

Based on “Symbolic AI” by

Prof. Dr. Andreas Dengel

Einführung in die KI,

Teil 2 Suchen

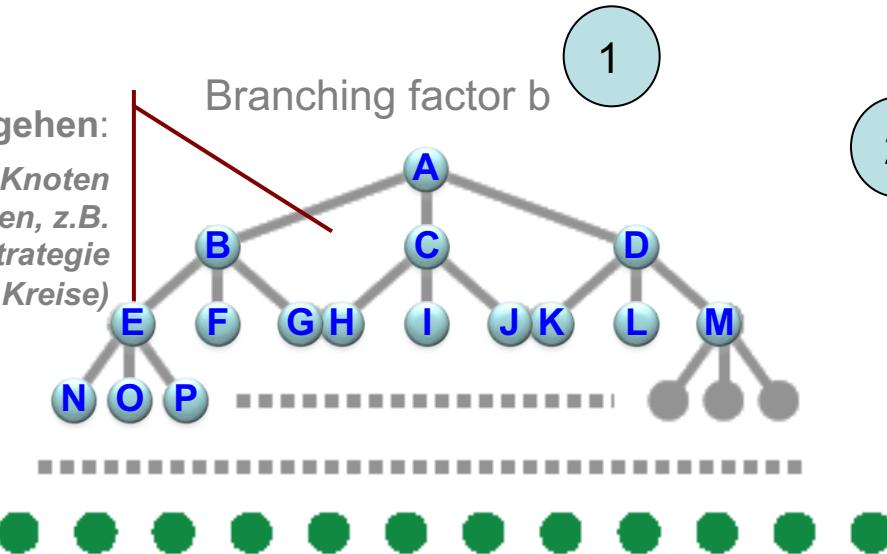
Prof. Dr. Paul Lukowicz

Erst Baum mit d und b, dann Bemerkungen, dann Beispiel

4

Durchgehen:

Wie sich Knoten
gemerkt werden, z.B.
irgendeine Strategie
(siehe farbige Kreise)



3

Given a goal in the search tree, we have to explore alternative **candidates** for search space exploration

Note:

candidates have to be kept in a data structure to remember them if a search fails

1

2

Depth

$d=0$

$d=1$

$d=2$

$d=3$

$d=m$

Maximal depth
of the tree

5

$$\leq \sum_{i=0}^d b^i = \frac{b^{d+1} - 1}{b - 1}$$

6

There are several well known algorithms for search space exploration some of which we will consider

Types of Strategy

- Uninformed (or blind) search strategies

The number of steps / the path costs from the start to the current node are unknown

It only can be decided whether or not the goal has been reached

- Informed (or heuristic) search strategies

Problem specific knowledge is used to make the search more efficient

- Depth-first-search
- Breadth-first-search
- Generate-and-test
- Hill climbing
- Best-first-search
- A*
- AO*
- Problem-reduction
- Branch and bound
- Dynamic programming
- Constraint satisfaction
-

Nach erstem Punkt auf Vorfolie

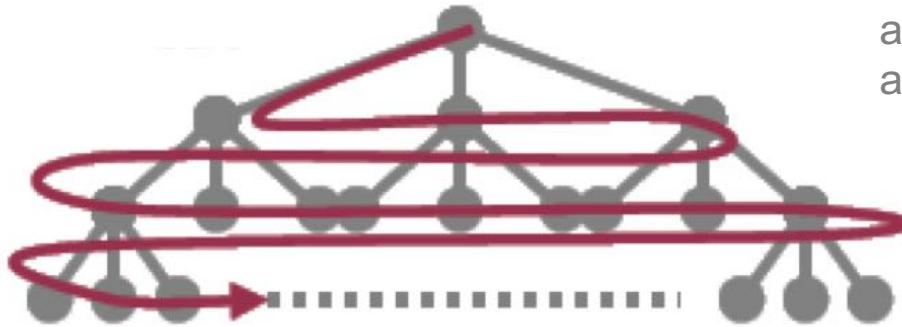
5 to build
list of candidates?

Breadth-first-search

1

2

If we knew that action costs
are constant and goal states
are on the final level



3

- Time requirement: $O(b^d)$ (Exponent d when action costs are constant!)
- Storage requirement: $O(b^d)$ (Exponent d when action costs are constant!)
- + Complete: If a solution exists, it will be found
- + Optimal with constant Aktionskosten

Note:

In some problems we have an infinite
branching factor, than BFS is incomplete

4

Nach erstem Punkt auf Vorfolie

2

How to build
list of candidates?

Depth-first-search



Vor 3 : Erst Algorithmus
modifizieren!!!

3

- Time requirement: $O(b^m)$, if $m = \text{maximal depth of the tree}$
- + Storage requirement: $O(bm)$, for paths already examined and evaluated
- Incomplete
- Not optimal

Note:

If no test for cycles is made, the DFS is incomplete because of infinite paths
(then it could happen that a solution is not found even if it exists)

5

4

Note:

This is required because all
successor nodes are stored in
a stack for backtracking

However, attention for infinite
paths!!!

There are further variations of Depth-First-Search

■ Limited Depth-First-Search

Define a global depth limit l , ...

Receives the qualitative characteristics of Depth-First-Search

Storage: $o(bl)$

Time: $o(b^l)$

Does not find a solution in depths $d > l$

Terminates in case of a finite branching factor

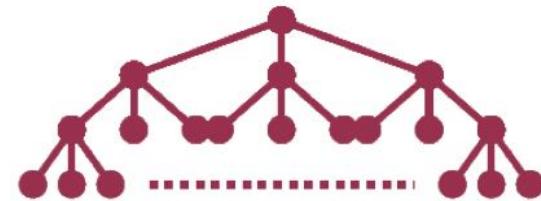
■ Iterative deepening search

Conduct depth-first-search until depth 1

... until depth 2

... until depth 3

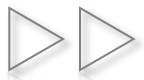
... etc., until problem is solved



- 😊 Complete
 - 😊 Optimal if constant action costs
 - 😊 Storage requirements: $o(bd)$
- ... but that's a waste of time, isn't it?

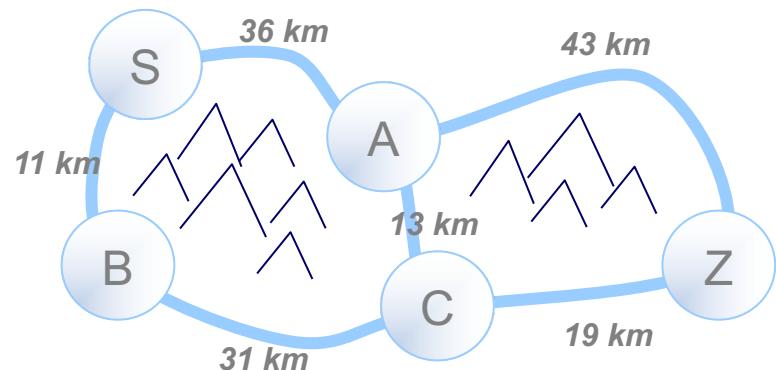
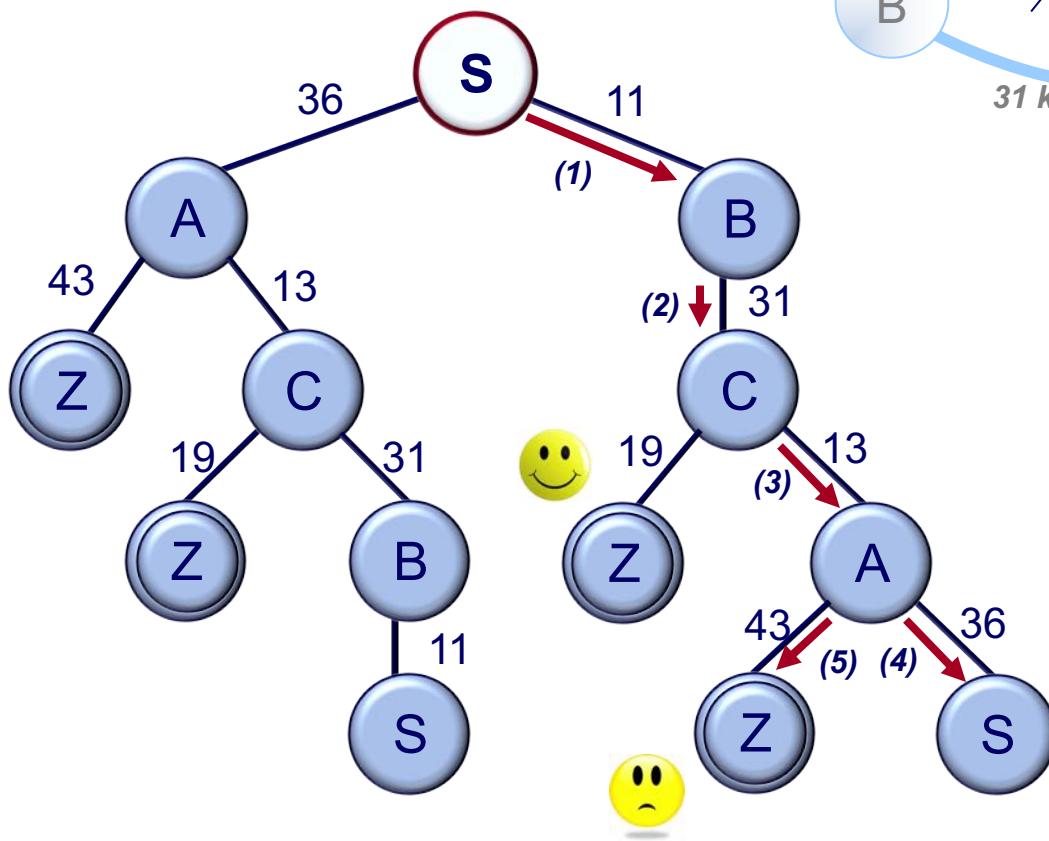
... to sum-up!

Criteria	BFS	DFS	Limited DFS	Iterative Deepening DFS
Criteria Time	b^d	b^m	b^l	b^d
Memory	b^d	bm	bl	bd
Optimal?	YES	NO	NO	YES
Complete?	YES	NO	YES, if $l \geq d$	YES



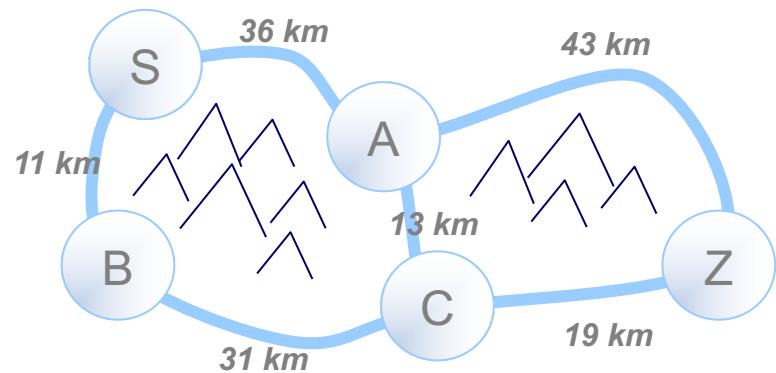
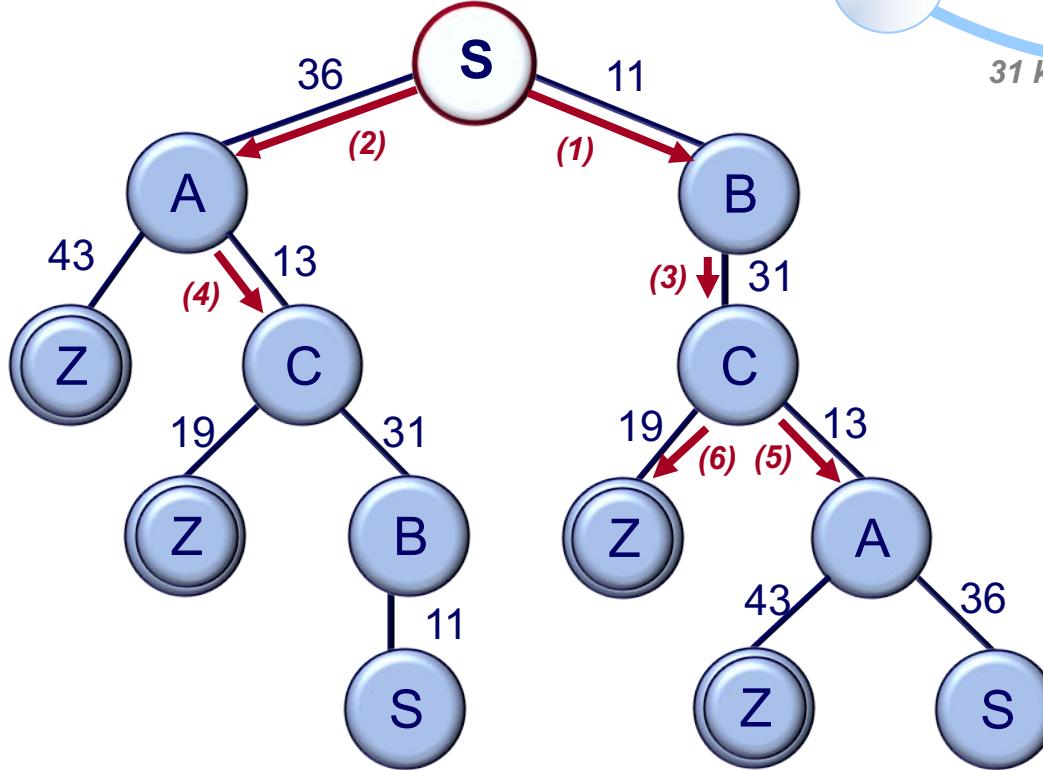
Steepest-Ascent-Hill-Climbing:

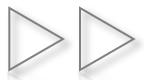
1. at every node take the „cheapest alternative
2. backtrack into solution found



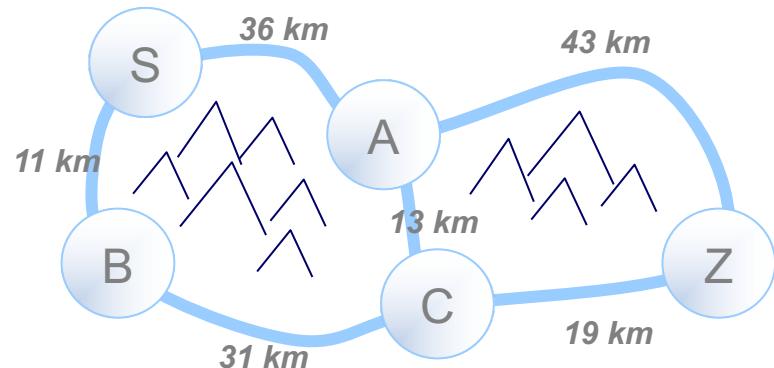
So, how does Best-First Search behave for our example

- Best-First-Search:

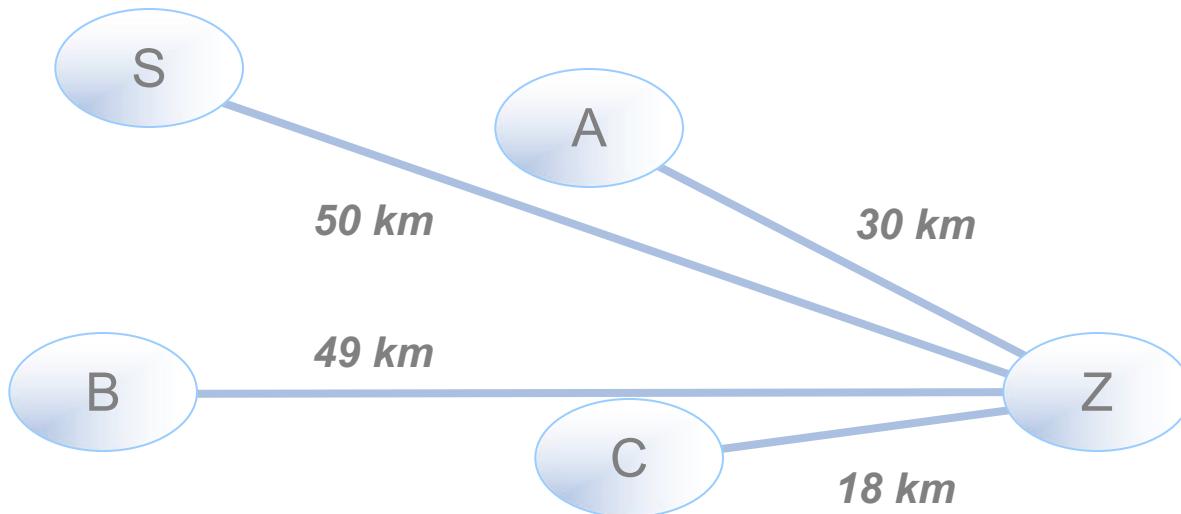




How about using some sort of how could the entire path could be ?

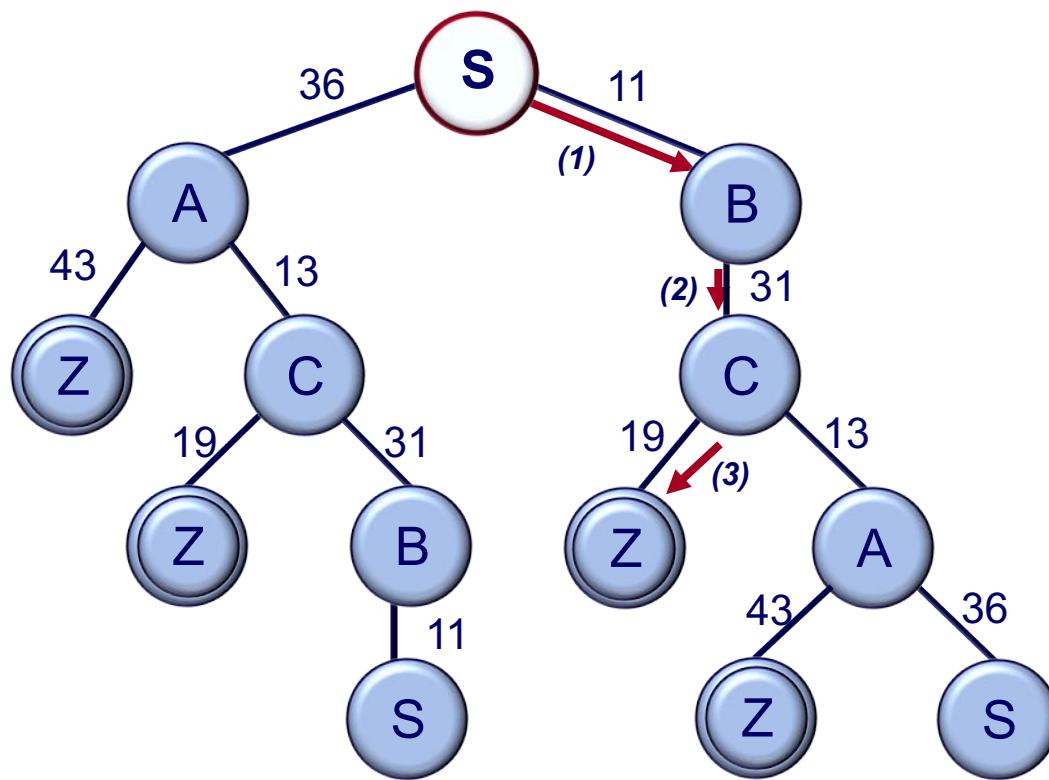
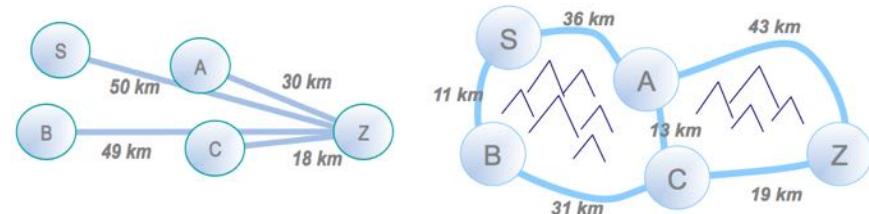


- Values of the estimating function $h(k) = \text{Euclidean distance}(k, Z)$



The A*-Algorithm (see e.g. Wikipedia) makes use of a strict monotone estimating function

■ A*-Algorithm:



Step 1: $\text{cost}(A) = 66$
 $\text{cost}(B) = 60$

Step 2: $\text{cost}(C) = 60$

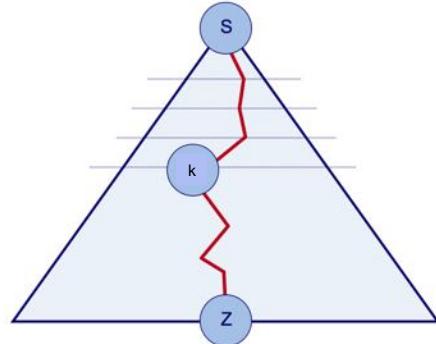
Step 3: $\text{cost}(Z) = 61$
 $\text{cost}(A) = 85$

Length of path: 61 km

If we make use of an admissible estimating function, the search is becoming optimal

- Since we can state the actual costs from one node k to the final node Z with $g(Z) - g(k)$ we can formulate the following condition for any estimating function:
- Definition: **Admissibility of an estimating function**

An estimating function h is admissible, if it never overestimates the costs it takes to get from one node to the final node, i.e. if it holds for any node k and for any final node Z which can be reached from k :



$$h(k) \leq g(Z) - g(k)$$

Note:
Estimated distance
from k to Z

A game tree may be used for showing the game positions for many two player games

Basic game characteristics are:

- Two players
- It is a sequential game - players take turns moving
- The game state can be represented by a **board** containing all the pertinent information
- Both players have knowledge of all possible next moves
(this is called a **perfect knowledge** game)

This excludes card games with hidden hands, for example, since we do not know all of our opponent's play choices

- Game moves are not random
 - Usually excluding games involving dice or spinning dials
- The games are finite, i.e. they all reach some terminal configuration representing a win, loss or draw result

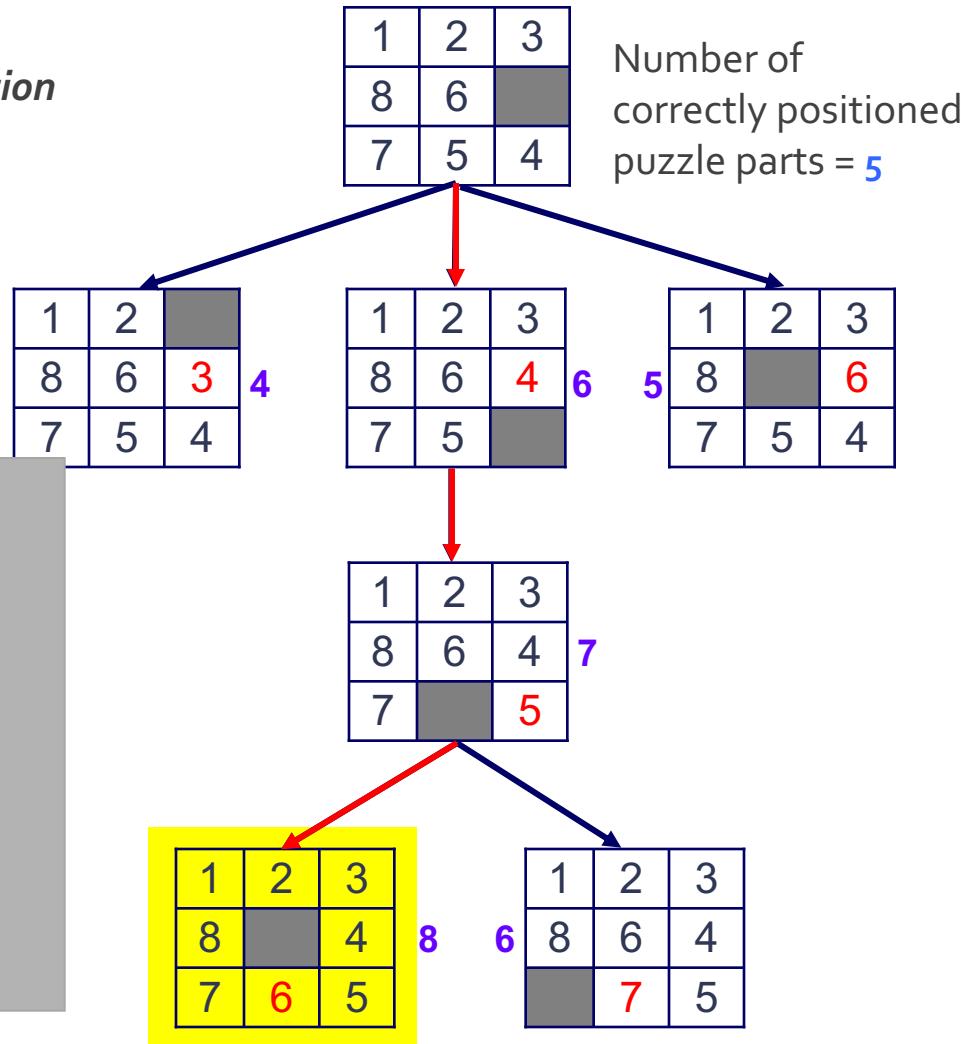
A better strategy would be including knowledge into the search process

- Define **estimating (or weighting) function**
here: Number of correct puzzles
- Follow the path with the highest improvement

The chess game:

The estimating function is based on the following parameters:

- Number of pieces left on the board
- weighted according to their function
- The number of pieces under attack
- Secured pieces
- Strategic (mid field) constellations



Chess is a two-person game with fair conditions

Computer evaluates
game situation

From its own perspective ...

... with function

B

e.g.

Piece	Score	In case of threat
King	1000	10
Castle	5	2,5
Knight	3	1,5
Pawn	1	0,5

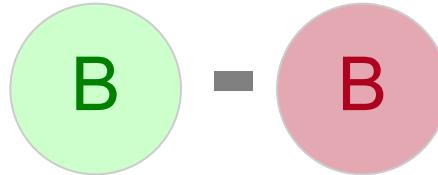


... and the one of the opponent

.... with function

B

Total evaluation of a game
situation is: :



- One player is trying to maximize while the other tries to minimize the evaluation value!

How to apply search in find in Two-Person-Games

Games like Chess, Checkers, Go oder Tic-Tac-Toe are two-person games called zero sum games

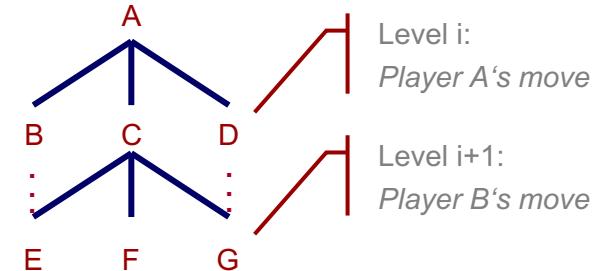
Principle of two-person games:

- The victory of one player causes the defeat of the other. This leads to a sum whose over-all advantage is 0
- The opponents alternate moving the pieces, i.e. chances are evenly distributed
- State spaces are characterized by **controversial** options to move

Player A is playing to win or for a draw

Player B is playing to win or for a draw, respectively

When a final node is reached, either player A or player B wins or the game ends in a draw



- In addition to looking for his advantage, player A simultaneously tries to prevent player B from making moves that are to his disadvantage

Search techniques

■ Min-Max-Principle:

If player A (or player MAX) is to make a move she/he tries to find a situation with a maximal weight

If player B (or player MIN) is to make a move she/he tries to find a situation with a minimal weight.

■ Example:

Tic-Tac-Toe would have the following weighting function holds:

$$100a + 10b + c - (100d + 10e + f)$$

X	X	O
	O	
X		

*a, b and c represent rows; columns and diagonals with
3, 2 and 1 „X“ – player MAX*

*d, e and f represent rows; columns and diagonals with
3, 2 und 1 „O“ – player MIN*

Example



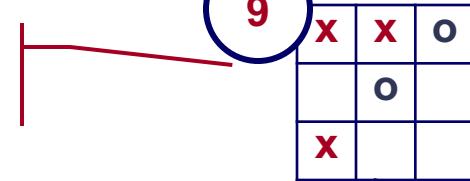
Example: Tic-Tac-Toe

MAX: x

MIN: o

Player MIN
to move

9



: 11 := 1*10 + 1*1

: 2

: 11

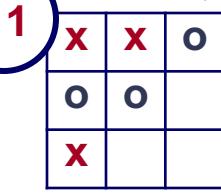
: 2

: 11

: 2

Player MAX
To move

-1

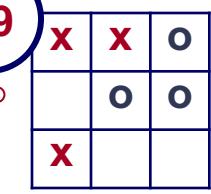


: 11

: 2

: 11

-9



: 11

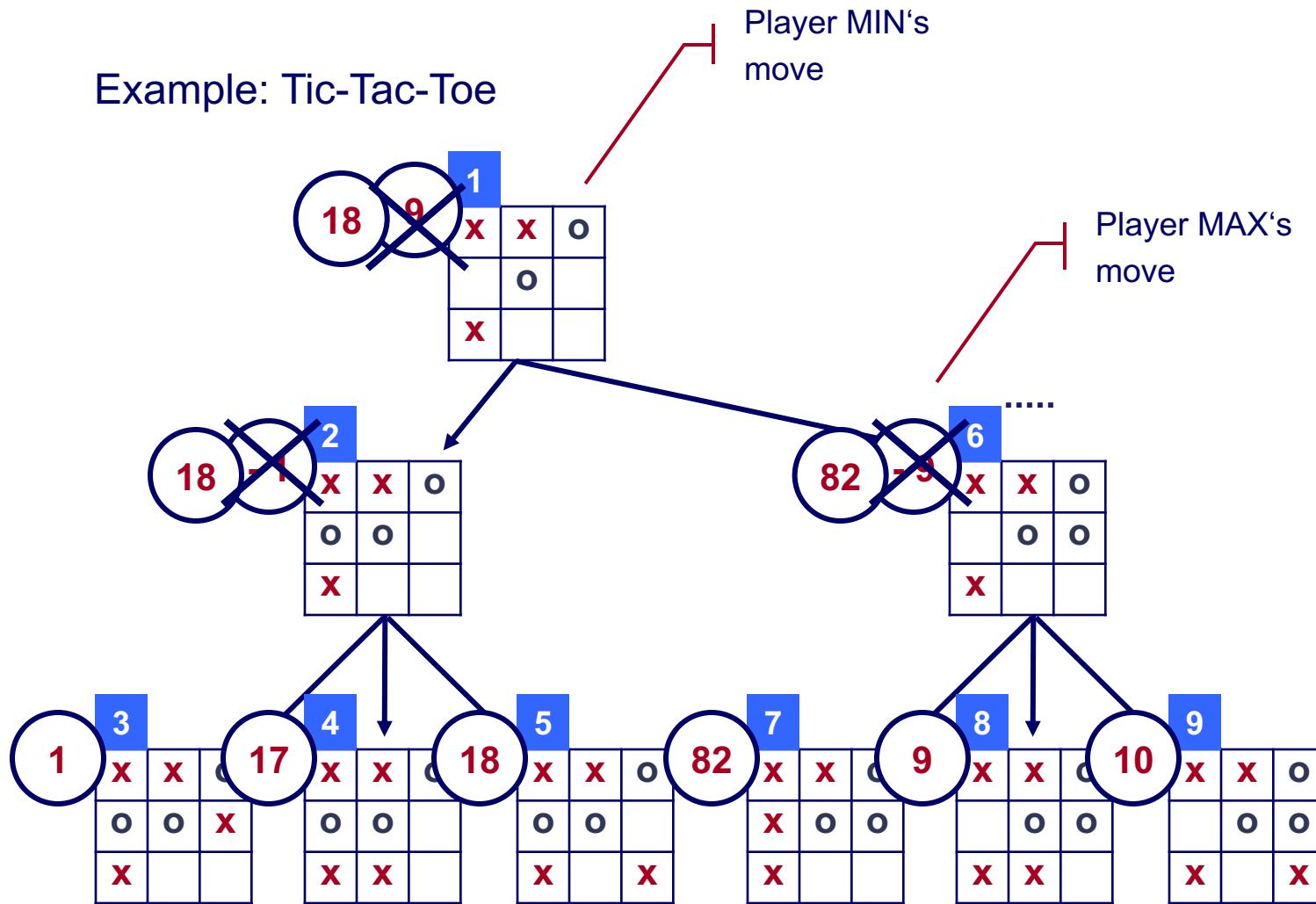
: 11

: 3

Continue example



Example: Tic-Tac-Toe



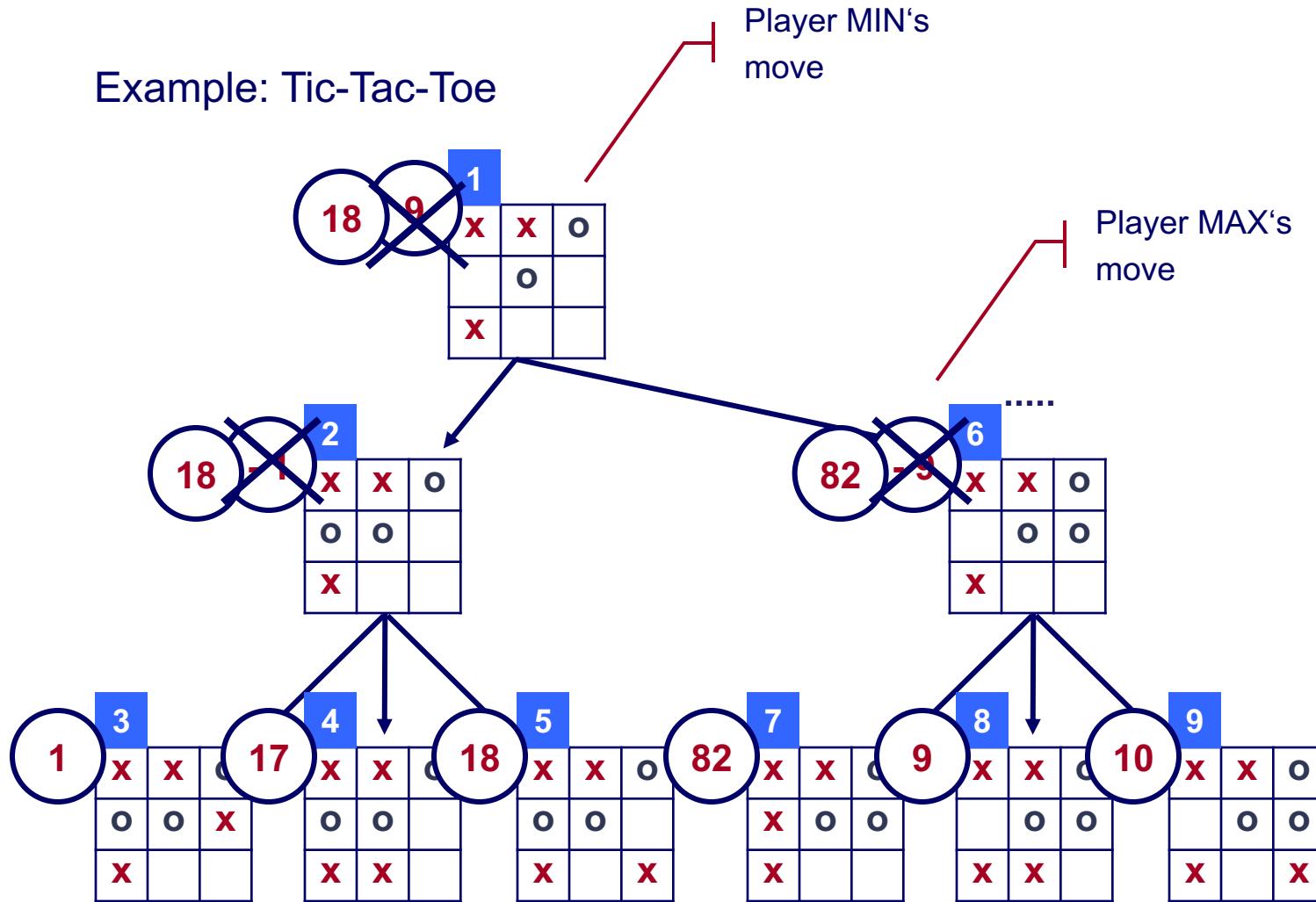


Could we further restrict the search space exploration

Continue example



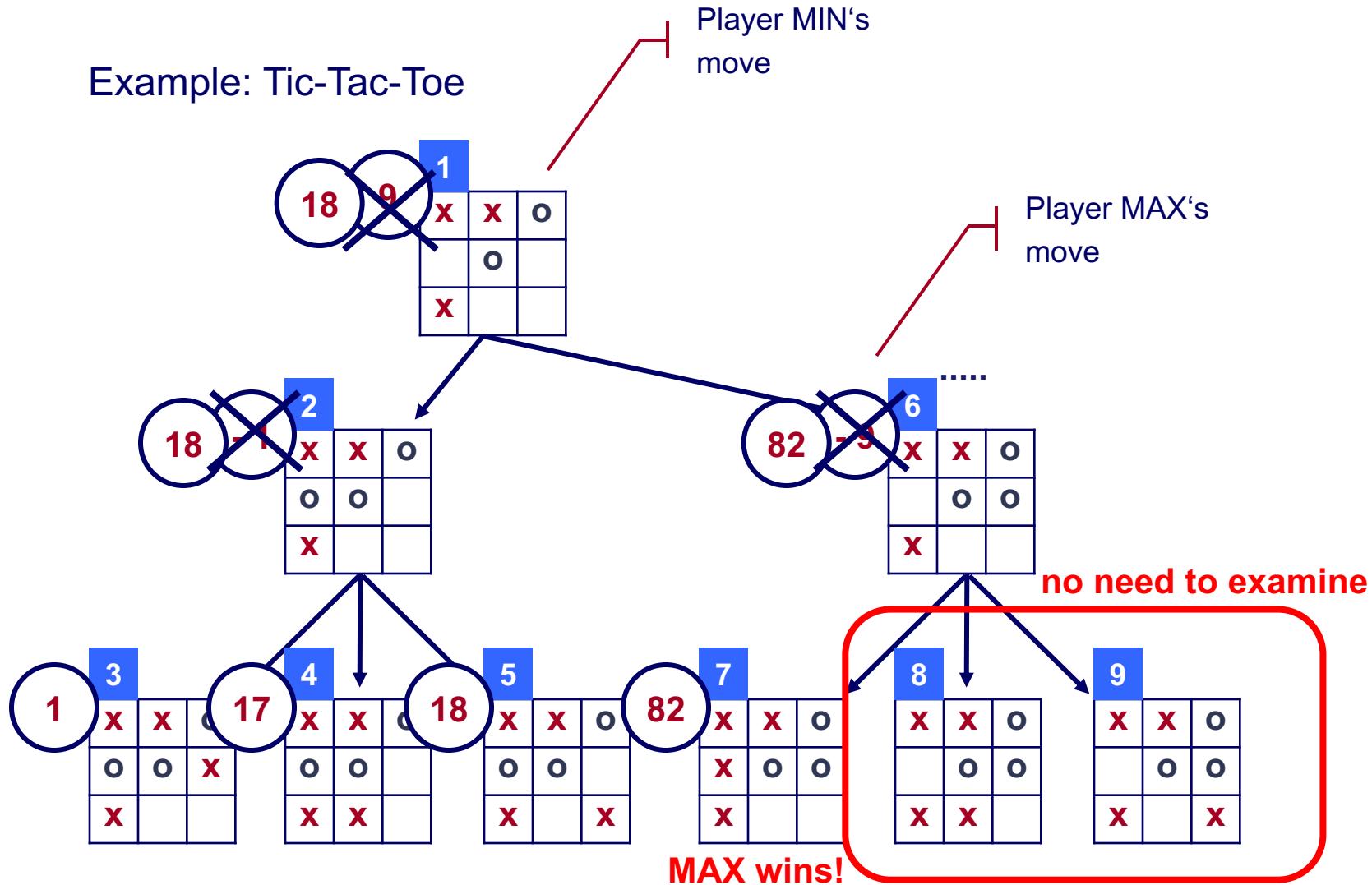
Example: Tic-Tac-Toe



Continue example



Example: Tic-Tac-Toe



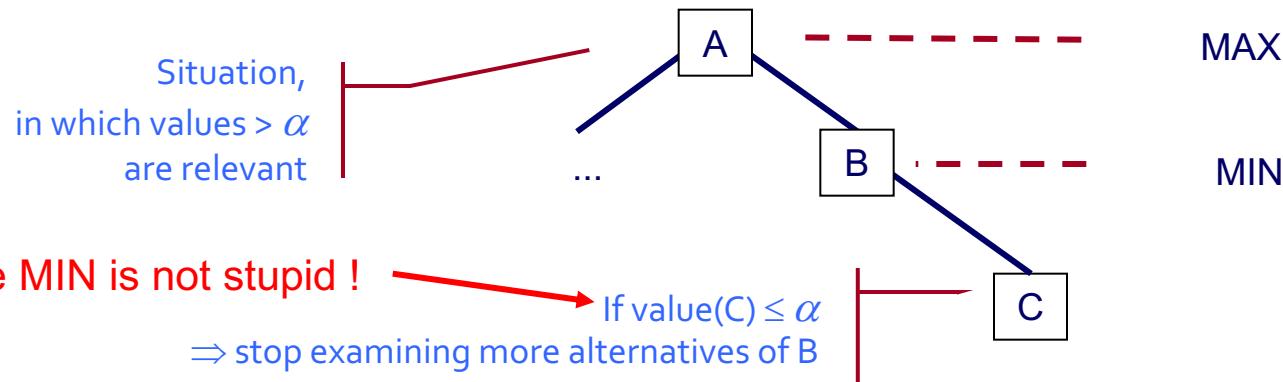
Search techniques

■ α - β -pruning of MINMAX :

The *maximizing* player MAX notes the *lower limits* for his situations to be reached using a parameter α

The *minimizing* player notes the *upper limits* for his situations to be reached using a parameter β

■ Effects: (for α)



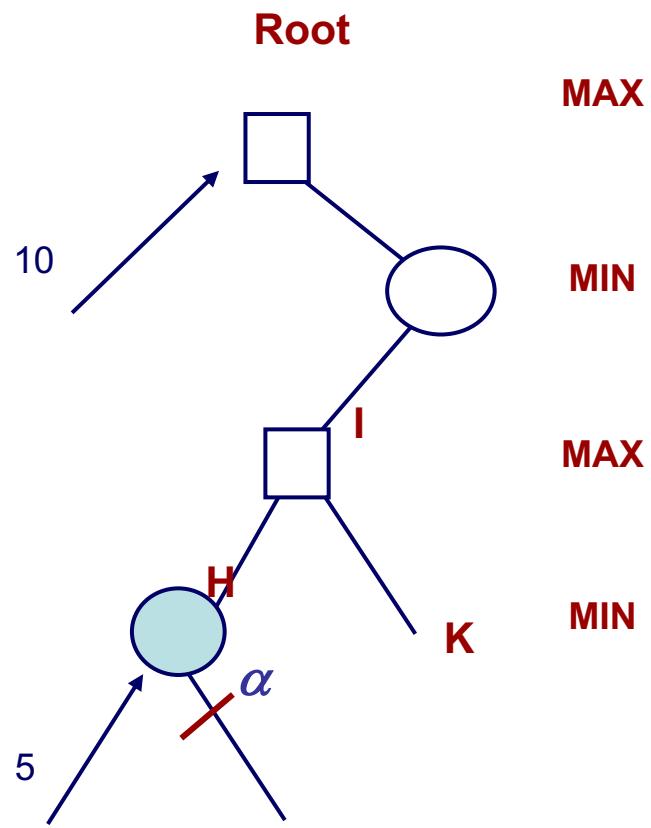
Alpha-Beta

Step-By-Step Examples



The effect of rules

- Example (α -bound):



H is MIN-node

The highest value of all MAX-predecessors is at the root (value = 10)

According to the β -bound-rule, the α -limit is 10.

*As soon as node **H** receives the value 5 from its left successor, it is below the α -limit.*

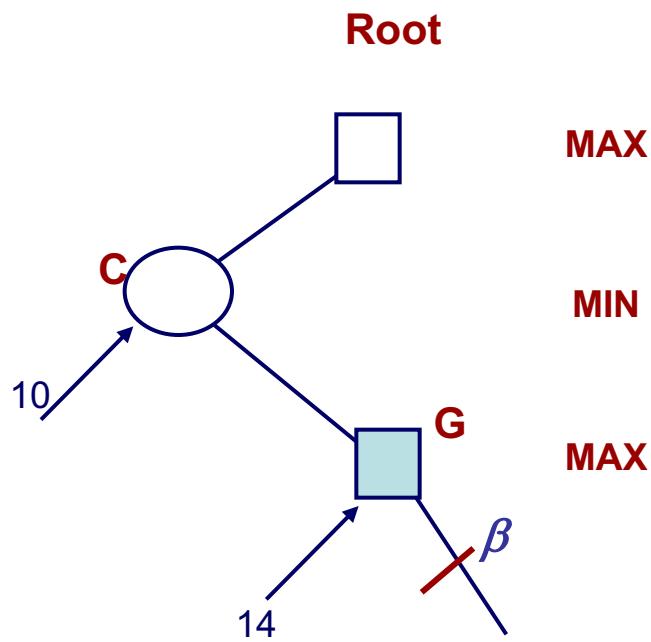
*H informs its predecessor **I** which continues with the evaluation of its other successor.*

*I hands the α -limit 10 also over to **K**.*

*If **H** would have delivered the value 15 instead of 5 to **I**, **I** would have a new value since 15 is more than 10. At the same time 15 had become the new α -limit for **K***

The effect of rules

- Example (β -bound):



G is MAX-node

The lowest value of all MIN-predecessors is 10 (node C). Thus, node C becomes the β -limit.

The evaluation of the node G produces the value 14 from its left successor – but 14 already exceeds the β -limit.

Thus, no further evaluation of G is necessary. G informs its predecessor.

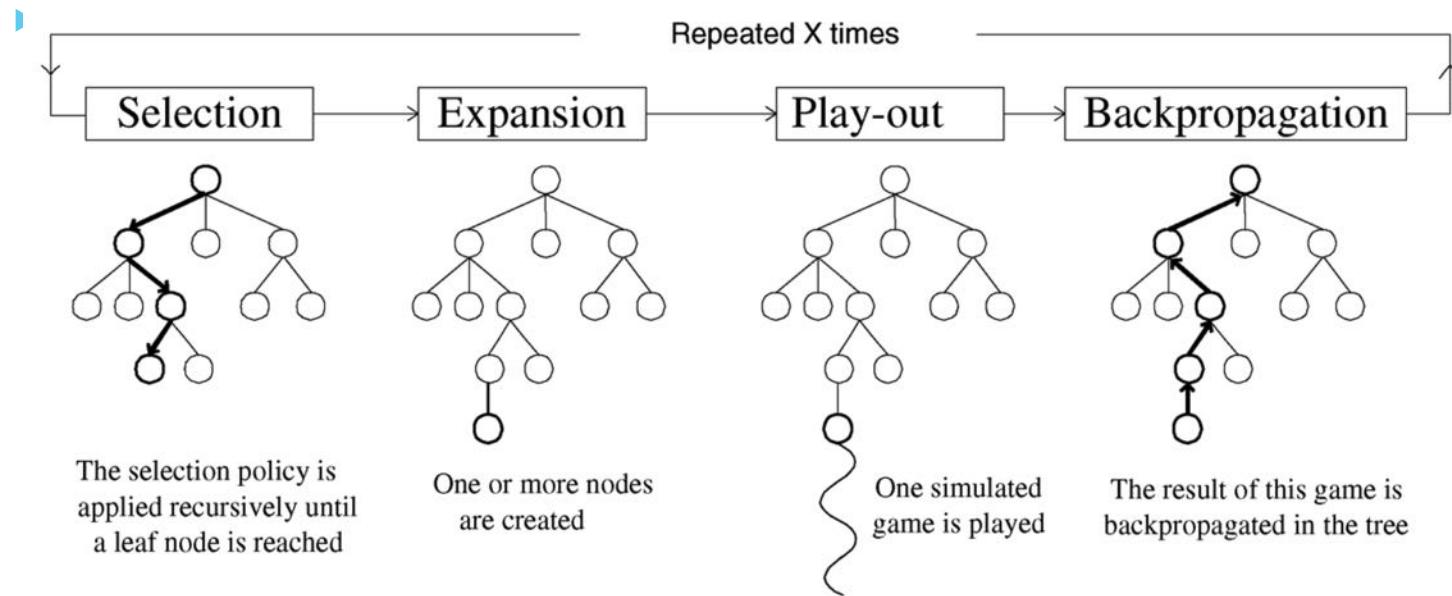
Since C does not possess any other successors, its value is now definitely 10. This value is given upwards.

When applying α - β -pruning of MINMAX, we may learn several things

- Terminate the search at the MIN-node if a successor exists with a value that is lower or equals the predecessor's lower limits α
- Terminate the search at the MAX-node if a successor exists with a value that is higher or equals the predecessor's upper limits β
- Savings depend on the order in which the moves are examined
- If we would consider an ideal order of the game tree, in practice we can save about half of the evaluations using α - β -pruning

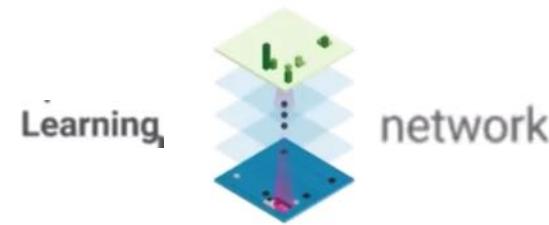
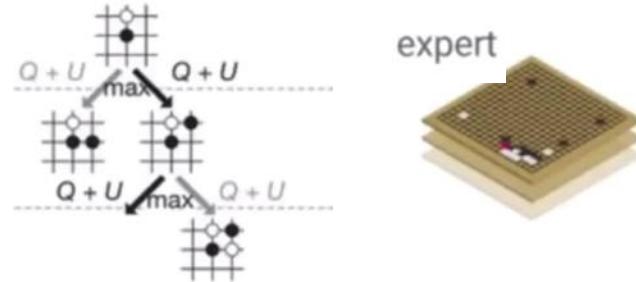
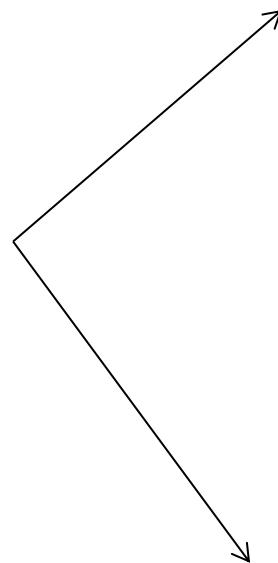
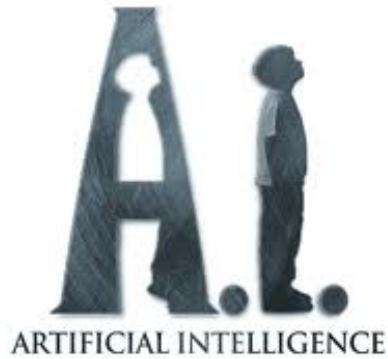
Back to Alpha Go (movie)

Monte Carlo Tree Search



Real-Time Monte Carlo Tree Search in Ms Pac-Man. Pepels et al. IEEE Transactions on Computational Intelligence and AI in Games (

**highly efficient complex (1) search
and (2) knowledge representation**

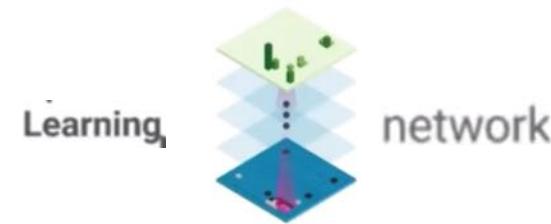
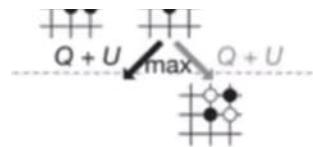
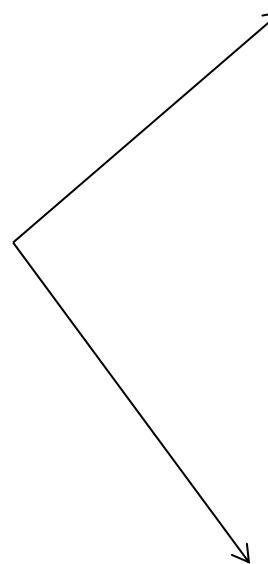
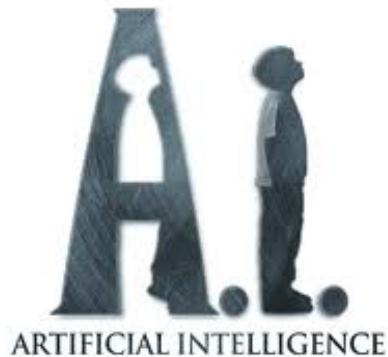


(3) learning: statistical analysis and optimization

highly efficient complex (1) search
and (2) knowledge representation



If we have knowledge that we can reason about things
and do not have to search, right ?



(3) learning: statistical analysis and optimization

Based on “Symbolic AI” by

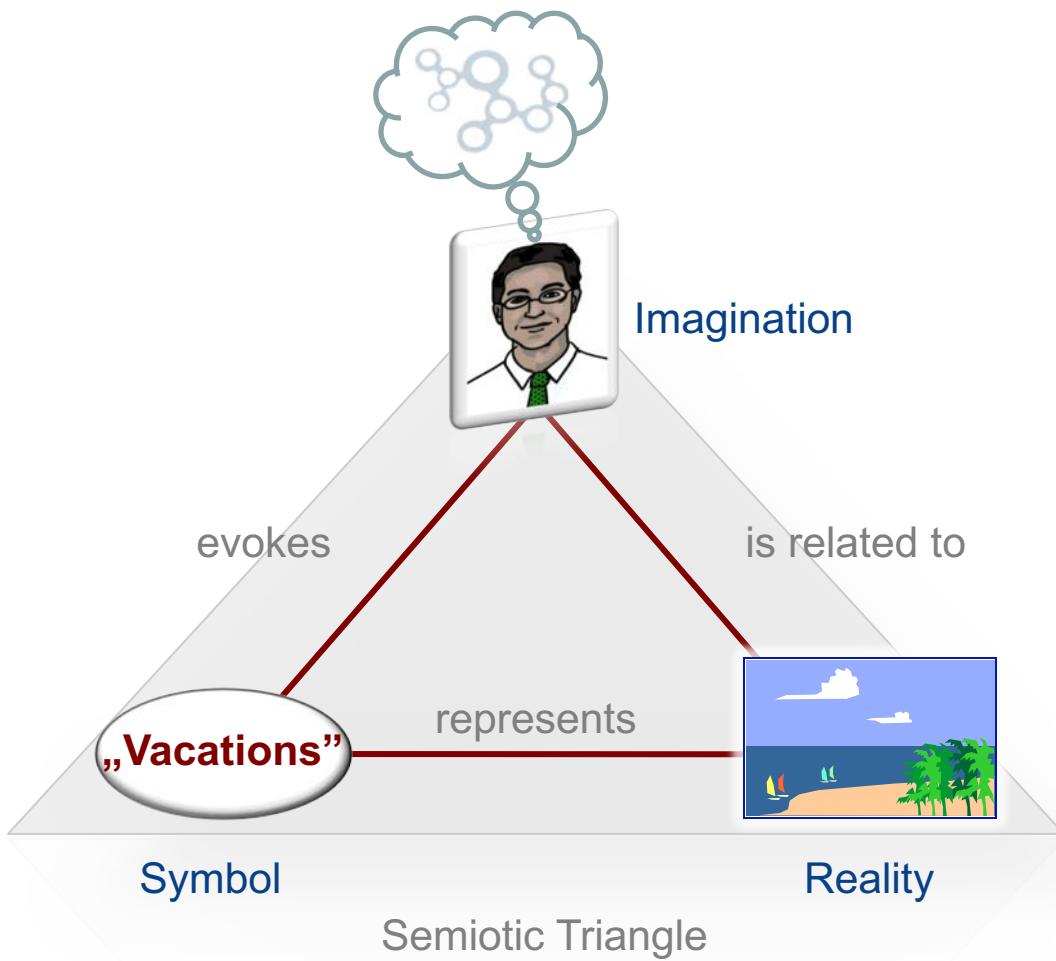
Prof. Dr. Andreas Dengel

Einführung in die KI,

Teil 2 Reasoning

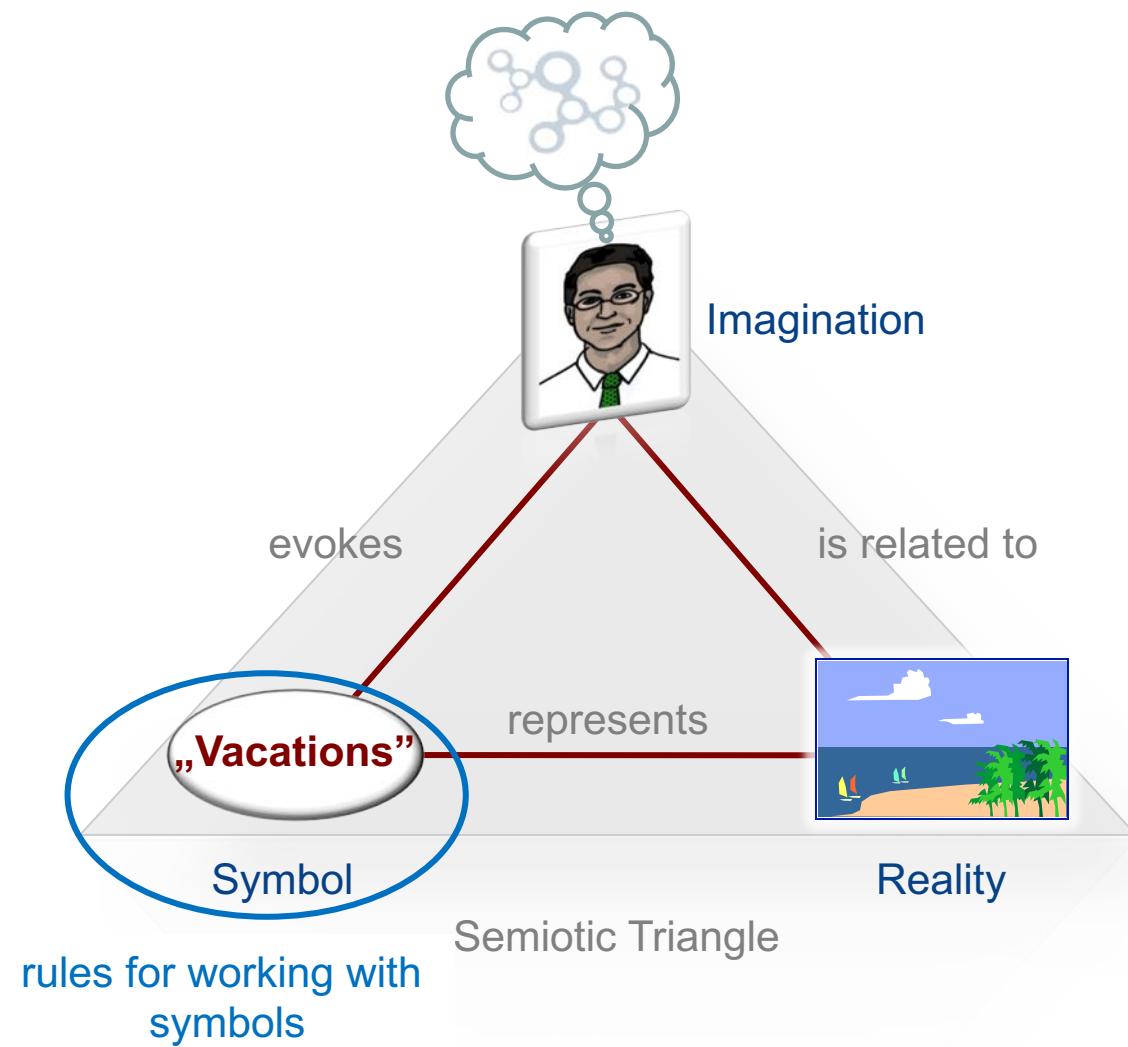
Prof. Dr. Paul Lukowicz

Symbol processing may be explained by the Semiotic Triangle



- Our environment consists of items, facts and events that are „real“ and determine our lives („what is going on“)
- In order to express their thoughts, people use signs, symbols, or characters that may be understood by others („what I couch or explicate“)
- People reading texts put contents together and create their very individual imagination („what I mean“)

Symbol processing may be explained by the Semiotic Triangle



- Our environment consists of items, facts and events that are „real“ and determine our lives („what is going on“)
- In order to express their thoughts, people use signs, symbols, or characters that may be understood by others („what I couch or explicate“)
- People reading texts put contents together and create their very individual imagination („what I mean“)

1. Inference types: Inference as formal reasoning

- *Reasoning* means to derive things that are not known yet from things that are known already (Shapiro, 1987)
- Inference stands for basic and formal reasoning
- The ability to deduce knowledge is a decisive element of human approaches to problem solving
 - On the basis of known facts ...*
 - ... we conclude that plausible events have already taken place
 - ... we speculate by assuming correlations
- There are types of inference conserving the truth and types not conserving the truth
 - *Deduction is a logically correct, i.e. a truth conserving inference*
 - *Induction and abduction are types of inference not conserving the truth*

1. Inference types

■ Abductive reasoning

Rule: All beans from this bag are white

Fact : These beans are white

Explaining fact: These beans are from this bag

■ Deductive reasoning

Explaining fact: These beans are from this bag

Rule: All beans from this bag are white

Fact : These beans are white

■ Inductive reasoning

Fact : These beans are white

Explaining fact: These beans are from this bag

Rule: All beans from this bag are white

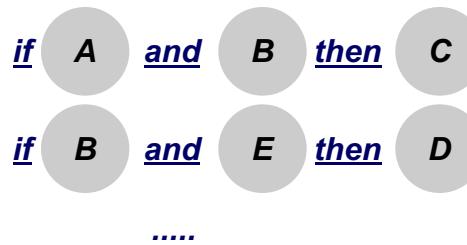
1. Inference types: Deduction and logic

- In contrast to induction and abduction deduction is a truth conserving and logically consistent inference
- Induction or abduction are used for reasoning only if deduction is not possible to apply, to complex or not promising to finish the task
 - ▶ *For now we concentrate on deductive reasoning as a principle in AI*
- In AI knowledge representation and reasoning require the formalization of knowledge and mechanisms of deduction
 - ▶ *We apply the logic of mathematics to formalize reasoning using mathematical means*

The ability to infer knowledge is central part of human problem solving skills

- Considering a given situation ...
 - ... we may infer that plausible events already took place
 - ... we make an educated guess because we are able to interpret a given context
- One central aim of AI is to investigate human inferring and to formally describe them
 - Expressiveness of logic allows an abstraction of human thinking*
- One example for expressing propositional logic inference is deduction for which in many cases, rules may be applied

General Form:



Example

■ Given Text

Fritz is a man. Fritz has six children. Ulrike, his oldest daughter has a son herself, named Frank, and a daughter, named Evi. The sons of Fritz are named Matthias and Frieder.

■ Knowledge Base:

- *is_a_man(fritz).*
- *is_daughter_of.ulrike, fritz).*
- *is_son_of(frank, ulrike).*
- *is_daughter_of(evi, ulrike).*
- *is_son_of(matthias, fritz).*
- *is_son_of(frieder, fritz).*

■ Rule from the rule base

- *is_grandson_of(X, Y) :- is_son_of(X, Z), is_daughter_of(Z, Y).*

■ Inferred Knowledge

- *is_grandson_of(frank, fritz).*

Beispiel Logikrätsel

Fünf Freunde haben gegeneinander ein Wettrennen gelaufen. Nach der anschließenden durchzechten Nacht können sie sich jedoch nur noch an wenige Details erinnern:

1. Tim ist vor Lukas im Ziel eingelaufen.
2. Janina war früher als Tim, Franz oder Lukas im Ziel.
3. Anna ist vor Janina oder Franz im Ziel angekommen.
4. Tim war vor Anna im Ziel oder Lukas war vor Anna im Ziel.
5. Franz schneller als Tim im Ziel.
6. Außerdem wissen sie noch, dass keine zwei von ihnen gleichzeitig im Ziel eingelaufen sind.

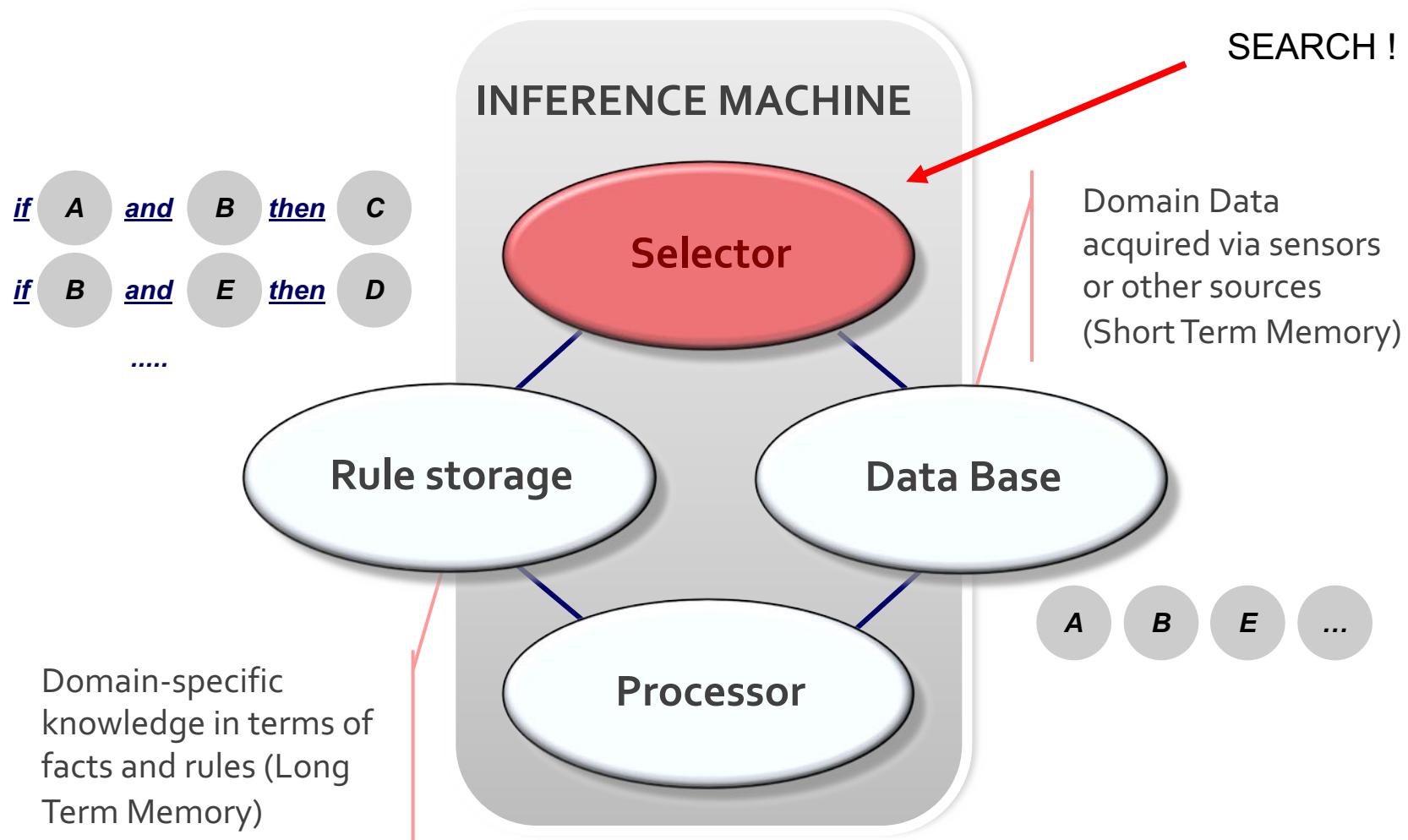
In welcher Reihenfolge sind die fünf denn nun eingelaufen?

Lösung

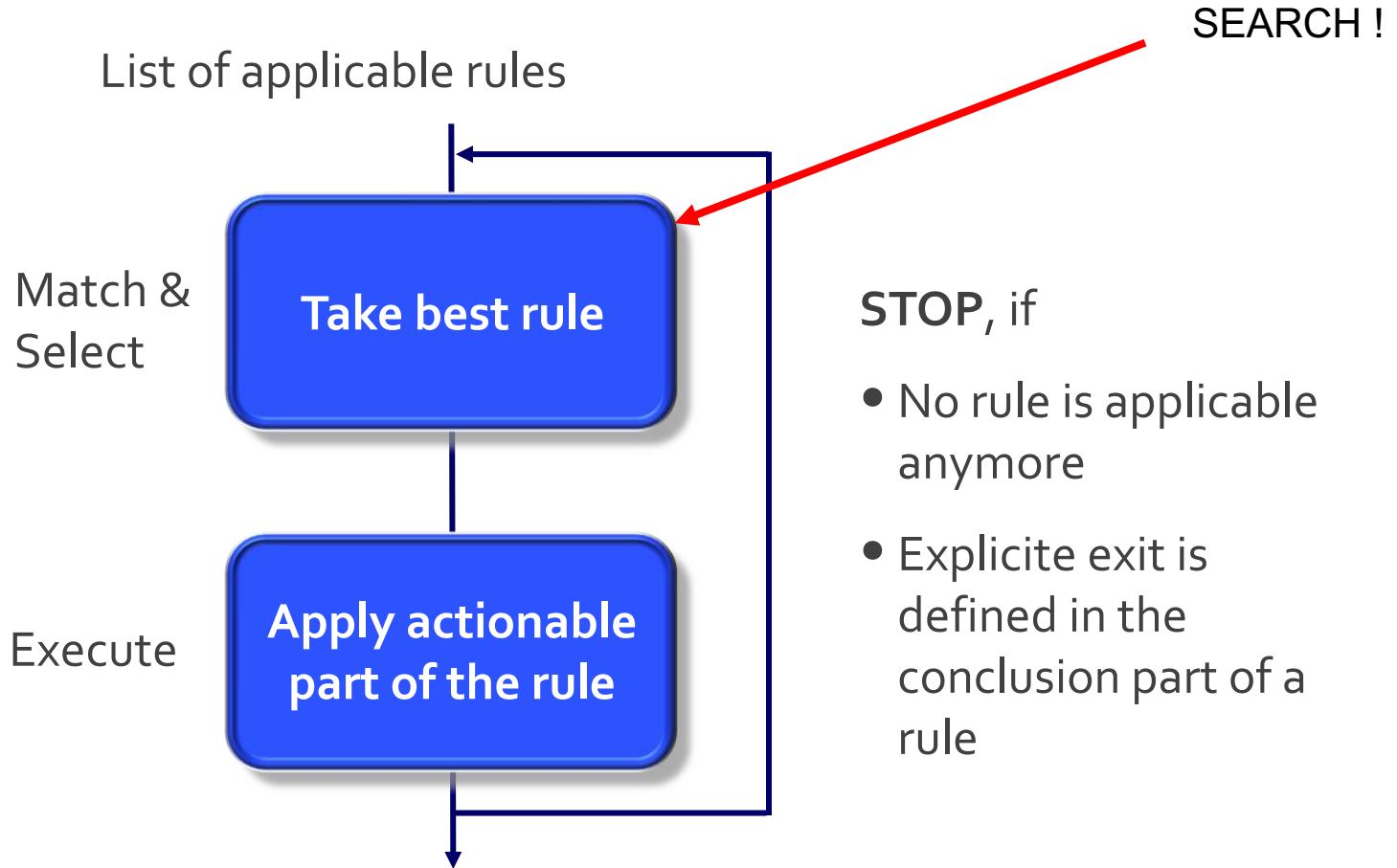
1. Kombiniert man die erste und fünfte Aussage, so erkennt man, dass Franz schneller als Tim und dieser wiederum schneller als Lukas im Ziel war.
2. Betrachte nun die dritte Aussage: Angenommen, Anna war vor Franz im Ziel, so führt die vierte Aufgabe zu einem Widerspruch, denn dann hätte Tim oder Lukas vor Franz im Ziel gewesen sein müssen. Also war Anna vor Janina im Ziel.
3. Also ist entweder Janina oder Lukas auf dem letzten Platz gelandet. Nach der zweiten Aussage war aber jemand hinter Janina. Also ist Lukas auf dem letzten Platz.
4. Betrachtet man die vierte Aussage, so muss wohl Tim vor Anna im Ziel gewesen sein, weil Lukas ja Letzter ist. Die beiden Frauen sind also zwischen Tim und Lukas platziert.

Lösung: Franz, Tim, Anna, Janina, Lukas

The inference engine selects rules to be processed (fired) based on the recent data (query)



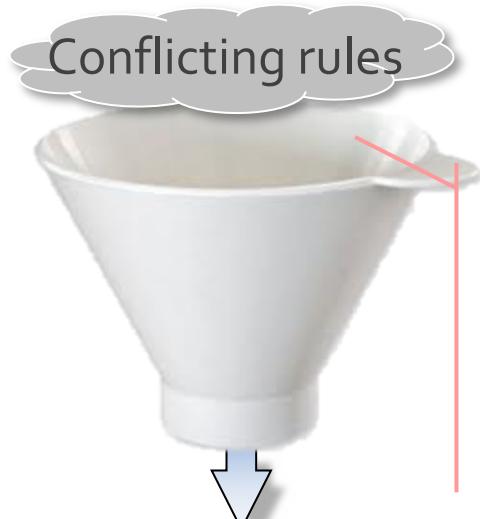
The working cycle to process rules is intuitively simple



Conflicts arise if more than one rule can be applied

- In such a case, the selector has to resolve the conflict by employing some kind of heuristics

meaning:

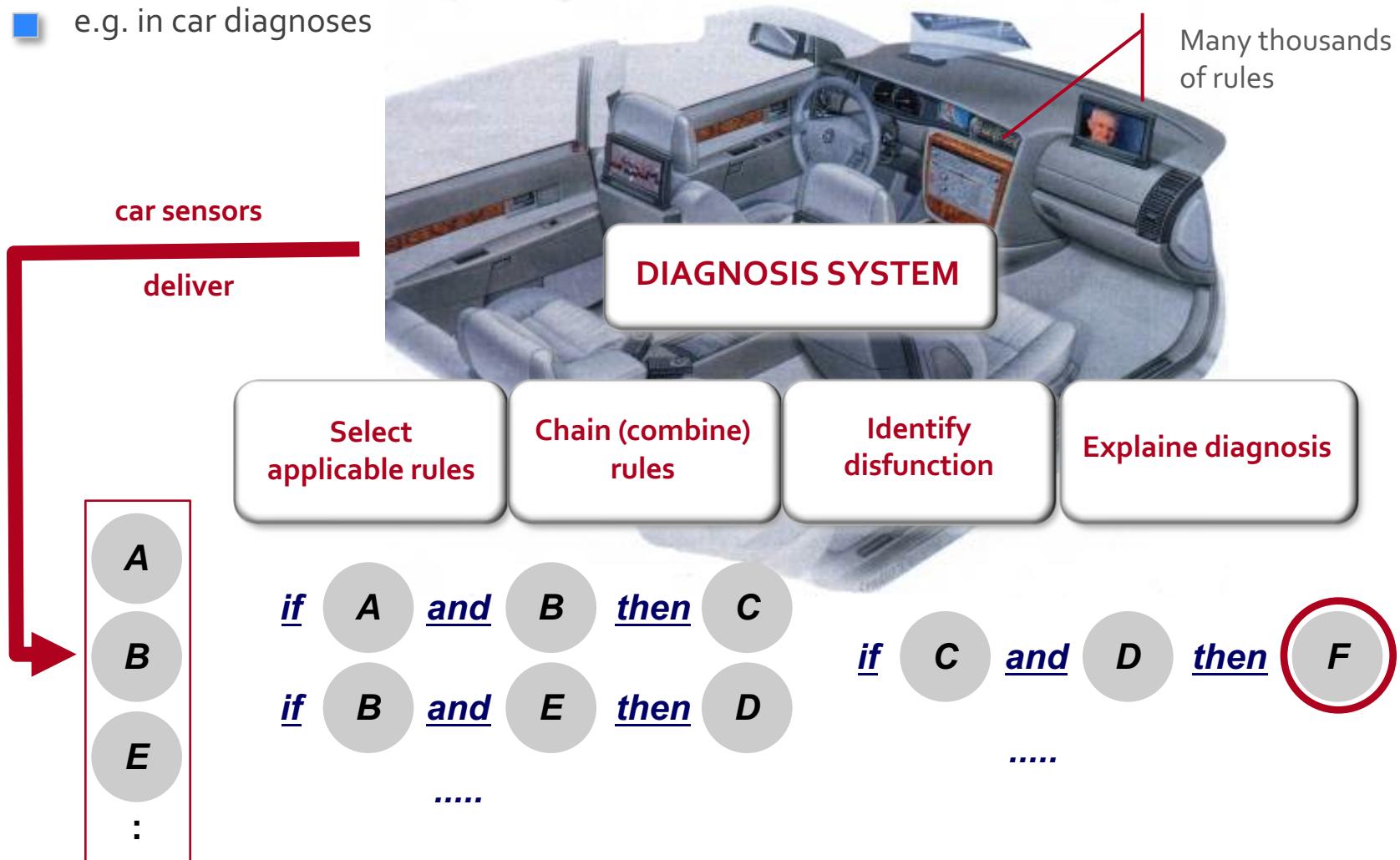


Apply heuristic:

- Follow order
- Select most special rule
- Select based on currentness of data
- Select based on additional knowledge

Many modern applications use large knowledge bases with many rules

■ e.g. in car diagnoses



There are different approaches to solve given problems



First approach: Imperative or procedural programming

The programmer uses computer language to describe how the problem can be solved in general

He or she implements a method of calculation to solve the problem

This method of calculation is running after the start of the program



Second approach: Logic programming

The programmer uses computer language to describe the problem.

He does not implement a method of calculation to solve the problem, because it exists in an interpreter language.

Such a method of calculation is automatically generated after the start of the program – this is the task of the interpreter

Remember Predicate Logic

■ Example knowledge

- All computer scientists are smart. Sophie ‘s dad is a computer scientist.

■ Representation

- $\forall x (\text{computer scientist}(x) \rightarrow \text{smart}(x)); \text{Computer scientist}(\text{dad}(\text{Sophie}))$

■ Definition

- Each variable (X, Y, ...) is a term.
- If f is a n-ary functional symbol and t_1, \dots, t_n are terms, then $f(t_1, \dots, t_n)$ is also a term.
- If P is an n-ary predicate symbol and t_1, \dots, t_n are terms, then $P(t_1, \dots, t_n)$ and $(t_1 = t_2)$ are formulars (atomic formulars).
- If F and G are formulars then $\neg F$, $(F \wedge G)$, $(F \vee G)$, $(F \rightarrow G)$, $(F \leftrightarrow G)$ are also formulars.
- If F is a formular and x is a variable, then $\forall X F$ and $\exists X F$ are also formulars.
- Atomic and negative atomic formulars are also called literals.

Logical programming is a kind of declarative programming

- Logical programming has its roots in automated proofs, an area of AI

Predicate logic is used (particularly the representation of clauses) as programming language

Resolution is used to demonstrate,

1. *that there is a solution to the given problem*
2. *the kind of the solution that exists to the problem*

- Kowalskis Formula:

$$\text{Algorithm} = \text{Logic} + \text{Control}$$

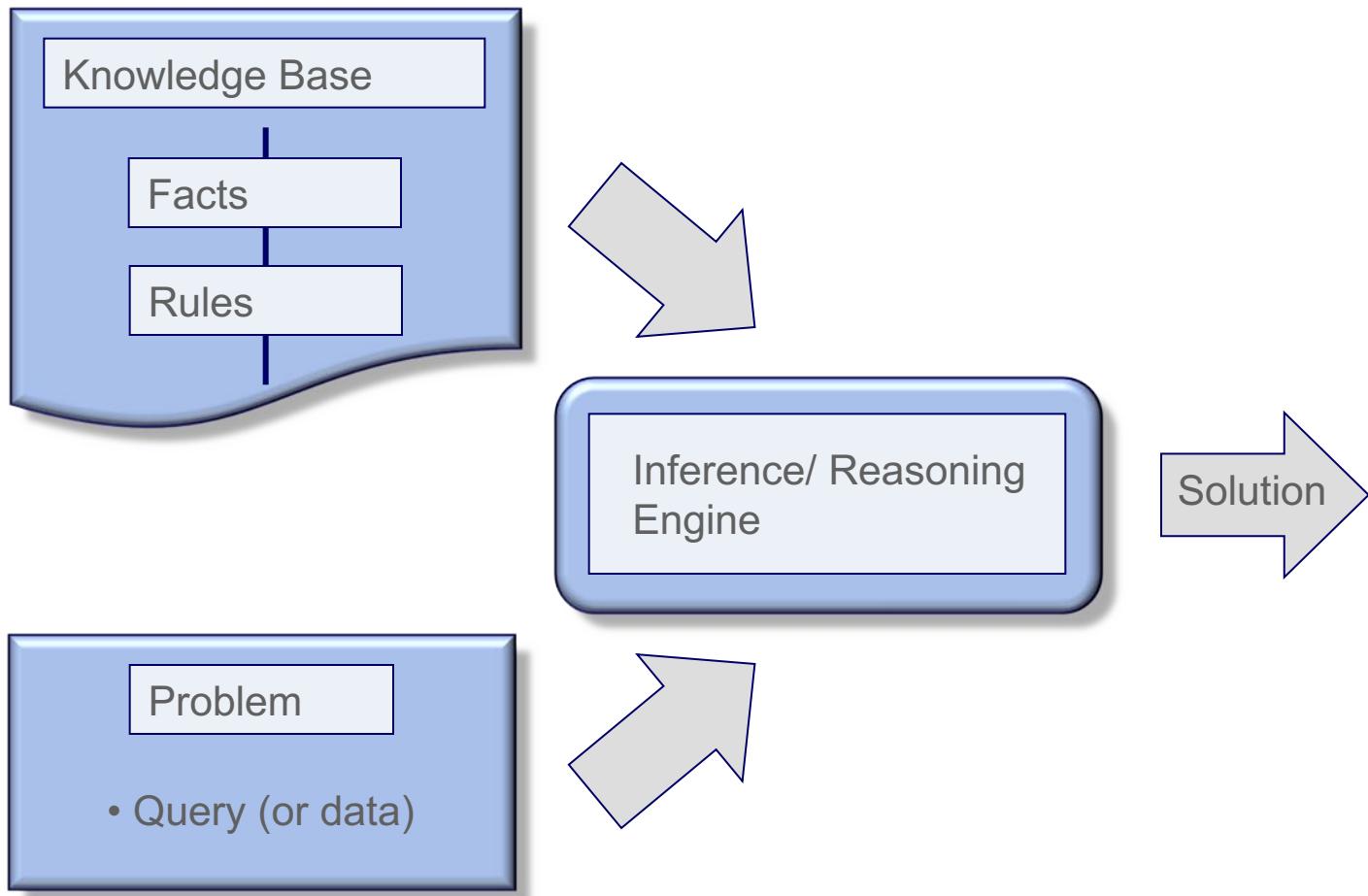
- Implemented in Prolog

**Algorithm =
Logic + Control**

Robert Kowalski
Imperial College, London

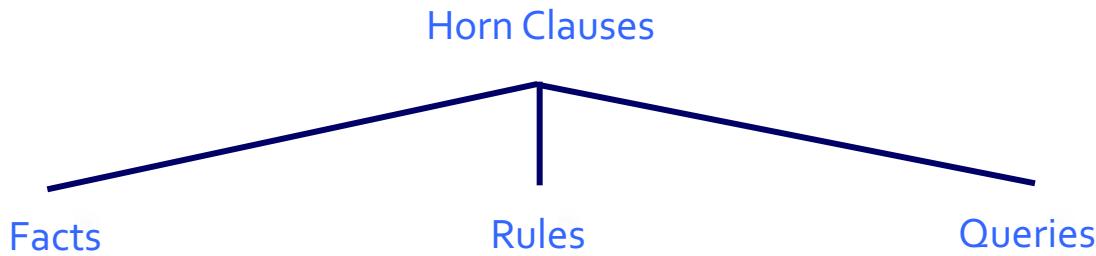
An algorithm can be regarded as consisting of a logic component, which specifies the knowledge to be used in solving problems, and a control component, which determines the problem-solving strategies by means of which that knowledge is used. The logic component determines the meaning of the algorithm whereas the control component only affects its efficiency. The efficiency of an algorithm can often be improved by improving the control component without changing the logic of the algorithm. We argue that computer programs would be more often correct and more easily improved and modified if their logic and control aspects were identified and separated in the program text.

The architecture of a predicative programming system is threefold



The basic elements of a PROLOG system are horn clauses

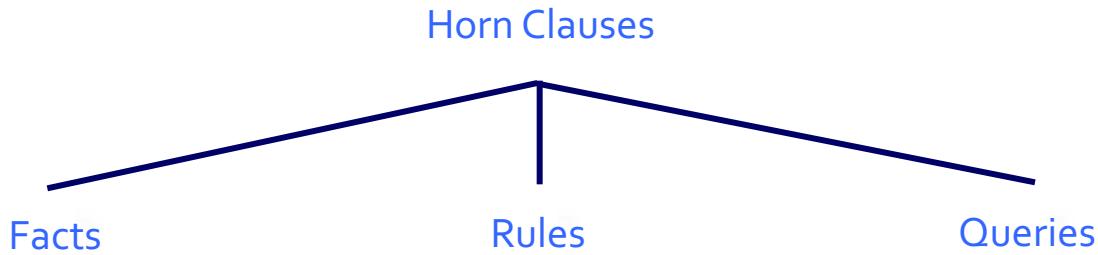
- Facts, Rules and queries may be built using horn clauses



- A predicative programming system aims at giving answers to questions by using the knowledge base
- The programmer is responsible to generate the knowledge base, i.e. define the basic facts and establish the corresponding rules
- **The programmer is not responsible for the construction of the inference / reasoning engine**
- The inference engine attempts to give answers to queries which are not yet covered by the existing facts

The basic elements of a PROLOG system are horn clauses

- Facts, Rules and queries may be built using horn clauses



- A predicative programming system aims at giving answers to questions by using the knowledge base
- The programmer is responsible to generate the knowledge base, i.e. define the basic facts and establish the corresponding rules
- The programmer is not responsible for the construction of the inference / reasoning engine
- **The inference engine attempts to give answers to queries which are not yet covered by the existing facts**

Logical programming: Advantages

- *Many problems can relatively easily be described by using rules and facts – but it might be hard to implement procedural algorithms for solving them, because the way is not straightforward (not deterministic)*
- *In general dealing with non determinism is solved in logical programming languages; there is no need for the programmer to deal with this aspect. Thus, logic programming is the better choice for problems of this kind*
- *The deduction mechanisms that have been realized in the interpreter may point to efficient algorithmic solutions, that the programmer might not have considered in advance*
- *In particular the applications in the fields automatic reasoning and planning are of this nature and are typically of interest for AI*

In logical programming, computing means proofing

- Example: Three lines form a logical program:

likes(popeye,spinach).

likes(popeye,icecream).

kisses(olivia,X) :- likes(X,spinach), likes(X,icecream).



kisses(olivia,X) :- likes(X,spinach), likes(X,icecream).

Equals the predicate logical formula

likes(X,spinach) \wedge likes(X,icecream) \rightarrow kisses(olivia, X)

- Start of the program: by querying for instance:

? - *kisses(olivia,Y).*

This resembles the question if there is an individual which is being kissed by Olivia

Answer: not only yes or no, but if yes also the individual's name

The programmer has to define facts and rules, and further formulates queries

- Facts: A. (A is a literal)
 - Rules: A:- B, C, (":-" means "if", "," means "and")
 - Queries: ?- B, C,
-
- Example (dialogue with a PROLOG system):

```
member(E, [E|REST]).
```

```
member(E, [KOPF|REST]) :- member(E, REST).
```

User: ?- member(7, [4, 6, 7, 9, 8]).

System: Yes

User: ?- member(10, [4, 6, 7, 9, 8])

System: No

User: ?- member(WAS, [4, 6, 7, 9, 8])

System: WAS = 4

User: ;

System: WAS = 6

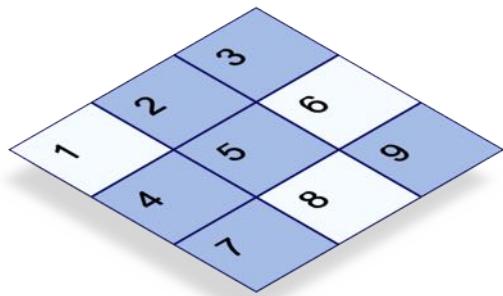
Are there other possible answers?

Exemplary problem



Chess

„Move the knight from field S to field G“
(Search a path on a simplified 3x3 chess board)



From field 1, a knight may move to fields 6 and 8

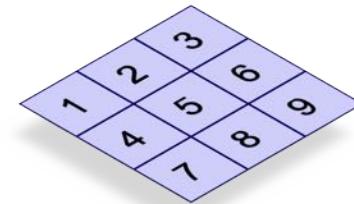
From field 2, a knight may move to fields 7 and 9

...

- Break, if field G is reached
- else:
 - test whether N has been already visited
 - Label N as visited
 - Search a path from N to G

Exemplary problem (cntd.)

„Move the knight from field S to field G“
(Search a path on a simplified 3x3 chess board)



- | | | |
|---|-----------------|---|
| ■ | Facts: | move(1, 6). move(1, 8).
move(2, 7). move(2, 9).
:
move(9, 2). move(9, 4). |
| ■ | Rules: | path(G, G, SEQUENCE).
path(S, G, SEQUENCE) :- move(S, N)
not_member(N, SEQUENCE)
path(N, G, [N SEQUENCE]). |
| ■ | Queries: | ?- path(1, 3, [1]). Answer: YES
?- path(1, WHERE, [1]) Answer: WHERE = 6 |

The semantic of an interpretation is depending on the control flow

Knowledge Base (simplified example):

Facts: move(c, d). move(d, e). move(e, c).

Rules:

Case (1) path(S, S).

path(S, Z) :- move(S, N), path(N, Z).

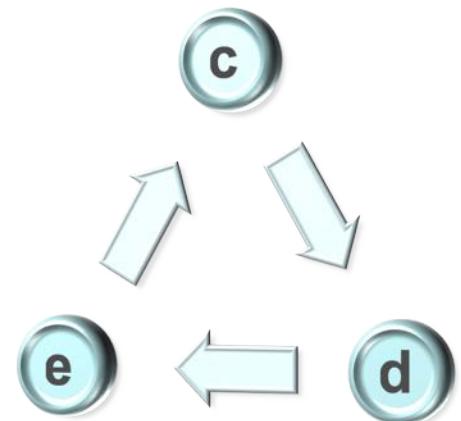
Case (2) path(S, Z) :- move(S, N), path(N, Z).

path(S, S).

Query: ?- path(d, d).

Antwort: Case (1): YES

Case (2): NO



Application Areas

- *Interpretation* – Inferring description from data
- *Prediction* – Inferring consequences of situations
- *Diagnosing* – Malfunctions from observables
- *Designing* – Configuring objects under constraints
- *Planning* – Designing actions
- *Monitoring* – Comparing observations with expectations
- *Debugging* – Prescribing remedies for malfunctions
- *Repairing* – Administrating prescribed remedies
- *Instructing* – Diagnosing and improving student's behavior
- *Controlling* – Governing overall system control

Application Examples

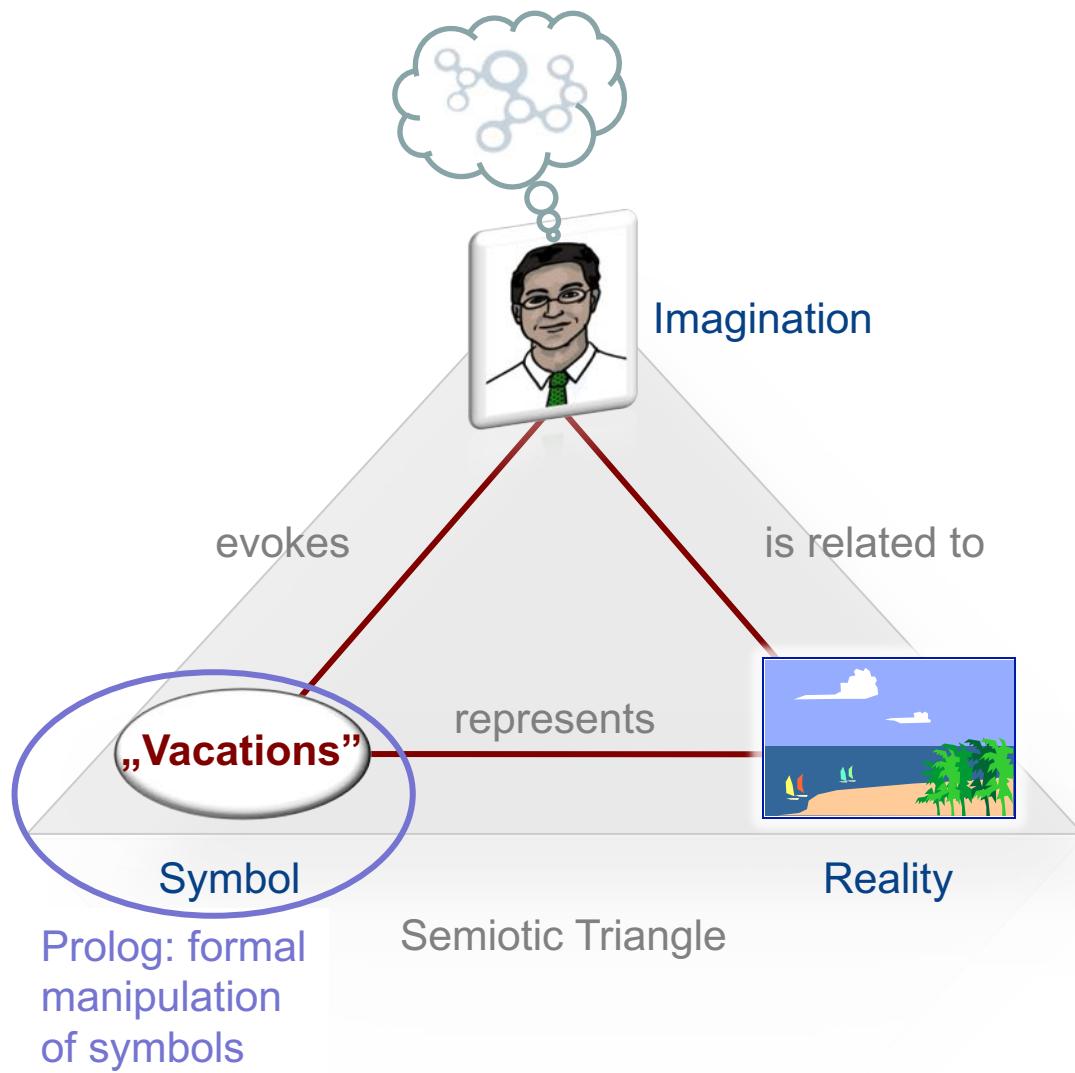
- *Dendral (1960s)* – Infers a compound's molecular structure from mass spectral and nuclear response data. Example:
 - If there are two peaks at x_1 and x_2 , so that the following applies:
 - $x_1 + x_2 = M + 28$ (M equals the mass of the entire molecule),
 - $X_1 - 28$ is a high peak,
 - $X_2 - 28$ is a high peak,
 - at least one of x_1 and x_2 is high,
 - then we are dealing with a Ketone-subgroup.
- *MYCIN* – Helps to diagnose and treat bacterial infections
 - Knowledge based system with fuzzy logic
 - In 1991 diagnosis programs reached level of doctors:
Lymph-node pathology, expert scoffs at the system's response Program explained the reasons Doctor finally agreed and admits his error

Application Examples (cont.)

- *Sophie (1976)* – Teaches students how to troubleshoot electrical circuits
 - SOPhisticated Instructional Environment
 - The system worked mainly through question answering and hypothesis evaluation
 - 50 circuit specific rules
- *INCO* (Integrated Communications Officer) - In charge of shuttle communications systems
 - Monitors communication between space shuttle and mission control
- *Wolfgang (1989)* – Composes music based on emotional computation

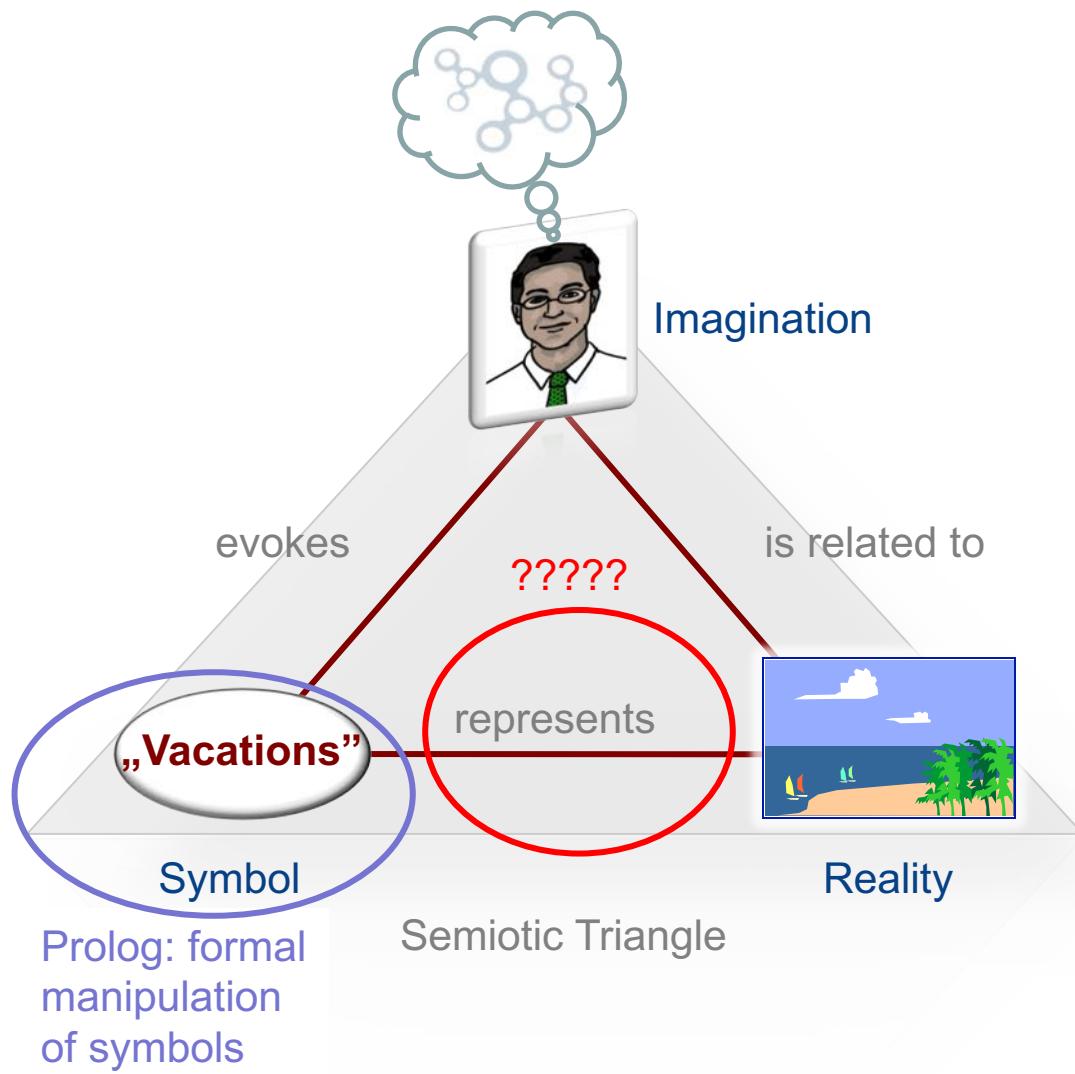
Mode detection rules	33
Good neighbor rules	4
Bad neighbor rules	4
Exception rules	9
Total	50

Symbol processing may be explained by the Semiotic Triangle



- Our environment consists of items, facts and events that are „real“ and determine our lives („what is going on“)
- In order to express their thoughts, people use signs, symbols, or characters that may be understood by others („what I couch or explicate“)
- People reading texts put contents together and create their very individual imagination („what I mean“)

Symbol processing may be explained by the Semiotic Triangle



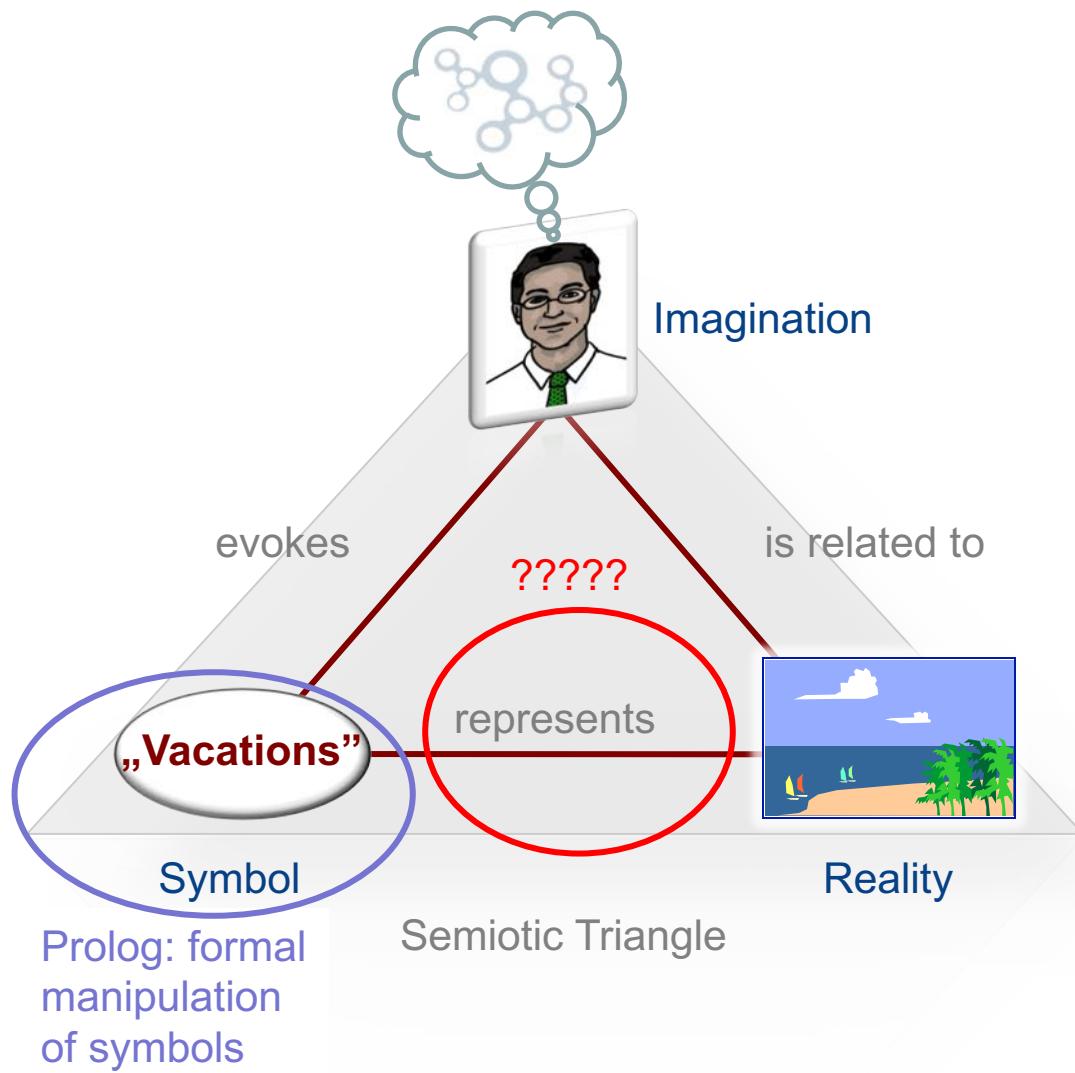
- Our environment consists of items, facts and events that are „real“ and determine our lives („what is going on“)
- In order to express their thoughts, people use signs, symbols, or characters that may be understood by others („what I couch or explicate“)
- People reading texts put contents together and create their very individual imagination („what I mean“)

Based on “Symbolic AI” by

Prof. Dr. Andreas Dengel

Einführung in die KI,
Teil 3 Wissensrepräsentation
Prof. Dr. Paul Lukowicz

Symbol processing may be explained by the Semiotic Triangle

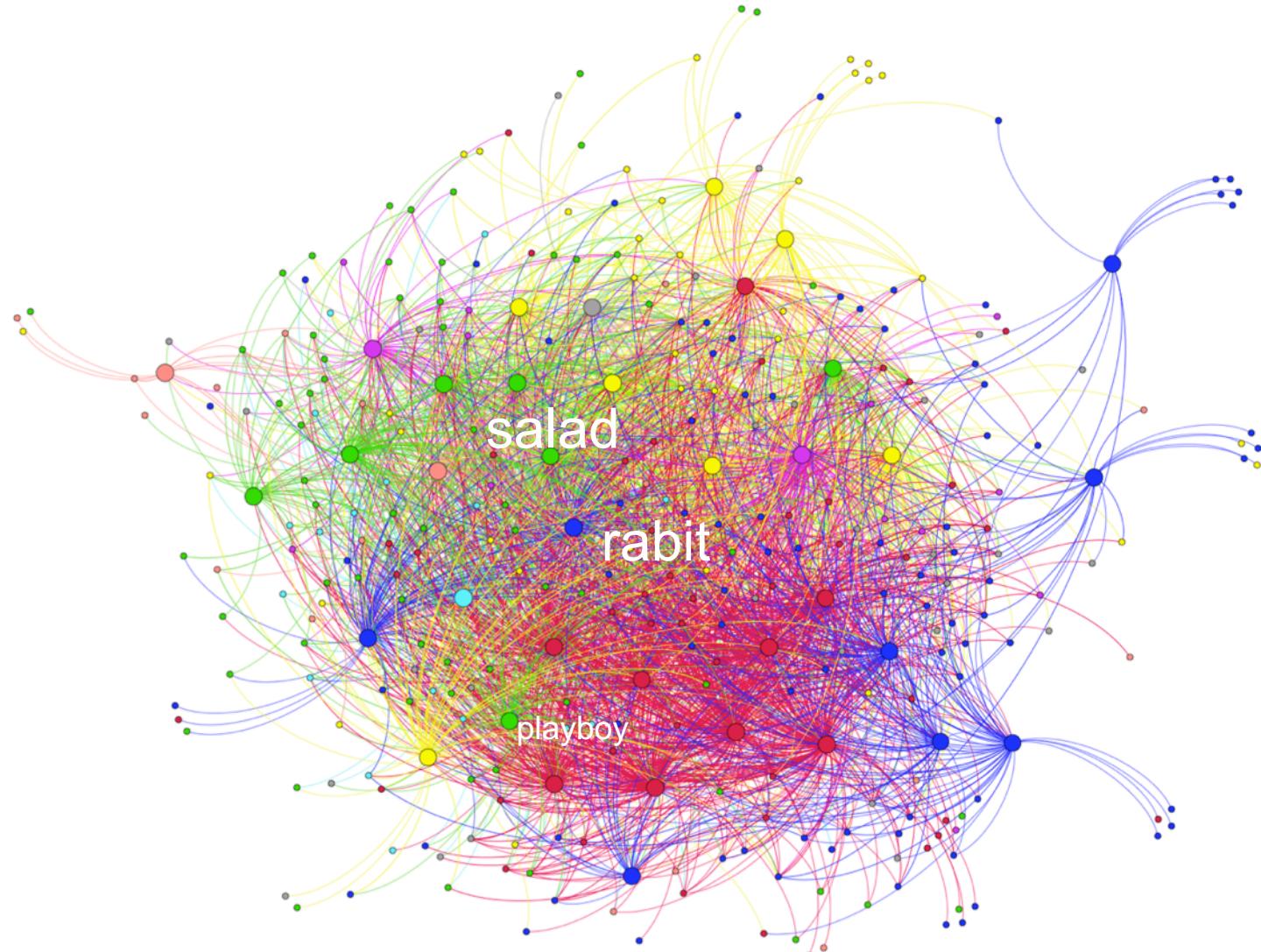


- Our environment consists of items, facts and events that are „real“ and determine our lives („what is going on“)
- In order to express their thoughts, people use signs, symbols, or characters that may be understood by others („what I couch or explicate“)
- People reading texts put contents together and create their very individual imagination („what I mean“)

An Example



The difference between symbols and knowledge: connections

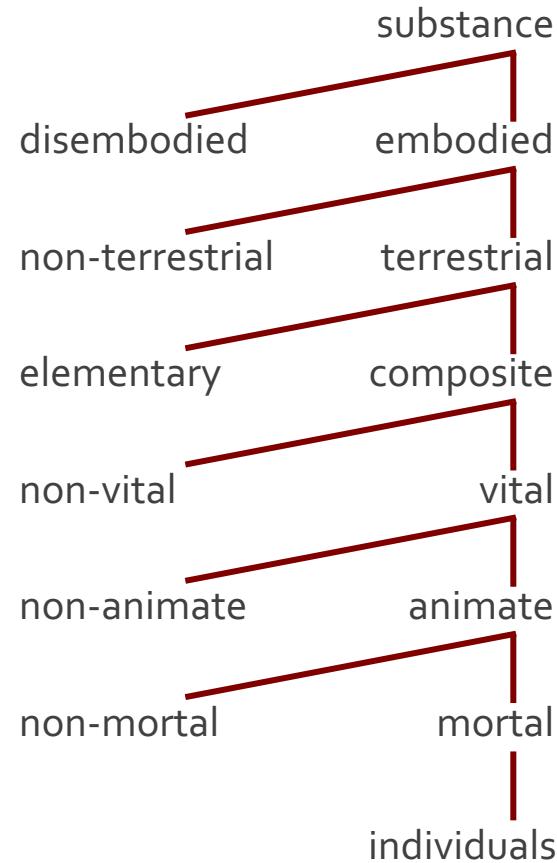


Semantic Networks are graphical representations for describing the meaning of a term

- Term is considered as a concept
- Concepts may be events, persons, objects, ideas, locations, ... and may be generic, i.e. are categories representing a class of things
- **The meaning of a concept (term) is described by its relations to other concepts (terms)**
 - ▶ Since sometimes such relations are not always hierarchical or follow an alphabetic order, they may be better represented by a network
 - ▶ Relations may describe any association between concepts that is very specific
- Knowledge is formalized via a labeled graph
 - ▶ Edges of the graph may be bidirectional but in most cases are implemented unidirectional
 - ▶ The access to a concepts is possible along the directed relations, i.e. all knowledge about a concept is accessible via its relations

Semantic Networks are one of the oldest ways for representing knowledge

- In the 3rd century BC the Greek philosopher Porphyrios designed the first Semantic Network on the basis of Aristotle's category system



Semantische Netze wurden eingeführt, um Sätze der natürlichen Sprache in formale Darstellung zu überführen

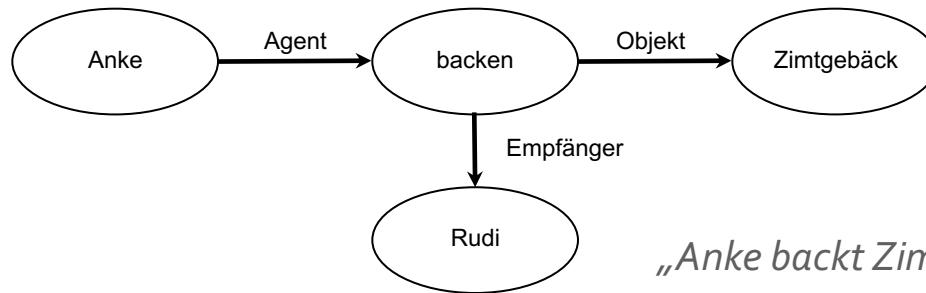
- Prinzip lässt sich mit einfachen Sätzen erläutern, die aus dem Tripel Subjekt-Prädikat(Verb)-Objekt bestehen, z.B.:



„Rudi mag Zimtgebäck.“

Subjekt und Objekt stehen dann für Knoten und Prädikate bzw. Verben für Kantenbezeichnungen

- Aussagen sind jedoch nicht allein auf eine einfache Form eingeschränkt

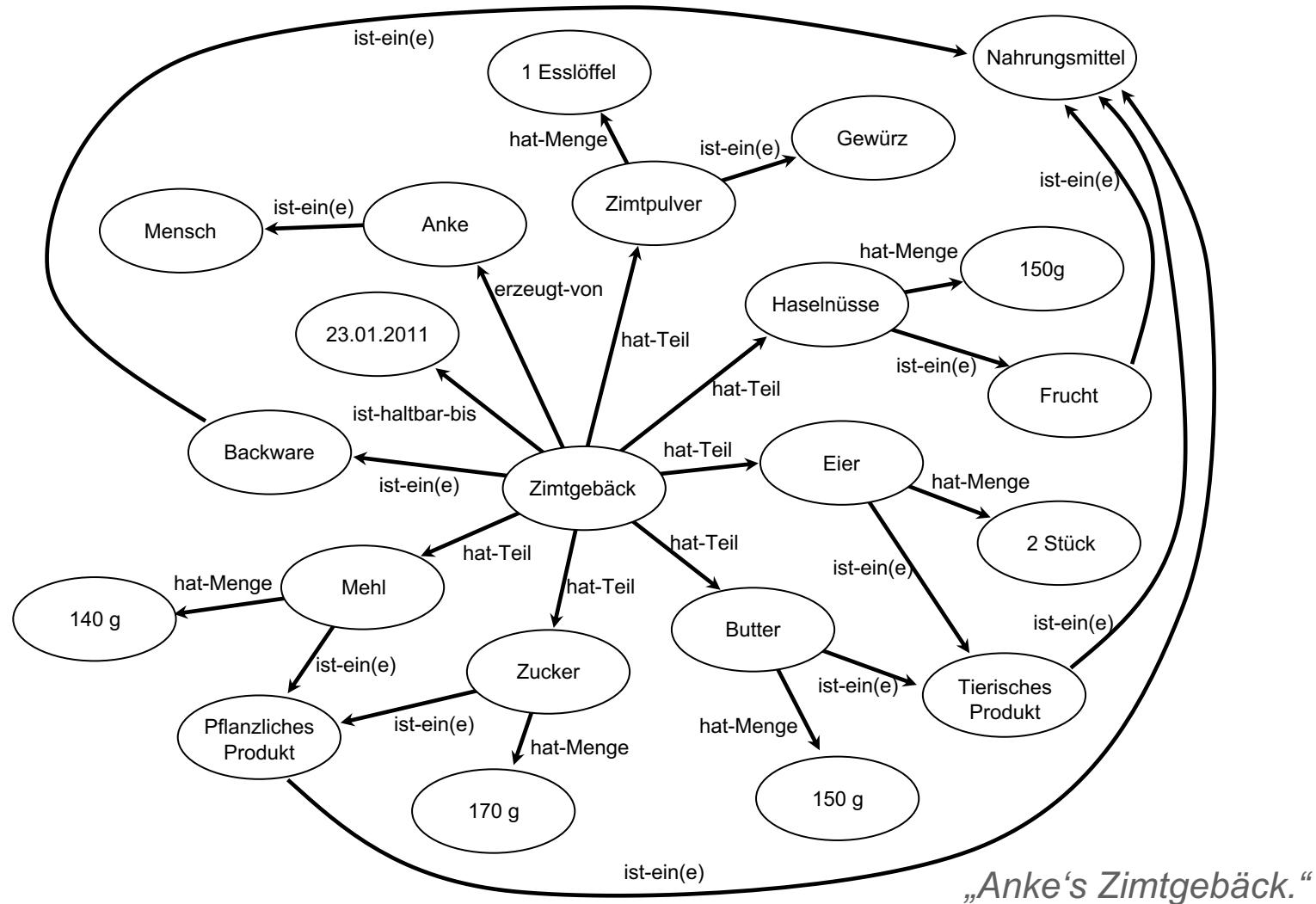


„Anke backt Zimtgebäck für Rudi.“

Verb wird zum zentralen Knoten und beschreibt die verschiedenen Rollen mittels der Relationen

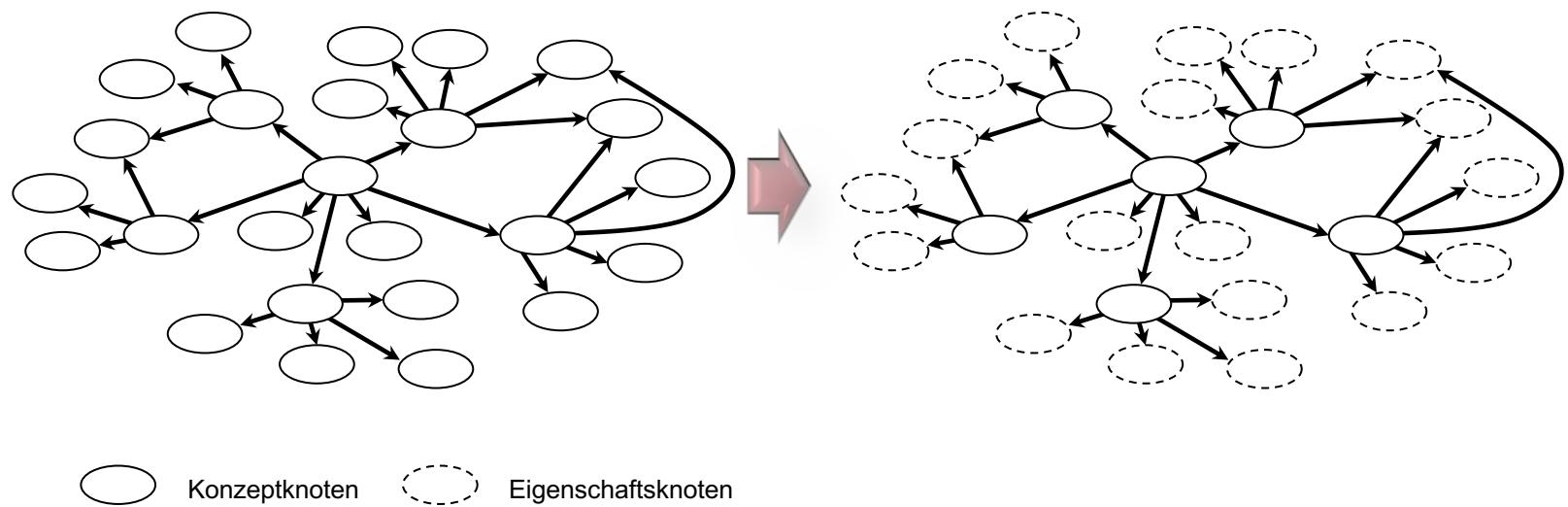
- ▶ Semantisches Netz für die stattfindende Aktion mit dem beteiligten Akteur und dem Empfänger des Objektes

Betrachten wir uns dazu nochmals unser Beispiel

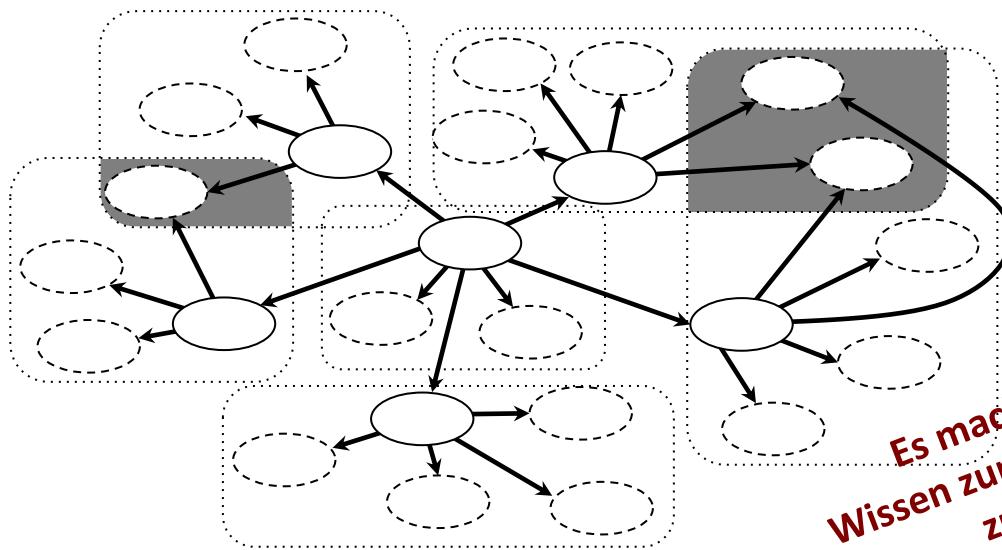


Die Verwendung von Relationen beschreibt die Semantik eines Objektes

- Semantische Netze machen keinen Unterschied, ob der Begriff im Knoten ein Konzept oder seine Eigenschaft repräsentiert
 - ▶ Interpretation, was Konzept und was Eigenschaft ist, kann nur durch nähere Betrachtung der verwendeten Begriffe erkannt werden



Es wäre sinnvoll, Konzepte mit ihren beschreibenden Eigenschaften als semantische Einheiten zusammenzufassen



○ Konzeptknoten ○ Eigenschaftsknoten ● Gemeinsame Beschreibungselemente

- Gemeinsam Beschreibungselemente erlauben allgemeiner geltende Eigenschaften zu generieren, die verallgemeinernde Konzepte besitzen

Wenn man komplexe Probleme lösen möchte, macht es durchaus Sinn das Wissen in kleiner Portionen zu unterteilen

- Regeln sind eine Alternative: Sie können in ihrer einfachen Form verkettet werden, um komplexe Probleme zu lösen
- Daneben gibt es eine weitere deklarative Form der Wissensrepräsentation:
Ein **Schema** bezeichnet das deklarative Wissen von Menschen zu einem bestimmten Konzept, etwa über ein Individuum, ein Objekt, ein Ereignis oder eine Handlung
 - ▶ Kognitive Struktur, die in sich selbstständig ist und damit die Verarbeitung seiner Informationen bestimmt
 - ▶ Ist in seiner Beschreibung abgrenzbar und funktioniert als Einheit, was wiederum bedeutet, dass seine definierenden Bestandteile zur gleichen Zeit aktiviert werden
 - ▶ Sie lassen sich durch Reize von außen oder über Beziehungen zu anderen Schemata von innen aktivieren

Schemata verknüpfen mentale Vorstellungen mit einer Realdefinition einer symbolischen Beschreibung

- 1** Schemata sind **kognitive Strukturen**, in denen allgemeines Wissen im Gedächtnis repräsentiert ist (Wissen über typische Zusammenhänge in einem Realitätsbereich ist in Schemata organisiert)
- 2** Schemata repräsentieren Wissen unterschiedlichster Inhaltsbereiche
- 3** Schemata enthalten sowohl episodisches als auch generisches Wissen
- 4** Schemata haben sowohl eine Struktur-, als auch eine Prozesskomponente
- 5** Schemata können ineinander eingebettet sein
- 6** Schemata weisen Variablen auf, die unterschiedliche Werte annehmen können

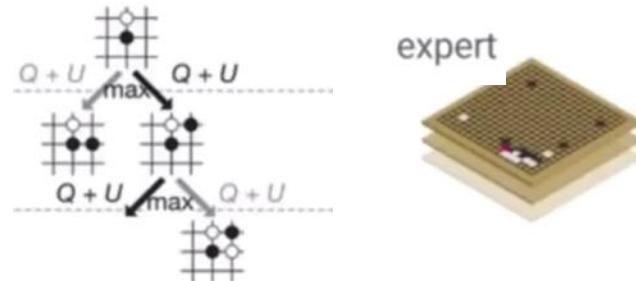
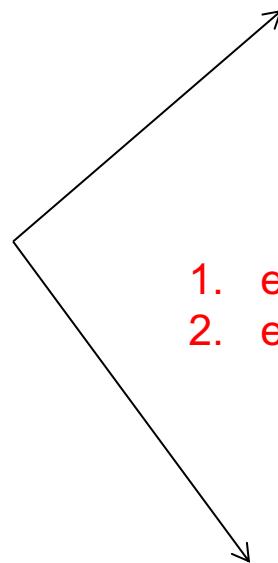
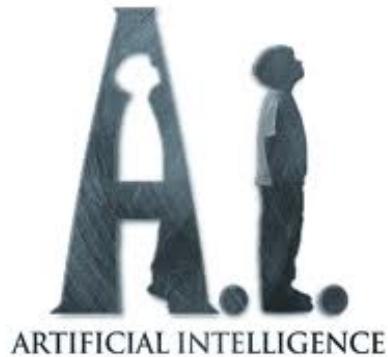
(Kon-)Text wirkt wie ein Schlüssel zu bestehenden mentalen Vorstellungen

■ Beispiel:

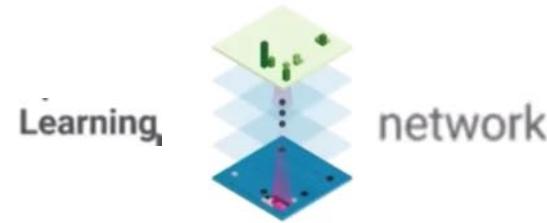
“ Als Rudi den Schlüssel in die Haustür steckt, wird er bereits stürmisch von Cindy und Bert empfangen. Nachdem er seinen Mantel abgelegt hat, zerren ihn die Kinder ungeduldig ins Wohnzimmer. Dort brennen bereits die Kerzen, Weihnachtslieder erfüllen den Raum und es duftet nach Zimtgebäck. »

- ▶ Erfahrungen, Pläne oder Wünsche bzw. Bilder (Imaginationen) werden in unserem mentalen Modellen abgerufen
- ▶ Es entsteht komplexe Szene in der Vorstellungswelt unseres Gehirns
- ▶ **Wir erkennen Details, die im Text nicht (explizit) erwähnt sind**

highly efficient complex (1) search and (2) knowledge **representation**



1. enables humans to easily encode their knowledge
2. enables computers to easily handle the knowledge



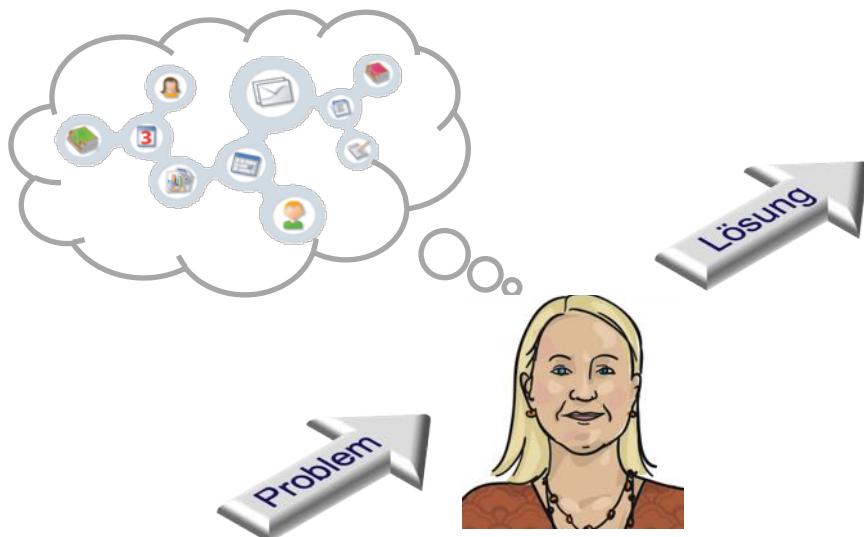
(3) learning: statistical analysis and optimization

Schemata spiegeln die Gedanken aus ontologischen und prädikativen Strukturen wider

- Minsky (1974): „A framework for representing knowledge“

- ▶ **Frames** als Grundlage für ein intuitives Denkmodell

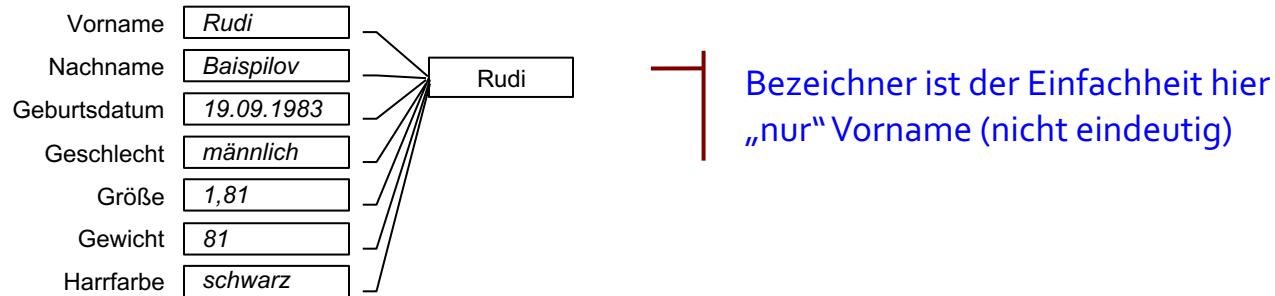
Frames sind nichts anderes als Schemata, die in ihrem Zusammenspiel ein komplexes Netzwerk mentaler Konzepte repräsentieren



- ▶ Knoten des Netzwerks repräsentieren die Schemata und die Kanten die Wechselwirkungen zwischen diesen
 - ▶ Frame-basierte Programmierung hatte sehr großen Einfluss auf das Design von Betriebssystemen, Benutzerschnittstellen und Anwendungssoftware

Frames repräsentieren mentale Konzepte unseres Denkens

- Konzepte sind durch Einbeziehung visueller oder auditiver Sinneseindrücke von einander differenzierbar
 - ▶ Uniforme Konzeptstruktur, die verschiedene Ausprägungen ihrer begrifflichen Bedeutung beschreiben, z.B.:

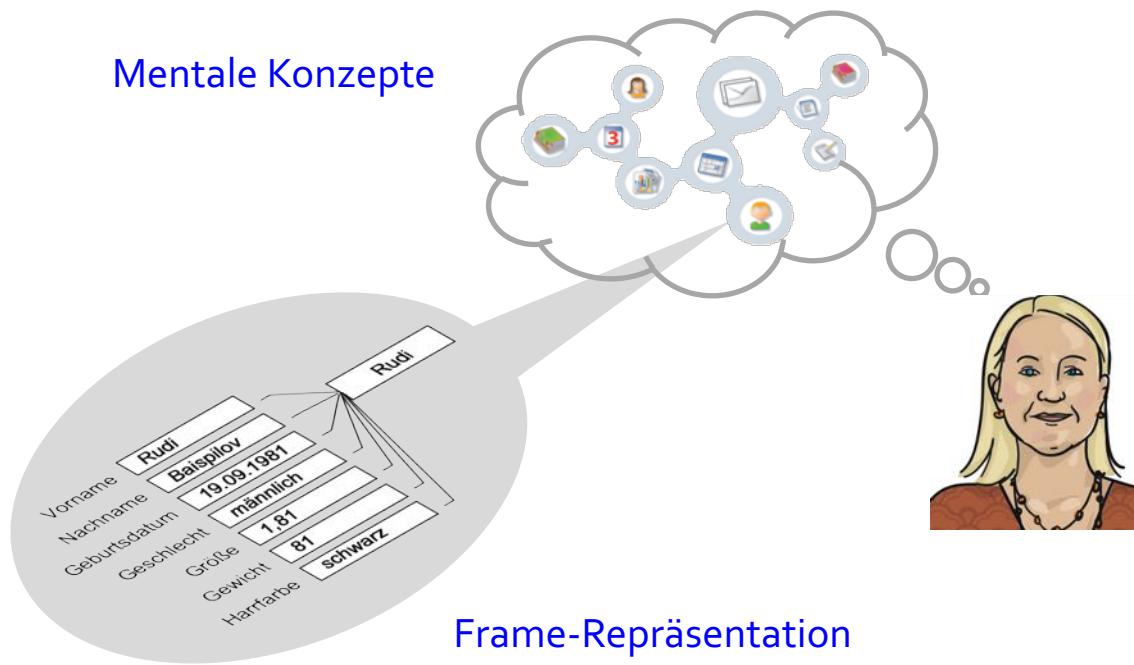


- ▶ Frame hat Bezeichner (eindeutiger Name, eindeutige ID oder URI)
- ▶ Frame hat Eigenschaften, die in Fächern (slots) organisiert sind,
- ▶ Fächer beschreiben Ausprägungen eines bestimmter Eigenschaften in Form von Attribut-Wert-Paaren

Mentale Konzepte unseres Denkens werden in Frame-Darstellung von Schemata transformiert

2

Durch Vernetzung mit anderen Frames werden Zusammenhänge und Wechselwirkungen und damit komplexe Sinneseindrücke dargestellt



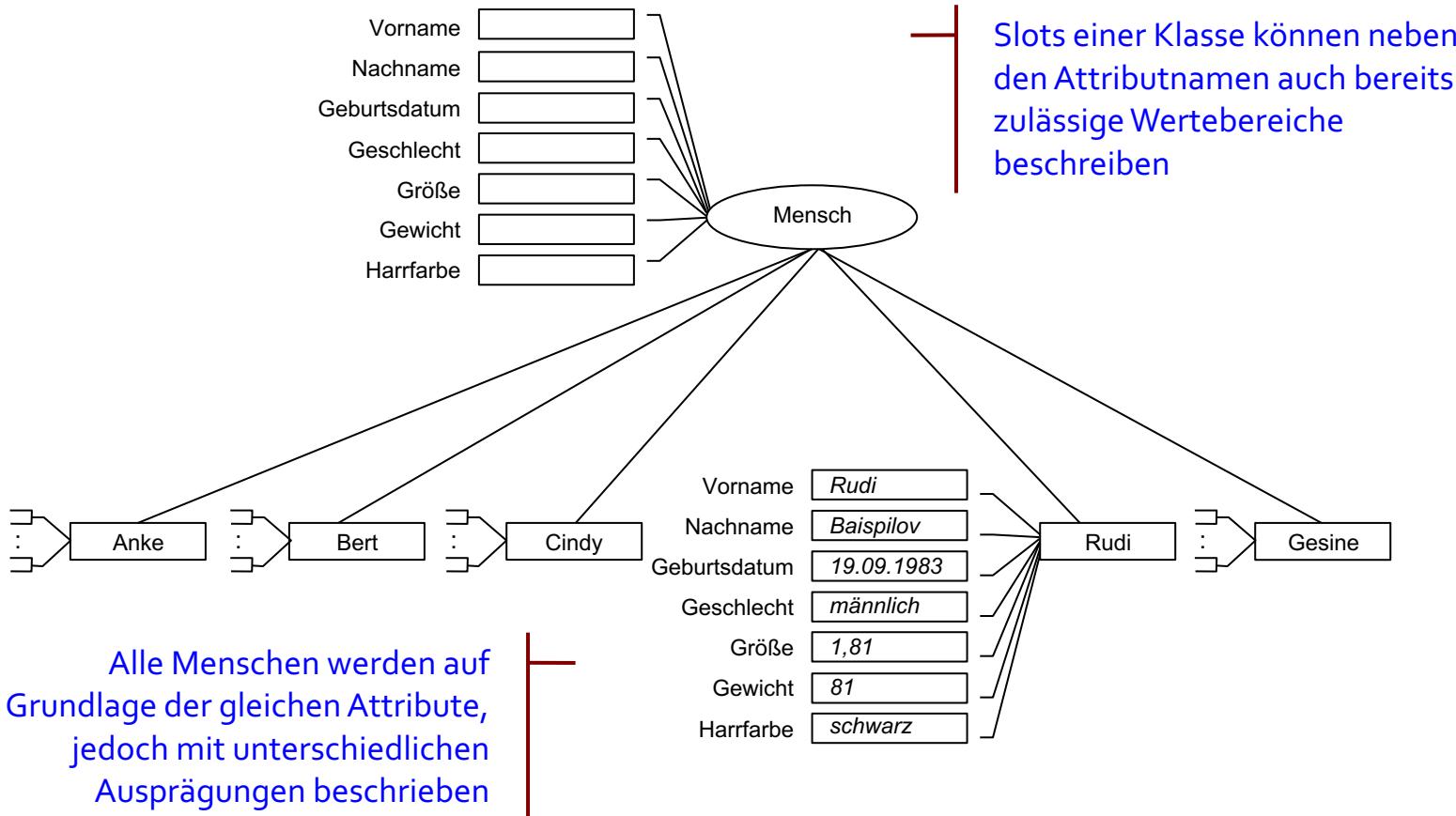
1

Frames sind abstrakte Informationseinheiten, mit deren Hilfe sich stereotype Wissen strukturell beschreiben lässt.

Die Darstellung in Frames kann unterschiedliche Abstraktionsgrade haben

- Eigenschaften (Menge der Attribute) eines Frames passen für unterschiedliche Konzepte gleicher Art
- Es wäre also sinnvoll, ein extra Frame einzuführen, welches die gemeinsamen Eigenschaften der beschriebenen Konzepte als Schablone zur Verfügung stellt
 - ▶ Solche „Musterframes“ nennt man Klassen
- Eine **Klasse** ist also ein „wertloses“ Frame, das für eine Menge individueller Exemplare angibt, welche Eigenschaften diese teilen
 - ▶ Exemplare heißen **Instanzen**
- Werte von Instanzen müssen explizit belegt werden oder nehmen default-Werte an, die in den Klassen angegeben sind
- Eine Klasse kann beliebig viele Instanzen erzeugen

Zusammenfassung von gleichartigen Frames zu einer Klasse nennt man Klassifikation



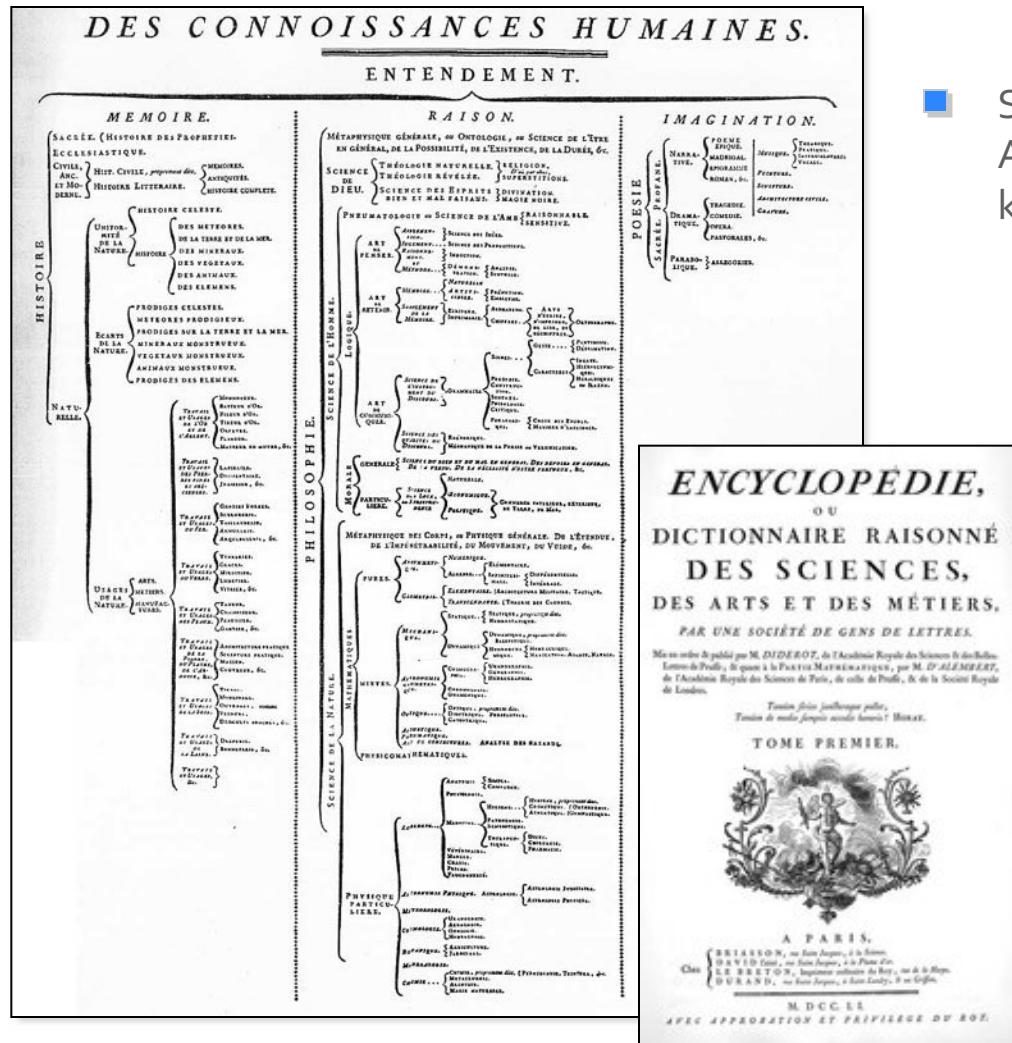
Klassen und Taxonomien sind ein typisches Mittel zur Beschreibung von Wissen



■ Beispiel: Tiger (*Panthera tigris tigris*)

tier	no.	denotation	FD	description
system	1	Lebewesen		(it does not appertain for the actual taxonomy)
main kingdom	2	Eukaryotae		
kingdom	1	Animalia		animals; heterotrophe nutation;
sub-kingdom	2	Metazoa		multicellular animals
part-kingdom	3	Eumetauoa		genuine multicellular animals
subsection	4	Bilateria		two side animals; Mesoderm is available; acting moving
main branch	3	Deuterostomia		new mouths; mouth of the early embryo becomes the new anus
tribe	1	Chordata		sp. cord ani.; Chorda dorsalis; elast. wand, intestine, RM; Notochord
sub-tribe	2	Vertebrata		vertebrates; segmented WS; inner axis skeleton; KRS
superclass	3	Tetrapoda	HAWORTH 1825	land vertebrates; four legs;
class	1	Mammalia	LINNEAUS 1758	mammals; genuine hairiness; milk glands;
subclass	2	Eutheria	GILL 1872	plazenta; genuine chorion- allantiosplacenta; trophoblast
order	1	Carnivora	BOWDICH 1821	carnivore; sole-, half sole- or toe-walker; large canine teeth
sub-order	2	Fissipedia		land predators (seals are Pinnipedia)
upper family	2	Cynofeloidea	HOUGH 1953	cat and dog breeds
family	1	Felidae	GRAY 1821	cats; binokular; former tree animals
lower family	2	Felinae	TROUESSART 1885	genuine cats; completely retractable claws;
gender group	3	Pantherini	POCOCK 1917	big cats
group	1	Panthera	OKEN 1816	genuine big-cats
kind	1	tigris	LINNEAUS 1758	tiger; (from the old Persian: it means arrow)
sub-king	1	tigris	see *)	nominatform; indian tiger;

Klassifikation hat eine lange Tradition



System von hierarchisch strukturiertem Allgemeinwissen (genealogy of human knowledge) in *Encyclopédie*

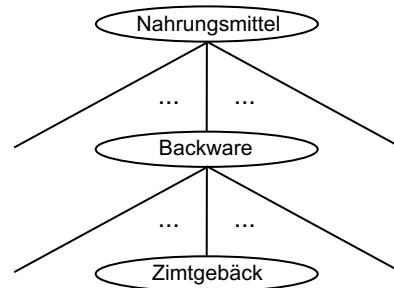
Rond d'Alembert (1717 - 1783) und Diderot (1713 – 1784)

Klassen stellen ein Basisvokabular zur Beschreibung von Wissen zur Verfügung

Ein Klassensystem ist eine Verkettungen von Unter- und Oberbegriffen

■ Beispiel:

„Der Begriff *Nahrungsmittel* ist eine Verallgemeinerung für *Backwaren* was wiederum eine umfassendere Bezeichnung für *Zimtgebäck* darstellt“

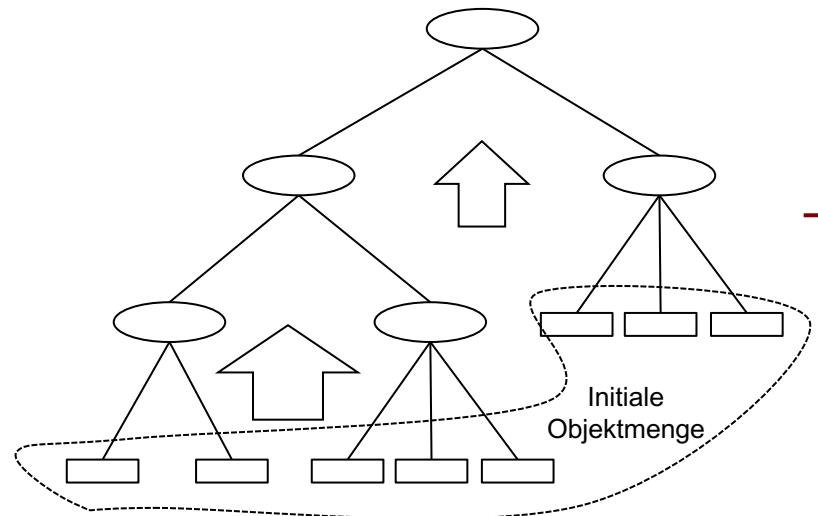


Nahrungsmittel ist ein Oberbegriff (Hyperonym) von *Backwaren* und *Zimtgebäck* ist ein Unterbegriff (Hyponym) von *Backwaren*

- Interpretation des Satzes führt zu einer Hierarchie von Klassen (**Taxonomie**) mit unterschiedlichem Abstraktionsgrad
 - ▶ Je allgemeiner ein Objekt ist, das beschrieben wird, desto unterspezifizierter bleiben seine Eigenschaften
 - ▶ Je spezieller man Objekte betrachtet, umso mehr werden sie mit Daten angereichert

In Framesystemen gibt es verschiedene Vorgehensweisen, solche Taxonomien aufzubauen

- **Generalisierung:** „Vom Speziellen zum Allgemeinen“

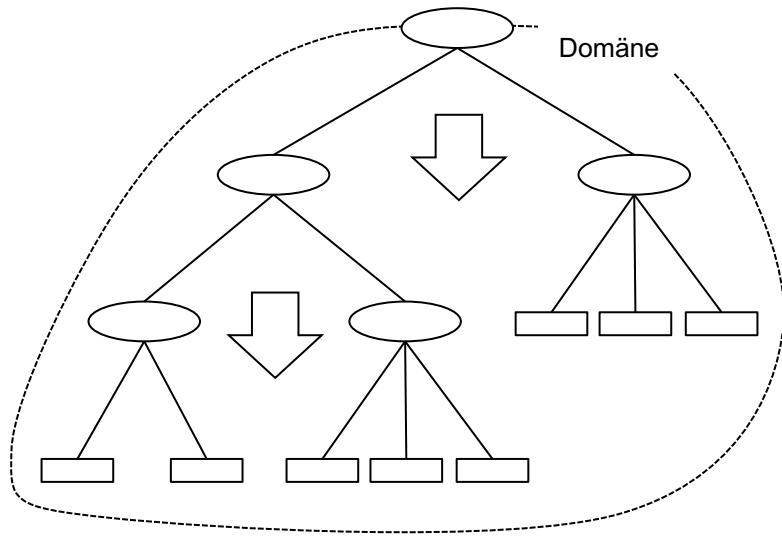


Mit einer Generalisierung verbunden ist die Extraktion von gemeinsamen Eigenschaften und deren Verlagerung in die Oberklasse

- ▶ Objekte werden verschieden klassifiziert
- ▶ Resultierenden Klassen werden sukzessive abstrahiert und während des Verallgemeinerungsprozesses in Taxonomien zusammengefasst
- Klassen besitzen demnach eine **Oberklasse**, wenn eine Verallgemeinerung für sie existiert



- **Spezialisierung:** „Vom Allgemeinen zum Speziellen“



Für Spezialisierung ist es wichtig, für die Modellbeschreibung relevante Unterscheidungsmerkmale zu identifizieren

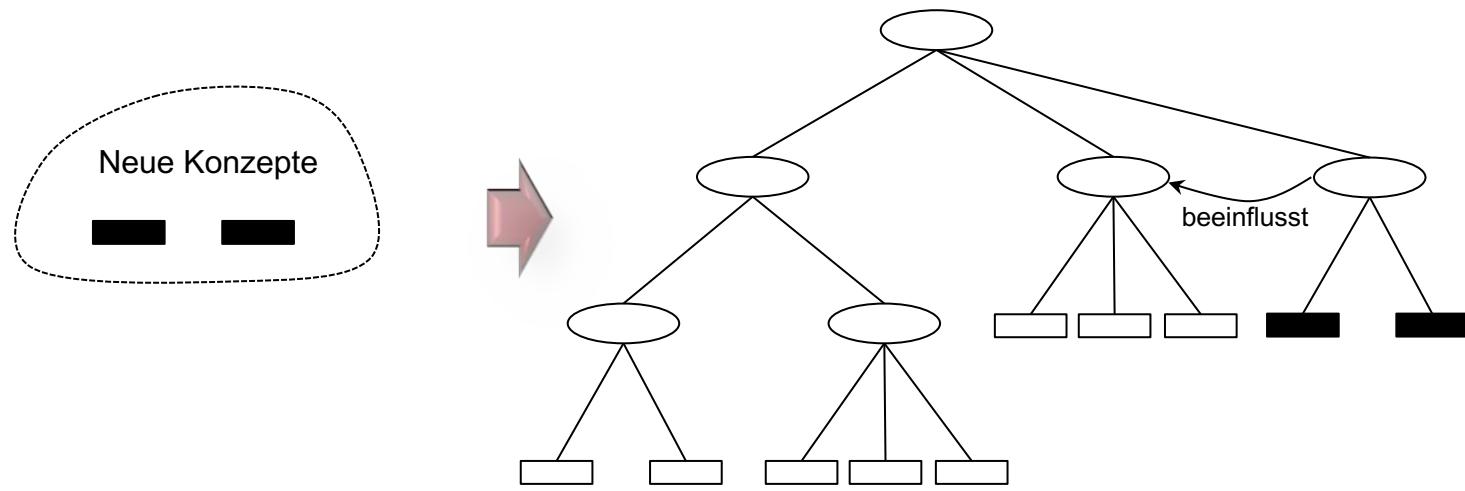
- ▶ Schrittweise Verfeinerung der Klassen durch Verwendung von Unterbegriffen
- ▶ Ergänzung der resultierenden **UnterkLASSE** mit weiteren, unterscheidenden Eigenschaften
- Klassen haben eine UnterkLASSE, wenn es eine Spezialisierung davon gibt

Für beide Ansätze zur Erstellung von Taxonomien gibt es einige wichtige Beobachtungen

- Generalisierung findet oft dann statt, wenn eine begrenzte Menge realer Konzepte vorliegt, beispielsweise bei Klassifikationsproblemen
 - ▶ Konzepte werden zunächst untersucht und aufgrund differenzierender Merkmale in Klassen unterteilt
 - ▶ Die entstehenden Klassen werden entsprechend genauso behandelt, so dass nach und nach ein Klassenbaum entsteht
- Spezialisierung ist ein typisches Instrument zur Top-Down-Beschreibung von Domänen, bei der Klassen bereits initial feststehen und sich systematisieren lassen, um die Konzepte der Domäne aufgrund gemeinsamer Merkmale zu charakterisieren
- Falls zu einem späteren Zeitpunkt neue Konzepte hinzu genommen werden müssen, so kann dies die bestehende Taxonomie beeinflussen

Einbindung neuer Konzepte in Klassenbaum kann Probleme auslösen

- Einbinden von Konzepten kann unter Umständen dazu führen, dass Veränderungen in der Fächerstruktur der bestehenden Klassen vorgenommen werden müssen



- ▶ Einführung neuer Fächer
- ▶ Verfeinerung bestehender Fächer
- ▶ Überschreiben bestehender Fächer

Abstraktion in Taxonomien bieten einige wesentlichen Vorteile bei der Modellierung

- Manche Eigenschaften müssen nicht in jeder betreffenden Klasse explizit definiert werden, z.B.:
 - ▶ Es genügt zu wissen, dass alle Nahrungsmittel ein Haltbarkeitsdatum haben
- Manche Eigenschaften müssen erst dann beschrieben werden, wenn die entsprechende Instanz erzeugt wurde, z.B.:
 - ▶ Name des Bäckers, der das Zimtgebäck hergestellt hat
- Zwischen Klassen und Unterklassen besteht eine Vererbungsrelation („ist-ein(e)-Relation“)
 - ▶ Es reicht aus, alle einer Klasse innewohnenden Eigenschaften nur einmal, möglichst an der Stelle zu beschreiben, wo sie als allgemeingültig für alle daraus abgeleiteten Spezialisierungen gelten
 - ▶ Klassen erben Eigenschaften von ihrer jeweiligen Oberklasse ohne weiteres Zutun

Vererbung ist eine Strukturbeziehung zwischen Klassen, die auch konstitutiv für alle abgeleiteten Instanzen ist

- Eine Unterklasse ist vollständig konsistent mit ihrer Oberklasse, erweitert diese jedoch um zusätzliche Eigenschaften
 - ▶ Da eine Unterklasse wiederum Oberklasse von weiteren Unterklassen sein kann, sind mehrere Stufen der Vererbung möglich



- Im Unterschied zu Semantischen Netzen sind die Klassenkonzepte entsprechend der Schemadefinition strukturiert, d.h. sie kapseln ihre beschreibende Eigenschaften

Während Klassen abstrakte Denkmuster darstellen, repräsentieren Instanzen real existierende Entitäten

- Unterscheidung durch speziellen Relationstyps „ist-Instanz-von“, z.B.:
 - ▶ Rudi „ist-Instanz-von“ Mensch
 - ▶ Ankes Zimtgebäck „ist-Instanz-von“ Zimtgebäck

Aussage „ist-Instanz-von“ gilt für gesamte Lebensdauer einer Instanz und ist wahr!
- Mit Erzeugung einer Instanz erhält diese gewisse Eigenschaften, z.B.:
 - ▶ Wenn „Ankes Zimtgebäck“ fertig ist, wird in sein Fach Hersteller der Wert „Anke“ eingetragen und in sein Fach Herstellungsdatum das Datum 20.12. eingetragen

```
graph LR; Zimtgebäck --> Anke[Anke]; Zimtgebäck --> Datum[20.12. ...]; Zimtgebäck --> Zutaten[Butter, Eier, ..]
```
- Ausprägungen einer Instanz sind existenzabhängig, z.B.:
 - ▶ Die Herstellung von Zimtgebäck erzeugt auch seine Bestandteile
- Löschung einer Instanz löscht auch deren Komponenten, z.B.:
 - ▶ Wenn das Zimtgebäck aufgegessen wurde, dann auch seine Zutaten

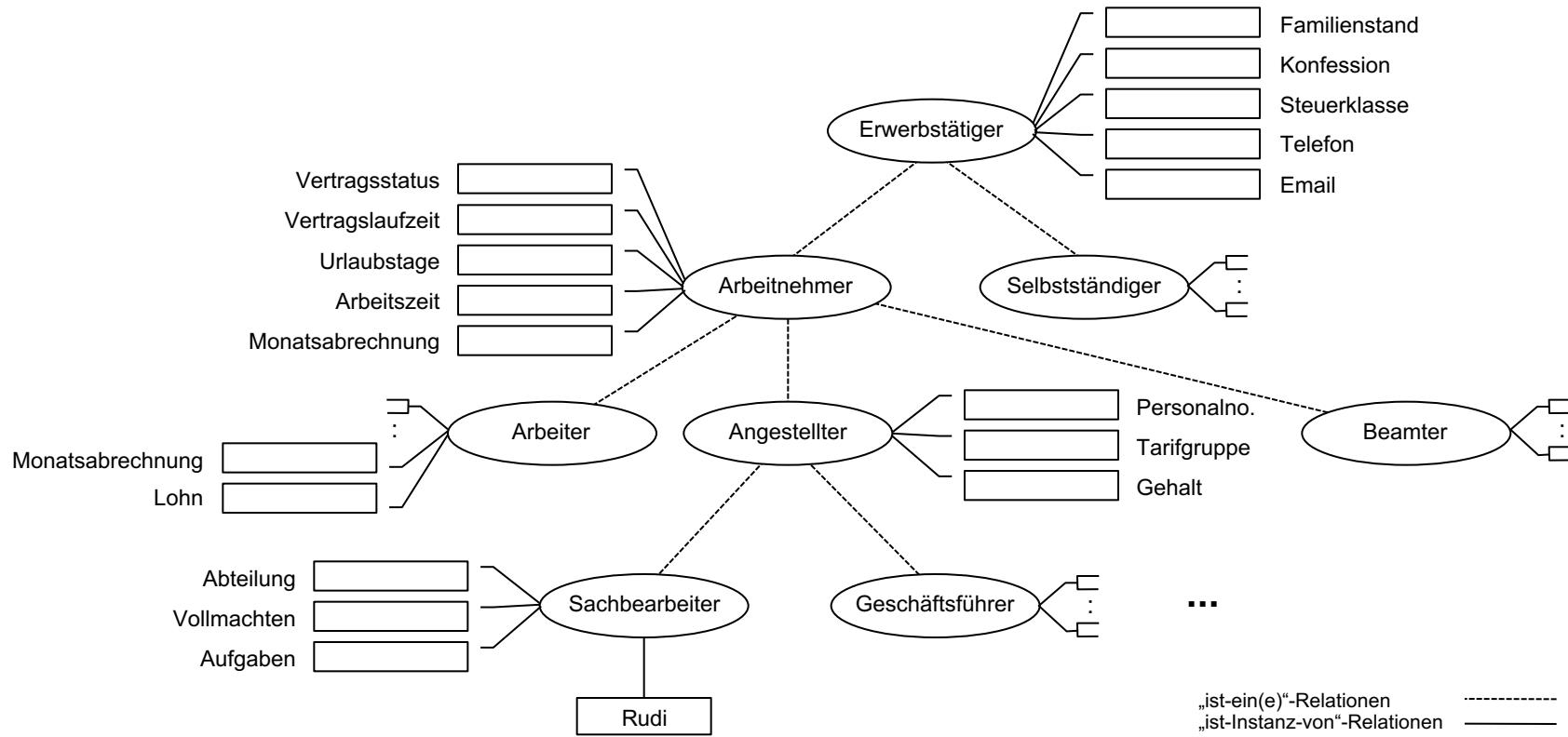
Man spricht in solchen Fällen auch von **echter Aggregation oder Komposition!**

Wenn Instanzen erzeugt werden, lassen sich Eigenschaften nicht immer ohne Weiteres mit Werten belegen

- Einige Eigenschaftsausprägungen ergeben sich durch Messen (Sensordaten), Abfragen (Eingabe) oder durch Zugriff auf existierende Drittquellen (Datenbanken, Internet), z.B.:
 - ▶ Werte für Hersteller und Zutaten von „Ankes Zimtgebäck“ ergeben sich entweder aus der Beobachtung, aus Fragen an Anke, welche Zutaten sie verwendet hat, oder indem wir die Angaben einfach aus Ankes Rezeptsammlung entnehmen
 - Andere Eigenschaftsausprägungen müssen erst berechnet werden:
 - ▶ Addiere zum Fertigstellungsdatum vier Wochen dazu und lege das berechnete Datum dann in das entsprechende Fach der Instanz „Ankes Zimtgebäck“ einträgt
- | Berechnung kann unter Umständen sehr komplex werden!
- **Beachte:** Ein Konzept kann also neben einem deklarativen Teil auch einen prozeduralen Teil enthalten, der gewisse Berechnungen bzw. Operationen ausführt

Bei Erzeugung von Taxonomien wird ein geschlossenes und kontrolliertes Vokabular vorausgesetzt

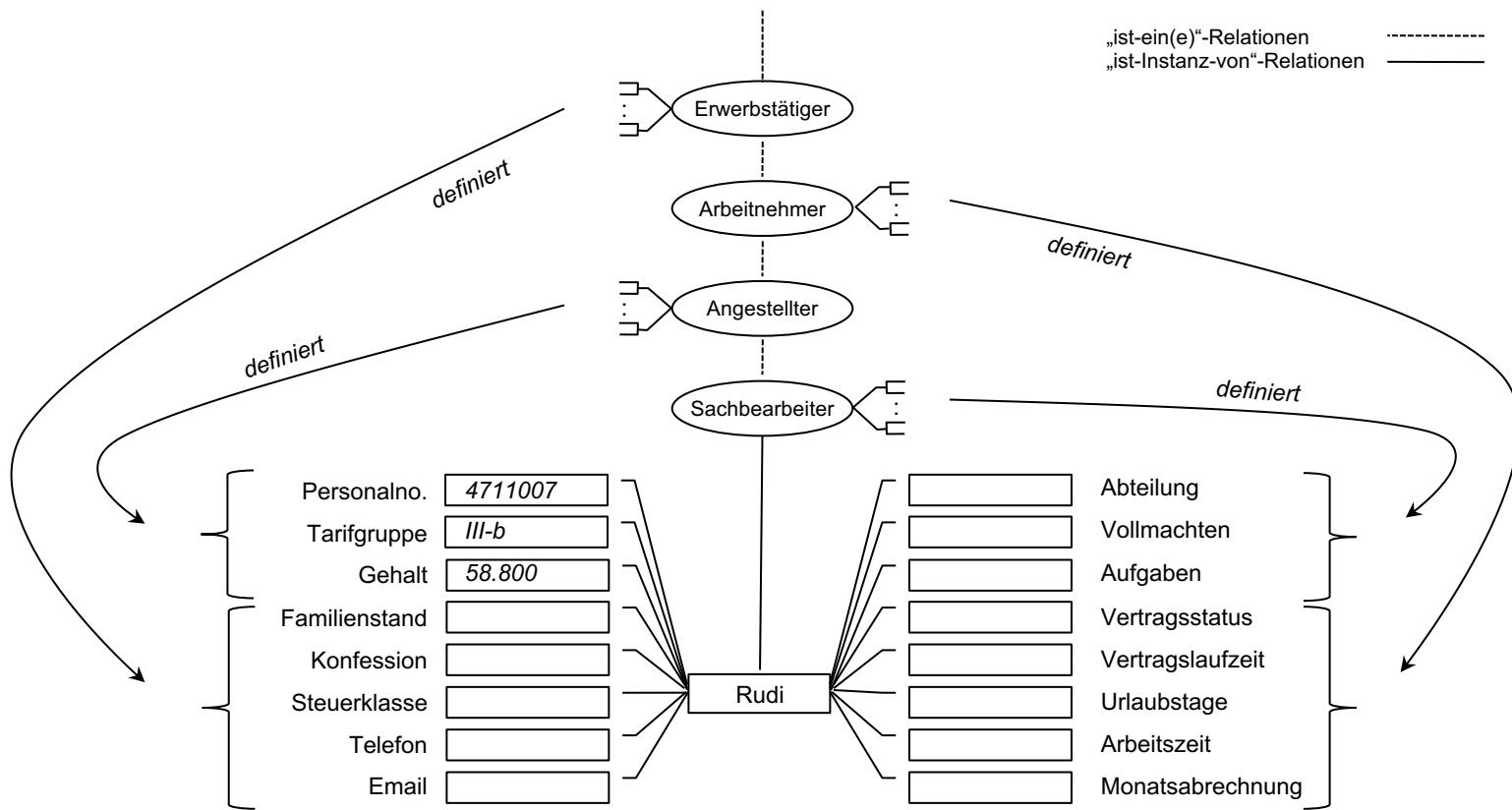
- Verwendung einer Menge von eindeutigen Termen zur Beschreibung einer Domäne, z.B.:

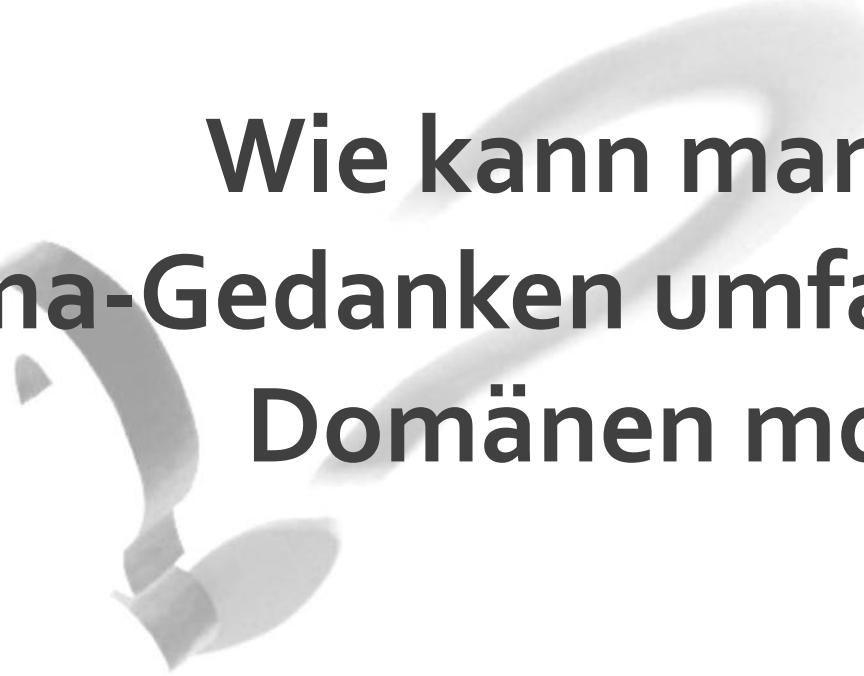


Eigenschaften von Unterklassen können Eigenschaften von Oberklassen neu definieren

- Klassen können Eigenschaften von ihren Oberklassen übernehmen oder neue Eigenschaften definieren
- Passt die verallgemeinernde Eigenschaft der Oberklasse nicht oder gelten für die Unterklassse Besonderheiten, so kann eine Eigenschaft neu definiert werden
 - ▶ Eigenschaft der Unterkasse überdeckt Eigenschaft der Oberklasse
- Man spricht in diesem Fall von einer **Überdeckungsbeziehung** zwischen Unter- und Oberklasse, z.B.:
 - ▶ Unterschiede bei Gehaltsabrechnung von Angestellten und Arbeitern

Eine Instanz besitzt alle Eigenschaften einer Klasse und deren Oberklassen





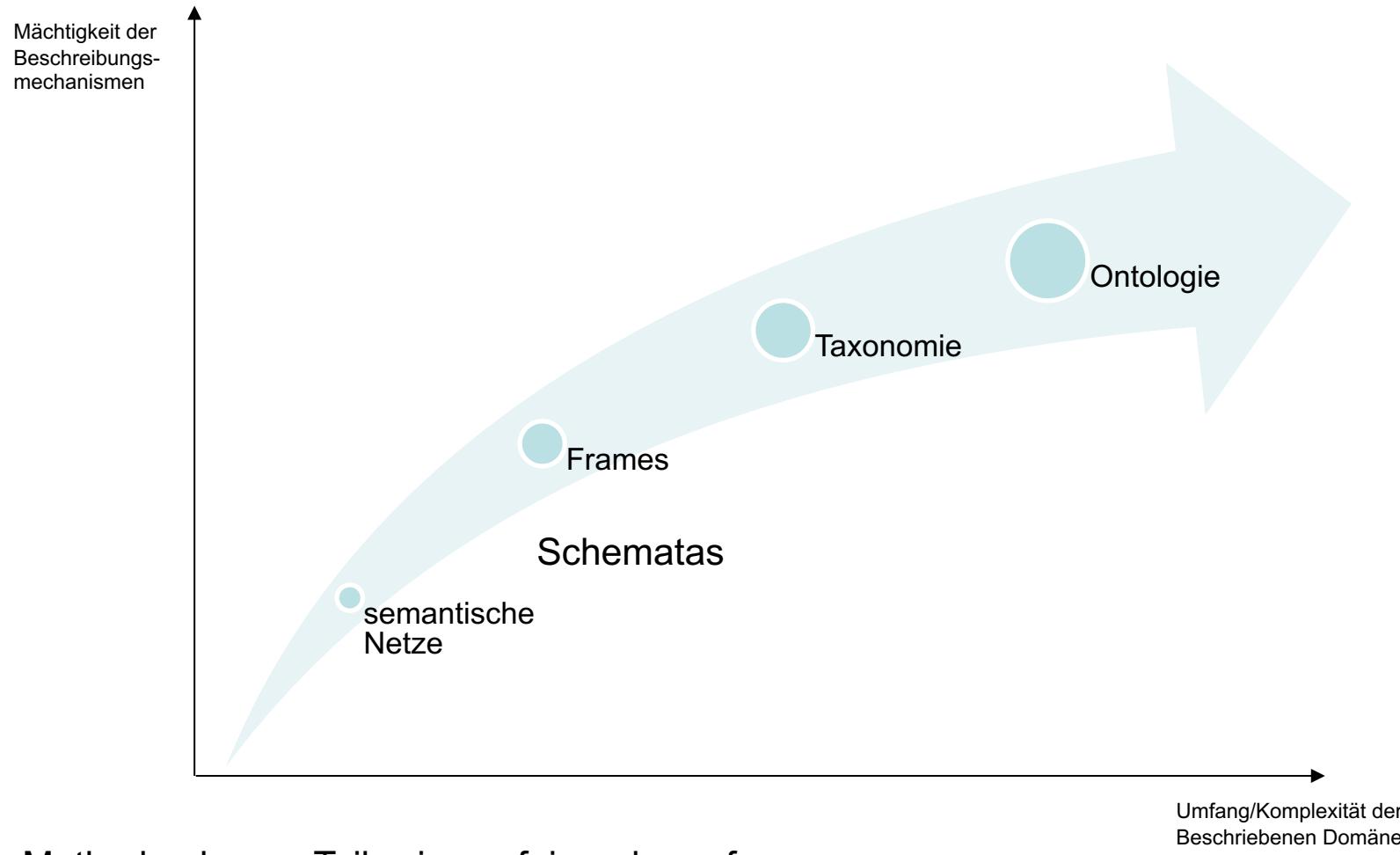
Wie kann man mit dem Schema-Gedanken umfangreiche Domänen modellieren

Grundlage jeder erfolgreichen Kommunikation ist die Verwendung eines gemeinsamen Vokabulars

- Die Ontologie (die Lehre vom Seienden) sucht als Disziplin nach Möglichkeiten, die Grundstrukturen der Realität korrekt und allgemeingültig zu beschreiben
- Definition(sversuch) einer Ontologie aus Sicht der Informatik:
 - ▶ Eine **Ontologie** ist eine formale, explizite Spezifikation einer **gemeinsamen Konzeptualisierung*** | Übertragung einer sozio-technischen Verständnisses von Konzepten ist nicht einfach!

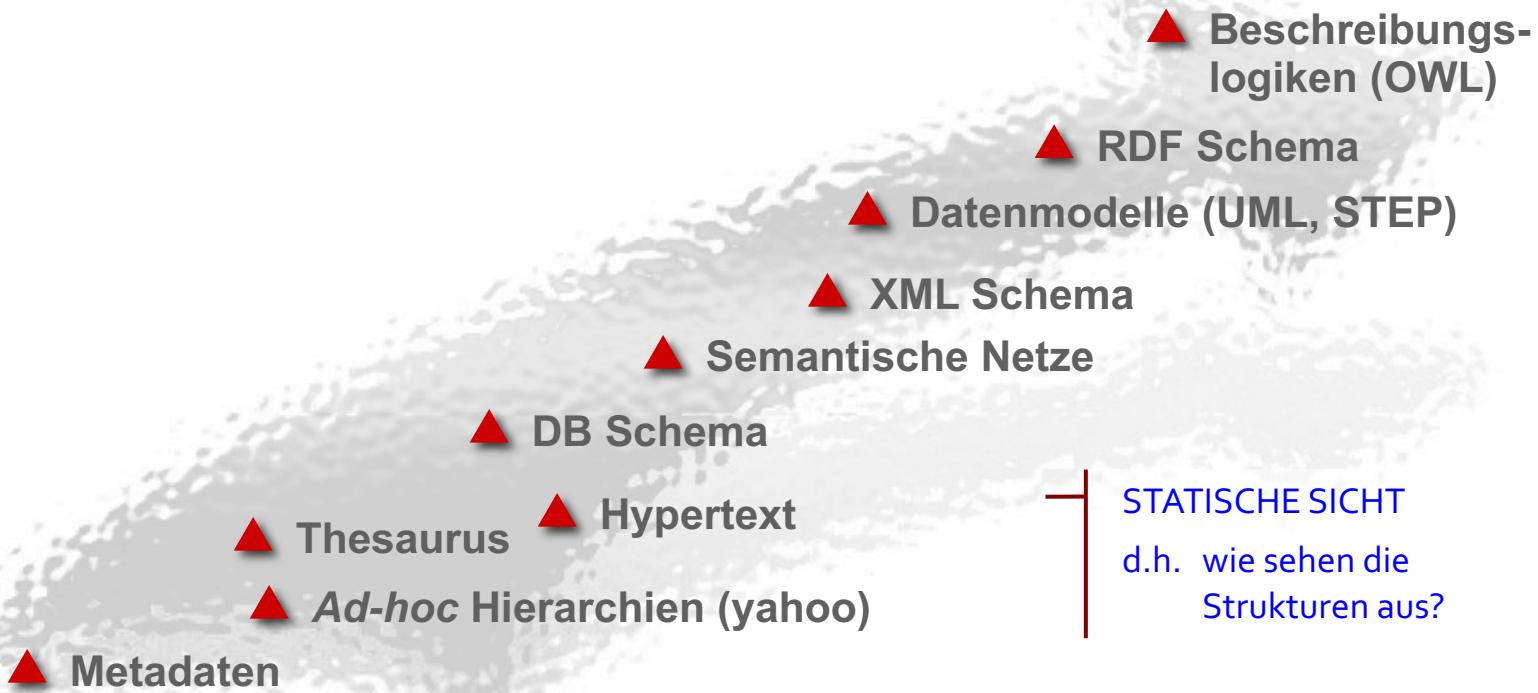
- 1 Nutzung gemeinsamer Symbole und Begriffe im Sinne einer Syntax
- 2 gemeinschaftliche (Ein-)Verständnis bezüglich deren Semantik
- 3 die Klassifikation der Begriffe in Form einer Taxonomie
- 4 die Vernetzung der Begriffe mit Hilfe assoziativer Relationen, wobei
- 5 Regeln und Definitionen darüber bestehen müssen, welche Relationen sinnvoll und erlaubt sind

Das große Ganze



Methoden bauen Teilweise aufeinander auf

Ontologien lassen sich mit fast allen Arten der Strukturierung sehr weit fassen



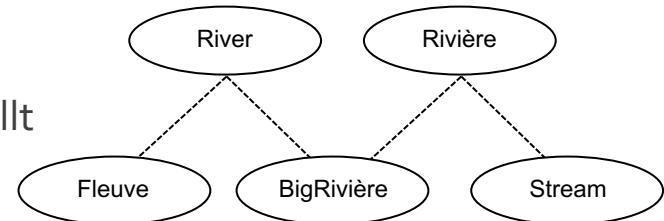
Entwickeln einer Ontologie ist ein kooperativer Prozess, um das Wissen vieler Menschen auf Normalform herunter zu brechen

- Akzeptanz der Begriffe (Konzepte) durch Gruppe beinhaltet ein gemeinsames Verständnis
 - ▶ Oft werden Ontologien für ein gewisses Interessensgebiet formuliert, also für begrenzten Ausschnitt der Welt (*Closed World Assumption*) und ggf. mit ausgewählten Zielsetzungen

Im Gegensatz zur Philosophie sprechen wir hier von **mehreren unterschiedlichen** Ontologien, je nach Domäne oder Zielsetzung!

Das Auffinden gleicher Terme (insbesondere bei mehreren Sprachen) ist ein nicht-triviales Problem

- Beim Übergang von einer deutschen zu einer englischen Ontologie müssen Begriffe übersetzt werden
 - ▶ Übersetzung ist im allgemeinen nicht bijektiv
Beispielsweise lässt sich das englische Verb ‚know‘ ins Deutsche sowohl mit ‚kennen‘ und als auch mit ‚wissen‘ übersetzen.
 - ▶ Übersetze der Terme ‚river‘ und ‚stream‘ vom Englischen ins Französische
Im Englischen ist das Differentiae der Terme die Größe
Im Französischen unterscheiden sich ‚fleuve‘ und ‚rivière‘ dadurch, dass ein ‚fleuve‘ ins Meer fließt und ein ‚rivière‘ ein Zustrom zu einem anderen Fluss ist
- Demnach ist ‚fleuve‘ ein Subtyp von ‚river‘. ‚stream‘ lässt sich als Subtyp von ‚fleuve‘ auffassen und ein ‚river‘ der nicht ins Meer fließt wäre dann eine Kreuzung aus ‚river‘ und ‚rivière‘. Die Abbildung stellt den Sachverhalt nochmal grafisch dar



Entwickeln einer Ontologie ist ein kooperativer Prozess, um Wissen vieler Menschen auf Normalform herunter zu brechen

- Akzeptanz der Begriffe (Konzepte) durch Gruppe beinhaltet ein gemeinsames Verständnis
 - ▶ Oft werden Ontologien für ein gewisses Interessensgebiet formuliert, also für begrenzten Ausschnitt der Welt (*Closed World Assumption*) und ggf. mit ausgewählten Zielsetzungen
- Im Gegensatz zur Philosophie sprechen wir hier von **mehreren unterschiedlichen** Ontologien, je nach Domäne oder Zielsetzung!
- Konzeptualisierungen der realen Welt lassen sich mit Hilfe der bisher kennengelernten Wissensrepräsentationsformen beschreiben
 - ▶ **Klassen** beschreiben Begriffskategorien
 - ▶ **Relationen** stellen die Wechselwirkungen zwischen den Klassen dar
 - ▶ **Regeln** (oder Axiome) werden benutzt, um Gegebenheiten der Domäne zu beschreiben, die immer wahr sein müssen
 - ▶ **Instanzen** repräsentieren real existierende Elemente der Domäne

Je nach Anwendungsgebiet und Generalität gibt es unterschiedliche Typen von Ontologien

- Domänenontologie
 - ▶ Eine Domänenontologie hat die Aufgabe das Wissen, welches in einem bestimmten Bereich Anwendung findet zu kapseln, beispielsweise im Bereichen wie Elektronik, Medizin, Mechanik
- Metadaten-Ontologie
 - ▶ Eine Metadaten-Ontologie wird verwendet, um ein Vokabular bereit zu stellen, mit dem es möglich ist, den Inhalt von on-line Informationsmaterial zu beschreiben.
- Allgemeine Ontologie
 - ▶ Soll das generelle Wissen über die Welt abbilden. Dabei werden Grundbegriffe wie Zeit, Raum, Zustand und Ereignis bereitgestellt. Die daraus resultierenden Ontologien sind dann losgelöst von ihrer ursprünglichen Domain vielfältig anwendbar (!?)

Es gibt auch noch andere Ontologieformen, wie z.B. Aufgabenontologien!

Im Projekt (Open-)CYC wird versucht, Allgemeinwissen methodisch zu formalisieren und verfügbar zu machen

- Lenat startete 1984 mit einem Budget von \$ 50 Mio. das bisher größte reine KI-Projekt

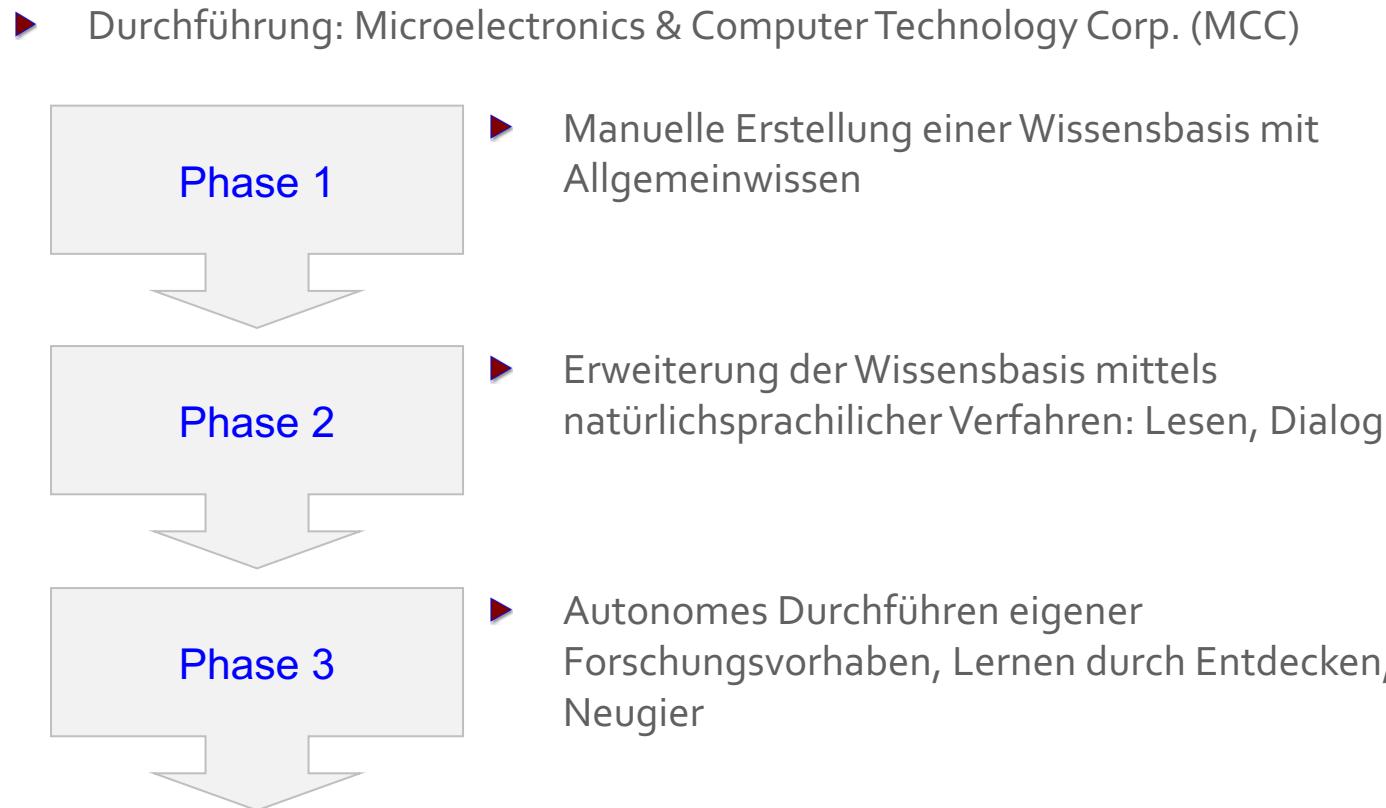


Abbildung von Allgemeinwissen stellt große Herausforderung dar

- Geeignete Darstellungen und eine Strategie zur Erfassung und Formalisierung erforderlich
 - ▶ Im Projektverlauf traten zahlreiche Probleme auf, die eine Veränderung der verwendeten Methodik nach sich zog, z.B.:
 - Zeitabhängige Aussagen
„Gerhard Schröder ist der neue deutsche Bundeskanzler“
 - Orts- und Zeitabhängige Aussagen
„Die Erde bebt“
- Mit Wachsen der Wissensbasis ist es schwierig, Widersprüche und Inkonsistenzen in Wissensbasis sind zu vermeiden

Um Probleme zu vermeiden kam das Mikrotheoriekonzept zur Anwendung

- Partitionierung der Wissensbasis in Themengebiete und Erstellung einer Mikrotheorie pro Themengebiet
 - ▶ Globale Inkonsistenzen erlaubt
 - ▶ Lokale Inkonsistenzen innerhalb einer Mikrotheorie verboten
 - Es ist unmöglich, formal zu zeigen, dass ...
 - 1 Jede Mikrotheorie *vollständig* ist
 - 2 Alle darin enthaltenen Aussagen *korrekt* sind
 - 3 Alle notwendigen Mikrotheorien *vorhanden* sind
 - Der geschätzte Aufwand von 200 Personenjahren in dem Projekt wurde um ca. 150% überschritten
- Dadurch ergaben sich andere Probleme bzgl. Überschneidungen und Redundanz

Für das Füllen der Wissensbasis hat sich folgende Vorgehensweise bewährt

- 1** Auswahl des zusammenhängenden Themengebiets
- 2** Beschreibung in Form von Freitext
- 3** Selektion der Begriffe (Konzepte) und der damit verbundenen Aussagen
- 4** Formalisierung und Kodierung in Sprache CycL
- 5** Einfügen in Wissensbasis
- 6** Stichprobenartige Kontrolle der Korrektheit und Vollständigkeit

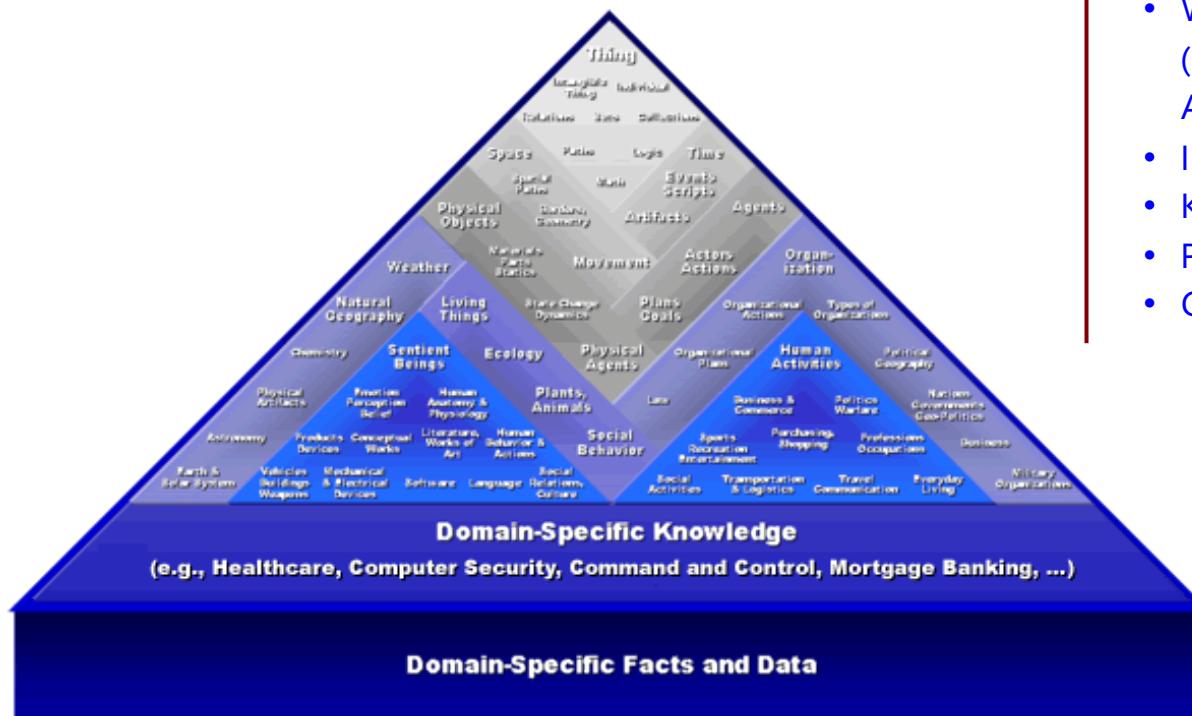
Trotzdem kann man von einem Erfolg des Projektes reden

- Heute existieren drei verfügbare Versionen

- ▶ Cyk (kommerziell, von Cycorp angeboten)
- ▶ ResearchCyk (für Forschungszwecke)
- ▶ OpenCyc (Open Source, <http://www.cyc.com/opencyc>)

Enthält Untermenge der Cyc-Funktionalität

- Wissensbasis in CycL
(6.000 Begriffe, 60.000 Aussagen)
- Inferene Engine
- Knowledge Base Browser
- Programmierschnittstellen
- CycML



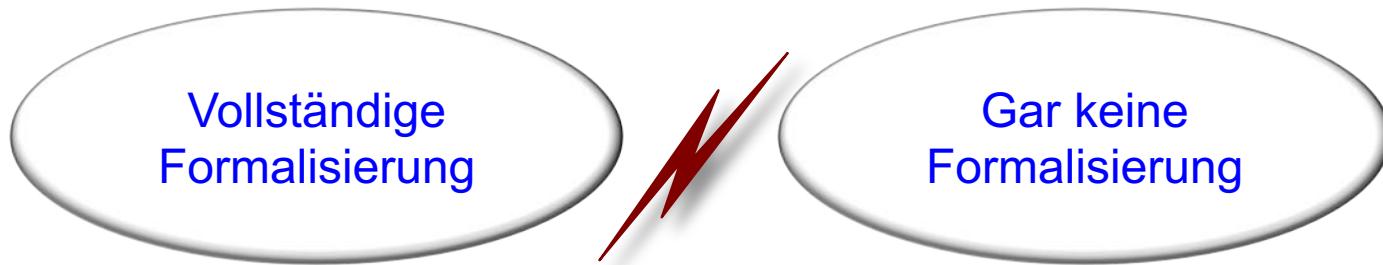
Wie kann man eine Ontologie für eine bekannte Domäne entwickeln

Betrachten wir uns einen konkreten Ontologie-Entwicklungsprozess

- 1** Bestimme die Domäne und das Anwendungsfeld der Ontologie
- 2** Denke an Wiederverwendbarkeit existierender Ontologien
- 3** Benenne wichtige Terme und nehme sie in die Ontologie auf
- 4** Definiere die Klassen und die Klassenhierarchie
- 5** Definiere die Eigenschaften der Klassen
- 6** Definiere die Restriktionen (Regeln) für die Eigenschaften
- 7** Kreiere Instanzen

Ontologie ist formales Modell einer Domäne zur Verbesserung der sozio-technischen Kommunikation zwischen Akteuren

- Es besteht eine Diskrepanz beim Grad der Formalisierung



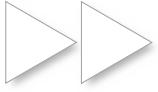
- ▶ Qualitativ gute Transkription von informellen Wissen (z.B. Textdokumente) in formale Repräsentation ist in absehbarer Zukunft nicht möglich
- ▶ Durch Fehlen jeglicher Ansatzpunkte wie Verzeichnisstruktur, Hyperlinks oder Layoutmerkmale wird relevantes Wissen nahezu unauffindbar

Ontologien bestehen aus verschiedenen Komponenten

■ Beispiel:

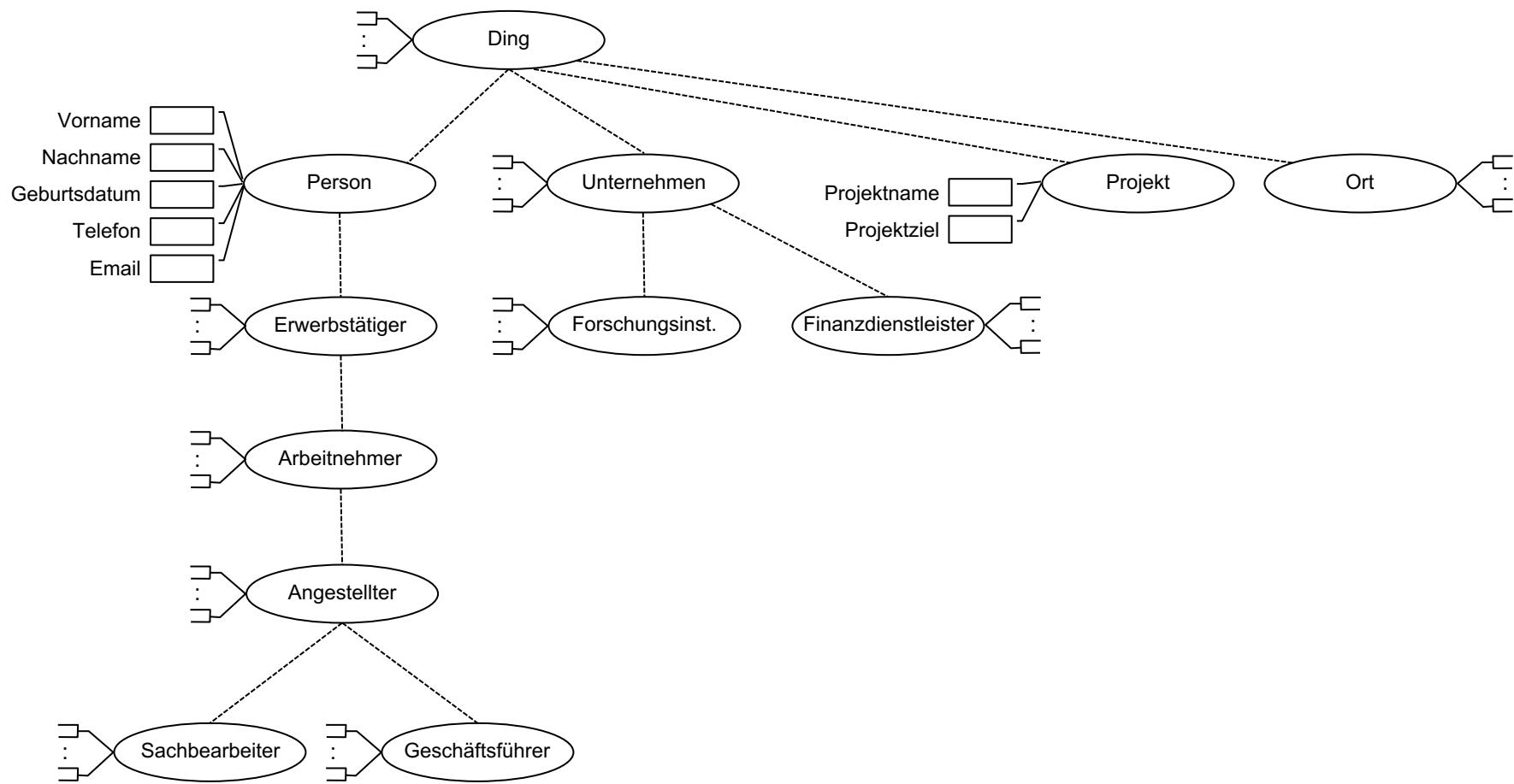
„Zimtgebäck bezeichnet ein Objekt, das ein Nahrungsmittel darstellt. Genauer gesagt gehört es zu Backwaren, die von Menschen hergestellt und begrenzt haltbar sind. Wann immer es dem Prozess des Essens unterzogen wird, existiert es nicht mehr. Das Zimtgebäck, das an Heiligabend bei Rudi im Wohnzimmer steht, hat Anke vor vier Tagen gebacken. Sie verwendete dabei Zutaten wie ...“

- **Klassen** (Konzepte) sind in Ontologien als Taxonomien organisiert, die sich Eigenschaften vererben
 - ▶ z.B. Lebensmittel ist Oberbegriff von Backwaren ist Oberbegriff von Kuchen
- **Instanzen** stehen für reale Elemente einer Domäne
 - ▶ z.B. das Zimtgebäck, das Anke am 20.12. gebacken hat

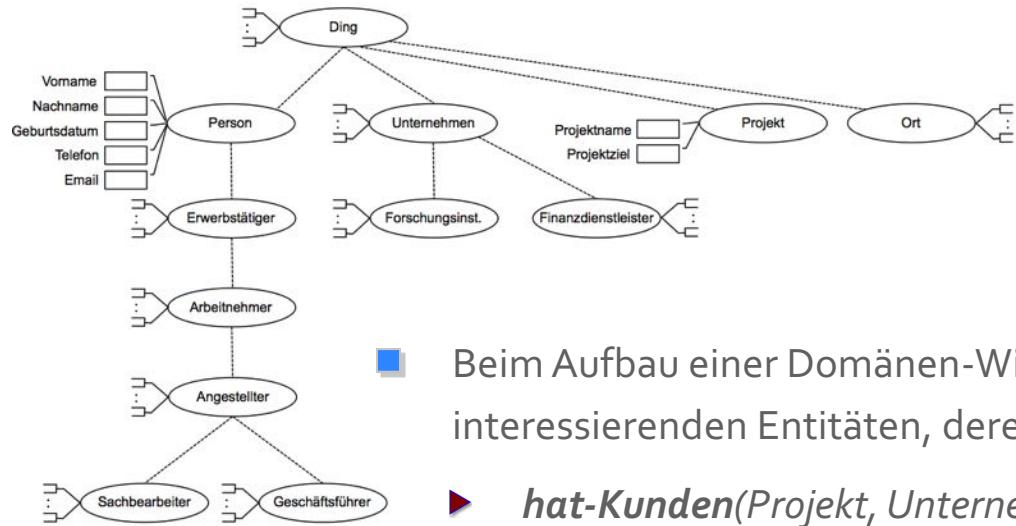


- **Relationen** beschreiben Abhängigkeiten von Klassen. Typische Relationen für Ontologien sind die klassischen „ist-ein(e)“- und „ist-Teil-von“-Relationen sowie Relation, die für die Domäne typisch sind
 - ▶ z.B. „hat-gebacken“, „ist-haltbar-bis“
- **Regeln** werden benutzt, um wahre Begebenheiten einer Domäne zu beschreiben
 - ▶ z.B. „Wenn die Zutaten eines Zimtgebäcks verdorben sind, so ist auch das Zimtgebäck ungenießbar“

Ausgangspunkt zum Aufbau einer Ontologie ist meist eine Taxonomie mit relevanten Begriffen und Attributen



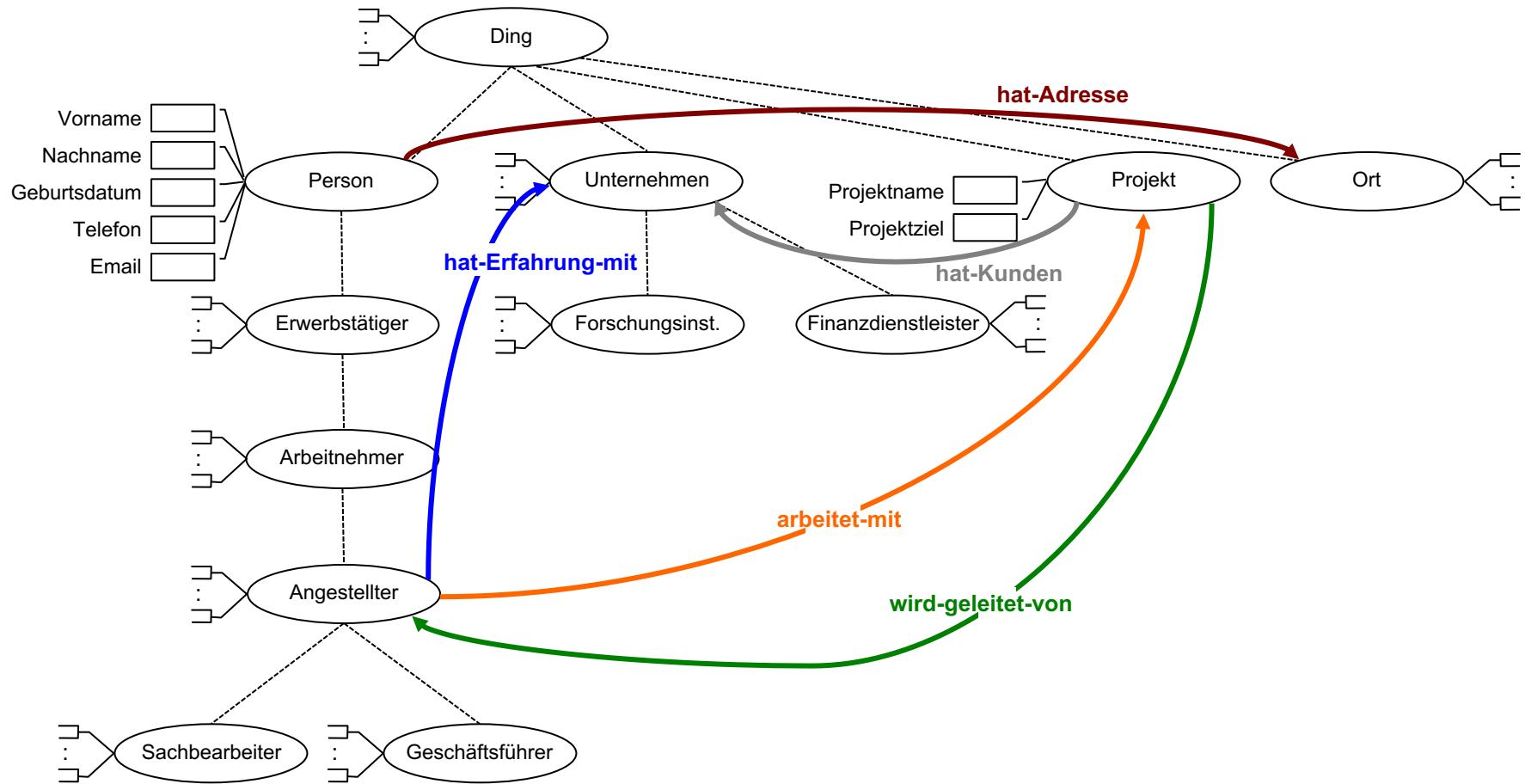
Neben der Standardrelation einer Taxonomie können andere Relationen eingeführt werden



Bei der Bestimmung der Relationen ist es wichtig, die richtigen Konzepte einzutragen!

- Beim Aufbau einer Domänen-Wissensbasis werden neben den interessierenden Entitäten, deren Beziehungen festgelegt, z.B.:
 - ▶ **hat-Kunden**(Projekt, Unternehmen): ein Projekt hat ein Unternehmen als Kunde
 - ▶ **wird-geleitet-von**(Projekt, Angestellter): ein Projekt hat einen Leiter
 - ▶ **arbeitet-mit**(Angestellter, Projekt): eine Angestellter ist für ein Projekt tätig
 - ▶ **hat-erfahrung-mit**(Angestellter, Unternehmen): eine Angestellter hat Erfahrung mit einem Unternehmen
 - ▶ **hat-adresse**(Person, Ort): eine Person hat eine Adresse

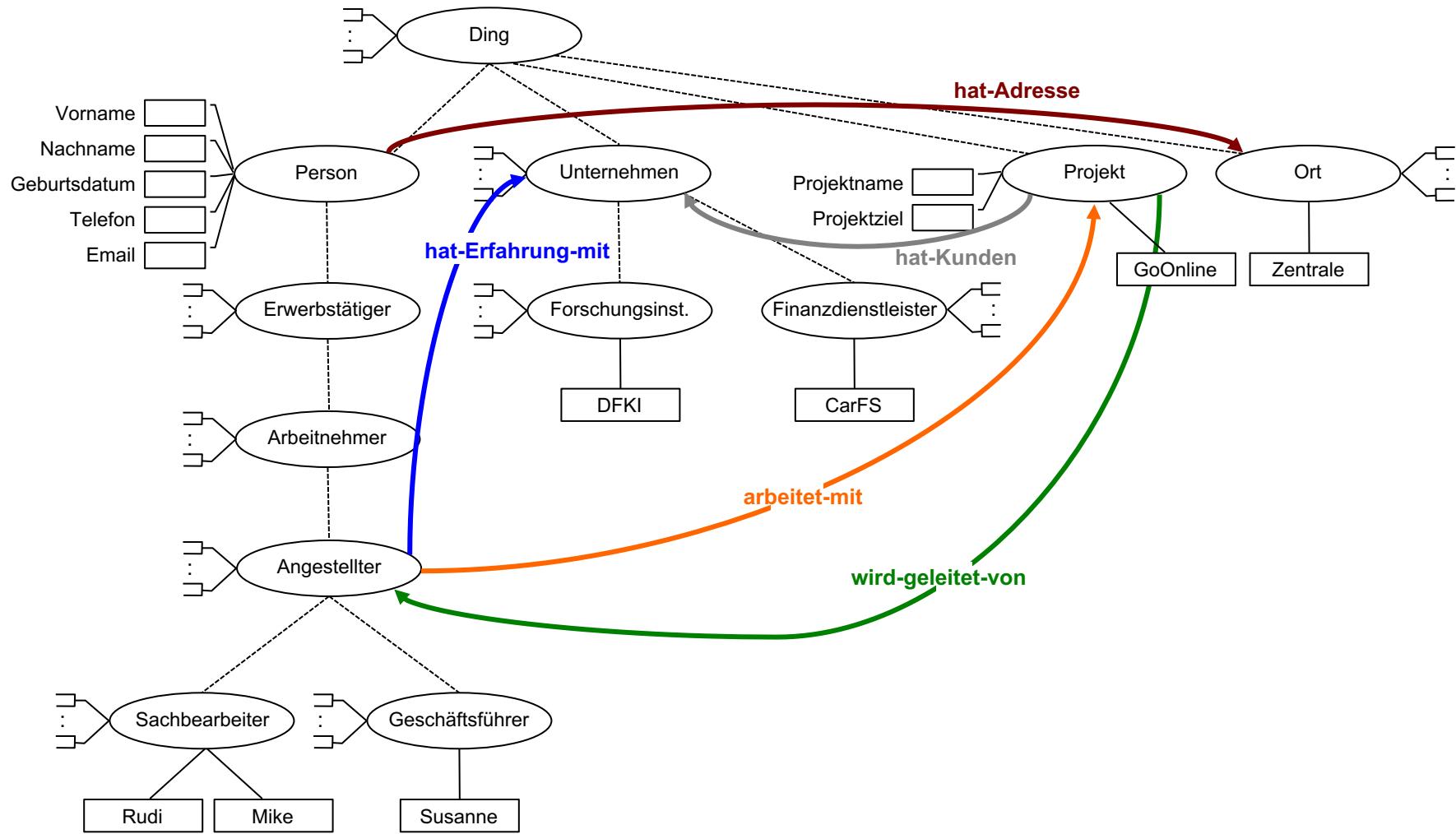
Mit den definierten Beziehungen kann die Beispiel-Ontologie ergänzt werden



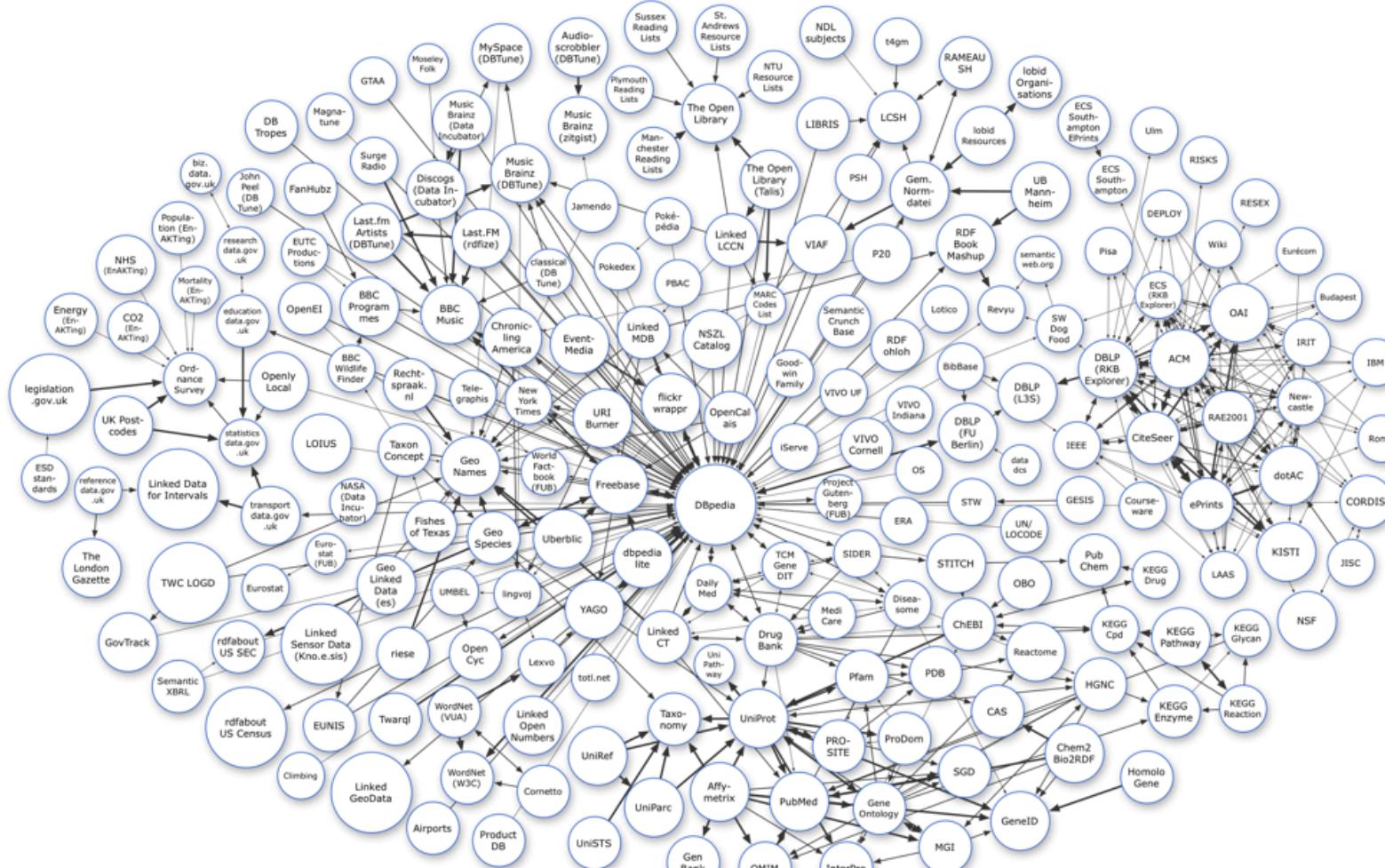
Die Verwendung von Regeln machen Ontologien zu einem mächtigen Wissens-Werkzeug

- Regeln machen allgemeingültige Aussagen über Begriffe und deren Zusammenhänge in Abhängigkeit der Attribute
- Es gibt eine spezielle Form von Regeln, die sogenannten **Constraints**, die Bedingungen für Attribute vorgeben, die erfüllt sein müssen
 - ▶ z.B. „*Ein Mitarbeiter hat mindestens 2 Jahre Erfahrung mit im Leasinggeschäft*“
 - ▶ z.B. „*Ein Angestellter gehört mindestens einem und höchstens 5 Projekten an*“
 - ▶ z.B. „*Ein Manager gehört höchstens 2 Projekten an*“

Die Entitäten der Domäne lassen sich dann über Instanzen einführen und mit Werten belegen

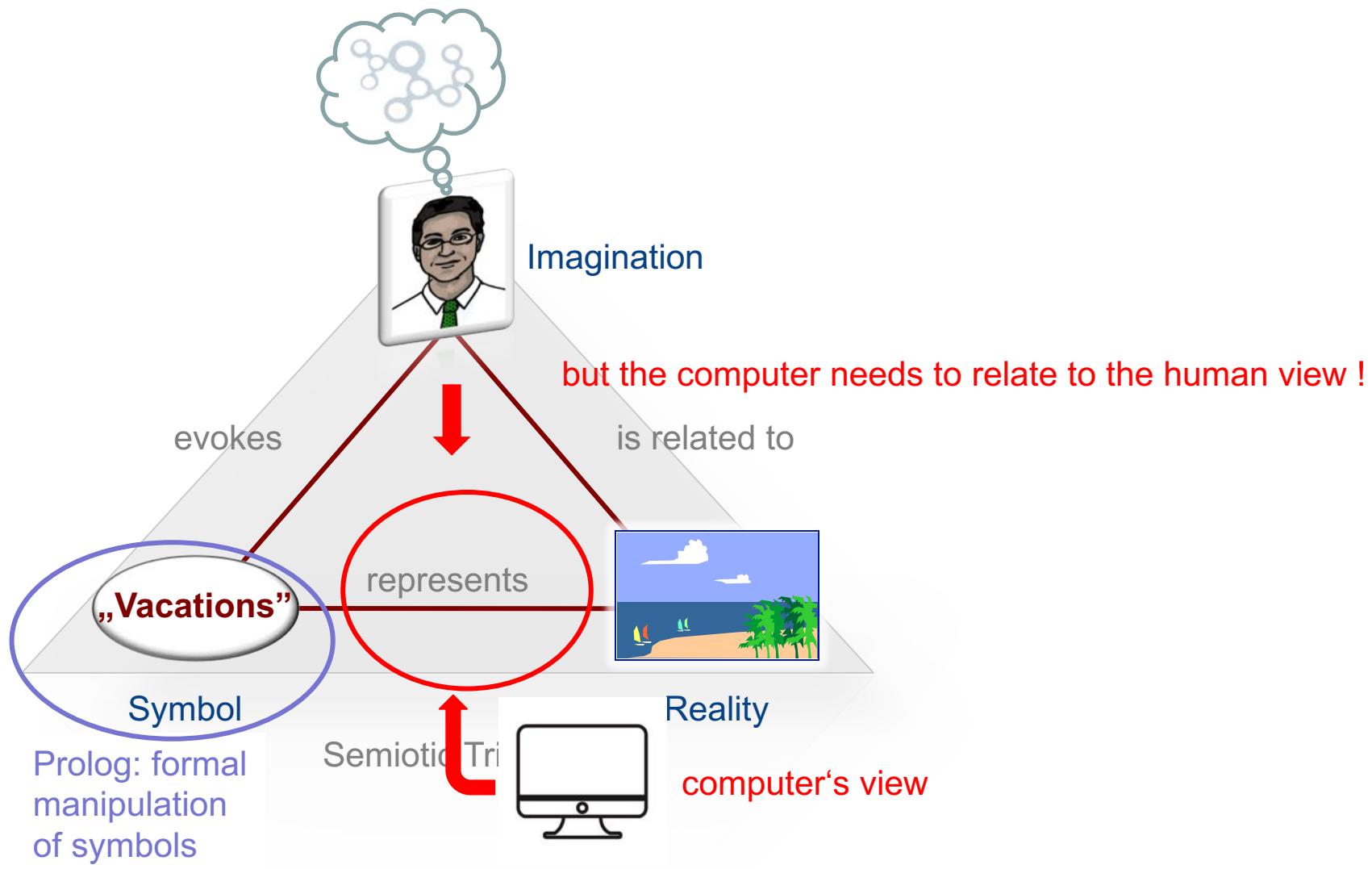


In der Linked Open Data Cloud entwickeln sich zahlreiche Ontologien

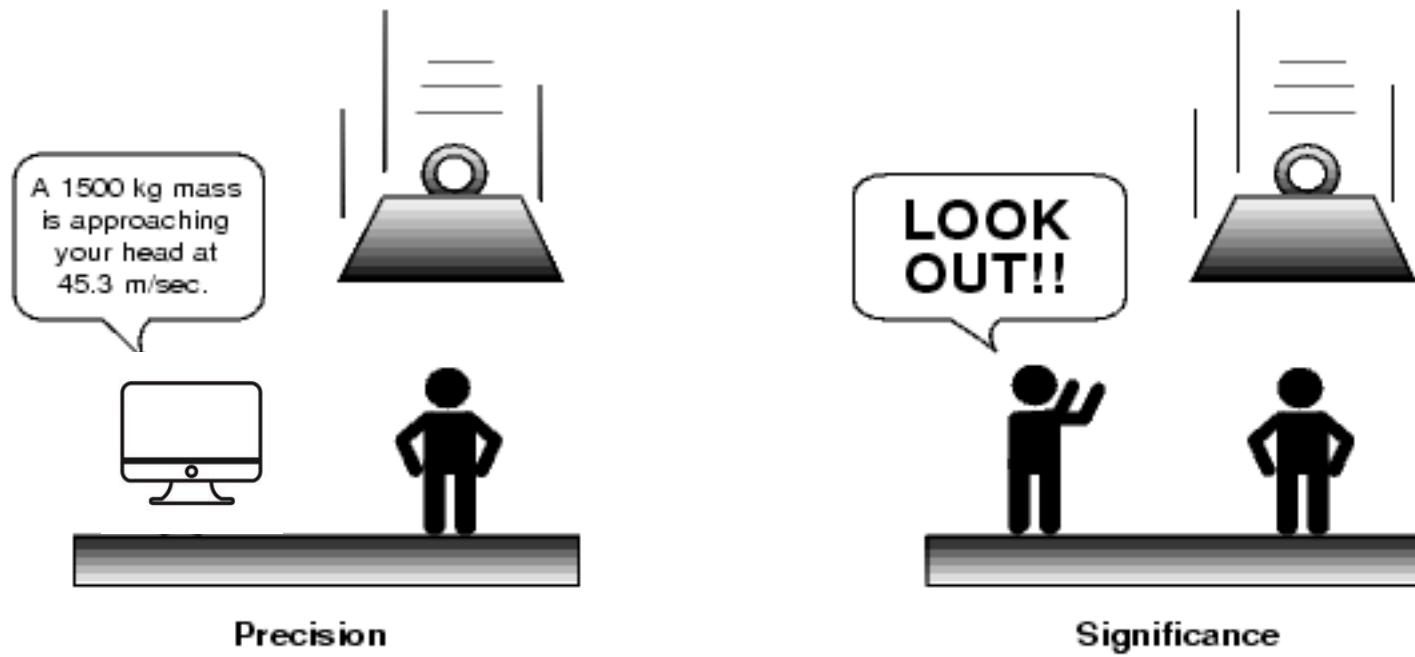


As of September 2010

Symbol processing may be explained by the Semiotic Triangle



In some situations we are not able to precisely describe the circumstances



Based on “Symbolic AI” by

Prof. Dr. Andreas Dengel

Einführung in die KI,

Teil 4 Fuzzy Logic

Prof. Dr. Paul Lukowicz

In some situations we are not able to precisely describe the circumstances

- Until now :

Statements are either right or wrong

- but:

It is often impossible to decide on the truth value of a statement since ...

- ... the sensorial information is not unique or*
- ... the measurement is not precise*
- ... the representation language only allows for a vague description*
- ... the views of the human experts may differ*
- ... the statement itself or used terms are not conclusive*
- ... there is no clear discrimination with regard to the interpretation of a statement*
- ... the description of the situation is incomplete*

Fuzzy sets and fuzzy reasoning has obtained a strong push in the mid eighties

- Professor Lotfi Zadeh, UC Berkeley, 1965
“People do not require precise, numerical information input, and yet they are capable of highly adaptive control.”
- Consider you want to cross a road at an arbitrary place (not on zebra crossing or pedestrian traffic light). You see a car coming towards you.
Which factors (variables) primarily influence your decision of going ahead?
 - Speed of approaching car **in terms of slow, fast, very fast**
 - Distance of approaching car **in terms of close, far, very far**
- ➡ You can (hopefully) make a decision to cross the road without having precise knowledge about the values of these variables

Types of uncertainty and the modeling of uncertainty

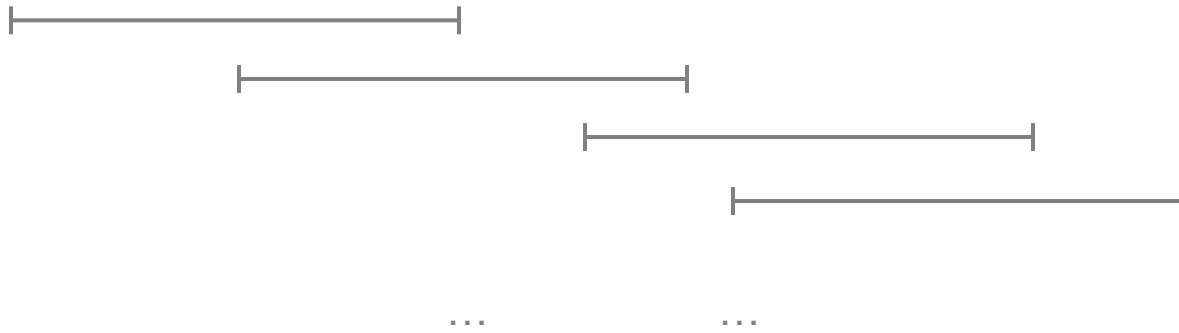
- Stochastic Uncertainty:
The probability of hitting the target is 0.8.
- Lexical Uncertainty:
 - "Tall Men", "Hot Days", or "Stable Currencies"
 - We will probably have a successful business year.
- Most words and evaluations we use in our daily reasoning are not clearly defined in a mathematical manner. This allows humans to reason on an abstract level!
- **Fuzzy logic** allows making decisions in a way that is close to the human way of thinking, and using natural language. On the contrary, using the classical crisp logic, we need to write algorithms and adapt to a machine's way of thinking

Nähern wir uns dem Thema Schritt für Schritt!

 Please use adjectives small, medium, tall, and very tall!



■ What does small, medium, (tall,) very tall mean?



Fuzzy sets and fuzzy reasoning addresses the lack in expressiveness of logic rules

Logic rule / Linguistic rule

- Logic rule (e.g.: XPS)

IF *pressure ≥ 1000 mbar*
AND *temperature ≥ 100 °C*
THEN *rotation = 2500 U/min*

- Linguistic rule

„*If the pressure rises higher than usual and the water temperature is too high then reduce the engine's rotation*“

IF *pressure is higher than usual*
AND *temperature is too high*
THEN *reduce rotation speed*

Note:

Fuzzy reasoning requires the evaluation of the truth value, of the requirements and of the measure of belief in order to validate a rule.

(according to human behavior)

Fuzzy sets are based on a mathematical model that allows to automatically process vague terms

- The idea is to **continuously** define the degree of **membership** for elements to sets or terms:

*Define the domain for the respective term or set (**linguistic variable**)*

Determine the respective membership function



e.g.:

The height of the man is ...

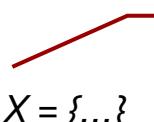
- Linguistic variables ...

... are used to transform expert knowledge which is commonly available into concepts based on the theory of fuzzy sets

... are particularly used to display different characteristics of an examined set

- Two tasks have to be solved:...

Define set of values for linguistic variable: Value set $X = \{ \dots \}$



e.g.:

Height = {small, medium, tall}

Define membership function for the term „A“:

$$\mu_A: X \rightarrow [0, 1]$$

Let's consider some examples and the corresponding explanation

- Linguistic variable „height“:

Value set $X = \{50, \dots, 250\}$

- Membership function for the term „tall“:

$$\mu_{\text{tall}}: X \rightarrow [0, 1]$$

Note:

$\mu_{\text{tall}}(x)$ illustrates the membership of the value $x \in X$ to the term „tall“

$\mu_{\text{tall}}(x) = 0$ means „ x does definitely not belong to the set X “

$\mu_{\text{tall}}(x) = 1$ means „ x definitely belongs to the set X “

$\mu_{\text{tall}}(x) = r, 0 < r < 1$ means „ x belongs to the set X with the degree r “

- Approach is not only applicable to describe vague terms but also precise terms can be modeled with respective value sets

Statement „ X is 190 cm tall“ is precise

Statement „ X is between 185 and 195 cm tall“ is already imprecise

Statement „ X is tall“ is vague

Fuzzy sets and fuzzy reasoning

- For all three statements membership functions can be given:

Observation:

In case 1 there is a 1-ary set

In case 2 we have a classical “sharp” set

In case 3 some values definitely belong to the term „tall“, some do not and others only to a certain, measurable degree

Note:

In principal all kinds of membership functions are possible

Fuzzy sets and fuzzy reasoning

1

$$\mu_{tall}(x) = \begin{cases} 1 & \text{if } x = 190 \\ 0 & \text{otherwise} \end{cases}$$

2

$$\mu_{tall}(x) = \begin{cases} 1 & \text{if } 185 \leq x \leq 195 \\ 0 & \text{otherwise} \end{cases}$$

3



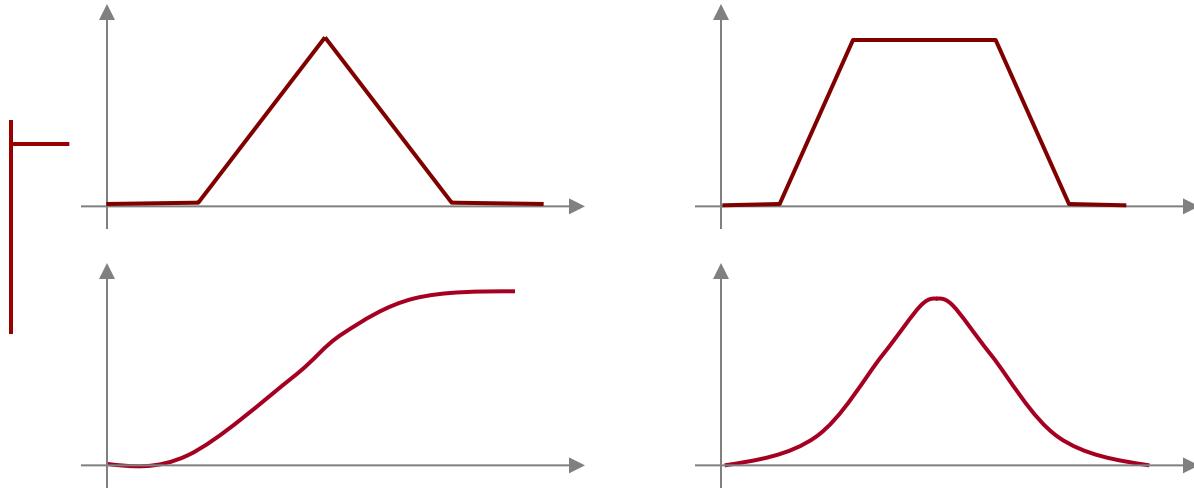


How can we express vagueness

For expressing different degrees of memberships, we may make use of a membership function

Examples for membership functions:

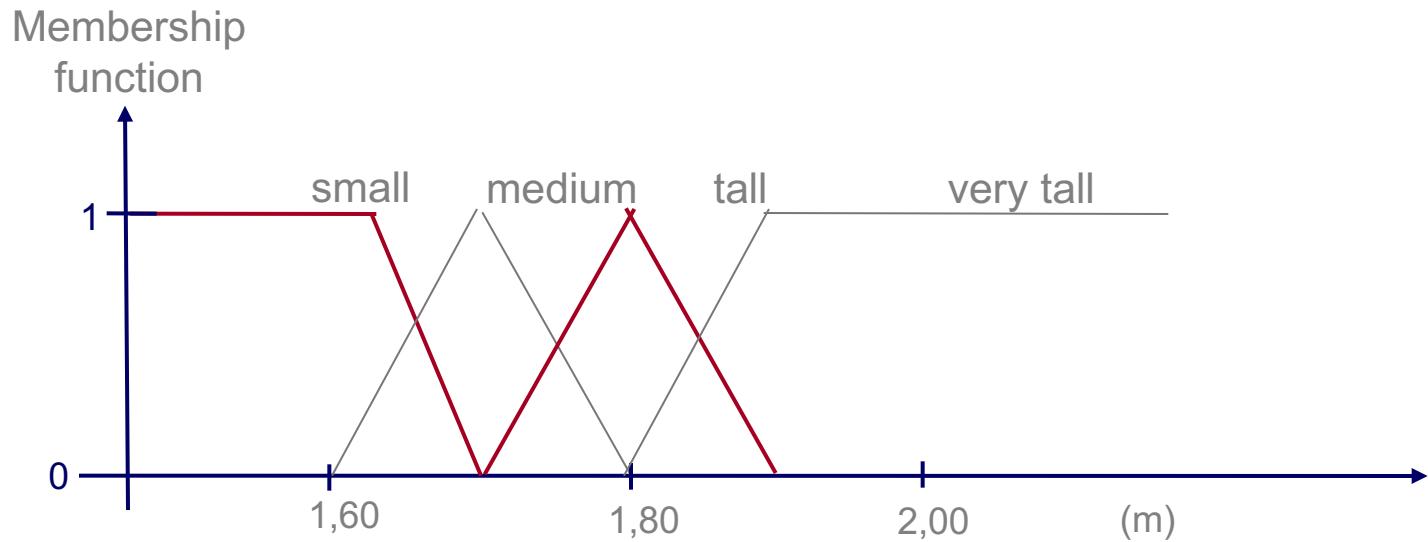
Note:
In practice triangles or trapezes proved to be most relevant



- An equilateral triangle of a fuzzy set A with the range $2d$ and the maximum x_0 has the membership function:

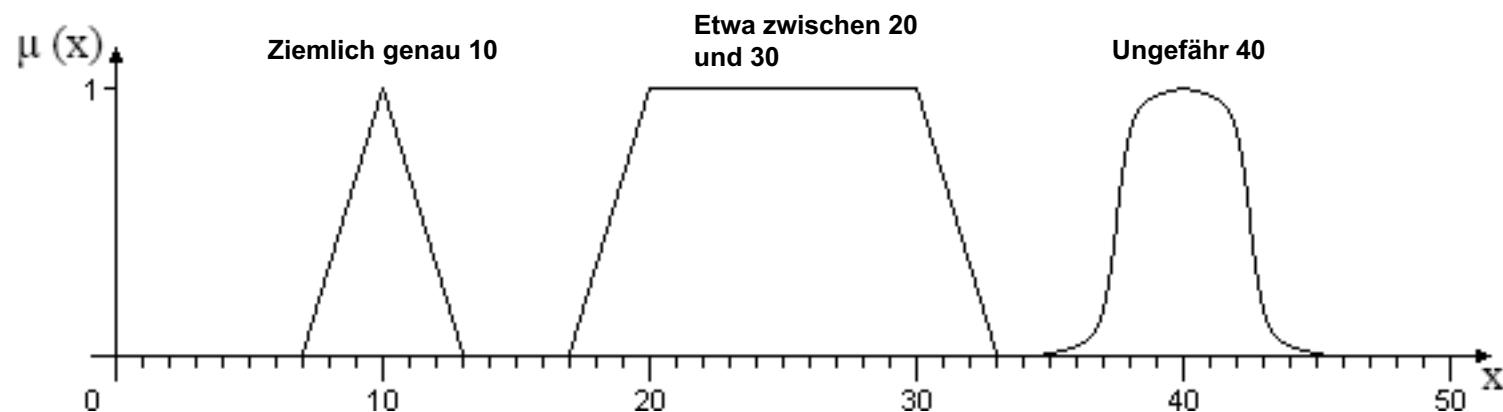
$$\mu_A(x) = 1 - \min \left\{ \frac{|x - x_0|}{d}, 1 \right\}$$

Fuzzy sets and fuzzy reasoning



Characteristic functions

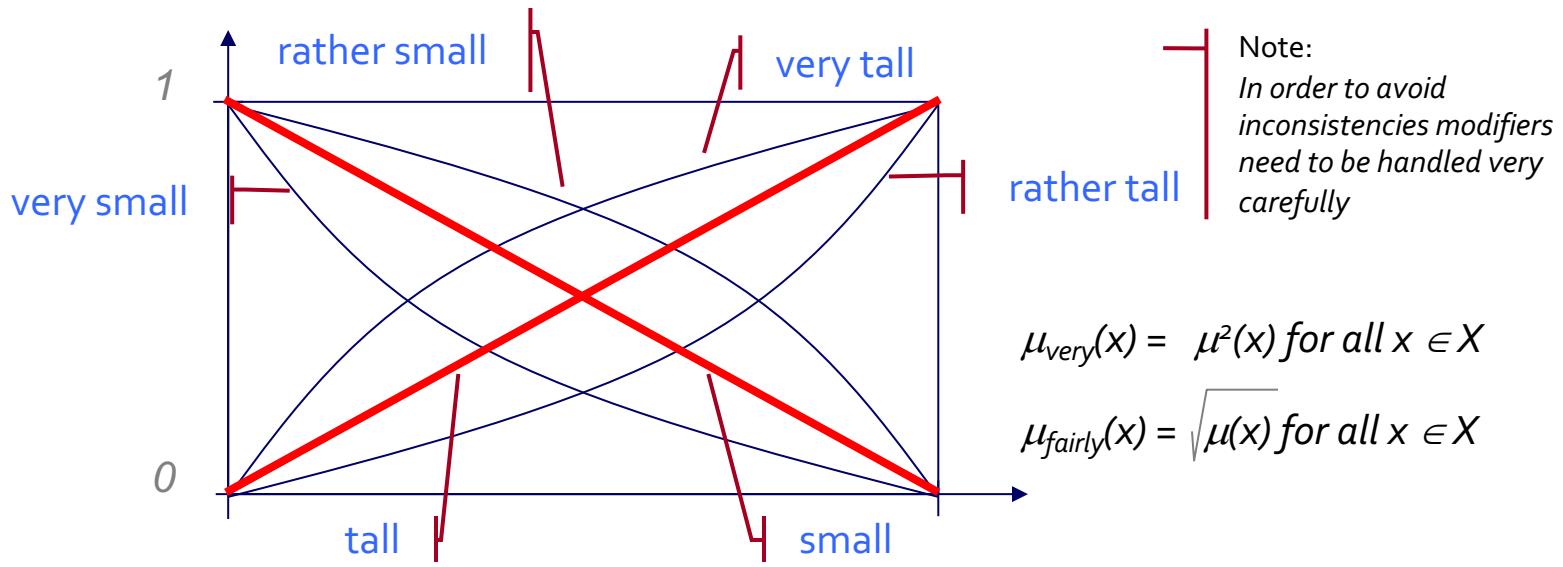
- Examples for triangle, trapezium and Gauß functions



In order to be more expressive, membership functions may be adapted to specific needs of a given domain

- According to natural language so called **modifiers** can be suggested for linguistic variables
 - e.g.: „very“, „more or less“, „almost“, ...

- Example:



Fuzzy-sets can easily be described and stored as sets or lists of the form

- Elements of the list represent distinctive points of the function

$$((x_1, \mu_B(x_1)), \dots ((x_m, \mu_B(x_m)))$$

with values $x_i \in X$

X is the considered domain

The list of the indicated value pairs define the prominent vertices of the fuzzy set

- The triangle form of the membership function can be represented as a list of the form

$$\mu_A: ((x_o - d, 0), (x_o, 1), (x_o + d, 0))$$

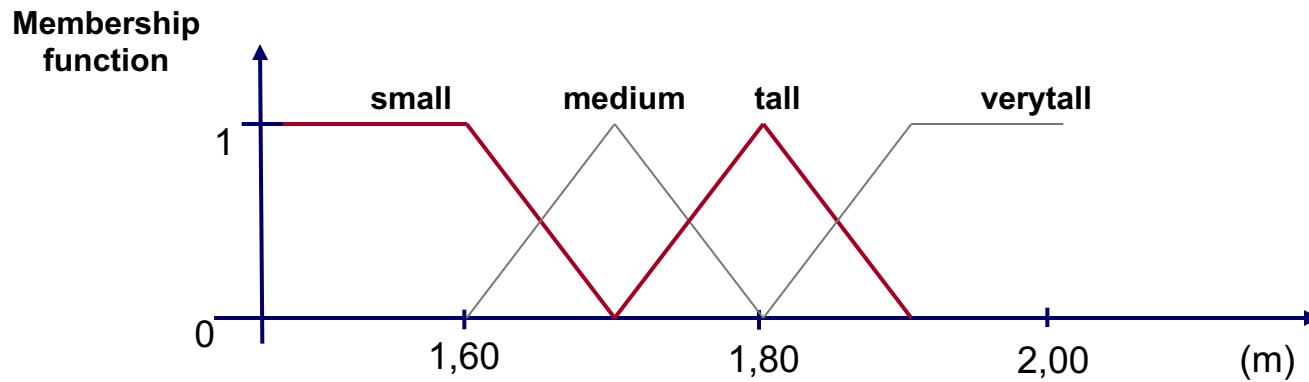
Note: The number of vertices depends on the form of the membership function

Fuzzy sets and fuzzy reasoning

- Example: (following p. 14, complete on the board)

Looking at height, for example four terms can be described in the form of fuzzy sets: small, medium, tall, very tall

With the following distribution (domain 50 to 250 cm)



It holds:

$$\mu_{\text{small}}: ((50, 1), ((160, 1), ((170, 0)))$$

...

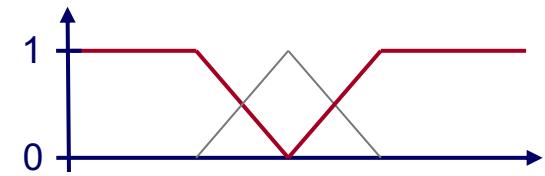
$$\mu_{\text{verytall}}: ((180, 0), (190, 1))$$

The classical set operations complement, intersection, and union may easily be generalized with fuzzy sets

- Using the fuzzy sets $\mu_A(x)$ and $\mu_B(x)$ with $x \in X$ we define:

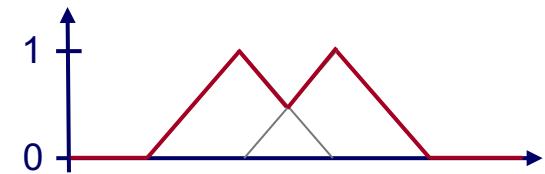
Complement:

$$\mu_{X \setminus A}(x) = 1 - \mu_A(x)$$



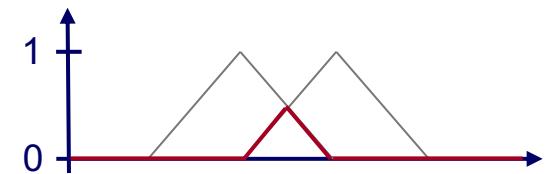
Union

$$\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x))$$



Intersection:

$$\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x))$$



On this basis, fuzzy rules can be defined and interpreted

This allows us to make some observations compared to classical logic

- The fuzzy logic can count as an extension of the classical logic
- If we combine the „more or less true“ of the fuzzy logic with the terms: small, medium, tall, very tall we receive a linguistic variable, capable to adopt values like rather, almost or medium tall

- Logical AND

Reasoning is only possible if all preconditions are fulfilled simultaneously

- Linguistic AND

Contains a certain degree of compensation; e.g. more of one and less of the other operator

Fuzzy rules are more like in practical life

Examples:

IF *the distance to the car ahead is small*
AND *the speed is high*
THEN *brake hard*

IF *clothes are hardly dirty*
OR *weight of clothes is low*
THEN *use less washing powder*

IF *clothes are really dirty*
THEN *use high temperature for the laundry*

Observations:

- Condition parts describe a situation in which rules can be applied while the values of the measurement are not discrete
- Rule conclusions also entail fuzzy statements of appropriate control values or variables for the observed situation
- Linguistic rules are applicable to describe expert behavior
- But an automatic rule requires to calculate a precise value of the control variable when precise measurements are given