Intelligente Systeme

Einführung

Prof. Dr. Rudolf Kruse Georg Ruß
Christian Moewes

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

Arbeitsgruppe Computational Intelligence
Institut für Wissens- und Sprachverarbeitung
Fakultät für Informatik
Otto-von-Guericke Universität Magdeburg

Wissensbasierte Systeme

Gliederung der Vorlesung

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.

- wobei A und B Aussagen sind.
- A (Wenn-Teil): Prämisse, Antezedenz
- B (Dann-Teil): Konklusion, Konsequenz

Begriffliches

- Prämisse erfüllt: “die Regel feuert”
- gilt die Regel immer: “deterministische Regel”
- Regeln werden häufig als A$\Rightarrow$ B geschrieben

- Beispiel Assoziationsregel in der Warenkorbanalyse: *Bier* $\Rightarrow$ *Chips*
- Expertenwissen aus der Analyse steckt in dieser Regel und kann wertvolle Hinweise zur Gestaltung des Ladenaufbaus liefern

Praktischer Nutzwert

- sehr guter Kompromiß zwischen Verständlichkeit der Wissensdarstellung und formalen Ansprüchen
- vgl: Konditionalsätze in natürlicher Sprache
- babylonische Tafeln regelten das 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 menschlichen Denken sehr nahekommen

Festlegungen

- Es wird häufig Wert auf eine möglichst einfache syntaktische Form der Regeln gelegt (Effizienz der Bearbeitung, Übersichtlichkeit)
- Häufig werden zumindest die beiden folgenden Bedingungen gefordert:
  - Die Verknüpfung $\vee$ (*oder*) darf nicht in der Prämisse einer Regel auftreten
  - die Konklusion einer Regel soll nur aus *einem* Literal, also einem positiven oder negierten Atom, bestehen.
- Regeln, die diesen beiden Bedingungen nicht genügen, müssen bei Bedarf mittels logischer Äquivalenzen vereinfacht werden

Beispiel

- **Die Regel:** *Wenn es morgen regnet oder schneit, gehen wir ins Kino oder bleiben zu Hause.*
- verletzt beide Bedingungen, daher wird sie zu vier Regeln umgeformt:
  - *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 in diesem Fall *vier* vereinfachte Regeln.

Gliederung der Vorlesung

Regelumformungen

- Im Rahmen der klassischen Logik werden Regeln durch die Implikation repräsentiert:

$$A \Rightarrow B \equiv \neg A \vee B$$

- Im Allgemeinen läßt sich unter Verwendung der Distributivgesetze und der Regeln von de Morgan immer erreichen, daß:
  - die Prämisse einer Regel "**if** A **then** B" eine Disjunktion von Konjunktionen K_i von Literalen ist und
  - die Konklusion B eine Konjunktion von Disjunktionen D_j von Literalen ist.

Regelumformungen

- Durch (ggf. wiederholte) Anwendung der folgenden beiden Schritte lassen sich komplexere Regeln in syntaktisch einfachere Regeln transformieren:

  1. Ersetze die Regel

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

     durch die $n \cdot m$ Regeln

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

  2. Ersetze die Regel

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

     (wobei K eine Konjunktion von Literalen ist) durch die 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

R1:	**if**			
	Karte	$=$	gültig	**and**
	PIN	$=$	richtig	**and**
	Versuche	$=$	nicht überschritten	**and**
	Betrag	$\leq$	Maximalbetrag	**and**
	Kontostand	$=$	ausreichend	
	then			
	Auszahlung	$=$	soll erfolgen	
R2:	**if**			
	Versuche	$=$	überschritten	
	then			
	Kartenrückgabe	$=$	nein	

- Allerdings werden die Äquivalenz-Regeln verletzt: Es soll *genau dann* eine Auszahlung erfolgen, wenn keine der Voraussetzungen verletzt ist, und die Karte soll *genau dann* einbehalten werden, wenn die Anzahl der zulässigen Versuche überschritten ist.

Beispiel: Geldautomat

- Daher muß die Regelbasis um die Gegenstücke zu R1 und R2 erweitert werden:

R1': **if**

Auszahlung	=	soll erfolgen	

then

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

R2': **if**

Kartenrückgabe	=	nein	

then

Versuche	=	überschritten	

Beispiel: Geldautomat

- Dies führt zu den logisch äquivalenten folgenden Regeln:

R1": **if**

Karte	=	ungültig	**or**
PIN	=	falsch	**or**
Versuche	=	überschritten	**or**
Betrag	>	Maximalbetrag	**or**
Kontostand	=	nicht ausreichend	

then

Auszahlung	=	soll **nicht** erfolgen

R2": **if**

Versuche	=	**nicht** überschritten

then

Kartenrückgabe	=	ja

Gliederung der Vorlesung

Wissensbasis eines regelbasierten Systems

- Besteht aus *Objekten* und deren Beschreibungen mittels einer endlichen Menge diskreter *Werte*
- Regeln repräsentieren Zusammenhänge zwischen Objekten oder Mengen von Objekten
- Objekte und Regeln bilden das *abstrakte Wissen* der Wissensbasis
- Bei Anwendung auf einen konkreten Fall kommt das *fallspezifische Wissen* hinzu:
  - unmittelbare Beobachtungen
  - abgeleitetes Wissen
- Häufig wird der Begriff *Evidenz* verwendet, 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

- Ein Kunde tritt an den Automaten heran und möchte Geld abheben. Er erfüllt alle Bedingungen, allerdings wünscht er eine Auszahlung, die nicht durch seinen Kontostand gedeckt ist. Das fallspezifische Wissen sieht demnach wie folgt aus:

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

- Hieraus kann mit Hilfe der Wissensbasis abgeleitet werden, daß die Auszahlung nicht erfolgen soll.

Gliederung der Vorlesung

1. Regeln

2. Regelumformungen

3. Wissensbasis

4. **Inferenz**
   - datengetrieben
   - zielorientiert
   - Beispiel: Signalsteuerung im Eisenbahnverkehr

5. Wissensbasierte Systeme

6. MYCIN – Ein verallgemeinertes regelbasiertes System

Inferenz im regelbasierten System

- Die grundlegende Inferenzregel in einem regelbasierten System ist der *modus ponens*:

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

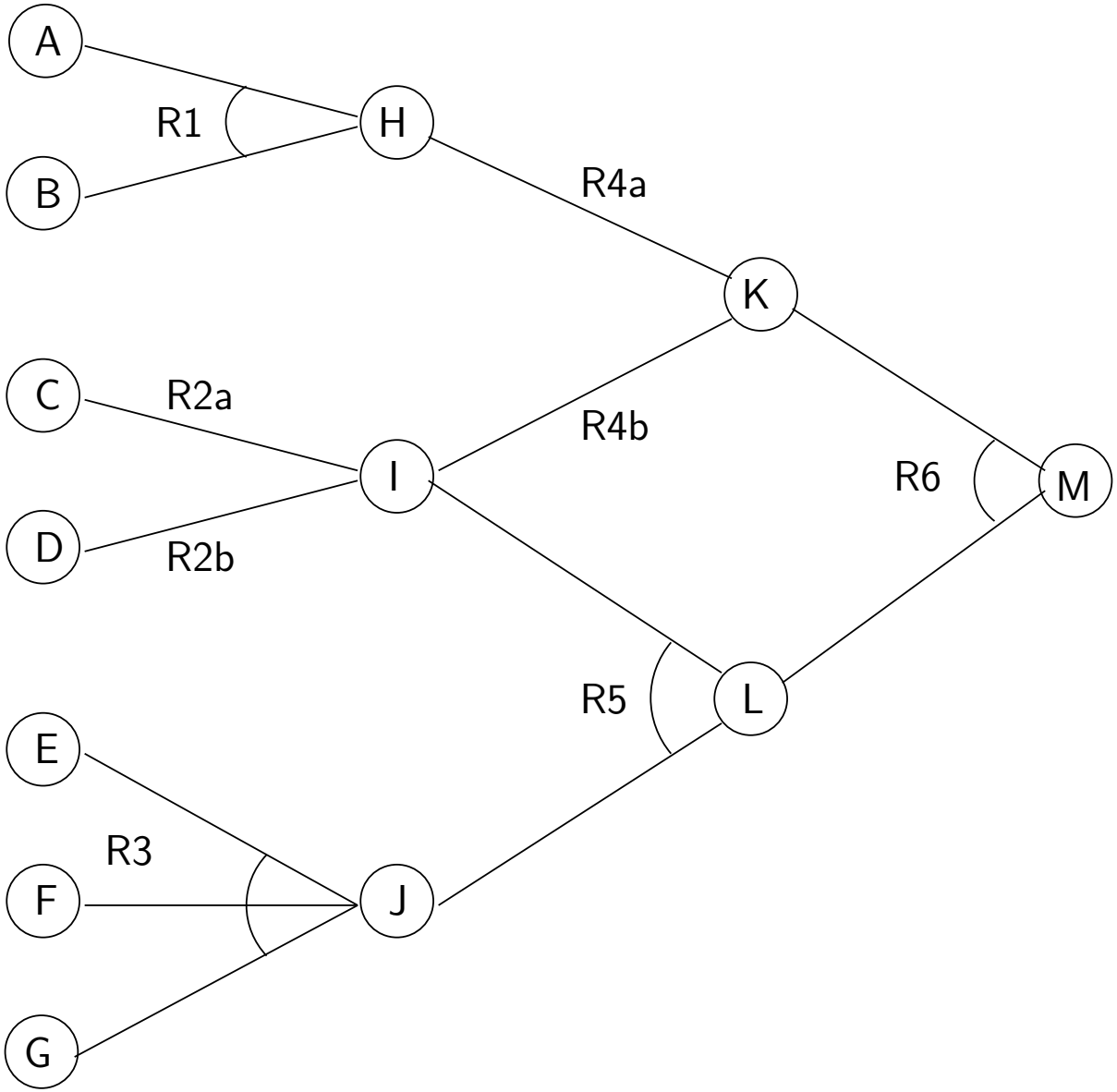
- Im Folgenden:
  - Verkettung von Regeln
  - Regelnetzwerke
  - Vorwärtsverkettung
  - Rückwärtsverkettung

Wissensbasis, Regelnetzwerk

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

R1:	if	$A \wedge B$	then	H
R2:	if	$C \wedge D$	then	I
R2a:	if	C	then	I
R2b:	if	D	then	I
R3:	if	$E \wedge F \wedge G$	then	J
R4:	if	$H \vee I$	then	K
R4a:	if	H	then	K
R4b:	if	I	then	K
R5:	if	$I \wedge J$	then	L
R6:	if	$K \wedge L$	then	M

Wissensbasis, Regelnetzwerk



Datengetriebene Inferenz

- alternative Bezeichnung: Vorwärtsverkettung
- fallspezifisches Wissen wird als Ausgangspunkt für den Inferenzprozess verwendet
- aus erfüllten Prämissen wird auf die Wahrheit der Konklusion geschlossen
- abgeleitete Fakten gehen erneut in die Wissensbasis ein

Datengetriebene Inferenz

Algorithmus: Datengetriebene Inferenz

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

Ausgabe: Die Menge der gefolgerten Fakten

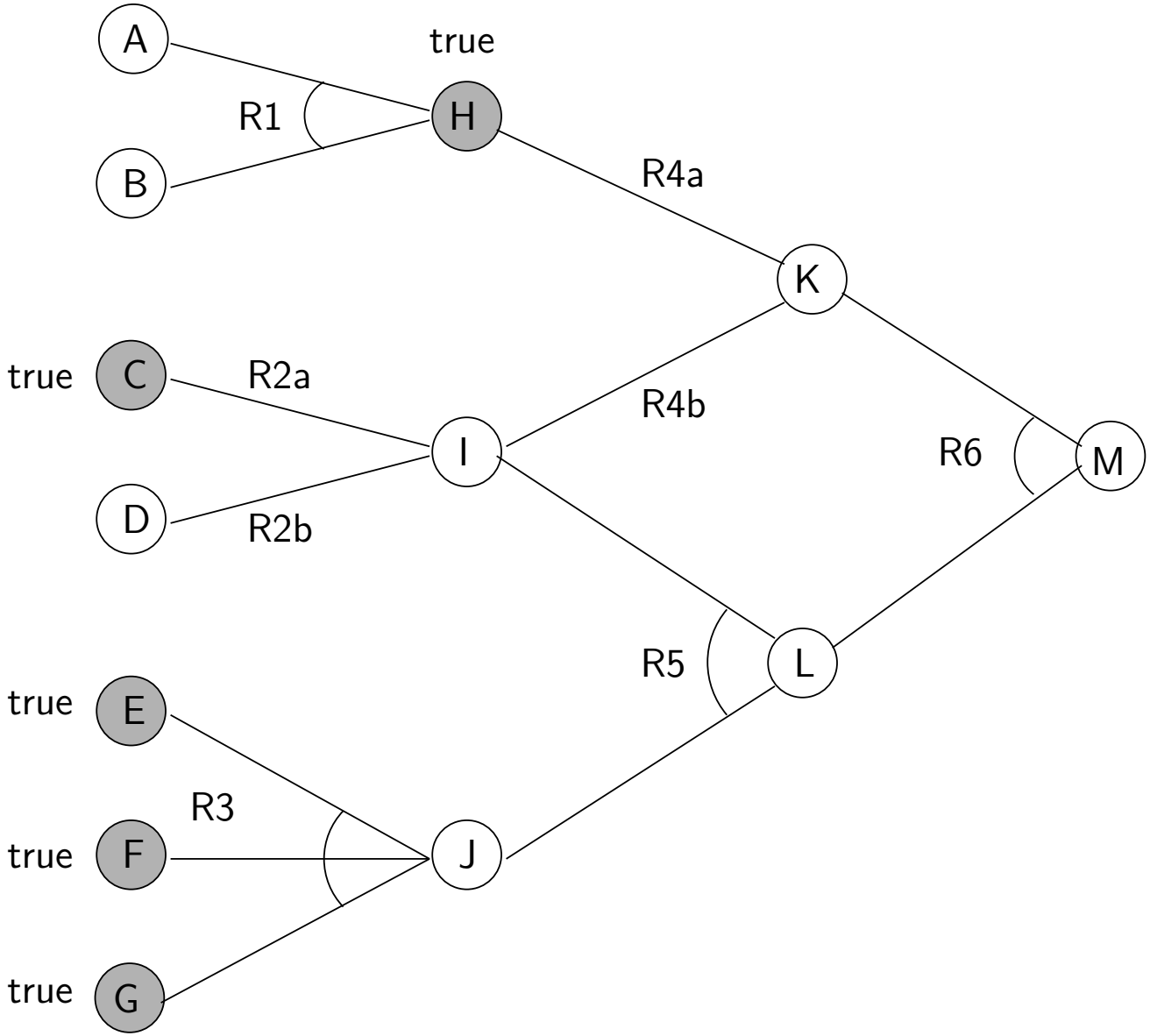
1. Sei $\mathcal{F}$ die Menge der gegebenen (evidentiellen) Fakten.
2. Für jede Regel **if** A **then** B der Regelbasis RB überprüfe:
   - Ist A erfüllt, so schließe auf B
   - $\mathcal{F} := \mathcal{F} \cup \{B\}$
3. Wiederhole Schritt 2, bis $\mathcal{F}$ nicht mehr vergrößert werden kann.

Datengetriebene Inferenz: Beispiel

Beispiel:

- gegeben seien Fakten $\mathcal{F} = \{H, C, E, F, G\}$
- damit feuern die Regeln R2a, R4a und R3
- somit vergrößert sich $\mathcal{F}$ zu $\mathcal{F} := \{H, C, E, F, G\} \cup \{I, J, K\}$
- in einem weiteren Durchlauf feuern nun auch R4b und R5, womit sich die Wissensbasis um L vergrößert
- Somit kann nun auch R6 feuern, was zur endgültigen Faktenmenge $\mathcal{F} := \{H, C, E, F, G, IJ, K\} \cup \{L, M\}$ führt

Datengetriebene Inferenz: Beispiel



Zielorientierte Inferenz

- alternative Bezeichnung: Rückwärtsverkettung
- Zielobjekt Z als Ausgangspunkt
- Durchsuchen der Regelbasis nach Regeln, die Z in der Konklusion enthalten
- Objekte der Prämissen werden zu Zwischenzielen

Zielorientierte Inferenz

Algorithmus: Zielorientierte Inferenz

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

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

1. BACKCHAIN($[q_1, \ldots, q_n]$)
2. **if** $n = 0$ then return(*yes*)
3. **if** $q_1 \in \mathcal{F}$
4. **then** BACKCHAIN($[q_2, \ldots, q_n]$)
5. **else for each** Regel $p_1 \wedge \ldots \wedge p_m \rightarrow q$ aus RB mit $q_1 = q$
6. **do if** BACKCHAIN($[p_1, \ldots, p_m, q_2, \ldots, q_n]$) = *yes*
7. **then return**(*yes*)
8. **endfor**
9. **endif**
10. **return**(*no*)

Zielorientierte Inferenz: Beispiel

- Die Wissensbasis enthalte die (zweiwertigen) Objekte O1, O2, O3 und die Regeln:
  - Regel 1: **if** O1 **then** O2
  - Regel 2: **if** O2 **then** O3
- Wir wollen den Zustand des Zielobjektes O3 in Erfahrung bringen:
  - O3 ist wahr, wenn O2 wahr ist
  - O2 ist wahr, wenn O1 wahr ist
  - Nun werden Informationen über O1 benötigt.

Problem der Widersprüchlichkeit

- Regelbasis kann zu widersprüchlichen Ableitungen führen, sobald Negation in Fakten oder der Konklusion von Regeln erlaubt wird
- praktisch 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*, so werden die Schlüsse F und $\neg F$ gezogen.
- Veranschaulichung: V - Vögel, P - Pinguine, F - Fliegen können

Problem der Widersprüchlichkeit

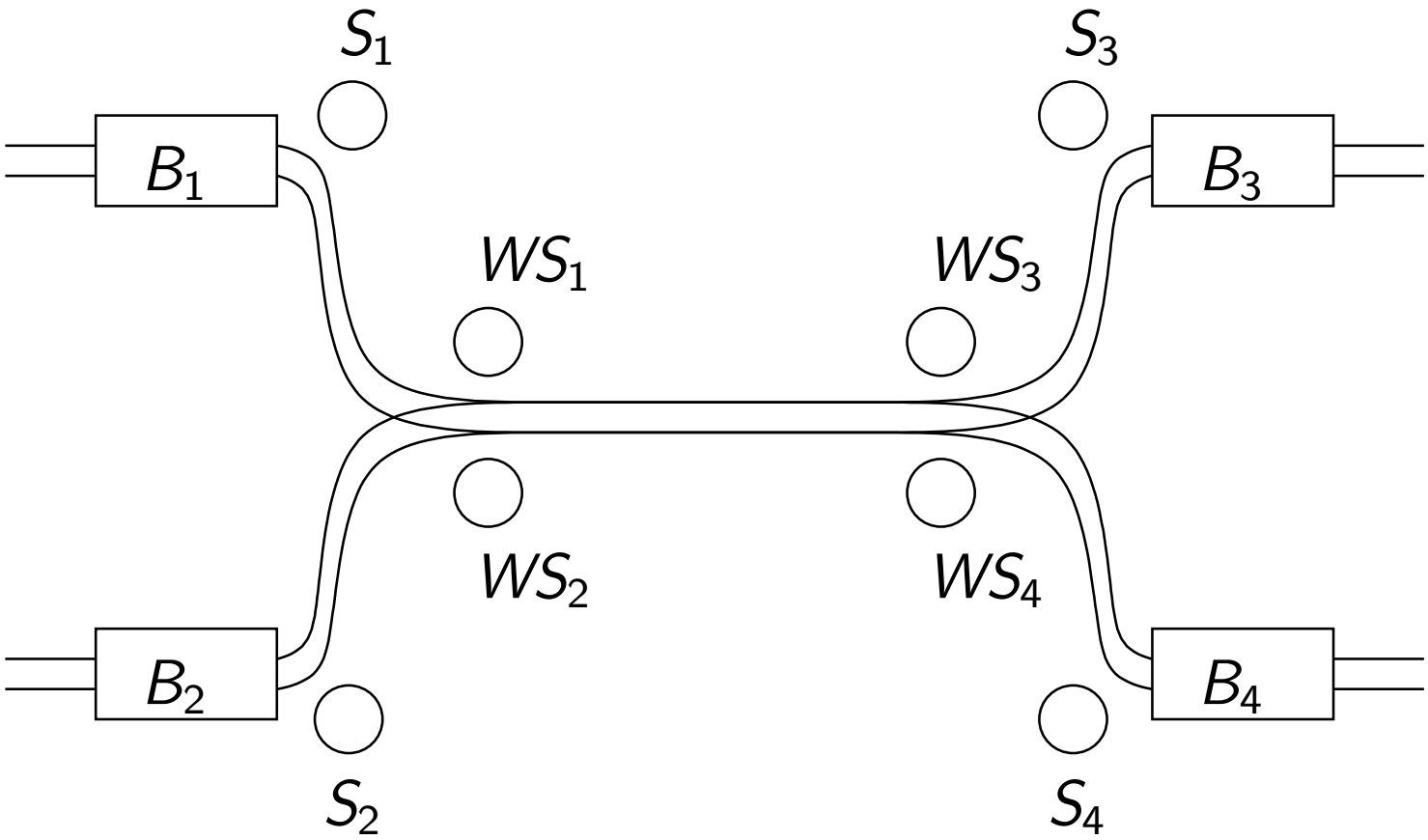
- Zwei Möglichkeiten der Widersprüchlichkeit der Regelbasis:
  - *klassisch-logisch inkonsistent:* es gibt keine Belegung der Objekte mit WErten, so daß alle Regeln erfüllt sind
  - *Ableitungen:* die Regelbasis führt zu widersprüchlichen Ableitungen
- Zweiter Fall tritt häufiger auf (und wird auch praktisch 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* als wichtige Warnfunktion und zur Verbesserung der Regelbasis

Die Erklärungskomponente

- Regeln sind im Allgemeinen recht gut verständlich
- Bei der Schlußfolgerung aus einer Regelbasis können daher die herangezogenen Regeln aufgelistet werden
- Eine Argumentationskette entsteht

- Beispiel (siehe "Datengetriebene Inferenz"):
  - Gegebene Fakten: H, C, E, F, G
  - Schlußfolgerungen:
    - I wegen Regel R2a
    - K wegen Regel R4a
    - J wegen Regel R3
    - L wegen Regel R5
    - M wegen Regel R6

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 ein regelbasiertes System gewährleistet werden

Beispiel: Signalsteuerung im Eisenbahnverkehr

- Vermeidung von Zusammenstößen auf der Strecke – immer nur höchstens eins 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:

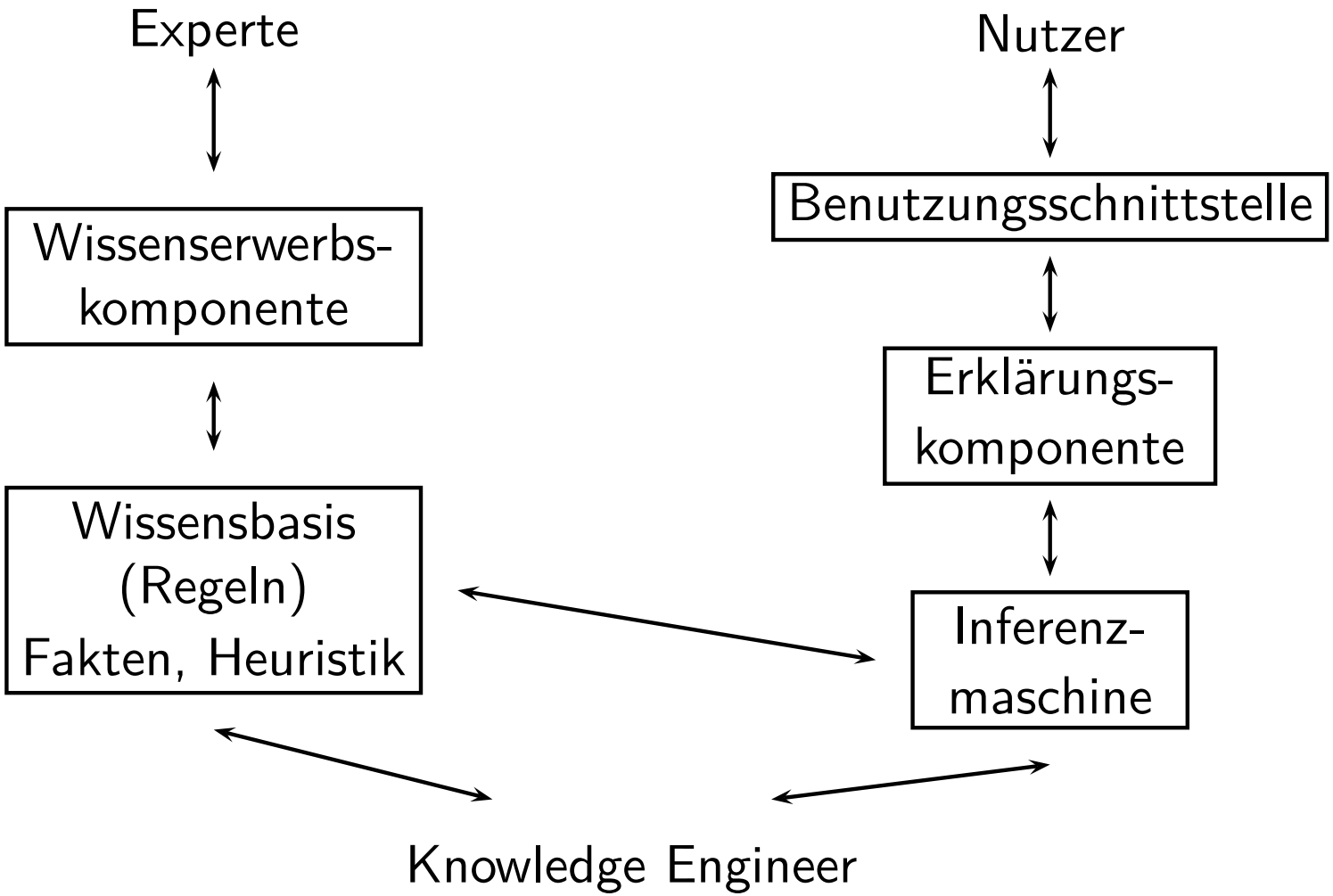
$\mathcal{C}_1$:	WS_1	$=$	rot	(R13)
	WS_3	$=$	rot	(R15)
	WS_2	$=$	rot	(R14)
	S_3	$=$	rot	(R17 und R2)
	S_4	$=$	rot	(R18 und R3)
	S_2	$=$	rot	(R1)

Monotones Schließen

- Die Menge der Schlußfolgerungen wächst *monoton* mit der Faktenmenge,
- bedingt durch die Allgemeingültigkeit der Regeln.
- Dies kann nicht immer garantiert werden, d.h. es kann durchaus vorkommen, daß bereits gezogene Schlußfolgerungen revidiert werden müssen,
- was zu *nichtmonotonem* Ableitungsverhalten führt.

Gliederung der Vorlesung

Wissensbasierte Systeme



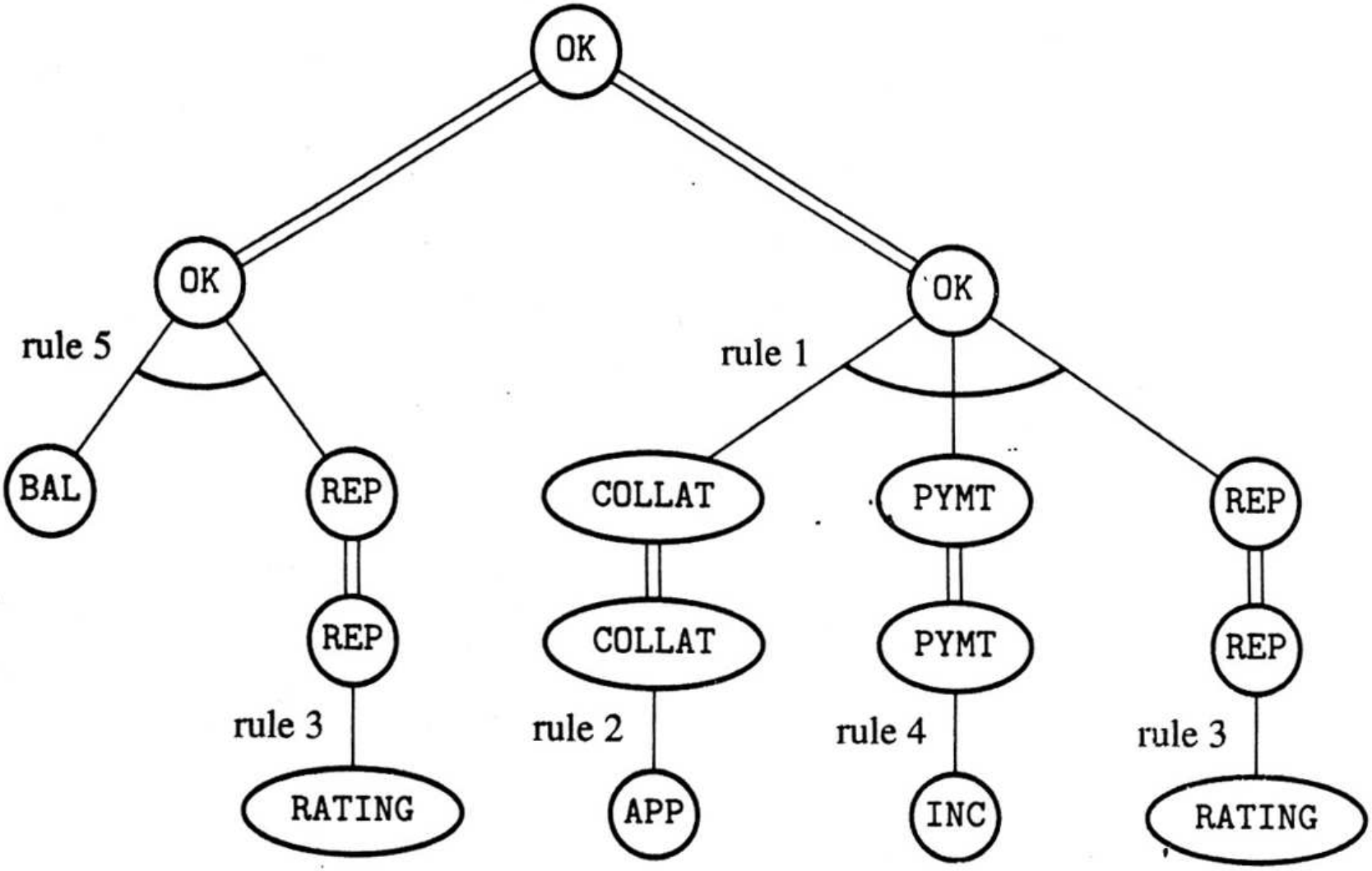
Beispiel: “Darlehensvergabe”

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

Beispiel: “Darlehensvergabe”

- Die folgenden Regeln könnten zur Entscheidungsfindung herangezogen werden:
  1. COLLAT $\wedge$ PYMT $\wedge$ REP $\Rightarrow$ OK
  2. APP $\Rightarrow$ COLLAT
  3. RATING $\Rightarrow$ REP
  4. INC $\Rightarrow$ PYMT
  5. BAL $\wedge$ REP $\Rightarrow$ OK

Entscheidungsbaum (Und/Oder-Baum)



Modi der Verwendung

- **Anfragemodus:**
  - Die Fakten werden aus einer Datenbank geholt oder der Benutzer wird gefragt.
- **Erklärungsmodus:**
  - Das System beantwortet Fragen, wie eine 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?!"

Wissensbasierte Systeme

- Eine einzelne Regel stellt (im Idealfall) eine Wissenseinheit dar.
- Eine Menge von Regeln zusammen mit einer Ausführungsstrategie stellt eine Art von Programm für die Lösung eines Problems oder einer Problemklasse dar.
- Prolog-Programm: *Regelbasiertes System*
- Regelbasierte Systeme sind historisch mit die ersten Formen von KI-Systemen. Sie wurden lange Zeit als prototypisch für *Expertensysteme* angesehen.
- Inzwischen benutzt nicht mehr jedes Expertensystem Regeln als Hauptinferenzmechanismus. Regelmodule sind jedoch immer noch wichtige Bestandteile von Expertensystemen.
- Regelsysteme gewinnen erneut Bedeutung z.B. bei der Kodifizierung von *Geschäftsprozess-Regeln*.

XCON/R1

Beispiel für ein Konfigurationssystem

- *XCON/R1* ist ein Werkzeug für die Konfiguration von *DEC Vax-Computern*. Es wird seit etwa 1980 entwickelt und löst folgende Aufgaben:
  - Identifikation fehlender Komponenten,
  - Platzierung der Komponenten bezüglich Bussen, Schnittstellen und Gehäusen
- XCON ist in *OPS5* implementiert (vorwärtsverarbeitendes regelbasiertes System).
- XCON wurde bekannt dadurch, dass es eine der ersten erfolgreichen kommerziellen Anwendungen von regelbasierten Expertensystemen darstellte und hatte dadurch großen Einfluss auf das industrielle Interesse an Expertensystemen.
- XCON wurde ständig weiterentwickelt. Es enthält heute annähernd 10.000 Regeln und löst routinemäßig Konfigurationsaufgaben 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.

Implementierung über Prolog

Beispiel

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

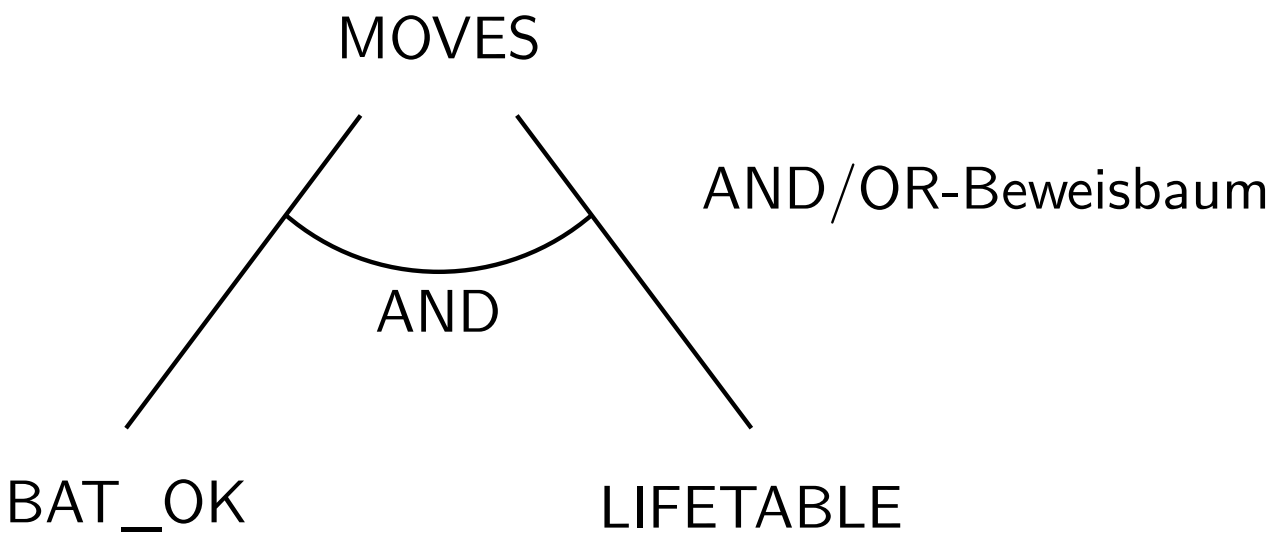
- Dieses Programm entspricht der Aussage

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

- Der Prolog Interpreter beweist diese Aussage mit “depth-first” und “backward chaining”.

Schlussfolgern mit Hornklauseln

Beispiel (Fortsetzung)



- Anmerkung:
  - Klausel 1 gegen Klausel 2 oder Klausel 3 geht nicht
  - aber Klausel 1 gegen Klausel 4 liefert zwei neue Ziele (siehe Baum)
  - diese können sofort bewiesen werden

Schlussfolgern mit Hornklauseln

Beispiel: (Blockwelt)

- Prolog:

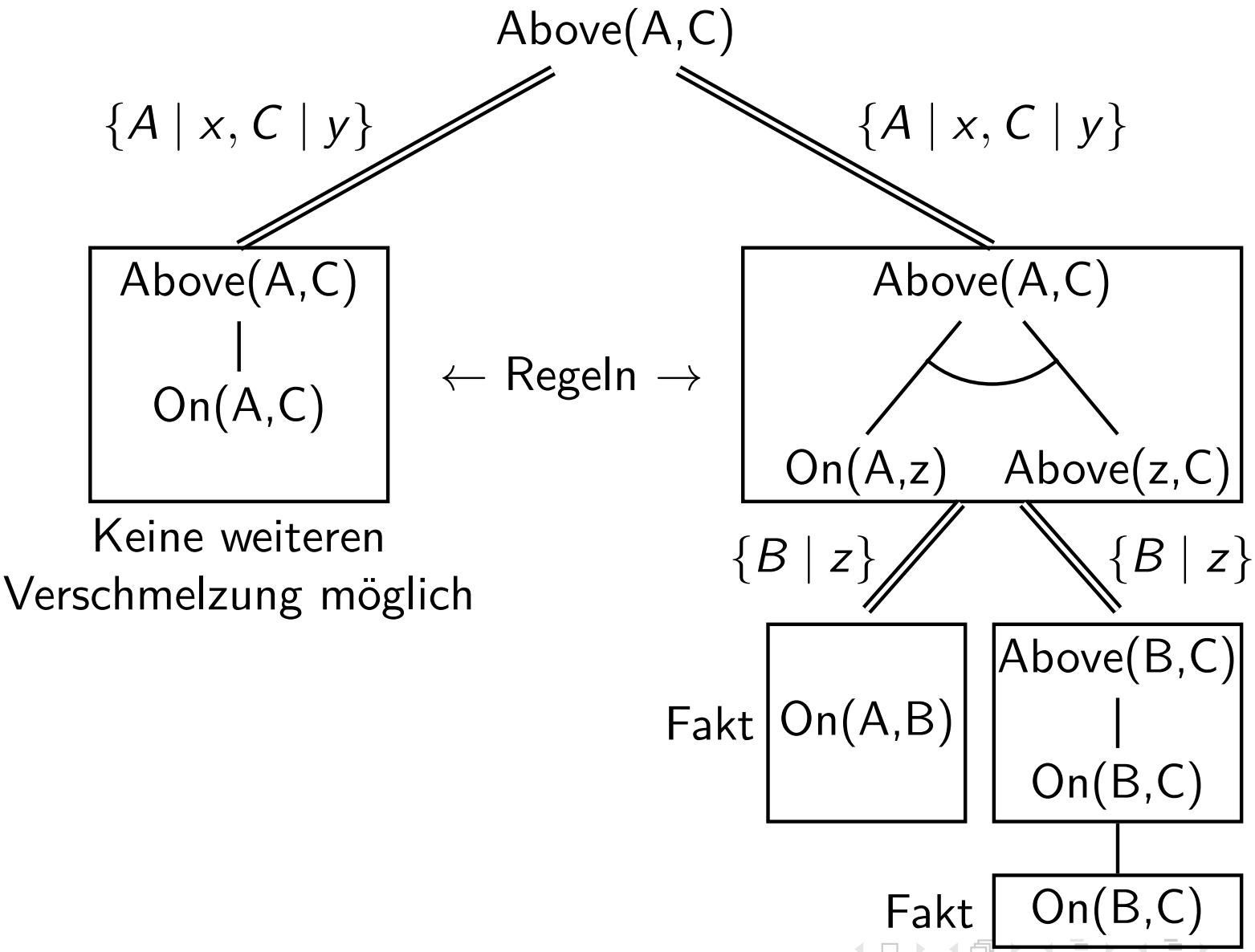
```
:- Above(A,C)
On(A,B) :-
On(B,C) :-
Above(x,y) :- On(x,y)
Above(x,y) :- On(x,z), Above(z,y)
```

- 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



Gliederung der Vorlesung

MYCIN – Ein verallgemeinertes regelbasiertes System

- Medizinisches Diagnosesystem, das auch unsicheres Wissen verarbeitet
- Wissensrepräsentation mittels Regeln
- Regeln sind mit Unsicherheitsfaktor (certainty factor) behaftet, der zwischen -1 und $+1$ liegt

Beispielregel 35:

```
PREMISE:  ($AND  (SAME CNTXT GRAM GRAMNEG)
                 (SAME CNTXT MORPH ROD)
                 (SAME CNTXT AIR ANAEROBIC))
ACTION:   (CONCLUDE CNTXT IDENTITY BACTEROIDES TALLY .6)
```

MYCIN – Ein verallgemeinertes regelbasiertes System

Bedeutung:

IF: 1) The gram stain of the organism is gramneg, and
2) The morphology of the organism is rod, and
3) The aerobicity of the organism is anaerobic,

THEN: There is suggestive evidence (0.6) that the identity of the organism is bacteroides

- Sicherheitsfaktoren (CF) drücken allgemein den *Grad des Glaubens* (degree of belief) an eine Hypothese aus.
- CF(A→B)>0: A unterstützt B
- CF(A→B)=1/-1: B ist definitiv wahr bzw. falsch, wenn A vorliegt
- CF(A→B)<0: A liefert Evidenz gegen B
- CF(A→B)=0: indifferente Haltung

MYCIN – Notationen

- $CF(A \rightarrow B) \in [-1, 1]$ bezeichnet den Sicherheitsfaktor einer Regel $A \rightarrow B$;
- $CF[B, \{A\}]$ bezeichnet den Sicherheitsfaktor der Konklusion B auf der Basis der CF der Regel $A \rightarrow B$ und der Prämisse A;
- allgemein ist $CF[B, \{A_1, \ldots, A_n\}]$ der Sicherheitsfaktor von B auf der Basis der n Regeln $A_1 \rightarrow B, \ldots, A_n \rightarrow B$ und der n Prämissen $A_1, \ldots, A_n$;
- $CF[B]$ repräsentiert schließlich den Gesamtsicherheitsfaktor von B bezüglich der Regelbasis, d.h.

$$CF[B] := CF[B, \{A | A \rightarrow B \text{ gehört zur Regelbasis}\}]$$

MYCIN – Propagationsregeln

1. Konjunktion
$$CF[A \wedge B] = min\{CF[A], CF[B]\}$$
2. Disjunktion
$$CF[A \vee B] = max\{CF[A], CF[B]\}$$
3. serielle Kombination
$$CF[B, \{A\}] = CF(A \rightarrow B) \cdot max\{0, CF[A]\}$$
4. parallele Kombination: Für $n > 1$ ist
$$CF[B, \{A_1, \ldots, A_n\}] = f(CF[B, \{A_1, \ldots, A_{n-1}\}], CF[B, \{A_n\}])$$
   - wobei die Funktion $f : [-1, 1] \times [-1, 1] \rightarrow [-1, 1]$ wie folgt definiert ist:

$$f(x, y) := \begin{cases} x + y - xy & \text{wenn } x, y > 0, \\ x + y + xy & \text{wenn } x, y < 0, \\ \frac{x+y}{1-\min\{|x|,|y|\}} & \text{sonst} \end{cases}$$

MYCIN – Funktionweise

- Der Arzt oder der Benutzer des Diagnosesystems bekommt während der Konsultation vom System Fragen gestellt.
- Nach etwa 50-60 Fragen gibt MYCIN eine hypothetische Diagnose aus, auf der dann eine passende Therapie basieren kann.
- Starke Erklärungskomponente, das *question answerung module*, das bei einer Konsultation verwendet wird
- Argumentationskette läßt sich transparent machen durch den *reasoning status checker*

MYCIN – Probleme

- MYCIN wurde niemals praktisch benutzt, auch wenn es viele Mediziner in der Diagnosegüte übertraf
- Probleme:
  - Ethische und rechtliche Schwierigkeiten – z.B. Verantwortung bei Fehldiagnose
  - Sehr stark vom behandelnden Arzt abhängig, sehr zeitaufwendig ($>$30min in einer typischen Sitzung)
  - zur damaligen Zeit (1970-1975) technisch sehr aufwendig aufgrund der fehlenden Rechenleistung und Vernetzung
- Dennoch sehr wertvoller Ansatz zur Demonstration der Mächtigkeit dieser Art der Wissensrepräsentation und Schlußfolgerung.
- Auch heute noch besteht das Problem, das Expertenwissen für ein derartiges System verfügbar zu machen.