



OTTO VON GUERICKE  
UNIVERSITÄT  
MAGDEBURG

INF

FAKULTÄT FÜR  
INFORMATIK

# Intelligente Systeme

## Wissensbasierte Systeme

**Prof. Dr. R. Kruse    C. Moewes    G. Ruß**

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

Institut für Wissens- und Sprachverarbeitung

Fakultät für Informatik

Otto-von-Guericke Universität Magdeburg

# Übersicht

## 1. Regeln

## 2. Regelumformungen

## 3. Wissensbasis

## 4. Inferenz

## 5. Resolution

## 6. Hornklauseln

## 7. Wissensbasierte Systeme

# 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
  1. Verknüpfung  $\vee$  (*oder*) darf nicht in Prämisse auftreten
  2. 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

1. Regeln

**2. Regelumformungen**

3. Wissensbasis

4. Inferenz

5. Resolution

6. Hornklauseln

7. Wissensbasierte Systeme



# 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

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

durch  $n \cdot m$  Regeln

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

2. ersetze Regel

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

(wobei  $K$  Konjunktion von Literalen ist) durch  $p$  Regeln

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

## Beispiel: Geldautomat

$R_1$ :	<b>if</b>	Karte	=	gültig	<b>and</b>
		PIN	=	richtig	<b>and</b>
		Versuche	$\neq$	überschritten	<b>and</b>
		Betrag	$\leq$	Maximalbetrag	<b>and</b>
		Kontostand	=	ausreichend	
	<b>then</b>	Auszahlung	=	soll erfolgen	
$R_2$ :	<b>if</b>	Versuche	=	überschritten	
	<b>then</b>	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

⇒ 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:

$R_1'$	<b>if</b>	Karte	=	ungültig	<b>or</b>
		PIN	=	falsch	<b>or</b>
		Versuche	=	überschritten	<b>or</b>
		Betrag	>	Maximalbetrag	<b>or</b>
		Kontostand	≠	nicht ausreichend	
	<b>then</b>	Auszahlung	≠	soll erfolgen	
$R_2''$	<b>if</b>	Versuche	≠	überschritten	
	<b>then</b>	Kartentrückgabe	=	ja	

# Übersicht

1. Regeln

2. Regelumformungen

**3. Wissensbasis**

4. Inferenz

5. Resolution

6. Hornklauseln

7. Wissensbasierte Systeme

# 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	≠	überschritten
Betrag	≤	Maximalbetrag
Kontostand	=	nicht ausreichend

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

# Übersicht

## 1. Regeln

## 2. Regelumformungen

## 3. Wissensbasis

## 4. Inferenz

Datengetriebene Inferenz

Zielorientierte Inferenz

Beispiel: Signalsteuerung im Eisenbahnverkehr

## 5. Resolution

## 6. Hornklauseln

# Inferenz im regelbasierten System

grundlegende Inferenzregel: *modus ponens*

<b>if A then B</b>	(Regel)
A wahr	(Faktum)
<hr/>	
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:

$R_1$ : 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  $K$

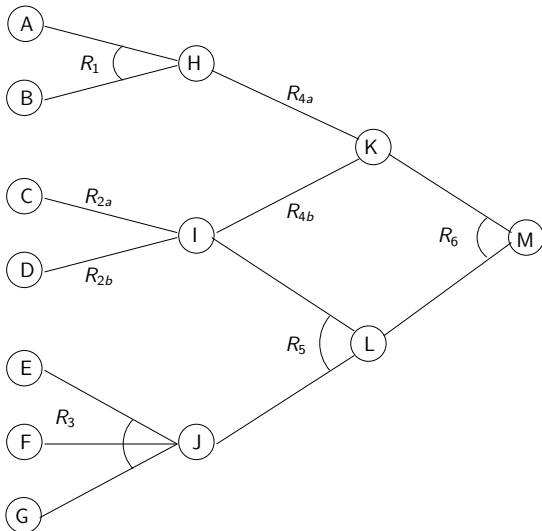
$R_{4a}$ : if  $H$  then  $K$

$R_{4b}$ : 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

# 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
-

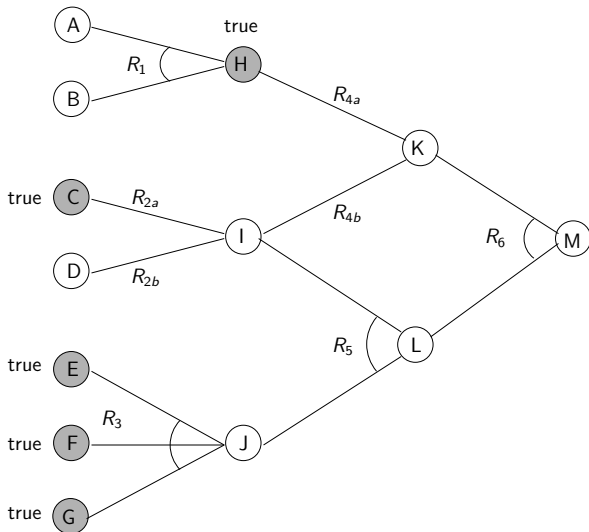
# Datengetriebene Inferenz: Beispiel

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  $R_5$
- ▶  $\mathcal{F} := \{H, C, E, F, G, I, J, K\} \cup \{L\}$
- ▶ somit feuert nun  $R_6$
- ▶ endgültige Faktenmenge  $\mathcal{F} := \{H, C, E, F, G, I, J, K, L\} \cup \{M\}$



# Datengetriebene Inferenz: 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)  $[q_1, \dots, q_n]$

**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: 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

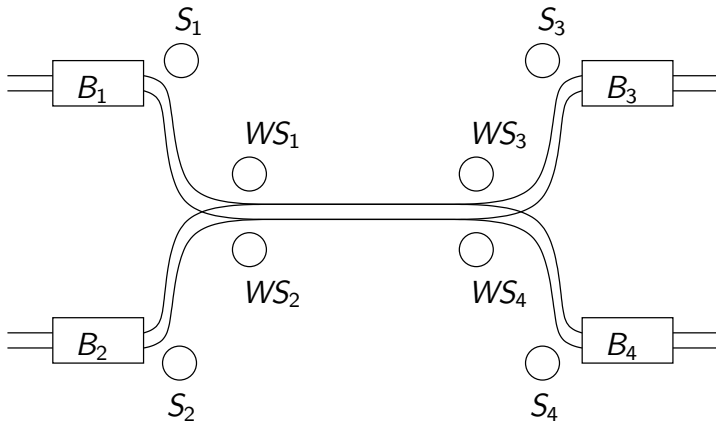
# Problem der Widersprüchlichkeit

- ▶ 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:
  - ▶  $\{\text{if } V \text{ then } F, \text{if } V \wedge P \text{ then } \neg F\}$  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

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

## Beispiel: Signalsteuerung im Eisenbahnverkehr

- ▶ kein Zug in einem belegten Bahnhof:

---

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

---

- ▶ mittlerer Teil der Strecke darf nicht durch wartende Züge blockiert werden:

---

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

---

## 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:	<b>if</b>	$WS_1$	=	grün	<b>then</b>	$WS_2$	=	rot
R22:	<b>if</b>	$WS_2$	=	grün	<b>then</b>	$WS_1$	=	rot
R23:	<b>if</b>	$WS_3$	=	grün	<b>then</b>	$WS_4$	=	rot
R24:	<b>if</b>	$WS_4$	=	grün	<b>then</b>	$WS_3$	=	rot

---

# Beispiel: Signalsteuerung im Eisenbahnverkehr

## Beispielsituation

- ▶  $B_1$  und  $B_3$  seien belegt:

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

- ▶ ableitbare Aussagen:

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

## 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)

---

# 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

1. Regeln

2. Regelumformungen

3. Wissensbasis

4. Inferenz

**5. Resolution**

6. Hornklauseln

7. Wissensbasierte Systeme

# Modus ponens und Resolution

- ▶ einfache, intuitive Regel der Inferenz

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

- ▶ durch Gültigkeit von  $A$  und  $A \rightarrow B$  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

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

# Verallgemeinerte Resolution

- ▶ Verallgemeinerung durch beliebige Anzahl von Literalen
- ▶ mit Literalen  $A_1, \dots, A_m, B, C_1, \dots, C_n$  erhält man

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

- ▶ 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 herausfand. 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

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

$$\text{Res}(1, 2) : (N)_5$$

$$\text{Res}(3, 4) : (P)_6$$

$$\text{Res}(1, 4) : (S \vee P)_7$$

- ▶  $P$  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 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 alle 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

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

$$\text{Res}(1, 6) : (C \vee \neg B)_7$$

$$\text{Res}(4, 7) : (C)_8$$

$$\text{Res}(2, 5) : (B \vee \neg C)_9$$

$$\text{Res}(3, 9) : (\neg C)_{10}$$

$$\text{Res}(8, 10) : ()$$

# Übersicht

1. Regeln

2. Regelumformungen

3. Wissensbasis

4. Inferenz

5. Resolution

**6. Hornklauseln**

7. Wissensbasierte Systeme

## Klauseln mit max. 1 positivem Literal

- ▶ Klausel in KNF enthält positive und negierte Literale
- ▶ Repräsentation als

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

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

- ▶ Transformation in

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

- ▶ 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

$$\neg A_1 \vee \dots \vee \neg A_m \vee B$$

bzw.

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

- ▶ Klausel mit einem positiven Literal ist ein **Fakt**
- ▶ in Klauseln mit negativem und einem positivem Literal heißt das positive literal **Kopf**



## Beispiel: Ski fahren

(schönes Wetter)<sub>1</sub>

(Schneefall)<sub>2</sub>

(Schneefall  $\rightarrow$  Schnee)<sub>3</sub>

(schönes Wetter  $\wedge$  Schnee  $\rightarrow$  Ski fahren)<sub>4</sub>

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

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

$MP(i_1, \dots, i_k)$  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:

(schönes Wetter)<sub>1</sub>

(Schneefall)<sub>2</sub>

(Schneefall  $\rightarrow$  Schnee)<sub>3</sub>

(schönes Wetter  $\wedge$  Schnee  $\rightarrow$  Ski fahren)<sub>4</sub>

(Ski fahren  $\rightarrow f$ )<sub>5</sub>

## Beispiel: Ski fahren

(schönes Wetter)<sub>1</sub>

(Schneefall)<sub>2</sub>

(Schneefall  $\rightarrow$  Schnee)<sub>3</sub>

(schönes Wetter  $\wedge$  Schnee  $\rightarrow$  Ski fahren)<sub>4</sub>

(Ski fahren  $\rightarrow f$ )<sub>5</sub>

Res(5, 4) : (schönes Wetter  $\wedge$  Schnee  $\rightarrow f$ )<sub>6</sub>

Res(6, 1) : (Schnee  $\rightarrow f$ )<sub>7</sub>

Res(7, 3) : (Schneefall  $\rightarrow f$ )<sub>8</sub>

Res(8, 2) : ()<sub>10</sub>

- ▶ „lineare Resolution“: große Reduktion des Suchraums
- ▶ fest Abarbeitungsreihenfolge
- ▶ SLD spielt wichtige Rolle in PROLOG

# Übersicht

1. Regeln

2. Regelumformungen

3. Wissensbasis

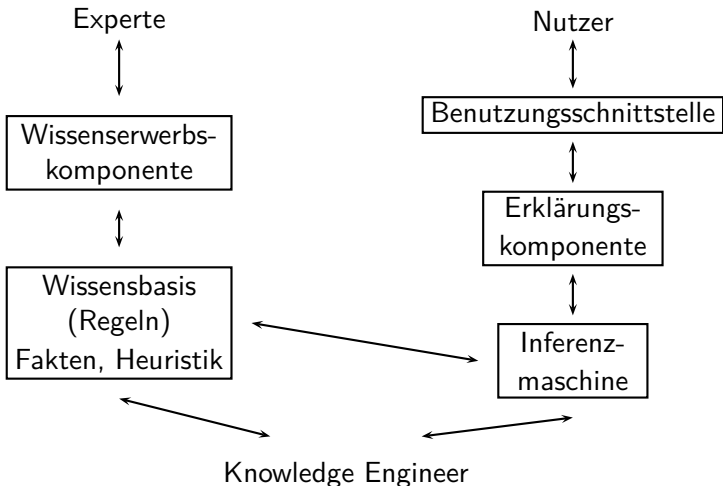
4. Inferenz

5. Resolution

6. Hornklauseln

**7. Wissensbasierte Systeme**

# Wissensbasierte Systeme



## 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.

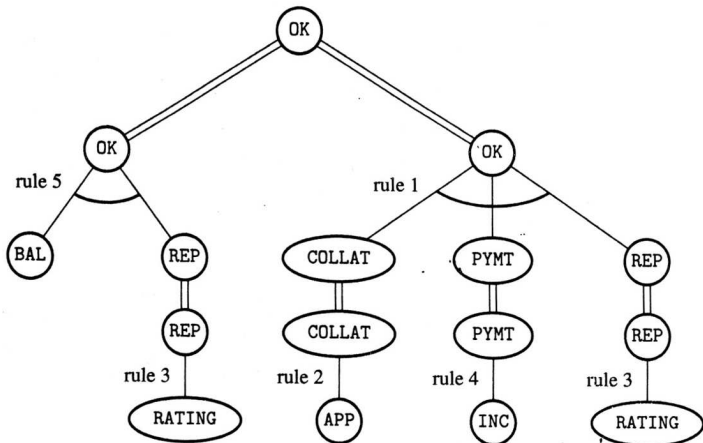
## Beispiel: „Darlehensvergabe“

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:

- ▶ 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 bewilligt werden sollte* oder nicht.“
- ▶ System: „Weil du mich gefragt hast?!“

# 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.

## Beispiel: Implementierung über Prolog

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

- ▶ Programm entspricht der Aussage:

$$(BAT\_OK \wedge LIFTABLE \wedge (BAT\_OK \wedge LIFTABLE \Rightarrow MOVES)) \Rightarrow MOVES$$

`:- MOVES`

Ziel (Anfrage)

`BAT_OK :-`

Fakten

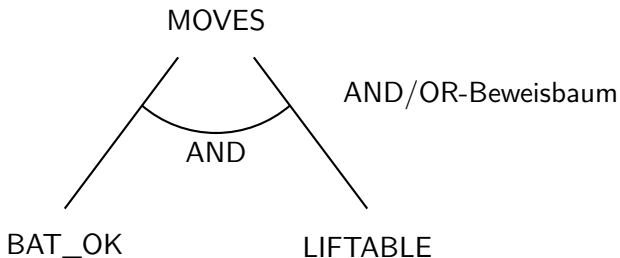
Prolog beweist

`LIFTABLE :-`

`MOVES :- BAT_OK, LIFTABLE` Regeln

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:

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

# Schlussfolgeren mit Hornklauseln

