# Artificial Intelligence

## Algorithms and Applications with Python

## Chapter 05

Dr. Dominik Jung
*dominik.jung42@gmail.com*

# Outline

| 5 | Knowledge Reasoning - Fundamental Algorithms and Concepts |
|---|---|

| 5.1 | Knowledge Reasoning |
|---|---|

| 5.2 | Propositional Logic |
|---|---|

| 5.3 | First Order Logic |
|---|---|

| 5.4 | Planning and Acting |
|---|---|

| 5.5 | Knowledge Representation and Engineering |
|---|---|

Lectorial 3: Building a Rule-based Agent for Credit Scoring

► **What we will learn:**
- We will design logical agents that can form representations of the world
- For that purpose we will formalize knowledge and reasoning processes with representation languages like propositional or first-order logic
- Use inference processes to derive new representations about the world, and use these new representations for decision-making and planning

Image source: ↗ Pixabay (2019) / ↗ CC0

► **Duration:**
- 270 min + 90 min (Lectorial)

► **Relevant for Exam:**
- 5.1 – 5.2, 5.4

The United States' National Parks'

- In the previous chapter build intelligent agents by modelling problems as search problems, and using clever search algorithms to solve these problems

- **Problem**: Search algorithms generate graphs or trees with successors and evaluate them, but do not "understand" anything

- In the past we tried to create intelligent agents by modelling problems as search problems, and using clever search algorithms

- However, as we have discussed, most real-world problems do not have well-definied solutions and hence our „search-approach" will not work for these kind of problems human can solve…

- What do humans do?
  → Humans know things and use it to act.

- Or, as AI researchers say, they have **real-world knowledge** and **do reasoning** to solve real-world problems



*That's what I do. I drink, and I know things.*

- Knowledge and reasoning is important when dealing with partially observable environments

- Knowledge-based agents and systems benefit from knowledge expressed in general forms combining in there inference process

- **Our new approach**: Find a way to represent knowledge and implement inferences procedures to derive solutions in our agents

**D** **Knowledge-based System**

An expert system or knowledge-based system is one that solves problems by applying knowledge that has been garnered from one or more experts in a field (Norvig, 1992)

# 5.1 Example: Credit Scoring System



## Scenario

You work in a big car company and are asked to build a credit scoring agent to automate most of the car credits. Consider the following aspects:

- Customers have to register succesfully with their credit card and correct pin when they want to buy the car with a financial leasing contract.
- Furthermore, the customer has to pay at least the minimum montly rate
- If he has no other open car credits, the contract is accepted, otherweise there has to be a manual check of the financial situation

**?** How can you / an AI conclude if a customer gets a credit or not?

| Variable | Values |
|---|---|
| Credit Card | {accepted, not accepted} |
| PIN | {true, false} |
| Montly credit rate | {≤ minimum amount, > minimum amount} |
| Other open credits | {true, false} |
| Accept | {yes, no} |
| Manual check | {ok, not ok} |

Adapted from Castillo, E. et al. (2012); Beierle, C., & Kern-Isberner, G. (2014) | Image source: ↗ Pixabay (2019) / ↗ CC0

# 5.1 Implement Credit Reasoning

- The central point of a knowledge-based system is to represent the problem-related knowledge as directly as possible

- For that purpose we have to keep the actual processing separate from knowledge representation

- One very intuitive approach for such a direct knowledge representation is the use of rules.

**D** A rule is a logical statement that relates two or more objects and includes two parts, the premise and the conclusion (Castillo, E et al. 2012, p.23)

```
[CS.1]
if
    CREDIT_CARD = accepted and
    PIN = true and
    MONTLY_CREDIT_RATE > MIN_AMOUNT and
    OTHER_CREDITS = false

then
    ACCEPT = yes
```
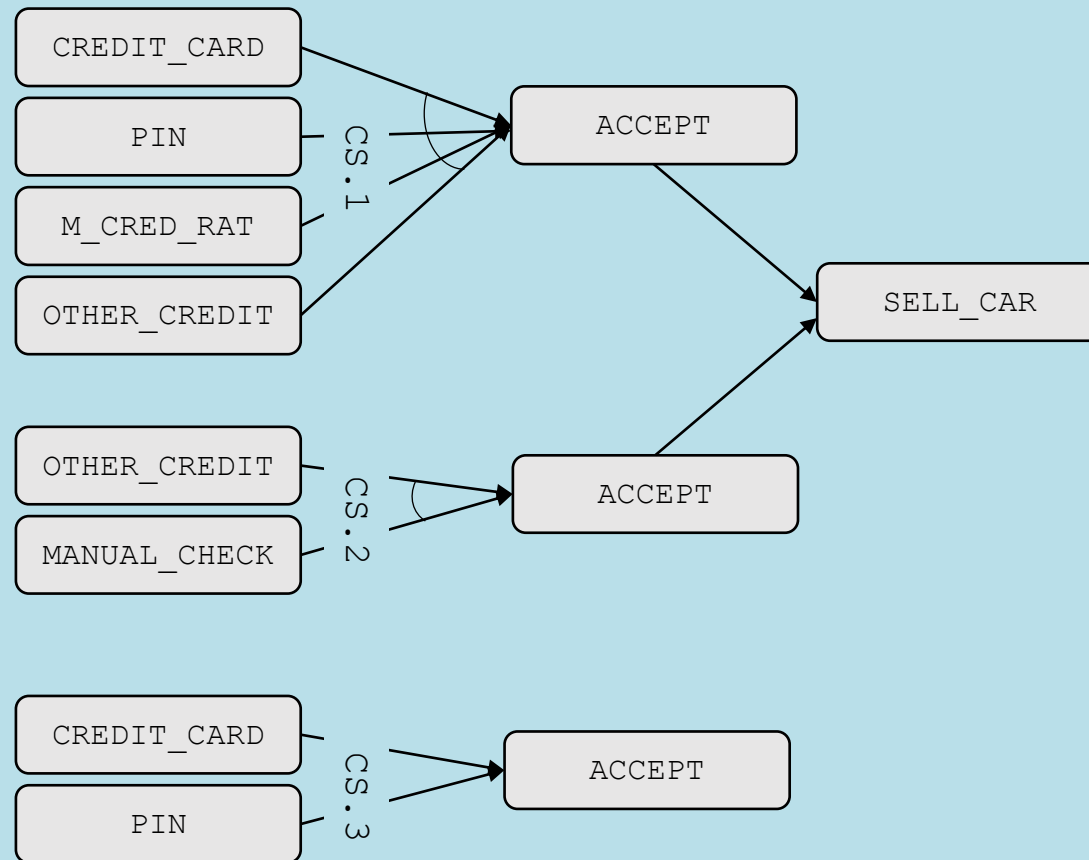
```
[CS.2]
if
    OTHER_CREDITS = true and
    MANUAL_CHECK = not_ok

then
    ACCEPT = no
```

Adapted from Castillo, E. et al. (2012); Beierle, C., & Kern-Isberner, G. (2014)

# 5.1 Rule Networks



```
[CS.1]
if
    CREDIT_CARD = accepted and
    PIN = true and
    MONTLY_CREDIT_RATE > MIN_AMOUNT and
    OTHER_CREDITS = false
then
    ACCEPT = yes
```

```
[CS.2]
if
    OTHER_CREDITS = true and
    MANUAL_CHECK = not_ok
then
    ACCEPT = no
```

```
[CS.3]
if
    CARD_DECLINED = true or
    SCHUFA_WARNING = true
then
    ACCEPT = no
```
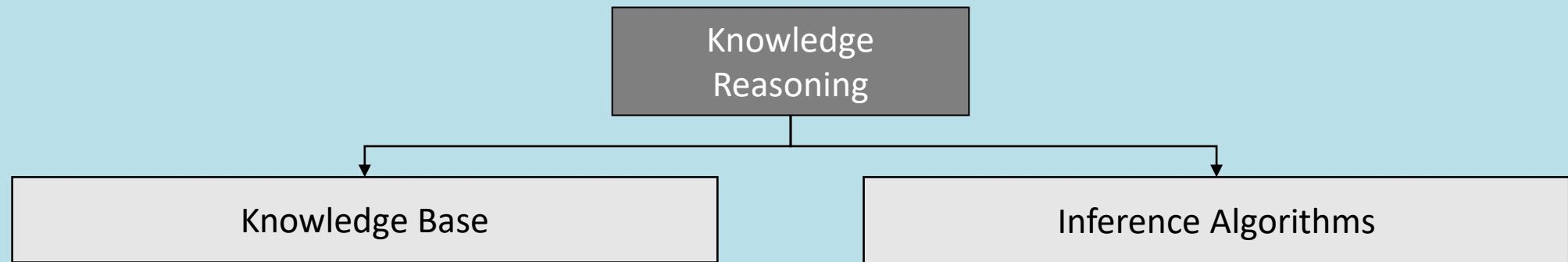
**D** **Knowledge Reasoning**

Knowledge Representation is the study of how "what we know" can at the same time be represented as comprehensibly as possible and reasoned with as effectively as possibly from an information system (based on Brachman & Levesque, 1992).

Knowledge Reasoning

Knowledge Base

Inference Algorithms

Seperation of concerns:
- clear distinction of problem description and problem solution
- knowledge of domain can be expressed directly

Adapted from Beierle, C., & Kern-Isberner, G. (2014); Rusell, S., & Norvig, P. (2016)

Inference Algorithms
(Domain-independent algorithms)

**+**

Knowledge Base
(Domain-specific content)

**= Intelligent Decision**

- Knowledge base (KB) = set of sentences in a formal language

- There are mainly two approaches to implement the agent's logic (or other type of AI system):
  - Declarative approach
  - Procedural approach

- Distinction between data and program

**D** **Knowledge Base**

The component of an expert system that contains the system's knowledge organized in collection of sentences about the system's domain.

- **Knowledge level**: What the agent knows, and the agent's goals.

- **Logical level**: how the knowledge representation of knowledge is stored. At this level, sentences can be encoded into different logics

- **Implementation level**: physical representation of logic and knowledge. At the implementation level agent perform actions as per logical and knowledge level.
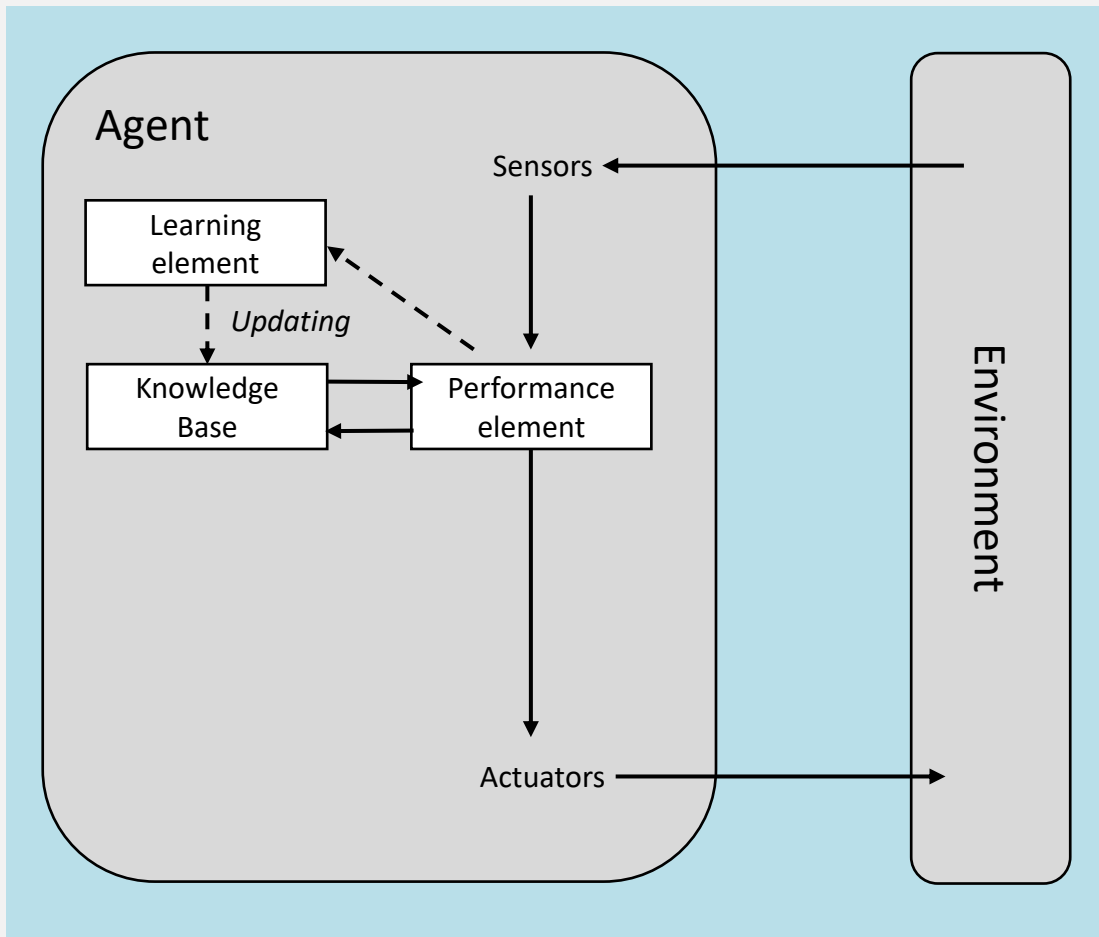
# 5.1 Knowledge-based Agent

▶ Adds percepts and action's results to knowledge base, and derives actions from the knowledge base



### Agent

- Learning element
  - *Updating*
- Knowledge Base
- Performance element
- Sensors
- Actuators

### Environment

## Inference Procedure

1. Agent TELLs the knowledge base what it perceives.
2. Agent ASKs the knowledge base what action it should perform
3. The Agent TELLs the knowledge base which action was chosen, and then executes the action.
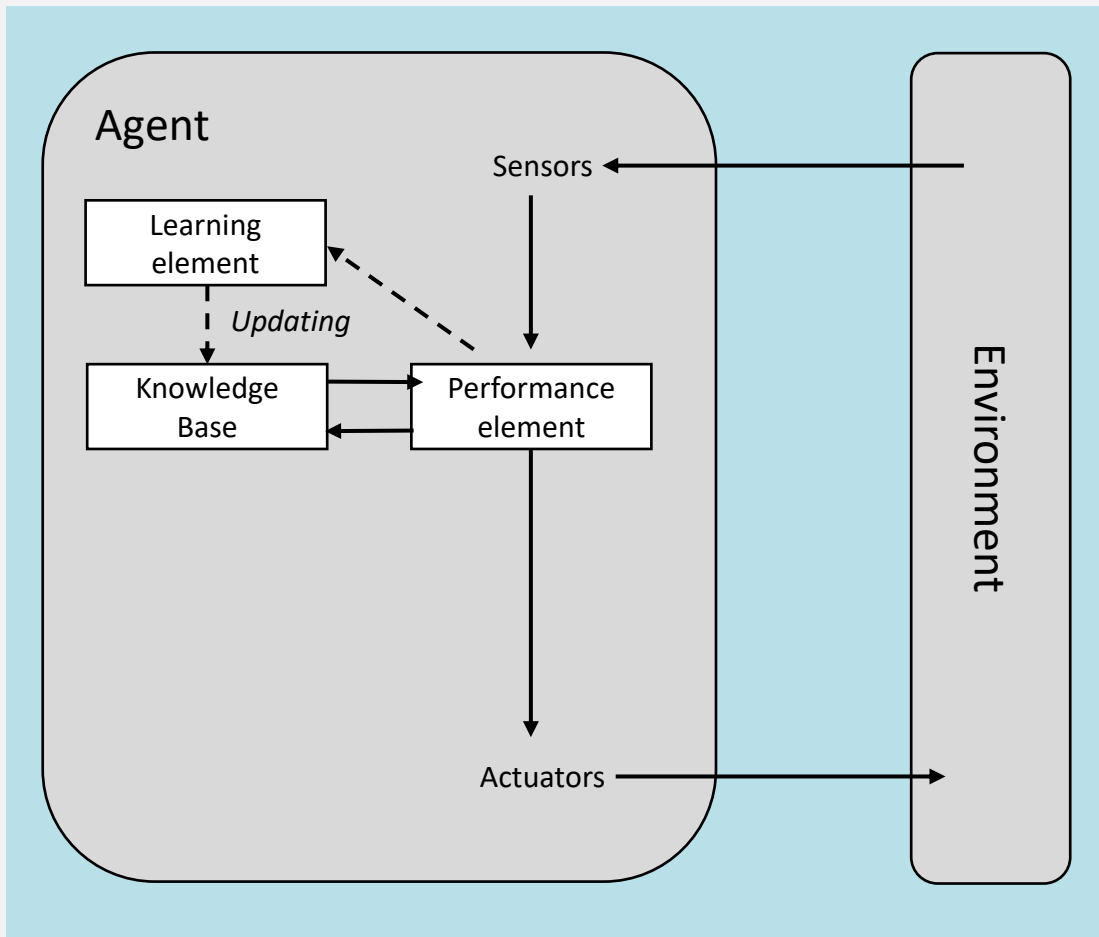
## Knowledge Base

Knowledge is represented in the knowledge base
- Case-Specific knowledge (temporary)
- Rule-based knowledge
  - Domain knowledge
  - General knowledge

Adapted from Rusell, S., & Norvig, P. (2016)

# 5.1 Knowledge-based Agent

▶ Adds percepts and action's results to knowledge base, and derives actions from the knowledge base



**Algorithm:** Knowledge-based Agent

$persistent$:
$KB, a\ knowledge\ base\ t,$
$a\ counter, initially\ 0, indicating\ time$

$TELL(KB, MAKE - PERCEPT - SENTENCE(percept, t))$

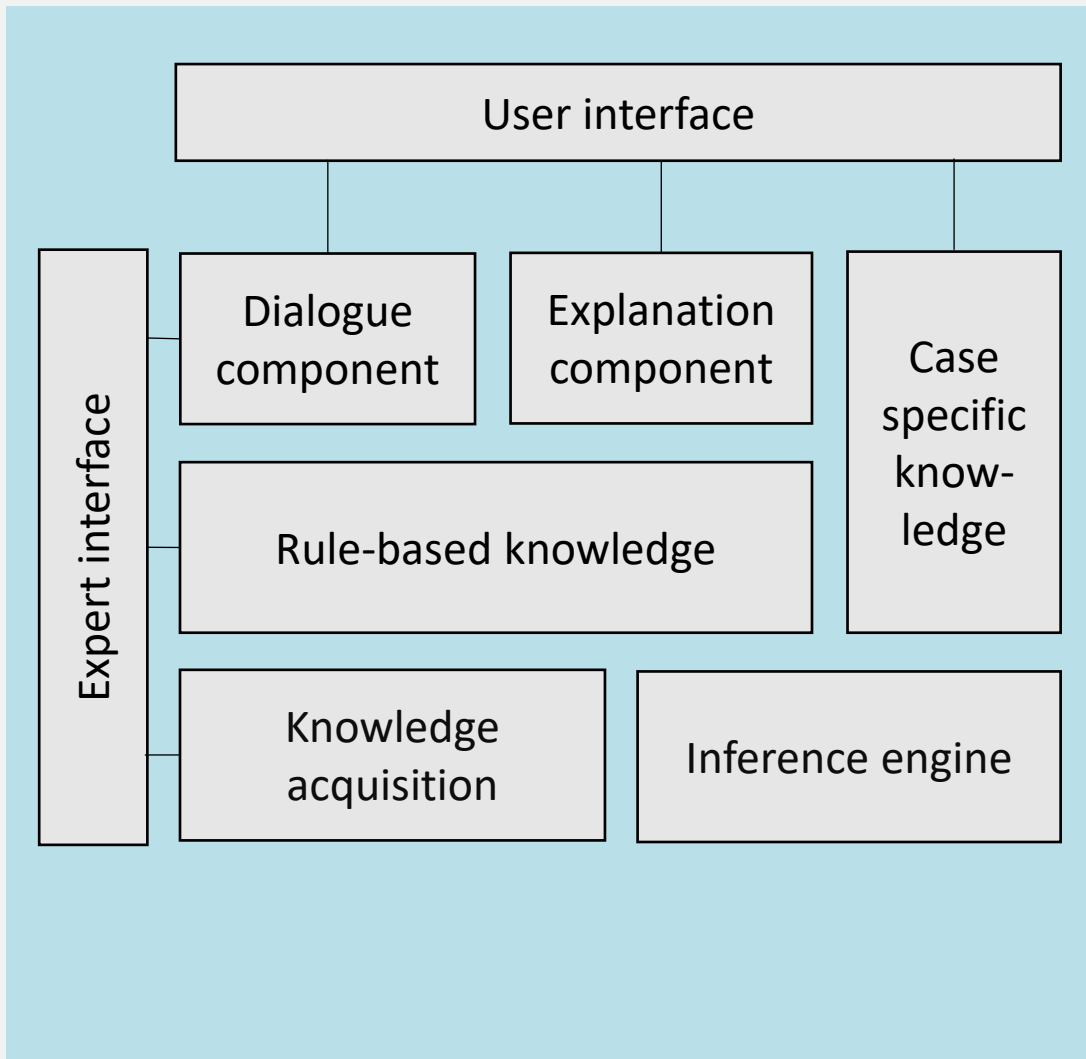$action \leftarrow ASK(KB, MAKE - ACTION - QUERY(t))$

$TELL(KB, MAKE - ACTION - SENTENCE(action, t))$

$t \leftarrow t + 1$

$return\ action$

Adapted from Rusell, S., & Norvig, P. (2016)

User interface

Expert interface

Dialogue component

Explanation component

Case specific know-ledge

Rule-based knowledge

Knowledge acquisition

Inference engine

Adapted from Beierle, C., & Kern-Isberner, G. (2014)

**From Knowledge to Expert Knowledge**

- We assume, that experts have above-average abilities to solve problems in a specific field

- Experts can solve problems using incomplete and uncertain Solving knowledge

- Experts use heuristic knowledge to solve specific problems and exploit their experience

- Experts are rare and expensive, and their knowledge can be lost if they are lost

# 5.1 Architecture of Knowledge-based System (KBS)
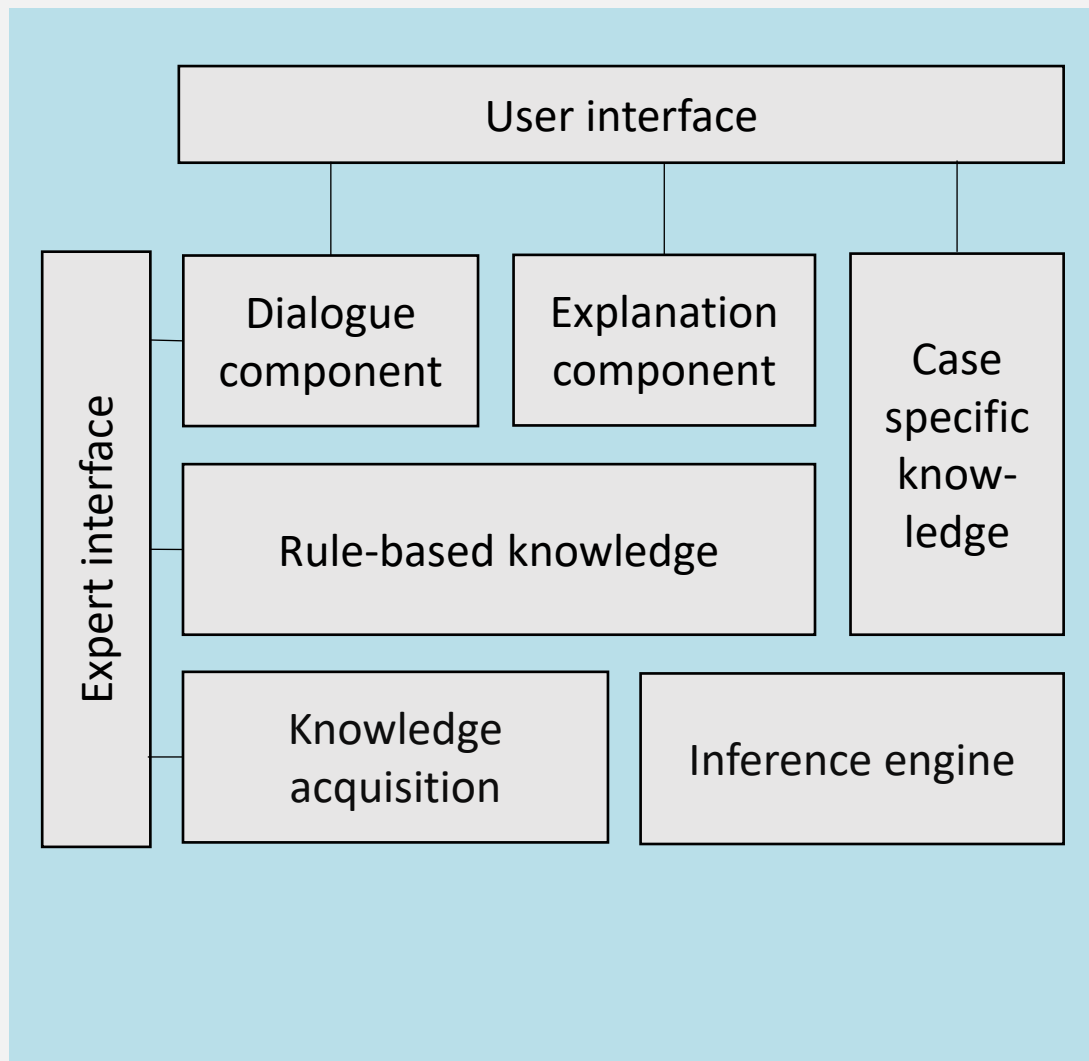


Adapted from Beierle, C., & Kern-Isberner, G. (2014)

## Inference Engine

- An inference engine tries to dervice answers from a knowledeg base
- It is the „brain" of the expert system that provides a methodology for reasoning about the information in the knowledge base, and for formulating conclusions

## Interfaces

- **Expert interface**: Add and store new expert knowledge in the knowledge base, revise existing knowledge

- **User interface**: Specify situation and ask for expert advice

- **Dendral**: Pioneering work developed in 1965 for NASA at Standford University

- **Drilling Advisor**: Developed in 1983 by Teknowledge for oil companies to replace human drilling advisors

- **Mycin**: Developed in 1970 at Stanford by Shortcliffe to assist internists in diagnosis and treatment of infectious diseases

- **Xcon/RI**: Developed in 1978 to assist the ordering of computer systems by automatically selecting the system components based on customer's requirements
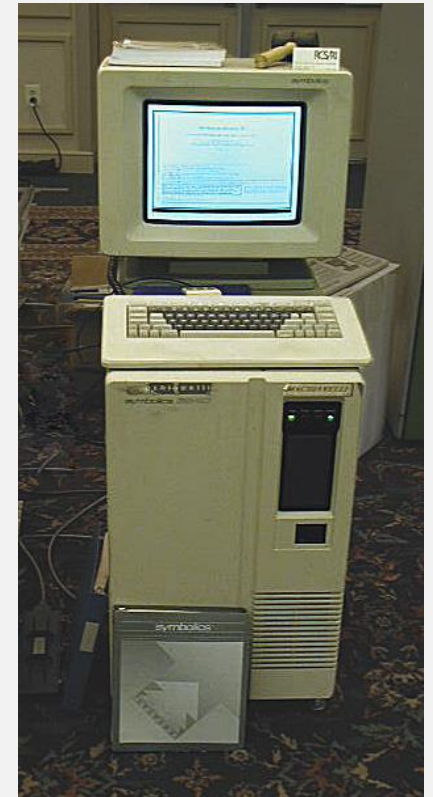
Image source: ↗ A Symbolics Lisp Machine (2019) by Michael L. Umbricht and Carl R. Friend (Retro-Computing Society of RI) / ↗ CC BY-SA 3.0

Adapted from Beierle, C., & Kern-Isberner, G. (2014)

- To build knowledge-based agents we have to specify the knowledge (what an agent has to know), and the agent logic (how knowledge is processed in the agent)

- For that purpose:
  - An agent should be able to represent states, actions, etc.
  - An agent should be able to incorporate new percepts, and to update internal representations of the world
  - An agent can deduce hidden properties of the world, and deduce appropriate actions

Adapted from Beierle, C., & Kern-Isberner, G. (2014); Rusell, S., & Norvig, P. (2016)

# Outline

| 5 | Knowledge Reasoning - Fundamental Algorithms and Concepts |
|---|---|

| 5.1 | Knowledge Reasoning |
|---|---|
| 5.2 | Propositional Logic |
| 5.3 | First Order Logic |
| 5.4 | Planning and Acting |
| 5.5 | Knowledge Representation and Engineering |

Lectorial 3: Building a Rule-based Agent for Credit Scoring

►**What we will learn:**
- We will design logical agents that can form representations of the world
- For that purpose we will formalize knowledge and reasoning processes with representation languages like propositional or first-order logic
- Use inference processes to derive new representations about the world, and use these new representations for decision-making and planning
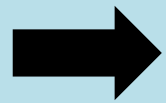
Image source: ↗ Pixabay (2019) / ↗ CC0

►**Duration:**
- 270 min + 90 min (Lectorial)

►**Relevant for Exam:**
- 5.1 – 5.2, 5.4

Knowledge Reasoning

Knowledge Base

Inference Algorithms

➡ Define formal system for handling facts so that in each case where the agent draws conclusion from the available information, that conclusion is guaranteed to be correct if the available information is correct.

Adapted from Rusell, S., & Norvig, P. (2016)

- A reasonig process usally consists of two parts: antecedant, inference rule, and conclusion
- The objective of reasoning is to find the missing component



A — I → C
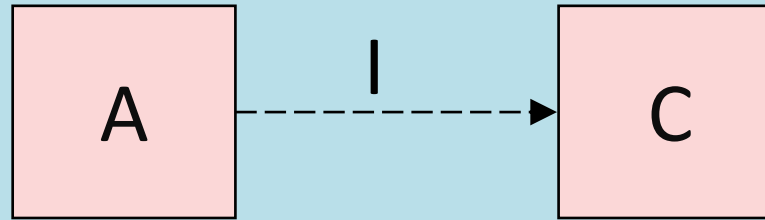
Knowledge          New Knowledge
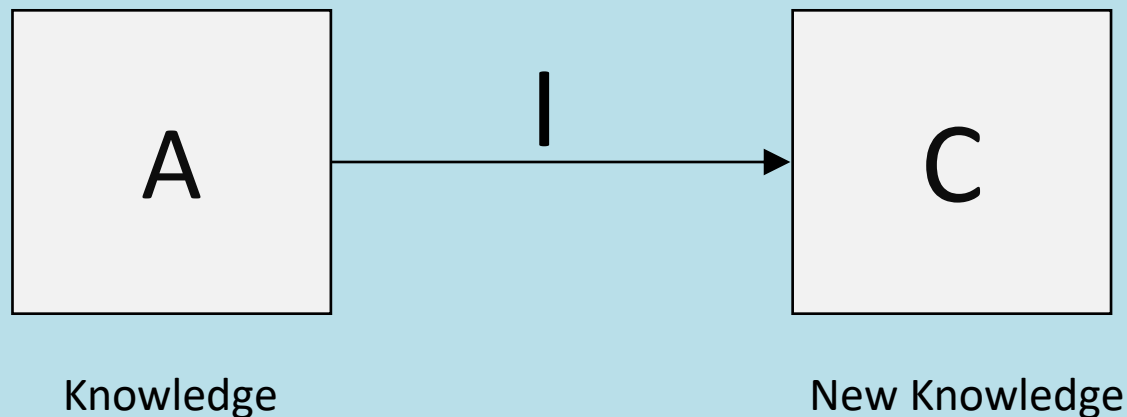
- Conclusion is missing: Deduction
- Inference is missing: Induction
- Antecedant is missing: Abduction

Adapted from Castillo, E. et al. (2012); Beierle, C., & Kern-Isberner, G. (2014); Peirce, C (1931, 103 ff.)

# 5.2 Induction

**Visualization**

A ⇢ I ⇢ C

**Characteristics**

- Given a set of D of basic examples. The hypothesis H follows inductively from D and background knowledge B
- <=> $B \cup H \rightarrow D, B \not\rightarrow D, B \cup D \not\rightarrow \neg H.$

- **Application**: Data Mining, Economy

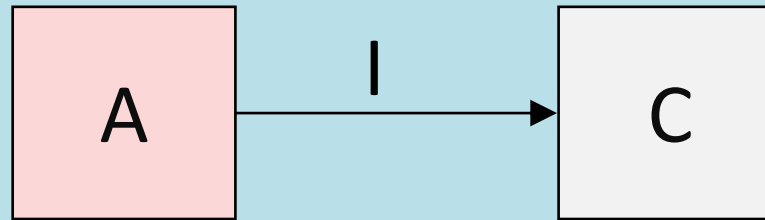Case: These beans are [randomly selected] from this bag.

Result: These beans are white.

Rule: All the beans from this bag are white.

Adapted from Castillo, E. et al. (2012); Beierle, C., & Kern-Isberner, G. (2014); Peirce, C (1931, 103 ff.)

# 5.2 Deduction

## Visualization

A ──I──> C

## Characteristics

- From a set of formulas F follows B
     <=> There is a sequence of rules to derive B

- $\dfrac{F, F \rightarrow B}{B}.$

- **Application**: Mathematics

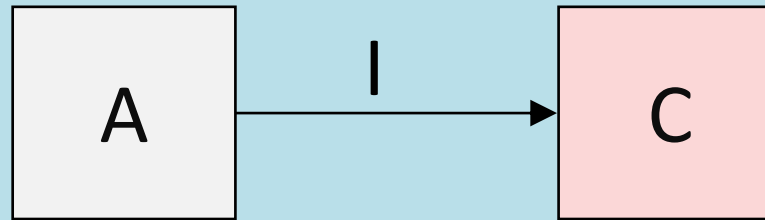Rule: All the beans from this bag are white.

Case: These beans are from this bag.

Result: These beans are white.

Adapted from Castillo, E. et al. (2012); Beierle, C., & Kern-Isberner, G. (2014); Peirce, C (1931, 103 ff.)

## Visualization

$$A \xrightarrow{\ I\ } C$$

## Characteristics

- H follows from background knowledge B and observations $D$

- abductive <=> $B \cup H \rightarrow D$

- **Application**: Medical Diagnosis, Car Repairing, Failure Explanation

Rule: All the beans from this bag are white.

Result: These beans [oddly] are white

.

Case: These beans are from this bag.

Adapted from Castillo, E. et al. (2012); Beierle, C., & Kern-Isberner, G. (2014); Peirce, C (1931, 103 ff.)

- **We need rules** and systematics for manipulating facts **to allow computational reasoning**

- **Logic** is such a formal system for **handling facts so that true conclusions may be drawn**

- We can apply logical rules to our knowledge base to deduce new information
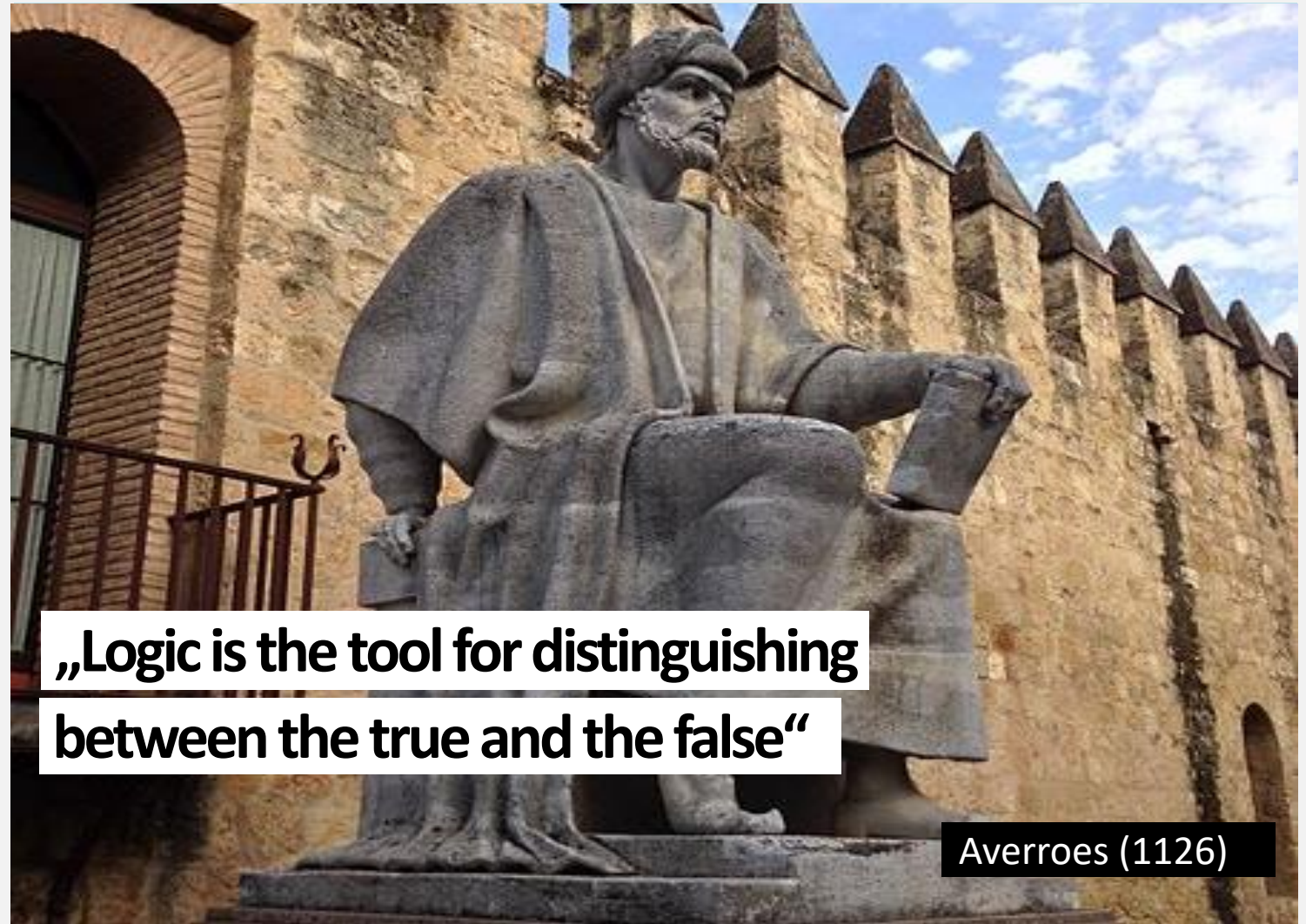


„Logic is the tool for distinguishing between the true and the false"

Averroes (1126)

- **Syntax:** rules for constructing valid sentences
  - e.g., $x + 2 = y$ is a valid arithmetic sentence

- **Semantics:** "meaning" of sentences, or relationship between logical sentences and the real world
  - Specifically, semantics defines truth of sentences
  - e.g., $x + 2 = y$ is true in a world ($\approx$ model) where $x = 5$ and $y = 7$
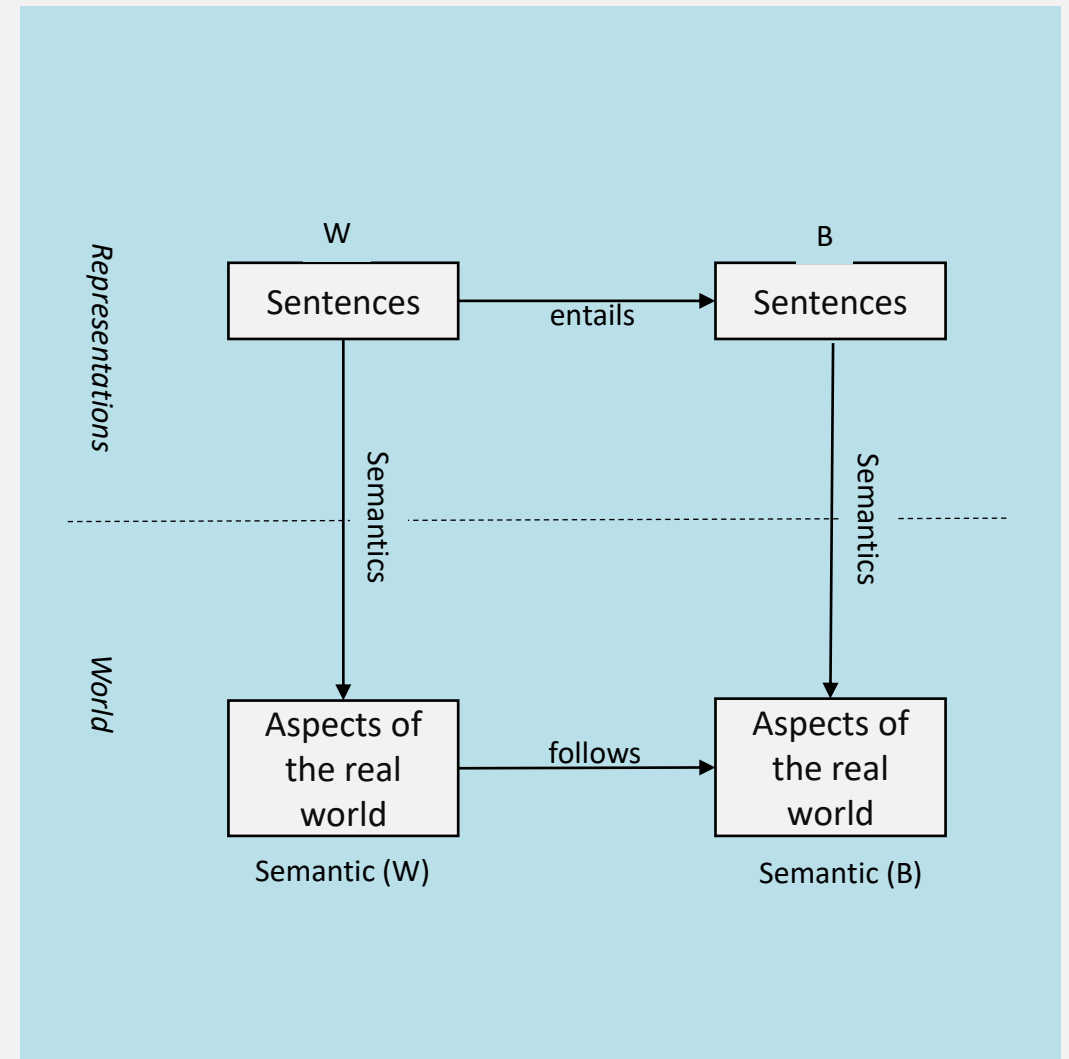


Adapted from Beierle, C., & Kern-Isberner, G. (2014; Rusell, S., & Norvig, P. (2016) | Image source: Rusell, S., & Norvig, P. (2016)

- **Atomic sentence (literal):**
  - A *proposition symbol* representing a true or false statement

- **Negation:** If **A** is a sentence, ¬**A** is a sentence

- **Conjunction:** If **A** and **B** are sentences, **A** ∧ **B** is a sentence

- **Disjunction:** If **A** and **B** are sentences, **A** ∨ **B** is a sentence

- **Implication:** If **A** and **B** are sentences, **A** ⇒ **B** is a sentence

- **Biconditional:** If **A** and **B** are sentences, **A** ⇔ **B** is a sentence

The related symbols **¬, ∧, ∨, ⇒, ⇔** are called *logical connectives*

- The semantic defines how the truth of a sentence is computed.

- For that, propositional logic has the following rules:

Given a model where…
   **A** is true, $\neg$**A** is false
   **A and B** is true, **A** $\wedge$ **B** is true
   **A or B** are true, **A** $\vee$ **B** is true
   **A** is **true** and **B** is false, **A** $\Rightarrow$ **B** is false
   **A and B** are both true or false, **A** $\Leftrightarrow$ **B** is true

Adapted from Rusell, S., & Norvig, P. (2016)

- The semantic defines how the truth of a sentence is computed.

- A truth table specifies the truth value of a composite sentence for each possible assignments of truth values to its atoms

| A | B | ¬A | A ∧ B | A ∨ B | A => B |
|---|---|-----|-------|-------|--------|
| false | false | true | false | false | true |
| false | true | true | false | true | true |
| true | false | false | false | true | false |
| true | true | false | true | true | true |

- The truth value of a more complex sentence can be evaluated *recursively* or *compositionally*

Adapted from Rusell, S., & Norvig, P. (2016)

- Two sentences are logically equivalent if they have the same truth value for every setting of their propositional variables

| A | B | $A \vee B$ | $\neg(\neg A \wedge \neg B)$ |
|---|---|---|---|
| false | false | false | false |
| false | true | true | true |
| true | false | true | true |
| true | true | true | true |

- `A OR B` and `NOT(`$\neg$`A` $\wedge$ $\neg$`(B))` are logically equivalent; Tautology = logically equivalent to True

Adapted from Rusell, S., & Norvig, P. (2016)

**D**

| Commutatitvity | $(a \; OR \; b) \; \equiv \; (b \; OR \; a)$ |
|---|---|
| | $(a \; AND \; b) \; \equiv \; (b \; AND \; a)$ |
| Associativity | $((a \; AND \; b) \; AND \; c) \; \equiv \; (a \; AND \; (b \; AND \; c))$ |
| | $((a \; OR \; b) \; OR \; c) \; \equiv \; (a \; OR \; (b \; OR \; c))$ |
| Double-negation elimination | $NOT(NOT(a)) \; \equiv \; a$ |
| Contraposition | $(a \; => \; b) \; \equiv \; (NOT(b) \; => \; NOT(a))$ |
| Implication elimination | $(a \; => \; b) \; \equiv \; (NOT(a) \; OR \; b)$ |
| De Morgan | $NOT(a \; AND \; b) \; \equiv \; (NOT(a) \; OR \; NOT(b))$ |
| | $NOT(a \; OR \; b) \; \equiv \; (NOT(a) \; AND \; NOT(b))$ |
| Distributitivity | $(a \; AND \; (b \; OR \; c)) \; \equiv \; ((a \; AND \; b) \; OR \; (a \; AND \; c))$ |
| | $(a \; OR \; (b \; AND \; c)) \; \equiv \; ((a \; OR \; b) \; AND \; (a \; OR \; c))$ |

Adapted from Rusell, S., & Norvig, P. (2016)

- **Entailment** means that a sentence follows from the premises contained in the knowledge base (or a model):

$$KB \models \alpha$$

- Knowledge base *KB* entails sentence $\alpha$ if and only if $\alpha$ is true in all models where *KB* is true
  - e.g., x + y = 4  entails  4 = x + y

- **Logical inference algorithm:** a procedure for generating sentences that follow from a knowledge base KB

- Two central characteristics: soundness and completeness

**D** **Soundness**

Inference algorithm derives true conclusions given true premises

**D** **Completeness**

Inference algorithm derives all true conclusions from a set of premises

Adapted from Castillo, E. et al. (2012); Beierle, C., & Kern-Isberner, G. (2014)

- To make implement intelligent behavior our agent has to check whether a sentence α is entailed by its KB or model

- For that we could enumerate all possible models of the KB (truth assignments of all its symbols), and check that α is true in every model in which KB is true
  - Is this sound?
  - Is this complete?

- **Problem**: if KB contains n symbols, the truth table will be of size $2^n$

- **Better idea**: use *inference rules*, or sound procedures to generate new sentences or *conclusions* given the *premises* in the KB

Adapted from Castillo, E. et al. (2012); Beierle, C., & Kern-Isberner, G. (2014)

- Modus Ponens

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

$\longleftarrow$ premises

$\longleftarrow$ conclusion

- And-elimination

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

Adapted from Castillo, E. et al. (2012); Beierle, C., & Kern-Isberner, G. (2014)

**And-introduction**

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

**Double negative elimination**

$$\frac{\neg \neg \alpha}{\alpha}$$

**Or-introduction**

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

**Unit resolution**

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

- Furthermore, we need to prove that our inference procedures are complete

- To prove a single sentence, e.g. $KB \models \alpha$, assume $KB \wedge \neg \alpha$ and derive a contradiction

$$\frac{\alpha \vee \beta, \neg \beta \vee \gamma}{\alpha \vee \gamma} \qquad \text{or} \qquad \frac{\alpha \vee \beta, \beta \Rightarrow \gamma}{\alpha \vee \gamma}$$

- For that purpose, we have to rewrite $KB \wedge \neg \alpha$ as a disjunctions of literals or conjunction of clauses (CNF)

Adapted from Castillo, E. et al. (2012); Rusell, S., & Norvig, P. (2016)

**D**

A sentence is in conjunctive normal form (CNF) if it is the conjunction of one or more clauses, where a clause is a disjunction of literals. A set of sentences is in CNF if each sentence is in CNF.

- $A \wedge B$

- $\neg A \wedge \neg B$

- $(A \vee B) \wedge (\neg C)$

Adapted from Castillo, E. et al. (2012); Rusell, S., & Norvig, P. (2016)

# Example: (p ⇒ q) ⇔ r

## Converting to CNF

1. Replace every occurrence of $\alpha \Leftrightarrow \beta$ by $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$

2. Replace every occurrence of $\alpha \Rightarrow \beta$ by $\neg\alpha \vee \beta$

3. Replace every occurrence of $\neg(\alpha \vee \beta)$ by $\neg\alpha \wedge \neg\beta$; every occurrence of $\neg(\alpha \wedge \beta)$ by $\neg\alpha \vee \neg\beta$; and every occurrence of $\neg\neg\alpha$ by $\alpha$. Repeat as long as possible

4. Replace every occurrence of (α ∧ β) ∨ γ by (α ∨ γ) ∧ (β ∨ γ), and every occurrence of α ∨ (β ∧ γ) by (α ∨ β) ∧ (α ∨ γ)

Given:

1. $\neg (p \rightarrow q) \vee (r \rightarrow p)$

2. $\neg (\neg p \vee q) \vee (\neg r \vee p)$

3. $(p \wedge \neg q) \vee (\neg r \vee p)$

4. $(p \vee \neg r \vee p) \wedge (\neg q \vee \neg r \vee p)$

**Algorithm: PL-RESULUTION(KB,$\alpha$)**

*Inputs*: *KB, a knowledge base*
$\quad\quad\quad\alpha$ *sentence of propositional logic*

*clauses* $\leftarrow$ *set off clauses in CNF of {KB$\land \neg\alpha$}*
*new$\leftarrow$ { }*

*loop do*
$\quad$ **for each** *pair of clauses $C_i, C_j$* **in** *clauses* **do**
$\quad\quad$ *resolvents $\leftarrow$ PL-RESOLVE($C_i, C_j$)*
$\quad\quad$ *if resolvents contains { }* **then return** *true*
$\quad\quad$ *new$\leftarrow$ new $\cup$ resolvents*
$\quad$ *if new clauses* **then return** *false*
$\quad$ *clauses $\leftarrow$ clauses $\cup$ new*

- Given formula in conjunctive normal form, repeat:
- Find two clauses with complementary literals,
- Apply resolution,
- Add resulting clause (if not already there)
- If the empty clause results, formula is not satisfiable
  - Must have been obtained from P and NOT(P)
- Otherwise, if we get stuck (and we will eventually), the formula is guaranteed to be satisfiable (proof in a couple of slides)

Adapted from Rusell, S., & Norvig, P. (2016)

- Keep applying resolution to clauses that contain complementary literals and adding resulting clauses to the list

- Break If there are no new clauses to be added, then KB does not entail α

- Or If two clauses resolve to form an empty clause, we have a contradiction and KB $\models$ α
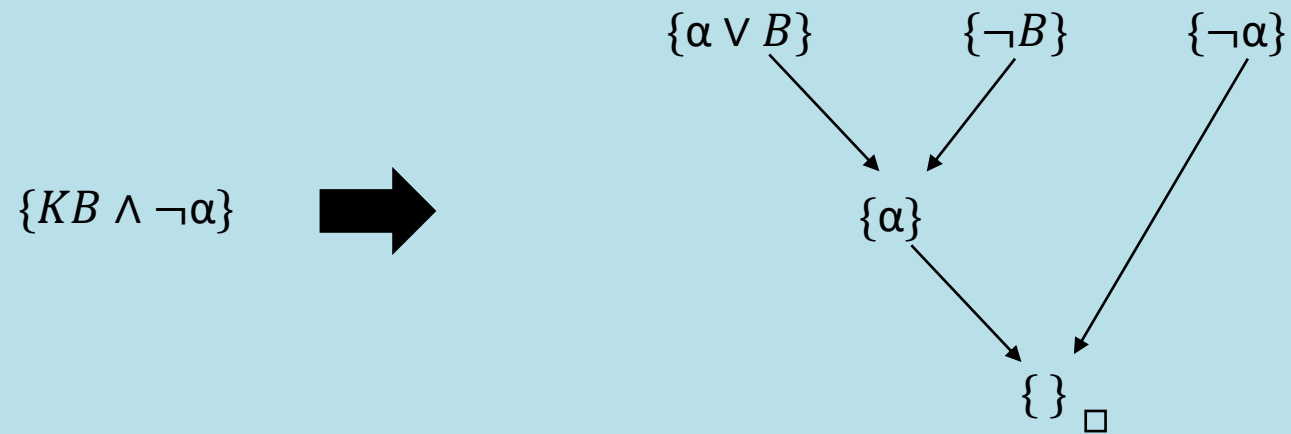
$$\{KB \wedge \neg\alpha\} \quad \Rightarrow$$

$$\{\alpha \vee B\} \quad \{\neg B\} \quad \{\neg \alpha\}$$

$$\{\alpha\}$$

$$\{\} \quad \square$$

Adapted from Castillo, E. et al. (2012); Rusell, S., & Norvig, P. (2016)

- Horn clauses are implications with only positive literals

$$x_1\ AND\ x_2\ AND\ x_4\ =>\ x_3\ AND\ x_6$$
$$TRUE\ =>\ x_1$$

- A definite clause is a disjunction with exactly one positive literal

- Knowledge bases with only definitive clauses have many positive characteristics, which allow efficient inference mechanisms (see Russel & Norvig, 2016)

Adapted from Castillo, E. et al. (2012); Rusell, S., & Norvig, P. (2016)

- So far, we know the modus ponens for simple 1-rule inference

$$\frac{\alpha \Rightarrow \beta, \alpha}{\beta}$$ ➡ $$\frac{\textbf{\textit{if}} \; MANUAL-CHECK = FAILED, \; \textbf{\textit{then}} \; ACCEPT = NO}{MANUAL-CHECK = FAILED}{ACCEPT = NO}$$

- However, rule-based Systems can use multiple rules for inference. There exists three popular approaches for that:
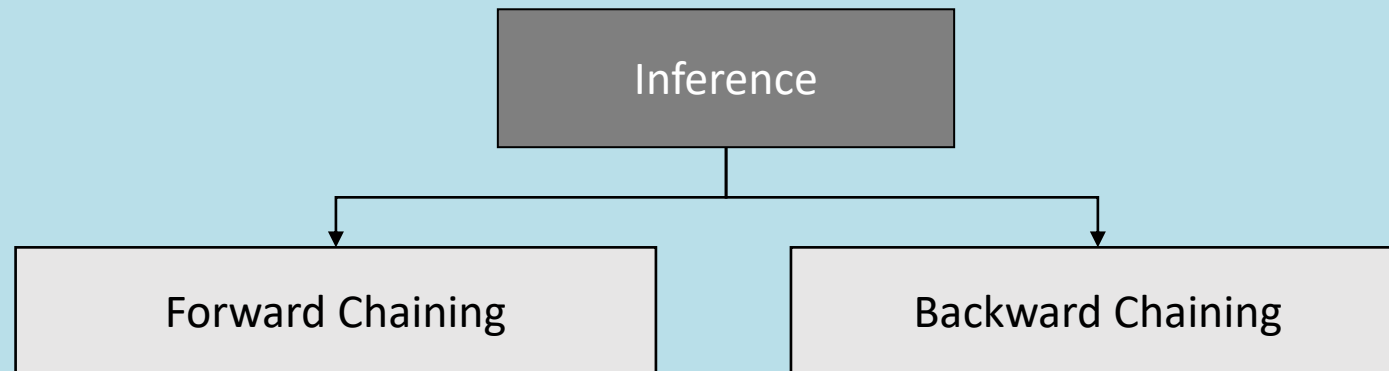
```
                    Inference

        Forward Chaining        Backward Chaining
```
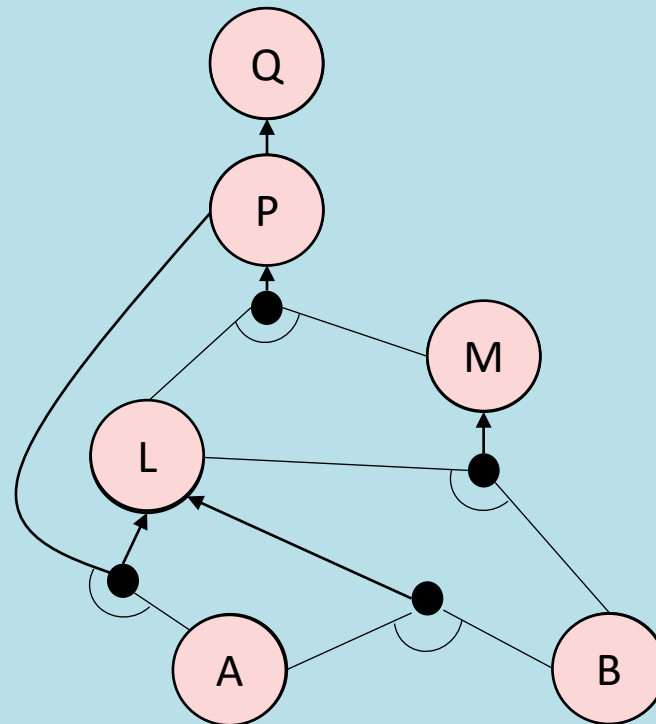
▪ **Idea**: find any rule whose premises are satisfied in the *KB*, add its conclusion to the *KB*, and keep going until query is found
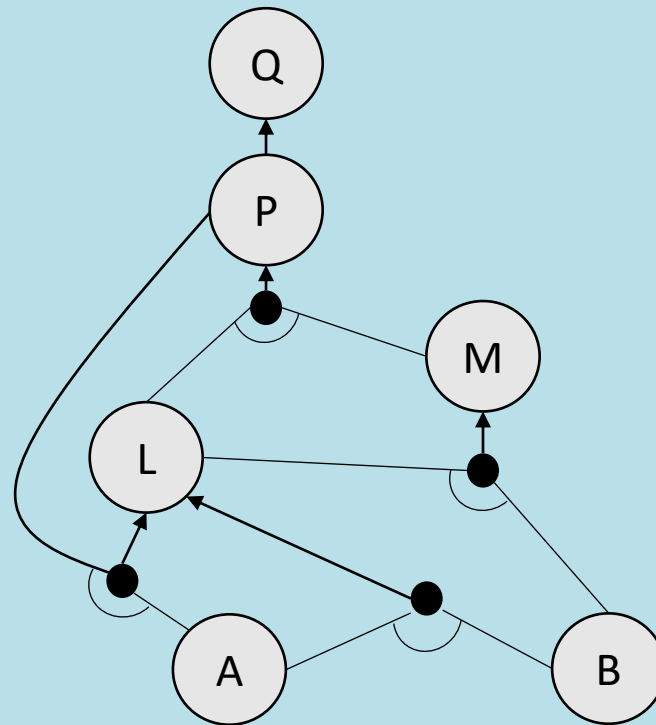


$$P \Rightarrow Q$$

$$L \wedge M \Rightarrow P$$

$$B \wedge L \Rightarrow M$$

$$A \wedge P \Rightarrow L$$

$$A \wedge B \Rightarrow L$$

$$A$$

$$B$$

Adapted from Castillo, E. et al. (2012); Beierle, C., & Kern-Isberner, G. (2014); Rusell, S., & Norvig, P. (2016)
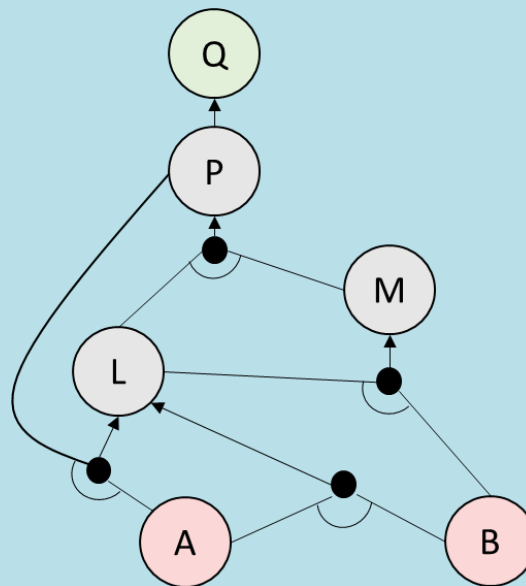
$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

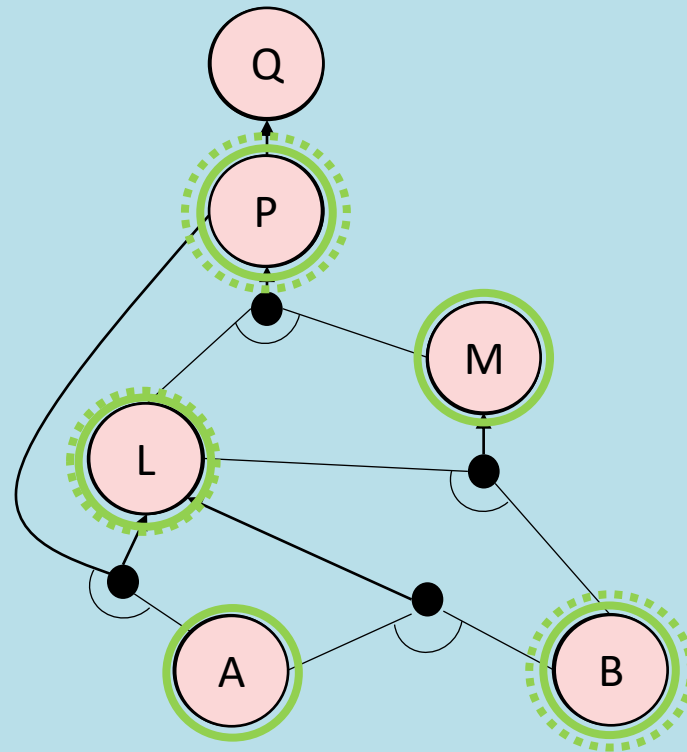Adapted from Castillo, E. et al. (2012); Rusell, S., & Norvig, P. (2016)

- **Idea**: work backwards from the query q to prove q,
  - check if q is known already or
  - prove all premises of some rule concluding q



$$P \Rightarrow Q$$
$$L \land M \Rightarrow P$$
$$B \land L \Rightarrow M$$
$$A \land P \Rightarrow L$$
$$A \land B \Rightarrow L$$
$$A$$
$$B$$

Adapted from Castillo, E. et al. (2012); Beierle, C., & Kern-Isberner, G. (2014); Rusell, S., & Norvig, P. (2016)

$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

- Forward chaining is data-driven, automatic processing
  - May do lots of work that is irrelevant to the goal

- Backward chaining is goal-driven, appropriate for problem-solving
  - Complexity can be much less than linear in size of KB

# Your turn!

**Task**

Your colleague Sarah plans to buy a new sportcar, but she can not decide which one. You remember your AI course and the chapter about logic and try to help her. Model the following problem with propositional logic:

- She wants to buy a Porsche but struggeles between 911 ($N$), Panamera ($P$) and Taycan ($T$)

- She says that she likes shopping ($S$), and hence she doesn't want buy a two-door car like the 911 because it has not enough place for shopping

- If you want to buy a Taycan, you need an electric fueling station in your garage ($E$)

She now assumes ($H_1$):

- If you like shopping, and have no electric fueling station you have to buy a Panamera

Modell this problem with propositional logic. Can you "prove" her assumption $H_1$? Explain.

# Outline

| 5 | Knowledge Reasoning - Fundamental Algorithms and Concepts |
|---|---|

| 5.1 | Knowledge Reasoning |
|---|---|
| 5.2 | Propositional Logic |
| 5.3 | First Order Logic |
| 5.4 | Planning and Acting |
| 5.5 | Knowledge Representation and Engineering |

Lectorial 3: Building a Rule-based Agent for Credit Scoring

►**What we will learn:**
- We will design logical agents that can form representations of the world
- For that purpose we will formalize knowledge and reasoning processes with representation languages like propositional or first-order logic
- Use inference processes to derive new representations about the world, and use these new representations for decision-making and planning



Image source: ↗ Pixabay (2019) / ↗ CC0

►**Duration:**
- 270 min + 90 min (Lectorial)

►**Relevant for Exam:**
- 5.1 – 5.2, 5.4

- Suppose you want to say "All humans are mortal"
  - In propositional logic, you would need about 6.7 billion statements

- Suppose you want to say "Some people can run a marathon"
  - You would need a disjunction of ~6.7 billion statements

- No notion of objects
- No notion of relations among objects
- RoommateCarryingUmbrella0 is instructive to us, suggesting
  - there is an object we call Roommate,
  - there is an object we call Umbrella0,
  - there is a relationship Carrying between these two objects
- Formally, none of this meaning is there
  - Might as well have replaced RoommateCarryingUmbrella0 by P

Adapted from Rusell, S., & Norvig, P. (2016)

- Propositional logic assumes the world consists of atomic facts

- First-order logic assumes the world contains objects, relations, and functions

- Objects: can give these names such as Car, Nationalpark, USA, …

- Relations: driving(., .), parkHasBeenVisited(.)
    - Driving(Person, Car), hasBeenVisited(Nationalpark)
    - Relations with one object = unary relations = properties

- Functions: Nationalpark(.)   **X**
    - Nationalpark(Car)

- Equality: Roommate(Person_A) = Person_B

- **Constants:**      Max_MPH, Dominik, Rocky Mountains, pi
- **Variables:**      x, y, a, b
- **Predicates:**      Person(Dominik), Siblings(Bart, Lisa), isNationalpark(RockyMountains),
- **Functions:**      FatherOf(Lisa), Age(Bart), Sqrt(x)
- **Connectives:**      $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$
- **Equality:**      =
- **Quantifiers:**      $\forall, \exists$
- **Term:**      Constant or Variable or Function($Term_1, \dots, Term_n$)
- **Atomic sentence:**      Predicate($Term_1, \dots, Term_n$) or $Term_1 = Term_2$
- **Complex sentence:**      made from atomic sentences using connectives and quantifiers

- Sentences are true with respect to a model and an interpretation

- Model contains objects (domain elements) and relations among them

- Interpretation specifies referents for

  constant symbols  →  objects
  predicate symbols →  relations
  function symbols  →  functional relations


- An atomic sentence **Predicate(Term$_1$, … , Term$_n$)** is true iff the objects referred to by **Term$_1$, … , Term$_n$** are in the relation referred to by predicate

It could be that we have a separate name for Roommate(Person0)
E.g., Roommate(Person0) = Person1
… but we do not **need** to have such a name


A function can be applied to any object
E.g., Roommate(Umbrella0)

Adapted from Rusell, S., & Norvig, P. (2016)

Variables: x, y, z, … can refer to multiple objects

New operators "for all" and "there exists"
    Universal quantifier and existential quantifier

for all x: CompletelyWhite(x) => NOT(PartiallyBlack(x))
    Completely white objects are never partially black

there exists x: PartiallyWhite(x) AND PartiallyBlack(x)
    There exists some object in the world that is partially white and partially black

Adapted from Rusell, S., & Norvig, P. (2016)

"John has an umbrella"

there exists y: (Has(John, y) AND IsUmbrella(y))

"Anything that has an umbrella is not wet"

for all x: ((there exists y: (Has(x, y) AND IsUmbrella(y))) =>
NOT(IsWet(x)))

"Any person who has an umbrella is not wet"

for all x: (IsPerson(x) => ((there exists y: (Has(x, y) AND IsUmbrella(y)))
=> NOT(IsWet(x))))

Adapted from Rusell, S., & Norvig, P. (2016)

"John has at least two umbrellas"

there exists x: (there exists y: (Has(John, x) AND IsUmbrella(x) AND Has(John, y) AND IsUmbrella(y) AND NOT(x=y))

"John has at most two umbrellas"

for all x, y, z: ((Has(John, x) AND IsUmbrella(x) AND Has(John, y) AND IsUmbrella(y) AND Has(John, z) AND IsUmbrella(z)) => (x=y OR x=z OR y=z))

Adapted from Rusell, S., & Norvig, P. (2016)

"Duke's basketball team defeats any other basketball team"
for all x: ((IsBasketballTeam(x) AND NOT(x=BasketballTeamOf(Duke)))
=> Defeats(BasketballTeamOf(Duke), x))
"Every team defeats some other team"
for all x: (IsTeam(x) => (there exists y: (IsTeam(y) AND NOT(x=y) AND
Defeats(x,y))))

Adapted from Rusell, S., & Norvig, P. (2016)

"Property P implies property Q, or propery P implies property Q (or both)"
for all x: ((P(x) => Q(x)) OR (Q(x) => P(x)))
(for all x: (P(x) => Q(x)) OR (for all x: (Q(x) => P(x)))

- $\forall$**x P(x)**

- Example: "Everyone at UNC is smart"

    $\forall$**x At(x,UNC)** $\Rightarrow$ **Smart(x)**

    Why not $\forall$**x At(x,UNC)** $\wedge$ **Smart(x)**?

- Roughly speaking, equivalent to the conjunction of all possible instantiations of the variable:

    **At(John, UNC)** $\Rightarrow$ **Smart(John)** $\wedge$ **…**

    **At(Richard, UNC)** $\Rightarrow$ **Smart(Richard)** $\wedge$ **…**

- $\forall$**x P(x)** is true in a model m iff **P(x)** is true with **x** being each possible object in the model

- **∃x P(x)**

- Example: "Someone at UNC is smart"

  **∃x At(x,UNC) ∧ Smart(x)**

  Why not **∃x At(x,UNC) ⟹ Smart(x)**?

- Roughly speaking, equivalent to the disjunction of all possible instantiations:

  **[At(John,UNC) ∧ Smart(John)] ∨**

  **[At(Richard,UNC) ∧ Smart(Richard)] ∨ ...**

- **∃x P(x)** is true in a model m iff **P(x)** is true with **x** being some possible object in the model

- $\forall$**x** $\forall$**y** is the same as $\forall$**y** $\forall$**x**

- $\exists$**x** $\exists$**y** is the same as $\exists$**y** $\exists$**x**

- $\exists$**x** $\forall$**y** is not the same as $\forall$**y** $\exists$**x**
  - $\exists$**x** $\forall$**y Loves(x,y)**
    - "There is a person who loves everyone"
  - $\forall$**y** $\exists$**x Loves(x,y)**
    - "Everyone is loved by at least one person"

- **Quantifier duality:** each quantifier can be expressed using the other with the help of negation
  - $\forall$**x Likes(x,IceCream)** $\qquad$ $\neg\exists$**x** $\neg$**Likes(x,IceCream)**
  - $\exists$**x Likes(x,Broccoli)** $\neg\forall$**x** $\neg$**Likes(x,Broccoli)**

- **Term$_1$ = Term$_2$** is true under a given model if and only if **Term$_1$** and **Term$_2$** refer to the same object

- E.g., definition of **Sibling** in terms of **Parent**:
  $\forall$**x,y Sibling(x,y)** $\Leftrightarrow$
  **[$\neg$(x = y) $\wedge$ $\exists$m,f $\neg$ (m = f) $\wedge$ Parent(m,x) $\wedge$ Parent(f,x) $\wedge$ Parent(m,y) $\wedge$ Parent(f,y)]**

- Brothers are siblings

  $\forall$**x,y Brother(x,y)** $\Rightarrow$ **Sibling(x,y)**

- "Sibling" is symmetric

  $\forall$**x,y Sibling(x,y)** $\Leftrightarrow$ **Sibling(y,x)**

- One's mother is one's female parent

  $\forall$**m,c (Mother(c) = m)** $\Leftrightarrow$ **(Female(m)** $\wedge$ **Parent(m,c))**

- FOL permits quantification over variables
- Higher order logics permit quantification over functions and predicates:

$$\forall P,x \; [P(x) \lor \neg P(x)]$$

$$\forall x,y \; (x=y) \Leftrightarrow [\forall P \; (P(x) \Leftrightarrow P(y))]$$

- All rules of inference for propositional logic apply to first-order logic
- We just need to reduce FOL sentences to PL sentences by instantiating variables and removing quantifiers

for all x: a
is equivalent to
NOT(there exists x: NOT(a))

Adapted from Rusell, S., & Norvig, P. (2016)

We are **not** allowed to reason in general about relations and functions
The following would correspond to higher-order logic (which is more powerful):
"If John is Jack's roommate, then any property of John is also a property of Jack's roommate"
(John=Roommate(Jack)) => for all p: (p(John) => p(Roommate(Jack)))
"If a property is inherited by children, then for any thing, if that property is true of it, it must also be true for any child of it"
for all p: (IsInheritedByChildren(p) => (for all x, y: ((IsChildOf(x,y) AND p(y)) => p(x))))

Adapted from Rusell, S., & Norvig, P. (2016)

**D**

Axioms: basic facts about the domain, our "initial" knowledge base

Theorems: statements that are logically derived from axioms

SUBST replaces one or more variables with something else

For example:

SUBST({x/John}, IsHealthy(x) => NOT(HasACold(x))) gives us

IsHealthy(John) => NOT(HasACold(John))

- **Substitution** of variables by *ground terms*:

**SUBST({v/g},P)**

- Result of SUBST({x/Harry, y/Sally}, Loves(x,y)):
  Loves(Harry,Sally)

- Result of SUBST({x/John}, King(x) $\wedge$ Greedy(x) $\Rightarrow$ Evil(x)):
  King(John) $\wedge$ Greedy(John) $\Rightarrow$ Evil(John)

- A universally quantified sentence entails every instantiation of it:

$$\frac{\forall v \; P(v)}{SUBST(\{v/g\}, P(v))}$$

  for any variable v and ground term g

- E.g., $\forall x$ King(x) $\wedge$ Greedy(x) $\Rightarrow$ Evil(x) yields:
  King(John) $\wedge$ Greedy(John) $\Rightarrow$ Evil(John)
  King(Richard) $\wedge$ Greedy(Richard) $\Rightarrow$ Evil(Richard)
  King(Father(John)) $\wedge$ Greedy(Father(John)) $\Rightarrow$ Evil(Father(John))

- An existentially quantified sentence entails the instantiation of that sentence with a new constant:

$$\frac{\exists v\; P(v)}{SUBST(\{v/C\}, P(v))}$$

- for any sentence P, variable v, and constant C that does not appear elsewhere in the knowledge base

- E.g., $\exists x\; Crown(x) \wedge OnHead(x, John)$ yields:

$$Crown(C_1) \wedge OnHead(C_1, John)$$

- provided $C_1$ is a new constant symbol, called a *Skolem constant*

- Every FOL KB can be *propositionalized* so as to preserve entailment
  - A ground sentence is entailed by the new KB iff it is entailed by the original KB

- **Idea:** propositionalize KB and query, apply resolution, return result

- **Problem:** with function symbols, there are infinitely many ground terms
  - For example, Father(X) yields Father(John), Father(Father(John)), Father(Father(Father(John))), etc.

- **Theorem** (Herbrand 1930):
  - If a sentence α is entailed by an FOL KB, it is entailed by a *finite* subset of the propositionalized KB

- **Idea:** For *n* = 0 to Infinity do
  - Create a propositional KB by instantiating with depth-n terms
  - See if α is entailed by this KB

- **Problem:** works if α is entailed, loops if α is not entailed

- **Theorem** (Turing 1936, Church 1936):
  - Entailment for FOL is **semidecidable**: algorithms exist that say yes to every entailed sentence, but no algorithm exists that also says no to every nonentailed sentence

From
for all x: a
we can obtain
SUBST({x/g}, a)


From
there exists x: a
we can obtain
SUBST({x/k}, a)
where k is a constant that does not appear elsewhere in the knowledge base (Skolem constant)
Don't need original sentence anymore

Adapted from Rusell, S., & Norvig, P. (2016)

for all x: there exists y: IsParentOf(y,x)

WRONG: for all x: IsParentOf(k, x)

RIGHT: for all x: IsParentOf(k(x), x)

Introduces a new function (Skolem function)

… again, assuming k has not been used previously

Adapted from Rusell, S., & Norvig, P. (2016)

for all x: Loves(John, x)

    John loves every thing

for all y: (Loves(y, Jane) => FeelsAppreciatedBy(Jane, y))

    Jane feels appreciated by every thing that loves her

Can infer from this:

FeelsAppreciatedBy(Jane, John)


Here, we used the substitution {x/Jane, y/John}

    Note we used different variables for the different sentences

General UNIFY algorithms for finding a good substitution

Adapted from Rusell, S., & Norvig, P. (2016)

Consider EdibleByWith

 e.g., EdibleByWith(Soup, John, Spoon) – John can eat soup with a spoon

for all x: for all y: EdibleByWith(Bread, x, y)

 Anything can eat bread with anything

for all u: for all v: (EdibleByWith(u, v, Spoon) => CanBeServedInBowlTo(u,v))

 Anything that is edible with a spoon by something can be served in a bowl to that something

Substitution: {x/z, y/Spoon, u/Bread, v/z}

Gives: for all z: CanBeServedInBowlTo(Bread, z)

Alternative substitution {x/John, y/Spoon, u/Bread, v/John} would only have given

CanBeServedInBowlTo(Bread, John), which is not as general

Adapted from Rusell, S., & Norvig, P. (2016)

for all x: (NOT(Knows(John, x)) OR IsMean(x) OR Loves(John, x))

John loves everything he knows, with the possible exception of mean things

for all y: (Loves(Jane, y) OR Knows(y, Jane))

Jane loves everything that does not know her

What can we unify?  What can we conclude?

Use the substitution: {x/Jane, y/John}

Get: IsMean(Jane) OR Loves(John, Jane) OR Loves(Jane, John)

Complete (i.e., if not satisfiable, will find a proof of this), **if** we can remove

literals that are duplicates after unification

Also need to put everything in canonical form first

Adapted from Rusell, S., & Norvig, P. (2016)

Deciding whether a sentence is entailed is semidecidable: there are algorithms that will eventually produce a proof of any entailed sentence

It is not decidable: we cannot always conclude that a sentence is not entailed

Adapted from Rusell, S., & Norvig, P. (2016)

First-order logic is not rich enough to model basic arithmetic
For any consistent system of axioms that is rich enough to capture basic arithmetic (in particular, mathematical induction), there exist true sentences that cannot be proved from those axioms

Adapted from Rusell, S., & Norvig, P. (2016)

## Suppose:

There are exactly 3 objects in the world,

If x is the spouse of y, then y is the spouse of x (spouse is a function, i.e., everything has a spouse)

## Prove:

Something is its own spouse

- **Propositional logic:** atomic statements are facts
  - Inference via resolution is sound and complete (though likely computationally intractable)

- **First-order logic:** adds variables, relations, and quantification
  - Inference is essentially a generalization of propositional inference
  - Resolution is still sound and complete, but not guaranteed to terminate on non-entailed sentences (semidecidable)
  - Simple inference procedures (forward chaining and backward chaining) available for knowledge bases consisting of *definite clauses*

# Your turn!

**Task**

Please prove:

- there exist x, y, z: (NOT(x=y) AND NOT(x=z) AND NOT (y=z))

- for all w, x, y, z: (w=x OR w=y OR w=z OR x=y OR x=z OR y=z)

- for all x, y: ((Spouse(x)=y) => (Spouse(y)=x))

- for all x, y: ((Spouse(x)=y) => NOT(x=y)) (for the sake of contradiction)

- Try to do this on the board…

# Outline

| 5 | Knowledge Reasoning - Fundamental Algorithms and Concepts |
|---|---|

| 5.1 | Knowledge Reasoning |
|---|---|
| 5.2 | Propositional Logic |
| 5.3 | First Order Logic |
| 5.4 | Planning and Acting |
| 5.5 | Knowledge Representation and Engineering |

Lectorial 3: Building a Rule-based Agent for Credit Scoring

►**What we will learn:**
- We will design logical agents that can form representations of the world
- For that purpose we will formalize knowledge and reasoning processes with representation languages like propositional or first-order logic
- Use inference processes to derive new representations about the world, and use these new representations for decision-making and planning



Image source: ↗ Pixabay (2019) / ↗ CC0

►**Duration:**
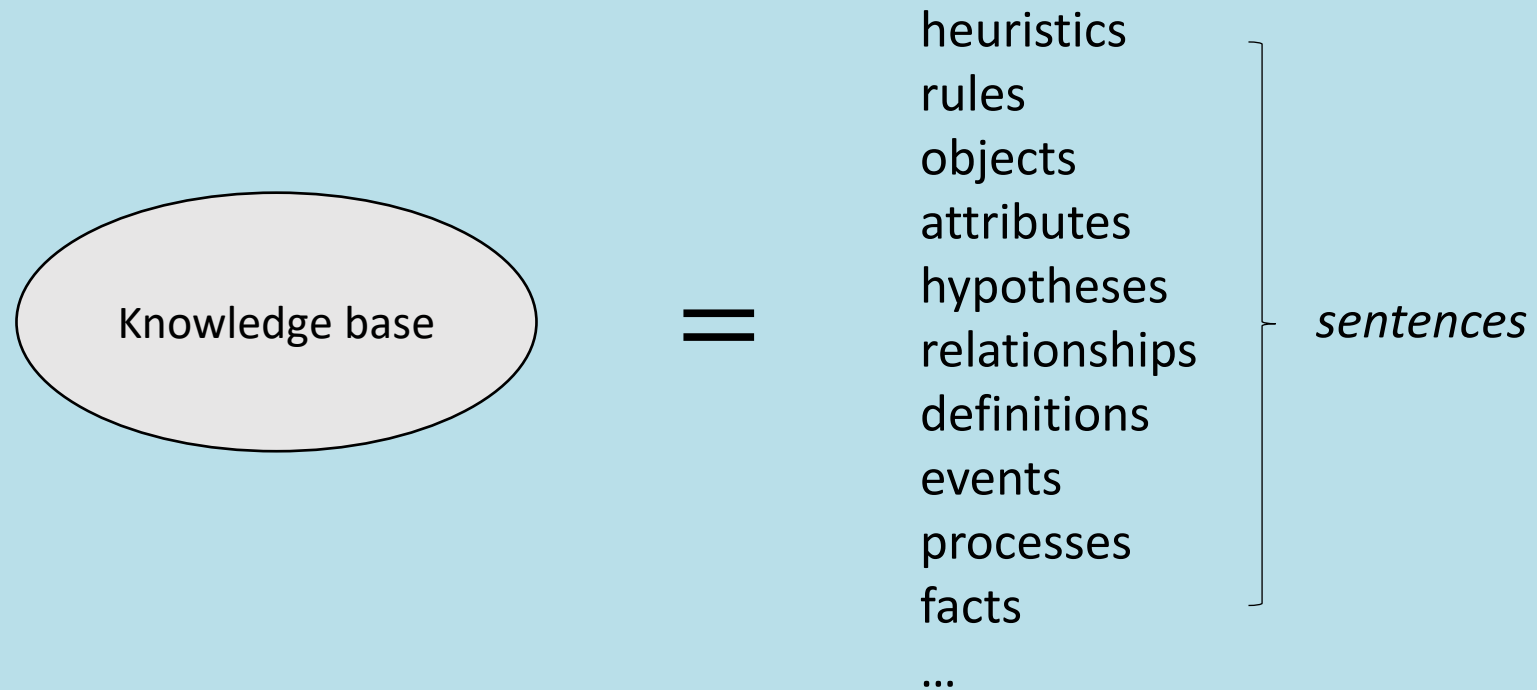- 270 min + 90 min (Lectorial)

►**Relevant for Exam:**
- 5.1 – 5.2, 5.4

Knowledge base **=**

heuristics
rules
objects
attributes
hypotheses
relationships
definitions
events
processes
facts
…

*sentences*

Adapted from Rusell, S., & Norvig, P. (2016)

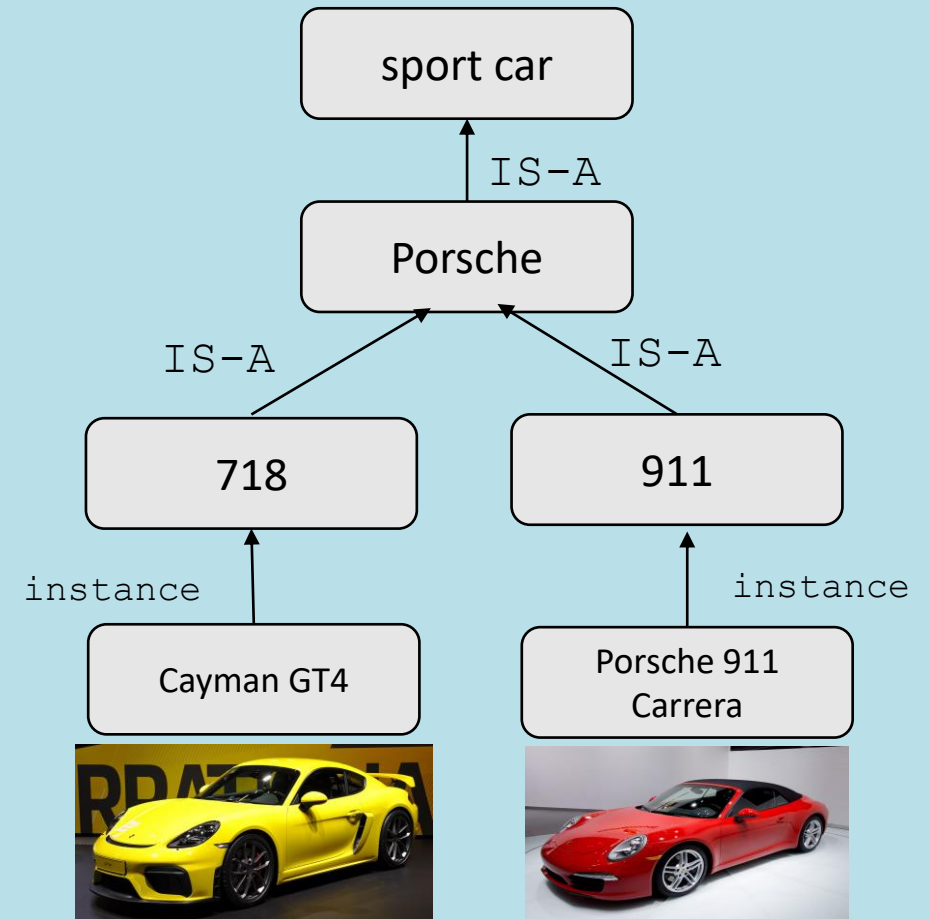Adapted from Rusell, S., & Norvig, P. (2016)

# 5.5 Simple Relational Knowledge

- It is the simplest way of storing facts which uses the relational method, and each fact about a set of the object is set out systematically in columns.

- This approach of knowledge representation is famous in database systems where the relationship between different entities is represented

- This approach has little opportunity for inference.

- **Example: The following is the simple relational knowledge representation.**

| Player | Weight | Age |
|--------|--------|-----|
| Player1 | 65 | 23 |
| Player2 | 58 | 18 |
| Player3 | 75 | 24 |

- All data must be stored into a hierarchy of classes

- All classes should be arranged in a generalized form or a hierarchal manner.

- Elements inherit values from other members of a class.

- This approach contains inheritable knowledge which shows a relation between instance and class, and it is called instance relation.

- Procedural knowledge approach uses small programs and codes which describes how to do specific things, and how to proceed

- In this approach, one important rule is used which is If-Then rule

- In this knowledge, we can use various coding languages such as LISP language and Prolog language

- We can easily represent heuristic or domain-specific knowledge using this approach

- But it is not necessary that we can represent all cases in this approach.

- A good knowledge representation system must possess the following properties:
  - **Representational Accuracy**: KR system should have the ability to represent all kind of required knowledge

  - **Inferential Adequacy**: KR system should have ability to manipulate the representational structures to produce new knowledge corresponding to existing structure

  - **Inferential Efficiency**: The ability to direct the inferential knowledge mechanism into the most productive directions by storing appropriate guides

  - **Acquisitional efficiency**: The ability to acquire the new knowledge easily using automatic methods.
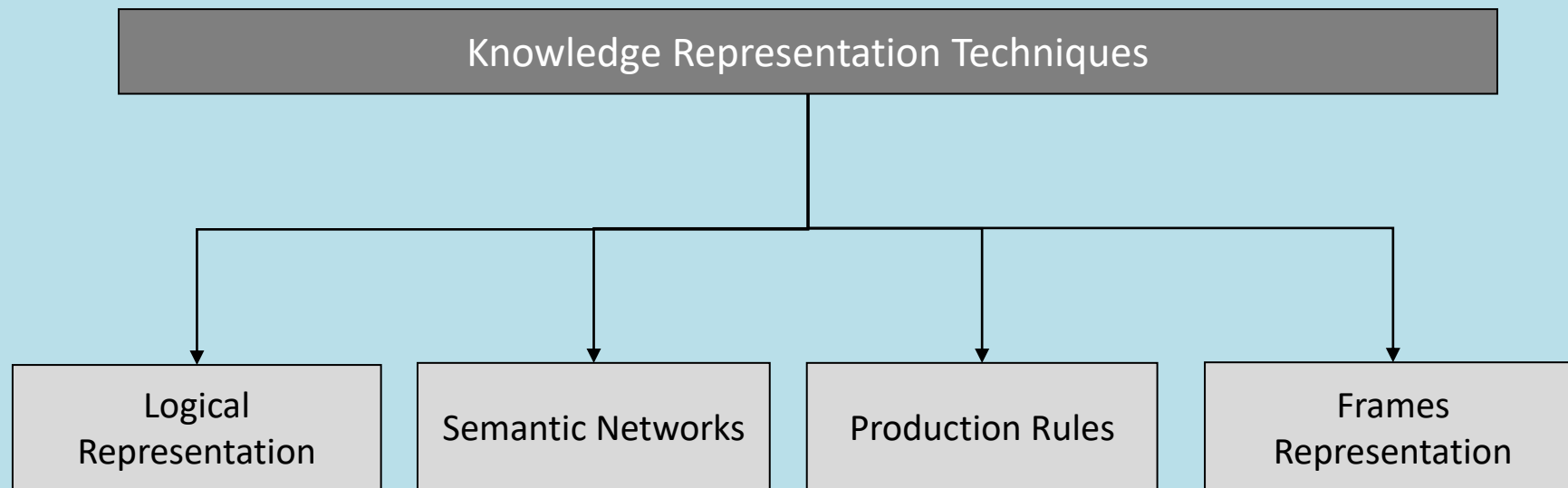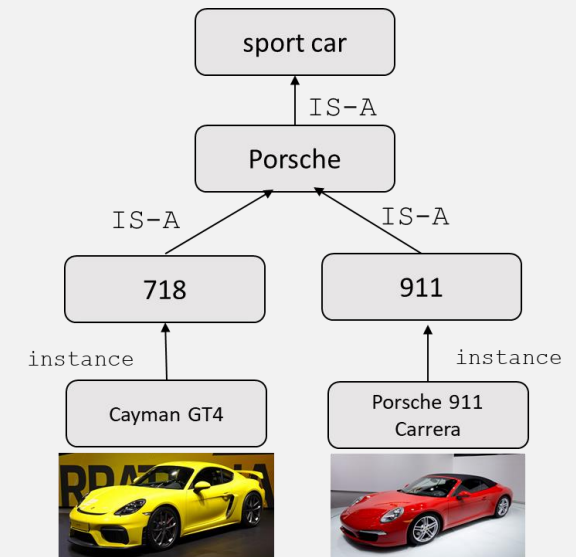
Adapted from Rusell, S., & Norvig, P. (2016)

# Your turn!

**Task**

Please expand the inheritable knowledge base by the following sentences:

- Porsche Panamera
- Tesla, Tesla Model X
- VW Golf

# As Long as There are Users of outdated Techniques, There is a Need of Specialist



Welt Online (2020)

# Exercises

**Workbook Exercises**

- Please read the chapters 7-12 from Rusell, S., & Norvig, P. (2016) to understand the concepts and algorithms of knowledge reasoning and representation. Then work through the related exercises. Focus on the contents we discussed in lecture.

- Read and understand the Wumpus World Problem from chapter 7. Try to model it with the concepts from this lecture.

**Coding Exercises**

- Try to build a simple rule-based agent with Python by implementing the decision rules a simple if-else checks. What is the problem of this approach if you think at expanding the knowledge based? Make notes and prepare for the lectorial of this chapter, we will find a more convinient way to process and represent knowledge in Python

**Literature**

1. Beierle, C., & Kern-Isberner, G. (2014). Methoden wissensbasierter Systeme: Grundlagen, Algorithmen, Anwendungen. Springer-Verlag.

2. Brachman, R. J., Levesque, H. J., & Reiter, R. (Eds.) (1992): *Knowledge representation*. MIT press.

3. Castillo, E., Gutierrez, J. M., & Hadi, A. S. (2012). Expert systems and probabilistic network models. Springer Science & Business Media.

4. Peirce, C. S. (1931): Collected papers of charles sanders peirce. Harvard University Press, 1931.

5. Rusell, S., & Norvig, P. (2016). *Artificial Intelligence: A Modern Approach.* Global Edition.
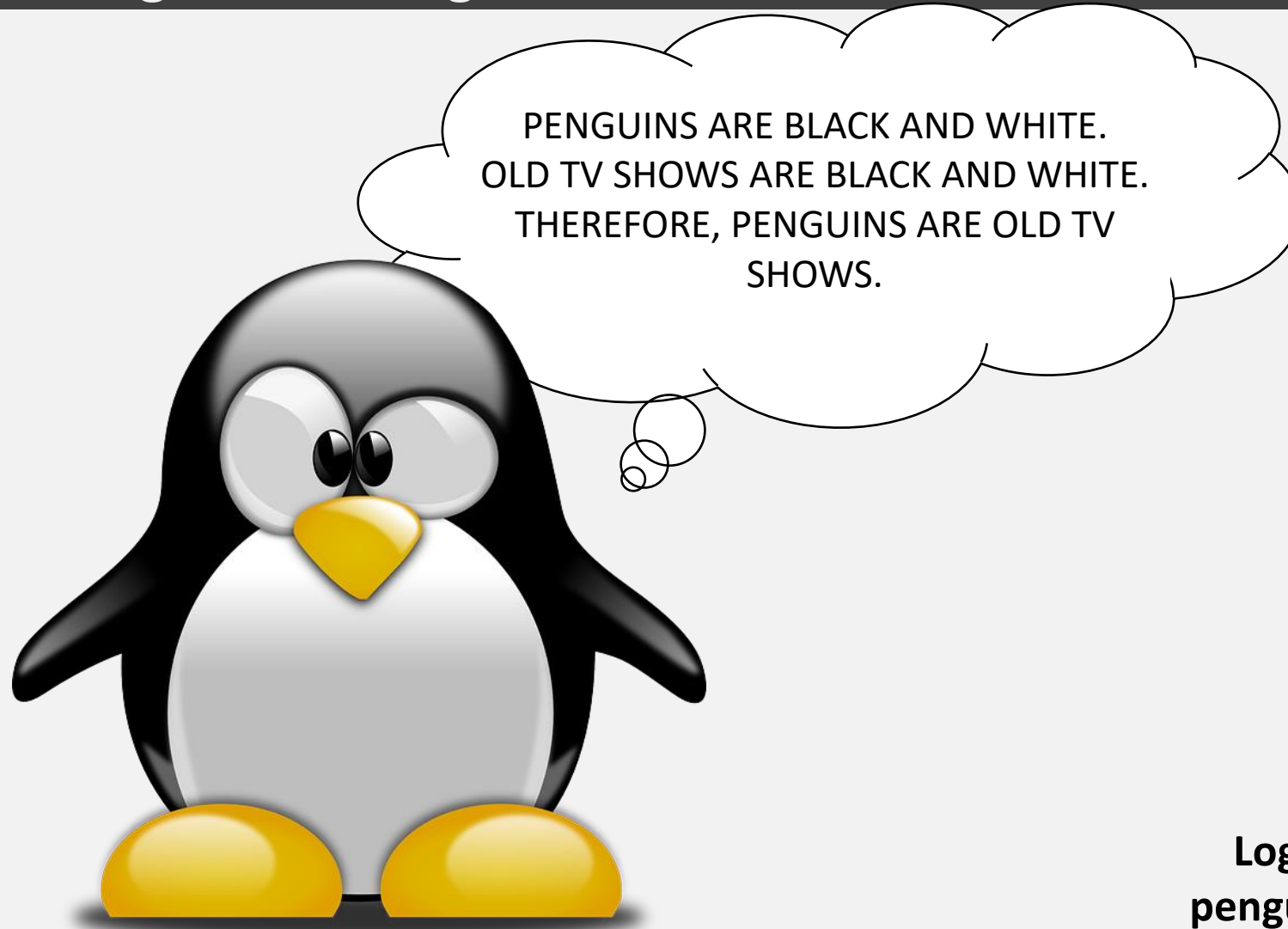
**Images**

*All images that were not marked other ways are made by myself, or licensed ↗CC0 from ↗Pixabay.*

# References

**Further reading**

- Even if knowledge-based systems are a bit outdated, there exists still many frameworks to developp such systems with state of the art technology like Python, Java etc. If you are interested I recommend to take a look at the following frameworks and libraries:
    - C Language Integrated Production System (CLIPS): An environment used to make rule or object based expert systems developed by the NASA (↗www.clipsrules.net )
    - Python Knowledge Engine Pyke (Pyke), which allows you to use logic programming to make expert systems in Python (↗www.pyke.sourceforge.net )
    - D3web, Java Knowledge Base System that uses XML (↗www.d3web.de )

- The popular game Age of Empires II from Microsoft Game Studios relies on CLIPS for its computer AI. I can recommend you to take a look at the guide from redmechanic (on ↗ Steam), from Leif Ericson(↗ aok.heavengames.com), or a copy of Microsoft's „Computer Player Strategy Builder Guide" from my git hub repository (Media/CPSB.zip) to implement your own knowledge-based AI. Play against it and try to win against yourself ;-)

# Glossary

| | |
|---|---|
| **Entailment** | *Necessary truth of one sentence given another* |
| **Inference** | *Deriving sentences from other sentences* |
| **Logical Agents** | *Logical agents apply inferenceto a knowledge baseto derive new information and make decisions* |
| **Knowledge Base** | *The component of an expert system that constains the systems's knowledge organized in collection of facts about the system's domain.* |
| **Knowledge Representation** | *The study of how "what we know" can at the same time be represented as comprehensibly as possible and reasoned with as effectively as possibly from an information system* |
| **Soundness** | *Derivations produce only entailed sentences* |
| **Completeness** | *Derivations can produce all entailed sentences* |