INF FAKULTÄT FÜR INFORMATIK

Intelligente Systeme

Regelbasierte Systeme

Prof. Dr. R. Kruse C. Braune C. Doell

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

Institut für Wissens- und Sprachverarbeitung

Fakultät für Informatik

Otto-von-Guericke Universität Magdeburg

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

sehr guter Kompromiss zw. Verständlichkeit der Wissensdarstellung und formalen Ansprüchen

vgl.: Konditionalsätze in natürlicher Sprache

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 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:

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

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

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
- 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

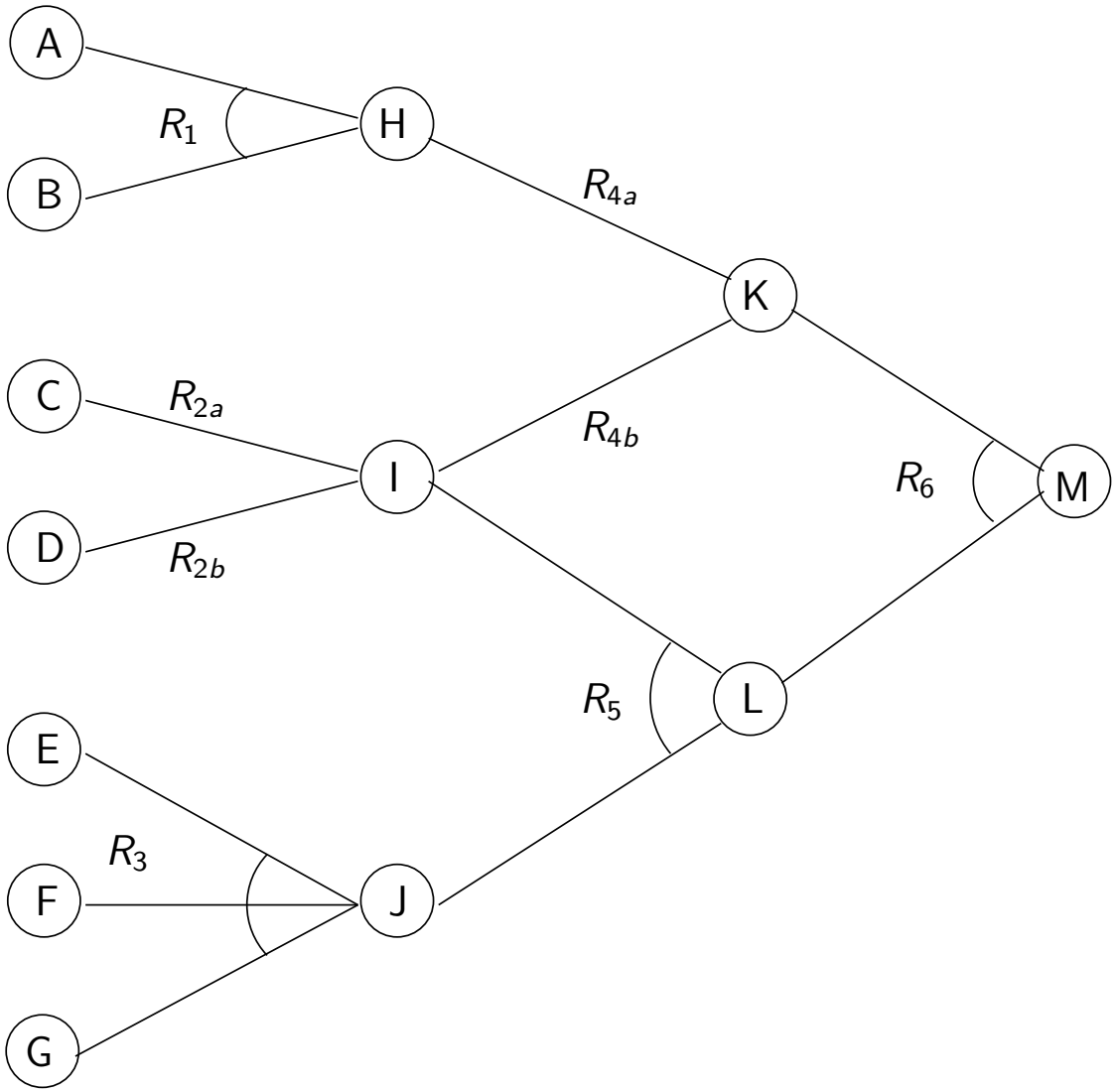
Wissensbasis, Regelnetzwerk

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

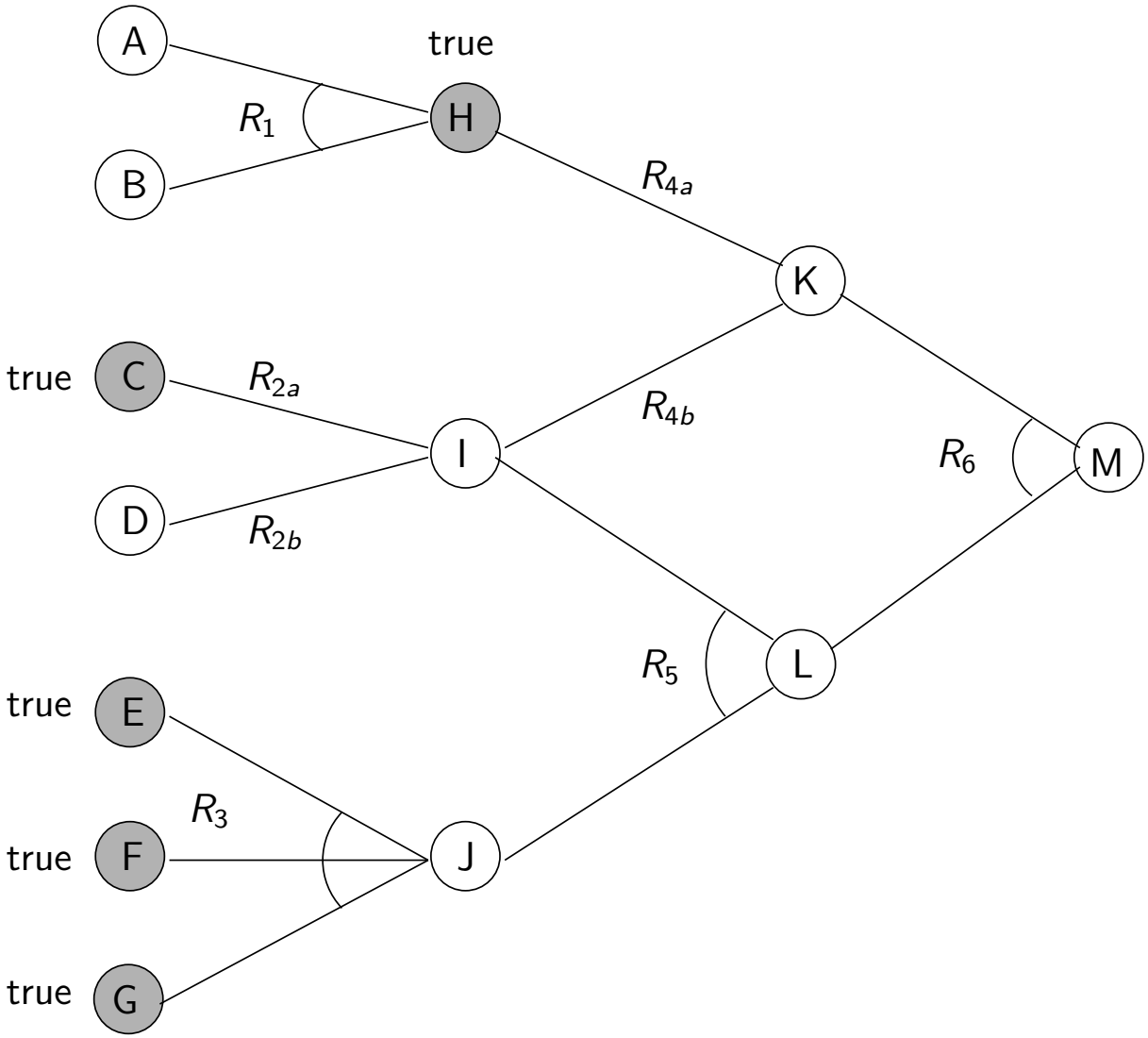
in weiterem Durchlauf feuern nun R_{4b} und R_5

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

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

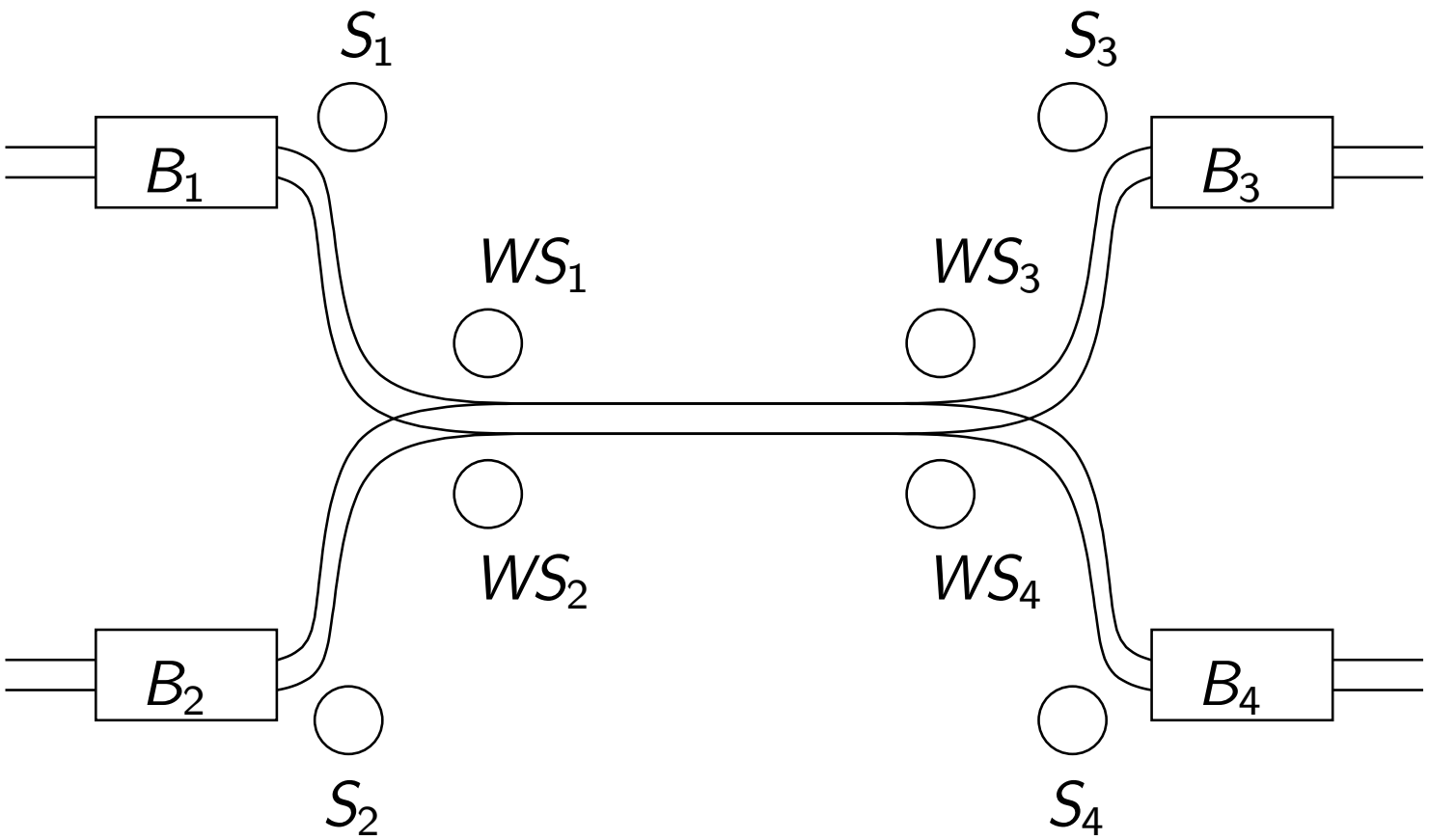
Argumentationskette entsteht

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