Inhaltsverzeichnis

1 Einführung in WBS

1.1 Repräsentation von Wissen

- Vollständigkeit
  ist alles notwendige vorhanden?
- Abstraktion
  frei von unwichtigen Details (Kannibal/Mönch : Farbe des Bootes)
- Ökonomie
  sparsamer Einsatz von Ausdrucksmitteln
  (Mehr Kannibalen als Mönche knapper als Aufzählung der Tupel für die diese Behauptung gilt)
- Redundanzfreiheit
  (keine Angabe von linkem und rechten Ufer.)
  Redundante Daten gefährlich → Aktualisierung, Löschen
- Tranzparenz
  Ist die Darstellung verständlich

1.2 Repräsentatives Wissen Deklaratives / Prozedurales Wissen

Algorithmen	*prozedurales Wissen*	Problemlösung
Daten	*deklaratives Wissen*	Wissensbasis
konventionelles System		Wissensbasiertes System

Deklarative Sprache: 4G Language (Was soll gemacht werden, uneffizient)	Prozedurale Sprache: 3G Language (Wie soll es gemacht werden, effizient)
MAX(Liste) O(n*n)	Lineare Suche O(n)

1.3 Erwartungen an Experten

- Wissen
- Problemlösung
- „beste“ Problemlösung
- kann Erklärungen zur Problemlösung abgeben.
- eignet sich neues Wissen an.
- ermittelt im Dialog mit dem Kunden das Wissen um das konkrete Problem.

1.4 Anwendung von XPS

- Diagnostik/ Selektion:
  (Medizin, Analyse von Schaltkreisen)
- Konstruktion/ Konfiguration:
  (Computeranlage, HiFi, Gestaltung von Platinen)

1.5 Potential und Grenzen

Für KI gedacht –¿ Mißerfolg.
erheben nicht mehr den Anspruch Experten ersetzen zu können.

Grenzen:

- Beziehen sich auf ein festgelegtes Wissensgebiet.
- Kennt seine Grenzen/ Kompetenzen nicht.
- Kritisch in Sicherheitstechnischen Bereichen.
- Kein intuitives Alltagswissen.
- Keine eigene Lernfähigkeit.
- intuitives Wissen nur schwer in explizite Form zu bringen (dt. Gramatik)

1.6 Suchverfahren

Färbeproblem:
Färbe benachbarte Länder mit unterschiedlichen Farben ein.
Generate and Test Prinzip — schlechte Peformance.

Wegesucheproblem:
Traveling Salesman
Ziel erreichbar.
Untersuchung auf Zyklenfreiheit.
Diejkstra, Floyd (Adjazenzmatrizen)

Weinkrügeproblem:

```
(Vol, Füllung)
(  9,       9), (7,0), (4,0) --> (9,6),(7,0),(4,3)
3! Möglichkeiten zum Umfüllen.
```

Zustände werden in Baum (Graf) eingeordnet. Kanten entsprechen Übergängen zwischen einzelnen Zuständen.

Tiefensuche:
Hoher Stackaufwand
unfair, da Zweig unendlich lang sein kann.

Breitensuche :
Hoher Speicheraufwand
fair, da auf jedenfall der Zielknoten gefunden werden kann.

Schrittweise Vertiefung:
Kombination aus T und B Suche.
Nur bis max. Tiefe möglich (Suchschranke).

Heuristische Suche:
I.A. Verbesserung des Suchverfahrens durch Einbringen von Wissen (welcher Ast enthält wahrscheinlich eine Lösung).
Keine Garantie für Optimierung

Besten Suche:
Verwende einen heuristischen Suchbaum mit Breiten/ Tiefensuche.
Einsortieren an die richtige Stelle anhand von heuristischen Regeln.

1.7 Allgemeiner Aufbau eines XPS

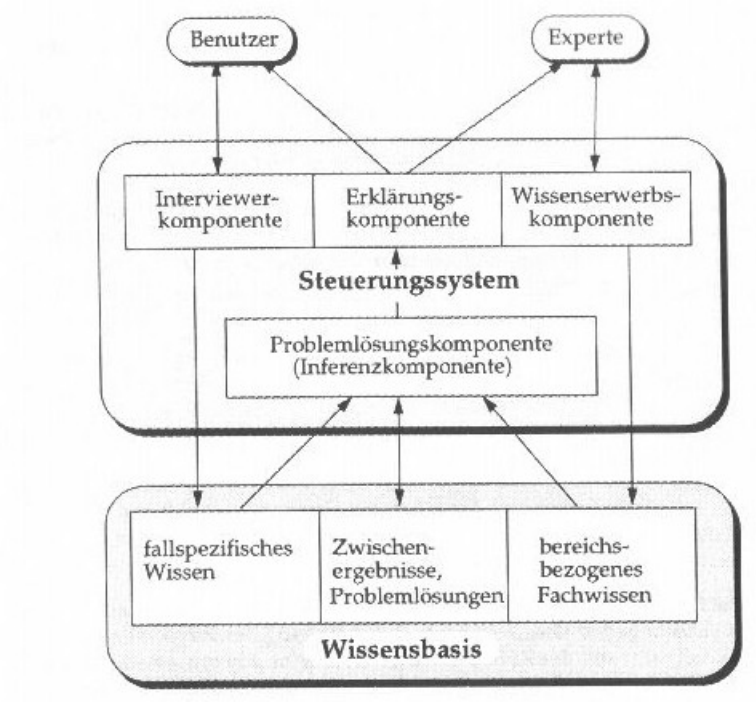

Abbildung 1: Allgemeiner Aufbau eines XPS

1.7.1 Steuerungssystem:

ist unabhängig von der Wissensbasis. besteht aus:

Problemlösungskomponente
Interpretiert Expertenwissen zur Lösung des benutzerspezifischens Problems. Erzeugt eine Problemlösung und reicht diese an die Erklärungskomponente weiter.

Interviewerkomponente
Führt den Dialog mit dem Benutzer (GUI)

Erklärungskomponente
Erläutert den von der Problemlösungskomponente gefundenen Lösungsweg. Kann zur Fehlerfindung vom Experten verwendet werden.

Wissenserwerbskomponente
Dient den Aufbau und der Wartung des Expertensystems.

1.7.2 Wissensbasis:

enthält:

Expertenwissen
Bereichsspezifisches Wissen
Oft in Form von Regeln.

Fallbezogenes Wissen
Kommt vom Benutzer und bezieht sich auf einen konkreten Anwendungsfall.
Liegt meist als Fakten vor.

Zwischenergebnisse und Problemlösungen
Werden dynamisch durch die Problemlösungskomponente erzeugt.

1.8 Verschiedene Formalisierungsmöglichkeiten

```
a)  grün ( c )
    Prädikat
           Konstante

b)  farbe ( c, grün )
    Prädikat
            Konstanten

c)  farbe ( c ) = grün
    Funktion
            Konstante
                Prädikat
                  Konstante
```

Beispiele „b und e haben die selbe farbe“

a)
blau (b) $\wedge$ blau (e) 4 $\vee$
rot (b) $\wedge$ rot (e) 4 $\vee \ldots$

b)
$\forall x$ farbe $(b, x) \Leftrightarrow$ farbe (e, x)

c)
farbe(b) = farbe(e)

1.9 3 Ebenen der Logik

Syntax
Grammatik der Sprache.
Prädikatenlogik: Konstanten, Variablen, Quantoren, Prädikate, Funktionen.

Semantik
Bedeutung/ Interpretation der Sprache.
Formeln mit gleicher Interpretation bzw. der selben Klauselmenge sind semantisch äquivalent.

Kalkül
(Resolutions) kalkül Regeln zur syntaktischen Umformung.

1.10 Prädikatenlogik

Vorteile:

1. Exaktheit
2. gute Kenntnis der theoretischen Grundlagen
   (Aristoteles; Frege(19 Jhd.))
3. Halbwegs effizientes Kalkül ($\rightarrow$ PROLOG)

Nachteile:

1. Exaktheit
2. zu Ausdrucksschwach für Alltagssituationen

Beispiel:

Vorgestern haben wir **frischen** Fisch gekauft.
Heute essen wir den **frischen** Fisch.

Erweiterungen der Prädikatenlogik:

Probabilistische Logik (unsicheres Wissen):
Sensoren geben nur mit einer bestimmten Wahrscheinlichkeit einen korrekten Wert an. Man kann nur angeben mit welcher Wahrscheinlichkeit die getroffene Folgerung zutrifft. (Wahrscheinlichkeitsbäume).

Fuzzy Logik (wages Wissen)
Schwammige Informationen (schwerer, größer etc.)
Über Wahrscheinlichkeitstabellen für Zuordnung.

nicht monotone Logik (unvollständiges Wissen)
Im Laufe der Diagnose erlangt man weiteres Wissen, welches zur Falsifizierung früher getroffener Folgerungen führen kann.
Beispiel: Pinguinproblem.
Überlegungen wie evtl. falsche Folgerungen behandelt werden: Benutzerrückfrage.
Rechtfertigungsberechtigte Systeme: Welche Regel hat Konklusion bewiesen (Tiefensuche)
Annahmebasierte Systeme: Breitensuche
Kontext (Alle Annahmen, die zu diesem Konflikt führen) analysieren.

temporale Logik (zeitabhängiges Wissen)
Wer einmal geheiratet hat kann nicht wieder ledig werden.
Kind jünger als Eltern.

2 Regelsysteme

2.1 Typische Regelsysteme

2.1.1 Ableitung von Wissen

1. **Wenn** Anna in Oslo ist, **dann** ist Jan nicht in Oslo
2. **Wenn** das Tier Haare hat, **dann** ist es ein Säugetier

2.1.2 Aktionen

1. **Wenn** der Gegener auf einem Feld eine Mühle schließen kann, **dann** setze einen eigenen Stein darauf.

2.1.3 Schreibweise der Regeln

$R : \pm A_1, \pm A_2, \pm A_3, \pm A_4, \ldots, \pm A_n \rightarrow B$

R ... Regelbezeichner
A_n... Prämissen
B ... Konklusion

Eine negative Prämisse bezieht sich auf ein negatives Faktum. Alle Prämissen sind mit UND verknüpft.

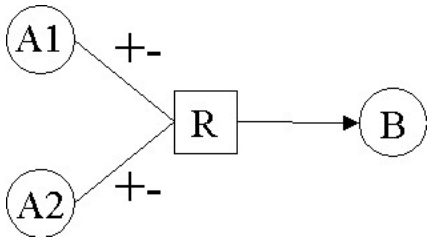

Abbildung 2: Darstellung einer Regel

2.2 Fakten

2.2.1 Statische Merkmale eines Faktums

1. Name (Identifikation, bzw. Nummer)
2. Text (z.B. „Anna ist ...“)
3. Frage-Text (z.B. „Ist Anna in Oslo?“)

Statische Merkmale bleiben im Verlauf einer Sitzung mit dem System unverändert.

2.2.2 Dynamische Merkmale

können sich im Verlauf einer Sitzung ändern.

1. Status (ist das Faktum *wahr/ falsch/ unbekannt*
2. goal (ist das Faktum als Hypothese eingetragen?)

2.3 Regeln

2.3.1 Statische Merkmale einer Regel

1. Name (bzw. Nummer)
2. Liste der Prämissen
3. Konklusion
4. weitere ...

2.4 Zugriff auf Merkmale von Fakten, Regeln

```
xps_get( Klasse, Typ, Name, Merkmal, Wert )
xps_set( Klasse, Typ, Name, Merkmal, Wert )
```

Klasse	Art des Merkmals `{dynamic, static}`
Typ	`{fact, rule}`
Name	Name, bzw. Nummer
Merkmal	z.B. Text, Fragetext bei Fakten Prämissen, Konklusion bei Regeln
Wert	nicht unifizierte Variable beim Lesen unifizierte Variable beim Schreiben.

3 Rückwärtsverkettung

Allgemeines Prinzip:

1. Hypothese aufstellen (goal)
2. Regeln suchen, die zur Hypothese passt (`Konklusion = Hypothese`)
3. Prämissen der Regel validieren (Rekursiv als neue Hypothese/ Zwischenziel)
4. Solange bis alle offenen Prämissen verifiziert.

3.1 Verfahren zum Prüfen einer Hypothese

Fall 1: Die Hypothese ist positiv

1. Prüfe den Status der Hypothese
   Status:
   wahr/ falsch → fertig
   unbekannt → weiter mit 2.
2. Prüfe ob es eine Regel für die Hypothese gibt.
   Falls Ja:
   Prüfe rekursiv deren Prämissen. Ist die Prämisse verifiziert, dann trage Status *wahr* ein (*dynamic*)
   Falls Nein:
   Falls es keine Regel gibt, die die Hypothese beweisen kann, frage den Benutzer und trage die Antwort als Status ein.

Fall 2: Hypothese negativ

Schwache Negation
Eine negative Prämisse $-X$ ist verifiziert, wenn die positive Hypothese X nicht verifiziert werden kann.
Entspricht der PROLOG-Negation.

Starke Negation
Eine negative Hypothese $-X$ ist verifiziert, wenn die positive Hypothese X definitiv **falsch** ist. Diese Aussage kann nur der Benutzer treffen: FRAGEN!

3.2 Erklärungskomponente

Mögliche Fragen an das System:

- Wie kommt das System zu seiner Schlußfolgerung?
  Lösung:
  Wird ein Faktum F aufgrund einer Regel R bestätigt, so trage R als Begründung in F ein.
- Warum stellt das System bei der Rückwärtsverkettung eine konkrete Frage?
  Antwort:
  Weil REGEL geprüft wird um HYPOTHESE zu verifizieren.

3.2.1 Steuerung zur Vermeidung unnötiger Fragen

Prüfe zuerst die Hypothesen, die vom Benutzer beantwortet werden können (Blätter).

Prüfe zuerst die Hypothesen, die in möglichst vielen Regeln vorkommen.

3.2.2 Begriffsdefinitionen

Regelkonflikt Eine Situation, in der mehrere Regeln anwendbar sind.

Konfliktmenge Menge der anwendbaren Regeln.

Konfliktlösung Strategie zur Auswahl einer Regel aus der Konfliktmenge

3.2.3 Möglichkeiten der Konfliktlösung

Keine Strategie Reihenfolge wird bestimmt durch die Reihenfolge der Regeln in der (Prolog)Datenbasis.

Prioritäten Vergabe von Prioritäten für die Regeln. Im Konfliktfall wird die Regel mit der höchsten Priorität ausgewählt (kleiner Wert)

Unterschiede: Prioritäten sind dynamisch, Reihenfolge der Regeln ist statisch.

3.2.4 Dynamische Änderung von Prioritäten

- Bestimmte Regeln werden häufiger aufgerufen als andere (Statistiken). Diese erhalten eine höhere Priorität.
- Aufgrund eines bestimmten Faktums werden gewisse Regeln wahrscheinlicher.
  Beispiel:
  Wenn das Licht nicht brennt, *dann* könnte es sein, daß die Batterie leer ist.
  ⇒Batterie-Regeln werden bevorzugt.

3.2.5 Heuristik

Eine Heuristik ist eine Regel, die in vielen Fällen, aber nicht allen, zutrifft.

Konkretes Beispiel:

R-Nr		Faktum „licht brennt“		auf Regel	Aktion
R20	:	-3	$\rightarrow$	7	(Priorität der Regel 7 wird erhöht)
R21	:	3	$\rightarrow$	-7	(Priorität der Regel 7 wird erniedrigt)

4 Vorwärtsverkettung

Situation:
„Symtome“ sind vollständig bekannt. Das System soll daraus Schlußfolgerungen ziehen.

Prinzip:

- Finde Regel, deren sämtliche Prämissen erfüllt sind.
- Regel „feuert“: Trage die Konklusion der Regel in die Faktenbasis ein. (Status = wahr)

4.1 Vergleich Regelsystem, Spiel

Regelsystem	Spiel
Regeln	Spielregeln
Fakten	Spielstellungen
Steuerung	Strategie

Bsp: Chipsspiel.

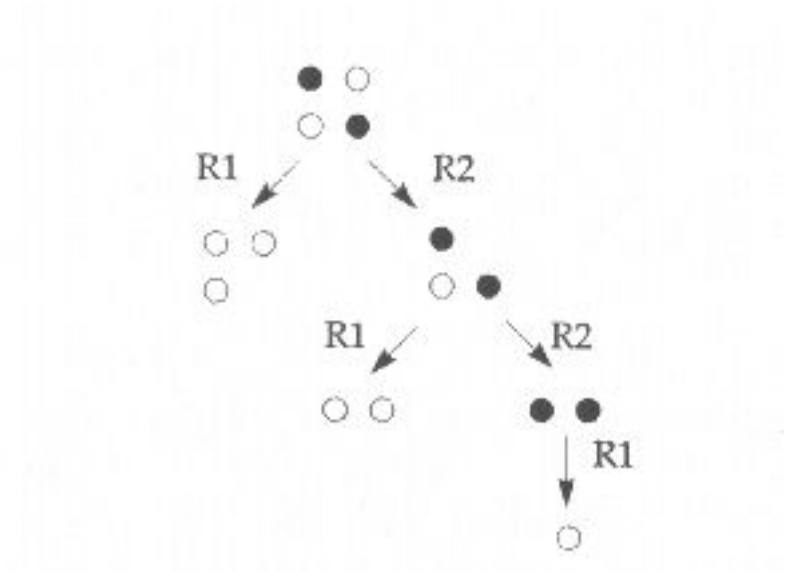

Abbildung 3: Chipsspiel - Suchbaum

Regeln legen die erlaubten Züge fest; Sie geben keinen Weg zum Erreichen des Spielziels an. Rückwärtsverkettung hier nicht angebracht.
Die Vorwärtsverkettung durchläuft den gesamten Suchraum. Es werden alle möglichen Spielstellungen erzeugt (Die von der Ausgangsstellung erreichbar sind).

4.2 Realisierung des Chipsspiels als Regelsystem

a)
Repräsentation einer Spielstellung:

```
status( G, R )
        Anzahl grüne Chips
           Anzahl rote Chips
```

b)
Repräsentation der Regel:

$r1(G,R) : status(G,R), G \geq 2 \rightarrow status(G-2, R+1)$
$r2(G,R) : status(G,R), G \geq 1, R \geq 1 \rightarrow status(G, R-1)$

4.3 Refraktionsprinzip

Prolog-Beispiel zur Vorwärtsverkettung:

```
forward :-
          anwendbare_Regel_finden(R),
          fire(R),
          forward.
```

Problem: Die Rekursion/Schleife endet nicht.
Wie kann man verhindern, daß immer wieder die Regel feuert?

Das Refraktionsprinzip:
Eine Regel darf nie zweimal mit denselben Prämissen feuern.

Beispiel am Chips-Spiel:

FB (inkr)	Regel
status(2,2)	$r_1(2,2)$
status(0,3)	$r_2(2,2)$
status(2,1)	$r_1(2,1)$
status(0,2)	...
⋮	⋮

Prinzip der Vorwärtsverkettung mit Refraktion:

- Regel auswählen deren Prämissen alle erfüllt sind
- Prüfe ob die Regel mit der entsprechenden Instantierung der Variablen schon gefeuert hat
- Wenn nein:
  Regel feuert: Konklusion wird in die Faktenbasis eingetragen (status=wahr);
  vermerken das die Regel mit dieser Instanz gefeuert hat.

4.4 Konfliklösungsstrategien

Begriffe:

Konfliktmenge Die Menge aller anwendbaren Regelinstanzen

Konfliklösung Die Auswahl einer Regelinstanz aus der Konflikmenge

Beispiel der Refraktion am Chips-Spiel mit Angabe der KM(Konflikmenge):

FB (inkr)	KM(ohne bereits gefeuerte RI)	ausg. RI
status(2,2)	$r_1(2,2)$,$r_2(2,2)$	$r_1(2,2)$
status(0,3)	$r_2(2,2)$	$r_2(2,2)$
status(2,1)	$r_1(2,1)$,$r_2(2,1)$	$r_1(2,1)$
status(0,2)	$r_2(2,1)$	$r_1(2,1)$
status(2,0)	$r_1(2,0)$	$r_1(2,0)$
status(0,1)	-	-

Eine Konfliktlösungsstrategie schreibt vor, welche Regelinstanzen aus der Konfliktmenge zu löschen sind.

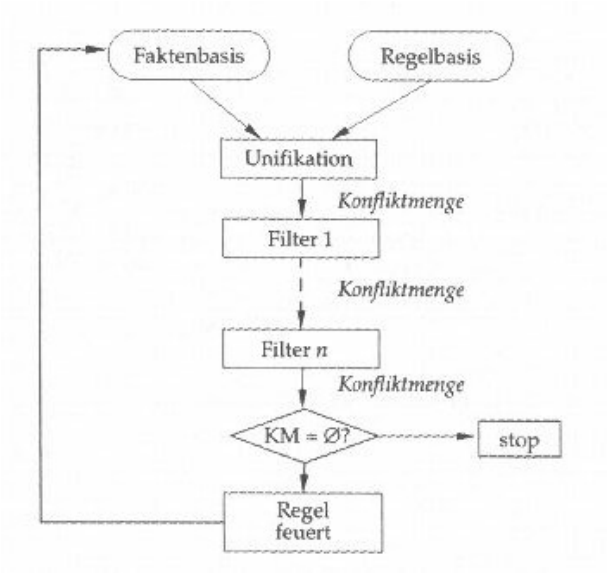

Abbildung 4: Konfliktlösungsstrategien

4.4.1 Konfliktlösungsstrategie: Refraktion

Lösche alle Regelinstanzen aus der Konfliktmenge, die schon gefeuert haben.
(Für immer löschen)

4.4.2 Konfliktlösungsstrategie: Priorität

Entferne alle Regeln aus der Konfliktmenge, die nicht höchste Priorität haben.
(Nur für einen Zyklus gelöscht)
Beispiel der Prioritätsstrategie am Chips-Spiel:

$p(r_1) < p(r_2)$ (Die Regel r_2 hat höhere Priorität als r_1)

Unterstrichene Regeln sind ausgewählt.

FB (inkr)	KM vorher	KM nachher
status(2,2)	r_1(2,2),r_2(2,2)	$\underline{r_2(2,2)}$
status(2,1)	r_2(2,2),r_1(2,1),r_2(2,1)	$\underline{r_2(2,1)}$
status(2,0)	r_1(2,2),r_1(2,1),r_1(2,0)	r_1(2,2),$\underline{r_1(2,1)}$,r_1(2,0)
status(0,2)	r_1(2,1),r_1(2,0)	r_1(2,2),$\underline{r_1(2,0)}$
status(0,1)	r_1(2,2)	...

4.4.3 Konfliktlösungsstrategie: Spezifität

Entferne aus der konfliktmenge alle Regeln, die allgemeiner sind als eine andere.

Beispiel für allgemeiner/spezifischer:

r_1 Wenn Temperatur > 100 ist, dann aktiviere die Kühlung

r_2 Wenn Temperatur > 100 und Druck > 200 löse Alarm aus

Bedeutung zwischen diesen Regeln:
Wenn r_2 anwendbar, dann ist auch r_1 anwendbar.
Die Menge der Anwendungsfälle $r_2 \subseteq$ Anwendungsfälle r_1.
In diesem Fall heißt:

r_1 ist **allgemeiner** als r_2 bzw.
r_2 ist **spezifischer** als r_1.

Übung: Gegeben ist das Regelsystem:

Prio=10	$r_1(Y)$	:	p(a,Y)	→	p(Y,a)
Prio=15	$r_2(Y,Z)$	:	p(a,Y), p(Y,Z)	→	p(a,Z)
Prio=5	$r_3(X,Y)$	:	q(X,Y)	→	p(Y,X)

a) Ist eine Regel spezifischer als eine andere?
b) Beschreiben Sie einen Ablauf der Vorwärtsverkettung bei Anwendung von Refraktion, Spezifität und Priorität(in der Reihenfolge).

zu a) Die Regel r_2 ist spezifischer als r_1.
zu b) Faktenbasis: {p(a,b), q(b,c)}

Unterstrichene Regeln sind ausgewählt.

FB (inkr)	KM vorher	KM nachher
p(a,b), q(b,c)	r_1(b),r_3(b,c)	$\underline{r_3(b,c)}$
p(c,b)	r_1(b)	$\underline{r_1(b)}$
p(b,a)	r_2(b,a)	$\underline{r_2(b,a)}$
p(a,a)	r_1(a),r_2(a,a),r_2(a,b)	$\underline{r_2(a,a)}$,r_2(a,b)
-	r_1(a),r_2(a,b)	$\underline{r_2(a,b)}$
-	r_1(a)	$\underline{r_1(a)}$
-	-	-

Geben Sie eine "opereationale" Definition von "spezifischer als" an, betrachten Sie folgende Beispiele:

a)	r_1	:	p, r	→	s
	r_2	:	p, $-q$, r	→	u
b)	r_1(X,Y)	:	p(X), q(X,Y)	→	r(Y)
	r_2	:	p(a), q(a,b)	→	s(a)
c)	r_1(X)	:	p(X)	→	s(X)
	r_2	:	p(a), $-q$(b)	→	r(b)

a)	Präm(r_1)	$\subset$	Präm(r_2)
b)	Präm(r_1(a,b))	$=$	Präm(r_2)
c)	Präm(r_1(a))	$\subset$	Präm(r_2) (Kombination aus a) und b))

Präm(X): Menge der Prämissen von X

Weitere Darstellung von allgemeiner und spezifischer:

r_1(Y)	:	-verheiratet(klaus,Y)	→	ledig(klaus)
r_2	:	-verheiratet(klaus,petra)	→	verheiratet(otto,petra)

Präm(r_2) = Präm(r_1(petra))
ist nicht für das Faktum verheiratet(klaus,sabine) gültig.

Wenn r_1 keine Variable mit negativen Prämissen enthält, dann heißt r_1 allgemeiner als r_2, wenn es eine Instanz $\overline{r_1}$ von r_1 gibt, so daß gilt:

Präm($\overline{r_1}$) $\subset$ Präm(r_2)

Enthalten die Regeln keine Variablen, so gilt folgende einfache Definition:

r_1 ist allgemeiner als r_2, falls Präm(r_1) $\subset$ Präm(r_2)

Warum muß die Refraktion an erster Stelle angewandt werden?

Beispiel:

FB = {a,b}
prio(r_1) < prio(r_2)

r_1 : a → c
r_2 : b → c

Strategie: Spezifität, Priorität

FB	KMvor	KMnach
a,b	r_1, r_2	r_1
c	r_1, r_2	-

4.4.4 Konfliktlösungsstrategie: Aktualität

Die "aktuelle" Regel feuert, d.h. die Regel, deren Prämissen sich auf die aktuellste Einträge in der Faktenbasis beziehen.

Zwei Varianten um die aktuellste Regel zu bestimmen:

Maximum Wähle die Regel, deren Prämissen das aktuellste Faktum, gegenüber anderen wählbaren Regeln, beinhaltet.

max(t(p_1)) > max(t(p_2))
p_1 ∈ Prämissen1
p_2 ∈ Prämissen2

Mittelwert Wähle die Regel, deren Prämissen im Mittel aktueller sind als bei anderen wählbaren Regeln.

avg(t(p_1)) > avg(t(p_2))
p_1 ∈ Prämissen1
p_2 ∈ Prämissen2

Beispiel am Chips-Spiel:

Syntax:
status$(R, B)_t$ Eintrag des Faktum zur Zeit t
Regel$(R, B)_t$ Regel auf ein Faktum, dessen Zeit t ist

prio(r_2) < prio(r_1)

Zeit t	FB	KMvor	KMnach
0	status$(2,2)_0$	$r_1(2,2)_0$,$r_2(2,2)_0$	$r_2(2,2)_0$
1	status$(2,1)_1$	$r_1(2,2)_0$,$r_2(2,1)_1$,$r_1(2,1)_1$	$r_2(2,1)_1$
2	status$(2,0)_2$	$r_1(2,2)_0$,$r_1(2,1)_1$,$r_1(2,0)_2$	$r_1(2,0)_2$
3	status$(0,1)_3$	-	