Einführung in die künstliche Intelligenz

(Dr. Bauer SS 2005 – verschriftlicht von Georg Kuschk)

Definition Künstliche Intelligenz :

Fachdisziplin der Informatik , deren Ziel es ist , Programme zu entwickeln , die sich verhalten , als verfügten sie über Intelligenz.
Unterscheidung in philosophische „starke KI“ und „schwache KI“ für konkrete Anwendungsprobleme.

1. Aussagenlogik

Es gibt genau zwei Wahrheitswerte : Wahr/Falsch bzw. 1/0

A - Aussage

$w(A)$ - Wahrheitswert von A

Rechnen im Galois-Feld $F_2 = \{0,1\}$ (1+0=1 , 1+1=0 , 1*0=0 , 1*1=1)

Addition und Subtraktion sind identisch ,d.h. 1-0 = 1 , 1-1=0

		Wahrheitswert
Konjunktion	$A \wedge B$	$a \cdot b$
Alternative	$A \vee B$	$a \cdot b + a + b$
Implikation (Subjunktion)	$A \Rightarrow B$	$a \cdot b + a + 1$
Bijektion (Äquivalenz)	$A \Leftrightarrow B$	$a + b + 1$
Negation	$\neg A$	$a + 1$
Disjunktion	$A \oplus B$	$b + 1$
NAND	$\neg(A \wedge B)$	$a \cdot b + 1$
NOR	$\neg(A \vee B)$	$(a+1) \cdot (b+1)$

$A \Rightarrow B$ ist semantisch äquivalent zu $\neg A \vee B$

$A \Leftrightarrow B$ ist semantisch äquivalent zu $(A \Rightarrow B) \wedge (B \Rightarrow A)$

de Morgansche Gesetze :

$$\neg(a \wedge b) = \neg a \vee \neg b$$
$$\neg(a \vee b) = \neg a \wedge \neg b$$

Def. Tautologie : Eine stets wahre Aussagenverbindung.

Def. Normalformen :

- Disjunktive Normalform (DNF) (Eine Disjunktion von Mintermen.)
- Konjunktive Normalform (KNF) (Eine Konjunktion von Maxtermen.)

Ein Minterm m_i ist eine Konjunktion , die jede Eingangsvariable negiert oder nicht negiert enthält.

Ein Maxterm M_i ist eine Disjunktion , die jede Eingangsvariable negiert oder nicht negiert enthält.
(gegenüber Mintermen wird hier jede Variable negiert (*,+ vertausen , sowie top,down))

2. Prädikatenlogik (1.Stufe)

Logik : Sprache $\rightarrow$ Syntax (Muster und Regeln der Sprache)
Semantik (Sinn und Bedeutung der Sprache)
Inferenz (Schlussfolgerung / Konklusion)

2.1 Syntax (Regeln)

Alphabet : alphanumerische Zeichen (Gross- / Kleinschreibung) , Logikzeichen , Rechenoperationen , ...

Wörter :

Objektkonstanten	(beginnen mit Kleinbuchstaben)
Funktionskonstanten	(beginnen mit Kleinbuchstaben)
Prädikatenkonstanten	(beginnen mit Kleinbuchstaben)
Variablen	(beginnen mit Grossbuchstaben)

Terme :

- Jede Objektkonstante und Variable ist ein Term.
- Es sei f eine n-stellige Funktionskonstante und $t_1,...,t_n$ Terme.
  Dann ist $f(t_1,...,t_n)$ ebenfalls ein Term.
- Weitere Terme gibt es nicht.

atomare Sätze/logische Formeln :

Es sei p eine n-stellige Prädikatenkonstante und $t_1,...,t_n$ Terme.
Dann ist $p(t_1,...,t_n)$ ein atomarer Satz.

Bsp :

nicht_verh eiratet(ad am, eva)

ist_kein_s pezialist($\underbrace{\text{eva}}_{\text{Objekt}}$, $\underbrace{\text{Mann}}_{\text{Variable}}$)

Literale :

Atomare Sätze (positive Literale) oder ihre Negation (negative Literale).

Sätze (logisch verknüpfte) :

Es seien s_1 und s_2 atomare Sätze/logische Formeln.
Dann sind $\neg s_1, \neg s_2, s_1 \wedge s_2, s_1 \vee s_2, s_1 \Rightarrow s_2, s_2 \Rightarrow s_1, s_1 \Leftrightarrow s_2$ ebenfalls atomare Sätze / log.F.

quantifizierte Sätze/Formeln :

Es sei s ein Satz und X eine (nicht zwingend in s vorkommende) Variable.
Dann sind $\forall X \; s$ und $\exists X \; s$ ebenfalls Sätze.
(Bzw. $\forall X : s$ und $\exists X : s$)
Analog sind auch hier die logischen Verknüpfungen ebenfalls quantifizierte Sätze.

Bsp 1 : $\exists X (p(X,Y) \Rightarrow r(f(a), X))$
f darf keine Prädikatenkonstante , sondern nur Funktionskonstante sein.

Bsp 2 : $\forall T \exists D$ topf(T) $\wedge$ deckel(D) $\wedge$ passt(T, D)
topf = {john, michael,.. .} deckel = {susi, julia,...}

Beispiele zu „wohlgeformten Sätzen“ :

1.) $\text{liebt}(\text{artu r}, \text{frankreich} \wedge \text{schweiz})$ nein , da atomarer Satz als Argument (P-Logik 2.Stufe)

2.) $\forall \text{land}\ \ \text{nachbar}(\text{fr ankreich}, \text{land})$ nein , da Quantor eine Variable erfordert

3.) $\forall X\ (\ \text{nachbar}(\text{fr ankreich}, \text{schweiz}) \Rightarrow \text{primzahl}(X\ )\ )$ ja

4.) $\forall X\ p(X) \Rightarrow \exists X\ p(X)$

Anmerkung : Der Gültigkeitsbereich einer Variablen ist deren Quantorenbereich.
Im obigen 4. Beispiel sind die beiden Variablen X also verschieden.

weitere Beispiele :

5.) Definiere : $\text{männlich}/1\ \ ,\ \ \text{weiblic h}/1\ \ ,\ \ \text{vegetar ier}/1\ \ ,\ \ \text{fleischer}/\ 1\ \ ,\ \ \text{liebt}/2$

Kein Mann ist sowohl Fleischer als auch Vegetarier.

$\neg\exists X\ \ \text{männlich}(X\ ) \wedge \text{fleischer}(\ X) \wedge \text{vegetarier}\ (X)$

bzw. $\forall X\ \ \neg\text{männlich}(X\ ) \vee \neg\text{fleischer}(\ X) \vee \neg\text{vegetarier}\ (X)$ (de Morgansche Regel)

Die einzigen vegetarischen Fleischer sind weiblich.

$\forall X\ \left(\text{vegetarier}\ (X) \wedge \text{fleischer}(\ X)\right) \Rightarrow \text{weiblich}(X\ )$

Alle Männer außer Fleischern lieben Vegetarier.

$\forall X \forall Y\ \left(\text{männlich}(X\ ) \wedge \neg\text{fleischer}(\ X) \wedge \text{vegetarier}\ (Y)\right) \Rightarrow \text{liebt}(X,\ Y)$

bzw. $\forall X\ (\ \text{männlich}(X\ ) \wedge \neg\text{fleischer}(\ X) \Rightarrow (\forall Y\ \text{vegetarie r}(Y) \Rightarrow \text{liebt}(X,\ Y))\ )$

Kein Mann liebt eine Frau , die Vegetarierin ist.

$\forall X \forall Y\ \left(\text{männlich}(X\ ) \wedge \text{weiblich}(Y\ ) \wedge \text{vegetarier}\ (Y)\right) \Rightarrow \neg\text{liebt}(X,\ Y)$

bzw. $\neg\exists X \exists Y\ \ \text{männlich}(X\ ) \wedge \text{weiblich}(Y\ ) \wedge \text{vegetarier}\ (Y) \wedge \text{liebt}(X,\ Y)$

6.) Redewendungen :

$\neg(\ \forall X\ \text{glänzt}(X)\ \Rightarrow \text{gold}(X)\ )$ bzw. $\exists X\ \text{glänzt}(X)\ \ \wedge \neg\text{gold}(X)$

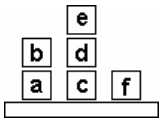
$\forall X\ \text{zögern}(X)\ \Rightarrow \text{verlieren}(\ X)$

Syntax		**Welt**
O – Objektkonstanten		$\tilde{O}$ - Menge von Objekten
F – Funktionskonstanten	$\rightarrow$ Semantik / Interpretation $\rightarrow$	$\tilde{F}$ - Menge von Funktionen (über $\tilde{O}$)
P – Prädikatenkonstanten		$\tilde{P}$ - Menge von Relationen (zwischen Objekten)
V – Variablen		

2.2 Semantik (Bedeutung)

Interpretation und Zuweisung von Wahrheitswerten an Sätze (bezogen auf konkreten Kontext/Welt).

Beispiel :



$\tilde{O} = \{a, b, c, d, e, f\} \quad \cup \{0,1,2\}$

Funktion Höhe $fh : \{a, b, c, d, e, f\} \to \{0,1,2\}$ Erweiterung der Objektmenge um {0,1,2}

$fh(a) = 0 \quad fh(b) = 1 \quad fh(c) = 0 \quad fh(d) = 1 \quad fh(e) = 2 \quad fh(f) = 0$

$\tilde{F} = \{fh\}$

Relationen :

$frei/1 \quad frei = \{b, e, f\} \subseteq \tilde{O}^1$

$tisch/1 \quad tisch = \{a, c, f\} \subseteq \tilde{O}^1$

$auf/2 \quad auf = \{(b,a),(d,c),(e,d)\} \subseteq \tilde{O}^2$

$über/2 \quad über = \{(b,a),(d,c),(e,d),(e,c)\} \subseteq \tilde{O}^2$

Def. Semantik :

O, F, P, V seien durch Syntax gegeben

Betrachte Welt $\tilde{O}, \tilde{F}, \tilde{P}$

Es seien I, U Abbildungen mit folgenden Eigenschaften :

I bijektiv , d.h. eineindeutig

$\forall o \in O: \quad I(o) \in \tilde{O}$

$\forall f \in F: \quad I(f) \in \tilde{F}$

$\forall p \in P: \quad I(p) \in \tilde{P}$

$\forall X \in V: \quad U(X) \subseteq \tilde{O}$

I heißt Interpretation , U heißt Variablenzuweisung

- Es sei t ein Grundterm (d.h. ohne Variablen)

$I(t) = I(f(t_1,...,t_n)) \quad =_{Def} \quad I(f)(I(t_1),...,I(t_n))$

Bsp : syn_a = (syntaktischer) Bezeichner für Objekt a

d.h. $I(syn\_a) = a$

$\Rightarrow I(syn\_fh(syn\_a)) = I(syn\_fh)(I(syn\_a)) = fh(a) = 0$

- p sei n-stellige Prädikatenkonstante und $t_1,...,t_n$ Grundterme.

  Wahrheitswert $WW\big(p(t_1,...,t_n)\big) = \begin{cases} 1 & : \ \big(T_{I,U}(t_1),...,T_{I,U}(t_n)\big) \in T_{I,U}(p) \\ 0 & : \ \text{sonst} \end{cases}$

  Bsp 1 : $WW(syn\_frei(syn\_a)) = T_{I,U}(syn\_frei(syn\_a))$:
  $I(syn\_a) = a$
  $I(syn\_frei) = frei = \{b, e, f\}$
  $a \notin \{b, e, f\}$, d.h. $I(syn\_a) \notin I(syn\_frei)$
  $\Rightarrow WW(syn\_frei(syn\_a)) = 0$

  Bsp 2 :
  $WW(syn\_über(syn\_e, syn\_d))$
  $WW = 1$, falls $\big(T_{I,U}(syn\_e), T_{I,U}(syn\_d)\big) \in T_{I,U}(syn\_über)$
  $\hat{=} \big(I(syn\_e), I(syn\_d)\big) \in I(syn\_über)$?
  $\hat{=} (e,d) \in über$?
  dies ist der Fall , also $WW = 1$

- Seien s_1, s_2 Sätze mit Wahrheitswerten $WW(s_1) = w_1$, $WW(s_2) = w_2$

  Dann ist : $WW(\neg s_1) = 1 - w_1$
  $WW(s_1 \wedge s_2) = w_1 \cdot w_2$
  $WW(s_1 \vee s_2) = w_1 w_2 + w_1 + w_2$
  $WW(s_1 \Rightarrow s_2) = w_1 w_2 + w_1 + 1$
  $WW(s_1 \Leftrightarrow s_2) = w_1 + w_2 + 1$ (siehe Rechenregeln Seite 1)

- Es sei X eine in einem Satz s vorkommende Variable und $\underset{\subseteq O}{\tilde{X}} = U(X)$

  $I^{-1}(\tilde{X})$ = Menge der Objektkonstanten aus O für die Objekte aus $\tilde{O}$, für welche die Variable X Repräsentant sein soll

  $s\,(X/x)$ = Substitution der Variablen X durch eine Objektkonstante aus $I^{-1}(\tilde{X})$

  $WW(\exists X\ s) = \begin{cases} 1: \text{Es gibt eine Substituti on } s(X/x) \text{ in } s \text{ , so daß } WW(s_{s(X/x)}) = 1 \\ 0: \text{sonst} \end{cases}$

  $WW(\forall X\ s) = \begin{cases} 1: \text{für jede (zulässige) Substituti on } s\,(X/x) \text{ gilt } WW(s_{s(X/x)}) = 1 \\ 0: \text{sonst} \end{cases}$

Def. freie Variable :

Eine Variable heißt frei in einem Satz , wenn sie nicht (durch Existenz- oder All-Quantor) quantifiziert wurde.

Def. geschlossener/offener Satz :

Ein Satz heißt geschlossen , wenn alle vorkommenden Variablen quantifiziert sind.
Ansonsten heißt er offen.

Def. Modell :

Interpretation (einschließlich Variablenzuweisung) für einen Satz , welche diesem Satz den Wahrheitswert 1 zuordnet.

I,U heißen Modell für s , wenn $WW(s)=1$ ist. (unter $T_{I,U}$)

Ein Satz heißt :

erfüllbar , wenn $\exists$ Modell

allgemeingültig , wenn $\forall T_{I,U} \quad WW(s)=1$ gilt

unerfüllbar , wenn $\neg\exists$ Modell

Δ : Menge von Sätzen

Δ ist erfüllbar , wenn es ein Modell für jeden Satz gibt.

Δ ist allgemeingültig , wenn für jedes Modell jeder Satz wahr ist.

Δ ist konsistent (im engeren Sinne) , wenn $\exists T_{I,U} : \forall s \in \Delta \quad WW(s)=1$

(Wenn keine widersprüchlichen Aussagen enthalten sind ; für WW(s)=0 muss die Negation enthalten sein)

Δ ist inkonsistent (im engeren Sinne) , wenn $\forall T_{I,U} : \exists s \in \Delta \quad WW(s)=0$

(d.h. wenn mindestens ein Satz aus Δ unerfüllbar ist)

Δ ist inkonsistent (im weiteren Sinne) , wenn Δ inkonsistent bzgl. einem festen $T_{I,U}$ ist.

Bsp : $\forall X \forall Y \quad syn\_auf(X,Y) \Rightarrow syn\_über(X,Y)$

Ist allgemeingültiger Satz , da $syn\_auf \subseteq syn\_über$

Def. logische Implikation :

Sei Δ eine Menge von Sätzen und $\Phi \in \Delta$ ein Satz.

$\Delta \models \Phi$ (Δ impliziert logisch den Satz Φ) gdw. $\forall T_{I,U}$ die Δ erfüllen , ist auch Φ erfüllt.

$(\Rightarrow \Delta \cup \{\neg\Phi\}$ ist unerfüllbar$)$

Def. logische Äquivalenz :

$\Phi \equiv \Psi$ (Φ ist logisch äquivalent zu Ψ) gdw. $\forall T_{I,U}$ gilt $WW(\Phi)=WW(\Psi)$

obiges Beispiel der Würfel :

syn_dazwischen/3 (1. Variable liegt zwischen den zwei anderen (Reihenfolge hierbei egal))

dazwischen = { (d,e,c) , (d,c,e) }

$\forall X \forall Y$ syn_auf$(X,Y) \Rightarrow$ syn_über(X,Y)

$\forall X \forall Y \exists Z$ syn_über$(X,Z) \wedge$ syn_über$(Z,Y) \Rightarrow$ syn_über(X,Y)

$\forall X \forall Y \forall Z \left((\text{syn\_über}(X,Z) \wedge \text{syn\_über}(Z,Y)) \vee (\text{syn\_über}(Y,Z) \wedge \text{syn\_über}(Z,X))\right)$

$\Rightarrow$ syn_dazwischen(X,Z,Y)

Achtung! In Prolog ist die Symmetrie einer Relation **nicht** folgendermaßen definieren :

$\forall X \forall Y \forall Z$ syn_dazwischen$(X,Y,Z) \Rightarrow$ syn_dazwischen(X,Z,Y)

Ein beliebiges Objekt heißt frei , falls kein Objekt existiert , welches auf ihm liegt.

D.h. : $\forall X (\text{frei}(X) \Leftrightarrow \neg\exists Y \text{ auf}(Y,X))$

Ein beliebiges Objekt steht auf dem Tisch , wenn es über keinem anderen liegt.

D.h. : $\forall X (\text{tisch}(X) \Leftrightarrow \neg\exists Y \text{ über}(X,Y))$

2.3 Inferenz (Schlussfolgerung)

gegeben : Δ - Menge von Sätzen und ein Satz Φ

Frage : $?\,\Delta \models \Phi$

Inferenz ist ein Prozess , um aus einer konsistenten Menge von Sätzen die Erfüllbarkeit eines gegebenen Satzes nachzuweisen.

Inferenzregeln :

(1) Bedingung (Prämisse) $\hat{=}$ Menge von Satzstrukturen

---

(2) Konklusion $\hat{=}$ Menge von Satzstrukturen

Inferieren :

Gegeben sei eine Menge von Sätzen , welche in die Struktur der Bedingung passen.
Dann gilt die Menge von Sätzen der Konklusion als abgeleitet.

Randbedingungen für Inferenzregeln :

Korrektheit (Alle abgeleiteten Sätze sind auch logisch impliziert.)
Vollständigkeit (Alle Sätze die logisch impliziert werden sollen auch ableitbar sein.)

Def. Ableitbarkeit :

Sei Δ eine (konsistente) Menge von Sätzen.
Ψ heißt **ableitbar** aus Δ , gdw. eine der folgenden Bedingungen erfüllt ist :

- $\Psi \in \Delta$
- Ψ ist Konklusion einer Inferenzregel , deren Prämisse durch Sätze aus Δ , oder Sätzen die aus Δ abgeleitet wurden , gebildet wird. - siehe $T(\Delta) = \{\Phi : \Delta \models \Phi\}$ - logischer Abschluss

Eine **Ableitung** von Ψ aus Δ ist eine Folge von Sätzen , die entweder zu Δ gehören , oder Konklusionen einer Inferenzregel sind.

Inferenzregeln :

- **Modus Ponens**

$$\frac{\Phi \quad \Phi \Rightarrow \Psi}{\Psi}$$

- **Modus Tollens**

$$\frac{\neg\Psi \quad \Phi \Rightarrow \Psi}{\neg\Phi}$$

(In zweiwertiger Logik überflüssig , da :

$$\Phi \Rightarrow \Psi \quad \equiv \quad \neg\Phi \vee \Psi \quad \equiv \quad \Psi \vee \neg\Phi \quad \equiv \quad \neg\Psi \Rightarrow \neg\Phi$$

$$\frac{\neg\Psi \quad \neg\Psi \Rightarrow \neg\Phi}{\neg\Phi}$$

)

- **UND-Einführung**

$$\frac{\Phi \quad \Psi}{\Phi \wedge \Psi}$$

- **UND-Beseitigung**

$$\frac{\Phi \wedge \Psi}{\Phi}$$
$$\Psi$$

- **Universelle Instanziierung**

$$\frac{\forall X\ \Phi}{\Phi_{s(X/t)}}$$

(X kommt in Φ vor , da sonst kein Sinn)

(Substitution von X durch Struktur t (t frei in Φ - d.h. in t dürfen keine Variablen auftauchen , die im Einzugsgebiet von Quantoren in Φ liegen.))

Bsp. :

$\forall Y\ \text{hasst}(\text{jane}, Y)$

zulässig : Y / jane , $Y / \text{mutter}(\text{jan e})$ (Sofern Mutter kein Prädikat , sondern eine Funktion ist.)

Obige Substitutionen sind unzulässig für folgende Sätze :

$\forall X \exists Y\ \text{hasst}(X, Y)$

$\exists Z\ \text{hasst}(\text{mutt er}(Z), Z)$

- **Existenzielle Instanziierung**

$$\frac{\exists X\ \Psi}{\Psi_{X/\Pi(X_1,\ldots,X_n)}}$$

- Π muss eine Funktionskonstante sein.
- Die Variablen $X_1,\ldots,X_n$ müssen frei sein.
- Anstelle der Variablen sind auch (neue) Objektkonstanten zulässig.

Bsp. :

zulässig : $\forall X \exists Y\ \text{hasst}(X, Y)$ Y / hugo = neue , „imaginäre“ Objektkonstante

unzulässig : $\exists Z\ \text{hasst}(\text{jane}, Z)$ Z / jane

Zusammenfassung :

Logische Implikation (Semantik) wird ersetzt durch Inferenz/Schlüsse ziehen (Syntax).

Beispiel für Inferenz :

explizit formuliertes Wissen :

1) Pferde sind schneller als jeder Hund.
2) Es existiert ein Windhund , der schneller ist als jeder Hase.
3) Harry ist ein Pferd.
4) Ralf ist ein Hase.

implizites Wissen :

5) Jeder Windhund ist ein Hund.
6) Transitivität von „schneller“

zu beweisen/abzuleiten : Ψ : Harry ist schneller als Ralf.

1) $\forall X \forall Y \text{ pferd}(X) \wedge \text{hund}(Y) \Rightarrow \text{schneller}(X, Y)$
2) $\exists X \text{ windhund}(X) \wedge (\forall Y \text{ hase(Y)} \wedge \text{schneller}(X, Y))$
3) pferd(harr y)
4) hase(ralf)
5) $\forall X \text{ windhund}(X) \Rightarrow \text{hund(X)}$
6) $\forall X \forall Y \forall Z \text{ schneller}(X, Y) \wedge \text{schneller}(Y, Z) \Rightarrow \text{schneller}(X, Z)$

zu zeigen : Ψ : schneller(harry, ralf)

7) Existenzielle Instanziierung in 2) X / bernd (d.h. bernd ist ein neues Objekt)
   windhund(b ernd) $\wedge (\forall Y \text{ hase}(Y) \Rightarrow \text{schneller}(\text{bernd}, Y))$
8) UND-Beseitigung in 7)
   windhund(b ernd)
9) UND-Beseitigung in 7)
   $\forall Y \text{ hase}(Y) \Rightarrow \text{schneller}(\text{bernd}, Y)$
10) Universelle Instanziierung in 9) Y / ralf
    hase(ralf) $\Rightarrow$ schneller(bernd, ralf)
11) Modus Ponens auf 10) und 4)
    schneller(bernd, ralf)
12) Universelle Instanziierung in 5) X / bernd
    windhund(b ernd) $\Rightarrow$ hund(bernd)
13) Modus Ponens auf 12) und 8)
    hund(bernd)
14) UND-Einführung auf 13) und 3)
    pferd(harr y) $\wedge$ hund(bernd)
15) Universelle Instanziierung in 1) X / harry Y / bernd
    pferd(harr y) $\wedge$ hund(bernd) $\Rightarrow$ schneller(harry, bernd)
16) Modus Ponens auf 15) und 14)
    schneller(harry, bernd)
17) UND-Einführung auf 16) und 11)
    schneller(harry, bernd) $\wedge$ schneller(bernd, ralf)
18) Universelle Instanziierung in 6) X / harry Y / bernd Z / ralf
    schneller(harry, bernd) $\wedge$ schneller(bernd, ralf) $\Rightarrow$ schneller(harry, ralf)
19) Modus Ponens auf 18) und 17)
    schneller(harry, ralf)

Bsp 2 : Beispiele zum Sequenzenkalkül der Aussagenlogik :

Zeige mittels Sequenzenkalkül : $a \to \neg b\,,\, c \to b\,,\, \neg a \to (b \to \neg c) \;\; \vdash_K \;\; \neg c$

$a \to \neg b\,,\, c \to b\,,\, \neg a \to (b \to \neg c)$	$\vdash_K$	a	(01) Annahme
$a \to \neg b\,,\, c \to b\,,\, \neg a \to (b \to \neg c)$	$\vdash_K$	$a \to \neg b$	(02) Annahme
$a \to \neg b\,,\, c \to b\,,\, \neg a \to (b \to \neg c)$	$\vdash_K$	$\neg b$	(03) Modus Ponens (1,2)
$a \to \neg b\,,\, c \to b\,,\, \neg a \to (b \to \neg c), c$	$\vdash_K$	$\neg b$	(04) Erweiterung
$a \to \neg b\,,\, c \to b\,,\, \neg a \to (b \to \neg c), c$	$\vdash_K$	c	(05) Erweiterung
$a \to \neg b\,,\, c \to b\,,\, \neg a \to (b \to \neg c), c$	$\vdash_K$	$c \to b$	(06) Erweiterung
$a \to \neg b\,,\, c \to b\,,\, \neg a \to (b \to \neg c), c$	$\vdash_K$	b	(07) Modus Ponens (5,6)
$a \to \neg b\,,\, c \to b\,,\, \neg a \to (b \to \neg c)$	$\vdash_K$	$\neg c$	(08) Widerspruch (4,7)

Bsp 3 : Zeige mittels Sequenzenkalkül : $p \to (q \to r)\,,\, p\,,\, \neg r \;\; \vdash_K \;\; \neg q$

$p \to (q \to r)\,,\, p\,,\, \neg r$	$\vdash_K$	p	(01) Annahme
$p \to (q \to r)\,,\, p\,,\, \neg r$	$\vdash_K$	$p \to (q \to r)$	(02) Annahme
$p \to (q \to r)\,,\, p\,,\, \neg r$	$\vdash_K$	$q \to r$	(03) Modus Ponens (1,2)
$p \to (q \to r)\,,\, p\,,\, \neg r, q$	$\vdash_K$	$q \to r$	(04) Erweiterung (3)
$p \to (q \to r)\,,\, p\,,\, \neg r, q$	$\vdash_K$	q	(04) Erweiterung auch auf rechter Seite
$p \to (q \to r)\,,\, p\,,\, \neg r, q$	$\vdash_K$	r	(05) Modus Ponens(4,3)
$p \to (q \to r)\,,\, p\,,\, \neg r, q$	$\vdash_K$	$\neg r$	(06) Annahme
$p \to (q \to r)\,,\, p\,,\, \neg r$	$\vdash_K$	$\neg q$	(07) Widerspruch (5,6)

Bsp 4 : Sequenzenkalkül der Prädikatenlogik

„x ist ein Mensch“ - $m(X)$

„x ist sterblich“ - $s(X)$

„Alle Menschen sind sterblich“ - $\forall X \; m(X) \to s(X)$

„Der Weihnachtsmann ist ein Mensch“ - $m(\text{weihnachts mann})$

Formelmenge $F = \{\forall X \; m(X) \to s(x) \quad , \quad m(\text{weihnachts mann})\}$

zu zeigen : $s(\text{weihnachts mann})$

Herleitung :

(1) F	$\vdash_K$	$\forall X \; m(X) \to s(X)$	Annahme
(2) F	$\vdash_K$	$m(\text{weihnachts mann}) \to s(\text{weihnachts mann})$	Spezialisierung (1) : $x = \text{weihnachts mann}$
(3) F	$\vdash_K$	$m(\text{weihnachts mann})$	Annahme
(4) F	$\vdash_K$	$s(\text{weihnachts mann})$	Modus Ponens (3) (2)

Nachteile der Inferenz :
- Instanziierung gezielt vorgenommen
- freie Auswahl der Inferenzregeln

aus Mängeln lernen $\Rightarrow$

Resolution

Die **Resolution** der Aussagenlogik ist ein Verfahren, um eine Formel *F*, die in KNF vorliegt, auf Erfüllbarkeit und Gültigkeit zu testen.
Resolution wird gewöhnlich nur benutzt, um die Unerfüllbarkeit einer Formel nachzuweisen, da dies meist recht schnell gezeigt werden kann, während der Nachweis der Erfüllbarkeit im WorstCase Exponentialzeit bedarf.
Zum Beweis der Allgemeingültigkeit einer Formel *F*, wird die Unerfüllbarkeit ihrer Negation gezeigt.

- Falls $\Delta \vdash \Psi$, dann ist $\Delta \cup \{\neg\Psi\}$ inkonsistent (Kontradiktion)
- Abschwächung der logischen Äquivalenz auf „Erfüllbarkeitsäquivalenz“
  (Δ und Ψ sind erfüllbarkeitsäquivalent , gdw. $\exists T_{I,U}$, so dass Δ und Ψ gleichzeitig wahr sind.)
- Erfüllbarkeitsäquivalente Umformung der Menge Δ und Satz Ψ $\Rightarrow$ Klauselmengen

zunächst eine Menge an Definitionen :

Def. Klauseln : Menge von Literalen (alternative (bzw. disjunktive) Verknüpfung von Literalen).
($L_1 \cup L_2 \cup L_3 \Rightarrow K = \{L_1, L_2, L_3\}$)

Def. Horn-Klauseln : Klauseln , welche mindestens ein positives Literal enthalten.

Klauseln = Alternativen (bzw. Disjunktionen)
Menge von Klauseln = Konjunktion von Klauseln

Umformung von PL1-Sätzen in eine Menge von Klauseln (KNF)

1.) Beseitigung von $\Rightarrow, \Leftarrow, \Leftrightarrow$

$\Psi \Rightarrow \Phi \quad \equiv \quad \neg\Psi \vee \Phi$

$\Psi \Leftrightarrow \Phi \quad \equiv \quad (\neg\Psi \vee \Phi) \wedge (\neg\Phi \vee \Psi)$

2.) Negation an die Literale heranziehen (über Quantoren hinweg) (de Morgan)

$\neg\exists X\Psi \quad \equiv \quad \forall X\neg\Psi$

$\neg\forall X\Phi \quad \equiv \quad \exists X\neg\Phi$

3.) Mehrfach vorkommende (jedoch unterschiedliche) Variablen umbenennen

Bsp. : $(\forall X\ p(X,X)) \wedge (\exists X\ q(X))$ wird zu $\forall X\ p(X,X) \wedge \exists Y\ q(Y)$

4.) Freie Variablen werden existenz-quantifiziert (Einschränkung der logischen Äquivalenz)

5.) Alle Quantoren nach vorn ziehen (Reihenfolge beachten) = Pränex-Normalform

6.) Skolemisierung aller existenz-quantifizierten Variablen
Ersetzen einer existenz-quantifizierten Variablen durch eine Funktion , die bisher im Satz noch nicht vorkommt. Argumente sind alle all-quantifizierten Variablen , in deren Allquantor-Wirkungsbereichen die existenz-quantifizierte Variable steht.

Bsp. : $\exists U\forall X\forall Y\exists Z\ p(U,X,Y,Z)$ wird zu $\forall X\forall Y\ p(a,X,Y,f(X,Y))$

(U wird durch 0-stellige Funktion ersetzt (=Objektkonstante) , Z wird durch eine 2-stellige Funktion ersetzt , wobei die Belegung von Z abhängt von X und Y .)

Die Einschränkung der Logik besteht darin : $\exists p(X)$ wird zu $p(a)$

$p(X)$ kann evt. für mehrere Belegungen von X gelten , $p(a)$ gilt nur für eine Variable.

7.) Alle verbliebenen (All-)Quantoren weglassen

8.) KNF herstellen (durch Distributivgesetze)

$(a \vee b) \wedge c \quad \equiv \quad a \wedge c \quad \vee \quad b \wedge c$

$(a \wedge b) \vee c \quad \equiv \quad (a \vee c) \wedge (b \vee c)$ in Klauseln : $\{a,c\} \wedge \{b,c\}$

9.) Die KNF als Menge von Klauseln aufschreiben

$(a \wedge b) \vee c \quad \hat{=} \quad \{\{a,c\},\{b,c\}\}$

Beispiele :

1.) $\forall X[(\forall Y\ p(X,y)) \Rightarrow \neg(\forall Y(q(X,Y) \Rightarrow r(X,Z)))]$

Beseitigung der Implikationen :

$\forall X[\neg(\forall Y\ p(X,Y)) \vee \neg(\forall Y(\neg q(X,Y) \vee r(X,Z)))]$

Negationen an die Literale heranziehen :

$\forall X[(\exists Y\ \neg p(X,Y)) \vee (\exists Y(q(X,Y) \wedge \neg r(X,Z)))]$ ($\wedge$ wegen de Morgan)

Mehrfach vorkommende Variablen umbenennen (hier Y zu V) :

Freie Variablen existenz-quantifizieren (hier Z) :

Quantoren nach vorne ziehen :

$\forall X\exists Y\exists V\exists Z[\neg p(X,Y) \vee q(X,V) \wedge \neg r(X,Z)]$

Skolemisierung aller existenz-quantifizierten Variablen :

$\forall X[\neg p(X, f_1(X)) \vee q(X, f_2(X)) \wedge \neg r(X, f_3(X))]$

Die verbliebenen All-Quantoren weglassen , KNF herstellen und als Menge von Klauseln aufschreiben :

$\{\{\neg p(X, f_1(X)), q(X, f_2(X))\}, \{\neg p(X, f_1(X)), \neg r(X, f_3(X))\}\}$

2.) $\neg\forall X\exists Y\ p(X,Y) \Rightarrow q(X,Y)$
$\neg\forall X\exists Y\ \neg p(X,Y) \vee q(X,Y)$
$\exists X\forall Y\ p(X,Y) \wedge \neg q(X,Y)$
$\forall Y\ p(a,Y) \wedge \neg q(a,Y)$
$p(a,Y) \wedge \neg q(a,Y)$ =KNF
$\{\{p(a,Y)\},\{\neg q(a,Y)\}\}$

3.) $(\neg\forall X\ p(X)) \Rightarrow (\exists X\ p(X))$
$(\forall X\ p(X)) \vee (\exists X\ p(X))$
$(\forall X\ p(X)) \vee (\exists Y\ p(Y))$
$\forall X\exists Y\ p(X) \vee p(Y)$
$\forall X\ p(X) \vee p(f(X))$
$p(X) \vee p(f(X))$
$\{\{p(X), p(f(X))\}\}$

Anmerkungen zu Resolution :
- Die leere Klausel {} ist nicht erfüllbar.
- Um zu überprüfen , ob $\Delta \vdash \Phi$ ableitbar ist , muss $\Delta \cup \{\neg\Phi\} \vdash \{\}$ abgeleitet werden.

Def. Unifikation („gleichmachen zweier Sätze“) :

Prozess zum Überprüfen , ob zwei Sätze Φ und Ψ durch geeignete Substitution ihrer Variablen identisch gemacht werden können.

Def. Substitution :

Eine endliche Menge von Zuordnungen zwischen Variablen und Termen , wobei :
- Jeder Variablen wird höchstens ein Term zugeordnet.
- Keine Variable , der ein Term zugeordnet ist , darf innerhalb eines Termes vorkommen , der einer anderen Variablen zugeordnet wird.

Beispiele :

zulässig : {X/a , Y/f(b) , Z/W} (nacheinander Ersetzung)
unzulässig : {X/g(Y) , Y/f(Z)}

Notationsvereinbarung :

Falls Ψ ein Satz ist , so soll $\Psi_s = \Phi$ der Satz sein , welcher aus Ψ durch die Substitution s entsteht.

Beispiel :

$\Psi = p(X,X,Y,U)$ $s = \{X/a, Y/f(b), Z/W\}$
$\Psi_s = p(a,a,f(b),U)$

Definition :

Seien $\Phi_1,...,\Phi_n$ Sätze. Diese heißen **unifizierbar** , wenn es eine s - Substitution gibt mit

$\Phi_{1s} = \Phi_{2s} = ... = \Phi_{ns}$.

Eine solche Subsitution heißt **Unifikator**.

Beispiel 1 :

$\Phi_1 = \{p(X)\}$, $\Phi_2 = \{p(X), p(f(Y))\}$

$s_1 = \{Y/a , X/f(a)\}$

$\Rightarrow \quad \Phi_1 = \{p(f(a))\} \quad , \quad \Phi_2 = \{p(f(a)), p(f(a))\} = \{p(f(a))\}$

$s_2 = \{X/f(Z) , Y/Z\}$

$\Rightarrow \quad \Phi_1 = \{p(f(Z))\} \quad , \quad \Phi_2 = \{p(f(Z)), p(f(Z))\} = \{p(f(Z))\}$

s_2 ist allgemeiner (weniger restriktiv) , da bei s_1 eine feste Bindung der Variablen vorgenommen wird.

Beispiel 2 :

$\Phi_1 = \{p(a,Y,Z)\}$, $\Phi_2 = \{p(X,b,Z)\}$

$s_1 = \{X/a , Y/b , Z/c\}$ (restriktiv)

$s_2 = \{X/a , Y/b\}$ (allgemeiner)

Def. : Faktor einer Klausel

Sei Φ eine Klausel und $j_1,...,j_n$ die Literale aus Φ.

Sei g der allgemeinste Unifikator von $j_1,...,j_n$.

Dann heißt $\Phi' = \Phi_g$ Faktor von Φ.

Beispiel :

$\Phi = \{p(X), p(Y)\}$ $\quad g = \{X/Y\}$

$\Phi' = \Phi_g = \{p(Y), p(Y)\} = \{p(Y)\}$

Resolution als Inferenzbildung (Resolventenbildung) – ({ } ableiten)

Seien C_1, C_2 Klauseln einer aussagenlogischen Formel, die in KNF vorliegt.
Gibt es ein Literal L, welches in C_1 positiv und in C_2 negativ vorkommt, so ist die Vereinigung beider Klauseln ohne das positive und negative Literal L eine Resolvente C_R.

$$L \in C_1$$
$$\overline{L} \in C_2$$
$$C_R = (C_1 \cup C_2) \setminus \{L, \overline{L}\}$$

Es darf immer nur genau ein Literal resolviert werden. Je nach Ausgangsklauseln ist die Bildung verschiedener Resolventen möglich.

Anders notiert: Aus

$$(A_1 \vee A_2 \vee \ldots \vee A_n) \wedge (B_1 \vee B_2 \vee \ldots \vee B_m \vee \neg A_1)$$

wird auf die Resolvente

$$A_2 \vee \ldots \vee A_n \vee B_1 \vee \ldots \vee B_m$$

geschlossen.

Die Resolvente ist **nicht äquivalent** zu den Ausgangsklauseln.
Die Bedeutung der Resolvente liegt vielmehr darin, dass die Ausgangsklauseln nur dann **beide gleichzeitig erfüllbar** sind, wenn auch die Resolvente erfüllbar ist (notwendige Bedingung).
Gelingt es die leere Klausel zu resolvieren, die stets unerfüllbar ist, ist somit die Unerfüllbarkeit der gesamten Formel gezeigt.

Φ $\quad \Phi \in \Phi' (\Phi' - \text{Faktor von } \Phi) \quad g\Phi' = g\Phi$

Ψ $\quad \Psi \in \Psi' (\Psi' - \text{Faktor von } \Psi) \quad g\Psi' = g\Psi$

und dann soll gelten : $\Phi g = \Psi g$

$\left(\left(\Phi' / \{\Phi\} \right) \cup \left(\Psi' / \{\neg \Psi\} \right) \right) g$

Spezialfälle

$$\frac{\begin{array}{ll} \Phi & j \in \Phi \\ \Psi & \neg j \in \Psi \end{array}}{\left(\Phi / \{j\} \right) \cup \left(\Psi / \{\neg j\} \right)}$$

Beispiele :

1.) $\Delta = \left\{ \{P, Q\}, \{\neg P, R\} \right\} \overset{RB}{\Rightarrow} \left\{ Q, R \right\}$

2.) $\Delta = \left\{ \{p(X), q(X, Y)\}, \{\neg p(A), r(b, Z)\} \right\} \underset{g = \{X / a\}}{\overset{RB}{\Rightarrow}} \left\{ q(a, Y), r(b, Z) \right\}$

Eigenschaften :
- Korrektheit (Resolventen werden auch logisch impliziert)
- Keine Vollständigkeit (Widerlegungsvollständigkeit bleibt erhalten)

Widerlegungsvollständigkeit : Falls die Menge von Klausel inkonsistent ist , so gelingt es , durch Resolution die leere Klausel (unerfüllbar) abzuleiten.

Ausgang : $\Delta \vdash ? \Phi$

Resolution : 1.) $\Delta, \neg\Phi$ in Klauselform

2.) $Kl(\Delta) \cup Kl(\neg\Phi)$

3.) Resolution solange , bis {} entsteht

Die Wissensbasis soll nun so vervollständigt werden , dass Sätze hinzugefügt werden , um die WB abzuschließen.

Prädikatenvervollständigung :

$COMP(\Delta, p) =_{Def}$ Konjunktion von Δ mit den Vervollständigungsformeln für p

D.h. beim Original- Δ ist für neue Objekte nicht entscheidbar , ob ein bereits vorhandenes Prädikat dafür gilt. (nicht alles entscheidbar)

Bei $COMP(\Delta, p)$ ist für alle vorhandenen Objekte alles entscheidbar.

Bsp :

$\Delta = \{ p(a) \}$

1.) $\forall X \; X = a \rightarrow p(X)$

2.) $\forall X \; p(X) \rightarrow X = a$

$$COMP(\Delta, p) = \Delta \wedge \left(\forall X \; p(X) \rightarrow X = a \right) \wedge \left(\forall X \; X = a \rightarrow p(X) \right)$$

Nichtmonotones Schließen

Klassische Logiken wie Aussagen- und Prädikatenlogik verfügen über die **Monotonie-Eigenschaft**.
Diese besagt im Wesentlichen, dass durch Schlussfolgerungen lediglich neues Wissen gewonnen, nicht aber bereits vorhandenes Wissen revidiert werden kann.
Was einmal bewiesen wurde, bleibt in einer monotonen Logik immer gültig , auch dann , wenn man zu einem späteren Zeitpunkt über neue Information verfügt.

Nichtmonotone Logiken ermöglichen eine Revidierung gewonnener Erkenntnisse.
Haben wir aus den Aussagen "Tux ist ein Vogel" und "Vögel können fliegen" geschlossen , dass Tux fliegen kann , so revidieren wir diesen Schluss , wenn wir die zusätzliche Information "Tux ist ein Pinguin" erhalten.

Jedes Problem ist prinzipiell in PL1 beschreibbar.
Aber : Nur endlich viele Sätze $\Rightarrow$ nur näherungsweise Beschreibung

gegeben : Wissensbasis Δ und $\Delta \vdash \Phi$

- Falls der Satz Φ aus der Wissensbasis logisch impliziert wird , so beinhaltet Φ im Vergleich zur Wissensbasis keine prinzipiell neue Information.
- Falls aus Δ ein Satz Φ nicht ableitbar ist , so ist nicht entscheidbar , ob Φ in Wirklichkeit gilt oder nicht.
- Nur ja/nein Antworten , keine graduierten Wahrheitswerte

Im Folgenden 3 Methoden :

1.) Es soll die Möglichkeit bestehen , auf die Negation eines „Grundtermes“ zu schließen , falls sein Original nicht ableitbar ist.
Ist es nicht möglich , einen Satz Φ abzuleiten , so wird die Wissensbasis durch $\neg\Phi$ erweitert.
$\Rightarrow$ abgeschlossene Welt
Falls neues Wissen erworben wird , kann Φ als gültig erkannt werden und $\neg\Phi$ muss aus der Wissensbasis wieder entfernt werden.
D.h. die Wissensbasis soll bezüglich Hinzufügen / Entfernen manipulierbar sein.
2.) Ausnahmebehandlung (Taxonomische Hierarchien)
3.) Wissenbasis unverändert lassen und Menge der Inferenzregeln ändern (Default-Regeln)

1.) Closed-World-Assumption (CWA)

CWA ergänzt die Wissensbasis Δ genau dann durch ein negiertes Grundatom $\neg\Phi$, wenn $\Delta \nvdash \Phi$.
(Alles was nicht abgeleitet werden kann , wird als falsch betrachtet.)

genauer :

Δ : Basismenge von Überzeugungen $\hat{=}$ echte Axiome
$T(\Delta) = \{\Phi : \Delta \vdash \Phi\}$ (logischer Abschluss von Δ)
$\Delta_{vÜ}$: Menge der vorausgesetzten Überzeugungen
$\neg\Phi \in \Delta_{vÜ} \Leftrightarrow \Phi \notin T(\Delta)$ (Φ - atomarer Grundterm)

Definition :

$$CWA(\Delta) =_{Def} \{\Phi : \{\Delta \cup \Delta_{vÜ}\} \vdash \Phi\}$$

Beispiel :

$\Delta = \{p(a) \quad , \quad p(a) \rightarrow q(a) \quad , \quad p(b)\}$
$q(b)$ ist nicht ableitbar , d.h. $\neg q(b) \in \Delta_{vÜ}$ und somit auch $\neg q(b) \in CWA(\Delta)$

Lemma zur Konsistenz :

Falls die Klauselform von Δ konsistent ist und eine Menge von Hornklauseln darstellt , so ist $CWA(\Delta)$ ebenfalls konsistent.

2.) Taxonomische Hierarchien (Taxonomie = Einteilung in Gruppen)

Hierarchisches Ordnen der Objekte in Teilklassen und Vererbung der Eigenschaften in Unterklassen.

Beispiel :

a_H : $ding(egon)$

b_H : $vogel(X) \rightarrow ding(X)$

c_H : $strauss(X) \rightarrow vogel(X)$

d_H : $fliegender\_strauss(X) \rightarrow strauss(X)$

(a,b,c,d = Hierarchie-Ebenen , fallend sortiert)

- $\Delta_H \subseteq \Delta$: Teilmenge von Sätzen , welche die Hierarchie-Ebenen beschreiben (auch zwischen Objekten)
- $\Delta_E \subseteq \Delta$: Sätze zur Beschreibung von Eigenschaften für Objekte innerhalb der Ebenen

Beispiel :

Kein Ding außer Vögel kann fliegen.

a_E : $ding(X) \wedge \neg vogel(X) \rightarrow \neg fliegen(X)$

Alle Vögel außer Strauße können fliegen.

b_E : $vogel(X) \wedge \neg strauss(X) \rightarrow fliegen(X)$

c_E : $strauss(X) \wedge \neg fliegender\_strauss(X) \rightarrow \neg fliegen(X)$

d_E : $fliegender\_strauss(X) \rightarrow fliegen(X)$

Problem :

Ausnahmen explizit getroffen , jedoch nicht vollständig. (z.B. Flugzeug = Ding , keinVogel , kann fliegen)

$\Rightarrow$ Klassifizierung von Objekten als „normal“/“unnormal“ durch Sätze und der Rest (wirkliche Ausnahmen) bekommt Eigenschaften nicht zugewiesen.

Bsp :

$ding(X) \wedge \neg unnormal\_1(X) \rightarrow \neg fliegen(X)$ (bzgl. Ebene a)

$vogel(X) \rightarrow unnormal\_1(X)$

3.) Default-Theorien

Hinzufügen neuer Inferenzregeln

Default Regeln :

$$D = \frac{A(X) : B(X)}{C(X)}$$ $X \triangleq$ „Schemavariable“

A - Vorbedingung , B - Konsistenzannahmen , C - Konsequenz

„Wenn A ableitbar ist und $\neg B$ nicht abgeleitet werden kann , dann leite C ab.“

Beispiel :

$$D = \frac{vogel(X) : fliegt(X)}{fliegt(X)}$$ $fliegt(X)$ = Default-Wert

Wissensbasis : $\{strauss(egon), strauss(X) \rightarrow vogel(X)\}$

Modus Ponens $\Rightarrow$ $vogel(egon)$

Default-Regel $\Rightarrow$ $fliegt(egon)$

Falls neues Wisssen erlangt wird $(\neg fliegt(egon))$, so ist aus der Wissensbasis weniger ableitbar.

$gebraten(egon)$
$gebraten(X) \rightarrow \neg lebend(X)$, d.h. $\neg lebend(egon)$
$D' = \dfrac{vogel(X) : lebend(X)}{fliegt(X)}$

Bsp 2 : $\dfrac{vogel(X) : fliegt(X) \wedge \neg strauss(X)}{fliegt(X)}$, $\dfrac{strauss(X) : \neg fliegt(X)}{\neg fliegt(X)}$

Bsp 3 : $\dfrac{freund(X,Y) \wedge freund(Y,Z) : freund(X,Z)}{freund(X,Z)}$

„Normalerweise sind deine Freunde auch meine Freunde."

3. Modallogik – Wissen und Überzeugungen (knowledge & belief)

Darstellung und Inferenz von Sätzen über das Wissen und die Überzeugungen anderer Agenten.

D.h. - Inferenzfähigkeit , Glauben und Wissen anderer
- Schlüsse ziehen , was andere schließen würden

Unterscheidung zwischen „ich weiß..." und „ich glaube..."

Glauben : Modaloperator $b/2$ (belief)
1. Operand : Agent-Grundterm (Term ohne Variable)
2. Operand : Überzeugung des Agent in Form eines logischen Satzes

Bsp : $b(du \quad , \quad ampel(rot) \rightarrow bremst(vor(du)) \quad)$

Wissen : Modaloperator $k/2$ (knowledge)
$k(a,\Phi) =_{Def} b(a,\Phi) \wedge \Phi$

$\hat{=}$ „Ich weiß , dass für a der Satz Φ gilt , wenn ich glaube , dass für a Φ gilt und Φ tatsächlich gilt."

3.1 Syntax

wohlgeformte Sätze der Modallogik :
1.) Alle PL1 – Sätze
2.) Falls Φ ein PL1 – Satz ohne freie Variable ist und a ein Grundterm (d.h. ein Term ohne Variable) , dann heißt $b(a,\Phi)$ Grundatom der Modallogik.
$k(a,\Phi) =_{Def} b(a,\Phi) \wedge \Phi$ ist dann ebenfalls wohlgeformt.
3.) Alle über logische Operationen erzeugbaren Sätze. (aus wohlgeformten Sätzen)

Beispiele :
1.) $\exists X \; b(r, p(X))$ nicht wohlgeformt , da X in $p(X)$ frei ist und existenz-quantifizierte Sätze oben nicht erwähnt werden.
2.) $b(r_1, b(r_2, p(a)))$ nicht wohlgeformt , da $b(r_2, p(a))$ kein PL1-Satz ist
3.) $b((\exists X \; g(X)), p(a))$ nicht wohlgeformt , da a kein Grundterm ist
4.) $b(r, (\exists X \; p(X)))$ wohlgeformt
5.) $p(a) \rightarrow b(r, p(a))$ wohlgeformt (wegen 3.)

$b(klaus, (\exists X \; schöne\_frau(X))) \; \hat{=}$
„Ich glaube , dass Klaus davon überzeugt ist bzw. weiß , dass es wenigstens eine schöne Frau gibt."

3.2 Semantik (für „Glaubensatome“)

Sei a ein Agent , Δ_a die Wissensbasis von a und r_a die Menge der Inferenzregeln von a .

Dann ist T_a die Menge der vom Agenten a aus Δ_a mittels Inferenzregeln aus r_a ableitbare Sätze. $\hat{=}$ Theorie von a

Im Allgemeinen ist $T_a \neq$ Menge der aus Δ_a logisch implizierten Sätze , da r_a nicht alle Inferenzregeln enthalten muss.

$$\Phi \in T_a \Leftrightarrow \Delta_a \vdash_{r_a} \Phi$$

$$WW(b(a,\Phi)) = \begin{cases} 1: \Phi \in T_a \\ 0: \quad \text{sonst} \end{cases}$$

$WW(b(a,\Phi))$ sei gegeben.

Wann darf Φ durch ein Ψ ersetzt werden , so dass $WW(b(a,\Phi)) = WW(b(a,\Psi))$ gilt ?

Bedingung : $\{\Delta_a \cup \Phi\} \vdash_{r_a} \Psi$

$WW_1 \neq WW_2 \Leftrightarrow \{\Delta_a \cup \Phi\} \nvdash_{r_a} \Psi$

Wenn $WW(b(a,\Phi)) = x$ und $\{\Delta_a \cup \Phi\} \vdash_{r_a} \Psi$, dann ist $WW(b(a,\Psi)) = x$

für $x = 0$ per Definition , d.h. wenn $\Psi \notin T_a$

für $x = 1$:

Wenn gilt : $\Delta_a \vdash_{r_a} \Phi$ und $\{\Delta_a \cup \Phi\} \vdash_{r_a} \Psi$, so ist

$T_a(\Delta_a) = T_a(\Delta_a \cup \Phi)$ und damit

$\Psi \in T_a(\Delta_a \cup \Phi)$

$\Rightarrow \Psi \in T_a(\Delta_a)$, d.h.

$WW(b(a,\Phi)) = 1 \quad \Rightarrow \quad WW(b(a,\Psi)) = 1$

Falls man glauben darf , dass für einen Agenten a die Sätze $\Phi_1,...,\Phi_n$ gelten und falls a in der Lage ist , aus $\Phi_1,...,\Phi_n$ auf einen Satz Φ_{n+1} zu schließen , dann soll geglaubt werden dürfen , dass Φ_{n+1} für a gilt.

Attachment-Regel : Folgt aus den vorausgesetzten Inferenzregeln des Agenten

allgemeinste Form :

$$\begin{array}{l} b(a,\Phi_1) \vee \Psi_1 \\ \ldots \\ b(a,\Phi_n) \vee \Psi_n \\ \Phi_1 \wedge \ldots \wedge \Phi_n \vdash_a \Phi_{n+1} \qquad \text{(Attachment-Regel : Annahme über Inferenzfähigkeit von } a \text{)} \\ \underline{\neg b(a,\Phi_{n+1})} \\ \Psi_1 \vee \Psi_2 \vee \ldots \vee \Psi_n \end{array}$$

bzw. spezieller :

$$\frac{\begin{array}{l} b(a,\Phi_1) \\ \ldots \\ b(a,\Phi_n) \\ \Phi_1 \wedge \ldots \wedge \Phi_n \mid\!\!-_a \Phi_{n+1} \end{array}}{b(a,\Phi_{n+1})}$$

Beispiel 1 :

$b(nora\ ,\ p \rightarrow q)$ I

$b(nora\ ,\ \neg q)$ II

Annahme , dass Nora den Modus Ponens beherrscht.
Es soll gezeigt werden , dass $b(nora\ ,\ \neg p)$.

Aus I folgt $b(nora\ ,\ \neg p \vee q)$ und damit $b(nora\ ,\ \neg q \rightarrow \neg p)$.

$$\frac{\begin{array}{l} b(nora\ ,\ \neg q \rightarrow \neg p) \\ b(nora\ ,\ \neg q) \\ (\neg q \rightarrow \neg p) \wedge \neg q \mid\!\!-_{nora} \neg p \\ \neg b(nora\ ,\ \neg p) \end{array}}{\{\}}$$

(Die Negation des zu Zeigenden wird in die Wissensbasis reingepackt und dann wird die unerfüllbare leere Klausel abgeleitet. -> Widerspruch , also war die reingepackte Negation falsch.)

$WW(b(nora, \neg p)) = 1 \Leftrightarrow \neg p \in \mathrm{T}_{nora}$

Der rechte Teil ist wahr , da nora Modus Ponens beherrscht.

Beispiel 2 :

- Wenn alle Raben schwarz sind , so dürfen wir glauben , dass rudi dies auch weiß.
  $(\forall X\ rabe(X) \rightarrow schwarz(X)) \rightarrow b(rudi\quad , \forall X\ rabe(X) \rightarrow schwarz(X))$
  Dies wird zu :
  $(rabe(egon) \rightarrow schwarz(egon)) \rightarrow b(rudi\quad , \forall X\ rabe(X) \rightarrow schwarz(X))$
  $(\neg rabe(egon) \vee schwarz(egon)) \rightarrow b(rudi\quad , \forall X\ rabe(X) \rightarrow schwarz(X))$
  $(rabe(egon) \wedge \neg schwarz(egon)) \vee b(rudi\quad , \forall X\ rabe(X) \rightarrow schwarz(X))$

- Ist fred ein Rabe , so dürfen wir glauben , dass rudi dies auch weiß.
  $rabe(fred) \rightarrow b(rudi\quad , rabe(fred))$
  Dies wird zu :
  $\neg rabe(fred) \vee b(rudi\quad , rabe(fred))$

Desweiteren möge rudi den Modus Ponens (und die Universelle Instanziierung) berherrschen (attachment).

zu zeigen : Wollen glauben dürfen , rudi wisse , dass fred schwarz ist.

Wissensbasis Δ_{rudi} in Klauselform überführen :

1.) $rabe(egon) \vee b(rudi, \forall X\ rabe(X) \rightarrow schwarz(X))$
2.) $\neg schwarz(egon) \vee b(rudi, \forall X\ rabe(X) \rightarrow schwarz(X))$
3.) $\neg rabe(fred) \vee b(rudi, rabe(fred))$
4.) $\neg b(rudi, schwarz(fred))$

Attachment-Regel : ($X = fred$ Universelle Instanziierung + Modus Ponens)

$(\forall X\ rabe(X) \rightarrow schwarz(X)) \ \wedge rabe(fred) \ \vdash_{rudi} \ schwarz(fred)$

Anwendung der allgemeinen Inferenzregel

a)

1.) $b(rudi\ , \underbrace{\forall X\ rabe(X) \rightarrow schwarz(X))}_{\Phi_1} \ \vee \underbrace{rabe(egon)}_{\Psi_1}$

3.) $b(rudi\ , \underbrace{rabe(fred))}_{\Phi_2} \ \vee \underbrace{\neg rabe(fred)}_{\Psi_2}$

4.) $\neg b(rudi\ , \underbrace{schwarz(fred))}_{\Phi_3} \ \vee \underbrace{\{\}}_{\Psi_3}$

$$\frac{\Phi_1 \wedge \Phi_2 \vdash_{rudi} \Phi_3}{\Psi_1 \vee \Psi_2 \vee \Psi_3} \qquad \text{(Attachment-Regel)} \quad \equiv rabe(egon) \vee \neg rabe(fred) \qquad \textbf{(I.)}$$

b)

2.) $b(rudi\ , \underbrace{\forall X\ rabe(X) \rightarrow schwarz(X))}_{\Phi_1} \ \vee \underbrace{\neg schwarz(egon)}_{\Psi_1}$

3.) $b(rudi\ , \underbrace{rabe(fred))}_{\Phi_2} \ \vee \underbrace{\neg rabe(fred)}_{\Psi_2}$

4.) $\neg b(rudi\ , \underbrace{schwarz(fred))}_{\Phi_3} \ \vee \underbrace{\{\}}_{\Psi_3}$

$$\frac{\Phi_1 \wedge \Phi_2 \vdash_{rudi} \Phi_3}{\Psi_1 \vee \Psi_2 \vee \Psi_3} \qquad \text{(Attachment-Regel)} \quad \equiv \neg schwarz(egon) \vee \neg rabe(fred) \qquad \textbf{(II.)}$$

(I.) und (II.) gilt , d.h.

$(rabe(egon) \vee \neg rabe(fred)) \ \wedge \ (\neg schwarz(egon) \vee \neg rabe(fred))$

$\equiv \underbrace{\neg rabe(fred)}_{falsch\ wegen\ 3.)} \vee \underbrace{(rabe(egon) \wedge \neg schwarz(egon))}_{falsch\ wegen\ 1.)}$???

wollten zeigen :

$\Delta_{rudi} \vdash_{r_{rudi}} b(rudi\ , schwarz(fred)) \quad \Leftrightarrow \quad \Delta_{rudi} \cup \{\neg b(rudi\ , schwarz(fred))\} \vdash_{rudi} \{\}$

$\Rightarrow \Delta_{rudi}$ inkonsistent

3.3 Erweiterung der Modallogik auf eingebettete Überzeugungen :

3.3.1 Syntax

1.) Jeder bisher wohlgeformte Satz ist wohlgeformt.
2.) Falls Φ wohlgeformt und geschlossen ist , und a ein Grundterm , so ist $b(a,\Phi)$ wohlgeformt.
(geschachtelte Sätze)
3.) Logische Verknüpfungen von geschachtelten Sätzen sind wohlgeformt.

Beispiel für jetzt nicht mehr wohlgeformten Satz : $b(a\ ,\ \underbrace{p(X) \rightarrow b(b,\forall Y\ p(Y))}_{\text{wohlgeformt}})$

hingegen : $p(X) \rightarrow b(a\ , b(b,\forall Y\ p(Y)))$ ist wohlgeformt

3.3.2 Semantik

a - Agent

Δ_a Wissen von a = Menge von - PL1-Sätzen (eigenes Wissen) und
- wohlgeformten Sätzen bzgl. des Glaubens über Wissen anderer Agenten

r_a - eigene Inferenzfähigkeit und
- Attachment-Regeln der Agenten , von denen a glaubt etwas zu wissen

T_a - Abschluß von Δ_a unter r_a ,
- eigene Schlüsse ,
- Schlüsse , die sich aus geglaubten Sätzen anderer durch geglaubte Inferenzfähigkeit dieser anderen ergeben

3.3.3 Beweismethoden

zu zeigen : $b(a_1\ , b(a_2,\Phi)) \vdash b(a_1\ , b(a_2,\Psi))$

Wir glauben , daß a_1 glaubt , a_2 wisse Φ.

Und wollen nun ableiten , daß wir glauben dürfen , a_1 glaube , daß dann a_2 auch Ψ wisse.

Beispiel 1 :

gegeben :

1.) $b\big(john\ , b(sam, p) \vee b(sam, q)\big)$
2.) $b(\ john\ , b(sam, p \rightarrow r))$
3.) $b(\ john\ , b(sam, \neg r))$

zu zeigen :

$b(\ john\ , b(sam, q))$

$b(\ john, \Phi_1) \vee \{\}$
$b(\ john, \Phi_2) \vee \{\}$
$b(\ john, \Phi_3) \vee \{\}$
$\Phi_1 \wedge \Phi_2 \wedge \Phi_3 \vdash_{john} \Phi_4 \ \hat{=}\ \big(b(sam, p) \vee b(sam, q)\big) \wedge b(sam, p \rightarrow r) \wedge b(sam, \neg r) \vdash b(sam, q)$
$\neg b(\ john\ , \Phi_4) \vee \{\}$

$\{\}$

neues Inferenzschema :

a)

$b(sam, p)$ $\quad \Phi_1 = p$

$b(sam, p \to r)$ $\quad \Phi_2 = p \to r$

$b(sam, \neg r)$ $\quad \Phi_3 = \neg r$

$\Phi_1 \wedge \Phi_2 \wedge \Phi_3 \mid\!\!-_{sam} \Phi_4$ $\quad \hat{=}\ p \wedge (p \to r) \wedge \neg r \mid\!\!-_{sam} q$

$\neg b(sam, q)$ $\quad \Phi_4 = q$

{} , d.h. $b(sam, q)$ gilt

(sam muss Modus Ponens beherrschen , sowie ableiten können und aus leerer Wissensbasis alles schliessen können.)

b)

analog wie a) , nur dass $\Phi_1 = q$ (Hier muss sam die UND-Beseitigung beherrschen.)

$\Rightarrow$ liefert ebenfalls {}

Damit ist bei john das Attachment erfüllt und das Ausgangsschema liefert {} ,
so dass $b(\, john\, , b(sam, q))$ gilt.

Beispiel 2 : wise-men-puzzle

a_1, a_2 - Agenten

Mindestens einer von ihnen hat einen weißen Punkt auf der Stirn (den er selber nicht sehen kann).
Beide wissen , dass jeder die Stirn des Anderen sieht.

Wir glauben :

1) Wenn a_1 keinen Punkt hat , dann darf a_1 glauben , dass a_2 weiß , dass a_1 keinen weißen Punkt hat. (analog für a_2)

1a) $b(a_1\ ,\ \neg punkt(a_1) \to b(a_2, \neg punkt(a_1)))$

1b) $b(a_2\ ,\ \neg punkt(a_2) \to b(a_1, \neg punkt(a_2)))$

2) Beide wissen , dass mindestens einer von ihnen einen weißen Punkt hat und dass der jeweils andere dies weiß.

2a) $b(a_1\ ,\ b(a_2, punkt(a_1) \vee punkt(a_2)))$

2b) $b(a_2\ ,\ b(a_1, punkt(a_2) \vee punkt(a_1)))$

3) a_2 sagt , er wisse nicht , ob er einen weißen Punkt hat.

$\Rightarrow a_1$ glaubt nicht , dass a_2 von seinem weißen Punkt weiss. (analog für a_2)

3a) $b(a_1\ , \neg b(a_2, punkt(a_2)))$ $\quad \hat{=}\ a_1$ darf glauben , dass für a_2 nicht gilt : „Ich habe einen weißen Punkt.“

3b) $b(a_2\ , \neg b(a_1, punkt(a_1)))$

1a) $b(a_1, \Phi_1) \vee \{\}$

2a) $b(a_1, \Phi_2) \vee \{\}$

3a) $b(a_1, \Phi_3) \vee \{\}$

4) $\neg b(a_1, punkt(a_1)) \quad \equiv \quad \neg b(a_1, \Phi_4) \vee \{\}$ (Negation hinzufügen)

$\Phi_1 \wedge \Phi_2 \wedge \Phi_3 \mid\!\!-_{a_1} \Phi_4$ (Attachment)

$$\equiv \quad \left(\neg punkt(a_1) \rightarrow b(a_2, \neg punkt(a_1))\right) \wedge \left(b(a_2, punkt(a_1) \vee punkt(a_2))\right)$$
$$\wedge \left(\neg b(a_2, punkt(a_2))\right) \mid\!\!-_{a_1} punkt(a_1)$$

$\Psi_1 \vee \Psi_2 \vee \Psi_3 \vee \Psi_4 \quad \equiv \quad \{\}$

D.h. , falls für a_1 das obige Attachment gilt , so ist der hinzugefügte Satz Φ_4 falsch , d.h. es gilt $\neg\Phi_4 = b(a_1, punkt(a_1))$

$\Rightarrow$ Wann gilt das Attachment von a_1 ?

$\Rightarrow$ Was muss a_1 von a_2 als Inferenzfähigkeit voraussetzen ?

$\Rightarrow$ Aus dem Attachment ein neues Inferenzschema bilden :

$\neg punkt(a_1) \rightarrow b(a_2, \neg punkt(a_1)) \quad \equiv \quad b(a_2, \Phi_1) \vee \Psi_1$
$\left(\Phi_1 = \neg punkt(a_1) \quad , \quad \Psi_1 = punkt(a_1)\right)$

$b(a_2, punkt(a_1) \vee punkt(a_2)) \quad \equiv \quad b(a_2, \Phi_2) \vee \Psi_2$
$\left(\Phi_2 = punkt(a_1) \vee punkt(a_2) \quad , \quad \Psi_2 = \{\}\right)$

$\neg b(a_2, punkt(a_2)) \quad \equiv \quad \neg b(a_2, \Phi_3) \vee \Psi_3$
$\left(\Phi_3 = punkt(a_2) \quad , \quad \Psi_3 = \{\}\right)$

Attachment von a_2 :

$\Phi_1 \wedge \Phi_2 \mid\!\!-_{a_2} \Phi_3 \quad \equiv \quad \neg punkt(a_1) \wedge \left(punkt(a_1) \vee punkt(a_2)\right) \mid\!\!-_{a_2} punkt(a_2)$

$\Psi_1 \vee \Psi_2 \vee \Psi_3 \quad \equiv \quad \Psi_1 = punkt(a_1)$

Dies gilt , wenn das Attachment von a_2 gilt – und dies wird vorausgesetzt.

Daraus folgt nun $b(a_1, punkt(a_1))$.

4. Induktion (maschinelles Lernen)

Maschinelles Lernen ist ein Oberbegriff für die "künstliche" Generierung von Wissen aus Erfahrung :
Ein künstliches System lernt aus Beispielen und kann nach Beendigung der Lernphase verallgemeinern.
D.h. es lernt nicht einfach die Beispiele auswendig, sondern es "erkennt" Gesetzmäßigkeiten in den Lerndaten.
So kann das System auch unbekannte Daten beurteilen.

Beispiel :

Karten ziehen (Karten werden nach unbekannter Regel belohnt)
Farben = {karo,herz,pik,kreuz}
Werte = {2,3,4,5,6,7,8,9,10,bude,dame,könig,as}
Beobachtung :
kreuz 4 : Belohnung
pik 7 : Belohnung
kreuz 2 : Belohnung
herz 5 : keine Belohnung
karo bube : keine Belohnung
pik dame : keine Belohnung
$\Rightarrow$ unendlich viele Theorien über die Belohnungs-Regeln möglich (z.B.Wetter schön$\Rightarrow$ Belohnung)

Suchen nach Hypothesen Φ , welche :

1.) die Daten Δ erklären :

$$T \cup \{\Phi\} \vdash \Delta$$

2.) und nicht im Widerspruch zu den Daten Δ und der Theorie T stehen :

$$T \vee \Delta \not\vdash \neg\Phi$$

Generierung der Hypothesen aus dem Vokabular der Theorie.

Vergleich von Hypothesen :

Φ_1 heißt besser als Φ_2 $=_{Def}$

Ist M_1 die Menge der Interpretationen die Φ_1 wahr machen , und M_2 die Menge der Interpretationen die Φ_2 wahr machen , so gilt $M_2 \subseteq M_1$.

5. Planen (Zustände , Zustandsänderungen , Aktionen)

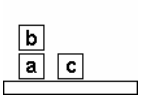
Bisher wurde die Welt durch wahrheitsbewertete Sätze beschrieben , d.h. sie war konstant.

Neu :
- Die Welt ist zu unterschiedlichen Zeitpunkten in unterschiedlichen Zuständen.
- Zustandswechsel durch Aktionen
- Suchen nach einer Folge von Aktionen , welche die Welt vom Ausgangszustand über Zwischenzustände in den Zielzustand überführen.

Beispiel :

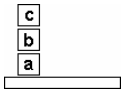
Tisch , 3 Würfel : a,b,c
Originalobjekte : a,b,c
Relationen : tisch/1 , frei/1 , auf/2
Im Zustand s_i gelten die Relationen $tisch_i$, $frei_i$, auf_i

Zustand s_1 :



$tisch_1 = \{a, c\}$, $frei_1 = \{b, c\}$, $auf_1 = \{(b, a)\}$

Zustand s_2 :



$tisch_2 = \{a\}$, $frei_2 = \{c\}$, $auf_2 = \{(c,b),(b,a)\}$

(insgesamt 13 Zustände)

Konzeptualisierung der Welt in ihrer Veränderung
- Zustände als eigenständige Domänen-Objekte
- Funktionen und Relationen über Zustände
- Sätze formulieren , die aussagen , in welchen Zuständen welche Originalobje kte welche Originalrelationen erfüllen

Zur Benennung der Zustände : Einführung von Objektkonstanten
(im Beispiel $s_1,...,s_{13}$)

Def. Zustandsdesignator : Ein Term , welcher einen Zustand beschreibt.

Einführung der Relation state/1 zur Unterscheidung von Originalobjekten und Zuständen.
(Wird wahr für alle Objekte , die Zustände sind.)

Bsp : $state = \{s_1,...,s_{13}\}$
$WW(state(a)) = 0$
$WW(state(s_1)) = 1$

Erweiterung der Originalrelationen um neues Argument :
$tisch/1 \rightarrow tisch/2$
$frei/1 \rightarrow frei/2$
$auf/2 \rightarrow auf/3$
z.B. $auf_1(b,a) \rightarrow auf(b,a,s_1)$
$auf_2(b,a) \rightarrow auf(b,a,s_2)$
$auf_2(c,b) \rightarrow auf(c,b,s_2)$
...

Ergebnis : Gesamte Welt ist in jedem Zustand beschrieben.

Die Originalrelationen werden zu Funktionen über Originalobje kten transformiert , so dass die Bilder der Originalobjekte einer Funktion (z.B. tisch,frei,auf) genau die Menge von Zuständen darstellen , in denen die Originalrelationen für die Originalobjekte gelten.

$tisch, frei, auf : \{a,b,c\} \rightarrow P(\{s_1,...,s_{13}\})$ (Potenzmenge)

z.B. $auf(a,b) = \{s_6, s_7, s_{13}\}$

Einführung einer Relation $t/2$, zur Beschreibung , ob die Originalobjekte $x_1,...,x_n$ in einem bestimmten Zustand s eine bestimmte Eigenschaft rel erfüllen.

$$WW\big(\, t\big(rel(x_1,...,x_n), s\big)\,\big) = \begin{cases} 1 & : \textit{für} \quad s \in rel(x_1,...,x_n) \\ 0 & : \qquad\qquad \text{sonst} \end{cases}$$

D.h. , t ist wahr gdw. die Originalrelation rel im Zustand s für die Objekte $x_1,...,x_n$ gilt.
(t beschreibt somit die Gültigkeit der Originalrelationen für alle Zustände.)

Bsp :

Die Beispielwelt von oben ist im Zustand s_1 , gdw :

$$WW\big(\, t\big(auf(b,a), s_1\big)\,\big) = 1$$
$$WW\big(\, t\big(tisch(a), s_1\big)\,\big) = 1$$
$$WW\big(\, t\big(tisch(c), s_1\big)\,\big) = 1$$
$$WW\big(\, t\big(frei(b), s_1\big)\,\big) = 1$$
$$WW\big(\, t\big(frei(c), s_1\big)\,\big) = 1$$

Def. Zustandsdeskriptor : Menge von Zustandsobjekten.

Axiome :

1.) $\forall D \forall z \quad \big[\, t(\overline{D}, z) \quad \Leftrightarrow \quad \neg t(D, z) \,\big]$

Beweis :

$$WW\big(t(D,z)\big) = 1 \quad \Leftrightarrow z \in D \qquad \Leftrightarrow z \notin \overline{D} \qquad \Leftrightarrow WW\big(t(\overline{D}, z)\big) = 0 \quad \Leftrightarrow WW\big(\neg t(\overline{D}, z)\big) = 1$$

2.) $\forall D_1 \forall D_2 \forall z \quad \big[\, t(D_1 \cap D_2, z) \quad \Leftrightarrow \quad t(D_1, z) \wedge t(D_2, z) \,\big]$

3.) $\forall D_1 \forall D_2 \forall z \quad \big[\, t(D_1 \cup D_2, z) \quad \Leftrightarrow \quad t(D_1, z) \vee t(D_2, z) \,\big]$

4.) $\forall D_1 \forall D_2 \forall z \quad \big[\, \big(D_1 \subseteq D_2 \wedge t(D_1, z)\big) \quad \Rightarrow \quad t(D_2, z) \,\big]$

Def. Zustands-Restriktionen :

Sätze , die , obgleich sie zustandsabhängige Funktionen enthalten , selbst zustandsunabhängig sind.

Beispiel in der Würfelwelt :

- Objekte , welche nicht auf anderen stehen , stehen auf dem Tisch.
  $\forall W_1 \forall Z \quad \big[t(tisch(W_1), Z) \quad \Leftrightarrow \quad \neg \exists W_2 \; t\big(auf(W_1, W_2), Z\big)\big]$
- Objekte sind frei , wenn kein anderes Objekt auf ihnen steht.
  $\forall W_1 \forall Z \quad \big[t(\, frei(W_1), Z) \quad \Leftrightarrow \quad \neg \exists W_2 \; t\big(auf(W_2, W_1), Z\big)\big]$
- Jedes Objekt kann nur auf genau einem anderen stehen.
  $\forall W_1 \forall W_2 \forall W_3 \forall Z \quad \big[t\big(auf(W_1, W_2), Z\big) \quad \Rightarrow \quad \big(t\big(auf(W_1, W_3), Z\big) \Leftrightarrow W_2 = W_3\big)\big]$

5.1 Aktionen

Verbindung zwischen Zuständen und Aktionen : Funktion do

$$do : \{Zustände\} \times \{Aktionen\} \rightarrow \{Zustände\}$$

Operatoren als Instanzen der Aktionen :

$$OPERATOR : \{Objekte\} \rightarrow \{Aktionen\}$$

Die Operatoren beschreiben Fakten , die als Resultat der Anwendung wahr werden.
(Indirekt auch Fakten , die falsch werden.)

Für die Aktionen Axiome aufstellen.

Frame-Problem :

Beschreibung der Merkmale von Zuständen , die bei Aktionen unverändert bleiben.
$\Rightarrow$ Formulierung von Frame-Axiomen (zur Beschreibung der unveränderten Dinge).

Die Anzahl der Frame-Axiome ist proportional zu :
#Aktionen * #Funktionen (zur Zustandsmengenbeschreibung)
$\Rightarrow$ Enormer Aufwand , um die unveränderten Eigenschaften zu beschreiben.

Im Würfel-Beispiel :

Einführung der Operatoren/Aktionen :

$stack/2$, $unstack/2$, $move/3$, $nop/0$

z.B. $do(unstack(c), S_2)$

Aktionen-Axiome :

$stack$:

$$\forall X \forall Y \forall S \left[t(frei(X)) \wedge t(tisch(X)) \wedge t(frei(Y), S) \wedge X \neq Y \right.$$
$$\left. \Rightarrow t(frei(X), do(S, stack(X,Y))) \wedge t(auf(X,Y), do(S, stack(X,Y))) \right]$$

$unstack$:

$$\forall X \forall Y \forall S \left[t(auf(X,Y), S) \wedge t(frei(X), S) \Rightarrow t(frei(X), do(S, unstack(X,Y))) \wedge \right.$$
$$\left. t(tisch(X), do(S, unstack(X,Y))) \wedge t(frei(Y), do(S, unstack(X,Y))) \right]$$

$move$:

$$\forall X \forall Y \forall Z \forall S \left[t(auf(X,Y), S) \wedge t(frei(X), S) \wedge t(frei(Z), S) \wedge X \neq Z \right.$$
$$\left. \Rightarrow t(auf(X,Z), do(S, move(X,Y,Z))) \wedge t(frei(Y), do(S, move(X,Y,Z))) \right]$$

Frame-Axiome :

1.) für Operator $stack$

1.1) Die vor $stack$ nicht-freien Würfel sind auch danach nicht-frei.

$$\forall U \forall X \forall Y \forall S \quad \left[\neg t(frei(U), S) \Rightarrow \neg t(frei(U), do(S, stack(X,Y))) \right]$$

1.2) Was vor $stack$ auf dem Tisch stand und durch $stack$ nicht berührt wurde , steht auch weiterhin auf dem Tisch.

$$\forall T \forall X \forall Y \forall S \quad \left[t(tisch(T), S) \wedge X \neq T \Rightarrow t(tisch(T), do(S, stack(X,Y))) \right]$$

1.3) Was vor $stack$ auf einem anderen Würfel stand , steht auch danach noch drauf.

$$\forall W_1 \forall W_2 \forall X \forall Y \forall S \quad \left[t(auf(W_1, W_2), S) \Rightarrow t(auf(W_1, W_2), do(S, stack(X,Y))) \right]$$

2.) für Operator *unstack*

2.1) Ein Würfel , der vor *unstack* frei war , ist es auch danach.

$$\forall U \forall X \forall Y \forall S \quad [t(frei(U), S) \Rightarrow t(frei(U), do(S, unstack(X, Y)))]$$

2.2) Alle Würfel , die auf dem Tisch standen , tun es auch weiterhin.

$$\forall T \forall X \forall Y \forall S \quad [t(tisch(T), S) \Rightarrow t(tisch(T), do(S, unstack(X, Y)))]$$

2.3) Ein Würfel , der auf einem anderen stand und nicht durch *unstack* berührt wurde , steht auch weiterhin auf diesem.

$$\forall W_1 \forall W_2 \forall X \forall Y \forall S \quad [t(auf(W_1, W_2), S) \Rightarrow t(auf(W_1, W_2), do(S, unstack(X, Y)))] \wedge X \neq W_1$$

3.) für Operator *move*

3.1) Alles , was vor *move* frei war und nicht Ziel für den zu bewegenden Stein war , ist auch weiterhin frei.

$$\forall U \forall X \forall Y \forall Z \forall S \quad [t(frei(U), S) \wedge U \neq Z \Rightarrow t(frei(U), do(S, move(X, Y, Z)))]$$

3.2) Alles , was vorher auf dem Tisch stand , steht auch weiterhin auf dem Tisch. (gdw.)

$$\forall T \forall X \forall Y \forall Z \forall S \quad [t(tisch(T), S) \Leftrightarrow t(tisch(T), do(S, move(X, Y, Z)))]$$

3.3) Jeder Würfel , der auf einem bestimmten Würfel stand und nicht durch *move* bewegt wurde , steht auch weiterhin auf diesem. (gdw.)

$$\forall W_1 \forall W_2 \forall X \forall Y \forall Z \forall S \quad [t(auf(W_1, W_2), S) \Leftrightarrow t(auf(W_1, W_2), do(S, move(X, Y, Z)))] \wedge X \neq W_1$$

4.) für Operator *nop*

$$\forall D \forall S \; [t(D, S) \Rightarrow t(D, do(S, nop))]$$, D = Menge von Zuständen

<u>Zusammengesetzte Aktionen :</u>

- nacheinander , nicht überlappend
- Reihenfolge ist entscheidend

<u>1. unbedingte Aktionsblöcke</u>

- endliche Folge von Aktionen
- keine obere Grenze für die Anzahl der Aktionen eines Blockes
- $\Rightarrow$ Aus jeder nicht-leeren Menge von Aktionen können unendlich viele Blöcke erzeugt werden - $do(S, [])$.

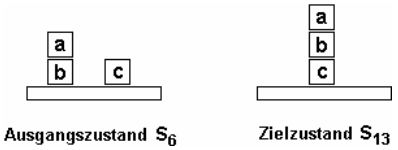
Darstellung : Liste von Termen , jeder Term ist Aktion oder ein weiterer Aktionsblock

Erweiterung von „ *do* “ , um die Aktionsblöcke wie Einzelaktionen in die Formulierung von Sätzen einzubeziehen :

1.) $do(S, []) = S$

2.) $do(S, [A \mid Block]) = do(do(S, A), Block)$

<u>Beispiel :</u>



Ausgangszustand S_6 Zielzustand S_{13}

$$do(S_6 , [unstack(a, b), stack(b, c), stack(a, b)])$$

$$do(stack(a, b), do(stack(b, c), do(unstack(a, b), S_6))) = S_{13}$$

2. bedingte Aktionsblöcke

Aktionen nur dann ausführen , wenn bestimmte Bedingungen erfüllt sind.
Ansätze :

1.) Bedingte Aktionen
2.) Produktionssysteme
3.) Markov-Prozeduren

1.) Bedingte Aktionen

Syntax :

Konditional-Ausdruck $if(\Phi, a, b)$ mit

if - dreistellige Funktion

Φ - Zustandsdeskriptor

a , b - Aktionen

$$if : Potenzmenge(Zustände) \times Aktionen^2 \rightarrow Aktionen$$

$$\forall P \forall S \forall A_1 \forall A_2 \left[t(P,S) \rightarrow do(if(P, A_1, A_2), S) = do(A_1, S) \right]$$
$$\forall P \forall S \forall A_1 \forall A_2 \left[\neg t(P,S) \rightarrow do(if(P, A_1, A_2), S) = do(A_2, S) \right]$$

P - Menge von Zuständen , welche die Bedingung für die Ausführung der Aktion A_1 darstellen

S - aktueller Zustand

A_1, A_2 - alternative Aktionen

Schachtelungen sind zulässig

Semantik :

Gehört der Ausgangszustand zur Zustandsmenge , d.h. Bedingung erfüllt , so wird die erste Aktion , anderenfalls die zweite Aktion ausgeführt.

2.) Produktionssysteme

sequentielle Abarbeitung :

- Auswahl einer Regel , deren Bedingungsteil durch den jeweils aktuellen Ausgangszustand erfüllt wird.
- Ausführung der Regel durch Aktivierung des Aktionsteils.
- Prozedur terminiert , falls keine Regel mehr angewandt werden kann.

Problem : Auswahl der nächsten Regel

formal : Produktionsregel $\Phi \rightarrow a$ mit

Φ - Zustandsdeskriptor

a - Aktionsdesignator

Da die Folge der Regeln endlich ist : Listen als Spezifikationssprache

Einführung eines Prädikats $dictates/3$, um das Resultat eines Produktionssystems zu definieren :
Argumente :

1. Produktionssystem (Liste der Produktionsregeln)
2. Zustand
3. Aktion

$$t(P,S) \rightarrow dictates([P \rightarrow A \mid L], S, A)$$

Falls S ein Zustand ist , der den Bedingungsteil P einer Produktionsregel (genauer : der ersten Regel in der Liste) erfüllt , so führe im Zustand S die Aktion A aus.

$$\neg t(P,S) \wedge dictates(L,S,B) \rightarrow dictates([P \rightarrow A \mid L], S, B)$$

Das Resultat der Ausführung des Produktionssystems ergibt sich aus der Ausführung einer Aktion B , falls der Zustand , in dem das Produktionssystem arbeiten soll , die erste Regel $P \rightarrow A$ nicht zur Ausführung bringen kann (da $S \neq P$) , aber im Rest L des Produktionssystems eine Regel mit Aktionsteil B enthalten ist , die ausgeführt werden kann.

Definition der Wirkung eines Produktionssystems bzw. von *dictates* :

Erweiterung von „ do “ :

1) $\neg\exists A\ dictates(P,S,A) \rightarrow do(P,S) = S$

Falls im Produktionssystem (beschrieben durch P) keine Regel ist , deren Bedingungsteil durch den aktuellen Zustand S erfüllt ist , so soll keine Zustandsänderung stattfinden.

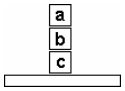
2) $dictates(P,S,A) \rightarrow do(P,S) = do(A,S)$

Das Ergebnis der Anwendung des Produktionssystems P auf den Zustand S ist der Zustand , der sich aus Anwendung der Aktion A , (der ersten Regel aus P , für die S den Bedingungsteil erfüllt) , ergibt.

Beispiel :

Suchen in der Würfelwelt eine Menge von Produktionsregeln , die jeden beliebigen Zustand in den gegebenen Zielzustand überführt.

Zielzustand :



$$\begin{aligned}[\quad & frei(c) \wedge auf(c,X) \Rightarrow unstack(c,X)\ , \\ & auf(b,c) \wedge tisch(c) \wedge frei(b) \Rightarrow stack(a,b)\ , \\ & frei(b) \wedge auf(b,a) \wedge tisch(a) \Rightarrow move(b,a,c)\ , \\ & frei(b) \wedge auf(b,a) \wedge auf(a,c) \Rightarrow unstack(b,a)\ , \\ & tisch(b) \wedge tisch(c) \wedge tisch(a) \Rightarrow stack(b,c)\ , \\ & auf(X_1,c) \wedge frei(X_1) \wedge auf(c,X_2) \Rightarrow unstack(X_1,c) \quad] \end{aligned}$$

Eigenschaften der Produktionssysteme:

- Bedingungen der einzelnen Regeln sind disjunkte Zustandsmengen.
- In jedem Zustand ist mindestens eine Regel anwendbar. (im Zielzustand nop)

Definition Konfliktmenge :

Menge aller in einem Zustand S ausführbaren Produktionsregeln.

Konfliktlösungs-Strategien :

- use first
- einfachste Regel zuerst
- least recently used
- least frequently used

3.) Markov-Prozeduren

Verallgemeinerung von Produktionssystemen

$=_{Def}$ Funktion $f : \{Zustände\} \rightarrow \{Aktionen\}$

Die Funktion f diktiert für jeden beschriebenen Zustand genau eine Aktion A.

Def. Markov-Programm :

Beschreibung einer Markov-Prozedur in einer formalen Programmiersprache (hier : Sprachen der PL1)

$\hat{=}$ Funktion (enkonstante) Π , welche die Prozedur und eine Menge von Sätzen Δ , welche die Prozedur beschreiben , bezeichnet.

Ein Markov-Programm heißt **vollständig** , wenn für jeden Zustand genau eine Aktion definiert ist.

Aufgrund der Komplexität der Welten sind für die KI partielle Markov-Programme charakteristisch.

6. Support-Logik

bisher : Sätze in PL1 (bzw. Einschränkung auf Hornklauseln) : 2 Wahrheitswerte

jetzt : Erweiterung auf Sätze mit Wahrheitswerten $\in [0,1]$
D.h. Abbildung eines Satzes auf einen Wahrscheinlichkeitswert.

Ansätze :

1.) $j : p$ **mit** $p = P(Satz\ j \quad ist\ wahr)$

2.) Belief / Plausibility

Zuordnung von zwei Werten an einen Satz $j : [np, pp]$ mit $0 \leq np \leq pp \leq 1$

np - Sicherheit , daß j wahr ist

pp - Möglichkeit , daß j wahr ist

$j : [np, pp] \quad \Rightarrow \quad \neg j : [1 - pp , 1 - np]$

(Wenn j mit Sicherheit np wahr ist , so bleibt nur noch die Möglichkeit $1 - np$, dass $\neg j$ wahr ist.)

Interpretation der Differenz $pp - np \geq 0$ als Maßzahl für den Mangel an Information über die exakt-wahrscheinlichkeitstheoretische Entscheidung.

$np = pp$: Wahrscheinlichkeit , daß j wahr ist.

$np = 0 \quad , \quad pp = 0$: Fakt j tritt definitiv nicht ein.

$np = 1 \quad , \quad pp = 1$: Fakt j ist in jedem Fall wahr. **(Alle PL1 Sätze werden mit [1,1] bewertet.)**

$np = \frac{1}{2} \quad , \quad pp = \frac{1}{2}$: Fakt j tritt mit 50%iger Wahrscheinlichkeit ein.

Beispiel :

Gericht mit 12 Geschworenen (5 schuldig , 6 unschuldig , 1 unschlüssig)
Schuldspruch$\hat{=}\ p$

$$p:\left[\frac{5}{12},\frac{6}{12}\right] \quad , \qquad \neg p:\left[\frac{6}{12},\frac{7}{12}\right]$$

Bewertung von Regeln :

- $k:-r \quad :[np, pp]$ $\qquad\qquad k:-r \quad \hat{=} \quad k \Leftarrow r$

Interpretation :

np : Sicherheit , daß k bei Erfüllung von r gilt

pp : Möglichkeit , daß k bei Erfüllung von r gilt

- $\neg k:-r \quad :[1-pp\ ,1-np]$ (Komplementarität)

Probleme :

1.) Rumpf einer Regel ist logische Verknüpfung von Literalen , die entweder als bewertete Fakten vorliegen oder abgeleitet werden können.
Gesucht : Bewertung für den gesamten Rumpf (Verrechnung in Disjunktion und Konjunktion)

2.) Gesucht : Modus Ponens
Gegeben :

$$\begin{array}{ll} r & :[n1, p1] \\ k:-r & :[n2, p2] \\ \hline k & :[n, p] \quad ? \end{array}$$

3.) Es gibt mehrere Wege , einen Kopf einer Regel abzuleiten.
Wie sind die Bewertungen der Teilergebnisse zu kombinieren ?

Support-Logic-Programming (SLOP)

Einführung eines neuen Operators „ : „

$$a(X) \coloneqq b(X), c(X):[s_1, s_2]$$
$$b(i):[s_5, s_6]$$
$$c(j):[s_7, s_8]$$
$$d(i):[s_9, s_{10}]$$
$$c(i):[s_{11}, s_{12}]$$

Anfrage : ?- a(X).
gewünschte Antwort : yes X=i [n,p]=[...,...]

D.h. gegeben : $A:[N1, P1]$, $B:[N2, P2]$
gesucht : $A \wedge B:[N, P]$ bzw. $A \vee B:[N, P]$

Definition : Kontingenztafel (zur Berechnung der Sicherheit N)

Kontingenztafeln sind Kreuztabellen absoluter Häufigkeiten bestimmter Merkmalsausprägungen.

Und	$N2$ B	$1-P2$ $\neg B$	$=1-(N2+1-P2)$?? $P2-N2$ $\neg(B\wedge\neg B)$ $\hat{=} B\vee\neg B$ d.h. B ist unklar
$N1$ A	$A\wedge B$ N_1N_2 M_{11}	$A\wedge\neg B$ $N_1(1-P_2)$ M_{12}	A $N_1(P_2-N_2)$ M_{13}
$1-P1$ $\neg A$	$\neg A\wedge B$ $N_2(1-P_1)$ M_{21}	$\neg A\wedge\neg B$ $(1-P_1)(1-P_2)$ $=N(\neg(A\vee B))$ M_{22}	$\neg A$ $(1-P_1)(P_2-N_2)$ M_{23}
$P1-N1$ $\neg(A\wedge\neg A)$ $=1-(N1+1-P1)$??	B $N_2(P_1-N_1)$ M_{31}	$\neg B$ $(1-P_2)(P_1-N_1)$ M_{32}	unklar $(P_2-N_2)(P_1-N_1)$ M_{33}

Bedingungen zur Berechnung von $M_{11}...M_{33}$:
Spalten- und Zeilenaufsummierung ergibt Spalten- bzw. Zeilenkopf.
(Da z.B. erste Zeile = alle jene Felder , wo A vorkommt) = Zeilenkopf

D.h. : $M_{11}+M_{12}+M_{13}$
$=N1N2+N1(1-P2)+N1(P2-N2)$
$=N1(N2+1-P2+P2-N2)$
$=N1$

$M_{21}+M_{22}+M_{23}=1-P1$
... usw .

Problem : 9 Unbekannte , aber nur 6 Gleichungen

1. Variante zur Berechnung :
Optimierungsproblem , d.h. füge noch Ungleichungen zu den 6 Gleichungen hinzu

2. Variante
- Ermittlung prinzipieller Formeln für Disjunktion
- Duch weitere Randbedingungen wird aus prinzipiell tatsächlich.

$\forall$ Satz C gelte : $N(C)=\sum_{\substack{X,Y \text{ beliebige Sätze,}\\ \text{so dass } X\wedge Y\supset C}} N(X\wedge Y)$ (, d.h. $X\wedge Y$ sollen C implizieren)

Beispiel :

- $N(A \vee B) \quad = M_{11} + M_{21} + M_{31} + M_{12} + M_{13}$ (Sicherheit , dass A oder B wahr ist)
  (M's = alljene Felder mit A oder B)

  wegen $M_{11} + M_{21} + M_{31} = N(B) = N2$ folgt

  $= N(B) + M_{12} + M_{13}$

  $= N(B) + M_{12} + M_{13} \quad + M_{11} - M_{11}$

  wegen $M_{11} + M_{12} + M_{13} = N(A) = N1$ folgt

  $= N(B) + N(A) - M_{11}$

  $= N(A) + N(B) - N(A \wedge B)$

- $P(A \vee B)$ (Möglichkeit , dass A oder B wahr ist)

  $= 1 - N(\neg(A \vee B))$

  $= 1 - N(\neg A \wedge \neg B)$

  $= 1 - \big((1 - P1)(1 - P2)\big)$

  $= 1 - \big(1 - P2 - P1 + P1P2\big)$

  $= P1 + P2 - P1P2$

  $= P(A) + P(B) - P(A \wedge B)$

weitere Randbedingungen :
Verhalten von A und B untereinander , d.h. $N(A \wedge B)$ und $P(A \wedge B)$:

Modell a) stochastische Unabhängigkeit von A und B :
N und P sollen sich wie Wahrscheinlichkeitsmaße verhalten.

$N(A \wedge B) = N(A) \cdot N(B)$
$P(A \wedge B) = P(A) \cdot P(B)$
$N(A \vee B) = N(A) + N(B) - N(A) \cdot N(B)$
$P(A \vee B) = P(A) + P(B) - P(A) \cdot P(B)$

Inferenzregeln :

$$\frac{A : [N1, P1] \qquad B : [N2, P2]}{A \wedge B : [N1 \cdot N2 \ , \ P1 \cdot P2] \qquad A \vee B : [N1 + N2 - N1 \cdot N2 \ , \ P1 + P2 - P1 \cdot P2]}$$

Es muss gelten : $0 \le N1 + N2 - N1 \cdot N2 \le P1 + P2 - P1 \cdot P2 \le 1$
Überprüfung :

$0 \le N1 + N2 - N1 \cdot N2 \quad \hat{=} \ 0 \le N1(1 - N2) + N2$

$N1 + N2 - N1 \cdot N2 \le P1 + P2 - P1 \cdot P2 \quad \hat{=}$

$P1 + P2 - P1 \cdot P2 \le 1 \quad \hat{=} \ P1(1 - P2) + P2 \le 1$

Modell b) „konservativ“ – keine Hintergrundinformationen :

$$\max\left(N(A)+N(B)-1\ ,\ 0\right) \ \leq\ N(A\wedge B)\ \leq\ \min\left(N(A),N(B)\right)$$
$$\max\left(P(A)+P(B)-1\ ,\ 0\right) \ \leq\ P(A\wedge B)\ \leq\ \min\left(P(A),P(B)\right)$$

Inferenzregeln :

$$\frac{A:[N1,P1]}{B:[N2,P2]}$$
$$A\wedge B:\left[\max\left(N1+N2-1,0\right)\ ,\ 1-N(\neg A\vee\neg B)\right] \qquad \text{wegen } P(A\wedge B)=1-N\left(\neg(A\wedge B)\right)$$
$$=\left[\max\left(N1+N2-1,0\right)\ ,\ 1-\max\left(N(\neg A),N(\neg B)\right)\right]$$
$$=\left[\max\left(N1+N2-1,0\right)\ ,\ 1-\max\left(1-P(A),1-P(B)\right)\right]$$
$$=\left[\max\left(N1+N2-1,0\right)\ ,\ \max\left(-P(A),-P(B)\right)\right]$$
$$=\left[\max\left(N1+N2-1,0\right)\ ,\ \min\left(P(A),P(B)\right)\right]$$
$$=\left[\max\left(N1+N2-1,0\right)\ ,\ \min\left(P1,P2\right)\right]$$

$$\frac{A:[N1,P1]}{B:[N2,P2]}$$
$$A\vee B:\left[N1+N2-N1\cdot N2\ ,\ 1-N(\neg A\wedge\neg B)\right] \qquad \text{wegen } N(A\vee B)=N(A)+N(B)-N(A\wedge B)$$
$$=\left[N1+N2-\min\left(N1,N2\right)\ ,\ 1-\max\left(N(\neg A)+N(\neg B)-1,0\right)\right]$$
$$=\left[\max\left(N1,N2\right)\ ,\ 1-\max\left(1-P1+1-P2-1,0\right)\right]$$
$$=\left[\max\left(N1,N2\right)\ ,\ 1-\max\left(1-P1-P2,0\right)\right]$$
$$=\left[\max\left(N1,N2\right)\ ,\ \min\left(P1+P2,1\right)\right]$$

Zur Definition von min/max :

$$\min(a,b):=\frac{a+b}{2}-\frac{|a-b|}{2}\ ,\qquad \max(a,b):=\frac{a+b}{2}+\frac{|a-b|}{2}$$

$\frac{a+b}{2}$ = Mittelpunkt des Intervalls , $\frac{|a-b|}{2}$ = Hälfte der Intervalllänge

Modell c) gegenseitiger Ausschluss von A und B :

Inferenzregeln :

$$\frac{A:[N1,P1]}{B:[N2,P2]}$$
$$A\wedge B:[0,0]$$

$$\frac{A:[N1,P1]}{B:[N2,P2]}$$
$$A\vee B:\left[\min\left(N1+N2,1\right),\min\left(P1+P2,1\right)\right]$$

Modell d) strenge Implikation von A auf B bzw. von B auf A : $\Leftrightarrow$

Inferenzregeln :

$$\frac{A:[N1,P1]}{B:[N2,P2]}$$
$$A\wedge B:[\min(N1,N2),\min(P1,P2)]$$

$$\frac{A:[N1,P1]}{B:[N2,P2]}$$
$$A\vee B:[\max(N1,N2),\max(P1,P2)]$$

Zu Modell a) : Modus Ponens :

(N und P sollen sich wie Wahrscheinlichkeitsmaße verhalten.)

$$\frac{A:=B:[N1,P1]\qquad B:[N2,P2]}{A:[N,P]\quad ?}$$

Vorbetrachtungen :

gegeben : $A:=B:[P(A\mid B),P(A\mid B)]$

$A:=\neg B:[P(A\mid\overline{B}),P(A\mid\overline{B})]$

$B:[P(B),P(B)]\qquad\Rightarrow\neg B:[1-P(B),1-P(B)]$

$$P(A)=P(A\mid B)\cdot P(B)+P(A\mid\overline{B})\cdot P(\overline{B})$$

(D.h. $P(A)=P(B\Rightarrow A)\cdot P(B)+P(\overline{B}\Rightarrow A)\cdot P(\overline{B})$)

Betrachte

$$\frac{\begin{array}{ll}A:=B & :[N1,P1]\\ A:=\neg B & :[N2,P2]\\ B & :[N3,P3]\end{array}}{A\qquad :[N,P]}$$

$\Rightarrow\quad\Rightarrow\neg A:=B\quad:[1-P1,1-N1]$

$\Rightarrow\quad\Rightarrow\neg A:=\neg B\quad:[1-P2,1-N2]$ **(*)**

$\Rightarrow\quad\neg B:[1-P3,1-N3]$ (impliziert (*))

$\Rightarrow N(A)=N(A:=B)\quad+\quad N(A:=\neg B)\quad+\quad N(A:=B\ \wedge\ A:=\neg B)$

Die Regeln $A:=B$ und $A:=\neg B$ und die Fakten B und $\neg B$ seien stochastisch unabhängige Ereignisse $\Rightarrow$

$N(A:=B)=N1\cdot N3$

$N(A:=\neg B)=N2\cdot(1-P3)$

$$\Rightarrow N(A)=N1N3+N2(1-P3)$$

Desweiteren gilt :

$N(\neg A:=B)=(1-P1)N3$ und $N(\neg A:=\neg B)=(1-P2)(1-P3)$

$$\Rightarrow P(A)=1-N(\neg A)$$
$$=1-(1-P1)N3-(1-P2)(1-P3)$$

$A := B \quad : [N1, P1]$
$A := \neg B \quad : [N2, P2]$
$B \quad : [N3, P3]$

$A \quad : [N1 \cdot N3 + N2 \cdot (1 - P3) \ , \ 1 - (1 - P1) \cdot N3 - (1 - P2) \cdot (1 - P3)]$

Für $A := \neg B$ fehlt die Bewertung (unklar) $\Rightarrow [N2, P2] = [0,1]$
In das obere Inferenzschema eingesetzt , ergibt dies nun schließlich den **Modus Ponens** :

$A := B : [N1, P1]$
$B \quad : [N2, P2]$

$A \quad : [N1 \cdot N2 \ , \ 1 - (1 - P1) \cdot N2] \qquad = [N1N2 \ , \ 1 - N2 + P1N2]$

Prüfen , ob $0 \le N1 \cdot N2 \le 1 - (1 - P1) \cdot N2 \le 1$ gilt :

- $0 \le N1 \cdot N2$ trivial
- $N1 \cdot N2 \le 1 - (1 - P1) \cdot N2$?
  $\Rightarrow \quad 1 - (1 - P1) \cdot N2 - N1 \cdot N2 \ge 0$?
  $\Rightarrow \quad 1 - N2 \cdot (1 - P1 + N1) = 1 - N2 \cdot (1 - \underbrace{(P1 - N1)}_{\ge 0 \text{ und } \le 1})$
  $\Rightarrow \quad \ge 0$
- $1 - (1 - P1) \cdot N2 \le 1$ trivial

Weiterhin offenes Problem :

Über unterschiedliche Wege seien für denselben Satz Nachweise der Gültigkeit geführt wurden.
Gesucht ist nun eine Kombination der beiden Ableitungsergebnisse $A:[N1,P1]$ und $A:[N2,P2]$
(Die Kombination von mehr als zwei Ableitungsergebnissen wird sukzessive auf zwei zurückgeführt.)

2.Weg	1.Weg und	$N1$ A_1	$1-P1$ $\neg A_1$	$P1-N1$ $\neg(A_1 \wedge \neg A_1) = A_1 \vee \neg A_1$
$N2$	A_2	$A_2 \wedge A_1$ $N1 \cdot N2$ M_{11}	$A_2 \wedge \neg A_1$ $N2(1-P1)$ M_{12}	$A_2 \wedge (A_1 \vee \neg A_1)$ $N2(P1-N1)$ M_{13}
$1-P2$	$\neg A_2$	$\neg A_2 \wedge A_1$ $N1(1-P2)$ M_{21}	$\neg A_2 \wedge \neg A_1$ $(1-P1)(1-P2)$ M_{22}	$\neg A_2 \wedge (A_1 \vee \neg A_1)$ $(1-P2)(P1-N1)$ M_{23}
$P2-N2$	$A_2 \vee \neg A_2$	$A_1 \wedge (A_2 \vee \neg A_2)$ $N1(P2-N2)$ M_{31}	$\neg A_1 \wedge (A_2 \vee \neg A_2)$ $(1-P1)(P2-N2)$ M_{32}	$(A_1 \vee \neg A_1) \wedge (A_2 \vee \neg A_2)$ $(P1-N1)(P2-N2)$ M_{33}

(z.B. $A_2 \vee \neg A_2$ ist keine Tautologie , da nicht mit PL1 bewertet - $A_2 \vee \neg A_2 \mathrel{\hat{=}} A_2$ ist unklar)
Spalten- und Zeilenaufsummierung ergibt weiterhin Spalten- bzw. Zeilenkopf.
Konfliktmasse der gesamten Tafel : $M_{12} + M_{21} \qquad (= N2(1-P1) + N1(1-P2))$
$\Rightarrow$ Normierung der gesamten Tafel mit der Konfliktmasse

$$N(A) = \frac{N(A_{Weg1} \wedge A_{Weg2}) + N(A_{Weg2} \wedge unklar_{Weg1}) + N(A_{Weg1} \wedge unklar_{Weg2})}{1 - Konfliktmasse}$$

$$= \frac{M_{11} + M_{13} + M_{31}}{1 - (M_{12} + M_{21})}$$

$$N = \frac{N1 \cdot N2 + N2(P1 - N1) + N1(P2 - N2)}{1 - (N2(1-P1) + N1(1-P2))}$$

$$= \frac{N1 \cdot N2 + N2(P1 - N1) + N1(P2 - N2)}{1 - Konfliktmasse}$$

$$P = P(A) = 1 - N(\overline{A})$$

$$N(\overline{A}) = \frac{M_{22} + M_{23} + M_{32}}{1 - (M_{12} + M_{21})}$$

$$= \frac{(1-P1)(1-P2) + (1-P1)(P2-N2) + (1-P2)(P1-N1)}{1 - Konfliktmasse}$$

$$= \frac{1 - P2 - P1 + P1P2 + P2 - P1P2 - N2(1-P1) + P1 - P1P2 - N1(1-P2)}{1 - N2(1-P1) - N1(1-P2)}$$

$$= \frac{1 - N2(1-P1) - P1P2 - N1(1-P2)}{1 - N2(1-P1) - N1(1-P2)}$$

$$= \frac{1 - N2(1-P1) - N1(1-P2)}{1 - N2(1-P1) - N1(1-P2)} - \frac{P1P2}{1 - N2(1-P1) - N1(1-P2)}$$

$$= 1 - \frac{P1P2}{1 - Konfliktmasse}$$

$$\Rightarrow P = P(A) = 1 - \left(1 - \frac{P1P2}{1 - Konfliktmasse}\right)$$

7. Unscharfe Mengen , unscharfe Logik (Fuzzy-Logik)

Schließen bei „Unsicherheit/Ungenauigkeit“

allgemeines Schema für Formulierung von Sätzen

O - Menge von Objekten

X, Y - (messbare) Eigenschaften der Objekte

$X : O \to \tilde{X} \subseteq R^n$, $X(o) = x \in \tilde{X}$

$Y : O \to \tilde{Y} \subseteq R^n$

$P(\tilde{X}), P(\tilde{Y})$ - Potenzmengen

$X(o)$ *is* A $\hat{=}$ „Ausprägung der Eigenschaft X für Objekte o liegt in der Menge A.“

, $o \in O$, $A \in P(\tilde{X})$

Unterschiedliche Varianten für A :

1.) $A = \{a\}$ d.h. $|A| = 1$

„ $X(o)$ *is* A“ wahr/falsch

bei wahr : Ausprägung der Eigenschaft X für Objekte o ist exakt festgelegt. (präziser , zweiwertiger Satz)

2.) $|A| > 1$

z.B. X = Körpergröße , o = Otto , $A = [170,180]$

„ $X(o)$ *is* A“ Satz ist entweder wahr oder falsch

Falls der Satz wahr ist , so kann aus dem Wahrheitswert nicht auf die exakte Ausprägung von X für o geschlossen werden. (unpräziser , zweiwertiger Satz)

3.) Bewertete Sätze

„ $X(o)$ *is* A“ mit $[N, P]$, $A \in P(\tilde{X})$

Präzise/unpräzise , zweiwertige Sätze , bei deren Gültigkeit eine gewisse Unsicherheit besteht.

4.) Verallgemeinerung der Wahrheitswerte

A - unscharfe Mengen

„ $X(o)$ *is* A“ - Wahrheitswert $m_A(X(o))$

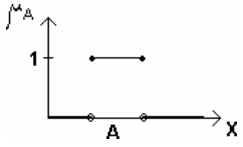
$\hat{=}$ „Grad der Zugehörigkeit des Wertes $X(o)$ in der unscharfen Menge A.“

5.) Verallgemeinerte Wahrheitswerte und bewertete Sätze

Definition char akteristische Funktion einer Menge :

Sei A eine gewöhnliche Menge $A \subseteq X$

charakteristische Funktion von A : $m_A(x) = \begin{cases} 1 & : x \in A \\ 0 & : \text{sonst} \end{cases}$

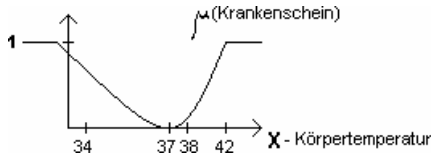


Definition Fuzzy Mengen :

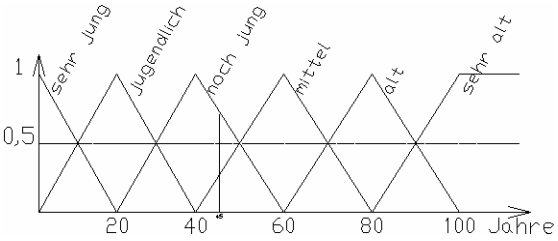
Eine Menge A heißt unscharfe Teilmenge von X , wenn mindestens ein $x \in X$ existiert , mit $m_A(x) \in (0,1)$.

$m_A(x) \hat{=}$ Grad der Zugehörigkeit von x zum Konzept A

Beispiel 1 :



Beispiel 2 : Fuzzyfunktion für das Alter eines Menschen



Definition Trägermenge , Kern :

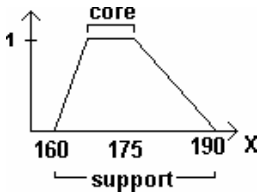
$$\text{support}(A) := \{x : m_A(x) > 0\}$$
$$\text{core}(A) := \{x : m_A(x) = 1\}$$

Beispiel :

Körpergröße mitteleuropäischer Männer

A = mittelgroß (unimodale charakteristische Funktion)

kleinsten und größten Wert suchen , der nicht mehr „mittelgroß ist“.



Definition unimodale Funktion :

Eine Funktion heißt unimodal , wenn jedes lokale Extremum gleichzeitig ein g lobales Extremum ist.

Definition Höhe einer unscharfen Menge A :

$$hgt(A) = \sup_{x \in \tilde{X}} \{m_A(x)\}$$

(= Maximaler Grad der Zugehörigkeit eines Wertes x zum Konzept A.)

Eine unscharfe Menge A heißt **normal** , gdw. $hgt(A) = 1$.

Wahrheitswert für gewöhnliche Mengen A :

$m_A : \tilde{X} \to \{0,1\}$

„$WW(X(o)\ is\ A)$“ $= 1 \quad \Leftrightarrow \quad m_A(X(o)) = 1 \Leftrightarrow X(o) \in A$

$= 0 \quad \Leftrightarrow \quad m_A(X(o)) = 0 \Leftrightarrow X(o) \notin A$

Wahrheitswert für unscharfe Mengen A :

„$WW(X(o)\ is\ A)$“ $=_{Def} \quad m_A(X(o))$

$\hat{=}$ Maß der Kompatibilität des Wertes $X(o)$ mit dem Konzept A

(Möglichkeit , dass der Satz wahr wird.)

7.1 Mengenoperationen mit unscharfen Mengen

1.) Mengenoperationen für gewöhnliche Mengen sollen weiterhin gelten.

2.) Es sollen möglichst viele Eigenschaften der Booleschen Algebra gelten :
$(P(\tilde{X}), \cup, \cap, \neg)$ = Boolesche Algebra

- Assoziativität : $(x \wedge y) \wedge z = x \wedge (y \wedge z)$, $(x \vee y) \vee z = x \vee (y \vee z)$
- Kommutativität: $x \wedge y = y \wedge x$, $x \vee y = y \vee x$
- Idempotenz : $x \wedge x = x$, $x \vee x = x$
- neutrale Elemente : $x \wedge 0 = 0$, $x \vee 0 = x$
  $x \wedge 1 = x$, $x \vee 1 = 1$
- Komplement : $x \wedge \neg x = 0$, $x \vee \neg x = 1$
- Involution : $\neg(\neg x) = x$
- Distributivität : $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$
  $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$
- De Morgan : $\neg(x \wedge y) = \neg x \vee \neg y$, $\neg(x \vee y) = \neg x \wedge \neg y$

Seien A, B unscharfe Mengen mit den charakteristischen Funktionen m_A, m_B :

- Teilmengenbeziehung
  $A \subseteq B \quad \Leftrightarrow \quad \forall x \in \tilde{X} \quad m_A(x) \leq m_B(x)$
- Komplementbildung
  $\overline{A} =_{Def}$ unscharfe Menge mit $m_{\overline{A}}(x) = 1 - m_A(x) \quad , \forall x \in \tilde{X}$

- <u>Vereinigung und Durchschnitt</u>

Varianten (siehe S.35 ff) :

	$m_{A\cup B}$	$m_{A\cap B}$	
1.	$\max\left(m_A(x)\,,\,m_B(x)\right)$	$\min\left(m_A(x)\,,\,m_B(x)\right)$	- strenge Implikation
2.	$\min\left(1\,,\,m_A(x)+m_B(x)\right)$	$\max\left(0\,,\,m_A(x)+m_B(x)-1\right)$	- keine Hintergrundinformation
3.	$m_A(x)+m_B(x)-m_A(x)\cdot m_B(x)$	$m_A(x)\cdot m_B(x)$	- stochastische Unabhängigkeit

1. Variante erfüllt alle Eigenschaften der Booleschen Algebra
2. Variante erfüllt nicht die Eigenschaft Idempotenz , da $A\cup A\neq A$ immer ein Trapez ergibt
3. Variante erfüllt nicht die Eigenschaften Idempotenz , Distributivität , De Morgansche Gesetze

<u>Bsp :</u>

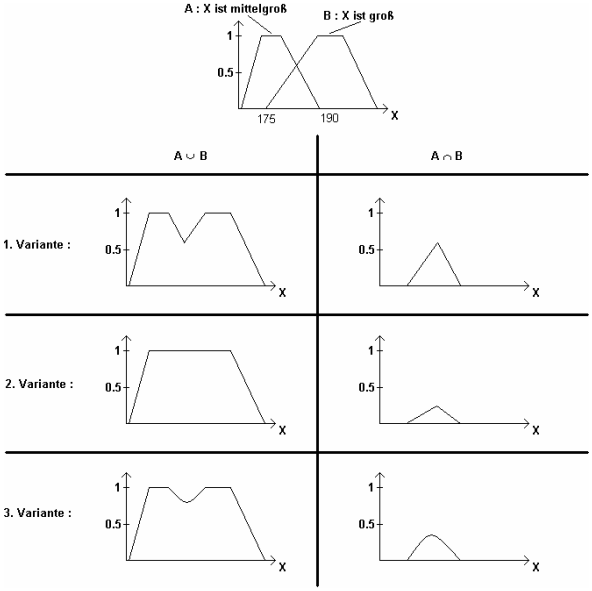
Überprüfung der Distributivität bei der 3. Variante :

$(A\cup B)\cap C \quad (x) \quad =?\quad (A\cap C)\cup(B\cap C)$

linke Seite : $m_{A\cup B}\cdot m_C = m_C\left(m_A+m_B-m_A\cdot m_B\right)=m_Am_C+m_Bm_C-m_Am_Bm_C$

rechte Seite : $m_{A\cap C}+m_{B\cap C}-m_{A\cap C}\cdot m_{B\cap C}=m_Am_C+m_Bm_C-m_Am_Cm_Bm_C$

$\Rightarrow \quad \neq$



$$m_{A\cup B}^{Variante1}\leq m_{A\cup B}^{Variante3}\leq m_{A\cup B}^{Variante2}$$
$$m_{A\cap B}^{Variante1}\geq m_{A\cap B}^{Variante3}\geq m_{A\cap B}^{Variante2}$$

D.h. Variante 1 ist von allen Varianten bzgl. der Vereinigung minimal und bzgl. des Durchschnitts maximal.

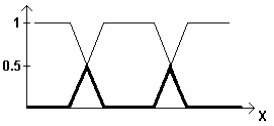
Problem bei der 1. Variante : $m_{A\cap\overline{A}}(x) = \min\left(m_A(x)\,,1-m_A(x)\right)$

$hgt(A\cap\overline{A}) = 0.5$, d.h. $A\cap\overline{A}\neq\varnothing$.

Wegen :

$$\begin{aligned} hgt(A\cap\overline{A}) &= \sup_{x\in\tilde{X}}\left\{m_{A\cap\overline{A}}(x)\right\} \\ &= \sup\left\{\min\left(m_A(x), m_{\overline{A}}(x)\right)\right\} \\ &= \sup\left\{\min\left(m_A(x)\,,1-m_A(x)\right)\right\} \\ &= \sup\left\{\frac{m_A(x)+1-m_A(x)}{2}-\frac{\left|m_A(x)-(1-m_A(x))\right|}{2}\right\} \\ &= \sup\left\{\frac{1}{2}-\frac{\left|2m_A(x)-1\right|}{2}\right\} \end{aligned}$$

= maximal für $\left|2m_A(x)-1\right|=0$, d.h. für $m_A(x)=\frac{1}{2}$



7.2 Zugehörigkeitsfunktionen von mehrdimensionalen Fuzzy-Mengen(/Relationen)

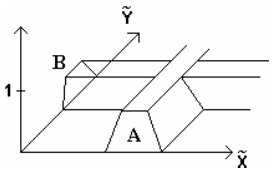
$A\times B\subseteq\tilde{X}\times\tilde{Y}$ m_A, m_B Zugehörigkeitsfunktionen

$m_{A\times B}(x,y) =_{Def} \min\limits_{x,y}\left(m_A(x)\,,\,m_B(y)\right)$

Zylindrische Erweiterung von A auf $\tilde{X}\times\tilde{Y}$:

$Zyl(A)$: $m_{A\times\tilde{Y}}(x,y) =_{Def} m_A(x)$

$Zyl(B)$: $m_{\tilde{X}\times B}(x,y) =_{Def} m_B(x)$



$\Rightarrow m_{A\times B}(x,y) = m_{(A\times\tilde{Y})\cap(\tilde{X}\times B)}$ (im einfachsten Fall Pyramidenstumpf)

WW („ $X(o)$ *is* A und/oder $Y(o)$ *is* B “) :

	und	oder
1. Variante	$\min\left(m_A(X(o))\,,\,m_B(Y(o))\right)$	$\max\left(m_A(X(o))\,,\,m_B(Y(o))\right)$
2. Variante	...	...
3. Variante	... siehe weiter oben	...

7.3 Modus Ponens in Fuzzy-Logik:

- Implikation

„*if* $X(o)$ *is* A *then* $Y(o)$ *is* B“

allgemeiner : allquantifiziert „*if* X *is* A *then* Y *is* B“

Wahrheitswert soll durch Zugehörigkeitsfunktion beschrieben werden : $\forall x \in \tilde{X}\ \forall y \in \tilde{Y}\ m_{A\to B}(x,y)$.

$$\frac{\begin{array}{l} X(o) \text{ is } A \\ \text{if } X \text{ is } A \text{ then } Y \text{ is } B \end{array}}{Y(o) \text{ is } B}$$

gesucht : Zugehörigkeitsfunktion m_B von B mit

$$m_B(y) = t - norm\left(m_A(x)\ ,\ m_{A\to B}(x,y)\right)$$

- beschreibt den Prämissenteil des Modus Ponens
- beschreibt die konjunktive Verknüpfung der beiden gegebenen Sätze

$m_{A\to B}(x,y)$ - Verteilung von Zugehörigkeiten / Möglichkeiten auf $\tilde{X}\times\tilde{Y}$

Schätzung von m_B , indem $t - norm\left(m_A(x)\ ,\ m_{A\to B}(x,y)\right)$ auf $\tilde{Y}$ projiziert wird.

(D.h. , den „Pyramidenstumpf“ auf die „Leinwand“ $\tilde{Y}$ projizieren.)

$$\Pr\left(m_{A\times B}(x,y)\right) = \sup_{x\in\tilde{X}}\left(m_{A\times B}(x,y)\right)$$

$$\Rightarrow \forall y \in \tilde{Y}\ m_B(y) \le \sup_{x\in\tilde{X}}\left(t - norm\left(m_A(x)\ ,\ m_{A\to B}(x,y)\right)\right)$$

Forderung : „vernünftiges“ Verhalten des Modus Ponens

d.h. $\forall y \in \tilde{Y}\ m_B(y) = \sup_{x\in\tilde{X}}\left(t - norm\left(m_A(x)\ ,\ m_{A\to B}(x,y)\right)\right)$

Dies wird erreicht , indem $t - norm$ und $m_{A\to B}$ entsprechend gewählt werden :

$$t - norm =_{Def} \min$$

$$m_{A\to B}(x,y) =_{Def} \begin{cases} 1 & : \ m_A(x) \le m_B(y) \\ m_B(y) & : \ sonst \end{cases}$$ (Lukasiewicz-Norm)

Hintergrund :

PL1 : $p \to q$

$$WW(p\to q) = \begin{cases} 1 & p \le q \\ 0 & sonst \end{cases} = \begin{cases} 1 & p \le q \\ WW(q) & sonst \end{cases} = WW\left(\neg p \vee q\right)$$

$$\Rightarrow m_B(y) = \sup_{x\in\tilde{X}}\left(\min\left(\ m_A(x)\ ,\ m_{A\to B}(x,y)\right)\right)$$

Verallgemeinerter Modus Ponens :

$$\frac{A' \qquad A \rightarrow B}{B'}$$

Ansatz : $m_{B'}(y) =_{Def} \sup_{x \in \tilde{X}} \left(\min(m_A(x) , m_{A \rightarrow B}(x, y)) \right)$

Eigenschaften :

1.) $A' \subseteq A \quad \Rightarrow B' = B$

2.) $\forall A', A \quad \Rightarrow B \subseteq B'$

3.) $a =_{Def} \sup \{ m_{A'}(x) : m_A(x) = 0 \}$

$\Rightarrow \forall y \in \tilde{Y} \; m_{B'}(y) \geq a$ („support“ von B' auf dem Niveau a)

$\hat{=}$ Größte Möglichkeit , bei der der Fakt erfüllt ist , die Prämisse aber verletzt wird.
(Grad der Undeterminiertheit)

4.) $a =_{Def} \inf \{ m_A(x) : m_{A'}(x) = 1 \}$ (core von B')

$m_{B'}(y) = 1 \quad \forall y \in \tilde{Y}$ mit $m_B(y) \geq b$

$\hat{=}$ Kleinste Möglichkeit , die Prämisse zu erfüllen , wobei der Fakt erfüllt ist.

8. PROLOG

PROgramming in LOGic

a , b Konjunktion
a ; b Disjunktion
b :- a Implikation
\+a not(a)

true liefert immer Yes
fail liefert immer No

regelkopf :- regelrumpf

Anfragen :
?- regelrumpf.
?- bruder (X,Y) , vater (X,Z).
?- mann (X).
?- weiblich (klara).

Beispiel 1 :

```
gerd.
peter.
krank(gerd).
gesund(X) :- not(krank(X)).

?- gesund(gerd).
No
?- gesund(peter).
Yes
```

Mit trace :

```
?- trace.
Yes

[trace] 11 ?- gesund(gerd).

   Call: (7) gesund(gerd) ? creep
^  Call: (8) not(krank(gerd)) ? creep
   Call: (9) krank(gerd) ? creep
   Exit: (9) krank(gerd) ? creep
^  Fail: (8) not(krank(gerd)) ? creep
   Fail: (7) gesund(gerd) ? creep
No
```

Beispiel 2 :

```
fak(0,1).
fak(X,Fakultaet) :- X>0 , Y is X-1 , fak(Y,Z) , Fakultaet is Z*X.

?- fak(5,X).
X = 120
Yes
```

Zunächst wird X pro Schritt um 1 erniedrigt (X>0) , dann wird multipliziert.
Die Iteration liegt in fak(Y,Z)

fak(5,X) -> fak(4,Z) -> ... -> fak(0,Z)	2.Regel
Unifikation Z=1 -> yes	1.Regel
Fakultaet is 1*1	die zuletzt aufgerufene 2.Regel
Fakultaet is 1*2	... usw. wieder nach oben arbeiten
Fakultaet is 2*3	
Fakultaet = 6*4	
Fakultaet = 24*5	

Cut - in Prolog eingebautes Prädikat Cut : !
(nimmt eine endgültige Belegung aller nachfolgenden Variablen vor)
(Achtung , bei mir (SWI-Prolog) werden die Klauseln von links nach rechts ausgewertet !)

Beispiel 1 :

```
q(X) :- a , ! , p(X).
a.
p(c).
p(b).

?- q(X).
X=c     ;
X=b     ;
No
                                    ;          (neue Belegungen suchen , bereits gefundene ignorieren)
```

Beispiel 2 :

```
q(X) :- p(X) , ! , a.
a.
p(c).
p(b).

?- q(X).
X=c     ;
No
```

Wirkungsweise des Cut-Operators :
Eine Regel, die einen Cut enthält, hat die Form p :- q, !, r.
Rein logisch gesehen ist der Cut bedeutungslos, da er immer beweisbar ist.
Der Cut beeinflusst jedoch das back-tracking-Verhalten des Prolog-Systems und ist daher nur prozedural zu verstehen:

Wenn das Literal p bewiesen werden soll , dann läuft der Beweis wiefolgt ab :
Zunächst werden immer die Literale in q bewiesen. Dafür gibt es zwei Möglichkeiten :

1. q kann nicht bewiesen werden. Dann wird die nächste Klausel zum Beweis von p ausprobiert, oder, falls sie nicht vorhanden ist, ein Backtrackingschritt ausgeführt.
   In diesem Fall hat der Cut also keine Wirkung.

2. q kann bewiesen werden. In diesem Fall ist das Literal p *nur dann* beweisbar, wenn r beweisbar ist, und zwar mit den Variablenbindungen, die durch den Beweis von q entstanden sind.
   Falls r also nicht beweisbar ist, so wird keine alternative Klausel für p und auch kein alternativer Beweis für q ausprobiert.
   Der Cut schneidet also alle alternativen Beweismöglichkeiten für p , die vor dem Auftreten des ! noch vorhanden waren, ab.“

Beispiel :

```
bluete(a).
blume(a).
res(X) :- blume(X),!,bluete(X).

?- res(X).
X=a     Yes



bluete(b).
blume(a).
res(X) :- blume(X),!,bluete(X).

?- res(X).
No
```

Unterschied "is" bzw "=" :

X = Y. Es wird versucht X und Y gleichzumachen.
X /= Y. X ist ungleich Y.

Bsp :

sokrates = sokrates
27=3*9
78=78

X is 56. Der Variablen wird eine Zahl (berechenbarer Ausdruck) zugewiesen.

Bsp :

X is Y/3.
X is 36 mod 15.

Bsp :

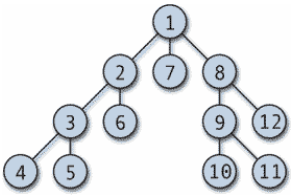
produkt(X,Y,Z) :- Z is X*Y.

Backtracking

= Problemlösungsmethode

Backtracking geht nach dem Versuch-und-Irrtum-Prinzip (*trial and error*) vor, d.h. es wird versucht eine erreichte Teillösung schrittweise zu einer Gesamtlösung auszubauen.
Wenn absehbar ist, dass eine Teillösung nicht zu einer endgültigen Lösung führen kann, wird der letzte Schritt bzw. die letzten Schritte zurückgenommen und stattdessen alternative Wege probiert.
Auf diese Weise ist sichergestellt, dass alle in Frage kommenden Lösungswege ausprobiert werden können.

Backtracking arbeitet nach dem Prinzip der Tiefensuche : (Nummerierung nach Folge der besuchten Knoten.)



Details zum Backtracking

Mit der Wissenbasis

p(a).
p(b).
c:-p(X).

reagiert Prolog auf die Anfrage ?- c. mit yes.
Ein Trace zeigt, dass sich das System mit der Ersetzung p(a) zufriedengibt. b wird also nicht in p eingesetzt.
Das liegt daran, dass das Prädikat c nullstellig ist.
Schreibt man c(X):-p(X), so werden alle Möglichkeiten für X getestet und ausgegeben.

Resolution, Backtracking und Unifikation in PROLOG

Im folgenden sollen die Prinzipien an konkreten Beispielen anschaulich erarbeitet werden.
Dazu wird folgende Wissensbasis vorausgesetzt :

```
maennlich(paul).
maennlich(fritz).
maennlich(steffen).
maennlich(robert).
weiblich(karin).
weiblich(lisa).
weiblich(maria).
weiblich(sina).

vater_von(steffen, fritz).
vater_von(fritz, karin).
vater_von(steffen, lisa).
vater_von(paul, maria).
mutter_von(karin, maria).
mutter_von(sina, paul).

elternteil_von(X,Kind):- vater_von(X,Kind).
elternteil_von(X,Kind):- mutter_von(X,Kind).
```

Matchen

An das System wird die Entscheidungsfrage ?– maennlich(steffen). gestellt.

PROLOG vergleicht diese Anfrage mit den in der Wissensbasis gespeicherten Fakten und Regeln, wobei zunächst geprüft wird, ob der Funktor mit gleicher Stelligkeit vorhanden ist. In diesem Fall führt es zu den Fakten:

```
maennlich(paul).
maennlich(fritz).
maennlich(steffen).
maennlich(robert).
```

Nun werden das in der Frage vorkommende Term steffen mit denen der Fakten nacheinander verglichen.

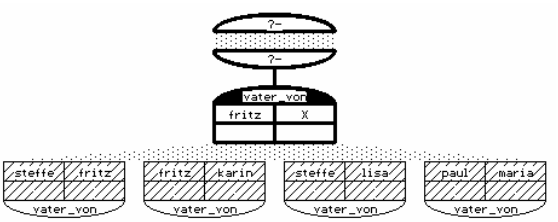
Findet PROLOG so wie hier eine Übereinstimmung, ergibt sich die Antwort yes, man spricht man vom Matchen.
Sollte in der gesamten Datenbasis keine Übereinstimmung vorhanden sein, so erscheint no.

Unifikation eines Terms mit einer ungebundenen Variable

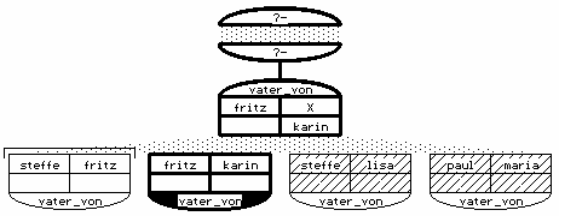
An das System wird die Ergänzungsfrage ?– vater_von(fritz,X). gestellt.

Wiederum werden alle Fakten und Regeln benutzt, die den gleichen Funktor vater_von besitzen.

```
vater_von(steffen, fritz).
vater_von(fritz, karin).
vater_von(steffen, lisa).
vater_von(paul, maria).
```



Nun wird versucht, das erste Argument der Anfrage fritz mit steffen zu matchen. Dies hat aber keinen Erfolg.
Damit kommt der zweite Fakt an die Reihe. Hier kommt es zur Übereinstimmung der jeweils ersten Argumente und es erfolgt die Bindung des Argumentes karin an die freie Variable X.
Diese wird damit gebunden (instanziert). Diesen Vorgang bezeichnet man als Unifikation oder Instanziierung.
Da es keine weitere freie Variable gibt, hat das System eine Lösung gefunden und gibt sie als X = karin aus.



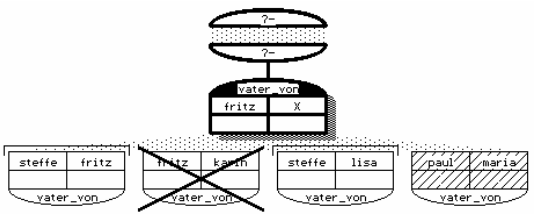
(Unifizieren zweier PROLOG-Ausdrücke heißt : Sie werden auf syntaktische Gleichheit überprüft.)

Damit ist die Lösungssuche aber nicht beendet, da ja noch ungeprüfte Klauseln existieren :

Backtracking

Backtracking ist die Fortsetzung der Lösungssuche beginnend am letzen Alternativpunkt.
Dazu werden alle ab diesem Punkt gebundenen Variablen wieder frei.

Zunächst wird die Bindung der Variablen X aufgehoben und X wieder frei.

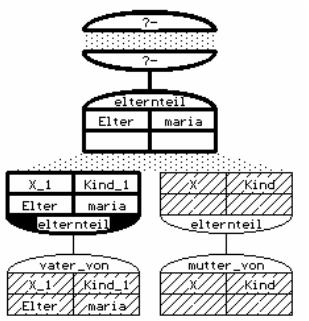


Die nachfolgende zweimalige Unifikation bleibt jedoch erfolglos, so dass vom System ein no ausgegeben wird.

Im Falle der Anfrage ?- vater_von(steffen,X). ergeben sich zwei Lösungen.

Unifikation zweier Variablen

An das System wird die Anfrage ?- elternteil_von(Elter,maria). gestellt. Damit ergibt sich



Das System findet zwei Regelköpfe, die der Anfrage entsprechen und verwendet den ersten Regelkopf. In diesem stehen aber die Variablen X und Kind. PROLOG benennt diese Variablen zunächst zu X_1 und Kind_1 um.
Anschließend unifiziert es die Variable X_1 mit der ungebundenen Variablen Elter und die Variable Kind_1 mit dem Argument maria.

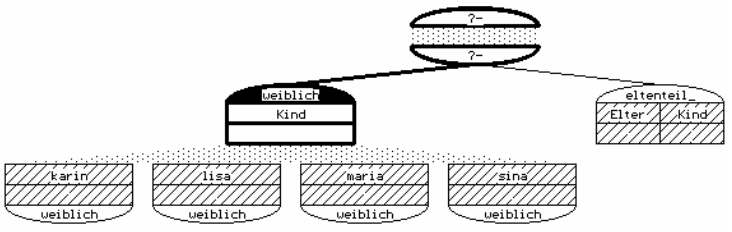
Unifikation ist das Gleichsetzen von Variablen. Hierfür gilt:

- sind beide Variablen ungebunden, dann stehen sie für den gleichen Wert, d. h.
  erhält eine Variable einen Wert, wird dieser automatisch der zweiten Variablen zugewiesen.
- ist eine Variable gebunden und die andere nicht, so erhält die freie Variable den Wert der gebundenen.
- sind beide Variablen gebunden, so ist die Unifikation gleich der von Termen

Resolution

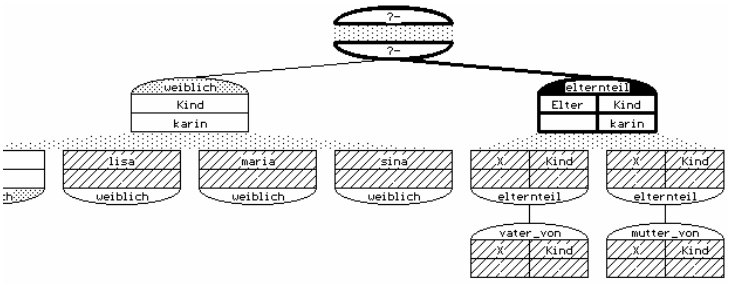
Resolution ist ein Beweisprinzip, das eine Anfrage durch schrittweise Ersetzung der Teilziele durch Fakten oder Regelrümpfe in die leere (wahre) Behauptung ?- . unter Ausgabe der zwischenzeitlich gebundenen Variablen überführt.
Notwendig sind dabei die Prinzipien der Unifikation und des Backtrackings.

An das System wird folgende Anfrage gestellt: ?- weiblich(Kind),elternteil_von(Elter,Kind).
Es ergibt sich das folgende Bild:

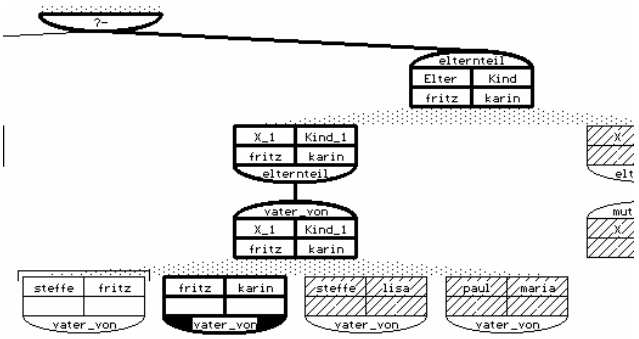


Das System versucht, das erste Teilziel weiblich(Kind)mit einem Faktum zu unifizieren, d. h. das Teilziel zu erfüllen.
Es wird erfüllt durch die Bindung von karin an die Variable Kind.
Damit wird die Anfrage reduziert auf ?- elternteil_von(Elter,karin).

Mit der gebundenen Variable Kind wird nun versucht, das zweite Teilziel elternteil_von(Elter,karin)zu erfüllen.

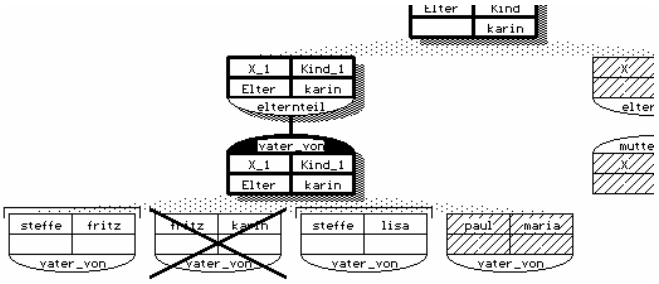


Da es zwei Regeln elternteil_von/2 gibt, ersetzt das System intern die Anfrage ?- elternteil_von(Elter,karin).
durch ?- vater_von(Elter,karin); mutter_von(Elter,karin).
Beachte : Semikolon steht für ODER. Es fand also ebenfalls eine Unifikation der Variablen X mit Elter statt.



Das System unifiziert das Teilziel vater_von(Elter,karin), so dass letztlich fritz an Elter gebunden wird.
Damit gibt es keine freien Variablen mehr, die ODER-Bedingung ist erfüllt und PROLOG gibt als erste gefundene Lösung Kind = karin und Elter = fritz aus.

PROLOG konnte aber nicht alle Fakten für die Lösungssuche verwenden, somit setzt Backtracking am letzen Alternativpunkt ein. Es ergibt sich:



Jedoch kann in der vater_von-Klausel keine Lösung mehr gefunden werden. Das System geht in den ODER-Zweig mutter_von und wird fündig.
Damit ist die Lösungssuche immer noch nicht beendet, denn im mutter_von-Zweig gab es noch unbenutzte Fakten. Somit setzt erneut Backtracking ein, das zu keinem weiteren Erfolg im mutter_von-Zweig führt.

Das System hat aber zu Beginn bei der Bestimmung der Variablenwertes für weiblich(Kind) einen Backtrackingpunkt gesetzt. Somit geht die Lösungssuche weiter, usw.

Wie beantwortet PROLOG Anfragen?

PROLOG versucht alle Teilanfragen zu erfüllen, d.h. zu zeigen, daß die Anfrage logisch aus den Fakten und Regeln folgen.
Kommen Variablen in der Anfrage vor, so versucht PROLOG diese mit Objekten zu instanziieren, und dann die instanziierte Anfrage zu erfüllen.
Ist eine Anfrage nicht aus der Datenbasis herleitbar, so kann PROLOG sie nicht erfüllen.

Achtung ! : tot :- tot.
?- tot. Liefert eine Endlosschleife

Erklärung : p :- q ,r.

deklarative Semantik :
p ist wahr wenn q und r wahr sind oder
Aus q und r folgt p .
prozedurale Semantik (PROLOG) :
Um Problem p zu lösen, löse zuerst q und dann r .

Einige Prolog-Funktionen auf Listen:

my_append / 3
```
my_append( [ ] , X , X ).
my_append( [X | Y] , Z , [X | W] ) :- my_append( Y , Z , W ).
```

my_reverse / 2
```
my_reverse( [ ] , [ ] ).
my_reverse( [ X | Y] , Z ) :- my_reverse( Y , W ) , my_append( W , [X] , Z).
```

my_union / 3
```
my_union( [ ] , L , L).
my_union( [Y | T] , L , R ) :- ist_element( Y , L ) , ! , my_union( T , L , R ).
my_union( [Y | T] , L , [Y]R ) :- my_union( T , L , R ).
```

my_intersection / 3
```
my_intersection( [ ] , _ , [ ] ).
my_intersection( [X | T] , L , [X | B] ) :- ist_element( X , L ) , ! , my_intersection( T , L , B ).
my_intersection( [ _ | T ] , L , B ) :- my_intersection( T , L , B).
```