INF FAKULTÄT FÜR INFORMATIK

Intelligente Systeme

Wissensbasierte Systeme

Prof. Dr. R. Kruse C. Braune C. Moewes

{kruse,braune,cmoewes}@iws.cs.uni-magdeburg.de

Institut für Wissens- und Sprachverarbeitung

Fakultät für Informatik

Otto-von-Guericke Universität Magdeburg

Übersicht

Wissensrepräsentation und Schlussfolgern

Aus der *kognitiven Psychologie* ist bekannt, dass menschliches Problemlösen und Lernen folgende Dinge involviert:

Repräsentation und Ausnutzung von *mehreren Arten von Wissen*

Kombination von *verschiedenen Methoden zum Schlussfolgern*: regelbasiertes Schließen, modellbasiertes Schließen, bedingungsbasiertes (constraint-based) Schließen

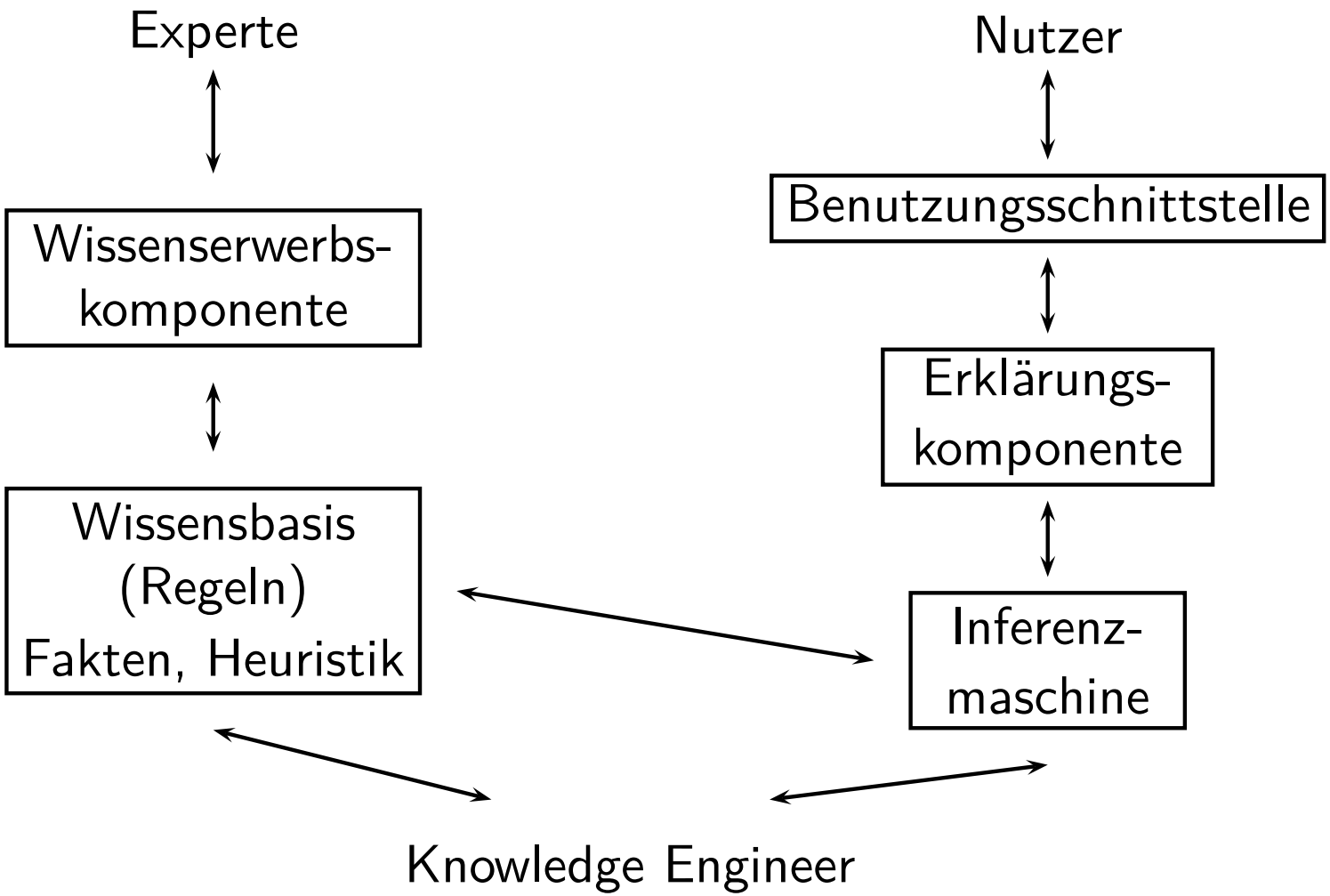
Fragen:

Wie können die reale Welt und das Schlussfolgern mit dem Computer modelliert werden?

Wie kann man Kenntnisse der Technologien gewinnen, die zum Entwurf von wissensbasierten Systemen gebraucht werden?

Wissensbasiertes System = System, dass mittels Wissen schlussfolgern kann

Wissensbasierte Systeme



Beispiel: „Darlehensvergabe“

OK	Darlehen sollte bewilligt werden.
COLLAT	Sicherheit für Darlehen ist genügend.
PYMT	Antragsteller kann Darlehenszahlungen tilgen.
REP	Antragsteller hat gute finanzielle Reputation.
APP	Beurteilung der Sicherheit ist ausreichend größer als Größe des Darlehens.
RATING	Antragsteller hat gute Bonität.
INC	Einkommen des Antragstellers ist größer als seine Ausgaben.
BAL	Antragsteller hat ausgezeichnete Bilanz.

Beispiel: „Darlehensvergabe“

Regeln zur Entscheidungsfindung:

$\text{COLLAT} \wedge \text{PYMT} \wedge \text{REP} \Rightarrow \text{OK}$

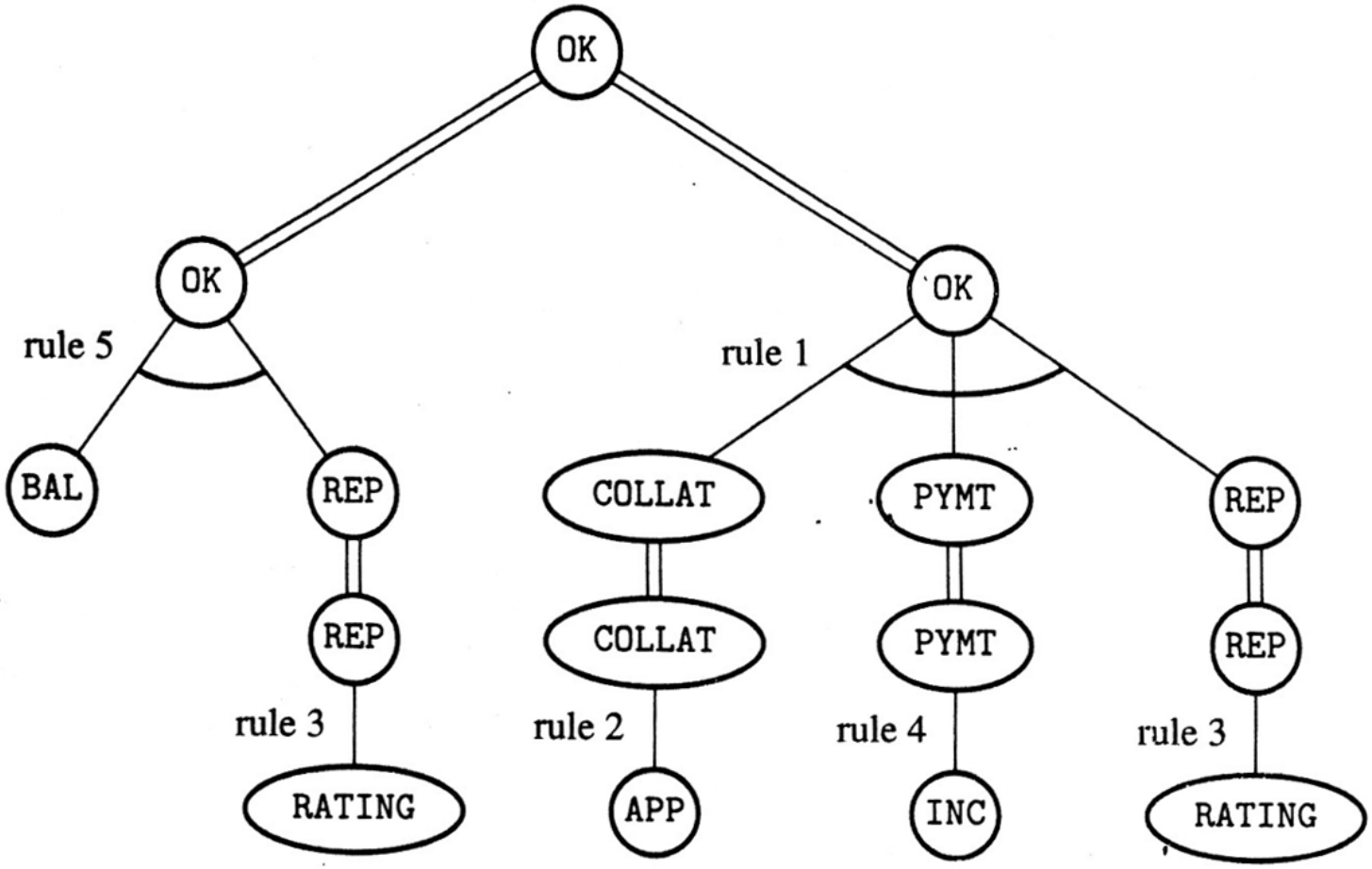
$\text{APP} \Rightarrow \text{COLLAT}$

$\text{RATING} \Rightarrow \text{REP}$

$\text{INC} \Rightarrow \text{PYMT}$

$\text{BAL} \wedge \text{REP} \Rightarrow \text{OK}$

Entscheidungsbaum (Und/Oder-Baum)



Modi der Verwendung

Anfragemodus:

Fakten werden aus Datenbank geholt oder Benutzer wird gefragt

Erklärungsmodus:

System beantwortet Fragen, wie Entscheidung gefällt wurde

Beispiele:

Nutzer: „Warum glaubst du, dass *das Darlehen bewilligt werden soll*?“

System: „Weil *die Sicherheit für das Darlehen genügend ist*, und *der Antragsteller die Darlehenszahlungen tilgen kann*, und *der Antragsteller eine gute finanzielle Reputation hat.*“

Anfrage- und Erklärungsmodus

Nutzer: „Warum glaubst du, dass *der Antragsteller eine gute finanzielle Reputation hat*?“

System: „Weil die Schufa gesagt hat, dass *der Antragsteller eine gute Bonität hat*.“

Nutzer: „Warum hast du mich gefragt, ob *das Einkommen des Antragstellers größer ist als seine Ausgaben* oder nicht?“

System: „Weil ich versucht habe, herauszufinden ob *der Antragsteller die Darlehenszahlungen tilgen kann* oder nicht.“

Nutzer: „Warum versuchst du herauszufinden, ob *der Antragsteller die Darlehenszahlungen tilgen kann* oder nicht?“

System: „Weil es eines der Kriterien zum Herausfinden ob *das Darlehen bewilligt werden sollte* oder nicht.“

Nutzer: „Warum versuchst du herauszufinden, ob *das Darlehen*

Wissensbasierte Systeme

Einzelne Regel = (im Idealfall) Wissenseinheit

Menge von Regeln zusammen mit Ausführungsstrategie = Programm für Lösung eines Problems oder Problemklasse

Prolog-Programm: *Regelbasiertes System*

Regelbasierte Systeme: historisch erste Formen von KI-Systemen

Lange Zeit als prototypisch für *Expertensysteme* angesehen

Inzwischen: nicht jedes Expertensystem benutzt Regeln als Hauptinferenzmechanismus

Regelmodule: wichtige Bestandteile von Expertensystemen

Regelsysteme gewinnen erneut Bedeutung
z.B. bei Kodifizierung von *Geschäftsprozess-Regeln*

XCON/R1

Beispiel für ein Konfigurationssystem

XCON/R1: Werkzeug für Konfiguration von *DEC Vax-Computern*

Seit ca. 1980 entwickelt und löst folgende Aufgaben:

- Identifikation fehlender Komponenten,
- Platzierung der Komponenten bezüglich Bussen, Schnittstellen und Gehäusen

Implementiert in *OPS5* (vorwärtsverkettetes regelbas. System)

Eine der ersten erfolgreichen kommerziellen Anwendungen von regelbasierten Expertensystemen

Hatte großen Einfluss auf industrielles Interesse an Expertensystemen

Wurde ständig weiterentwickelt: heute ca. 10.000 Regeln

Löst routinemäßig Aufgaben mit 100–200 Komponenten

Beispiel einer XCON/R1 Regel

IF

the most current active context is distributing massbus devices, and there is a single-port disk drive that has not been assigned to a massbus, and there are no unassigned dual-port disk drives, and the number of devices that each massbus should support is known, and there is a massbus that has been assigned at least one disk drive and that should support additional disk drives, and the type of cable needed to connect the disk drive to the previous device on the massbus is known

THEN

assign the disk drive to the massbus.

Übersicht

Was sind Regeln?

Formalisierte Konditionalsätze der Form

Wenn A **dann** B.

Bedeutung:

Wenn A wahr (erfüllt, bewiesen) ist,
dann schließe, dass auch B wahr ist.

A und B sind Aussagen

A (Wenn-Teil): Prämisse, Antezedenz

B (Dann-Teil): Konklusion, Konsequenz

Begriffliches

„Regel feuert“: Prämisse erfüllt

„Deterministische Regel“: Regel immer gültig

Häufige Schreibweise einer Regel: $A \rightarrow B$

Beispiel: Assoziationsregel (Warenkorbanalyse) *Bier* $\rightarrow$ *Chips*

Regel beinhaltet Expertenwissen (aus Analyse)

Liefert wertvolle Hinweise (z.B. zur Gestaltung des Ladenaufbaus)

Praktischer Nutzwert

sSehr guter Kompromiss zw. Verständlichkeit der Wissensdarstellung und formalen Ansprüchen

vVgl.: Konditionalsätze in natürlicher Sprache

bBabylonische Tafeln regelten Alltagsleben der Menschen und benutzten *Wenn-dann*-Konstrukte

Menschen sind intuitiv mit Regeln vertraut

Expertenwissen lässt sich häufig sehr gut mit Regeln modellieren, da Regeln menschlichem Denken sehr nahekommen

Festlegungen

Notwendig: möglichst einfache syntaktische Form der Regeln (Effizienz der Bearbeitung, Übersichtlichkeit)

Häufig gefordert: mindestens zwei Bedingungen

- Verknüpfung $\vee$ (*oder*) darf nicht in Prämisse auftreten
- Konklusion soll nur aus *einem* Literal (positives oder negiertes Atom) bestehen

Falls Regeln beiden Bedingungen nicht genügen, dann u.U. Vereinfachung mittels logischer Äquivalenzen

Beispiel

Regel: *Wenn es morgen regnet oder schneit, gehen wir ins Kino oder bleiben zu Hause.* verletzt beide Bedingungen; daher umformen zu 4 Regeln:

Wenn es morgen regnet und wir nicht ins Kino gehen, dann bleiben wir zu Hause.

Wenn es morgen regnet und wir nicht zu Hause bleiben, dann gehen wir ins Kino.

Wenn es morgen schneit und wir nicht ins Kino gehen, dann bleiben wir zu Hause.

Wenn es morgen schneit und wir nicht zu Hause bleiben, dann gehen wir ins Kino.

Einer komplexen Regel entsprechen hier *4* vereinfachte Regeln.

Übersicht

Regelumformungen

In klassischer Logik: Implikation repräsentiert Regel

$$A \rightarrow B \equiv \neg A \vee B$$

I.A. durch Distributivgesetze und de Morgan'schen Regeln immer erreichbar:

Prämisse A ist Disjunktion von Konjunktionen K_i von Literalen

Konklusion B ist Konjunktion von Disjunktionen D_j von Literalen

Regelumformungen

Transformation von komplexeren Regeln in syntaktisch einfachere durch (ggf. wiederholte) Anwendung der folgenden Schritte:

1. Ersetze Regel

$$\textbf{if } K_1 \vee \ldots \vee K_n \textbf{ then } D_1 \wedge \ldots \wedge D_m$$

durch $n \cdot m$ Regeln

$$\textbf{if } K_i \textbf{ then } D_j,\ i \in \{1, \ldots, n\},\ j \in \{1, \ldots, m\}$$

2. Ersetze Regel

$$\textbf{if } K \textbf{ then } L_1 \vee \ldots \vee L_p$$

(wobei K Konjunktion von Literalen ist) durch p Regeln

$$\textbf{if } K \wedge (\bigwedge_{k \neq k_0} \neg L_k) \textbf{ then } L_{k_0}, k_0 \in \{1, \ldots, p\}$$

Beispiel: Geldautomat

R_1 :	**if**	Karte	$=$	gültig	**and**
		PIN	$=$	richtig	**and**
		Versuche	$\neq$	überschritten	**and**
		Betrag	$\leq$	Maximalbetrag	**and**
		Kontostand	$=$	ausreichend	
	then	Auszahlung	$=$	soll erfolgen	
R_2:	**if**	Versuche	$=$	überschritten	
	then	Kartenrückgabe	$=$	nein	

Äquivalenzregeln werden allerdings verletzt:

Auszahlung *genau dann wenn* keine Voraussetzung verletzt

Karte einbehalten *genau dann wenn* Anzahl der zulässigen Versuche überschritten

Beispiel: Geldautomat

$\Rightarrow$ Erweiterung der Regelbasis um Gegenstücke zu R_1 und R_2:

R_1':	**if**	Auszahlung	$=$	soll erfolgen	
	then	Karte	$=$	gültig	**and**
		PIN	$=$	richtig	**and**
		Versuche	$\neq$	überschritten	$=$ **and**
		Betrag	$\leq$	Maximalbetrag	**and**
		Kontostand	$=$	ausreichend	
R_2':	**if**	Kartenrückgabe	$=$	nein	
	then	Versuche	$=$	überschritten	

Beispiel: Geldautomat

Führt zu logisch äquivalenten Regeln:

R_1'':	**if**	Karte	$=$	ungültig	**or**
		PIN	$=$	falsch	**or**
		Versuche	$=$	überschritten	**or**
		Betrag	$>$	Maximalbetrag	**or**
		Kontostand	$\neq$	nicht ausreichend	
	then	Auszahlung	$\neq$	soll erfolgen	
R_2'':	**if**	Versuche	$\neq$	überschritten	
	then	Kartenrückgabe	$=$	ja	

Übersicht

Wissensbasis eines regelbasierten Systems

Besteht aus *Objekten* und deren Beschreibungen mittels endlicher Mengen diskreter *Werte*

Regeln repräsentieren Zusammenhänge zw. Objekten oder Mengen von Objekten

Objekte und Regeln bilden *abstraktes Wissen* der Wissensbasis

Bei Anwendung auf konkreten Fall kommt *fallspezifisches Wissen* hinzu:

- Unmittelbare Beobachtungen und
- Abgeleitetes Wissen,

auch *Evidenz* genannt um zu betonen, dass für ein Faktum Anhaltspunkte oder Beweise vorliegen

Beispiel: Geldautomat – abstraktes Wissen

Parameter	**mögliche Werte**
Karte	{gültig, ungültig}
PIN	{richtig, falsch}
Versuche	{überschritten, nicht überschritten}
Kontostand	{ausreichend, nicht ausreichend}
Betrag	{$\leq$ Maximalbetrag, $>$Maximalbetrag}
Auszahlung	{soll erfolgen, soll nicht erfolgen}
Kartenrückgabe	{ja, nein}

Beispiel: Geldautomat – fallspezifisches Wissen

Kunde tritt an Automaten heran und möchte Geld abheben
Er erfüllt alle Bedingungen, allerdings wünscht er eine Auszahlung, die nicht durch seinen Kontostand gedeckt ist
Fallspezifisches Wissen:

Karte	$=$	gültig
PIN	$=$	richtig
Versuche	$\neq$	überschritten
Betrag	$\leq$	Maximalbetrag
Kontostand	$=$	nicht ausreichend

mit Hilfe der Wissensbasis $\Rightarrow$ Auszahlung soll nicht erfolgen

Übersicht

Inferenz im regelbasierten System

grundlegende Inferenzregel: *modus ponens*

if A **then** B	(Regel)
A wahr	(Faktum)
B wahr	(Schlussfolgerung)

im Folgenden:

Verkettung von Regeln

Regelnetzwerke

Vorwärtsverkettung

Rückwärtsverkettung

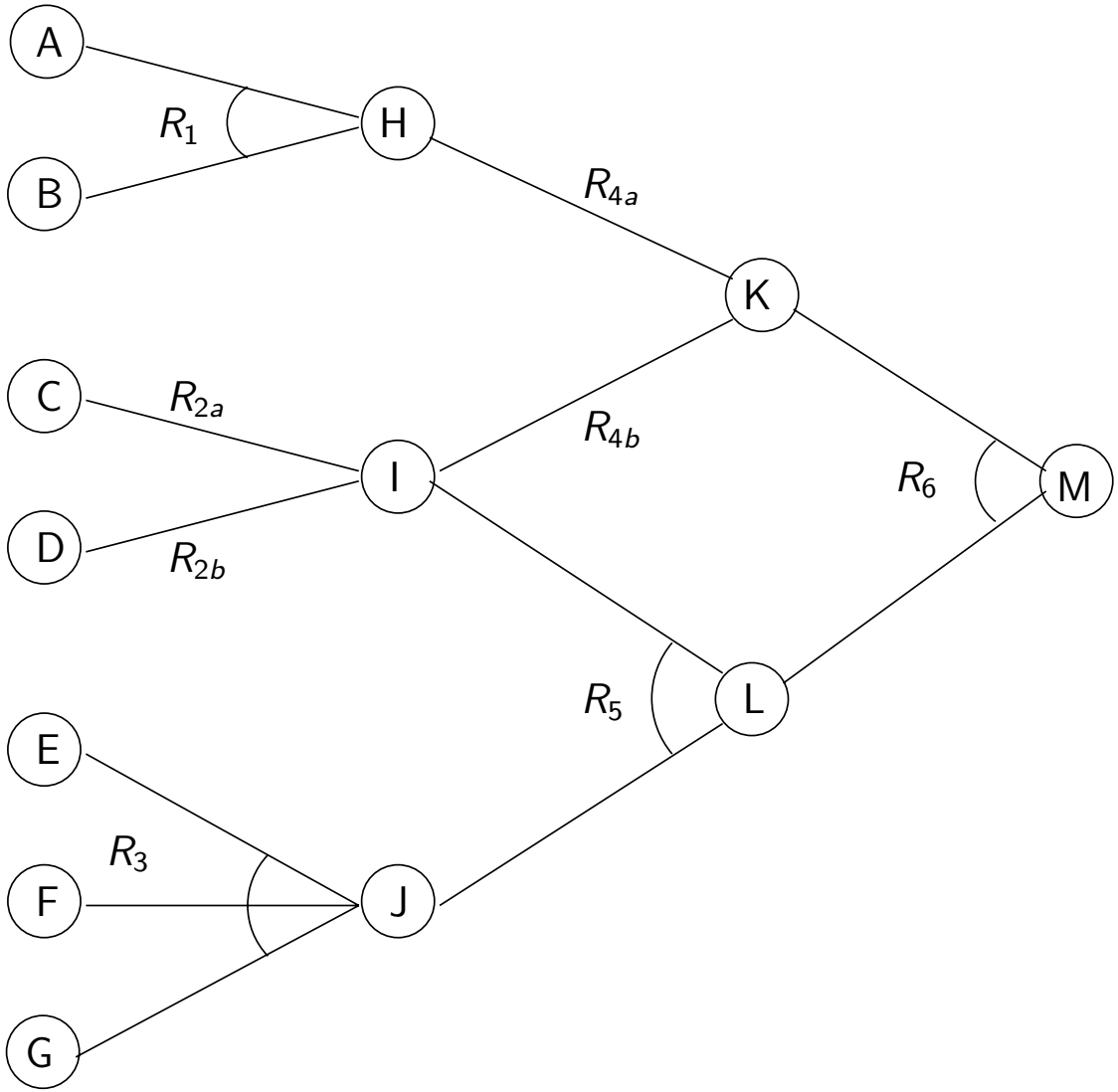
Wissensbasis, Regelnetzwerk

Objekte: A, B, C, D, E, F, G, H, I, J, K, L, M

Regeln:

$$
\begin{array}{lllll}
R_1: & \text{if} & A \wedge B & \text{then} & H \\
R_2: & \text{if} & C \vee D & \text{then} & I \\
R_{2a}: & \text{if} & C & \text{then} & I \\
R_{2b}: & \text{if} & D & \text{then} & I \\
R_3: & \text{if} & E \wedge F \wedge G & \text{then} & J \\
R_4: & \text{if} & H \vee I & \text{then} & K \\
R_{4a}: & \text{if} & H & \text{then} & K \\
R_{4b}: & \text{if} & I & \text{then} & K \\
R_5: & \text{if} & I \wedge J & \text{then} & L \\
R_6: & \text{if} & K \wedge L & \text{then} & M
\end{array}
$$

Wissensbasis, Regelnetzwerk



Datengetriebene Inferenz

Alternative Bezeichnung: Vorwärtsverkettung

Fallspezifisches Wissen als Ausgangspunkt für Inferenzprozess

Aus erfüllten Prämissen wird auf Wahrheit der Konklusion geschlossen

Abgeleitete Fakten gehen erneut in Wissensbasis ein

Datengetriebene Inferenz

Algorithmus 1 ForwardChain

Eingabe: Wissensbasis RB (Objekte, Regeln), Menge $\mathcal{F}$ von Fakten

Ausgabe: Menge der gefolgerten Fakten

1: sei $\mathcal{F}$ Menge der gegebenen (evidentiellen) Fakten
2: **for each** Regel **if** A **then** B aus RB $\{$
3: ist A erfüllt, so schließe auf B
4: $\mathcal{F} := \mathcal{F} \cup \{B\}$
5: $\}$
6: gehe zu Schritt 2, bis $\mathcal{F}$ nicht mehr vergrößert werden kann

Datengetriebene Inferenz: Beispiel

Beispiel:

Gegeben seien Fakten $\mathcal{F} = \{H, C, E, F, G\}$

Damit feuern Regeln R_{2a}, R_{4a} und R_3

Somit vergrößert sich $\mathcal{F}$ zu $\mathcal{F} := \{H, C, E, F, G\} \cup \{I, J, K\}$

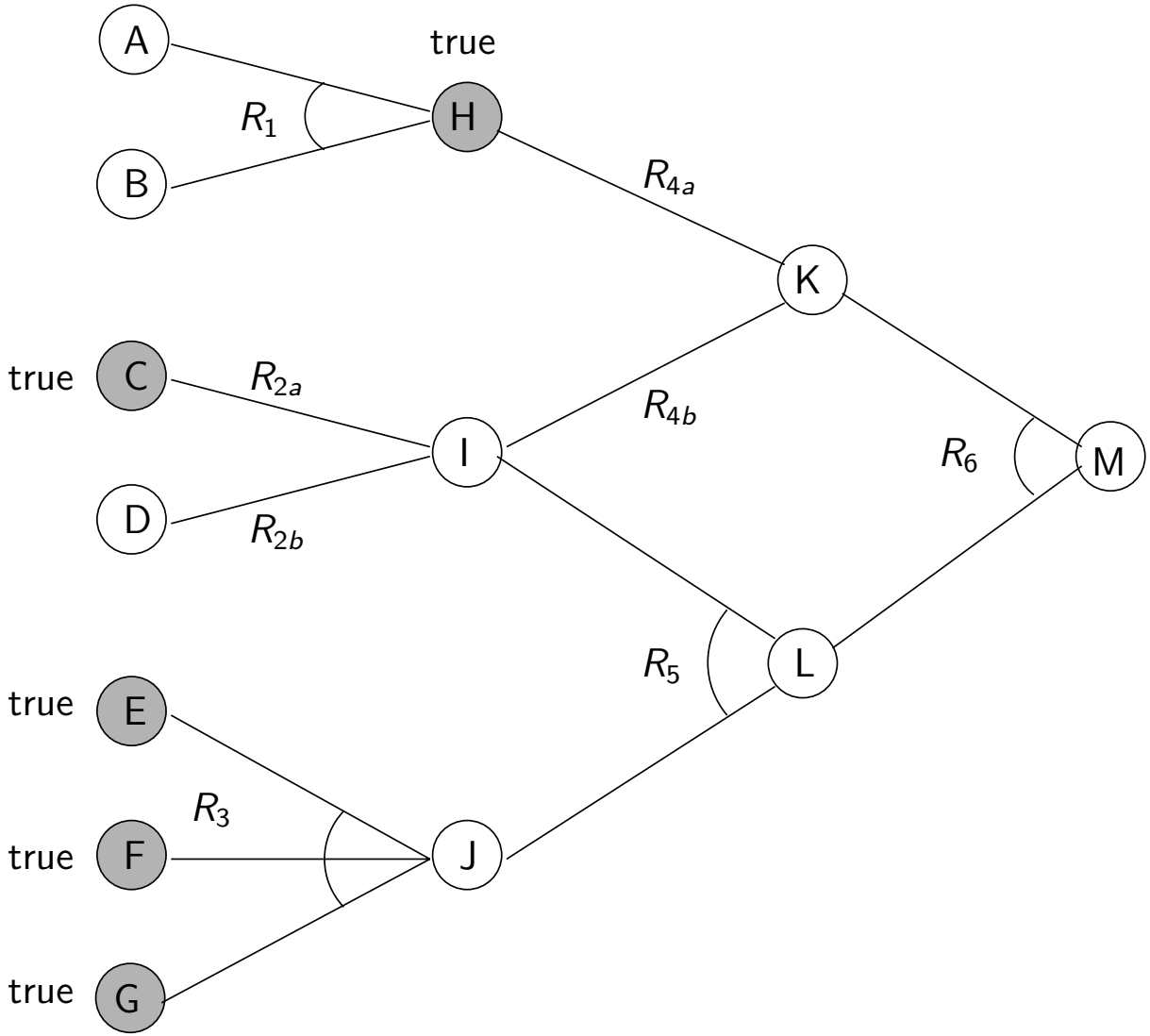
In weiterem Durchlauf feuern nun R_{4b} und R_5

$\mathcal{F} := \{H, C, E, F, GI, J, K\} \cup \{L\}$

Somit feuert nun R_6

Endgültige Faktenmenge $\mathcal{F} := \{H, C, E, F, G, IJ, K, L\} \cup \{M\}$

Datengetriebene Inferenz: Beispiel



Zielorientierte Inferenz

Alternative Bezeichnung: Rückwärtsverkettung

Zielobjekt Z als Ausgangspunkt

Durchsuchen der Regelbasis nach Regeln, die Z in Konklusion enthalten

Objekte der Prämissen werden zu Zwischenzielen

Zielorientierte Inferenz

Algorithmus 2 BackChain

Eingabe: Regelbasis RB, (evidentielle) Faktenmenge $\mathcal{F}$, Liste von Zielen (atomaren Anfragen) $[q_1, \ldots, q_n]$

Ausgabe: *yes*, falls alle q_i ableitbar sind, sonst *no*

```
 1: if n = 0 {
 2:     return yes
 3: }
 4: if q_1 ∈ F {
 5:     BackChain([q_2, ..., q_n])
 6: } else {
 7:     for each Regel p_1 ∧ ... ∧ p_m → q aus RB mit q_1 = q {
 8:         if BackChain([p_1, ..., p_m, q_2, ..., q_n]) = yes {
 9:             return yes
10:         }
11:     }
12: }
13: return no
```

Zielorientierte Inferenz: Beispiel

Wissensbasis enthalte (zweiwertigen) Objekte O_1, O_2, O_3 und Regeln:

- Regel 1: **if** O_1 **then** O_2
- Regel 2: **if** O_2 **then** O_3

Frage: Zustand des Zielobjektes O_3

- O_3 ist wahr, wenn O_2 wahr ist
- O_2 ist wahr, wenn O_1 wahr ist
- Informationen über O_1 benötigt

Problem der Widersprüchlichkeit

Falls Negation in Fakten oder Konklusion von Regeln erlaubt:
Regelbasis kann zu widersprüchlichen Ableitungen führen

in der Praxis häufig auftretendes Problem

Experten benutzen zur Formulierung der Regelbasis häufig:

- Implizite, unausgesprochene Annahmen,
- oder übersehen, dass Ausnahmen zu Regeln auftreten können

Beispiel-Regelbasis: **if** V **then** F, **if** $V \wedge P$ **then** $\neg F$

Instantiiere V und P mit *wahr* $\Rightarrow$ Schlüsse F und $\neg F$

Veranschaulichung: V – Vögel, P – Pinguine, F – Fliegen können

Problem der Widersprüchlichkeit

2 Möglichkeiten der Widersprüchlichkeit:

- *Klassisch-logisch inkonsistent:* es gibt keine Belegung der Objekte mit Werten, so dass alle Regeln erfüllt sind
- *Ableitungen:* Regelbasis führt zu widersprüchlichen Ableitungen

2. Fall ist häufiger (auch praktisch genutzt)

Beispiel:

- $\{\textbf{if } V \textbf{ then } F, \textbf{if } V \wedge P \textbf{ then } \neg F\}$ nicht inkonsistent
- $\{\textbf{if } V \textbf{ then } F, \textbf{if } V \wedge P \textbf{ then } \neg F,\ V \wedge P\}$ ist inkonsistent

Konsistenzüberprüfung: wichtige Warnfunktion

Dient der Verbesserung der Regelbasis

Die Erklärungskomponente

Regeln sind i.A. recht gut verständlich

Bei Schlussfolgerung aus Regelbasis können dazu herangezogene Regeln aufgelistet werden

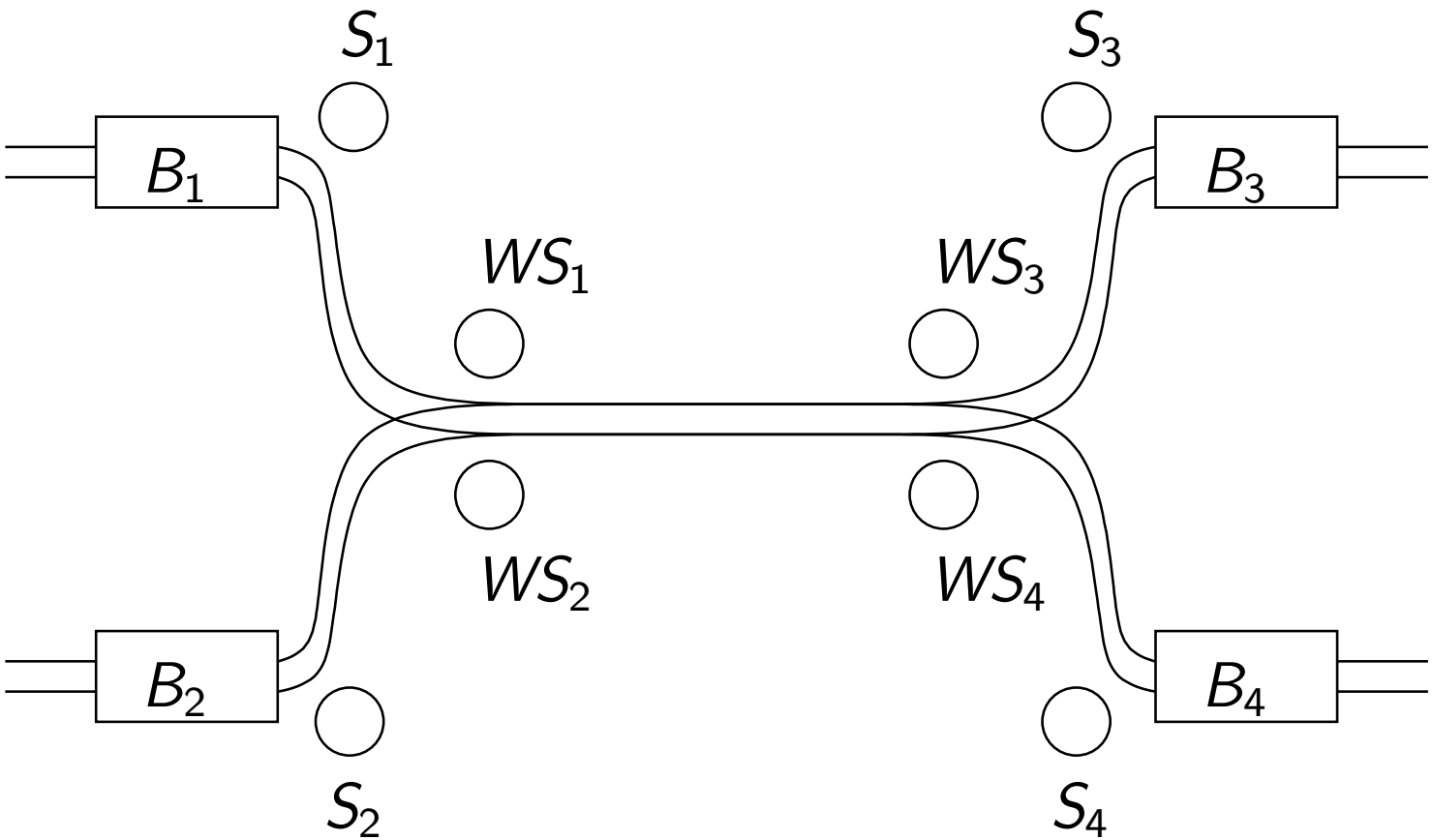
Argumentationskette entsteht

Beispiel (siehe „Datengetriebene Inferenz“):

- Geg. Fakten: H, C, E, F, G
- Schlussfolgerungen:

  I wegen Regel R_{2a}
  K wegen Regel R_{4a}
  J wegen Regel R_3
  L wegen Regel R_5
  M wegen Regel R_6

Beispiel: Signalsteuerung im Eisenbahnverkehr



Beispiel: Signalsteuerung im Eisenbahnverkehr

Objekte	mögliche Werte
S_1, S_2, S_3, S_4	{rot,grün}
WS_1, WS_2, WS_3, WS_4	{rot,grün}
B_1, B_2, B_3, B_4	{frei, belegt}

S_i – Bahnhofssignale

WS_i – Weichensignale

B_i – Bahnhöfe

Alle gezeigten Strecken sind eingleisig

Gefahrloser Bahnverkehr soll durch regelbasiertes System gewährleistet werden

Beispiel: Signalsteuerung im Eisenbahnverkehr

Vermeidung von Zusammenstößen auf Strecke — immer nur höchstens 1 der S_i auf grün, Rest rot:

R1:	**if**	S_1	=	grün	**then**	S_2	=	rot
R2:	**if**	S_1	=	grün	**then**	S_3	=	rot
R3:	**if**	S_1	=	grün	**then**	S_4	=	rot
R4:	**if**	S_2	=	grün	**then**	S_1	=	rot
R5:	**if**	S_2	=	grün	**then**	S_3	=	rot
R6:	**if**	S_2	=	grün	**then**	S_4	=	rot
R7:	**if**	S_3	=	grün	**then**	S_1	=	rot
R8:	**if**	S_3	=	grün	**then**	S_2	=	rot
R9:	**if**	S_3	=	grün	**then**	S_4	=	rot
R10:	**if**	S_4	=	grün	**then**	S_1	=	rot
R11:	**if**	S_4	=	grün	**then**	S_2	=	rot
R12:	**if**	S_4	=	grün	**then**	S_3	=	rot

Beispiel: Signalsteuerung im Eisenbahnverkehr

Kein Zug in einem belegten Bahnhof:

R13:	**if**	B_1	$=$	belegt	**then**	WS_1	$=$	rot
R14:	**if**	B_2	$=$	belegt	**then**	WS_2	$=$	rot
R15:	**if**	B_3	$=$	belegt	**then**	WS_3	$=$	rot
R16:	**if**	B_4	$=$	belegt	**then**	WS_4	$=$	rot

Mittlerer Teil der Strecke darf nicht durch wartende Züge blockiert werden:

R17:	**if**	$WS_1 = rot$	$\wedge$	$WS_2 = rot$	**then**	S_3	$=$	rot
R18:	**if**	$WS_1 = rot$	$\wedge$	$WS_2 = rot$	**then**	S_4	$=$	rot
R19:	**if**	$WS_3 = rot$	$\wedge$	$WS_4 = rot$	**then**	S_1	$=$	rot
R20:	**if**	$WS_3 = rot$	$\wedge$	$WS_4 = rot$	**then**	S_2	$=$	rot

Beispiel: Signalsteuerung im Eisenbahnverkehr

WS_1/WS_2 bzw. WS_3/WS_4 sind Weichensignale, d.h. es kann höchstens eine der beiden zugehörigen Strecken freigegeben werden:

R21:	**if**	WS_1	=	grün	**then**	WS_2	=	rot
R22:	**if**	WS_2	=	grün	**then**	WS_1	=	rot
R23:	**if**	WS_3	=	grün	**then**	WS_4	=	rot
R24:	**if**	WS_4	=	grün	**then**	WS_3	=	rot

Beispiel: Signalsteuerung im Eisenbahnverkehr

Beispielsituation

B_1 und B_3 seien belegt:

$\mathcal{F}_1$:	B_1	$=$	belegt
	B_3	$=$	belegt

ableitbare Aussagen:

$\mathcal{C}_1$:	WS_1	$=$	rot	(R13)
	WS_3	$=$	rot	(R15)

Beispiel: Signalsteuerung im Eisenbahnverkehr

Zug fährt in Bahnhof B_2 ein:

$\mathcal{F}_2$:	B_1	$=$	belegt
	B_3	$=$	belegt
	B_2	$=$	belegt

Ableitbare Aussagen:

$\mathcal{C}_1$:	WS_1	$=$	rot	(R13)
	WS_3	$=$	rot	(R15)
	WS_2	$=$	rot	(R14)
	S_3	$=$	rot	(R17)
	S_4	$=$	rot	(R18)

Beispiel: Signalsteuerung im Eisenbahnverkehr

Fahrt für Zug in Bahnhof B_1 wird freigegeben, d.h. S_1 wird auf grün gestellt:

$\mathcal{F}_2$:	B_1	$=$	belegt
	B_3	$=$	belegt
	B_2	$=$	belegt
	S_1	$=$	grün

Ableitbare Aussagen:

$\mathcal{C}_1$:	WS_1	$=$	rot	(R13)
	WS_3	$=$	rot	(R15)
	WS_2	$=$	rot	(R14)
	S_3	$=$	rot	(R17 und R2)
	S_4	$=$	rot	(R18 und R3)
	S_2	$=$	rot	(R1)

Monotones Schließen

Menge der Schlussfolgerungen wächst *monoton* mit Faktenmenge

Bedingt durch Allgemeingültigkeit der Regeln

Kann nicht immer garantiert werden

D.h. es kann vorkommen, dass bereits gezogene Schlussfolgerungen revidiert werden müssen

Führt zu *nichtmonotonem* Ableitungsverhalten

Übersicht

Modus ponens und Resolution

Einfache, intuitive Regel der Inferenz

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

Durch Gültigkeit von A und $A \to B$ kann B geschlussfolgert werden

Alternative: **Resolutionregel**

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

Klausel $A \vee C$ heißt *Resolvente*

Einfache Transformation führt zu

$$\frac{A \vee B, \quad B \to C}{A \vee C}$$

Verallgemeinerte Resolution

Verallgemeinerung durch beliebige Anzahl von Literalen

Mit Literalen $A_1, \ldots, A_m, B, C_1, \ldots, C_n$ erhält man

$$\frac{(A_1 \vee \ldots \vee A_m \vee B), \quad (\neg B \vee C_1 \vee \ldots \vee C_n)}{(A_1 \vee \ldots \vee A_m \vee C_1 \ldots \vee \vee C_n)}$$

Literale B und $\neg B$ heißen *komplementär*

Resolutionsregel entfernt ein Paar von komplementären Literalen aus zwei Klauseln

Restlichen Literale werden als neue Klausel kombiniert

Widerspruchsbeweis

Zu zeigen: aus Wissensbasis KB folgt Anfrage Q

Lösung: durch Widerspruch von $KB \wedge \neg Q$

Ziel: Erzeugen zweier Klauseln (A) und $(\neg A)$ aus KNF, die zur leeren Klauselmenge als Resolvente führen

Beispiel: Englische Familie

Obwohl ich 7 lange Jahre Englisch mit großem Erfolg studiert habe, muss ich zugeben, dass ich total verwirrt bin, wenn ich Engländer Englisch sprechen hören. Kürzlich habe ich aufgrund nobler Gefühle drei Anhalter mitgenommen, Vater, Mutter und deren Tochter, die Engländer waren und nur Englisch sprachen, wie ich schnell herausfandt. Bei jedem der Sätze, die folgten, schwankte ich zwischen zwei Interpretationen. Sie erzählten mir Folgendes (die zweite mögliche Bedeutung in steht dabei in Klammern): Der Vater: „Wir fahren nach Spanien (Wir sind aus Newcastle).“ Die Mutter: „Wir fahren nicht nach Spanien und sind aus Newcastle (Wir stoppten in Paris und fahren nicht nach Spanien).“ Die Tochter: „Wir sind nicht aus Newcastle (Wir stoppten in Paris).“ Was hat es mit dieser bezaubernden englischen Familie auf sich?

Beispiel: Englische Familie

Lösung in drei Schritte: Formalisierung, Transformation in KNF, Beweis

Für gewöhnlich: Formalisierung am schwierigsten

S für „Wie fahren nach Spanien“

N für „Wir sind aus Newcastle“

P für „Wir stoppten in Paris“

Somit erhalten wir mit drei Aussagen von Vater, Mutter, Tochter:

$$(S \vee N) \wedge [(\neg S \wedge N) \vee (P \wedge \neg S)] \wedge (\neg N \vee P)$$

Beispiel: Englische Familie

$$(S \vee N) \wedge [(\neg S \wedge N) \vee (P \wedge \neg S)] \wedge (\neg N \vee P)$$

Herausziehen von $\neg S$ in der Mitte bringt KNF

Durchnummerieren der Klauseln führt zu

$$KB \equiv (S \vee N)_1 \wedge (\neg S)_2 \wedge (P \vee N)_3 \wedge (\neg N \vee P)_4$$

Jetzt Resolutionsbeweis erst mal ohne Anfrage Q

Notation: $\text{Res}(m, n) : (\text{Klausel})_k$ bedeutet, (Klausel) durch Resolution von m und n entstanden und numeriert mit k

Beispiel: Englische Familie

$$KB \equiv (S \vee N)_1 \wedge (\neg S)_2 \wedge (P \vee N)_3 \wedge (\neg N \vee P)_4$$

$$\begin{aligned} \text{Res}(1,2) &: (N)_5 \\ \text{Res}(3,4) &: (P)_6 \\ \text{Res}(1,4) &: (S \vee P)_7 \end{aligned}$$

P könnte auch durch $\text{Res}(4,5)$ oder $\text{Res}(2,7)$ entstehen

Keine weiteren Resolutionsschritte ohne neue Klauseln möglich

Leere Klausel nicht erzeugt: also ist KB widerspruchsfrei

Beispiel: Englische Familie

Bisher haben wir N und P hergeleitet

Zu zeigen: $\neg S$ durch Hinzufügen von $(S)_8$ zur Menge aller Klauseln als negierte Anfrage

Demnach $\text{Res}(2, 8) : ()_9$

Also gilt $\neg S \wedge N \wedge P$

Die „bezaubernde englische Familie“ kommt bewiesenermaßen aus Newcastle, stoppten in Paris, fahren aber nicht nach Spanien

Beispiel: Hochsprung

Drei Mädchen üben Hochsprung für ihre Note im Sportunterricht. Die Stange liegt bei $1.20\,\mathrm{m}$. „Ich wette“, sagt das erste Mädchen zum Zweiten, „dass ich es genau dann drüber schaffen werden, wenn du es nicht schaffst.“ Wenn das zweite Mädchen das Gleiche zum Dritten sagen würde, die genau das Gleiche wiederum zum Ersten sagen würden, wäre es dann für älle Drei, ihre Wetten zu gewinnen?

Beispiel: Hochsprung

Formalisierung:

Sprung des 1. Mädchens glückt: A

Sprung des 2. Mädchens glückt: B

Sprung des 3. Mädchens glückt: C

Wette des 1. Mädchens": $(A \leftrightarrow \neg B)$

Wette des 2. Mädchens": $(B \leftrightarrow \neg C)$

Wette des 3. Mädchens": $(C \leftrightarrow \neg A)$

Behauptung:

$$Q \equiv \neg((A \leftrightarrow \neg B) \wedge (B \leftrightarrow \neg C) \wedge (C \leftrightarrow \neg A))$$

durch Resolution zu zeigen: $\neg Q$ ist unerfüllbar

Beispiel: Hochsprung

Transformation in KNF
Wette des 1. Mädchens:

$$(A \leftrightarrow \neg B) \equiv (A \rightarrow \neg B) \wedge (\neg B \rightarrow A) \equiv (\neg A \vee \neg B) \wedge (A \vee B)$$

Analog für 2. und 3. Mädchen
Negierte Behauptung:

$$\neg Q \equiv (\neg A \vee \neg B)_1 \wedge (A \vee B)_2 \wedge (\neg B \vee \neg C)_3 \wedge (B \vee C)_4 \wedge (\neg C \vee \neg A)_5 \wedge (C \vee A)_6$$

Beispiel: Hochsprung

$$\neg Q \equiv (\neg A \vee \neg B)_1 \wedge (A \vee B)_2 \wedge (\neg B \vee \neg C)_3 \wedge (B \vee C)_4 \wedge (\neg C \vee \neg A)_5 \wedge (C \vee A)_6$$

$$\begin{aligned} \text{Res}(1,6) &: (C \vee \neg B)_7 \\ \text{Res}(4,7) &: (C)_8 \\ \text{Res}(2,5) &: (B \vee \neg C)_9 \\ \text{Res}(3,9) &: (\neg C)_1 0 \\ \text{Res}(8,10) &: () \end{aligned}$$

Übersicht

Klauseln mit max. 1 positivem Literal

Klausel in KNF enthält positive und negierte Literale

Repräsentation als

$$\neg A_1 \vee \ldots \vee \neg A_m \vee B_1 \vee \ldots \vee B_n$$

mit Variablen $A_1, \ldots, A_m$ und $B_1, \ldots, B_n$

Transformation in

$$A_1 \wedge \ldots \wedge A_m \rightarrow B_1 \vee \ldots \vee B_n$$

Z.B. „Wenn das Wetter schön ist and Schnee auf dem Boden liegt, dann fahre ich Ski oder werde arbeiten.“

Hier Schlussfolgerung: Schwimmen geht er definitiv nicht!

Z.B. „Wenn das Wetter schön ist and Schnee auf dem Boden liegt, dann fahre ich Ski.“

Hier wissen wir definitiv, was er machen wird

Hornklauseln

Hornklauseln = Klauseln mit maximal einem positiven Literal

$$\neg A_1 \vee \ldots \vee \neg A_m \vee B$$

bzw.

$$A_1 \wedge \ldots \neg A_m \rightarrow B$$

Klausel mit einem positiven Literal ist ein **Fakt**

In Klauseln mit negativem und einem positivem Literal heißt das positive literal **Kopf**

Beispiel: Ski fahren

$$(\text{schönes Wetter})_1$$
$$(\text{Schneefall})_2$$
$$(\text{Schneefall} \rightarrow \text{Schnee})_3$$
$$(\text{schönes Wetter} \wedge \text{Schnee} \rightarrow \text{Ski fahren})_4$$

Wenn wir wissen wollen, ob Ski fahren gilt, dann Modus ponens nutzen:

$$\frac{A_1 \wedge \ldots \wedge A_m, \quad A_1 \wedge \ldots \wedge A_m \rightarrow B}{B}$$

$MP(i_1, \ldots, i_k)$ sei Anwendung des Modus ponens auf Klauseln i_1 bis i_k

Beispiel: Ski fahren

$$MP(2,3) : (\text{Schnee})_5$$
$$MP(1,5,4) : (\text{Ski fahren})_6$$

Mit Modus ponens: komplette Berechnung für Formeln, die aus Hornklauseln bestehen

Für große Wissensbasen für MP zu vielen unnötigen Formeln (wenn mit falscher Formel begonnen wird)

Darum: Berechnung nutzen, die mit Anfrage anfängt und rückwärts arbeitet bis Fakten erreicht sind

backward chaining

Backward Chaining für Hornklauseln

SLD-Resolution wird genutzt

SLD: selektive regelgetriebene lineare Resolution für definite Klauseln

Beispiel:

$$(\text{schönes Wetter})_1$$
$$(\text{Schneefall})_2$$
$$(\text{Schneefall} \rightarrow \text{Schnee})_3$$
$$(\text{schönes Wetter} \wedge \text{Schnee} \rightarrow \text{Ski fahren})_4$$
$$(\text{Ski fahren} \rightarrow f)_5$$

Beispiel: Ski fahren

$$(\text{schönes Wetter})_1$$
$$(\text{Schneefall})_2$$
$$(\text{Schneefall} \rightarrow \text{Schnee})_3$$
$$(\text{schönes Wetter} \wedge \text{Schnee} \rightarrow \text{Ski fahren})_4$$
$$(\text{Ski fahren} \rightarrow f)_5$$

$$\text{Res}(5,4) : (\text{schönes Wetter} \wedge \text{Schnee} \rightarrow f)_6$$
$$\text{Res}(6,1) : (\text{Schnee} \rightarrow f)_7$$
$$\text{Res}(7,3) : (\text{Schneefall} \rightarrow f)_8$$
$$\text{Res}(8,2) : ()_{10}$$

„Lineare Resolution“: große Reduktion des Suchraums

Fest Abarbeitungsreihenfolge

SLD spielt wichtige Rolle in PROLOG

Beispiel: Implementierung über Prolog

„Roboterarm bewegt sich, wenn Batterie i.O. ist und Gewicht nicht zu schwer ist“

Programm entspricht der Aussage:

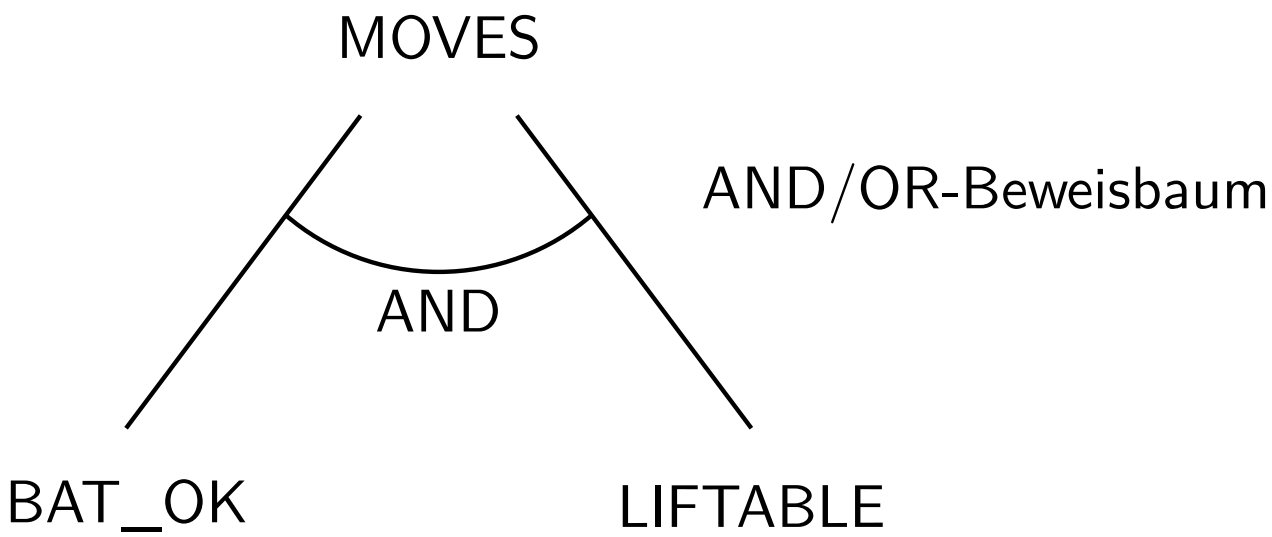
$$(\text{BAT\_OK} \wedge \text{LIFTABLE} \wedge (\text{BAT\_OK} \wedge \text{LIFTABLE} \Rightarrow \text{MOVES})) \Rightarrow \text{MOVES}$$

```
:- MOVES                          Ziel (Anfrage)
BAT_OK :-                         Fakten
LIFTABLE :-
MOVES :- BAT_OK, LIFTABLE         Regeln
```

Prolog beweist

Aussage mit „depth-first“ und „backward chaining“

Beispiel: Schlussfolgern mit Hornklauseln



Anmerkung:

Klausel 1 gegen Klausel 2 oder Klausel 3 geht nicht

Aber Klausel 1 vs. Klausel 4 liefert 2 neue Ziele (siehe Baum)

Diese können sofort bewiesen werden

Schlussfolgern mit Hornklauseln

Beispiel: Blockwelt

Ist Bauklotz A über C, wenn A auf B und B auf C?
Prolog:

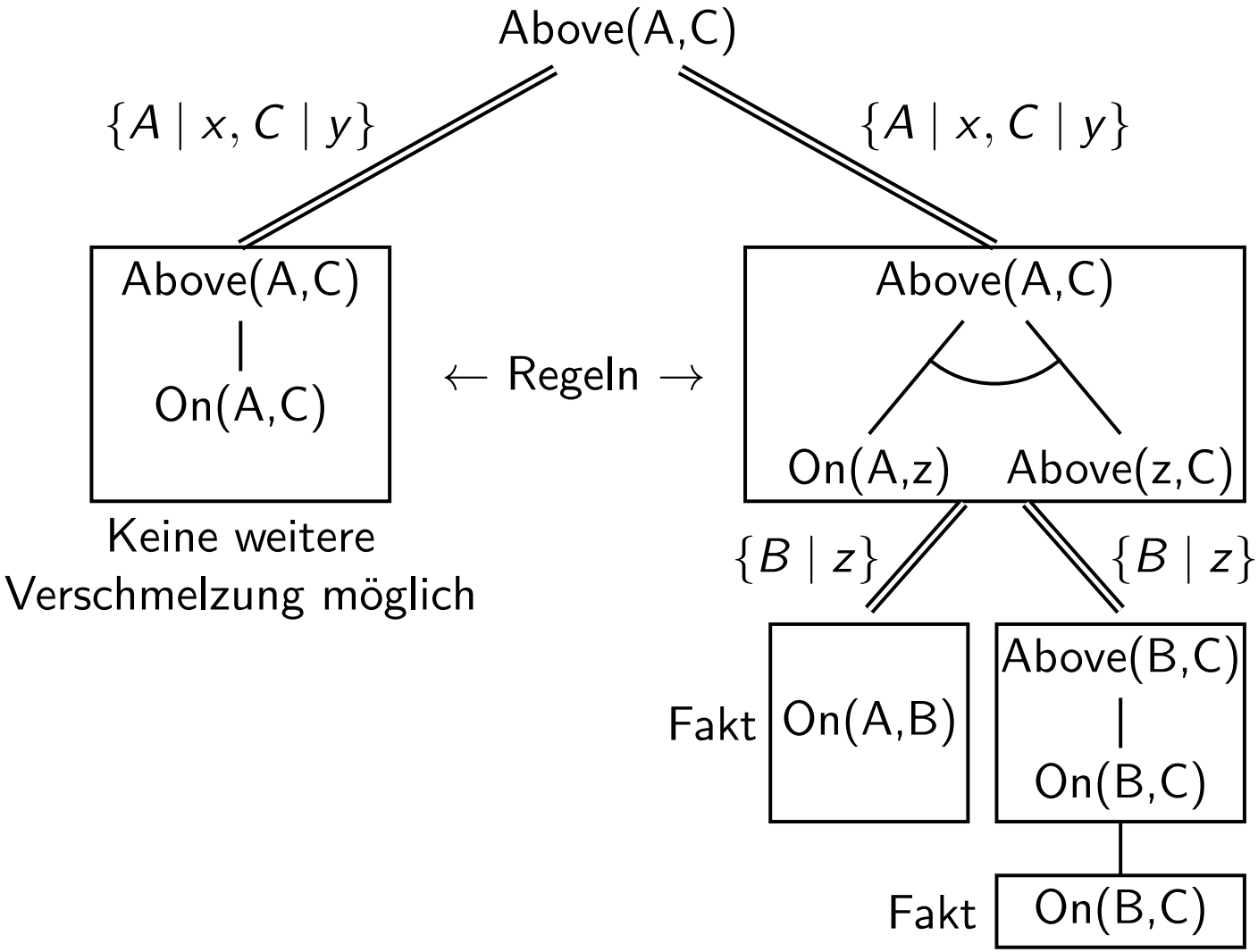
```
:- Above(A,C)
On(A,B) :-
On(B,C) :-
Above(x,y) :- On(x,y)
Above(x,y) :- On(x,z), Above(z,y)
```

PK1:

$$(\forall x, y, z)\ \mathrm{On}(x, y) \Rightarrow \mathrm{Above}(x, y)$$

$$(\forall x, y)\ (\exists z)\ \mathrm{On}(x, y) \wedge \mathrm{Above}(z, y) \Rightarrow \mathrm{Above}(x, y)$$

Schlussfolgern mit Hornklauseln



Übersicht

Wonach suchen wir, um eine Maschine zu verkörpern?

Geist

Gedanken

Intelligenz

Rationalität

Neuroanatomie

Wissen

Pyramide des Wissens

Giarratano und Riley (2005)

Weisheit: Nutzen von Wissen in einer vorteilhaften Weise

Metawissen: Regeln über Wissen

Wissen: Regeln über das Nutzen von Informationen

Informationen: potentiell nützlich für Wissen

Daten: potentiell nützliche Informationen

Intellektuelle Ursprünge

Vor 2000 Jahren: der **Syllogismus** (Logik von Aristoteles, Konklusionen)

Im 17. Jhd.: die **Algebra der Gedanken** (Leibniz: menschliches Schlussfolgern kann auf Berechnungen zurückgeführt werden, regelbasierte Manipulation)

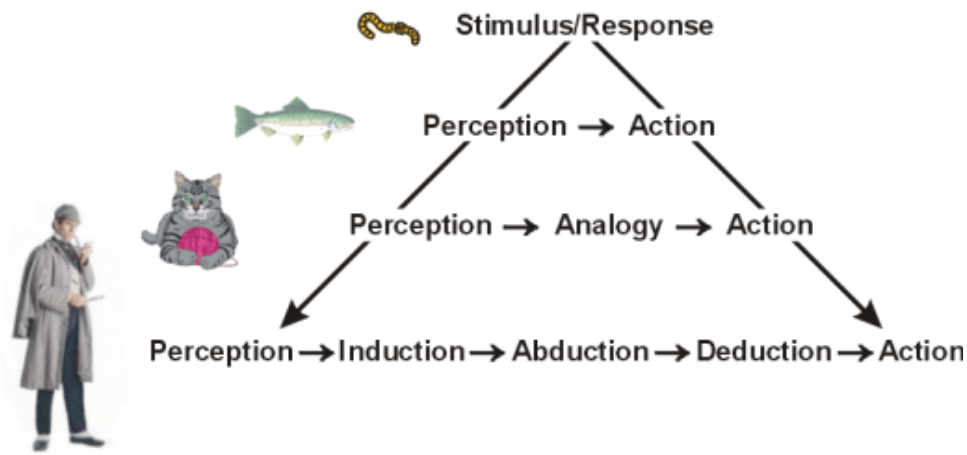
Im 19. Jhd.: **Boole'sche Logik**

Auch 19. Jhd.: **analytische Maschine** (Babbage, Ada Lovelace)

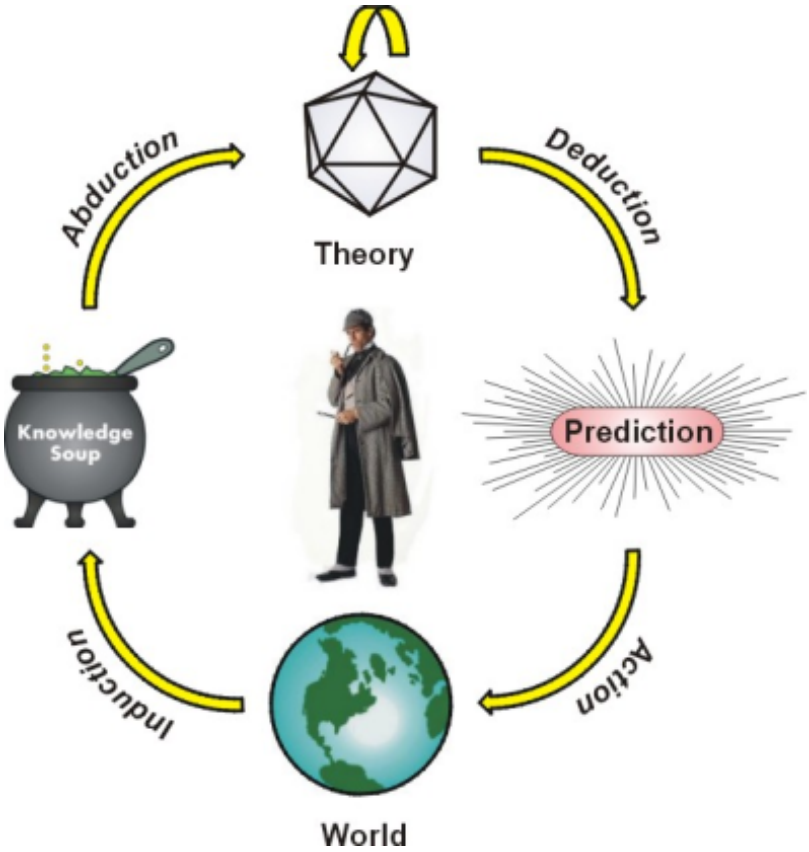
Im 20. Jhd.: Turings Ideen über Gedanken und Berechenbarkeit

1927 (McCarthy): Schlussfolgern kann auf Berechnungen reduziert werden

Kognitive Prozesse



Entwicklung der Kognition



Peirce'sche Logik des Pragmatismus

Beispiel: Das unordentliche Zimmer

von D.M. Gabbay (2010)

Eine Mutter geht in das Schlafzimmer ihrer Teenager-Tochter. Ihr erster Eindruck ist, dass dort eine große Unordnung herrscht. Es liegt überall irgendetwas herum. Der Eindruck der Mutter ist, dass dies untypisch für ihre Tochter sei, sich so zu verhalten. Was war passiert?
Vermutung: Ihre Tochter hat Probleme mit einem Jungen.

Weitere Analyse: Die Mutter bemerkt ein zusammengebrochenes Regal. Hat ihre Tochter es zerbrochen? Nach weiterer Betrachtung bemerkt sie anhand des Musters der Unordnung, dass das Regal zusammenbrach durch das zu hohe Gewicht und einfach alles verteilte. Aber eigentlich ist das keine Unordnung, sondern ergibt (anziehungskräftemäßig) Sinn.

Beispiel: Das unordentliche Zimmer

Verschiedene Modi von Schlussfolgerungen werden hier genutzt:
Schließen mittels neuronaler Netze; Sie erkennt die Unordnung sofort, so wie wir ein Gesicht erkennen.

Nicht-monotone Deduktion: Die Mutter schließt aus dem Kontext und ihrem Wissen über ihre Tochter, dass das Mädchen nicht so unordentlich ist. „Was ist passiert?“ fragt sie demnach.

Abduktion: Sie bietet eine sinnvolle Erklärung an, dass das Mädchen Probleme mit einem Jungen hat: Es ist normal in dem Alter.

Deduktion: Dann wendet sie Deduktion an und erkennt, dass die Unordnung von der Schwerkraft herrührt. Diese Deduktion ist nicht mehr ein „Neuronaler Netz“-Eindruck. Es ist eine sorgfältige Berechnung.
Merke: Punkt 4 kann ein „Neuronaler Netz“-Eindruck sein: jemand der viele Fälle von zerbrochenen Regalen sieht, erkennt dieses Muster.
Punkt 2 könnte auch ein Bayes-Netz gewesen sein.

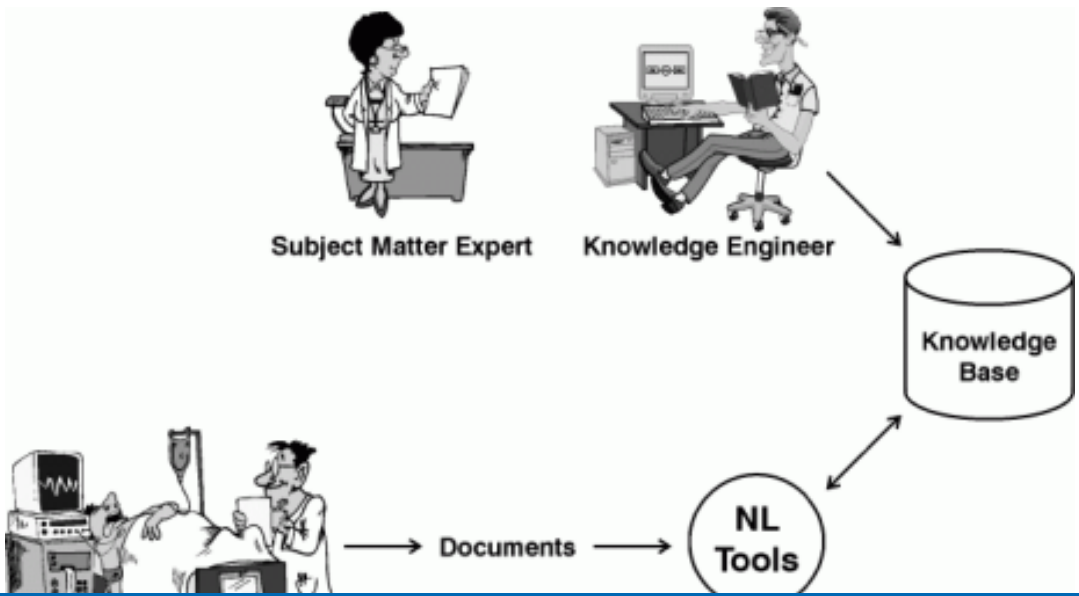
Übersicht

Knowledge Engineer

Seine Aufgabe: Entwurf von effizienten regelbasierten Systemen
Arbeitet mit den Experten zusammen und beobachtet diese, um Expertisen zu extrahieren und um diese in eine Menge von Regeln zu implementieren

Kennt sich aus mit Expertensystemen, aber nicht notwendigerweise in der Anwendungsdomäne

Kennt die Möglichkeiten der Technologien und weiß sie zu nutzen



Vergleich zur klassischen Programmierung

Eigenschaft	klass. Programmierung	Expertensystem
Gesteuert von...	Anordnung der Aussagen	Inferenzmaschine
Regelung und Daten	implizierte Integration	explizite Separation
Lösung durch...	Algorithmus	Regeln
Eingabe	als richtig angenommen	unvollständig, falsch
Unerwartete Eingabe	schwer händelbar	sehr zugänglich
Ausgabe	immer richtig	variiert mit dem Problem
Erklärung	generell sequentiell	opportunistische Regeln
Programmentwurf	strukturiert	kaum oder keine Struktur
Modifizierbarkeit	schwer	möglich
Erweiterung	in großen Sprüngen	inkrementell

Vergleich zur klassischen Programmierung

Algorithmen + Datenstrukturen = Programme

Wissen + Inferenz = wissensbasiertes System