
11. Wissensbasierte Systeme

Prof. Dr. Rudolf Kruse

University of Magdeburg

Faculty of Computer Science

Magdeburg, Germany

rudolf.kruse@cs.uni-magdeburg.de



11.1 Schlussfolgern mit Hornklauseln

Beispiel: Prolog

: -MOVES

Ziel (Anfrage)

BAT_OK :-

Fakten

LIFTABLE :-

MOVES :- BATOK, LIFTABLE

Regeln

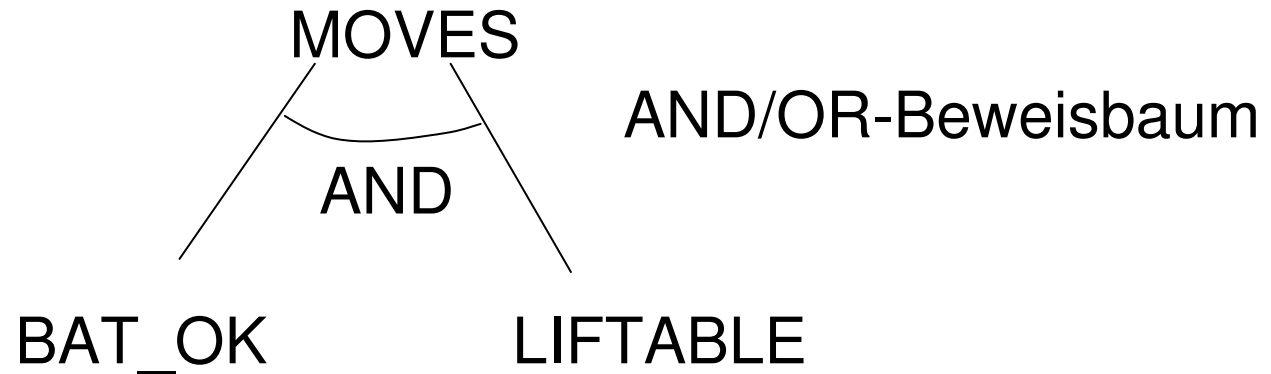
Dieses Programm entspricht der Aussage

$$(BAT_OK \wedge LIFTABLE \wedge (BAT_OK \wedge LIFTABLE \Rightarrow MOVES)) \Rightarrow MOVES$$

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), die sofort bewiesen werden können.

Schlussfolgern mit Hornklauseln

Beispiel: (Blockwelt)

Prolog:

`:- Above (A, C)`

`On (A, B) :-`

`On (B, C) :-`

`Above (x, y) :- On (x, y)`

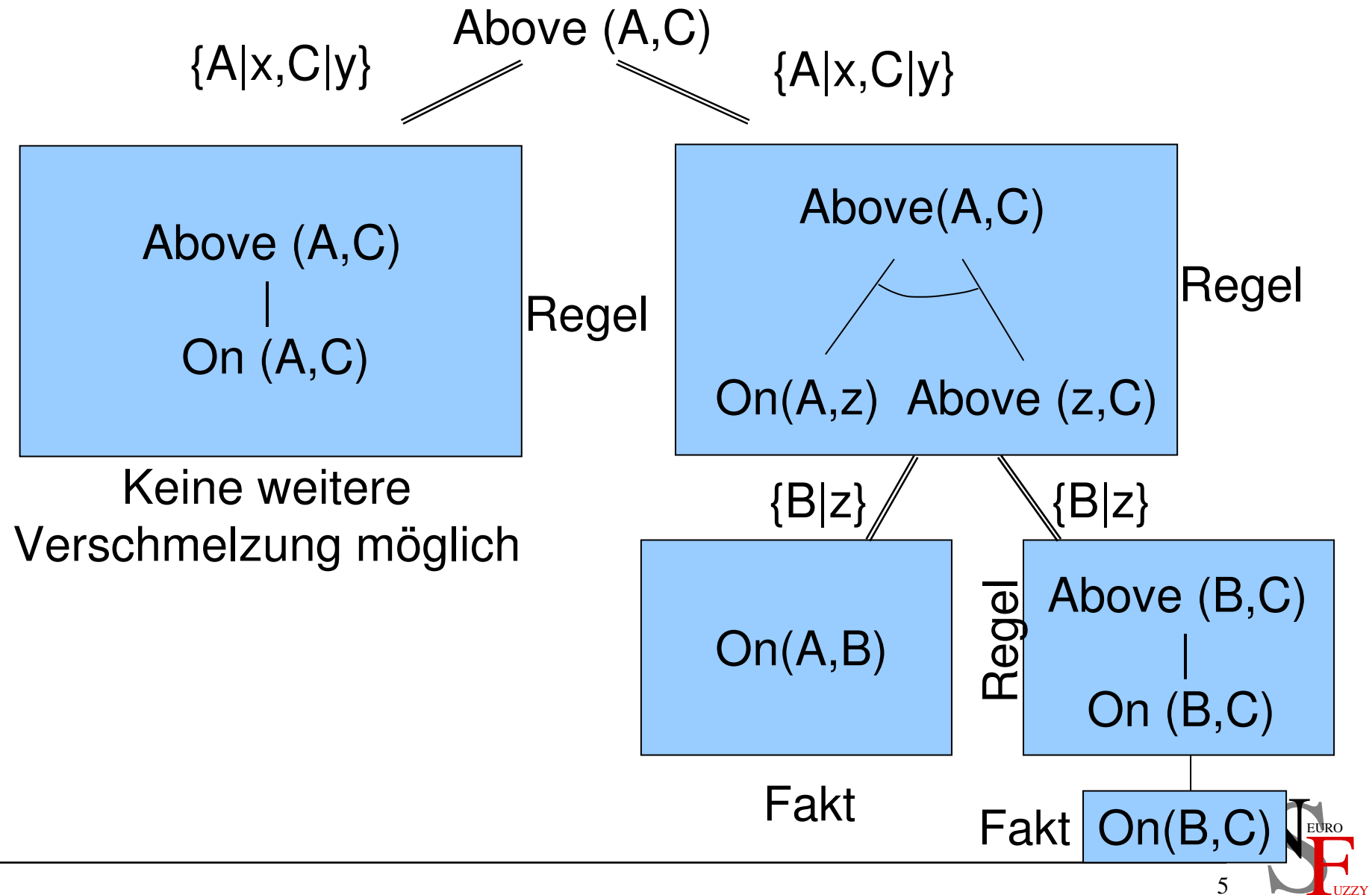
`Above (x, y) :- On (x, z), Above (z, y)`

PK1:

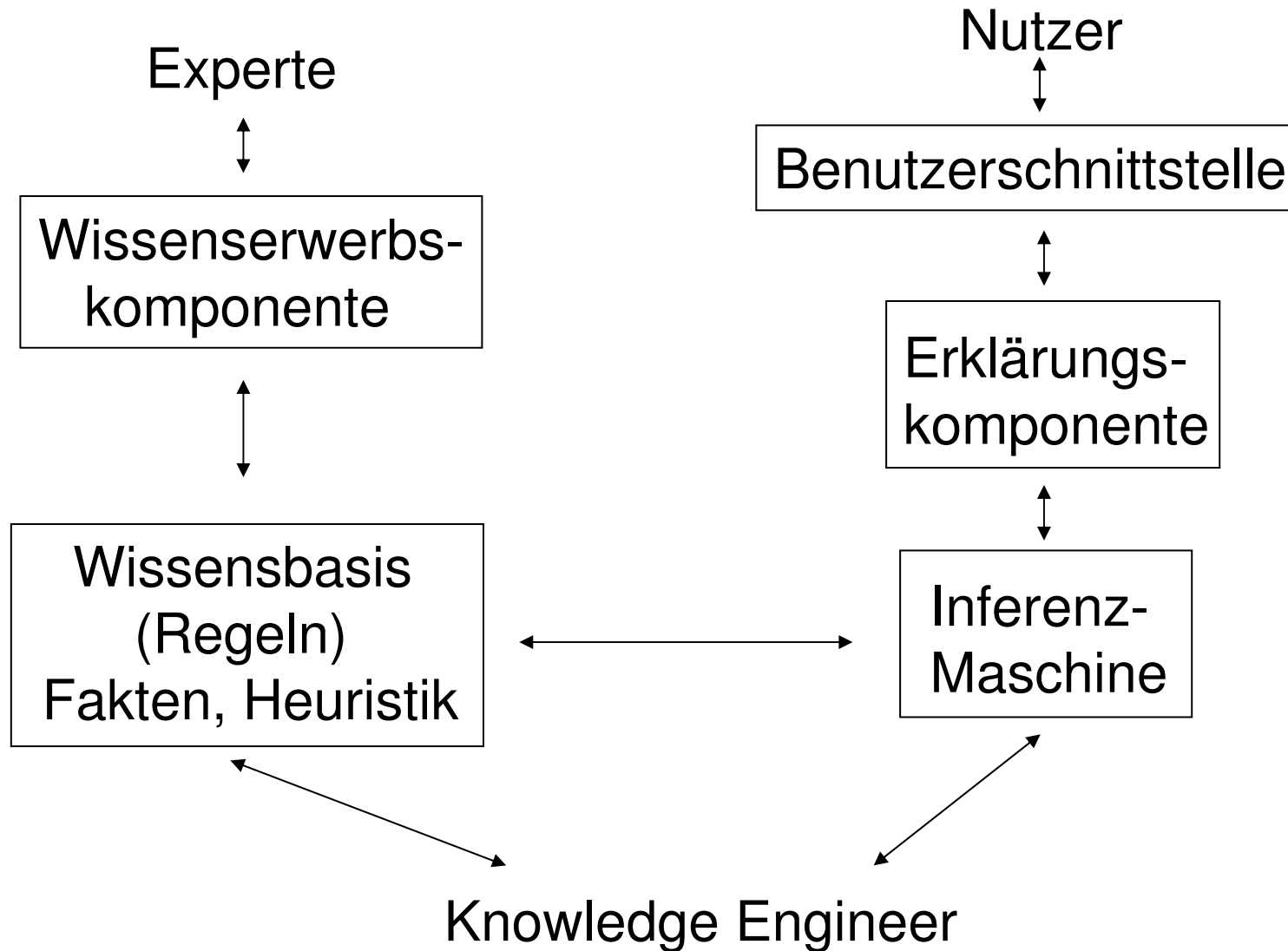
$$\forall x, y, z : On(x, y) \rightarrow Above(x, y)$$

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

Schlussfolgern mit Hornklauseln



11.2 Regelbasierte Expertensysteme



Regelbasierte Expertensysteme

Beispiel: „Darlehensvergabe“

- **OK** (The loan should be approved.)
- **COLLAT** (The collateral for the loan is satisfactory.)
- **PYMT** (The applicant is able to make the loan payments.)
- **REP** (The applicant has a good financial reputation.)
- **APP** (The appraisal on the collateral is sufficiently greater than the loan amount.)
- **RATING** (The applicant has a good credit rating.)
- **INC** (The applicant`s income exceeds his/her expenses.)
- **BAL** (The applicant has an excellent balance sheet.)

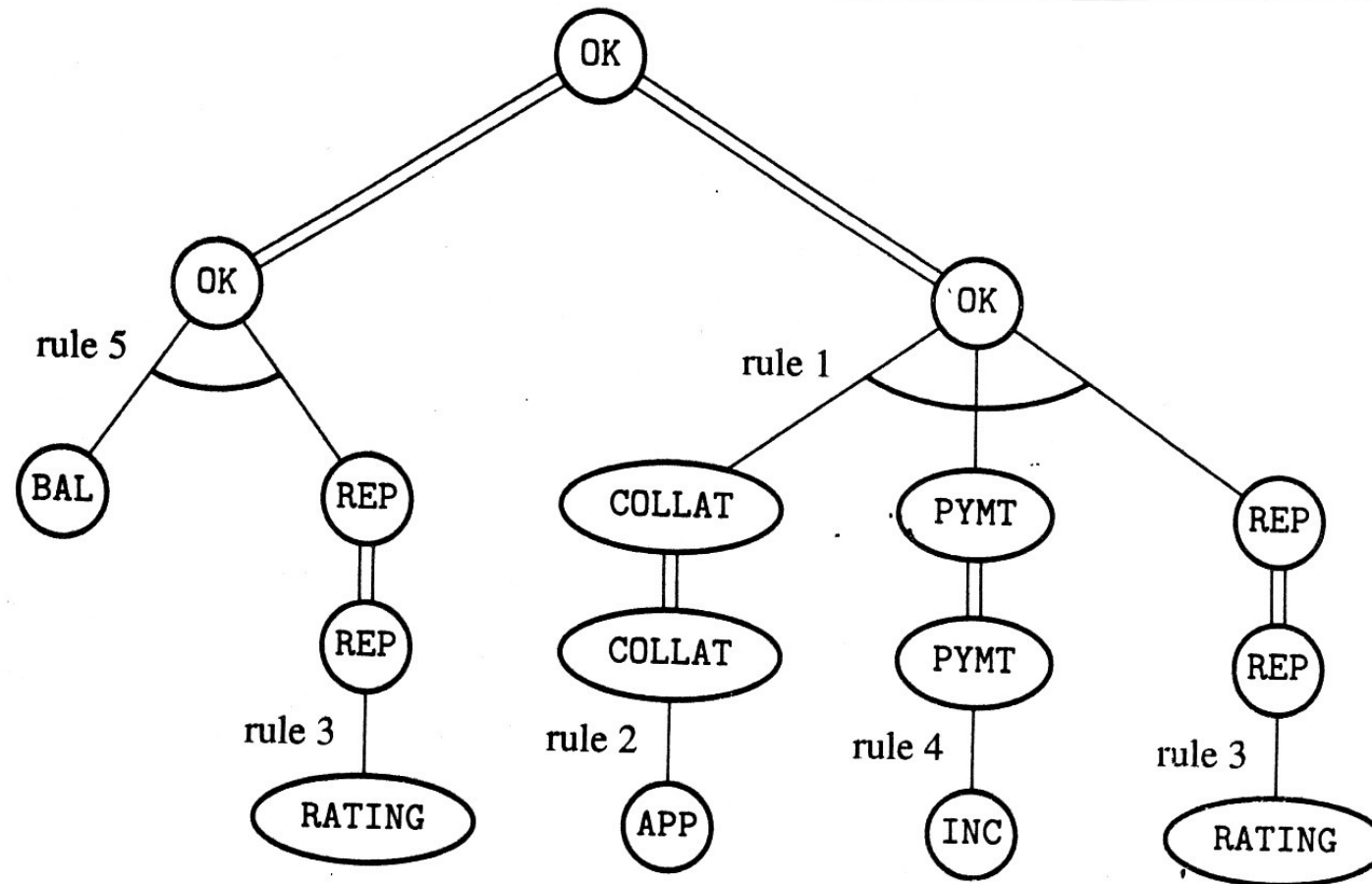
Regelbasierte Expertensysteme

Die folgenden Regeln könnten zur Entscheidungsfindung herangezogen werden:

1. $\text{COLLAT} \wedge \text{PYMT} \wedge \text{REP} \Rightarrow \text{OK}$
2. $\text{APP} \Rightarrow \text{COLLAT}$
3. $\text{RATING} \Rightarrow \text{REP}$
4. $\text{INC} \Rightarrow \text{PYMT}$
5. $\text{BAL} \wedge \text{REP} \Rightarrow \text{OK}$

Regelbasierte Expertensysteme

Entscheidungsbaum (und/oder-Baum)



Regelbasierte Expertensysteme

Modi der Verwendung:

- **Anfrage:** Die Fakten werden aus einer Datenbank geholt oder der Benutzer wird gefragt.
- **Erklärungsmodus:** Das System beantwortet Fragen, wie eine Entscheidung gefällt wurde.

Beispiele:

User: **Why do you believe that *The loan should be approved?***

System: **Because *The collateral for the loan is satisfactory, and The applicant is able to make the loan payments, and The applicant has a good financial reputation.***

...

Regelbasierte Expertensysteme

User: Why do you believe that *The applicant has good financial reputation?*

System: Because the credit-rating service said that *The applicant has a good credit rating.*

User: Why did you ask me about whether or not *The applicant's income exceeds his/her expenses?*

System: Because I was trying to establish whether or not *The applicant is able to make the loan payments.*

User: Why are you trying to establish whether or not *The applicant is able to make the loan payments?*

System: Because that is one of the criteria in establishing whether or not *The loan should be approved.*

User: Why are you trying to establish whether or not *The loan should be approved?*

System: Because you asked me to?

Regeln und Regelbasierte Systeme

Eine Regel im allgemeinen Sinn ist eine *wenn–dann–Konstruktion*, bestehend aus einer *Bedingung* und einer *Aktion*:

wenn *Bedingung* dann *Folgerung*

Die zwei Teile einer Regel können je nach Kontext unterschiedlich interpretiert werden; z.B.

- Schlussregeln: when Prämisse, dann Konklusion
- Hypothesenbildung (etwa bei Diagnose): when Evidenz, dann Hypothese
- Produktionen: when Bedingung, dann Aktion

Regeln werden häufig auch als *Produktionen* oder *Produktionsregel* bezeichnet.

Regelbasierte 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*

Regelauswertung

Vorwärtsverkettung (engl. forward chaining)

- **Erweiterung der Wissensbasis:** Sobald neue Fakten eingefügt werden, berechnet das System auch die möglichen Folgerungen.
- ***Data-driven*:** Datengesteuertes Vorgehen
- **Prämissen-orientiertes Schließen:** Die Verkettung wird durch die linken Seiten der Regeln gesteuert Produktionssystem

Regelauswertung

procedure FORWARD-CHAIN(KB, p)

if there is a sentence in KB that is a renaming of p **then return**

Add p to KB

for each ($p_1 \wedge \dots \wedge p_n \Rightarrow q$) **in** KB such that for some i , $\text{UNIFY}(p_i, p) = \theta$ succeeds **do**

 FIND-AND-INFER($KB, [p_1, \dots, p_{i-1}, p_{i+1}, \dots, p_n], q, \theta$)

end

procedure FIND-AND-INFER($KB, premises, conclusion, \theta$)

if $premises = []$ **then**

 FORWARD-CHAIN($KB, \text{SUBST}(\theta, conclusion)$)

else for each p' **in** KB such that $\text{UNIFY}(p', \text{SUBST}(\theta, \text{FIRST}(premises))) = \theta_2$ **do**

 FIND-AND-INFER($KB, \text{REST}(premises), conclusion, \text{COMPOSE}(\theta, \theta_2)$)

end

Regelauswertung

Rückwärtsverkettung (engl. backward chaining)

- **Beantwortung von Anfragen an das System. Der Bedarf an bestimmter Information löst die Anwendung von Regeln aus.**
- ***Demand-driven*: Zielgesteuertes Vorgehen**
- **Konklusions-orientiertes Schließen: Die Verkettung wird durch die rechten Seiten der Regeln gesteuert.**

Regelauswertung

function BACK-CHAIN(KB, q) **returns** a set of substitutions

BACK-CHAIN-LIST($KB, [q], \{\}$)

function BACK-CHAIN-LIST($KB, qlist, \theta$) **returns** a set of substitutions

inputs: KB , a knowledge base

$qlist$, a list of conjuncts forming a query (θ already applied)

θ , the current substitution

static: $answers$, a set of substitutions, initially empty

if $qlist$ is empty **then return** $\{\theta\}$

$q \leftarrow \text{FIRST}(qlist)$

for each q'_i **in** KB such that $\theta_i \leftarrow \text{UNIFY}(q, q'_i)$ succeeds **do**

 Add $\text{COMPOSE}(\theta, \theta_i)$ to $answers$

end

for each sentence $(p_1 \wedge \dots \wedge p_n \Rightarrow q'_i)$ **in** KB such that $\theta_i \leftarrow \text{UNIFY}(q, q'_i)$ succeeds **do**

$answers \leftarrow \text{BACK-CHAIN-LIST}(KB, \text{SUBST}(\theta_i, [p_1 \dots p_n]), \text{COMPOSE}(\theta, \theta_i)) \cup answers$

end

return the union of $\text{BACK-CHAIN-LIST}(KB, \text{REST}(qlist), \theta)$ for each $\theta \in answers$

Beispiel

”The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.”

***American(x) $\wedge$ Weapon(y) $\wedge$ Nation(z) $\wedge$ Hostile(z) $\wedge$ Sells(x, z, y)
 $\Rightarrow$ Criminal(x)***

Owns(Nono, x) $\wedge$ Missile(x) $\Rightarrow$ Sells(West, Nono, x)

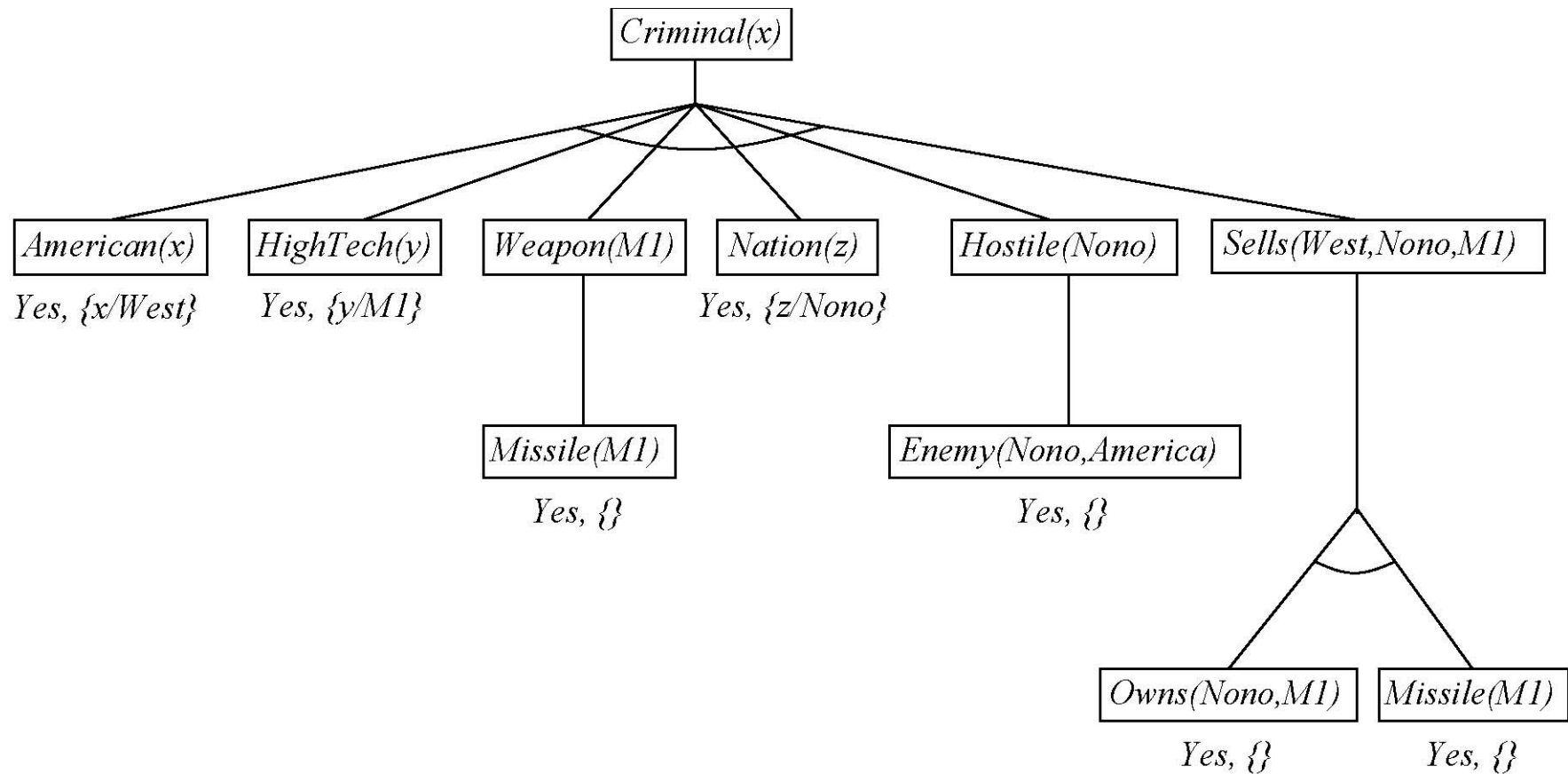
Missile(x) $\Rightarrow$ Weapon(x)

Enemy(x, America) $\Rightarrow$ Hostile(x)

***American(West), Nation(Nono), Enemy(Nono, America), Owns(Nono, M 1),
Missile(M 1), Nation(America).***

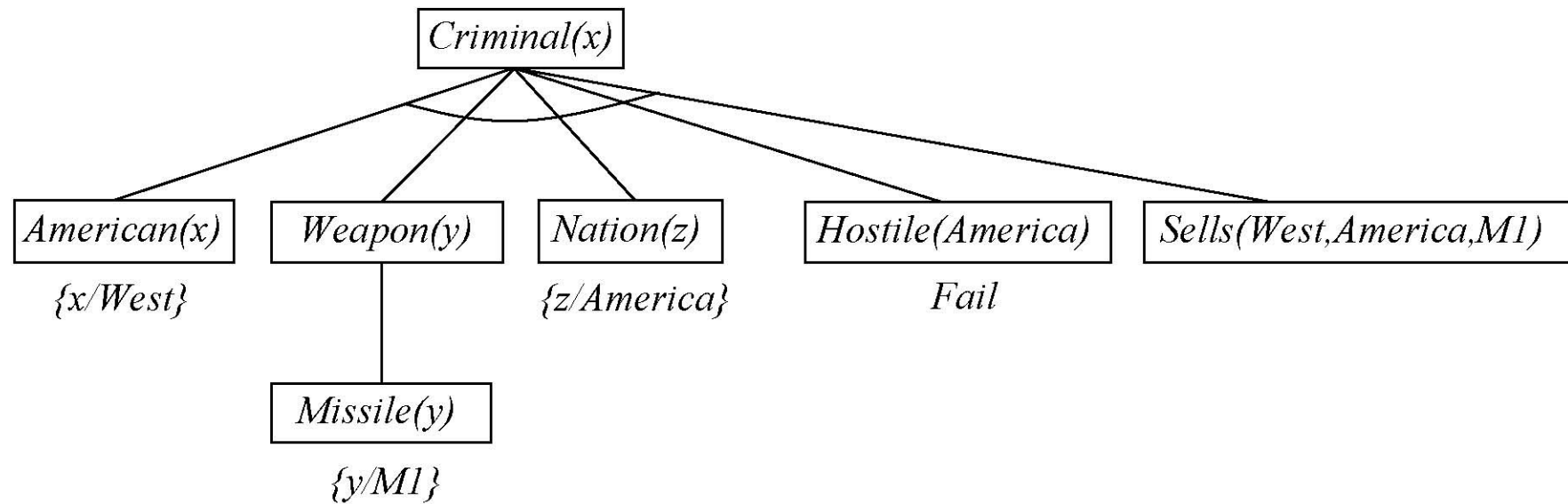
Beispiel

Ist Colonel West ein Krimineller?



Beispiel

Fortsetzung



Aufbau eines Regelsystems

Komponenten

Datenbasis

Menge von strukturierten Datenobjekten:

Arbeitsspeicher, “Kurzzeitgedächtnis” (short-term memory buffer)

Der Arbeitsspeicher enthält eine Menge von Grundaussagen, die den momentanen Zustand des modellierten Weltausschnitts beschreiben.

Regelbasis

Menge von Regeln: Die Anwendung einer Regel verändert im allgemeinen den Inhalt des Arbeitsspeichers.

Regel-Interpreter

“Inferenzmaschine”, die die Abläufe im System steuert.

Regelinterpretation

Grundschemata der Vorwärtsverkettung

Es werden Regeln gesucht, deren Bedingungsteil erfüllt ist. Aus der Menge dieser Regeln werden eine oder mehrere ausgewählt und angewandt. Dies führt typischerweise zur Veränderung der Datenbasis und der Möglichkeit, dass neue Bedingungsteile erfüllt werden.

Dieser Zyklus wird so lange wie möglich wiederholt.

Der Prozess terminiert, falls

- *es keine Regel mehr gibt, deren Bedingungsteil erfüllt ist;*
- *ein Lösungskriterium erfüllt ist (vgl. Zielzustand bei Suchaufgaben);*
- *ein Abbruchkriterium erfüllt ist (z.B. das Erreichen einer Maximum-Schranke für die Anzahl der Schritte).*

Regelinterpretation

Folgende Aufgaben sind zu lösen.

- *Identifikation derjenigen Regeln, deren Bedingungsteile erfüllt sind und die damit für die Anwendung aktiviert werden können:*

Instantiierung bzw. Matching.

- *Auswahl einer Regel aus der Menge der aktivierbaren Regeln für die Ausführung.*

Konfliktlösung notwendig

Alternative: Alle aktivierbaren Regeln werden angewandt.

Instantiierung von Regeln

Ein Datenobjekt erfüllt eine Teilbedingung, wenn seine Attributwerte die entsprechenden einschränkenden Prädikate im Muster der Teilbedingung erfüllen (matching).

- *Im Muster nicht eingeschränkte Attribute können beliebige Werte haben.*
- *Variable im Muster werden an die entsprechenden aktuellen Werte gebunden.*
- *Eine Variable, die mehrfach im Muster vorkommt, muss konsistent gebunden werden (oder: das erste Vorkommen bindet, folgende Vorkommen schränken ein wie eine Konstante).*

Ein Datenobjekt, das eine Teilbedingung erfüllt, wird an die Teilbedingung gebunden. Eine explizite Bindung an eine Objektvariable wird durch eine Teilbedingung erreicht (genaue Syntax ist implementierungsabhängig). Über die Variable kann später auf das Datenobjekte zugegriffen werden.

Durch das Matching wird an jede (positive) Teilbedingung ein Datenobjekt gebunden: die Regel wird instantiiert. Bindungen an Variable, die sich aus dem Matching ergeben haben, sind lokal für die Instanz der Regel.

Zu einer Regel kann es zu einem Zeitpunkt (d.h. Zustand des Arbeitsspeichers) gleichzeitig mehrere verschiedene Instanzen geben.

Aktivierung von Regeln

Eine instantiierte Regel kann aktiviert werden: feuern.

Aktivierung bedeutet, dass der Aktionsteil der Regel ausgeführt wird.

Wenn mehrere Regeln instantiiert sind, d.h. aktiviert werden können, muss eine Auswahl getroffen werden, da die Regeln in Konflikt stehen können.

Konfliktauflösung notwendig

Strategien zur Konfliktauflösung, d.h. zur Auswahl der als nächste anzuwendenden Regel, auf der Grundlage von

- *Regelprioritäten*
- *Beziehungen zwischen Regelinstantiierungen*

Aktivierung von Regeln

Konfliktauflösung aufgrund von Regelprioritäten:

- *Totale Ordnung auf Regeln, z.B. Reihenfolge der Eingabe.*
- *Partielle Ordnung: typischerweise Ordnung zwischen Gruppen von Regeln; nützlich für strukturierte Regelmengen.*

Konfliktauflösung aufgrund von Beziehungen zwischen Regelinstantiierungen: insbesondere Spezialisierungsrelationen. Eine Regelinstanz I ist spezieller als eine Instanz K , wenn sie z.B.

- *mindestens so viele Prämissen hat wie K und die Prämissen mindestens so einschränkend sind wie die von K .*
- *mehr Datenobjekte betrifft als K .*

Optimierung von Regelanwendungen

Bei naiver Implementierung der Regelinstantiierung müssen in jedem Schleifendurchlauf alle Eingaben mit den Bedingungsteilen aller Regeln verglichen werden.

Problem bei großen Regelmengen.

Eine Effizienzsteigerung kann erreicht werden durch

Einschränkung der zu überprüfenden Datenelemente: Bei jeder Eingabe bzw. Regelanwendung wird jeweils nur ein kleiner Teil des Arbeitsspeichers verändert; nur dieser muss betrachtet werden.

Einschränkung der zu überprüfenden Bedingungen: Die Bedingungsteile der Regeln sind im allgemeinen nicht disjunkt; gleichartige Tests können zusammengefasst und damit der Testaufwand reduziert werden.

Optimierung von Regelanwendungen

Auf der Grundlage dieser Überlegungen liefert der Rete-Algorithmus eine effiziente Implementierung des Matching mit Teilbedingungen und der Instantiierung von Regeln (Rete: lat. Netz).

Der Rete-Algorithmus wird in vielen regelbasierten Systemen verwendet, in denen es auf Effizienz ankommt.

Rete-Algorithmus am Beispiel:

$A(x) \wedge B(x) \wedge C(y) \Rightarrow add_D(x)$

$A(x) \wedge B(y) \wedge D(x) \Rightarrow add_E(x)$

$A(x) \wedge B(x) \wedge E(x) \Rightarrow delete_A(x)$

■ *Arbeitsspeicher: A(1), A(2), B(2), B(3), B(4), C (5)*

Optimierung von Regelanwendungen

Auf der Grundlage dieser Überlegungen liefert der Rete-Algorithmus eine effiziente Implementierung des Matching mit Teilbedingungen und der Instantiierung von Regeln (Rete: lat. Netz).

Der Rete-Algorithmus wird in vielen regelbasierten Systemen verwendet, in denen es auf Effizienz ankommt.

Rete-Algorithmus am Beispiel:

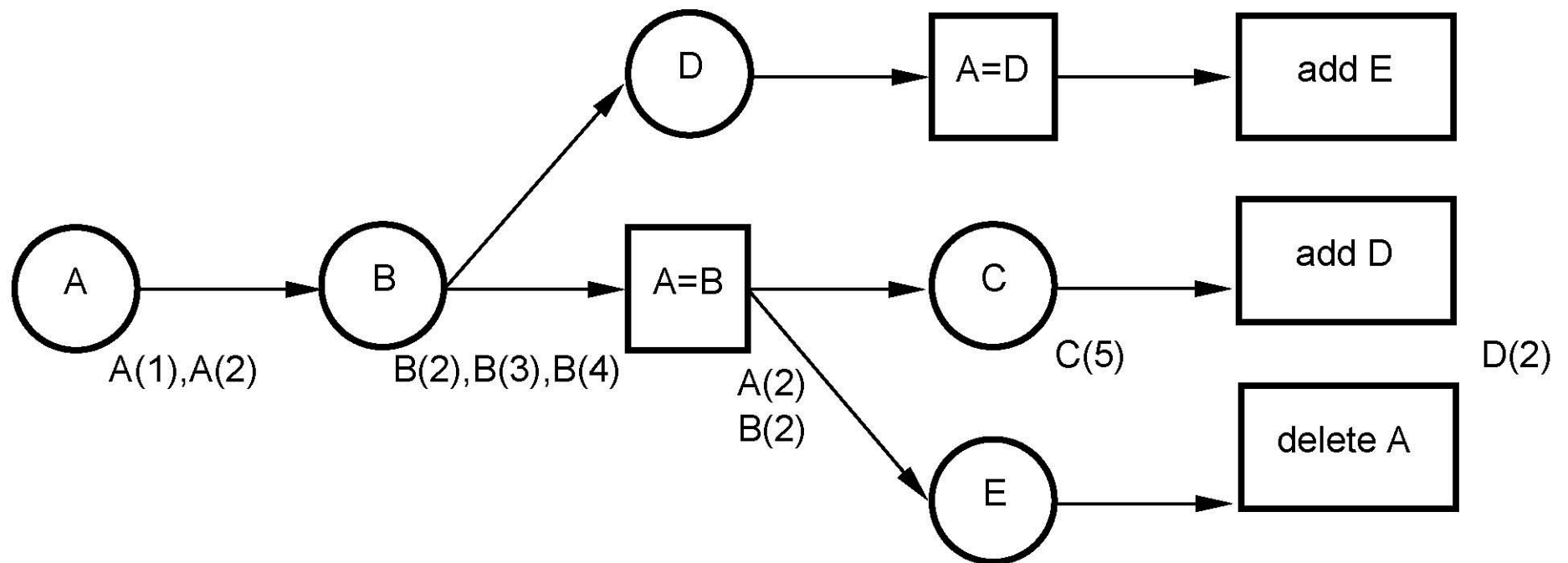
$A(x) \wedge B(x) \wedge C(y) \Rightarrow add_D(x)$

$A(x) \wedge B(y) \wedge D(x) \Rightarrow add_E(x)$

$A(x) \wedge B(x) \wedge E(x) \Rightarrow delete_A(x)$

■ *Arbeitsspeicher: A(1), A(2), B(2), B(3), B(4), C (5)*

Optimierung von Regelanwendungen



Der Rete Algorithmus

- *Voraussetzung: Der Arbeitsspeicher ändert sich nur langsam.*
- *Das Matching wird nicht wiederholt. Alle Information wird im Netz aufbewahrt, d.h. Duplikation über die Zeit wird vermieden. Motivation: Zwischen den Zeitpunkten t und $t+1$ sind fast alle Fakten gleich geblieben.*
- *Die Bedingungen der Regeln werden durch ein Netzwerk von Tests repräsentiert; gemeinsame Teilbedingungen werden nur einmal getestet.*
- *Eine Änderung im Arbeitsspeicher wird sofort durch das Netz propagiert, zu i.a. geringen Kosten.*
- *Kompilierung von Regelmengen in relativ effizienten maschinen-nahen Code.*

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.

Planungssystem für VW-Gruppe

Strategie der VW GRUPPE

Marketing Strategie	individuelle Fahrzeug-Spezifikationen durch Kunden	Bestseller-orientierte Fahrzeug-Spezifikationen durch Hersteller
Komplexität	Große Anzahl möglicher Varianten	Kleine Anzahl möglicher Varianten

Fahrzeug- Spezifikation



Item	short back	2,8L 150 kW spark	Type alpha	4	leather, Type L3	yes	
Item family	body variant	engine	radio	door layouts	seat covering	vanity mirror	

Varianten des Golf

- etwa 200 item families (Variablen, Merkmale)
- von 2 bis 50 items pro item family
(Werte pro Variable)



- mehr als 2^{200} mögliche Fahrzeug Spezifikationen
- Wahl der gültigen Spezifikationen ist beschränkt durch **REGEL SYSTEME**
(10.000 technische Regeln, noch mehr marketing- und produktions-orientierte Regeln)

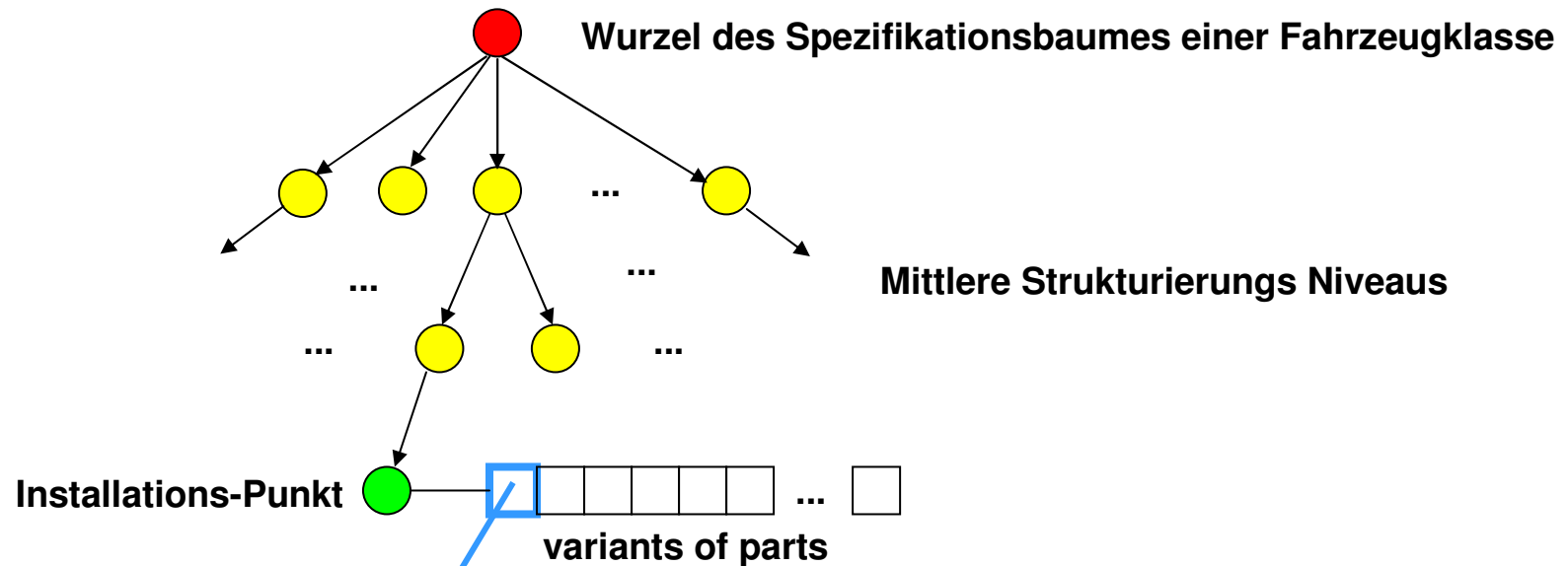
Beispiel

if $engine = e_1$ then $transmission = t_3$

if $engine = e_4$ and $auxiliary\ heater = h_1$

then $generator \in \{g_3, g_4, g_5\}$

Planungs-Problem: Vorhersage des Teilebedarfs



Installation Bedingung: Disjunktion von Merkmal Kombinationen

Installations Rate bei jedem Installations Punkt ergeben in der Summe 1

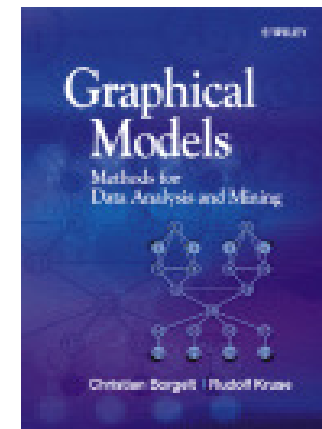
BEISPIEL : > 100.000 Merkmal Kombinationen werden in der „Golf“ Klasse genutzt

Vorschlag zur Problemlösung

- Modellierung des Verhaltens des Planers (wissensbasiertes System)
- Modellierung von Regeln: Constraint-Netze
- Nutzung historischer Daten: Data Mining
- Kombination dieser Wissensquellen: Informations-Fusion
- Planungsunterstützung: Belief Change, Visualisierung

Probleme dieser Art wurden von 1990-2000 in multinationalen EU-Projekten behandelt: DRUMS 1, DRUMS 2 and FUSION

Unsere Empfehlung in 2000 für VW :
Relationale Graphische Modelle nutzen



Grundlegende Lösungsideen

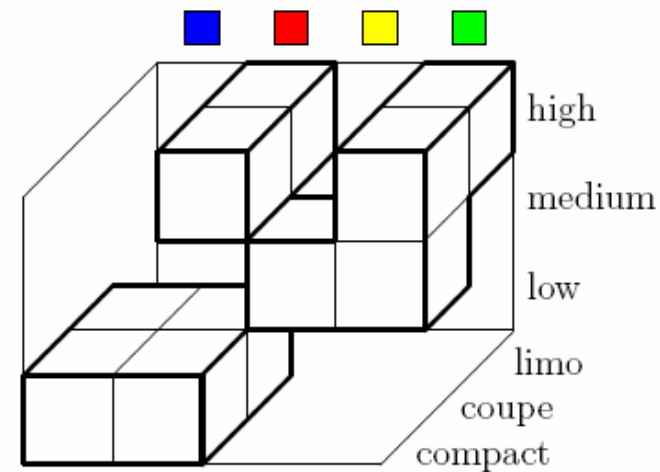
■ Vereinfachtes Beispiel

- 10 unterschiedliche ausgestattete Autos,
- 3 Merkmale
- 36 mögliche Kombinationen
- Schlußfolgerungen ?

color	speed	type
■	low	compact
■	low	coupe
■	low	compact
■	low	coupe
■	high	coupe
■	high	limo
■	medium	coupe
■	medium	coupe
■	high	coupe
■	high	limo

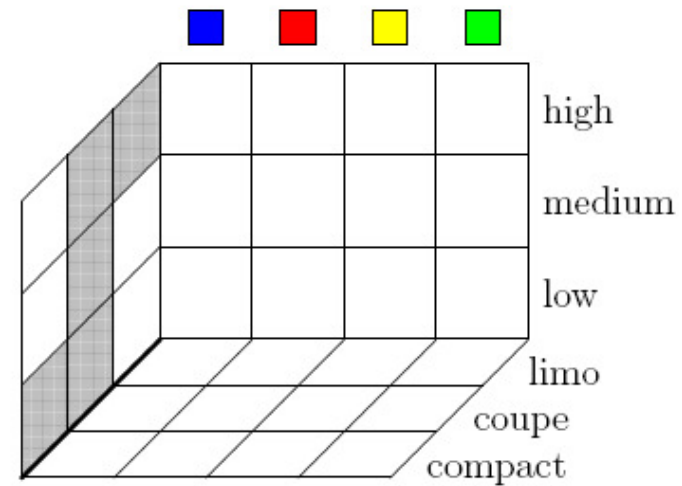
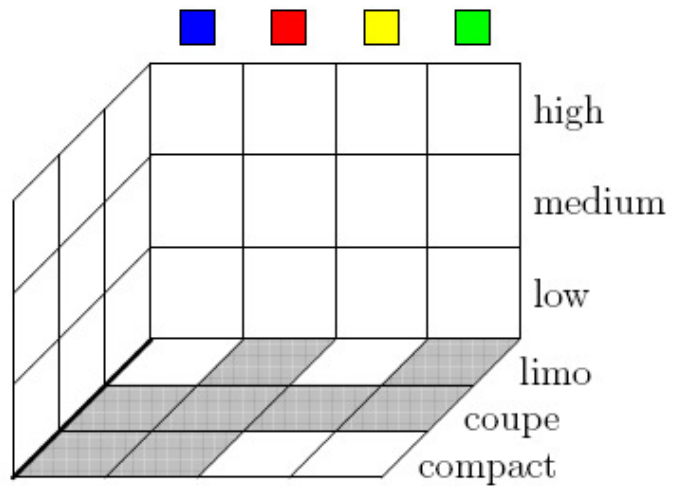
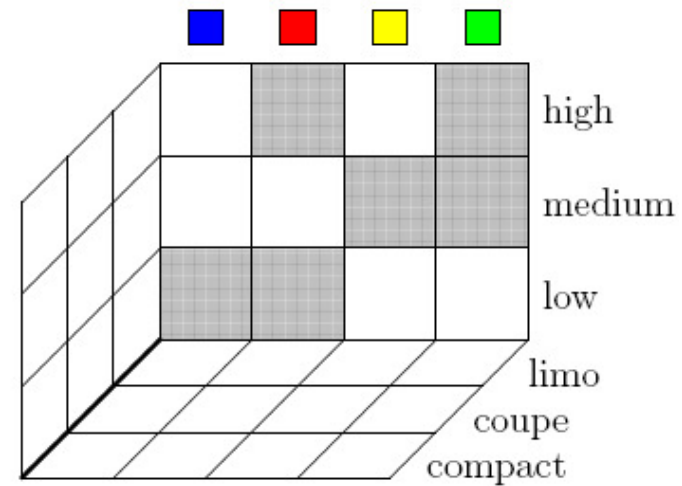
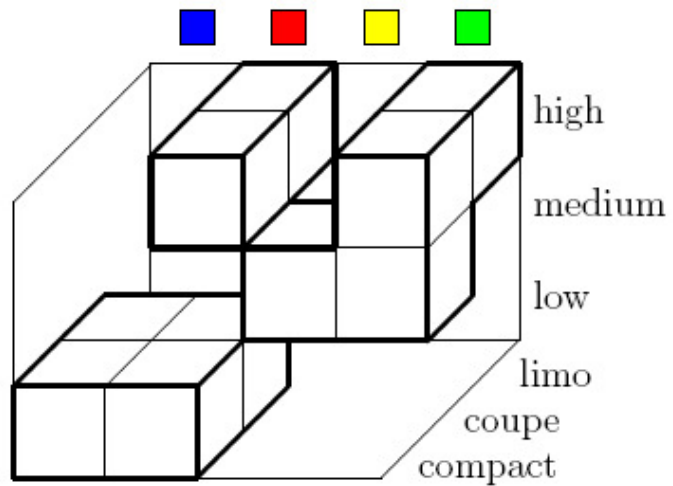
Relation

color	speed	type
■	low	compact
■	low	coupe
■	low	compact
■	low	coupe
■	high	coupe
■	high	limo
■	medium	coupe
■	medium	coupe
■	high	coupe
■	high	limo



Jeder Würfel entspricht einem Tupel

Projektionen



Regelsysteme, dargestellt über Constraints

Regeln :

$r_1 :$ if $A = a_1$ then $B \in \{b_3, b_4, b_5\}$

$\wedge$ $r_2 :$ if $B = b_2$ then $A = a_2$



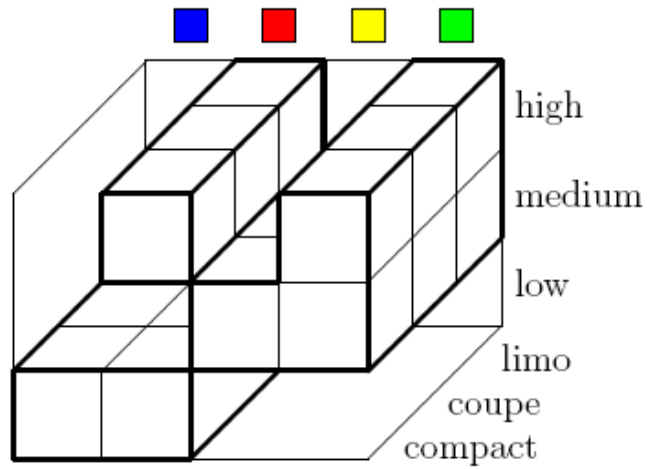
Regel Schema : (A,B)

$Dom(A) = \{a_1, a_2, a_3\}$

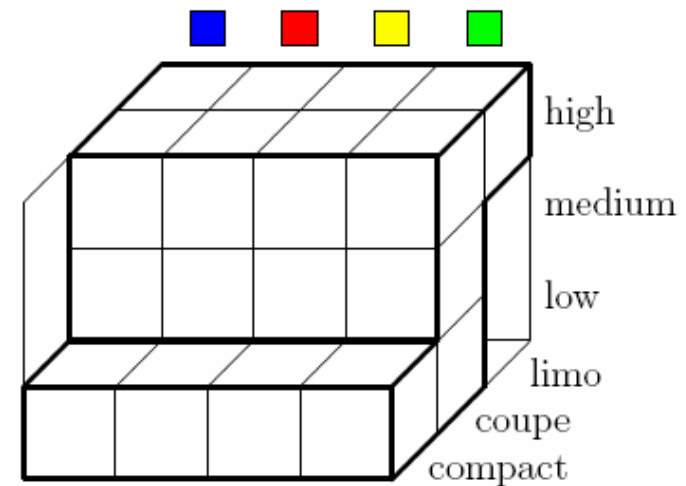
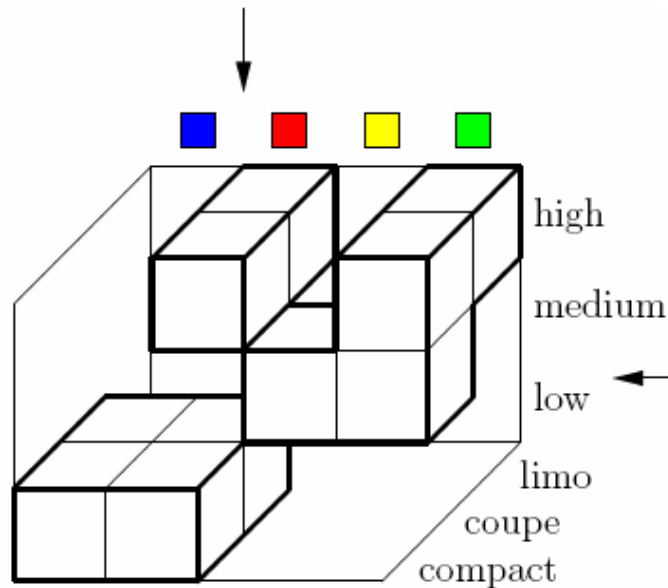
$Dom(B) = \{b_1, b_2, b_3, b_4, b_5\}$

	b_1	b_2	b_3	b_4	b_5
a_1					
a_2					
a_3					

Zylindrische Erweiterung und Schnitte

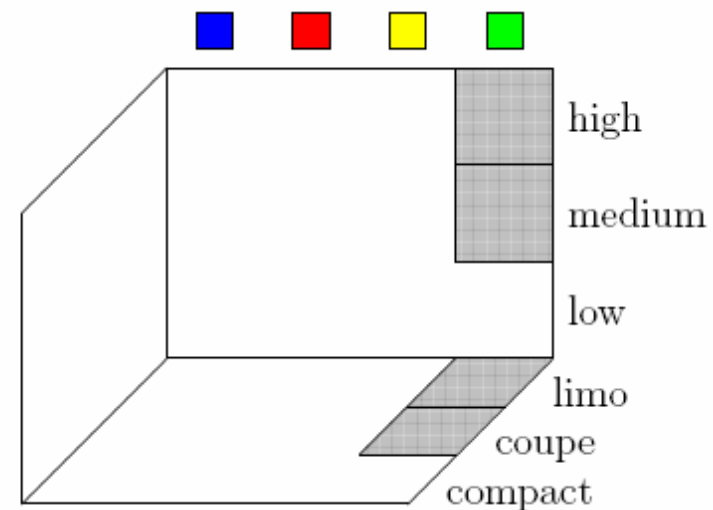
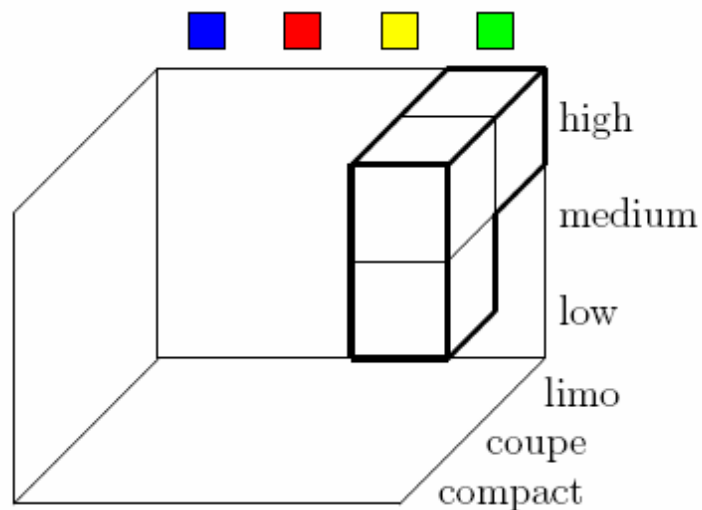


Zerlegung in
Unterräume ist
hier möglich

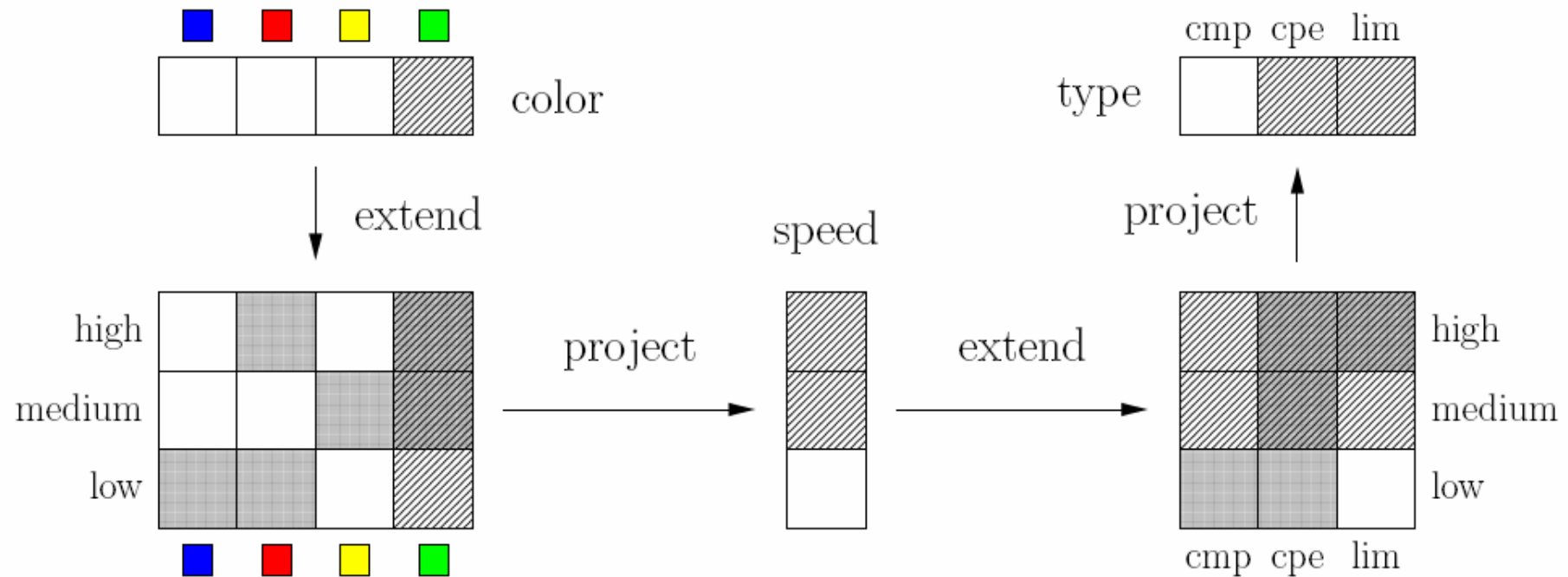


Schlußfolgern

- Information: Das Auto ist grün.
- Dann sind nicht mehr alle Kombinationen möglich und man kann schließen: Das Objekt ist
 - lomo oder coupe
 - high oder medium

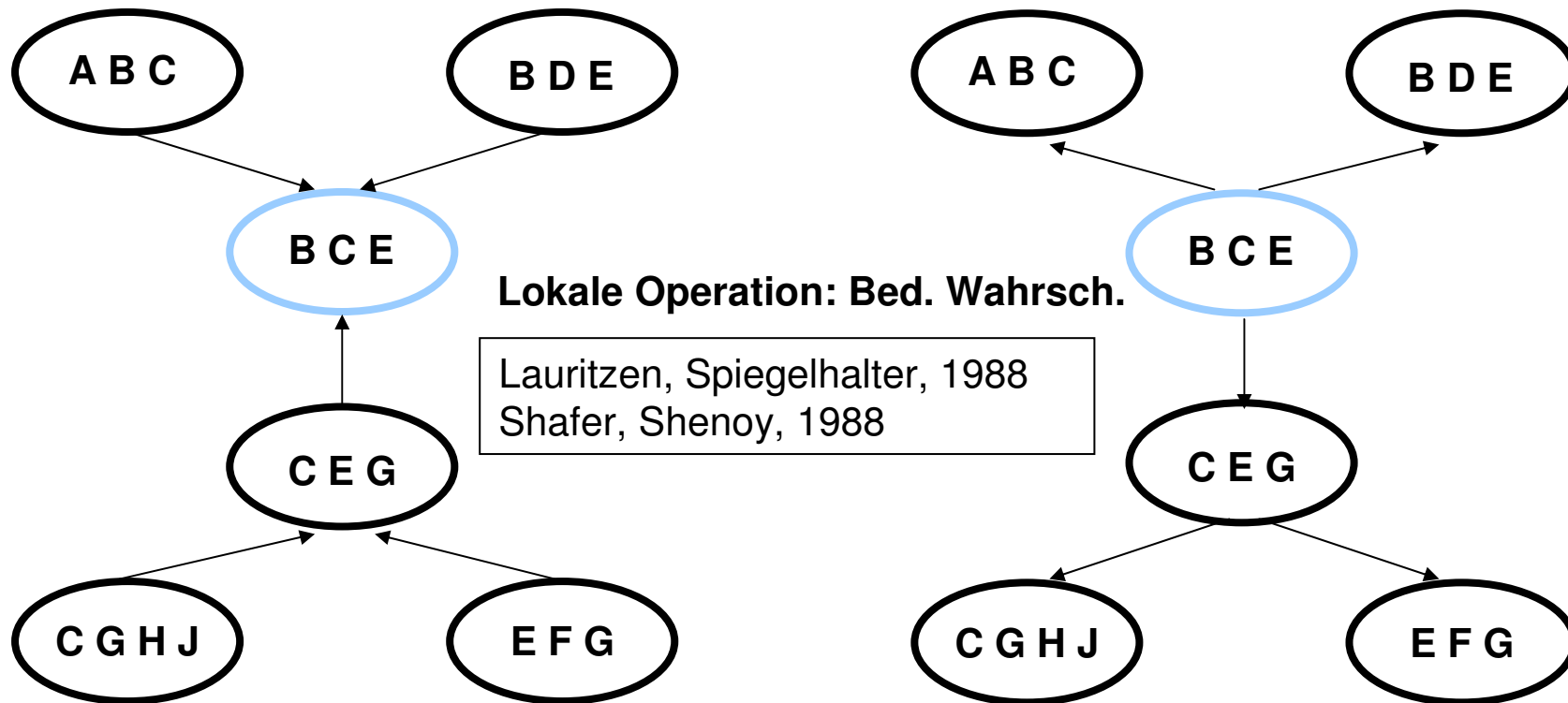


Schlußfolgern mit Projektionen



Propagation in Cliques-Bäumen

1. Lokale Berechnungen hinsichtlich der Cliques



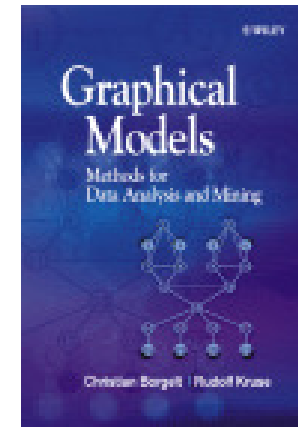
2. Information sammeln

3. Information verteilen

Relationale Graphische Modelle

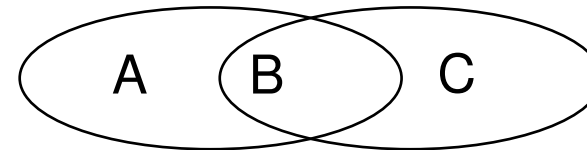
■ Relationale Graphische Modelle

≡ Zerlegung in niederdimensionale Räume sowie
Relationen auf niederdimensionalen Räumen



■ Beispiel

Constraints zwischen
A und B sowie B und C

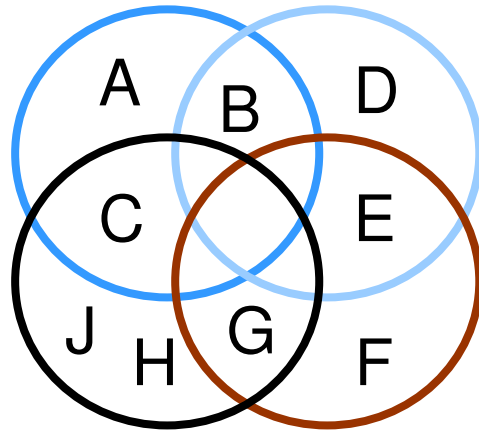


Hypergraph

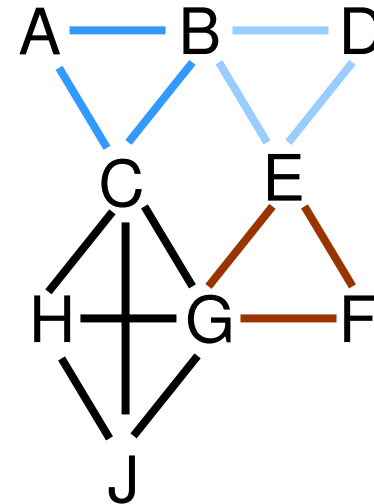
Idee: Spezifikation von

- Abhängigkeitsstruktur
- Relationen auf niederdimensionalen Räumen

Transformation in Hyperbaum Struktur



Hypergraph



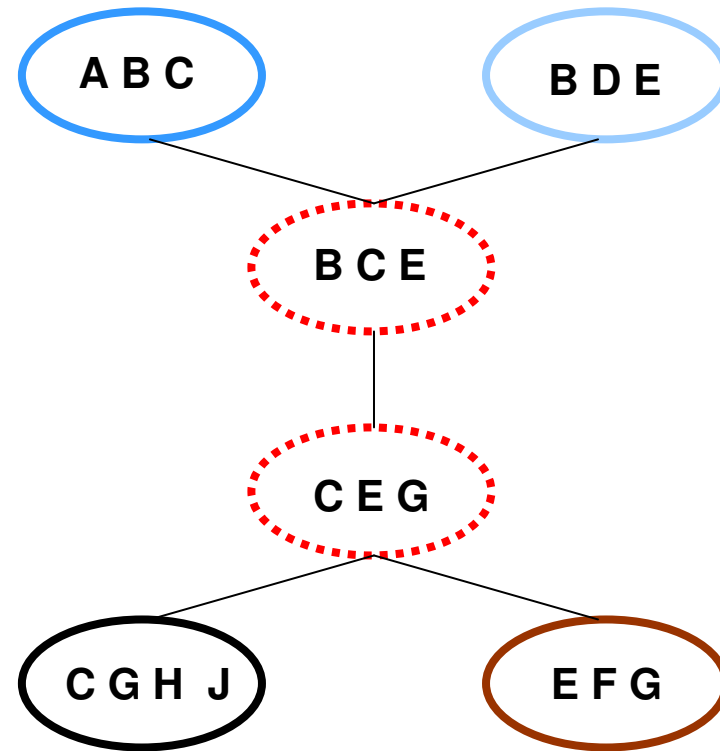
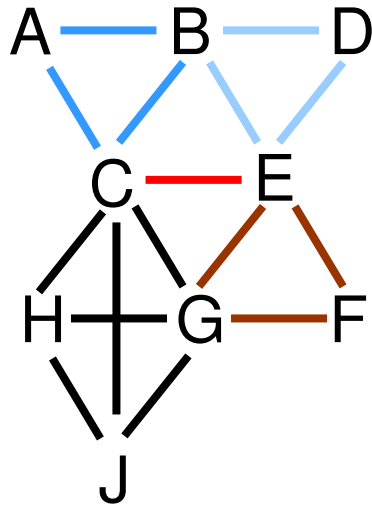
Ungerichteter Graph

Interpretation as a conditional independence graph:

$$(X, Y) \notin \mathcal{E} \iff X \perp Y \mid V - \{X, Y\}$$

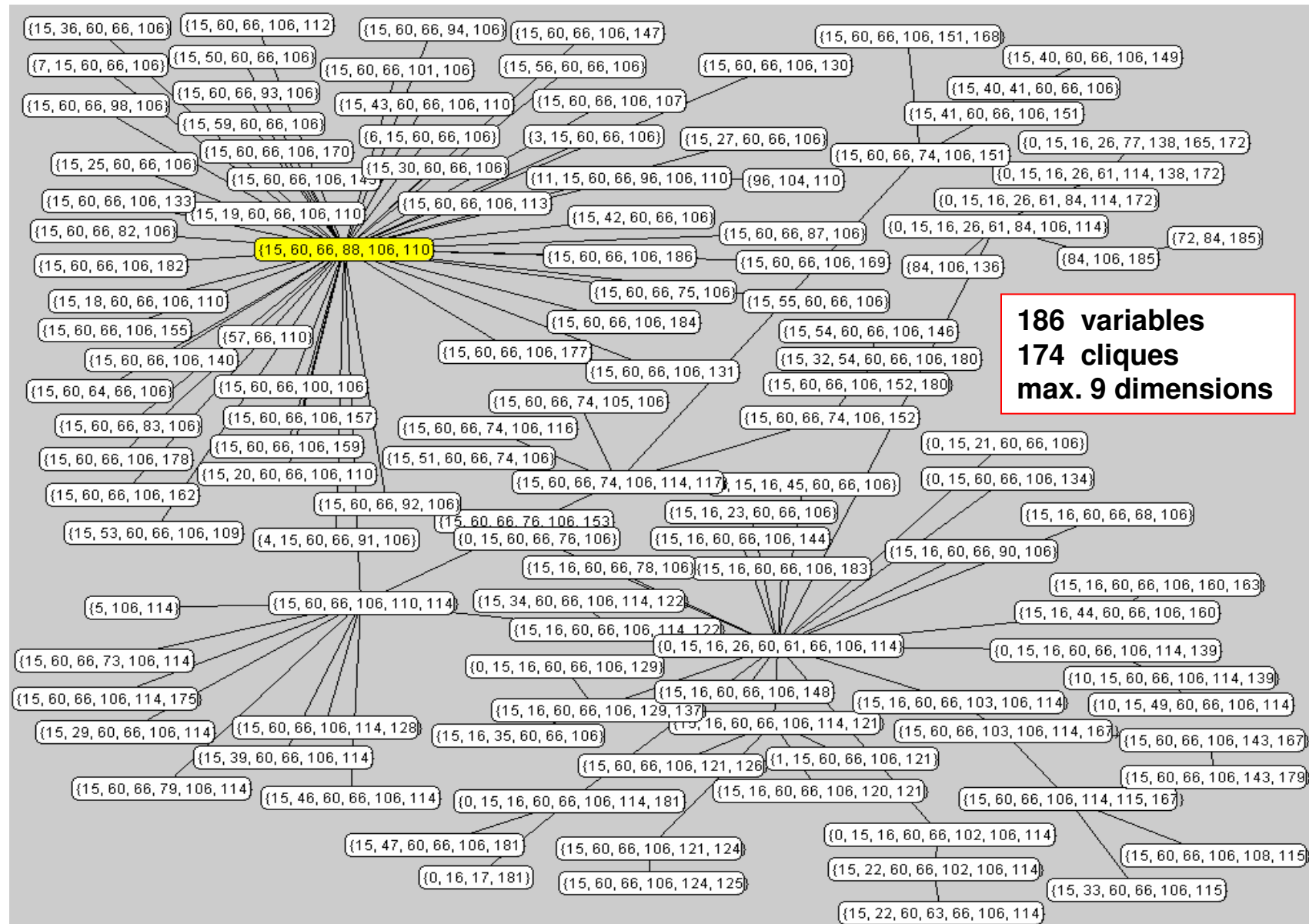
Transformation in Hyperbaum Struktur

Triangulation



Skeleton of Tree of Cliques
(hypertree structure)

Hyperbaum Struktur (VW Bora)



Mycin

MYCIN ist ein häufig zitiertes und einflussreiches Beispiel für ein regelbasiertes medizinisches Diagnosesystem.

- *Entwicklung Anfang der siebziger Jahre.*
- *Unterstützung der Diagnose von Infektionskrankheiten, Vorschlag von Therapiemaßnahmen.*
- *Regeln leiten Hypothesen aus Symptomen bzw. Beobachtungen ab.*
- *Vorgehen bei der Diagnose:*
 - Rückwärtsverkettend*
 - Interaktiv: Fehlende bzw. nützliche Information wird vom Benutzer angefordert.*

Im Unterschied zu bisher besprochenen Systemen: Mit den ca 500 Regeln ist ein numerisches Maß für die (Un-)Sicherheit der Aussagen bzw. Schlussfolgerungen verbunden (Sicherheitsfaktor, engl. certainty factor). Beim Schließen werden die Sicherheitsfaktoren ähnlich wie Wahrscheinlichkeiten miteinander kombiniert.

Mycin

Diagnosen in MYCIN: Identifikation wahrscheinlicher Krankheitserreger

Bezugsrahmen:

- *Form des Mikroorganismus: Stab (Bazillus), Kugel (Coccus) oder Spirale (Spirillum)*
- *Organismus wächst ärobisch (in Luft) oder anärobisch (ohne Luft)*
- *Reaktion auf Färbung (stain): gram positiv oder negativ*

Beobachtung von Kulturen liefert Information

Mycin

Beispiel-Regel

IF

- (1) the stain of the organism is gram positive, and*
- (2) the morphology of the organism is coccus, and*
- (3) the growth conformation of the organism is chains,*

THEN

there is suggestive evidence (0.7) that the identity of the organism is streptococcus.

Das Expertensystem Pathfinder

- Typical example from *Pathfinder*, a medical expert system

