



OTTO VON GUERICKE
UNIVERSITÄT
MAGDEBURG

INF

FAKULTÄT FÜR
INFORMATIK

Intelligente Systeme

Regelbasierte Systeme

Prof. Dr. R. Kruse **C. Braune**

{rudolf.kruse, christian.braune}@ovgu.de

Institut für Intelligente Kooperierende Systeme

Fakultät für Informatik

Otto-von-Guericke-Universität Magdeburg

Architektur Regelbasierter Systeme

Ein regelbasiertes System (rule-based system, rule engine) besteht aus Fakten, Regeln und einem Regelinterpretierer (inference engine).

Die Fakten und Regeln bilden zusammen die Wissensbasis.

Der Teil der Wissensbasis, der die Fakten enthält, wird als Faktenbasis bezeichnet.

Ein sehr einfaches regelbasiertes System ist Prolog.

Was sind Regeln?

Regeln sind formalisierte Konditionalsätze der Form

Wenn A dann B.

mit der Bedeutung:

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

A und B sind dabei Aussagen

A (Wenn-Teil): Prämisse, Antezedenz

B (Dann-Teil): Konklusion, Konsequenz

Begriffliches

Es gibt weitere Arten von Regeln, wie z.B. Produktionsregeln:

Wenn der Abstand zu klein ist, dann verringere die Geschwindigkeit.

Ist die Prämisse erfüllt, so wird die Aktion durchgeführt - "die Regel feuert"

Regel werden zum Beispiel bei der Warenkorb-Analyse in Form von Assoziationsregeln genutzt.

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

Sehr guter Kompromiss zw. Verständlichkeit der Wissensdarstellung und formalen Ansprüchen

Babylonische 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 dem menschlichem Denken nahekommen

Festlegungen

Wünschenswert ist eine möglichst einfache syntaktische Form der Regeln (Effizienz der Bearbeitung, Übersichtlichkeit)

Häufig werden zwei Bedingungen gefordert:

- 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.*

Beide Bedingungen werden verletzt, deswegen werden aus der Regel folgende vier Regeln gewonnen:

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.

Regelumformungen

In der klassischen Logik gilt:

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

Durch Distributivgesetze und de Morgan'schen Regeln lässt sich folgende Darstellung erreichen:

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:

Ersetze Regel

$$\text{if } K_1 \vee \dots \vee K_n \text{ then } D_1 \wedge \dots \wedge D_m$$

durch $n \cdot m$ Regeln

$$\text{if } K_i \text{ then } D_j, i \in \{1, \dots, n\}, j \in \{1, \dots, m\}$$

Ersetze Regel

$$\text{if } K \text{ then } L_1 \vee \dots \vee L_p$$

(wobei K Konjunktion von Literalen ist) durch p Regeln

$$\text{if } K \wedge \left(\bigwedge_{k \neq k_0} \neg L_k \right) \text{ then } L_{k_0}, k_0 \in \{1, \dots, p\}$$

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	Kartenzurückgabe	=	nein	

Äquivalenzregeln werden allerdings verletzt:

Auszahlung *genau dann, wenn* keine Voraussetzung verletzt

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

Beispiel: Geldautomat

Die Regelbasis muss um Gegenstücke zu R_1 und R_2 erweitert werden.

⇒ 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** Kartenzurückgabe = 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	≠	nicht ausreichend	
	then	Auszahlung	≠	soll erfolgen	
R''_2 :	if	Versuche	≠	überschritten	
	then	Kartenzurückgabe	=	ja	

Wissensbasis eines regelbasierten Systems

Besteht aus *Objekten* und deren Beschreibungen durch endliche Mengen diskreter *Werte*

Regeln repräsentieren Zusammenhänge zwischen *Objekten* oder *Mengen von Objekten*

Objekte und Regeln bilden *abstraktes Wissen* der Wissensbasis

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

- unmittelbare Beobachtungen
- abgeleitetes Wissen

Dieses wird 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	≠	überschritten
Betrag	≤	Maximalbetrag
Kontostand	=	nicht ausreichend

Mit Hilfe der Wissensbasis kann man zeigen, dass die Auszahlung soll nicht erfolgen darf.

Inferenz im regelbasierten System

Grundlegende Inferenzregel: *modus ponens*

$$\begin{array}{ll} \text{if } A \text{ then } B & \text{(Regel)} \\ A \text{ wahr} & \text{(Faktum)} \\ \hline B \text{ wahr} & \text{(Schlussfolgerung)} \end{array}$$

Im Geldautomatenbeispiel:

$$\begin{array}{l} \text{if Kontostand} = \text{nicht ausreichend then Auszahlung} = \text{soll nicht erfolgen} \\ \text{Kontostand} = \text{nicht ausreichend wahr} \\ \hline \text{Auszahlung} = \text{soll nicht erfolgen wahr} \end{array}$$

Wissensbasis: Regelnetzwerk

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

Regeln:

R_1 : if $A \wedge B$ then H

R_2 : if $C \vee D$ then I

R_{2a} : if C then I

R_{2b} : if D then I

R_3 : if $E \wedge F \wedge G$ then J

R_4 : if $H \vee I$ then K

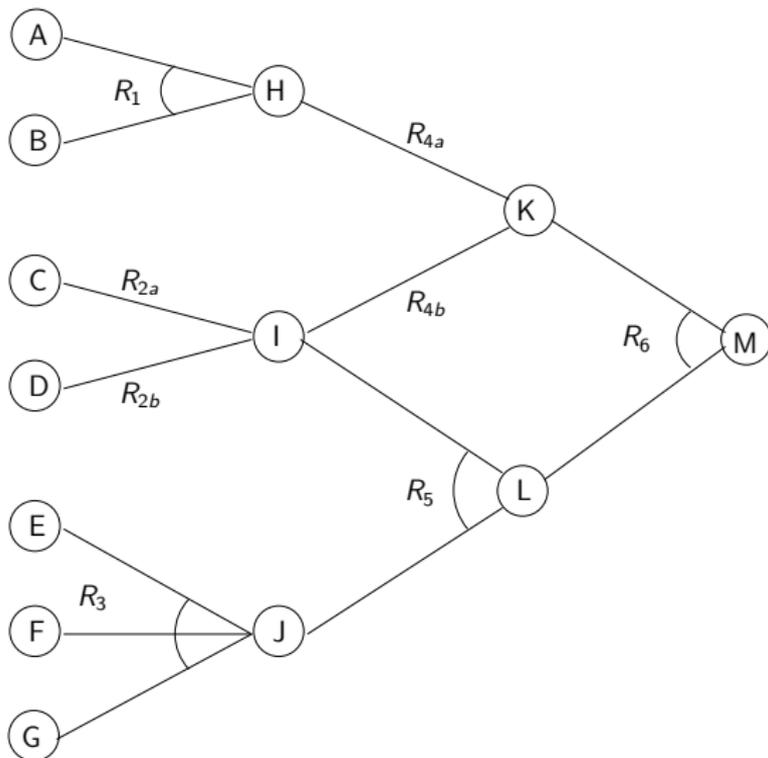
R_{4a} : if H then K

R_{4b} : if I then K

R_5 : if $I \wedge J$ then L

R_6 : if $K \wedge L$ then M

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

Das Verfahren endet, wenn keine neuen Fakten mehr generiert werden können

Datengetriebene Inferenz: 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: Regelnetzwerk

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

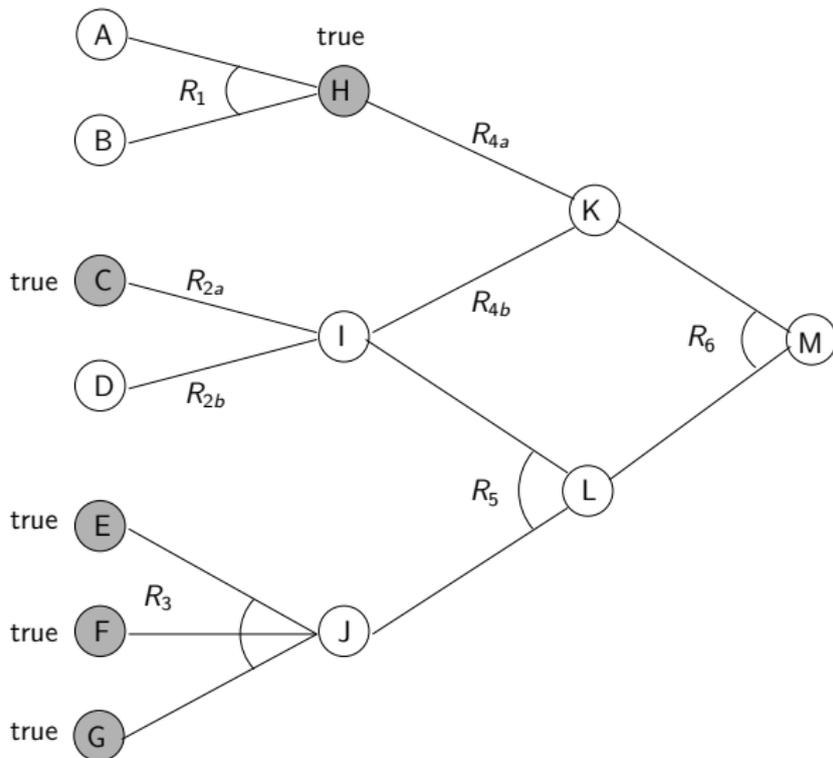
In weiterem Durchlauf feuern nun R_{4b} und R_5

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

Somit feuert nun R_6

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

Datengetriebene Inferenz: Regelnetzwerk



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: BackChain

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

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

```
1: if  $n = 0$  {
2:   return yes
3: }
4: if  $q_1 \in \mathcal{F}$  {
5:   BACKCHAIN( $[q_2, \dots, q_n]$ )
6: } else {
7:   for each Regel  $p_1 \wedge \dots \wedge p_m \rightarrow q$  aus RB mit  $q_1 = q$  {
8:     if BACKCHAIN( $[p_1, \dots, p_m, q_2, \dots, q_n]$ ) = yes {
9:       return yes
10:    }
11:  }
12: }
13: return no
```

Zielorientierte Inferenz

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

Das 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-Grammatik: **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

Regelbasen können auf zwei Arten widersprüchlich sein::

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

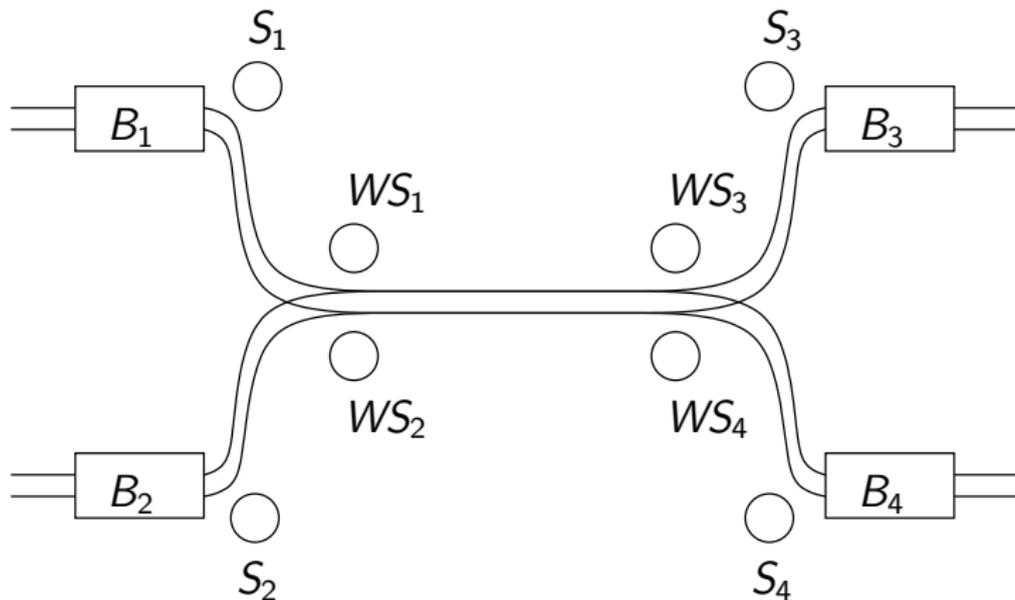
Beispiel:

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

Konsistenzüberprüfung: wichtige Warnfunktion

Dient der Verbesserung der Regelbasis

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	$S_1 = \text{grün}$	then	$S_2 = \text{rot}$
R2:	if	$S_1 = \text{grün}$	then	$S_3 = \text{rot}$
R3:	if	$S_1 = \text{grün}$	then	$S_4 = \text{rot}$
R4:	if	$S_2 = \text{grün}$	then	$S_1 = \text{rot}$
R5:	if	$S_2 = \text{grün}$	then	$S_3 = \text{rot}$
R6:	if	$S_2 = \text{grün}$	then	$S_4 = \text{rot}$
R7:	if	$S_3 = \text{grün}$	then	$S_1 = \text{rot}$
R8:	if	$S_3 = \text{grün}$	then	$S_2 = \text{rot}$
R9:	if	$S_3 = \text{grün}$	then	$S_4 = \text{rot}$
R10:	if	$S_4 = \text{grün}$	then	$S_1 = \text{rot}$
R11:	if	$S_4 = \text{grün}$	then	$S_2 = \text{rot}$
R12:	if	$S_4 = \text{grün}$	then	$S_3 = \text{rot}$

Beispiel: Signalsteuerung im Eisenbahnverkehr

Kein Zug in einem belegten Bahnhof:

R13:	if	$B_1 = \text{belegt}$	then	$WS_1 = \text{rot}$
R14:	if	$B_2 = \text{belegt}$	then	$WS_2 = \text{rot}$
R15:	if	$B_3 = \text{belegt}$	then	$WS_3 = \text{rot}$
R16:	if	$B_4 = \text{belegt}$	then	$WS_4 = \text{rot}$

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

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

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:

$$\begin{array}{l} \mathcal{F}_1: B_1 = \text{belegt} \\ B_3 = \text{belegt} \end{array}$$

Ableitbare Aussagen:

$$\begin{array}{l} \mathcal{C}_1: WS_1 = \text{rot} \quad (\text{R13}) \\ WS_3 = \text{rot} \quad (\text{R15}) \end{array}$$

Beispiel: Signalsteuerung im Eisenbahnverkehr

Zug fährt in Bahnhof B_2 ein:

$$\begin{array}{l} \mathcal{F}_2: B_1 = \text{belegt} \\ B_3 = \text{belegt} \\ B_2 = \text{belegt} \end{array}$$

Ableitbare Aussagen:

$$\begin{array}{l} \mathcal{C}_1: WS_1 = \text{rot} \quad (\text{R13}) \\ WS_3 = \text{rot} \quad (\text{R15}) \\ WS_2 = \text{rot} \quad (\text{R14}) \\ S_3 = \text{rot} \quad (\text{R17}) \\ S_4 = \text{rot} \quad (\text{R18}) \end{array}$$

Beispiel: Signalsteuerung im Eisenbahnverkehr

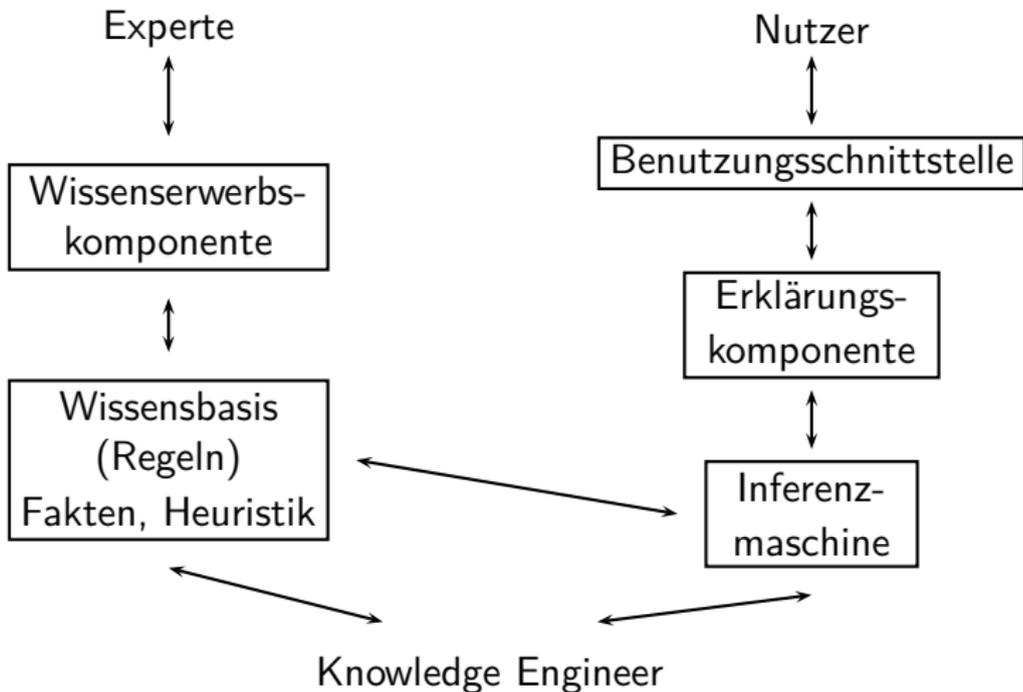
Fahrt für Zug in Bahnhof B_1 wird freigegeben, d.h. S_1 wird auf grün gestellt:

$\mathcal{F}_2:$	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)

Regelbasierte Systeme in realen Anwendungen



Beispiel: „Darlehensvergabe“

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

Beispiel: „Darlehensvergabe“

Regeln zur Entscheidungsfindung:

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

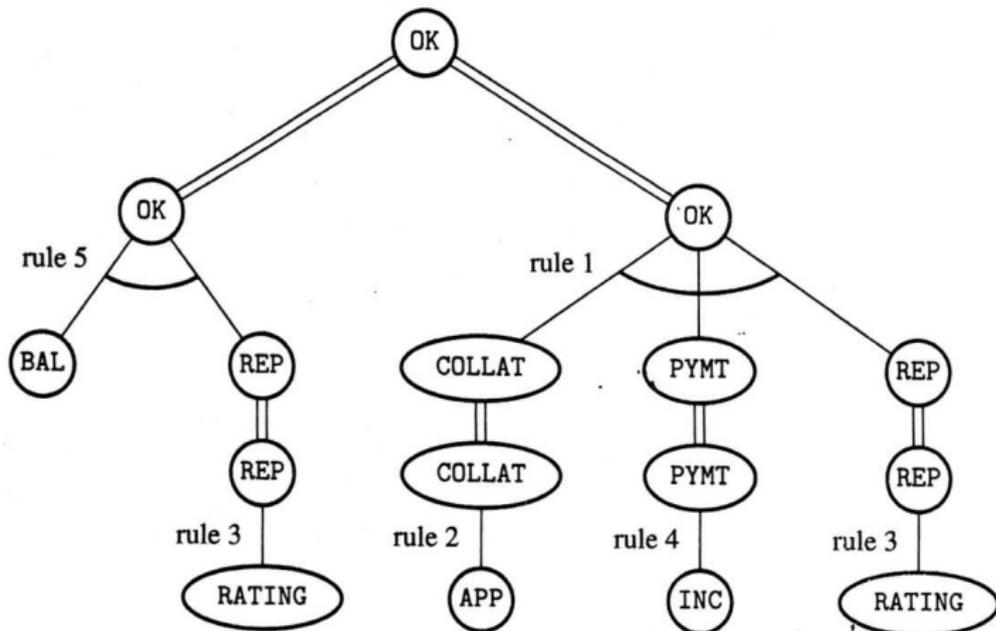
$\text{APP} \Rightarrow \text{COLLAT}$

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

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

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

Entscheidungsbaum (Und/Oder-Baum)



Modi der Verwendung

Anfragemodus:

Fakten werden aus Datenbank geholt oder Benutzer wird gefragt

Erklärungsmodus:

Das System beantwortet Fragen, wie die 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 bewilligt werden sollte* oder nicht.“

System: „Weil du mich gefragt hast?!“

XCON/R1

Beispiel für ein Konfigurationssystem

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

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.

Erweiterungen es regelbasierten Schließens

In unseren Beispielen wächst die Menge der Schlussfolgerungen monoton mit der Größe der Faktenmenge. In realen Anwendungen kann Allgemeingültigkeit der Regeln nicht immer garantiert werden. Es kann also vorkommen, dass bereits gezogene Schlussfolgerungen revidiert werden müssen. Dies führt zu nichtmonotonomem Ableitungsverhalten.

Manchmal ist das Wissen in realen Anwendungen unscharf, unvollkommen oder mit Unsicherheit behaftet, deshalb werden oft kompliziertere Regelsysteme wie Fuzzy-Regeln, Default-Regeln oder probabilistische Regeln genutzt.

Das manuelle Erstellen von großen Regelsystemen ist zeitaufwendig und teuer, deswegen werden oft maschinelle Lernverfahren für die Regelgenerierung eingesetzt.

Literatur

Methoden wissensbasierter Systeme – Grundlagen, Algorithmen,
Anwendungen

C. Beierle, G. Kern-Isberner, Springer, 2014

Grundkurs Künstliche Intelligenz – Eine praxisorientierte Einführung
W. Ertel, Springer, 2016