

# Knowledge-Based Systems (KBS)

## Machine Learning and Knowledge Engineering

Winter semester 2021/2022

Prof. Dr. Martin Atzmüller

Osnabrück University & DFKI  
Semantic Information Systems Group  
<https://sis.cs.uos.de/>

# 1 – KBS – Introduction and Overview

- ① Motivation & First Characterization
- ② Outline/KBS Topics & Contents
- ③ KBS & Artificial Intelligence
- ④ KBS & Machine Learning
- ⑤ KBS & Expert Systems
- ⑥ Example KR for KBS
- ⑦ KBS by Example

# KBS – Characterization

Title of the course:

KNOWLEDGE-BASED SYSTEMS: MACHINE LEARNING AND  
KNOWLEDGE ENGINEERING

Three core ingredients:

- Knowledge-Based Systems
- Machine Learning
- Knowledge Engineering

## KBS – Characterization

Some exemplary questions that could come up in this context:

- What is knowledge and what is a knowledge-based system?
- What is machine learning, and how does it fit into KBS?
- Has this something to do with intelligence?
- If so, what is then intelligence?
- Is artificial intelligence itself different from human intelligence?
- Will computers eventually rule the world?

# Scope & Focus of this Course

## KNOWLEDGE-BASED SYSTEMS: MACHINE LEARNING AND KNOWLEDGE ENGINEERING

- In this course: Focus more on questions of the first two types above, approaching such questions from a methodological perspective  
    ↪ KBS, (semi-/automatic) machine learning, knowledge representation, knowledge engineering, hybrid approaches, ...
- Therefore, we do not consider biological, political or social aspects of these systems
- However: Some questions relating to human-centered *design* of KBS and its according methods

# Motivation

## What is a Knowledge-Based System?

Two exemplary definitions:

- “A computer system that uses a knowledge base to support reasoning processes in order to solve an application problem. Expert systems are examples, but knowledge-based systems can take many other forms and can be found in many areas of artificial intelligence.” (Oxford Reference)
- “A knowledge-based system (KBS) is a computer program that reasons and uses a knowledge base to solve complex problems. The term is broad and refers to many different kinds of systems.” (Wikipedia)

# Knowledge-Based Systems (KBS)

Intuition – important components (from those definitions above):

- “Knowledge base”,
- “reasoning processes”,
- KBS “solve an application problem”,
- KBS “solves complex problems”
- ... “ term is *broad* and refers to *many different systems*”

So, what about *Machine Learning*?

## KBS and the role of machine-learning methods

“Since its origins, AI (Artificial Intelligence) has swung between formal reasoning techniques based on knowledge (such as rule based systems and Bayesian Networks ) and machine learning based techniques. With the recent renaissance in deep learning, the pendulum has swung once again. This has reached a point where deep learning is now so synonymous with AI, that many people consider any solution which does not include deep learning to not really be AI. [...] Some currently believe that we should only utilize deep learning based techniques, relying solely on computational power and the availability of data. [...]

I, for one, am an advocate for using deep learning together with the formal reasoning techniques and believe that the next big leap in AI can be made only by finding the right trade-offs in combining the two methodologies.”<sup>1</sup> (Segev Wasserkrug)

---

<sup>1</sup><https://www.linkedin.com/pulse/knowledge-based-machine-learning-ai-segev-wasserkrug/>



# Module Contents

From the UOS/computer science module handbook – specializations:

<b>MK-9-K</b>	<b>Knowledge-based Systems: Machine Learning and Knowledge Engineering</b> 9 LP (Knowledge-based Systems: Machine Learning and Knowledge Engineering)
---------------	--

Dozent	Atzmüller
SWS	4V+2U

---

Methods, algorithms and techniques for the design and development of knowledge-based systems using machine learning and knowledge engineering. These include, for example, knowledge acquisition, knowledge representation, knowledge graphs, semi-automatic knowledge engineering methods, knowledge processing of vague knowledge, update and refinement of knowledge bases, fundamental machine learning concepts, rule-based/logic-based formalisms, case-based reasoning, probabilistic methods, interpretable machine learning, neural-network-based approaches, hybrid AI systems.

---

Voraussetzungen	<b>Erwartet:</b> SK-DBS, KI-KI
-----------------	--------------------------------

---

# Knowledge-Based Systems – Contents

- ➊ Introduction & Overview
- ➋ Knowledge Engineering & Representation Formalisms
- ➌ Knowledge Acquisition & Machine Learning
- ➍ Interpretable Learning
- ➎ Knowledge Discovery & Data Mining
- ➏ Graph & Link Mining
- ➐ Knowledge Graphs
- ➑ Rule-Based Approaches and Case-Based Reasoning
- ➒ Uncertainty Modeling & Answer-Set Programming
- ➓ (Deep) Neural Networks
- ➔ Interpretability, Explanation & Explainability
- ➕ Informed Machine Learning

## KBS – First Intuitions

- In general, there is a clear separation of knowledge and its processing by some group of algorithms
- More specifically: Separation between *problem description* and *problem solving (method)*
- Knowledge about the domain can be directly included
- The knowledge is typically stored in a *knowledge base*.
- It is accumulated by processes of knowledge engineering.
  - These can make use of various *knowledge sources*, and
  - use several sorts of *knowledge representations*

## KBS – First Intuitions (2)

- Knowledge – one of the core components
- Various definitions, ideas, conceptualizations what “knowledge” means
- In Knowledge Discovery (in Databases, KDD), for example: “valid, novel, potentially useful, and ultimately understandable patterns or relationships within a dataset in order to make important decisions” [Fayyad 1996]
- For us - working definition: Knowledge as
  - organized/structured information, with a
  - declarative representation
- Distinction between *expert system* and knowledge-based system
  - Expert system – special KBS: Knowledge (mainly) from expert
  - KBS: More general, knowledge can be acquired/provided/retrieved in a more general way

## KBS – First Intuitions (3)

- Knowledge acquisition - several options, for example:
  - Asking experts from the respective fields
  - (Semi-)automatic machine learning
  - Exploiting existing data bases
  - Also possible - combinations
- The exploitation, adaptation, derivation, and acquisition of existing and/or new knowledge is done by an inference engine
- Example techniques/methods: logical rules or specific (learning, abstraction, etc.) methods on the knowledge base in order to deduce/construct/infer new information and knowledge

# Properties of Experts and Expert Systems

- Again: Expert systems  $\sim$  KBS, for which experts provide the modeled knowledge
- Strengths and Weaknesses:
  - Experts have capabilities above average in solving specific problems, even if these problems are new or do not have a unique solution.
  - Experts use heuristic knowledge, for solving special problems, based on their experience
  - Experts make use of their common sense (world) knowledge
  - Experts often act intuitively in a correct way, however, cannot explain their decision(s).
  - They can solve problems in uncertain and incomplete knowledge contexts.
  - Experts are rare and typically expensive
  - Their performance is not constant, but depends on how they are feeling that day
  - One expert (e.g., for a report/expert opinion) is often not sufficient
  - Expert knowledge can get lost
  - Expert knowledge can often not be directly transferred

## KBS – Medical Example

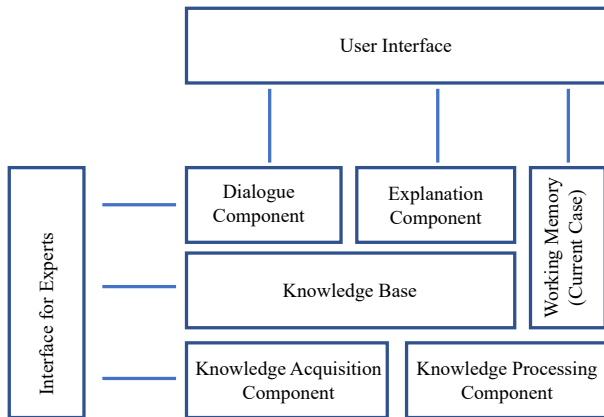
- Medical domain – prominent application domain for KBS
- In particular, diagnosis: *symptoms* → *diagnosis*
- Several tasks:
  - How can medical information, that are collected, organized and stored be queried efficiently, and how can they be complemented? A KBS can help in retrieving and providing relevant information.
  - How can learning from experience be supported? How can the modeled knowledge be refined – via maintenance – to the current/upcoming state-of-the-art?
  - Given a set of symptoms, how can the most likely diagnosis be determined?
  - What are the relationships between the set of (non-observable) diagnoses, and the (partially) observable set of symptoms? How can these be modeled?
  - Which additional information needs to be obtained in order to make the diagnosis more certain? How much validity is obtained with this, individually?

## KBS – Basic Considerations

- KBS: Clear separation of knowledge and its processing by some group of algorithms
  - Knowledge base
  - Knowledge processing (methods)
- How is the knowledge stored in the knowledge base, ie., in which form?
  - Rule-based
  - Predicate logic
  - ...
- Determines methods for knowledge processing which can be applied
- Further components:
  - Knowledge acquisition component
  - Explanation component
  - Dialogue component



# KBS – Basic Architecture



# KBS and Their Relation to AI

- KBS trigger many important scientific and practical developments in the field of artificial intelligence
- Also, vice versa, knowledge-based systems are influenced by recent developments in artificial intelligence
- In particular, machine learning, computational methods, but also knowledge-representation and reasoning (which are core fields of KBS)

## Excursus: Artificial Intelligence (AI)

- Recently, impressive developments in AI as well as machine learning (deep learning)
- But: Current trend (again!) to focus more on knowledge-based methods, in combination with machine learning
- In general, it is much more complex to define AI (methods), since it is a more general term compared to knowledge-based methods
- Also:  $\rightsquigarrow$  AI course ...
- Pragmatic definition (but also slightly problematic) “Artificial Intelligence is the study of how to make computers do things at which, at the moment, people are better.” Rich (1983)
- Hybrid AI: Knowledge-based + automatic machine learning  
 $\rightsquigarrow$  symbolic + subsymbolic methods

# History of AI

## Turing Test for Intelligent Machines

- In 1950, the famous british computer scientist Alan Turing proposed to test a machine's ability to exhibit intelligent behavior indistinguishable from a human (Wikipedia).
- In 1956, many of the attendees of the first AI workshop at Dartmouth College, USA, predicted that a machine as intelligent as a human being would exist in no more than a generation.

# History of AI

1930/1940

- Gödel, Church, and Turing: First theoretical aspects
- Gödel's completeness theorem: Correspondence between semantic truth and syntactic provability in first-order logic
- Church: Lambda-calculus – functional programming
- Turing: Proof of non-decidability of the Halting problem
- McCulloch, Pitts, and Hebb: First mathematical models for neural networks

# History of AI

1950/1960/1970

- The rise of first programmable computers
- Newell and Simon: Logic Theorist program – problem solving skills – the “first artificial intelligence program”
- McCarthy: LISP programming language, processing of symbolic structures
- Resolution: Completeness for predicate logic proven
- PROLOG: Logic programming, Horn clause calculus in predicate logic

# History of AI

1980/1990

- Computational power (increase)  $\rightsquigarrow$  first practical application(s) of neural networks
- NETtalk: a parallel network that learns to read aloud
- First successful commercial expert system, R1 (internally XCON: eXpert CONfigurer); DEC – configuration of computers (VAX machines)
- Handling of imprecise knowledge
- Bayesian reasoning
- Fuzzy Logic: extension of two-state logic to infinite values in the interval between zero and one; practical applications (for example, see Elkan 1993):
- System like CART, ID3, or C4.5: can generate decision trees with promising correctness/accuracy rates. Therefore, such ML models can be used as practical expert systems

# History of AI

2000/2010/2020

- Data Mining: Derive explicit knowledge from data bases
- Combining logic and neural networks in hybrid systems
- Massive improvements in deep learning
- Distributed large agent systems
- Big data integration / usage



## Hybrid Approaches: Symbolic/Subsymbolic AI

*Symbolic or knowledge / rule-based* AI models central cognitive abilities of humans like *logic*, deduction and planning in computers. Mathematically exact operations can be defined. Decision making can be supported.

*Subsymbolic* or statistical AI tries to learn a model of a process (e.g., an action of a robot or the classification of sensor data), from training data.

*Hybrid approaches involve the combination of both.*

## Current Status of AI

So, since the formulation of the Turing Test in 1950, there have been many success stories of AI:

- Computers can compete in *games* (Chess 1997, Go 2015).
- In health care, computers can support sick, handicapped or old-aged people.
- Hotlines of call centers are supported by computers based on NLP, image recognition, and *intelligent decision support*.
- Algorithms from *deep learning* are increasingly used by banks and insurance companies for decision making.
- Companies are developing *autonomously driving cars* (cf. Google) and devising autonomous space missions.

## Future of AI

- AI reports about popular developments can be found in many newspapers:  
<https://www.bbc.com/news/business-48994128>
- AI is considered to be a *key technology* for society. As for other new technologies, the questions arises, how AI will influence our future and how we can guide its development.
- Key questions about the *chances and risks* have to be answered in the fields of, e.g., ethics, law, biology, and computer science.  $\rightsquigarrow$  *responsible AI, trusted AI* – also for knowledge-based systems
- This is also connected to *explainable machine learning* (leading to trusted machine learning), see below.

# Explainable AI

With the availability of powerful AI methods, more and more are applied for decision support. Explicative methods are:

- Interpretable,
- Transparent, and
- Explainable

Benefits:

- Explicative approaches make models and methods understandable
- They enable computational sensemaking
- Trust of the user applying those methods is increased

# Declarative Programming

Techniques from symbolic AI can help to solve problems of the digitization wave. *Modern intelligent information systems* need to integrate and reason about hybrid knowledge bases (inference) with expert knowledge.

Declarative Programming (DP) is an advanced paradigm for modeling and solving complex problems in, e.g., data science, AI, NLP, and for establishing systems for the web.

- Traditional, *imperative* programming languages tell the computer exactly how to accomplish a goal.
- Modern DP languages only specify the goal to the computer, e.g. Database, Rule, and Ontology Languages.

## Declarative Rule Bases

Declarativity: forward reasoning, treatment of negation and aggregation, e.g., with ASP-tools (Clingo or dlv).

### DSL Notation

- Logical rules of the form  $A \wedge B \rightarrow C \vee D$  can be written in DSL notation as **if** A **and** B **then** C **or** D
- deduction with complex rules is possible (generic, heuristic, defeasible, with default negation, ...)

### Applications

- Diagnosis in Industry, Medicine, etc.
- Business Rules in E-Commerce
- Root Cause Analysis in Computer Networks
- Discovering Anomalous Links in Networks

# Learning

- According to Michalski and Kodratoff, Y. (1990) “Research in machine learning has been concerned with building computer programs able to construct new knowledge or to improve already possessed knowledge by using input information.”
- Although the literature does not provide a common definition of learning, there is some consensus according to the distinction between the acquisition of new knowledge and the attainment of performance improvements

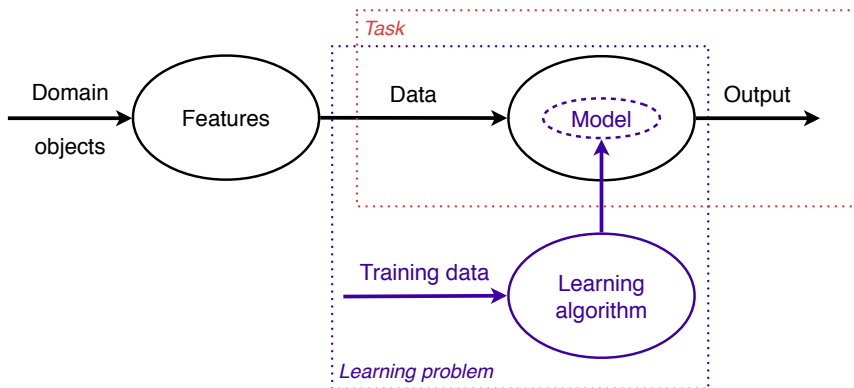
# Machine Learning – Overview

- Many definitions
- “The use and development of computer systems that are able to learn and adapt without following explicit instructions, by using algorithms and statistical models to analyse and draw inferences from patterns in data.” (Oxford, Dictionary)
- Tasks:
  - Predictive: Predicting a target variable from features
  - Descriptive: Exploiting underlying structure in the data
- Both can be used for acquiring, formalizing, discovering *knowledge* ...
- ... via finding structure(s) in data



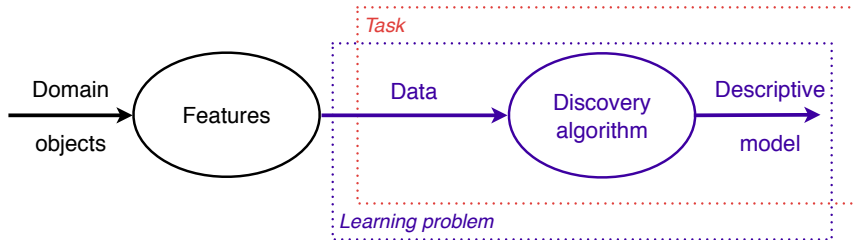
# Machine Learning - General

(cf. Peter A. Flach (2012). Machine Learning. The Art and Science of Algorithms that Make Sense of Data. Cambridge University Press



# Machine Learning - Descriptive

(cf. Peter A. Flach (2012). Machine Learning. The Art and Science of Algorithms that Make Sense of Data. Cambridge University Press



## Semi-Automatic Approaches

- Combining machine learning with human-in-the-loop
- Advantages
  - Level of control of the derived knowledge – quality control
  - Iterative refinement options
  - Background knowledge can be incrementally supplied
- Example: Learning diagnostic scores (see “KBS by example” slides below ...)  
(cf. Atzmüller et al. (2006) Semi-automatic learning of simple diagnostic scores utilizing complexity measures. Artificial Intelligence in Medicine. 37:19–30)

# Hybrid Approaches & Informed Machine Learning

- Core idea: Integrate knowledge-based and (automatic/machine-learning) approaches
- Use knowledge already in building machine learning models
- Which types of knowledge?
- Some examples:
  - Intuitive domain/expert knowledge (rules, etc.)
  - Common sense/world knowledge (language, visual concepts)
  - Application domain constraints (linear equations, physics models, etc.)

## Expert Systems

Again: Expert systems are a special (sub-)type of knowledge-based systems (historically, the first knowledge-based systems)

- Goal: Capture expert knowledge and reasoning capabilities (of experts) for special tasks, in order to get to similar (or better) problem solving.
- Application: technical/medical diagnosis
- Problems: Knowledge is often not explicable (knowledge acquisition problem)
  - common sense knowledge
  - experience
  - complex problem solving strategies
  - large search spaces
  - heterogeneous forms of knowledge
  - etc.

Methods: Knowledge representation & inference, reusable ontologies and problem solving methods, declarativity

## Expert System Tasks

There are different categorization schemes, characterizing tasks from different perspectives (Breuker et al., Clancey, Puppe).

Important exemplary tasks:

- Classification
- Configuration
- Construction/Design
- Diagnosis
- Planning
- Simulation

We will generalize this towards KBS in the following.

## KBS – Problem Solving Types

We will mainly consider the three-fold categorization of problem solving types by Puppe [Puppe, 2012]:

- **Classification**
- *Configuration*
- (Simulation)

Here, knowledge acquisition/engineering can often be directly applied, e.g., machine learning for classification. (In this course, we will focus on *classification* and to some extent on *configuration*)

## Problem solving type: Classification

Applicable for problems with the following characteristics

- Domain consists of two finite, disjoint sets – one containing observations, the other problem solutions; also, there is typically uncertain, complex knowledge about the relationships between these two sets
- Problem is defined by a given subset of observations (symptoms, findings) which may be incomplete
- Result of classification is the selection of one or more solutions for the problem
- If quality of the solutions can be improved by considering additional observations, one of the tasks of classification is to determine which additional observations are to be requested



## Classification: Problem types

- Static fault finding/diagnosis
- Dynamic fault finding/diagnosis
- Assessment
- Multiple assessment
- Precedence selection
- Object identification

## Classification – Diagnosis

Diagnosis Task: Identify reasons for faulty behavior of a system, such that a therapy can be applied.

Application areas:

- Medicine: Diagnosis of diseases
- Car/automotive: Diagnosis of motor faults

## Example – Diagnosis

	Probability	Name	Type	QuantitativeDescription
Component	0.2	Boostconverter		
Inputs				
CDI		CurrentDrawIn	double,0.0,4.0,A	
VBO		VBatOldIn	double,0.0,4.3,V	
CBO		CurBatOldIn	double,0.0,4.0,A	
BRP		BatResistanceParam	double,0,1	
BVP		BaseVoltageParam	double,0,10	
BRP		BoostResistanceParam	double,0,1	
SBP		SigmoidBaseParam	double,0,10	
SNP		SigmoidNegativeOffsetParam	double,0,10	
SSP		SigmoidScalarParam	double,0,1	
SOP		SigmoidOffsetParam	double,0,1	
Outputs				
				$VBat = VBatOldIn -$ $CurBatOldIn * BatResistanceParam$ $c = (BaseVoltageParam -$ $CurrentDrawIn * BoostResistanceParam) * CurrentDrawIn$ $eff = (SigmoidScalarParam / (1 + pow(SigmoidBaseParam, (CurrentDrawIn - SigmoidNegativeOffsetParam) * -1))) + SigmoidOffsetParam$ $CurrentDrawOut = ((VBat - math.sqrt((VBat * VBat - 4 * BatResistanceParam * eff * c, 0.0))) / (2 * BatResistanceParam))$
CDO		CurrentDrawOut	double,-5.0,4.0,0.05,0.15,A	
				$if(VBatOldIn > 2.8):$ $VOut = (BaseVoltageParam - CurrentDrawIn * BoostResistanceParam)$ $else:$ $VOut = 0.0$
VO		VOut	double,-5.0,5.2,0.05,0.0,V	
ErrorFunctions				
	0.2	Boostconverter is producing overvoltage		$if(VOut > 5.2):$ $error = True$ $else:$ $error = False$

(cf. Djebko et al. (2019) Model-Based Fault Detection and Diagnosis for Spacecraft with an Application for the SONATE Triple Cube Nano-Satellite. Aerospace 6(105))

## Other application areas:

- Quality control
- Production process diagnostics
- Repair diagnostics
- Monitoring
- Network diagnostics
- Medical diagnostics/therapy
- Image and object recognition
- ...

# Problem-solving type: Construction

- Configuration
- Assignment
- Planning

# Configuration

Configuration: Assembly of a system based on single components.

Knowledge base:

- Catalogue of components
- Constraints on compatibility

Input: Requirements/restrictions on the solution

Important: Basic elements are selected from the given set, parametrized and finally assembled to form a solution object – fulfilling the given requirements/constraints.

Example case: R1/XCON - Configuration of VAX computers

# Assignment

Assignment: Mapping of one set of objects onto another, taking into account restrictions.

Knowledge base:

- Set of objects
- Restrictions

Important difference to configuration/planning: Elementary objects are given and fully characterized by themselves (so, no further parametrization of those objects itself is necessary); relations between objects need to be determined.

Scheduling - special case of assignment, where objects are mapped onto time intervals.

# Planning

Planning: Determination of a sequence of actions; beginning with a start state, a respective goal state is reached, respecting a set of certain conditions.

Knowledge base:

- Definition of operators

Input: Description of start and goal state

Solution: Plan = sequence of actions

Application case:

- Production planning
- Travel planning
- Robotics
- ...



# Example Knowledge Representations for KBS

- Various knowledge representations
- Depends on application & domain
- Examples:
  - Rules
  - Graphs
  - Diagnostic scores (learning example below)
  - Also: Cases, Bayesian networks, ...

## Knowledge Representation: Rules

Rules are a classic and widely used representation for expert systems:

```
IF (battery OK)
AND (Value FuelGauge > 0)
AND (PetrolFilter clean)
THEN (Problem = IgnitionSystem)
```

- Modeling human problem solving
- Natural representation of expert knowledge
- Experiences based on problems which have been previously solved by the expert
- These experiences have often be formalized into rules (or: “rules of thumb”)
- If an expert encounters a new problem, then often the according rule(s) can be applied

# Semantic Networks / Knowledge Graphs

- Specifying properties of subjects
- Basically: *subject*, *predicate*, *object*, where *predicate* determines the type of the property, and *object* is the according property value.
- Alternatively: *individual*, *property*, *value*

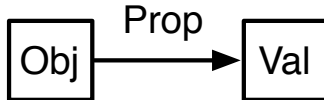
Logic (predicate):

*prop(Individual, Property, Value)*

triple representation:

*⟨individual, property, value⟩*

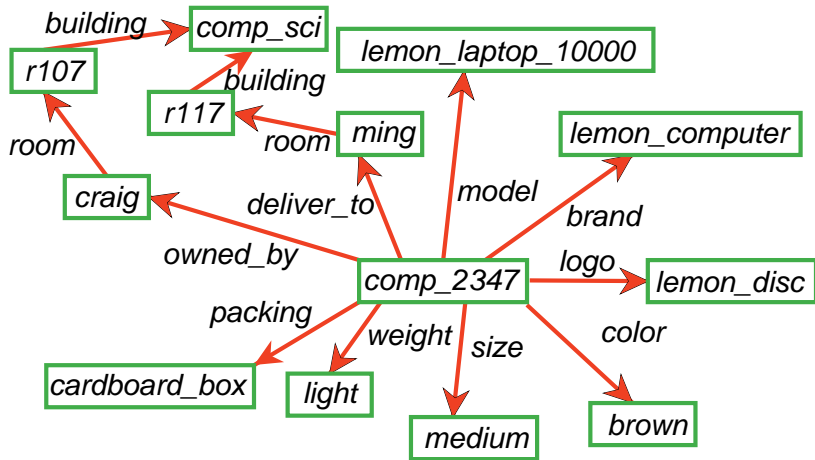
## Graphical Representation



Since many triples can be included, complex graphs can be constructed via a set of triples.

## More Complex Example – Knowledge Graph

(cf. David Poole and Alan Mackworth, Artificial Intelligence: foundations of computational agents, 2nd edition, Cambridge University Press, 2017)



## Equivalent Logic Program

```
prop(comp_2347, owned_by, craig).  
prop(comp_2347, deliver_to, ming).  
prop(comp_2347, model, lemon_laptop_10000).  
prop(comp_2347, brand, lemon_computer).  
prop(comp_2347, logo, lemon_disc).  
prop(comp_2347, color, brown).  
prop(craig, room, r107).  
prop(r107, building, comp_sci).  
⋮
```

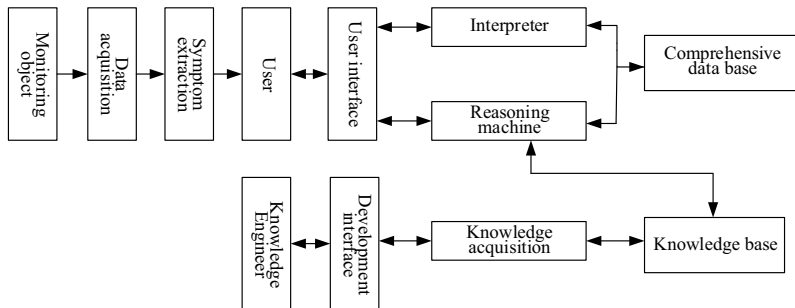
## KBS by Example

- ① Expert system for fault detection in wind turbines
- ② KnowWE - the *Knowledge Wiki Environment*
- ③ Machine Learning – Semi-Automatic Learning of Simple Diagnostic Scores

## Example 1: Fault detection in wind turbines

(cf. Deng et al. (2017) Rule - based Fault Diagnosis Expert System for Wind Turbine. IST 2017)

- Expert system for fault detection in wind turbines
- KBS/Expert system structure





# Common faults: Fault detection in wind turbines


Fault	Fault phenomena	Causes of failure
High speed shaft unbalance	<ol style="list-style-type: none"> <li>1. The spectrum of its 1X (X said multiplier) based</li> <li>2. Radial runout</li> <li>3. Amplitude increases with increasing speed</li> <li>4. Axis orbit for a more round or oval</li> </ol>	Rotor of the rotating machine due to the material density distribution, assembly factors, processing errors and movement of the erosion and deposition and other factors led to the center of mass and the rotation center there is a certain degree of eccentricity.
High speed axis misalignment	<ol style="list-style-type: none"> <li>1. The spectrum is its 1X, 2X or higher harmonic</li> <li>2. Axial vibration</li> <li>3. The amplitude does not vary with the speed</li> <li>4. Axis trajectory was banana-shaped or 8-shaped</li> </ol>	Installation in the construction of the ultra-poor, the thermal expansion of housing, chassis deformation or displacement, high-speed shaft bending.
High speed shaft bending	<ol style="list-style-type: none"> <li>1. The spectrum is its 1X, 2X or higher harmonic</li> <li>2. When the rotor speed is fixed, the vibration amplitude is fixed</li> </ol>	The bending of the shaft is permanent bending and temporary bending. The former is mainly caused by design and manufacturing defects, improper long-term parking methods, etc. The latter is mainly overloaded but can be recovered.
High speed shaft parts loosening	<ol style="list-style-type: none"> <li>1. The spectrum is its 1X and harmonic</li> <li>2. Vertical vibration greater</li> <li>3. Vibration amplitude is not fixed</li> </ol>	The fastener is not fastened, and the fastener is loose or the fastener is damaged under the vibration of the base.
High-speed shaft rubbing	<ol style="list-style-type: none"> <li>1. Most of the spectrum is its 1X component, there are harmonics, sub synchronous and natural frequency components</li> <li>2. Axis trajectory was banana-shaped or 8-shaped</li> </ol>	The speed may be too fast or the internal temperature of the wind turbine may change abruptly or the distance between the rotating parts and the fixed parts may be too small or the part offsets may cause friction.
Blade fracture	<ol style="list-style-type: none"> <li>1. The spectrum is <math>f_z = f_t \times Z</math> ft-impeller frequency Z-Number of impeller blades</li> </ol>	Long-term outdoor harsh environment of the work, the blade surface will be fouling or ice leaves, so that uneven stress due to the blade load fatigue crack, and ultimately fracture.

## Reasoning: Fault detection in wind turbines

- Expert system for fault detection in wind turbines
- Reasoning strategy (forward reasoning):

Rule "rotor imbalance"			Diagnosis		Result
Symptoms	Weight	Confidence threshold	Confidence level	True value	T > K Rule is activated
spectrum (1X-based, spectrum into fir tree)	0.4	0.8	1.0	1.0	
phase (stable)	0.1	0.8	1.0	1.0	
Axis Trajectory (Oval)	0.2	0.8	1.0	1.0	
rotation feature (forward movement)	0.1	1.0	1.0	1.0	
vibration direction (radial)	0.05	1.0	1.0	1.0	
the critical vibration (the difference between horizontal and vertical direction is not significant)	0.05	0.8	1.0	1.0	
other (vibration varies with speed)	0.1	1.0	0.8	0.8	
CF= 0.98      K=0.7			T=0.9604		

# Example 2: KnowWE – System



Your trail: Main

≡

**d3web Home**

- Product Overview
- Downloads
- Success Stories
- History

**Users**

- Tutorials & Screencasts
- Documentation
- User FAQ
- Demos

**Developers**

- Getting Started
- How-Tos
- Developer FAQ
- API Documentation
- Sources ↗
- Repository ↗

**Get Social**

- Forum ↗
- YouTube ↗

**Miscellaneous**

- All pages
- Recent changes
- Installed plugins
- Contact / Impressum
- Privacy Statement

## KnowWE

KnowWE is a semantic wiki that provides the possibility to define and maintain ontologies together with strong problem-solving knowledge. Ontologies can be formulated using the RDF(S) or OWL standard: RDF(S) can be included by using the turtle markup, whereas the markup for OWL uses the Manchester syntax. Additionally, KnowWE provides markups for the formalization of the strong problem-solving methods of [d3web-Core](#). This includes

- Decision trees
- Scoring and abstraction rules
- Decision tables
- Flowchart knowledge with DiaFlux models
- Set-covering models

KnowWE offers a toolset for the entire knowledge engineering life cycle including

- Testing and visually debugging knowledge bases
- Refactoring of knowledge bases, e.g., rename question, rename solution, etc.
- Evaluation of knowledge bases including the analysis of knowledge bases for anomalies
- Continuous integration of knowledge by knowledge builds

Since, KnowWE provides a sophisticated plugin architecture, most of the features are included as plugins. New plugins can be easily introduced; see the developer [How-Tos](#) for more details.


## System Requirements

Technically, KnowWE builds on the development of the wiki [JSPWiki](#) ↗.

The installation of KnowWE requires the Java Platform, Standard Edition JDK 6 or better.

For operating systems, the distribution runs with MS-Windows XP, Windows 7 and Mac OS 10.6 or better.

We officially support Firefox 3.6 or higher and Google Chrome. Currently, the Internet Explorer is not officially supported.



Search 🔍

Info ▾

More... ▾

«

## Example 2: KnowWE – Overview

(cf. Baumeister et al. (2011) KnowWE: A Semantic Wiki for Knowledge Engineering. Applied Intelligence 35:323–344)

KnowWE – the Knowledge Wiki Environment:

<https://www.d3web.de/Wiki.jsp?page=KnowWE>

- Knowledge-based system – based on semantic wiki
- Semantic wiki: Text can be complemented with semantic annotations – “semantic formatting”
- KnowWE:
  - Includes problem solving methods, built on a semantic Wiki
  - Also: integrates strong problem solving methods of *d3web*:  
<https://www.d3web.de/>
  - Several classes of problem solving instances – classification, construction
  - Also: Knowledge acquisition; machine learning approaches can be connected/used, like learning from Big data, information extraction/learning from text via plugins.

# KnowWE – Example Article

KnowWE: CDBadIgnitionTiming

http://localhost:8080/KnowWE/Wiki.jsp?page=CDBadIgnitionTiming

KnowWE

CDBadIgnitionTiming

G'day, *Joachim Baumeister* (authenticated) Log out My Prefs

Quick Navigation

Your trail: Main, Car-Diagnosis, PageIndex, CDBadIgnitionTiming, ObjectInfoPage

View Attach Info Edit More... Interview

**Car Diagnosis**

Main

Master V1

Test Suite

**Solutions**

Update - Clear - Show findings

**Established Solutions:**

- Bad ignition timing

**Suggested Solutions:**

- Clogged air filter

**Administration**

Main

Page Index

Rename Concepts

Installed TagHandlers

KnowWE 20100119\_07:48

JSPWiki v2.8.3-dm-19

**Bad Ignition timing**

adapted from Wikipedia

**General**

"Ignition timing ", is the process of setting the time that a spark will occur in the combustion chamber (during the power stroke) relative to piston position and crankshaft angular velocity.

Setting the correct ignition timing is crucial in the performance of an engine. The ignition timing affects many variables including engine longevity, fuel economy, and engine power.

**Typical Symptoms**

Bad ignition timing can have multiple symptoms: For example, some starting problems are related to bad ignition timing. So a bad ignition timing can result in a barely or not starting engine. Additionally, bad ignition timing can be the reason for performance problems such as delayed take-off or weak acceleration. Furthermore, bad ignition timing frequently causes engine noises such as ringing or knocking.

**Repair It**

Engine noises

- ☐ ringing
- ☒ knocking
- ☐ -?-

Setting the **knocking** while monitoring engine power output with a dynamometer is one way to correctly set the ignition timing. After advancing or retarding the timing, a corresponding change in power output will usually occur. A load type dynamometer is the best way to accomplish this as the engine can be held at a steady speed and load while the timing is adjusted for maximum output.

Please note that newer engines typically use electronic ignition systems (ignition controlled by a computer). Most computers from original equipment manufacturers (OEM) are not able to be modified so changing the timing is not possible.

Overall timing changes are still possible, depending on the engine design. Aftermarket engine control units allow to make changes to the timing map. This allows the timing to be advanced or retarded based on various engine applications.

**Knowledge**

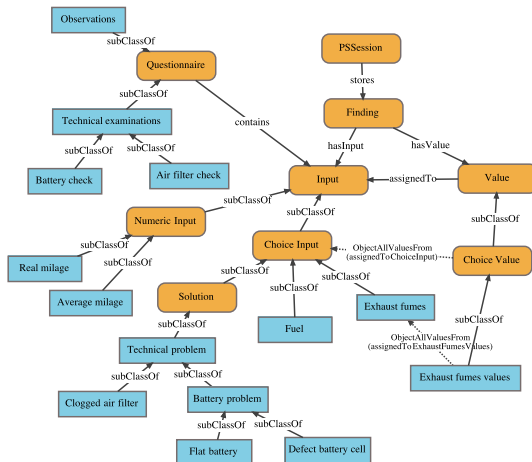
```

IF NOT ((Engine start = does not start OR Engine start = engine barely starts))
THEN Bad ignition timing = N5

IF (Engine noises = ringing OR Engine noises = knocking)
THEN Bad ignition timing = PS
  
```

# KnowWE – Knowledge Graph

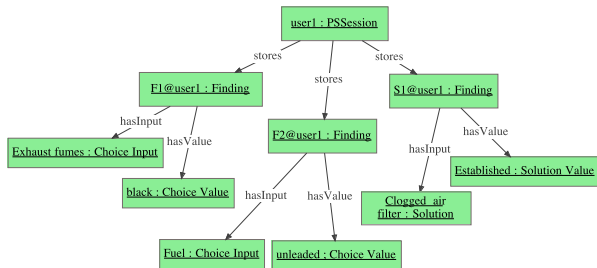
**Fig. 4** (Color online)  
Connecting the application ontology with the task ontology by subclassing. Concepts of the task ontology are depicted in *rounded rectangles (orange)*, whereas concepts of the application ontology are represented by *non-rounded rectangles (blue)*



# KnowWE – Problem Solving

- Example: Problem solving session

**Fig. 6** Example: A problem-solving session instantiating two findings entered by the user (*Exhaust fumes = black* and *Fuel = unleaded*) and instance representing a derived solution (*Clogged air filter*)



# Example: Learning Diagnostic Scores

## ① What is a Diagnostic Score?

**Problem description:** Diagnostic scores are applicable, if the goal is to find the best solution from a list of candidates, where the choice does typically not depend on one key feature like in decision trees, but on the overall picture. Further, it is necessary, that each feature is meaningful by itself and must not to be logically combined for rating solutions (this would form another pattern, e.g. heuristic decision tables).

**Solution description:** The basic idea of diagnostic scores is to rate all possible individual features systematically both in favor and against all solution objects. The rating should be done by a standardized scheme, e.g. with different predefined categories or with numbers or pseudo-probabilistic percentages. Since they are usually quite robust against small variations, a rather coarse scheme with few categories is often sufficient. Each solution object has an account, where it sums up the positive or negative ratings from the observed features in the case. The solution object with the highest account (and a sufficient difference to the second best) is chosen.

(Puppe (2000). Knowledge Formalization Patterns. Proc. EKAW)

## ② How can we learn diagnostic scores from data?

↪ (cf. Atzmueller et al. (2006) Semi-automatic learning of simple diagnostic scores utilizing complexity measures. Artificial Intelligence in Medicine. 37:19–30)



## Learning Diagnostic Scores

### Definition 1 (Attribute and Attribute Values)

Let  $\Omega_A$  be the universe set of all attributes available in the problem domain. A value  $v \in \text{dom}(a)$  assigned to an attribute  $a \in \Omega_A$  is called an *attribute value* and we call  $\Omega_{\mathcal{F}}$  the set of all possible attribute values (findings) in the given problem domain. An attribute value  $f \in \Omega_{\mathcal{F}}$  is denoted by  $a:v$  for  $a \in \Omega_A$  and  $v \in \text{dom}(a)$ . The set  $F_a \subseteq \Omega_{\mathcal{F}}$  of possible attribute values for a given attribute  $a$  is defined as  $F_a = \{f \in \Omega_{\mathcal{F}} \mid f = a:v \wedge v \in \text{dom}(a)\}$ . Each attribute value  $f \in \Omega_{\mathcal{F}}$  is defined as a possible input of a diagnostic knowledge system.

Other common names for attribute values are *findings* and *observations*. A diagnostic system usually comes up with a solution for a given problem. These solutions are defined as diagnoses.

## Learning Diagnostic Scores (1)

### Definition 2 (Diagnosis)

Let  $d$  be a *diagnosis* representing a possible output, i.e., a solution, of the diagnostic knowledge system. We define  $\Omega_{\mathcal{D}}$  to be the universe of all possible diagnoses for a given problem domain.

A symbolic state  $dom(d) = \{unlikely, probable\}$  is assigned to a diagnosis  $d \in \Omega_{\mathcal{D}}$  with respect to a given problem. The value range of a diagnosis  $d$  is denoted by  $dom(d)$ .

## Learning Diagnostic Scores (2)

A collection of cases is given as an input to the learning system.

### Definition 3 (Case)

A *case*  $c$  is defined as a tuple

$$c = (\mathcal{F}_c, \mathcal{D}_c, \mathcal{I}_c),$$

where  $\mathcal{F}_c \subset \Omega_{\mathcal{F}}$  is a set of attribute values given as input to the case. Often  $\mathcal{F}_c$  is also called the set of *observations* for the given case. The set  $\mathcal{D}_c \subseteq \Omega_{\mathcal{D}}$  contains the diagnoses describing the solution of the case  $c$ , and  $\mathcal{I}_c$  contains additional (meta-) information describing the case  $c$  in more detail. The set of all possible cases for a given problem domain is denoted by  $\Omega_C$ .

For the learning task, we consider a case base  $CB \subseteq \Omega_C$  containing all available cases that have been previously solved.

## Learning Diagnostic Scores (3)

A simple and intuitive way for representing inferential knowledge is the utilization of *diagnostic scores*. Then, simple scoring rules are applied.

### Definition 4 (Simple Scoring Rule)

A *simple scoring rule*  $r$  is denoted as follows:

$$r = f \xrightarrow{s} d,$$

where  $f \in \Omega_{\mathcal{F}}$  is an attribute value, and  $d \in \Omega_{\mathcal{D}}$  is the targeted diagnosis. For each rule a symbolic confirmation category  $s \in \Omega_{scr}$  is attached with

$$\Omega_{scr} = \{ S_3, S_2, S_1, 0, S_{-1}, S_{-2}, S_{-3} \}.$$

## Learning Diagnostic Scores (4)

- Let  $\Omega_R$  be the universe of all possible rules for the sets  $\Omega_{\mathcal{F}}$ ,  $\Omega_{\mathcal{D}}$  and  $\Omega_{scr}$ .
- We call  $\mathcal{R} \subseteq \Omega_R$  the *rule base* containing the inferential knowledge of the problem domain.
- Scores are used to represent a qualitative approach for deriving diagnoses with symbolic confirmation categories.
- These symbolic categories state the degree of confirmation or disconfirmation of a particular diagnosis.

# Learning Diagnostic Scores (5)

**Algorithm 1** (*Learning simple diagnostic scores*).

---

**Require** Case base  $CB \subseteq \Omega_C$

1. **for all** diagnoses  $d \in \Omega_D$  **do**
  2. Learn a diagnostic profile  $P_d$
  3. **for all** attributes  $a \in \{a \mid a \in \Omega_{(A)}, \exists f \in F_a, f \in P_d\}$  **do**
  4. **for all** attribute values  $f \in F_a$  **do**
  5. Construct binary variables  $D, F$  for  $d$  and  $f$ , which measure if  $d$  and  $f$  occur in cases of the case base  $CB$ .
  6. Compute  $\chi^2_{fd} = \chi^2(F, D)$
  7. **if**  $\chi^2_{fd} \geq \chi^2_\alpha$  **then**
  8. Compute the correlation /  $\phi_{fd}$  coefficient  
 $\phi_{fd} = \phi(F, D)$
  9. **if**  $|\phi_{fd}| \geq \text{threshold}_c$  **then**
  10. Compute the quasi-probabilistic score  $qps$ ,  
 $qps = \text{sgn}(\phi_{fd}) \cdot \text{prec}(r)(1 - \text{FAR}(r))$  using the pseudo-rule:  $f \rightarrow d$
  11. Map the  $qps$ -score to a symbolic category  $s$  using a conversion table
  12. Apply background knowledge to validate the diagnostic scoring rule, if available
  13. Create a diagnostic scoring rule (if valid):  $f \xrightarrow{s} d$
- 

Contingency table:

	$D = \text{true}$	$D = \text{false}$
$F = \text{true}$	a	b
$F = \text{false}$	c	d

The frequency counts denoted in the table are defined as follows:

$$a = N(D = \text{true} \wedge F = \text{true}), \quad b = N(D = \text{false} \wedge F = \text{true}), \\ c = N(D = \text{true} \wedge F = \text{false}), \quad d = N(D = \text{false} \wedge F = \text{false}),$$

where  $N(\text{cond})$  is the number of times the condition  $\text{cond}$  is true for cases  $c \in \mathcal{C}$ .

## Learning Diagnostic Scores (6)

- To identify dependencies between attribute values and diagnoses, we apply the  $\chi^2$ -test *for independence* with a certain threshold  $\chi^2_\alpha$  corresponding to confidence level  $\alpha$ . For binary variables the formula for the  $\chi^2$ -test simplifies to

$$\chi^2(F, D) = \frac{(a + b + c + d)(ad - bc)^2}{(a + b)(c + d)(a + c)(b + d)}. \quad (1)$$

- We require a certain minimal support threshold for an attribute value  $f$  co-occurring with diagnosis  $d$ . The default threshold is set to 5, i.e., the attribute value has to occur together with the diagnosis at least five times.

## Learning Diagnostic Scores (7)

- For the remaining dependencies we generate rules described as follows:
  - For all dependent tuples  $(F, D)$  we derive the quality of the dependency using the  $\phi$ -coefficient

$$\phi(F, D) = \frac{ad - bc}{\sqrt{(a+b)(c+d)(a+c)(b+d)}}, \quad (2)$$

which measures the degree of association between two binary variables. We use it to discover positive or negative dependencies.

- If the absolute value of  $\phi(F, D)$  is less than a certain threshold *threshold<sub>c</sub>*, i.e.,  $|\phi(F, D)| < \text{threshold}_c$ , then we do not consider this weak dependency for rule generation.
- If  $\phi(F, D) < 0$ , then we obtain a negative association between the two variables, and we will generate a rule  $f \xrightarrow{s} d$  with a negative category  $s$ .
- If  $\phi(F, D) > 0$ , then we construct a rule  $f \xrightarrow{s} d$  with a positive category  $s$ .



## Learning Diagnostic Scores (8)

- For determining the exact symbolic confirmation category of the remaining rules  $r = f \rightarrow d$ , we utilize two measures used in diagnosis: *precision* and the *false alarm rate (FAR)*, which is also known as the *false positive rate*, or  $1 - \text{specificity}$ .
- The precision of a rule  $r$  is defined as

$$\text{prec}(r) = \frac{TP}{TP + FP}, \quad (3)$$

- The false alarm rate *FAR* for a rule  $r$  is defined as

$$\text{FAR}(r) = \frac{FP}{FP + TN}. \quad (4)$$

## Learning Diagnostic Scores (9)

- The symbols  $TP$ ,  $TN$ ,  $FP$  denote the number of *true positives*, *true negatives*, and *false positives*, retrieved from the contingency table. For a positive dependency between attribute value  $f$  and diagnosis  $d$ ,  $TP = a$ ,  $TN = d$  and  $FP = b$ . For a negative dependency the situation is different, since we try to predict the absence of the diagnosis, so  $TP = b$ ,  $TN = c$  and  $FP = a$ .
- To score the dependency, we first compute a *quasi probabilistic score* ( $qps$ ) which we then map to a symbolic category – e.g., using a table mapping intervals to symbols
- The numeric  $qps$  score for a rule  $r$  is computed as follows

$$qps(r) = \text{sgn}(\phi(D, F)) \cdot \text{prec}(r)(1 - \text{FAR}(r)) . \quad (5)$$

# Learning Diagnostic Scores: Context

## SonoConsult: Documentation and diagnostic KBS

(for details: Puppe et al. (2008) Application and Evaluation of a Medical Knowledge System in Sonography (SONOCONSULT), Proc. ECAI)

### Sonographie

**Name, Vorname:** Mustermann, Manuel, 01.10.40  
**Fragestellung:** Oberbauch-Screening; Leberzirrhose

**Befund** vom 17.11.04; gute Untersuchungsbedingungen

**Leber:**Höhe in MCL 11 cm; Tiefe in MCL 10 cm; verplumpt; Oberfläche unregelmäßig, knotig, gebuckelt; Unterrand stumpf; Verformbarkeit deutlich vermindert; Binnenstruktur deutlich echovermehrt; mittleres Reflexmuster; Kalibersprung der Pfortaderäste intrahepatisch; Rarefizierung der Pfortaderäste intrahepatisch

**D. hepatocholedochus:** Durchmesser 5 mm; unauffällig

**Gallenblase:**unauffällig

**Milz:**längs 14 cm, tief 6 cm; Parenchym unauffällig

**Pfortadersystem:** Pfortaderdurchmesser 14 mm; keine wesentliche Zunahme des Durchmessers bei

Inspiration; Milzvenendurchmesser 12 mm; Hinweis auf wiedereröffnete Nabelvene

*Duplexsonographie: Pfortader Fluß orthograd mit gleichmäßigem Flußprofil, Flußgeschwindigkeit 12 cm/s; Milzvene Flußgeschwindigkeit 12 cm/s; wiedereröffnete Nabelvene*

**Flüssigkeit im Abdomen:** freie Flüssigkeit im Sinne von Aszites, mäßig ausgeprägt

**Abdominelle Gefäße:** Arteria hepatica (duplexsonographisch): nicht durchgeführt

**Vena cava:** unauffällig

**Lymphknoten:** in beurteilbaren Regionen nicht erkennbar bzw. nicht vergrößert

**Pleuraerguss:** beidseits nicht nachweisbar

**Perikarderguss:** nicht nachweisbar

### **Beurteilung:**

*Schlussfolgerungen von SonoConsult:*

**Portale Hypertension (K76.6) bei Leberzirrhose (K74.6); portalhypertensiv bedingter Aszites (R18); Splenomegalie (R16.1) bei portaler Hypertension (K76.6)**

Die diagnostischen Schlussfolgerungen müssen durch den Untersucher/Befunder geprüft werden.

# Learning Diagnostic Scores: Results

No.	threshold <sub>c</sub>	MR <sup>a</sup>	RBS <sup>b</sup> (uncrt. cat) <sup>c</sup>	AF <sup>d</sup>	SC <sup>e</sup>	ACC <sup>f</sup>
Experiment E0: no knowledge used, no pruning						
1	0.2	30.58 ± 15.93	2201.50 (1586.40)	391.60	3.52	0.94
2	0.3	20.75 ± 10.14	1493.90 (929.70)	348.90	3.26	0.93
3	0.4	14.82 ± 6.93	1067.20 (566.80)	293.70	2.99	0.91
4	0.5	10.93 ± 5.18	786.80 (334.00)	245.80	2.69	0.90
5	0.6	8.46 ± 3.71	609.20 (205.40)	208.00	2.38	0.84

28

M. Atzmueller et al.

Experiment E1: using partition class and abnormality knowledge, no pruning

1	0.2	8.35 ± 5.09	600.90 (395.10)	177.10	2.59	0.89
2	0.3	5.97 ± 3.25	430.10 (241.90)	149.20	2.36	0.87
3	0.4	4.50 ± 2.09	323.90 (155.60)	130.20	2.10	0.85
4	0.5	3.37 ± 1.45	242.90 (90.70)	112.60	1.77	0.85
5	0.6	2.65 ± 1.03	190.50 (55.80)	99.90	1.53	0.79

Experiment E2: using partition class and abnormality knowledge and pruning

1	0.2	3.60 ± 4.05	258.90 (53.10)	122.70	1.09	0.89
2	0.3	2.86 ± 2.19	205.60 (17.40)	99.50	0.99	0.87
3	0.4	2.42 ± 1.31	174.50 (6.20)	90.00	0.95	0.85
4	0.5	2.12 ± 0.96	152.70 (0.50)	82.50	0.92	0.85
5	0.6	1.87 ± 0.73	134.70 (0.00)	77.90	0.91	0.79

<sup>a</sup> MEAN RULES: average number of rules per diagnostic score.<sup>b</sup> RULE BASE SIZE: total number of rules (confirmation categories equally weighted, cf. Section 4).<sup>c</sup> Total number of rules with an uncertain confirmation category.<sup>d</sup> APPLIED ATTRIBUTE VALUES: number of different attribute values used.<sup>e</sup> SCORE CATEGORIES: number of different confirmation categories used.<sup>f</sup> Accuracy of the learned diagnostic scores.

# Summary

## What Did We Learn?

- Intuitions of KBS, basic considerations and architecture
- Expert systems as special knowledge based systems
- Problem solving methods (and types) of expert systems
- Relations to AI and Machine Learning
- Examples of prominent knowledge representations
- KBS by Example – from diagnostics to knowledge wikis and a machine-learning-driven approach (diagnostic score)