# 8. Bildverstehen und Wissensrepräsentation

## *Begriffsklärungen*

### *Bildverstehen:*

beinhaltet

- Bedeutungserfassung einer Bildstruktur durch die Zuweisung von Namen
- Auffinden wichtiger Details (Bewertung)
- Assoziation mit anderen Bildern
- Begreifen von ursächlichen Zusammenhängen (kausales Verstehen des Bildinhaltes)
- intuitives Erfassen von Zusammenhängen (intuitives Verstehen des Bildinhaltes)
- Ziehen von Schlussfolgerungen aus den gefundenen Zusammenhängen

insbesondere:

- Erkennen von Objekten, Ereignissen, Szenen
- Erkennen von exemplarischen Situationen
- Erkennen von Absichten
- Ableitung von Schlüssen (z.B. hinsichtlich neuer Aktionen)

notwendig: Verwendung von *Modellen*

Modell: Darstellung prototypischer Eigenschaften, Strukturen und Methoden einer Klasse von Bildprimitiven, Bildkomponenten, -objekten, Szenen oder Situationen oder eines idealen Individuums einer Klasse (Levi 2002).

*ikonisches Modell*: analog zum geometrischen und topologischen Bildinhalt aufgebaut. (Verwendet werden üblicherweise Matrix-, Vektor- oder Stringdarstellungen.)

*symbolisches Modell*: keine direkte Analogie zur Geometrie und Topologie des Bildinhaltes vorhanden. (Darstellung erfolgt üblicherweise relational, prädikatenlogisch, oder durch Graphen oder Grammatiken; Näheres siehe unten)

weitere Definitionen:

*Bildprimitiv:* erste Aggregationsstufe klassifizierter Pixel (z.B. Kantenpunkte). Beispiele sind Kanten und Regionen.

*Bildkomponente:* Komponente eines Bildobjektes. Objekt hier: ein realer Gegenstand, bzw. die Region(en), die ihm im Bild entsprechen. (z.B. Hände, Augen, Nase...)

*Bildstruktur:* Struktur, die aus Bildprimitiven oder Bild-komponenten zusammengesetzt ist und die auch selbst wieder Teil komplexer Bildelemente (z.B. einer Bildszene) sein kann. Beispiele: Menschen, Roboter...

*Bildszene:* Struktur, die aus Bildstrukturen zusammengesetzt ist. Beispiele: Hörsaal, Kreuzung, Autobahn, Hafen, Bahnhof.

*Bildsituation:* semantische Beschreibung von Abläufen, die einer Bildszene zugeordnet werden können. Beispiele: zuhörende Studenten, Linksabbieger, Überholer, weinende und zuwinkende Menschen (Abschiedssituation).

Von der Stufe der "Bildkomponente" aufwärts erfolgt die Darstellung in der Regel symbolisch.

Unterscheidung von *high level* und *low level*-Bildanalyse.

Strategien des Bildverstehens:

*bottom-up:*

- geht vom Bild aus, leitet Repräsentationen von Objekten ab
- versucht, diese mit in einer *Wissensbasis* gespeicherten Repräsentationen für bekannte Objekte zu vergleichen

*top-down:*

- geht aus von hoher Repräsentationsebene (z.B. "finde eine bestimmte Objektkategorie"), legt dann einen Bildverarbeitungsverlauf fest

*bidirektionale Strategien:* kombinieren bottom-up und top-down

Wie kann nun eine symbolische Repräsentation aussehen,
d.h. wie wird Wissen dargestellt?

# *Wissensrepräsentationsmethoden der KI*

## *Produktionssysteme (Regelsysteme)*

– häufig in Expertensystemen benutzt
– besonders geeignet zur Darstellung von prozeduralem Wissen

Ein KI-Produktionssystem besteht aus den Hauptkomponenten:

- Daten
- Menge von Produktionsregeln (Operationen)
- Kontrollsystem.

Produktionsregel: hat die allgemeine Form

**`if`** *Prämisse* **`then`** *Conclusio*

oder, bei prozeduraler Interpretation:

**`if`** Bedingung *X* ist erfüllt **`then`** Aktion *Y* ist ausführbar

Das Kontrollsystem analysiert, welche Prämissen aufgrund der aktuellen Daten erfüllt sind, und entscheidet, welche der somit anwendbaren Regeln ausgeführt werden sollen.
Die Conclusio (Aktion) kann dann eine Änderung der Daten bewirken.

Als Programmier-Paradigma (Fallregel- oder van Wijngaarden-Paradigma):
typische Unterschiede zum klassischen prozeduralen (von Neumann-) Paradigma

- keine lokalen Daten
- keine starre Sequenz von Programmbefehlen
- keine hierarchische Organisation
- kein Aufruf einer Regel von einer anderen Regel

Beispiel:

Wissensbasis:

| | | | |
|---|---|---|---|
| Daten | object A | **is_a** | ball |
| | object B | **is_a** | ball |
| | object C | **is_a** | shoe |
| Regel | **if** ball | **then** | circular |

Die Regel kann auf die Daten zu Objekt A und B angewandt werden; es kann z.B. die Information abgeleitet werden, dass 2 der Objekte "circular" sind.

Mit einer rein enumerativen Wissensrepräsentation müsste zur Gewinnung dieser Information eine Auflistung der Objekte etwa so aussehen:

object A **is_a** (ball, circular)
etc.

was wesentlich speicherplatzintensiver wäre.

*Strategien der Problemlösung* mit Produktionssystemen:

Ausgehend von einem Startzustand (gegebene Menge von Daten) soll ein Zielzustand durch Regelanwendungen erreicht werden.

- vollst. Graph eines Produktionssystems: verbindet jeden Zustand mit seinen Nachfolgezuständen bzgl. Regelanwendung.

Aufgabe der Kontrollstrategie: Finde einen Weg vom Start- zum Zielzustand.

- *unwiderrufliche* (irrevocable) Kontrollstrategien
- *versuchende* (tentative) Kontrollstrategien (z.B. Backtracking; Graphensuchalgorithmen)
- *Forward-Produktionssysteme*: gehen vom Startzustand aus; versuchen, durch Anwendung von Regeln (F-rules) das Ziel zu erreichen
- *Backward-Produktionssysteme*: gehen vom Ziel aus; Anwendung von B-rules, um zum Startzustand zu kommen
- *Bidirectional PS*: Kombination beider Strategien, gehen gleichzeitig von Start- und Zielzustand aus

## *Prädikatenlogik erster Ordnung*

- Darstellung von Faktenwissen und Regelwissen (Inferenzen)
- sehr einheitliche Repräsentationsform mit etabliertem Theorie-Hintergrund
- Programmiersprache PROLOG
- Nachteil: keine Darstellung von unvollständigem oder unscharfem Wissen.

Bausteine: *Formeln*
Syntaktische Grundbausteine der Formeln:

- Prädikatensymbole
- Konstantensymbole
- Variablensymbole
- Funktionssymbole

jedes Prädikaten- und jedes Funktionssymbol besitzt eine nichtverschwindende Anzahl von Argumenten (übl. Klammerschreibweise)

*Term* **:** Konstantensymbol | Variablensymbol | Funktionssymbol mit Termen als Argumenten
*atomare Formel* **:** Prädikatensymbol mit Termen als Argumenten

Syntaktisch korrekte *Formeln* werden aus atomaren Formeln mitteln Konnektoren (und, oder, nicht, Implikation, Äquiv.) zusammengesetzt
Erweiterung: Zulassen von Quantoren ($\forall$, $\exists$).

*Interpretation* einer Formel:
stellt die Verbindung zwischen den Symbolen und einem gegebenen Problembereich (z.B. Bildinformationen) her

- jedem Konstantensymbol wird ein Objekt, jedem Prädikatensymbol eine Relation und jedem Funktionssymbol eine Funktion des jeweiligen Problemkreises zugeordnet

Beispiel: kollinear($x$, $y$, $z$)
die Belegung der Variablen $x$, $y$, $z$ mit den Konstantensymbolen A, B, C liefert für die Formel den Wert TRUE, wenn die A, B, C entsprechenden Punkte auf einer Geraden liegen.

Variablen in einer Formel im Wirkungsbereich eines Quantors heißen *gebundene* Variablen.
*freie* Variablen: außerhalb d. Wirkungsbereichs eines Quantors.

genauer:

**Definition**

- Hat eine Formel $A$ die Gestalt $\forall xB$ oder $\exists xB$, so heißt $B$ der Wirkungsbereich des *Präfixes* $\forall x$ bzw. $\exists x$ von $A$.
- Ein Auftreten einer Variablen $x$ in einer Formel $A$ heißt gebunden, wenn es innerhalb des Wirkungsbereichs eines Präfixes $\forall x$ oder $\exists x$ einer Teilformel von $A$ stattfindet.
- Ein Auftreten einer Variablen $x$ in einer Formel $A$ heißt frei, wenn es nicht gebunden ist und nicht unmittelbar rechts neben einem Quantor stattfindet.

$$\forall x(p_0(x, y) \rightarrow \forall z(\exists y\ p_1(y, z) \vee \forall x\ p_2(f(x), x)))$$

gebundene Vorkommen
freie Vorkommen

Ein *Satz* ist eine Formel ohne freie Variablen.

- Subkalkül des Prädikatenkalküls 1. Ordnung: *Propositionenkalkül*, verzichtet auf Variablen und Quantoren
- Erweiterung des Prädikatenkalküls 1. Ordnung: *Prädikatenkalkül 2. Ordnung* – Variablen und Quantoren nicht nur für Objekte, sondern auch für Relationen zugelassen ($\Rightarrow$ Gewinn an Mächtigkeit, aber Verlust an Entscheidbarkeit)

Zwei Formeln heißen *äquivalent*, wenn sie für jede gemeinsame Interpretation den selben Wahrheitswert liefern.

Ist $M$ eine Menge von Formeln und $I$ eine Interpretation, so heißt $I$ *Modell* für $M$, wenn sich für jede Formel in $M$ unter $I$ der Wahrheitswert TRUE ergibt.

*M allgemeingültig*: jede Interpretation ist Modell für $M$.

Beispiele für allgemeingültige prädikatenlogische Formeln:

1. $\neg\forall xA \leftrightarrow \exists x\neg A$,
2. $\neg\exists xA \leftrightarrow \forall x\neg A$
3. $\forall x\forall yA \leftrightarrow \forall y\forall xA$,
4. $\exists x\exists yA \leftrightarrow \exists y\exists xA$
5. $\forall x(A \wedge B) \leftrightarrow \forall xA \wedge \forall xB$
6. $\exists x(A \vee B) \leftrightarrow \exists xA \vee \exists xB$
7. $\forall\vec{y}(A \wedge QxB \leftrightarrow Qx(A \wedge B))$,
   falls $x \notin Frei(A)$ und $\vec{y}$ alle freie Variablen in $A \wedge QxB$ sind.
8. $\forall\vec{y}(QxA \wedge B \leftrightarrow Qx(A \wedge B))$,
   falls $x \notin Frei(B)$ und $\vec{y}$ alle freie Variablen in $A \wedge QxB$ sind.
9. $\forall\vec{y}(A \vee QxB \leftrightarrow Qx(A \vee B))$,
   falls $x \notin Frei(A)$ und $\vec{y}$ alle freie Variablen in $A \wedge QxB$ sind.
10. $\forall\vec{y}(QxA \vee B \leftrightarrow Qx(A \vee B))$,
    falls $x \notin Frei(B)$ und $\vec{y}$ alle freie Variablen in $A \wedge QxB$ sind.

(11) $((\forall xA)\rightarrow B) \leftrightarrow (\exists x(A\rightarrow B))$ *und dual*
(folgt aus 1 und 2)

Eine Formelmenge $M$ heißt *erfüllbar*, wenn für $M$ ein Modell existiert.

Eine Formel $f$ *folgt logisch aus* $M$, wenn jedes Modell für $M$ auch Modell für $f$ ist.

Eine *Inferenzregel* ist eine Vorschrift, um aus einer gegebenen Menge von Formeln neue Formeln abzuleiten.

Beispiele:

*modus ponens*

$$\frac{f \qquad f\rightarrow g}{g}$$

*reductio ad absurdum*

$$\frac{f \rightarrow g \qquad f \rightarrow \neg g}{\neg f}$$

Resolutionsregel (wichtig in PROLOG):

$$\frac{f \vee g \qquad \neg g \vee h}{f \vee h}$$

- Eine Menge *R* von Inferenzregeln heißt *widerspruchsfrei*, wenn jede Formel, die sich mittels *R* aus einer Menge *M* von Formeln ableiten lässt, auch logisch aus *M* folgt.
- *R* heißt *vollständig*, wenn jede Formel, welche logisch aus einer Menge *M* von Formeln folgt, auch mittels Regeln aus *R* aus *M* gewonnen werden kann.

(Anwendung: automatisches Theorem-Beweisen)

Im Prädikatenkalkül 1. Ordnung gibt es vollständige und widerspruchsfreie Inferenzregeln. (Speziell: Beweisverfahren mit Hilfe der Resolutionsregel, Verwendung in PROLOG.)

*Partielle Entscheidbarkeit* des Prädikatenkalküls 1. Ordnung:

- Wenn ein Theorem aus den Axiomen folgt, kann immer in endlich vielen Schritten durch Anwendung der Inferenzregeln ein Beweis gefunden werden.
- Aber: wenn eine Formel *nicht* aus den Axiomen folgt, ist die Termination des Ableitungsprozesses mittels der Inferenzregeln nicht sichergestellt.

*ein Logisches Programm:*

Darstellung von Wissen in Form von Formeln der Prädikatenlogik 1. Ordnung, evtl. in spezieller Form (z.B. Horn-Klauseln; PROLOG).

*Beispiel:*
(1) Jedes Bild enthält mindestens ein Objekt.
(2) Aufnahme 1 ist ein Bild und enthält nur Würfel.
(3) Aufnahme 1 enthält mindestens einen Würfel.

Übersetzung in Formeln:
(1) $\forall x\,(\text{BILD}(x) \rightarrow \exists y\,(\text{IN}(y, x)))$
(2) BILD(AUFNAHME1)
(2a) $\forall x\,(\text{IN}(x, \text{AUFNAHME1}) \rightarrow \text{WÜRFEL}(x))$
(3) $\exists x\,(\text{WÜRFEL}(x) \wedge \text{IN}(x, \text{AUFNAHME1}))$

Beachte: Formel (3) folgt logisch aus (1), (2) und (2a).

Verbindung zu Produktionssystemen:

Anwendung der Inferenzregeln im Prädikatenkalkül lässt sich als Anwendung eines Produktionssystems auffassen.

speziell in der PROLOG-Variante: tentatives Backward-Produktionssystem mit Backtracking.

## *Wissensdarstellung mit relationalen Strukturen*

**Definition** *Relationalstruktur RS*

$$RS = (N, R)$$

$N = \{n_1, n_2, \ldots, n_l\}$ = Knotenmenge, $n_i$ beschreiben häufig Objektkomponenten. $R = \{R_1, \ldots, R_m\}$ =Relationenmenge;

$$R_i \subseteq \underbrace{N \times \ldots \times N}_{s_i} = N^{s_i}$$

$R_i$: $s_i$-stellige Relation auf der Knotenmenge N; i=1,...,m $s_i \geq 1$.

Spezialfall: p-partiale Relationalstruktur

$$N = N_1 \cup N_2 \cup \ldots \cup N_p \quad , \qquad N_i \cap N_j = \varnothing$$

$$i \neq j, \quad i,j = 1, \ldots, p$$

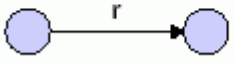
$$R_i \subseteq \underbrace{N_{i_1} \times N_{i_2} \times \ldots N_{i_{s_i}}}_{s_i}$$

Spezialfall: (gerichtete) *Graphen* ($s_i = 2$)

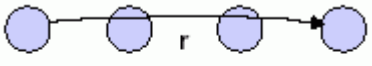
grafische Darstellung von Relationen:

Graphical representation

binary relation:



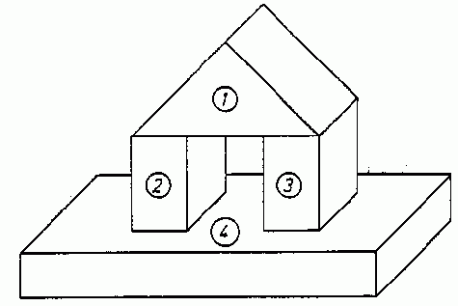

n-ary relation:



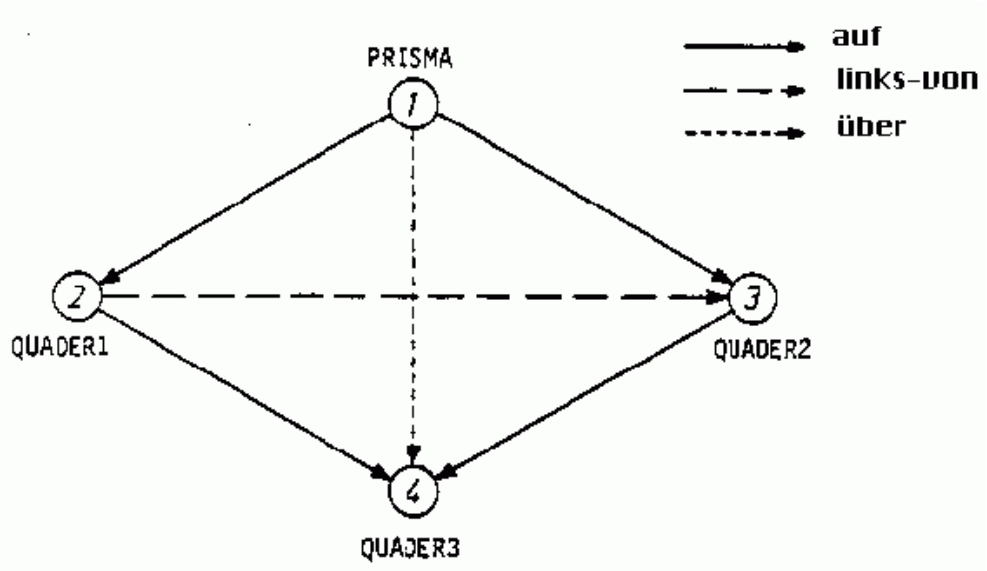

"hypergraph"

Beispiel (aus Levi 2002):

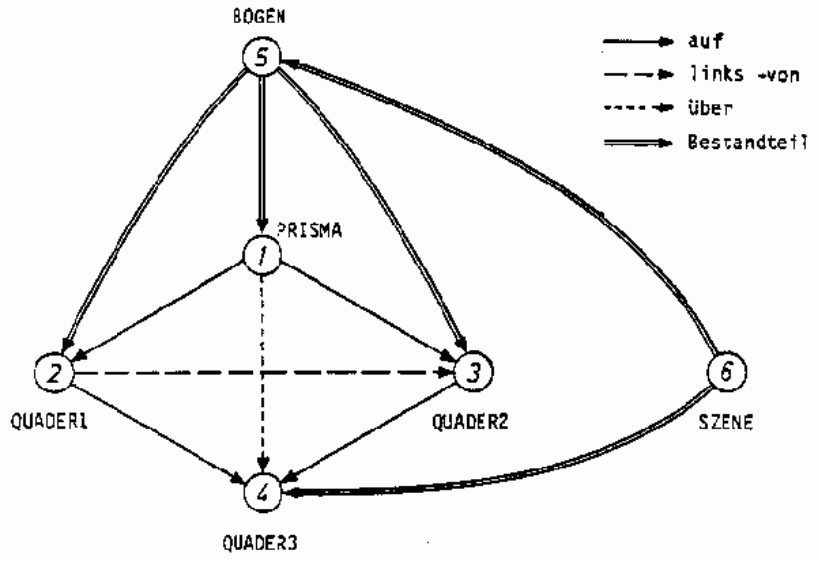
Szene



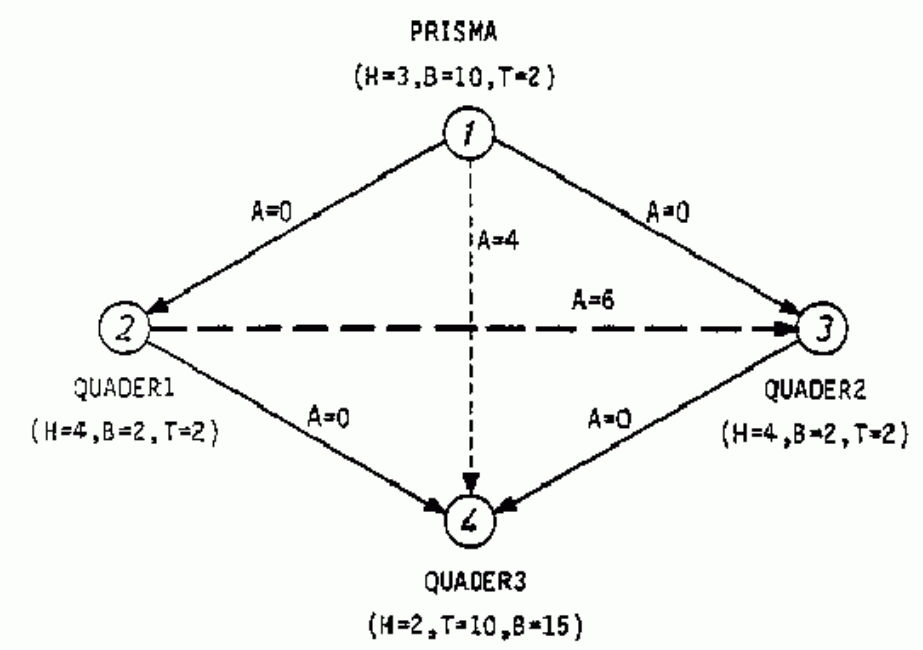

Graph:



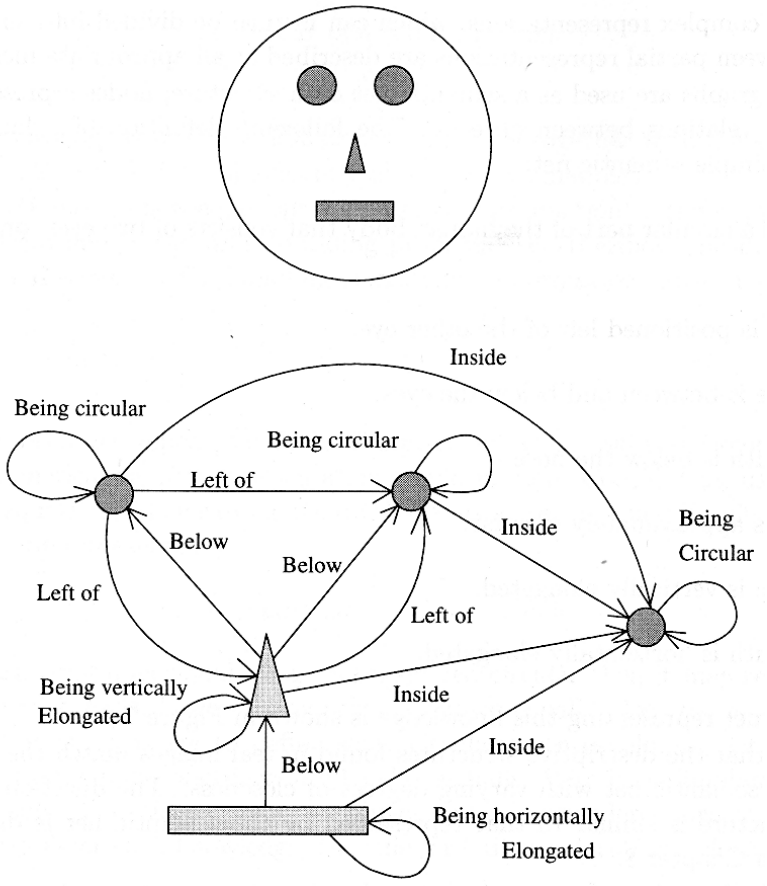

Ergänzung durch hierarchische Szenenbeschreibung (mit zusätzlichem Kantentyp *part-of*):



attributierter Graph (Höhe, Breite, Tiefe, Abstand):

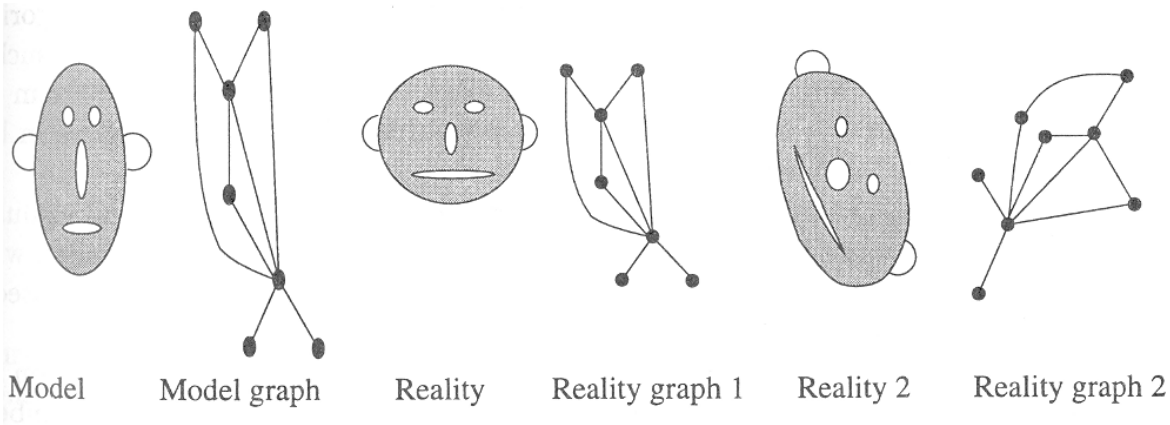



Relationen-Modell eines Gesichts (aus Sonka et al. 1999):

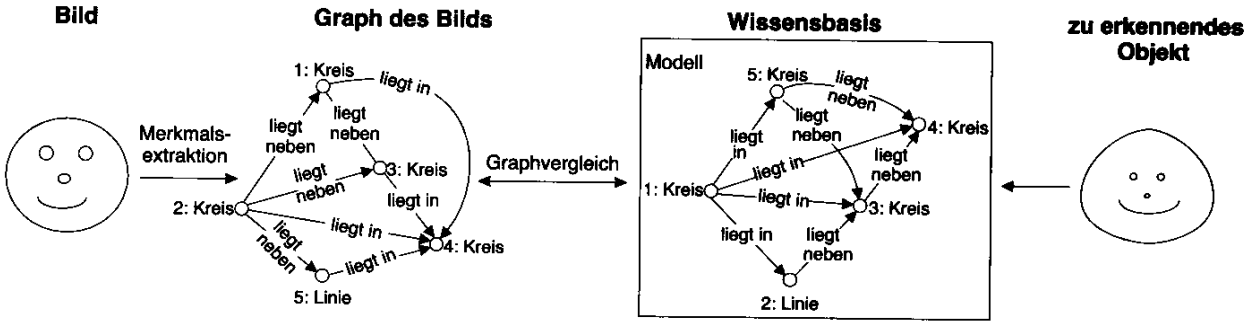



vgl. *Szenengraph* in VRML und Java 3D

⇒ Wiedererkennung von Objekten wird zu einem Graph-Matching-Problem (Auffinden von Graphen-Isomorphismen)

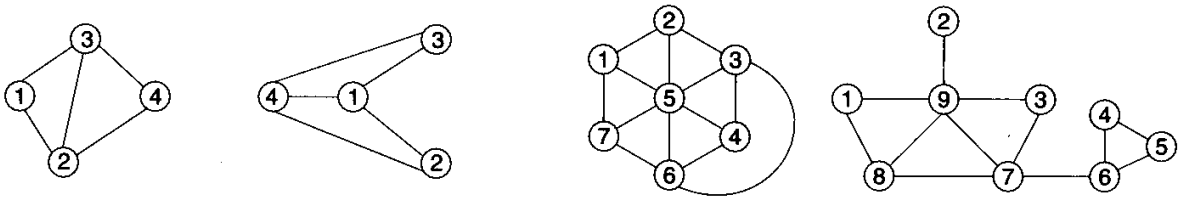



- Die zu erkennenden Objekte müssen vorab als Modelle abgelegt werden (Repräsentation als Graphen).
- Aus einem zu analysierenden Bild müssen die Merkmale extrahiert werden, die man für den Aufbau eines Graphen-Modells benötigt.
- Auf der Basis der extrahierten Merkmale wird der Graph einer Szene erstellt.
- Nun können die Graphen verglichen werden (Suche nach Subgraph-Isomorphien).



Beispiele für Graphen-Isomorphien:

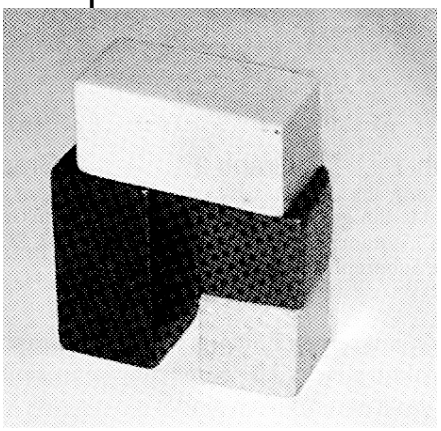
- Graph 1 ist isomorph zu Graph 2 und zu mehreren Subgraphen von Graph 3
- Graph 4 ist auf zweifache Weise Subgraph von Graph 3



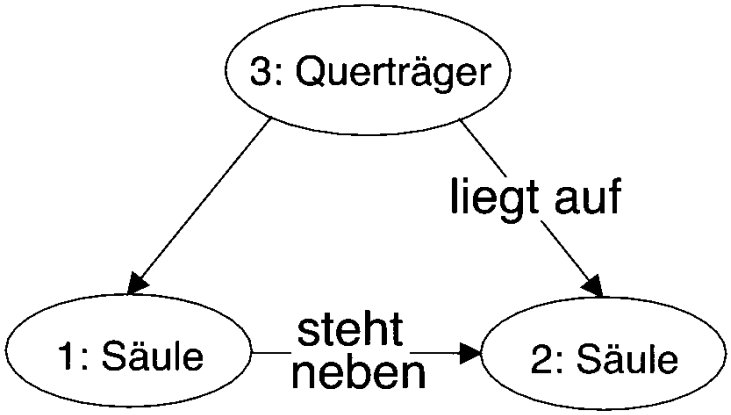

übergeordnete Konzepte können durch bestimmte, attributierte Subgraphen definiert werden

z.B. "Torbogen"

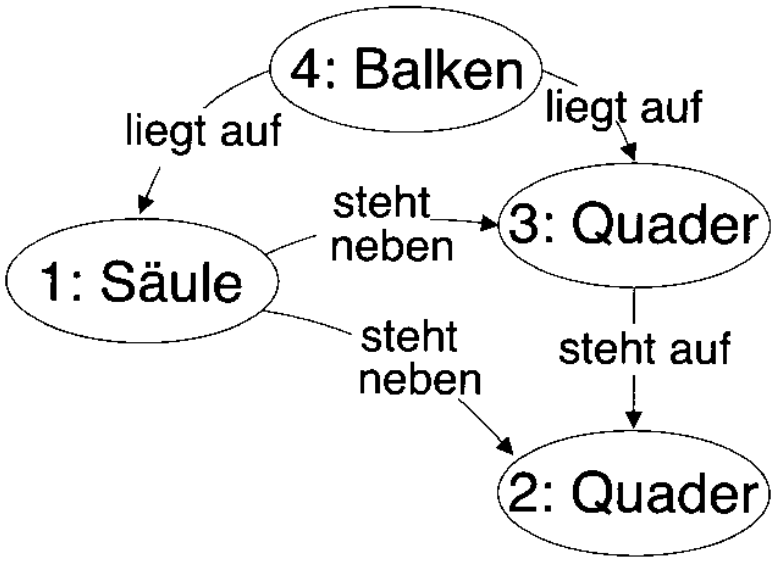
Beispiel



Graphen-Definition eines Torbogens:



Problem: der aus dem obigen Bild extrahierte Graph würde anders aussehen als dieser Prototyp.

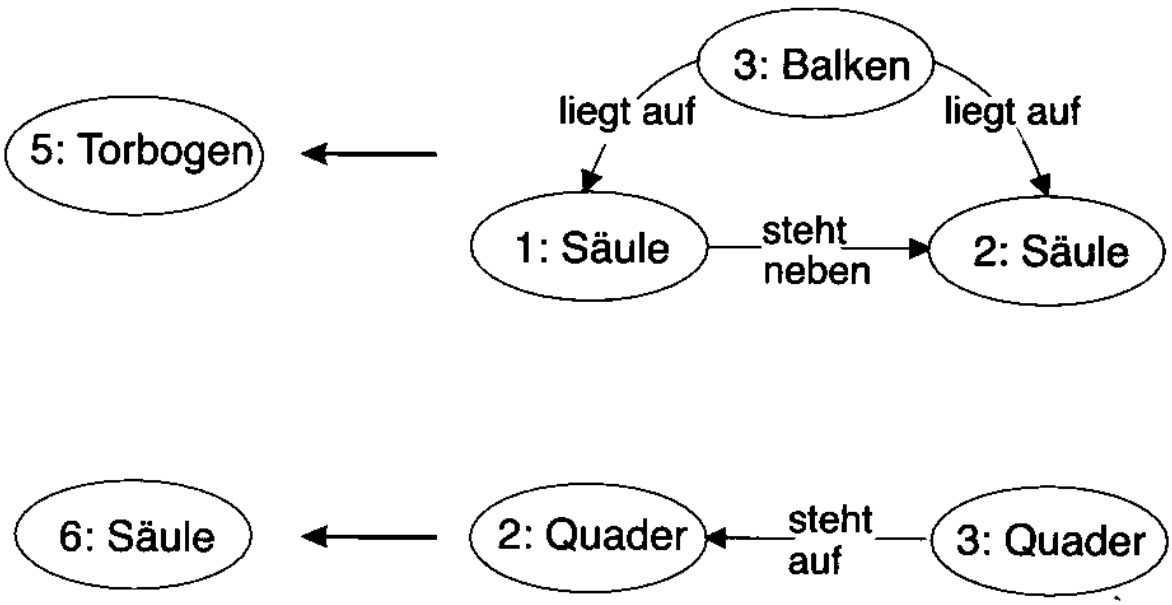



⇒ würde nicht als Torbogen erkannt!

Abhilfe:

- Verwendung von Graph-Grammatiken
- dadurch Kombination (Schachtelung) der Objekterkennungsschritte

einfache Graph-Grammatik für das Torbogen-Beispiel:



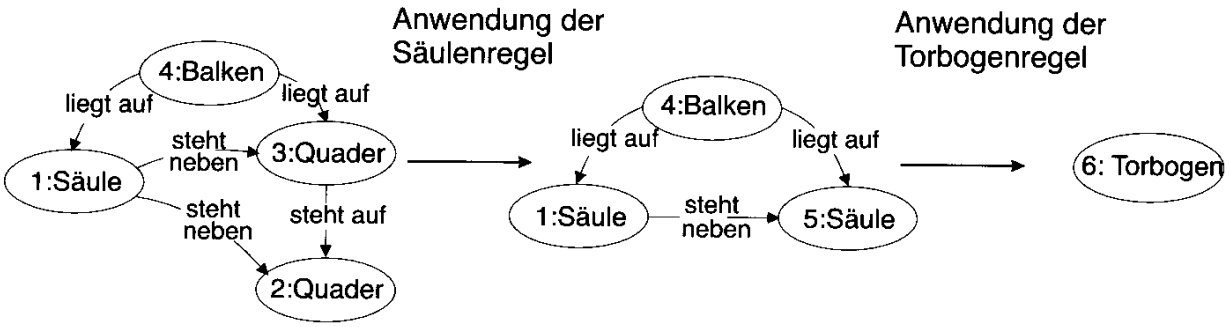
einfaches Isomorphie-Matching reicht jetzt nicht mehr für die Objekterkennung

- es muss berücksichtigt werden, wie man in einem Graphen G einen Teilgraphen G', der auf der rechten Seite einer Regel steht, durch den Graphen G'' auf der linken Seite der Regel ersetzt. Es muss dabei gewährleistet sein, dass G'' auf dieselbe Weise in G eingebettet ist wie zuvor G' (Persistenz der Relationen): "korrekte Einbettung" von G''.
- es muss gesteuert werden, in welcher Reihenfolge die Grammatikregeln angewendet und die Graphen ersetzt werden

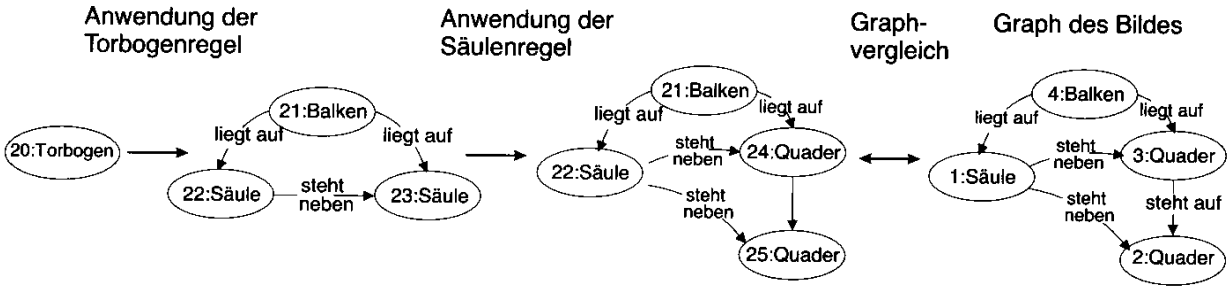
dafür 2 Strategien:

*datengetriebene Verarbeitung (bottom-up)*
man geht vom Graphen des Eingabebildes aus und ersetzt Subgraphen solange anhand der Regeln, bis man zu einem oder mehreren Objekten gelangt ist, die als Zielobjekte def. sind. (Anwendung der Grammatik-Regeln von links nach rechts)

Im Bsp.: erst "Säule"-Regel, dann "Torbogen"-Regel anwenden

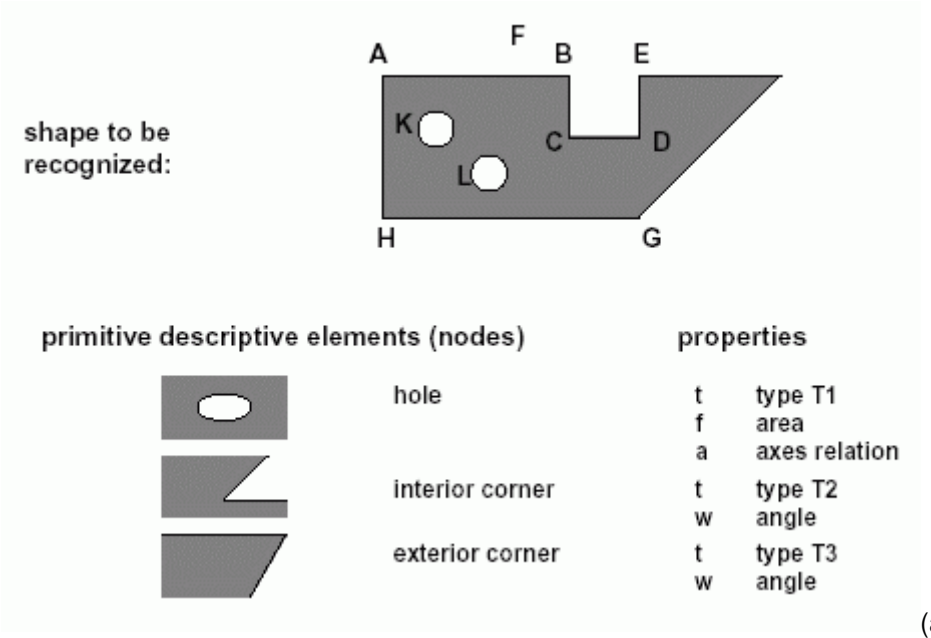


*modellgetriebene Verarbeitung (top-down)*
man beginnt mit den Regeln, die Zielobjekte produzieren, und wendet die Regeln sukzessive von links nach rechts an – Synthese neuer Graphen, bis der synthetisierte Graph isomorph zum Graphen des Eingabebildes oder zu einem seiner Subgraphen wird.



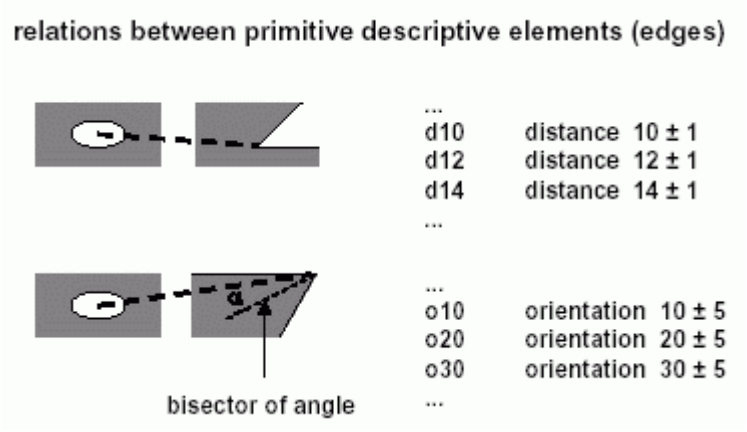

Es sind auch Hybridverfahren aus beiden Verarbeitungs-strategien möglich.

Neben der reinen Graphen-Isomorphie müssen bei der Objekterkennung oft die Attribute in Bild und Modell verglichen werden:



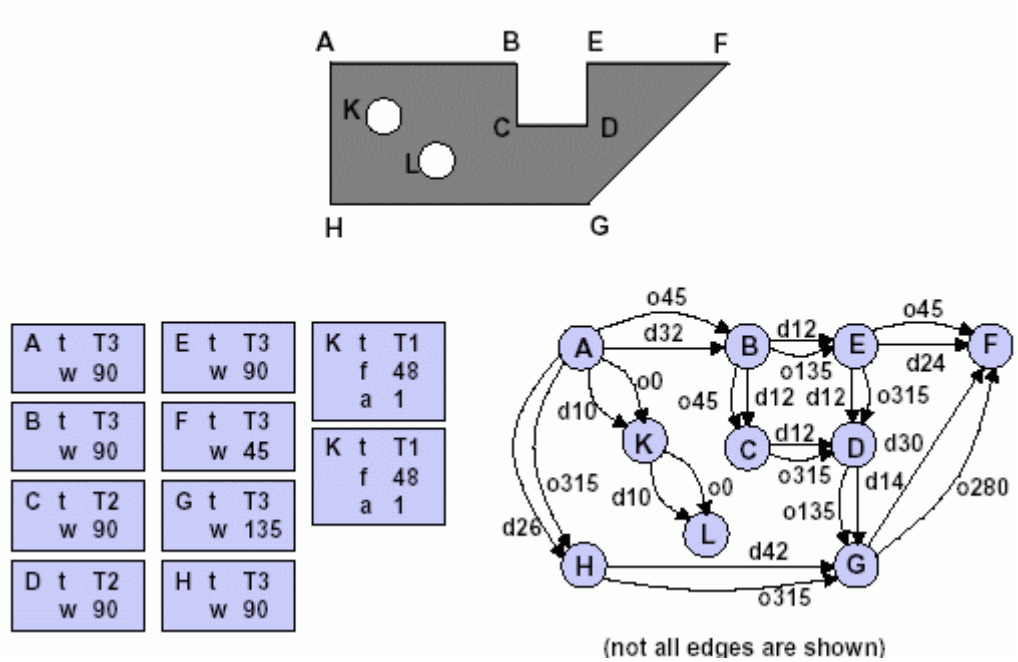

(aus Neumann 2001)

Knotenattribute




Kantenattribute

Darstellung des Modellobjekts als attributierter Graph:

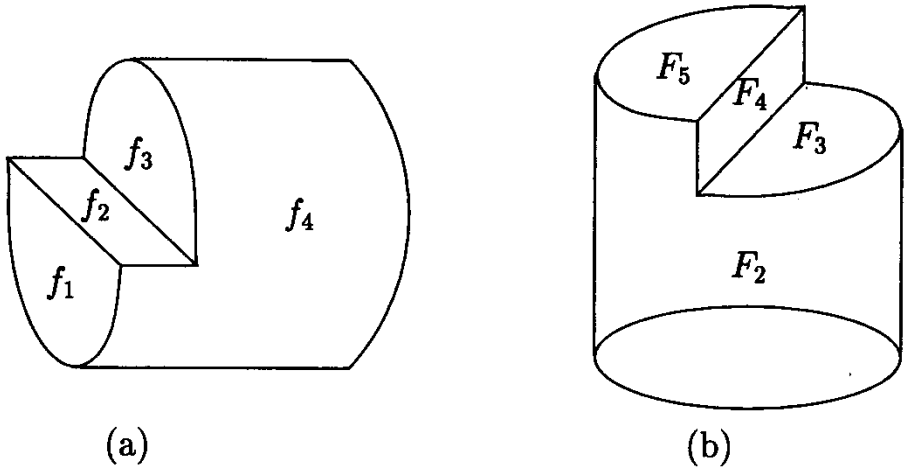



Zur Isomorphie-Feststellung mit einem gegebenen Szenengraphen wird manchmal ein neuer Graph definiert, der *Kompatibilitäts-Graph* (auch *Assoziationsgraph*). Jeder möglichen Knoten-Zuordnung zwischen Modellgraph und Szenengraph entspricht ein Knoten im Kompatibilitäts-Graphen.

- Knoten des Kompatibilitäts-Graphen: Paare mit kompatiblen Knoten-Attributen
- Kanten des Kompatibilitäts-Graphen: konsistente Zuordnungen mit kompatiblen Kanten-Attributen

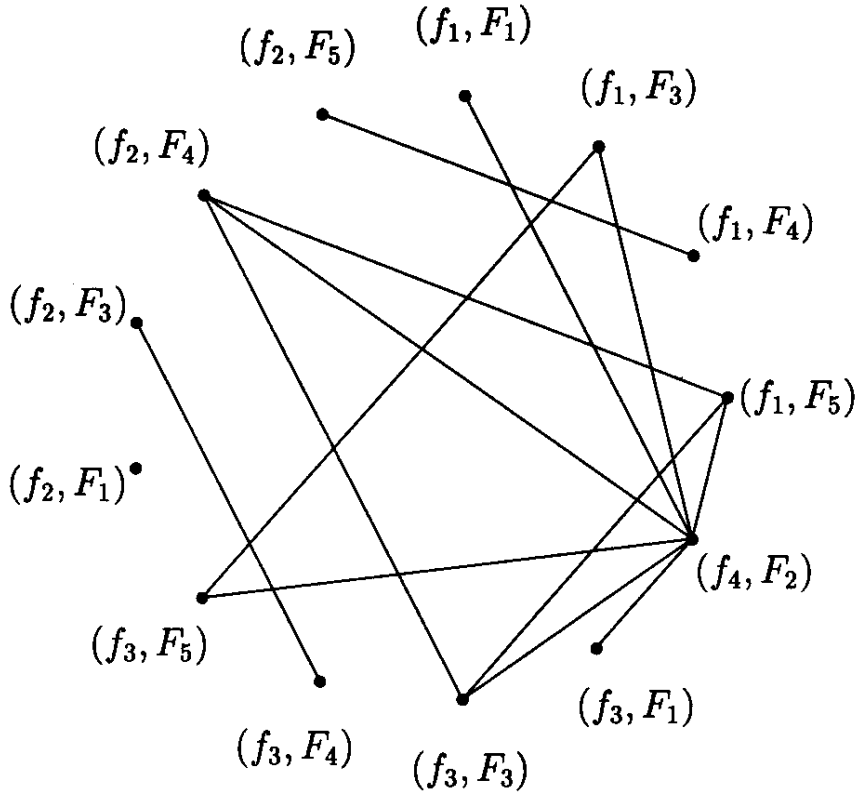
Beispiel:

Szene (a) und Modell (b)



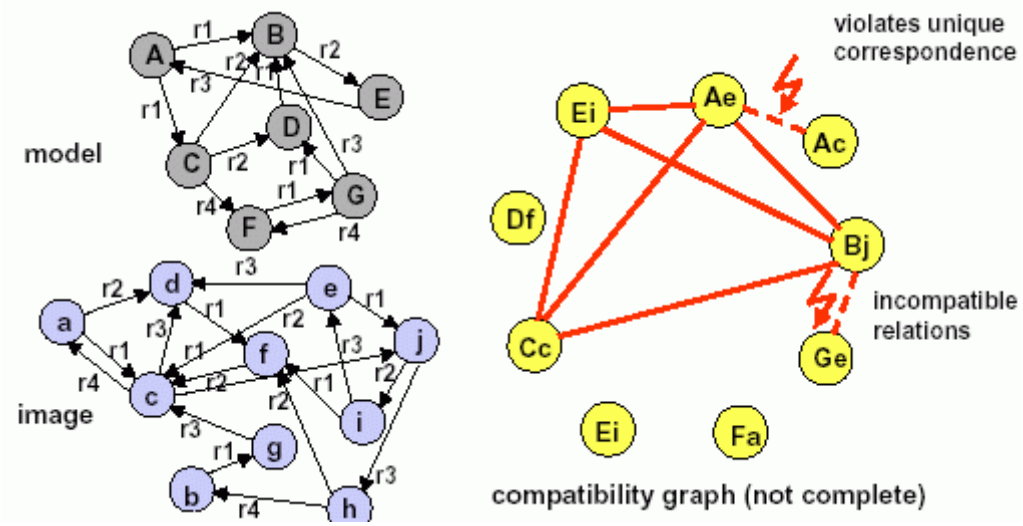

(aus Jiang & Bunke 1997)

zugehöriger Kompatibilitätsgraph:




vollständige Subgraphen (*Cliquen*) im Kompatibilitätsgraphen entsprechen partiellen konsistenten Zuordnungen

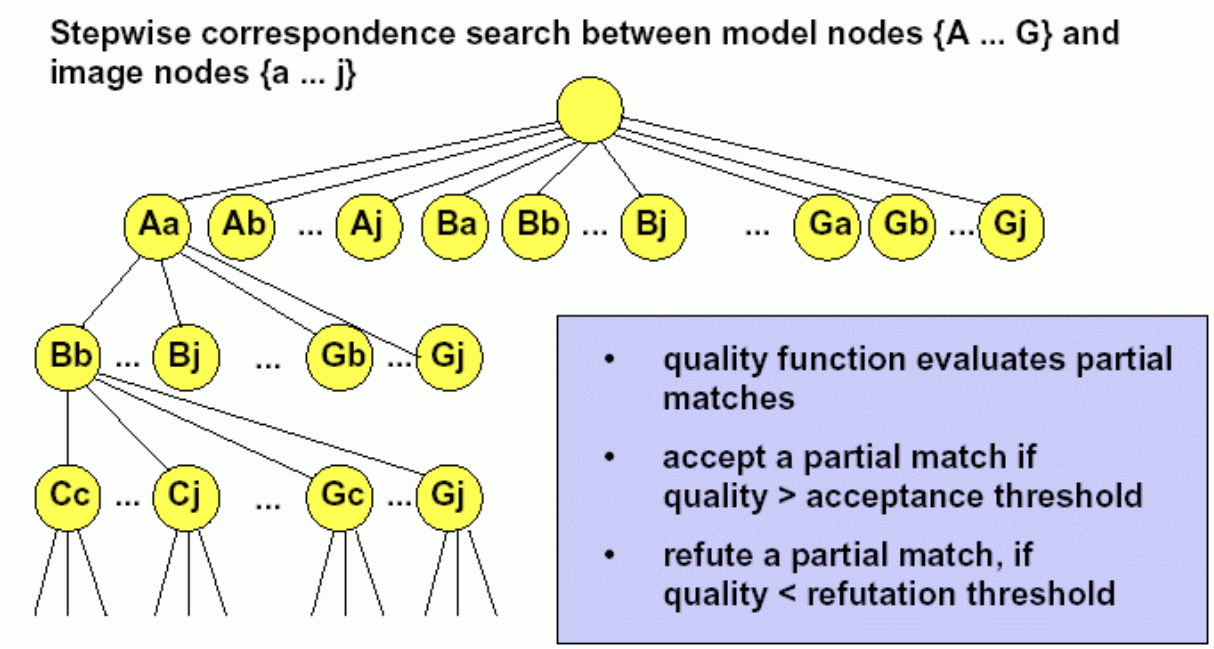
- Annahme: "richtige" Zuordnung entspr. *maximaler Clique* (hier die Clique, die ($f_2$, $F_4$) enthält)




⇒ Matching-Problem wird zurückgeführt auf das Finden maximaler Cliquen im Kompatibilitätsgraphen

- in der Literatur gibt es dafür Algorithmen, z.B. Bron & Kerbusch 1973
- aber: exponentieller Aufwand bzgl. Knotenzahl des Kompatibiltätsgraphen
- effiziente, aber suboptimale Lösungen verwenden heuristische Suchstrategien

Beispiel: relationales Matching mit heuristischer Suche




(aus Neumann 2001)

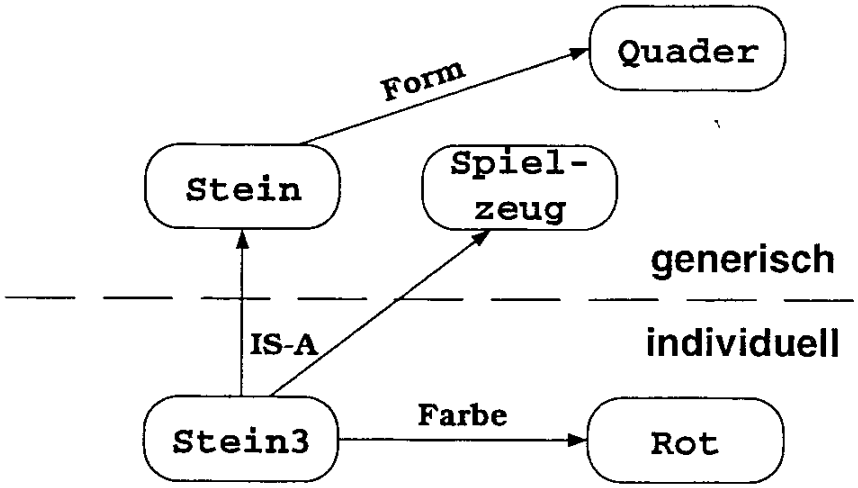
Erweiterung der Objektrepräsentation durch relationale Strukturen:

*Semantische Netze*

(seit 60er Jahren; 70er: M. Minsky; Grundlage mehrerer KI-Sprachen und Expertensysteme)

semantisches Netz = spezialisierte, attributierte relationale Struktur, die auch Prozeduren enthalten kann, um bestimmte Relationen und Attribute zu berechnen, und die vom Vererbungsprinzip der objektorientierten Programmierung Gebrauch macht (nach Levi 2002)

semantische Netze können individuelle und generische Komponenten enthalten:




Vererbung (*inheritance*) erfolgt entlang der IS-A-Kanten (Instanz "Stein3" erbt Eigenschaften von "Stein").

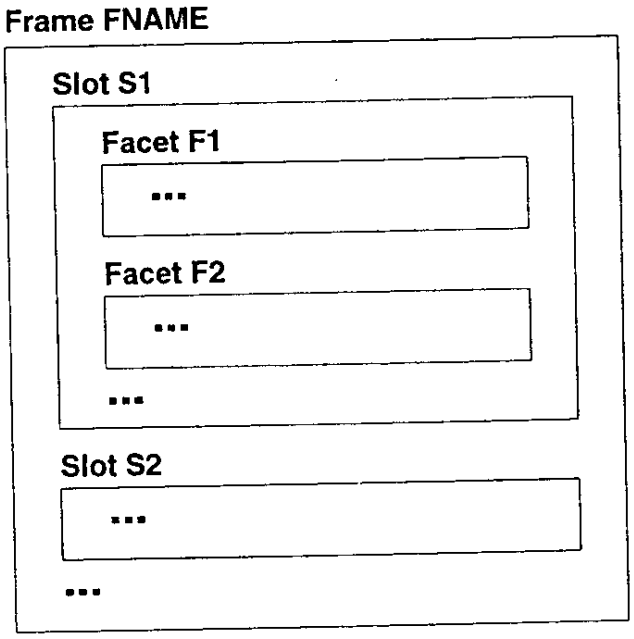
⇒ **`Form(Stein3)`** wird als "**`Quader`**" ausgewertet

weitere Eigenschaften:

- Verwendung von Defaults (vgl. OOP)
- *Dämonen* (demons): Prozeduren, die automatisch dann aufgerufen werden, wenn sie benötigt werden
- als Datenstruktur für die Knoten eines semant. Netzes oft verwendet: *Frames*

*Frame*: Datenstruktur, ähnlich der "Klasse" aus der OOP, jedoch für allgemeine Konzepte und spezielle Instanzen gleichermaßen verwendet

- enthält *Slots* (Merkmale, Relationen) mit zugehörigen *Facets* (Werte-Feldern)




spezielle Slots:

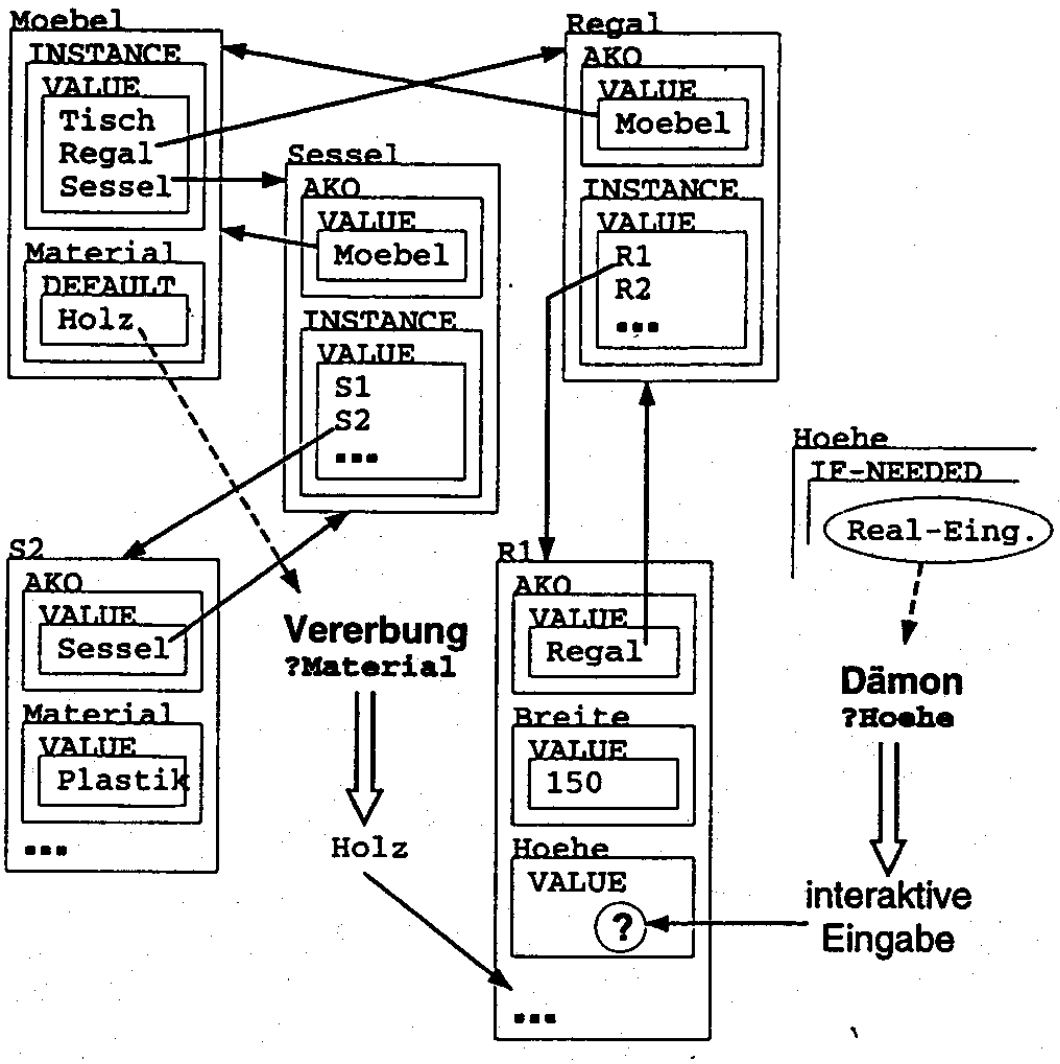
- AKO = a kind of: Generalisierungsrelation, verweist auf Superkonzept, invers zu INSTANCE, zeigt auf anderes Frame (Kante eines semant. Netzes)
- INSTANCE: Spezialisierung, verweist auf Subkonzept, invers zu AKO
- CLASSIFY: hat nur die Werte GENERIC oder INDIVIDUAL, vermerkt, ob generische Klasse ("Konzept") oder Instanz vorliegt
- PART OF: verweist auf übergeordnetes Frame im Sinne *räumlicher* Zugehörigkeit

spezielle Facets:

- VALUE: Wert des Slots
- REQUIRE: Bedingungen für Value (z.B. Wertebereich)
- DEFAULT: Default-Wert
- IF-ADDED, IF-REMOVED, IF-NEEDED: Dämonen

Beispiel (aus Pinz 1994):

```
(FRAME (Moebel (INSTANCE
                  (VALUE (Regal, Sessel, Tisch,...)))
               (Material
                  (DEFAULT (Holz)))
               (CLASSY GENERIC)))
(FRAME (Sessel (AKO (VALUE (Moebel)))
               (CLASSY GENERIC)
               (INSTANCE (VALUE (S1, S2,...)))))
(FRAME (Regal (AKO (VALUE (Moebel)))
              (CLASSY GENERIC)
              (INSTANCE (VALUE (R1, R2,\dots )))
              (Hoehe (IF-NEEDED (Real-Eingabe "Hoehe")))
              ...
              ))
(FRAME (S2 (CLASSY INDIVIDUAL)
           (Material (VALUE (Plastik)))
           (AKO (VALUE (Sessel)))))
(FRAME (R1 (CLASSY INDIVIDUAL)
           (AKO (VALUE (Regal)))
           (Breite (VALUE (150)))))
```



Verbindung zur Prädikatenlogik:

- individuelle Frames (Instanzen) beinhalten nur Aussagen über konstante Objekte und lassen sich in prädikatenlogische Formeln ohne Variablen und Quantoren übertragen
- generische Frames (Konzepte) legen Klassen individueller Objekte fest und können als Formeln mit Allquantoren dargestellt werden

Nachteil der Wissensrepräsentation mit semantischen Netzen:

großer Speicherplatz- und Suchaufwand, Unübersichtlichkeit bei großen Netzen

- deshalb Kombination mit anderen Formen der Wissens-darstellung