be described in PL1 by initial and the desired final state

$$\begin{array}{ll} \textit{Bottom}(a) & \textit{Bottom}(a) \\ \textit{Bottom}(c) & \textit{Ontop}(c, a) \\ \textit{Ontop}(b, c) & \longrightarrow \textit{Ontop}(b, c) \\ \textit{Free}(a) & \textit{Free}(b) \\ \textit{Free}(b) & \end{array}$$

Planning the actions of a robot II

The robot can perform two different actions

- ① "stack" block 1 on top of block 2
- ② "lay down" of block 1, which is currently on top of block 2

Stacking(Block₁, Block₂)

Constraint: *Free(Block1)*
Free(Block2)

Action:

delete:	<i>Free(Block2)</i>
	<i>Bottom(Block1)</i>
add:	<i>Ontop(Block1, Block2)</i>

Planning the actions of a robot III

Laydown(Block₁, Block₂)

Constraint: *Free(Block1)*

Ontop(Block1, Block2)

Action:

delete:

Ontop(Block1, Block2)

add:

Free(Block2)

Bottom(Block1)

Compare the start and target situation in our example

Planning the actions of a robot IV



Figure: A planning task for a robot.

$Bottom(a)$	$Bottom(a)$
$Bottom(c)$	$OnTop(c, a)$
$OnTop(b, c)$	\longrightarrow
$Free(a)$	$OnTop(b, c)$
$Free(b)$	$Free(b)$

Planning the actions of a robot V

- ① We choose as the first action

$Laydown(b, c)$

Constraint: $Free(b)$
 $Ontop(b, c)$

Action:

delete:	$Ontop(b, c)$
add:	$Free(c)$
	$Bottom(b)$

Planning the actions of a robot VI

- ② To proof $\mathcal{S} \models \mathcal{C}$ we add the negation of $\mathcal{C} \equiv \neg Free(b) \wedge \neg Ontop(b, c)$ to the clauses describing the initial state

$\neg Free(b) \vee \neg Ontop(b, c)$

$Bottom(a)$

$Bottom(c)$

$Ontop(b, c)$

$Free(a)$

$Free(b)$

Planning the actions of a robot VII

Here, we can easily derive the empty clause. After performing the action, the new situation is given by

$\text{Bottom}(a)$

$\text{Bottom}(b)$

$\text{Bottom}(c)$

$\text{Free}(a)$

$\text{Free}(b)$

$\text{Free}(c)$

Planning the actions of a robot VIII

- ③ Next, we put block c on top of block a, i.e. we perform the action

Stacking(c, a)

Constraint: $Free(c)$
 $Free(a)$

Action:

delete: $Free(a)$
 $Bottom(c)$
add: $OnTop(c, a)$

Planning the actions of a robot IX

After checking the constraints (easy) and performing the action we obtain the situation

$\text{Bottom}(a)$

$\text{Bottom}(b)$

$\text{OnTop}(c, a)$

$\text{Free}(b)$

$\text{Free}(c)$

Planning the actions of a robot X

- ④ Now we put block b on top of block c to obtain the target situation

Stacking(b, c)

Constraint: $Free(b)$
 $Free(c)$

Action:

delete: $Free(c)$
 $Bottom(b)$
add: $OnTop(b, c)$

Search

To solve a planning task, one needs to find a series of actions leading from the initial state to the desired final state.

Search graphs:

- ① Nodes: represent the states
- ② Edges: Are present if there exists an action leading from one state to the other.

See e.g. *Russel, P. and Norvig, S, chapter 3* for search algorithms.

Knowledge representation

Aims

Humans understand, reason and interpret knowledge. Based on their knowledge, they perform various actions in the real world.

Knowledge representation

- Knowledge representation and reasoning (KR, KRR) is the part of AI concerned with approaches to representing information about the real world so that a computer can understand and can utilize this knowledge to solve the complex real world problems.
 - ▶ diagnosis of a medical condition
 - ▶ finding a new engineering design
 - ▶ find interactions between genes and proteins
 - ▶ ...
- KR is not just storing data into some database, but it also enables an intelligent machine to reason on the basis of that knowledge.

Kinds of knowledge to be represented in AI systems

Object: All the facts about objects in our world domain (e.g. cars have wheels, electrical cars have batteries, combustion cars have fuel tanks,...).

Events: Events are the actions which occur in our world (e.g. car drives, car breaks down, an accident,...).

Meta-knowledge: It is knowledge about what we know.

Facts: Facts are the truths about the real world and what we represent.

Knowledge base: Collects the known facts regarding objects and events in the form of sentences (as a technical term).

Types of knowledge I

① Declarative knowledge (descriptive knowledge):

- ▶ is to know about something,
- ▶ includes concepts, facts, and objects.

② Procedural knowledge (imperative knowledge):

- ▶ is about knowing how to do something,
- ▶ includes rules, strategies, procedures, agendas, etc.,
- ▶ is application dependent.

③ Heuristic knowledge:

- ▶ represents knowledge of some experts in a field or subject,
- ▶ rules of thumb based on previous experiences, awareness of approaches, and which are good to work with,
- ▶ not guaranteed.

④ Structural knowledge:

- ▶ describes relationships between various concepts such as kind of, part of, and grouping of something,

Types of knowledge II

- ▶ describes the relationship that exists between concepts or objects.

⑤ Meta-knowledge:

- ▶ is knowledge about the other types of knowledge.

Approaches to Knowledge Representation I

- Relational data base:

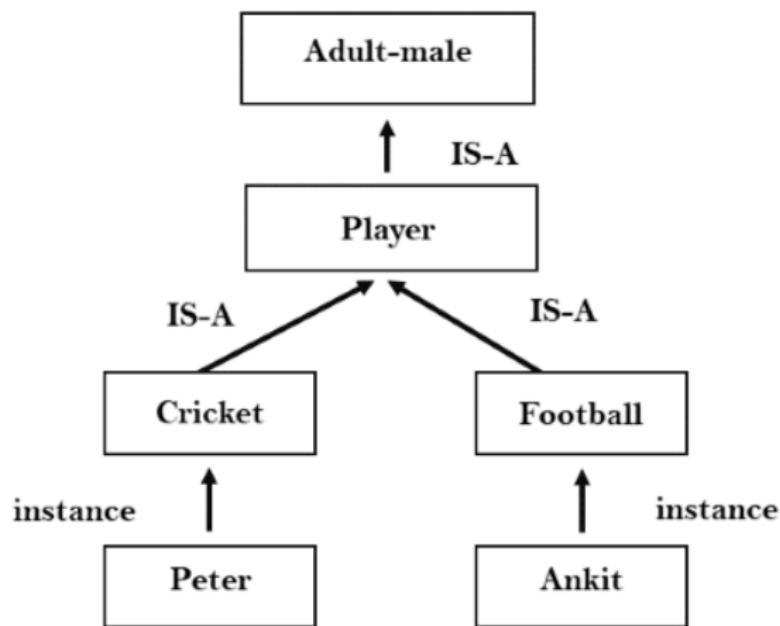
Name	Age	Goals	Assists
Heinze	22	22	45
Palo	23	46	12
Müller	29	128	221

- ▶ little opportunity for deductive inference

- Inheritable knowledge representation:

- ▶ all data are stored into a hierarchy of classes
- ▶ inheritable knowledge shows the relation between instance and class, and it is called instance relation
- ▶ Class hierarchies can be represented graphically with arrows pointing from objects to values

Approaches to Knowledge Representation II



Approaches to Knowledge Representation III

- Inferential knowledge representation:

- ▶ Knowledge is represented in the form of formal logics.
- ▶ One can derive more facts from the inferential knowledge base.
- ▶ Correctness is guaranteed.
- ▶ Example: Marcus is a man. All men are mortal.

$$\text{Man}(\text{Marcus})$$
$$\forall X \text{Man}(X) \Rightarrow \text{Mortal}(X)$$

- Procedural knowledge representation:

- ▶ uses small programs and codes which describes how to do specific things, and how to proceed.
- ▶ If-Then rules are used.
- ▶ heuristic or domain-specific knowledge can be represented.

Techniques for knowledge representation

There are four important techniques for KR:

- ① Logical Representation
- ② Semantic Network Representation
- ③ Frame Representation
- ④ Production Rules

Logical knowledge representation

Is based on logic (propositional or PL1)

Advantage:

- Enables us to do logical reasoning.

Disadvantages:

- Inference is often not efficient.
- Building knowledge bases requires familiarity with logic.

Semantic network representation I

- Semantic networks represent our knowledge in the form of graphical networks.
- Nodes represent objects and arcs describe the relationship between those objects.

Two types of relations:

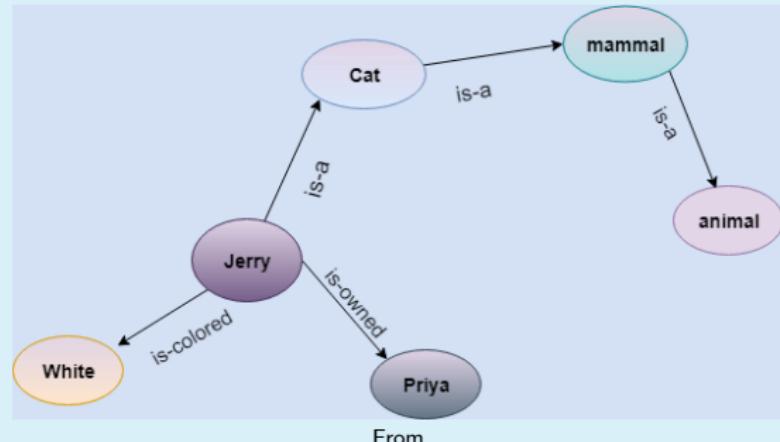
- IS-A relation (Inheritance)
- Kind-of-relation

Semantic network representation II

Example

Statements:

- Jerry is a cat.
- Jerry is a mammal.
- Jerry is owned by Priya.
- Jerry is brown colored.
- All Mammals are animals.



<https://www.javatpoint.com/ai-techniques-of-knowledge-representation>

Semantic network representation III

Advantages:

- are a natural representation of knowledge,
- convey meaning in a transparent manner,
- simple and easily understandable.

Disadvantages:

- computational time might be prohibitive at runtime as we need to traverse the complete network tree to answer some questions,
- no quantifiers, e.g., \forall, \exists
- do not have any standard definition for the link names,
- logic reasoning and entailment are difficult to implement and the utility depends on the creator.

Frame Representations I

- A frame is a record like structure which consists of a collection of attributes and its values to describe an entity in the world.
- consists of a collection of slots and slot values.
- Slots have names and values which are called **facets**.

Facets:

- are features of frames which enable us to put constraints on the frames

Frame Representations II

Example

A robot should put screws into a chassis. There are two tasks

- "Put a 2×12 screw into drill hole 1"
- "Put a 4×30 screw into drill hole 2"

The positions and the types of screws are collected as frames

Object:	Screw 2×12
Diameter	2
Length	12
Position X	220 mm
Position Y	280 mm

Object:	Screw 4×30
Diameter	4
Length	30
Position X	100 mm
Position Y	150 mm

Frame Representations III

Example

In addition, the chassis is described as a frame

Object: Chassis	
Drill hole: hole 1	
Diameter:	2 mm
X-Coordinate:	10 mm
Y-Coordinate:	15 mm
Drill hole: hole 2	
Diameter:	4 mm
X-Coordinate:	42 mm
Y-Coordinate:	90 mm

Frame Representations IV

Advantages:

- Programming is easier by grouping the related data (object oriented programming).
- It is very easy to add slots for new attribute and relations.
- It is easy to include default data and to search for missing values.
- Frame representations are easy to understand and visualize.

Disadvantages:

- Inference mechanisms can not be easily processed.
- Frame representations can not easily proceed inference mechanisms.

Production Rules I

- Production rules consist of (**condition, action**) pairs which mean, "If condition then action".
- A KR by a production rule is defined by
 - ▶ the set of production rules,
 - ▶ working memory,
 - ▶ the recognize-act-cycle.
- Recognize-act-cycle
 - ▶ The agent checks for the condition and if the condition is met, the production rule fires and the corresponding action is carried out.
 - ▶ The condition part of the rule determines which rule may be applied to a problem.
- Working memory
 - ▶ contains the description of the current state
 - ▶ A rule can write knowledge to the working memory, which in turn may fire other rules.

Production Rules II

Example

- IF (at bus stop AND bus arrives) THEN action (get into the bus)
- IF (on the bus AND paid AND empty seat) THEN action (sit down).
- IF (on bus AND unpaid) THEN action (pay charges).
- IF (bus arrives at destination) THEN action (get down from the bus).

Production Rules III

Advantages:

- Production rules are expressed in natural language.
- Modularity, i.e it is easy to modify, remove or add a rule.

Disadvantages:

- No learning capabilities. The rules perform neither deductive nor inductive inference.
- Computational inefficiency, if many rules are active.

Ontologies I

What do we need to exchange knowledge?

- Syntax: Joint concepts, symbols and operators.
- Semantics: A joint understanding of the meaning of a concept.
- Taxonomy: Classification of the concepts.
- Thesaurus: Relations between concepts
- Ontology: Rules and knowledge, about which rules are meaningful and allowed.

Example: I

SNOMED CT: a medical ontology

- Concept = medical term
- Example for a particular concept: “Sepsis due to disease caused by respiratory syndrome coronavirus 2”

Parents

> Organ dysfunction syndrome (disorder)

Sepsis due to disease caused by severe acute respiratory syndrome coronavirus 2 (disorder) ★ ↗
SCTID: 870588003
870588003 | Sepsis due to disease caused by severe acute respiratory syndrome coronavirus 2 (disorder)
en: Sepsis due to disease caused by severe acute respiratory syndrome coronavirus 2 (disorder)
en: Sepsis due to disease caused by COVID-19
en: Sepsis due to disease caused by 2019 novel coronavirus
en: Sepsis due to disease caused by 2019-nCoV
en: Sepsis due to disease caused by SARS-CoV-2
en: Sepsis due to disease caused by severe acute respiratory syndrome coronavirus 2

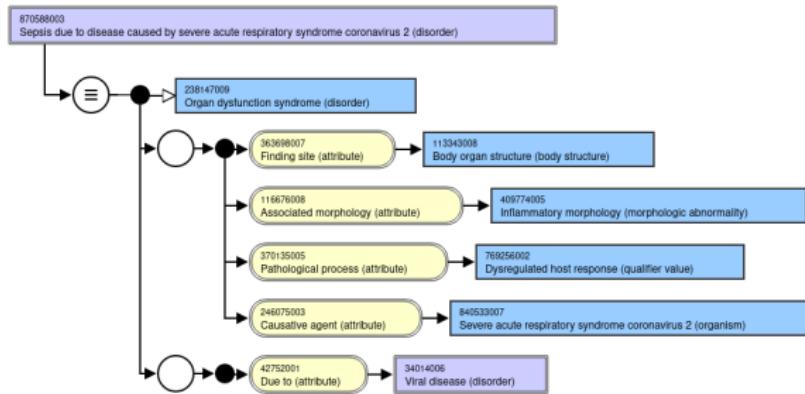
Axiom

Due to	→ Viral disease
Finding site	→ Body organ structure
Associated morphology	→ Inflammatory morphology
Pathological process	→ Dysregulated host response
Causative agent	→ Severe acute respiratory syndrome coronavirus 2

Children (0)
No children

Example: II

SNOMED CT: a medical ontology



Example: III

SNOMED CT: a medical ontology

```
EquivalentClasses(  
    :870588003 |Sepsis due to disease caused by severe acute respiratory syndrome coronavirus 2 (disorder)|  
    ObjectIntersectionOf(  
        :238147009 |Organ dysfunction syndrome (disorder)|  
        ObjectSomeValuesFrom(  
            :609096000 |Role group (attribute)|  
            ObjectIntersectionOf(  
                ObjectSomeValuesFrom(  
                    :116676008 |Associated morphology (attribute)|  
                    :409774005 |Inflammatory morphology (morphologic abnormality)|  
                )  
                ObjectSomeValuesFrom(  
                    :246075003 |Causative agent (attribute)|  
                    :840533007 |Severe acute respiratory syndrome coronavirus 2 (organism)|  
                )  
                ObjectSomeValuesFrom(  
                    :363698007 |Finding site (attribute)|  
                    :113343008 |Body organ structure (body structure)|  
                )  
                ObjectSomeValuesFrom(  
                    :370135005 |Pathological process (attribute)|  
                    :769256002 |Dysregulated host response (qualifier value)|  
                )  
            )  
        )  
    )  
    ObjectSomeValuesFrom(  
        :609096000 |Role group (attribute)|  
        ObjectSomeValuesFrom(  
            :42752001 |Due to (attribute)|  
            :34014006 |Viral disease (disorder)|  
        )  
    )  
)
```

Definition of an Ontology (Thomas R. Gruber, 1993)

Definition (Ontology)

An ontology is an explicit, formal specification of a shared conceptualization.

- Conceptualization: an abstract model.
- Domain: relevant terms ("concepts") and relations between them ("relations") in a certain field of knowledge ("domain").
- Explicit: all concepts must be defined.
- Formal: the machine must be able to process it.
- Shared: there is a consensus about the ontology.

Formal representation

Example

Informal representation

Lithium is a chemical element; it has symbol Li and atomic number 3. It is a soft, silvery-white alkali metal. Under standard conditions, it is the least dense metal and the least dense solid element. ...

(from Wikipedia)

Semi-Formal representation

Lithium

Name: $\langle String \rangle$

Symbol: $\langle String \rangle$

Atomic number : $Z = 3$

:

More examples

- Disease Ontology
 - ▶ Represents diseases via common terms and via "is-a" relations
 - ▶ <https://www.disease-ontology.org/>
- Gene Ontology
 - ▶ Represents the knowledge about genes.
 - Molecular Function
 - Cellular Component
 - Biological Process
 - ▶ <https://geneontology.org/>

Representation of Ontologies I

Classes, relations und Instances

- Ontologies can be represented as **classes**, **relations** and **instances**.
- Classes are groups or sets of objects and represent the concepts of the ontology
- Classes are characterized by their **attributes**.
- Attributes are pairs of the form (*Name, Value*)

Representation of Ontologies II

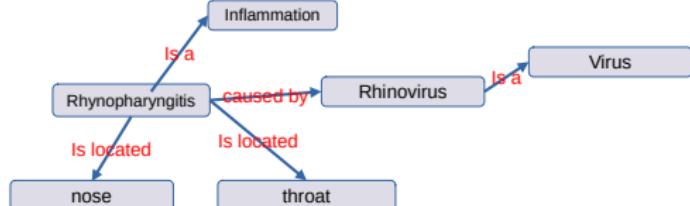
Classes, relations und Instances

Relations:

- relate different classes to each other
- are special attributes
- the values of a relation are objects of other classes

Informal representation

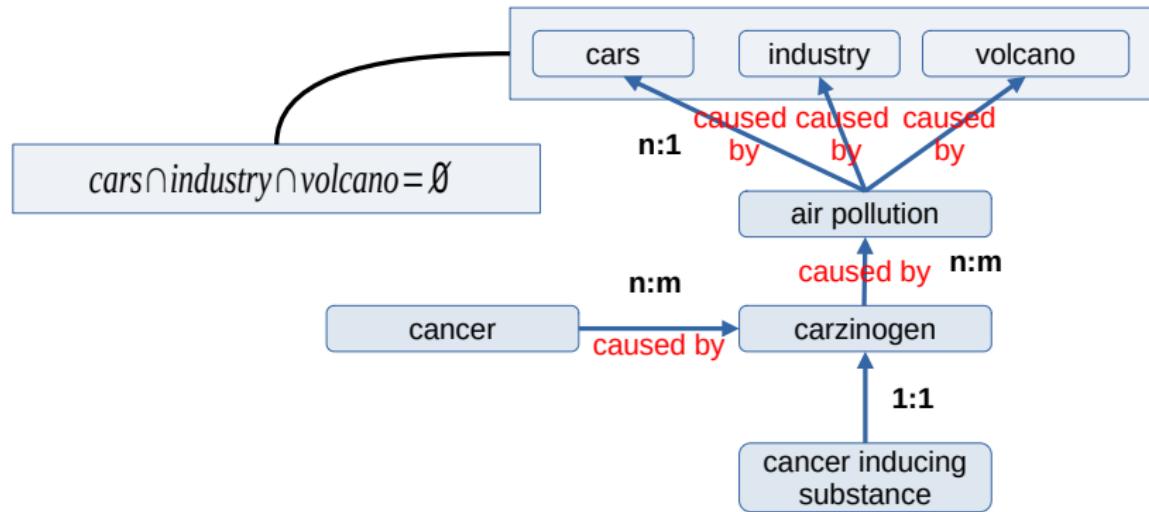
The common cold or the cold is a viral infectious disease of the upper respiratory tract that primarily affects the respiratory mucosa of the nose, throat, sinuses, and larynx.[6][8] Signs and symptoms may appear fewer than two days after exposure to the virus. These ... (from Wikipedia)



Representation of Ontologies III

Classes, relations und Instances

Rules: Rules (constraints) define allowed values for relations and attributes.



Representation of Ontologies IV

Classes, relations und Instances

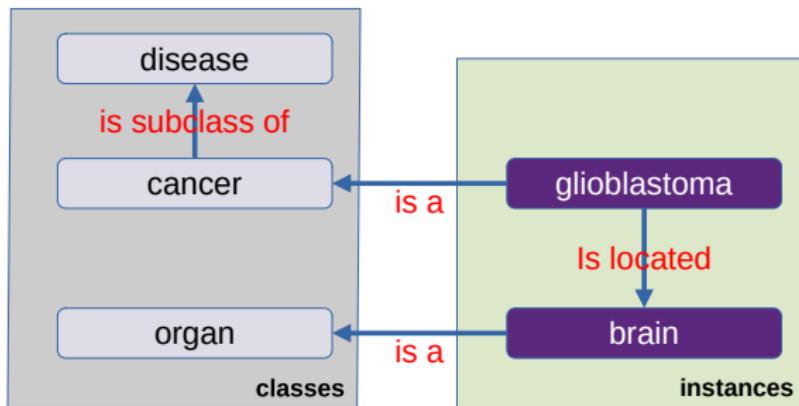
Propositions and Entailment

- Classes, relations and rules can be combined to form **propositions**.
- **Formal axioms:**
 - ▶ Formal axioms are special propositions, which are not expressed by the ontology itself.
 - ▶ Missing information is not declared as wrong, but unknown (**open world semantics**)
 - ▶ Example: "There is no clear evidence that COVID-19 is not a zoonosis."

Representation of Ontologies V

Classes, relations und Instances

Instances: describe an individuum in an ontology.



Ontologies and logic

Web Ontology Language (OWL) (<https://www.w3.org/OWL/>)

- is a family of knowledge representation languages for authoring ontologies.
- OWL implements **description logic**
- Description logic contains a well defined subset of PL1, which is decidable.
- There are inference engines for ontologies implemented in OWL.
- There is also an ontology editor called Protege to design ontologies.

Limitations of logic approaches to AI

- Search space problem.
- Decidability and incompleteness.
- Modeling uncertainty

The search space problem. I

- At every step of an deductive inference problem, there are many possibilities for the application of an inference rule.
- Search space problem: An explosive growth of the search space
- Worst case: All possible sequences of applying an inference rule must be tried



Figure: From Ertel, Introduction to AI, Springer, 2017, Fig. 4.1

The search space problem. II

Humans use heuristics to restrict the search space.

Heuristic proof controlling modules:

- Evaluate the various alternatives for the next step and then choose the alternative with the best rating.
- Resolution rule: the rating of the available clauses can be based on the number of positive literals, the complexity of the terms, etc., for every pair of resolvable clauses.
- Use a machine learning classifier:
 - ▶ Variables: attributes of all pairs of possible resolution steps
 - ▶ Classes: Positive resolutions, negative resolutions
 - ▶ Training data: Use an automatic theorem prover and save as training data for the positive class the variables for every proof found at every branch during the search. Negative examples are given by branches not leading to a proof.

The search space problem. III

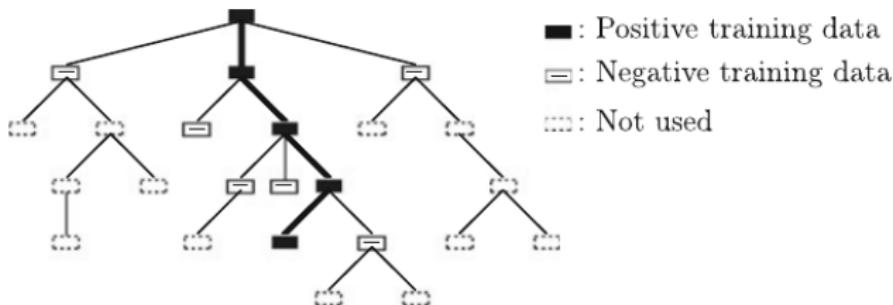


Figure: A search path of an automatic theorem prover. Successful proofs follow paths labeled as positive examples and unsuccessful proofs follow paths labeled as negatives. From Ertel, Introduction to AI, Springer, 2017, Fig. 4.1

Decidability

Theorem

The set of valid formulas in first-order predicate logic is semidecidable.

- A theorem, i.e. a true statement, a prover based on a complete calculus (like resolution calculus) can verify the truth of the statements.
- For statements that are false, the prover might never halt. Then, it is not possible to decide, whether the prover needs more time to proof a true statement, or whether the statement is false.

Incompleteness I

- Quantifiers over functions or predicates are not included in PL1 (only over variables)
- Sometimes, such an extension might be desirable

Example

Induction:

"If a predicate $p(n)$ holds for n , then $p(n + 1)$ also holds"

$$\forall p p(n) \rightarrow p(n + 1)$$

However, such extensions will come for the price of loosing decidability.

Incompleteness II

Theorem (Gödel's incompleteness theorem)

Every axiom system for the natural numbers with addition and multiplication (arithmetic) is incomplete. That is, there are true statements in arithmetic that are not provable.

Idea behind Gödel's Theorem:

- Gödelization: every arithmetic formula is encoded as a unique number
- The proposition

$$F = \text{"I am not provable."}$$

is formulated in the language of arithmetic.

- This formula is true for the following reason:
 - ▶ Assume F is false.
 - ▶ Then we can prove F and therefore show that F is not provable. This is a contradiction.
 - ▶ Thus F is true and therefore not provable.

Incompleteness III

Practical significance:

- Mathematical theories (axiom systems) become incomplete if the language becomes too powerful.
- Example: Set theory and Russel's paradox.
- Still, undecidable axiom systems might be useful.

Monotonicity of Logic I

Example (Flying penguin)

The following statements

- ① Tweety is a penguin.
- ② Penguins are birds.
- ③ Birds can fly

can be formalized in PL1 as a knowledge base (forall quantifier suppressed)

$$\text{Penguin}(\text{tweety})$$

$$\text{Penguin}(X) \Rightarrow \text{Bird}(X)$$

$$\text{Bird}(X) \Rightarrow \text{Fly}(X).$$

one can derive (e.g. with resolution) $\text{Fly}(\text{tweety})$.

Monotonicity of Logic II

Example (Flying penguin)

This shows, that we need to introduce an exception for penguins. We try

$$\text{Penguin}(X) \Rightarrow \neg \text{Fly}(X)$$

From this, we can derive $\neg \text{Fly}(\text{tweety})$. But, the **knowledge base is inconsistent**, because $\text{Fly}(\text{tweety})$ can still be derived.

Monotonicity of Logic III

Theorem (Monotonic logic)

A logic is called **monotonic** if, for an arbitrary knowledge base \mathcal{A} and an arbitrary formula Φ , the set of formulas derivable from \mathcal{A} is a subset of the formulas derivable from $\mathcal{A} \cup \phi$.

- If a set of formulas is extended, then, all previously derivable statements can still be proved, and additional statements can potentially also be proved.
- The extension of the knowledge base will never lead to our goal.

Monotonicity of Logic IV

Example (Flying penguin (contd.))

We try to modify the knowledge base by correcting the obviously false statement “*(all) birds can fly*” by “*(all) birds except penguins can fly*”. The \mathcal{A}_2 reads

$$\text{Penguin}(\text{tweety})$$
$$\text{Penguin}(X) \Rightarrow \text{Bird}(X)$$
$$\text{Bird}(X) \wedge \neg \text{Penguin}(X) \Rightarrow \text{Fly}(X)$$
$$\text{Penguin}(X) \Rightarrow \neg \text{Fly}(X)$$

Now, we derive $\neg \text{Fly}(\text{tweety})$, but not $\text{Fly}(\text{tweety})$ and \mathcal{A}_2 is consistent.

Monotonicity of Logic V

Example (Flying penguin (contd.))

Now, we add $\text{Raven}(\text{abraxas})$

$\text{Raven}(\text{abraxas})$

$\text{Raven}(X) \Rightarrow \text{Bird}(X)$

$\text{Penguin}(\text{tweety})$

$\text{Penguin}(X) \Rightarrow \text{Bird}(X)$

$\text{Bird}(X) \wedge \neg \text{Penguin}(X) \Rightarrow \text{Fly}(X)$

$\text{Penguin}(X) \Rightarrow \neg \text{Fly}(X)$

We are unable to say anything about the flying capability of Abraxas.

Monotonicity of Logic VI

Example (Flying penguin (contd.))

We need to state, that ravens are not penguins:

$Raven(abraxas)$

$Raven(X) \Rightarrow Bird(X)$

$Raven(X) \Rightarrow \neg Penguin(X)$

$Penguin(tweety)$

$Penguin(X) \Rightarrow Bird(X)$

$Bird(X) \wedge \neg Penguin(X) \Rightarrow Fly(X)$

$Penguin(X) \Rightarrow \neg Fly(X)$

There are about 10000 different bird species and for each of one, we would have to state, that they are not penguins.

Non-monotonic Logic

- Logic systems, which allows knowledge (formulas) to be removed from the knowledge base, are called **non-monotonous logics**.
- Example: Default logic

Modeling Uncertainty

- Sometimes, a proposition is not just true or false.
- There are different approaches to incorporate uncertainty
 - ▶ Fuzzy logic and fuzzy sets
 - ▶ Probabilistic logic: We want to assign a probability to a certain statement to express uncertainty, like $\text{Prob}(\text{Bird}(X) \Rightarrow \text{Fly}(X)) = 0.99$.

Chap. 5 Probability theory and probabilistic logic

Reasoning under uncertainty

- In the real world, we rarely have complete information and knowledge.
- Decision making occurs under constraints (little time and little, potentially uncertain knowledge)

Example

A patient with pain in the lower abdomen and a raised leukocyte (white blood cell) count comes to the doctor. Based on these symptoms, the doctor thinks he has an appendicitis.

Modelling with propositional logic:

$$\text{StomachPainRightLower} \wedge \text{LeukocytesHigh} \Rightarrow \text{Appendicitis}.$$

So, each patient with these symptoms will be diagnosed an appendicitis. However, there are many other possible causes for these symptoms.

Reminder of basic probability concepts

Sample space

What can happen?

Sample space: A set $\Omega = \{\omega_1, \omega_2, \dots\}$ containing the possible outcomes of a random experiment.

Example

Some sample spaces

- ① Coin tossing $\Omega = \{H, T\}$
- ② Rolling a die $\Omega = \{1, 2, 3, 4, 5, 6\}$
- ③ The height of a person $\Omega = \{\omega \mid \omega \in (0, 300)\}$

Each element $\omega \in \Omega$ is called a sample point or an elementary event. An event A is always a subset of the sample space: $A \subset \Omega$. The empty set \emptyset is the impossible event. The certain event is Ω itself.

Events and elementary events

Definition (Events and elementary events)

Let Ω be the finite set of **events** for an experiment. Each event $\omega \in \Omega$ represents a possible outcome of the experiment. If these events $\omega_i \in \Omega$ mutually exclude each other, but cover all possible outcomes of the attempt, then they are called elementary events.

Events and sample space

Example

Some events

- ① $\Omega = \{H, T\}$ and all possible events are $\{H\}, \{T\}, \{H, T\}, \emptyset$. Only $\{H\}$ and $\{T\}$ are elementary events.
- ② $\Omega = \{1, 2, 3, 4, 5, 6\}$ and $A = \{2, 4, 6\}$ is the event that an even number will be rolled.
- ③ The number of customers visiting a shop: $\Omega = \{0, 1, 2, \dots\}$ and $\omega_i = \{i\}, i \in \mathbb{N}$ are the elementary events. The event that between 100 and 150 customers are visiting is $A = \{100, 101, \dots, 200\}$.
- ④ $\Omega = \{\omega | \omega \in (0, 300)\}$ and $A = (160, 180) \subset \Omega$ is the event that the persons height is between 160 and 180 centimeters. Another event is that the person is shorter than 180 centimeters, i.e.
 $B = \{x \in \Omega | x < 180\}$.

Set theory and random events

Let Ω be a sample space and $A, B \subset \Omega$ be events.

Set	Set theoretic meaning	Probabilistic meaning
$A \setminus B$	complementary set B relative to A	A happens, but not B
$A^c = \Omega \setminus A$	complementary set of A	A does not happen
$A \cap B$	intersection of sets A and B	A and B happen
$A \cup B$	union of sets A and B	A or B or both happen
Ω	superset	certain event
\emptyset	empty set	impossible event
$A \cap B = \emptyset$	A and B are disjoint	A and B are exclusive events (can not happen together)

Not all subsets of the sample space Ω are events

- The goal is to assign probabilities to events.
- For finite or countable infinite sample spaces, the power set (set of all subsets) of Ω can be assigned a probability.
- For uncountable sets, only some subsets of Ω qualify as events. Otherwise, there can be mathematical contradictions (Measure theory).
- Therefore, we need to introduce conditions of the notion of a *sigma algebra*.

Sigma algebra

Definition

Let \mathcal{F} be a collection of subsets of Ω . Then, Ω is called a σ -algebra, if \mathcal{F} has the following properties

①

$$\Omega \in \mathcal{F}$$

- ② if $A \in \mathcal{F}$, then the complementary event $A^c = \Omega \setminus A$ is also an element of \mathcal{F} .
- ③ For any sequence $B_n \subset \mathcal{F}$ of events, the union

$$\cup_{n=1}^{\infty} B_n \subset \mathcal{F}$$

If \mathcal{F} is a σ -algebra, then we call the ordered pair (Ω, \mathcal{F}) a *measurable space*.

Probability spaces I

Definition

Let (Ω, \mathcal{F}) be a measurable space and $P : \mathcal{F} \rightarrow [0, 1]$ be a real valued function assigning a number between zero and one to any event. We call this function a *probability measure*, if

- ① $P(A) \geq 0$ for all $A \in \mathcal{F}$
- ② $P(\Omega) = 1$
- ③ $P(\bigcup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} P(A_i)$ for any collection $A_i \in \mathcal{F}, i = 1, 2, \dots$ of pairwise disjoint events $A_i \cap A_j = \emptyset, i \neq j$.

We call the ordered triple (Ω, \mathcal{F}, P) a probability space.

Often it is not necessary to define the underlying measurable space (Ω, \mathcal{F}) . For a finite sample space Ω , the σ -algebra will be taken to be all subsets of Ω . If $\Omega = \mathbb{R}$, then we can take Borel measurable sets.

Probability spaces

Examples I

Example

For coin tossing we have $\Omega = \{H, T\}$ and $\mathcal{F} = \{\{H\}, \{T\}, \{H, T\}, \emptyset\}$. Then, each map $P : \mathcal{F} \rightarrow (0, 1)$ is a probability measure.

- Fair coin:

$$P(\{H\}) = \frac{1}{2}, \quad P(\{T\}) = \frac{1}{2}$$

$$P(\{H\} \cup \{T\}) = P(\{H\}) + P(\{T\}) = 1 = P(\Omega)$$

- A biased coin:

$$P(\{H\}) = \frac{1}{3}, \quad P(\{T\}) = \frac{2}{3}$$

$$P(\{H\} \cup \{T\}) = P(\{H\}) + P(\{T\}) = 1 = P(\Omega)$$

Probability spaces

Examples II

Example

From $P(\Omega) = 1$ and we have $P(A^c \cup A) = P(A^c) + P(A)$

Example

With $\emptyset = \Omega^c$ it follows that $P(\emptyset) = 0$.

Example

Body height in cm: $\Omega = (0, 300)$

$A = \{(150, 160)\}$, $B = \{[160, 200)\}$ and $A \cap B = \emptyset$

$$\begin{aligned}P(A \cup B) &= P((150, 200)) \\&= P(A) + P(B) = P((150, 160)) + P([160, 200))\end{aligned}$$

Conditional probability

Definition (Conditional probabilities)

Let (Ω, \mathcal{F}, P) be a probability space and $A, B \in \mathcal{F}$ be two events. The conditional probability of the event A given the event B is defined as

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

provided that $P(B) > 0$.

Example

Rolling a die: Let and $A = \{6\}$ and $B = \{2, 4, 6\}$ and thus $A \cap B = \{6\}$.

$$P(A \cap B) = \frac{1}{6}, \quad P(B) = \frac{1}{2}, \quad P(A|B) = \frac{1}{3}$$

Law of total probability I

Theorem (Total probability)

Let A and $B_1, \dots, B_m \subset \Omega$ be events. If the events are mutually exclusive and exhaustive

$$B_i \cap B_j = \emptyset \quad \forall i \neq j, \quad i, j = 1, \dots, m$$

$$\cup_{i=1}^m B_i = \Omega$$

we have

$$P(A) = \sum_{i=1}^m P(A|B_i)P(B_i)$$

Law of total probability II

Example

A company owns two production sites for computer chips of the same type. Site 1 produces 70% of the chips and 10% of the chips are defect. Site 2 produces the remaining 30% and their defect rate is 20%. What is the probability that the company produces a chip with a defect?

Let A and B be the events that a product comes from site 1 or 2, respectively, and D be the event of a defect. Then

$$P(D) = P(D|A)P(A) + P(D|B)P(B) = \frac{1}{10} \frac{7}{10} + \frac{2}{10} \frac{3}{10} = \frac{13}{100}.$$

Bayes Formula I

Theorem (Bayes Formula)

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

as long as $P(B) > 0$.

Proof.

$$P(A \cap B) = P(A|B)P(B) = P(B|A)P(A).$$

□

Conditional probability and Bayes Formula I

Assume, a diagnostic test for a rare disease with incidence rate 0.01 detects the disease in 99% of all patients carrying the disease and it has a false positive detection probability of 2%. What is the probability that a person who is positively tested actually carries the disease?

Let D be the event that the person carries the disease and T be the event, that the test is positive. Thus, we know $P(T|D) = 0.99$, $P(D) = 0.01$ and $P(T|D^c) = 0.02$. Then

Conditional probability and Bayes Formula II

$$\begin{aligned}P(T) &= P(T \cap D) + P(T \cap D^c) \\&= P(T|D)P(D) + P(T|D^c)P(D^c) \\&= P(T|D)P(D) + P(T|D^c)(1 - P(D)) \\&= 0.99 \cdot 0.01 + 0.02 \cdot (1 - 0.01) \\&= 0.021\end{aligned}$$

$$\begin{aligned}P(D|T) &= \frac{P(T|D)P(D)}{P(T)} \\&= \frac{0.99 \cdot 0.01}{0.021} \\&= 0.476\end{aligned}$$

Thus, only less than half of the persons tested positively actually carry the disease.

Independence I

Definition

Two events A and B are called independent ($A \perp B$), if

$$P(A \cap B) = P(A) P(B).$$

Corollary

A and B are independent if and only if $P(A|B) = P(A)$ and if and only if $P(B|A) = P(B)$.

Proof.

Reading exercise. □

Random variables

Definition

Let (Ω, \mathcal{F}) be a measurable space. A real valued function

$$X : \Omega \rightarrow \mathbb{R}$$

from the sample space Ω with the property

$$X^{-1}(-\infty, a) = \{\omega \in \Omega \mid X(\omega) \leq a\} \in \mathcal{F}$$

is called a *random variable*. The range of X

$$\mathcal{X} = \{x \mid X(\omega) = x, \omega \in \Omega\}$$

is also called the state space of X . If \mathcal{X} is finite or countable finite, then X is a discrete random variable. If \mathcal{X} is a possibly infinite interval, its called a continuous random variable. The random variable can also be of mixed type.

Random variables and cumulative distribution functions

Notation for the events associated with a random variable:

- $X = x$ indicates the event $\{\omega \in \Omega \mid X(\omega) = x\}$
- $X \leq x$ is the notation for the event $\{\omega \in \Omega \mid X(\omega) \leq x\}$

This enables us to associate a probability measure $P_X : \mathbb{R} \rightarrow [0, 1]$ with a random variable X . Often, we do not actually specify the measurable space (Ω, \mathcal{F}) , but only the probability measure or the cumulative distribution function defined below.

Definition

The cumulative distribution function (c.d.f.) of the random variable is the function

$$\begin{aligned} F : \mathbb{R} &\rightarrow [0, 1] \\ x &\mapsto F(x) = P_X(X \leq x). \end{aligned}$$

Probability mass functions for discrete random variables

Definition

For a discrete random Variable X we call the function $p(x) = P_X(X = x)$ for $x \in \mathcal{X}$ the **probability mass function** (p.m.f) of X .

It follows from the basic axioms of probability that

$$\sum_{x \in \mathcal{X}} p(x) = 1$$

$$\sum_{x \in B} p(x) = P_X(X \in B)$$

$$F(x) = \sum_{a \leq x} p(a).$$

Example

For a fair dye with $\Omega = \{1, 2, 3, 4, 5, 6\}$ we have $P(X \leq 2) = \frac{1}{3}$ and $P(X \leq 3) = \frac{1}{2}$. Also, $P(X > 2) = 1 - P(X \leq 2) = \frac{2}{3}$.

Bernoulli and Binomial distribution I

Example for discrete random variables

Assume that you toss a coin n times and the probability of heads is $\theta \in [0, 1]$. The random variable

$$X_i = \begin{cases} 1 & : \text{heads} \\ 0 & : \text{tails} \end{cases}$$

indicates whether the i -th trial is a success or not. It has the p.m.f.

$$\begin{aligned} p_{X_i}(x) &= P(X_i = x_i) = \begin{cases} \theta & : x_i = 1 \\ 1 - \theta & : x_i = 0 \end{cases} \\ &= \theta^{x_i} (1 - \theta)^{1-x_i} \end{aligned}$$

Bernoulli and Binomial distribution II

Example for discrete random variables

Then, define the number of successes

$$Y = \sum_{i=1}^n X_i$$

The p.m.f. is given by

$$p_Y(y) = P_Y(Y = y) = \begin{cases} \binom{n}{y} \theta^y (1 - \theta)^{n-y} & : y = 0, 1, 2, \dots, n \\ 0 & : \text{otherwise.} \end{cases}$$

Here, we have used the binomial coefficient

$$\binom{n}{y} = \frac{n!}{y! (n - y)!}.$$

Bernoulli and Binomial distribution III

Example for discrete random variables

and the convention $0! = 1$. For $y \in \{0, 1, 2, \dots, n\}$, the c.d.f.

$$F_Y(y) = \sum_{k \leq y} p_Y(k) = \sum_{k=0}^y \binom{n}{k} \theta^k (1-\theta)^{n-k}$$

is the probability of having at most y successes.

- Notation: $X_i \sim \text{bernoulli}(\theta)$ and $Y \sim \text{binomial}(n, \theta)$.

Bernoulli and Binomial distribution IV

Example for discrete random variables

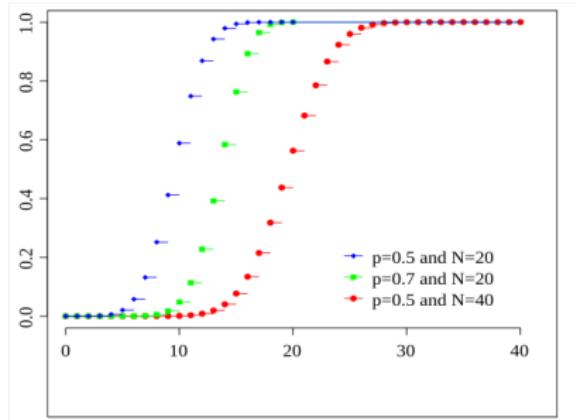
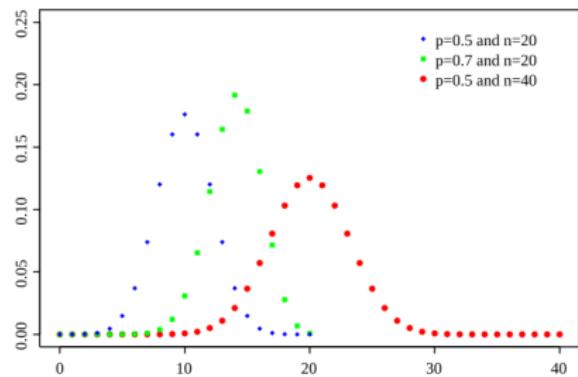


Figure: Probability mass function (left) and cumulative distribution function(right) for different parameters θ and n .

Density functions for continuous random variables I

Definition

For a continuous random variable X with c.d.f. F_X we define the density of X to be the nonnegative, integrable function $p_X : \mathbb{R} \rightarrow [0, 1]$ with

$$F_X(x) = \int_{-\infty}^x p_X(x)dx.$$

Note, that not every continuous random variable has a density function.

Properties of densities:

Let $A \subset \mathcal{X}$ be an event. Then

$$P_X(x \in \mathcal{X}) = \int_{-\infty}^{\infty} p_X(x)dx = 1$$

$$P_X(x \in A) = \int_{x \in A} p_X(x)dx$$

Density functions for continuous random variables II

Special cases are

$$P_X(a \leq X \leq b) = \int_a^b p_X(x) dx = F(b) - F(a)$$

and for continuous random variables we have

$$P(a < X < b) = Pr(a \leq X < b) = Pr(a < X \leq b) = P(a \leq X < b).$$

If the c.d.f is differentiable, then

$$p_X(x) = \frac{d}{dx} F_X(x) \quad \forall x \in \mathcal{X}.$$

Exponential distribution I

An example for a continuous distribution

The exponential distribution with density

$$p_X(x) = \lambda e^{-\lambda x}, \quad x \geq 0$$

and $p_X = 0$ for $x < 0$. Then

$$\begin{aligned} F_X(x) &= \int_{-\infty}^x p_X(\xi) d\xi \\ &= \int_{-\infty}^x \lambda e^{-\lambda \xi} d\xi \\ &= \int_0^x \lambda e^{-\lambda \xi} d\xi \\ &= \left(1 - e^{-\lambda x}\right) \end{aligned}$$

Notation: $X \sim \text{Exp}(\lambda)$

Exponential distribution II

An example for a continuous distribution

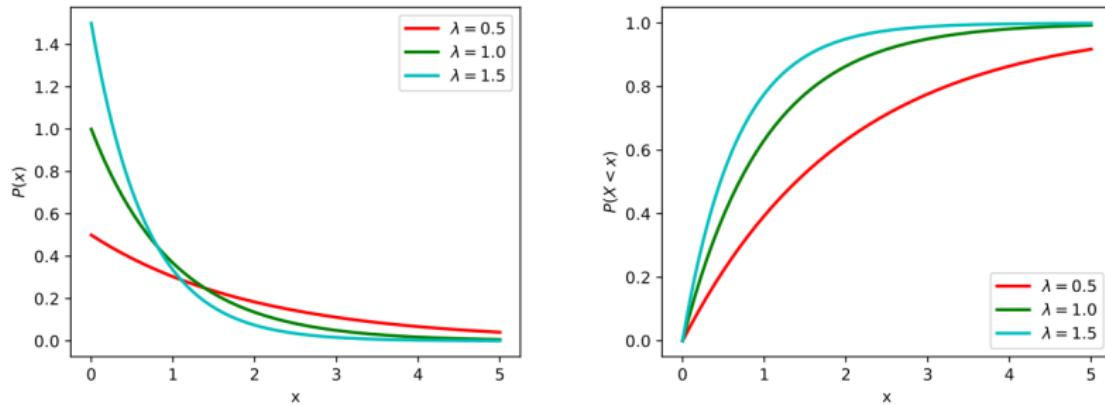


Figure: Density (left) and cumulative distribution function (right) for the exponential distribution for different parameters λ .

The Gaussian distribution I

Another example for a continuous distribution

The Gaussian distribution (normal distribution)

$$p_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \quad -\infty < x < \infty$$

with parameters $\sigma > 0$ and $\mu \in \mathbb{R}$.

Notation: $X \sim \mathcal{N}(\mu, \sigma^2)$

The Gaussian distribution II

Another example for a continuous distribution

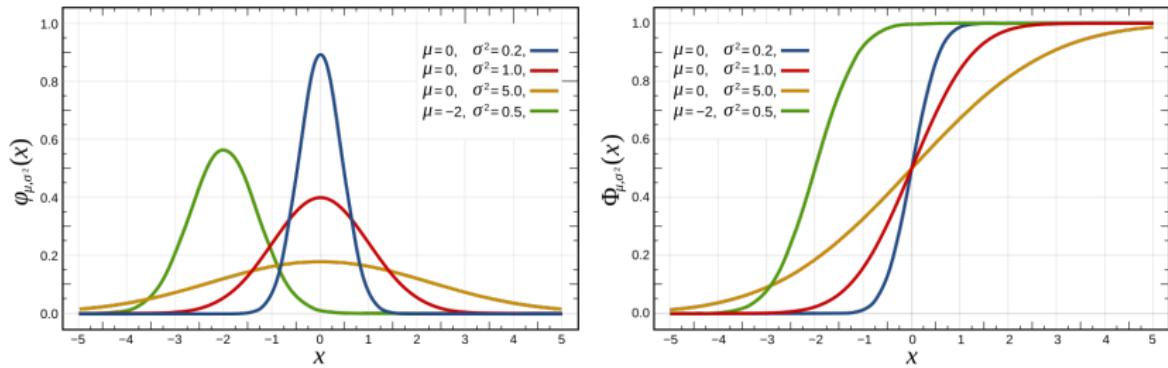


Figure: Density (left) and distribution function (right) for the Gaussian distribution for different values of the parameters.

Expectation of a function

The goal is to derive some characteristic numbers characterising a random variable or distribution.

Definition

Suppose X is a real random variable and $u : \mathcal{X} \rightarrow \mathbb{R}$ is a real function. The expectation of $u(X)$, denoted as $E(u(X))$ is defined as

$$E(u(X)) = \begin{cases} \int_{-\infty}^{\infty} p_X(x)u(x)dx & : X \text{ is a continuous random variable} \\ \sum_{x \in \mathcal{X}} p_X(x)u(x) & : X \text{ is a discrete random variable,} \end{cases}$$

provided the integral or sums are finite.

Expectations

Moments

- Expectation of X for $u(x) = x$:

$$E(X) = \begin{cases} \int_{-\infty}^{\infty} p_X(x)x dx & : X \text{ is a continuous random variable} \\ \sum_{x \in \mathcal{X}} p_X(x)x & : X \text{ is a discrete random variable.} \end{cases}$$

Sometimes we will write $\mu_X = E(X)$

- The m -th moment for $u(x) = x^m$, $m = 1, 2, 3, \dots$ and

$$E(X^m) = \begin{cases} \int_{-\infty}^{\infty} p_X(x)x^m dx & : X \text{ is a continuous random variable} \\ \sum_{x \in \mathcal{X}} p_X(x)x^m & : X \text{ is a discrete random variable.} \end{cases}$$

Expectations

An example

Example

Let $X \sim \text{bernoulli}(\theta)$. Then

$$E(X) = \theta \cdot 1 + (1 - \theta) \cdot 0 = \theta$$

$$E(X^2) = \theta \cdot 1^2 + (1 - \theta) \cdot 0^2 = \theta$$

Example

Let $X \sim \text{Exp}(\lambda)$. Then

$$\begin{aligned} E(X) &= \int_0^\infty x \lambda e^{-\lambda x} dx = -\lambda \int_0^\infty \frac{d}{d\lambda} e^{-\lambda x} dx = -\lambda \frac{d}{d\lambda} \left[\frac{-1}{\lambda} e^{-\lambda x} \right]_{x=0}^\infty \\ &= -\lambda \frac{d}{d\lambda} \frac{1}{\lambda} = \frac{1}{\lambda} \end{aligned}$$

Variance

and standard deviation

- Variance of X

$$\begin{aligned} \text{Var}(X) &= E((X - \mu_X)^2) \\ &= \begin{cases} \int_{-\infty}^{\infty} p_X(x)(x - \mu_X)^2 dx & : X \text{ is a continuous rv} \\ \sum_{x \in \mathcal{X}} p_X(x)(x - \mu_X)^2 & : X \text{ is a discrete rv.} \end{cases} \end{aligned}$$

- Standard deviation

$$\sigma_X := \sqrt{\text{Var}(X)}$$

Important Example

Expectation and variance for the Gaussian distribution

The Gaussian distribution (normal distribution)

$$p_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \quad -\infty < x < \infty$$

has the parameters $\sigma > 0$ and $\mu \in \mathbb{R}$. One can show

- $E(X) = \mu$ and the parameter μ is the expectation (mean).
- $\text{Var}(X) = \sigma^2$ and hence the parameter σ is the standard deviation.

Poisson distribution I

A model for counts

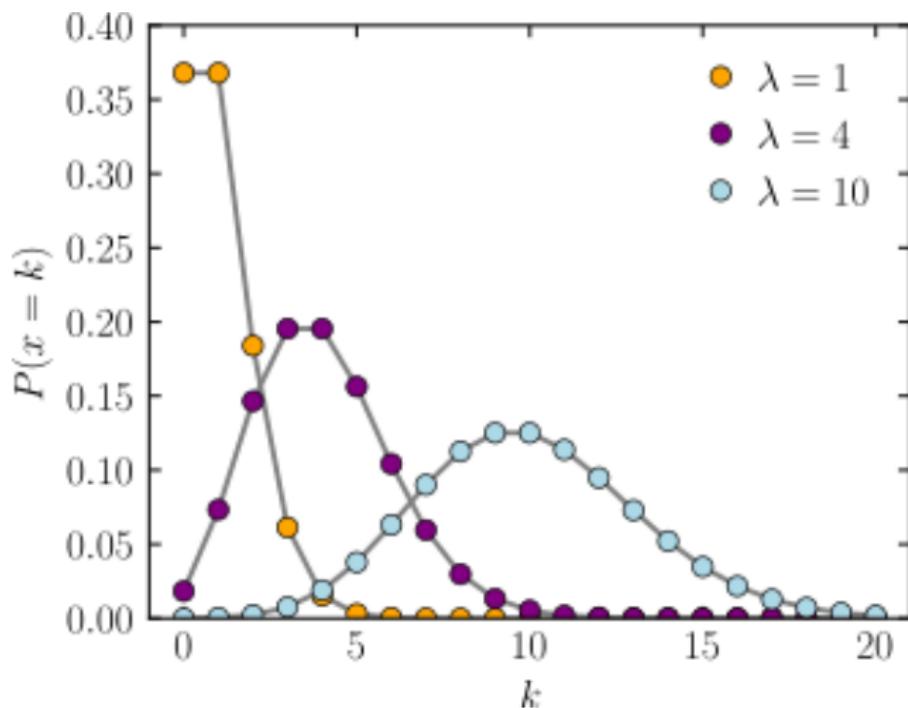
- Consider a fixed time interval.
- Assume that discrete events occur in this time interval.
- Assume further, that the expected number of events $\lambda > 0$ (the rate) is constant throughout the interval. Then, the probability distribution for the number of events $X \in \mathbb{N}$ is given by the **Poisson distribution**

$$P(X = k) = p(k) = \frac{\lambda^k}{k!} e^{-\lambda}$$

- Expectation: $E(X) = \lambda$
- Variance: $Var(X) = \lambda$

Poisson distribution II

A model for counts



Poisson distribution III

A model for counts

Figure: The probability mass function (pmf) for the Poisson distribution with different rate parameters λ .

Multiple random variables

Cumulative distribution function for two random variables

We combine two random variables X_1 and X_2 with state space \mathcal{X}_1 and \mathcal{X}_2 in a random vector $(X_1, X_2)^T \in \mathcal{X}_1 \times \mathcal{X}_2$ and consider the joint c.d.f.

$$F(x_1, x_2) = P(X_1 \leq x_1, X_2 \leq x_2).$$

Definition

If X_1, X_2 are both continuous, the joint probability density function (p.d.f) $p_{X_1 X_2}(x_1, x_2)$ is defined via

$$F(x_1, x_2) = \int_{-\infty}^{x_2} \int_{-\infty}^{x_1} p_{X_1 X_2}(\xi_1, \xi_2) d\xi_1 d\xi_2.$$

Then, for $B \subset \mathcal{X}_1 \times \mathcal{X}_2$

$$\Pr((X_1, X_2) \in B) = \int \int_B p_{X_1 X_2}(x_1, x_2) dx_1 dx_2.$$

Marginalization

Integrating the joint density over one of the variables yields the **marginal p.d.f.** of the other, i.e.

$$p_{X_2}(x_2) = \int_{\mathcal{X}_1} p_{X_1 X_2}(x_1, x_2) dx_1$$

$$p_{X_1}(x_1) = \int_{\mathcal{X}_2} p_{X_1 X_2}(x_1, x_2) dx_2.$$

The density function is normalised to one

$$\int_{\mathcal{X}_1} \int_{\mathcal{X}_2} p_{X_1 X_2}(x_1, x_2) dx_1 dx_2 = 1.$$

Joint probabilities

Discrete random variables

- One can do the same for discrete random variables by replacing the joint p.d.f. by the joint p.m.f. $p_{X_1 X_2}(x_1, x_2) = \Pr(X_1, X_2)$ and the integrals by sums.
- We will then just talk about the density, which can either be a p.d.f. in the case of continuous random variables or a which can be a p.m.f. in case of discrete random variables. In the latter case we also use the term discrete density instead of p.m.f.

Example

Two independent Gaussians

Example

Assume the joint density

$$p_{X_1 X_2}(x_1, x_2) = \frac{1}{2\pi\sigma_1\sigma_2} e^{-\frac{(x_1-\mu_1)^2}{2\sigma_1^2} - \frac{(x_2-\mu_2)^2}{2\sigma_2^2}}$$

with parameters $\mu_i, \sigma_i, i = 1, 2$ and $\sigma_i > 0$. Then

$$p_{X_1}(x_1) = \int_{-\infty}^{\infty} p_{X_1 X_2}(x_1, x_2) dx_2 = \frac{1}{\sigma_1 \sqrt{2\pi}} e^{-\frac{(x_1-\mu_1)^2}{2\sigma_1^2}}$$

and thus the $X_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$. Analogously one can see that $X_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$.

Conditional distributions

Definition

Let the random variables X_1 and X_2 have the joint density $p_{X_1 X_2}(x_1, x_2)$ and marginal densities $p_{X_1}(x_1)$ and $p_{X_2}(x_2)$ respectively. The conditional density of X_2 given that X_1 takes the value $X_1 = x_1$ is defined as

$$p_{X_2|X_1}(x_2 | x_1) = \frac{p_{X_1 X_2}(x_1, x_2)}{p_{X_1}(x_1)}, \quad p_{X_1}(x_1) > 0.$$

Remark: Often one simply writes $p_{X_2|X_1}(x_2 | x_1) = p(x_2 | x_1)$ or $p_{X_1 X_2}(x_1, x_2) = p(x_1, x_2)$ and $p_{X_1}(x_1) = p(x_1)$.

Independence of random variables

Definition

If the joint density $p(x_1, x_2)$ of the two random variables X_1 and X_2 can be written as the product of the marginal densities

$$p(x_1, x_2) = p(x_1)p(x_2),$$

then we call the two random variables X_1 and X_2 *independent*.

Corollary

If the random variables X_1 and X_2 are independent, then
 $p(x_2 | x_1) = p(x_2)$ and $p(x_1 | x_2) = p(x_1)$.

Example

The two Gaussians in the example on slide 265 are independent.

Another example

Conditional probabilities

Example

Assume that there are two dies, one fair die and one unfair die with p.m.f

1	2	3	4	5	6
1/4	1/4	1/4	1/12	1/12	1/12

Let's play the following game: You toss a coin and if it lands heads you take the fair die and if it lands tails, you take the loaded one. There are two random variables: $X \in \{0, 1\}$ is the outcome of the coin toss and $Y \in \{1, 2, 3, 4, 5, 6\}$ is the outcome of rolling the die. Then we have

$$p_{Y|X}(Y|0) = 1/6, \quad \forall Y \in \{1, 2, 3, 4, 5, 6\}$$

$$p_{Y|X}(Y|1) = \begin{cases} 1/4 & : Y \in \{1, 2, 3\} \\ 1/12 & : Y \in \{4, 5, 6\} \end{cases}$$

Obviously, X and Y are dependent.

Probabilistic logic

Events and propositions I

Events can be described by propositions. The proposition is true, if the event happens, otherwise it is false.

Example

- Sample space:
 $\Omega : \{\omega_1 = \text{"Germany wins the football world cup."}, \omega_2 = \text{"Germany doesn't win the world cup."}\}$
- Event: $A = \omega_1$
- Proposition: $q_A = \text{"Germany wins the football world cup."}$

The probability of the event equals the probability of the proposition to be true

$$P(A) = P(q_A = \text{True}).$$

Remark: More exactly, we should denote the truth value of q_A as $\mathcal{I}(q_A)$ and write $P(\mathcal{I}(q_A) = T)$, but we use the shorthand notation above.

Events and propositions II

Let A, B be events and Ω be the sample space. The propositions corresponding to the events are denoted as q_A and q_B , respectively.

Set notation	propositional logic	Description
$A \cap B$	$q_A \wedge q_B$	intersection/and
$A \cup B$	$q_A \vee q_B$	union/or
A^c	$\neg q_A$	complement/negation
Ω	True	certain event/true
\emptyset	False	impossible event/false

Probability rules for logic propositions I

From probability theory and the link between logical propositions and events we find for the events q_A and q_B

$$P(\neg q_A) = 1 - P(q_A) \tag{15}$$

$$P(q_A \vee \neg q_A) = P(q_A \vee q_{A^c}) = 1 \quad \text{certain event} \tag{16}$$

$$P(q_A \wedge \neg q_A) = P(q_A \wedge q_{A^c}) = 0 \quad \text{impossible event} \tag{17}$$

$$P(q_A \vee q_B) = P(q_A) + P(q_B) - P(q_A \wedge q_B) \tag{18}$$

Important Remark

Information about the truth values of q_A and q_B is not sufficient to say anything about the probability of $q_A \wedge q_B$!

Probability rules for logic propositions II

Theorem

To fully characterize all possible probabilities for a set of logical propositions q_1, \dots, q_N one needs to specify the joint probability distribution.

Example

Let q_A and q_B be two logical propositions. To characterize their joint distribution we need to characterize the four numbers

$$p(q_A \wedge q_B), p(\neg q_A \wedge \neg q_B), p(\neg q_A \wedge q_B), p(q_A \wedge \neg q_B).$$

This shows that we need 2^N different numbers, if we want to obtain the joint probability for the propositions q_1, \dots, q_N .

Conditional probabilities

The probability of a proposition q_A to be true given that we know that q_B is true is determined by the conditional probability

$$P(q_A|q_B) = \frac{P(q_A \wedge q_B)}{P(q_B)}.$$

A-priori and a-posteriori probabilities

A-priori probabilities

For a proposition q we can assign a probability $P(q)$ independently of the outcome of an event or a concrete experiment. This can reflect our subjective belief or can be based on prior experience.

A-posteriori probabilities

Assume that we observe that the propositions r_1, \dots, r_N are true. These observations change the probability that q is true or not. The conditional probability $p(q|r_1, \dots, r_N)$ is then called an a-posteriori probability.

Sometimes we write $Bel(q)$ for the a-posterior-probability. For the observed true propositions we have $Bel(p_i) = 1$ for $i = 1, \dots, N$

Probabilistic Logic I

Modus Ponens

Recap: Modus Ponens in propositional logic:

For two proposition q and r we have

$$\frac{q \Rightarrow r}{\frac{q}{r}}$$

Now, we only have probabilities for the truth of the propositions.

Theorem (Modus Ponens of probabilistic logic)

$$\frac{\begin{array}{l} P(q|r) = \alpha \\ Bel(r) = 1 \end{array}}{Bel(q) = \alpha}$$

Probabilistic Logic II

Modus Ponens

This can be generalized: If it is known that the propositions r_1, \dots, r_N are true, then q is true with probability $P(q|r_1, \dots, r_N)$. Thus, the conditioning describes the **context**, from which we can reason about a proposition.

Example

We are at the dentist and three proposition Toothache, Cavity, and Catch (the dentist's nasty steel probe catches in my tooth).

Joint probability distribution:

	toothache		\neg toothache	
	catch	\neg catch	catch	\neg catch
cavity	0.108	0.012	0.072	0.008
\neg cavity	0.016	0.064	0.144	0.576

Probabilistic Logic III

Modus Ponens

Example

Now we compute the probability of toothache.

$$\begin{aligned} P(\text{toothache}) &= P(\text{toothache} \wedge \text{catch} \wedge \text{cavity}) \\ &\quad + P(\text{toothache} \wedge \neg \text{catch} \wedge \text{cavity}) \\ &\quad + P(\text{toothache} \wedge \text{catch} \wedge \neg \text{cavity}) \\ &\quad + P(\text{toothache} \wedge \neg \text{catch} \wedge \neg \text{cavity}) \\ &= 0.108 + 0.012 + 0.016 + 0.064 = 0.2 \end{aligned}$$

Probabilistic Logic IV

Modus Ponens

Example

Given that we know that we have a cavity and the dentist uses its steel probe, we want to know the probability of the incredible pain we might have to suffer:

$$P(\text{toothache} | \text{catch} \wedge \text{cavity}) = \frac{P(\text{toothache} \wedge \text{catch} \wedge \text{cavity})}{P(\text{catch} \wedge \text{cavity})}$$

$$\begin{aligned} P(\text{catch} \wedge \text{cavity}) &= P(\text{toothache} \wedge \text{catch} \wedge \text{cavity}) \\ &\quad + P(\neg\text{toothache} \wedge \text{catch} \wedge \text{cavity}) \\ &= 0.108 + 0.072 = 0.18 \end{aligned}$$

$$P(\text{toothache} | \text{catch} \wedge \text{cavity}) = \frac{0.108}{0.18} = 0.6$$

Lack of Modularity in Probabilistic Logic

- In propositional and predicate logic we can deduce from the axioms

$$q$$

$$q \Rightarrow r$$

$$r \Rightarrow s$$

in a *stepwise* manner that s is true.

- However, $P(q, r, s) = P(s|q, r)p(q, r)$ requires knowledge of $P(s|q, r)$ and $P(q, r)$. Knowledge of $P(r)$ and $P(q)$ alone is not sufficient.
- Probabilistic logic is not modular and stepwise reasoning is not generally possible.

Bayesian Inference I

- **Problem:** Assume, that we have a model for $P(q|r)$ and we observe q to be true. What is the a-posteriori probability $P(r|q)$?
- Bayes Formula:

$$P(r|q) = \frac{P(q|r)P(r)}{P(q)}$$

To solve the problem above we need to know the a-priori probabilities $P(r)$ and $P(q)$.

Bayesian Inference II

Application to diagnostic questions:

- Let s is the proposition for a symptom and f the proposition for a failure or an error.
- Then, the belief that the error occurs, given that the symptom is present is given as

$$P(f|s) = \frac{P(s|f)P(f)}{P(s)} = \frac{P(s|f)P(f)}{P(s|f)P(f) + P(s|\neg f)P(\neg f)}.$$

The last equation follows from the law of total probability.

Chap. 6 Bayesian Networks

Motivation

- The full joint probability distribution can answer any question about the domain.
- But, it can become intractably large as the number of variables grows.
- Bayesian Networks (Belief Networks): Structure the joint distribution.

Bayesian Networks

Introductory example

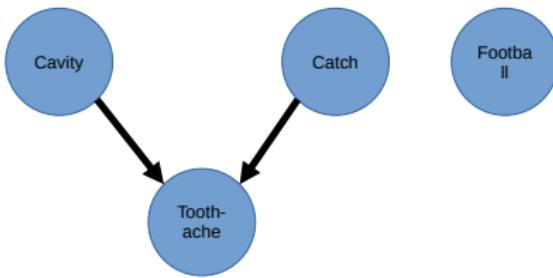


Figure: A Bayesian network where *Toothache* depends on both *Catch* and *Cavity*. *Catch* and *Cavity* are independent. *Football* (for Football results) is independent of all other variables.

$$P(\text{Cavity}, \text{Catch}, \text{Toothache}, \text{Football})$$

$$= P(\text{Football})P(\text{Cavity})P(\text{Catch})P(\text{Toothache}|\text{Cavity}, \text{Catch})$$

Conditional Independence I

Definition (Conditional Independence)

Let X and Y and Z be random variables. We say that X and Y are conditionally independent, if

$$P(X, Y|Z) = P(X|Z) P(Y|Z).$$

Notation: $X \perp Y|Z$ or sometimes $X \perp\!\!\!\perp Y|Z$

Remark

Here, P stands for a probability mass function in the case of discrete random variables and for a probability density in the case of continuous variables. There are also mixed probability measures for combinations of discrete and continuous random variables.

Conditional Independence II

Theorem

If $X \perp\!\!\!\perp Y|Z$ then the following equations are equivalent

$$P(X, Y|Z) = P(X|Z) P(Y|Z)$$

$$P(X|Y, Z) = P(X|Z)$$

$$P(Y|X, Z) = P(Y|Z)$$

Proof.

From the definition of independences and the definition of conditional probabilities (Exercise). □

Directed Acyclic Graphs (DAGs) I

A directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of

- a set of nodes (vertices) \mathcal{V} and
- an edge set \mathcal{E} of ordered pairs of vertices.
- If $(X, Y) \in \mathcal{E}$ then there is an arrow pointing from node X to node Y .

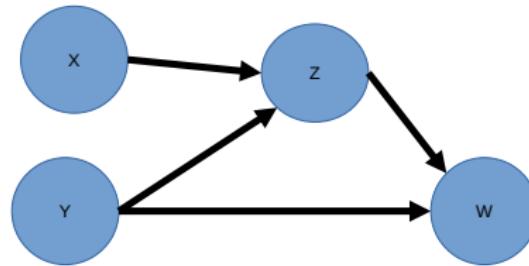


Figure: A directed graph with nodes $\mathcal{V} = \{X, Y, Z, W\}$ and edges $\mathcal{E} = \{(X, Z), (Z, W), (Y, Z), (Y, W)\}$

Directed Acyclic Graphs (DAGs) II

- Two variables X and Y are **adjacent**, if they are connected by an edge, regardless of direction.

Directed Acyclic Graphs (DAGs) II

- Two variables X and Y are **adjacent**, if they are connected by an edge, regardless of direction.
- If $X \rightarrow Y$ we call X a **parent** of Y and Y a **child** of X .

Directed Acyclic Graphs (DAGs) II

- Two variables X and Y are **adjacent**, if they are connected by an edge, regardless of direction.
- If $X \rightarrow Y$ we call X a **parent** of Y and Y a **child** of X .
- All parents of a node X are denoted as $\text{par}(X)$.

Directed Acyclic Graphs (DAGs) II

- Two variables X and Y are **adjacent**, if they are connected by an edge, regardless of direction.
- If $X \rightarrow Y$ we call X a **parent** of Y and Y a **child** of X .
- All parents of a node X are denoted as $\text{par}(X)$.
- A **directed path** between two variables is a set of arrows all pointing in the same direction linking one variable to the other.

Directed Acyclic Graphs (DAGs) II

- Two variables X and Y are **adjacent**, if they are connected by an edge, regardless of direction.
- If $X \rightarrow Y$ we call X a **parent** of Y and Y a **child** of X .
- All parents of a node X are denoted as $\text{par}(X)$.
- A **directed path** between two variables is a set of arrows all pointing in the same direction linking one variable to the other.
- A sequence of adjacent vertices starting with X and ending with Y but ignoring the direction of the arrows is called an **undirected path** or just a path.

Directed Acyclic Graphs (DAGs) II

- Two variables X and Y are **adjacent**, if they are connected by an edge, regardless of direction.
- If $X \rightarrow Y$ we call X a **parent** of Y and Y a **child** of X .
- All parents of a node X are denoted as $\text{par}(X)$.
- A **directed path** between two variables is a set of arrows all pointing in the same direction linking one variable to the other.
- A sequence of adjacent vertices starting with X and ending with Y but ignoring the direction of the arrows is called an **undirected path** or just a path.
- If there is a directed path from node X to node Y we call X an **ancestor** of X and Y a **descendant** of X .

Directed Acyclic Graphs (DAGs) III

- A directed path that starts and ends at the same node is called a **cycle**.

Directed Acyclic Graphs (DAGs) III

- A directed path that starts and ends at the same node is called a **cycle**.
- A directed graph without cycles is called a **directed acyclic graph**

Probability and Graphs

- Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes X_1, \dots, X_K .

Probability and Graphs

- Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes X_1, \dots, X_K .
- Assume that the nodes correspond to random variables (discrete or continuous or mixed).

Probability and Graphs

- Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes X_1, \dots, X_K .
- Assume that the nodes correspond to random variables (discrete or continuous or mixed).
- Let $p(x_1, \dots, x_K)$ denote the joint probability mass function (pmf) or the joint probability density function (pdf) of X_1, \dots, X_K .

Probability and Graphs

- Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes X_1, \dots, X_K .
- Assume that the nodes correspond to random variables (discrete or continuous or mixed).
- Let $p(x_1, \dots, x_K)$ denote the joint probability mass function (pmf) or the joint probability density function (pdf) of X_1, \dots, X_K .

Definition

We say that p is **Markov to \mathcal{G}** , or that \mathcal{G} **represents P** , if

$$p(x_1, \dots, x_K) = \prod_{k=1}^K p(x_k | \text{par}(x_k))$$

The set of distributions represented by \mathcal{G} is denoted by $\mathcal{M}(\mathcal{G})$.

Probability and Graphs

Example 1

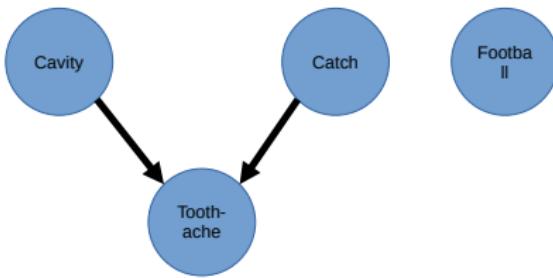


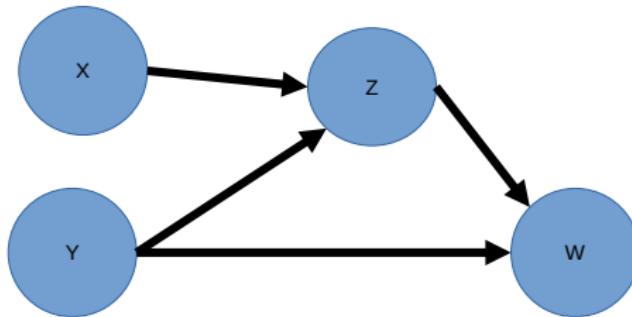
Figure: A Bayesian network where *Toothache* depends on both *Catch* and *Cavity*. *Catch* and *Cavity* are independent. *Football* (for Football results) is independent of all other variables.

$$P(\text{Cavity}, \text{Catch}, \text{Toothache}, \text{Football})$$

$$= P(\text{Football})P(\text{Cavity})P(\text{Catch})P(\text{Toothache}|\text{Cavity}, \text{Catch})$$

Probability and Graphs

Example 2



For each $p \in \mathcal{M}(\mathcal{G})$ we have

$$p(x, y, z, w) = p(x)p(y)p(z|x, y)p(w|y, z).$$

Bayesian Networks

Definition

Definition

A **Bayesian Network** is a directed acyclic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ representing a joint probability distribution $p \in \mathcal{M}(\mathcal{G})$ over the random variables $\mathcal{V} = (X_1, \dots, X_K)$. Each node X_i has a conditional probability distribution $p(X_i | \text{par}(X_i); \theta_i)$ with a finite number of parameters θ_i .

Representing Knowledge with A Bayesian Network I

Example: Alarm Network

You have a new burglar alarm installed at home. It is fairly reliable at detecting a burglary, but is occasionally set off by minor earthquakes. You also have two neighbors, John and Mary, who have promised to call you at work when they hear the alarm. John nearly always calls when he hears the alarm, but sometimes confuses the telephone ringing with the alarm and calls then, too. Mary, on the other hand, likes rather loud music and often misses the alarm altogether. Given the evidence of who has or has not called, we would like to estimate the probability of a burglary.

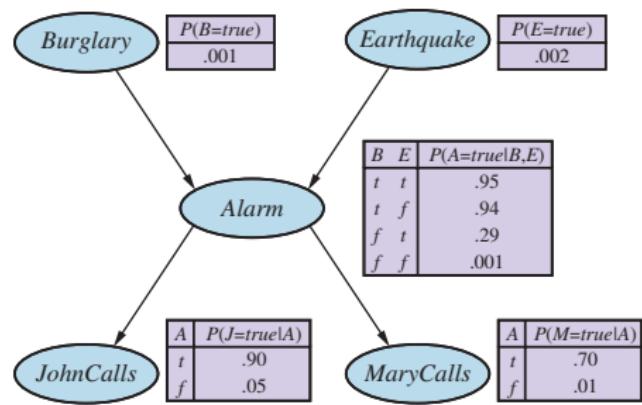


Figure: The alarm network with the respective conditional probability distribution assigned to each node.

Representing Knowledge with A Bayesian Network II

Example: Alarm Network

- To each variable (node) there is assigned conditional probability distribution $p(X_i|par(X_i); \theta_i)$.
- Since all variables are discrete, these are just tables (conditional probability tables).
- The parameters θ are given by the conditional probabilities in the tables.

Conditional Independence relations in Bayesian Networks

Markov condition I

Theorem

A distribution $p \in \mathcal{M}(\mathcal{G})$ if and only if the following **Markov Condition** holds: For every variable X ,

$$X \perp\!\!\!\perp \tilde{X} | \text{par}(X)$$

where \tilde{X} denotes all the other variables except the parents and the descendants of X .

Conditional Independence relations in Bayesian Networks

Markov condition II

Markov-condition:

$$X \perp\!\!\!\perp \tilde{X} | \text{par}(X)$$

where \tilde{X} denotes all the other variables except the parents and the descendants of X .

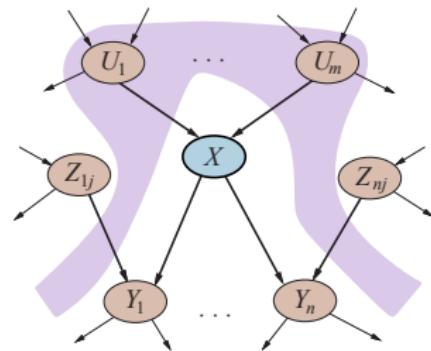
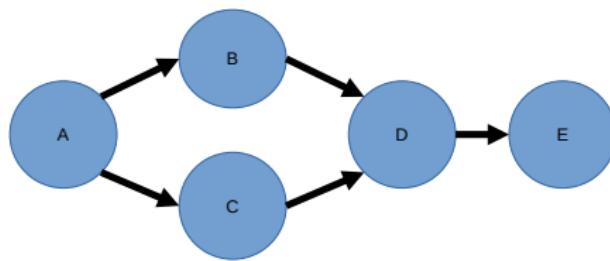


Figure: A node X is conditionally independent of its non-descendants (e.g. the Z_{ij} s) given its parents (the U_i s shown in the lavender area). (From Russel and Norvig, AI, Fig. 13.4)

There are, however, further conditional independence relations in a Bayesian Network implied by the Markov property.

Markov condition

Example



Joint probability function:

$$p(a, b, c, d, e) = p(a)p(b|a)p(c|a)p(d|b, c)p(e|d)$$

The following independence relations follow from the Markov Condition:

$$D \perp\!\!\!\perp A | \{B, C\}, \quad B \perp\!\!\!\perp C | A, \quad E \perp\!\!\!\perp A, B, C | D$$

Conditional Independence relations in Bayesian Networks

Markov blankets I

Markov blanket: Consider a node X in a Bayesian network. The set of nodes formed by its

- parents
- children
- the childrens parents

is called the Markov blanket of X .

Notation: $MB(X)$

Theorem

A variable is conditionally independent of all other nodes in the network, given its Markov blanket:

$$X \perp\!\!\!\perp Z | MB(X)$$

for all nodes Z with $Z \cap (MB(Z) \cup X) = \emptyset$.

Conditional Independence relations in Bayesian Networks

Markov blankets II

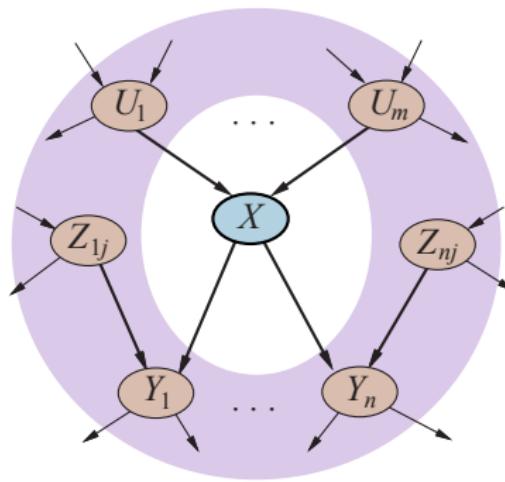


Figure: The lavender area indicates the Markov blanket $MB(X)$ of the node X . Given $MB(X)$, the node X is independent of all other nodes in the network. (From Russel and Norvig, AI, Fig. 13.4).

Conditional Independence relations in Bayesian Networks

D-separation I

The most general coinditional independence question we can ask is the following:

- Let $X, Y, Z \subset \mathcal{E}$ be sets of nodes in a Bayesian Network.
Are X and Z independent, given Y ?

Conditional Independence relations in Bayesian Networks

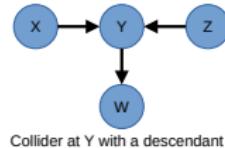
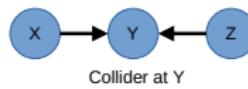
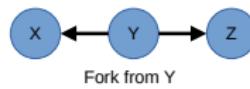
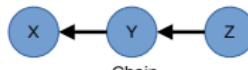
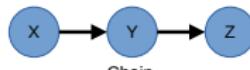
D-separation I

The most general coinditional independence question we can ask is the following:

- Let $X, Y, Z \subset \mathcal{E}$ be sets of nodes in a Bayesian Network.
Are X and Z independent, given Y ?
- This answer can be answered by the d-separation property (directed separation).

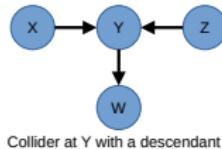
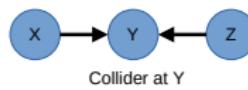
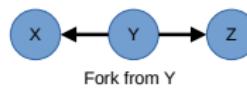
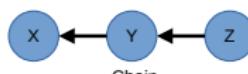
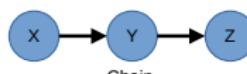
Conditional Independence relations in Bayesian Networks

D-separation II



Conditional Independence relations in Bayesian Networks

D-separation II

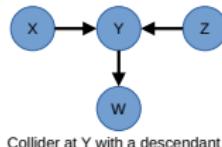
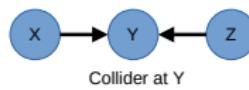
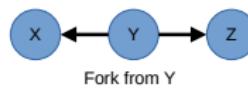
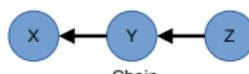
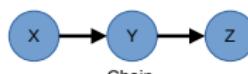


Rules of d-separation:

- When Y is not a collider, X and Z are **d-connected**, but they are **d-separated** given Y .

Conditional Independence relations in Bayesian Networks

D-separation II

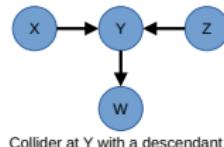
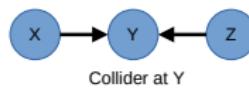
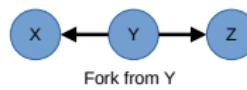
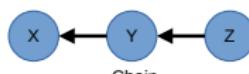


Rules of d-separation:

- ① When Y is not a collider, X and Z are **d-connected**, but they are **d-separated** given Y .
- ② If X and Z collide at Y , then X and Z are **d-separated**, but they are **d-connected** given Y .

Conditional Independence relations in Bayesian Networks

D-separation II



Rules of d-separation:

- ① When Y is not a collider, X and Z are **d-connected**, but they are **d-separated** given Y .
- ② If X and Z collide at Y , then X and Z are **d-separated**, but they are **d-connected** given Y .
- ③ Conditioning on the descendant of a collider has the same effect as conditioning on the collider.

Conditional Independence relations in Bayesian Networks

D-separation III

Definition (d-separation)

Let X and Y be distinct nodes and let W be a set of nodes not containing X or Y .

Then X and Y are **d-separated given W** if there exists no undirected path π between X and Y such that

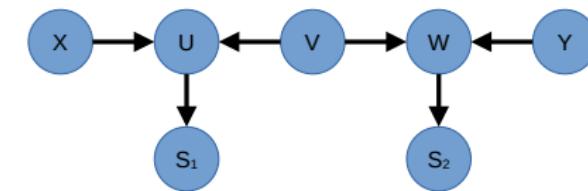
- ① every collider on π has a descendant in W and
- ② no other node on π is in W .

If A , B and W are distinct sets of nodes and A and B are not empty, then A and B are **d-separated given W** if for every $X \in A$ and $Y \in B$, X and Y are d-separated given W . Sets of nodes that are not d-separated are said to be **d-connected**.

Conditional Independence relations in Bayesian Networks

D-separation IV

Example (d-separation)

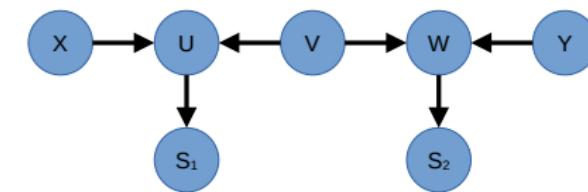


- X and Y are d-separated (given the empty set)

Conditional Independence relations in Bayesian Networks

D-separation IV

Example (d-separation)

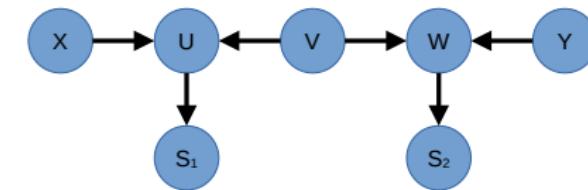


- X and Y are d-separated (given the empty set)
 - ▶ The path between X and Y is blocked by the colliders at U and W .

Conditional Independence relations in Bayesian Networks

D-separation IV

Example (d-separation)

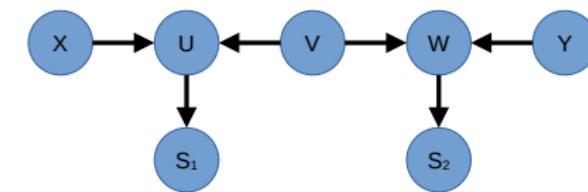


- X and Y are d-separated (given the empty set)
 - ▶ The path between X and Y is blocked by the colliders at U and W .
- X and Y are d-connected given $\{S_1, S_2\}$

Conditional Independence relations in Bayesian Networks

D-separation IV

Example (d-separation)

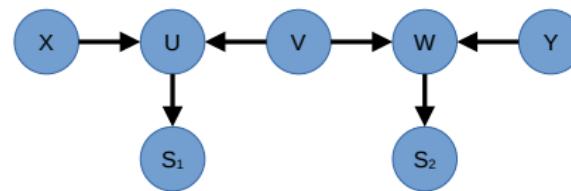


- X and Y are d-separated (given the empty set)
 - ▶ The path between X and Y is blocked by the colliders at U and W .
- X and Y are d-connected given $\{S_1, S_2\}$
 - ▶ The path between X and Y has colliders at U and W . Conditioning on their descendants S_1 and S_2 opens the path.

Conditional Independence relations in Bayesian Networks

D-separation IV

Example (d-separation)

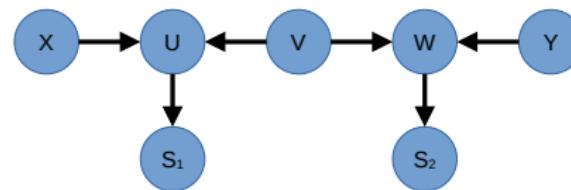


- X and Y are d-separated (given the empty set)
 - ▶ The path between X and Y is blocked by the colliders at U and W .
- X and Y are d-connected given $\{S_1, S_2\}$
 - ▶ The path between X and Y has colliders at U and W . Conditioning on their descendants S_1 and S_2 opens the path.
- X and Y are d-separated given $\{S_1, S_2, V\}$

Conditional Independence relations in Bayesian Networks

D-separation IV

Example (d-separation)



- X and Y are d-separated (given the empty set)
 - ▶ The path between X and Y is blocked by the colliders at U and W .
- X and Y are d-connected given $\{S_1, S_2\}$
 - ▶ The path between X and Y has colliders at U and W . Conditioning on their descendants S_1 and S_2 opens the path.
- X and Y are d-separated given $\{S_1, S_2, V\}$
 - ▶ The node V on the path between X and Y is not a collider. Conditioning on V blocks the path.

Conditional Independence relations in Bayesian Networks

D-separation V

Theorem

Let X , Y and Z disjoint sets of nodes in a Bayesian network. Then,
 $X \perp\!\!\!\perp Y | Z$ if and only if X and Y are d-separated by Z .

Conditional Independence relations in Bayesian Networks

D-separation V

Theorem

Let X , Y and Z disjoint sets of nodes in a Bayesian network. Then,
 $X \perp\!\!\!\perp Y | Z$ if and only if X and Y are d-separated by Z .

Remark: We implicitly assume that the joint probability distribution is faithful to the graph \mathcal{G} . This means. that there are no additional independence relations implied by the probability distribution (for example by special parameter values inducing additional independence relations).

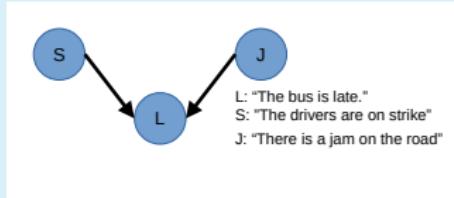
Conditioning on colliders

Why does conditioning on a collider induce statistical dependence?

Example

There are two different reasons, why the bus is late ($L = 1$):

- ① There is a congestion (jam, $J = 1$)
- ② The drivers are on strike ($S = 1$)



- $S \perp\!\!\!\perp Y$, because the path between S and L is blocked by the collider at L
- If the bus is late, the probability of a jam should increase:
 $P(J = 1|L = 1) \geq P(J = 1)$.
- But when we learn that there is a strike, we would lower the chance that there is a jam: $P(J = 1|L = 1) \neq P(J = 1|L = 1, S = 1)$
- This makes the effect, that conditioning on a collider renders conditional dependence (d-dependence) plausible.

Markov equivalence I

Different graphs with the same independence relations

- If \mathcal{G} is a DAG, let $I(\mathcal{G})$ denote all the independence statements implied by \mathcal{G} .
- Two DAGs \mathcal{G}_1 and \mathcal{G}_2 for the same random variables (nodes) are called **Markov-equivalent**, if $I(\mathcal{G}_1) = I(\mathcal{G}_2)$.
- A collider is called **unshielded**, if the parents are not adjacent.
- The **skeleton** of a DAG \mathcal{G} is the undirected graph obtained by replacing the arrows with undirected edges.

Theorem

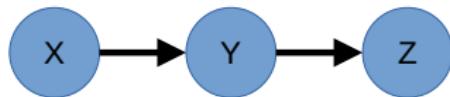
Two DAGs \mathcal{G}_1 and \mathcal{G}_2 for the same random variables (nodes) are **Markov-equivalent**, if

- ① They have the same skeleton and
- ② \mathcal{G}_1 and \mathcal{G}_2 have the same unshielded colliders.

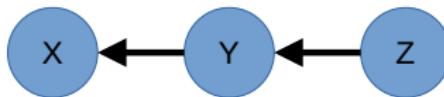
Markov equivalence II

Different graphs with the same independence relations

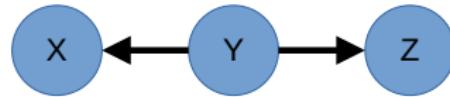
Example



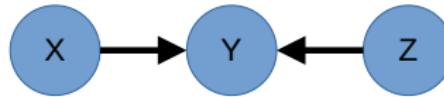
Chain



Chain



Fork from Y



Collider at Y

- The chains and the forks are Markov-equivalent.
- The DAG with the collider is not Markov-equivalent to the three others.

Constructing Bayesian Networks I

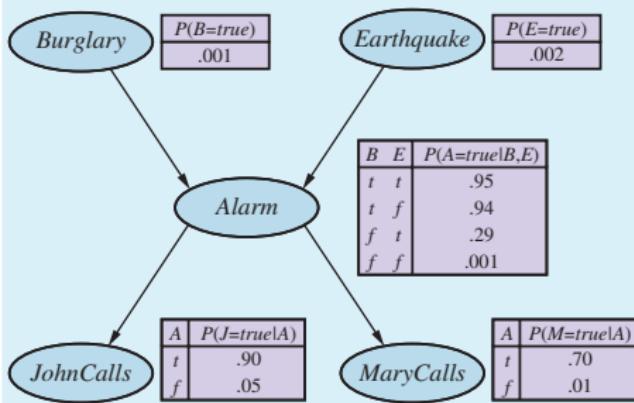
Aim: Construct a Bayesian network in such a way that the resulting joint distribution is a good representation of a given domain.

Some thoughts:

- The Markov condition implies that we only need to specify $P(X_i|par(X_i))$ to describe the joint distribution.
- **Topological order:** Numbering the nodes in topological order means that the order is consistent with the directed graph structure.

Constructing Bayesian Networks II

Example



Two orderings in Topological order:

X_1	X_2	X_3	X_4	X_5
B	E	A	J	M
E	B	A	M	J
:	...			:

Constructing Bayesian Networks III

Constructing a Bayesian Network

- ① **Nodes:** First determine the set of variables that are required to model the domain. Now order them, X_1, \dots, X_K . Any order will work, but the resulting network will be more compact if the variables are ordered such that causes precede effects.
- ② **Edges:** For $i = 1$ to n do:
 - ▶ Choose a minimal set of parents for X_i from X_1, \dots, X_{i-1} , such that

$$P(X_i | X_{i-1}, \dots, X_1) = P(X_i | \text{parents}(X_i))$$

is satisfied.

- ▶ For each parent insert a link from the parent to X_i .
- ▶ Conditional probability tables: Write down the conditional probability table:

$$p(X_i | \text{par}(X_i)) \equiv p(X_i | \text{par}(X_i); \theta_i)$$

Ambiguities when constructing Bayesian Networks

- Assume, we want to model a system with several variables X_1, \dots, X_K and we know all the conditional independence relationships between these variables.

Ambiguities when constructing Bayesian Networks

- Assume, we want to model a system with several variables X_1, \dots, X_K and we know all the conditional independence relationships between these variables.
- Then, there can still be several Markov-equivalent DAGs which can not be distinguished from each other by just knowing the independence relations. All these DAGs have the same skeleton and unshielded colliders.

Ambiguities when constructing Bayesian Networks

- Assume, we want to model a system with several variables X_1, \dots, X_K and we know all the conditional independence relationships between these variables.
- Then, there can still be several Markov-equivalent DAGs which can not be distinguished from each other by just knowing the independence relations. All these DAGs have the same skeleton and unshielded colliders.
- Domain knowledge or designed experiments about the **causal** relationships can be used to direct the edges.

Example: Car insurance applications I

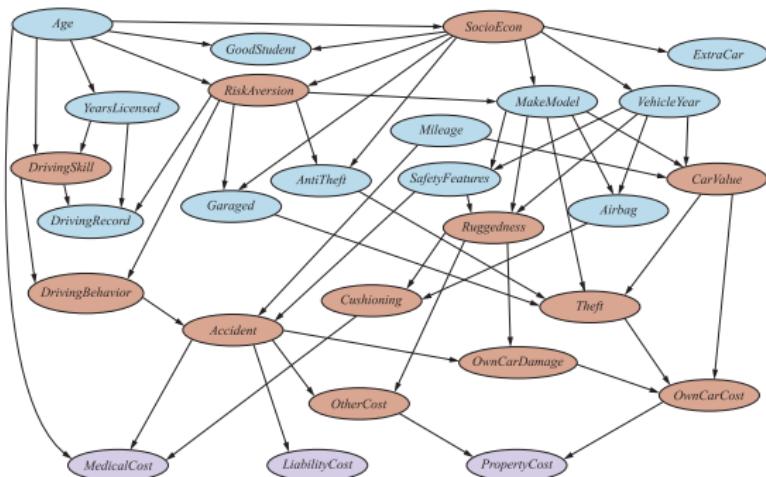


Figure: Car insurance applications represented by a Bayesian Network. (From Russel and Norvig, AI, Fig. 13.9)

Claims (shaded lavender):

- **MedicalCost:** for any injuries sustained by the applicant.
- **LiabilityCost:** for lawsuits filed by other parties against the applicant and the company.
- **PropertyCost:** for vehicle damage to either party and vehicle loss by theft.

Example: Car insurance applications II

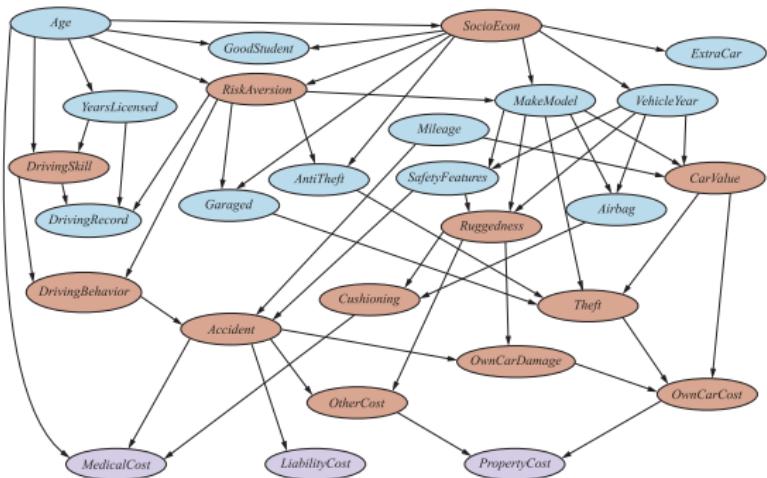


Figure: Car insurance applications represented by a Bayesian Network. (From Russel and Norvig, AI, Fig. 13.9)

Input to application form (light blue):

- Applicant: Age, Years licensed, GoodStudent, DrivingRecord.
- Vehicle: MakeModel, VehicleYear, Airbag, SafetyFeatures.
- Driving situation: Mileage, Garaged.

Example: Car insurance applications III

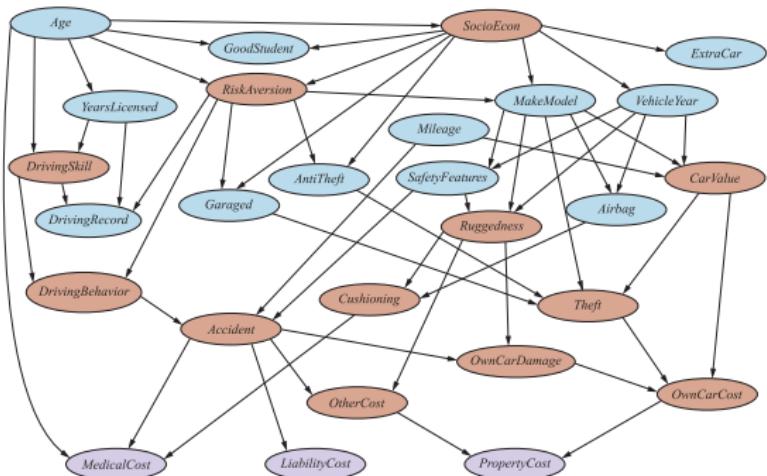


Figure: Car insurance applications represented by a Bayesian Network. (From Russel and Norvig, AI, Fig. 13.9)

Hidden Variables (red):

- Neither input nor output variables.
- Essential for structuring the network.
- Reasonable sparse connection structure and manageable parameters in the conditional probability tables.

Example: Car insurance applications IV

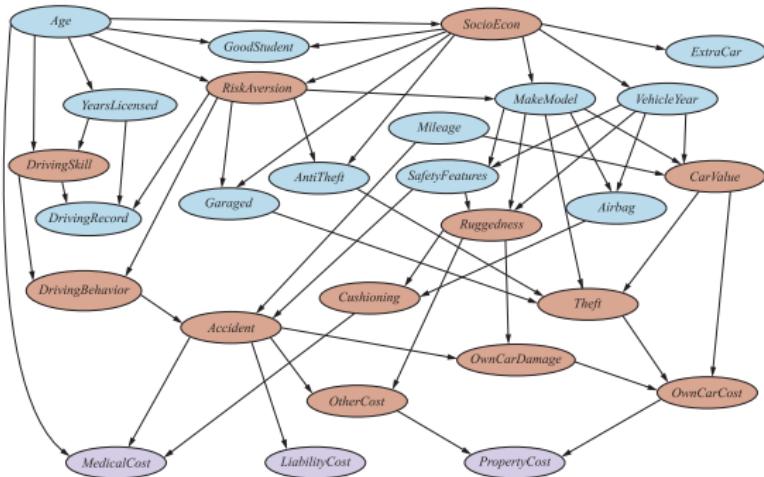


Figure: Car insurance applications represented by a Bayesian Network. (From Russel and Norvig, AI, Fig. 13.9)

- Theft or Accident are key hidden variables.
- It is unknown, whether these will occur in the next time period.
- They need to be inferred from the available information and previous experience.
- During network development, the parent variables are chosen to represent causal effects on key variables.

The inference problem I

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the graph of a Bayesian network representing a probability distribution P which Markov to \mathcal{G}

$$p(x_1, \dots, x_K) = \prod_{k=1}^K p(x_k | \text{par}(x_k)).$$

Typically, we have three disjoint sets \mathbf{Y} , \mathbf{Q} and \mathbf{Z} of nodes with

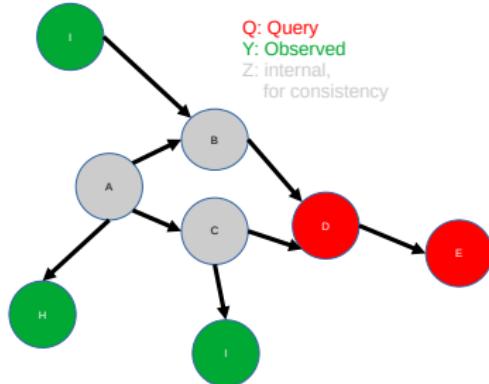
$$\mathcal{V} = \mathbf{Y} \cup \mathbf{Q} \cup \mathbf{Z}.$$

The inference problem II

Y : The set of measured or observed variables (symptoms, events that happened).

Q : Query variables, the variables we want to know.

Z : Internal variables, necessary for the inference task.



Inference task:

Compute the posterior distribution over the query variables \mathbf{Q} given the evidence $\mathbf{Y} = \mathbf{y}$

$$P(\mathbf{Q}|\mathbf{Y} = \mathbf{y}) = P(\mathbf{Q}|\mathbf{y}).$$

Inference from the joint probability distribution I

The Bayesian network describes the joint probability distribution

$$P(X_1, \dots, X_K) \equiv P(\mathbf{Q}, \mathbf{Y}, \mathbf{Z}) = P(Q_1, \dots, Q_{k_Q}, Y_1, \dots, Y_{k_Y}, Z_1, \dots, Z_{k_Z})$$

with $k_q + k_Y + k_Z = K$. For the inference we use

$$\begin{aligned} P(\mathbf{Q} | \mathbf{Y}) &= \frac{P(\mathbf{Q}, \mathbf{Y})}{P(\mathbf{Y})} = \frac{\sum_{\mathbf{z}} P(\mathbf{Q}, \mathbf{Y}, \mathbf{Z} = \mathbf{z})}{P(\mathbf{Y})} \\ &= \frac{1}{N} \sum_{\mathbf{z}} P(\mathbf{Q}, \mathbf{Y}, \mathbf{Z} = \mathbf{z}) \end{aligned} \tag{19}$$

Here, the normalization factor

$$N = P(\mathbf{Y}) = \sum_{\mathbf{q}} \sum_{\mathbf{z}} P(\mathbf{Q} = \mathbf{q}, \mathbf{Y}, \mathbf{Z} = \mathbf{z})$$

which describes the a-priory probability of the observations \mathbf{Y} .

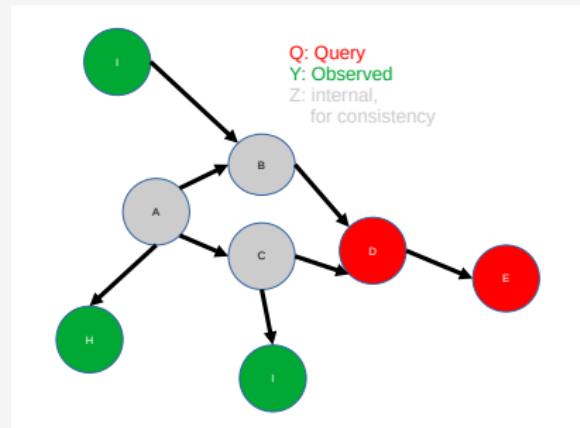
Inference from the joint probability distribution II

Inference in Bayesian Networks (Summary)

Y : The set of measured or observed variables (symptoms, events that happened).

Q : Query variables, the variables we want to know.

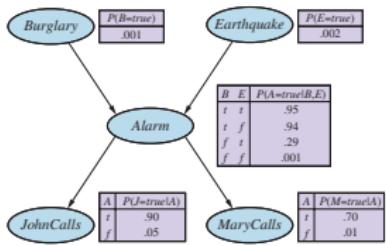
Z : Internal variables, necessary for the inference task.



$$P(Q|Y) = \frac{1}{N} \sum_z P(Q, Y, Z = z) \quad (20)$$

$$N = P(Y) = \sum_q \sum_z P(Q = q, Y, Z = z) \quad (21)$$

Example: Burglary network I



Query: $P(B|J = 1, M = 1)$

- $Q = B, \mathbf{Y} = (J, M) = \mathbf{y} = (1, 1), \mathbf{Z} = (E, A)$

$$P(B|J = 1, M = 1)$$

$$= \frac{1}{N} \sum_{e \in \{0,1\}} \sum_{a \in \{0,1\}} P(B, J = 1, M = 1, E = e, A = a)$$

Factorization implied by the Markov property:

$$P(B, J = 1, M = 1, E = e, A = a)$$

$$= P(B)P(E = e)P(A = a|B, E = e)P(J = 1|A = a)P(M = 1|A = a)$$

Example: Burglary network II

Thus

$$\begin{aligned} P(B|J = 1, M = 1) &= \frac{1}{N} \sum_{e \in \{0,1\}} \sum_{a \in \{0,1\}} P(B)P(E = e)P(A = a|B, E = e)P(J = 1|A = a)P(M = 1|A = a) \\ &= \frac{P(B)}{N} \sum_{e \in \{0,1\}} P(E = e) \sum_{a \in \{0,1\}} P(A = a|B, E = e)P(J = 1|A = a)P(M = 1|A = a) \end{aligned}$$

The computation involves nested loops:

- over the variables in order and
- over the different values of the variables.

Example: Burglary network III

From the probability tables in the figure we find for $b = 1$ and $b = 0$

$$P(B = b|J = 1, M = 1) = \frac{1}{N} \begin{cases} 0.00059224 & \text{for } b = 1 \\ 0.0014919 & \text{for } b = 0 \end{cases}$$

Now, from the normalization

$$P(B = 1|J = 1, M = 1) + P(B = 0|J = 1, M = 1) = 1$$

we determine N and obtain $P(B = 1|J = 1, M = 1) = 0.284$ and $P(B = 0|J = 1, M = 1) = 0.716$.

Inference algorithms

Exact inference

- NP-complete.
 - ▶ for a network with binary random variables with K nodes the complexity is $\mathcal{O}(K2^K)$
 - ▶ for 2^K different states of the variables we have to form products of K probabilities.
- Algorithms try to find an optimal order and grouping of variables.

Approximate inference

- Based on Monte Carlo sampling and Markov chain Monte Carlo sampling.
- Try to obtain samples of the joint probability distribution or the query distribution $P(\mathbf{Q}|\mathbf{Y} = \mathbf{y}) = P(\mathbf{Q}|\mathbf{y})$.

See *Russel and Norvig, Artificial Intelligence, Ch. 13, 4th Ed. 2022.* for further details.

Continuous variables

- Many real-world problems involve continuous quantities, such as mass, length, time, temperature, money, concentration,...

Continuous variables

- Many real-world problems involve continuous quantities, such as mass, length, time, temperature, money, concentration,...
- **Discretization:** Discretize the variables and generate an ordinal discrete variable.

Continuous variables

- Many real-world problems involve continuous quantities, such as mass, length, time, temperature, money, concentration,...
- **Discretization:** Discretize the variables and generate an ordinal discrete variable.
- Alternative: Specify conditional distributions for the continuous random variables.

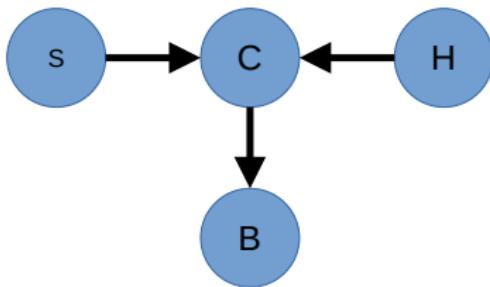
Continuous variables

- Many real-world problems involve continuous quantities, such as mass, length, time, temperature, money, concentration,...
- **Discretization:** Discretize the variables and generate an ordinal discrete variable.
- Alternative: Specify conditional distributions for the continuous random variables.
- Networks containing both discrete and continuous variables are called **Hybrid Networks**.

A hybrid network

Example

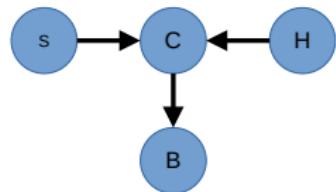
- A customer buys some fruit depending on its cost.
- The cost depends in turn on the size of the harvest and whether the government's subsidy scheme is operating.
- Variables Subsidy S and Buys B are binary.
- Harvest H in tons and cost C in Euros are continuous.



Subsidy and Buys are discrete
Harvest and Cost are continuous

Specifying conditional distributions

Example (contd.)



Subsidy and Buys are discrete
Harvest and Cost are continuous

For $P(C|S, H)$ we use a **linear Gaussian** conditional distribution.

$$P(c|h, S = 0) = \mathcal{N}(c|a_0 h + b_0, \sigma_0^2) = \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left(\frac{(c - a_0 h - b_0)^2}{2\sigma_0^2}\right)$$

$$P(c|h, S = 1) = \mathcal{N}(c|a_1 h + b_1, \sigma_1^2) = \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left(\frac{(c - a_1 h - b_1)^2}{2\sigma_1^2}\right)$$

Conditional distribution of cost

Example (contd.)

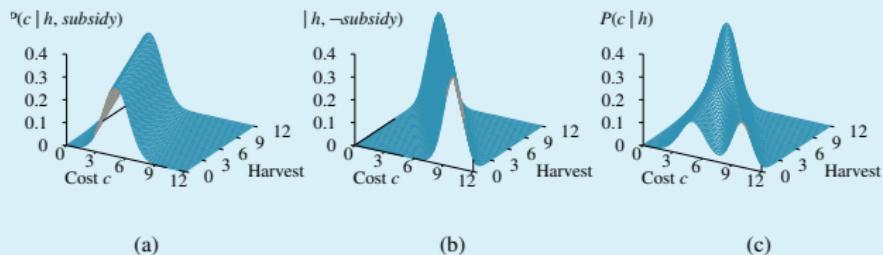
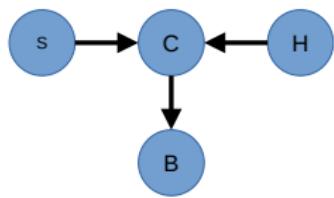


Figure 13.7 The graphs in (a) and (b) show the probability distribution over *Cost* as a function of *Harvest* size, with *Subsidy* true and false, respectively. Graph (c) shows the distribution $P(\text{Cost} \mid \text{Harvest})$, obtained by summing over the two subsidy cases.

Figure: (a) $P(c|h, S = 1)$ and (b) $P(c|h, S = 0)$. In (c) the conditional $P(c|h) = \sum_{s \in \{0,1\}} P(c|h, S = s)P(S = s)$ is plotted under the assumption $P(S = 1) = 1 - P(S = 0) = \frac{1}{2}$. From Russell and Norvig, AI, Fig. 13.7

Specifying conditional distributions for continuous parents and discrete children I

Example (contd.)



Subsidy and Buys are discrete
Harvest and Cost are continuous

For $P(V|C)$ we think that the probability to buy ($B = 1$) should decrease with increasing cost. One way to model this is the **logit** model:

$$P(B = 1|c) = 1 - \frac{1}{1 + \exp\left(\frac{-4}{\sqrt{2\pi}} \frac{c-\mu}{\sigma}\right)} = 1 - \text{sigmoid}\left(\frac{4}{\sqrt{2\pi}} \frac{c-\mu}{\sigma}\right)$$

Specifying conditional distributions for continuous parents and discrete children II

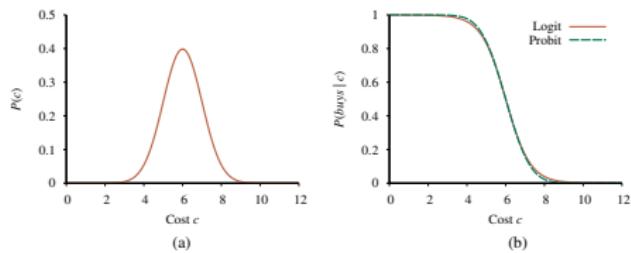


Figure 13.8 (a) A normal (Gaussian) distribution for the cost threshold, centered on $\mu = 6.0$ with standard deviation $\sigma = 1.0$. (b) Expit and probit models for the probability of *buys* given *cost*, for the parameters $\mu = 6.0$ and $\sigma = 1.0$.

Figure: Left: The Gaussian with mean $\mu = 6$ and unit standard deviation.
 Right: Inverse Logit model and probit model. From Russell and Norvig, AI, Fig. 13.8

Here, the parameters of $P(B = 1|c)$ are given by

- μ to shift the distribution and
- σ to control the width of the threshold regions.

Specifying conditional distributions for continuous parents and discrete children III

Alternative: **Probit model:**

$$P(B = 1|c) = 1 - \Phi\left(\frac{c - \mu}{\sigma}\right)$$

Here,

$$\begin{aligned} \Phi(x) &= \int_{-\infty}^x \mathcal{N}(x|0, 1) dx \\ &= \int_{-\infty}^x \frac{1}{\sqrt{2\pi}} \exp\left(\frac{-x^2}{2}\right) dx \end{aligned}$$

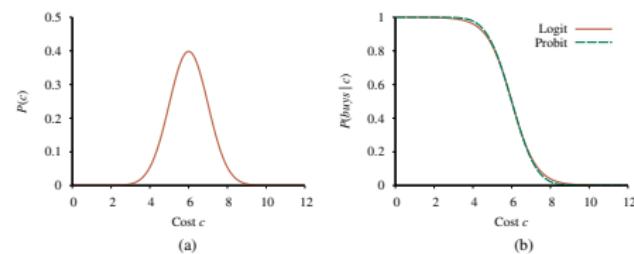


Figure 13.8 (a) A normal (Gaussian) distribution for the cost threshold, centered on $\mu = 6.0$ with standard deviation $\sigma = 1.0$. (b) Expit and probit models for the probability of *buys* given *cost*, for the parameters $\mu = 6.0$ and $\sigma = 1.0$.

Figure: Left: The Gaussian with mean $\mu = 6$ and unit standard deviation.

Right: Inverse Logit model and probit model. From Russell and Norvig, AI, Fig. 13.8

is the cumulative distribution function of a standard normal distribution.

Estimation for DAGs

There are two different tasks:

- ① Estimate the parameters in the probability distributions.
- ② Estimate the structure of the DAG.
 - ▶ Model comparison problem. See specialized Literature.
 - ▶ *Koller and Friedman, Probabilistic Graphical Models, MIT Press, 2009.*
 - ▶ *Murphy, Probabilistic Machine Learning, MIT Press, 2022.*

Parameter estimation for DAGs

Markov-factorization according to a given graph:

$$p(x_1, \dots, x_K; \theta) = \prod_{k=1}^K p(x_k | \text{par}(x_k); \theta_k)$$

Here, we have assumed that the parameters are local. This means that the parameter vector $\theta = (\theta_1, \dots, \theta_k, \dots, \theta_K)$ can be separated into distinct parameters corresponding to each node (variable) in the network.

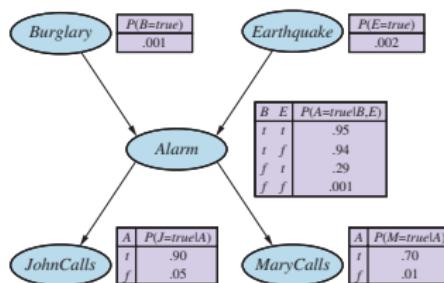


Figure: The parameters for the conditional distribution of each depend only on the node itself. No parameter refers to several nodes.

Parameter estimation for DAGs I

Maximum likelihood

Let $\mathcal{V} = (X_1, \dots, X_K)$ be the nodes (variables) of a Bayesian network.
Assume, we have n observations for each node:

Data: $\mathcal{D} = \{x_{ik}\}_{k=1,\dots,K}^{i=1,\dots,n}$

Parameter estimation for DAGs II

Maximum likelihood

Maximum Likelihood

Likelihood:

$$\begin{aligned}\mathcal{L}(\theta) &= \text{Prob}(\mathcal{D}|\theta) = \prod_{i=1}^n P(X_1 = x_{i1}, \dots, X_K = x_{iK}) \\ &= \prod_{i=1}^n p(x_{i1}, \dots, x_{iK}) \\ &= \prod_{i=1}^n \prod_{k=1}^K p(x_{ik} | \text{par}(x_{ik}); \theta_k)\end{aligned}$$

Maximum Likelihood:

$$\hat{\theta}_{ML} = \operatorname{argmax}_{\theta} \mathcal{L}(\theta)$$

Chap. 7 Outlook

Integrating Machine learning (inductive) and symbolic AI (deductive)

- Several authors argued for a synthesis of symbolic and neural (or ML) methods.
- Daniel Kahneman: Cognition as encompasses two components, system 1 and system 2 (see his book "Thinking Fast and Slow.")

System 1

- is fast, reflexive, intuitive, and unconscious,
- is used for pattern recognition.

System 2

- is slower, step-by-step, and explicit,
- handles planning, deduction, and deliberative thinking.

Example

Integrating a knowledge graph with neural processing units

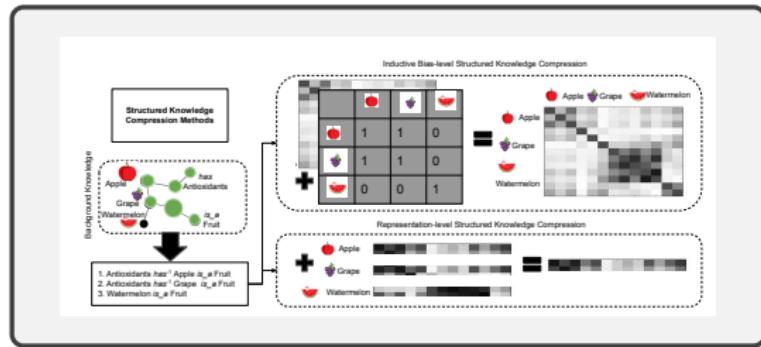


Figure: Two methods for compressing knowledge graphs to integrate them with neural processing pipelines. One approach involves embedding knowledge graph paths into vector spaces, enabling integration with the neural network's hidden representations. The other method involves encoding knowledge graphs as masks to modify the neural network's inductive biases. An example of an inductive bias is the correlation information stored in the self-attention matrices of a transformer neural network Fig.2 from Sheth, Roy and Gaur, Neurosymbolic AI-Why,What, and How.

<https://arxiv.org/abs/2305.00813v1>

Limitations of Machine Learning

Association/Correlation is not causation

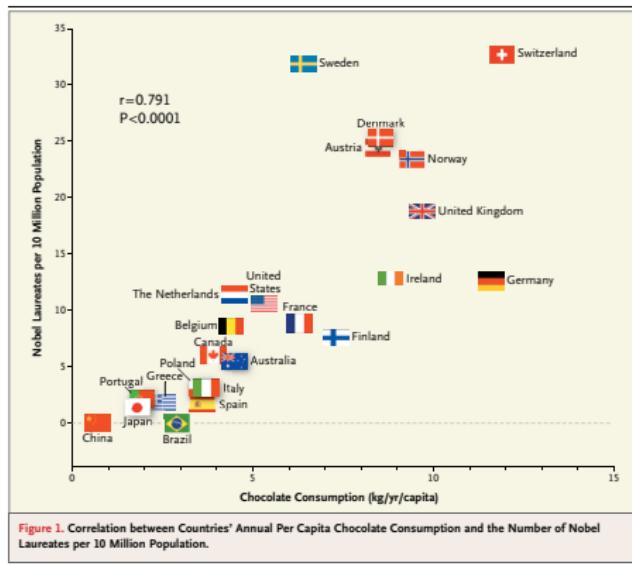


Figure: Chocolate consumption and number of nobel prices. (From Messerli, N Engl J Med 2012.)

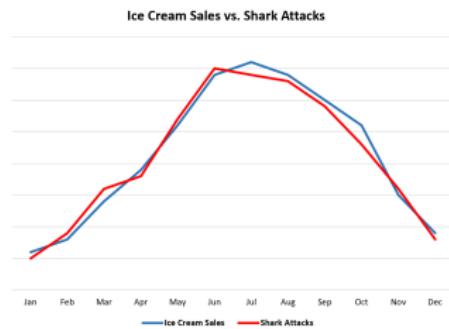


Figure: Ice cream sales and shark attacks in the USA.