Arbeitsblätter zur Lehrveranstaltung

Inferenzmethoden

des Studiengangs

Informatik

an der

Technischen Universität Ilmenau

apl. Prof. Dr.-Ing. habil. Rainer Knauf

Inhaltsverzeichnis

1 Einführung in die Wissensverarbeitung

Einige Teilgebiete der Künstlichen Intelligenz (KI)

- Wissensrepräsentation,
- maschinelles Beweisen (Deduktion),
- Sprachverarbeitung,
- Bildverarbeitung,
- Spielprogramme,
- algorithmisches Lernen (induktive und analoge Inferenz),
- Robotik,

- KI–Sprachen und
- **Wissensbasierte Systeme**.

Eigenschaften Wissensbasierter Systeme

1. Wissensbasierte Systeme sind in der Lage, **Darstellungen**(!) **des Wissens in symbolischer Form** getrennt von Wissensverarbeitungskomponenten **aufzubewahren**.

   > Eine scharfe geistige Trennung zwischen Wissen an sich und formalen Systemen zur Darstellung und Verarbeitung desselben ist unabdingbar für das Verständnis der Grundkonzepte der Wissensverarbeitung. Diese Trennung ist der erste Schritt zur Demystifizierung des Begriffs „Künstliche Intelligenz“.

2. Wissensbasierte Systeme können aus der abgespeicherten Information Schlüsse und Konsequenzen ableiten, die in ihnen nicht explizit repräsentiert; ihnen aber inhärent sind. Die dazu angewandten Methoden heißen **Inferenzmethoden**.

   Typischerweise ist der Inferenz Wissensbasierter Systeme ein **Nichtdeterminismus** innewohnend.

3. Eine weitere Eigenschaft Wissensbasierter Systeme ist die **Erklärungsfähigkeit**, d.h. sie sollten in der Lage sein, auf Anforderung die Kette der Folgerungsschritte (resp. Resolutionsschritte) dem Nutzer transparent und plausibel zu machen. Die Akzeptanz solcher Systeme bei Nutzern hängt wesentlich von der Qualität der Erklärungsfähigkeit ab.

4. Eine aus der Sicht des Autors gleichfalls wichtige Eigenschaft Wissensbasierter Systeme ist die **Lernfähigkeit**. Dazu gehört z.B. die Fähigkeit, anhand von Nutzerinformationen (etwa über den Erfolg oder Misserfolg vergangener Sitzungen) neue Wissensinhalte abzuleiten.

2 Architektur und Realisierungskonzepte der Komponenten Wissensbasierter Systeme

Architektur Wissensbasierter Systeme

Die Abbildung 1 zeigt die Umsetzung der o.g. Eigenschaften durch eine Struktur und das (andeutungsweise) funktionelle Zusammenwirken der Komponenten.

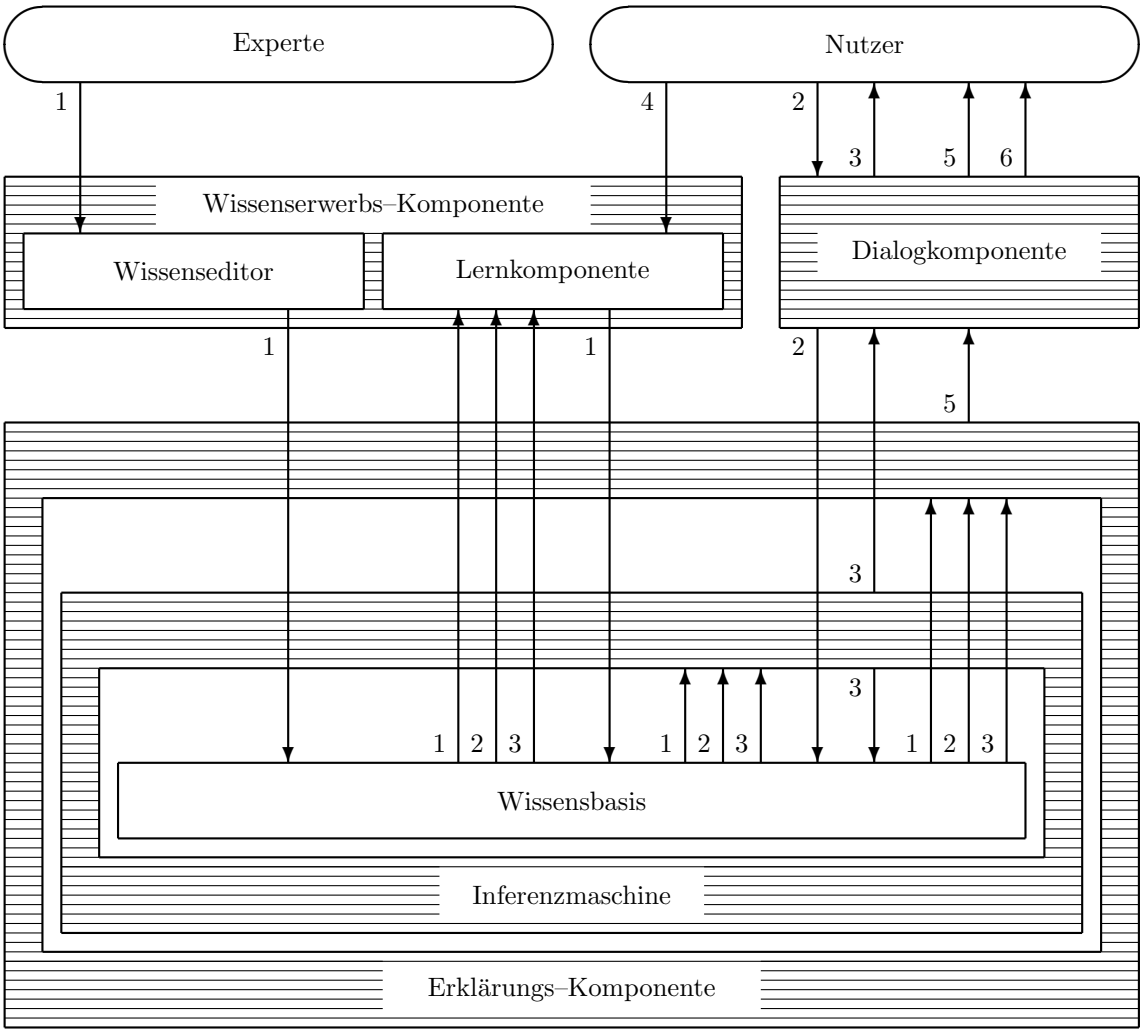
2.1 Wissensdarstellungen und ihre Abbildung auf PROLOG–Strukturen

Typen des darzustellenden Wissens

- bereichsspezifisches Expertenwissen (welches sich während der Konsultation mit dem Nutzer nicht ändert),
- fallspezifisches Faktenwissen (welches der Nutzer während der Konsultation eingibt) und
- Zwischen– und Endergebnisse (die das System während der Konsultation ableitet).

Ebenen des darzustellenden Wissens

1. die Darstellung auf Nutzerebene (problemorientierte Darstellung),
2. die Darstellung in der (den) Programmiersprache(n), in der das Wissensbasierte System implementiert ist (implementierungsorientierte Darstellung) und
3. die computerinterne Darstellung (Bytes).



1 . . . bereichsspezifisches Expertenwissen
2 . . . fallspezifisches Faktenwissen
3 . . . Zwischen– und Endergebnisse
4 . . . Erfolgsbewertung des Nutzers
5 . . . Erklärungs–Information
6 . . . Bedien–Information (Hilfe–Komponente)

Abbildung 1: Architektur Wissensbasierter Systeme

Formen der problemorientierten Wissensdarstellung

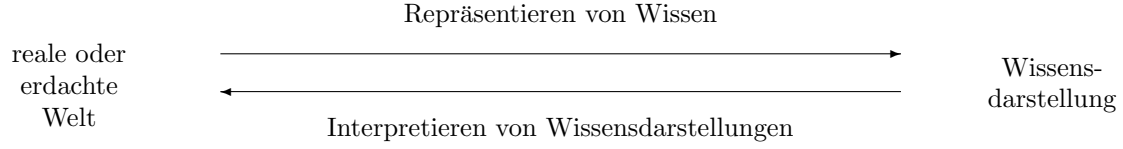
- Semantische Netze,
- Frames,
- der Prädikatenkalkül der ersten Stufe (PK1),
- Produktionssysteme und
- prozedurale Repräsentationen.

Methoden der problemorientierten Wissensdarstellung

Die **deklarativen Repräsentationsideen** basieren auf der Annahme, dass die Darstellung von Wissen unabhängig davon betrachtet werden kann, wie es verarbeitet wird. Es wird eine saubere Trennung zwischen Darstellung und Verarbeitung von Wissen vorgenommen.
Prozedurale Wissensrepräsentationen betonen den Verarbeitungsaspekt. Hierzu wird dem zu repräsentierenden Wissen zusätzlich Wissen über die Verarbeitung dieses Wissens (Metawissen) (explizit oder implizit) hinzugefügt und damit die o.g. scharfe Trennung „verwaschen“ bzw. ganz aufgehoben.

2.1.1 Prädikatenkalkül der ersten Stufe

Die Theorie des PK1 als Wissensdarstellungskalkül aus der Sicht der Wissensverarbeitung



Repräsentation:
Eine **Repräsentation** im PK1 ist eine eindeutige Zuordnung von Prädikaten–, Funktions– und Individuensymbolen zu Prädikaten, Funktionen und Objekten über einer Objektmenge U.
Interpretation:
Eine **Interpretation** des PK1 besteht aus einer Objektmenge U zusammen mit einer Abbildung $\mathcal{I}$, welche jedem n-stelligen Funktionssymbol eine n-stellige Funktion und jedem n-stelligen Prädikatensymbol eine n-stellige Relation über U zuweist. Eine solche Interpretation heißt auch „Interpretation $\mathcal{I}$ in U“.
Zusammenhang Prädikat – Relation:
Ein n-Tupel von Elementen aus U gehört zu der einem Prädikatensymbol p zugewiesenen Relation R, gdw. das dem Prädikatensymbol p zugeordnete Prädikat dieses n-Tupel auf *wahr* abbildet.

Sortenlogik

1. Variablen und Konstanten haben eine Sorte S, die als Index mitgeführt wird, z.B. X_S, c_S usw.
2. Jedem n-stelligen Funktionssymbol f ist ein $(n+1)$-Tupel von Sorten $[S_1, \ldots, S_n, S_{n+1}]$ zugeordnet. Es werden nur Terme der Gestalt $f(t_1, \ldots, t_n)$ zugelassen, bei denen t_i von der Sorte S_i ist; die Sorte des gesamten strukturierten Terms $f(t_1, \ldots, t_n)$ ist dann S_{n+1}.
3. Jedem n-stelligen Prädikat p ist ein n-Tupel $[S_1, \ldots, S_n]$ von Sorten zugeordnet und es sind nur Atomformeln $p(t_1, \ldots, t_n)$ zugelassen, bei denen t_i von der Sorte S_i ist.

Semantisch gesehen wird einer Sorte S für jede Interpretation $\mathcal{I}$ eine Teilmenge des Universums U zugeordnet: $\mathcal{I}(S) \subseteq U$. Bei der Verarbeitung sortenbehafteter Prädikate und Terme müssen die vorgenommenen Substitutionen sortentreu sein.

Prädikatenlogik mit Gleichheit

Gleichheit von Termen:

1. $t \equiv t$ gilt fr jeden Term t.
2. Fr alle n-stelligen Funktionssymbole f und alle Terme t_i und t'_i gilt:

$$\begin{array}{ll} & (f(t_1,\ldots,t_n) \equiv f(t'_1,\ldots,t'_n)) \\ \text{, wenn} & (t_1 \equiv t'_1 \wedge \ldots \wedge t_n \equiv t'_n) \end{array}$$

3. Fr alle n-stelligen Prdikatensymbole p und alle Terme t_i und t'_i gilt:

$$\begin{array}{ll} & (p(t_1,\ldots,t_n) \rightarrow p(t'_1,\ldots,t'_n)) \\ \text{, wenn} & (t_1 \equiv t'_1 \wedge \ldots \wedge t_n \equiv t'_n) \end{array}$$

Universum:

$$\begin{array}{rcl} [a] & = & \{b : a \equiv b\} \\ U_{\equiv} & = & \{[a] : a \in U\} \end{array}$$

Paramodulationsregel (auf HORN–Klauseln angewandt):
Sei $\{K_1,\ldots,K_n\}$ eine Menge von Fakten und Regeln, $H \equiv \bigwedge_{i=1}^{m} H_i$ eine Frage (eine Hypothese).
Eine der Klauseln K_j sei

$$(s \equiv t) \leftarrow \bigwedge_{k=1}^{p} B_k \quad ,$$

Es gebe Termeinsetzungen ϑ_1 und ϑ_2 in die Variablen von s und einem in einem der H_i (etwa H_l mit $1 \leq l \leq m$) vorkommenden Term r, so dass $\vartheta_1(s) \equiv \vartheta_2(r)$.

$$M \equiv \bigwedge_{i=1}^{n} K_i \wedge \neg H$$

ist kontradiktorisch ($kt\ M$), wenn M nach Ersetzen von H in M durch

$$(\bigwedge_{i=1}^{l-1} \vartheta_2(H_i)) \wedge \vartheta_2(H'_l) \wedge (\bigwedge_{k=1}^{p} \vartheta_1(B_k)) \wedge (\bigwedge_{i=l+1}^{m} \vartheta_2(H_i))$$

noch immer kontradiktorisch ist.
H'_l entsteht hierbei aus H_l durch Ersetzen von r durch $\vartheta_1(t)$.
Desweiteren mu bei der Prdikatenlogik mit Gleichheit zur vollstndigen Resolution zu jeder Wissensbasis die Klausel

$$\forall X(X \equiv X)$$

hinzugefgt werden und der Satz von ROBINSON erweitert werden:

$$M \equiv \bigwedge_{i=1}^{n} K_i \wedge \neg H$$

ist kontradiktorisch, gdw. durch eine endliche Anzahl von Anwendungen der Resolutionsmethode und/oder der Paramodulationsregel die leere Klausel

$$\Box \equiv false \leftarrow true$$

entsteht.

Die Praxis der Logischen Programmierung

(Nutz-)Effekte der Beschränkung auf HORN–Klauseln:

1. über HORN–Klauseln gibt es ein korrektes und vollständiges Ableitungsverfahren, d.h.

   $$\{K_1, \ldots, K_n\} \models H, \text{ gdw.} \{K_1, \ldots, K_n\} \vdash_* H$$

   wobei $\vdash_*$ (hier) die Ableitbarkeit nach ROBINSON durch Hintereinanderanwendung der Resolutionsmethode bedeutet.

2. O.g. Ableitungsverfahren ist algorithmisierbar.

3. In einer (PROLOG–) Wissensbasis aus HORN–Klauseln lassen sich keine Widersprüche formulieren, d.h. es gilt für jede PROLOG–Wissensbasis

   $$\bigwedge_{i=1}^{n} K_i \neq false$$

2.1.2 Semantische Netze

In Abbildung 2 sind 2-stellige Relationen über einer Menge von Personen in Form eines Semantischen Netzes beispielhaft illustriert.

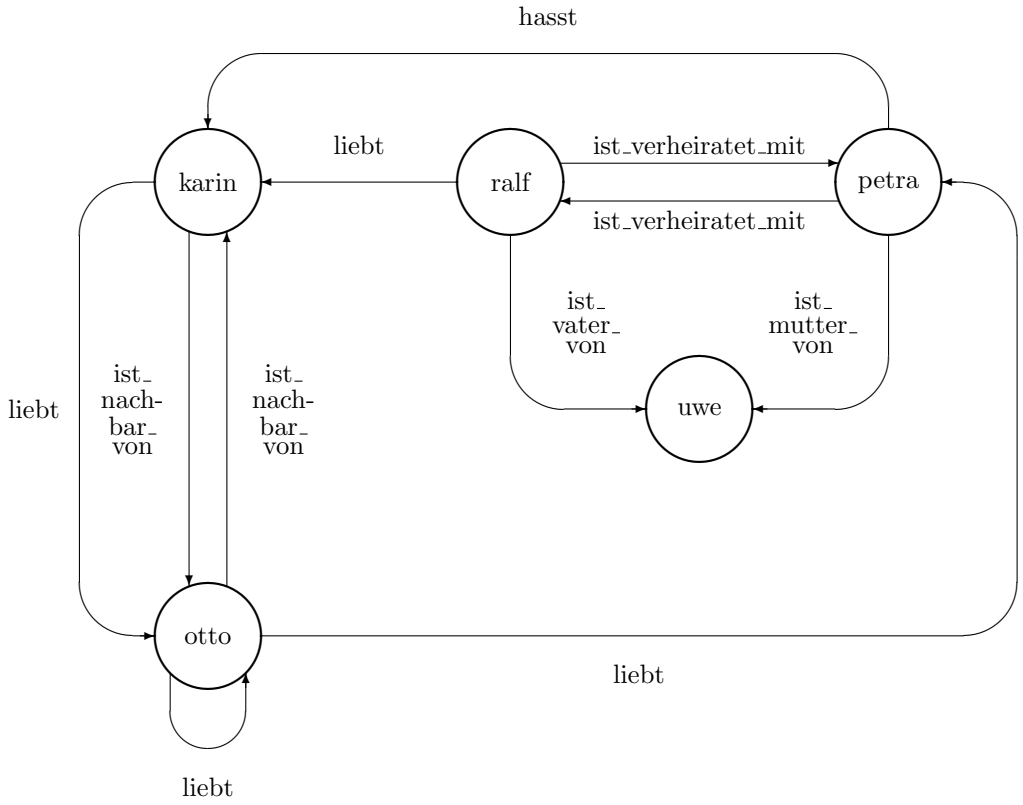


Abbildung 2: Ein Semantisches Netz

2.1.3 Frames

Eine allgemeine Notationsform eines Frames zeigt die Abbildung 3.
Begriffe und Eigenschaften:

Framename		
Oberkonzepte		
Unterkonzepte		
Slotname 1:	Slotwert 1	
	Merkmal 1_1:	Ausprägung 1_1
	...	
	Merkmal 1_{n_1}:	Ausprägung 1_{n_1}
...		
...		
Slotname m:	Slotwert m	
	Merkmal m_1:	Ausprägung m_1
	...	
	Merkmal m_{n_m}:	Ausprägung m_{n_m}

Abbildung 3: Notationsform eines Frames

- Wenn die Slots eines Frames (teilweise) gefüllt sind, so spricht man von einer **Instanz** eines Frames.
- **Defaultwerte** eines Slots sind vorläufige Werte, die in Ermangelung besseren Wissens als „Arbeitshypothese“ gewählt werden. Sie werden i.allg. durch das Voranstellen von *default* gekennzeichnet und können ohne weiteres „überschrieben“ werden.
- Von **generischen Werten** spricht man, wenn Slotinhalte für alle Instanzen unveränderlich sind.
- Durch **Slotbedingungen** (**Constraints**) werden die Wertebereiche von Slots eingeschränkt.
- In einem Slot können auch Regelsysteme stehen, die auf Anforderung in Aktion treten, etwa um einen Slotwert zu berechnen.
- Auch „werteberechnende“ Prozeduren können Slotinhalt sein.

  Eine Stelle, welche die Bedingungen prüft, unter denen eine solche Prozedur ausgeführt wird, heißt **Dämon**. Ein Dämon bildet demnach die Schnittstelle zwischen deklarativen und prozeduralen Elementen der Wissensrepräsentation „Frame“.

  Solche prozeduralen Zusätze können in mehreren Formen auftreten, z.B. als

  – *if–needed* – Prozeduren (wenn–nötig – Prozeduren), die den Slotwert nicht automatisch, sondern nur auf Anforderung berechnen, was bei aufwendigen Berechnungen sinnvoll erscheint, oder als

  – *if–added* – Prozeduren (wenn–belegt – Prozeduren), die von einem Dämon genau dann angestoßen werden, wenn ein bestimmter Slot einen bestimmten Wert erhalten hat.
- Durch die Möglichkeit der Unterbrigung (wiederum) ganzer Frames in einem Slot sind beliebig tief strukturierte Slotinhalte denkbar.

Mit Hilfe solcher Frames kann man nun Wissen über Konzepte einer (realen oder erdachten) Welt notieren. Beispiele hierfür zeigt die Abbildung 4.

2.1.4 Objektorientierte Darstellungen

Grundgedanken:

Personenkraftwagen		
Oberkonzepte: *Kraftfahrzeuge*		
Unterkonzepte: *Limousine, Caravan, Transporter*		
Motor:	*kraftstoffgetrieben*	
	Kraftstoff:	*default Benzin*
	Zylinder:	*>= 2*
	Hubraum:	*> 500 cm^3*
Getriebe:	*default Schaltgetriebe*	
	Gänge:	*default 5*
Räder:	*4*	

Limousine	
Oberkonzepte: *Personenkraftwagen*	
Unterkonzepte: *Kleinwagen, Mittelklassewagen, Wagen gehobener Klasse*	
Gesamtmasse:	*< 2 t*
Sitzplätze:	*default 5*

Abbildung 4: Beispiele für Frames

1. Um komplexe Strukturen (Software, Wissen, ...) beherrschbar zu halten, sollten sie **modularisiert** werden. Die dabei entstehenden Moduln nennt man dann schlicht **Objekte**, an die man gewisse Anfragen (Botschaften) stellen (senden) kann, die ggf. beantwortet werden. Die bessere Beherrschbarkeit ergibt sich durch
   - die Erhöhung der übersichtlichkeit mit der Blockbildung,
   - eine gewisse „Klarheit“ durch das Verbergen unwichtiger Details in den Objekten und
   - die Anwendungsunabhängigkeit durch die Möglichkeit der Datenabstraktion.
2. Eine gewisse „Kompaktheit“ der Repräsentation (von Daten, Wissen, ...) erreicht man durch die Zusammenfassung von Objekten mit gemeinsamen Eigenschaften zu Klassen. Da diese Gemeinsamkeiten innerhalb einer Klasse „vererbt“ werden können, brauchen sie nur einmal notiert werden. Dieser Vererbungsbegriff ist derselbe, wie der aus der Frame–Hierarchie bekannte.

Ein **Objekt** besteht aus lokalen Daten und einer Menge von (auf diese Daten zugreifenden) Operationen, die auch **Methoden** genannt werden. Andere als die durch die Methoden bereitgestellten Operationen können mit den Daten nicht ausgeführt werden.
Objekte kommunizieren über Botschaften (bzw. Nachrichten). Eine auszusendende Nachricht besteht i.allg. aus 3 Teilen:

- Name des Empfänger–Objektes,
- Name der auszuführenden Operation und
- eine Liste von Argumenten (die gleichfalls Objekte sind), mit denen die Operation ausgeführt werden soll.

Wenn ein Empfänger eine solche Botschaft bekommt, so muss er

1. prüfen, ob die auszuführende Operation von ihm auch bereitgestellt wird,
2. die Operation anstoßen und

3. eine Antwort an den Sender zurückmelden.

Die dadurch verursachten Effekte sind zweierlei Art:

1. lokale Effekte:
   Die Operation greift auf lokale Daten zu und verändert diese gegebenenfalls.
2. globale Effekte:
   Die Operation bewirkt, dass Botschaften an andere Objekte geschickt werden, was deren Zustand gegebenenfalls verändert.

Objekte mit gleichen Operationen und gleich strukturierten Daten werden zusammengefasst; ihre Gemeinsamkeiten werden einmalig durch eine **Klasse** definiert. Die einzelnen Objekte heißen dann **Instanzen** dieser Klasse. Die Klassen selbst sind gleichfalls Objekte.
Auch innerhalb von Klassen gibt es hierarchische Beziehungen, damit bestehende Gemeinsamkeiten kompakt notiert werden können:

> Die Klasse B ist eine Spezialisierung der Klasse A (oder A ist Oberklasse von B), wenn
>
> 1. die Klasse B zusätzliche Datenstrukturen gegenüber A aufweist oder
> 2. B zusätzliche Operationen gegenüber A aufweist,
>
> und ansonsten isomorph ist. B erbt alle Operationen und Datenstrukturen aus A. Hierbei ist es zulässig, dass gleichnamige Operationen in B anders implementiert sind als in A.

Die Ober–Unterklassen – Relationen zwischen Objekten können verschieden starken Resriktionen unterliegen:

1. In einer „strengen“ Variante wird von der Hierarchie Baumförmigkeit gefordert, d.h. jedes Objekt (mit Ausnahme der Wurzel) hat genau eine Oberklasse.

   Dies ist z.B. die Vererbungsform des „Klassikers“ objektorientierter Sprachen, SMALLTALK 80.

   Die durch diese Restriktion erzielte Effizienz erkauft man sich durch eine gewisse Schwerfälligkeit bei horizontalen Transfers in der Hierarchie. Hierzu muss man sich i.allg. „globaler Daten“ bedienen, die dem Konzept der Objektorientierung widersprechen.

2. In einer liberaleren Variante, die auch **Mehrfachvererbung** oder **multiple Vererbung** genannt wird, kann ein Objekt mehrere Oberklassen haben.

   Dies ist besonders dann von Interesse, wenn man für ein Objekt Eigenschaften verschiedener „Herkunft“ repräsentieren möchte. So kann man z.B.

   - als Oberklassen von „Licht“ sowohl „Welle“ als auch „Teilchen“ zulassen und
   - als Oberklassen von „Weizen“ sowohl „Pflanze“ als auch „Nahrungsmittel“ zulassen.

Eine in jedem Falle geforderte Bedingung ist, dass kein Objekt direkte oder indirekte Oberklasse seiner selbst sind darf. Eine graphische Darstellung der Klassenhierarchie muss demnach einen azyklischen Graphen ergeben.

2.1.5 Produktionssysteme

Theoretischen Grundlagen:

- HORN – Klauseln des Aussagenkalküls für die Wissensdarstellung und
- Schlussregeln für die Inferenz, wobei in allen bekannten Systemen als einzige Schlussregel der „modus ponens“ nach ARISTOTELES

  $$\frac{A \quad A \to B}{B}$$

  Anwendung findet.

Komponenten und deren Zusammenwirken:

1. eine Faktenbasis, d.h. Menge einfacher Aussagen
2. eine Menge von Produktionsregeln der o.g. Form und
3. eine Steuerkomponente, die **zyklisch**
   (a) alle auf die aktuelle Faktenbasis anwendbaren Regeln (die **Konfliktmenge**) bestimmt (sämtliche Bedingungen dieser Regeln müssen in der Faktenbasis als wahre Aussagen verzeichnet sein),
   (b) nach einer festgelegten Strategie (z.B. prioritätengesteuert) aus der Konfliktmenge eine Regel auswählt, d.h. eine **Konfliktlösung** vornimmt und
   (c) den Aktionsteil der gewählten Regel ausführt, der i.allg. darin besteht, die Faktenbasis zu verändern.

2.1.6 Prozedurale Repräsentationen aus dem Blickwinkel der Wissensverarbeitung

Bei dieser Repräsentationsform besteht das Wissen allein aus Prozeduren, die eine nicht explizit repräsentierte Faktenmenge, nämlich die Daten, manipuliert. Die Reihenfolge der Aufrufe der Prozeduren muss für das konkrete Problem „vorgedacht“ sein. Alle Entscheidungsprozesse, die während der Problemlösung auftreten, müssen bei der Formulierung der Prozeduren vorgesehen werden. Es gibt praktisch keine nichtdeterminierten Prozeduren, d.h. für jede Situation gibt es genau einen Lösungsweg (und nicht etwa mehrere alternativ anwendbare Prozeduren) für ein und denselben Aufruf. Den Gewinn an Effizienz erkauft man sich durch einen Verlust an universeller Verwendbarkeit.

2.1.7 Transformation der problemorientierten Wissensdarstellung auf die Maschinenebene

Ein „Weg“ des Wissens von der (dem „Expertenhirn“ entlockten) problemorientierten Ebene bis zur computerinternen Darstellung wird durch die Abbildung 5 illustriert.

2.2 Inferenzmechanismen

Grundlage für das Inferieren ist **Metawissen**, z.B.

1. Wissen über Verallgemeinerungsmöglichkeiten
2. Wissen über Möglichkeiten des analogen Schließens
3. Wissen über Integritätsbedingungen einer Wissensbasis
4. Wissen über Eigenschaften von Aussagen
5. Wissen über Verfahren des sukzessiven Ableitens zur Simulation des Folgerns

Noch eine Stufe höher (die man vielleicht *Meta–Metawissen* bezeichnen könnte) könnte man das für das Ableiten nötige Wissen über

- Eigenschaften von Ableitungsverfahren wie z.B. Vollständigkeit und Korrektheit sowie
- das Verhältnis zwischen Folgern und Ableiten

einordnen.
Vollständig ist ein Ableitungsverfahren genau dann, wenn alle Aussagen H, die aus einer Menge von Aussagen $\{A_1, \ldots, A_n\}$ *folgen*, nach diesem Ableitungsverfahren auch *ableitbar* sind.
Korrekt ist ein Ableitungsverfahren genau dann, wenn alle Aussagen H, die aus einer Menge von Aussagen $\{A_1, \ldots, A_n\}$ *ableitbar* sind, aus dieser auch *folgen*.
Das Verhältnis zwischen Folgern und Ableiten lässt sich demnach wie folgt formulieren: $\{A_1, \ldots, A_n\} \models H$, wenn [*gdw.*] durch ein korrektes [*und vollständiges*] Ableitungsverfahren gezeigt werden kann , dass $\{A_1, \ldots, A_n\} \vdash H$.

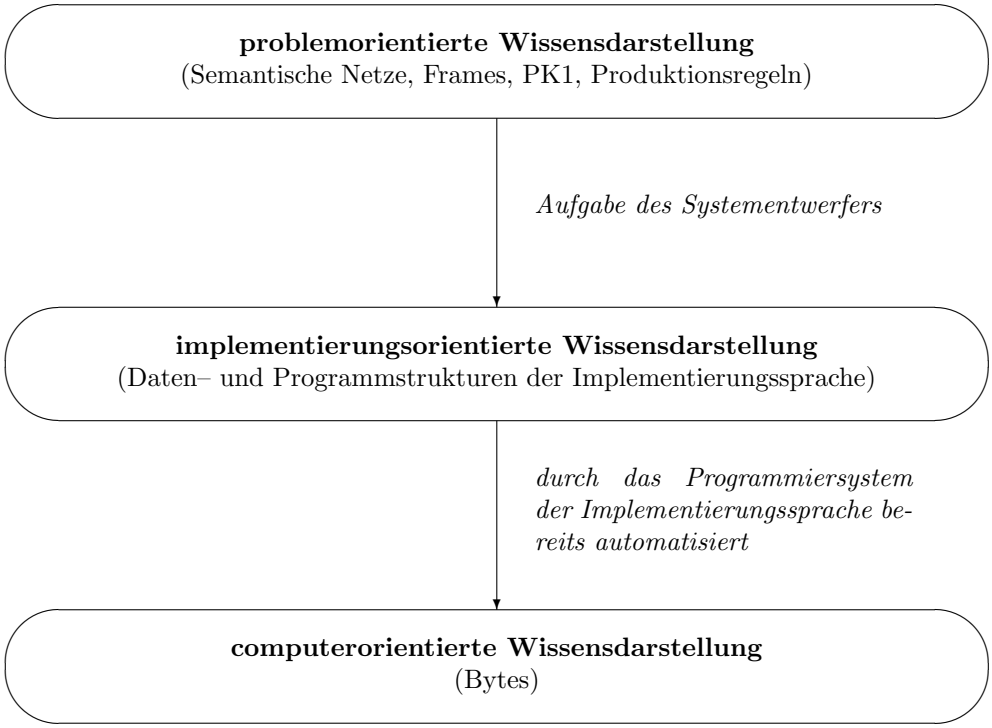


Abbildung 5: Transformationen zwischen den Ebenen der Wissensdarstellung

Die Grenze zwischen Wissen und Metawissen ist oft fließend. Deshalb werden in der Fachliteratur mitunter Mehrebenenmodelle vorgestellt, in denen Zwischenstufen zwischen dem bereichsspezifischen Expertenwissen und dem Metawissen definiert werden.
Beispiel: 4–Ebenen–Modell für das modellbasierte Wissensakquisitionswerkzeug KADS–II (siehe Tabelle 1).

2.2.1 Deduktion

Das Ziel der Deduktion ist es, genau das **Folgern** zu simulieren:

> Sei $M = \{A_1, \ldots, A_n\}$ eine Menge von Aussagen, H eine Aussage (Hypothese).
>
> H **folgt aus** M ($M \models H$), falls jede Interpretation, die zugleich alle Aussagen A_i wahr macht (jedes **Modell** von M), auch H wahr macht (auch ein Modell von H ist):

$$M \models H \quad \text{, wenn} \quad ag \quad (\bigwedge_{i=1}^{n} A_i \rightarrow H)$$

$$\text{bzw.} \quad kt \quad (\bigwedge_{i=1}^{n} A_i \wedge \neg H)$$

Hülleneigenschaften der Folgerungsrelation

Sei A die Menge aller (denkbaren) Aussagen; $P(A)$ die Potenzmenge davon, d.h. die Menge aller Teilmengen von A. Der Folgerungsbegriff definiert eine Relation Fl zwischen Elementen aus $P(A)$. Zur Relation Fl gehören alle Paare $[M, Fl(M)]$ mit $M, Fl(M) \in P(A)$, bei denen $Fl(M)$ die Menge der aus M folgerbaren Aussagen ist:

$$Fl(M) := \{H | M \models H\}$$

Tabelle 1: Mehrebenen–Modell in KADS–II

Wissenskategorie	Organisation	Wissenstypen
Metawissen ↓ *steuert*	Strategien	Pläne, Metaregeln, Annahmen
Aufgabenwissen ↓ *wendet an*	Aufgabenzerlegung	Aufgaben, Problemlösungsmethoden, Kontrollstrukturen
Inferenzwissen ↓ *benutzt*	Inferenzstruktur	primitive Inferenzaktionen, Wissensrollen
Bereichswissen	Bereichstheorie, Bereichsmodelle, Benutzertheorie, Fallmodell	Begriffe, Eigenschaften, Relationen, situationsspezifische Informationen

Es gelten folgende Sätze:

1. **Satz der Einbettung**: $M \subseteq Fl(M)$
   Jede in M enthaltene Aussage folgt auch aus M.

2. **Satz der Monotonie**: $M_1 \subseteq M_2 \longrightarrow Fl(M_1) \subseteq Fl(M_2)$
   Wenn eine Menge M_1 Teilmenge einer Menge M_2 ist, dann ist auch die Menge der aus M_1 folgerbaren Aussagen eine Teilmenge der aus M_2 folgerbaren Aussagen.

3. **Satz der Abgeschlossenheit**: $Fl(Fl(M)) \subseteq Fl(M)$
   Hat man für eine gegebene Menge M von Aussagen $Fl(M)$ gebildet, so führt das nochmalige Bilden der daraus folgerbaren Aussagen nicht zu einer Erweiterung gegenüber $Fl(M)$.

4. **Endlichkeitssatz**:
   $M \models H \longrightarrow \exists M^*((M^* \subseteq M) \wedge (M^* \models H) \wedge (card(M^*) < \infty))$
   Wenn $M \models H$, so gibt es stets eine endliche Teilmenge M^* von M, so dass $M^* \models H$.

5. **Ableitungstheorem**: $(M \models (H_1 \rightarrow H_2)) \longrightarrow (M \cup \{H_1\} \models H_2)$
   Wenn $M \models (H_1 \rightarrow H_2)$, dann $M \cup \{H_1\} \models H_2$.

6. **Deduktionstheorem**: $(M \cup \{H_1\} \models H_2) \longrightarrow (M \models (H_1 \rightarrow H_2))$
   Wenn $M \cup \{H_1\} \models H_2$, dann $M \models (H_1 \rightarrow H_2)$.

7. **Unvollständigkeitssatz** (zugunsten der Verständlichkeit etwas „schlampig“ formuliert):

   Für jede widerspruchsfreie Menge M von Aussagen gilt: Es gibt eine Aussage H derart, dass weder $M \models H$ noch $M \models \neg H$.

Grundlage für die Realisierbarkeit korrekter und vollständiger Ableitungsverfahren „$\vdash$“ ist der GÖDELsche **Vollständigkeitssatz**:

> Es gibt einen Algorithmus, der die aus einer gegebenen Menge von Aussagen folgerbaren Aussagen aufzählen kann.

Wenn es diesen gibt, muss es natürlich erst recht ein algorithmisierbares Verfahren geben, welches von einer gegebenen Hypothese H entscheidet, ob sie aus einer Menge von Aussagen M folgt. Zwei derartige Verfahren sind

- die Resolutionsmethode und deren Hintereinanderanwendung nach ROBINSON „$\vdash_*$“ und
- das natürliche Schließen nach GENTZEN „$\vdash_{NI}$“.

Die Resolutionsmethode und deren Hintereinanderanwendung nach ROBINSON „$\vdash_*$“

Auf HORN–Klauseln angewandt stellt sie sich wie folgt dar:

Sei $\{K_1, \ldots, K_n\}$ eine Menge von Fakten und Regeln,

$$H \equiv \bigwedge_{i=1}^{m} H_i$$

eine Frage (eine Hypothese).

Eine der Klauseln K_j sei

$$A \leftarrow \bigwedge_{k=1}^{p} B_k \quad ,$$

wobei A und die B_k Atomformeln sind und alle auftretenden Variablen bezüglich der ganzen Klausel allquantifiziert sind.

Es gebe Termeinsetzungen (Substitutionen) ϑ_1 und ϑ_2 in die Variablen von A und eines der H_i (etwa H_l mit $1 \leq l \leq m$), so dass $\vartheta_1(A) \equiv \vartheta_2(H_l)$.

$$M \equiv \bigwedge_{i=1}^{n} K_i \wedge \neg H$$

ist kontradiktorisch ($kt \quad M$), wenn M nach Ersetzen von H in M durch

$$\underbrace{(\bigwedge_{1=1}^{l-1} \vartheta_2(H_i))}_{\text{die ersten } l-1 \text{ Teilhypothesen}} \quad \wedge \quad \underbrace{(\bigwedge_{k=1}^{p} \vartheta_1(B_k))}_{\substack{\text{Einsetzen des Körpers der Klausel statt } H_1}} \quad \wedge \quad \underbrace{(\bigwedge_{i=l+1}^{m} \vartheta_2(H_i))}_{\text{restliche Teilhypothesen}}$$

noch immer kontradiktorisch ist.

Substitution:

Eine **Substitution** ϑ ist eine Abbildung von einer endlichen Menge X von Variablen in die Menge der Terme. Sie wird i.allg. notiert als Menge von Paaren:

$$\vartheta = \{[x, t] | x \in X, t = \vartheta(x)\}$$

Substitution in strukturierten Termen:

$$\vartheta(f(Y, Z)) = f(\vartheta(Y), \vartheta(Z))$$

Hintereinanderausführung von Substitutionen „$\circ$“:

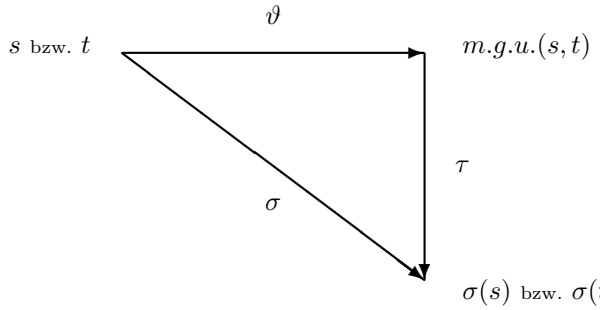
$$\sigma \circ \vartheta(t) = \sigma(\vartheta(t))$$

Substitutionen, die zwei Terme (syntaktisch) identisch machen (und nur solche sind für die Resolutionsmethode von Interesse), unifizieren diese bzw. heißen **Unifikator**:

ϑ **unifiziert** zwei Terme s und t (oder: ist **Unifikator** von s und t), falls $\vartheta(s) = \vartheta(t)$ ist.

Sinnvollerweise sucht man bei der Resolutionsmethode nicht irgendeinen Unifikator, sondern den **allgemeinsten Unifikator** (*most general unifier*), damit nicht durch eine unnötige Spezialisierung der Terme (zukünftige) Resolutionsmöglichkeiten in nachvolgenden Resolutionsschritten verbaut werden.

> Eine Substitution ϑ heißt **allgemeinster Unifikator** zweier Terme s und t, d.h. $\vartheta = m.g.u.(s,t)$, wenn
>
> 1. ϑ ein Unifikator von s und t ist und
> 2. für jeden Unifikator σ von s und t eine Substitution τ existiert, so dass $\sigma = \tau \circ \vartheta$ ist:



Um einen Algorithmus angeben zu können, welcher den allgemeinsten Unifikator zweier gegebener Terme s und t ermittelt, benötigen wir zunächst den Begriff der **Unterscheidungsterme**, die wie folgt gebildet werden:

> Lese s und t simultan von links nach rechts. Man nehme die erste Stelle, an der sich s und t unterscheiden und nenne $\hat{s}$ und $\hat{t}$ diejenigen Teilterme von s bzw. t, die an dieser Stelle beginnen. $\hat{s}$ und $\hat{t}$ sind die Unterscheidungsterme von s und t.

Ein Algorithmus, der für zwei Terme s und t entscheidet, ob sie unifizierbar sind und im positiven Falle den allgemeinsten Unifikator liefert, ist der folgende:

> **Algorithmus vom allgemeinsten Unifikator:**
> Setze $i := 0$, $\vartheta_i := id$ (identische Substitution[1]), $s_i := s$ und $t_i := t$.
> $\odot$ Teste, ob s_i und t_i syntaktisch übereinstimmen.
> Falls ja, setze $\vartheta := \vartheta_i$. ϑ ist allgemeinster Unifikator. *stop*
> Falls nein, bilde die Unterscheidungsterme $\hat{s}_i$ und $\hat{t}_i$ von s_i und t_i.
> Teste, ob $\hat{s}_i$ und $\hat{t}_i$ beides keine Variablen sind.
> Falls ja, so sind s und t nicht unifizierbar.[2] *stop*
> Falls nein (sei etwa $\hat{s}_i$ eine Variable), so teste, ob $\hat{t}_i$ die Variable $\hat{s}_i$ enthält.
> Falls ja, so sind s und t nicht unifizierbar. *stop*
> Falls nein, setze
> $\vartheta' := \vartheta_i \cup \{\vartheta_i(\hat{s}_i, \hat{t}_i)\}$[3]
> $s' := \vartheta'(s_i)$ und $t' := \vartheta'(t_i)$
> $i := i + 1$
> $\vartheta_i := \vartheta'$ und $s_i := s'$ und $t_i := t'$
> und gehe nach $\odot$.

Mit der Resolutionsmethode und der (darin enthaltenen) Unifikation werden Resolutionsschritte vollzogen, deren Ziel der **Satz von ROBINSON** benennt:

$$M = \bigwedge_{i=1}^{n} K_i \wedge \neg H$$

[1] D.h. jede Variable wird durch sich selbst ersetzt.

[2] Wenn es sich um zwei strukturierte Terme handelt, dann haben sie verschiedene Funktionssymbole (sonst wären es nicht die *Unterscheidungs*terme) und sind schon deshalb nicht unifizierbar.

[3] D.h. ersetze auf allen rechten Seiten im Unifikator ϑ_i die Variable $\hat{s}_i$ durch den Term $\hat{t}_i$.

ist kontradiktorisch ($kt \quad M$), gdw. durch wiederholte Anwendung der Resolutionsmethode schließlich (in endlich vielen Schritten) die Hypothese durch die (kontradiktorische) leere Klausel $false \leftarrow true$ (Symbol: $\square$) ersetzt werden kann.

Da sich bekanntlich nicht jeder beliebige Ausdruck in eine Konjunktion von HORN–Klauseln umformen lässt, scheint eine Verallgemeinerung dieser Methode auf Ausdrücke in Klauselform sinnvoll:

Sei $\{K_1, \ldots, K_n\}$ eine Menge von Ausdrücken in Klauselform. Es gebe darin 2 variablenfremde[4] Klauseln K_k und K_l mit

$$\begin{aligned} K_k &\equiv \bigvee_{i=1}^{m_A} A_i \leftarrow \bigwedge_{i=1}^{m_B} B_i \equiv \bigvee_{i=1}^{m_A} A_i \vee \bigvee_{i=1}^{m_B} \neg B_i \\ K_l &\equiv \bigvee_{i=1}^{m_C} C_i \leftarrow \bigwedge_{i=1}^{m_D} D_i \equiv \bigvee_{i=1}^{m_C} C_i \vee \bigvee_{i=1}^{m_D} \neg D_i \end{aligned}$$

(wobei A_i, B_i, C_i und D_i Atomformeln sind) derart, dass eine der A_i (etwa A_s) und eine der D_i (etwa D_t) miteinander unifizierbar sind, d.h. $\vartheta(A_s) \equiv \vartheta(D_t)$ mit $\vartheta = m.g.u.(A_s, D_t)$.

$$M \equiv \bigwedge_{i=1}^{n} K_i$$

ist kontradiktorisch ($kt \quad M$), wenn M nach dem Hinzufügen von

$$\begin{aligned} K &\equiv \bigvee_{i=1}^{s-1} \vartheta(A_i) \vee \bigvee_{i=s+1}^{m_A} \vartheta(A_i) \vee \bigvee_{i=1}^{m_C} \vartheta(C_i) \leftarrow \\ &\quad \bigwedge_{i=1}^{m_B} \vartheta(B_i) \wedge \bigwedge_{i=1}^{t-1} \vartheta(D_i) \wedge \bigwedge_{i=t+1}^{m_D} \vartheta(D_i) \\ &\equiv \bigvee_{i=1}^{s-1} \vartheta(A_i) \vee \bigvee_{i=s+1}^{m_A} \vartheta(A_i) \vee \bigvee_{i=1}^{m_C} \vartheta(C_i) \vee \\ &\quad \bigvee_{i=1}^{m_B} \vartheta(\neg B_i) \vee \bigvee_{i=1}^{t-1} \vartheta(\neg D_i) \vee \bigvee_{i=t+1}^{m_D} \vartheta(\neg D_i) \end{aligned}$$

noch immer kontradiktorisch ist. K heißt *Resolvente* von K_k und K_l.

Allein durch die Resolutionsmethode ist das ROBINSON'sche Ableitungsverfahren bei Ausdrücken in Klauselform allerdings noch nicht vollständig; hier bedarf es noch einer sogenannten **Faktorenregel**:

Sind in einer Klausel K zwei Atomformeln A_i und A_j innerhalb des Klauselkopfes oder –körpers untereinander unifizierbar mit $\vartheta = m.g.u.(A_i, A_j)$, so ist $\vartheta(K)$ eine Resolvente.[5]

Der Satz von ROBINSON müsste dann wie folgt verallgemeinert werden:

$$M = \bigwedge_{i=1}^{n} K_i$$

ist kontradiktorisch ($kt \quad M$), gdw. sich durch wiederholte Anwendung von Resolutionsmethode und Faktorenregel schließlich (in endlich vielen Schritten) die (kontradiktorische) leere Klausel $false \leftarrow true$ (Symbol: $\square$) ableiten lässt.

[4] D.h. sie dürfen keine gleich benannten Variablen haben. Ggf. muss vor der Resolution eine entsprechende Variablenumbenennung vollzogen werden. Dies ist ohne weiteres möglich, da gleich benannte Variablen in verschiedenen Klauseln völlig unabhängig voneinander sind.

[5] Die dabei syntaktisch identisch werdenden Atomformeln im Klauselkopf oder –körper in der Resolvente werden dabei nur nur ein Mal notiert.

Da es durch dieses Verfahren sicher sehr viele Wege geben kann, aus eine Konjuktion von Ausdrücken in Klauselform die leere Klausel $\square$ abzuleiten, erscheint die Suche nach *einem* derartigen Weg kaum systematisierbar. Zwei Möglichkeiten, diese Suche effizient zu gestalten, sind:

1. das Löschen „überflüssiger“ Klauseln

   „überflüssig“ sind

   (a) tautologische Klauseln, d.h. solche, in denen die gleiche Atomformel A sowohl im Klauselkopf als auch im Klauselkörper vorkommt.

   (b) Klauseln K_j, die von „allgemeineren“ Klauseln K_i *subsumiert* werden:

   K_i subsumiert K_j, falls

   i. durch ein und dieselbe Substitution ϑ jedes Literal aus K_i einem Literal aus K_j identisch wird und

   ii. K_i höchstens genauso viele Literale wie K_j hat.[6]

2. die Einschränkung der Regelanwendung auf bestimmte Fälle

   Es ist vernünftig, die Kontradiktorizität einer Klauselmenge M nicht in einer Teilmenge $M_1 \subset M$ zu suchen, die erfüllbar ist. Diesen Gedanken greift die *Stützmengenstrategie* (engl. *set of support*) auf:

   Wenn $M = M_1 \cup M_2$ (mit $M_1 \cap M_2 = \emptyset$) eine Klauselmenge mit erfüllbarem M_1 ist, dann lässt die Stützmengenstrategie für M nur solche Resolventenbildungen zu, für die mindestens eine der beteiligten Klauseln in M_2 ist oder durch (ggf. mehrere) Resolventenbildungen aus M_2 gewonnen wurde.

   Demnach sollte man auch die Faktorenregel sinnvollerweise nur auf Klauseln aus M_2 oder daraus gewonnenen Klauseln anwenden.

Ein weiteres korrektes und vollständiges Ableitungsverfahren ist:

Das natürliche Schließen nach GENTZEN „$\vdash_{NI}$“

Sei M eine Menge von Aussagen; H, H_1, H_2, $H*$, A und B Aussagen. Es gilt

1. Einbettung:
   $M \vdash_{NI} H$, wenn $H \in M$.
2. (a) UND–Einführung:
   $M \vdash_{NI} (H_1 \wedge H_2)$, wenn $M \vdash_{NI} H_1$ und $M \vdash_{NI} H_2$.
   (b) UND–Beseitigung:
   $M \vdash_{NI} H_1$ und $M \vdash_{NI} H_2$, wenn $M \vdash_{NI} (H_1 \wedge H_2)$.
   (c) ODER–Einführung:
   $M \vdash_{NI} (H_1 \vee H_2)$, wenn $M \vdash_{NI} H_1$ oder $M \vdash_{NI} H_2$.
   (d) ODER–Beseitigung:
   $M \vdash_{NI} H$, wenn $M \vdash_{NI} (H_1 \vee H_2)$ und $M \cup \{H_1\} \vdash_{NI} H$ und $M \cup \{H_2\} \vdash_{NI} H$.
   (e) IMPLIKATIONs–Einführung:
   $M \vdash_{NI} (H_1 \to H_2)$, wenn $M \cup \{H_1\} \vdash_{NI} H_2$.
   (f) IMPLIKATIONs–Beseitigung:
   $M \vdash_{NI} H_2$, wenn $M \vdash_{NI} (H_1 \to H_2)$ und $M \vdash_{NI} H_1$.
   (g) äQUIVALENZ–Einführung:
   $M \vdash_{NI} (H_1 \leftrightarrow H_2)$, wenn $M \vdash_{NI} (H_1 \to H_2)$ und $M \vdash_{NI} (H_2 \to H_1)$.
   (h) äQUIVALENZ–Beseitigung:
   $M \vdash_{NI} (H_1 \to H_2)$ und $M \vdash_{NI} (H_2 \to H_1)$, wenn $M \vdash_{NI} (H_1 \leftrightarrow H_2)$.

[6] Diese zweite Bedingung verhindert, dass eine Klausel die aus ihr durch die Anwendung der (gerade mühsam erarbeiteten und essentiell benötigten) Faktorenregel ggewonnenen Klauseln (Faktoren) subsumiert.

(i) NICHT–Einführung:
$M \vdash_{NI} \neg H$, wenn $M \cup \{H\} \vdash_{NI} H^*$ und $M \cup \{H\} \vdash_{NI} \neg H^*$.

(j) NICHT–Beseitigung:
$M \vdash_{NI} H$, wenn $M \vdash_{NI} \neg\neg H$.

(k) ALLQUANTOR–Einführung:
$M \vdash_{NI} \forall X A(X)$, wenn $M \vdash_{NI}^{(a)} A(a)$.

(l) ALLQUANTOR–Beseitigung:
$M \vdash_{NI}^{(a)} A(a)$, wenn $M \vdash_{NI} \forall X A(X)$.

(m) EXISTENZQUANTOR–Einführung:
$M \vdash_{NI} \exists X A(X)$, wenn $M \vdash_{NI} A(a)$.

(n) EXISTENZQUANTOR–Beseitigung:
$M \vdash_{NI} B$, wenn $M \vdash_{NI} \exists X A(X)$ und $M \cup \{A(a)\} \vdash_{NI}^{(a)} B$.

3. $M \vdash_{NI} H$, gdw. das durch sukzessives Hintereinanderanwenden der Regeln gemäß 1. und 2. möglich ist.

$\vdash_{NI}^{(a)}$ ... „Die Konstante a wird in keinem der Ableitungsschritte benutzt“

Nachteilig an dem Verfahren nach GENTZEN ist, dass es kaum algorithmisierbar ist:

1. Es gibt keine Methode, den nächsten Ableitungsschritt *zielgerichtet* zu unternehmen.

   Es gibt keine Möglichkeit zu evaluieren, wie nahe man der Lösung ist.

   Da jedoch stets mindestens eine der Regeln anwendbar ist[7], hilft hier auch kein Backtrack–Mechanismus zwecks Weitersuche in der Breite, denn die Tiefensuche gerät nie in eine Sackgasse.

2. Eine (auch denkbare) Breitensuche scheitert an der Größe des Suchraumes.

2.2.2 Induktion

Ziel der Induktion in der Wissensverarbeitung ist es,

- aus einer Menge elementarer wahrer Aussagen allgemeine Regeln abzuleiten bzw.
- aus allgemeinen Regeln und einem neuen Beispiel (Fakt) neue allgemeine Regeln zu generieren, die das Beispiel mit erfassen.

Die dazu im praktischen Einsatz befindlichen Verfahren sind solche, die Klassifizierungsregeln generieren:
Gegeben ist

- eine Menge von Objekten I,
- ein endlicher Merkmalsraum, in dem jedem Objekt aus I anhand der Ausprägungen seiner Merkmale genau ein Punkt zugeordnet werden kann,
- eine endliche Menge von Objektklassen U, die die Objektmenge I in disjunkte Teilmengen partitionieren:

  $$U = \Pi(I) = \{U_1, \ldots, U_n\}$$

  mit

  $$\bigcup_{i=1}^{n} U_i = I \qquad \textit{und} \qquad \forall i \forall j((1 \leq i \leq n) \wedge (1 \leq j \leq n) \wedge (i \neq j) \rightarrow (U_i \cap U_j = \emptyset))$$

  und

[7] Auch das ist eine interessante Eigenschaft des Ableitungsverfahrens, die gar nicht so selbstverständlich ist

- die bekannte Zugehörigkeit einiger (einer endlich großen Anzahl) Objekte aus I zu je einer Klasse aus U, d.h. Fakten der Art

$$(Klasse = K) \wedge \bigwedge_{i=1}^{n} (Merkmal_i = Auspraegung_{ij})$$

  Solche Objekte bekannter Klassenzugehörigkeit sollen im weiteren **Beispiele** genannt werden.

Gesucht ist

- ein Regelwerk, welches
  - beliebige Objekte anhand der Ausprägungen seiner Merkmale klassifiziert und
  - zu keinem der Beispiele im Widerspruch steht.

  Das Regelwerk aus HORN–Klauseln des PK0, d.h. aussagenlogische Ausdrücken der Art

$$(Klasse = K) \leftarrow \bigwedge_{i=1}^{n} (Merkmal_i = Auspraegung_{ij})$$

  bestehen.

Ansätze zur Lösung dieses Problems seien im folgenden anhand eines kleinen Beispiels demonstriert:

Gegeben sei

- eine Objektmenge I: *eine Menge von Personenkraftwagen der Mittelklasse*
- ein (der Darstellbarkeit halber 3-dimensional gehaltener) Merkmalsraum, der in Bild 6 gezeigt ist

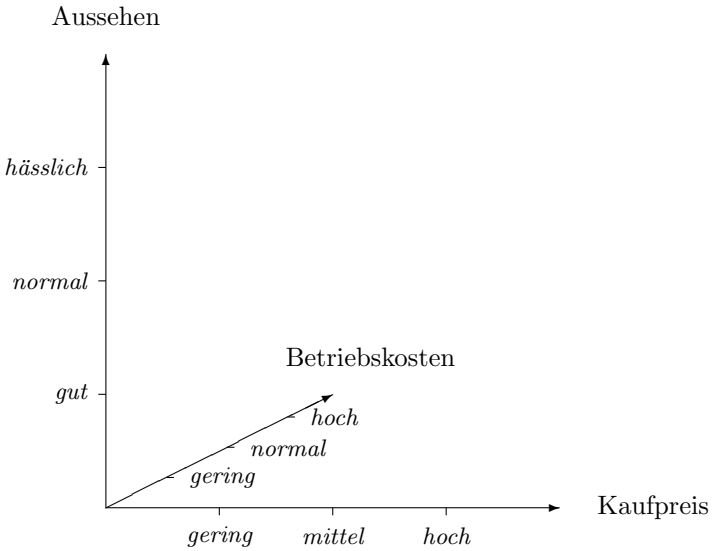


Abbildung 6: Ein Merkmalsraum

- eine Menge disjunkter Objektklassen $U = \{U_1, U_2, U_3\}$ mit

  U_1: Menge der empfehlenswerten PKWs
  U_2: Menge der indifferenten PKWs
  U_3: Menge der abratenswerten PKWs

- die Zugehörigkeit einiger Objekte aus I zu jeweils einer Klasse aus U, die in Tabelle 2 gezeigt ist

Tabelle 2: Eine Beispielmenge zur Induktion eines Regelwerks

Kaufpreis	Aussehen	Betriebskosten	KLASSE
mittel	*hässlich*	*gering*	*indifferent*
gering	*hässlich*	*gering*	*indifferent*
gering	*hässlich*	*normal*	*abratenswert*
hoch	*normal*	*normal*	*indifferent*
mittel	*gut*	*normal*	*empfehlenswert*
mittel	*normal*	*hoch*	*abratenswert*
mittel	*gut*	*hoch*	*abratenswert*
hoch	*normal*	*hoch*	*abratenswert*
gering	*normal*	*gering*	*empfehlenswert*
mittel	*normal*	*gering*	*empfehlenswert*
gering	*gut*	*gering*	*empfehlenswert*
hoch	*hässlich*	*normal*	*abratenswert*
hoch	*gut*	*normal*	*indifferent*
gering	*gut*	*normal*	*empfehlenswert*

Gesucht ist:

- ein Regelwerk, welches PKWs nach der Ausprägung ihrer Merkmale in eine der 3 Klassen einordnet und nicht im Widerspruch zu o.g. Beispielen steht.

Zur Lösung derartiger Aufgaben sind aus der Literatur eine ganze Reihe von Verfahren bekannt, von denen zwei hier vorgestellt werden sollen; das erste zeichnet sich durch besondere Trivialität aus, das zweite durch „Minimalität“ des entstehenden Regelwerkes zur Klassifikation:

„left–to–right“ – Regelgenerierung

Man partitioniert die Beispielmenge entsprechend der auftretenden verschiedenartigen Ausprägungen des ersten („linkesten“) Merkmals (seien es n_1 derartige Ausprägungen) in n_1 Teilmengen.

Im Beispiel sind diese Teilmengen die PKWs mit hohem, mittlerem und geringem Kaufpreis, wodurch die Beispielmenge in die drei Teil – Beispielmengen gemäß Tabelle 3 partitioniert wird.

Gehören alle Objekte einer dieser Teilmengen (z.B. die mit der Ausprägung $Auspraegung_{1j}$ mit $1 \leq j \leq n_1$ des Merkmals $Merkmal_1$) ein und derselben Klasse K_1 an, so kann man daraus schon eine Regel ableiten:

$$(Klasse = K_1) \longleftarrow (Merkmal_1 = Auspraegung_{1j})$$

Ansonsten muss man die Teilmengen ihrerseits in n_2 verschiedene Teilmengen entsprechend den n_2 auftretenden Ausprägungen des zweiten Merkmals partitionieren.

In der Tabelle 4 ist dies am Beispiel derjenigen Objekte getan worden, für die das Merkmal „Kaufpreis“ die Ausprägung *hoch* hat.

Gehören alle Objekte einer solchen Menge (z.B. die mit der Ausprägung $Auspraegung_{2k}$ mit $1 \leq k \leq n_2$ des Merkmals $Merkmal_2$) ein und derselben Klasse K_2 an, so entstehen Regeln mit 2 konjunktiv verknüpften Prämissen:

$$(Klasse = K_2) \longleftarrow (Merkmal_1 = Auspraegung_{1j}) \wedge (Merkmal_2 = Auspraegung_{2k})$$

Dies ist im Beispiel bei den hässlichen PKWs mit hohem Kaufpreis und bei den gut aussehenden PKWs mit hohem Kaufpreis der Fall:

Tabelle 3: Teil – Beispielmengen nach der ersten Partitionierung

Kaufpreis = *hoch*

Aussehen	Betriebskosten	KLASSE
normal	*normal*	*indifferent*
normal	*hoch*	*abratenswert*
hässlich	*normal*	*abratenswert*
gut	*normal*	*indifferent*

Kaufpreis = *mittel*

Aussehen	Betriebskosten	KLASSE
hässlich	*gering*	*indifferent*
gut	*normal*	*empfehlenswert*
normal	*hoch*	*abratenswert*
gut	*hoch*	*abratenswert*
normal	*gering*	*empfehlenswert*

Kaufpreis = *gering*

Aussehen	Betriebskosten	KLASSE
hässlich	*gering*	*indifferent*
hässlich	*normal*	*abratenswert*
normal	*gering*	*empfehlenswert*
gut	*gering*	*empfehlenswert*
gut	*normal*	*empfehlenswert*

Tabelle 4: Einige Teil – Beispielmengen nach der zweiten Partitionierung

(Kaufpreis = *hoch*) $\wedge$ (Aussehen = *hässlich*)

Betriebskosten	KLASSE
normal	*abratenswert*

(Kaufpreis = *hoch*) $\wedge$ (Aussehen = *normal*)

Betriebskosten	KLASSE
normal	*indifferent*
hoch	*abratenswert*

(Kaufpreis = *hoch*) $\wedge$ (Aussehen = *gut*)

Betriebskosten	KLASSE
normal	*indifferent*

$$(\text{Klasse} = \textit{abratenswert}) \longleftarrow (\text{Kaufpreis} = \textit{hoch}) \wedge (\text{Aussehen} = \textit{hässlich})$$
$$(\text{Klasse} = \textit{indifferent}) \longleftarrow (\text{Kaufpreis} = \textit{hoch}) \wedge (\text{Aussehen} = \textit{gut})$$

Das Verfahren wird so lange fortgesetzt, bis jedes Beispiel in mindestens einer Regel erfasst ist.

In unserem Beispiel führt dies zu folgendem Regelwerk, welches (der übersicht halber) in der Abbildung 7 als Baum dargestellt ist.

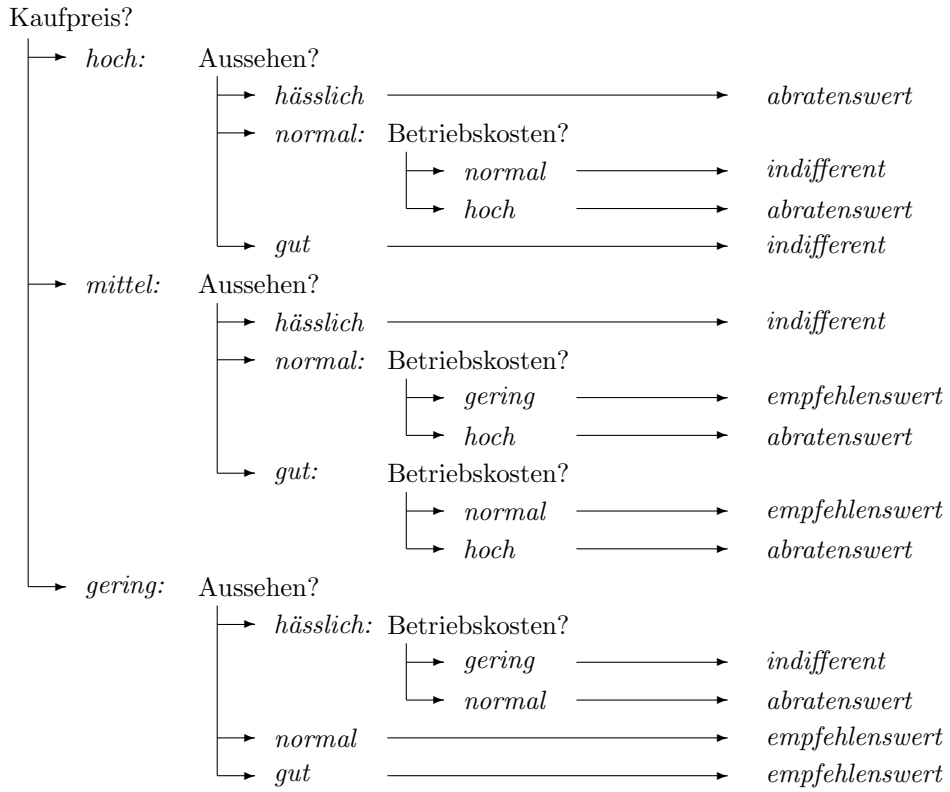


Abbildung 7: Regelbaum nach der „left–to–right“ – Regelgenerierung

In einer Produktionsregel–Notation sieht das Ergebnis etwa so aus:

$$\begin{aligned}
&(\text{Klasse} = \textit{abratenswert}) \longleftarrow (\text{Kaufpreis} = \textit{hoch}) \wedge \\
&\qquad\qquad\qquad\qquad\quad (\text{Aussehen} = \textit{hässlich}) \\
&(\text{Klasse} = \textit{indifferent}) \longleftarrow (\text{Kaufpreis} = \textit{hoch}) \wedge \\
&\qquad\qquad\qquad\qquad\quad (\text{Aussehen} = \textit{normal}) \wedge \\
&\qquad\qquad\qquad\qquad\quad (\text{Betriebskosten} = \textit{normal}) \\
&(\text{Klasse} = \textit{abratenswert}) \longleftarrow (\text{Kaufpreis} = \textit{hoch}) \wedge \\
&\qquad\qquad\qquad\qquad\quad (\text{Aussehen} = \textit{normal}) \wedge \\
&\qquad\qquad\qquad\qquad\quad (\text{Betriebskosten} = \textit{hoch}) \\
&\ldots
\end{aligned}$$

Es entstehen 13 Produktionsregeln. Wenn man bedenkt, dass diese aus 14 Beispielen hervorgegangen sind, konnte hier offensichtlich nicht viel durch die Induktion verallgemeinert werden.

Ein optimierendes Verfahren zur Generierung einer minimalen Anzahl von Regeln ist die

Regelgenerierung nach dem ID3 – Algorithmus

Der Grundgedanke ist die Verwendung der **Entropie der Information** (ein Maß für die Aussagekraft, den Informationsgehalt bzw. allgemein für die „Ordnung“ die der betrachteten Welt innewohnend ist) zur Optimierung eines Regelbaumes. Wenn bei einem Versuch P n verschiedene Ereignisse (Ergebnisse) auftreten können, so ist die Entropie der Information über das Versuchsergebnis

$$H(P) = -\sum_{i=1}^{n} p_i \cdot ld(p_i)$$

wobei p_i die Wahrscheinlichkeit des Auftretens des i-ten Ereignisses beim Versuch P ist.
Dieser Wert ist bei gleichwahrscheinlichem Auftreten aller n möglichen Versuchsergebnisse am größten. Je „schiefer“ die Verteilung der Häufigkeiten der n verschiedenen Versuchsergebnisse ist (m.a.W.: je „ungleicher“ die Wahrscheinlichkeiten sind), desto geringer wird dieser Wert.
Beim ID3–Algorithmus werden nun die Entropien der einzelnen Merkmale über die Klassen innerhalb des vorhandenen Beispielraumes bestimmt. Das Merkmal mit der niedrigsten Entropie liefert die größte Aussagekraft bei der Entscheidung über die Klassenzugehörigkeit.
Um die Entropie eines Merkmals zu bestimmen, muss zuerst der Beispielraum anhand der auftretenden Ausprägungen des betrachteten Merkmals in Teilmengen zerlegt werden. Für jede dieser Teilmengen, d.h. für jede Ausprägung *Auspr*, bestimmt man die Entropie wie folgt:

$$H(Auspr_i) = -\sum_{j=1}^{m} p(Klasse_j \mid Auspr_i) \cdot ld(p(Klasse_j \mid Auspr_i))$$

mit	m ...	Anzahl der bei der Ausprägung i auftretenden Klassen
	$p(Klasse_j \mid Auspr_i)$ ...	Wahrscheinlichkeit der Zugehörigkeit des Objekts zur Klasse j, wenn das betrachtete Merkmal die Ausprägung i hat

Die Klassen, die bei der Ausprägung i nicht auftreten (d.h. diejenigen, bei denen $p(Klasse_j \mid Auspr_i) = 0$ ist), werden hierbei nicht berücksichtigt (weggelassen), denn

$$\lim_{p \to 0}(p \cdot ld(p)) = 0$$

Um die (mittlere) Entropie des Merkmals ingesamt zu ermitteln, muss man die Entropien seiner Ausprägungen mit den Wahrscheinlichkeiten ihres Auftretens wichten:

$$H(Merkmal) = \sum_{i=1}^{n} p(Auspr_i) \cdot H(Auspr_i)$$

mit	n ...	Anzahl der Ausprägungen des Merkmals
	$p(Auspr_i)$ ...	Wahrscheinlichkeit des Auftretens der Ausprägung i
	$H(Auspr_i)$ ...	Entropie der Ausprägung i nach o.g. Formel

In unserem Beispiel führen diese Berechnungen zu folgenden Ergebnissen:

Merkmal Kaufpreis:

H(Kaufpreis=*gering*)
$= -p(empfehlenswert \mid gering) \cdot ld(p(empfehlenswert \mid gering))$
$\quad -p(indifferent \mid gering) \cdot ld(p(indifferent \mid gering))$
$\quad -p(abratenswert \mid gering) \cdot ld(p(abratenswert \mid gering))$
$= -\frac{3}{5} \cdot ld(\frac{3}{5}) - \frac{1}{5} \cdot ld(\frac{1}{5}) - \frac{1}{5} \cdot ld(\frac{1}{5})$
$= -\frac{3}{5} \cdot (-0.737) - \frac{1}{5} \cdot (-2.232) - \frac{1}{5} \cdot (-2.232)$
$= 1.371$

H(Kaufpreis=*mittel*)
$= -\frac{2}{5} \cdot ld(\frac{2}{5}) - \frac{1}{5} \cdot ld(\frac{1}{5}) - \frac{2}{5} \cdot ld(\frac{2}{5})$
$= 1.522$

H(Kaufpreis=*hoch*)
$= -\frac{2}{4} \cdot ld(\frac{2}{4}) - \frac{2}{4} \cdot ld(\frac{2}{4})$
$= 1$

H(Kaufpreis)
$= \quad p(gering) \cdot H(\text{Kaufpreis= } gering)$
$\quad +p(mittel) \cdot H(\text{Kaufpreis= } mittel)$
$\quad +p(hoch) \cdot H(\text{Kaufpreis= } hoch)$
$= \frac{5}{14} \cdot 1.371 + \frac{5}{14} \cdot 1.522 + \frac{4}{14} \cdot 1$
$= 1.319$

Für die Merkmale „Aussehen“ und „Betriebskosten“ gestaltet sich die Rechnung ähnlich und führt zu folgenden Ergebnissen:

Merkmal Aussehen: H(Aussehen) = 1.319
Merkmal Betriebskosten: H(Betriebskosten) = 1.026

Die kleinste Entropie liefert demnach das Merkmal „Betriebskosten“, so dass dieses Merkmal an der Wurzel des Regelbaumes steht:

Betriebskosten?
- *gering:* $Unterbaum_1$
- *normal:* $Unterbaum_2$
- *hoch:* $Unterbaum_3$

Nunmehr wird die Beispielmenge entsprechend der 3 Ausprägungen des Merkmals „Betriebskosten“ in 3 Teilmengen geteilt, die separat behandelt werden und nur noch die Merkmale „Kaufpreis“ und „Aussehen“ aufweisen. Hierbei entstehen $Unterbaum_1$, $Unterbaum_2$ und $Unterbaum_3$.

Das Verfahren für einen solchen Unterbaum ist beendet, wenn alle Beispiele einer entstehenden Teilmenge derselben Klasse angehören, wie das z.B. beim $Unterbaum_3$ der Fall ist:

$Unterbaum_3$:

Kaufpreis	Aussehen	Klasse
mittel	*normal*	*abratenswert*
mittel	*gut*	*abratenswert*
hoch	*normal*	*abratenswert*

Hier artet der Unterbaum in einen Blattknoten aus, welcher sofort die Klassenzugehörigkeit verrät:

Betriebskosten?
- *gering:* $Unterbaum_1$
- *normal:* $Unterbaum_2$
- *hoch* → *abratenswert*

Komplizierter gestaltet sich die Ermittlung des Unterbaumes $Unterbaum_2$:

$Unterbaum_2$:

Kaufpreis	Aussehen	Klasse
gering	*hässlich*	*abratenswert*
hoch	*normal*	*indifferent*
mittel	*gut*	*empfehlenswert*
hoch	*hässlich*	*abratenswert*
hoch	*gut*	*indifferent*
gering	*gut*	*empfehlenswert*

Merkmal Kaufpreis:

H(Kaufpreis=*gering*)
$= -p(empfehlenswert \mid gering) \cdot ld(p(empfehlenswert \mid gering))$
$\quad -p(abratenswert \mid gering) \cdot ld(p(abratenswert \mid gering))$ [8]
$= -\frac{1}{2} \cdot ld(\frac{1}{2}) - \frac{1}{2} \cdot ld(\frac{1}{2})$
$= 1$

H(Kaufpreis=*mittel*)
$= -1 \cdot ld(1)$
$= 0$

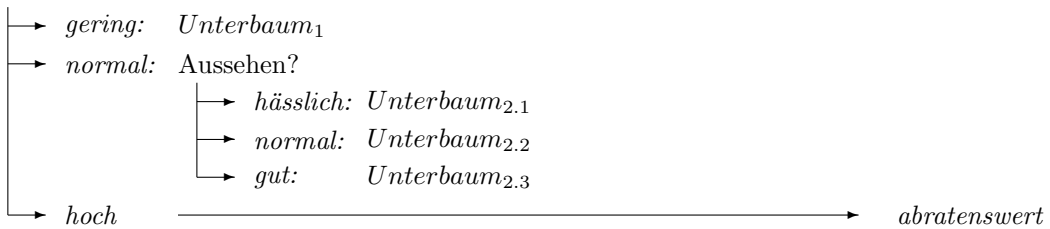
H(Kaufpreis=*hoch*)
$= -\frac{2}{3} \cdot ld(\frac{2}{3}) - \frac{1}{3} \cdot ld(\frac{1}{3})$
$= 0.918$

H(Kaufpreis)
$= \quad p(gering) \cdot H(\text{Kaufpreis= } gering)$
$\quad +p(mittel) \cdot H(\text{Kaufpreis= } mittel)$
$\quad +p(hoch) \cdot H(\text{Kaufpreis= } hoch)$
$= \frac{2}{6} \cdot 1 + \frac{1}{6} \cdot 0 + \frac{3}{6} \cdot 0.918$
$= 0.792$

Analog errechnet ergibt sich für das Merkmal „Aussehen“ das folgende:

Merkmal Aussehen: H(Aussehen) = 0.459

Die niedrigste Entropie liefert also das Merkmal „Aussehen“, so dass sich der Regelbaum nun wie folgt darstellt:



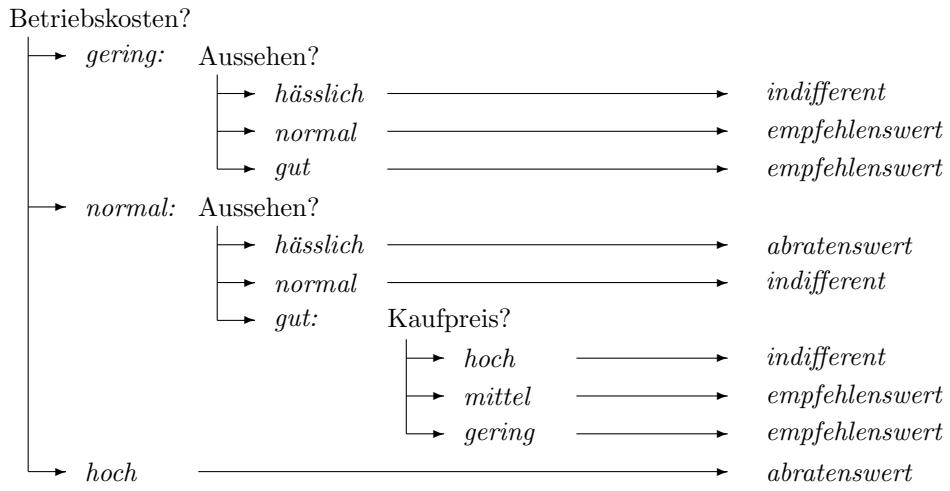


Abbildung 8: Regelbaum nach der „ID3“ – Regelgenerierung

Die weitere Behandlung der Unterbäume nach diesem Verfahren ergibt letztlich das in Abbildung 8 gezeigte Endergebnis.

In einer Produktionsregel – Notation sieht das wie folgt aus:

$$
\begin{array}{lll}
(\text{Klasse} = \textit{indifferent}) & \longleftarrow & (\text{Betriebskosten} = \textit{gering}) \quad \wedge \\
 & & (\text{Aussehen} = \textit{hässlich}) \\
(\text{Klasse} = \textit{empfehlenswert}) & \longleftarrow & (\text{Betriebskosten} = \textit{gering}) \quad \wedge \\
 & & (\text{Aussehen} = \textit{normal}) \\
(\text{Klasse} = \textit{empfehlenswert}) & \longleftarrow & (\text{Betriebskosten} = \textit{gering}) \quad \wedge \\
 & & (\text{Aussehen} = \textit{gut}) \\
\ldots & &
\end{array}
$$

Die nach dem ID3–Algorithmus entstandene Wissensbasis ist bezüglich der Beispielmenge logisch äquivalent derjenigen des zuvor gezeigten Verfahrens der „left–to–right“ – Regelgenerierung, d.h. sie ordnet alle gegebenen Beispiele der gleichen Klasse wie beim vorherigen Verfahren zu. Aufgrund der Optimierung kommt man hier jedoch schon mit 9 Regeln aus.
Durch eine Erweiterung kann dieses Verfahren auch auf der Grundlage einer Beispielmenge mit unscharfer Klassenzuordnung basieren. Dadurch kann es auch dann Anwendung finden, wenn die Klassenzugehörigkeit der Beispiele nicht genau bekannt ist. Hier muss dann die Wahrscheinlichkeit der Zugehörigkeit zu einer Klasse quantifiziert werden, was durch die relative Häufigkeit in der Beispielmenge abgeschätzt werden kann.

Induktionsverfahrenüber prädikatenlogischen Ausdrücken

Diese haben i.allg. zum Ziel, aus einer Menge variablenfreier Ausdrücke $\{A_1, \ldots, A_n\}$ einen allgemeinen variablenbehafteten Ausdruck B zu induzieren, so dass für alle A_i gilt:

$$ag(B \rightarrow A_i)$$

B ist dann eine logische Begründung für alle beobachteten Situationen A_i. Meist gibt es sehr viele Begündungen für die beobachteten Situationen A_i, die jedoch unterschiedlich interessant sein können. Hier ist i.allg. eine „beste“ Begründung gefragt:

B heißt *bester induktiver Schluss* (oder einfachste Begründung) für
$M = \{A_1, \ldots, A_n\}$, falls gilt:

[8] Indifferente PKWs mit geringem Kaufpreis gibt es in der Beispielmenge nicht.

1. B ist eine logische Begründung für M und
2. Jede andere logische Begründung C für M begründet auch B.

Dieser Sachverhalt kann bildlich wie in Abbildung 9 gezeigt werden, wobei die Pfeile tatsächlich als Implikationssymbole allgemeingültiger Aussagen interpretiert werden können.

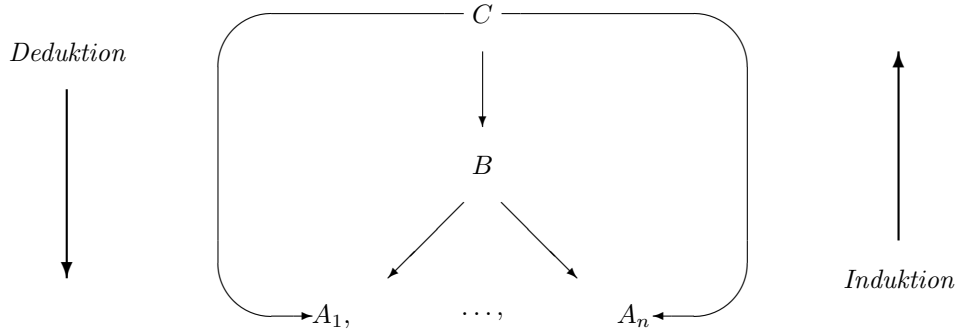


Abbildung 9: B ist der „beste induktive Schluss“ für $\{A_1, \ldots, A_n\}$

Ein (zu) einfacher Weg des Findens eines solchen „besten induktiven Schlusses“B wäre

$$B = \bigwedge_{i=1}^{n} A_i$$

Dies scheint jedoch kaum praktischen Anforderungen zu genügen und wird um so sinnloser, je größer n ist.
Solche „besten induktiven Schlüsse“ B lassen sich u.a. auch als die speziellsten gemeinsamen Muster aller Ausdrücke $\{A_1, \ldots, A_n\}$ auffassen.
Für dieses Verfahren müssen alle A_i Kombinationen (ganau genommen Konjunktionen) von Atomformeln sein, deren Variablen allquantifiziert sind. Der zu findende induktive Schluss B ist ein Ausdruck der Gestalt

$$\forall X_1 \ldots \forall X_m \Phi$$

wobei Φ keine Quantoren enthält.
Der „Trick“ dieses Verfahrens besteht darin, diejenigen Atomformeln, die das gleiche Prädikatensymbol aufweisen, durch die Angabe eines „gemeinsamen Musters“ zusammenzufassen. Dabei entstehen variablenbehaftete Atomformeln mit vorangestellten Allquantoren, die konjunktiv verknüpft werden. Durch Ausklammern der Allquantoren[9] entsteht dann Φ.
Das Problem reduziert sich somit auf ein sogenanntes *Antiunifikationsproblem*:

1. Für quantorfreie Ausdrücke A und B gilt $B \geq A$, genau dann, wenn $ag(B \rightarrow A)$.[10]
2. B heißt **Antiunifikation** einer Menge von Aussagen $\{A_1, \ldots, A_n\}$, falls $\forall i : B \geq A_i$.
3. Die **speziellste Antiunifikation** Φ von $\{A_i, \ldots, A_n\}$, ist diejenige Antiunifikation, für die gilt: Alle anderen Antiunifikationen C sind „allgemeiner“ als Φ, d.h. $C \geq \Phi$.

   Die Termersetzung ϑ, die Φ aus allen A_i erzeugt ($\vartheta(A_1) = \vartheta(A_2) = \ldots = \vartheta(A_n) = \Phi$), heisst **speziellster gemeinsamer Antiunifikator** (*most specific antiunifier*) von $\{A_1, \ldots, A_n\}$:

   $$\vartheta = m.s.a.(\{A_1, \ldots, A_n\})$$

Das Ziel der Antiunifikation ist demnach das Finden eines *speziellsten gemeinsamen Antiunifikators* einer Menge von Atomformeln $\{A_1, \ldots, A_n\}$.
Das Problem des Findens des speziellsten Antiunifikators (auch most specific antiunifier $m.s.a.$ genannt) zweier Atomformeln wird durch das Verfahren, welches den $m.s.a.$ zweier Terme findet, mitgelöst, so dass die Angabe des letzteren hier genügt.

[9] Man denke daran, dass die Atomformelnn vorher variablenfremd zu machen sind!
[10] Sind A und B Konjunktionen von Atomformeln, dann bedeutet dies nichts anderes als „A ist allgemeiner als“ B. D.h. für jede Atomformel B_i in B gibt es in A eine Atomformel A_j und eine Substitution ϑ derart, dass $\vartheta(A_j) \equiv B_i$ ist.

Ein Verfahren zur Ermittlung der speziellsten Antiunifikation 2er Terme

Man versteht Antiunfikation am besten als (potentiell nicht-deterministischen) Reduktionsprozeß. Man gibt ein Regelsystem an. Dann ist dreierlei zu tun:

1. Man muß den Formalismus des Reduktionssystems mit der zu l
ösenden Aufgabe in Beziehung setzen.

2. Man muß Termination nachweisen.

3. Man muß die Korrektheit der Ergebnisse – im jeweiligen Anwendungsszenario – nachweisen.

Es seien zwei Terme t_1 und t_2 gegeben. In beiden Termen wird sukzessive die Antiunifikation ausgef
ührt, bis sie beide syntaktisch gleich sind. Die Stellen, an denen noch etwas zu tun ist, werden in einer Markierungsmenge M notiert, und zwar durch Positionen im Term. Die sukzessive aufgebauten Substitutionen legt man in Σ ab, wobei die linke Seite jeder Substitutionsregel nur einmal auftaucht, denn es wird ja nur in ein und demselben Term – der Antiunifikation – ersetzt. Deshalb enth
ält Σ Tripel.
Eine Ausgangssituation f
ür zwei Terme t_1 und t_2 wird formalisiert als

$$[\, t_1, t_2, \{\varepsilon\}, \emptyset \,]$$

und eine Zielsituation der Form

$$[\, t, t, \emptyset, \Sigma \,]$$

bedeutet, dass t als die speziellste Antiunifikation konstruiert wurde und Σ die beiden Substitutionen enth
ält, die erlauben, daraus t_1 bzw. t_2 zu erzeugen.

Das Regelsystem

Vor jeder Regel werden die Bedingungen der Regelanwendung angegeben.

R1 $p \in M,\ t_1.p \neq t_2.p,\ [x, t_1|_p, t_2|_p] \in \Sigma$

$$[\, t_1, t_2, M, \Sigma \,] \quad \Longrightarrow \quad [\, t_1[p \hookleftarrow x], t_2[p \hookleftarrow x], M \setminus \{p\}, \Sigma \,]$$

R2 $p \in M,\ t_1.p \neq t_2.p,\ \nexists\, [x, t_1|_p, t_2|_p] \in \Sigma,\ x \notin \pi_1(\Sigma)$

$$[\, t_1, t_2, M, \Sigma \,] \quad \Longrightarrow \quad [\, t_1[p \hookleftarrow x], t_2[p \hookleftarrow x], M \setminus \{p\}, \Sigma \cup \{[x, t_1|_p, t_2|_p]\} \,]$$

R3 $p \in M,\ t_1.p = t_2.p,\ t_1|_p \neq t_2|_p,\ \alpha(t_1.p) = n$

$$[\, t_1, t_2, M, \Sigma \,] \quad \Longrightarrow \quad [\, t_1, t_2, M \setminus \{p\} \cup \{p1, \ldots, pn\}, \Sigma \,]$$

R4 $p \in M,\ t_1|_p = t_2|_p$

$$[\, t_1, t_2, M, \Sigma \,] \quad \Longrightarrow \quad [\, t_1, t_2, M \setminus \{p\}, \Sigma \,]$$

Ein paar umgangssprachliche Erkl
ärungen sollen zun
ächst dem Formalismus den Schrecken nehmen, bevor Details der Notation eingef
ührt werden.

zu **R1** An einer noch zu bearbeitenden Position in M stehen verschiedene Terme, die bereits schon einmal durch eine Variable x eliminiert worden sind. Dieselbe Variable kann hier benutzt werden. Diese Position ist dann erledigt; mehr ist nicht zu tun.

zu **R2** An einer noch zu bearbeitenden Position in M stehen verschiedene Terme, die in dieser Form noch nicht ersetzt worden sind. Es wird eine neue Variable x gew
ählt und f
ür diese Terme eingef
ügt. Entsprechend wird die Substitutionsliste erg
änzt und die Position gestrichen.

zu **R3** An einer noch zu bearbeitenden Position in M stehen Terme mit demselben Funktionssymbol. An dieser Position ist nichts zu
ändern, aber alle Teiltermpositionen werden in die Liste der zu bearbeiten Aufgaben aufgenommen.

zu **R4** An einer noch zu bearbeitenden Position p stehen jeweils dieselben Terme. Diese Position ist erledigt; mehr ist nicht zu tun.

Notationen
Positionen in Termen sind W
örter
über der Menge der nat
ürlichen Zahlen $I\!N$. Das leere Wort heißt ε.[11]

Erkl. Wenn t irgendein Term ist, der an der Position p den Teilterm $f(t_1, \ldots, t_n)$ hat, dann hat er an den Positionen $p1, \ldots pn$ die entsprechenden Teilterme $t_1, \ldots, t_n$.

Wenn p eine Position im Term t ist, dann bezeichnet $t.p$ den Operator an dieser Position im Term t und $t|_p$ den in t an der Position beginnenden Teilterm. Trivialerweise ist f
ür jeden Term t und die Wurzelposition ε stets $t = t|_\varepsilon$.

Bspl. Betrachten wir den Term $t = f(\sin(x+1), 1-\cos(\varphi))$. Das Funktionssymbol an der Position ε ist f und das Symbol an der Position 1 ist sin. Insgesamt haben wir:
$t.\varepsilon = f \quad t.1 = \sin \quad t.2 = - \quad t.11 = + \quad t.21 = 1 \quad t.22 = \cos \quad t.111 = x \quad t.112 = 1 \quad t.221 = \varphi$
Der Teilterm an der Position 1 ist $\sin(x+1)$ und der Teilterm an der Position 11 ist $x+1$. F
ür die Teilterme "auf der rechten Seite" des 2-stelligen Operators f haben wird insgesamt also:
$t|_2 = 1-\cos(\varphi) \quad t|_{21} = 1 \quad t|_{22} = \cos(\varphi) \quad t|_{221} = \varphi$

Wenn in einem Term t an einer Position p ein Term t' eingesetzt wird, so notieren wir das durch $t[p \hookleftarrow t']$.

Bspl. Betrachten wir den Term $t = f(\sin(x+1), 1-\cos(\varphi))$. Eine Ersetzung von z an der Position 1 wird notiert als $t[1 \hookleftarrow z]$ und liefert $t[1 \hookleftarrow z] = f(z, 1-\cos(\varphi))$.

Eigentlich geh
ört zur Beschreibung der Arit
ät eines Funktionssymbols nicht nur die Stelligkeit, sondern auch die Sortigkeit. Da die Arit
ät nur dort ins Spiel kommt, wo Sortenunterschiede nicht mehr auftreten k
önnen, werden diese ignoriert. α gibt zu einem Funktionssymbol dessen Stelligkeit an. F
ür Konstanten und Variablen ist dieser Wert 0.
Mit π_1 wird die Projektion auf das erste Argument bezeichnet. So ist insbes. $\pi_1(M)$ die Menge aller Variablen, die in Tripeln aus M vorkommen.

Termination

Es ist fast trivial, die Termination dieses Reduktionssystems zu beweisen. Man macht das am besten nur über M.
3 Regeln verkleinern M jeweils echt. Die einzige Regel, die M vergr
ößert, ist **R3**. Diese Regel f

[11] Achtung: Da diese Auffassung kein irgendwie geartetes n-
äres Zahlsystem voraussetzt, gibt es auch keine Konfusionen.

ührt allerdings zu einer gegebenen Position nur l
ängere Positionen ein. Wegen der Endlichkeit der Ausgangsterme kann **R3** nur endlich oft angewendet werden. Das gen
ügt.

Beispiele

Ein erstes Beispiel
Gegeben seien die Terme

$$t_1 = p(f(a,b), f(a,b), X, a)$$
$$t_2 = p(f(a,X), g(a,b), b, Y)$$

Die Ausgangssituation ist demnach

$$[p(f(a,b), f(a,b), X, a), p(f(a,X), g(a,b), b, Y), \{\varepsilon\}, \emptyset]$$

Die Regelanwendungen ergeben

Regel	t_1	t_2	M	Σ
	$p(f(a,b), f(a,b), X, a)$	$p(f(a,X), g(a,b), b, Y)$	$\{\varepsilon\}$	$\emptyset$
R3	$p(f(a,b), f(a,b), X, a)$	$p(f(a,X), g(a,b), b, Y)$	$\{1,2,3,4\}$	$\emptyset$
R3	$p(f(a,b), f(a,b), X, a)$	$p(f(a,X), g(a,b), b, Y)$	$\{11,12,2,3,4\}$	$\emptyset$
R4	$p(f(a,b), f(a,b), X, a)$	$p(f(a,X), g(a,b), b, Y)$	$\{12,2,3,4\}$	$\emptyset$
R2	$p(f(a,X_0), f(a,b), X, a)$	$p(f(a,X_0), g(a,b), b, Y)$	$\{2,3,4\}$	$\{[X_0,b,X]\}$
R2	$p(f(a,X_0), X_1, X, a)$	$p(f(a,X_0), X_1, b, Y)$	$\{3,4\}$	$\{[X_0,b,X],$ $[X_1,f(a,b),g(a,b)]\}$
R2	$p(f(a,X_0), X_1, X_2, a)$	$p(f(a,X_0), X_1, X_2, Y)$	$\{4\}$	$\{[X_0,b,X],$ $[X_1,f(a,b),g(a,b)],$ $[X_2,X,b]\}$
R2	$p(f(a,X_0), X_1, X_2, X_3)$	$p(f(a,X_0), X_1, X_2, X_3)$	$\emptyset$	$\{[X_0,b,X],$ $[X_1,f(a,b),g(a,b)],$ $[X_2,X,b],$ $[X_3,a,Y]\}$

Noch ein (scheinbar ganz hnliches) Beispiel
Gegeben seien die Terme

$$t_1 = p(f(a,b), f(a,b), X, b)$$
$$t_2 = p(f(a,X), g(a,b), b, X)$$

Die Ausgangssituation ist demnach

$$[p(f(a,b), f(a,b), X, b), p(f(a,X), g(a,b), b, X), \{\varepsilon\}, \emptyset]$$

Die Regelanwendungen ergeben

Regel	t_1	t_2	M	Σ
	$p(f(a,b), f(a,b), X, b)$	$p(f(a,X), g(a,b), b, X)$	$\{\varepsilon\}$	$\emptyset$
R3	$p(f(a,b), f(a,b), X, b)$	$p(f(a,X), g(a,b), b, X)$	$\{1,2,3,4\}$	$\emptyset$
R3	$p(f(a,b), f(a,b), X, b)$	$p(f(a,X), g(a,b), b, X)$	$\{11,12,2,3,4\}$	$\emptyset$
R4	$p(f(a,b), f(a,b), X, b)$	$p(f(a,X), g(a,b), b, X)$	$\{12,2,3,4\}$	$\emptyset$
R2	$p(f(a,X_0), f(a,b), X, b)$	$p(f(a,X_0), g(a,b), b, X)$	$\{2,3,4\}$	$\{[X_0,b,X]\}$
R2	$p(f(a,X_0), X_1, X, b)$	$p(f(a,X_0), X_1, b, X)$	$\{3,4\}$	$\{[X_0,b,X],$ $[X_1,f(a,b),g(a,b)]\}$
R2	$p(f(a,X_0), X_1, X_2, b)$	$p(f(a,X_0), X_1, X_2, X)$	$\{4\}$	$\{[X_0,b,X],$ $[X_1,f(a,b),g(a,b)],$ $[X_2,X,b]\}$
R1	$p(f(a,X_0), X_1, X_2, X_0)$	$p(f(a,X_0), X_1, X_2, X_0)$	$\emptyset$	$\{[X_0,b,X],$ $[X_1,f(a,b),g(a,b)],$ $[X_2,X,b]\}$

Das Problem, die Antiunifikation einer *Menge* von (i.allg. mehr als zwei) Atomformeln zu definieren, ist nun recht einfach zu lösen: Statt der Tripel in der Termersetzung Σ entstehen für eine n-elementige Menge $\{t_1, t_2, \ldots, t_n\}$ dann $n+1$-Tupel der Art

$$[Variable, Ersetzung\_in\_t_1, Ersetzung\_in\_t_2, \ldots, Ersetzung\_in\_t_n]$$

Wie die Vorbedingungen für die Anwendung der Regeln R1-R4 in diesem Falle zu formulieren sind, verbleibt dem Leser als Übungsaufgabe![12]

2.2.3 Abduktion

Die dritte Methode des Schließens, die **Abduktion**, ist eine Schlussweise, bei der aus beobachteten Erscheinungen auf mögliche Ursachen geschlossen wird. Ein solcher Schluss erfolgt demnach umgekehrt zur natürlichen Kausalität.
Im einfachsten Fall stellt sich ein abduktiver Schluss wie folgt dar:

$$\frac{A \rightarrow B \quad B}{A}$$

Dieser Schluss ist natürlich außerordentlich fragwürdig, denn es gilt mitnichten

$$\{A \rightarrow B, B\} \models A$$

Bei wissensverarbeitenden Diagnosesystemen ist eine solche Schlussweise jedoch an der Tagesordnung, denn dort wird von einer vorliegenden Symptomatik auf den vorliegenden Fehler geschlossen, obwohl die natürliche Kausalität genau entgegengesetzt orientiert ist. Dieser Sachverhalt ist in der Abbildung 10 illustriert.

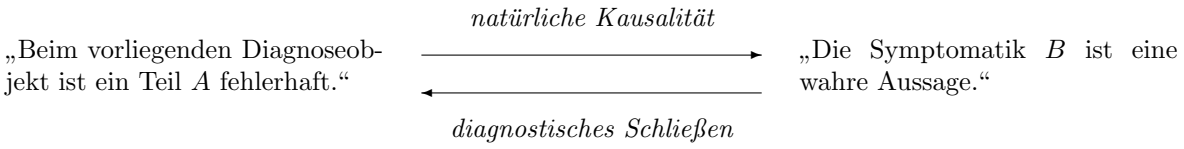


Abbildung 10: Das diagnostische Schließen entgegen der natürlichen Kausalität

Eine Symptomatik ist hierbei eine zusammengesetzte Aussage des Aussagenkalküls, deren elementare Bestandteile i.allg. Symptome genannt werden.
Um die praktische Anwendung einer solch fragwürdigen Schlussweise zu legitimieren, sollten die Regeln möglichst so beschaffen sein, dass sie ein **vollständiges** und **eindeutiges** Regelsystem bilden. Für die Diagnoseproblematik würde das bedeuten, dass alleüberhaupt denkbaren Diagnosen (vollständig) erfasst werden und für eine gegebene Symptomatik (Konklusion) nur eine einzige Diagnose (eindeutig) als Prämisse in Frage kommt, d.h.

- alle denkbaren Prämissen des Diskursbereiches müssen in (mindestens) einer Regel vorkommen und
- die Konklusionen sämtlicher in einer Situation zu prüfenden Regeln müssen einander ausschließen, d.h. sie müssen paarweise disjunkt sein:

$$\forall i \forall j((i \neq j) \rightarrow ((Konklusion_i \wedge Konklusion_j) \equiv false))$$

Die Bezeichnungen „Prämissen“ und „Konklusionen“ beziehen sich hier auf die natürliche Kausalität. Je eklatanter diese Anforderungen verletzt werden, desto fragwürdiger wird dieser Schluss. Liegen beide Eigenschaften 100%ig vor, so kann die Implikation durch die Äquivalenz ersetzt werden. Da eine Äquivalenz nichts anderes als eine Implikation in beiden Richtungen ist, handelt es sich dann folglich um einen (formallogisch sauberen) deduktiven Schluss.

[12] Es ist zu beachten, dass es zur Anwendung von R1 und R2 genügt, dass *ein Paar* von Termen t_i und t_j existiert mit $t_i.p \neq t_j.p$. Für die Anwendung der Regeln R3 und R4 hingegen müssen natürlich *alle* Funktionssymbole an der Stelle p bzw. alle an der Stelle p beginnenden Teilterme identisch sein.

2.3 Erklärungs – Komponente

Wozu dient die Erklärungs – Komponente?

1. Die primäre Aufgabe dieser Komponente ist die **Erklärung der** zur Problemlösung herangezogenen **Inhalte der Wissensbasis und der darüber getroffenen Inferenzschritte**.
2. Ein nicht zu verachtender Nebeneffekt ist die Erweiterung des Horizonts des Fachwissens beim Nutzer; er lernt etwasüber die Inhalte der Wissensbasis und ihre Anwendung auf den vorliegenden Fall. Auf diese Weise erfähren die Nutzer eine **Instruktionüber die Inhalte der Wissensbasis**.
3. Die Erklärungs – Komponente kann gleichfalls zum **Test und** zur **Fehlersuche beim Entwurf des Systems** genutzt werden.

Was soll dem Nutzer erklärt werden?

Vom Nutzer wird die Erklärungskomponente aktiviert, indem er sich

- das Zustandekommen einer Anfrage des Systems erklären lässt, d.h. die Gegenfrage

  „Warum fragst Du mich das?“

  an das System stellt (**WHY–Erklärung**),
- die vom System gefundenen Zwischen– und Endergebnisse begründen lässt, d.h. die Frage

  „Wie kommst Du zu diesem Ergebnis?“

  stellt (**HOW–Erklärung**) oder
- eine Darstellung des dem System bisher bekannten fallspezifischen Faktenwissens auflisten lässt (**WHAT–Erklärung**).

Wie kann die Erklärung der Inferenzschritte realisiert werden?

Die Erklärung ist i.allg. eine schrittweise Angabe der getroffenen Inferenzschritte und kann sowohl

- von der Situation zu Beginn der Sitzung ausgehen und bis zu dem aktuell ins Kalkül gezogenen Inferenzschritt gehen (nennen wir es **Vorwärts–Erklärung**) als auch
- in umgekehrter Richtung erfolgen (nennen wir es **Rückwärts–Erklärung**).

Welche Anforderungen sind an eine Erklärungskomponente zu stellen?

1. Erklärungen müssen **informativ** sein.
2. Erklärungen sollten sowohl für den Entwerfer als auch für jede Klasse von Nutzern **instruktiv** sein.
3. Die Informationsmenge sollte sich **auf das Wesentliche beschränken**.

2.4 Dialog – Komponente

Die Dialog – Komponente bildet die Schnittstelle zum Nutzer und hat die **Aufgaben**,

- das fallspezifische Faktenwissen zu akquirieren,
- dem Nutzer die Zwischen– und Endergebnisse mitzuteilen,
- auf Anforderung die vom Nutzer gewünschte Erklärungsinformation über den Inferenzprozess in geeigneter Form zu präsentieren und
- dem Nutzer die Bedienung des Systems transparent zu machen.

Welchen Anforderungen sollte eine Dialogkomponente gerecht werden?

1. **Konsistenz**

   Gleichartige Interaktionen sollten auch gleichartige Vorgänge auslösen.
   Ist die Konsistenz innerhalb eines Systems gegeben, so spricht man von **innerer Konsistenz**, ist sieüber mehrere Systeme eines Paketes hinweg erfüllt, spricht man von **äußerer Konsistenz**.

2. **Adaptierbarkeit und Adaptivität**

   Die Nutzer–Schnittstelle sollte an den Erfahrungsstand und die Kompetenz des Nutzers angepasst sein. Dies bezieht sich u.a. auf

   - die Informationskodierung und die Benutzung von Fachausdrücken,
   - die Art und Detailliertheit von Erklärungen,
   - die Möglichkeiten zur Definition und Nutzung von Aktionssequenzen und
   - die Interaktionstechnik (computergeführter oder nutzergeführter Dialog).

3. **Hilfekomponenten**

   Sinnvoller ist ein **Hilfesystem**, welches dem Nutzer auf Anforderung und/oder bei Fehlbedienungen erklärende Informationenüber die Bedienung liefert.

4. **Führung eines Protokollsüber die Dialogvergangenheit**

   Um dem Nutzer jederzeit erklären zu können, warum er in eine bestimmte Dialogsituation geraten ist, ist es i.allg. erforderlich, auf die Dialogvergangenheit (die Vorgeschichte) zurückzugreifen. Zu diesem Zweck sollte die Dialogvergangenheit protokolliert werden. Außerdem kann ein solches Protokoll auch dazu verwendet werden, dem Nutzer jederzeit die Möglichkeit zu geben, in die jeweils vorhergehende Dialogsituation zurückzukehren bzw. das Sitzungsgeschehen nachträglich auszuwerten.

2.5 Wissenserwerbs – Komponente

Die Aufgabe dieser Komponente ist (lediglich) die **Akquisition bereichsspezifischen Expertenwissens**, d.h.

- der Unterstützung des Wissenserwerbs im Dialog mit dem Ersteller der Wissensbasis gleichfalls
- die Akquisition bereichsspezifischen Expertenwissens anhand von Nutzerinformationen (z.B. durch Auswertung von Fallbeispielen aus einer Reihe von Sitzungen).

Der Wissenserwerb mit einer Wissenserwerbskomponente setzt voraus, dass das Wissen bereits analysiert und eine (oder mehrere) formale Wissensdarstellung(en) gefunden wurde(n). Die Wissenserwerbskomponente setzt darauf auf und unterstützt die (manuelle und automatische) Erstellung von Wissensbasen in dieser (diesen) Darstellungsform(en).
Zusammenfassend und systemstisierend seien die durch eine Wissenserwerbskomponente zu unterstützenden Funktionen in der Abbildung 11 dargestellt.

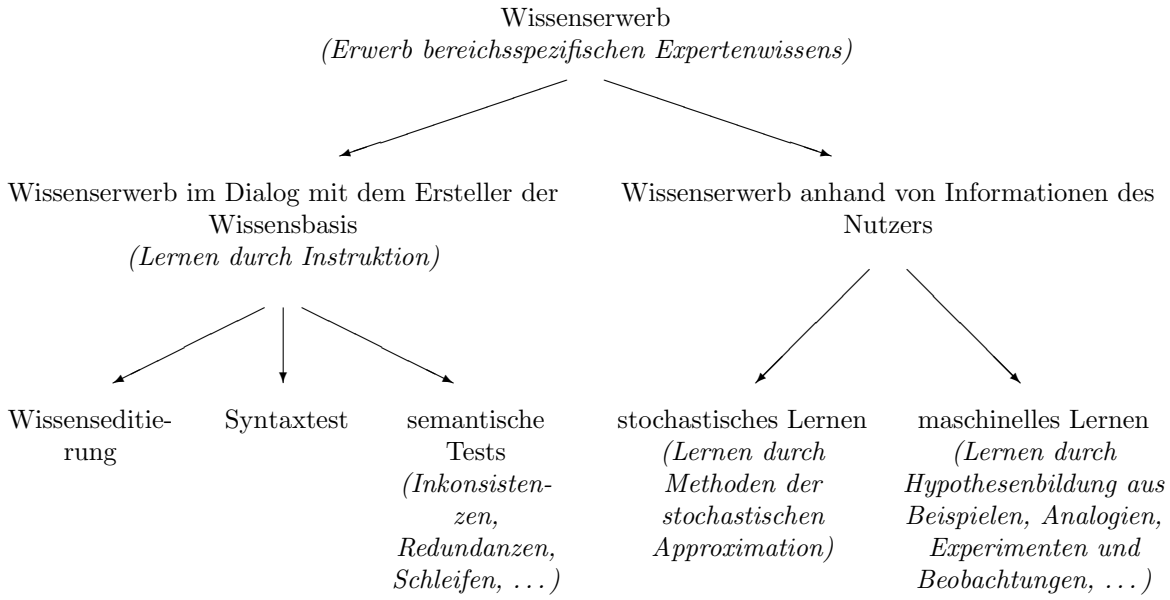


Abbildung 11: Kategorien des Wissenserwerbs