Einführung in die Künstliche Intelligenz

Thomas Baar[*]
Robert Magnus
Torsten Trzewik
u.a.

Berlin, den 29. Dezember 1993

Zusammenfassung

Script zur Vorlesung „Einführung in die KI“, gehalten von Prof. Dr. H. D. Burkhard im Sommersemester 1993 an der Humboldt-Universität zu Berlin. Es enthält noch einige Fehler und deckt auch nur einen Teil der gehaltenen Vorlesung ab. Korrekturvorschläge werden jederzeit dankbar entgegengenommen.

[*] Ein besonderer Dank geht an Dirk Verworner für den TeX-Satz und an alle weiteren Lektoren, die sich oft die Zeit nahmen, mich an meine häufigen Fehler zu erinnern.

Inhaltsverzeichnis

ELIZA
Turing-Test

Kapitel 1

Intelligenz

Die Mehrzahl von „Chaos"? – „KI"!

1.1 Einleitung

Intelligenz ist die geistige Leistung des Menschen, Probleme zu erkennen und zu lösen. Sie tritt dabei in vielfältigen Formen auf. So gehören dazu sprachliches Verständnis, Redegewandtheit, räumliches Denken, Gedächtnis, Auffassungsgabe, schlußfolgerndes Denken und andere Gebiete.

Nun können Programme verschiedene Aspakte der Intelligenz simulieren und versuchen menschliche Fähigkeiten zu erreichen. Zwei Beispiele dafür sind ELIZA und der Turing-Test. ELIZA ist ein Dialogsystem, das von Joseph Weizenbaum 1966 am MIT[1] geschaffen wurde. Der Computer simuliert dabei einen Psychiater, der sich mit dem Patienten per Terminal unterhält. Hier ein Beispiel für einen typischen ELIZA-Dialog:

```
==> (visit-doctor 'Bill)
(HELLO, BILL)
(WHAT SEEMS TO BE THE TROUBLE?)

**(everyone hates me)
(WHY DO YOU SAY EVERYONE HATES YOU)

**(whenever i enter a room no one will look me in the eye)
(MANY PEOPLE HAVE THE SAME SORTS OF FEELINGS)

**(they laugh at me behind my back)
(YOU SEEM TO THINK THEY LAUGH AT YOU BEHIND YOUR BACK)

**(they get together and talk about how they hate me)
(MANY OF MY PATIENTS HAVE TOLD ME THE SAME THING)

**(even my mother hates me)
(YOU FEEL THAT EVEN YOUR MOTHER HATES YOU)
```

[1] Massachusetts Institute of Technology

```
**(even when i was a baby she said i was ugly and stupid)
(PLEASE CONTINUE)

**(she hit me all the time)
(WHY DO YOU BELIEVE SHE HIT YOU ALL THE TIME)
```

Obwohl ELIZA nur durch Umstellen der erhaltenen Antwort und eventuell Austausch von Synonymen eine neue Frage generiert, hatte es nach außen den Anschein eines intelligenten Wesens. Der Erfolg seines Programms überraschte selbst seinen Schöpfer und veranlaßte ihn, seine Auffassungen über die Verantwortung eines Informatikers zu überdenken.

Das zweite Beispiel, der Turing-Test, ist etwas älter. Er stammt aus den Anfangszeiten der Informatik und wurde von ALAN TURING entwickelt. Das Szenario ist relativ einfach. Eine Person sitzt vor einem Terminal und kommuniziert mit einem Partner. Sie soll allein durch den Dialog herausfinden, ob der Partner ein Mensch oder ein Computer ist. Der besondere Trick liegt in der Verwendung eines lernfähigen Computers, der sich semantische Sachverhalte (Ironie, hintergründige Fragen usw.) merkt und selber weiterverwendet. Die Frage ist nun, ob irgendwann einmal der Fall eintritt, daß der Computer von einem Menschen nicht mehr zu unterscheiden ist. Dann wäre die Äquivalenz von natürlicher und künstlicher Intelligenz gezeigt.

1.2 Probleme der Realisierung

Die Probleme, die bei der Realisierung von KI-Systemen auftreten, liegen im natürlichen Gegensatz der simulierten Anwendung und der zugrundeliegenden Algorithmen. Eine kleine Gegenüberstellung eines *von-Neumann*-Rechner mit einem *neuronalen Netz*[2] stellt das recht deutlich dar.

Rechner		Neuronales Netz
exakt	-	unexakt
seriell	-	parallel
starr	-	flexibel
fehlersensibel	-	fehlertolerant

Natürlich kann ein neuronales Netz auf einem Rechner partiell simuliert werden, das menschliche Denken hat sicherlich aber auch auf symbolischer Ebene vergleichbare Eigenschaften eines neuronalen Netzes.

Um diese Probleme in den Griff zu bekommen, hat die KI Anleihen aus anderen Gebieten nehmen müssen, auf die sie wiederum einen Einfluß über Rückkopplung (*spin off*) hatte. Die folgende Übersicht zeigt nur einen kleinen Ausschnitt und ist keineswegs erschöpfend.

KI	Psychologie, Neurologie, Sprachwissenschaften
verteilte KI	Betriebswissenschaften, Soziologie, Rechtswissenschaften
Robotik	Mechanik, Elektronik, Audiovisuelle Systeme

WINSTON hatte 1984 die Ziele der KI im Bauen intelligenter, nützlicher Systeme und Verstehen der natürlichen Intelligenz charakterisiert.

[2] Modell des Gehirnes

1.3 Gebiete der KI

Genauso vielfältig, wie der Einfluß der KI auf andere Wissenschaften, ist die KI selbst. Dazu gehören:

Sinnesleistung
Es wird versucht, die menschlichen Leistungen zu verstehen und zu simulieren. Kerngebiete sind dabei das Sehen, besser gesagt, das optische Erkennen; das Hören bzw. Sprechen und die räumliche Orientierung.

Expertensysteme
Programme, die gestützt auf eine Datenbasis (Wissen), auf Fragen Auskunft geben. Dabei erweist es sich als schwierig, das menschliche Wissen (*Expertenwissen*) für den Rechner umzusetzen.

Lehr- und Lernprogramme
Systeme zur Wissensvermittlung. Sie sind von der Struktur eng mit den *Expertensystemen* verwandt.

Übersetzungen
Die Übertragung eines Textes in eine andere Sprache ist z.B. vom Kontext des Textes und der kulturellen Basis des Autors abhängig.

Schachprogrammierung
Eines der Gebiete, das (partiell) durch die Verwendung von lernfähigen Programmen gekennzeichnet ist.

Schlu"ssregeln@Schluszregeln
Inferenz
kombinatorische
Explosion

Kapitel 2

Formalismus in der Wissensverarbeitung

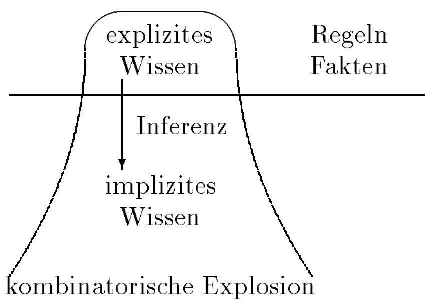
Alles, was der Rechner kann, vermag auch der Mensch, jedenfalls theoretisch. Es wäre also denkbar, einen Programmablauf auf einem Blatt Papier nachzuvollziehen. Begründet liegt dies in der Determiniertheit eines Programms; alle Schritte des Ablaufs sind fest bestimmt, selbst Zufallszahlen sind nur Pseudozufallszahlen und vorhersagbar.

Dagegen bedeutet die Simulation von menschlichen Fähigkeiten durch den Rechner das Gegenteil. Dazu muß das menschliche Handeln für den Rechner nachvollziehbar sein. Hierfür sind formale Methoden notwendig.

Einige Dinge werden sich einer formalen Beschreibung wohl immer entziehen, insbesondere Dinge des Fühlens. So kann sich ein Veterinärmediziner über die Empfindungen bei einer Trächtigkeitsuntersuchung noch so den Mund fußlig reden, wenn der Student das nicht einmal selber fühlt, weiß er darüber nichts.

Andere Sachen, wie das Denken, sind gut formalisierbar. So gibt es den Kalkül des natürlichen Schließens, eine formale Beschreibung der, vom Menschen verwendeten, logischen Schlußregeln.

Aber selbst bei solchen formalen Beschreibungen besteht das Problem, die richtigen Schlußregeln anzuwenden. Da diese nicht bekannt sind, sind Heuristiken zum Finden (Suchstrategien) gefragt, weil durch das Anwenden aller möglichen Regeln (Inferenz), die Menge des abgeleiteten Wissens zu groß wird (kombinatorische Explosion). Wir werden diesen Sachverhalt in Kapitel 4 näher beleuchten.

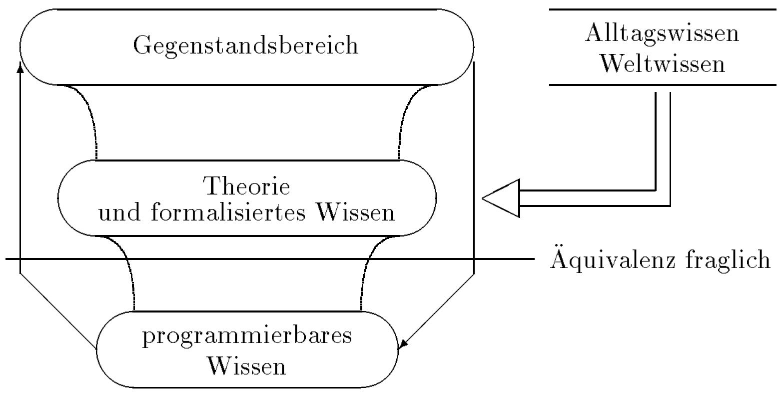


Beispiel Schachspiel:

explizites Wissen: Schachbrett, Figuren, Spielregeln

implizites Wissen: bester nächster Zug, vielleicht eine explizite Gewinnstrategie (ist bis jetzt aufgrund der kombinatorischen Explosion nicht gefunden worden. Es wäre zudem schade um das Schachspiel)

Ein weiteres Problem hat sich insbesondere mit der Entwicklung von Expertensystemen herauskristallisiert. Wenn der Experte sein Wissen in Regeln und Fakten beschreibt, bleibt die Frage, ob er sich sicher sein kann, daß diese Liste vollständig ist. Hat er unter Umständen Regeln vergessen, die zwar nicht zum Fachgebiet gehören, aber trotzdem für Aufgabenstellungen relevant sind ?



Kapitel 3

Constraints – Die Beschränkungen

Vieles im Leben unterliegt (natürlichen) Beschränkungen. Diese Beschränkungen haben sich als gut erwiesen und somit durchgesetzt. Eine der Wichtigsten ist vielleicht die Lebensdauer selbst. Man stelle sich vor, sie wäre unendlich ....

Andere Beschränkungen für uns sind zum Beispiel die Lebensnormen in Form von Gesetzestexten. Wenn jemand unter dem Problem der Ebbe im Portemonnaie leidet, dann sollte er keine nutzlosen (?) Gedanken daran verschwenden, das Problem durch Diebstahl, Raub oder Betrug zu lösen.

Der Suchraum, d.h. die Menge aller potentiellen Problemlösungen, wird also in diesem Fall auf den Gelderwerb durch Verkauf, Arbeit usw. eingeschränkt. Allerdings wird der Suchraum noch weiter eingeschränkt, diesmal durch die aktuelle konjunkturelle Lage. Hier haben wir jetzt also schon mehrere Bedingungen, die unser Problem beeinflussen. Die zu suchende Arbeit in unserem Beispiel

- muß der konjunkturellen Lage genügen und
- darf nicht gegen Gesetze verstoßen.

Manchmal ein schwieriges Unterfangen!

Jetzt wollen wir aber wieder zur KI zurückkehren und sehen, wie solche Beschränkungen ausgenutzt werden.

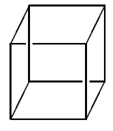
3.1 Bildverstehen

Das Problem ist schnell umschrieben. Es soll gezeigt werden, wie man ein zweidimensionales Abbild eines dreidimensionalen Körper interpretieren kann und soll. Bei der Auswahl konzentrieren wir uns dabei auf ebenflächig begrenzte Körper.

Solche Interpretationen sind aber selbst für den Menschen nicht trivial. Gelegentlich lassen wir uns auch ziemlich täuschen – der Computer erst recht.

Ein bekannter Künstler, Maurits Escher, hat viele solcher „Fehler“ in seinen Bildern verwendet. In seinem Bild „Belvedere“ sitzt links unten auf einer Bank ein Mann, der einen unmöglichen Würfel in der Hand hält.

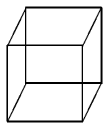
Vor ihm liegt eine Zeichnung auf dem Boden.



Der unmögliche Würfel

Die schauen wir uns mal genauer an.

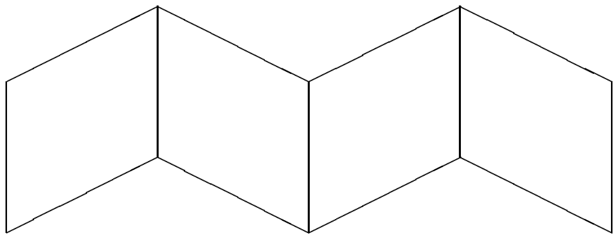
Was ist an diesem Würfel paradox?



Der paradoxe Würfel

Das Entscheidende ist, daß wir uns hier für eine Vorder- und eine Rückseite entscheiden müssen. Es stehen zwei Möglichkeiten zur Auswahl, welche ist die Richtige? Im ersten Fall werden wir vielleicht die quadratische Fläche links unten als Vorderseite betrachten, und der Fall ist klar. Es ist aber auch möglich, daß die Fläche rechts oben die Vorderseite ist.

Es ist eine Frage der persönlichen Perspektive bzw. der Gewohnheit, welcher Variante man den Vorzug gibt. Ein weiteres Beispiel für ein zweideutigkeites Bild ist das folgende:



Hierbei besteht das Problem in den vertikalen Linien. Sie können parallel zum Untergrund, dem Blatt, verlaufen. In diesem Fall kann man sich zwei aufgeschlagene Bücher vorstellen, die mit dem Rücken nach oben nebeneinander liegen. Sie können aber auch senkrecht stehen, wie bei einer spanischen Wand — bekannt aus älteren Filmen, hinter denen sich dann die hübschen Frauen umzogen. Nun, der Phantasie sind keine Grenzen gesetzt ....

Im allgemeinen aber wird die Phantasie bei solchen Bildern kaum angeregt, von gewissen Ausnahmen einmal abgesehen. Um abschweifende Diskussionen zu vermeiden, was mit dem dargestellten Bildern alles gemeint sein kann, werden wir an unsere Zeichnungen gewisse Voraussetzungen stellen:

- Es gibt keine Schatten oder Bruchlinien.
- Verdeckte Kanten sind nicht sichtbar. Das Beispiel des paradoxen Würfels ist also im folgenden nicht erlaubt.
- Alle Eckpunkte sind Schnittpunkte genau dreier aufeinandertreffender Flächen. Die Spitzen der Cheops-Pyramiden sind damit nicht erlaubt.
- Ein allgemeiner Beobachtungspunkt wird verlangt. Bei geringen Ortsveränderungen des Punktes darf kein Schnittpunkt seinen Typ wechseln.

Diese Forderungen sind auch sinnvoll hinsichtlich der Komplexität des Problems. Es wird in mancher Hinsicht einfacher.

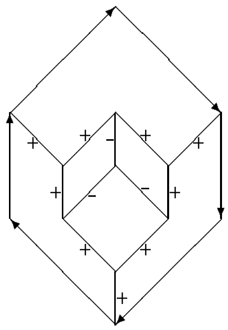
Nun müssen noch die verschiedenen Schnittpunkt- und Linientypen definiert werden. Es werden folgende Linien unterschieden:

Begrenzungslinien Sie bilden die äußeren Kanten des Objektes. Gekennzeichnet werden sie mit einem Pfeil „$\rightarrow$“. Die Pfeile der Begrenzungslinien sind so gerichtet, daß die Körperfläche immer rechts liegt.

Innenlinien Sie sind die Kanten im Inneren des Objektes. Es werden zwei Arten unterschieden.

konkave Linien Beide Begrenzungsflächen schließen den Beobachtungsstandpunkt ein. Ein Beispiel wäre der Blick in ein geöffnetes Buch. Sie sind mit einem „−“ markiert.

konvexe Linien Beide Grenzflächen sind vom Beobachter abgewandt, wie bei einem Würfel. In der Zeichnung sind sie mit einem „+“ versehen.



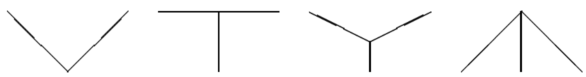
Durch Beschriften der einzelnen Kanten einer Zeichnung erhalten wir zum Beispiel einen großen Würfel, bei dem an einer Ecke ein kleiner Würfel ausgespart ist.

Sind auch andere Beschriftungen möglich und richtig?

Wenden wir uns nun den Schnittpunkten zu. Ziel unserer Betrachtungen ist es, eine richtige Beschriftung der Kanten zu finden. Ob dabei physikalisch widersprüchliche Konstellationen auftreten, kann man nur anhand der Schnittpunkte entscheiden. So ist ein Schnittpunkt von drei Begrenzungslinien nicht möglich.

Allerdings gibt es einige, physikalisch unmögliche Figuren (z.B. von Escher), die konsistent (ohne Widerspruch) beschriftet werden können.

Es stellt sich die Frage, welche Arten von Schnittpunkten es aufgrund unserer Voraussetzungen gibt. Es sind folgende vier:

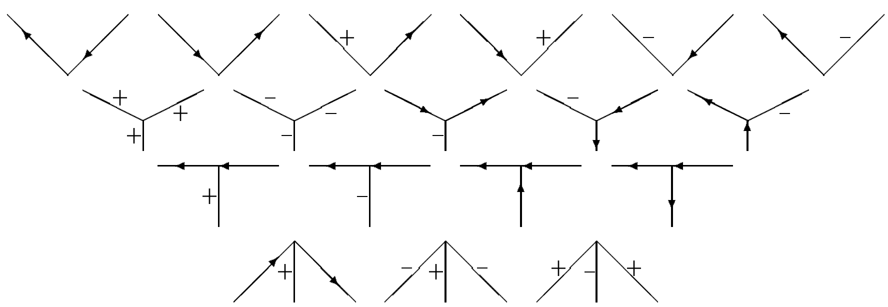


Ecke T-Stück Gabel Pfeil

Bis auf T-Stücke sind alle anderen in unserem Eingangsbild enthalten.

Wie viele Instanzen dieser vier Grundtypen kann es nun geben?

Unter Instanz ist dabei eine vollständige Eck- und Kantenbeschriftung zu verstehen. Bei Kanten gibt es die vier Möglichkeiten „←“, „→“, „−“ und „+“. Daraus ergeben sich 4^3 Beschriftungsmöglichkeiten für eine Gabel. Insgesamt also $4^3+4^3+4^3+4^2 = 208$ Möglichkeiten, von denen aber nur 18 physikalisch möglich sind.

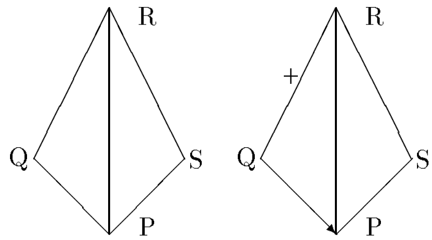


Für die praktische Arbeit sollen nun noch zwei Verfahren vorgestellt werden.

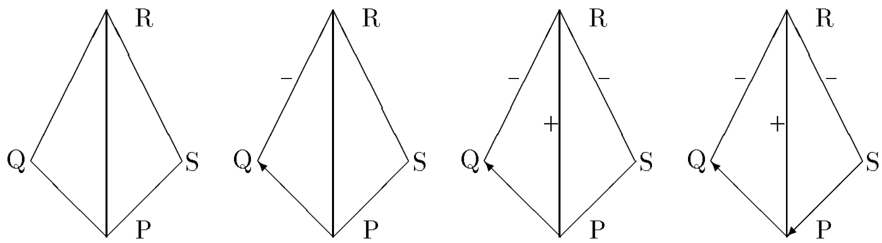
Verfahren Try-and-Error

Man beschriftet die Kante sukzessive. Bei einem Fehler nimmt man die letzte Beschriftung zurück und versucht es mit einer anderen (*Backtracking*).

Das wäre ein Versuch. Er bricht aber ab, da sich für P keine mögliche Eckkombination ergibt.



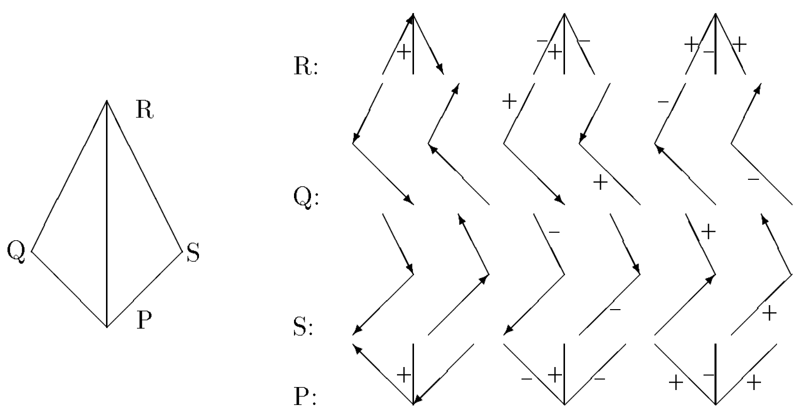
Durch *Backtracking* ergibt sich für unser Problem folgende Lösung:



Das Verfahren liefert uns zwar eine Lösung des Problems, aber dabei werden alle Varianten einfach durchprobiert. Dadurch ist die Effektivität nicht besonders hoch.

Verfahren von Waltz

Es werden jeweils Eckpaare (*Kanten*) betrachtet und nicht verträgliche Eckteile gestrichen. Betrachten wir dazu das gleiche Beispiel wie oben:



Links ist das zu beschriftende Gebilde zu sehen und rechts alle möglichen Varianten für die entsprechden Kanten.

Betrachten wir nun die Kante QR. Die Beschriftung nach Punkt R bschränkt sich auf „$\rightarrow$“, „+“ und „$-$“, die für Punkt Q auf „$\leftarrow$“, „$\rightarrow$“, „+“ und „$-$“. Die richtige Beschriftung muß im Durchschnitt beider Mengen liegen. Dadurch fallen die Varianten 1 und 4 für Q weg. Nun kann man eine anliegende Kante betrachten, z.B. Kante QP, und es würde die dritte Möglichkeit für Q wegfallen. Das wiederum hat Auswirkungen auf Kante QR. Diese könnte nicht mehr konvex („+“) sein und dadurch entfällt die dritte Möglichkeit bei R. Der Rest der Kanten läuft nach demselben Ausschlußverfahren.

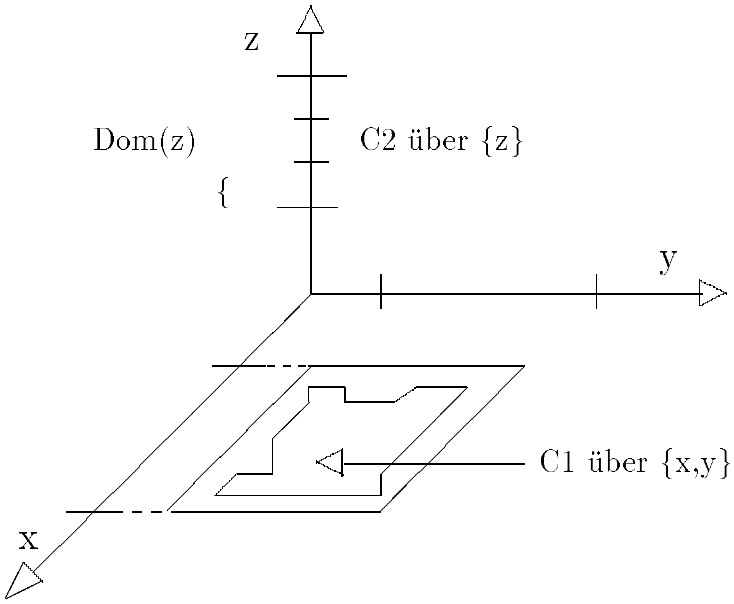
Auch dieser Algorithmus findet eine Lösung, falls eine existiert. Zusätzlich ist es hier leicht festzustellen, ob mehrere Lösungen (Mehrdeutigkeiten) existieren.

3.2 Formale Beschreibung

Im folgenden suchen wir eine Variablenbelegung für die Lösung eines Problems. Ein Beispielproblem wäre: Finde zwei natürliche Zahlen x und y, so daß $x + y = 7$ unter der Vorausetzung $x > y > 2$.

Definition *Constraint*:

Gegeben ist eine Variablenmenge $V = \{v_1, \ldots, v_n\}$ mit den Wertebereichen $Dom(v_i)$ für alle v_i mit $i = 1, \ldots, n$. Sei wieterhin $Dom(V) = Dom(v_1) \times \ldots \times Dom(v_m)$.



1. Ein *Constraint* C über $V' = \{v'_1, \ldots, v'_m\} \subseteq V$ ist eine Teilmenge
$$C \subseteq Dom(v'_1) \times \ldots \times Dom(v'_m) \quad .$$
2. Ein *Constraint-Netz* $\mathcal{C}$ über V ist eine Menge $\mathcal{C} = \{C_1, \ldots, C_k\}$, wobei jedes C_j ein Constraint über einer Menge $V_j \subseteq V$ ist.
3. Sei β eine Belegung, die jedem $v_i \in V$ ein Element aus $Dom(v_i)$ zuordnet. Eine Belegung β der Variablen *erfüllt* ein Constraint C über $V' = \{v'_1, \ldots, v'_m\}$, falls $[\beta(v'_1), \ldots, \beta(v'_m)] \in C$. Dann ist $[\beta(v'_1), \ldots, \beta(v'_m)]$ eine *lokale Lösung* für C.
4. Eine lokale Lösung heißt *globale Lösung*, falls β alle $C \in \mathcal{C}$ erfüllt.

Erläutern wir nun diese abstrakte Definition am Eingangsbeispiel:

$$V = \{x, y\} \qquad Dom(x) = Dom(y) = \mathbb{N}$$

$$\begin{array}{ll} C_1 \subseteq Dom(y) & C_1 = \{y \in \mathbb{N} : y > 2\} \\ C_2 \subseteq Dom(x) \times Dom(y) & C_2 = \{[x, y] \in \mathbb{N} \times \mathbb{N} : x > y\} \\ C_3 \subseteq Dom(x) \times Dom(y) & C_3 = \{[x, y] \in \mathbb{N} \times \mathbb{N} : x + y = 7\} \end{array}$$

$$\mathcal{C} = \{C_1, C_2, C_3\}$$

Eine lokale Lösung bezüglich C_1 ist $x = 1$ und $y = 3$. Für eine globale Lösung müssen alle drei Bedingungen erfüllt werden. Eine passende Belegung β ist $x = 4$ und $y = 3$.

Bei der Suche nach Lösungen können folgende Probleme auftreten:

- Passen die einzelnen lokalen Lösungen zusammen, d.h. sind die einzelnen *Constraints* nicht widersprüchlich (inkonsistent)?
  Ein Beispiel wäre die Forderung $x + y = 6$ statt $x + y = 7$.
- Gibt es einen Algorithmus, der aus lokalen Lösungen globale Lösungen extrahiert (*Constraint Propagation*)?

Definition :

Gegeben sei ein Constraint-Netz $\mathcal{C}$ über $V = \{v_1, \ldots, v_n\}$ und zu jeder Variablen v_i sei $D_i \subseteq Dom(v_i)$.

1. $D = D_1 \times \ldots \times D_n$ heißt *lokal konsistente Einschränkung* bezüglich $C \in \mathcal{C}$, falls für alle $d_i \in D_i$ mit $i = 1, \ldots, n$ eine lokale Lösung $[a_1, \ldots, a_{i-1}, d_i, a_{i+1}, \ldots, a_n] \in D$ für C existiert.
2. $D = D_1 \times \ldots \times D_n$ heißt *global konsistente Einschränkung*, falls für alle $d_i \in D_i$ mit $i = 1, \ldots, n$ eine globale Lösung $[a_1, \ldots, a_{i-1}, d_i, a_{i+1}, \ldots, a_n] \in D$ für alle $C \in \mathcal{C}$ existiert.
3. D heißt *inkonsistent*, wenn D keine globale Lösung enthält.

Bemerkung :

1. Es gibt globale Lösungen gdw. global konsistente Einschränkungen existieren. Falls eine globale Lösung existiert, so existiert auch eine maximale global konsistente Einschränkung.
2. Falls D inkonsistent ist, so enthält D keine global konsistente Einschränkung.
3. D kann globale Lösungen enthalten (D ist nicht inkonsistent), ohne selbst global konsistente Einschränkung zu sein.

Beispiel :

1. Es sei ein Constraint-Netz $\mathcal{C} = \{C_1, C_2, C_3, C_4\}$ wie folgt gegeben:

$$\begin{array}{lll} V = \{x, y, z\} & \text{und} & Dom(x) = Dom(y) = Dom(z) = [0, 1] \\ \mathcal{C} = \{C_1, C_2, C_3, C_4\} & & \\ & & \\ C_1 = \{[x, y] : x > y\} & \text{für} & V_1 = \{x, y\} \\ C_2 = \{[y] : y > 0.5\} & \text{für} & V_2 = \{y\} \\ C_3 = \{[x, z] : x + z = 1\} & \text{für} & V_3 = \{x, z\} \\ C_4 = \{[x, z] : x < z\} & \text{für} & V_4 = \{x, z\} \end{array}$$

Eine lokale Lösung bezüglich C_3 wäre $[0.5, 0.7, 0.5]$. Eine lokal konsistente Einschränkung dazu wäre

$$D = \{[x, y, z] : y = \frac{\sqrt{2}}{2} \text{ und } x^2 + z^2 \leq 1\} \quad .$$

Eine global konsistente Einschränkung kann es nicht geben, weil keine globale Lösung existiert. Die gegebenen Voraussetzungen sind inkonsistent. Aus C_1, C_2 und C_4 folgt nämlich

$$z > x > y > 0.5 \quad .$$

Nimmt man die Einschränkung C_3 mit $x + z = 1$ dazu, ergibt sich ein Widerspruch.

2. Sei das Constaint-Netz wie oben definiert, aber ohne C_4. Wenn

$$\begin{aligned} D_x &= \{0.7, 0.9\} \\ D_y &= \{0.6, 0.8\} \\ D_z &= \{0.1, 0.3\} \end{aligned}$$

ist, ist die Menge $D = D_x \times D_y \times D_z$ global konsistente Einschränkung.

Jetzt greifen wir noch mal die Frage nach einem Algorithmus zum Finden von global konsistenten Einschränkungen auf. Für die Beantwortung ist es nützlich, sich einmal Anwendungsbeispiele von *Constraint-Satisfaction* vor Augen zu führen.

Bilderkennung und Beschriftung der Kanten. Die zu belegenden Variablen sind die Kanten mit $Dom(Kante) = \{„\rightarrow“, „\leftarrow“, „-“, „+“\}$. Die Constraints bestehen in der Existenz von nur wenigen physikalisch möglichen Ecktypen.

Gleichungssysteme. Dabei ist es egal, ob es sich um linare oder nichtlineare Systeme handelt, auch Ungleichungen sind möglich.

Lösungen für Diophantische Gleichungen

Das letzte Anwendungsgebiet ist das berühmte **10. Hilbertsche Problem**, formuliert im Jahre 1900. Gerade diese Anwendung liefert den Schlüssel zur Beantwortung unserer Frage. Man hat herausgefunden, daß es algorithmisch unentscheidbar ist, ob es eine solche Lösung gibt, und somit existiert auch kein deterministisches Verfahren, das alle Lösungen generiert.

Für uns heißt das, daß global konsistente Einschränkungen für dieses Problem nicht *immer* algorithmisch aus den lokalen Einschränkungen berechnet werden können. Wenn es für dieses spezielle Problem keinen Algorithmus gibt, dann kann es auch keinen Algorithmus für alle derartigen Probleme geben, da er sonst auch das spezielle Problem lösen würde. Diese Schlußweise nennt man *Reduktion*.

Obwohl wir uns dieses Dilemmas bewußt sind, wollen wir nicht verzagen und wenigstens ein Schema angeben, über dessen Unzulänglichkeiten später noch diskutiert wird.

Schema für Constraint Propagation

1. Wähle $D \subseteq Dom(V)$ als initiale Einschränkung.
2. Wähle ein $C \in \mathcal{C}$.
3. Bestimme eine neue Einschränkung $D \subseteq D_{alt}$, so daß D eine lokal konsistente Einschränkung bezüglich C ist.
4. Zwischenauswertung, dabei können folgende Fälle auftreten:
   (a) Ich habe eine globale Lösung in D gefunden.
   (b) $D = \emptyset$.
   (c) Es ist keine weitere Einschränkung zu erwarten, da schon alle C probiert wurden, ohne daß sich D ändert. Wann tritt d) auf?
   (d) D ist inkonsistent.

   Daraus ergeben sich folgende Möglichkeiten zur Weiterarbeit:
   - Weiter bei Punkt 2, falls keiner der vier Fälle zutrifft.

- Backtracking zu Punkt 3 für 4.b), 4.c) oder 4.d).
- Wenn Backtracking nicht aussichtsreich ist oder 4.a) erfüllt ist, kann das Verfahren gestoppt werden.

Bemerkung :

1. Die Mengen D sind im allgemeinen keine global konsistenten Einschränkungen.
2. Durch das Einschränken in Schritt 3 wird D lokal konsistente Einschränkung bezüglich des C aus Schritt 2. Normalerweise ist D keine lokal konsistente Einschränkung bezüglich der früheren C's. Eine Möglichkeit, diesen Effekt zu verhindern, ist die sukzessive Einschränkung von D durch vollständig wechselnde C.
3. Hinweise zur Wahl von D in Schritt 3:
   - Wird D zu klein gewählt, können Lösungen verloren gehen (eventuell D inkonsistent).
     - D wird durch das Backtracking vergrößert.
     - Dabei lehrt die Erfahrung, daß nicht chronologisch (PROLOG), sondern widerspruchsorientiert vorgegangen werden sollte.
   - Wird D zu groß gewählt, dauert der Prozeß lange, eventuell unendlich.

Satz :

1. Gelten bei der Anwendung des Schemas folgende Voraussetzungen:
   - $D = Dom(V)$ im Schritt 1 und
   - D ist maximal lokale Einschränkung von D_{alt} im Schritt 3,

   dann enthält D stets die maximal global konsistente Einschränkung, sofern eine solche existiert.
2. Wenn das Verfahren durch Bedingung 4.c) abbricht, dann enthält D eine globale Lösung, falls sie existiert.

Beweis :

- zu 1: Der Beweis ist indirekt.
  Sei $D(1), D(2), \ldots, D(i), \ldots$ die Folge der konsistenten Einschränkungen D.
  Dann existiert ein $D(i)$, mit $D(i)$ enthält die maximal global konsistente Einschränkung (mindestens im Schritt 0).
  Aber $D(i+1)$ enthält diese Einschränkung nicht.
  Es gilt $D(i+1) \subset D(i)$ und es folgt die Existenz einer globalen Lösung a mit $a \in D(i)$ und $a \notin D(i+1)$.
  Somit kann $D(i+1)$ keine maximal lokale Einschränkung von $D(i)$ sein, weil a ja auch lokale Lösung ist und in $D(i+1)$ nicht mehr enthalten ist.
- zu 2: Der Beweis folgt aus 1.

Bemerkung :

Die Vewendung dieses Verfahren führt nicht immer zu einer Lösung. Dies ist Fall, wenn es keine gibt oder das Verfahren nicht abbricht.

Beispiel :

1.

$$V = \{v_1, v_2\} \quad \text{und} \quad Dom(v_1) = Dom(v_2) = \mathbb{N}$$

$$\begin{array}{lll} C_1 = \{[d_1, d_2] : d_1 = d_2\} & \text{für} & V' = \{v_1, v_2\} \\ C_2 = \{[0]\} & \text{für} & V' = \{v_1\} \end{array}$$

Schema:

$$\begin{array}{lll} (1) & D = \mathbb{N} \times \mathbb{N} & \\ (2) & C_1 & \\ (3) & D = \{[1,1]\} & \\ (2) & C_2 & \\ (3) & D = \emptyset & \text{Inkonsistenz, deshalb Backtracking} \end{array}$$

2. Die Vorausetzungen seien wie oben gegeben

$$\begin{array}{llll} (1) & D = \mathbb{N} \times \mathbb{N} & & \\ (2) & C_2 & (3) & D_1 = \{0\} \quad D_2 = \{\mathbb{N}\} \\ (2) & C_1 & (3) & D_1 = \{0\} \quad D_2 = \{0\} \\ (2) & C_2 & & \text{keine Änderung,} \\ (2) & C_1 & & D \text{ bleibt konstant} \end{array}$$

Wie man sieht, ist die Reihenfolge der Regelauswahl durchaus entscheidend.

3.

$$V = \{v_1, v_2\} \quad \text{und} \quad Dom(v_1) = Dom(v_2) = [0, 1]$$

$$\begin{array}{lll} C_1 = \{[d_1, d_2] : d_1 = d_2\} & \text{für} & V' = \{v_1, v_2\} \\ C_2 = \{[d_1, d_2] : d_1 = 2d_2\} & \text{für} & V' = \{v_1, v_2\} \end{array}$$

Schema:

$$\begin{array}{llll} (1) & D = [0,1] \times [0,1] & & \\ (2) & C_2 & (3) & D_1 = [0,1] \quad D_2 = [0, 0.5] \\ (2) & C_1 & (3) & D_1 = [0, 0.5] \quad D_2 = [0, 0.5] \\ (2) & C_2 & (3) & D_1 = [0, 0.5] \quad D_2 = [0, 0.25] \end{array}$$

Es ergibt sich ein unendlicher Prozeß bei jeweils maximaler, lokal konsistenter Einschränkung.

4.

$$V = \{v_1, v_2, v_3\} \quad \text{und} \quad Dom(v_i) = \{a, b\}$$

$$\begin{array}{lll} C_1 = \{[d_1, d_2] : d_1 \neq d_2\} & \text{für} & V' = \{v_1, v_2\} \\ C_2 = \{[d_1, d_3] : d_1 \neq d_3\} & \text{für} & V' = \{v_1, v_3\} \\ C_3 = \{[d_2, d_3] : d_2 \neq d_3\} & \text{für} & V' = \{v_2, v_3\} \end{array}$$

Schema:

$$\begin{array}{llll} (1) & D = \{a,b\} \times \{a,b\} \times \{a,b\} & & \\ (2) & C_1 & (3) & D \text{ bleibt konstant} \\ (2) & C_2 & (3) & D \text{ bleibt konstant} \\ (2) & C_3 & (3) & D \text{ bleibt konstant} \end{array}$$

Der Algorithmus bricht ab (nach Punkt 3 der Weiterarbeit), es ergibt sich keine Lösung und $D \neq \emptyset$.

Aufgabe :

Man formalisiere folgende Aufgabe und finde die Lösung mit dem Schema für *Constraint-Propagation.*

Bei einem 100m Lauf kamen die sieben gestarteten Läufer wie folgt im Ziel an:

- Bernd direkt vor Joachim, aber unmittelbar hinter Roger,
- Dieter direkt hinter Jürgen und
- Hermann erzielte den mittleren der sieben Plätze.

Außerdem ist von den Läufern folgendes bekannt

- Bernd und Dieter kommen aus Greifswald.
- Den zweiten Platz schaffte ein Rostocker.
- Uwe hat blonde Haare.

Ferner seien die Beziehung Name – Person eineindeutig und jeder hat nur einen Wohnort.

Welchen Platz erreichte jeder der sieben Läufer?

Kapitel 4

Wissensrepräsentation und Lösungsstrategien

4.1 Einleitung

Die Sprache hat sich herausgebildet, um Gedanken und Gefühle zum Ausdruck zu bringen. Dabei vollbringen wir eine Transformation von Semantik zur Syntax; beim Lesen ist es umgekehrt. Die Sprache stellt Wörter (Symbole) und Vorschriften zu deren Anordnung (Grammatik) bereit. Die Semantik gibt an, mit welchen Bedeutungen die einzelnen Symbole belegt (interpretiert) werden und stellt somit ein Modell dar.

Es ist kein Zufall, daß hier Begriffe aus der Logik fallen. Ein Ableitungskalkül (Axiome + Vorschriften, wie Formeln umgeformt werden), wie in einem Theorembeweiser implementiert, arbeitet immer nur syntaktisch mit den Formeln. Semantisches Folgern kann dann nur in der gleichen Interpretation der neugewonnenen Formeln erfolgen. Die Eigenschaft der Vollständigkeit eines Regelsystems sichert, daß alles was gefolgert (semantisch) auch syntaktisch abgeleitet werden kann.

Nachdem wir uns klargemacht haben, daß wir bloß syntaktisch (Bitmanipulation) arbeiten können, wollen wir einige Anforderungen an eine gute Repräsentation benennen und einige Probleme aufzeigen. Eine gute Repräsentation sollte:

- Möglichkeiten bieten, Objekte zu benennen und deren Beziehungen untereinander zu beschreiben
- menschlich anschaulich, aber trotzdem rechnerkonform sein

Das erste Merkmal erfordert natürlich eine geeignete Datenstruktur, und da Folgerungen gezogen werden sollen, auch eine entsprechende Auswertestrategie.

$$
\begin{tikzcd}
\text{Datenstruktur} \arrow[r] & \text{explizites Wissen} \arrow[d, "\text{Inferenz}"] \\
\text{Datenstruktur} & \text{implizites Wissen} \arrow[l, "\text{Interpretation}"']
\end{tikzcd}
$$

Dabei sind folgende Auswertungsverfahren zu berücksichtigen:

- formales Schließen (syntaktisches Ableiten),

- prozedural,
- Suchen (Probieren von Möglichkeiten),
- Simulation (Modelle auswerten),
- Fallvergleich (Erfahrung) und
- Metaschlüsse.

Metaschlüsse erfordern Meta-Inhalte. Dazu muß man vom Inhalt der Aussage abstrahieren und Aussagen über Aussagen treffen. Eines der bekanntesten Ergebnisse in dieser Richtung sind wohl die Unvollständigkeitssätze des Wiener Mathematikers Kurt Gödel aus den 30-er Jahren. Ein Teilresultat ist, daß es keine Axiomatisierung (Menge von Axiomen + Ableitungsregeln) der Arithmetik der natürlichen Zahlen derart gibt, daß auch alle wahren Aussagen durch das Axiomensystem beweisbar wären. Das ist deshalb so bemerkenswert, weil der bekannte Mathematiker David Hilbert Anfang des 20. Jahrhunderts nach einem solchen System gesucht hat. Er hat sich also eine Stufe tiefer an den Inhalt der Aussagen geklammert.

Für den Informatiker sind solche Meta-Schlüsse wohl besonders bei der Suche nach effizienten Algorithmen wichtig.

Es gibt sogenannte NP-vollständige Probleme, Probleme bei denen man keinen deterministischen, polynomiellen Algorithmus kennt (und fast jeder glaubt, daß es keinen gibt, nur bewiesen ist es noch nicht). Wenn man solch ein Problem vor sich hat, ist es wohl Zeitverschwendung (?), dort einen polynomiellen Algorithmus zu suchen und man sollte den Metaschluß, daß es keinen gibt, akzeptieren.

4.1.1 Formen der Wissensdarstellung

Zustandsraumsuche

Dieses Modell lehnt sich an die Automatentheorie an. Die Knoten sind dabei Zustände, oder natürlicher ausgedrückt, Situationen, und es gibt Operatoren (Kanten) als Zustandsübergänge. Dabei ist:

- explizites Wissen: Ausgangszustände Operatoren (Zulässigkeitsbedingungen)
- implizites Wissen: erreichbare Zustände, Operatorenfolge
- Inferenzstrategie: Graph-Suche

Problemzerlegung

Hier geht es um die Umsetzung gegebener Fakten in ein logisches Grundschema. Folgender Sachverhalt sei gegeben:

Ein Mann kauft ein Auto, wenn er Geld hat und ein Auto braucht. Er hat Geld, wenn er gearbeitet, geklaut oder etwas verkauft hat. Er braucht ein Auto, wenn er damit fahren muß oder zuviel Werbung gesehen hat.

Dieser Sachverhalt in Prolog[1] dargestellt:

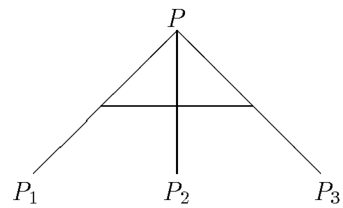
```
kauft(Mann,auto)   :- hat_geld(Mann), braucht(Mann,auto).
hat_geld(Mann)     :- hat_gearbeitet(Mann).
```

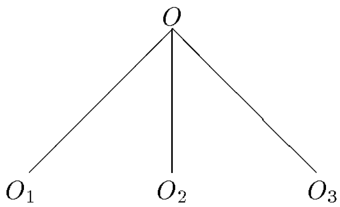
[1] Man beachte, daß `Mann` (großgeschrieben) in PROLOG lediglich eine Variable ist und auch mit einer Frau belegt werden kann

```
hat_geld(Mann)      :- hat_geklaut(Mann).
hat_geld(Mann)      :- hat_verkauft(Mann,Irgend_etwas).

braucht(Mann,auto) :- muss_fahren(Mann).
braucht(Mann,auto) :- hat_zuviel_Werbung_gesehen(Mann,auto).
```

Dies ist aber auch in sogenannten UND/ODER- Bäumen darstellbar.

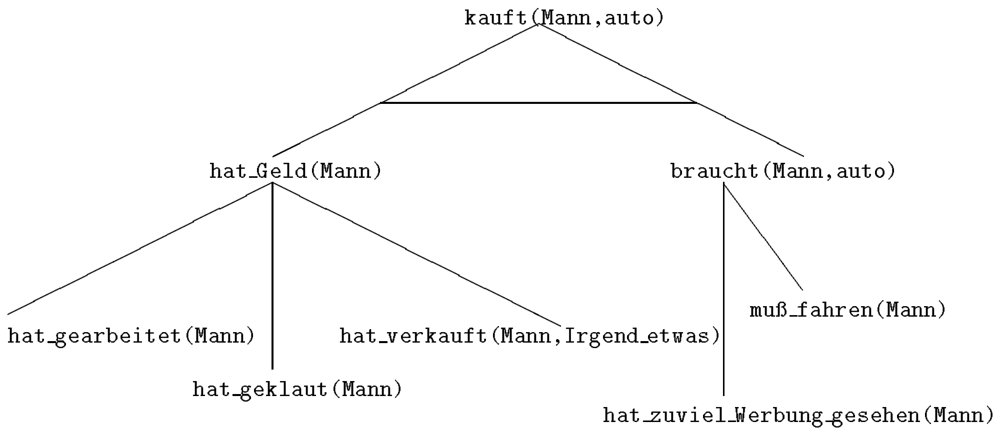
Wenn P ein Problem ist, das die Lösung aller Teilprobleme $P_1, \ldots, P_n$ erfordert, dann wird dies durch das nebenstehende Bild ausgedrückt (UND-Verknüpfung).





Wenn O ein Problem ist, welches gelöst ist, wenn eines der Teilprobleme $O_1, \ldots, O_n$ gelöst ist, dann haben wir diesen Teilbaum (ODER-Verknüpfung).

Der Baum für unser Beispiel sieht so aus:



Nun sind solche Repräsentationen schön und gut, aber der Rechner kann damit noch nichts anfangen. Sie müssen erst noch compiliert werden.

4.1.2 Compilation in der Wissensverarbeitung

Prozedurale Programmiersprachen (PASCAL, C) stellen eine Möglichkeit dar, Algorithmen zu beschreiben. Solch ein Programm wird dann in Maschinencode übersetzt, der auch prozedural abgearbeitet wird.

In der KI sollen Probleme selbständig gelöst werden, und deshalb sind die Sprachen auch mehr deklarativer (beschreibender) Natur. Trotzdem soll ein ausführbarer Maschinencode herauskommen. Deshalb müssen solche Sprachen um Auswertungs-(Inferenz-)komponenten angereichert werden. Diese Anreicherungen dienen zum einen zur Erhöhung der Effektivität, als auch zur Abarbeitung überhaupt. Solche Ergänzungen können sein:

- Auswertungsstrategien,
- Standardwissen,
- Streichen von nicht benötigtem Wissen und
- Vorverarbeitung (Constraints nur einmal verwenden).

4.2 Regelbasierte Systeme (Produktionssysteme)

Die Grundidee ist hierbei das Darstellen der Welt anhand der Ausgangssituation und den dazugehörigen WENN-DANN-Regeln.

Es wird eine Regel angewendet, wenn ihr WENN-Teil zur aktuellen Situation (Menge von Fakten) paßt. Manchmal passen mehrere Regeln und man muß eine auswählen. Die Regelanwendung kann eine neue Situation bewirken.

Situation	+	`IF`	*Bedingung*	`THEN`	*Aktion*	$\Longrightarrow$	*Aktion*
	passen		|		|		
	(*matchen*)		linke Seite		rechte Seite		

Einige Sachverhalte kann man ganz gut damit beschreiben, z.B. :

Fakt: *heute_ist_Sonntag*

Regeln:

IF *ich_bin_etwas_beschwipst*	***THEN*** *ich_fahre_kein_Auto*
IF *heute_ist_arbeitsfrei*	***THEN*** *es_gibt_Sektfrühstück*
IF *heute_ist_Sonntag*	***THEN*** *heute_ist_arbeitsfrei*
IF *ich_hab_Sekt_getrunken*	***THEN*** *ich_bin_etwas_beschwipst*
IF *es_gibt_Sektfrüehstück*	***THEN*** *ich_hab_Sekt_getrunken*

Diese Regeln und Fakten sind explizites Wissen. Implizites Wissen wäre alles das, was man daraus folgern könnte (Wissen produzieren, deshalb Produktionssysteme), also, daß ich Sekt trinke, etwas beschwipst bin und kein Auto fahre.

Ein anderes (etwas ernsthafteres) Beispiel:

Türme von Hanoi : *Umsetzen eines Turmes auf eine andere Stelle mit der Maßgabe, daß zwischendurch nur Türme entstehen. Dazu stehen 3 Plätze zur Verfügung. Es gibt einen Startplatz, auf dem der Turm steht, einen Zielpatz, auf den der Turm umgesetzt werden soll, und einen Platz als Zwischenspeicher. Ein Turm ist ein Stapel von Scheiben, wobei eine größere Scheibe nie auf einer kleineren liegt (geordnet). Während der Umsetzung muß jeder Turm geordnet sein.*

Zuerst formalisieren wir, das heißt, wir einigen uns über den Objektbereich (hier nullstellige Funktionssymbole) und die Beziehungen zwischen ihnen (Prädikate).

nullstellige Funktionssymbole

$S_1, \ldots, S_n$ - Scheiben

T_1, T_2, T_3 - Plätze für Türme (im weiteren auch als Scheiben anzusehen, die größer als gewöhnliche Scheiben und nicht verschiebbar sind)

Prädikatssymbole

x, y und z seien im folgenden Variablen, wobei

$$x \in \{S_1, ..., S_n\} \qquad y, z \in \{S_1, ..., S_n\} \cup \{T_1, T_2, T_3\} \quad .$$

Prädikat		Bedeutung
`liegt_auf`(x, y)	-	x liegt direkt auf y.
`kleiner_als`(x, y)	-	x ist kleiner als y.
`oben`(y)	-	y liegt oben auf einem der 3 Stapel.

Die Ausgangssituation kann nun formalisiert beschrieben werden:

- `liegt_auf`(S_i, S_{i+1}) mit $i = 1, \ldots, n-1$
- `liegt_auf`(S_n, T_1)
- `oben`(S_1)
- `oben`(T_2)
- `oben`(T_3)
- `kleiner_als`(S_1, S_2), `kleiner_als`(S_1, S_3), ..., `kleiner_als`(S_1, S_n)

$\vdots$

- `kleiner_als`(S_{n-2}, S_{n-1}), ..., `kleiner_als`(S_{n-2}, S_n)
- `kleiner_als`(S_{n-1}, S_n)
- `kleiner_als`(S_1, T_i), `kleiner_als`(S_2, T_i), ..., `kleiner_als`(S_n, T_i) mit $i = 1, 2, 3$

Nun fehlen noch die Produktionsregeln und eine Zielregel. Diese ergeben sich auf natürliche Weise.

```
IF oben(x) AND oben(y) AND kleiner_als(x,y)
  AND liegt_auf(x,z)
THEN
    REMOVE oben(y)
    REMOVE liegt_auf(x,z)
    ADD liegt_auf(x,y)
    ADD oben(z).

IF oben(S1) AND oben(T1) AND oben(T2)
THEN
    stop.
```

4.2.1 Komponenten von Regelsystemen

Unser Einführungsbeispiel hat hoffentlich deutlich gemacht, worum es bei Regelsystemen geht. Es soll mit Hilfe von Regeln versucht werden, von einem Ausgangszustand zu einem Zielzustand zu gelangen. Die Regeln sind die Übergangsfunktionen.

Ein Regelsystem besteht aus 3 Komponenten:

- Regelmenge,
- Datenbasis und
- Interpreter/Ablaufsteuerung.

Datenbasis (Arbeitsspeicher)

Das Regel- oder Produktionssystem produziert Wissen. Das aktuell gültige Wissen wird im Arbeitsspeicher hinterlegt und als Menge von *Working-Memory-Elements* (*WME*) bezeichnet. Solche WME's sind Fakten (Prädikate) und in unserem Beispiel wären das z.B. zur Zeit der Ausgangssituation:

`liegt_auf`(S_1, S_2), `liegt_auf`(S_2, S_3) ...
`oben`(S_1), `oben`(T_2), `oben`(T_3), usw.

Die Datenbasis kann mit `REMOVE` und `ADD` verändert werden. Die entsprechenden WME's werden entfernt oder hinzugefügt (siehe Beispiel).

Regelmenge

Es muß zwischen Regel und Regelinstanz unterschieden werden.

Regel:

IF	$Bedingung_1$	*THEN*	$Aktion_1$
	$Bedingung_2$		$Aktion_2$
	$\vdots$		$\vdots$
	$Bedingung_m$		$Aktion_m$
	LHS *(left hand side)*		***RHS*** *(right hand side)*

Dabei können in LHS und RHS Variablen auftreten. Jede mögliche Belegung dieser macht aus einer Regel eine Regelinstanz.

Regelinstanz: Regel + Belegung der Variablen

Zur Abarbeitung gelangen immer die Regelinstanzen. Dazu muß die linke Seite der Regelinstanz wahr werden (LHS ist eine logische Verknüpfung von Fakten (WME's), z.B

oben(S_1) **AND oben**(S_2) **AND kleiner_als**(S_1,S_2)
AND liegt_auf(S_1,S_4)

Regelinstanzen liegen nicht statisch, sondern immer nur dynamisch vor. Es muß also während der Abarbeitung eine Belegung der Variablen einer Regel gefunden werden, so daß die linke Seite der resultierenden Regelinstanz wahr wird. Diesen Vorgang zum Finden einer Belegung nennt man *Matching*. Wenn die rechte Seite der Regelinstanz abgearbeitet wird, die ursprüngliche Regel also angewendet wird, kommt es zum *Feuern* der Regel bzw. Regelinstanz.

Interpreter/Ablaufsteuerung

Wann wird was groß geschrieben?

Er steuert den *RECOGNIZE-ACT-ZYKLUS* und umfaßt 3 Phasen.

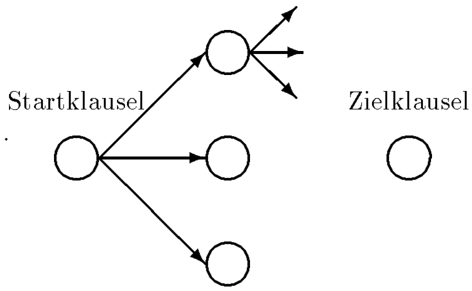
1. MATCHING Bestimmung der feuerbaren Regelinstanzen.
2. CONFLICT RESOLUTION Im Allgemeinen gibt es mehrere feuerbare Regelinstanzen (auch wenn es nur 1 Regel gibt). Die Aufgabe besteht in der Auswahl einer Regelinstanz.
3. ACTION Abarbeiten der ausgewählten Regelinstanz.

Nun kann die Ablaufsteuerung prinzipell auf 2 Arten geschehen. Entweder man versucht das Ziel ausgehend von der aktuellen Situation zu erreichen oder dem Ziel näher zu kommen (*FORWARD-CHAINING*), oder es wird ausgehend vom Ziel nach Regeln gesucht, durch deren Ausführung das Ziel erreicht werden würde, und die Vorbedingungen dieser Regelinstanzen werden zu neuen Zielen (*BACKWARD-CHAINING*). PROLOG wäre ein Vertreter der letztgenannten Steuerung. Schließlich ist auch eine Verkettung beider Techniken denkbar.

Vorwärtsverkettung - FORWARD-CHAINING

Verfahren (zunächst ohne Backtracking)

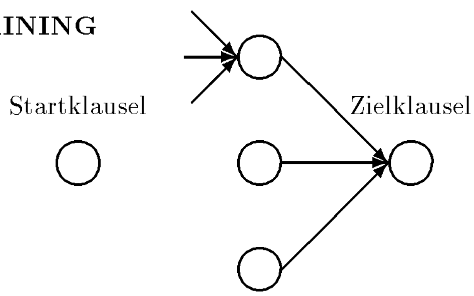
1. Bestimmen der Regelinstanzenmenge
   $\mathcal{R} := \{R : R$ feuert bei initialer Datenbasis$\}$.
2. $\mathcal{R} = \emptyset \longrightarrow EXIT(no)$
3. Wähle $R \in \mathcal{R}$ (CONFLICT RESOLUTION).
4. R abarbeiten (ACTION).
5. Ziel erreicht? $\longrightarrow$ EXIT(yes)
6. $\mathcal{R}$ aktualisieren.
   - nicht mehr feuerbare Regeln entfallen
   - neu feuerbare Regeln werden zu $\mathcal{R}$ hinzugenommen
   - eventuell R streichen (oder in CONFLICT RESOLUTION dafür sorgen, daß R nicht im nächsten Schritt ausgewählt wird, um Zyklen zu verhindern)
7. weiter mit 2.



Rückwärtsverkettung - BACKWARD-CHAINING

Verfahren:

1. $C := \{$ zu erfüllende Ziele $\}$
2. $C = \emptyset \longrightarrow$ EXIT(yes)
3. Wähle $Z \in C$.
4. Wenn Z gemäß Datenbasis erfüllbar ist,
   dann setze $C := C/\{Z\}$, weiter mit 2.
5. Existieren Regeln, die Z erfüllen würden, dann wähle eine Regel R_1 davon aus, setze $C := C \setminus \{Z\} \cup \{$ Vorbedingungen der Regel $R_1\}$ und mache bei 2. weiter.
6. EXIT(no) sonst.



4.2.2 Suchstrategie – oder welche Regel ist die richtige?

Um etwas zu erfinden, sind zwei nötig. Der eine stellt Kombinationen her; der andere wählt aus, erkennt, was er wünscht und was für ihn wichtig ist in der Masse der Dinge, die der erste ihm überlassen hat. Was wir Genie nennen, ist nicht so sehr das Werk des ersteren als vielmehr die Bereitschaft des letzteren, den Wert dessen, was ihm vorgelegt wurde, zu erfassen und daraus auszuwählen.

Der Dichter und Philosoph Paul Valéry (1871–1945)

Bei den beiden oben genannten Verfahren steht man im allgemeinen immer vor dem Problem, welche der anwendbaren Regeln günstig und auszuwählen ist. Theoretisch wäre es vorstellbar, alle möglichen Regelinstanzen,deren Anzahl sei n, in einem Zustand anzuwenden, wodurch n neue Zustände zu betrachten wären. Wäre einer dieser Zustände der Zielzustand, hätte man eine Lösung gefunden.

Wäre das nicht der Fall, müßte für jeden dieser n Zustände die jeweilige Menge der passenden Regelinstanzen abgearbeitet werden. Werden im Durchschnitt mal angenommen m Regelinstanzen abgearbeitet, wären im nächsten Schritt dann schon $n \cdot m$ Zustände zu betrachten. …

Dieses Verfahren wird Breite-Zuerst-Suche genannt und ist eigentlich nur dann günstig, wenn der minimale Weg (Folge von Regelanwendungen) zwischen Ausgangs- und Zielzustand gesucht wird. Dabei muß der Weg relativ kurz sein, weil anderenfalls die Anzahl der zu betrachtenden Zustände zu groß wird (kombinatorische Explosion).

Die Breite-Zuerst-Suche wird also in den meisten Fällen an der Komplexität scheitern. Deshalb werden wir nicht alle Regelinstanzen abarbeiten, sondern nur eine einzige, und zwar die, von der wir uns erhoffen, daß sie uns der Lösung (Ziel) ein Stück näher bringt.

Folgende Gesichtspunkte der Auswahl sind möglich:

- Keine Regel/Regelinstanz wiederholen, um Schleifen zu vermeiden.
- Jüngere Regelinstanzen bevorzugen. Führt zur Tiefe-Zuerst-Suche
- Ältere Regelinstanzen bevorzugen. Führt zur Breite-Zuerst-Suche
  Dabei ist das Alter einer Regelinstanz abhängig vom Alter derjenigen WME's, die im Bedingungsteil der Instanz stehen. Das Alter der WME ergibt sich aus dem Zeitpunkt der Erzeugung (Produktion) oder letzten Änderung jener WME.
- Regeln mit vielen Bedingungen bevorzugen. Dadurch kann man Ausnahmen behandeln, z.B.

```
IF sitzt_in_Bibliothek(x)              THEN arbeitet(x)
IF sitzt_in_Bibliothek(x) AND muede(x) THEN schlaeft(x)
```

- Metaregeln bevorzugen.

4.2.3 Strukturierung von Wissen

Um Probleme zu lösen, braucht man nur relativ wenige, dafür aber die richtigen Regeln. So wird es wenig nützlich sein, zum Beispiel beim Umformen einer mathematischen Gleichung vielleicht Wetterregeln anwenden zu wollen. Man kann sich beim Suchen nach Regeln auf die beschränken, die im Kontext der Mathematik zu finden sind. So könnte man die Menge aller Regeln verschiedenen Gebieten zuordnen. Also :

$$
\begin{array}{ll}
\texttt{IF}\ context_1\ \texttt{AND}\ \ldots & \texttt{THEN}\ \ldots \\
\texttt{IF}\ context_1\ \texttt{AND}\ \ldots & \texttt{THEN}\ \ldots \\
\vdots & \\
\texttt{IF}\ context_2\ \texttt{AND}\ \ldots & \texttt{THEN}\ \ldots \\
\texttt{IF}\ context_2\ \texttt{AND}\ \ldots & \texttt{THEN}\ \ldots
\end{array}
$$

Um zu erfahren, in welchem Kontext gearbeitet wird, wären Metaregeln anwendbar.

$$\texttt{IF}\ \ldots \qquad \texttt{THEN ADD}\ context_1$$

Die Vorteile einer solchen Strukturierung liegen auf der Hand.

- Modularität,
- Erklärbarkeit,
- Erweiterbarkeit und
- Übersichtlichkeit.

Nachteile sind:

- höherer Steuerungsaufwand und
- anwendbare Regeln aus anderen Kontexten werden nicht gefunden

4.2.4 Backtracking – oder Play it again, ...

Nun kann es trotz noch so ausgefeilter Konfliktlösungsstrategien passieren, daß die ausgewählten Regeln nicht zum Ziel führen.

Wann wird dieser Fakt bemerkt?

Wenn keine Regel mehr anwendbar ist, war der Weg eine Sackgasse. Wenn immer wieder Regeln anwendbar sind, kann man sich auch auf dem Holzweg befinden. Die Entscheidung, wann ein anderer Weg einzuschlagen ist, ist nicht trivial.

Vergleichbar ist die Situation vielleicht mit der eines Jungunternehmers, dessen Geschäft nicht wie erwartet läuft und der zum Überleben immer neue Kredite von der Bank braucht. Wann sagt diese *nein*?

Wenn nun festgestellt wurde, daß die bisherige Regelauswahl nicht zum Ziel führt, muß *man* sich revidieren, einen alten Zustand wieder herbeiführen und eine andere Regel auswählen. Die bisherigen Zustände und die dadurch hervorgerufenen Regelanwendungen müssen für ein solches Vorgehen protokolliert werden.

Auf welchen Zustand setze ich das System zurück?

Diese Frage ist insbesondere für die Effizienz wichtig. PROLOG geht immer nur zum letzten Zustand zurück und versucht dort eine andere Regelanwendung. Effektiver (aber nicht so sicher) sind semantische Abhängigkeiten (intelligentes Backtracking). In PROLOG läßt sich so etwas begrenzt durch ein `!` (Cut) simulieren. Begrenzt deshalb, weil die Abhängigkeiten dem Programmierer bekannt sein müssen.

4.2.5 Unsicheres Wissen

Nun lassen sich viele Dinge in der Wirklichkeit nicht in immergültige Regeln fassen, sondern diese Regeln sind mit Unsicherheiten behaftet. Eines der anschaulichsten Beispiele dafür ist wohl der tägliche Wetterbericht. Dort wird zum Beispiel, als Konsequenz dessen, von Regen*wahrscheinlichkeit* gesprochen. Es kann aber auch Unsicherheit bezüglich eines Faktes bestehen. Zum Beispiel (siehe oben):

$$\texttt{sitzt\_in\_Bibliothek}(x)\ \texttt{\&}\ \texttt{muede}(x) \rightarrow \texttt{schlaeft}(x)$$

Das Prädikat `sitzt_in_Bibliothek(`x`)` kann noch einen Booleanwert bekommen, aber bei `muede(`x`)` ist das schon nicht mehr der Fall.

Es hat sich eine Disziplin der Logik, die *Fuzzy-Logik*, herausgebildet, die sich genau mit solchen Problemen der Unsicherheiten befaßt. Dort muß man den Grad des Vertrauens in einzelne Regeln und Fakten bewerten, und die Fuzzy-Logik stellt Mittel bereit, wie diese Unsicherheiten zu propagieren sind und welchen Grad des Vertrauens man zum Ergebnis haben kann.

In unserem Beispiel wäre das der Grad des Vertrauens in den Fakt `schlaeft(`x`)`. Dieser Grad ist in den meisten Fällen eine (reelle) Zahl zwischen 0 und 1, und es ist nicht falsch, sich darunter die Wahrscheinlichkeit für das Ereignis `schlaeft(`x`)` vorzustellen.

Wir wollen nun aber nur auf ein für die Informatik zwar wichtiges, aber ohne größere Schwierigkeiten aus der elementaren Wahrscheinlichkeitstheorie ableitbares Theorem hinaus.

Bayes Theorem

Seien U und S Ereignisse. Dann gilt:

$$P(Ursache/Symptom) = \frac{P(Symptom/Ursache) \cdot P(Ursache)}{P(Symptom)}$$

$$P(U/S) = \frac{P(S/U) \cdot P(U)}{P(S)}$$

Wenn $S_1, \ldots, S_n$ unabhängige Ereignisse sind, gilt allgemeiner:

$$P(U/S_1\& \ldots \& S_n) = P(U) \cdot \frac{P(S_1/U) \cdot \ldots \cdot P(S_n/U)}{P(S_1\& \ldots \& S_n)}$$

Dabei sei:

$P(U)$ – Wahrscheinlichkeit für Eintritt des Ereignisses U
$P(S_1\&S_2)$ – Wahrscheinlichkeit für gleichzeitiges Auftreten der Ereignisse S_1 und S_2
$P(U/S)$ – Wahrscheinlichkeit für Eintritt des Ereignisses U unter der Voraussetzung, daß Ereignis S eingetreten war.

Dann gilt:

$$\begin{aligned} P(U\&S) &= P(U/S) \cdot P(S) \\ P(U\&S) &= P(S/U) \cdot P(U) \\ \Rightarrow 1) \qquad P(U/S) &= P(U) \cdot \frac{P(S/U)}{P(S)}. \end{aligned}$$

Nach dem Multiplikationssatz für die Wahrscheinlichkeitsrechnung gilt für unabhängige Ereignisse $S_1, \ldots, S_n$:

$$P(S_1\& \ldots \& S_n) = P(S_1) \cdot \ldots \cdot P(S_n).$$

Wenn $S_1, \ldots, S_n$ unabhängig, dann sind auch Einschränkungen dieser Ereignisse unabhängig, also insbesondere $S_1/U, \ldots, S_n/U$. Dann gilt:

$$\Rightarrow 2) \qquad P(S_1/U) \cdot \ldots \cdot P(S_n/U) = P(S_1/U\& \ldots \& S_n/U) = P((S_1\& \ldots \& S_n)/U).$$

Aus 1) und 2) ergibt sich:

$$\begin{aligned} P(U/(S_1\& \ldots \& S_n)) &= P(U) \cdot \frac{P((S_1\& \ldots \& S_n)/U)}{P(S_1\& \ldots \& S_n)} \\ &= P(U) \cdot \frac{P(S_1/U) \cdot \ldots \cdot P(S_n/U)}{P(S_1) \cdot \ldots \cdot P(S_n)}. \end{aligned}$$

4.3 OPS5 – Official Production Rule System

OPS5 ist nun ein System, in dem spezielle Produktionssysteme geschrieben werden können. Die allgemeinen Konzepte der Produktionssysteme können hier in konkreten Realisierungen wiedergefunden werden, insbesondere Konfliktlösungsstrategien. Eine weitere Besonderheit ist der Zeitstempel, um junge/alte Regelinstanzen zu bestimmen.

Ein OPS5-Programm besteht aus:

- Angabe der Konfliktlösungsstrategie **LEX** oder **MEA**
- Definition der Datenstrukturen für den Arbeitsspeicher
  Sie erfolgt mit Hilfe von `literalize` z.B. `literalize block name size place` und bewirkt die Deklaration einer WME-Klasse mit

Name	`block`
Attribute	`name`
	`size`
	`place`

  Einzelne Instanzen von `block` werden mit `make` erzeugt.

- Produktionsteil (Regeln und Kommandos)

Für eine Syntaxbeschreibung wird auf den Anhang A verwiesen.

Für jedes WME wird der Zeitpunkt der Erzeugung bzw. letzten Änderung verwaltet. Das ist wie gesagt wichtig für die Konfliktlösungsstrategie. Ein Beispielprogramm ist im Anhang B zu finden.

Abarbeitung

Bestimmung der feuerbaren Regelinstanzen:

Die Bedingungen der LHS müssen mit den WME's matchen. Dazu:

1. Übereinstimmung der Klassennamen
2. Attribute matchen
   (a) Werte sind gleich ⟶ o.k.
   (b) In `<cond_element>` ist die Bedingung angegeben, die erfüllt wird z.B.
   `{ > 5  < 10 }` ⟶ UND-Verknüpfung,
   `<< < 5  > 10 >>` ⟶ ODER-Verknüpfung.
   (c) In `<cond_element>` steht die Variable (Notation: `<x>`) und kann entsprechend belegt werden.

Variablen:

- treten in Regeln auf,
- haben keinen Typ,
- haben auf Regelabarbeitung beschränkte Lebensdauer und
- sind, falls einmal belegt, dann in der ganzen Regel so belegt (Konsistenz). Bei negierten Bedingungen (`- <cond_element>`) erfolgt aber keine Variablenbindung. Diese (negierten) Bedingungen sind erfüllt, falls kein WME matcht.

Konfliktlösungsstrategien

generelles Vorgehen: Aussondern, bis nur eine Regelinstanz übrigbleibt.

Strategie LEX abgeleitet von lexikographischer Ordnung (siehe Schritt 2.)

1. Schritt *Refraction* (lt. Wörterbuch: (Strahlen-)Brechung)
   Regelinstanzen, die gefeuert haben, streichen (dabei Instanz = Regel + Variablenbelegung + Zeitstempel der Bedingungen (`<cond_element>`); falls danach die Regelinstanzenmenge leer ist, Vorgang abbrechen.
   Folgende Regel wäre immer wieder aufgrund des Zeitstempels anwendbar:
   ```
   (p loop
       (start)
   --> (make start))
   ```
2. Schritt *Recency* (dt. Neuheit)
   Hier wird lexikographisch vorgegangen. Die Menge der Zeitstempel der Bedingungen einer Regelinstanz wird geordnet (keine doppelt !) und zu einer Liste (Wort) zusammengefaßt. Danach wird die Regelinstanz mit der größten Liste ausgewählt. Analog: Welches Wort steht am weitesten hinten im Wörterbuch.
   Verfahren

   - Sei n_1 der größte Zeitstempel, der bei WME's in den Regelinstanzen der Konfliktmenge vorkommt.
     $\longrightarrow$ Alle Regelinstanzen, bei den n_1 nicht als Zeitstempel vorkommt, streichen. Falls keine Einermenge, mit Verfahren weiter. Ansonsten ist der Konflikt der Regelauswahl gelöst.
   - Sei n_2 der zweitgrößte Zeitstempel der aktuellen Konfliktmenge.
     $\longrightarrow$ Alle Regelinstanzen, bei denen n_2 nicht als Zeitstempel vorkommt, streichen. Wenn die Einermenge erreicht ist, dann analog weiter. Falls nichts mehr möglich und keine Einermenge erreicht, weiter bei 3.

3. Schritt *Test-Specificity*
   Instanzen mit mehr Tests (Bedingungen) bevorzugen.
4. Schritt *zufällige Auswahl*

Strategie MEA (*means-ends-analysis*)

Hier erhält das erste `cond_element` jeder Regel besonderes Gewicht. (Deshalb sind im Syntax auch keine negativen Bedingungen als erste Bedingungen zugelassen)

1. wie bei LEX
2. unterteilt sich in
   (a) Sei t_1 der maximale Zeitstempel der ersten Bedingung einer jeden Regel.
   $\longrightarrow$ Alle Instanzen, die t_1 nicht als ersten Zeitstempel haben, streichen.
   (b) weiter wie bei LEX

Durch die MEA-Strategie kann der Ablauf eines Programms besser bestimmt werden. (siehe im Beispiel: (`goal` ...))

Effizienz in OPS5 – der Rete-match-Algorithmus

Siehe Seite 21!

Normalerweise laufen während eines *Recognize-Act-Zyklus* folgende Schritte ab:

1. Bestimmung der Regelinstanzenmenge (*matchen*),
2. Konfliktlösung und
3. Aktion.

Nun verändert die Aktion meist nicht so viele WME's, so daß alle Regelinstanzen davon betroffen sind. Deshalb werden den WME's Verweise auf diejenigen Regeln mitgegeben, bei denen sie als `cond-elements` etwas bewirken können (Aktivierung). Dadurch läßt sich die Konfliktmenge beim nächsten Zyklus schneller bestimmen und auch schneller lösen.

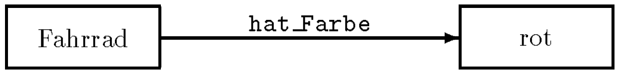
4.4 Semantische Netze

Semantische Netze dienen der strukturierten Wissensdarstellung und orientieren sich an natürlicher Spache. Daß dabei oft ein Teil des ursprünglichen Wissens verloren geht, werden wir noch sehen. Insofern haben Semantische Netze oft keine eindeutige Semantik.

Semantische Netze sind gerichtete Graphen. Dabei werden interpretiert:

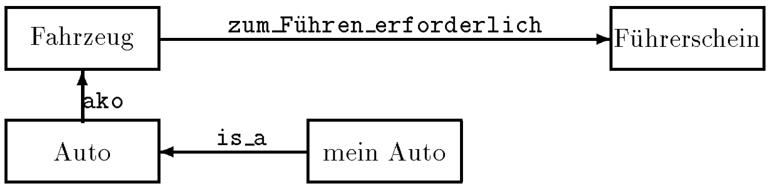
Knoten	$\longrightarrow$	Objekte
beschriftete Kanten	$\longrightarrow$	Relationen

Die Beschriftung der Kanten soll eine Bedeutung intendieren, so wie wir uns bei bestimmten Relationsnamen ja auch eine bestimmte Standardsemantik vorstellen.



Zur Inferenz innerhalbs eines solchen Netzes sind Eigenschaften von Kanten besonders wichtig. Einige Kanten mit bestimmten Eigenschaften sind:

`is_a`	Element einer Menge
`ako (a_kind_of)`	Teilmenge
`has_part`	Bestandteile

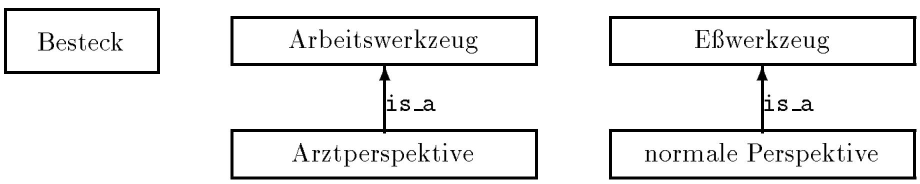


Eine Eigenschaft von **`ako`** und **`is_a`** ist, daß alle Aussagen der (Ober)Menge auch für Elemente oder Teilmengen gelten (Transitivität). So kann geschlossen werden:

Für mein Auto ist ein Führerschein erforderlich.

Begriffe haben auch eine starke Kontextabhängigkeit. Fragt man z.B. einen Chirurg nach Besteck, dann wird er vielleicht Skalpell usw. antworten, obwohl normalerweise Messer, Gabel und Löffel damit gemeint sind.

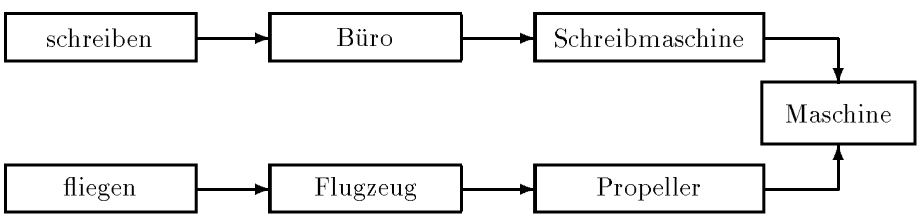
Zu diesem Zweck werden Knoten für die Perspektive eingeführt:



Die Bestimmung, daß es sich um eine Arztperspektive handelt, muß implizit durch den Text erfolgen. So sucht man nach anderen Wörtern, die dann zusammen eine eindeutige Perspektiveinordnung oder Bedeutungsbestimmung erlauben. Ein Beispiel für diesen Sachverhalt wäre:

Ich kann Maschine schreiben.
Meine Maschine fliegt 11.00 Uhr.

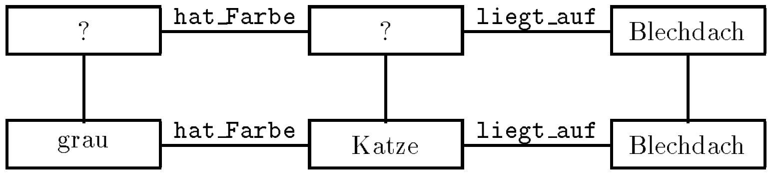
Als Wissensgrundlage sei folgendes Netz gegeben:



Hier wäre es z.B. möglich zu erkennen, daß man es im 1.Fall mit einer Schreibmaschine zu tun hat. Hierzu ist eine Verbindung zwischen Schreiben und Maschine zu suchen.

Suchen und Graph-Matching wird auch bei der Beantwortung von Fragen benötigt, z.B.:

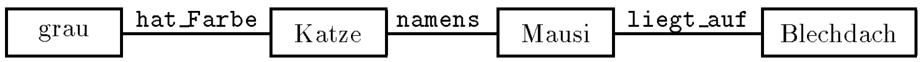
Welche Farbe hat das Objekt, das auf dem Blechdach liegt ?
Eine graue Katze liegt auf dem Blechdach.



Schon dieses (simple) Problem ist relativ schwierig zu lösen, denn der beste bekannte Graphmatchalgorithmus ist exponentieller Natur.

Hätte man aber als Datenbasis:

Eine graue Katze namens Mausi liegt auf dem Blechdach.



In diesem Fall müßte vom Namen abstrahiert werden, was dann noch mehr Aufwand erfordert. Wenn dieser Aufwand nicht in der benötigten Größenordnung betrieben werden kann, ist zu fragen was bei einem Abbruch gefolgert werden kann ?

Ein anderes Problem der Inferenz ist, daß Eigenschaften der Menge nicht notwendigerweise Eigenschaften der Elemente sind. So könnte die spezielle Katze *Mausi* auch schwarz sein. Dieses Problem kann mit sogenannten *default*-Werten gelöst werden. Das sind Eigenschaften der Menge, die bis auf Widerruf für alle Elemente gelten. Aber:

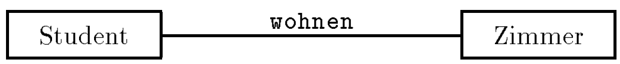
Inferenz: Darf der Tisch verheizt werden ?

Zum Schluß noch etwas zur Semantik. Um einen natürlich sprachlichen Text zu verstehen, hat der Mensch wirklich den ganzen Text und Alltagswissen zur Verfügung. Ein Computer soll nun aus einem Skelett des Textes und unzureichendem Alltagswissen Inferenzen ziehen.

Ein typisches Beispiel wäre das Wort „Student“! Ohne Kontext kann das sein:

ein spezieller Student
irgendein Student
alle Studenten
Berufsbezeichnung

Dadurch würden folgende Aussagen auf das gleiche semantische Netz abgebildet:



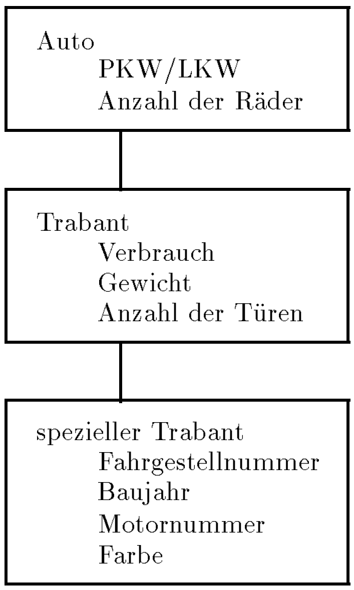
Der Student wohnt in dem Zimmer.
Der Student wohnt in irgendeinem Zimmer.
Alle Studenten wohnen in einem Zimmer.
Irgendein Student wohnt in einem Zimmer.
usw.

Wenn alle diese Aussagen in ein einziges Netz codiert werden können, dann kann dieses Netz auch als das alles interpretiert werden.

Ein anderes, schönes Beispiel für die mehrdeutige Semantik befindet sich im Buch *KI* von WINSTON S.282-283.

4.5 Frames – oder der Rahmen machts

Frames sind Musterbeschreibungen für Objekte. Sie enthalten Attribute, denen die speziellen Objekte einen bestimmten Wert zuweisen. Dazu ein Beispiel:



Wie in der OOP ist auch hier eine Vererbung möglich. Wenn jetzt mein Trabant betrachtet werden soll, wäre dies eine Instanz von spezieller Trabant. Es müßten die Attribute mit speziellen Werten belegt werden, also Farbe =„rot“, Baujahr=„1975“ usw. Weiterhin sind auch Attribute der Superframes zu belegen.

Das kann zum Teil automatisch passieren (z.B. Auto $\longrightarrow$ PKW, Anzahl der Räder = 4) oder auch mit default-Werten (z.B. Verbrauch = 8l/100km).

Default-Werte lassen sich überschreiben. So kann es sein, daß aufgrund einer defekten Benzinleitung mein Trabbi 11l/100km verbraucht. Das könnte bei dieser Instanz vermerkt werden. (Allerdings müßte dann auch am real existierenden Objekt etwas geändert werden.)

Die hier erwähnten Attribute werden auch als Slots (Schlitze) bezeichnet. Diese Schlitze sind bei einer Instanz mit Informationen aufzufüllen. So ist z.B. unter jeder Motorhaube ein Kärtchen zu finden, wo man Baujahr und Fahrgestellnummer aufgedruckt findet.

Diese Kärtchen existieren millionenfach, aber bei meinem speziellen Kärtchen wurde bei der Auslieferung das Baujahr und die Fahrgestellnummer eingestanzt und somit die Slots gefüllt. Damit ist es einzigartig geworden, sozusagen ein Unikat.

Nun können Slotwerte nicht nur einfache Werte sein. Ein typisches Beispiel wäre ein Fahrzeughalter, wobei es sich meistens um eine Person handelt. Uns interessieren an dieser Person nur bestimmte Aspekte, wie Name, Wohnort und Versicherungsklasse.

Frames haben also folgenden Aufbau:

$$
\begin{array}{llll}
\text{Framename} & & & \\
supersets & :: & < Liste\_von\_Framenamen > & \\
subsets & :: & < Liste\_von\_Framenamen > & \\
Slot_1 & : & Aspektname_{1,1} & Aspektwert_{1,1} \\
\vdots & & \vdots & \vdots \\
Slot_1 & : & Aspektname_{1,n} & Aspektwert_{1,n} \\
 & \vdots & & \\
Slot_k & : & Aspektname_{k,1} & Aspektwert_{k,1} \\
\vdots & & \vdots & \vdots \\
Slot_k & : & Aspektname_{k,j} & Aspektwert_{k,j}
\end{array}
$$

Supersets würde in Analogie zu den semantischen Netzen eine `ako`- bzw. `is_a`-Relation bedeuten.

Obwohl man sich bei einem Frame nur auf die wesentlichen Fakten beschränkt, können das noch zu viele sein. So ist es sicherlich nicht immer vonnöten zu wissen, wo beispielsweise der Fahrzeughalter wohnt. Oder die Slotwerte sind veränderlich, z.B. die Versicherungsklasse ist abhängig von aktuellem Zeitpunkt und letztem Unfall.

Diesen Aufwand der Berechnung treibt man daher nur im Bedarfsfalle. Die entsprechenden Prozeduren (Dämonen) werden mit `if_needed` spezifiziert. Sie liefern Werte für den Slotwert. Desweiteren gibt es noch `if_added`-Prozeduren.

Prozeduren (Dämonen)

- `if_needed`
  - Aktivierung bei Bedarf
  - füllt Slotwert in Abhängigkeit von (anderen) Slotwerten

Sollte das nicht noch ausformuliert werden?

- `if_added` (Aktionen, Schreibprozeduren)
  - unvorhergesehene Erweiterung der Slotbedingungen ⟶ Konsequenzen abarbeiten (siehe Beispiel)
  - belegen/ändern eines Slotwertes ⟶ z.B. bei einer Änderung des Familienstandes auch die Steuerklasse ändern

Slotbedingungen

dito.

- Wertebereich
- Constraints (z.T. konträr bei Mehrfachvererbung; Beispiel: Trabant als PKW, Geschäftsinventar ⟶ Auswirkung auf Kostenrechnung)

Slotwerte können, wie wir gesehen haben, auf verschiedene Weisen belegt werden (z.B. *default*, Vererbung, `if_needed`).

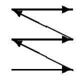
Aber für welche Belegungsart entscheidet man sich?

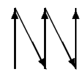
```
Frame1
subset :: Frame2

  Wert1:
   default   : 5
  Wert2:
   if_needed : IF Wert1 = 5   THEN 7
   default   : 8


Frame2
superset :: Frame1

  Wert1:
  Wert2:
   if_needed : IF Wert1 =\= 5  THEN 9
   default   : 10
```

Wenn eine Instanz von `Frame2` gebildet wird, deren `Wert1` jetzt wirklich 5 sein soll, wie groß ist dann der `Wert2`? Sicherlich wird die `if_needed`-Prozedur dem *default*-Wert vorzuziehen sein, aber was, wenn sie, wie in diesem Fall, kein Ergebnis bringt?

Frame1	Wert2	if_needed	default	Vererbung
Frame2	Wert2	if_needed	default	Vererbung

Es gibt 2 Möglichkeiten:

- horizontale Vorgehensweise 

  Belegung von `Wert2` in `Frame2` mit default (also 10)
- vertikale Vorgehensweise 

  Anwendung der ererbten `if_needed`-Prozedur aus `Frame1` (also 7)

Als Abschluß sei noch folgenes Restaurantbeispiel dargeboten:

```
Generic RESTAURANT Frame
  Specialization-of: Business-Establishment
  TYPES:
      range:      (Cafeteria, Seat-Yourself, Wait-To-Be-Seated)
      default:    Wait-To-Be-Seated
      if-needed:  IF plastic-orange-counter THEN Fast-Food,
                  IF stack-of-trays THEN Cafeteria,
                  IF wait-for-waitress-sign or reservations-made
                  THEN Wait-To-Be-Seated,
                  OTHERWISE Seat-Yourself
  Location:
      range:      an ADDRESS
      if-needed:  (Look at the MENU)
  Name:
      if-needed:  (Look at the MENU)
  Food-Style:
      range:      (Burgers, Chinese, American, Seafood, French)
      default:    American
      if-added:   (Update Alternatives of Restaurant)
  Times-of-Operation:
      range:      a Time-of-Day
      default:    open evenings except Mondays
  Payment-Form:
      range:      (Cash, CreditCard, Check, Washing-Dishes-Script)
  Event-Sequence:
      default:    Eat-at-Restaurant Script
  Alternatives:
      range:      all restaurants with same FoodStyle
      if-needed:  (Find all Restaurants with the same FoodStyle)
```

4.6 Scripts – das Drehbuch

Heißt das Scripts oder Scripte?

Scripts beschreiben einen zeitlichen Verlauf analog den Frames, die Objekte beschreiben. Dabei wird ein üblicher Ablauf zugrunde gelegt, der aber aufgrund von Werten variieren kann.

Beispiel:

> Nach der Arbeit gehe ich in die Kneipe, die Kellnerin schaut mich an und fragt: „Wie immer?“. Ich nicke und bekomme mein Bier serviert.

Das wäre ein üblicher Ablauf, wobei das Nicken ein default-Wert ist. Falls mit dem Kopf geschüttelt wird, hat der Ereignisablauf auch Alternativen parat, z.B. Essen oder Trinken bestellen.

Also abhängig von Slotwerten (in diesem Fall Bestellung) kommt ein ganz bestimmtes Verhalten zustande. Manchmal ist das Verhalten auch unbestimmt (man weiß nicht was zu tun ist), wobei Menschen dann meistens zum Unwohlsein oder überstürtzen Entscheidungen neigen.

Typische Inhalte eines Scripts sind also:

- Voraussetzungen (Anfangsbedingungen),
- Ergebnisse,
- Requisiten,
- Rollen und
- Ablauf mit Varianten.

An der Stelle muß der Fall von Scripts mindestens verhindert werden!

Aus Scripts kann man auch folgern. Aus unserem *Nach-der-Arbeit-in-die-Kneipe-gehen*-Script können erfahrungsgemäß Ehefrauen am meisten folgern.

Es geht aber auch umgekehrt. Aus dem Ergebnis (z.B. Kurt kam hunrig aus dem Restaurant) kann auf das Script (konkreter Ablauf) geschlossen werden. Wahrscheinlich hatte das Restaurant schon Küchenschluß.

Zum Teil sind Schlüsselwörter mit Scripts so stark verbunden, daß die Erwähnung des Wortes eine starke Assoziation auslöst. Wer denkt nicht bei Strumpfmaske an einen Banküberfall?

4.7 Repräsentation natürlicher Sprache

Das Verstehen natürlicher Sprache ist eines der härtesten Probleme der KI. Dessen Lösung würde einen gewaltigen Fortschritt auf folgenden Anwendungsgebieten darstellen:

- maschinelle Übersetzung,
- natürlichsprachliches Interface und
- intelligente Textverarbeitung.

Mit einem natürlichsprachlichen Interface könnten gerade auch Probleme des Alltagswissens (*commmon sense*) vielleicht in den Griff bekommen werden, denn wieviel Wissen nimmt der Mensch gerade durch Hören auf und ist sich dessen gar nicht bewußt. Ein Beispiel dafür ist: *Ich sah die Fliege, die die Mutter erschlug.* Dieser Satz wird erst eindeutig, wenn das Alltagswissen hinzugenommen wird. Sobald statt Fliege das Wort *Tochter* eingesetzt wird, denken wir uns den Sachverhalt eigentlich umgekehrt.

Wie denkt man umgedreht?

Ein weiteres Beispiel wäre: *Ich sehe den Mann auf dem Berg mit dem Fernrohr.*

Dieser Satz kann selbst mit Alltagserfahrung nicht erfaßt werden, hier fehlen noch eine Reihe von Informationen.

Die natürliche Sprache kann mehrdeutig und außerdem unlogisch sein. Als Beispiel ein einfacher Satz und seine Verneinung:

Satz	*Die Kinder schlafen.*
Negation	*Die Kinder schlafen nicht.*

In der Prädikatenlogik würde dieser Sachverhalt so dargestellt:

Formel	$\forall x : schlaeft(x)$
Negation	$\exists x : nicht\ schlaeft(x)$

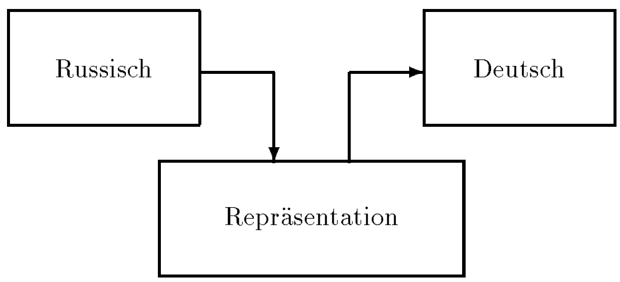
In der logischen Umsetzung erhält man dabei einen anderen Sachverhalt als bei der sprachlichen. Und trotzdem kommen die Menschen ganz gut damit zurecht. Vielleicht hat sich die Sprache gerade so herausgebildet, weil vom Menschen viele Entscheidungen rein emotional und nicht rational getroffen werden.

Weiterhin ist unser Gehirn nicht dafür gebaut, z.B. das Wachstum einer geometrischen Reihe abzuschätzen. Das machen sich vor allem Versicherungs- und Bankvertreter zunutze, wenn es um Zins und Zinseszins geht. Gefährlich wird dieser Umstand, wenn Planungsexperten sich bei der Abschätzung des Bevölkerungswachstums auf ihre Anschauung verlassen.

Zwei Experimente verdeutlichen diesen Fakt.

1. Man lasse schätzen, wie dick (theoretisch) ein 1mm starkes Blatt Papier wird, nachdem es 50 mal gefaltet wurde und rechne dann nach.
2. Man zeige jemanden eine Seite aus irgendeinem Buch, lege dieses Buch weg und sorge dafür, daß er (sie) jetzt nicht solch eine Schrift woanders sieht. Dann wird ein Groschen gezeigt und geschätzt, wieviele Buchstaben dieser Groschen überdeckt.

Nun zurück zu unserer Sprache. Die Repräsentation sollte unabhängig von der konkreten Sprache sein. Eine solche Repräsentation ist zum Beispiel bei der Übersetzung notwendig.



Im Russischen gibt es aufgrund der Landschaft unheimlich viele Wörter für Landschaftsbeschreibungen, Träumen usw. Es existieren zwar deutsche Übersetzungen, doch treffen sie oft nicht genau. Generell differenziert der Mensch bei Dingen, mit denen er viel zu tun hat, stärker. Somit waren neue Wörter für eine Nuance zweckmäßig.

Vergleichbar ist das mit einem Opernfan, der nur noch an ausgesprochenen Spitzentenören Gefallen findet, weil andere ihm zuviele Fehler machen. Fehler, die man als Laie zum Beispiel nicht bemerken würde.

Weiterhin sind in die Repräsentation solche Informationen einfließen zu lassen wie Gestik und Mimik, oder auch, für wen die Information bestimmt ist.

Wenn zum Beispiel ein Mecklenburger nach einer guten Mahlzeit gefragt werden wird, wie ihm denn nun das Essen geschmeckt hätte, und seine Antwort wäre: „Ja, konnte man essen.“, dann ist das für ihn ein ungeheures Lob. Die Bedienung würde das aber sicherlich als eine Beleidigung auffassen.

Nachdem nun die Anforderungen zusammengetragen wurden, einige Methoden.

4.7.1 Verwendete Methoden

Alles ein Bischen dürftig auskommentiert!

- Syntaxanalyse: Anhand der Grammatik den Satz in Subjekt, Prädikat und Objekt aufsplitten.
- Semantische Primitive für einzelne Gruppen: Dabei starke Kontextabhängigkeit beachten. Daraus ergeben sich die Beziehungen zwischen den Primitiven.

- Verweis bezüglich Pronomen
  (Beispiel: *Adam stieg in den Zug. Er fuhr nach Hamburg.*)
- Zeitbeziehungen herausstellen

Semantische Primitive

1. Tätigkeiten (Verben):
   (Fast) alle Tätigkeiten setzen sich aus Basisaktionen zusammen. Solche Basisaktionen sind: *come, get, give, go, keep, let, make, put, take, have, say, see, send.*
2. Akteure (Substantive):
   Die Akteure werden in Gruppen eingeteilt und aufgrund der jeweiligen Gruppe (z.T. mit Hilfe des Falls (Grammatik) bestimmt), zu der sie gehören, in Beziehung gesetzt.

   Beispiel: *John öffnet mit dem Schlüssel die Tür.*

Fälle:	`Agent`	*John*
	`Counter_Agent`	
	`Object`	*Tür*
	`Result`	
	`Instrument`	*Schlüssel*
	`Source`	
	`Goal`	
	`Experiencer`	
	`Location`	
	`Time`	
	`Path`	

Nun können Verben durch *case*-Frames charakterisiert werden. Dadurch kann die Semantik eines Verbs eingeschränkt werden.

```
open
   Object     :    (notwendig)
   Instrument :    (optional)
   Agent      :    (optional)
```

4.8 Repräsentation von Raum und Zeit

Den Hintergrund bildet der sich neigende fahle Herbsthimmel. Der Wind treibt rasch dahineilende scheckige Wölkchen über eine ferne Gebirgskette. Davor wogt die mit Wermut bewachsene rotbraune Steppe. Eine Straße, noch schwarz und feucht vom letzten Regen, durchzieht sie. Zu ihren Seiten liegt in dichten Büscheln verdörrtes umgeknicktes Steppengras. Der ausgewaschenen Räderspur folgen - je weiter, desto verschwommener - die Fußstapfen zweier Wanderer. Die beiden Fußgänger selbst scheinen nur noch einen Schritt machen zu müssen, um hinter dem Rahmen zu verschwinden. Der eine von ihnen ....

Tschingis Aitmatow „Djamila“

Dieses Zitat zeigt doch, wie gut wir räumlich, d.h. in Bildern, denken können. Wessen Phantasie ist jetzt nicht angeregt worden? Aber der Computer?

Räumliches Denken ist Gewohnheit und nichts besonderes. Das liegt sicherlich auch daran, daß wir uns im Raum (begrenzt) bewegen können.

Zeitliches Denken dagegen ist eigentlich nur dem Menschen eigen und ein signifikantes Zeichen von Intelligenz. Die Folgen seiner Handlungen abzuschätzen, erfordert ein hohes Maß an Vorstellungskraft. Wenn beispielsweise ein Affe sich einen Stuhl besorgt, um an die zu hoch hängenden Bananen zu kommen, bezeichnen wir das als intelligentes Verhalten.

Raum und Zeit sind in der Sprache eng miteinander verquickt. So gibt es Präpositionen, die sowohl räumliche als auch zeitliche Aspekte zum Ausdruck bringen können. Solche wären: *in, nach, an, lange, ...* (im Gegensatz zu: *weit, bei, während, ...*).

Da eine Handlung einen zeitlichen Verlauf hat, können wahlweise ohne Sinnentstellung räumliche oder zeitliche Wörter gewählt werden, zum Beispiel:

Ich ging in den Hörsaal. Dort (dann) schlief ich ein.

Auf solche Bezugnahmen fußen auch einige Darstellungsformen von Raum und Zeit.

4.8.1 Mögliche Darstellungsformen

Bitte Einteilung in Abschitte überprüfen!
∘ ausform.
∘ pktstzg

- ohne Koordinaten z.B. *nach*, *nebeneinander*, *vor*, *während*, dabei wird die Welt durch relationale Beziehungen modelliert
- mit Koordinaten
  - kontinuierlich (Vorstellung)
  - diskret (rechnerkonform)
- punktförmig
  - modellhafte Abstraktion (z.B. ein punktförmiger Affe am torsionsfreien Draht)

Als Mittel für die Darstellung können unterschiedliche Konzepte genutzt werden:

Mengenlehre	$\longrightarrow$	in,...
Topologie	$\longrightarrow$	bei, an, zwischen
Metrik	$\longrightarrow$	nahe, bei, an
Anordnung	$\longrightarrow$	vor, hinter, in
Gestalt	$\longrightarrow$	rund, konvex

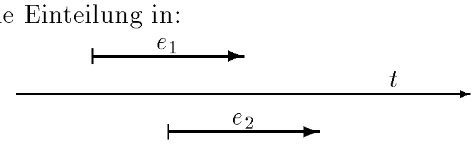
4.8.2 Zeit

$\nearrow$ S. 36 !

Zeitliche Vorausschau ist, wie schon erwähnt, signifikant für intelligentes Verhalten. Deshalb hat sich eine Reihe von Modellen und Kalkülen herausgebildet, um insbesondere die Probleme der Planung besser zu beherrschen. Generell läßt sich (wie in der KI wohl überhaupt) sagen, daß alle Modelle mehr oder minder ihre Vor- und Nachteile haben, und deshalb immer dasjenige, zum Sachverhalt passende Modell ausgewählt werden muß.

Eine grobe Klassifizierung wäre die Einteilung in:

- lineare Zeit

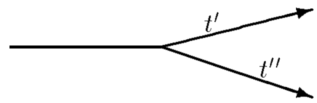


  Hier wird die zeitliche Abfolge von Ereignissen direkt in Bezug gesetzt (Ordnung). Da diese Abfolge nicht immer bekannt oder interessant ist, wird unterschieden in:

  1. Nebenläufigkeit - zeitliche Beziehung zwischen Ereignissen unbekannt
     „Als sie sich nach einer Stunde trafen, hatte er Brot (A) und Blumen (B) gekauft, während sie auf der Post (C) war."
     $\longrightarrow$ keine zeitliche Beziehung zwischen A, B, C.

   - Parallelität
     „Als er in das Geschäft ging, entschwand er ihren Blicken.“
   - Sequentialität
     „Nachdem er das Licht der Welt erblickt hatte, fing er an zu brüllen.“

- verzweigte Zeit

  Als Modell denkbar für Planung (Zukunft verzweigt) oder Diagnose (Vergangenheit verzweigt).
  „Wenn morgen schlechtes Wetter ist, gehe ich zur Uni, sonst an den Strand.“
  Wenn nur lineare Zeit zur Verfügung stehen würde, müßten Ereignisse in Beziehung gesetzt werden, die nichts miteinander zu tun haben, z.B. *esse_Mittag_in_der_Mensa* und *schlecke_Eis_in_der_Sonne*

Eine andere Klassifikation läßt sich durch Zeitbezug herstellen:

↗ S. 36 !

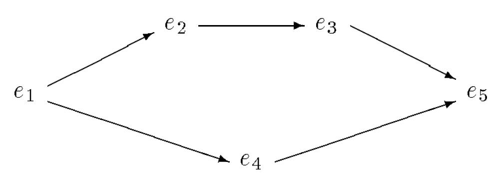
Darstellung ohne explizite Zeit

Hier wird der zeitliche Verlauf durch eine Abfolge von Ereignissen dargestellt. Dabei sind wieder die Möglichkeiten eines linearen Zeitverlaufs zu beachten:

- sequentiell

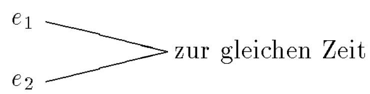
  $$e_1 \longrightarrow e_2 \longrightarrow \cdots \longrightarrow e_n$$

- nebenläufig (Halbordnung)

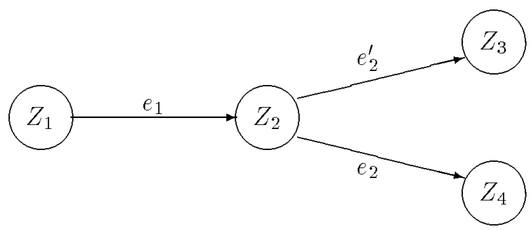
  

  Beobachter sieht nebenläufige Prozesse e_2 und e_4. Dann wären folgende Beobachtungen möglich:
  e_2 vor e_4, e_2 und e_4 parallel oder e_4 vor e_2.
  Aber immer ist z.B. e_1 vor e_2 $\longrightarrow$ zeitlich (oder kausal) abhängig.

- parallel

Auch das *Automatenmodell* kommt ohne explizite Zeit aus. Das ist ein Graph, in dem Zustände durch Knoten und Ereignisse durch Kanten dargestellt werden. Ereignisse sind von Zuständen abhängig, beeinflussen jene aber auch (siehe auch Produktionssysteme).



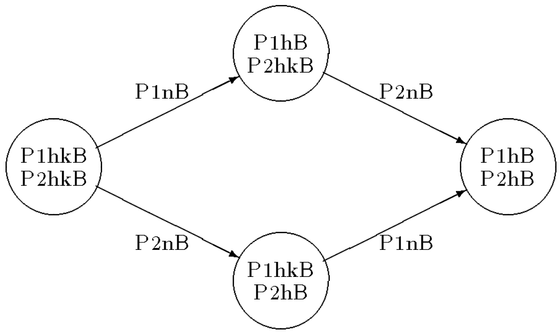
In den Knoten sind jetzt jeweils alle Parameter eines Systems kodiert. Das führt zu einer globalen Sicht, welche nicht immer zweckmäßig ist.

Beispiel: *Zwei Personen wollen Bücher kaufen.*

Als Abkürzungen werden verwendet:

P1hkB	- Person1 hat kein Buch	
P1hB	- Person1 hat Buch	Analog Person2
P1nB	- Person1 nimmt Buch	

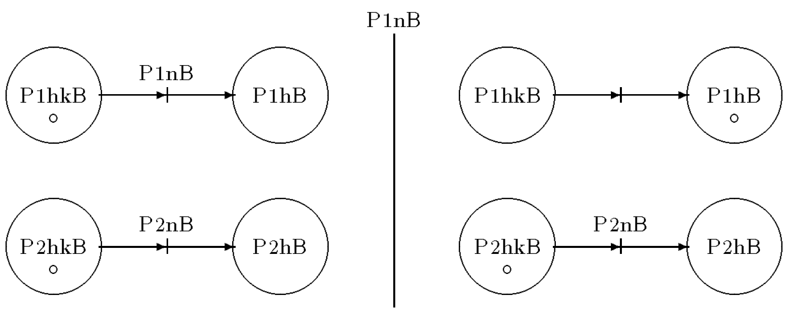
Das könnte folgenden Automaten implizieren:



Aufgabe

Man schreibe einen Automaten für dieses Szenario mit 5 Personen auf!

Da solche Automaten sich sehr schnell aufblähen, kann auch eine verteilte (lokale) Sichtweise angebracht sein. Als Beschreibungsmittel dafür haben sich die *Petrinetze* herausgebildet. Unser Bücherbeispiel würde wie folgt aussehen:



Die Knoten stellen (lokale) Bedingungen dar. Falls sie markiert sind, sind die Bedingungen erfüllt. Eine Transition (|) ist ein Ereignis. Diese bewirkt die Markierung der Nachbedingung (P1hB) und das Löschen der Markierung in den Vorbedingungen.

Darstellung mit expliziter Zeit

[↗ S. 36 !]

Hier wird nun direkt mit Zeiten operiert. Dabei müssen bestimmte Axiome erfüllt werden (siehe unten Punkt- und Intervallstrukturen).

Das Bezugnehmen auf die Zeit kann zum Beispiel mit Hilfe von modalen Operatoren (P, F, H, G) erfolgen.

Sei q ein Ereignis, so sind die modalen Operatoren:

Pq	Es war q.
Fq	q wird sein.

Hq	q war immer.
Gq	q wird immer sein.

In die Prädikatenlogik übersetzt bedeutet dies (t_0 entspricht Gegenwart):

$$\begin{array}{lcl} Pq & : & \exists t(t < t_0 \ \wedge \ q(t)) \\ Fq & : & \exists t(t > t_0 \ \wedge \ q(t)) \\ Hq & : & \forall t(t < t_0 \ \rightarrow \ q(t)) \\ Gq & : & \forall t(t > t_0 \ \rightarrow \ q(t)) \end{array}$$

Auch hier gibt es zwischen den Operatoren Äquivalenzen, z.B.:

$$\begin{array}{lcl} Gq & \longleftrightarrow & \neg F \neg q \\ Hq & \longleftrightarrow & \neg P \neg q \end{array}$$

Als Anwendung ist die Repräsentation der natürlichen Sprache denkbar.
Beispiel:

Peter schläft.	q	entspricht	$Q(t_0)$
Peter hat geschlafen./ Peter schlief.	Pq	entspricht	$\exists t(t < t_0 \ \wedge \ Q(t))$
Peter wird schlafen.	Fq	entspricht	$\exists t(t > t_0 \ \wedge \ Q(t))$
Peter hatte geschlafen.	PPq	entspricht	$\exists t \exists t'(t < t_0 \ \wedge \ t' < t \ \wedge \ Q(t'))$
Peter wird geschlafen haben.	FPq	entspricht	$\exists t \exists t'(t > t_0 \ \wedge \ t' < t \ \wedge \ Q(t'))$

Im weiteren werden wir 2 Formen der Zeitbeschreibung kennenlernen, die *Punkt*- und die *Intervallstruktur*.

Eine Struktur im prädikatenlogischen Sinne ist ein Tripel $[D, F, P]$, wobei D Gegenstandsbereich, F Menge von Funktionen über D und P Menge von Prädikaten ist. Damit wird also der Gegenstandsbereich, über den Aussagen gemacht werden soll, strukturiert. Im folgendem werden keine Funktionssymbole vorkommen und deshalb statt Tripel nur Paare angegeben werden.

Definition *Punktstruktur*:

1. Es ist $[T, \{<\}]$ *Punktstruktur*, falls
   T eine nichtleere Menge (Zeitpunkte) ist und
   $<$ binäre Relation über T (Präzedenz – lt. Duden früherer Fall).
2. Belegungsfunktion V: Aussagenvariable $\longrightarrow 2^T$
   Dabei ist $V(p)$ die Menge aller Zeitpunkte t, an denen p *true* zugeordnet wird.

   Das ist kein Satz!
3. Ein *Modell* ist ein Tripel $[T, <, V]$ wobei i.a. „$<$“ eine strikte partielle Ordnung (transitiv, irreflexiv) ist.

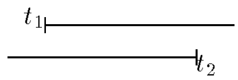
Definition :

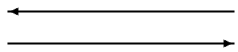
Ein Ausdruck A ist *wahr* zur Zeit t im Modell M ($M \models A[t]$), falls gilt:

$A[t] = p(t)$	d.h. $M \models p(t)$	falls $t \in V(p)$ für Variable p
oder		
$A[t] = \neg A'[t]$	d.h. $M \models \neg A'[t]$,	falls nicht $M \models A[t]$
oder		
$A[t] = [B \wedge C][t]$	d.h. $M \models [B \wedge C][t]$	falls $M \models B[t]$ und $M \models C[t]$
oder		
$A[t] = FA'[t]$	d.h. $M \models FA'[t]$	falls $\exists t' : t' > t \wedge M \models A'[t']$
oder		
$A[t] = PA'[t]$	d.h. $M \models PA'[t]$	falls $\exists t' : t' < t \wedge M \models A'[t']$

Die anderen modalen Operatoren G und H können durch Äquivalenzen mit P und F ausgedrückt werden.

Nun können je nach Zweckmäßigkeit gewisse Bedingungen an die Zeit gestellt werden. Einige davon wären:

- gebundene Zeit, $\exists t_1 \neg \exists t_2 (t_2 < t_1)$
  $\exists t_2 \neg \exists t_1 (t_1 > t_2)$
  

- ungebundene Zeit, $\forall t_1 \exists t_2 (t_2 < t_1)$
  $\forall t_1 \exists t_2 (t_2 > t_1)$
  
- diskret, $\forall t_1 \forall t_2 (t_1 < t_2 \rightarrow \exists t_3 (t_1 < t_3 \;\wedge\; \neg \exists t_4 (t_1 < t_4 < t_3)))$
  $\forall t_1 \forall t_2 (t_1 < t_2 \rightarrow \exists t_3 (t_3 < t_2 \;\wedge\; \neg \exists t_4 (t_3 < t_4 < t_2)))$
- dicht, $\forall t_1 \forall t_2 (t_1 < t_2 \rightarrow \exists t_3 (t_1 < t_3 < t_3)$
- kontinuierlich (Schnittaxiom Dedekindscher Schnitt),
  $$\forall P((\forall x \forall y (P(x) \wedge \neg P(y) \rightarrow x < y)) \wedge \exists x P(x) \wedge \exists y \neg P(y))$$
  $$\rightarrow \exists z (\forall u (z < u \rightarrow \neg P(u)) \wedge (\forall v (z > v \rightarrow P(v)))$$
  Dies ist ein Axiom, notiert in der Prädikatenlogik 2.Stufe. Als Beispiel für eine Relation P könnte $P(t) = <(t, \sqrt{2})$ genommen werden.
- linear, $\forall t_1 \forall t_2 (t_1 < t_2 \;\vee\; t_1 = t_2 \;\vee\; t_1 > t_2)$
  

- linkslinear oder
  $$\forall t_1 \forall t_2 \forall t_3 (t_2 < t_1 \;\wedge\; t_3 < t_1) \rightarrow (t_2 < t_3 \;\vee\; t_2 = t_3 \;\vee\; t_2 > t_3)$$
- rechtslinear.
  $$\forall t_1 \forall t_2 \forall t_3 (t_1 < t_2 \;\wedge\; t_1 < t_3) \rightarrow (t_2 < t_3 \;\vee\; t_2 = t_3 \;\vee\; t_2 > t_3)$$

Nun soll die Gegenstandsmenge keine Punktmenge, sondern eine Menge von Intervallen sein. Das hat den Vorteil, daß Ereignisse auch eine zeitliche Ausdehnung haben können.

Definition *Intervalstruktur*:

Eine *Intervallstruktur* ist ein Paar $[I, \{\subseteq, <\}]$, wobei

- I eine nichtleere Menge (von Intervallen),
- $\subseteq$ binäre Relation (die Teilmengenrelation) und
- $<$ binäre Relation ist.

Dabei sei $<$ wieder Präzedenz im dem Sinne, das ein Zeitintervall *vollständig vor* einem anderem liegt.

An die beiden Relationen werden noch gewisse Forderungen gestellt, wie:

- $\subseteq$ steht für eine partielle Ordnung (transitiv, reflexiv, antisymmetrisch).
- $<$ steht für eine strikte partielle Ordnung (transitiv, irreflexiv).

Intervallkalkül von Allen

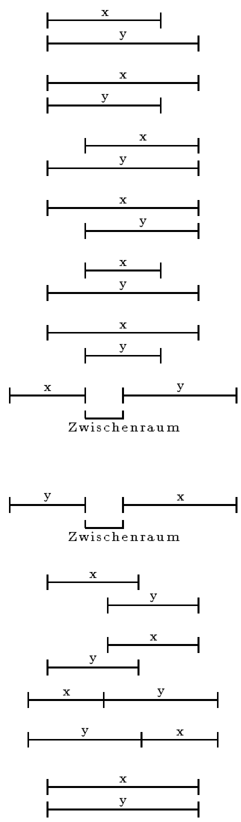
Dieser Kalkül dient zur Beschreibung von

- Zeit,
- Ereignissen,
- Aktionen (schieben Ereignisse an) und

- Absichten.

Es werden 13 Relationen zwischen Zeitintervallen eingeführt:

$s(x,y)$	STARTS(x,y)
$s_i(x,y)$	INVERSE von STARTS
$f(x,y)$	FINISHES(x,y)
$f_i(x,y)$	INVERSE von FINISHES
$d(x,y)$	DURING(x,y)
$d_i(x,y)$	INVERSE von DURING
$b(x,y)$	BEFORE(x,y) ist aber verschieden von Präzedenz, da trennendes Intervall erforderlich
$b_i(x,y)$	INVERSE von BEFORE(x,y)
$o(x,y)$	OVERLAPS(x,y)
$o_i(x,y)$	INVERSE von OVERLAPS
$m(x,y)$	MEETS(x,y)
$m_i(x,y)$	INVERSE von MEETS
$e(x,y)$	EQUALS(x,y)



Die inverse Relation zu EQUALS ist EQUALS wieder selber. Deshalb wird sie weggelassen. Die eingeführten Relationen müssen die folgenden Axiome erfüllen (so von Allen aufgestellt).

1. Für jedes Intervall t und jede der 13 Relationen r gibt es ein Intervall t' mit $r(t,t')$ bzw. $r(t',t)$.
$$\forall r\ \ [(\forall t_1 \exists t_2 : r(t_1,t_2)) \wedge (\forall t_2 \exists t_1 : r(t_1,t_2))]$$
2. Axiome, die den gegenseitigen Ausschluß der Relationen sichern, z.B.
$$o(x,y) \rightarrow \neg m(x,y)$$
3. Axiome zur Beschreibung des transitiven Verhaltens (Verkettung), z.B.
$$m(t_1,t_2) \wedge d(t_2,t_3) \ \ \rightarrow \ \ o(t_1,t_3) \vee d(t_1,t_3) \vee s(t_1,t_3)$$

[Die Axiome sind doch nicht original so ?!]

Es wurde noch eine Relation HOLDS eingeführt, welche wir aber erst später einführen. Für diese Relation werden natürlich auch Axiome verlangt.

Nun zeigte LADKIN 1986, daß diese Axiome genau die Theorie der Intervalle über eine unbegrenzte, lineare Ordnung darstellen. Das macht die Modellsuche insofern einfach, als daß wir von einer linearen Zeit (z.B. reelle Zeitachse) ausgehen und uns auf unsere Anschauung verlassen können.

In einer Intervallstruktur $[I, \{\subseteq, <\}]$ lassen sich die 13 Relationen zwischen den Zeitintervallen wie folgt ausdrücken:

$$
\begin{array}{rll}
s(t_1, t_2) & \text{gdw.} & t_1 \subseteq t_2 \\
 & & \wedge\ \exists t_3 (t_3 \subseteq t_2\ \wedge\ t_1 < t_3) \\
 & & \wedge\ \neg\exists t_4 (t_4 \subseteq t_2\ \wedge\ t_4 < t_1) \\
f(t_1, t_2) & \text{gdw.} & t_1 \subseteq t_2 \\
 & & \wedge\ \exists t_3 (t_3 \subseteq t_2\ \wedge\ t_3 < t_1) i \\
 & & \wedge\ \neg\exists t_4 (t_4 \subseteq t_2\ \wedge\ t_1 < t_4) \\
d(t_1, t_2) & \text{gdw.} & t_1 \subseteq t \\
 & & \wedge\ \exists t_3 (t_3 \subseteq t_2\ \wedge\ t_3 < t_1) \\
 & & \wedge\ \exists t_4 (t_4 \subseteq t_2\ \wedge\ t_1 < t_4) \\
b(t_1, t_2) & \text{gdw.} & t_1 < t_2 \\
 & & \wedge\ \exists t_3 (t_1 < t_3\ \wedge\ t_3 < t_2) \\
o(t_1, t_2) & \text{gdw.} & \exists t_3 (t_3 \subseteq t_1\ \wedge\ t_3 < t_2) \\
 & & \wedge\ \exists t_4 (t_4 \subseteq t_2\ \wedge\ t_1 < t_4) \\
 & & \wedge\ \exists t_5 (t_5 \subseteq t_2\ \wedge\ t_5 < t_1) \\
m(t_1, t_2) & \text{gdw.} & t_1 < t_2 \\
 & & \wedge\ \neg\exists t_3 (t_1 < t_3\ \wedge\ t_3 < t_2) \\
e(t_1, t_2) & \text{gdw.} & t_1 \subseteq t_2\ \wedge\ t_2 \subseteq t_1
\end{array}
$$

Nun wollen wir beweisen, daß mit unserer Übersetzung in die Punktstruktur die Axiome von Allen weiterhin erfüllt bleiben. Wir begnügen uns mit einem Beispielaxiom, um das Prinzip zu verdeutlichen. Das auserwählte Axiom lautet:

$$\text{before}(t_1, t_2) \wedge \text{overlaps}(t_2, t_3) \rightarrow \text{before}(t_1, t_3)$$

[oben stand $b(x, y)$ und hier before(x, y)]

Dazu schreiben wir uns die Übersetzungen daneben und leiten dann ab:

$$
\begin{array}{rll}
\text{before}(t_1, t_2) & \text{gdw.} & t_1 < t_2 \wedge\ \exists t_4 (t_1 < t_4 \wedge t_4 < t_2) \\
\text{overlaps}(t_2, t_3) & \text{gdw.} & \exists t_5 (t_5 \subseteq t_2 \wedge t_5 < t_3) \\
 & & \wedge\ \exists t_6 (t_6 \subseteq t_3 \wedge t_2 < t_6) \\
 & & \wedge\ \exists t_7 (t_7 \subseteq t_3 \wedge t_7 < t_2)
\end{array}
$$

Behauptung:

$$\text{before}(t_1, t_3) \quad \text{gdw.} \quad t_1 < t_3 \wedge\ \exists t_8 (t_1 < t_8\ \wedge\ t_8 < t_3)$$

Beweis:

Da gilt:

$$x < y\ \wedge\ u \subseteq y\ \rightarrow\ x < u \text{ (Axiom der Intervallstruktur)}$$

$$x < y\ \wedge\ u \subseteq y\ \rightarrow\ x < u \text{ (Axiom der Intervallstruktur)}$$

folgt aus:

$$t_1 < t_2\ \wedge\ \exists t_4 \exists t_5 (t_1 < t_4\ \wedge\ t_4 < t_2\ \wedge\ t_5 \subseteq t_2\ \wedge\ t_5 < t_3)$$

die Tatsache:

$$t_1 < t_2\ \wedge\ \exists t_4 \exists t_5 (t_1 < t_4\ \wedge\ t_4 < t_5\ \wedge\ t_5 < t_3)\ .$$

Da $<$ transitiv ist, folgt daraus:

$$t_1 < t_2 \ \wedge \ \exists t_4(t_1 < t_4 \ \wedge \ t_4 < t_3) \quad \text{sowie} \quad t_1 < t_3 \ ,$$

woraus schließlich folgt

$$t_1 < t_3 \ \wedge \ \exists t_4(t_1 < t_4 \ \wedge \ t_4 < t_3) \ ,$$

was bis auf Umbenennung der Behauptung entspricht.

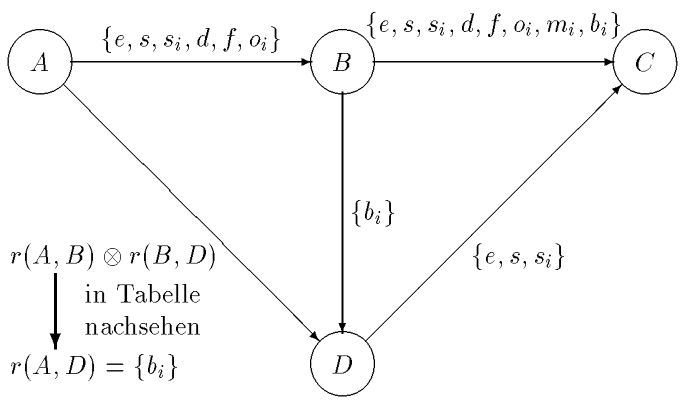
Daß nicht noch andere Beziehungen zwischen t_1 und t_3 bestehen, wurde durch diesen Beweis nicht widerlegt, aber es ist tatsächlich so. **[Tut barbarisch klingen, tut das !]**

Verkettungen werden bei der Inferenz eines Sachverhaltes ausgenutzt. Wir werden jetzt ein Beispiel abarbeiten und uns hinterher einer formalen Beschreibung zuwenden.

Beispiel: (A, B, C, D sind Zeitintervalle)

- A beginnt während B
- B beginnt nicht vor C
- B liegt zeitlich völlig nach D
- C und D beginnen gleichzeitig

Formalisiert können wir diesen Sachverhalt in einem *Allen-Netz* (wird noch definiert) folgendermaßen darstellen: **[Nicht ständig diese Klammern !]**



Was kann über $r(A, D)$ gesagt werden ?

Nun, es gilt $r(A, D) = r(A, B) \times r(B, D)$. Laut Tabelle im Anhang ergibt sich daraus

$$r(A, D) = \{b_i\} \ .$$

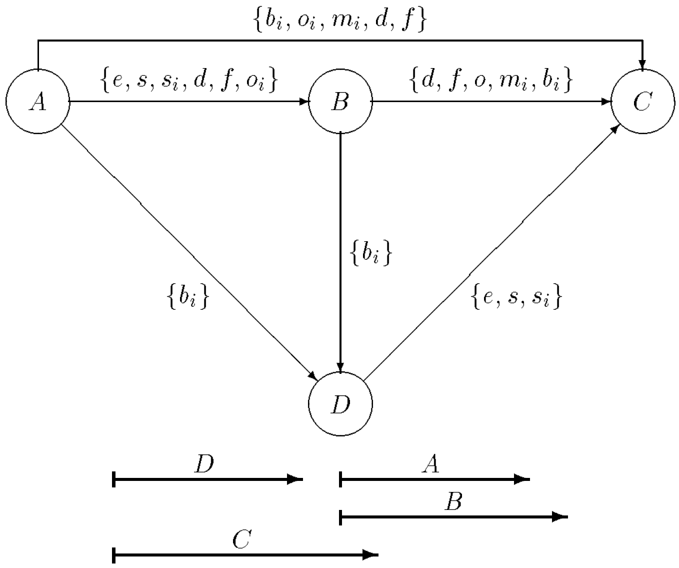
Somit können Aussagen über Intervallbeziehungen generiert werden, die im Text unerwähnt blieben. Ein anderes Verfahren ist die Einschränkung von Möglichkeiten.

Da $r(B, D) \times r(D, C) = \{d, o_i, f, m_i, b_i\}$ ist, kann die ursprüngliche Annahme über $r(B, C)$ (so wie der Sachverhalt es hergibt) eingeschränkt werden, d.h. durch Inferenz kommt neues Wissen hinzu, nämlich daß es sich bei $r(B, C)$ nicht um die Relationen s, s_i oder $=$ handeln kann.

Nun können noch weitere Konsistenzprüfungen vorgenommen werden, z.B.

$$r(A, B) \times r(B, C) = r(A, D) \times r(D, C) \ .$$

Wenn sich bei solchen Konsistenzprüfungen nichts mehr ändert (Stabilität erreicht) kann aufgehört werden. Was das Ergebnis bedeutet, werden wir noch diskutieren. Das Ergebnis unseres Beispiels ist:



Definition *Allen-Netz*:

Ein *Allen-Netz* ist gegeben durch $A = [T, C]$, wobei

- T eine Menge von Intervallen und
- C eine Abbildung $C : T \times T \to 2^{\{s, s_i, \ldots, e\}}$ ist. Dabei ist $\{s, s_i, \ldots, e\}$ die Menge der eingeführten Relationen.

Dabei gilt für C die Bedingung

$$C(t_1, t_2) = INV(C(t_2, t_1)) \ .$$

$INV(M)$ ist definiert als:

$$\begin{array}{rcl} INV(\emptyset) & = & \emptyset \\ INV(M_1 \cup s) & = & INV(M_1) \cup s_i \\ INV(M_1 \cup s_i) & = & INV(M_1) \cup s \\ INV(M_1 \cup f) & = & INV(M_1) \cup f_i \\ & \text{usw.} & \end{array}$$

Solch ein *Allen-Netz* läßt sich als Graph mit Knoten T und Kantenbeschriftungen $C(t, t')$ darstellen.

Definition *Modell*:

Ein *Modell* für $A = [T, C]$ ist eine Belegung $\beta : T \to I$ so daß für alle $t_1, t_2 \in T$ gilt, daß $\beta(t_1)$ und $\beta(t_2)$ in einer durch C zugelassenen Relation stehen.

Bemerkung :

Bei dieser Definition fällt auf, daß eine pure Belegung ein Modell sein kann, obwohl ein Modell in der Logik immer eine Interpretation ist, das heißt eine Belegung der vorkommenden Prädikate mit Wahrheitswerten *true* und *false*.

Diese Belegung der Prädikate ist hier implizit gegeben, weil als Intervallbereich I nur Intervalle der reellen (rationalen usw.) Zeitachse in Frage kommen mit einem Anfang $A(t)$ und Ende $E(t)$, wobei $A(t) < E(t)$.

Wenn 2 solche Intervalle betrachtet werden, ist natürlich genau gegeben, welche Beziehung (Relation im Allen Kalkül) zwischen ihnen besteht und damit auch eine Interpretation.

Definition *global konsistent*:

$A = [T, C]$ heißt *global konsistent*, falls es ein Modell besitzt; entsprechend heißt

$A = [T, C]$ *global inkonsistent*, falls es kein Modell besitzt.

Bemerkung :

Das Netz $A = [T, C]$ ist global konsistent, gdw. eine *eindeutige* Abbildung

$$C' : T \times T \rightarrow \{s, s_i, \ldots, e\}$$

existiert mit

$$C' \subseteq C \qquad \text{(d.h. } C'(t,t') \in C(t,t')\text{, für alle } t, t' \in T\text{)}$$

und $A' = [T, C']$ ist global konsistent.

Hier ist C' eindeutig, d.h. im Allen-Netz $A' = [T, C']$ sind die Kantenbeschriftungen nur Einermengen bzw. einzelne Elemente aus $\{s, s_i, \ldots, e\}$.

Falls $A = [T, C]$ global konsistent ist, gibt es in der Regel mehrere C', so daß $A' = [T, C']$ global konsistent ist, d.h. es gibt auch mehrere verschiedene Modelle von $A = [T, C]$.

Das liegt an der Eindeutigkeit der Zuordnung von $A' = [T, C']$ mit einem Modell (in unserem Fall eine Menge von Intervallen auf der (reellen) Zeitachse). **[Klammern in Klammern find ich echt SPITZE]** Wenn 2 konkrete Intervalle gegeben sind, trifft genau eine der gegebenen 13 Relationen $s, s_i, \ldots, e$ auf diese beiden Intervalle zu.

Nun wollen wir uns 2 Problemen zuwenden, die sich gerade mit solchen *Teil*modellen beschäftigen. **[Verhindere bitte das ≫Teilmodellen≪ in mehreren Schriften ! Entweder das ganze Wort oder garnicht !]**

Problem 1 (Existenz eines Modells)
Es ist die globale Konsistenz eines Allen-Netzes $A = [T, C]$ zu entscheiden.

Äquivalente Fragen wären:

- Existiert ein Modell ?
- Existiert eine eindeutige Abbildung $C' \subseteq C$, wobei das Netz $[T, C']$ global konsistent ist ?

Problem 2 (Konstruktion eines Modells)
Berechne ein eindeutiges C' zu $A = [T, C]$ mit $C' \subseteq C$ und $[T, C']$ ist global konsistent.

Das erste Problem ist offenbar eine Abschwächung von Problem 2. **[Das erste Problem oder Problem 1, aber nicht das 1. Problem]** Da Problem 1 aber wieder ein NP-vollständiges Problem ist, kann Problem 2 auch nur NP-vollständig sein (wieder *Reduktion* siehe Constraint Propagation). Beide Probleme sind praktisch nur durch Probieren zu lösen.

Das schließt aber nicht die Existenz eines Verfahrens aus, mit dem wir uns der Lösung annähern können. Dabei beschränken wir uns auf die Betrachtungen auf lokale Konsistenzen. Falls ein Netz an einer Stelle lokal inkonsistent ist, ist das Netz auch global inkonsistent.

Definition *lokal konsistent*:

Ein Netz $A = [T, C]$ ist *lokal konsistent* an der Stelle $T' = \{T_1, T_2, T_3\} \subseteq T$, falls das Netz $A' = \{T', C_{/T'}\}$ global konsistent ist. Dabei ist $C_{/T'}$ die Einschränkung von C auf T', d.h.

$$C_{/T'} = \begin{cases} C(t,t') & \text{falls } t,t' \in T', t \neq t' \\ \text{nicht definiert} & \text{sonst.} \end{cases}$$

Bemerkung :

Das Teilnetz A' ist gerade ein Teildreieck von A. Falls A genau n Knoten besitzt, gibt es $\binom{n}{3}$ solche Teildreiecke.

Nun wollen wir das angekündigte Verfahren zur lokalen Konsistenzprüfung formal betrachten. Es ist das gleiche, welches im Einführungsbeispiel verwendet wurde. Daher dürfte der Ablauf aus der Intuition heraus klar sein; aufeinanderfolgende Konsistenzprüfungen in Teildreiecken und überflüssige Beschriftungen entfernen. Vorher wird das Netz an den unbekannten Kanten noch „aufgefüllt“ (siehe Schritt 1 des Verfahrens).

Verfahren:

Ausgangspunkt: $A = [T, C]$ ist gegeben. Es sei P der Definitionsbereich von C und r_0 eine andere Bezeichnung für C, d.h.

$$r_0(t,t') = C(t,t')$$

Der Algorithmus benutzt

- einen Stack (Keller) K für aktuell zu prüfende Intervallpaare und
- $r : T \times T \longrightarrow 2^{\{s, s_i, \ldots, e\}}$ für aktuell berechnete Relationen.

Vorgehensweise:

Die Menge $r(t,t')$ sukzessive verkleinern (überflüssige Beschriftungen der Kanten streichen) durch Vergleich mit Nachbarn (lokale Transitivität) bis zur Stabilisierung. Stabilität tritt wegen Endlichkeit des Netzes ein.

1. Schritt:
   Initialisiere $K := P$
   und $r(t,t') = \begin{cases} r_0(t,t') & \text{falls } (t,t') \in P \\ \{s, s_i, \ldots, e\} & \text{sonst (Auffüllen).} \end{cases}$
2. Schritt:
   - Falls $K = [\,]$ $\longrightarrow$ EXIT (lokal konsistent),
   - sonst $(t_1, t_2) := pop(K)$ und bei Schritt 3 weiter.
3. Schritt:
   Für alle $t \in T$ die Schritte 3a) und 3b) ausführen !
   a) Falls $r(t_1, t) \supset r(t_1, t) \cap (r(t_1, t_2) \times r(t_2, t))$ gilt
      - Setze $r(t_1, t) := r(t_1, t) \cap (r(t_1, t_2) \times r(t_2, t))$
      - Falls $r(t_1, t) = \emptyset$ $\longrightarrow$ EXIT (lokal inkonsistent),
        sonst $push(t_1, t)$ (einkellern).

   b) Falls $r(t, t_2) \supset r(t_1, t_2) \cap (r(t, t_1) \times r(t_1, t_2))$ gilt
      - Setze $r(t, t_2) := r(t_1, t_2) \cap (r(t, t_1) \times r(t_1, t_2))$
      - Falls $r(t, t_2) = \emptyset$ $\longrightarrow$ EXIT (lokal inkonsistent),
        sonst $push(t, t_2)$ (einkellern).

Behauptung:

Der Algorithmus bricht nach maximal $O(n^3)$-Schritten ab (n ist Anzahl der Knoten im Allen-Netz).

Begründung:

Wie oft kann die zugehörige Menge $r(t_1, t_2)$ zu einer Kante $[t_1, t_2]$ aus K mittels *pop* herausgenommen werden und somit der Schritt 3) angewendet werden ?

Es sind höchstens 13 mal, weil die Kardinalität der Menge $r(t_1, t_2)$ sich bei einer *push*-Operation mindestens um eins verringert.

Da es n^2 solcher Paare $[t_1, t_2] \in T \times T$ gibt, ist die Anzahl der Durchläufe des Schrittes 3) mit $O(n^2)$ abschätzbar.

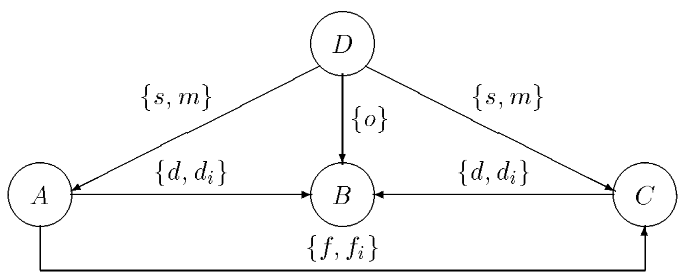
In jedem dieser Durchläufe finden $O(n)$ Berechnungen statt, so daß die Gesamtkomplexität $O(n^3)$ beträgt.

Bemerkung :

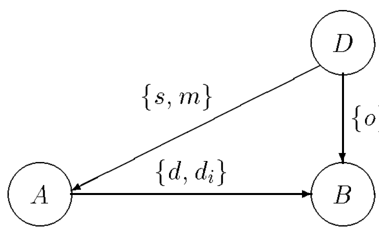
Das Verfahren arbeitet korrekt bezüglich der Bewertung von Inkonsistenzen, d.h. man kann sich darauf verlassen, daß bei EXIT (lokal inkonsistent) das gesamte Netz inkonsistent ist.

Als Preis dafür, einen schnellen Algorithmus zu haben, muß hingenommen werden, daß aus der lokalen Konsistenz (EXIT lokal konsistent) nicht die globale Konsistenz folgt.

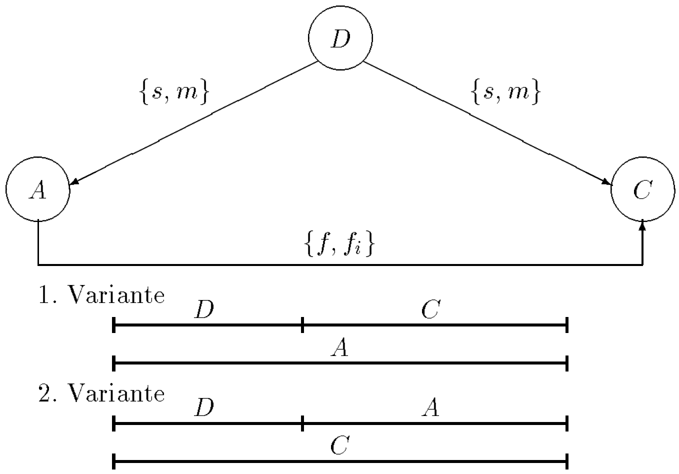
Dazu ein Beispiel:



Dieses Netz ist überall lokal konsistent und stabil.



Analog können die anderen Teilnetze betrachtet werden. Aber das Netz ist global inkonsistent, weil folgende zwei Varianten nicht mit den anderen Teilnetzen in Übereinstimmung zu bringen sind.


D
{s, m}
{s, m}
A
C
{f, fi}
1. Variante
D
C
A
2. Variante
D
A
C

Anhang A

OPS5 – Eine Syntaxbeschreibung

Vereinfachte Syntax für OPS5 :

```
<regel>                 ::= ( p <regelname> <LHS> --> <RHS>)
<LHS>                   ::= <cond_element> {[-] <cond_element>}
                                      ; bei - <cond_element> negierte Bedingung
<RHS>                   ::= {<action>}
<cond_element>          ::= (<class_name> {<attribut_value_pair>})
                                      ; müssen nicht alle da sein
<attribut_value_pair>   ::= ^ <attribut> <value>
<action>                ::= (make <class_name> {<attribut_value_pair>})
                                      ; nicht alle erforderlich, Rest NIL (undefiniert)
                          | (remove <number>)
                                      ; entfernt WME, das mit i-ter Bedingung der LHS
                                      ; gematcht hat
                          | (modify <number> {<attribut_value_pair>})
                                      ; entsprechend remove, bekommt neuen Zeitstempel
                          | (write ...)
                          | (halt)
                          | (built ...); neue Regel
```

Anhang B

OPS5-Beispielprogramm

Folgendes Beispielprogramm nimmt Blöcke der Größe nach von einer Halde und plaziert sie der Größe entsprechend.

```
(strategy mea)

(literalize goal task index)
(literalize block name size place)
(literalize start)


(p begin
   (start)
   -->
   (make block ^ name A ^ size 10 ^place heap)
   (make block ^ name B ^ size 20 ^place heap)
   (make block ^ name C ^ size 30 ^place heap)
   (make block ^ name D ^ size 40 ^place heap)
   (remove 1)
   (make goal ^ task add ^ index 1)
)

(p pick-up
   (goal ^ task add)
   (block ^ size     <size>       ^ place heap)
  -(block ^ size { > <size> }     ^ place heap)
   -->
   (modify 2 ^ place hand)
)

(p holding
   (goal  ^task add)
   (block ^ place hand)
   -->
   (modify 1 ^ task put-down)
)
```

```
(p put-down
   (goal  ^ task put-down  ^ index <rank>)
   (block ^ place hand)
   -->
   (modify 2 ^ place <rank>)
   (bind <a> (compute <rank> + 1 ) ) ; arithm. Berechnung und
   Bindung der Variablen a
   (modify 1 ^ task add  ^ index <a> )
)

(p stop
   (goal   ^ task add)
  -(block  ^ place heap)
   -->
   (remove 1)
   ( wm )         ; druckt Arbeitsspeicher aus
   (halt)         ; beendet Abarbeitung
)

(defun rerun nil
   ( remove *)    ; loescht gesamten Arbeitsspeicher
   (make start)
   (run)          ; startet Abarbeitung
```

`rerun` ist eine neu definierte LISP-Funktion. Gestartet wird das Programm auf dem Toplevel mit: `:- (rerun)`

Anhang C

Dialog zum Thema Künstliche Intelligenz

COMPUTER Hotel International, guten Tag. Womit kann ich Ihnen dienen?

ANRUFER Ja, ähm, ich wollte wissen, ob Sie ein Doppelzimmer mit Bad haben.

COMPUTER Ja, wir haben 120 Doppelzimmer mit Bad.

ANRUFER Was, alle frei?

COMPUTER Nach freien Zimmern haben sie nicht gefragt.

ANRUFER Also, wieviele haben sie frei?

COMPUTER Heute haben wir nichts frei.

ANRUFER Ich komme ja auch erst übermorgen, nur eine Nacht.

COMPUTER Ach so. Übermorgen haben wir noch fünf frei. Sie möchten eins dieser herrlichen, preiswerten, Zimmer, nicht wahr? Bitte buchstabieren Sie Ihren Namen.

ANRUFER Em. Ü. El. El. Eh. Er.

COMPUTER Danke, Herr Emü el Ellehr. Ich bestätige Ihre Buchung: ein ruhiges Doppelzimmer mit Bad. Übermorgen für eine Nacht, zum Preis von 450 DM. Wann bitte kommen Sie an?

ANRUFER Das weiß ich nicht ... wo ist denn mein Flugschein? ...

COMPUTER Das weiß nur die Flugauskunft, ich verbinde.

Flugauskunft, guten Tag.

ANRUFER ... ach ja, hier. Flug LH 4523, wann kommt der?

COMPUTER Flug El Al 4523 ist nicht in der Datei.

ANRUFER Nicht El Al, El Ha, El Haaaaaa, Haaaaaaarghhh!

COMPUTER Zentralrechner Bundesgesundheitsamt. Sie haben behandlungsbedürftige Erkältungssymptome. Zur weiteren Diagnose sagen sie bitte 'aahh'.

ANRUFER Aaa. Bin ich denn bescheuert? Ich wollte ein Zimmer buchen und keinen Arzt. Oh, hätte ich doch bloß noch einen guten alten Apple!

COMPUTER *schweigt.*

ANRUFER Hallo, sind Sie noch da?

Computer Zentrallager. Entschuldigen Sie die Verzögerung. Ich habe einen Apple, Baujahr 1976 im 'Used Comp Shop', Castrop-Rauxel, zu einem Preis von 1800 DM. Möchten Sie ihn haben?

Anrufer Nein. Vergiß es!

Computer Ihr Benutzercode erlaubt keine Löschung von Zentrallagerdaten.

Anrufer Ich will ein Zimmer, Du Hirsch, Du damischer!!

Computer String Recognition Error at Susra$('damischer' $$$Aswi42S): Syntax Check at 200DE Rec83749: Error Recovery in Progress. Please wait.

Anrufer Herrje, verschwinde. Ich will zurück zum anderen Computer.

Computer Zentralrechner Bundesgesundheitsamt. Zur Beurteilung, ob Sie bescheuert sind, einige Fragen zu Ihrem Gesundheitszustand. Gab es früher Probleme?

Anrufer Du verdammter $&#. Du kannst mich mal #$&%¿¡.

Computer Achtung, Achtung, hier spricht der Polizeirechner. Ihnen wird eine Straftat nach Paragraph 1523 StPO, Beleidigung zentraler Intelligenzsysteme, zur Last gelegt. Ihre Personal-Chipkarte wird eingezogen. Ich weise Sie auf Ihr Aussageverweigerungsrecht hin. Sie können außerdem einen Computer des Rechtsanwaltverbandes konsultieren.

Anrufer *bricht zusammen.*

Anhang D

Verkettung von Relationen über Intervallen

ySz xRy	b	b_i	d	d_i	o	o_i	m	m_i	s	s_i	f	f_i
before b	b	no info	$b\ o\ m\ d\ s$	b	b	$b\ o\ m\ d\ s$	b	$b\ o\ m\ d\ s$	b	b	$b\ o\ m\ d\ s$	b
after b_i	no info	b_i	$b_i\ o_i\ m_i\ d\ f$	b_i	$b_i\ o_i\ m_i\ d\ f$	b_i	$b_i\ o_i\ m_i\ d\ f$	b_i	$b_i\ o_i\ m_i\ d\ f$	b_i	b_i	b_i
during d	b	b_i	d	no info	$b\ o\ m\ d\ s$	$b_i\ o_i\ m_i\ d\ f$	b	b_i	d	$b_i\ o_i\ m_i\ d\ f$	d	$b\ o\ m\ d\ s$
contains d_i	$b\ o\ m_i\ d\ f$	$b_i\ o_i\ d_i\ m_i\ s_i$	$o\ o_i\ IN\_con\ =$	d_i	$o\ d_i\ f_i$	$o_i\ d_i\ s_i$	$o\ d_i\ f_i$	$o_i\ d_i\ s_i$	$d_i\ f_i\ o$	d_i	$d_i\ s_i\ o_i$	d_i
overlaps o	b	$b_i\ o_i\ d_i\ m_i\ s_i$	$o\ d\ s$	$b\ o\ m\ d_i\ f_i$	$b\ o\ m$	$o\ o_i\ IN\_con\ =$	b	$o_i\ d_i\ s_i$	o	$d_i\ f_i\ o$	$d\ s\ o$	$b\ o\ m$
overlapped by o_i	$b\ o\ m\ d_i\ f_i$	b_i	$o_i\ d\ f$	$b_i\ o_i\ m_i\ d_i\ s_i$	$o\ o_i\ IN\_con\ =$	$b_i\ o_i\ m_i$	$o\ d_i\ f_i$	b_i	$o_i\ d\ f$	$o_i\ b_i\ m_i$	o_i	$o_i\ d_i\ s_i$
meets m	b	$b_i\ o_i\ m_i\ d_i\ s_i$	$o\ d\ s$	b	b	$o\ d\ s$	b	$f\ f_i\ =$	m	m	$d\ s\ o$	b
met-by m_i	$b\ o\ m\ d_i\ f_i$	b_i	$o_i\ d\ f$	b_i	$o_i\ d\ f$	b_i	$s\ s_i\ =$	b_i	$d\ f\ o_i$	b_i	m_i	m_i
starts s	b	b_i	d	$b\ o\ m\ d_i\ f_i$	$b\ o\ m$	$o_i\ d\ f$	b	m_i	s	$s\ s_i\ =$	d	$b\ m\ o$
started-by s_i	$b\ o\ m\ d_i\ f_i$	b_i	$o_i\ d\ f$	d_i	$o\ d_i\ f_i$	o_i	$o\ d_i\ f_i$	m_i	$s\ s_i\ =$	s_i	o_i	d_i
finishes f	b	b_i	d	$b_i\ o_i\ m_i\ d_i\ s_i$	$o\ d\ s$	$b_i\ o_i\ m_i$	m	b_i	d	$b_i\ o_i\ m_i$	f	$f\ f_i\ =$
finished-by f_i	b	$b_i\ o_i\ m_i\ d_i\ s_i$	$o\ d\ s$	d_i	o	$o_i\ d_i\ s_i$	m	$s_i\ o_i\ d_i$	o	d_i	$f\ f_i\ =$	f_i

Index